Reliable Low-Level Image Analysis

Ullrich Köthe

September 21, 2007

Contents

1 Introduction and Motivation

You must be able to walk firmly on the ground before you start walking on a tightrope.

Henri Matisse

Vision is the most versatile among all human senses. It can be used for tasks as diverse as object and shape recognition, self-localization and motion control, reading a text or a musical score, and graphical data analysis, to name just a few. It works equally well in various contexts, from natural to artificial surroundings. But this versatility comes at a price: an individual measurement (i.e. the intensity at a particular wave length, time and retina location) has no meaning in itself. Only a collection of connected measurements (i.e. an image or image sequence) makes sense, but its interpretation is extremely difficult. This phenomenon is often called the *semantic gap* of image analysis. Traditional technical systems avoid the semantic gap by assigning a well-defined meaning to every measurement. For example, a speedometer tells us the velocity of a car at a particular time, a thermometer indicates the temperature of the surrounding air, and so on. Current vision systems employ the same strategy: they are always tailored to a particular, very narrow application.

Yet, a universal artificial vision system would be highly desirable. First, such a system would be much more robust against variations in the input data or modifications of the task. When the environment cannot be tightly controlled, the system must be able to deal with unexpected situations or unusual structures. In fact, detection of the unexpected is the main point of many vision applications. By definition, the unexpected cannot be exactly described beforehand, so robustness is a necessity. Second, solutions for narrow domains are often one-of-a-kind systems that must be developed from scratch or at least adapted by an image analysis expert. The lack of reusable, generic solutions for common tasks appears exceedingly frustrating for user and developers alike. Universal vision systems will eventually be much more satisfactory and cheaper. While such systems are unlikely to appear in the near future, any progress toward genericity will simplify the development of successful vision applications.

All non-trivial vision systems consist of a large number of modules. Today, these modules are mainly selected on the basis of the developer's experience and preferences, and expensive custom-build modules are common. There is no systematic way for deciding which modules are the most appropriate in a given task, and how module parameters should be adjusted. While experimental validation of complete vision systems on representative application data has become state-of-the-art during the last decade, this so called “black-box” approach to testing is insufficient for optimizing modularization: When

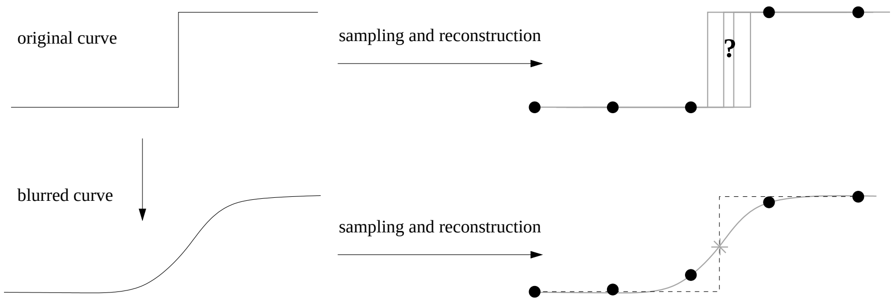


Figure 1.1: Top: When a piecewise constant function is sampled, it is impossible to locate the jump to sub-pixel accuracy in the reconstructed function. Bottom: If the function is blurred before sampling, the inflection point of the reconstructed function (gray star) is a good estimate of the original jump (dashed). Figure adapted from [Jähne 97].

a black-box test finds that system A fails in 10% of the trials, while system B fails in only 3%, one will prefer system B, but one doesn't gain any deeper insight into the reasons of the difference. Figures on overall system performance are insufficient for pinpointing which modules are causing problems and why.

However, it is likewise insufficient to look at individual modules in isolation, as the following example demonstrates. Edges in digital images are never perfectly sharp, but always somewhat blurry. In general, sharper edges are preferable because their location can be estimated with higher accuracy, and several authors suggest to apply sharpening methods (e.g. total variation minimization [Dibos & Knopfler 00, Rudin et al. 92] or robust estimation [Boomgaard & Weijer 03]) in order to reduce blurring prior to edge detection. However, depending on the subsequent edge detector, this suggestion may have quite the opposite effect. Consider figure 1.1 top, where a perfect step edge is sampled without any blurring. It can be seen that the step can easily be detected in the digital representation, but its exact position can only be reconstructed with pixel accuracy – the transition could have occurred at any location between the two pixels on either side of the step. The maximum error of pixel-accurate edge detectors cannot drop below the pixel size, no matter how sharp the transition is. In contrast, a blurry edge (figure 1.1 bottom) can potentially be reconstructed almost exactly from its samples, and the position of the original transition may be estimated with sub-pixel accuracy (e.g. at the inflection point of the reconstructed curve) – localization errors as low as 0.01 pixels are possible. From a signal theoretic point of view, too sharp edge transitions violate Shannon's sampling theorem and suffer from aliasing (see chapter 3). Thus, a module creating sharp edge transitions is not a value in and of itself – it has to be considered in context.

In this work, properties of individual modules and module interactions in larger systems will receive equally careful treatment. Indeed, we will follow the holistic approach taken by the emerging field of *algorithm engineering*[1] which suggests to consider real world modeling, algorithm development, coding, and validation as equally important aspects of a successful problem solution. In the field of image analysis, this means that realistic scene

[1] see `http://www.algorithm-engineering.de`

models, properties of image acquisition devices, experimental evaluation of modules and systems, analysis of the difference between continuous and discrete theories etc. have to be considered with equal care. Interestingly, this wider point of view also leads to better understanding of the details: It leads to a modularization where each module is making realistic assumptions about its input, and produces outputs with well-defined properties and for well-defined receivers. In an (as yet Utopian) optimal system, each module can be characterized and optimized individually, but the modularization is designed in such a way that the performance of the whole system can be predicted from the properties of the modules it is comprised of.

Of course, a single work cannot realize this vision in its entirety. Instead, we restrict our attention to *low-level image analysis.* This decision has been taken for the following reasons:

- Image interpretation cannot be based directly on the pixel data (except in very restricted contexts). It is necessary to first transform the raw iconic information into more meaningful primitives. This is precisely the purpose of low-level image analysis which is therefore an intrinsic part of more or less any image analysis solution, including the human visual system (whose processing starts with local, orientation dependent operators). Improvements in reliability and generality of low-level modules will have an immediate impact.

- Guarantees on a module's behavior can only be given relative to dependable properties of the module's input. As long as proofs about the output of low-level modules are not available, guarantees on higher level modules (which use low-level results as input) are out of reach. In contrast, low-level image analysis works directly on the image data, which can be constrained so as to facilitate such proofs, provided the constraints remain sufficiently realistic for the conclusions to be useful in practice. Thus, it is necessary to understand the possibilities and limitations of low-level algorithms before moving on to higher-level modules.

In our attempt to formally understand the limitations of low-level image analysis, the concept of an *ideal geometric image* plays a key role. We think of the ideal geometric image as a perfect 2-dimensional projection of a 3-dimensional scene, at infinite resolution and without blurring and noise. No real imaging device can observe the ideal geometric image – blurring by the lens, digitization, noise and other effects cause the real image to differ significantly from the ideal one. In fact, an infinite amount of information is lost since the real image is a discrete function on the digital plane, whereas the ideal image is an analog function on the continuous plane. Nonetheless, we know from experience that many important properties of the ideal image can still be recovered. Thus, we define *low-level image analysis* as the task of reconstructing interesting properties of the ideal geometric image from the actually observed digital image[2]. Knowledge of the ideal image's properties is highly desirable because these properties are independent of the specifics of

[2]It should be noted that both ideal and real images reside in 2-dimensional space, so our definition implies that low-level image analysis is not concerned with the 3-dimensional world.

a particular image acquisition device and are thus suitable starting points for higher-level analysis (e.g. 3D scene reconstruction). The present work will investigate to what extend and with what accuracy this reconstruction is possible, in spite of the information loss.

Another distinction we are interested in is the one between bottom-up and top-down processing. Bottom-up strategies are characterized by the fact that they solely rely on information contained in the stimulus to be processed (i.e. the pixel data), whereas top-down strategies are based on additional information outside the stimulus. If a system overemphasizes top-down strategies, it will only be able to see what it is supposed to see: Constraints and assumptions override the information coming from the data. In contrast, when there is too little top-down guidance, the system may be unable to correctly interpret images with high noise or with occlusions, ending up at random conclusions. Obviously, the two strategies have to be balanced in the right way for a system to perform optimally. The optimal trade-off can only be found if the capabilities and limitations of the modules realizing the two strategies are precisely known. In most of this work, we concentrate on bottom-up strategies, but top-down strategies should eventually be analyzed with the same care.

At this point of the discussion, people sometimes question the value of reliable low-level bottom-up processing in vision. After all, the human visual system can correctly handle input of very low quality, where higher-level top-down analysis apparently takes precedence. When the high-level system can tolerate bad low-level input, work on optimal low-level modules might be considered a waste of resources. In our opinion, this point of view severely underestimates the importance of low-level vision. First, image analysis is clearly much easier with better input data, even in the very robust human visual system. Otherwise, you wouldn't see so many people wearing eye glasses! Second, raw image data are highly ambiguous. Dealing with these ambiguities is a major (perhaps *the* major) challenge in automated image interpretation. Less-than-optimal low-level modules (as are common today) will often increase data ambiguity instead of reducing it, for example by creating artifacts such as spurious edges. Since these artifacts make an already difficult problem even more difficult, we would like to avoid them whenever possible. Third, recent psychological experiments suggest that bottom-up processing plays a much more important role in human vision than was previously believed. In these experiments, unknown images are presented to subjects for very short periods of time (20 to about 100 ms). Given the measured response times of the subjects to these stimuli and the known length of the visual and motoric pathways, there is only little opportunity for top-down processing and feedback to kick in. Yet, subjects are able to solve many basic recognition tasks (e.g. whether the image contains an indoor or outdoor scene, or whether or not it contains an animal) with very low error rates (see [Serre et al. 05] for an overview). Therefore, we are convinced that efforts put into reliable low-level vision will highly pay-off. In addition, several of our findings (e.g. the geometric sampling theorems, chapter 6) apply to top-down methods as well.

Within the realm of low-level analysis we have chosen a *segmentation* approach. In other words, we attempt to reconstruct region boundaries defined in the ideal geometric image. The question whether these boundaries correspond to actual object boundaries in the 3-dimensional world is (by our definition) beyond the scope of low-level segmentation.

The performance of a low-level segmentation algorithm is defined by how well it reconstructs the boundaries present in the ideal geometric image, regardless of their meaning in the 3D world. We favor the segmentation approach due to the following reasons:

- There is a number of applications where segmentation itself is a major analysis goal, most notably in medical and scientific imaging. Even when low-level segmentation alone is not sufficient, subsequent higher-level algorithms or interactive procedures will profit from more accurate low-level results.

- Segmentation aims for a complete (i.e. dense) description of the image: every point of the plane is classified as belonging to an edge, to a corner/junction, or to the interior of a homogeneous region. When a dense description is available, it is relatively easy to derive a sparse description (e.g. a set of isolated interest points) from it, but not vice versa.

- If an algorithm fails to compute the correct segmentation in some areas, this will immediately become visible in the results. In contrast, such problems may go unnoticed when we only aim at a sparse image description, where algorithm failures may be indistinguishable from feature-less areas. In this sense, segmentation forces one to precisely analyze where and why errors occur – it is impossible to simply sweep failures under the carpet. Thus, even when segmentation is not the ultimate solution, it is an important tool for studying the problem.

Low-level image segmentation has received a lot of attention over the years, being one of the oldest ideas in image analysis. Yet, there are many indications that its theoretical limits are not nearly reached by today's algorithms, and are, in general, not even known with a satisfactory degree of precision. A number of examples shall illustrate this fact.

Consider figure 1.2, where we want to recognize the characters on the license plate of a car which are easily readable for a human. When we try standard methods such as the edge detectors proposed by Canny [Canny 86], Shen and Castan [Shen & Castan 92], or the watershed transform [Vincent & Soille 91], the results are pretty bad (especially for the more complicated characters 4, 9, G, and W) and subsequent high level algorithms will have a hard time interpreting these responses correctly. Edges computed by means of thresholding happen to be the best, but this is only a coincidence because this particular image does not suffer from a shading gradient which would destroy the effectiveness of thresholding.

The usual conclusion at this point is that bottom-up approaches are unsuitable in this application. Instead, one would typically consider an approach based on pattern recognition and machine learning, since the number of possible characters is limited to 36 or so (depending on the country), and their possible shapes are predetermined by law. However, while we might quickly arrive at a practical solution in this manner, we will not learn anything about *why* bottom-up methods performed so poorly. Are there fundamental limits in the sense that no purely data-driven algorithm can ever achieve significantly better results, or did we just try an unfortunate selection of algorithms?

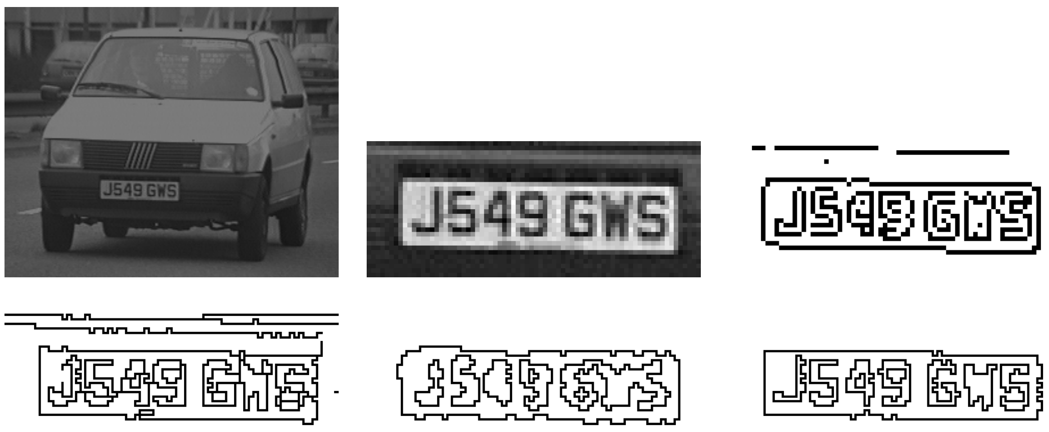


Figure 1.2: Original image and license plate region. Although the resolution is quite low (width of the characters' lines is less than 2 pixels), a human can easily recognize the characters in the original image. In contrast, segmentations with the Canny algorithm, Shen-Castan algorithm, watershed algorithm, and thresholding are not very accurate.

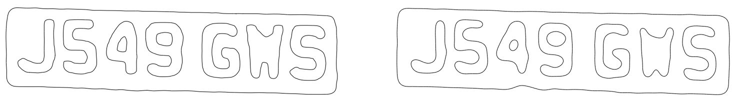


Figure 1.3: Segmentation of the license plate with the subpixel watershed algorithm (left) and the subpixel zero-crossing algorithm (right) after twofold oversampling. See chapter 5 for a description of these methods.

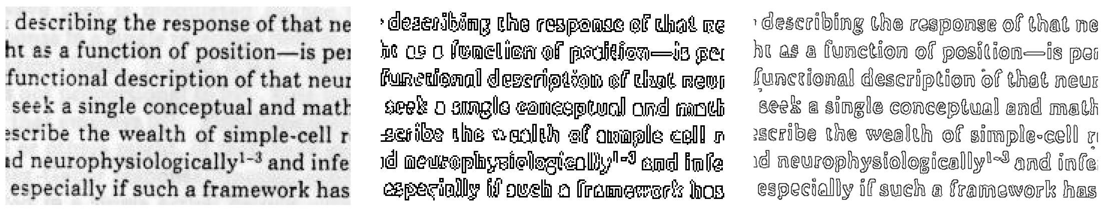

describing the response of that ne
ht as a function of position—is pe
functional description of that neur
seek a single conceptual and matl
escribe the wealth of simple-cell r
id neurophysiologically¹⁻³ and infe
especially if such a framework has

Figure 1.4: Improved segmentation on up-sampled images: original (left), Canny edges at original (center) and doubled (right) resolution. Bi-cubic interpolation was used for twofold up-sampling in both dimensions.

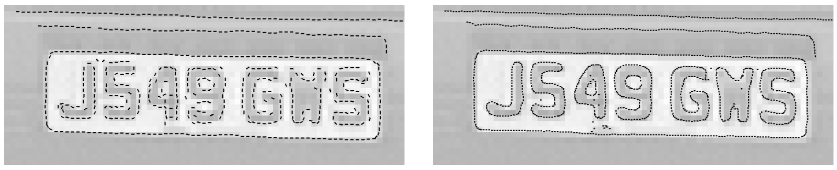


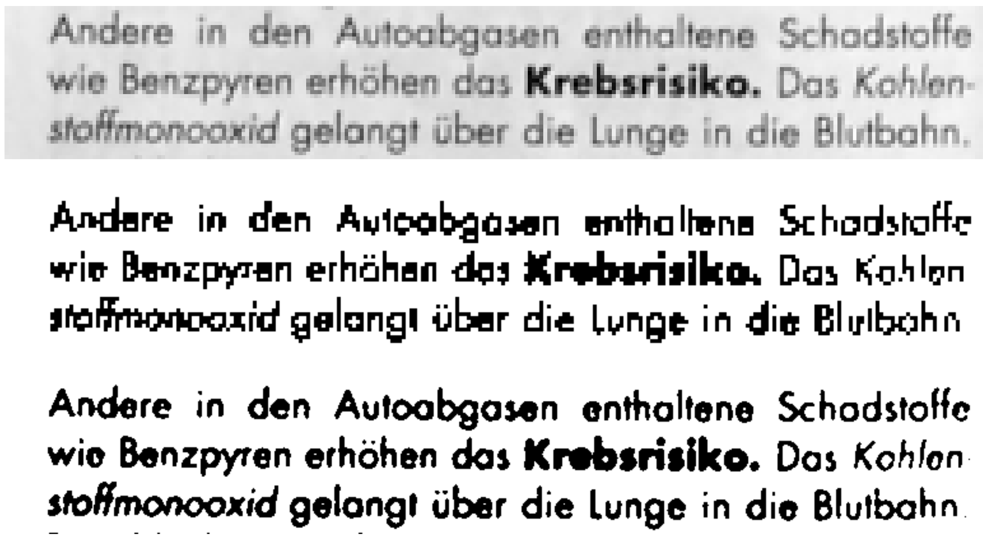
Figure 1.5: Improved segmentation on up-sampled images: Canny sub-pixel edgels at original (left) and doubled (right) resolution. Bi-cubic interpolation was used for twofold up-sampling in both dimensions.

In this particular example, we can definitely achieve much better results with bottom-up methods, as figure 1.3 shows. These segmentations have been obtained from exactly the same data, without using prior knowledge about the characters, and indeed without making any assumptions beyond a basic edge definition. The accuracy of the result becomes especially apparent when one observes that the algorithm correctly detected the difference between the straight upper bar of the number "5" and the curved upper bar of the letter "S". Clearly, there is much more information in the data than the traditional algorithms were able to extract. But why do traditional algorithms fail? What can be done to avoid loosing information? And how do our findings on this particular example apply to a larger class of images? These are some of the questions we are going to deal with in this work.

Traditionally, low-level image analysis tries to quickly reduce the amount of data. In view of the small computer memories in the early days of image analysis, this was a necessity, but it may not be adequate today, where memory is no longer the primary concern. Look at figure 1.4. It shows a text fragment with relatively low resolution (left) and the edges detected by means of the standard Canny algorithm (center). While the original text is fairly easy to read, it is almost unreadable in the edge image. Again, most people would conclude that edge detection wasn't the right approach. However, if we repeat edge detection after interpolating the original image to twice the original resolution (i.e. after increasing the image size instead of reducing it), text readability is significantly improved (figure 1.4 right). To the best of my knowledge, this phenomenon was first reported by [Overington 92], but anecdotal evidence has also been provided by others. The improvement appears surprising at first since interpolation does not increase the information content of the image. The only possible explanation is that information gets lost when we perform edge detection at the original resolution.

One could put forward the objection that improvements on interpolated images might just be artifacts of pixel-accurate edge representations. After all, a pixel-based edge representation can store at most one edge point per pixel, and any edge pair must be separated by at least one non-edge pixel. Figure 1.5 left demonstrates that a subpixel edge detector can indeed detect edges with higher accuracy than its pixel-accurate counterpart at the same resolution (compare figure 1.2 top right). But figure 1.5 right shows that the results improve even further when we work on an interpolated image, in agreement with what we had already observed for pixel-accurate methods.

For the sake of a first, intuitive understanding of this phenomenon, suppose we are


Andere in den Autoabgasen enthaltene Schadstoffe wie Benzpyren erhöhen das **Krebsrisiko.** Das *Kohlenstoffmonooxid* gelangt über die Lunge in die Blutbahn.

Andere in den Autoabgasen enthaltene Schadstoffe wie Benzpyren erhöhen das **Krebsrisiko.** Das *Kohlen stoffmonooxid* gelangt über die Lunge in die Blutbahn

Andere in den Autoabgasen enthaltene Schadstoffe wie Benzpyren erhöhen das **Krebsrisiko.** Das *Kohlen stoffmonooxid* gelangt über die Lunge in die Blutbahn

Figure 1.6: Top: Original low-resolution gray-value image of a text. Center: Binarisation at original resolution. Bottom: Binarisation after threefold oversampling. The geometric accuracy of the oversampled binary image (while far from perfect) is much better.

segmenting an idealized printed document that contains only pure black and white (letters and paper respectively) with perfectly sharp edges. That is, the ideal geometric image has 1 bit per point, but infinite resolution. When a digital image of this document is taken, some blurring by the optical system and sensor of the camera is unavoidable (and, in fact, is required in order for Shannon's sampling theorem to be fulfilled). Thus, the pixels of the digital image contain gray-values at, say, 8 bits per pixel. Since the original image was purely black and white, these gray-values are expressing the subpixel geometry of the depicted letters by measuring how much foreground and background was contained in the receptive field of the blurring kernel centered at each pixel. Now, when we binarise the digital image at the given resolution, we return to a representation with 1 bit per pixel, and the geometric information contained in the other 7 bits is lost. In contrast, when we interpolate the gray-value image threefold in each direction before binarisation, we transform each 8-bit pixel into nine 1-bit pixels, i.e. the overall information content is roughly preserved, see figure 1.6. Even better results are possible with a subpixel-accurate, polygonal edge representation, as seen in figure 1.3.

One may still object that these low-resolution situations are not very typical for the images we encounter in practice. After all, one tries to adjust the camera's resolution and field of view so that the objects of interest are clearly visible. However, in certain situations this may be impossible, for example in microscopy, where the objects of interest are often barely resolvable. Moreover, even when the objects themselves are large with respect to the pixel grid, the correct image interpretation may depend on very small details (cf. 1.7 left), or the boundaries of neighboring objects may almost touch each other (cf. figure 1.7 right). Similar effects can result from foreshortening and partial occlusion. In all these cases, one cannot circumvent the necessity of dealing with fine

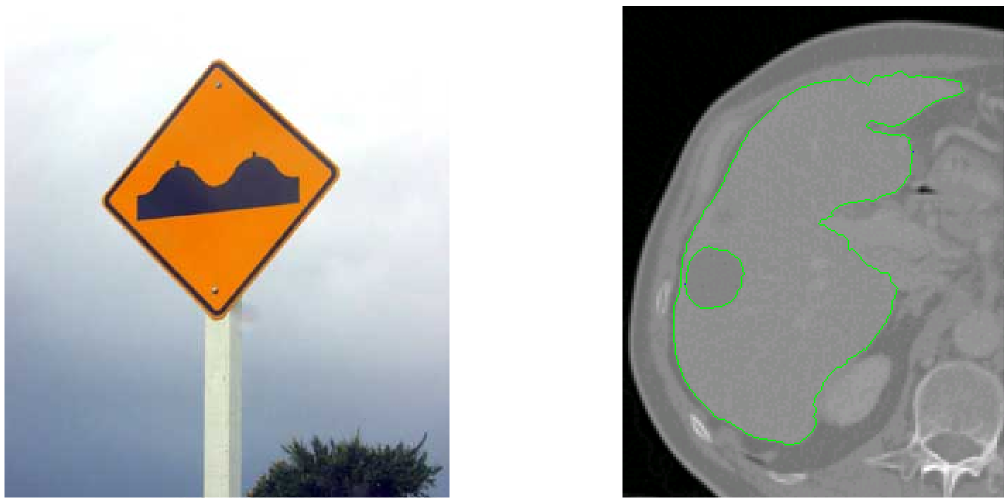


Figure 1.7: Left: The street bump sign is well resolved, but the point of the picture depends on two tiny details. Right: While both the tumor and enclosing liver are large objects in this CT image, their boundaries almost touch, and care is required to prevent unintentional region merging.

detail (possibly with subpixel accuracy) despite the objects themselves being big. Yet another nice example for the importance of image detail is depicted in figure 1.8: Coarse and fine scale interpretations of this (deliberately constructed) illusion are completely different.

Another source of difficulties is the fact that we use *discrete* devices to look at the *continuous* world. Unfortunately, the consequences of this mismatch are often not explicitly considered. For example, it is common to derive the theory of an image analysis module in the continuous domain. But when the theory is later realized on a discrete computer, there is often no formal proof that the discrete results are indeed close (i.e. within given error bounds) to the desired continuous ones. While such proofs are common in other fields like physics, they are not directly applicable to image analysis: Traditional discretisation analysis is typically conducted in the form of asymptotic convergence theorems and assumes that the resolution of the discretization can be refined as needed. In contrast, once an image of a scene has been taken, its resolution is fixed and cannot be refined arbitrarily. At best, one can take a new image at higher, but still finite resolution. Thus, a discretization analysis providing absolute error bounds for fixed resolution is needed. Since this is difficult, it has been suggested that discretization problems could be avoided when the module's theory were directly formulated in the discrete domain. Making sure that the module's computations conform to theory is then much easier. However, a new difficulty arises here: it is now unclear how computed results relate to phenomena in the real world, because the relationship between the discrete and continuous domains has not been dealt with. In order to really understand the performance of image analysis modules, we must analyze them in both the continuous and discrete domains, much like what Shannon did in his famous sampling theorem to connect continuous and discrete

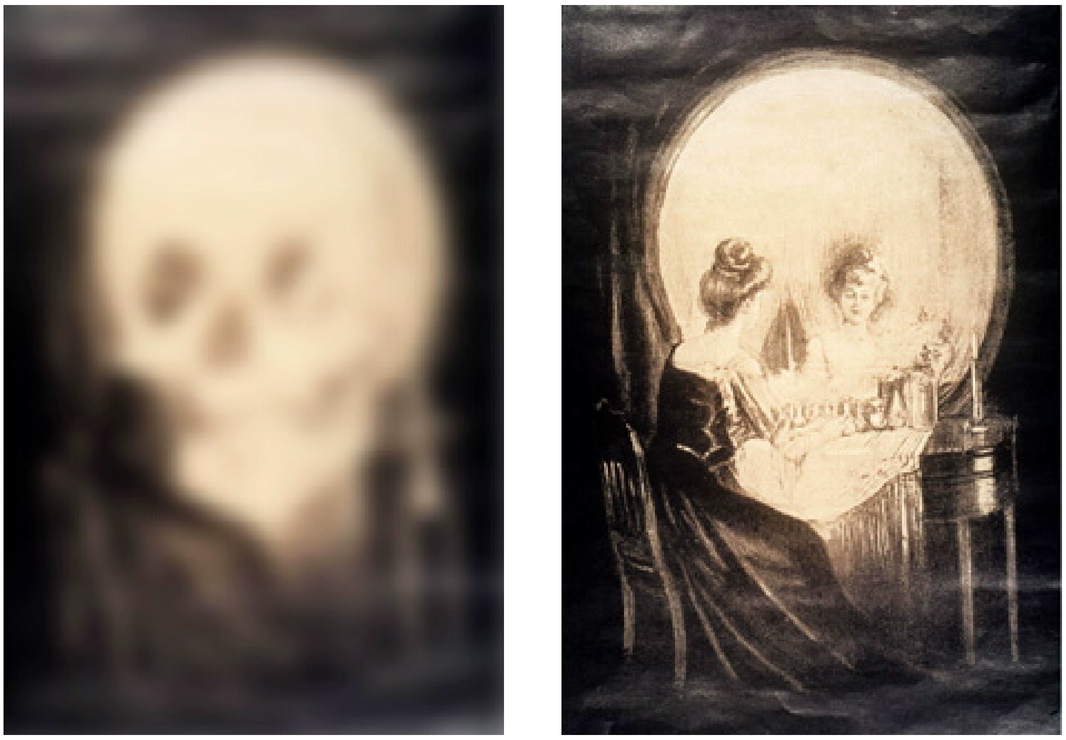


Figure 1.8: At coarse resolution, the image appears to depict a skull, whereas the detailed view reveals that we are looking at a woman in a mirror. While the illusion was constructed deliberately in this image, similar phenomena occur in real images as well.

signal representations.

This work is structured as follows: In chapter 2 we revisit the problems discussed here in a more formal way. We introduce our definition of low-level segmentation and review possible measures of success in this task.

Chapter 3 is devoted to a detailed analysis of the image acquisition process in cameras and the human eye. In particular, we investigate to what degree Shannon's sampling theorem is fulfilled by these devices, and how well the analog image function can be reconstructed from a given digital image. Spline interpolation and noise-normalization are introduced as key tools in this reconstruction process.

Chapter 4 introduces basic concepts of topology and discusses their consequences for image segmentation. In particular, it is shown that any sufficiently powerful representation for 2-dimensional segmentation results must deal with three feature types simultaneously, namely vertices, edges, and regions. Image analysis algorithms need explicit access to these features, and to the geometry of their embedding into the image plane. We introduce the GeoMap as a suitable abstract data type for these requirements, and propose several efficient implementation strategies.

Chapter 5 presents algorithms for GeoMap creation. Many of them are modifications of well-known segmentation algorithms, which are adapted to the new GeoMap framework.

Others, including variants of subpixel-accurate contour tracing and methods on the basis of Delaunay triangulations, have not previously been applied to image segmentation problems. These methods offer a wide choice of boundary detection options for application developers and can easily be exchanged because they are all based on the unified GeoMap framework.

In chapter 6 we analyze how the resolution of the sampling grid and the accuracy of edge detection methods influences the validity of segmentation results with respect to a given ground truth. We derive a number of geometric sampling theorems that state conditions under which important topological properties (such as number and neighborhood of regions) will be preserved despite the inherent information loss of digitization. For the first time, we are also able to give such guarantees for noisy and blurred images. This boundary sampling theorem may be considered as one of the central results of this work.

Chapter 7 investigates the accuracy of a number of gradient-based segmentation methods (e.g. Canny's algorithm, watershed transform) in both their pixel-accurate and subpixel-accurate versions. Formulas for systematic and statistical errors of edge, corner, and junction position and orientation, and for the likelihood of over- and undersegmentation are derived. Subsequent experimental evaluation shows remarkable consistency with theoretical predictions. Inserting the error bounds thus determined into the boundary sampling theorem derived in the previous chapter, one can predict whether or not the objects of interest can be reliably segmented in a given imaging situation.

In chapter 8, we apply the error analysis of the previous chapter to the problem of boundary tangent estimation. Several tangent estimators are analyzed both theoretically and experimentally, and it is found that in most situations the simplest algorithms are preferable.

Chapter 9 introduces a number of alternative boundary definitions on the basis of tensors, including anisotropic structure tensors and the boundary tensor. These methods are of interest because they carry the promise to improve the boundary accuracy near corners and junctions, which turned out to be a weak point of gradient-based algorithms in chapter 7. However, careful experimental evaluation of these algorithms shows that none of the alternative algorithms consistently outperforms the gradient. This finding is in line with the practical experience of many application developers, and puts the corner/junction problem as a high priority on the future research agenda.

2 The Low-Level Segmentation Problem

Abstract

Segmentation is usually defined as partitioning the image into meaningful regions. What is considered as “meaningful” depends on the application, but the term mainly refers to the correct delineation of certain scene objects of interest. We argue that one cannot reasonably expect that *low-level* segmentation (i.e. data-driven, bottom-up algorithms) will produce correct application-dependent boundaries, because this problem does in general require higher level information and feedback (i.e. top-down strategies). Therefore, we propose another, application independent definition of low-level segmentation: Its goal is the reconstruction of properties of an ideal geometric image from the information obtained from the real digital image. In this chapter, we investigate the implications of this new definition and analyze how one can objectively measure whether or not the goal has been achieved. We discuss the definition of ground truth and the problem of comparing a computed segmentation with the ground truth. In particular, we define the key notions of *structure preserving* reconstructions and of *r-similar* reconstructions.

2.1 Definition of the Problem

This work will be concerned with low-level image segmentation – but exactly what does that mean? It is clear that our goal of creating well-characterized modules for low-level image analysis can only be achieved when we are able to precisely define what these modules are supposed to do. Lack of precision in the definition of module behavior is a major reason for the difficulties in creating reliable computer vision systems. Vagueness already begins with the term *low-level*: obviously, the term refers to the first stages of image processing and analysis, but where exactly does low-level processing end?

In order to answer this question, we need to understand how information is transformed during the image acquisition process. Image acquisition creates a discrete 2-dimensional representation of a 3-dimensional scene. The relationship between the two domains is immensely complicated, because it depends on a large number of variables. In order to understand image acquisition, it must be split into several subprocesses, and it is not unreasonable to expect that image analysis – its inverse process – can be split into corresponding subproblems. A tremendously successful model for the first part of image acquisition is the notion of a *pinhole camera*: It performs a perfect perspective projection of the scene onto a plane which we call the *ideal geometric image*, see figure 2.1. The properties of this projection, the information loss involved, and the task of reconstructing 3D information from a set of projections are quite well understood and have received in-depth treatment in various monographs, e.g. [Faugeras & Luong 01, Hartley & Zisserman 04].

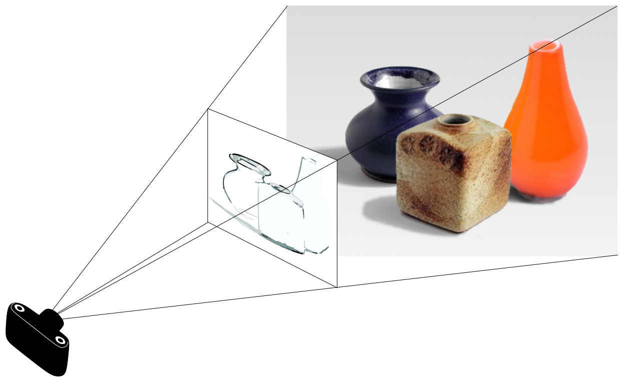


Figure 2.1: Observation of a scene by a camera. The ideal geometric image is created by the geometric rays passing through a virtual image plane in front of the camera (for clarity, only the objects' outlines are shown). It corresponds to perspective projection of the 3D scene onto a continuous 2-dimensional domain, as realized by a perfect pinhole camera.

The most important characteristic of the ideal geometric image is that it still resides in the continuous domain and has infinite resolution. However, the geometric image cannot be observed by any real device. Therefore, the second part of a realistic image acquisition model must describe how the ideal geometric image is transformed into the digital image which we actually get from a camera. The present work is based on the observation that the information loss incurring during the transition from the analog to the digital domain is as dramatic as the one due to projection from 3D to 2D. It is necessary to understand this information loss in order to build reliable image analysis systems. Therefore, we define the task of low-level image analysis as follows:

Definition 2.1. *The purpose of* low-level image analysis *is to recover information about the ideal geometric image from the corresponding digital image.*

According to this definition, low-level image analysis is *not* concerned with 3-dimensional objects – as soon as 3D information is required or has to be recovered, we are no longer speaking of low-level processing. It follows that the reconstruction of true object boundaries cannot be the goal of low-level analysis because it requires knowledge beyond what is available in purely 2D representations. We shall see that many interesting results can be obtained despite (or perhaps because of) restricting our attention to 2D[1].

From the point of view of low-level image analysis, the main advantage of introducing an ideal geometric image is the possibility to define "ground truth": by assuming that the ideal geometric image is given, we can create digital test images which reproduce, as well as possible, the corresponding output of real cameras. Low-level image analysis modules can than be characterized by comparing their results with the known ground truth. These comparisons realistically predict module behavior in real imaging situations

[1]Others put the borderline of low-level processing at the transition from a pixel-based representation to an initial symbolic (e.g. polygonal) representation. We prefer our definition because it is based on the *goal* of processing rather than its *means*.

if and only if the ideal geometric images and their transformations into digital images are modeled with sufficient realism.

Unfortunately, our understanding of the relationship between analog and digital representations in multi-dimensional spaces is not yet satisfactory. We provided some evidence for this fact in chapter 1. We are convinced that any realistic theory has to consider analog and digital representations with equal care. It is insufficient to rely on pure analog theories (e.g. differential equations), where digitization is merely treated as an implementation detail, and it is likewise insufficient to merely work in the digital domain (like in mathematical morphology) and ignore how the digital image was created in the first place. The first approach may be plagued with artifacts arising from the fact that the digital system is not a sufficiently accurate realization of the analog theory. In contrast, theory and implementation may match perfectly in the second approach, but the relationship between computed results and phenomena in the continuous real world remains unknown.

In the remainder of the present chapter we will look into the questions of how ideal geometric images can be modeled and how they can be compared with the output of image segmentation algorithms. In order to describe the structure of an ideal geometric image, we use the concept of *plane partitions*:

Definition 2.2. *A* partition P *of the plane* $\mathbb{R}^2$ *is defined by a finite set of points* $V = v_1, v_2, ...$ *called* vertices *and a set of pairwise disjoint* arcs $A = a_i \subset \mathbb{R}^2$ *such that every arc is a mapping of the open interval* $(0, 1)$ *onto the plane, the start and end points* $a_i(0)$ *and* $a_i(1)$ *are in* V *(but not in* a_i*). Start and end points need not be distinct. The union of the vertices and arcs is the* boundary *of the partition:* $B = V \cup A$*, and the* regions $R = r_i$ *are the connected components (i.e. maximal connected sets) of the complement of* B*. Since* B *is a closed set, all regions are open. Vertices, arcs, and regions are also called* 0-, 1-, *and* 2-cells *according to their dimensions.*

Since arcs are subsets of $\mathbb{R}^2$, the boundary is bounded and no arc runs to infinity. Consequently, there exists a single region containing the point at infinity called the *infinite* region of the plane partition. It describes the area outside of the actual field of view, and its interior is of no interest for image analysis. It should be noted that region boundaries form an explicit part of the representation. As we will see later, this is necessary for consistent topological interpretation of the partition, and allows edge- and region-based segmentation methods to complement rather than exclude one another.

Many image analysis tasks are simplified when the scene can be described by only two region types: “foreground” and “background”. In the terminology of plane partitions, this means that there exists a labeling of the regions with two labels such that every arc is in the closure of exactly one foreground and one background region. Such a plane partition is called *binary*. Many image segmentation algorithms produce binary image partitions, e.g. thresholding, snakes [Kass et al. 88] and most variants of the level-set method [Osher & Paragios 03], but all these methods share a big restriction: While it is possible for the fore- and background to consist of several connected components, the boundary cannot form junctions of odd degrees in a binary plane partition. Since junctions of degree 3 are very common in real images (especially T-junctions resulting from occlusion), binary partitions are not sufficient in many situations, so we will not restrict ourselves to binary partitions if possible.

Plane partitions are the basis of our definition of an ideal geometric image's content:

Definition 2.3. *Let P be a plane partition with boundary B and regions r_i, and let $b_{ij} = \partial r_i \cap \partial r_j \subseteq B$ the common boundary of regions i and j. The* indicator function *of region r_i is defined as $\rho_i : \mathbb{R}^2 \to [0,1]$ with the properties (i) $\rho_i(x,y) = 1$ in the interior of r_i, (ii) $\rho_i(x,y) = 0$ in the interior of r_i^C and (iii) $0 \leq \rho_i(x,y) \leq 1$ on the boundary ∂r_i such that $\sum_i \rho_i(x,y) \equiv 1$ holds in the entire plane. Furthermore, let $f_i(x,y)$ be bounded, continuous and possibly vector valued functions defined at least within the closure $\bar{r}_i$ of region i, such that $f_i(x,y) \neq f_j(x,y)$ holds for almost all points in the common boundary b_{ij}. Then the function:*

$$f_{geometric}(x,y) = \sum_i \rho_i(x,y)\, f_i(x,y) \tag{2.1}$$

is called an ideal geometric image.

Since the functions f_i and f_j are distinct on the common boundary b_{ij}, the geometric image is a piece-wise smooth function whose discontinuities occur on the boundary B. This piece-wise smooth image model was also employed by [Förstner 99]. It should be noted that the behavior of the function $f_{\text{geometric}}$ on the boundary B is largely irrelevant in practice because B is a set of measure zero. Since discontinuity between regions is only required almost everywhere, contrast inversion along an arc is explicitly allowed as long as the gray-levels are only equal in a single point. In practice, we want the functions f_i to be simple functions. When all f_i are constant, we arrive at the well-established step edge model.

While equation (2.1) is clearly a simplification of reality, it is more general than what may be apparent on first sight. Let us look at a number of possible objections:

- One may argue that some boundaries in the ideal geometric image will lack contrast (e.g. due to shading) and will not show up as discontinuities. This is true, but according to definition 2.1, the detection of these boundaries is outside the realm of low-level image analysis. It turns out that truly zero-contrast boundaries are actually rare in practice. Even if the contrast is very low, it is usually non-zero and detection of the boundary remains possible, see figure 2.2. The real difficulty lies in the distinction of weak, but important boundaries from slightly stronger, unimportant ones. This problem may or may not be solvable by low-level means.

- Shading and shadows can also cause the opposite kind of errors: true regions can be split up into several smaller ones. However, this is not a problem in the geometric image because it has infinite resolution. No matter how small the resulting regions are, they can be correctly represented here. The real problem is that the detection of small regions becomes more difficult in *digital* images, because the region size decreases relative to the sampling rate.

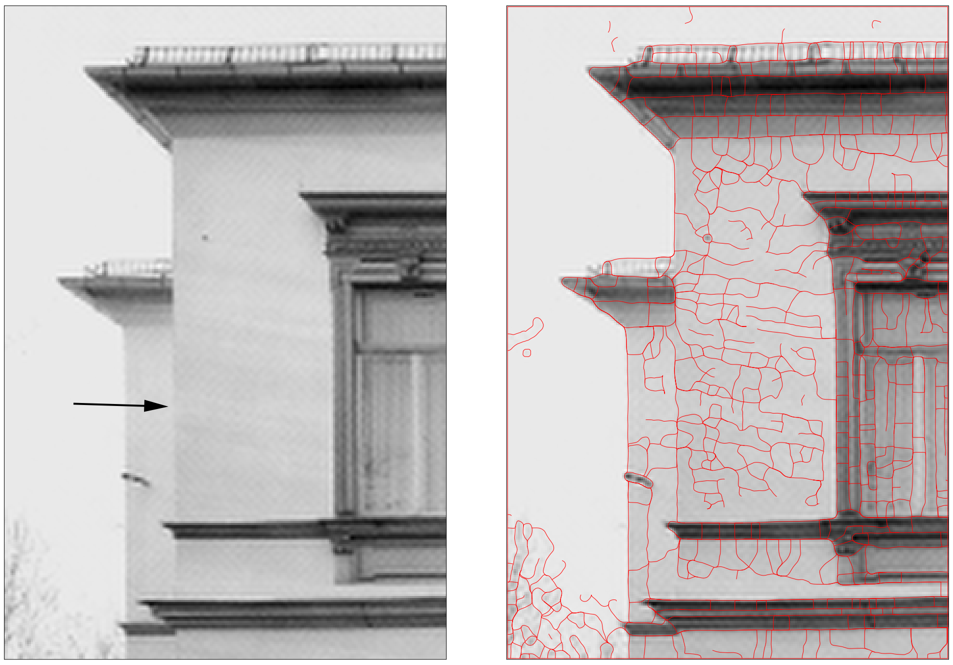


Figure 2.2: Left: The arrow marks a portion of the building's edge with very low contrast. Right: Edge detectors (here the subpixel watershed algorithm, see section 5.1.2) are still able to detect this edge with good accuracy. The real difficulty is to distinguish an important weak edge from irrelevant edges with similar contrast (e.g. from edges that are caused by noise).

- When regions in the geometric image are arbitrarily small (relative to the sampling rate of the digital image), their detection becomes impossible. So one may argue that our model raises an unsolvable problem. This is again true. But it is equally true that sufficiently large regions can be reconstructed correctly. Finding precise limits of reconstructability is one of the key questions in this work. Our distinction between geometric and digital images facilitates the search for these limits and allows us to investigate the artifacts occurring under insufficient resolutions.

- Another objection is that boundaries are not really discontinuous, but fuzzy. This is certainly true as soon as one arrives at atomic scales, but it can happen at much coarser scales due to diffraction at shadow edges or simply due to a fuzzy transition between neighboring objects (e.g. the edge of a cloud). However, in most cases these fuzzy transitions are very narrow in comparison to the size of the camera's point spread function and are, therefore, indistinguishable from true discontinuities in the digital image. Many of the detection methods discussed in this work will be applicable even if the scale of the transition is similar to the scale of the point spread function. Eventually, the geometric image model should be generalized to explicitly incorporate blurry edges, but the model yields sufficiently interesting results without this generalization.

- The restriction to simple functions f_i within each region may raise concerns whether textures can be treated in our model. To respond to this objection, we distinguish three kinds of texture: (i) Consider an area with many similar regions which are clearly separated at an infinite resolution, e.g. the leaves of a tree or the grains of sand. When the distance from the observer is large, these regions are no longer separable at the resolution of the digital image, but may still exhibit some regularity. These *geometric textures* are thus a way of dealing with objects that are too small to be resolved individually[2]. Clearly, geometric textures are covered by our framework. (ii) Some textures arise from noise introduced in the process of image acquisition. These textures are not a property of the geometric image, but must be modeled as part of the digitization process and are thus also covered by our framework. (iii) Finally, some textures arise because the surface reflectance of a 3-dimensional object is governed by a random process which cannot be explained in terms of constituting regions, even at an infinite resolution. These surfaces are rare in practice and can be handled by choosing appropriate f_i.

In conclusion, the property of *infinite resolution* makes the geometric image a sufficiently general model for the analysis of low-level image analysis. Generalizations concerning the complexity of the region functions f_i and of their discontinuities at the boundary can easily be incorporated when the present version (2.1) of the model becomes sufficiently understood. The problems of scale and finite resolution can already be studied without these generalizations because they arise when the geometric image is subjected to blurring and digitization in the camera and are not primarily properties of the geometric image itself.

Most boundaries in an ideal geometric image are closely related to the 3D boundaries of real world objects, because the projection of a 3D boundary usually gives rise to a 2D boundary. But there is no 1-to-1 correspondence, and we do not believe that perfect reconstruction of object boundaries is possible without explicit reference to 3-dimensional concepts. However, definition 2.1 restricts low-level processing to the 2-dimensional domain. We define *low-level image segmentation* accordingly:

Definition 2.4. Low-level image segmentation *is the task of recovering, as well as possible, the plane partition of ideal geometric images from the observed digital images by means of 2-dimensional bottom-up processing.*

While we concentrate on segmentation in this work, it should be stressed that it is not the only worthwhile goal of low-level image analysis. One may also base subsequent analysis on incompletely detected boundaries (as most authors do), isolated interest points [Mikolajczyk & Schmid 04, Lowe 04], or configurations of duplet/triplet features

[2] The human visual system can even observe geometric textures when the individual elements are still visible, e.g. the bricks of a wall, or the textons in Julesz' experiments [Julesz 81]. In this case, texture and textons are visible simultaneously, showing that human vision is a multi-scale process where the texture interpretation arises at coarser resolutions. This still fits in our definition by assuming that the texture interpretation is computed at the scale where individual elements become too small to be resolvable.

[Granlund & Moe 04, Johansson & Moe 05], to name just a few possibilities. The main difference between these approaches and segmentation is that they only provide sparse descriptions of the geometric image's properties, whereas segmentation produces a *dense representation*: every point in the plane is assigned to one subset of the partition. Segmentation plays a special role because sparse descriptions can always be derived from dense ones, but not vice versa. On the other hand, segmentation is a much more ambitious goal, and precise knowledge of its possibilities and limitations is highly desirable.

If ideal geometric images were directly observable, low-level segmentation would be trivial: one could simply look for discontinuities. But the digital camera image is a blurred and sampled version of the geometric image, and information about regions and their boundaries is no longer explicitly accessible. Due to the information loss, it is rarely possible to recover the original partition exactly. We investigate how well it can be approximated without using knowledge about the 3-dimensional nature of the problem. In other words, we attempt to give a precise meaning to the formulation "as well as possible" by investigating the theoretical limits of various approaches to low-level segmentation.

It is interesting to contrast our notion of low-level segmentation with that of traditional definitions. Most frequently, segmentation has been defined as a maximal partition of the pixel plane such that each region is homogeneous according to some criterion. Similarly, edge detection has been defined as the process of marking local maxima in the rate of change of some criterion. These definitions are based on the *means* employed in the solution (homogeneity or inhomogeneity of a criterion). In contrast, our definition states the *goal* of the module's action (reconstruction of the geometric image). We prefer this definition because it implies a method of *performance evaluation* (quantitative comparison of the module's output with a known correct solution), whereas the traditional definitions fail to suggest methods which compare the performance of different criteria. In the remainder of the present chapter, we will discuss how a segmentation process's success according to definition 2.4 can be measured, and why some apparently good measures of correctness may not in practice be sufficient.

2.2 Measures of Success

2.2.1 Similarity between Plane Partitions and their Reconstructions

Due to information loss during digitization, the reconstructed shape is almost never identical to its continuous original. Therefore, we must first define precisely how we are going to quantify how well the reconstruction algorithm performed. To make the definition useful in image analysis, it should correspond closely to human perception of shape similarity. In the sequel we discuss a number of common possibilities and analyze their advantages and limitations. Many important properties of shape reconstruction can be understood by investigating whether the relationship between a region and its complement changes under the reconstruction. Therefore, we illustrate most of our arguments with binary partitions, but they are equally valid for non-binary partitions.

When we ask if the shapes in a partition have been preserved, we must look at two fundamentally different properties: topological similarity and geometric similarity. Topology answers qualitative questions like "Has the number of regions changed?", "Are neighborhood relations between regions preserved?", "Has a reconstructed region as many holes as the corresponding original?". In contrast, geometry provides quantitative measures of similarity such as "How much has a reconstructed region been distorted relative to the corresponding original?". Topology describes relations between sets as a whole, whereas geometry describes the relation between individual corresponding points.

Correspondence between shapes is easily established when the shapes are topologically equivalent:

Definition 2.5. *Two sets A and A' are topologically equivalent if there exists a homeomorphism $f: A \to A'$. A homeomorphism is a bijective mapping between A and A' such that both f and f^{-1} are continuous.*

In 2D, this implies that the number of connected components and the number of holes in each component must be preserved. Moreover, open and closed sets must remain open or closed respectively. Consider, for example, figure 2.3. The gray region in subfigure (b) may be considered the reconstruction of the gray shape in figure (a) on the depicted square grid. [3] In this case, the reconstruction is obviously topologically equivalent to the original. However, the same is true in subfigures (c) and (d): although we do not perceive the reconstruction on the coarser square grid as correct, the two sets are topologically equivalent. This happens because topological equivalence is defined independently of any embedding space, and the blob within the hole of the original shape can be mapped onto the upper blob of the reconstruction. When the shapes in figure 2.3(c) and (d) are embedded in $\mathbb{R}^3$ (i.e. the images represent slices through the 3-dimensional space), it is easy to imagine a continuous transformation moving the small black blob from the hole to the outside. But human perception implicitly assumes a 2-dimensional embedding. Then, not only the topology of the shapes, but also the topology of their 2D complements (i.e. the white background regions) must be preserved. The homeomorphism must therefore be extended to the entire embedding plane into an $\mathbb{R}^2$-*homeomorphism*. Intuitively, an $\mathbb{R}^2$-homeomorphism acts as if the plane were a rubber membrane that can be stretched in arbitrary ways, but may not be cut or folded. The concept is also known as a *morphing* transformation. Under this stronger requirement, only the shapes in figure 2.3(a) and (b) are topologically equivalent, because no $\mathbb{R}^2$-homeomorphism can possibly move the small blob out of the enclosing hole. Generalizing this to arbitrary partitions, we arrive at the following definition:

Definition 2.6. *Two plane partitions $P_1$and P_2 are topologically equivalent, when there exists a homeomorphism $f: \mathbb{R}^2 \to \mathbb{R}^2$ that maps every cell of P_1 onto a cell of P_2.*

Note that all cells are mapped simultaneously by a single homeomorphism. The homeomorphism establishes a 1-to-1 correspondence between the cells of P_1 and P_2. Obviously, the dimensions and neighborhood relations of all cells are preserved in this mapping.

[3] The illustrations in this section are mostly taken from [Stelldinger & Köthe 05]. In that paper, the reconstruction is defined as the union of all pixels whose sampling point is within the original set. A pixel on an arbitrary (even irregular) grid is defined as the Voronoi region around a sampling point, see our definition 4.9. However, the problems illustrated in this section are not specific to this particular reconstruction method.

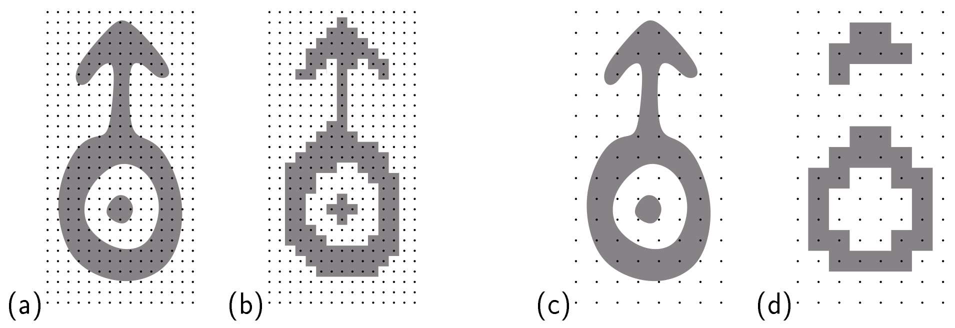


Figure 2.3: (a) and (b): The reconstruction on a high-resolution grid is topologically equivalent to the original and perceived as correct; (c) and (d): At lower resolution, the reconstruction is incorrect because the black blob in the large black region's hole disappears, while a new blob appears in the exterior region. However, the gray sets are still topologically equivalent, because the embedding space is irrelevant in definition 2.5. The reconstruction error is only captured by considering homeomorphisms of the entire plane $\mathbb{R}^2$ (definition 2.6), i.e. by requiring topological equivalence of the background as well.

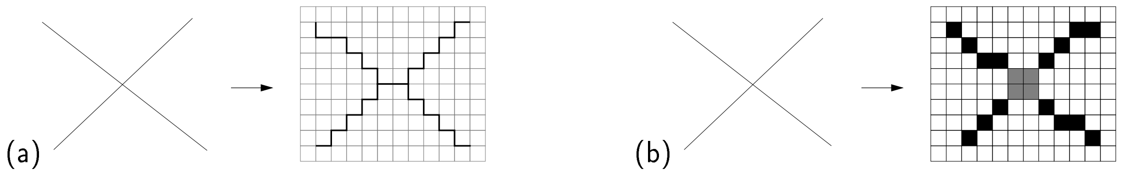


Figure 2.4: Examples where the digitization of Euclidean lines does not preserve topology: a) a junction of degree 4 is split into a pair of junctions of degree 3 (the lines are digitized into interpixel edges); b) the reconstructed arcs and vertices have non-zero area (the lines are digitized into 8-connected pixel chains). A formal definition of these digitization methods is given in section 4.3.2.

Unfortunately, it is very difficult to guarantee complete topological equivalence in low-level segmentation. A common problem is that 4-junctions often get split up into two 3-junctions, see figure 2.4a. The digitized lines are no longer homeomorphic to the original. The same happens when the digitized lines have non-zero area, whereas the original lines formed a set of measure zero, figure 2.4b. This figure illustrates the problem with pixel-based digitization methods, but it can also occur in alternative approaches, e.g. shape reconstruction by means of Delaunay triangulation (see section 5.3). Our intuition considers the digitizations in figure 2.4 as correct, although they are not homeomorphic to the original. We should therefore relax the notion of shape similarity to include these cases. This can be achieved by means of *homotopy*:

Definition 2.7. *Two mappings $f, g : A \to A'$ are called* homotopic*, when there exists a continuous mapping $F : A \times [0, 1] \to A'$ such that $F(a, 0) = f(a)$ and $F(a, 1) = g(a)$ for all $a \in A$. Two sets A and A' are of the same* homotopy type *when there exists two mappings $f : A \to A'$ and $g : A' \to A$ such that the composition $g \circ f : A \to A$ is homotopic to the identity map $i_A : A \to A$, and the composition $f \circ g : A' \to A'$ is homotopic to the identity map $i_{A'} : A' \to A'$.*

In the 2-dimensional domain, the relation between homotopy and plane partitions is most easily understood by building the *homotopy tree* introduced in [Serra 82]. Serra's original definition applies to a given set in the plane and its complement. We can apply this concept to plane partitions by identifying the boundary B with the given set, and the regions with the complement set. This leads to the following algorithmic definition:

Definition 2.8. *Let P be a plane partition with regions $R = \{r_i, i = 1, ...\}$ and boundary B, and denote by b_j the connected components of the boundary. The* homotopy tree *of P is created as follows: take the infinite region r_0 as the root of the tree. Let B_0 be the set of boundary components that intersect the closure of r_0. The members of B_0 become the children of node r_0. Then for every $b_j \in B_0$ find the set $R_j \subset R \backslash \{r_0\}$ of regions whose closure intersects b_j. The members of R_j become children of b_j. Repeat this process recursively until all elements of P have been added to the tree.*

In other words, the tree is built by tracing the incidence relation between regions and boundary components from the exterior of the image toward the interior. Starting with a root node representing the infinite region, the tree consists of alternating levels of boundary and region nodes. The creation procedure works just as well when the boundaries have non-zero area. Homotopy trees are significant in our context due to the following fundamental property:

Theorem 2.1. *Let P be a plane partition and $\hat{P}$ its reconstruction. P and $\hat{P}$ are of the same homotopy type if and only if their homotopy trees are isomorphic.*

The original proof in [Serra 82] is again easily adapted to plane partitions by considering boundaries and open regions as complementing sets. When the homotopy trees of P and $\hat{P}$ are isomorphic, there exists a 1-to-1 mapping between their regions and boundary components such that all corresponding entities are of the same homotopy type. Now, the differences between the original and reconstructed boundaries depicted in figure 2.4 are explicitly permitted. However, it is also permitted that reconstructed regions exhibit "isthmuses", i.e. they may only be connected through a single point, see figure 2.5. This is impossible in the original partition because regions are required to be open sets. It can happen in the reconstruction when more than three pixels meet at a single pixel corner (i.e. it can happen in a square grid, but not in a hexagonal one). Such configurations are undesirable because they give raise to the infamous *connectivity paradox* [Rosenfeld 70]. To avoid this, we require topological equivalence between regions in order for the reconstruction to be considered successful:

Definition 2.9. *Let $P = B \bigcup \cup_i r_i$ be a plane partition with boundary B and regions r_i, and let $\hat{P} = \hat{B} \bigcup \cup_j \hat{r}_j$ a reconstruction of P with boundary $\hat{B}$ and regions $\hat{r}_j$. The reconstruction is called* structure preserving *when the homotopy trees of P and $\hat{P}$are isomorphic, and corresponding regions are homeomorphic.*

The points in a structure preserving reconstruction can be mapped onto the points of the original partition, and the mapping is 1-to-1 for points within regions, and possibly 1-to-many for points in the boundary. This mapping ensures that (i) reconstructed regions

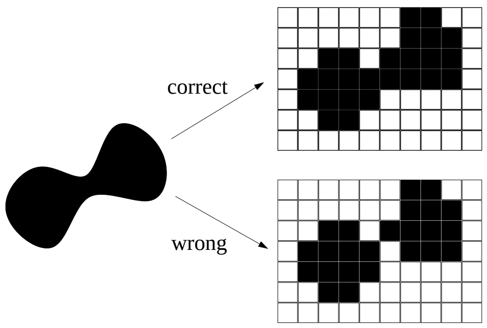


Figure 2.5: When the reconstructed region is represented as the union of pixels whose centers are in the original region, the shape of the reconstruction depends on the relative position between the pixels and the original region. It may happen that the reconstructed set is only 8-connected, i.e. only connected through a single point of the plane (an “isthmus”). This reconstruction is not homeomorphic to the original region, although the homotopy type is preserved.

are open sets (i.e. have no isthmuses), (ii) the number of regions, the number of holes in every region, and the inclusion tree between regions are all preserved, and (iii) the reconstructed boundary is of the same homotopy type as the original boundary. The definition also allows a topological classification of various reconstruction errors:

Definition 2.10. *Let the homotopy trees of P and $\hat{P}$ be different. Create a new partition P' by adding a single arc a' to P (when the start and end points of a' coincide, adding a new vertex there is also permitted). When the homotopy tree of P' is isomorphic to the one of $\hat{P}$, we say that $\hat{P}$ contains a* spurious *or* false positive *boundary (namely the part of $\hat{B}$ that is mapped onto a' by the homotopy). Likewise, create P'' by removing an arc (and possibly an isolated vertex) from P. If the homotopy trees of P'' and $\hat{P}$ are isomorphic, the removed arc is missing in $\hat{P}$or a* false negative.

In practice, P and $\hat{P}$ will often differ by more than a single arc. Then, the transformation of P will involve several arc additions and/or removals. In the most general case, this may amount to isomorphic subgraph matching, which is a known NP-hard problem. We will come back to the issue how to determine false positives/negative at the end of this chapter. Here, we just assume that we can somehow determine their *numbers*. These numbers are usually dependent on the parameters of the reconstruction algorithm: when the sensitivity of the algorithm is increased, the number of false negatives (missed boundary parts) will decrease, but the number of false positives (spurious boundary parts) will increase, and vice versa. The optimal trade-off depends on the application. However, it is still possible to obtain an application-independent characterization of reconstruction performance by means of the *receiver operating characteristic* (ROC). Let TP, FP, TN, and FN be the numbers of true and false positives, and true and false negatives respectively, measured with a particular choice of algorithm parameters on a particular set of test images. To be comparable, these numbers must be normalized. *Sensitivity* or *recall* of the reconstruction algorithm is defined as

$$\text{recall} = \frac{TP}{TP + FN}$$

Its *miss rate* is

$$\text{miss rate} = 1 - \text{recall} = \frac{FN}{TP + FN}$$

the *specifity*

$$\text{specifity} = \frac{TN}{TN + FP}$$

and the *false alarm rate* is

$$\text{false alarm rate} = 1 - \text{specifity} = \frac{FP}{TN + FP}$$

All four characteristics are between zero and one. When we adjust the algorithm parameters so that the reconstruction doesn't contain any boundary at all, TP and FP are zero, and therefore recall and false alarm rate are zero as well. Similarly, when the algorithm assigns all possible points to the boundary, TN and FN are zero, and recall and false alarm rate assume a value of one. For other choices of algorithm parameters, one obtains (recall, false alarm rate) pairs between these extremes. Every pair defines a point in the so called ROC diagram, where recall is associated with the abscissa, and false alarm rate with the ordinate. A reconstruction algorithm that just guesses will produce pairs on the straight line where recall and false alarm rate are equal. Points above this line indicate better-than-chance performance. The upper convex hull of all points is a monotonically increasing curve from $(0, 0)$ to $(1, 1)$ called the *receiver operating curve* (ROC curve). The area A_{ROC} under this curve is a useful characterization of the reconstruction algorithm in a single, parameter-independent number. $A_{\text{ROC}} = 0.5$ for a reconstruction algorithm that just guesses, and $A_{\text{ROC}} = 1$ for a perfect algorithm. Some authors prefer equivalent ROC-diagrams where the false alarm rate is drawn against the miss rate. This is just the previous diagram rotated by 90° counter-clockwise. Now, the lower convex hull running from $(0, 1)$ to $(1, 0)$ is the curve of interest (lower curves are better), and $A'_{\text{ROC}} = 1 - A_{\text{ROC}}$ should be small.

Alternatively, one can compute a *precision-recall curve*, where recall is plotted along the abscissa, and precision $= \frac{TP}{TP+FP}$ along the ordinate. The upper convex hull again characterizes algorithm performance, and higher curves are preferable. Unfortunately, the PR curve becomes unstable at low recall due to the small total number of responses under this condition (when there is only one response, it can either be a true or false positive, resulting in precision values of 1 and 0 respectively), and approaches the fraction of true positives among all possible responses at high recall (which is undesirable because it is a sample-dependent quantity). Unlike A_{ROC}, the area under the PR curve doesn't have a clear interpretation in terms of chance vs. perfect performance. On the other hand, differences between algorithms on the *same* data set are sometimes more pronounced in the precision-recall curves than in the ROC.

Another value summarizing the overall performance of an algorithm is the *cross-entropy loss*. It can be computed when a set of candidate boundary parts is given and the algorithm analyzes whether each candidate actually represents a boundary piece of the ground-truth partition. This is a common approach to reducing oversegmentation. The algorithm to be evaluated must estimate the probability p_i for candidate i to be a part

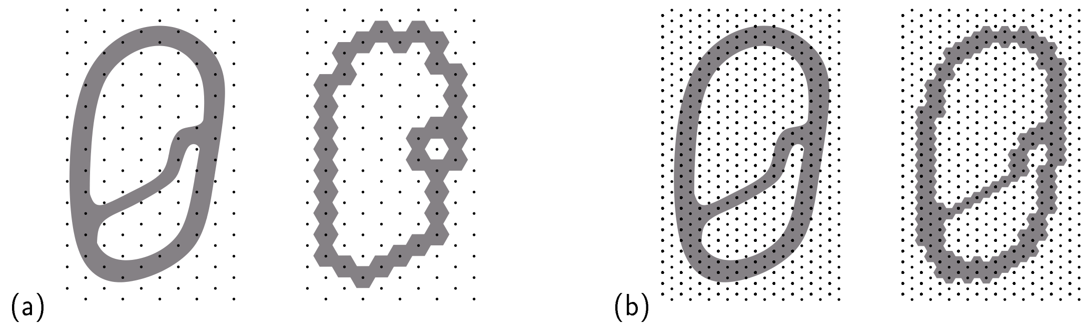


Figure 2.6: a) At low resolution, the reconstruction is topological equivalent (and therefore structure preserving) to the original plane partition, but the geometric similarity (as measured by the Hausdorff distance) is not very high because one of the holes is too small in the reconstruction. b) The Hausdorff distance between original and reconstruction is small at higher resolution.

of the true boundary instead of performing a hard classification. Then the cross-entropy loss is defined as

$$L = -\frac{1}{|I_P \cup I_N|}\left[\sum_{i \in I_P} \log p_i + \sum_{i \in I_N} \log(1 - p_i)\right]$$

where I_P and I_N denote the sets of true positives and true negatives respectively. L vanishes when the reconstruction is perfect, i.e. when $p_i = 1$ for all true positives and $p_i = 0$ for all true negatives.

Even when the reconstruction is structure preserving or topologically equivalent to the original partition, this does not automatically mean that the two partitions are perceived as similar. Frequently, two topological reconstruction errors compensate each other. But this usually requires to map some points of the reconstruction onto relatively distant points of the original. Consider, for example, figure 2.6a: the black region has two holes in both the original partition and the reconstruction. Yet, due to sampling problems, one hole is much too small in the reconstruction, so that any morphing transformation between the two shapes necessarily requires significant displacements. Shape comparison should therefore include a measure of "nearness", i.e. geometric similarity. The simplest such measure is the *Hausdorff distance*:

Definition 2.11. *The* Hausdorff distance d_H *between two sets* A *and* A'*is defined as*

$$d_H(A, A') = \max\left(\max_{a \in A} d\left(a, A'\right), \max_{a' \in A'} d\left(a', A\right)\right)$$

where $d(a, A') = \sup_{a' \in A'} d\left(a, a'\right)$ *is the distance between point* a *and set* A'*, and* $d(a, a')$ *is the Euclidean distance between points* a *and* a'*.*

The Hausdorff distance is a metric. In the context of shape similarity, it is most useful to measure the Hausdorff distance between the boundaries B and $\hat{B}$ of the original and reconstruction respectively. On a square raster, this can be efficiently computed by means of the distance transform of the boundary pixels, whose computation takes only linear time in the total number of pixels. The examples in figure 2.6a and b can now be distinguished by their Hausdorff distance. However, the Hausdorff distance is extremely sensitive to outliers: a single wrong point can lead to a very big value of the Hausdorff distance, even if the sets to be compared are otherwise identical. In contrast, when the two sets are distorted everywhere, but only by a small maximum amount, this situation cannot be distinguished from the case where there is only a single localized distortion. Therefore, [Baddeley 92] proposed another distance that averages displacements over an entire set C of points:

Definition 2.12. *Let C be a domain such that $A \subset C$ and $A' \subset C$. Than the family of* Baddeley distances *with $p \in [1, \infty)$ is defined as*

$$d_{B,p}(A, A') = \left(\frac{1}{|C|} \int_C \left| d(c, A) - d(c, A') \right|^p \, dc \right)^{1/p}$$

This is a metric for all $p \in [1, \infty)$. Baddeley uses $p = 2$ in his experiments. The domain C can be the entire image or a subset of it. When the distance between the boundaries B and $\hat{B}$ is to be computed, it is useful to choose C as a strip of width μ around the two boundaries: $C = (B \oplus \mathcal{B}_\mu) \cup \left(\hat{B} \oplus \mathcal{B}_\mu\right)$ where $\oplus$ denotes dilation, and $\mathcal{B}_\mu$ is the open ball with radius μ ($\mu = 5$ in Baddeley's experiments). Restricting C to this strip reduces the influence of topological errors (false positives and false negatives) and thus increases the sensitivity of $d_{B,p}$ to geometric accuracy. In a discrete domain, the integral over C is replaced with a sum over the pixels in C. Like the Hausdorff distance, Baddeley's distance can be efficiently computed on a raster by means of the distance transform, which takes linear time in the number of pixels. Figure 2.7 demonstrates the difference between the two distances. We observe that Baddeley's distance is more informative than the Hausdorff distance, although it seems that the penalty given to a series of many small gaps (third image) is somewhat too small.

Geometric distances cannot directly signal topological errors. Nevertheless, they can be used to get an approximate idea of topological accuracy by replacing the topological error definition 2.10 with a geometric one, e.g. [Heath et al. 97]: a point is considered an outlier (false positive) if its distance from the correct boundary exceeds a certain value. A piece of the correct boundary is considered missing (false negative) when its distance to the reconstructed boundary exceeds a certain value. The thresholds for these decisions should be slightly higher than the expected maximum geometric error on correctly detected boundaries (see section 7.2.2.3). The geometric definitions lead to simpler descriptions of the similarity between original and reconstruction, because geometric errors are relatively easy to compute.

High geometric similarity (i.e. small geometric distance) is not sufficient to ensure that reconstructions are perceived as being similar to the original. Figure 2.8 demonstrates this. It is necessary to require both: preservation of topology and high geometric similarity. In [Stelldinger & Köthe 05], we proposed the notion of *weak r-similarity* to describe

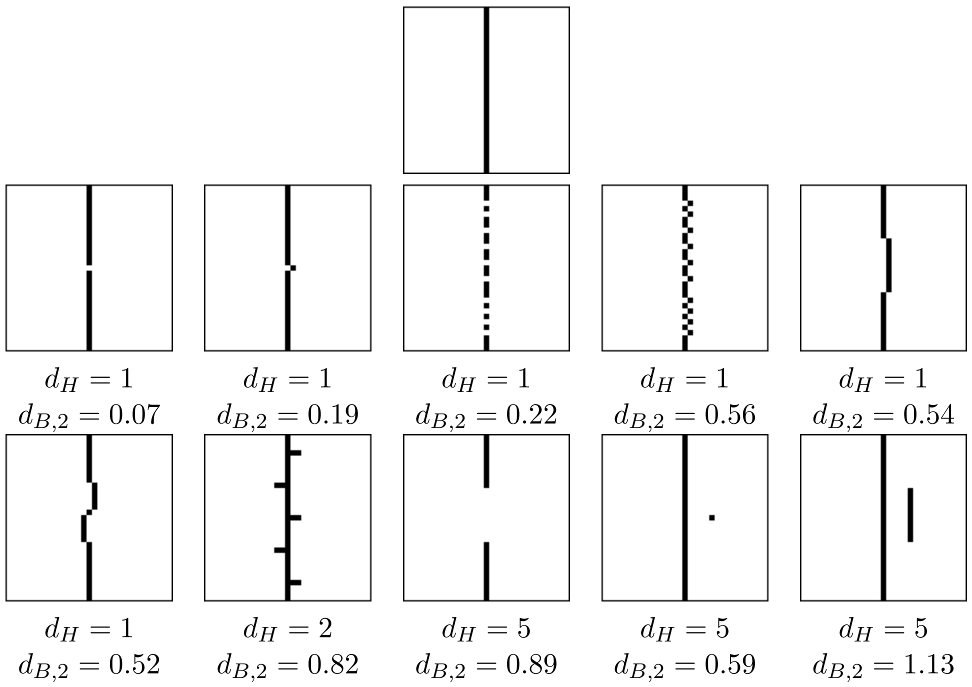

$d_H = 1$ $d_{B,2} = 0.07$
$d_H = 1$ $d_{B,2} = 0.19$
$d_H = 1$ $d_{B,2} = 0.22$
$d_H = 1$ $d_{B,2} = 0.56$
$d_H = 1$ $d_{B,2} = 0.54$
$d_H = 1$ $d_{B,2} = 0.52$
$d_H = 2$ $d_{B,2} = 0.82$
$d_H = 5$ $d_{B,2} = 0.89$
$d_H = 5$ $d_{B,2} = 0.59$
$d_H = 5$ $d_{B,2} = 1.13$

Figure 2.7: Comparison between the Hausdorff distance d_H and Baddeley's distance $d_{B,2}$ (with $\mu = 5$). The distorted images differ from the ground truth (shown at the top) by 1 or 10 false positives and/or false negatives.

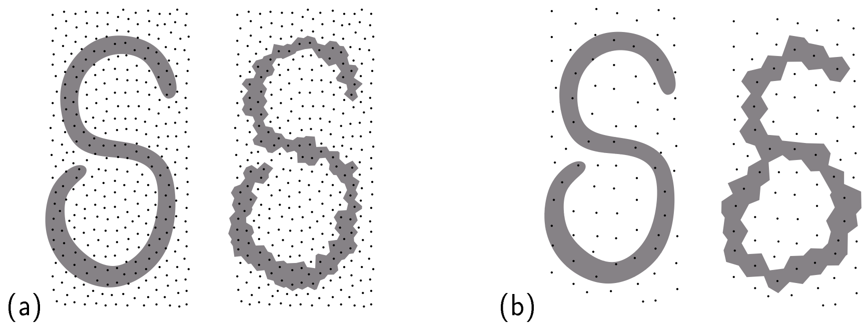


Figure 2.8: a) At high resolution, the letter S is reconstructed correctly. b) At lower resolution, the two shapes are not topologically equivalent and are not perceived as similar, although both the Hausdorff and Baddeley distances between original and reconstruction remain small.

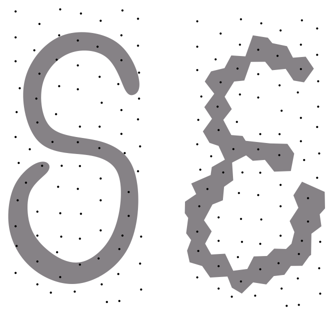


Figure 2.9: Even when the original and reconstructed shapes are topologically equivalent and have small Hausdorff distance, perceptual similarity is not guaranteed: the s may turn into an ε at low resolution.

this requirement for binary plane partitions. The definition is generalized to arbitrary partitions as follows:

Definition 2.13. *A partition P and its reconstruction $\hat{P}$ are* weakly r-similar *when the reconstruction is structure preserving, and the Hausdorff distance between the boundaries B and $\hat{B}$ is at most r.*

Unfortunately, not even weak r-similarity (with some small r) is always sufficient. Figure 2.9 gives an example where the sampling grid differs only minimally from the one in figure 2.8b. Now, the two shapes are topologically equivalent and have small Hausdorff distance, but are still perceived as different. This can be explained as follows: the supremum operations in the Hausdorff distance map every point of A to the nearest point in A' and vice versa. In general, these two mappings are different, and they also differ from the mappings that ensure topology preservation: since many points can be mapped to the same nearest point, the former mappings do not usually define a homeomorphism and are not even structure preserving. This problem can be avoided when we force the mappings for topological and geometric comparisons to be the same. A homeomorphism implies geometric similarity when it doesn't displace the points very much:

Definition 2.14. *A homeomorphism $f : A \rightarrow A'$ with $A, A' \subseteq \mathbb{R}^2$ is called an r-homeomorphism, if $|f(a) - a| \leq r$ for all $a \in A$.*

A thus restricted homeomorphism defines a *morphing distance* $d_M(A, A') = r$ between the sets A and A'. The morphing distance is a metric because the concatenation of an r-homeomorphism and a s-homeomorphism is at least an $(r+s)$-homeomorphism. [Stelldinger & Köthe 05] introduced the term *strong r-similarity* when the morphing distance between a binary plane partition and its reconstruction does not exceed r. We again generalize the definition to arbitrary plane partitions:

Definition 2.15. *A partition P and its reconstruction $\hat{P}$ are* strongly r-similar *when the reconstruction is structure preserving, and the homeomorphisms f_i between corresponding regions r_i and $\hat{r}_i$ are r-homeomorphisms.*

When the reconstruction is not only structure preserving, but even topologically equivalent to the original (i.e. there is also a homeomorphism between the boundaries), strong r-similarity implies that a single r-homeomorphism exists which maps the entire reconstruction onto the original. Then the topology is not only preserved globally, but also locally in the following sense: Pick any open subregion of $\mathbb{R}^2$ that is partitioned by the boundary B in a certain way (including junctions, holes etc.). Then the existence of a homeomorphism ensures that a corresponding open region exists which gets partitioned in the same way by the reconstructed boundary $\hat{B}$. Existence of an r-homeomorphism tightens this property so that the corresponding region is not just somewhere in the plane, but is at most at distance r from the original region, i.e. in a localized neighborhood. Since d_M is an upper bound on the Hausdorff distance, strong r-similarity implies weak r-similarity. The undesirable reconstructions shown in the previous figures are not strongly r-similar to their original for reasonably small r (i.e. $d_H \ll d_M$). Strong r-similarity requires sufficiently high resolution in these examples.

Unfortunately, computing the morphing distance between two regions is very difficult, because an uncountable number of transformations must be considered. In practice, it is often only possible to derive upper bounds for the morphing distance, e.g. by direct proof (see for example section 6.1.1), or by restricting the permissible class of homeomorphisms, e.g. to thin-plate-spline transformations (see for example [Belongie et al. 02]). Often, we must settle on weak r-similarity, despite its shortcomings.

A possible break-through methodology that avoids many complications discussed above may be the *Normalized Probabilistic Rand Index* (NPR index) recently introduced by [Unnikrishnan et al. 07]. Instead of evaluating the correctness of every point in the reconstruction separately, these authors propose to look at all *point pairs*. Then the question is no longer whether a point is correctly assigned to a particular region or boundary, but whether its *relation* to all other points is preserved with respect to the ground truth segmentation. This approach has two advantages:

- It can handle incomplete or weak ground truth. This is especially important when algorithms are tested against manual ground truth from human observers, because segmentations of different persons (or of the same person at different times) will rarely be in exact agreement.

- It avoids the difficult matching problem of measured features against ground truth features.

The NPR index approach assumes that the probability p_{ij} that points i and j belong to the same feature can be estimated from the ground truth. For example, if a set of G manual ground truth segmentations is available, and all agree that the two points belong to the same feature (i.e. to the same region), then $p_{ij} = 1$. If all agree that they do belong to different features, $p_{ij} = 0$. In general, $p_{ij} = \mathrm{count}(l_i = l_j)/G$ (where $\mathrm{count}(l_i = l_j)$ is the number of observers that assigned i and j to the same feature), so that values p_{ij} between 0 and 1 signal disagreement between observers. Now let $c_{ij} \in \{0, 1\}$ be the decision of the segmentation algorithm to be evaluated whether i and j belong to the same feature. Then the ***probabilistic rand index*** of a test segmentation S_{test} relative to a set $\{S_{\text{GT}}\}$ of ground truth segmentations is defined as

$$\text{PR}\left(S_{\text{test}}, \{S_{\text{GT}}\}\right) = \frac{1}{\binom{N}{2}} \sum_{i,j>i} \left[p_{ij}^{c_{ij}} \left(1 - p_{ij}\right)^{1-c_{ij}}\right]$$

In order to be comparable across images, this index has to be normalized. Thus, the NPR index is defined as

$$\text{NPR} = \frac{\text{PR} - E\left[\text{PR}\right]}{1 - E\left[\text{PR}\right]}$$

where $E\left[\text{PR}\right]$ is the expectation of the probabilistic rand index across the entire set of test images. NPR values above zero indicate that an algorithm performs better than chance. [Unnikrishnan et al. 07] report very promising results for this performance measure on the Berkeley Segmentation Dataset (cf. section 2.2.2.3 below). Further details and an algorithm for efficient computation of the NPR index can be found in that paper.

2.2.2 Ground-Truth Definition and Matching

The comparison methods described in the previous section critically depend on two requirements: (i) We need realistic digital images together with the associated ground-truth partition. (ii) We must be able to find best matches between the cells of the ground-truth partition and its reconstruction. These are very difficult problems, and no completely satisfying solutions exist so far. Three basic approaches can be distinguished for ground-truth definition:

Simulation of the camera: An artificial geometric image is defined in the continuous domain, e.g. as a polygonal plane partition with constant intensities f_i in every region r_i. A digital image is then created computationally, by applying a suitable image acquisition model to the geometric image. This method yields very accurate correspondences between geometric and digital images. It can make good predictions of real algorithm behavior when the model matches the real situation. Unfortunately, this is often not the case. Test images are frequently lacking geometric configurations which are notoriously difficult to analyze, for example small or narrow regions, complex junctions, contrast inversions, occlusion, or non-constant f_i. The simulated digitization process often employs unrealistic camera models. Therefore, the simulation approach earned a reputation of making unreliable performance predictions.

Better simulated images can in principle be obtained by ray tracing of a geometric scene model. However, this technique may not be as realistic as necessary because existing ray tracing algorithms approximate real image acquisition only to the point where the results appear convincing for a human observer, and will take arbitrary shortcuts when humans cannot see the difference. Further research is necessary to turn ray tracing into a valid simulation tool for the evaluation of image segmentation algorithms.

Manual ground-truth: Lack of realism can be avoided when actual camera images are used instead of simulated ones. This poses the problem of how to find the original geometric image. Marking the ground-truth partition manually is still the best and most common method. However, even when experts are performing the manual segmentation there is considerable disagreement: Exactly where is the correct boundary? Does a particular boundary belong to the ground truth or not? One reason is the subjectivity of human judgments, which should be reduced by creating "consensus ground-truth" from the responses of many uncorrelated observers. Recently, this approach was extended to a set of algorithms whose errors are uncorrelated [Yitzhaky & Peli 03]. Automatic consensus ground-truth, while still in its infancy, may eventually solve the subjectivity problem, and will provide much larger test image collections than are possible with human labor.

Another factor might be even more important for the success of manual ground-truthing: the human experts are working on the same digital image as the segmentation algorithm to be tested. This contradicts our requirement that ground-truth must be defined in the geometric image, i.e. at infinite resolution. Although this is impossible, it can be approximated: the ground-truth should be defined at a considerably higher resolution than the digital image to be processed. This can be achieved in two ways: either one takes two images of the same scene at different resolution (this requires a powerful registration algorithm to align the ground-truth with the low-res image), or one creates the low-res image from the high-res one by means of a subsampling procedure which realistically models a digital camera (similar to the simulation approach).

Images of test objects: Definition of the geometric image can be simplified by using 3-dimensional test objects with known properties (in some contexts, e.g. medicine, the objects are called "phantoms"). These objects are then imaged with real cameras. Common object types include resolution test charts for 2-D imaging, or custom-built models of human organs for 3-D imaging (e.g. computer tomography). The test objects must possess certain properties which can be easily verified in the segmentation result, in spite of possible distortions during projection and digitization, e.g. the number and neighborhood of connected regions, or collinearity, parallelity, intersection, length ratios and angles between edges. It is also possible to measure these properties in the 3-dimensional world by an independent measurement modality, e.g. traditional geodesy of photogrammetry. However, it is difficult to create large collections of test objects and scenes, and they often lack the complexity of real scenes. Their main purpose is the verification of simulations – when the results match, simulation can be used to perform much larger test series with equally reliable results.

2.2.2.1 Generated Test Images

Let us first look at generated test images which simulate the process of image acquisition in the computer. Artificially created images have been used for edge detector

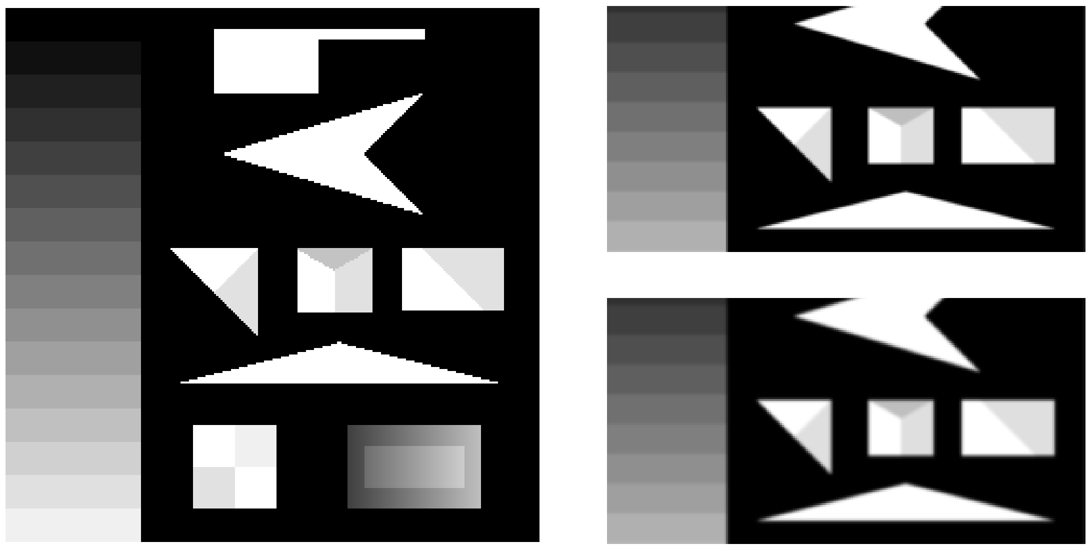


Figure 2.10: Left: Popular test image for edge and corner detectors, from [Smith & Brady 97]. However, it was directly sampled from the geometric image and lacks a simulated PSF. Note the staircasing artifacts at slanted edges. Right: Reproduction of the same test image with Gaussian PSF at $\sigma = 0.5$ and $\sigma = 0.85$ respectively.

evaluation since its beginnings, and are also frequently applied in testing corner detector performance. Figure 2.10 left shows a popular example that first appeared in [Smith & Brady 97]. It presents a reasonable collection of edge, corner, and junction features for a quick overview of algorithm performance, although the number of different angles is quite limited. Since the distance between neighboring features is not very big, it can also be used to study if and when the responses of nearby features start to interfere with each other. However, in anticipation of our detailed discussion in chapter 3 we can already state that results on this test image will not realistically predict performance on real images, because the simulation lacks a realistic camera model: the geometric image must be smoothed by a simulated camera point spread function (PSF) before sampling. Since many edges are aligned with the coordinate axes, this is perhaps not readily visible, but note the strong staircasing artifacts at slanted edges.

A straightforward method avoiding these artifacts is to render the test image at a much higher resolution (we use 8-fold oversampling), then smooth with a suitably scaled PSF filter, and finally downsample to the desired resolution. We are showing in section 3.2.3 that Gaussian filters are reasonable approximations of the PSF of real digital cameras. Figure 2.10 right depicts reproductions of the same test image with Gaussian PSFs at $\sigma = 0.5$ and $\sigma = 0.85$. At the latter value, the resulting image can be considered as effectively band-limited (cf. section 3.2) and artifact-free. The former value represents a typical PSF of real cameras, where some aliasing is tolerated (note the residual staircasing at slanted edges) for the sake of improved subjective image sharpness. It goes without saying that filtering of figure 2.10 left *after* sampling will not make the simulation any more realistic.

The advantage of the oversampling-smoothing-downsampling technique is that it can be applied to arbitrary input images. However, applying a digital filter to an oversampled image is still an approximation of what's really going on in the camera. It would be preferable to compute test images by exact numerical integration of the analytic convolution integrals $PSF \star f$ (where f is the test image at infinite resolution and without blurring). This is actually possible in case of test images containing a single corner or junction, and corresponding models have been proposed by [Rohr 92, Deriche & Giraudon 93]. Assuming constant gray levels within the adjacent regions, a junction with arbitrary degree and arbitrary angles can be constructed from a suitable superposition of simple corners. A corner whose apex is at the coordinate origin, one arm is parallel to the x-axis, and the other encloses an angle α with the x-axis can be written as:

$$c_\alpha(x,y) = \begin{cases} \Theta\left(\tan(\alpha)x - y\right)\Theta(y) & \text{if } \alpha \neq \pm\frac{\pi}{2} \\ \Theta\left(x\right)\Theta(y) & \text{otherwise} \end{cases}$$

where Θ denotes the unit step function. The convolution of such a corner with a Gaussian PSF at scale σ_{PSF} can be expressed as

$$f_{\alpha,\sigma_{\text{PSF}}}(x,y) = \tag{2.2}$$
$$\begin{cases} \Phi_{\sigma_{\text{PSF}}}(x)\,\Phi_{\sigma_{\text{PSF}}}(y) - \int_{\xi=-\infty}^{x} g_{\sigma_{\text{PSF}}}(x)\,\Phi_{\sigma_{\text{PSF}}}\left(y - \tan(\alpha)\,(x-\xi)\right)\,d\xi & \text{if } \alpha \neq \pm\frac{\pi}{2} \\ \\ \Phi_{\sigma_{\text{PSF}}}(x)\,\Phi_{\sigma_{\text{PSF}}}(y) & \text{otherwise} \end{cases}$$

where $g_{\sigma_{\text{PSF}}}(.)$ and $\Phi_{\sigma_{\text{PSF}}}(.)$ denote the Gaussian and its integral at scale σ_{PSF}:

$$\begin{aligned} g_\sigma(x) &= \frac{1}{\sqrt{2\pi}\sigma}\,e^{-\frac{x^2}{2\sigma^2}} \\ \Phi_\sigma(x) &= \int_{-\infty}^{x} g_\sigma(x')\,dx' = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right)\right) \end{aligned}$$

When the corner is formed by edges at two arbitrary angles α_1, α_2 ($\alpha_1 < \alpha_2$) and the enclosed region has intensity v, we get

$$v\,\left(f_{\alpha_2,\sigma_{\text{PSF}}} - f_{\alpha_1,\sigma_{\text{PSF}}}\right)$$

An arbitrary junction (with arbitrary many segments, arbitrary edge angles and intensities) can be composed by adding the constituting corner segments:

$$f = \sum_k v_k\left(f_{\alpha_{k+1},\sigma_{\text{PSF}}} - f_{\alpha_k,\sigma_{\text{PSF}}}\right) \tag{2.3}$$

In order to create example junctions quickly, we pre-computed the integrals (2.2) with α running from $0°$ to $360°$ in increments of $15°$, and with an image size of 401×401 at $\sigma_{\text{PSF}} = 10$. Arbitrary junctions are computed as weighted sums of the appropriate corners, followed by subsampling to the desired resolution. This subsampling does not

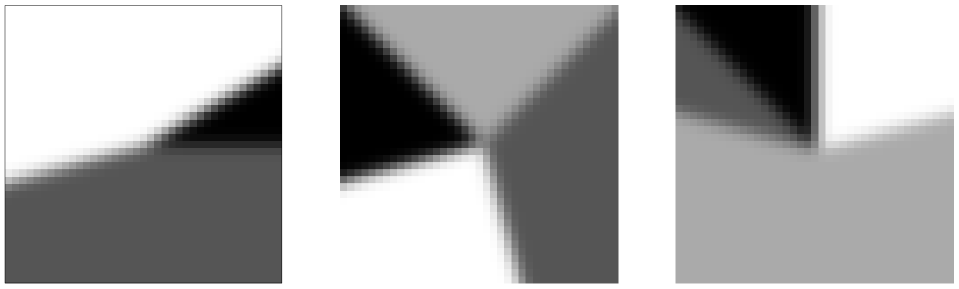


Figure 2.11: Example junctions generated by means of (2.3) for $\sigma_{\text{PSF}} = 1$.

require additional smoothing because this was already included in the numeric solution of (2.2). In most experiments we use $\sigma_{\text{PSF}} = 1$, so that we have to subsample 10-fold. In addition, sub-pixel junction locations with 0.1 pixel shifts can be simulated by translating the upper-left image corner within a 10×10 window before downsampling. Since all pixel values have been computed by direct numerical evaluation of the analytic model, approximation errors are avoided up to the numeric precision of the integration routine. Figure 2.11 shows example test images representative of the ones we are using in this work.

Since the ground-truth of simulated images is usually relatively simple (a few geometric objects represented by polygons and elliptic arcs), matching of the resulting segmentations against the original plane partition is relatively straightforward. We assume that the segmentation consists of regions, arcs, and vertices. Then we compute the asymmetric Hausdorff distance of every arc $\hat{a}_i$ in the segmentation to the ground-truth boundary B

$$d\left(\hat{a}_i, B\right) = \max_{q \in \hat{a}_i} d(q, B)$$

where $d(q,B) = \min_{b \in B} d(q,b)$ is the distance from point q to set B, and $d(q,b)$ is the Euclidean distance between points. The points of $\hat{a}_i$ will usually be the *edgels* representing the arc. Then we consider all $\hat{a}_i$ as successful matches, when $d\left(\hat{a}_i, B\right)$ is below a threshold. This simple heuristic works well for artificial test images because false positive arcs almost always contain at least one point that is sufficiently far away from the true boundary to be rejected by the threshold.

2.2.2.2 Test Objects and Scenes

Baker and Nayar created simple test objects and used them to create images very similar to the generated test images described in the previous section [Baker & Nayar 99]. They used accurately manufactured cuboids and cylinders as phantom objects. Sequences of several hundred images were taken from these objects by a robot that automatically varied object pose and illumination. Due to the properties of perspective projection, straight lines in 3D remain straight lines in 2D, and circles (the cylinders' bottoms) become ellipses. The image series is publicly available[4], and figure 2.12 depicts three

[4] `http://www1.cs.columbia.edu/CAVE/`

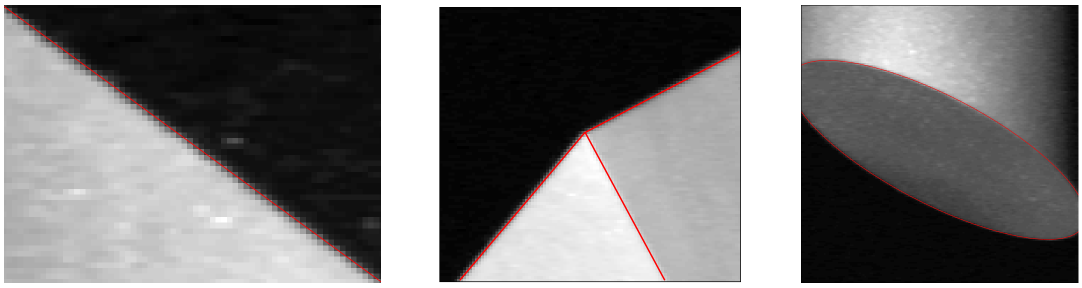


Figure 2.12: Images benchmark2-sn7 (left), benchmark5-t8 (center), and benchmark7-a2 (right) from [Baker & Nayar 99], with consensus ground truth according to algorithm 2.1 shown in red. Edge detectors used for ground truth estimation: subpixel watershed algorithm on the Gaussian gradient magnitude and on the structure tensor trace, Canny algorithm with spline-based subpixel correction, and Haralick algorithm with subpixel-accurate zero-crossing detection. All operators were applied with $\sigma_{\text{filter}} = 2$.

example images. In addition to using a real camera, these images have natural shading gradients rather than constant gray-value within each region. However, the gray-level quantization is somewhat coarse (only about 30 to 60 gray levels), and noise seems to be correlated between horizontally neighboring pixels.

Since the ground-truth for these images is not directly available, [Baker & Nayar 99] propose to compare algorithms by how well they reproduce the known invariant of each image. They assume that every algorithm returns edgels which are points supposedly lying on the desired edge and having the correct tangent direction θ. Then it is measured how well these edgels align on a straight line or ellipse, or whether they define a common point of intersection. Predicates measuring these properties are called *global measures of coherence*. As an example, we explain GMC$_1$, a predicate that measures edgel collinearity. Let $e_i = (x_i, y_i, \theta_i, c_i)$ be the coordinates, angle, and confidence of edgel i, and assume that all edgels returned by a given algorithm on a given image are sorted by decreasing confidence. Each edgel defines a projective line $\lambda_i = (l_{1,i} = -\sin\theta_i, l_{2,i} = \cos\theta_i, l_{3,i} = x_i \sin\theta_i - y_i \cos\theta_i)$. Then GMC$_1$ is defined as

$$\mathrm{GMC}_1(n) = \frac{1}{E_n(x)E_n(y)} \Big((E_n(x))^2\, Var_n(l_1) + (E_n(y))^2\, Var_n(l_2) + Var_n(l_3) \Big)$$

where $E_n(.)$ and $Var_n(.)$ denote the expectation and variance of a quantity over the first n edgels in the sorted list. The resulting functions GMC$_1(n)$ are averaged over many images. To compare two detectors j and k, the relative measure of coherence is computed:

$$\mathrm{RGMC}_1^{j,k}(n) = \frac{\mathrm{GMC}_1^j(n) - \mathrm{GMC}_1^k(n)}{\mathrm{GMC}_1^j(n) + \mathrm{GMC}_1^k(n)}$$

This measure is always in the range $[-1, 1]$, and positive values indicate that algorithm j is superior to algorithm k. Unfortunately, the GMC approach has a number of disadvantages:

- The GMC approach implicitly assumes that the edgel density along an edge is constant for all algorithms. In [Baker & Nayar 99], this is ensured by comparing only algorithms which detect at most one edgel per pixel, but this doesn't work for the algorithms we are analyzing in this work. For example, the sub-pixel watershed algorithm (section 5.1.2) creates edgels whose spacing along the edge is only about 0.1 pixel. Then, "choosing the n strongest edgels" means very different things for different algorithms, and computation of RGMC is meaningless.

- The proposed formulas for computing edgel properties such as collinearity and common point of intersection are not optimal. Points far from the coordinate origin may have stronger influence, measurement errors in the edgel parameters are not considered, and results may be unstable (especially the ones for parallel edges and edge intersection). Improved estimation algorithms for these properties can be found in [Utcke 06], and the GMC approach can in principle be modified accordingly.

- The distinction between geometric and topological errors is only implicit: the strongest edgels are usually close to the true edge, so that GMC_1 is dominated by geometric errors for small n. When all "good" edgels are used up, false positives will increasingly influence the GMC values. We would rather like clear distinctions between the error classes, and figures of merit should respect the distinction.

We do not use the GMC approach for these reasons. Instead, we estimate the ground-truth in each image by a technique inspired by [Yitzhaky & Peli 03] called *consensus ground truth.* Since we know the number and type of features (edges or ellipses) in each test image, this approach works very well here:

Algorithm 2.1: Consensus ground truth

Input: Test image containing a number of objects bounded by a known number of straight lines or ellipses.

1. Compute edgels with a number of different algorithms and take the union of the n strongest edgels from every algorithm.
2. Perform a robust fit (using, for example, the RANSAC method [Fischler & Bolles 81]) of a straight line or an ellipse.
3. Perform a least-squares fit using only the inliers determined in step 2.
4. If there are several features in the image, remove the inliers from the edgel set, and goto step 2.

The edge detectors used in consensus ground-truthing should have high accuracy and uncorrelated errors. According to our findings later in this work, the following four detectors are the most suitable: the sub-pixel watershed algorithm (section 5.1.2) on the basis of both the Gaussian gradient magnitude and the structure tensor trace, the sub-pixel version of Haralick's second derivative detector (section 5.1.1), and Canny's algorithm with sub-pixel correction (section 5.2). The ground-truth we computed by means of this method for the Baker/Nayar images is shown in red in figure 2.12. Subsequent edge detector evaluation is done in the same way as with simulated images.

2.2.2.3 Manual Ground Truth

The most common method for ground-truth definition in natural images is manual labeling by a human or a group of humans. This has the advantage that image interpretation ambiguities are resolved by the superior human visual system. Since manual labeling is expensive (especially when one attempts to collect many images in order to improve statistical reliability) these databases are rare. One of the largest and most popular is "Berkeley Segmentation Dataset" developed by Malik's group and described in [Martin et al. 01]. Its public version comprises 300 images with segmentations by about 5 to 7 subjects each[5]. This data set is of great help in the evaluation of segmentation algorithms, but it is not optimal in the context of our definition of *low-level* segmentation (cf. definition 2.4) on two reasons:

- Subjects were allowed to judge edge relevance according to subjective criteria. Thus, individual segmentations differ significantly from each other. Moreover, subjects largely employed high-level rules in their decisions, which means that mainly edges of salient objects have been marked, whereas equally strong edges in the background or around less important objects are missing, see figure 2.13. Moreover, there is significant disagreement between subjects exactly which edges are salient in this sense. According to our understanding, it is not the task of *low-level* image analysis to distinguish between important and unimportant boundaries – this distinction is performed by later analysis stages. High boundary strength (as measured by low-level algorithms) and high boundary importance (as labeled by human subjects) are considerably different concepts. This becomes especially apparent when Malik's segmentations are used to train classifiers for low-level boundary strength: there is high overlap between the positive and negative categories in the training data, resulting in an unnecessarily high error rate of the classification [Boetius 06].
- The manual segmentations are not very accurate in terms of localization. Some edges are 2 pixels and more to the side of the true edge. When we match accurate edges from low-level segmentation algorithms against inaccurate ground truth edges, the evaluation will significantly over-estimate the errors of the edge detectors, see figure 2.14. Very accurate edge detectors are especially suffering from this effect. In a learning framework, this again leads to significant overlap between the positive and negative categories [Boetius 06].

Another openly accessible image set with manual ground truth is provided at the University of South Florida[6] and described in [Heath et al. 97]. Its outstanding characteristic

[5]see `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/`

[6]see `http://figment.csee.usf.edu/edge/roc/` and `http://marathon.csee.usf.edu/edge/edge_detection.html`

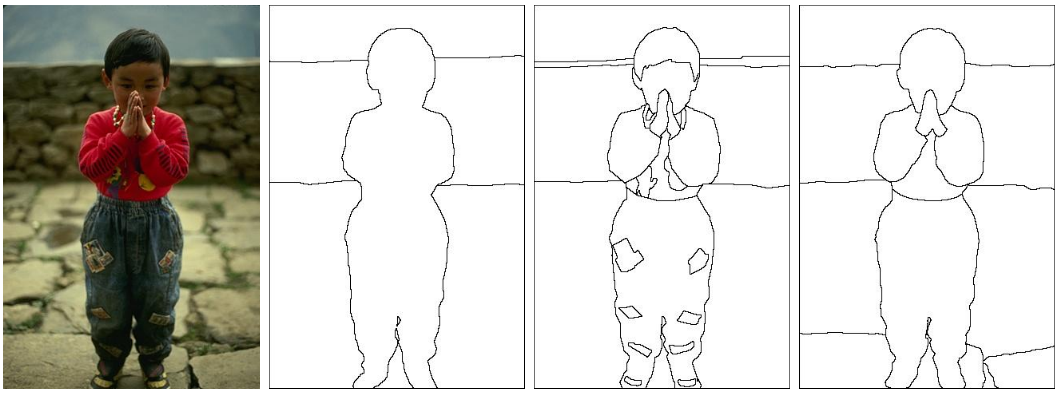


Figure 2.13: Image 187029 from the Berkeley Segmentation Dataset along with manual segmentations by three subjects. Which one should we test segmentation algorithms against? What about edges in the background region (stones in the wall and pavement) ?

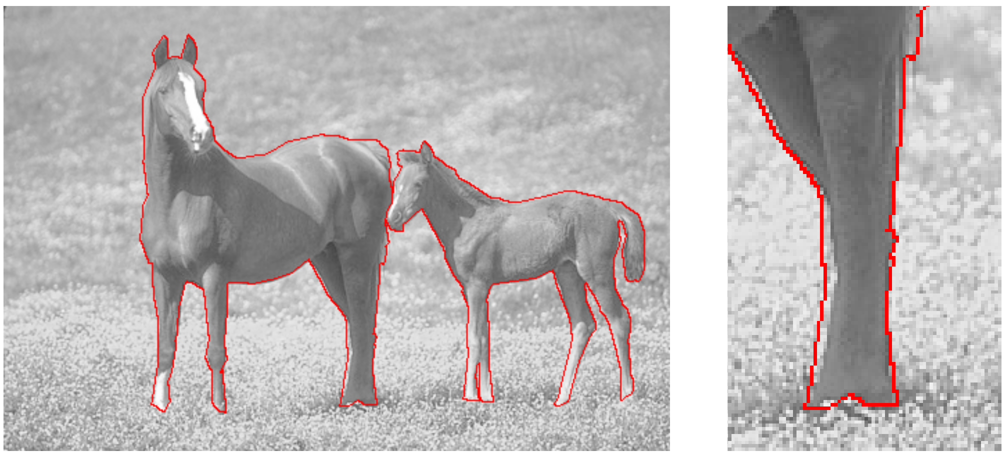


Figure 2.14: Image 113016 from the Berkeley Segmentation Dataset. The distance between the true edge and the manual segmentation is 3 pixels near the horse's leg (right).

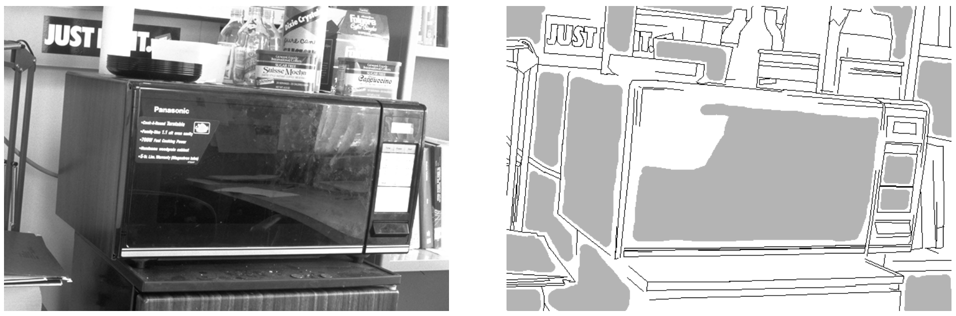


Figure 2.15: Image 50 (left) from the University of South Florida edge detector evaluation data set and the corresponding ground truth image (right) with "no edge" (gray), "edge" (black), and "don't care" (white) labels.

is that the ground truth labeling is three valued: "edge", "no edge", and "don't care", see figure 2.15. The "don't care" class accounts for structures which no low-level edge detector is supposed to handle reliably, such as textures. The detected edgel points are matched against the edge points in the "edge" class which are not farther away than a distance d (typically, $d = 2 \ldots 3$ pixels). Matched edge points are counted as true positives, unmatched points from the ground truth are false positives, and unmatched points from the detector (including those located in the "no edge" region) are false positives. The numbers are used to compute the receiver operator characteristic of an edge detector (cf. previous section).

While the idea of multi-valued ground truth is very interesting, the assignment of image points to the different classes is still problematic. Consider, for example, the image in figure 2.15. The reflections in the door of the oven are labeled as "no edge", although they contain clearly visible edges that many low-level algorithms would certainly detect. In contrast, the areas close to true edges are marked as "don't care", and edge detectors which produce false positives near true edges wouldn't be penalized. Similarly, most fine detail (e.g. pieces of small text) is assigned to the "don't care" class, so that experiments can never identify edge detectors which are particularly good at handling detail. Obviously, it is very difficult to achieve agreement upon criteria for correct ground-truth labeling, even if multiple labels are permitted.

In order to improve the quality of manual ground truth for low-level segmentation, we believe that two issues need to be considered:

1. Manual segmentation should be based on firm statements of the goal of the algorithms being tested. "Mark important object boundaries" is too vague in the context of low-level segmentation. Our definition 2.4 is an attempt towards a formal definition of correct low-level segmentation results: the boundaries of the ideal geometric image (*all* its boundaries) should be recovered.

2. Ground truth definition should make use of additional data which is not available to the algorithms being tested.

To understand the second statement, recall the original meaning of the term "ground truth": It designates measurements taken on the *ground* in order to verify hypotheses formulated on the basis of remote sensing data. The point of this kind of ground truth is that verification measurements are taken independently of the remote sensing data. In contrast, the manual ground truth for image analysis algorithms is always defined on the very image that is also the input of the algorithms to be tested. Since human subjects are not given additional independent data (other than their prior knowledge), the ground truth may not be as accurate as it should be. However, it is very difficult to obtain reliable segmentation ground truth from other measurement modalities (e.g. geodesic measurements) because it is impossible to include important image acquisition phenomena such as occlusion and shadows.

Therefore, we propose a compromise method: manual ground truth should be defined in a high-resolution image, whereas image analysis is performed at lower resolutions. The original high-resolution image can be transformed into the lower resolution one by a realistic simulated camera (including point spread function and noise). The ground truth data has to be transformed as well, and this naturally leads to multi-valued ground truth labeling: Certain boundaries should still be visible at the low resolution, and these form the "edge" class. Likewise, certain homogeneous regions should remain in the "no edge" class. The same may be true for weakly textured regions: at low resolution the texture may have been smoothed away, so that these regions will also belong to the "no edge". The same applies to weak edges – they may or may not be detectable at low resolution and should be assigned to a "may be edge" class. In other cases, objects close to each other (e.g. individual letters) may get merged into one (a text block), and the ground truth boundary should be adapted accordingly. Yet other boundaries may be too strong to be smoothed away, but no longer be reliably resolvable at the low resolution, so these might be assigned to a "don't care" class or perhaps to a "texture" class. At the same time, we also get subpixel accurate boundary locations, even if the ground truth was only marked with pixel accuracy. Unfortunately, the definition of an algorithm that transforms the ground truth from high to low resolution is an open problem. However, we are convinced that this transition is in principle possible, for example by means of (α, β)-reconstruction (algorithm 5.12).

3 Analysis of the Image Acquisition Process

Abstract

The ideal geometric image is noise-free and has infinite resolution, whereas the real digital image is distorted by noise and blurring, and contains only a finite amount of data. The relationship between ideal and real images is a key characteristic of any image acquisition device, and it is of fundamental importance to understand this relationship when one wants to reconstruct features of the ideal image from the real one. In this chapter, we analyze how real camera systems blur the ideal image (by their point spread function), whether this blurring together with the sensor resolution conforms to Shannon's sampling theorem, and how much the result is degraded by noise. On the basis of this analysis, we show that spline interpolation and noise normalization are important practical preprocessing steps that reduce the difference between the real and ideal images.

3.1 The Linear Model of the Image Acquisition Process

Image analysis can only produce reliable results if it is based on a realistic model of the image acquisition process. The model has to describe how information is altered by the components of an imaging device, and how the information in the image is related to the real world. The model then serves as a starting point for determining suitable image analysis methods. To yield relevant results, the model must describe the behavior of actual image acquisition devices with sufficient realism. On the other hand, it must remain tractable to be of any practical value. Currently the best compromise between realism and tractability is achieved by first introducing the ideal geometric image (see chapter 2), and then treating cameras as *linear shift-invariant systems* that transform the ideal image into the actual digital image by a convolution operation. This approach was pioneered by [Duffieux 46, Schade 48]. A good motivation of this approach from a physics perspective can be found in [Goodman 05], discussions in the specific context of image analysis and fidelity are, for example, given by [Granlund & Knutsson 95, Jähne 97, Park & Rahman 99, Förstner 99]. In other words, image acquisition is split into three steps:

- The 3D scene is first projected onto an idealized image plane by a perfect geometric projection, e.g. a perspective or fish-eye projection. The resulting *ideal geometric image* is continuous and has infinite resolution, but it is not directly observable.

- Under the influence of real optics and real sensors, the ideal image is blurred by the system transfer function of the imaging device. The resulting image is still continuous and will be referred to as the *analog camera image*.
- The analog camera image is sampled and quantized by the camera sensor. Discretisation also introduces noise into the image. The resulting image is the *digital image* we are actually going to analyze.

Figure 3.1 illustrates this image acquisition model. As far as low-level image analysis is concerned, the ideal geometric image $f_{\text{geometric}}(x, y)$ is considered as fixed, but unknown. Blurring of the geometric image by the optics of the camera can be described as a convolution with the optical *point spread function* (PSF):

$$f_{\text{optical}}(x, y) = PSF_{\text{optical}}(x, y) \star f_{\text{geometric}}(x, y)$$

This is an approximation, albeit a good one: The amount of blurring actually increases slightly towards the image periphery in real optical systems, and it varies with focus (i.e. object distance) and light wavelength (chromatic aberration). The error of the approximation is small when the lens is color corrected and the depth of sharpness covers the objects of interest entirely. We assume that this is the case throughout this work since the theoretical treatment of those effects is very complicated.

The optically blurred image is then digitized by the camera's receptor array. Photons are not only absorbed at the exact location of the sampling points, but in a certain area around them. The sensitivity profile of each receptor can be modeled by another PSF. The *analog camera image* $\breve{f}(x, y)$ is the result of convolving the optically blurred image with the sensor PSF:

$$\breve{f}(x, y) = PSF_{\text{sensor}}(x, y) \star f_{\text{optical}}(x, y)$$

The effects of both PSFs are conveniently combined into a single system PSF:

$$\begin{aligned} PSF_{\text{system}}(x, y) &= PSF_{\text{sensor}}(x, y) \star PSF_{\text{optical}}(x, y) \\ \breve{f}(x, y) &= PSF_{\text{system}}(x, y) \star f_{\text{geometric}}(x, y) \end{aligned}$$

In some cameras, an additional optical low-pass filter is placed between the lens and the sensor plane. Its PSF can be convolved into the system PSF in the same way. The Fourier transform of the PSF is called the *system transfer function*, and its magnitude is the *magnitude transfer function* (MTF).

Spatial discretization is mathematically modeled by multiplication with a superposition of delta functions located at the sampling points. In case of a square grid with sampling distance h, we get

$$\begin{aligned} \hat{f}\,(x, y) &= \sum_{i,j} \delta\,(x - h\,i, y - h\,j)\; \breve{f}(h\,i,\, l\,j) \\ \hat{f}_{kl} &= \breve{f}(h\,k,\, h\,l) \end{aligned}$$

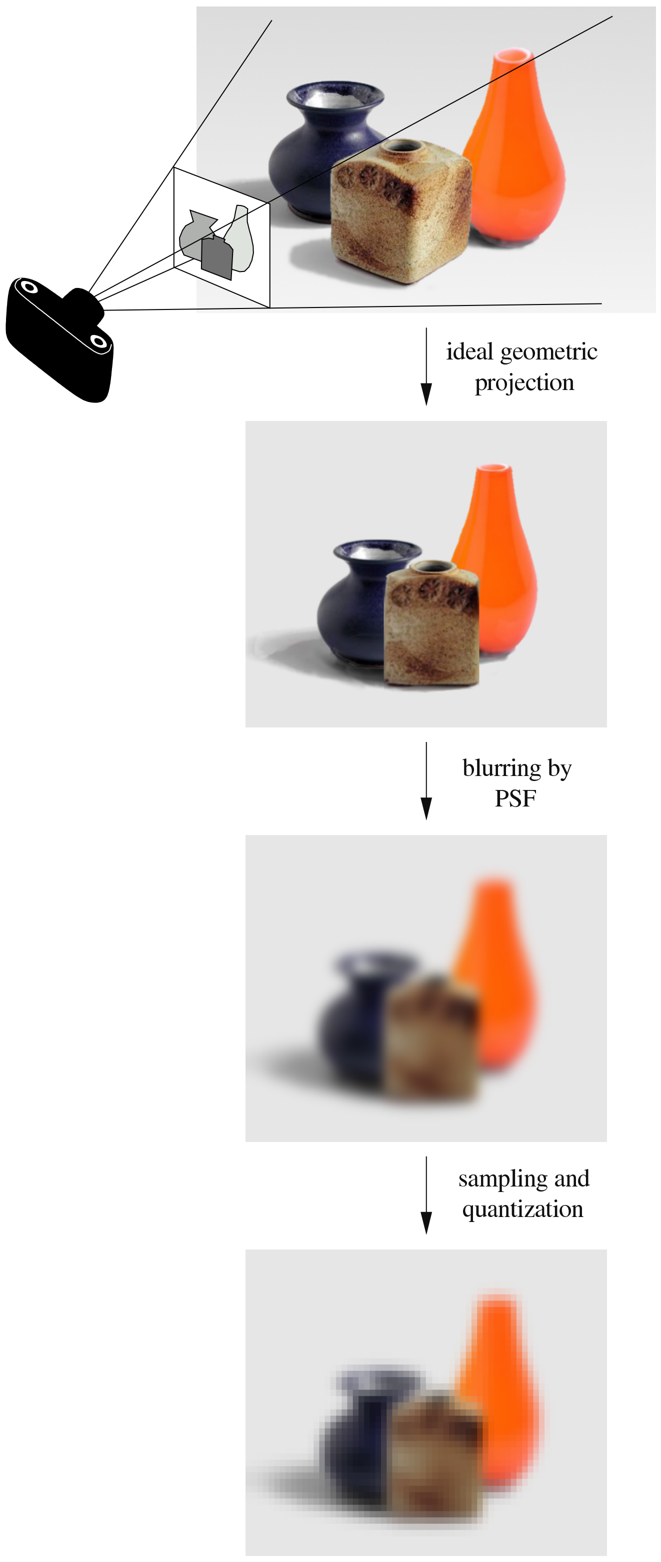


Figure 3.1: The image acquisition model employed in this work (compare figure 2.1).

Finally, noise $\hat{n}_{kl}$ is added to the sampled image, and the colors or intensities are quantized

$$f_{kl} = \lfloor \hat{f}_{kl} + \hat{n}_{kl} \rfloor$$

f_{kl} is the digital image we actually observe. The round-off error of quantization is best handled as just another source of noise. It is uniformly distributed between $-\frac{1}{2}$ and $\frac{1}{2}$ and has thus a variance of $\frac{1}{12}$ gray levels. The number of quantization levels should be chosen so that other noise sources have significantly larger variance (otherwise, the camera's capabilities would be wasted). Since the sensor noise standard deviation of modern cameras usually exceeds one gray level, this is generally fulfilled. However, assuming additive noise is clearly a simplification, because noise can actually depend on the data (e.g. on the local intensity). We will discuss this problem in section 3.4.

The point of a linear camera model is that the analog camera image can be exactly reconstructed from the sampled image provided the former is band-limited:

$$\mathcal{F}[\breve{f}](u, v) = 0 \text{ if } |u| < \nu_N \text{ and } |v| < \nu_N$$

where $\nu_n = \frac{1}{2h}$ is the Nyquist frequency of a grid with sampling distance h[1]. This is the result of Shannon's famous *sampling theorem* (see for example [Poularikas 96, section 1.6]). Due to noise and finite image size, the reconstruction can never be perfect in practice, but it will be very accurate when noise levels are low. The sampling theorem can be best understood in the Fourier domain, where spatial discretization corresponds to a convolution of the spectrum of the analog image $\breve{f}$ with a sum of delta functions:

$$\mathcal{F}[\hat{f}] = \mathcal{F}[\breve{f}] \star \left(\sum_{i,j} \delta\left(u - \frac{i}{h}, v - \frac{j}{h}\right) \right)$$

The spectrum of $\breve{f}$ is replicated an infinite number of times at intervals equal to the sampling frequency. The square $(-\nu_N, \nu_N)^2$ containing the replication at the origin is called the *sampling pass-band*, whereas the other replications are referred to as the *sampling side-bands*. If $\mathcal{F}[\breve{f}]$ is zero beyond the Nyquist frequency, as the sampling theorem requires, replications do not overlap, and the spectrum of $\breve{f}$ can be recovered from the spectrum of $\hat{f}$ by applying a suitable low-pass filter (e.g. the ideal interpolator, see section 3.3) which removes all spectrum replications in the sampling side-bands.

It is important that the analog camera image is indeed band-limited, because aliasing artifacts cannot be prevented otherwise. These artifacts are very undesirable, because

[1]In the context of optical systems, it is most convenient to define the Fourier transform as

$$F(u, v) = \mathcal{F}[f(x, y)] = \iint f(x, y)\, e^{-2\pi\,\mathrm{i}(x\,u + y\,v)} dx\,dy$$

and its inverse as

$$f(x, y) = \mathcal{F}^{-1}[F(u, v)] = \iint F(u, v)\, e^{2\pi\,\mathrm{i}(u\,x + v\,y)} du\,dv$$

Under these definitions, length and spatial frequency are direct inverses of each other, without the necessity to adjust units.

they cannot in general be distinguished from genuine features of the depicted scene. Erroneous feature measurements arising from aliasing can only be removed by additional assumptions about the image content, making image interpretation more difficult and less general. Since the geometric image cannot be band-limited due to its discontinuities, blurring with a band-limited PSF is necessary to turn an otherwise unpredictable sampling operation into a well-defined and reversible process. While this blurring deteriorates the geometric image, it is nevertheless a necessary prerequisite for successful image analysis. In the next two sections, we will look at how well this requirement is fulfilled in real optical systems and how the analog image can be reconstructed in practice.

3.2 The Linear Model in Digital Cameras and the Human Eye

3.2.1 The Diffraction Limited System

According to our linear image acquisition model, the analog camera image must be band-limited so that reconstruction from a sampled representation becomes possible. To justify this model, we investigate how well the underlying assumptions are fulfilled in real optical systems. It turns out that any real optical system is indeed band-limited. Consider a perfect lens. It is characterized by the property that all rays from a single object point are perfectly focused on a single image point. But even if a lens is extremely close to perfection, it must have a finite size. A lens of finite size is modeled by combining the perfect lens model with an aperture that is placed in the image-side principle plane of the lens. The light wave emitted at a given object point leaves the lens as a perfectly spherical wave which converges exactly onto the geometric image point. But as this spherical wave passes through the lens aperture, it is diffracted. The resulting diffraction pattern on the image plane at distance b behind the aperture is determined by the Fourier transform of the aperture shape [Goodman 05, Jähne 97]. In case of a circular aperture with diameter d we get (in polar coordinates)

$$p(r) = I_0 \, \frac{J_1\left(\frac{\pi\, r}{f_{\#}\,\lambda}\right)}{r}$$

where J_1 is the first-order modified Bessel function, $f_{\#} = b/d$ is the relative aperture (also known as the *f-stop number*), and λ is the wavelength and I_0 the amplitude of the wave function before diffraction. In case of incoherent illumination, the intensity of the perceived diffraction image is proportional to the square of the diffraction pattern amplitude. The resulting function is called *Airy function*:

$$I(r) \sim p(r)^2 \sim \mathrm{airy}(r) = \frac{J_1\left(\frac{\pi r}{f_{\#}\,\lambda}\right)^2}{\pi r^2}$$

The Airy function can be interpreted as the point spread function of a circular aperture, provided that it is normalized to unit integral over its domain $\mathbb{R}^2$ in order to preserve signal energy. It consists of a central, Gaussian-like peak that contains over 80% of the energy, surrounded by a series of dark and light rings. Its Fourier transform can be interpreted as the optical transfer function (OTF) of the pinhole camera with finite aperture size. It is equal to the convolution of the d-disk with itself and reads

$$P_{\text{diffraction}}(\nu, \nu_0) = \begin{cases} \frac{2}{\pi}\left(\cos^{-1}\left(\frac{\nu}{\nu_0}\right) - \frac{\nu}{\nu_0}\sqrt{1-\left(\frac{\nu}{\nu_0}\right)^2}\right) & \text{if } \nu \leq \nu_0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $\nu_0 = \frac{1}{f_\# \lambda}$ (in cycles per μm) or $\nu_0 = \frac{\pi d}{180° \lambda}$ (in cycles per degree, with aperture diameter d) is the band-limit of the system. This OTF s depicted in figures 3.3 and 3.6. Since this OTF cannot be exceeded by any real optical system it is known as the OTF of the *ideal diffraction-limited system*[2]. The band limit depends on the wavelength of the light. When the formula is applied to white light, the average wave length of the visible spectrum $\lambda = 0.55\,\mu$m is usually used. Two important characteristics of optical systems can be directly derived from the ideal OTF:

1. The *Rayleigh criterion* is an estimate of the optical resolution. It states that the images of two point sources can be distinguished when the peak of the Airy pattern of one source is located in the first dark ring of the other. This means that the distance between the two peaks must be $r_R = 1.22\,f\#\,\lambda$, and the height of the trough between the peaks is 74% of the peaks' height. This corresponds to a Michelson contrast
$$\text{contrast} = \frac{I_{\text{peak}} - I_{\text{trough}}}{I_{\text{peak}} + I_{\text{trough}}}$$
of 15%. This old rule of thumb is still widely used for comparing the resolution of two optical systems, although more recent measurements have shown that it is somewhat pessimistic: New estimates of the minimal required contrast for a pattern to be distinguishable are between 3% and 5% [Williams 85b]. In terms of the Rayleigh criterion, this means that two spots remain discernible at $0.86\,r_R$, or that their intensities may differ by 50% when the distance is r_R.

2. The *Strehl ratio* estimates the perceived sharpness of an image. It is the ratio between the areas under the actual OTF of the system and under the ideal OTF of the diffraction limited system. According to Fourier theory, this is equivalent to the ratio of the center heights of the corresponding point spread functions. The higher the Strehl ratio (i.e. the better it approaches unity), the closer is the real system to the diffraction limited system.

A discrete optical system would perfectly conform to Shannon's sampling theorem when the sampling frequency were at least twice the limit frequency of the diffraction limited

[2]The optimality of the diffraction limited system stems from the fact that the OTF of any system is the product of the diffraction limited OTF with the OTFs of all additional effects. The OTFs of passive optical elements can never have attenuation above unity, so the product cannot exceed the diffraction limited OTF. See for example [Goodman 05] for a more detailed discussion.

system $\nu_{\text{sampling}} > \frac{2}{f_{\#}\lambda}$. This corresponds to a minimal sample spacing of $h < 0.41\,r_R$, e.g. slightly less than what follows from the updated Rayleigh criterion. However, sampling at this rate would be overkill in most practical situations, because the ideal OTF is never achieved exactly. Instead, one chooses the sampling density according to the *effective band-width* of the real optical system.

To get a more formal understanding of the concept of an effective band-width, we follow the proposal of [Park & Rahman 99, Rahman & Jobson 03] to treat aliasing as just another kind of noise, albeit a scene-dependent one. Recall that the spectrum before sampling is the product of the spectrum of the ideal geometric image and the system transfer function

$$\breve{F}(\nu_1, \nu_2) = F_{\text{g}}(\nu_1, \nu_2)\, P_{\text{MTF}}(\nu_1, \nu_2)$$

The spectrum $\hat{F}$ after sampling is the superposition of infinitely many replications of the spectrum $\breve{F}$ plus sensor noise:

$$\hat{F}(\nu_1, \nu_2) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \breve{F}(\nu_1 - k, \nu_2 - l) + \hat{F}_{\text{noise}}(\nu_1, \nu_2)$$

where the sensor noise $\hat{F}_{\text{noise}}$ is band-limited within $\left[-\frac{1}{2}, \frac{1}{2}\right] \times \left[-\frac{1}{2}, \frac{1}{2}\right]$. The sum can be split up into the contribution from the sampling pass-band $\hat{F}_0$ (i.e. the replication at the origin), and the aliasing contributions from the sampling side-bands $\hat{F}_{\text{alias}}$ (all other replications):

$$\hat{F}(\nu_1, \nu_2) = \hat{F}_0(\nu_1, \nu_2) + \hat{F}_{\text{alias}}(\nu_1, \nu_2) + \hat{F}_{\text{noise}}(\nu_1, \nu_2)$$

with

$$\begin{aligned} \hat{F}_0(\nu_1, \nu_2) &= \breve{F}(\nu_1, \nu_2) \\ \hat{F}_{\text{alias}}(\nu_1, \nu_2) &= \sum_{k,l \neq (0,0)} \breve{F}(\nu_1 - k, \nu_2 - l) \end{aligned} \tag{3.2}$$

Note that the aliasing noise $\hat{F}_{\text{alias}}$ is additive. The spectrum of the reconstructed signal $\tilde{F}$ is obtained by multiplying $\hat{F}$ with the transfer function of the ideal interpolator, i.e. the box function over the interval $\left[-\frac{1}{2}, \frac{1}{2}\right] \times \left[-\frac{1}{2}, \frac{1}{2}\right]$. If the system MTF is band-limited within $\left[-\frac{1}{2}, \frac{1}{2}\right] \times \left[-\frac{1}{2}, \frac{1}{2}\right]$, and the noise is zero, the reconstructed spectrum $\tilde{F}_{\text{ideal}}$ will be equal to the spectrum $\breve{F}$ before sampling. Otherwise, the reconstruction contains errors that can be quantified. According to [Huck et al. 99, Rahman & Jobson 03], a suitable figure of merit is the mutual information between the actual spectrum and the ideal spectrum

$$H = \mathcal{E}\left[\tilde{F}\right] - \mathcal{E}\left[\tilde{F} | \tilde{F}_{\text{ideal}}\right]$$

where $\mathcal{E}\left[\tilde{F}\right]$ is the entropy of the reconstructed spectrum, and $\mathcal{E}\left[\tilde{F} | \tilde{F}_{\text{ideal}}\right]$ is the conditional entropy of the reconstructed spectrum given the ideal spectrum. The mutual information is maximized if the second term is zero, i.e. if the actual and ideal spectra are identical. Under realistic assumptions discussed in [Huck et al. 99], this is equal to the entropy of the reconstructed signal minus the entropy of aliasing and noise

$$H = \mathcal{E}\left[\tilde{F}\right] - \mathcal{E}\left[\tilde{F}_{\text{alias}} + \tilde{F}_{\text{noise}}\right]$$

which can be expressed as

$$H = \frac{1}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \log_2 \left(1 + \frac{\left|\hat{F}_0(\nu_1, \nu_2)\right|^2}{\left|\hat{F}_{\text{alias}}(\nu_1, \nu_2)\right|^2 + \left|\hat{F}_{\text{noise}}(\nu_1, \nu_2)\right|^2} \right) d\nu_1 \, d\nu_2$$

To simplify the expression further, it is assumed that the autocorrelation of the geometric image is proportional to $e^{-r/\mu}$, where r is the distance from the origin, and μ is the *mean spatial detail*, i.e. the average size of image regions relative to the sampling distance of the camera. This autocorrelation describes natural images well because it reflects their *self-similarity*: the general form of the autocorrelation is independent of the viewing distance, and only the parameter μ changes. Then the power spectrum of the ideal geometric image becomes

$$|F_{\text{g}}(\nu_1, \nu_2)|^2 = \frac{K^2}{\left(1 + (2\pi\mu)^2 \left(\nu_1^2 + \nu_2^2\right)\right)^{3/2}} \tag{3.3}$$

The noise shall be white with zero mean and standard deviation σ_{N}. The noise power spectrum equals σ_{N}^2 in the sampling passband, and zero elsewhere. The ratio K/σ_{N} is the *signal-to-noise ratio*. Furthermore, it is assumed that the spectrum replications due to sampling are uncorrelated, so that the contributions of mixed terms in the square of the aliasing spectrum vanish. Then we get

$$H = \frac{1}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \log_2 \left(1 + \frac{|F_{\text{g}}(\nu_1, \nu_2) P_{\text{MTF}}(\nu_1, \nu_2)|^2}{\sum_{k,l \neq (0,0)} |F_{\text{g}}(\nu_1 - k, \nu_2 - l) P_{\text{MTF}}(\nu_1 - k, \nu_2 - l)|^2 + \sigma_{\text{N}}^2} \right) d\nu_1 \, d\nu_2$$

We can analyze how the information content H of the reconstructed image changes when the spatial detail μ, the signal-to-noise ratio and the MTF are varied. We compare three MTFs – the diffraction limited MTF, a Gaussian and an exponential MTF – and express H as a function of the MTF's spatial scale (in case of the diffraction limited MTF, this is the ratio between the grid's Nyquist frequency and the MTF's cutoff). Computation of H with different parameters confirms, first of all, a number of obvious facts: (i) The information content of the digital image decreases with increasing noise. (ii) If there were no aliasing, the information content would monotonically decrease with increasing MTF scale, i.e. with increasing smoothing. (iii) When the scene doesn't contain much detail, i.e. μ is big, the influence of the MTF is small – it only attenuates frequencies that are not present in the data anyway.

Interesting results are obtained when the scene indeed contains significant detail, i.e. when μ becomes smaller than approximately 3. Then the information content does not

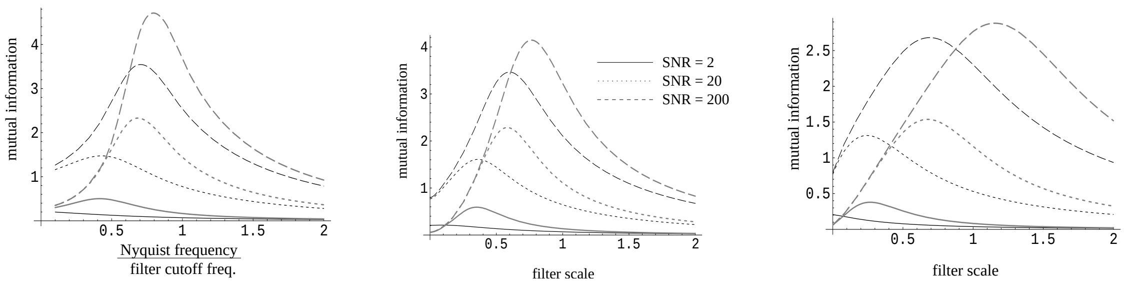


Figure 3.2: The mutual information between the sampled and original images as a function of the filter size σ (SNR = 2: solid, SNR = 20: dotted, SNR = 200: dashed; $\mu = 1$: black, $\mu = 0.1$: gray). Left: diffraction limited MTF, center: Gaussian MTF $P_{\mathrm{MTF}}(\nu_1, \nu_2) = e^{-2\pi^2\sigma^2(\nu_1^2+\nu_2^2)}$; right: exponential MTF $P_{\mathrm{MTF}}(\nu_1, \nu_2) = e^{-2\pi\,\sigma\,\sqrt{\nu_1^2+\nu_2^2}}$.

only decrease for large scale MTFs (excessive blur), but also for very small ones (strong aliasing), see figure 3.2. A clear optimum exists between these extremes: the information content is maximized by the MTF which optimally balances these two sources of information loss[3]. The precise scale of the optimum depends on the signal-to-noise ratio and the mean spatial detail. A very detailed geometric image requires more smoothing to suppress aliasing. But when the noise standard deviation σ_{N} increases, more aliasing can be tolerated because it will be masked by the statistical noise. The mutual information at the optimum is always higher for the Gaussian MTF. Very similar results are reported in [Park & Rahman 99], who use the *image fidelity*

$$F = 1 - E\left[\left|\breve{F} - \breve{F}_{\mathrm{ideal}}\right|^2\right] / E\left[\left|\breve{F}_{\mathrm{ideal}}\right|^2\right]$$

(where E is the expectation) as another plausible figure of merit for the comparison between ideal and real reconstructions.

In digital images with 256 gray-levels, signal-to-noise ratios between 20 and 200 are quite realistic: typically, the noise standard deviation is at most a few gray-levels, and the edge contrast a few tens of gray-levels, up to perhaps 200. Under these conditions, the optimal diffraction limited and Gaussian MTFs have scales between $\sigma \approx 0.4$ and $\sigma \approx 0.85$, whereas the scale of the optimal exponential lies between $\sigma \approx 0.3$ and $\sigma \approx 1.1$. The corresponding attenuations at the Nyquist frequency are between 40% and 3% for either MTF type. Therefore, these MTFs can be considered as *effectively band-limited* under suitable imaging conditions.

Since the MTF cannot easily be changed for a given lens and sensor, the MTF of a multi-purpose optical device should be designed to work well under all expected imaging conditions. To be on the safe side with respect to aliasing, the MTF scale should be on the upper end of the useful range, so that the attenuation at the Nyquist frequency becomes approximately 4%. If one allows for some aliasing in order to improve perceived image sharpness, $\sigma \approx 0.5$ might be used, with an attenuation of only 30% at the Nyquist

[3]We note in passing that this method can also be used to determine optimal filter scales for image pyramid construction.

frequency. We will see below that the human eye roughly realizes the first possibility (although with an exponential-like MTF), whereas digital cameras follow the second strategy.

3.2.2 The Human Eye

By looking at the human visual system, we can gain additional insight into where the effective band limit should be placed in practice. The eye is remarkable in that the sampling density in the center of the fovea (the area of the retina with highest sensor density) seems to match exactly the resolution of the optical system of the eye. This can be concluded from experiments that compare the threshold contrasts for detection vs. recognition of a grating pattern. In the detection mode, subjects have to signal whether or not a pattern is present in the stimulus, irrespective of if it is seen correctly or just as any pattern. In the recognition mode, the subjects must correctly identify the spatial frequency of the pattern. If subjects were able to detect the presence of a high frequency pattern without being able to recognizing it correctly, this would suggest that they in fact perceived a lower frequency Moiré pattern, and not the true stimulus. However, this has not been observed in experiments: In the fovea, the thresholds for both detection and recognition are equal, and the highest observable frequency is about 60 cycles per degree of visual angle [Hirsch & Curcio 89, Thibos et al. 87, Williams & Hofer 03]. The possibility that Moiré effects are somehow filtered out by neural processes was excluded by two control experiments: First, when the same experiment is repeated at peripheral locations in the retina, Moiré patterns can occur, and the recognition threshold is well below the detection threshold . Second, it is possible to create Moiré patterns in foveal vision, when high frequency gratings are created directly in the retinal plane (by means of laser interferometry), so that they are not subjected to the optical blurring of the cornea and lens [Williams 85a].

These results suggest that the sampling density of the human fovea is indeed just as high as the effective bandwidth of the optical transfer function requires. But exactly what does this mean? Which attenuation depth must not be exceeded in order to consider an optical system as effectively band-limited? To answer this question, we must compare measurements of the optical properties and sampling density of the human eye. Up to until very recently, the foveal cone density could not be measured in-vivo, but only in anatomical preparations. This is unfortunate because it means we cannot correlate the optical performance with the sensor density of one and the same person. We must rely on averages, although there is considerable variation among individuals. The most comprehensive measurements of human foveal cone density were conducted by [Curcio et al. 90]. The average center-to-center cone spacing for 18 individuals described in that paper was $h_{\text{cone}} = 2.55\,\mu\text{m}$, with a minimum of $1.9\,\mu\text{m}$ and a maximum of $3.4\,\mu\text{m}$. Since cones are, to good approximation, ordered in a hexagonal raster, the corresponding Nyquist frequency is

$$\nu_{\text{Nyquist, human}} = \frac{1}{\sqrt{3}\,h_{\text{cone}}} = 0.226 \text{ cycles per } \mu\text{m}$$

To compare this number with other optical systems, we must express it in terms of the size of actually observable objects, e.g. as the subtended visual angle on the object side (in degrees). The conversion factor can be computed as follows: Since the lens is immersed in fluid, not air, the focal lengths f_o on the object side and f_i on the image side of the eye differ, and the lens equation becomes

$$\frac{f_o}{b_o} + \frac{f_i}{b_i} = 1 \tag{3.4}$$

where b_o and b_i are the distances of the object and image from the principle planes of the cornea/lens system. The magnification is given by

$$\frac{B_i}{B_o} = \frac{f_o}{f_i}\frac{b_i}{b_o}$$

where B_o and B_i are the sizes of the object and its image. If we assume that the object distance is large with respect to the focal length (i.e. $f_o \ll b_o$), it follows from (3.4) that $b_i \approx f_i$ and we get

$$B_i = f_o \frac{B_o}{b_o} = f_o \tan \beta_o$$

where β_o is the visual angle subtended by the object. Note that the image side focal length drops out of the equation. If we further assume that the object size is much less than its distance, the tangent is almost equal to the angle (in radians), and we get

$$\beta_o \approx \frac{B_i}{f_o}$$

The average object-side focal length of the eye is 16.7 mm. Therefore, the Nyquist frequency of the foveal cone raster becomes

$$\nu_{\text{Nyquist, human}} = \frac{f_o}{\sqrt{3}\, h_{\text{cone}}} \frac{\pi}{180^\circ} = 66 \text{ cycles per degree}$$

This number is slightly higher than the detection and recognition acuity of 60 cycles per degree according to the measurements described above. It supports the hypothesis that the eye's optical resolution is matched well to the sensor distance: Frequencies above the Nyquist frequency are indeed suppressed thanks to optical blurring by the cornea/lens system and the sensitivity profile of the cones. To find out how much attenuation is required in order to suppress aliasing, we must look at measurements of the optical transfer function of the eye. These measurements cannot be performed by means of psychological experiments because the observer's response would not only depend on the eye's optical properties, but also on the neural processing in the retina and the brain[4]. We need the transfer function without neural contributions, because aliasing is a purely physical effect that occurs in the process of sampling by discrete receptors, i.e. before any neural processing can take place.

[4]Not all publications clearly distinguish the "contrast sensitivity function" (describing the combined effect of the eye's optics *and* the brain's neural processing) from the optical transfer function (describing the optics alone), although these functions differ considerably.

In order to isolate the influence of the optical system, an objective, purely optical measurement strategy is required. Three methods are in common use: the double-pass method [Campbell & Gubisch 66, Westheimer 86, Artal & Navarro 94], laser interferometry [Campbell & Green 65, Williams et al. 94], and aberrometry [Liang & Williams 97, Marcos 03]. The latter method is the most involved, but allows to estimate the complete wave aberration of the eye, i.e. the optical path length as a function of the ray's entry point into the eye. The optical transfer function (including phase) can be calculated from the wave aberration data. The other two methods only allow to measure the modulation transfer function (the magnitude of the OTF), and give 1-dimensional slices or circular averages of it. In the double-pass method, a point source is projected onto the retina, and the light that is reflected out of the eye is imaged. Since the light passes through the eye twice, the intensity of this image is proportional to the autocorrelation of the PSF, and its Fourier transform is the square of the MTF. In the interferometric approach, a pair of coherent laser beams are directed into the eye so that they form an interference pattern (in the form of a sine wave grating) *directly in the foveal plane.* The spatial frequency of the grating depends only on the wave length of the laser light and on the distance of the two beams as they enter the eye – frequencies up to 200 cycles per degree (well beyond the eye's Nyquist frequency) are possible. Under normal circumstances, frequencies above 60 cycles per degree would be suppressed by the optical system, but in case of interferometric stimulation, diffraction and aberrations only cause a phase shift in the resulting interference pattern. The interference pattern can be used in two ways: First, one can measure how the (known) contrast of the pattern is attenuated when the pattern is viewed through the eye's optical system from the outside. This gives an objective measurement of the optical transfer function. Second, one can measure detection thresholds of the brain for the interference patterns resulting from neural processing plus cone sensitivity alone, independently of the eye's optics.

At present, the best available measurements seem to be those of Williams and his collaborators [Williams et al. 94, Liang & Williams 97, Williams & Hofer 03]. In a series of experiments with several subjects, they determined the MTF and OTF as functions of pupil size, using all three methods mentioned, partly on the same subjects. The most interesting findings with respect to our discussion were

- For pupil sizes at and below 2 mm, the eye's optics are well described by the model of a diffraction limited system. Interestingly, the band-limit of the diffraction limited system for a 2 mm pupil (i.e. at a relative aperture $f_{\#} = 8.35$) is 63 cycles per degree. That is, the eye achieves theoretically optimal performance as long as the optical band-limit is below the Nyquist frequency of the cone raster.

- As the pupil size increases, the OTF is exceedingly limited by aberrations. The best balance between diffraction and aberrations, and consequentially the best optical performance, is achieved at a pupil size of about 3 mm. At larger pupil sizes, the PSF becomes rather anisotropic. The shape of the PSF varies strongly between individuals, but is almost mirror symmetric in the two eyes of a single person.

- Interestingly, above 3 mm pupil size, the attenuation depth at 60 cycles per degree remains almost constant at 3 to 5%. In other words, aberrations cause just enough blurring to avoid aliasing, but not more (as this would waste sensor resolution).

For our present discussion, a pupil diameter around 3 mm ($f_{\#} = 5.6$) is of highest interest because it corresponds to normal viewing conditions, where the eye's performance is optimal. [Williams et al. 94] found that the following MTF function gives a very good fit to the data at 3 mm pupil diameter:

$$P_{\text{optical,human}}(\nu) = P_{\text{diffraction}}(\nu, \nu_0)\left(0.3481 + 0.6519\, e^{-0.1212^\circ\, \nu}\right) \tag{3.5}$$

where $\nu_0 = 82.7$ cycles per degree because they used laser light with a wave length of 633 nm. The attenuation depth of this function at the Nyquist frequency is 3.7%. Their measurements of the eye MTF using the double-pass method were slightly, but consistently lower: Above ≈ 20 cycles per degree, the double-pass MTF stayed at about 60% of the interferometric MTF [Williams et al. 94]. The reasons seem to be unclear.

An alternative functional approximation of the PSF on the basis of double-pass measurements at a 3 mm pupil was given by [Westheimer 86]

$$p_{\text{optical, human}}(r) = 0.925\, e^{-2.59|r|^{1.36}} + 0.048\, e^{-2.43|r|^{1.74}}$$

where r is the radial distance in minutes of arc. Although this function is often cited (e.g. [Beckmann & Legge 02, Cormack 05, Pattanaik et al. 98, Vimal et al. 89, Winkler 99]), it appears to be wrong, as was pointed out by [Wachtler et al. 96]. They found that the function was inconsistent with the tabulated raw data reported alongside the formula and concluded that a numerical error must have been made during the transformation of the measured line spread data into the desired point spread function. They calculated a corrected PSF from the original data as

$$p_{\text{optical, human}}(r) = 0.108\, e^{-1.15\, r^2} + 0.050\, e^{-0.67|r|}$$

(in contrast to [Wachtler et al. 96], we normalized the PSF to have unit integral). The corresponding MTF is

$$P_{\text{optical, human}}(\nu) = 0.295\, e^{-0.000657\, \nu^2} + \frac{0.705}{(1 + 0.00680\, \nu^2)^{3/2}} \tag{3.6}$$

with ν in cycles per degree. The corrected function is not only a good fit for the measured double-pass data of [Westheimer 86], but also corresponds to the results of other authors. For example, it reaches 60% of the interferometric MTF at high frequencies, in accordance to the observations reported by [Williams et al. 94]. The MTFs (3.5) and (3.6) have almost identical Strehl ratios (38.7% and 38.3%). Figure 3.3 shows these functions together with the diffraction limited MTF for a 3 mm pupil.

In another series of experiments, [Williams 85a] measured the neural contrast sensitivity for interferometric gratings created directly in the foveal plane. Frequencies between 10 and 200 cycles per degree were presented to the subjects. At frequencies beyond the eye's Nyquist frequency, subjects perceived the expected Moiré patterns (instead of the

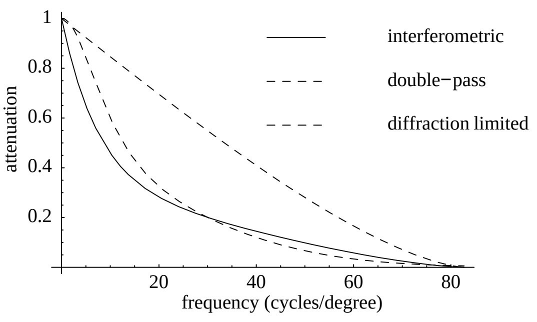


Figure 3.3: Functional approximation of the eye's optical MTF over frequency. Solid: determined by interferometric method (3.5); dotted: determined by double-pass method (3.6), dashed: diffraction limited system for comparison. The Nyquist frequency of the cone raster is 66 cycles per degree.

actual gratings). These patterns remained detectable up to an actual frequency of 200 cycles per degree, albeit with decreasing contrast. This shows that the absence of Moiré patterns in normal viewing is indeed a consequence of optical filtering and not due to some hypothetical neural mechanism.

To get the total attenuation before sampling, the optical MTF must be multiplied with the MTF corresponding to the sensitivity profile of an individual cone. Any subsequent filtering will then occur on sampled data where the fundamental and aliasing energies can no longer be distinguished. The first attempt at an objective measurement of the cone sensitivity profile seems to be [MacLoad et al. 92]. The experimental technique is a refinement of the just mentioned investigation of Moiré patterns caused by interferometric gratings. In this experiment, two high frequency gratings were presented simultaneously. Since human neural processing includes non-linear processing steps, theory predicts that the superposition of these gratings should look like a Moiré pattern that arose from a rotated low-frequency grating. This expected pattern was indeed seen by the subjects. According to theory, the threshold contrast for its detection is proportional to the square of the MTF of all linear filtering steps occurring *before* the first non-linear computation. The data could be fitted very well to a Gaussian aperture function with full width at half height of 1.1 μm, which is less than half of the cone spacing. In a recent publication [Hofer et al. 05a], an improved estimate derived from newer measurements was given as 61.5% of the cone spacing, i.e. 1.57 μm. It corresponds to the following Gaussian MTF

$$P_{\text{cone}}(\nu) = e^{-\frac{(0.014\,\nu)^2}{2}}$$

(ν in cycles per degree). Its attenuation depth at the Nyquist frequency is 64%. Since this aperture is very narrow, the authors of [MacLoad et al. 92] conclude that it probably arises solely from the cone sensitivity profile and does not include contributions of neural linear filtering prior to the first non-linearity. That is, the first non-linear computation occurs quite early in the retinal processing chain. The combined attenuation depth of the optical and cone MTFs is 2.4% at the Nyquist frequency, and 4% at 60 cycles per degree, in very good agreement with the observation that 4% is the lowest contrast, and 60 cycles per degree the highest frequency humans can perceive under optimal viewing conditions [Williams 85b].

So far, we have treated the fovea as if it only contained one receptor type. But actually, there are three: S-, M-, and L-cones corresponding to maximum spectral sensitivities at short, medium, and long wave lengths. However, these receptors are not equally distributed in the fovea. In fact, S-cones are quite rare (3.9% to 6.6% of the total population) because short wavelengths are severely out of focus (as much as 2 diopters) due to the eye's chromatic aberrations. Hence, denser sampling of the blue end of the spectrum wouldn't make sense. Since the spectral sensitivity peaks of M- and L-cones differ by only 30 nm, chromatic aberrations can be neglected for these cone types. Thanks to adaptive mirror optics which compensate for the eye's aberrations, it has recently become possible to take images of the fovea where individual cones are resolved and can be classified [Roorda & Williams 99, Hofer et al. 05b]. These investigations found that the L/M cone ratios exhibit enormous variability between individuals – from 0.37 (70% M-cones) to 16.5 (almost no M-cones)! Moreover, the two cone types are not distributed at random, but tend to form clusters of like type. Except for extreme cases, the L/M ratio has apparently no influence on visual performance. In light of these and other recent findings, the question of how foveal color perception works has again become an active area of research. With regard to the luminance channel, there is consensus that the brain can interpolate an accurate luminance image at full resolution from the responses of L- and M-cones because their spectral sensitivities are so similar. [Osorio et al. 98] estimate the worst-case luminance error to be comparable to adding a sinusoidal grating with 1% contrast to the image.

In summary, we are led to the conclusion that the human eye can be considered as an effectively band limited system, and the band limit corresponds to the Nyquist frequency of the cone raster, or is even lower (at pupil diameters below 2 mm). The fraction of the MTF that falls outside the sampling passband is $\approx 3.7\%$.

3.2.3 Digital Cameras

Now we want to compare these measurements with the situation in digital cameras. On the one hand, cameras are simpler to analyze because they lend themselves easily to experimentation. On the other hand, matters are complicated by the huge variability of the available designs. Moreover, manufacturers don't disclose many vital details. So it is necessary to measure the MTF of the particular camera to be used. But before doing that, we will discuss a few general principles.

In contrast to the eye, which evolved according to the image analysis requirements of the brain, cameras are not usually optimized with image analysis in mind. Typically, camera images are meant to be printed or displayed for human viewing, and the reproduction should look as similar as possible to the original scene, as judged by human observers. Under these conditions, the image data have to pass through three different PSFs (when the processing chain is approximated by a linear system): the camera PSF, the display PSF, and the PSF of the human eye. It is evident that the final image will only be similar to what the viewer would see by looking directly on the scene if the camera and display PSFs have only minor attenuation relative to the eye PSF. The key property to be optimized is thus the perceived sharpness of the image. If the image is attenuated to below 5% at the Nyquist frequency before it arrives at the eye, it will look blurry. The standard trade-off is to accept a certain amount of aliasing in order to keep

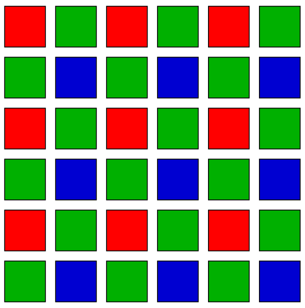


Figure 3.4: The spectral sensitivity pattern (Bayer filter) of the pixels in most single chip color cameras.

the camera MTF at 10 to 35% at the Nyquist frequency [Williams & Burns 01]. When the attenuation depth is above 35%, aliasing reaches unacceptable levels.

Like the human eye, the camera MTF is bounded by the MTF of the ideal diffractive system (3.1). In the context of camera properties, the band limit is usually expressed in line pairs per millimeter. For f-stop $f_{\#} = 8$ and green light at $\lambda = 550$ nm, we get an ideal band-limit of 227 lp/mm (line pairs per millimeter, same as cycles per millimeter). When a diffraction limited lens is used, the sensor Nyquist frequency must lie between 60 an 80 percent of this value to obtain the desired 10-30% attenuation. This can indeed be achieved with modern sensor arrays with 2.5 to 3 μm pixel pitch. However, this design will not lead to very high picture quality because the PSF lacks uniformity over the operating range of the lens: The contrast worsens dramatically as one moves to the periphery of the field of view (proportional to $\cos\alpha$ for radial lines, and to $\cos^3\alpha$ for tangential ones, where α is the angle of eccentricity [Fiete 04]), and it varies with aperture setting and zoom (if available).

A further complication is introduced by the fact that in most color cameras a single chip is responsible for reproducing all three color channels (with the exception of professional video cameras, which use three separate chips). In most cameras[5], every pixel is covered by a specific color filter (red, green, or blue), and these filters are arranged according to the Bayer filter pattern depicted in figure 3.4. It consists of three rectangular subgrids at half (red and blue) and $\sqrt{2}/2$ times the resolution (green) of the raw pixel grid. The green subgrid is rotated by 45°. When the Bayer pattern is used, the Nyquist frequency is no longer determined by the raw pixel pitch, but at best corresponds to the pixel pitch of the green subgrid. The color image at full resolution must be interpolated from the responses of the color subgrids. This is similar to the human eye, where a full resolution image is reconstructed from the S-, M- and L-cone responses. However, the spectral sensitivity of the M- and L-cones is very similar, so that the eye is able to interpolate as if the luminance image were available at full pixel resolution. This is in general not the case with camera interpolation: The spectral sensitivities of the channels differ significantly, and the resulting luminance image will only have the effective resolution of the green channel (which dominates recovery of luminance information with a ratio of 6:3:1 over the red and blue channels). This means in particular that frequencies beyond the Nyquist

[5]A notable exception to this interlaced design is the Foveon X3 chip [Lyon & Hubel 02] where red, green, and blue sensors are stacked on top of each other in every pixel, so that full color resolution is retained at the expense of reduced signal-to-noise ratio.

frequency of the green subgrid will give rise to aliasing artifacts that cannot be removed in the interpolation step. Thus, a Bayer grid with 3 μm raw pixel pitch has a Nyquist frequency (in the green subgrid) of 118 lp/mm.

Like in the human eye, the aperture of the individual sensor elements also contributes to the total system MTF. In a full frame or frame transfer CCD chip, the sensor aperture is almost equal to the pixel square, i.e. a *fill factor* close to 100% is achieved. To good approximation, the sensor MTF is the Fourier transform of a box, i.e. the product of two sinc-functions:

$$P_{\text{sensor}}(\nu_1, \nu_2) = \frac{\sin(\pi\,\nu_1)\sin(\pi\,\nu_2)}{\pi^2 \nu_1 \nu_2} \tag{3.7}$$

where ν_1, ν_2 are the horizontal and vertical spatial frequency coordinates. At the Nyquist frequency of the raw grid, this MTF has an attenuation depth of 64%, whereas this value is 81% at the Nyquist frequency of the green subgrid. Other sensor types, such as interline transfer CCD and CMOS chips, have much lower fill-factors. The fill-factor can be improved by placing a micro lens on top of each pixel which collects photons from a larger fraction of the pixel square onto the sensitive portion of the sensor [Erhardt et al. 84]. Exact numbers are rarely disclosed, but in general the resulting fill factors are in the order of 70%, i.e. the effective side length s_0 of the sensitive area is 0.84 times the pixel spacing. The MTF of these sensors attenuates less:

$$P_{\text{sensor}}(\nu_1, \nu_2) = \frac{\sin(s_0\,\pi\,\nu_1)\sin(s_0\pi\,\nu_2)}{s_0^2 \pi^2 \nu_1 \nu_2}$$

For the given value $s_0 = 0.84$, the attenuation depth at the Nyquist frequencies of the raw grid and green subgrid are 73% and 86% respectively. Taking the diffraction limited MTF and sensor MTF together, we arrive at an attenuation depth of 32% at Nyquist frequency for the green subgrid of a 3 μm chip. This provides for subjectively very sharp images (at least in the image center) at the price of significant aliasing.

In high quality cameras, the MTF of the optical system is optimized for being as uniform as possible over the entire field of view and being almost free of various kinds of aberrations (spherical and chromatic aberrations, koma, vignetting etc.). This is in general achieved by sacrificing some resolution at the image center, so that the lens is no longer diffraction limited. Figure 3.5 shows MTF data for a typical high-quality lens we are going to use in experiments. It can be seen that the MTF is more uniform over the field of view and over various zoom settings than would be expected for a nearly diffraction limited system. Since high-quality lenses have lower resolution than low-quality ones, they are used with larger sensors: typical pixel pitches are 4 to 6 μm for professional digital video cameras, and 7 to 12 μm for digital single lens reflex (SLR) cameras. While larger sensors are more expensive, they also contribute to image improvement because their signal-to-noise ratio is better than that of smaller sensors. This is especially important for CMOS sensors which suffer from higher noise than CCD sensors.

Even if the lens MTF is lower than the diffraction limit, it may still be too high to prevent aliasing at the pixel spacing of the sensor. In that case, an additional optical low-pass filter must be placed in front of the sensor array. Another motivation for using such a filter is the fill factor with respect to the Bayer filter pattern: Even if the raw pixels had a

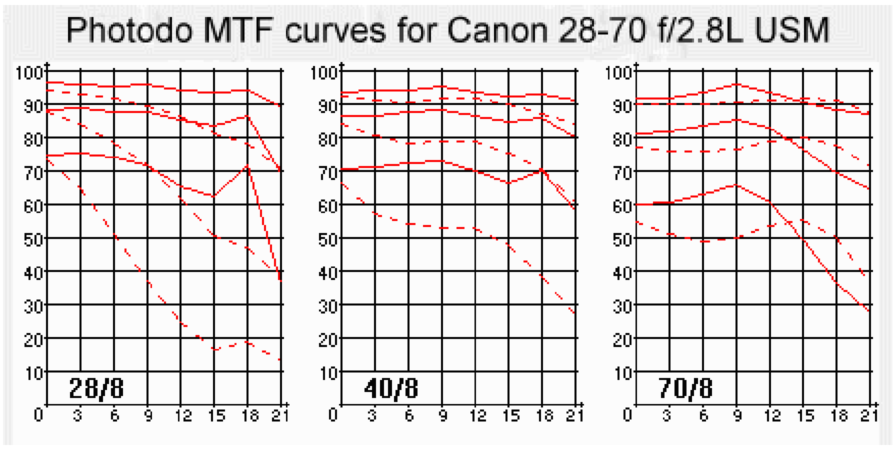


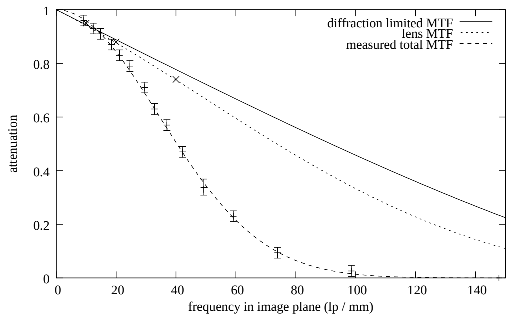
Figure 3.5: The MTF for the Canon EF 28-70/2.8 zoom lens we are going to use in experiments. The graphs show attenuation depth (in %) vs. distance from the image center (in mm). Solid lines represent the contrast of sagittal gratings (aligned like spokes in a wheel), dashed ones meridional (tangential) gratings, where measurements were taken (from top to bottom) at 10, 20, and 40 lp/mm in the image plane. Graphs from left to right are for focal lengths 28 mm, 40 mm, and 70 mm, all at $f_{\#} = 8$. Data measured and made available by Photodo AB, Lund, Sweden, www.photodo.com.

fill factor of 100%, the green subgrid's fill factor would only be 50%, and the red and blue ones' 25%. An optical lowpass filter can modify the point spread function so that a fill factor close to 100% is recovered in the green subgrid. The most common optical lowpass filter design takes advantage of the double-refraction effect of certain crystalline materials such as quartz [Pritchard 73, Greivenkamp 90]. When depolarized light passes through such a crystal at a particular orientation (namely at 45° of the crystal's principal axis), two polarized output rays emerge at a certain distance from each other. The distance can be controlled by the thickness of the crystal. A pair of rotated crystals, together with a polarization retarder in between, can split each incoming ray into four subrays located at the corners of a square. The corresponding transfer function is

$$P_{\text{prefilter}}(\nu_1, \nu_2) = \cos(\pi \nu_1 s_o) \cos(\pi \nu_2 s_o)$$

where s_o is the side length of the square. It is even possible to achieve different amounts of blurring for different colors [Greivenkamp 90], as would be appropriate for a Bayer filter camera.

Since exact measurements of the properties of the lens, optical prefilters and sensor apertures of our cameras don't seem to be publicly available, we must measure the MTF on our own. There are several approaches to MTF measurement. The most convenient is to take images of a resolution test chart, e.g. the Siemens star [Jähne 97]. However, this gives only rough estimates of the camera properties, especially at frequencies near and beyond the Nyquist frequency. The most accurate results can be achieved by taking images of sinusoid or rectangle gratings at one spatial frequency at a time. Since only one known grating is present in the target, one can always assign the measured contrast



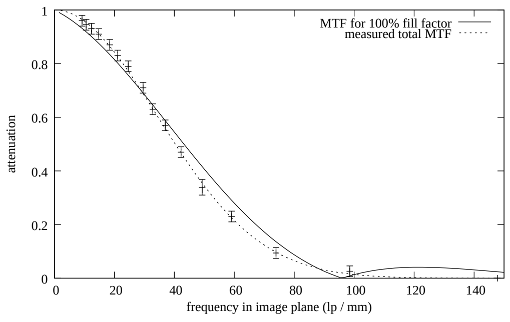


Figure 3.6: Left: Transfer functions of the Canon EOS D60 with lens EF 28-70/2.8 at focal length $f = 28\,\text{mm}$ and aperture $f_{\#} = 8$. The lens MTF is a fit to data points taken from figure 3.5 (indicated by $\times$). Measurements were taken in the green subgrid, whose Nyquist frequency is 48 lp/mm. Right: Comparison of the measured MTF with the expected MTF for square pixels with 100% fill factor at the resolution of the green subgrid (i.e. the product of the lens MTF and a sinc-function according to (3.7)).

to the correct frequency, even when a high-frequency grating is only observed as a Moiré pattern. Moreover, since the contrast can be measured over several periods of the grating, measurements have a good signal-to-noise ratio even at low contrast. Figure 3.6 shows the MTF measured in this way for the Canon EOS D60 digital SLR camera with zoom lens EF 28-70/2.8 at a focal length of 28 mm and a relative aperture of $f_{\#} = 8$ (these are the settings with minimal attenuation). This camera has a CMOS sensor with $7.4\,\mu$m pixel pitch and a Bayer filter according to figure 3.4. The Nyquist frequency of the raw grid is 68 lp/mm, and of the green subgrid 48 lp/mm. An optical prefilter, whose properties are not publicly specified, is mounted on top of the chip. The measured total MTF can be well approximated by a Gaussian with standard deviation $\sigma = 4.65\,\mu$m, which is 63% of the raw pixel pitch, and 44% of the pixel pitch in the green subgrid. For comparison, we also show an approximate lens MTF that is fitted to measurements taken from figure 3.5 (precisely, this curve is the product of the diffraction limited MTF and a Gaussian with $\sigma = 1.27\,\mu$m). The attenuation depth of the total MTF at the Nyquist frequency of the green subgrid is 37%. Figure 3.6 right shows that the optical prefilter has approximately the same effect as if the green subgrid had a fill factor of 100%.

Unfortunately, the method illustrated above is rather expensive. A much simpler method that still gives reasonably accurate MTF measurements is the *slanted edge* technique standardized in [ISO 12233:2000]. In this approach a high-contrast step edge is imaged, and the MTF can be recovered as the Fourier transform of the image's derivative perpendicular to the edge. To improve the method's signal-to-noise ratio and accuracy, the resolution of the edge is first enhanced by means of a simple super-resolution algorithm which computes a four-fold oversampled 1-D edge by taking averages of many scanlines of the 2-D image. Before averaging, each scanline is shifted according to the edge's subpixel location in that line. Super-resolution is only achieved when these subpixel locations differ slightly from line to line, i.e. when the edge is slightly slanted (typically by about 5°). This gives the method its name. Figure 3.7 shows some MTFs we measured

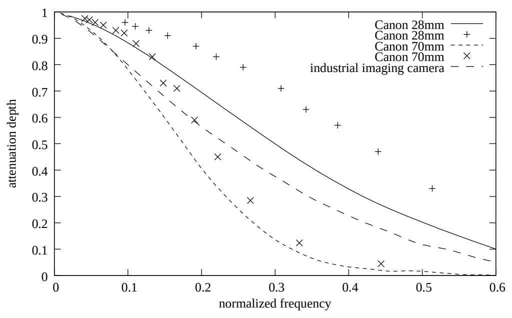


Figure 3.7: MTFs measured by the slanted edge technique for Canon EOS D60 at 28mm and 70mm focal length, and a typical industrial imaging camera (curves). For comparison, measurements with the grating technique are also presented (crosses). The frequency coordinate is normalized so that the Nyquist frequency is 0.5 cycles per pixel for all cameras.

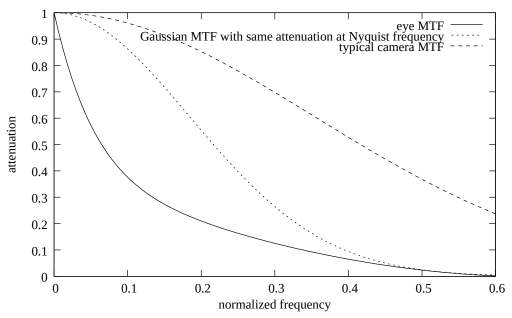


Figure 3.8: Comparison of the eye MTF according to (3.5) with the camera MTF according to figure 3.6 and with the Gaussian MTF (3.8) that has the same effective band limit as the eye MTF. The frequency coordinate is normalized to a Nyquist frequency of 0.5.

this way. It can be seen that the slanted edge technique somewhat underestimates the MTF.

The normalized transfer functions for the eye, the real camera and a hypothetical “band-limited camera” are compared in figure 3.3. The MTF of this band-limited camera is a Gaussian that has the same attenuation at the Nyquist frequency as the MTF of the human eye:

$$P_{\text{band-limited camera}}(\nu) = e^{-\frac{(2\pi\,\nu)^2 0.865^2}{2}} \tag{3.8}$$

It is evident that the eye MTF has the strongest attenuation at all frequencies. The band-limited camera has approximately the same behavior near and beyond the Nyquist frequency, but attenuates low frequencies less. In contrast, the real camera MTF allows significant aliasing. We can compare the effect of the three MTFs by means of a simulated image acquisition process [Park & Rahman 99]: We first take an image at high resolution, and then downsample it to 1/4 of its linear resolution, using the three MTFs (normalized to the Nyquist frequency of the *new* resolution) as prefilters. The aliasing noise is computed from the contributions of the appropriate spectrum replications according to (3.2). The results can be seen in figure 3.9. The signal-to-aliasing-noise ratios for the real camera, band-limited camera, and the eye are 40, 920, and 423 respectively. In other words, the SNR of the band-limited camera is 23 times higher than that of the real camera. One could ask what happened if we applied an additional post-filter to the image of the real camera. When the image has an exponential autocorrelation, i.e. its power spectrum conforms to (3.3), aliasing is concentrated at high frequencies, and a post-filter could successfully reduce aliasing noise. When the post-filter is a Gaussian with $\sigma = 0.71$, the total amount of smoothing is equivalent to the band-limited prefilter. However, the resulting signal-to-aliasing-noise ratio is 250, i.e. only 25% of what we get

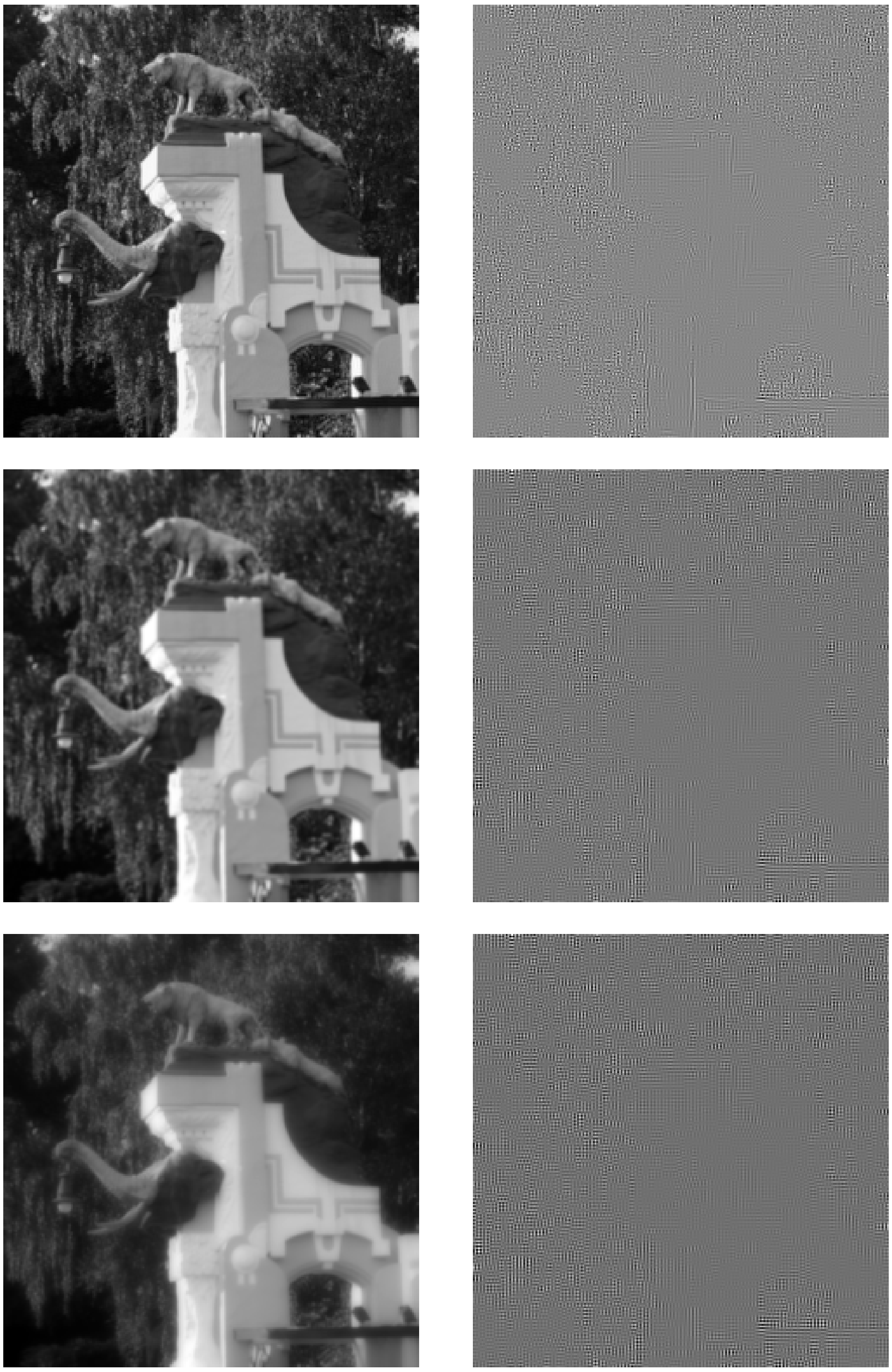


Figure 3.9: Left column from top to bottom: Simulated image acquisition with camera MTF, effectively band-limited Gaussian, and eye MTF. Right column: Corresponding aliasing noise (the second and third aliasing images have been amplified by a factor of 20 relative to the first one to make the noise visible at all).

with the band-limited pre-filter. To achieve the same SNR as the latter, the post-filter must have $\sigma = 1.1$, i.e. the image has to be smoothed more. As far as aliasing is concerned, using a band-limited camera is preferable. The consequences of aliasing noise will be considered in detail in section 7.2.2.1.

3.3 Reconstruction of the Analog Camera Image

Even if the PSF were perfectly band-limited, this wouldn't buy us anything if we weren't able to reconstruct the analog image from the actual digital image. In theory, a band-limited image can be reconstructed from the (noise-free) sampled image by convolution with the ideal interpolator

$$\tilde{f}(x,y) = \sum_{k,l} f_{kl}\, \mathrm{sinc}(x-k)\,\mathrm{sinc}(y-l) \tag{3.9}$$

where $\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$. This is always true when the MTF is band-limited: the ideal interpolator suppresses the spectrum repetitions caused by sampling, independent of how the camera MTF is shaped below the band limit. Unfortunately, the ideal interpolator cannot be realized in practice because it decays much too slowly (only as $\mathcal{O}(x^{-1})$) so that unacceptable cut-off errors occur when the filter is restricted to a finite window of reasonable size. Interpolating splines are a natural approximation because they can be easily and efficiently implemented and converge to the ideal interpolator as their order increases.

3.3.1 Spline Interpolation

B-spline interpolation is defined similarly to the ideal interpolator:

$$\tilde{f}(x,y) = \sum_{k,l} c_{kl}\, b_n(x-k) b_n(y-l) \tag{3.10}$$

where $b_n(x)$ are n^{th}-order B-spline basis functions, i.e. piecewise 1D polynomials of degree n, but the weights c_{kl} must now differ from the samples f_{kl} in order to guarantee interpolation.[6] Fortunately, the c_{kl} can be easily computed by means of recursive filters p_n:

$$c_{kl} \quad = \quad p_{n,\text{horizontal}} \star p_{n,\text{vertical}} \star f_{kl} \tag{3.11}$$

The recursive filters p_n will be explained below. B-splines of order n are defined by the n-fold convolution of the 0^{th}-order B-spline with itself:

$$b_n(x) = \underbrace{b_0(x) \star \cdots \star b_0(x)}_{(n+1)\text{ terms}} \tag{3.12}$$

[6]Surprisingly, this is overlooked in some books and papers describing "spline interpolation", e.g. [Eberly 96, Pratt 01, Allebach 05]. These authors apply equation (3.10) directly to the samples (i.e. $c_{kl} = f_{kl}$) and consequently end up with an approximation instead of an interpolation.

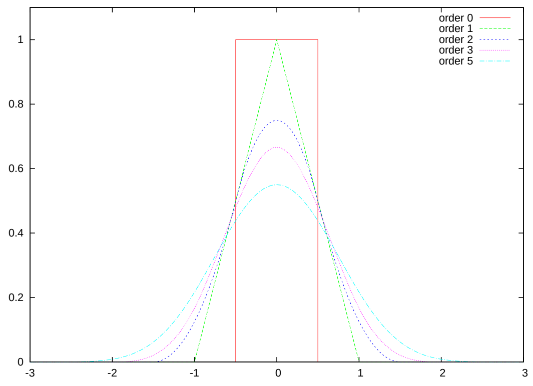


Figure 3.10: The first few B-spline functions.

where $b_0(x)$ is simply the unit rectangle function

$$b_0(x) = \text{rect}(x) = \begin{cases} 1 & \text{if } -\frac{1}{2} \le x < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

The first few B-spline functions are depicted in figure 3.10. Equation (3.12) is equivalent to the recursion relation

$$b_n(x) = \frac{\left(\frac{n+1}{2} + x\right) b_{n-1}\left(x + \frac{1}{2}\right) + \left(\frac{n+1}{2} - x\right) b_{n-1}\left(x - \frac{1}{2}\right)}{n}$$

However, computing spline values via this recursion is quite slow, so in practice we prefer the explicit expressions given in table 3.1. Zero order spline interpolation is equivalent with nearest-neighbor interpolation (every pixel is filled with the value of its center point), and first order is equivalent to bilinear interpolation.

Since splines are piecewise polynomial functions, the polynomial's coefficients remain constant within certain regions. These regions have the same size as the pixels and are called *facets*. When $\vec{s}$ is a grid point, the corresponding facets extend from $\vec{s} - \left(\frac{1}{2}, \frac{1}{2}\right)^T$ to $\vec{s} + \left(\frac{1}{2}, \frac{1}{2}\right)^T$ for even order splines, and from $\vec{s}$ to $\vec{s} + (1,1)^T$ for odd order ones.

The transfer functions of the B-splines are powers of the sinc-function

$$B_n(u) = \left(\frac{\sin(\pi\, u)}{\pi\, u}\right)^{n+1} = \text{sinc}\,(u)^{n+1}$$

B-splines have a number of important properties:

1. They are $(n-1)$-times continuously differentiable.
2. The support of a B-spline is the smallest possible for a spline function of the given order. B-splines are only non-zero for $|x| < (n+1)/2$ and therefore efficiently computable.
3. The B-splines converge towards Gaussians as $n \to \infty$, and spline interpolation according to (3.10) approaches sinc interpolation.

order	expressions
1	$b_1(x) = \begin{cases} 1-\lvert x\rvert & \lvert x\rvert \le 1 \\ 0 & \text{else} \end{cases}$
2	$b_2(x) = \begin{cases} \frac{3}{4}-\lvert x\rvert^2 & \lvert x\rvert \le \frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2}-\lvert x\rvert\right)^2 & \frac{1}{2} \le \lvert x\rvert \le \frac{3}{2} \\ 0 & \text{else} \end{cases}$
3	$b_3(x) = \begin{cases} \frac{2}{3}-\lvert x\rvert^2+\frac{1}{2}\lvert x\rvert^3 & \lvert x\rvert \le 1 \\ \frac{1}{6}\left(2-\lvert x\rvert\right)^3 & 1 \le \lvert x\rvert \le 2 \\ 0 & \text{else} \end{cases}$
4	$b_4(x) = \begin{cases} \frac{115}{192}-\frac{5}{8}\lvert x\rvert^2+\frac{1}{4}\lvert x\rvert^4 & \lvert x\rvert \le \frac{1}{2} \\ \frac{55}{96}+\frac{5}{24}\lvert x\rvert-\frac{5}{4}\lvert x\rvert^2+\frac{5}{6}\lvert x\rvert^3-\frac{1}{6}\lvert x\rvert^4 & \frac{1}{2} \le \lvert x\rvert \le \frac{3}{2} \\ \frac{1}{24}\left(\frac{5}{2}-\lvert x\rvert\right)^4 & \frac{3}{2} \le \lvert x\rvert \le \frac{5}{2} \\ 0 & \text{else} \end{cases}$
5	$b_5(x) = \begin{cases} \frac{11}{20}-\frac{1}{2}\lvert x\rvert^2+\frac{1}{4}\lvert x\rvert^4-\frac{1}{12}\lvert x\rvert^5 & \lvert x\rvert \le 1 \\ \frac{17}{40}+\frac{5}{8}\lvert x\rvert-\frac{7}{4}\lvert x\rvert^2+\frac{5}{4}\lvert x\rvert^3-\frac{3}{8}\lvert x\rvert^4+\frac{1}{24}\lvert x\rvert^5 & 1 \le \lvert x\rvert \le 2 \\ \frac{1}{120}\left(3-\lvert x\rvert\right)^5 & 2 \le \lvert x\rvert \le 3 \\ 0 & \text{else} \end{cases}$

Table 3.1: Explicit expressions for the B-splines up to order 5.

At this point a comparison of splines with the *facet model* of [Haralick & Shapiro 92] may be of interest. Both approaches reconstruct the image function by means of piecewise polynomial functions. Haralick obtains these functions by a least squares fit to the samples in every $n \times n$ window, where n^2 must at least equal the number of coefficients, but can be greater if smoothing is desired. However, since these functions are computed independently in every facet, the resulting analog image will be discontinuous across facet borders. Consequently, it is impossible to guarantee feature (e.g. edge) continuity between neighboring pixels. This is in contrast to property 1 of spline interpolation. Since the computational effort for the facet model and spline interpolation are comparable, and smoothing can be achieved in the latter approach by a suitable prefilter, splines should be preferred.

The coefficients c_{kl} for B-spline interpolation can be computed very efficiently by the prefilters p_n. Since the B-splines themselves are bell-shaped and act as smoothing filters, the interpolation condition can only be fulfilled if the c_{kl} represent a sharpened version of the image f_{kl}. At the sampling points, the amount of sharpening must exactly counter

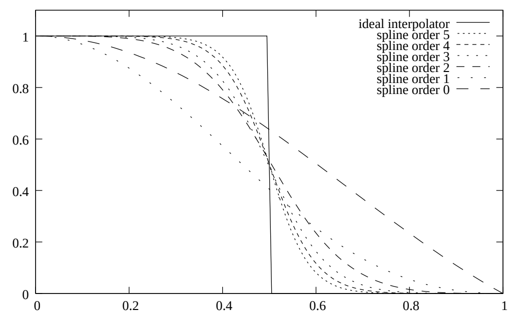


Figure 3.11: As the spline order increases, the transfer functions $B_n(u)P_n(u)$ of spline interpolation converge towards the ideal interpolator.

the effect of B-spline smoothing. Therefore, the transfer function P_n of the prefilters $p_n(i)$ must be the inverse of the transfer function of the sampled B-splines $b_n(i)$. For orders 0 and 1, P_0 and P_1 are just identity filters, whereas the 1-dimensional transfer functions for higher orders are

$$\begin{aligned}
P_2(u) &= \frac{4}{3+\cos(2\pi\,u)} \\
P_3(u) &= \frac{3}{2+\cos(2\pi\,u)} \\
P_4(u) &= \frac{192}{115+76\cos(2\pi\,u)+\cos(4\pi\,u)} \\
P_5(u) &= \frac{60}{33+26\cos(2\pi\,u)+\cos(4\pi\,u)} \\
P_n(u) &= \left(\sum_{k=0}^{\lfloor n/2\rfloor} b_n(k)\,\cos(2k\pi\,u)\right)^{-1}
\end{aligned}$$

In 2D, these prefilters can be expressed as sets of $\lfloor n/2 \rfloor$ first-order recursive filters. Each of them must be applied in the four major directions of the image (left to right, right to left, top to bottom, bottom to top). Table 3.2 lists the filter coefficients for the first few spline orders. Figure 3.11 compares the transfer functions of spline interpolation $B_n(u)P_n(u)$ with the ideal interpolator.

Only one addition and multiplication per pixel are required for each run of a first-order recursive filter. The algorithmic complexity for the computation of the coefficients c_{kl} is therefore $4\lfloor n/2 \rfloor \mathcal{O}(w \cdot h)$, i.e. linear in the number of pixels, with a very small constant factor. Once the coefficients c_{kl} have been computed, we can calculate function

spline order n	0	1	2	3	4	5
p_1	-	-	$2\sqrt{2}-3$	$\sqrt{3}-2$	-0.3613412259	-0.4305753471
p_2	-	-	-	-	-0.0137254293	-0.0430962882

Table 3.2: The coefficients of the recursive prefilters to compute the spline coefficients c_{ij} from the image samples f_{ij}.

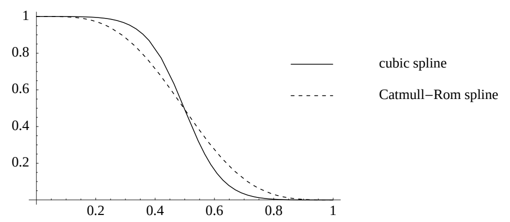


Figure 3.12: The transfer functions of cubic spline and Catmull-Rom interpolation. Cubic spline interpolation approximates the ideal interpolator better, with only marginally greater computationally effort.

values $\tilde{f}(x, y)$ at arbitrary real valued coordinates by evaluating (3.10). The required effort increases with the degree of the spline. For example, we need 44 multiplications per evaluation for a 3rd-order spline, and 102 for a 5th-order one (these numbers include the effort for computing the required B-spline values and for evaluating the convolution sum). In practice, this translates into roughly 2 million evaluations per second for the 3rd-order spline and half as many for the 5th-order one, both measured on a 3GHz Linux machine using the GNU C++ compiler version 3.3. When the sample positions are not entirely random, so that only a few different facet coordinates are actually needed, the B-spline values can be pre-computed, and the effort reduces to that of standard convolution. If the new points are located on a grid, the computational burden can be further reduced by taking advantage of the separability of the 2-dimensional B-spline functions.

It is also instructive to compare the quality and computational effort of spline interpolation with other interpolation methods. One popular alternative is the Catmull-Rom function [Catmull & Rom 74] which is defined in 1D by

$$\mathrm{CatmullRom}(x) = \begin{cases} 1 - \frac{5}{2}\,|x|^2 + \frac{3}{2}\,|x|^3 & \text{if } |x| \le 1 \\ 2 - 4\,|x| + \frac{5}{2}\,|x|^2 - \frac{1}{2}\,|x|^3 & \text{if } 1 < |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

This definition can be inserted in (3.10) instead of the cubic B-spline $b_3(.)$ to interpolate function values at arbitrary real-valued coordinates. Since the Catmull-Rom function is a cardinal function (i.e. it evaluates to 1 at $x = 0$ and to 0 at all other integers), the coefficients c_{kl} are equal to the discrete image values f_{kl}, so that prefiltering is not necessary. Apart from this, the computational effort for Catmull-Rom and cubic spline interpolation is the same, because both basis functions are piecewise cubic polynomials supported between -2 and 2. The performance advantage of Catmull-Rom interpolation is small when many points must be interpolated, so that the prefiltering effort can be amortized over a large number of spline evaluations. On the other hand, the quality of cubic spline interpolation is significantly higher – its transfer function approximates the ideal interpolator much better (see figure 3.12), and the resulting interpolant is twice continuously differentiable, whereas the Catmull-Rom interpolant has only one continuous derivative. Unless the prefiltering effort cannot be tolerated at all, spline interpolation is always preferable. Similar arguments can be put forward for other alternative interpolators.

The quality of spline interpolation can be demonstrated in a simple experiment: Rotate a given image by a small angle (e.g. 10°) and compute the gray values of the rotated image by means of spline interpolation. Then rotate the resulting image by 10° again, and repeat this process until a full revolution of 360° is achieved after 36 steps. If the interpolation of the intermediate images worked perfectly, the final image would be identical to the original, so that their root mean square (RMS) difference would be zero. Since spline interpolation approximates the ideal interpolator, we expect the RMS to decrease as the spline order increases, and that the error is already quite small for 5th-order splines. Figure 3.13 shows some results. It can be seen that nearest-neighbor interpolation completely destroys the image, and linear interpolation results in significant blurring. In contrast, the result of 5th-order interpolation is still very close to the original after 36 interpolation steps.

It is often useful to write (3.10) in an alternative form that makes the polynomial in each facet explicitly accessible. For example, this is necessary when one wants to apply algorithms that only work on polynomials, such as polynomial root finders. It also stresses the similarity of spline interpolation to Haralick's facet model – the two models only differ in the way how coefficients are calculated and in the consequential fact that splines are differentiable across facet borders. To derive the polynomial form of the interpolation formula, we first define the facet index (i_f, j_f) of the coordinate (x, y)

$$(i_f, j_f) = \begin{cases} (\lfloor x \rfloor, \lfloor y \rfloor) & \text{if } n \text{ odd} \\ (\lfloor x + \frac{1}{2} \rfloor, \lfloor y + \frac{1}{2} \rfloor) & \text{if } n \text{ even} \end{cases}$$

where n is the order of the spline, and $\lfloor . \rfloor$ denotes the floor operation. Then we define local facet coordinates

$$(u, v) = (x - i_f, y - j_f)$$

The domain of the facet coordinates is defined by $u, v \in [0, 1)$ if n is odd, and $u, v \in \left[-\frac{1}{2}, \frac{1}{2}\right)$ if n is even. Then we can express $\tilde{f}(x, y)$ as

$$\tilde{f}(x, y) = \mathbf{v}^T \mathbf{B}\, \mathbf{u} \tag{3.14}$$

where $\mathbf{B} = \mathbf{B}(i_f, j_f)$ is the coefficient matrix for the current facet and $\mathbf{u}$ and $\mathbf{v}$ are vectors of monomials of the local facet coordinates

$$\begin{aligned} \mathbf{u} &= \left(1, u, u^2, ..., u^n\right)^T \\ \mathbf{v} &= \left(1, v, v^2, ..., v^n\right)^T \end{aligned}$$

To compute $\mathbf{B}$ we define a matrix $\mathbf{C} = \mathbf{C}(i_f, j_f)$ containing the spline coefficients c_{ij} from a window around the current facet index according to the following rule

$$C_{kl} = c_{i_f - \lfloor n/2 \rfloor + k,\, j_f - \lfloor n/2 \rfloor + l} \qquad k, l = 0, ..., n$$

with reflective boundary conditions. Then $\mathbf{B}$ is given by

$$\mathbf{B} = \mathbf{W}\,\mathbf{C}\,\mathbf{W}^T$$

where $\mathbf{W}$ is a position-independent matrix characterizing the B-spline of order n

$$W_{ij} = \frac{1}{i!} \frac{d^i}{d\,x^i} b_n\left(\lfloor n/2 \rfloor - j\right) \qquad i, j = 0, ..., n$$

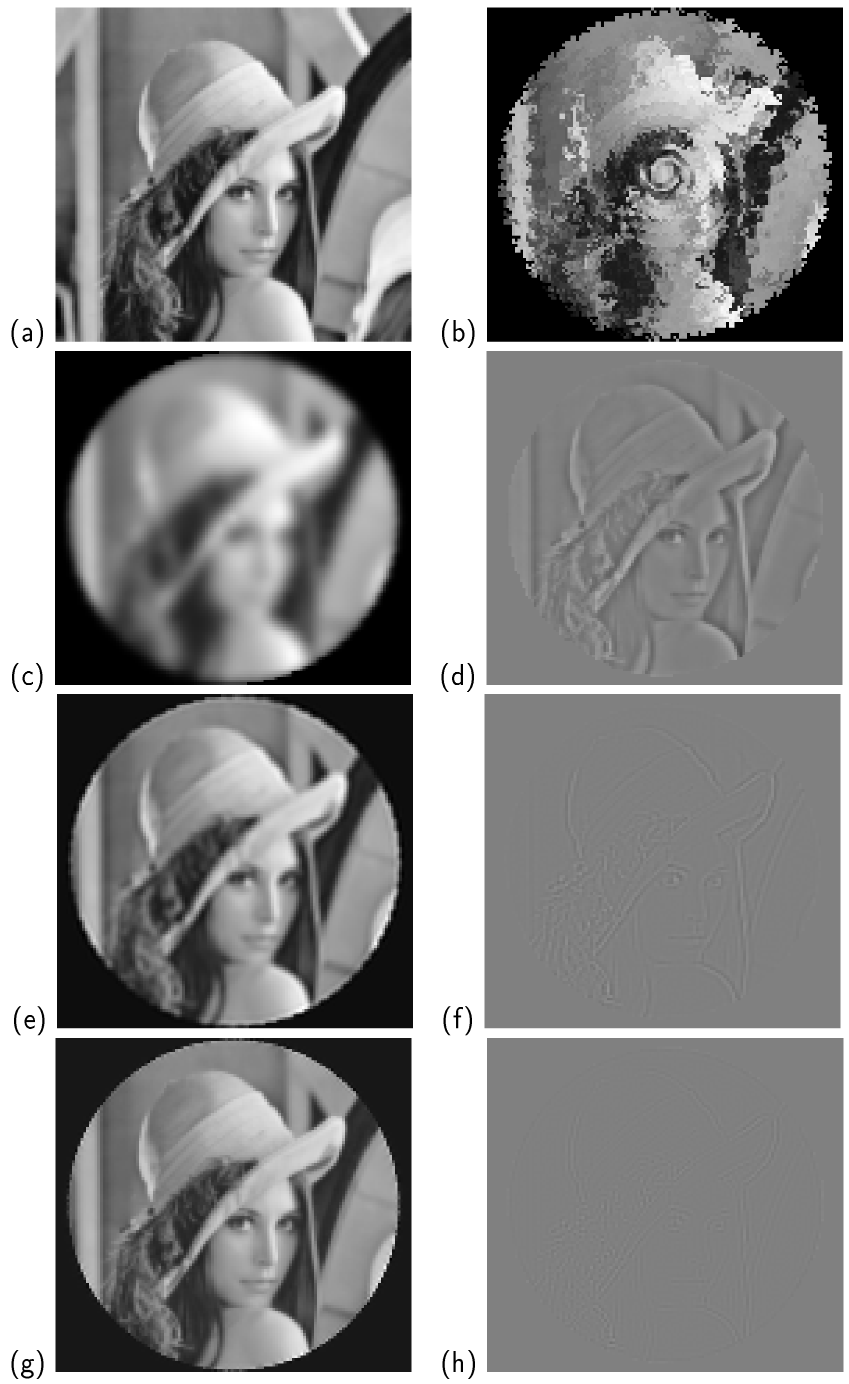


Figure 3.13: Image rotation by spline interpolation (36 consecutive rotations of 10°): (a) original image (range is 0...255); (b) result using 0th-order spline (nearest-neighbor interpolation); (c) result and (d) difference to original for 1st-order spline (linear interpolation), RMS = 17 gray levels; (e) result and (f) difference for 3rd-order spline, RMS = 4.4 gray levels; (g) result and (h) difference for 5th-order spline, RMS = 2.3 gray levels.

order	expressions
1	$\mathbf{W} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$
2	$\mathbf{W} = \begin{pmatrix} \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix}$
3	$\mathbf{W} = \begin{pmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 \\ -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \end{pmatrix}$
4	$\mathbf{W} = \begin{pmatrix} \frac{1}{384} & \frac{19}{96} & \frac{115}{192} & \frac{19}{96} & \frac{1}{384} \\ -\frac{1}{48} & -\frac{11}{24} & 0 & \frac{11}{24} & \frac{1}{48} \\ \frac{1}{16} & \frac{1}{4} & -\frac{5}{8} & \frac{1}{4} & \frac{1}{16} \\ -\frac{1}{12} & \frac{1}{6} & 0 & -\frac{1}{6} & \frac{1}{12} \\ \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} \end{pmatrix}$
5	$\mathbf{W} = \begin{pmatrix} \frac{1}{120} & \frac{13}{60} & \frac{11}{20} & \frac{13}{60} & \frac{1}{120} & 0 \\ -\frac{1}{24} & -\frac{5}{12} & 0 & \frac{5}{12} & \frac{1}{24} & 0 \\ \frac{1}{12} & \frac{1}{6} & -\frac{1}{2} & \frac{1}{6} & \frac{1}{12} & 0 \\ -\frac{1}{12} & \frac{1}{6} & 0 & -\frac{1}{6} & \frac{1}{12} & 0 \\ \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} & 0 \\ -\frac{1}{120} & \frac{1}{24} & -\frac{1}{12} & \frac{1}{12} & -\frac{1}{24} & \frac{1}{120} \end{pmatrix}$

Table 3.3: Matrices needed for the explicit computation of the spline polynomials within each facet.

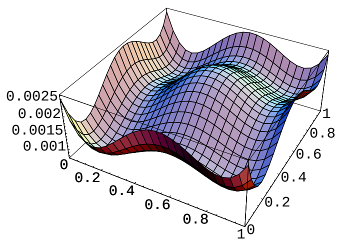
The B-spline derivatives in this expression must be computed according to the recursion

$$\frac{d}{dx} b_n\left(x\right) = b_{n-1}\left(x + \frac{1}{2}\right) - b_{n-1}\left(x - \frac{1}{2}\right) \tag{3.15}$$

Note that the n^{th} derivative of $b_n(x)$ is discontinuous. The recursion (3.15), together with (3.13), ensures that correct limits are taken at discontinuities. Table 3.3 lists explicit expressions for the matrices $\mathbf{W}$ for the first few spline orders. Much more detail about spline interpolation can be found in [Unser et al. 93].

3.3.2 Experiment: Detection of Extrema and Saddle Points in Spline-Interpolated Images

The number and location of extrema and saddle points are the simplest geometric features we can measure in an analog function. Therefore, we construct a band-limited image



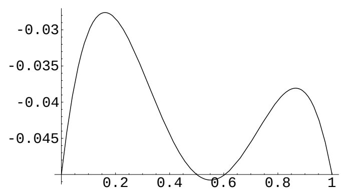


Figure 3.14: Our test function f for critical point detection (left) is the Euclidean product of the function s_3 (right).

where 9 such points are located at known positions. Such a function can easily be obtained as a linear combination of integer-translated sinc-functions. For our experiments, we choose the function

$$\begin{aligned} f(\vec{x}) &= s_3(x_1)s_3(x_2) \\ \text{with} \quad s_1(x,k) &= \operatorname{sinc}(x+k) + \frac{4}{5}\operatorname{sinc}(x-k-1) \\ s_2(x,k) &= s_1(x,k)/s_1(0.5,k) \\ s_3(x) &= s_2(x,1) - s_2(x,2) - 0.05 \end{aligned} \tag{3.16}$$

As can be seen in figure 3.14 this function has 9 critical points in the unit square $[0,1]\times[0,1]$ – four minima, four saddles, and one maximum. Since the function is band-limited with a Nyquist frequency of $\frac{1}{2}$, sampling at unit distance would be sufficient provided the image were infinitely large and we actually used exact sinc interpolation. Since neither condition can be fulfilled in practice, we must sample at smaller steps. Moreover, we allow the grid to be rotated and translated arbitrarily, i.e. $\vec{x} = h\,\mathcal{R}^T(\alpha)(\vec{x}' - \vec{d})$ where h is the sample distance, $\mathcal{R}(\alpha)$ a rotation matrix, $\vec{x}'$ a grid point and $\vec{d}$ a (sub-pixel) shift (detailed parameter settings are given below). Figure 3.15 shows an example reconstructions obtained at $h = 1/2.6$ (which turns out to be the lowest sampling rate where we can reasonably approximate exact reconstruction) with splines up to order 5. The 5th-order spline reconstruction is almost indistinguishable from the original at this sample distance, whereas the nearest neighbor and linear reconstructions (orders 0 and 1) hardly resemble it at all.

When we want to detect extrema and saddle points in a spline reconstruction of the test image, we must distinguish differentiable splines (order 2 and higher) and non-differentiable ones (orders 0 and 1). In the latter case we use the following discrete definitions:

- 0th-order spline, 4-neighborhood: a minimum (maximum) is a point whose 4-neighbors are lower (higher) than the center. A saddle has two opposite neighbors that are higher and two that are lower.
- 0th-order spline, 8-neighborhood: a minimum (maximum) is a point whose 8-neighbors are lower (higher) than the center. A saddle point is characterized by the property

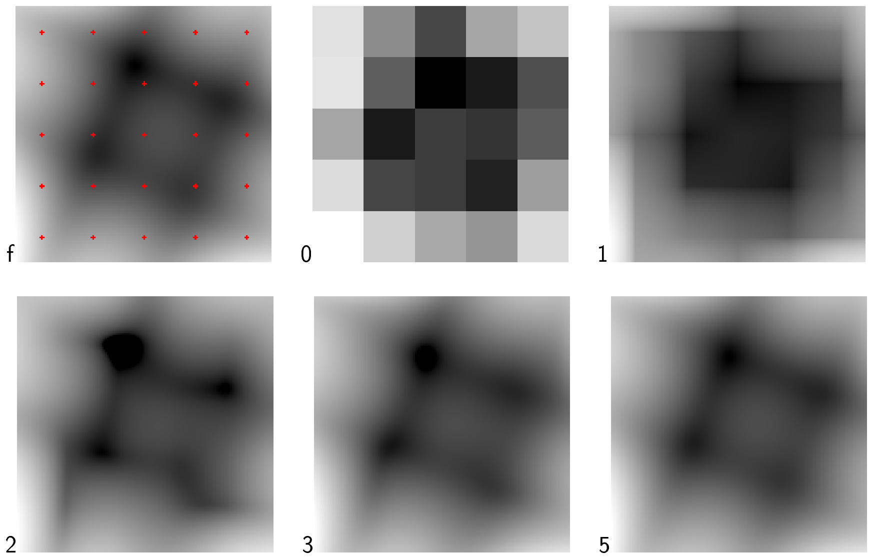


Figure 3.15: Reconstruction of the test function (sample distance 0.39, $\alpha = -22.5°$, $d_1 = 0.3$, $d_2 = -0.2$) by various splines. Left to right, top to bottom: Location of the sampling points (red) on the test function f, reconstruction by spline orders 0, 1, 2, 3, 5 (the region used for reconstruction was four times larger than the region depicted).

that the difference between the center and its neighbors changes sign more than two times while one walks over the 8 neighbors in counter-clockwise order.

- 1^{st}-order spline: minima, maxima, and saddles are defined as in the previous case. In addition, there may be saddles in the interior of a facet, when the bilinear function is hyperbolic in this facet (i.e. has a point with vanishing gradient).

One could extend these definitions for the case that neighbors have the same value as the center, but this is non-trivial and not necessary for the present experiment. In case of differentiable splines, the extrema and saddles are exactly the points where the gradient vanishes. In the present experiment, these are always isolated points. Since a spline is a piecewise polynomial function, we can in principle solve the localization problem directly: use (3.14) to compute an explicit expression for the gradient in every facet (pixel) and set it to zero. This gives us a polynomial system with two equations in two variables. After eliminating one of the variables, we can find the roots of the remaining polynomial by standard methods. Among these roots, we keep only those whose corresponding coordinates are actually located in the current facet. Unfortunately, the order of the polynomial to be solved increases as $2n(n-1)+1$, where n is the spline order. This quickly becomes very inefficient and prone to numerical instabilities, especially considering that one rarely finds more than two critical points per facet in practice. We only employ the direct method for 2^{nd}-order splines.

For higher order splines, it is more appropriate to use an iterative approach. Consider the second order Taylor series expansion of the function $\tilde{f}(\vec{x})$:

$$\tilde{f}(\vec{x}_0 + \Delta\vec{x}) = \tilde{f}(\vec{x}_0) + \nabla\tilde{f}(\vec{x}_0)\Delta\vec{x} + \frac{1}{2}(\Delta\vec{x})^T H(\vec{x}_0)\,\Delta\vec{x}$$

where $H(\vec{x}_0)$ is the Hessian matrix at point $\vec{x}_0$. If $\vec{x}_0 + \Delta\vec{x}$ is a critical point, the gradient with respect to $\Delta\vec{x}$ of this expression must be zero. Solving $\partial\tilde{f}/\partial(\Delta\vec{x}) = 0$ for $\Delta\vec{x}$ gives $\Delta\vec{x} = H^{-1}(\vec{x}_0)\,\nabla\tilde{f}(\vec{x}_0)$, which can be used to define a sequence of iterative corrections that quickly converge to a critical point. In order to find all critical points, we have determined experimentally that four different starting points per pixel are required for typical natural images containing significant detail. Since the total number of critical points is always much less, many points will be detected several times, so that efficient duplicate removal is also required. This gives the following algorithm:

Algorithm 3.1: Iterative critical point detection in a 2D spline

Input: A spline function $\tilde{f}(\vec{x})$ of order $n \geq 3$ over the domain $[0, w] \times [0, h]$. The desired accuracy ϵ of the algorithm.

1. For $x = 0, \frac{1}{2}, 1, \frac{3}{2} ..., w$ and $y = 0, \frac{1}{2}, 1, \frac{3}{2}, ...h$:

   a) Choose $\vec{x}^{(0)} = (x, y)^T$ as iteration starting point and compute

   $$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \Delta\vec{x}^{(k)} = \vec{x}^{(k)} + H^{-1}\left(\vec{x}^{(k)}\right)\nabla\tilde{f}\left(\vec{x}^{(k)}\right) \tag{3.17}$$

   until $\left|\Delta\vec{x}^{(k)}\right| \leq \epsilon$. Let the point of convergence be $\vec{s}$.

   b) Check whether the resulting critical point $\vec{s}$ is already known (this can be done quickly by means of an efficient data structure for nearest neighbor queries, e.g. a kD-tree [Wendland 05]). If not, store $\vec{s}$ as a new critical point and determine its type by means of the eigenvalues of the Hessian at $\vec{s}$: if both eigenvalues are positive, $\vec{s}$ is a minimum, if both are negative, it is a maximum, and if their signs differ, we found a saddle point.

It is remarkable that this algorithm even works for splines of order 2, where the Hessian is constant within every facet, but discontinuous across the facet borders. In theory, critical point detection could fail here because the iterations may oscillate between neighboring facets. But in practice, we have never actually encountered convergence problems for $n = 2$.

To study critical point detection systematically, we created 1620 test images from the function f, namely at sample distances $h \in \{\frac{1}{2.6}, \frac{1}{3}, \frac{1}{5}, \frac{1}{8}, \frac{1}{32}\}$, rotation angles $\alpha \in \{-45°, -40°, -22.5°, -10°, 0°, 12°, 22.5°, 30°, 45°\}$, and shifts $d_1, d_2 \in \{-0.4, -0.2, 0.0, 0.1, 0.3, 0.5\}$. Critical points were detected by looking in the 4- or 8-neighborhood at the given resolutions, and by means of adaptive sampling on the basis of spline interpolation of various orders. The most striking finding was that even at the highest resolution $h = \frac{1}{32}$ it is insufficient to only look in the 4- or 8-neighborhood: When the function is not aligned

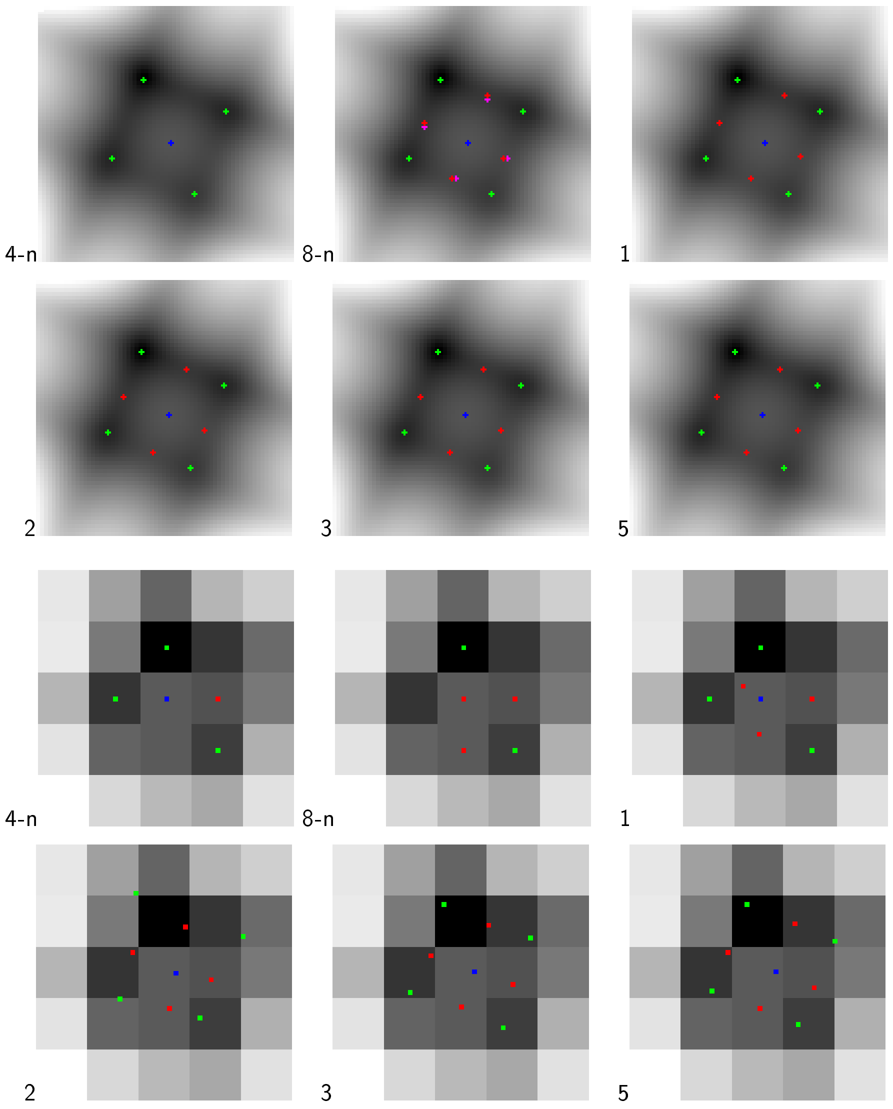


Figure 3.16: Critical point detection at $h = \frac{1}{32}$ (upper rows) and $h = \frac{1}{2.6}$ (lower rows) ($\alpha = -22.5°$, $d_1 = 0.3$, $d_2 = -0.2$). Results of 4- and 8-neighborhood methods, spline methods of order 1, 2, 3 and 5 (all overlayed over the nearest-neighbor interpolation). Minima, maxima, saddles, and false positives are marked green, blue, red, and magenta respectively.

with the coordinate axes, these non-adaptive methods frequently result in missing points or mis-detections. The upper rows of figures 3.16 and 3.17 show two representative cases. These failures are due to the fact that a 3×3 neighborhood does not contain enough shape information for reliable decisions about critical points. Different kinds of errors are typical: Nearest-neighbor interpolation with 4-neighborhood classification usually misses critical points, whereas 8-neighborhood classification finds too many (error rates in both cases are over 85% when $\alpha \neq 0°$, and saddles are the point type with most errors).

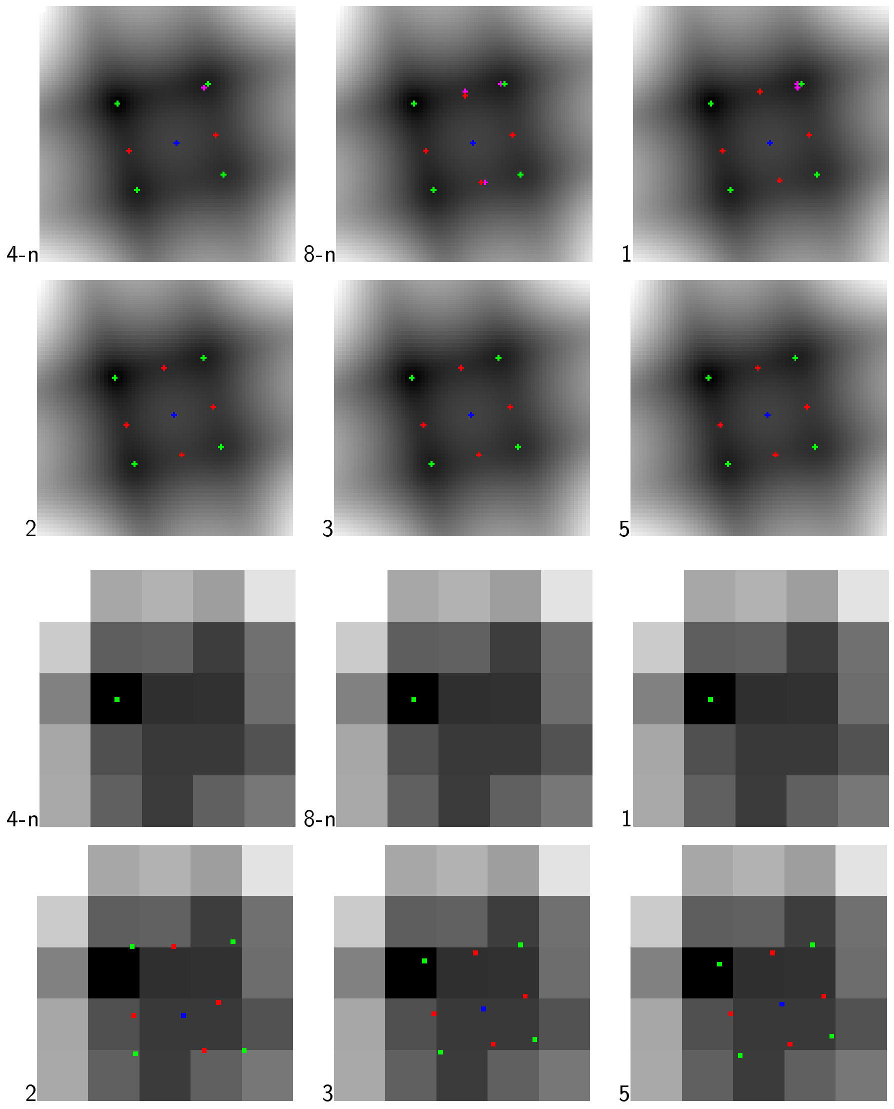


Figure 3.17: Critical point detection at $h = \frac{1}{32}$ (upper rows) and $h = \frac{1}{2.6}$ (lower rows) ($\alpha = 12°$, $d_1 = 0.3$, $d_2 = 0.5$). Results of 4- and 8-neighborhood methods, spline methods of order 1, 2, 3 and 5 (all overlayed over the nearest-neighbor interpolation). Minima, maxima, saddles, and false positives are marked green, blue, red, and magenta respectively.

Linear interpolation behaves similar to the 8-neighborhood case (it also depends only on the 3×3 neighborhood and applies the exact same rules at minima and maxima), but has fewer false positives at saddles (40% total error rate when $\alpha \neq 0°$). In contrast, higher order splines work perfectly at this resolution, with maximum localization errors of 0.027 pixels for the second-order spline and below 10^{-4} pixels for the order 3 and 5 ones. At this resolution, one starting point per pixel is sufficient for the Newton iterations to find all critical points.

When the resolution is reduced, the performance of the simple methods further deteriorates, whereas the higher-order methods perform still properly at $h = \frac{1}{5}$. Their maximum localization errors are now 0.21, 0.065 and 0.009 pixels for the splines of order 2, 3 and 5 respectively, and in a few cases four iteration starting points per pixel are required in order to find all critical points. At $h = \frac{1}{3}$, the second-order spline misses a maximum in 2 of the 324 test images, whereas the higher orders still find all points, and maximum localization errors are 0.72, 0.32, and 0.17 pixels respectively. The sample distance $h = \frac{1}{2.6}$ is the coarsest resolution where the fifth-order spline reliably detects all critical points, orders 2 and 3 fail in 10% and 5% of the images respectively. The maximum errors are 1.9, 0.82 and 0.26 pixels. The lower rows in figures 3.16 and 3.17 show results for this resolution. The minimum sampling distance where a 5th-order spline represents the function well enough to reliably detect all 9 critical points under all parameter choices is $h = 1/2.6$, which means that on average there are 1.3 critical points per pixel. A similar experiment on real image data is described in section 3.4.3.

Our experiment shows that the reliable detection of extrema and saddle points is not at all trivial, even without any noise. Simple comparison of pixel values in a 4- or 8-neighborhood works surprisingly badly, in striking contrast to popular belief. We would like to point out that this is a fundamental problem that cannot be solved by going to higher resolution: a 3×3 set of pixels is simply to small to decide whether the center is a special point. One must look at larger neighborhoods and use subpixel detectors, as the following experiment illustrates once again. Consider an anisotropic Gaussian blob

$$255\, e^{-\frac{r^2}{2}\left(\frac{\cos^2(\alpha)}{\sigma_1^2}+\frac{\sin^2(\alpha)}{\sigma_2^2}\right)}$$

with $\sigma_1 = 30$ and $\sigma_2 = 2$, rotated by an angle $\alpha = 22.5°$. This function has only a single maximum at the center of the blob. However, when we sample this function, a large number of false critical points are detected along the ridge, as figure 3.18(b,c) shows. In figure 3.19, the pixel values from a subregion of this image are shown. It can be clearly seen how these sampling artifacts arise: Consider a sampling point located near the true ridge of the Gaussian hill (e.g. the central pixel in figure 3.19). The level line through that point is a narrow ellipse, with the point located near the ellipse's tip. Since all "uphill" points are confined to the interior of the ellipse, i.e. to a small sector of the point's neighborhood, it is quite likely that none of the neighboring sampling points falls within this sector. Consequently, there is no higher neighbor in the grid, and the center point is falsely classified as a maximum. Similar arguments hold for minima in narrow valleys and for saddle points. Obviously, this problem cannot be solved by higher sampling rates: No matter how small a sampling step we take, there will always be locations where the "uphill" part of a ridge will be lost between two sampling points, giving rise to a false maximum. In contrast, when we use higher order splines, the correct maximum and only the correct maximum is detected (figure 3.18d).

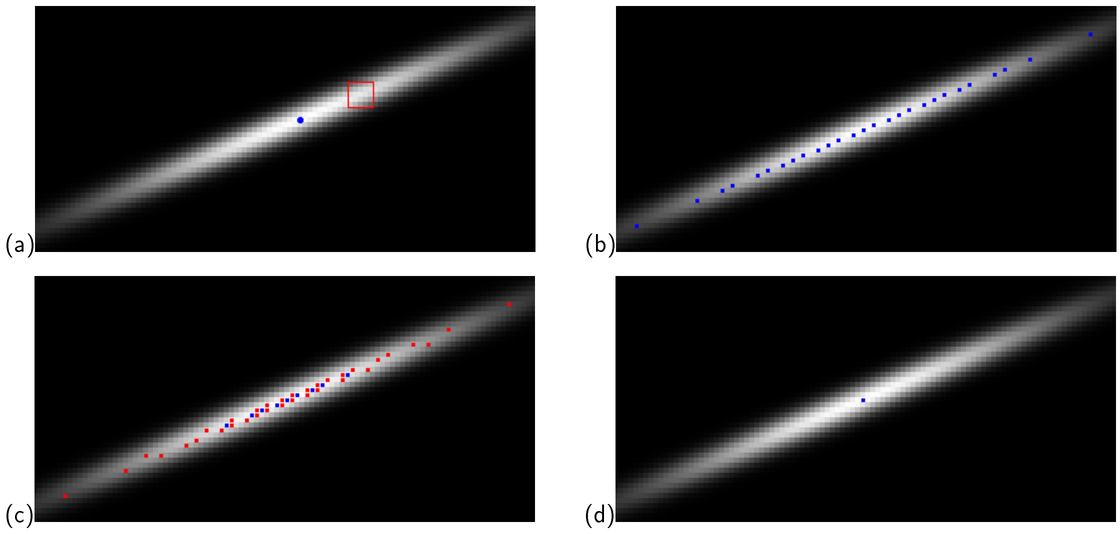
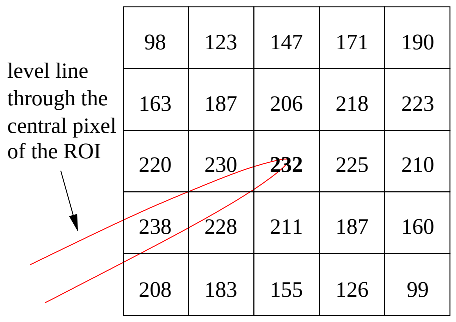


Figure 3.18: (a) Anisotropic Gaussian blob with its true maximum marked. The red square indicates the ROI for figure 3.19; (b) Maximum detection in the 4-neighborhood produces many false positives; (c) In the 8-neighborhood, there are still several false positive maxima (blue) and a large number of false positive saddle points (red); (d) The correct result can be recovered by algorithm 3.1 with splines of order 2 and above.


level line through the central pixel of the ROI

98	123	147	171	190
163	187	206	218	223
220	230	**232**	225	210
238	228	211	187	160
208	183	155	126	99

Figure 3.19: Illustration of false positive detection. The table shows the gray levels in the red region of 3.18(a). The central pixel of this region is a false maximum in either the 4- or 8-neighborhood because the uphill point on the ridge is lost between pixels “230” and “228”. On the same reason, pixels “230” and “225” are falsely marked as saddles by the 8-neighborhood method.

3.4 Noise Normalization and Noise Filtering

According to the image acquisition model, noise is added to the image during or after sampling. We can therefore consider the noise spectrum to be band limited. It can be shown (see e.g. [Goudail & Réfrégier 04]) that linear filter techniques are statistically optimal when noise is additive and Gaussian distributed with constant mean and variance throughout the image plane. However, noise in real imaging devices is not in general Gaussian distributed. We will analyze the noise and describe ways to transform it into Gaussian white noise.

3.4.1 Noise in CCD Cameras

There are three main sources of noise in a CCD camera: photon counting noise, quantization noise, and electronic noise [Healey & Kondepudy 94, Jähne 02, Förstner 99]. If the measured intensity is proportional to the number of photons detected by the sensor, photon counting noise is Poisson distributed:

$$p(\tilde{I} \,|\, I) = \frac{e^{-I} I^{\tilde{I}}}{\tilde{I}!}$$

where $\tilde{I} \geq 0$ and $I \geq 0$ are the measured and true intensities respectively. Mean and variance of this distribution are equal to I. When I is not very small (i.e. $I > 20$), the Poisson distribution can be well approximated by a Gaussian with mean and variance equal to I. Therefore, photon counting noise can be realistically modeled by a Gaussian whose variance increases linearly with intensity I:

$$\sigma_P^2(I) = a\,I + b \tag{3.18}$$

The parameters a and b account for the possibility that the actually reported gray-levels are linear functions of the originally measured intensities.

Quantization noise models the round-off error when real-valued intensities are transformed into integers. The round-off error is uniformly distributed between $-\frac{1}{2}$ and $\frac{1}{2}$. Hence, its variance is

$$\sigma_Q^2 = \int_{-\frac{1}{2}}^{\frac{1}{2}} x^2\,dx = \frac{1}{12}$$

If gray levels are represented with 8 bits per pixel or more, this is generally much less than the contribution of other noise sources (otherwise, camera capabilities would be wasted by too coarse quantization). Finally, electronic noise depends on the details of the sensor and subsequent processing and is not easily modeled without intimate knowledge of the sensor. It is relatively small in most modern sensors. For example, the noise standard deviation of a uniform black image in the Canon EOS D60 is 0.2% of the maximum white intensity (see below), i.e. in the same order as the round-off error. We further assume that noise does not depend on the position of the sensor in the sensor array, and that the noise of neighboring samples is uncorrelated (according to [Förstner 99], the correlation coefficient is less than 50%). Therefore, photon counting noise is the main noise source we have to consider.

Noise depending on the image intensity is problematic because it is incompatible with the assumption of statistical independence between signal and noise. If this assumption is violated, the decision of whether a particular feature measurement represents a scene property or is caused by noise depends on the intensities of all pixels in the operating window and requires non-linear processing techniques. To avoid these complications, [Förstner 99] proposes a noise normalization transform that equalizes the noise variance by means of a non-linear scaling of the image intensities. After the transformation, the assumption of additive Gaussian noise is fulfilled reasonably well. Suppose the dependence of the noise variance on intensity is given by a function $\sigma_N^2 = s(I)$. After transforming intensities according to a function $\hat{I} = t(I)$ to be determined, the noise variance shall be constant $\hat{\sigma}_N^2 = \sigma_0^2$:

$$\sigma_0^2 = \text{const.} = \int \Big(t(\tilde{I}) - t(I)\Big)^2 p(\tilde{I} \,|\, I)\, d\tilde{I} \quad \text{for all } I$$

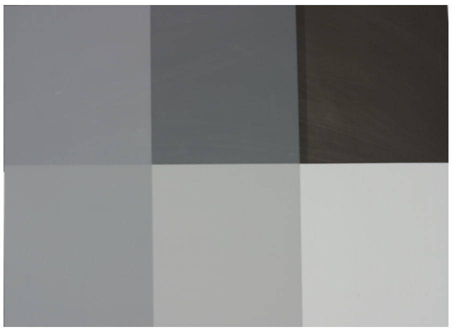


Figure 3.20: Part of the target pattern used to determine how the noise variance depends upon image intensity.

where $p(\tilde{I}\,|\,I)$ is again the probability of measuring intensity $\tilde{I}$ when the true intensity was I. When $t(\tilde{I})$ is expanded into a Taylor series around I, we get $t(\tilde{I}) \approx t(I) + \frac{d}{dI}t(I)(\tilde{I} - I)$. Then, the variance after the transformation can be approximated as

$$\sigma_0^2 \approx \left(\frac{d}{dI}t(I)\right)^2 \int \left(\tilde{I} - I\right)^2 p(\tilde{I}\,|\,I)\,d\tilde{I} = \left(\frac{d}{dI}t(I)\right)^2 s(I)$$

Taking the square root and re-arranging, we find

$$d\,t(I) = \sigma_0 \frac{dI}{\sqrt{s(I)}}$$

with general solution

$$t(I) = \sigma_0 \int_{I_{\min}}^{I} \frac{dI}{\sqrt{s(I)}} + C \tag{3.19}$$

where the constants σ_0 and C can be chosen freely, for example so that the variance becomes unity and the minimal intensity $\hat{I}_{\min} = t\,(I_{\min})$ is zero. In case of the linear relationship (3.18) between variance and intensity (e.g. Poisson noise), the integral can be solved analytically, and we get

$$t(I) = \frac{2\,\sigma_0}{a}\sqrt{a\,I + b} + C \tag{3.20}$$

To determine the noise normalization transform of real cameras, we took images of the gray tone target depicted in figure 3.20. We computed the variance in Gaussian-weighted windows of size $\sigma = 8$ around each pixel, but discarded all measurements where the local gradient at scale 8 was bigger than 0.1% of the contrast between the gray regions. In this way, variance measurements were not contaminated by edge artifacts. Figure 3.21 shows the noise variance as a function of intensity for the two cameras tested. In case of the Canon EOS D60, the intensities are taken from the raw image before white-point normalization and Bayer-grid interpolation. It can be seen that the dependencies are indeed nearly linear – the variance grows only slightly faster (in the Canon camera) or slower (in the industrial camera) than linearly for high intensities. Since we are going to use the green subgrid of the Canon camera as a gray-level image, we need the corresponding noise

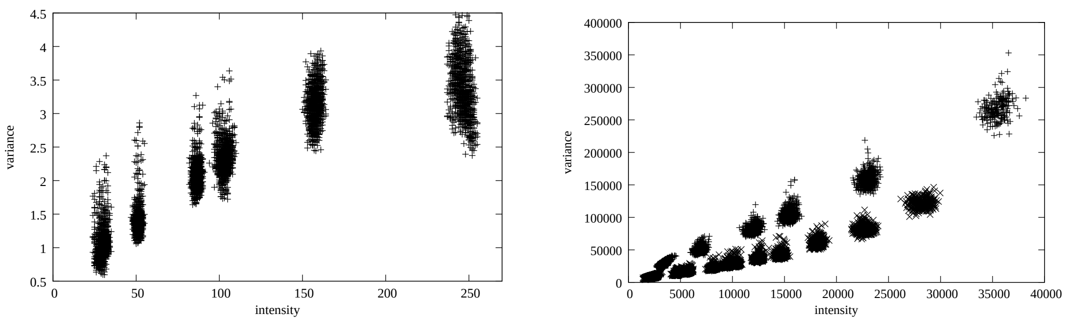


Figure 3.21: Dependence of the noise variance upon the intensity for an industrial camera (left) and the Canon EOS D60 (right). In the right graphs, the upper clusters represent the red channel, the lower ones the green and blue channels. Camera settings were: f-stop 8, chip sensitivity ISO 100.

normalization transform. The noise variance can be expressed as a function of intensity by means of a least squares fit to a parabola:

$$\sigma_N^2(I) = c_2\, I^2 + c_1\, I + c_0 = 10^{-4}\, I^2 + 1.22\, I + 5300$$

The solution to (3.19) for positive c_2 is:

$$\tilde{I} = \frac{1}{\sqrt{c_2}} \left(\log\left(\frac{2c_2 I + c_1}{\sqrt{c_2}} + 2\sqrt{c_2\, I^2 + c_1\, I + c_0} \right) - \log\left(\frac{c_1}{\sqrt{c_2}} + 2\sqrt{c_0} \right) \right) \tag{3.21}$$

The integration constant was chosen so that a zero gray level is mapped onto zero. The maximum intensity is mapped onto a gray level of approximately 190. Although the formula is somewhat complex, this is not a problem in practice because I is quantized, and we can use a look-up table to speed up computations. Figure 3.22 shows this transform. It turns out that it is very similar to the standard gamma correction $\tilde{I} = \left(\frac{I}{I_{\max}}\right)^{0.4545}$ that is routinely applied in many cameras and scanners to transform RGB data into the perceptually more uniform R'G'B' or sRGB color spaces [Poynton 96]. A similar nonlinearity is believed to be the first step in human vision [Graham & Hood 92, Georgeson & Freeman 96]. Noise normalization and perceptual uniformity are achieved by essentially the same transformation. It can be speculated whether this is a coincidence.

When we want to analyze full color data from the Canon camera, we must first interpolate the Bayer pattern to a full color image, and the color channels must be normalized for correct white balance. In principle, we could do this on our own, which would have the advantage that all intermediate steps were known and could be analyzed. But it is difficult to achieve good results without detailed knowledge of the properties of the various camera components. Since these data are not publicly available, we prefer to rely on Canon's proprietary conversion algorithm. Therefore, we repeated the noise analysis after conversion to full RGB. The results are shown in figure 3.23 left. It can be seen that Canon's algorithm performs too much gamma correction: The noise variance does now

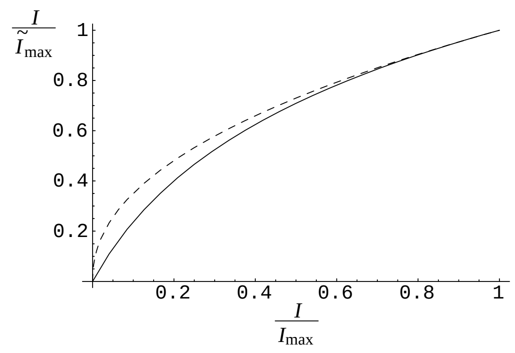
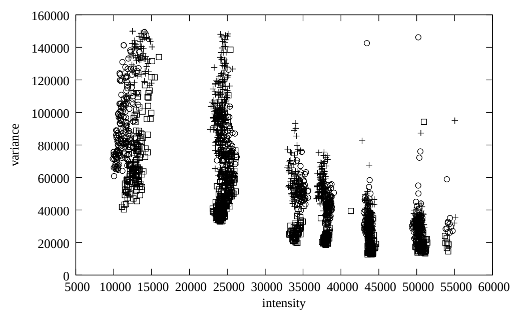


Figure 3.22: Comparison of the noise normalization transform according to (3.21) (solid line) with the standard gamma correction with $\gamma = 0.4545$ (dotted). The axes depict normalized intensity, i.e. $\frac{\tilde I}{\tilde I_{\max}}$ vs. $\frac{I}{I_{\max}}$. It should be noted that the two transforms are almost equal for high intensities, but gamma correction brightens low intensities too much.



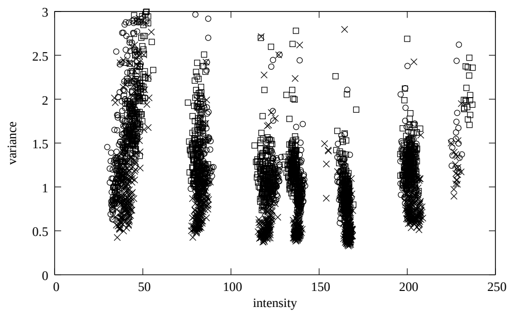


Figure 3.23: Dependence of the noise variance upon the intensity for the Canon EOS D60 after Bayer filter interpolation and white balance adjustment (left). The same after noise normalization (right). Squares: red channel, crosses: green channel, circles: blue channel (hard to distinguish because the noise distributions are very similar in all channels).

decrease with intensity. This observation is in line with the steeper ascent of the gamma curve at low intensities in figure 3.22. Noise in the red channel is still somewhat higher than in the other channels, but the difference is not as big as in the raw data. A linear least-squares fit to the data is

$$\sigma_N^2(I) = -1.638\,I + 107347$$

The corresponding noise normalization transform is

$$\tilde I = -\frac{\sqrt{-1.638\,I + 107347}}{0.819} + 390$$

where the constant is chosen so that the zero intensity is preserved. Note that the same transformation is applied to all color channels, because white balance would be destroyed otherwise which may or may not be a problem, depending on the application. Figure 3.23 right shows the result of this transformation. Except at low intensities, the average noise variance is indeed unity. Higher values at low intensity are mostly in the red channel.

In practice, it is not always possible to calibrate the camera noise with a calibration target such as figure 3.20. In that case, the calibration must be performed with a real image, where we don't know the homogeneous regions beforehand. Therefore, a *robust noise estimation procedure* must be employed, which excludes regions with significant variation of the noise-free image information. Furthermore, it is not always possible to access the raw image data. After a number of unknown pre-processing steps, e.g. in the camera, the noise variance is often no longer described by a simple function like a linear or quadratic one. This calls for a *non-parametric* estimation method. A method that meets both requirements was implemented by Förstner et al. and partly described in [Förstner 99]. We give a complete description of the algorithm by combining information from the paper and the original implementation.

First, suppose that the image consists of a single region with uniform gray-level and additive Gaussian noise with variance σ_N^2. Then the response of the symmetric difference filter $(0.5, 0, -0.5)$ to this image is a Gaussian random variable with zero mean and variance $\sigma_N^2/2$. The gradient squared magnitude $g^2 = g_x^2 + g_y^2$ resulting from applying the symmetric difference in $x-$ and $y-$direction is the sum of two squares of Gaussian random variables, hence follows a χ^2-law with 2 degrees of freedom. Its distribution is an exponential

$$p(g^2) = \frac{1}{\mu} e^{-g^2/\mu}$$

with parameter $\mu = \sigma_N^2$. Consequently, we can determine the variance of the original image values by estimating the mean of the gradient squared magnitude. Now suppose that the image contains some outliers. They will give raise to gradient magnitudes that are much higher than the expected value μ of the exponential law. A robust estimate of μ should exclude these outliers. When we already have a good guess μ_0 for μ, we can obtain an improved estimate from just the lower part of the distribution up to $\lambda^2\mu_0$ $(0 < \lambda < \infty)$, which is supposedly not contaminated by outliers:

$$\tilde{\mu}_0 = \frac{1 - e^{-\lambda^2}}{1 - (1 + \lambda^2)e^{-\lambda^2}} \frac{\int_{h=0}^{\lambda^2\mu_0} h\, p(h)\, dh}{\int_{h=0}^{\lambda^2\mu_0} p(h)\, dh} \tag{3.22}$$

where $p(h) = p(g^2)$ is the histogram of gradient squared magnitudes, and the factor $\frac{1-e^{-\lambda^2}}{1-(1+\lambda^2)e^{-\lambda^2}}$ adjusts the estimate for the fact that only part of the histogram was used to compute the mean. When this procedure is iterated (starting with a fixed μ_0 whose choice is uncritical), it quickly converges to a robust estimate of $\mu = \sigma_N^2$.

Finally, consider a real image. We require it to contain several homogeneous regions at different gray-levels in order for the noise estimation to be possible. The homogeneous regions are separated by edges and possibly textures, which will be considered as outliers with respect to the noise estimation model. The following algorithm was reverse-engineered from Förstner's original implementation:

Algorithm 3.2: Non-parametric noise normalization

Input: An image f with the following two properties: (i) The conditional probabilities of observed intensities, given the true intensity, are spatially uncorrelated Gaussian random variables with intensity-dependent variance, and (ii) there exist sufficiently many homogeneous regions for estimating true intensities and variances within circular windows of radius r. This radius must be large enough to yield accurate estimates, but small enough for condition (ii) to hold. Förstner suggests $r = 6$ for standard images.

1. Compute the squared gradient magnitude g^2 of f by means of the symmetric difference filter.
2. Mask inhomogeneous regions:
   a) Mask pixels were g^2 is higher than the maximum expected noise gradient under worst-case assumptions. This threshold is uncritical, and Förstner uses a value of 2000 for standard 256-graylevel images.
   b) Enlarge the masked region by dilation with a disc of radius r.
3. At all unmasked pixels, use a window of radius r to estimate the average grayvalue and the noise variance according to the iterative procedure described above (equation (3.22)).
4. Sort the estimated [grayvalue, variance] pairs by increasing grayvalue and group the pairs into n clusters of equal size (n is again uncritical, Förstner uses $n = 20$). Compute a cluster representative by assigning the average noise variance within the cluster to the cluster's midpoint grayvalue.
5. Linearly interpolate between the cluster representatives to obtain the function $\sigma_N^2(I)$ (use constant extrapolation outside the range of the cluster centers). Compute the transformation $t(I)$ by numeric integration of (3.19) according to

$$t(I) = t(I-1) + \frac{1}{\sqrt{\sigma_N^2(I)}}$$

In general, any image analysis algorithm that is explicitly or implicitly based on the assumption of additive Gaussian noise (and these are the majority) will profit from noise normalization. Noise normalization is usually much cheaper than the development of a new analysis algorithm adapted to a specific kind of noise. For example, consider edge detection and let edge strength be estimated by means of Gaussian derivative filters. When the noise is not normalized, the strength of spurious edges caused by noise will depend on the image intensity, so that different thresholds are required in different parts of the image. In contrast, a uniform threshold on the gradient magnitude is sufficient after noise normalization, and it can be derived analytically. Using Parseval's identity, the expected value of the square of a 2-dimensional Gaussian first derivative along the x-direction, applied to a Gaussian white noise image with variance σ_N^2, can be computed as

$$E\left[g_x^2\right] = \frac{\sigma_N^2}{4\pi^2} \iint \left(2\pi\, x\, e^{-2\pi^2(\nu_1^2+\nu_2^2)s^2}\right)^2 d\nu_1\, d\nu_2 = \frac{\sigma_N^2}{8\pi s^4}$$

where s is the standard deviation of the Gaussian filter. The gradient squared magnitude $g^2 = g_x^2 + g_y^2$ is thus exponentially distributed with parameter $\mu = \frac{\sigma_N^2}{4\pi s^4}$. Since we apply the filter to an already noise-normalized image, we have $\sigma_N^2 = 1$. Hence, we can compute the gradient magnitude above which the probability that the observed gradient was caused by noise becomes less than ϵ:

$$|g_{s,\epsilon}| \geq \sqrt{\frac{-\ln \epsilon}{4\pi\, s^4}}$$

For example, when we set a threshold of $0.5/s^2$, the probability of false positives is 5%, at $0.6/s^2$ it drops to 1%, and at $0.75/s^2$, we already achieve 0.1%. Similar thresholds can be derived for a color gradient, when the noise in each color channel is independent and has been normalized. If the color gradient is computed by adding the squared gradients of the three colors channels, a χ^2-distribution with 6 degrees of freedom (instead of 2) must be used: $p(g^2) = \frac{27\,g^4}{3\,\mu'^3} e^{-3g^2/\mu'}$ with mean $\mu' = 3\mu = \frac{3\sigma_N^2}{4\pi s^4}$. Then the gradient magnitudes for 5%, 1% and 0.1% probability of false positives are $0.72/s^2$, $0.81/s^2$, and $0.95/s^2$ respectively. A more detailed analysis of optimal threshold estimation will be performed in section 7.2.2.2.

In most images, the effect of noise normalization is not very spectacular. This is easily understood when we look at it as a two-step process: First, noise normalization performs a transformation similar to γ-correction which equalizes the noise variance over the entire image by a non-linear brightness increase. Second, the image intensity is divided by the standard deviation of the resulting noise distribution in order to achieve *unit noise variance.* Since the variance before intensity scaling is usually above unity, noise normalization has the effect of reducing the edge contrast in terms of absolute intensity (usually to values between 1/4 and 3/4 of the contrast in the original image). However, this transformation does not deteriorate the signal-to-noise ratio, as long as image intensities are represented with floating point accuracy. Figure 3.24 shows the result of noise normalization in two example images.

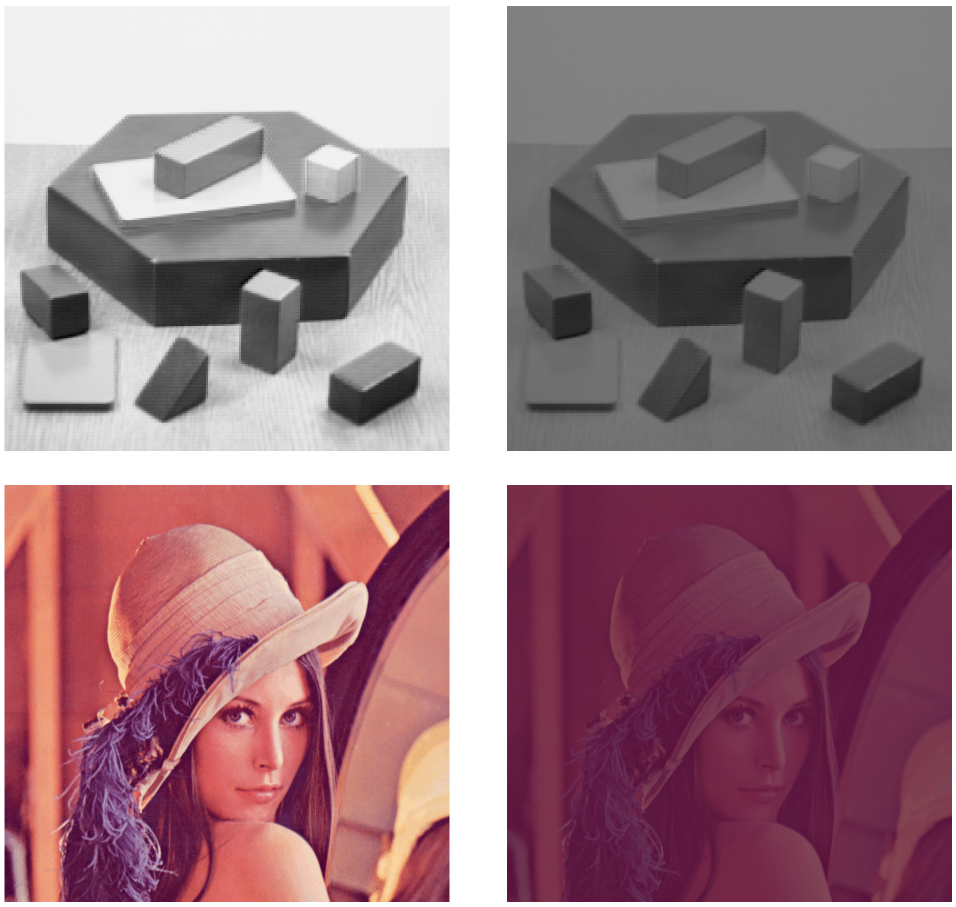


Figure 3.24: Non-parametric noise normalization according to algorithm 3.2 for two example images. The normalized images look worse than the originals, but this is just a consequence of the fact that these images have been optimized for image analysis, not printing.

3.4.2 Speckle Noise

The non-parametric noise normalization algorithm described above requires the conditional probabilities to be Gaussian distributed. This requirement is not always fulfilled. An important exception are images suffering from *speckle noise*. Speckle noise is formed by self-interference of the reflected waves under coherent illumination, for example in radar, laser, and ultrasound imaging. Our presentation follows the analysis of [Michailovich & Tannenbaum 06] who reviewed the properties of speckle noise and proposed a suitable noise normalization transform. Speckle noise has three important properties: (i) it is non-Gaussian, (ii) it is combined with the signal by multiplication, not addition[7], and (iii) it is spatially correlated because it is subjected to smoothing by the system PSF. The precise form of the noise distribution depends on the imaging

[7]There is also an additive noise component, but it is usually much smaller than the speckle and can be neglected.

modality, but it can usually be described by a *generalized gamma distribution*

$$p(z) = \frac{\gamma\, z^{\gamma\nu-1}}{\alpha^{\gamma\nu}\Gamma(\nu)} \exp\left(-\left(\frac{z}{\alpha}\right)^{\gamma}\right) \qquad \text{with } z \geq 0 \text{ and } \alpha, \gamma, \nu > 0$$

where $\Gamma(\nu)$ is the Gamma function. By selecting certain values for the parameters γ and ν, a number of important distributions arise as special cases, for example Rayleigh ($\gamma = 2$, $\nu = 1$), exponential ($\gamma = 1$, $\nu = 1$), Weibull ($\nu = 1$) and log-normal ($\nu \to \infty$).

Multiplicative noise can be turned into additive noise by taking the logarithm of the variable: $\ln(s\,z) = \ln(s) + \ln(z)$, where s is the noise-free signal. When z follows a generalized gamma distribution, the distribution of its logarithm is

$$p\left(\ln(z)\right) = \frac{\gamma}{\Gamma(\nu)} \exp\left[\gamma\nu\left(\ln(z) - \ln(\alpha)\right) - \exp\left[\gamma\left(\ln(z) - \ln(\alpha)\right)\right]\right]$$

For a wide range of parameters, this distribution is similar to a Gaussian distribution, but has longer tails, so that $\ln(z)$ will contain outliers. [Michailovich & Tannenbaum 06] derived the following noise normalization algorithm from their analysis of the speckle problem:

Algorithm 3.3: Normalization of speckle noise

Input: An image with multiplicative, spatially correlated speckle noise.

1. Decorrelate the noise by a deconvolution filter defined in the Fourier domain as

$$H(\nu_1, \nu_2) = \frac{1}{\sqrt{|P_{\text{MTF}}(\nu_1, \nu_2)|^2 + \frac{\sigma_N^2}{\sigma_S^2}}}$$

   where P_{MTF} is the magnitude transfer function of the sensor, and σ_N^2/σ_S^2 is the ratio of the noise and signal variances. [Michailovich & Tannenbaum 06] recommend to treat this quotient simply as a tuning parameter controlling how much decorrelation is applied to frequencies that have been highly damped by the action of the MTF.

2. Take the logarithm of the decorrelated signal to turn multiplicative into additive noise.

3. Perform outlier shrinkage: Apply a 3×3 or 5×5 median filter to the logarithmic image. Whenever the absolute difference between the unfiltered image value and the median exceeds a threshold λ, replace the value with (median $\pm\, \lambda$), where the sign depends on whether the unfiltered value was larger or smaller than the median, otherwise retain the unfiltered value. The threshold λ should be chosen so that $5 - 7\%$ of the pixels are classified as outliers.

The positive effect of this noise normalization on the results of subsequent wavelet denoising, total variation filters, and anisotropic diffusion is convincingly demonstrated by [Michailovich & Tannenbaum 06], see figure 3.25. Yet another noise normalization algorithm suitable for phase contrast imaging was reported by [Goudail & Réfrégier 04] with equally convincing results.

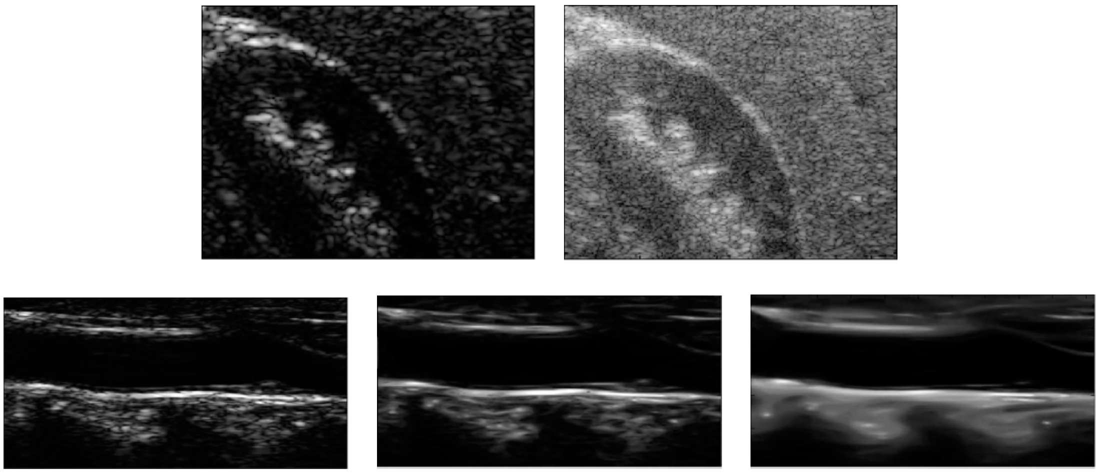


Figure 3.25: Top row: ultrasound image before and after noise normalization. Bottom row: original image (left), anisotropic diffusion without noise normalization (center), anisotropic diffusion after noise normalization (right). Images from [Michailovich & Tannenbaum 06].

3.4.3 Experiment: Critical Point Detection in Real Images

After noise normalization the question arises how much noise reduction we need in order to be able to reliably detect image features. To get an experimental intuition, we repeat the experiment on critical point detection (section 3.3.2) with a real camera. We displayed the picture computed from the logarithm of equation (3.16) on a LCD screen, see figure 3.26 left (taking the logarithm was necessary because the dynamic range of the target would otherwise exceed the capabilities of the display and camera). This picture was then photographed with the Canon EOS D60 with f-stop $f_{\#} = 8$ and focal length 28 mm, figure 3.26 right. The viewing distance was adjusted so that a 10×10 square of pixels on the screen was imaged onto a single camera pixel, so that the sampling of the target pattern was invisible to the camera. The fact that the camera does not produce band-limited images was no problem in this experiment because the target image itself is effectively band-limited.

The camera image was then analyzed in the interpolated RGB image as well as in the red, green, and blue subgrids of the Bayer raster: The images were first smoothed by a Gaussian filter, and critical points were then detected in a spline reconstruction of the smoothed image by means of algorithm 3.1 explained in section 3.3.2. It turned out that we were able to detect exactly the desired critical points in all four cases after smoothing with a Gaussian with standard deviation $\sigma = 1$ and spline interpolation of any order between 2 and 5. Figure 3.27 shows the detected critical points for fifth order spline interpolation. The red and blue subgrids represent the lowest resolution where we were able to reliably detect all critical points. It should be noted that the asymmetric placement of the central maximum was correctly recovered. In the red and blue subgrids, the average distance between the four central minima was 3.95 ± 0.15 pixel, the average distance between the four critical points closest to each other (the

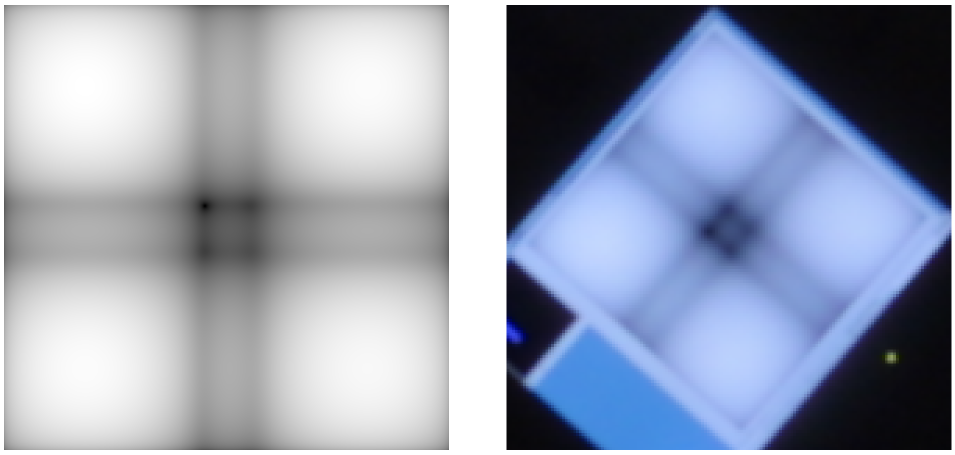


Figure 3.26: Left: The target pattern for the experiment. It has a size of 895×895, and the distance between the four central minima is 90 pixels. Right: Camera image of the target (Bayer filter image interpolated to full resolution color image). The camera was rotated by $45°$ to align the target with the green subgrid. The depicted ROI has size 160×160, the distance between the four central minima is ≈ 8 pixels.

central maximum and the adjacent saddles and minimum) was 1.42 ± 0.02 pixel. The ratio between the two lengths is 2.78, whereas the theoretically correct value would have been 2.25. Apparently, the various blurring filters in the processing chain introduce a bias that amplifies the asymmetry of the central maximum. But we still regard this as a remarkable result, especially considering that no false positives or negatives occurred in critical point detection. When the resolution was reduced by half an octave, the minimum with lowest contrast was lost. Thus, the red and blue subgrid results in figure 3.27 represent approximately the lowest resolution where correct detection was possible. They correspond to a 5.6-fold oversampling relative to the theoretical limit (where the two distances would have been 0.7 and 0.31 respectively). This should be compared with the 2.6-fold oversampling required in the simulations (section 3.3.2) where no blurring and no noise degraded the image quality.

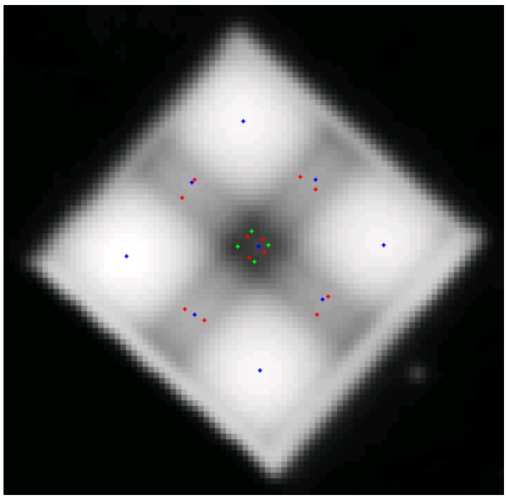
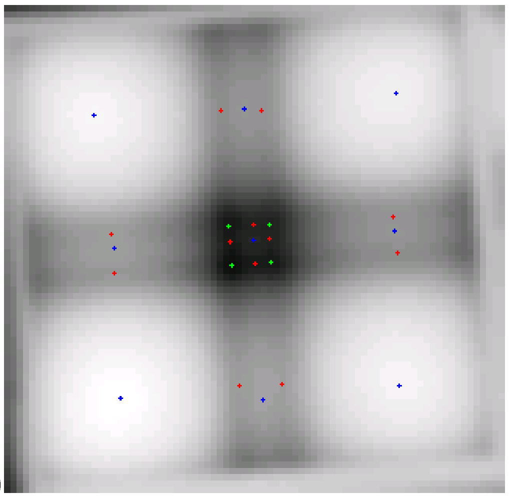
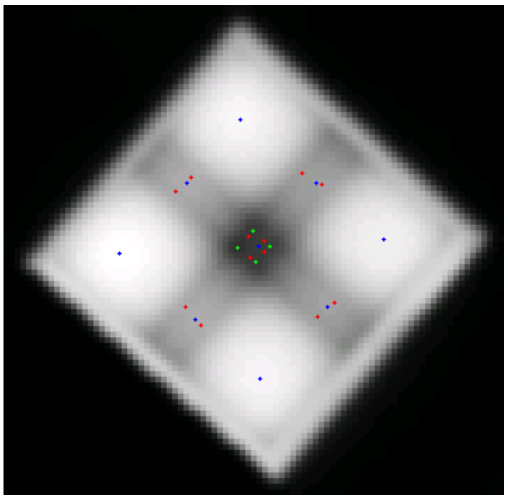
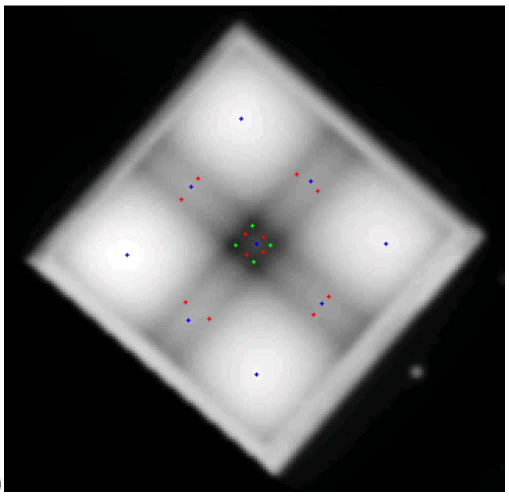
(1) 
(2) 
(3) 
(4) 

Figure 3.27: Left to right: detected critical points (green: minima, red: saddles, blue: maxima) in the red (1), green (2), and blue (3) subgrids of the Bayer raster, and in the the interpolated full color image (4). The second image is rotated because the green subraster of the Bayer is oriented diagonally. The resolution of the green subgrid is half an octave higher than the resolution of the red and blue subgrids, and the resolution of the interpolated image is a full octave higher.

4 The Representation of Segmentation Results

Abstract

No image segmentation method can be more powerful than what is supported by the underlying data representation. In this chapter, we analyze the requirements posed on this data representation by typical segmentation algorithms on the one hand, and the requirements posed by the concepts of topology on the other hand. Conformance to the requirements of topology is of fundamental importance for the definition of basic shape properties like connectedness and neighborhood. We introduce the GeoMap as an efficient and easy-to-use representation that fulfills all requirements and can be implemented in many interesting ways. We discuss GeoMap implementation by polygonal and by grid-based data structures and demonstrate some basic GeoMap applications.

4.1 Topology for Segmentation

Single pixels don't tell much about the image content[1]. It is necessary to consider sets of pixels and their relations. Some of the most fundamental relations are topological in nature, for example *connectedness*, *neighborhood*, and *boundary*. These relations are implicitly defined by the geometric plane partitions we discussed in chapter 2 (definition 2.2), but there are a number of good reasons to make topology explicit:

- Valuable cues for the interpretation of image structure can be derived from topological information, for example whether a region has holes, which arcs its boundary is composed of, or which other regions are neighbors of it. Therefore, topological information should be readily available to algorithms.
- Topological criteria are important for the evaluation and comparison of segmentation algorithms, as became apparent in section 2.2. Many statements about geometric shape similarity include implicit statements about topology. If made explicit, these topological properties serve as powerful shape comparison criteria in their own right.
- The combination of explicit geometric and topological information leads to very powerful data representations (so called *GeoMaps*, see section 4.2) which facilitate the unification and integration of hitherto incompatible algorithms, such as edge- and region-based segmentation methods.

[1]Multi-spectral images with dozens or hundreds of channels per pixel are notable exceptions.

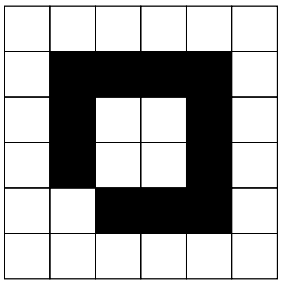


Figure 4.1: Illustration of the connectivity paradox (see text for explanations).

- By taking advantage of topological information, these data representations provide simple operations for orderly (consistency preserving) manipulation of a segmentation, e.g. for removing undesired edges from an over-segmentation.
- Dealing with topology explicitly prevents us from considering topological errors as minor problems – gaps in edges (as are common in e.g. Canny's algorithm) are no longer something that can be ignored or handled by some simple heuristics.

The importance of topology for image analysis was first recognized by [Rosenfeld 70] who analyzed the so-called *connectivity paradox*: When the 8-neighborhood is used for both the foreground and the background, there is no guarantee that a simple closed curve splits the pixel plane into an interior and an exterior part. Similarly, if 4-neighborhood is used, the plane can be split by a curve that is not closed, see figure 4.1. [Rosenfeld 70] pointed out that these problems can be avoided on the square raster when different neighborhoods are used for the foreground and background (i.e. 4-neighborhood for the foreground and 8-neighborhood for the background, or vice versa). It is also well-known that the hexagonal grid does not suffer from these problems because only three pixels meet at every pixel corner.

The first analysis of topological issues independent of a particular grid layout was conducted by [Kovalevsky 89] who introduced the notion of *cell complexes* into the field of image analysis. Later it became apparent that the concept of *combinatorial maps* provides even more powerful tools for the representation and analysis of image topology. Combinatorial maps were originally introduced by [Tutte 84] and have been well-established for a long time in computer graphics and geometric modeling [Mäntylä 88, Kettner 98, Duford & Puitg 00]. In the context of irregular image pyramids, [Kropatsch 95] was the first to introduce dual-graph representations which later turned out to be equivalent to combinatorial maps [Brun & Kropatsch 01], but are significantly more complicated. Several other authors were lead to combinatorial maps as a natural representation for interpixel boundaries (cf. section 4.3.2), e.g. [Braquelaire & Domenger 99, Winter 95]. Interestingly, the interpixel boundary representation was already discovered by [Brice & Fennema 70], but was probably not practical at the time due to its large memory requirements. Additional theoretical justification of the interpixel approach is given by [Khalimsky et al. 90] who introduced a topology for a square raster that treats points with even and odd coordinates differently, and [Ahronovitz et al. 95] who defined finite topological spaces by regular subdivisions of $\mathbb{R}^2$ into convex polytopes.

A good review of the basic ideas of combinatorial maps and their realization by means of interpixel boundaries can be found in [Braquelaire 05]. [Köthe 03b] demonstrated that combinatorial maps can also be constructed from thin 8-connected edge images, which, for example, result from Canny's algorithm and some variants of the watershed transform. Later, the concept of topological data representations was generalized to include plane partitions whose boundaries do not form a single connected set, e.g. the *topological graph of frontiers* [Fiorio 96] and the *border map* [Bertrand et al. 99]. This is important in practice because regions with holes and inclusions are very common, for example in pictures showing a wall with windows, or a biological cell with nucleus and organelles. An integrated treatment of many different approaches in the unified framework of *XPMaps* was given in [Köthe 02]. In the sequel, we will review and further develop these ideas. In particular we will show how geometric information can be combined with topological information in both pixel-based and polygon-based representations, and how these representations can be derived from image data.

The mathematical theory of *topological spaces* describes the minimal requirements that must be met in order to allow topological relations to be established. These requirements are formulated in the following axioms:

Definition 4.1. *A tuple (E, O) consisting of a set E of abstract entities and a set O of subsets of E (called the* open subsets*) is called a* topological space *if the following axioms are fulfilled:*

(1) The empty set and E are both open: $\emptyset \in O$, $E \in O$.

(2) The union of arbitrary many open sets is an open set: $o_1 \ldots, o_k \in O \Rightarrow \bigcup_k o_k \in O$.

(3) The intersection of two open sets is an open set: $o_1, o_2 \in O \Rightarrow o_1 \cap o_2 \in O$.

If E is a continuous space, the elements e of E are called *points*. In finite spaces they are often called *cells*. Topological relations can now be defined solely on the basis of the axioms: The *boundary* ∂S of a subset S of E is the set of all points (or cells) such that every open set containing one of these points (cells) intersects both S and its complement $S^c = E \backslash S$. The *closure* of a set is the union of the set and its boundary: $\bar{S} = S \cup \partial S$. The *interior* of a set is the difference between the set and its boundary: $S^0 = S \backslash \partial S$. Two non-intersecting sets are *neighbors* when their boundaries intersect. A set is *connected* if it cannot be partitioned into two subsets that are not neighbors of each other. A set S is *simply connected* if both S and S^c are connected.

To simplify working with a topological space, it is useful to define a *topological basis*. A basis is a set of open sets such that all open subsets of E can be written as unions and finite intersections of sets from the basis. In case of the n-dimensional Euclidean space $\mathbb{R}^n$, the most common basis is the set of open balls $B_\epsilon(\vec{x}) = \{\vec{y} \in \mathbb{R}^n : |\vec{x} - \vec{y}| < \epsilon\}$ with all possible centers $\vec{x} \in \mathbb{R}^n$ and all possible radii $\epsilon > 0$. In a finite topological space, the most natural basis is formed by the set of *open stars*, where the open star of a cell is defined as the smallest open set containing the cell (see definition 4.2 below).

If the axioms of a topological space are not fulfilled, it is impossible to establish consistent definitions for the relations mentioned. For example, when naive neighborhood definitions are used, it is impossible to come up with a satisfactory definition for the

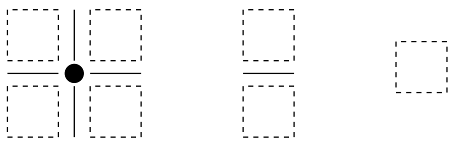


Figure 4.2: The open stars of a vertex, arc, and region (from left to right) in a square tessellation of the plane. If the infinite region is contained in an open star, the pictures must be modified in obvious ways.

boundary of a region [Pavlidis 82]. Another well-known problem is the connectivity paradox we discussed above. Obviously, inconsistent intermediate data will cause errors and contradictions in subsequent processing steps. They may result in artifacts whose detection and correction during high-level processing is very difficult. Therefore, it is preferable to solve these problems at their root, and image segmentation methods conforming to the requirements of topology make an important contribution towards this goal.

It is easy to prove that a plane partition according to definition 2.2 does indeed fulfill the topological axioms when we consider the vertices, arcs, and regions as three kinds of cells with dimension $z = 0, 1, 2$ respectively. Then the Euclidean topology of the plane induces a neighborhood relation between the cells, which is used to define the smallest open neighborhood (the *open star*) of each cell:

Definition 4.2. *Let T be a partition of the Euclidean plane into cells (regions, arcs, vertices) according to definition 2.2. Then the* open star *(smallest open neighborhood) of a cell z is defined as the set of cells z' that intersect with every Euclidean neighborhood of the points in z:*

$$\textit{open-star}(z \in T) = \{z' \in T : \; \forall \vec{x} \in z, \forall \epsilon > 0 : B_\epsilon(\vec{x}) \cap z' \neq \emptyset\}$$

where $B_\epsilon(\vec{x})$ denotes the open ball of radius ϵ around the point $\vec{x}$.

The cell z itself is always a member of its open star. All other members have a higher dimension than z. The set of open stars forms a basis for the finite topological space defined by the cells of the plane partition, i.e. one can create all possible open sets as unions of the open stars. The interpretation of plane partitions in terms of open stars and finite topological spaces suggests an alternative definition of topological equivalence:

Definition 4.3. *Two partitions of the plane are topologically equivalent if there exists a 1-to-1 mapping between the cells of the division such that the open stars are preserved.*

It can be shown that this definition is equivalent to definition 2.6 (topological equivalence due to existence of a homeomorphism), but it is easier to work with because it is based on a finite set of cells instead of an infinite set of points. Some authors, e.g. [Ahronovitz et al. 95], further require that all regions be convex polytopes, i.e. do not contain holes. We will not do this in this work. The simplest example for a star topology is obtained by tessellating the plane into equal squares. Then the open stars of the vertices, arcs, and regions look as depicted in figure 4.2, and one can establish a 1-to-1 relation between the partition's regions and the pixels of an appropriately sized image.

The fact that vertices, arcs, and regions partition the plane induces a very important relationship between their numbers – Euler's equation. In its most common form, Euler's equation requires the boundary (i.e. the union of arcs and vertices) to consist of a single connected component. Then it holds that

$$n - e + f = 2 \tag{4.1}$$

where n, e, f denote the number of vertices, arcs, and regions respectively, see e.g. [Tutte 84]. When the boundary set is not connected, the relationship must be slightly modified in order to take the number of connected boundary components k into account:

$$n - e + f - k = 1 \tag{4.2}$$

Proof. Since the proof is rarely found in books, we include it here. The claim is easy to prove by induction. The trivial "partition" that only consists of the infinite region has $f = 1$ and $n = e = k = 0$, the partition consisting of a single vertex has $n = f = k = 1$, $e = 0$, and a partition with connected boundary has $k = 1$ and $n - e + f = 2$, so the formula is valid for these cases. Suppose we have proved the formula for some k, i.e. $n^{(k)} - e^{(k)} + f^{(k)} - k = 1$. Now we add new vertices and arcs which form a new connected boundary component. For this component alone the relation $n^{(1)} - e^{(1)} + f^{(1)} - 1 = 1$ holds. By adding the two equations we get $n^{(k)} + n^{(1)} - e^{(k)} - e^{(1)} + f^{(k)} + f^{(1)} - k - 1 = 2$. It holds that $n^{(k)} + n^{(1)} = n^{(k+1)}$, $e^{(k)} + e^{(1)} = e^{(k+1)}$, but $f^{(k)} + f^{(1)} = f^{(k+1)} + 1$, because the surrounding region of the newly added component is a region that already existed in the k-component partition and must not be counted twice. Thus, $n^{(k+1)} - e^{(k+1)} + f^{(k+1)} - (k + 1) = 1$ as claimed. $\square$

While a plane partition according to definition 2.2 implicitly contains the topological information necessary to define neighborhoods and boundaries, this information is not explicitly accessible in a data structure that implements the geometric definition: We cannot answer questions like "Which arcs belong to the boundary of a certain region?", or "Does a sequence of arcs enclose a hole in a region?" (to name just two examples) without invoking nontrivial geometric computations. In fact, we cannot even directly identify the regions themselves because they are only defined by a negative property – regions are connected sets of points that are *not* vertices and do *not* belong to arcs.

The first topological data representation that made regions explicit was the *region adjacency graph* (RAG) introduced by [Pavlidis 77]. It contains a node for each region, and two nodes are connected by an edge when the corresponding regions are neighbors. However, it turned out that this representation is too weak for encoding complete topological information, as is demonstrated in figure 4.3 where two quite different images are described by isomorphic region adjacency graphs. As [Kovalevsky 89] pointed out for the first time, it is necessary to consider not only regions, but entities of all dimensions (i.e. regions, arcs, and vertices) together. To this end, he introduced the notion of *cell complexes* into the field of image analysis:

Definition 4.4. *A* cell complex *is a triple* (Z, dim, B) *where* Z *is a set of cells,* dim *is a function that associates a non-negative integer* dimension *to each cell, and* $B \subset Z \times Z$ *is the* bounding relation *that describes which cells bound which other cells. A cell may bound only cells of larger dimension, and the bounding relation must be transitive. If the largest dimension is* k*, we speak of a* k*-complex.*

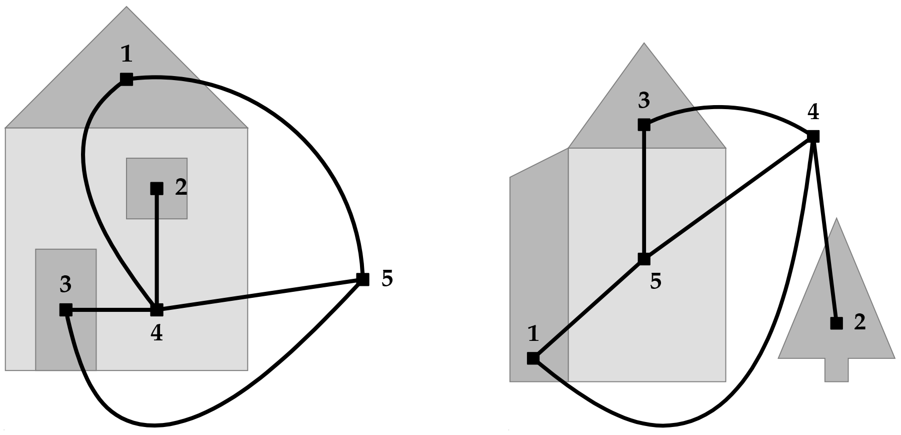


Figure 4.3: Two different images may have the same region adjacency graph (i.e. there is a one-to-one mapping between the nodes of the two graphs so that edges are preserved). Illustration from [Meine 03].

The different feature types have been transformed into abstract cells that are distinguished by their dimensions. A cell complex becomes a topological space by additionally defining the *open sets* as follows: A set of cells is called open if, whenever cell z belongs to the set, all cells bound by z do also belong to the set. Kovalevsky proved that these definitions indeed guarantee that the axioms of a topological space are fulfilled, and that any discrete topological space can be described in terms of a cell complex of appropriate dimension.

In image analysis, we want to represent the topological structure of 2-dimensional images, so we are naturally interested in cell complexes whose largest cell dimension is 2, and whose bounding relation is consistent with a plane partition according to definition 2.2. We call those *planar cell complexes*. Given a plane partition, it is easy to define an associated cell complex: Create a 0-, 1-, and 2-cell for every vertex, arc, and region respectively, and define the bounding relation so that a cell bounds all cells (except itself) belonging to the corresponding open star. This bounding relation is indeed transitive since definition 4.2 guarantees that whenever z_1's open star contains z_2, it also contains all members of z_2's open star.

Unfortunately, we may still loose information by converting a plane partition into a cell complex. That is, plane partitions may have identical cell complex representations even if they are not homeomorphic. Figure 4.4 shows an example. It should be noted that the three plane partitions in this figure differ mainly in the ordering of the arcs around the central vertex. This ordering information is lost in the cell complex. It could be recovered by standard graph drawing algorithms [Di Battista et al. 99] (with a slight simplification because 2-cells are already known and need not be constructed), but the result is not always unique: When the boundary graph is only 1-connected (i.e. would become disconnected if one vertex and all adjacent arcs were removed), several orderings will be consistent with the bounding relation. This is precisely what happens in figure 4.4.

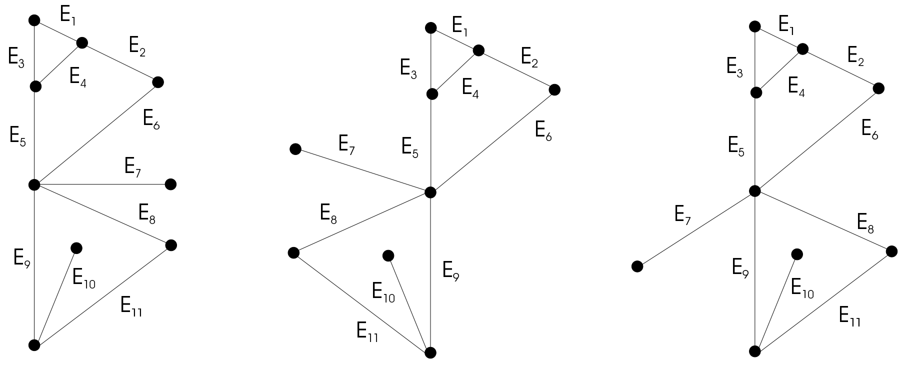


Figure 4.4: Three plane divisions that are not topologically equivalent but have the same cell complex representation.

A unique ordering could be enforced by associating additional geometric information with the cells, but this counters our goal of making topological properties explicit. We have to encode the ordering of the arcs around vertices and regions in an abstract way. This can be achieved by means of *combinatorial maps* [Tutte 84, Duford & Puitg 00]. Combinatorial maps are based on *permutations* of *darts*. A permutation of a set is a bijective mapping of the set onto itself. In effect, a permutation assigns unique successors and predecessors to every member of the set. When the set contains only finitely many elements, we can successively move from an element to its successor until we return to the element where we started, and this will only take a finite number of steps. The set of elements reachable by means of a successor chain starting at element e is called the *orbit* of e in the permutation. Orbits form equivalence classes, because the same set of elements is reachable from every member of an orbit. For example, consider the set $S = \{1, 2, 3, 4, 5\}$ and the permutation $P = \{1 \to 2, 2 \to 5, 5 \to 1, 3 \to 4, 4 \to 3\}$. Its orbits are the sets $S_1 = \{1, 2, 5\}$ and $S_2 = \{3, 4\}$. In a combinatorial map, the permutations are defined over a set of *darts*. Darts are also known as *half-edges* because every edge of a plane partition is represented by a pair of opposite darts:

Definition 4.5. *A* combinatorial map *is a quadruple $(D, \sigma, \alpha, \phi)$ where D is a set of* darts *(or* half-edges*) and σ, α, ϕ are permutations over the darts: Each of them assigns a unique successor $\alpha(d)$, $\sigma(d)$, $\phi(d)$ to every dart. The cycles of these permutations (their orbits) are called* nodes, edges, *and* faces *respectively. The α-orbits must have length two, and the ϕ-permutation must be related to the others by*

$$\phi(d) = \sigma^{-1}(\alpha(d)) \tag{4.3}$$

where $\sigma^{-1}(d)$ is the σ-predecessor of d. Moreover, we require the map to be connected: each dart must be reachable from every other one by a suitable sequence of σ- and α-steps.[2]

[2]Combinatorial maps can also be defined by using four darts per edge ("quad-edges" or "crosses"), e.g. [Tutte 84]. This allows the representation of non-orientable manifolds and the definition of generalized maps (G-maps) which are easy to extend into arbitrary dimensions [Lienhardt 91]. However, these extensions are not needed in 2D image segmentation, so we prefer the simpler and more efficient half-edge definition.

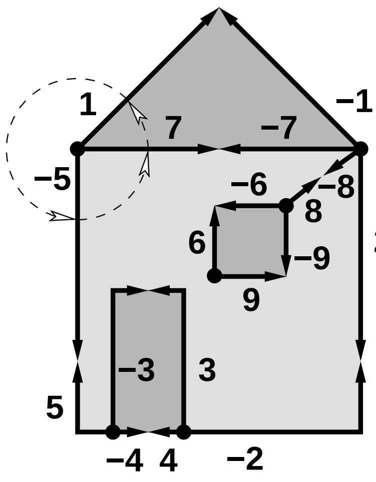


Figure 4.5: The combinatorial map representation of the house example. Darts are indicated by a bold arrow and an associated label. Edges (α-orbits) are formed by pairing darts with labels n and $-n$. The σ-orbit (1,-5,7) is indicated by dashed arrows, the other σ-orbits are (-1,-7,-8,2), (-6,-9,8), (6,9), (-3,5,-4), and (-2,3,4). Illustration adapted from [Meine 03].

The notation $\alpha^*(d)$, $\sigma^*(d)$, $\phi^*(d)$ will be used to indicate the orbits where d is a member of. A combinatorial map is *planar* if the numbers of nodes, edges, and faces in it fulfill Euler's equation (4.1). A planar map can be embedded on a sphere or in the Euclidean plane. In the latter case, it is necessary to designate one face as the *exterior* face, i.e. as the one which contains the point at infinity. Figure 4.5 shows an example.

Every planar combinatorial map has a unique dual map that is obtained by reversing the roles of the σ and ϕ permutations (i.e. σ-orbits become faces, ϕ-orbits nodes). There are two important ways to create planar combinatorial maps. First, one can start with a map that is known to be planar and transform it to the desired map by a sequence of topological operators that preserve planarity. We will discuss these operators in section 4.4.1. Second, one can create a combinatorial map from a plane partition (which is specified geometrically) by turning each arc into two darts and defining the permutations according to the counter-clockwise ordering of the arcs around their vertices. Chapter 5 will be devoted to this approach.

We define open sets by associating a combinatorial map with a cell complex:

Definition 4.6. *The* associated cell complex *of a combinatorial map is defined as follows: Interpret every node, edge, and face of the map as a 0-, 1-, or 2-cell, respectively. Let a cell bound another one of higher dimension whenever the orbits corresponding to the two cells have a dart in common. Call a set of cells* open *if, whenever cell z belongs to the set, all cells bounded by z also belong to the set.*

These definitions imply the following theorem:

Theorem 4.1. *Every combinatorial map with open sets defined according to definition 4.6 is a finite topological space.*

Proof. First, observe that the bounding relation defined by definition 4.6 indeed meets the requirements of definition 4.4. In particular, it is transitive because the underlying relation "orbits have a dart in common" is transitive, as can be seen as follows: Consider an edge orbit, say α_1^*, that contains darts d_1 and d_2, who also belong to face orbits ϕ_1^* and

ϕ_2^* respectively. Suppose further that node orbit σ_1^* contains dart d_1. For transitivity to hold, σ_1^* must share a dart with both ϕ_1^* and ϕ_2^*. This is immediately true for ϕ_1^* because of d_1. In case of ϕ_2^*, consider the ϕ-successor of d_2. Due to the condition $\phi(d) = \sigma^{-1}(\alpha(d))$, dart $d_3 = \phi(d_2)$ is identical to the σ-predecessor of d_1. Thus, d_3 is both in σ_1^* and ϕ_2^*.

Second, observe that the open sets of a combinatorial map are exactly the open sets of the associated cell complex. Therefore, the theorem stating that cell complexes fulfill the axioms of a topological space (cf. [Kovalevsky 89]) implies the same for combinatorial maps. □

It is interesting to note that [Kovalevsky 01a] augments the original cell complex (definition 4.4) with *ordered adjacency lists* that represent the embedding of a cell complex in the plane. The ordering of these lists is not part of the original cell complex definition and is, in fact, equivalent to the permutations and orbits of combinatorial maps. Another alternative was introduced by [Kropatsch 95], who resolves the ambiguities in the region adjacency graph by always considering the RAG together with its dual graph, and perform all manipulations simultaneously in both graphs. This approach is also equivalent to combinatorial maps, but the bookkeeping is much more complicated.

4.2 Combining Topology and Geometry in the GeoMap

Combinatorial maps allow to represent the complete topological structure of a plane partition, but they lack geometric content. Since we need both geometric and topological descriptions in image analysis, [Meine & Köthe 05a] introduced the *GeoMap* as a data representation that combines both. It also deals with a problem that was not addressed in the definition of combinatorial maps: Regions in real images may have holes, i.e. the boundary graph of the partition may be disconnected. We cannot simply solve this problem by dropping the restriction that a combinatorial map be connected: The definition of combinatorial maps does not provide a way to tell which component forms the surrounding part and which one the hole, or which face contains the hole. The standard solution to this problem is the introduction of auxiliary edges which connect the hole with its surroundings, see for example edge $(8, -8)$ in figure 4.5. However, auxiliary edges must be inserted between two specific nodes, and this requirement breaks the natural symmetry of the representation: why should two nodes be singled out by bounding an auxiliary edge that wasn't in the data? Moreover, in pixel-based representations of a GeoMap, it may not even be possible to find a legal place for an auxiliary edge due to the finite grid resolution.

Therefore, we prefer to introduce an additional relation "contains" to encode containment. This decision has an important consequence: when a region has holes, its boundary is not a connected set. Consequently, there is no longer a 1-to-1 correspondence between ϕ-orbits and faces – a face can now be bounded by several ϕ-orbits. In order to handle this, we introduce the notion of *contours*: Each ϕ-orbit now represents a contour, and a face may be bounded by many contours. In this respect, GeoMaps are extensions of the border maps proposed by [Bertrand et al. 99] and the XPMaps introduced by [Köthe 02]. However, these maps encode only the combinatorial (topological) structure

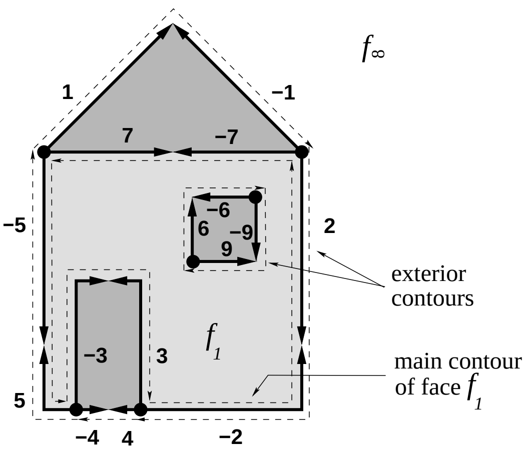


Figure 4.6: Illustration of the GeoMap concept. In contrast to the combinatorial map of the house example (figure 4.5), the auxiliary edge (8,-8) has been removed. Instead, the ϕ-orbit (6,-9) has become the exterior contour of the window and is contained in face f_1. Orbit (9,-6) is the main contour of the region filling the window, and orbit (-2,-7,-5,-3) is the main contour of face f_1. The exterior contour of the house as a whole is orbit (1,2,4,5), which is contained in the infinite face f_∞. Note that the two exterior ϕ-orbits are traversed in clockwise orientation, whereas all other ϕ-orbits are traversed counter-clockwise.

of a plane partition, whereas GeoMaps encode a consistent combination of topological *and* geometric information:

Definition 4.7. *A* GeoMap *is a tuple* $(V, A, F, D, \sigma, \alpha, \phi, \mathsf{exterior}, \mathsf{contains})$*, where*

- V*,* A *and* F *are sets of vertices, arcs, and regions forming a plane partition according to definition 2.2. The infinite face is denoted* f_∞*. For every arc* $a_k \in A$ *define the* inverse arc *as* $a_{-k}(t) = a_k(1-t)$*.*
- D *is a set of darts.* σ*,* α*, and* ϕ *are permutations of the darts whose orbits* σ^**,* α^**, and* ϕ^* *represent the* nodes, edges, *and* contours *of the map respectively. Darts are labeled so that label*$(\alpha(d)) = -$*label*(d) *holds for all darts. For brevity, this will also be written as* $\alpha(d) = -d$*.*
- exterior *is a function* $\phi^* \rightarrow \{true, false\}$*, and* contains *is a binary relation between regions and contours.*

The GeoMap's entities must have the following properties (compare figure 4.6):

1. *Every* α*-orbit* $(d, -d)$ *is uniquely associated with a pair* (a_k, a_{-k}) *such that label*$(d) = k$*.*
2. *There is a 1-to-1 correspondence between the vertices and the* σ*-orbits (nodes)*[3]*. Let* A_j *be the set containing all arcs and inverse arcs whose starting vertex is* v_j *(i.e. for all* $a_k \in A_j$ *we have* $a_k(0) = v_j$*). Then the* σ*-orbit associated with* v_j *contains exactly the darts associated with the members of* A_j*, and the* σ*-permutation orders these darts according to the angle of incidence of the associated arcs at* v_j*, i.e. according to the tangent direction of* a_k *at* $t = 0$*.*
3. *The* ϕ*-permutation is derived from the other two permutations according to* $\phi(d) = \sigma^{-1}(\alpha(d))$*. When the arcs and inverse arcs associated with the darts in contour* ϕ^*_m *are traversed according to their* ϕ*-order, they form a closed curve* Γ_m*.*

[3]Therefore, we will often use the terms "vertex" and "node" as synonyms in the sequel.

4. *The function* exterior(ϕ^*_m) $=$ *false if and only if* ϕ^*_m *has mathematically positive orientation. This condition can be checked as follows:* exterior(ϕ^*_m) $=$ *false holds if and only if the path integral*

$$P_m = \frac{1}{2} \int_{\Gamma_m} x\, dy - y\, dx \tag{4.4}$$

measuring the enclosed area of the curve Γ_m *has positive sign (provided that* x *and* y *are defined in a right-handed coordinate system). When* $P_m < 0$ *(i.e.* exterior(ϕ^*_m) $=$ *true), the contour* ϕ^*_m *is the exterior contour of a map component which forms a hole in some region.*

5. *Every region* f_n *is associated with the set* Φ_n *of all contours that contain a dart whose associated arc is adjacent to region* f_n*. With the exception of* Φ_∞*, the set* Φ_n *contains exactly one contour with* exterior(ϕ^*_n) $=$ *false. This is the outer or* main contour *of* f_n *(i.e. the one enclosing the region). Thus, there is a 1-to-1 relation between regions and non-exterior contours. Furthermore,* Φ_n *may contain an arbitrary number of contours where* exterior(ϕ^*_l) $=$ *true. For every such contour there must be an entry* contains(f_n, ϕ^*_l) *in the contains relation, whereas no such entry must exist for any main contour. The set of contained contours can be found as follows: Evaluate the path integral*[4]

$$W_{n,l} = \int_{\Gamma_n} \frac{-(y - y_l)}{(x - x_l)^2 + (y - y_l)^2} dx + \frac{(x - x_l)}{(x - x_l)^2 + (y - y_l)^2} dy \tag{4.5}$$

where Γ_n *is the curve of the main contour of region* f_n*, and* (x_l, y_l) *is an arbitrary point within* ϕ^*_l*. When* (x_l, y_l) *is in the interior of* Γ_n*,* $W_{n,l} = 2\pi$*, otherwise* $W_{n,l} = 0$*. When* $W_{n,l} = 2\pi$ *holds for several candidate regions* n*, the one with smallest* $P_n > 0$ *must be chosen. If* $W_{n,l} \neq 2$ *for all* n*, although* $P_l < 0$*, the contour is contained in the infinite region, and an entry* contains(f_∞, ϕ^*_l) *must be inserted into the contains relation.*

Conditions (1) to (5) essentially define a general (but not very efficient) algorithm that transforms a given plane partition into a GeoMap by making its topological properties explicit. In practice we can often take advantage of special properties of the plane partition that lead to significant simplifications and speed-ups, especially in the construction of the exterior function and contains relation. Details of these possibilities will be discussed below. By construction, GeoMaps are planar. They fulfill the generalized Euler formula $v - e + f - k = 1$, where v, e, f, and k are the number of vertices, arcs, faces, and exterior ϕ-orbits respectively. The GeoMap is the most general representation for 2-dimensional segmentation problems with finitely many cells, and the geometry and topology of any finite plane partition can be expressed in terms of a GeoMap.

The GeoMap can be implemented as a powerful abstract data type, see [Meine 03, Meine & Köthe 05a] for details. It provides the following types of operations:

[4]The integral follows from Cauchy's integral formula for analytic functions of complex arguments.

- Query the entities of the GeoMap, i.e. ask for a list of darts, nodes, edges, and faces, or a list of contours of a given face.
- Inspect the topology of the GeoMap by asking for the labels of the start and end node of a dart, its edge and its left and right faces, or by navigating on the set of darts according to the order defined by the permutations.
- Inspect the geometry of the GeoMap by asking for representative points of every entity.
- Modify the structure of the GeoMap by elementary Euler operators (see section 4.4.1) that can be composed to achieved any desired modification.

These operations form a standardized, easy-to-use interface to the GeoMap data type, which can thus form the common basis of many different algorithms. The interface encapsulates a number of potentially complicated algorithms that realize the required GeoMap functionality in an implementation-specific way. The encapsulated algorithms may vary widely between implementations, but the interface stays always the same. In spite of the simplicity of the interface, GeoMaps data structures and associated algorithms can be implemented very efficiently, and certain implementations are fast enough for interactive work on very large images. All segmentation methods to be discussed in this work are realized in terms of GeoMaps.

The most important element of the GeoMap interface is the DartTraverser. It represents a dart and provides all functionality directly related to darts:

```
query:      startNode: DartTraverser -> Node
            endNode:   DartTraverser -> Node
            leftFace:  DartTraverser -> Face
            rightFace: DartTraverser -> Face
            edge:      DartTraverser -> Edge
navigation: nextAlpha: DartTraverser -> DartTraverser (alpha successor)
            prevAlpha: DartTraverser -> DartTraverser (alpha predecessor)
            nextSigma: DartTraverser -> DartTraverser (sigma successor)
            prevSigma: DartTraverser -> DartTraverser (sigma predecessor)
            nextPhi:   DartTraverser -> DartTraverser (phi successor)
            prevPhi:   DartTraverser -> DartTraverser (phi predecessor)
```

Every σ-orbit (node), every α-orbit (edge) and every ϕ-orbit (contour) has one distinguished dart, its *anchor*. In principle, the anchor can be chosen arbitrarily among the darts in the orbit. But a reproducible rule is desirable from a computational point of view, so that the anchors of two isomorphic GeoMaps don't differ: We always choose the dart associated with the forward arc in every α-orbit, and the one with smallest label in the other two orbit types. The following functions are therefore available:

```
anchor:   Edge -> DartTraverser
anchor:   Node -> DartTraverser
contours: Face -> sequence_of(DartTraverser)
```

The latter returns all anchors associated with a given face. By convention, the first anchor in the sequence always belongs to the outer contour of the face, whereas the others represent embedded holes according to the contains relation. The Euler operators to be described in section 4.4.1 are also parametrized in terms of DARTTRAVERSERS.

One key advantage of the combination of geometry and topology in the GeoMap is that it becomes possible, even easy, to compute many kinds of attributes and properties for the cells and to use them to improve the capabilities of segmentation algorithms. We already mentioned that topological properties like the number of holes or the number of corners may be important cues for object recognition. These properties are easily obtainable from a GeoMap. Similarly useful are geometric properties like the area of a region, the length and curvature of an edge, the distance between two vertices or the angle between two edges. Since the cells of the GeoMap have a geometric representation, these are equally easy to compute. The same is true for perceptual criteria ("Gestalt" properties) such as proximity and good continuation, which are also geometric in nature.

In addition, it is very important to make use of the statistical properties of gray-levels or colors. Therefore, the GeoMap provides facilities to compute and store these properties if desired. For example, one may be interested in the minimal or average gradient magnitude (of the original image) along a given edge, or the mean and variance of the color distribution in a region. Since it depends on the application, which statistics are of interest and how they should be determined, the GeoMap supports these facilities in the form of callbacks: whenever the GeoMap receives a call to retrieve or update some application property, it forwards this call to the appropriate application-provided callback function, along with all relevant data describing the cell(s) involved. In particular, the GeoMap offers a set of *representative points* for each cell that can be used to sample an image function in order to collect the desired statistics. Given a cell, these points are passed to the callback function in form of an iterator that returns the points successively. The machinery to find the appropriate points is completely hidden behind the GeoMap interface. This solution is another example how the use of GeoMap data structure helps in the simplification and unification of algorithms. Much more detail about these concepts will be provided in [Meine 08].

4.3 GeoMap Realizations

4.3.1 Polygonal GeoMaps

The obvious realization of the GeoMap is by means of a polygonal plane partition:

Definition 4.8. *A* polygonal GeoMap *is a GeoMap whose arcs are polygonal lines. The points defining a polygonal arc are called* knots.[5]

Each arc is uniquely defined by an ordered sequence of knots and their connections by straight lines. The start and end knots of the polygon do not themselves belong to the arc,

[5]One could also require the arcs to be straight lines. The expressiveness of the two definitions is the same, because all interior knots of a polygonal arc could be turned into vertices of the GeoMap. In practice, we found definition 4.8 easier to work with.

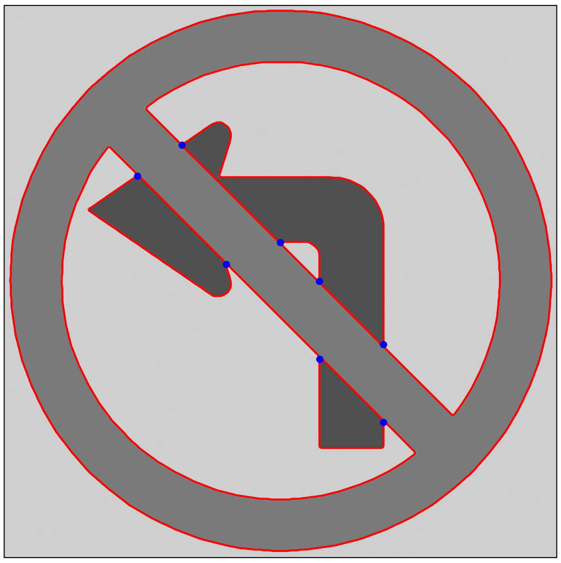


Figure 4.7: A polygonal GeoMap that encodes the partition of the image shown in the background (red: arcs, blue: vertices).

but must be vertices of the GeoMap. The σ-order of the GeoMap is realized by doubly linked cyclic lists of arcs around each vertex. Arbitrary arc geometries can be represented with any desired accuracy by placing intermediate knots sufficiently densely. Knots may have arbitrary coordinates in the plane, as long as the requirements of definition 2.2 are fulfilled, i.e. arcs do not intersect. The contains relation can be established efficiently by means of a fast polygon containment test, see e.g. [Heckbert 94]. The main task in the polygonal GeoMap approach is the creation of a suitable polygonal plane partition to begin with. We will deal with this problem in section 5.1, where we introduce algorithms that derive subpixel-accurate polygonal arcs from a given boundary indicator. Figure 4.7 shows an example.

Representative points for each cell (to be used for computing cell statistics) are defined as follows: Vertices are single points, and polygonal arcs are sequences of knots, so these points can be used directly to sample the contents of the respective cells. A polygonal arc can be resampled if a higher knot density is required for the computation of reliable statistics. In order to get points for the regions, we overlay the polygonal map with a square grid of adjustable pixel pitch. The grid points lying within a given region represent that region. These points can be efficiently determined by means of a connected components algorithm.

4.3.2 Grid-Based GeoMaps

Alternatively, one can define GeoMaps on top of a sampling grid. This approach builds upon low-level segmentation algorithms people are most familiar with, e.g. region labeling or edge detectors that mark pixels as boundary pixels. However, to make these methods applicable in the GeoMap framework it is necessary to develop provably correct algorithms that transform grid-based representations into GeoMaps. This is not a trivial problem, because naive definitions of neighborhood and connectivity in a grid easily lead

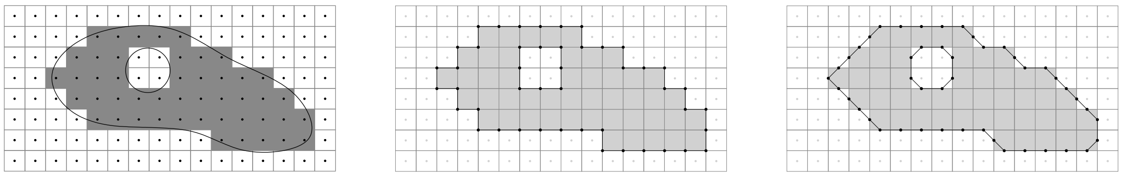


Figure 4.8: Left: Subset digitization $\hat{P}$ (gray) of a region P with a hole (indicated by solid contours). Center: The *interpixel* boundary (or *crack edge*) is the Euclidean boundary $\partial\hat{P}$ of the digitization $\hat{P}$. It consists of polygons whose knots are located at pixel corners. Right: The *mid-crack boundary* of $\hat{P}$ is a set of polygons whose knots are located at the center points of the cracks.

to topological contradictions (cf. the connectivity paradox in section 4.1) which prevent the construction of consistent GeoMaps. In order to avoid these problems, we start with a number of formal definitions of relevant terms:

Definition 4.9. *A countable set $S \subset \mathbb{R}^2$ of points is called a* grid *on $\mathbb{R}^2$ when the intersection $S \cap A$ with any bounded subset $A \subset \mathbb{R}^2$ contains only finitely many sampling points. The Voronoi regions associated with each sampling point $\vec{s} \in S$ are called* pixels:

$$\text{pixel}_S(\vec{s}) := \{\vec{x} : \forall \vec{s}' \in S \backslash \{\vec{s}\} : |\vec{x} - \vec{s}| \leq |\vec{x} - \vec{s}'|\}$$

Two sampling points are called directly adjacent *(or* adjacent*, without qualification) if their corresponding pixels share an edge. The shared edge of two directly adjacent pixels is called a* crack*. Two sampling points are* indirectly adjacent *if their pixels share only a corner. A* discrete path *in the grid is a sequence of directly adjacent sampling points.*

In this section, we will only apply this definition to square grids, where direct and indirect adjacency refer to the familiar 4- and 8-neighborhood. Moreover, we simplify the camera model by assuming that the point spread function is a δ-function and there is no noise. In other words, the regions and boundaries of the ideal geometric image are now directly digitized, similar to the rasterization of geometric models in computer graphics. The analysis of digitization models provides useful insight into what pixel configurations occur in grid-based shape representations, and how they are best treated from a topological point of view. The discretization of a *region* is most commonly described by means of *subset digitization*:

Definition 4.10. *The* subset digitization $\hat{P}$ *of a a set $P \subset \mathbb{R}^2$ is defined as the union of all pixels whose sampling points are located within P.*

If the grid is a square grid, subset digitization is also known as *Gauss digitization*, see figure 4.8 left. It should be noted that any region image whose connected components have been labeled according to the 4-neighborhood can be interpreted as the subset digitization of some (possibly unknown) plane partition. There are two common ways of defining the *boundary* of a subset digitization:

Definition 4.11. *Let P be a plane partition and $\hat{P}$ its subset digitization on a square grid. The Euclidean boundary $\partial\hat{P}$ of $\hat{P}$ is called the* interpixel boundary *or* crack edge *of $\hat{P}$ (figure 4.8 center). It consists of polygons whose segments are cracks, and whose knots are located at pixel corners. Alternatively, the* mid-crack edge *of $\hat{P}$ is defined as the set of polygons whose knots are at the cracks' center points (figure 4.8 right). Precisely, the mid-crack edge is obtained from the crack edge as follows: (a) Split all segments of the crack edge polygons at their center points. (b) Remove all knots from the resulting polygons that are located at pixel corners and whose degree is 2 (knots with other degrees, representing line endings and junctions, must remain in the mid-crack polygons).*

Mid-crack edges are important because they have higher geometric accuracy (in a sense to be made precise in section 6.2.1) than crack edges due to the reduced staircasing of diagonal lines.

Crack edges and mid-crack edges are indirectly defined digital boundaries, because they are derived from the subset digitization of a region. Alternatively, we can define digital boundaries directly on the basis of Euclidean arcs. One common representative of this approach is the *grid-intersection digitization:*

Definition 4.12. *Let $a : [0, 1] \to \mathbb{R}^2$ be a planar arc and S an axis-aligned square grid. The* grid lines *are the horizontal and vertical lines through the points of S. Let G be the set of intersection points between a and the grid lines. Then the* grid-intersection digitization *of a is the set of all pixels that contain a point of G (points in G that are exactly on a crack are arbitrarily assigned to one of the adjacent pixels).*

This digitization model is illustrated in figure 4.9 left. It results in a topologically thin digital arc, i.e. it is impossible to remove pixels without destroying the arc's connectivity (in the 8-neighbor sense). This model is, for example, realized by the well-known Bresenham algorithm for straight line drawing. It is also the typical result of region growing algorithms which leave explicit boundaries between regions, e.g. some versions of the watershed algorithm to be discussed later. Alternatively, one can digitize arcs by marking all pixels that it touches:

Definition 4.13. *The* supercover digitization *of a planar arc $a : [0, 1] \to \mathbb{R}^2$ is the union of all pixels whose intersection with the arc is non-empty.*

This kind of digital curve is shown in 4.9 right. It is always a superset of the grid intersection digitization, and typically results when edge pixels are marked by means of Canny's algorithm [Canny 86]. To enable uniform treatment of all types of digital boundaries, we first transform crack edges and mid-crack edges into an equivalent representation where boundaries are represented by marked pixels rather than polygons. In case of crack edges derived from a labeled region image, this can be done by means of the *crack insertion algorithm* which produces an image with twice the size of the original image by inserting extra rows and columns between the original pixels. These additional pixels are marked with a boundary label where appropriate:

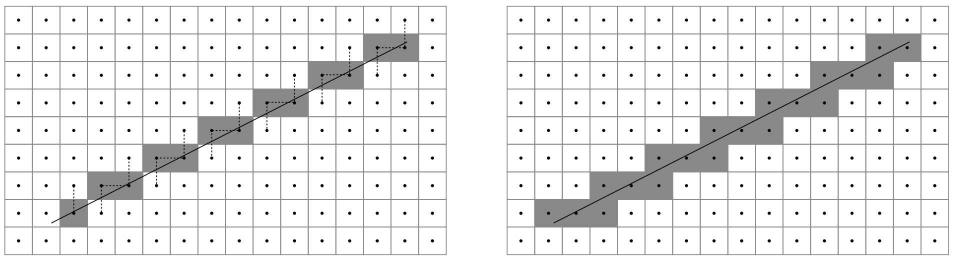


Figure 4.9: Left: Grid-intersection digitization of a straight line. The dashed lines are the relevant parts of the grid lines that contain intersection points. Right: Supercover digitization of a straight line.

Algorithm 4.1: Crack Insertion Algorithm

Input: A discrete region image I_{Region} on the integer domain $[0, ..., w-1] \times [0, ..., h-1]$, whose connected components are labeled according to the 4-neighborhood.

1. Create an image I_{Crack} on the integer domain $[-1, ..., 2w-1] \times [-1, ..., 2h-1]$ and label it as follows:
    a) Map all labels from $I_{\text{Region}}(i, j)$ onto the corresponding points $I_{\text{Crack}}(2i, 2j)$.
    b) For all points of the form $I_{\text{Crack}}(2i+1, 2j)$: If its horizontal neighbors $I_{\text{Crack}}(2i, 2j)$ and $I_{\text{Crack}}(2i+2, 2j)$ have different labels, or one is outside the image, label $I_{\text{Crack}}(2i+1, 2j)$ as a boundary point. Otherwise, copy the (unique) region label of the two neighbors.
    c) For all points of the form $I_{\text{Crack}}(2i, 2j+1)$: If its vertical neighbors $I_{\text{Crack}}(2i, 2j)$ and $I_{\text{Crack}}(2i, 2j+2)$ have different labels, or one is outside the image, label $I_{\text{Crack}}(2i, 2j+1)$ as a boundary point. Otherwise, copy the (unique) region label of the two neighbors.
    d) For all points of the form $I_{\text{Crack}}(2i+1, 2j+1)$: If none of the direct neighbors $I_{\text{Crack}}(2i, 2j+1)$, $I_{\text{Crack}}(2i, 2j-1)$, $I_{\text{Crack}}(2i+1, 2j)$, $I_{\text{Crack}}(2i-1, 2j)$ (where points outside the image are ignored) is marked as boundary, they necessarily belong all to the same region. In this case, copy their region label to the current point. Otherwise, mark $I_{\text{Crack}}(2i+1, 2j+1)$ as a boundary point.

Figure 4.10a illustrates the action of this algorithm[6]. We call the result of the crack insertion algorithm an *image with explicit inter-pixel boundaries* or a *crack-edge image*. The corresponding mid-crack image (figure 4.10b) can be computed from the crack-edge image by means of topology-preserving thinning, see below. It is easy to prove that both the explicit crack edges and the regions in the crack-edge image are 4-connected. In fact, they are even *well-composed* [Latecki 98]:

[6]It should be noted that the explicit execution of this transformation (which results in four times as many pixels as the original label image) can be avoided when the GeoMap is implemented in a clever way, see for example [Braquelaire & Domenger 99]. However, explicit crack edges lead to great simplifications in the subsequent algorithms and proofs.

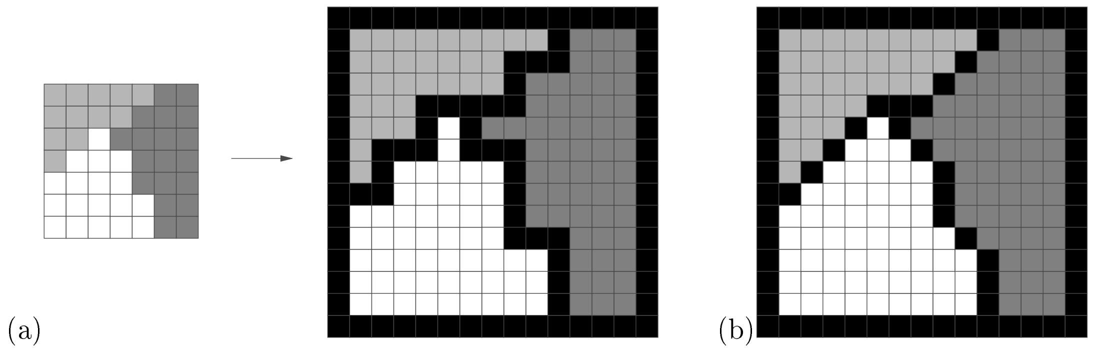


Figure 4.10: (a) A region image labeled according to the 4-neighborhood and the corresponding image with explicit interpixel boundaries resulting from the crack insertion algorithm (boundaries are marked in black). (b) The mid-crack image resulting from thinning the boundary of the crack-edge image.

Definition 4.14. *A set in a square grid is called* well-composed *when its connected components are identical under 4- and 8-connectivity. A binary image partition is well-composed if both the foreground and the background are well-composed.*

[Latecki 98] proved that this condition is equivalent to the condition that there is no 2×2 pixel block where foreground and background are arranged in a checkerboard-like pattern, i.e. where two diagonally opposite pixels are foreground, and the other two are background. In our case, the boundaries can be interpreted as foreground and the union of all regions as background, and non-existence of checkerboard configurations is guaranteed by the construction of the boundary in the crack insertion algorithm. Well-composedness ensures that topological inconsistencies such as the connectivity paradox cannot occur.

In contrast, the boundaries created by grid-intersection digitization are only 8-connected, and the resulting edge images are not well-composed. To achieve topological consistency here, it is necessary to use different neighborhood definitions for the foreground and background: We require the boundary to be 8-connected, and regions to be 4-connected [Rosenfeld 70]. Then, checkerboard configurations are always resolved so that the boundary is connected. Furthermore, we require 8-connected boundaries to be *topologically thin*:

Definition 4.15. *A set in a square grid is* topologically thin *when it is impossible to take a pixel from the set and assign it to the set's complement without changing the connected components of either the set or the complement. If a set is not topologically thin, it contains so called* simple pixels *whose removal will not change connectivity.*

Arbitrary boundaries can be made thin by means of the following thinning algorithm which recursively removes all simple pixels. A pixel can be classified as being simple or not by applying definition 4.15 in its 3×3 neighborhood only. In general, the result of thinning depends on the order in which simple pixels are considered, see figure 4.11a. Instead of using an arbitrary order (e.g. scan order), we prefer a thinning algorithm with explicit priority. The priority can be determined by some measure of confidence that

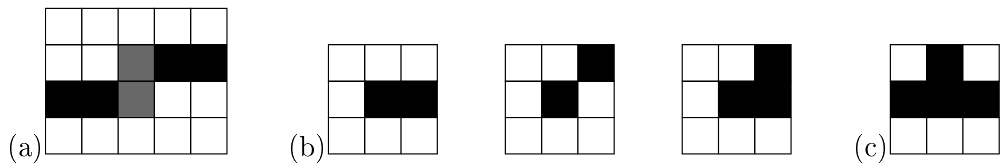


Figure 4.11: (a) Illustration of why priorities are useful. Either of the gray pixels can be removed, but not both. The one with higher confidence of belonging to the true edge should be kept. (b) Special thinning rules for terminations: Three edge termination configurations (modulo rotation and reflection) must be excluded from thinning although their center is a simple point. (c) Special thinning rules for T-junctions: The center point of this configuration is also simple but should be kept.

a pixel actually belongs to the boundary. Pixels with low confidence (e.g. small local gradient magnitude) are then removed first:

Algorithm 4.2: Thinning with Priority

Input: An image with a non-thin boundary, and additional information (e.g. the gradient magnitude) to compute priorities.

1. For every boundary pixel that is a 4-neighbor of the background: If it is a simple pixel, compute its priority and put it into a priority queue.
2. While the priority queue is not empty:
   a) Remove the pixel at the top of the queue and check whether it is still simple (this property may have changed due to relabeling of a neighbor). If yes, relabel it into background. Check whether pixels in the 4-neighborhood have become simple after the relabeling. If so, put them into the priority queue.

Since many edge detection algorithms do not produce closed boundaries, we must be careful to avoid removal of the terminations of edges during thinning. Since edge terminations are simple points, this has to be done by an exception to the rule of simple points: In the configurations shown in figure 4.11b, the center is not removed, despite its being simple. Another useful exception is to leave T-junctions untouched, figure 4.11c. The region topology is independent of whether or not we remove the center pixel of this configuration, but we found empirically that the geometric accuracy of the segmentation is almost always improved by leaving the T-junction intact. These exceptions to the rules of definition 4.15 are possible because the rules of definition 4.16 below correctly classify boundary pixels into nodes and edges even if these simple points remain in the boundary.

When the thinning algorithm (with special rules for terminations and T-junctions) is applied to a crack-edge image we obtain the corresponding *mid-crack image*, see figure 4.10b. Thinning is simplified here because no priorities are needed: Two simple pixels can never be neighbors in a crack-edge image, so that the result of thinning is independent

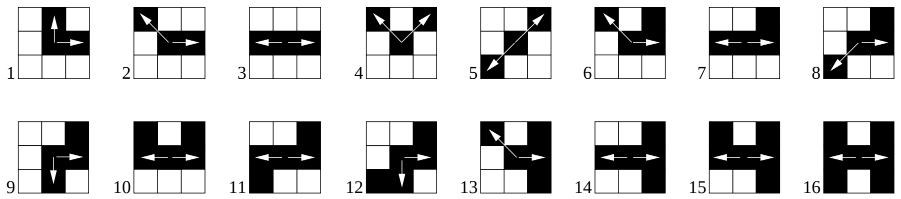


Figure 4.12: All possible configurations (up to reflection and rotation) in a pixel-based boundary representation where the center pixel is classified as an edge pixel. The continuation of the edge to either side is marked with an arrow. In all other configurations, the center is marked as a node pixel.

of the processing order. It is easily seen that the boundary pixels of the mid-crack image correspond to the knots of the midcrack polygons: When $\{\vec{x}_i\}$ denotes the set of knot positions in the midcrack polygons, exactly the pixels at coordinates $2\vec{x}_i$ are marked as boundary pixels in the mid-crack image.

We have now introduced various types of images where certain pixels are marked in a well-defined way as belonging to the boundary. The next step is to classify these boundary pixels into either edge or node pixels. Interestingly, this classification can be done by only considering 3×3 blocks of pixels. The basic idea of the classification algorithm is to consider a boundary pixel as an edge pixel whenever it has two uniquely defined neighbors that can be interpreted as its predecessor and successor in an arc. In the simplest case, this requirement is fulfilled when exactly two neighbors (which must not themselves be neighbors of each other) of the current pixel are marked as boundary. But this definition does not cover all cases that one intuitively considers as edge candidates, see for example configuration 7 in figure 4.12. A more general definition is obtained by preferring 4-neighbors over 8-neighbors whenever there is a choice between several candidates. This rule can be formalized as follows:

Definition 4.16. *A boundary pixel is classified as either an* edge *or a* node *pixel according to the following rules: Traverse the pixel's 8-neighborhood counter-clockwise and identify connected sectors of boundary pixels. If there are not exactly two such sectors, the center is a node pixel. Otherwise, check whether each of the two sectors contains at most one 4-neighbor. If this is not true, the center is again a node pixel, otherwise an edge pixel. In the latter case, one pixel from either sector is designated as the edge's continuation: the 4-neighbor, if the sector contains one, and the (unique) 8-neighbor otherwise.*

Figure 4.12 shows all possible configurations (up to rotation and reflection) that are classified as edge pixels according to this definition. The continuation of the edge to either side is marked by arrows. These neighbors are not necessarily themselves edge pixels, they can also belong to the node where the edge starts or ends. Not all configurations can actually occur in all types of boundary images: By construction, corners and junctions of an explicit inter-pixel boundary can only occur at pixels whose coordinates are both odd numbers. Therefore, only configurations 1, 3, 7, 10, 11, 14, 15, and 16 are possible in these images. In contrast, configurations 1, 9, and 12 cannot occur when the boundary is topologically thin. On first sight, it might appear that even more cases are impossible for

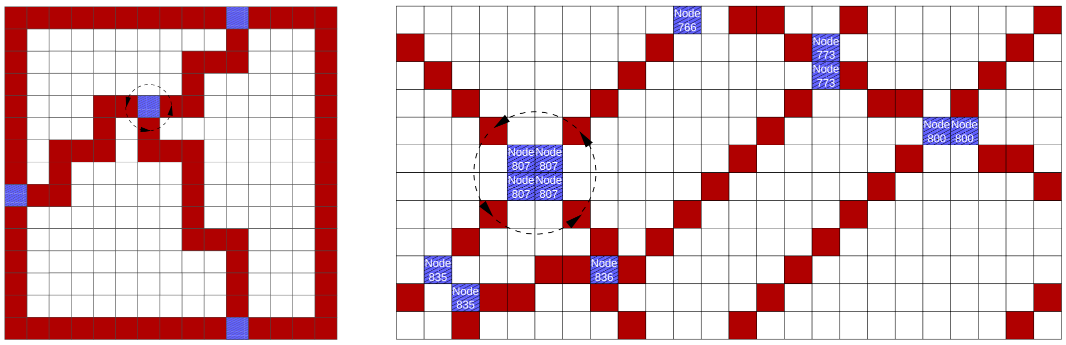


Figure 4.13: Classification of boundary pixels (edge pixels are marked red, nodes blue, regions white) of an image with explicit inter-pixel boundaries (left) and with thin 8-connected boundaries (right). The right example shows that nodes consisting of several pixels are natural in this kind of boundary image. The σ-orbits around one node in each image are also plotted. Illustrations from [Meine 03].

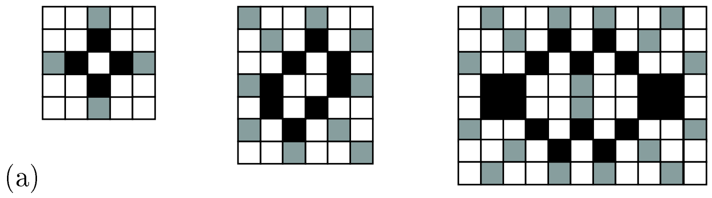
thin boundaries, e.g. number 15. But in these cases there exist boundary continuations outside the depicted 3×3 region which make the configuration indeed thin. It should be noted that the set of node pixels resulting from an interpixel boundary is identical to the set resulting from the corresponding mid-crack representation, provided that edge terminations and T-junctions have been excluded from thinning.

After classification, we compute the 8-connected components of all pixels classified as node pixels. Each component will become a node of the final GeoMap. It may appear surprising at first that nodes resulting from pixel-based boundary representations are permitted to consist of several pixels, instead of a single point like in polygonal GeoMaps. But permitting spatially extended nodes is the simplest solution that guarantees topological consistence of the resulting GeoMap. Note that nodes consisting of multiple pixels cannot occur in images with explicit inter-pixel and mid-crack boundaries, because corners and junctions in these images are always located at pixels with odd coordinates. In other image types, we have found that it is always a sign of insufficient resolution when nodes with more than four pixels occur frequently. Figure 4.13 depicts examples for the result of boundary pixel classification.

Nodes consisting of several pixels do not necessarily form simply connected components – they may have holes. Figure 4.14(a) shows a few examples where this happened. But with the exception of the first example, this problem is extremely rare in practice, except when the resolution is clearly too low to segment the depicted objects. Therefore, we do not bother to treat holes within nodes in any sophisticated way, but simply fill these holes with a standard flood-fill algorithm and add them to the node.

After node labeling we must label the edges. This is done by means of edge tracing[7]: We start with an edge pixel that has at least one continuation belonging to a node. Then we recursively follow along the opposite continuation until we again arrive at a node.

[7]This edge tracing algorithm already appeared in [Haralick & Shapiro 92]. However, the all-important classification and continuation rules (our definition 4.16) were left open as application-specific choices in that work. Therefore, no correctness proof of the algorithm was possible.



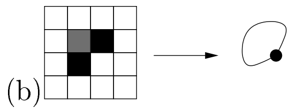


Figure 4.14: (a) Examples for configurations where a node (marked black) is not simply connected. Except for the left one, these configurations are extremely unlikely and only occur when the resolution is insufficient for resolving the depicted objects properly. (b) Undesirable edge configuration. According to the classification rules, the three boundary pixels are interpreted as one edge (gray) and two connected node pixels. This is topologically a self-loop, which is consistent but undesirable because the enclosed region does not contain region pixels. Thinning removes this and similar configurations.

All edge pixels along the way belong to the same edge and are labeled accordingly. This algorithm even works when edge pixels from different edges happen to be 8-neighbors near a node (compare figure 4.13 right) – they will never be continuations of each other. Every thus traced edge is used to construct a dart of the GeoMap to be build. The opposite dart in the same α-orbit is found by tracing the edge backwards. The σ-orbits are defined by running around each node with a standard left-hand-on-the-wall border tracing algorithm, and storing the darts in the order they are found. This is illustrated for two nodes in figure 4.13.

As always, the ϕ-orbits are defined by the relationship $\phi(d) = \sigma^{-1}(\alpha(d))$. Determining the exterior function and contains relation is much easier than for polygonal plane partitions because we can take advantage of the grid. We first label connected components of region pixels according to the 4-neighborhood. This labeling is always consistent with the boundary topology (cf. the theorem below). All ϕ-orbits whose edge pixels have the region label n to their left (where "left" is defined relative to the traversal direction of the ϕ-orbit) belong to the contour set Φ_n. The main contour among the contours in Φ_n is easily identified because it is the first contour of region n that one finds when the image is traversed in standard scan order. The exterior function and contains relation are readily constructed from this information.

The following theorem establishes that the algorithms described in this section indeed result in topologically valid GeoMaps. In the literature, an analogous theorem has only been available for explicit inter-pixel boundaries, where the proof is simple. A unified proof covering all kinds of pixel-based boundary representations is given for the first time.

Theorem 4.2. *When a boundary image is classified and labeled according to the rules described above, the result is of the same homotopy type as a suitably chosen polygonal plane partition and therefore constitutes a valid GeoMap.*

Proof. We prove the theorem by explicitly constructing a polygonal plane partition. First, observe that regions are labeled by using 4-connectivity, whereas the boundary is either well-composed or 8-connected. It is thus guaranteed that different regions are always separated by closed boundaries – the “connectivity paradox” cannot occur.

Second, edges cannot cross without forming a node at the crossing. We see this by analyzing the configurations in figure 4.12. The claim is obvious for configurations 1 to 5 and can be proved by contradiction for the others. Consider, for example, configuration 11. It can in principle be interpreted as a combination of configurations 3 and 5, in which case two independent edges would cross at its center. For this to happen both the right and upper right pixels in configuration 11 must be classified as edge pixels as well, and must thus have a well-defined continuation. It is easy to see that the right neighbor (if it is an edge pixel) can only have continuations at its left (i.e. to the center of configuration 11) and its top (the upper right pixel of configuration 11). Likewise, the edge of the upper right pixel cannot be continued towards the center pixel, because its 4-neighbor (i.e. the right pixel in configuration 11) takes precedence. Therefore, all three pixels belong to the same edge, contrary to the assumption that two different edges cross. In the same way, the claim can be proved for all other configurations.

We can now construct a polygonal arc for every edge by simply connecting the centers of successive edge pixels with straight lines. In addition, we draw straight lines from the enters of the first and last points of every edge to the nearest point (in Euclidean sense) of the adjacent node region. Since nodes are simply connected regions by construction, we can reduce them in a homotopy-preserving way to single representative points, and the end points of the arcs can be connected to this point by suitably extending the arcs. This always results in a valid polygonal plane partition, which can be transformed into a GeoMap according to definition 4.7. □

Transforming a pixel-based GeoMap into a polygonal one is not only of academic interest for the above proof. It is also a useful transition in practice because polygonal representations lend themselves naturally to geometric analysis and manipulation. The only practical problem is the definition of the representative points of the nodes and their connection to the start/end points of the incident edges. For nodes consisting of up to 4 pixels (i.e. all nodes in an interpixel or mid-crack representation, and most in a thin-boundary representation), we determined that it is sufficient to just use the node’s center of mass and connect it to the edges segments by straight lines. In all possible cases of up to 4 node pixels, the order induced by the angles of these lines is identical to the σ-order of the corresponding edges. For more complex nodes, this is still being investigated. But recall that nodes larger than four pixels usually indicate insufficient resolution.

Instead of just connecting consecutive pixel centers like in the proof, edges can also be transformed into maximal straight segments by means of *digital straight segment* algorithms (see section 4.4.2 for algorithms and example images). These algorithms are designed so that the polygonal representation is simplified without altering the topology. The resulting polygonal GeoMaps are not only more compact, but are also geometrically more accurate because the creation of maximal straight segments acts as a noise filter.

4.4 Manipulation of a GeoMap

The initial result of low-level segmentation is rarely the desired one. It is therefore necessary to provide functions which modify a GeoMap in an orderly way until we arrive at the desired segmentation. We distinguish three manipulation types:

1. purely geometric ones that only change the position of nodes and edges without changing topology
2. node and edge removals that don't change the geometry of the remaining cells, and
3. node and edge insertions which necessarily involve both topology and geometry (the location of the new cell).

Although cell insertions and removals are inverses of each other, there is a fundamental asymmetry in their computational complexity. Obviously, we can remove at most as many edges as there are in the GeoMap, but there are $\mathcal{O}(f\, n_f^2)$ possibilities for the topological placement of every newly inserted edge (where f is the number of faces, and n_f is the average number of nodes bounding a face), and even more for its geometric location. For example, if we assume that the new edge runs from the upper left to the lower right corner of a square region with $p \times p$ pixels, and allow only steps down and to the right, there exist already 2^{p-1} different paths. Reducing an initial over-segmentation (by edge removal) is therefore in general much simpler than closing gaps in an initial under-segmentation (by edge insertion). When the latter is attempted, a clear and simple decision rule for the location of new edges is required. To avoid these complications, we will restrict ourselves to manipulation types (1) and (2), where a number of general statements can be made.

4.4.1 Euler Operators and Contraction Kernels

We first consider operations that remove boundary cells without changing the topology and geometry of the remaining boundary. These operators are called *Euler operators* because Euler's equation (4.2) must remain valid after the transformation in order to preserve the GeoMap's planarity [Mäntylä 88, Köthe 00][8]. Arbitrary complex manipulations can be reduced to a small set of elementary Euler operators as shown in figure 4.15. These operators are elementary in the sense that they change the number of edges exactly by one. They are complete in the sense that any desired simplification can be expressed as a sequence of Euler operations. For example, to remove the complete boundary between two neighboring regions (which may consist of several edges) we must first execute one merge faces operation, followed by the appropriate number of remove bridge operations. Below we present algorithms for the necessary modifications of the GeoMaps.

The operation merge faces deletes an edge and merges the two adjacent faces that had been separated by the edge. Merge edges removes a node of degree two and merges the two edges it bounds. Contract edge also removes an edge, but instead of the adjacent faces the two end points are merged into one. This is equivalent to merge faces on the

[8]Some authors prefer the term "superficial operators" to the term "Euler operators".

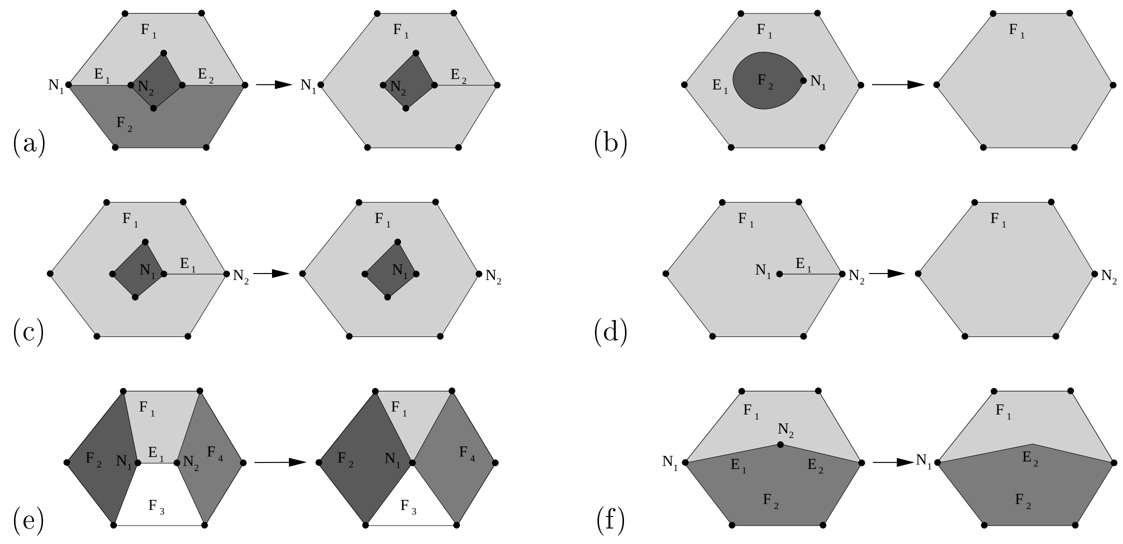


Figure 4.15: The most important elementary Euler operators. (a) and (b) two variants of merge faces, (c) and (d) two variants of remove bridge, (e) contract edge, (f) merge edges (a variant of contract edge when one of the end nodes of E_1 has degree 2).

dual map (i.e. the one were the roles of nodes and faces have been switched). Finally, remove bridge deletes an edge that is embedded in only one face, i.e. whose left and right face are identical. We explain the details of these algorithms in the following algorithms:

Algorithm 4.3: Merge faces

Input: Edge E_1 is to be removed. Let d be the dart running form node N_1 to node N_2 and $d' = \alpha(d)$ its opposite dart. Let F_1 be the face to the left of d which is to survive after the merge operation. Ensure that the orbit $\phi^*(d')$ is the outer contour of F_2 (otherwise, flip the roles of d and d' and therefore those of F_1/F_2 and N_1/N_2)[9]. Note that degree(N_1) > 1 and degree(N_2) > 1 because F_1 and F_2 are distinct faces.

1. Update the contains relation: replace all entries contains (F_2, ϕ_k^*) with contains (F_1, ϕ_k^*). Delete the entry exterior $(\phi^*(d'))$ from the exterior function of the GeoMap.
2. When N_1 and N_2 are identical (i.e. E_1 is a loop) and degree(N_1) $= 2$ (cf. figure 4.15b):
   a) Delete the entries exterior $(\phi^*(d))$ and contains $(F_1, \phi^*(d))$ from the GeoMap.
   b) Remove the darts d and d'. This will automatically remove orbits $\alpha^*(d)$ (i.e. edge E_1), $\sigma^*(d)$ (i.e. node $N_1 = N_2$), $\phi^*(d')$ (i.e. the face F_2), and $\phi^*(d)$ (i.e. the hole in face F_1) because these orbits wouldn't contain any darts after deletion of d and d'.

[9]This is always possible because $\phi^*(d')$ and $\phi^*(d')$ cannot both be exterior contours.

3. Otherwise (cf. figure 4.15a):

   a) Let $d_\phi \neq d$ be a dart such that $\phi^*(d_\phi) = \phi^*(d)$. If no such dart exists, choose $d_\phi \neq d'$ such that $\phi^*(d_\phi) = \phi^*(d')$ (at least one of these choices is always possible). Likewise, choose $d_\sigma \neq d$ with $\sigma^*(d_\sigma) = \sigma^*(d)$ and $d'_\sigma \neq d'$ with $\sigma^*(d'_\sigma) = \sigma^*(d')$.

   b) Remove d and d' from their respective σ-orbits[10]. Due to the condition $\phi = \sigma^{-1} \circ \alpha$, this also removes these darts from the ϕ-orbits. Consequently, the previously separate orbits $\phi^*(d)$ and $\phi^*(d')$ are automatically turned into a single contour $\phi^*(d_\phi)$ (which bounds the surviving face F_1), whereas the orbit previously bounding face F_2 and the face itself are removed. Note that the preconditions of merge faces are defined so that the entries exterior $(\phi^*(d_\phi))$ and contains $(F_1, \phi^*(d_\phi))$ can remain untouched.

   c) Remove the darts d and d' and thus the orbit $\alpha^*(d)$ (i.e. edge $E_{:1}$).

   d) Update the anchors of the orbits $\phi^*(d_\phi)$, $\sigma^*(d_\sigma)$, and $\sigma^*(d'_\sigma)$ in order to ensure that d and d' are no longer used as anchors.

The remove bridge operator is similar to merge faces in that an edge is removed. But this time, the edge is a bridge, i.e. its left and right face are identical. Remove bridge may create a new hole within this face, see figure 4.15c.

Algorithm 4.4: Remove bridge

Input: Edge E_1 is a bridge to be removed. Let d be the dart in E_1 running form node N_1 to node N_2 and $d' = \alpha(d)$ its opposite dart. Let F_1 be the face containing E_1.

1. Let $\xi =$ exterior $(\phi^*(d))$ and remove this entry from the exterior function of the GeoMap. Also, remove contains $(F_1, \phi^*(d))$, if this entry exists in the contains relation of the GeoMap.

2. If degree$(N_1) = 1$ and degree$(N_2) = 1$:

   a) Remove the darts d and d'. This will automatically remove orbits $\alpha^*(d)$ (i.e. edge E_1), $\sigma^*(d)$ (i.e. node N_1), $\sigma^*(d')$ (i.e. node N_2), and $\phi^*(d) = \phi^*(d')$ (i.e. the hole in face F_1) because these orbits wouldn't contain any darts after deletion of d and d'.

3. Otherwise:

   a) If degree$(N_1) > 1$ (cf. figure 4.15c):

      i. Let $d_\phi = \sigma^{-1}(d)$.

[10] Formally, removing d from its σ-orbit means the following: Let $d_+ = \sigma(d)$ and $d_- = \sigma^{-1}(d)$ be σ-successor and predecessor of d. Now, modify the σ-permutation of the GeoMap so that $\sigma(d_-) = d_+$ and remove d from the permutation.

ii. Remove d from its σ-orbit. This automatically removes d from its ϕ-orbit as well and turns $\phi^*(d_\phi)$ into a separate contour which bounds F_1. Update the anchor of $\phi^*(d_\phi)$.

iii. If $\xi = \mathsf{true}$ or if $\phi^*(d_\phi)$ is oriented in mathematically negative sense: Create new entries $\mathsf{exterior}\,(\phi^*(d_\phi)) = \mathsf{true}$ and $\mathsf{contains}\,(F_1, \phi^*(d_\phi))$ in the GeoMap.

iv. Otherwise create a new entry $\mathsf{exterior}\,(\phi^*(d_\phi)) = \mathsf{false}$ in the GeoMap.

b) Otherwise (cf. 4.15d):

i. Remove the orbit $\sigma^*(d)$ and therefore the node N_1.

c) Proceed likewise with dart d' and node N_2.

d) Remove d and d' and therefore the orbit $\alpha(d)$ (i.e. edge E_1).

The operation contract edge provides yet another possibility to remove an edge. Instead of merging two adjacent faces, it merges the two end nodes of the edge. It is thus equivalent to a merge face operation in the dual GeoMap. We assume that both end nodes of the edge to be contracted have a degree of at least 2, because otherwise contract edge is identical to the variant of remove bridge depicted in figure 4.15d:

Algorithm 4.5: Contract edge

Input: Edge E_1 is the edge to be removed. Let d be the dart in E_1 running form node N_1 to node N_2 and $d' = \alpha(d)$ its opposite dart. make sure that $\mathsf{degree}(N_1) > 1$ and $\mathsf{degree}(N_2) > 1$.

1. Let $d_+ = \sigma(d)$ and $d_- = \sigma^{-1}(d)$ be σ-successor and predecessor of d. Likewise, let $d'_+ = \sigma(d')$ and $d'_- = \sigma^{-1}(d')$ be σ-successor and predecessor of d'.

2. Change the σ-permutation such that $\sigma(d'_-) = d_+$ and $\sigma(d_-) = d'_+$. This removes d and d' from their respective σ- and ϕ-orbits.

3. Remove the darts d and d' and therefore the orbit $\alpha^*(d)$ (i.e. edge E_1).

4. Update the anchors of the orbits $\phi^*(d_+)$, $\phi^*(d_-)$, and $\sigma^*(d_+)$.

The operation merge edges is just a variant of contract edge where $\mathsf{degree}(N_2) = 2$. It replaces an "edge-node-edge" chain by a single edge. We can check that the above operators indeed preserve GeoMap planarity by looking at Euler's equation. Let Δn, Δe, Δf, Δk be the changes in the number of nodes, edges, faces and exterior contours (holes) caused by an Euler operator. Then any Euler operator must satisfy the condition

$$\Delta n - \Delta e + \Delta f - \Delta k = 0$$

In operator merge faces, we have $\Delta n = 0$, $\Delta e = -1$, $\Delta f = -1$, $\Delta k = 0$ (case 3) or $\Delta n = -1$, $\Delta e = -1$, $\Delta f = -1$, $\Delta k = -1$ (case 2). Thus, the condition is always satisfied. In operator remove bridge, we must distinguish three cases: when both end nodes of the bridge have degree equal to one, we get $\Delta n = -2$, $\Delta e = -1$, $\Delta f = 0$, $\Delta k = -1$, when one end node has degree one, we get $\Delta n = -1$, $\Delta e = -1$, $\Delta f = 0$, $\Delta k = 0$, and if neither end node has degree one, we get $\Delta n = 0$, $\Delta e = -1$, $\Delta f = 0$, $\Delta k = 1$. The operator contract edge implies $\Delta n = -1$, $\Delta e = -1$, $\Delta f = 0$, $\Delta k = 0$. Again the required identity holds in all cases.

It is easy to see that this set of Euler operators is complete in the sense that any GeoMap can be reduced to a single face (the infinite face F_∞): first apply merge faces until the number of faces is reduced to 1. The result is a GeoMap whose only face is F_∞ with a number of holes that do not contain any faces, but only consist of edges and nodes (which therefore form trees). Note that the number of edges in a tree is one less than the number of nodes. Thus, we now have $f = 1$ and, when k is the number of trees, $n = e + k$. Now, apply the variant of remove bridge that requires exactly one end node of the bridge to have degree one until no such bridges remain. Each application of this operation cuts one leave (i.e. one edge and one node) from one of the trees. We end up with k trees that just consist of a single edge with two nodes, i.e. $e = k$ and $n = 2k$. Finally, use the variant of remove bridge requiring both ends of the bridge to have degree 1. Every application of this operation changes the GeoMap so that $\Delta n = -2$, $\Delta e = -1$, $\Delta f = 0$, $\Delta k = -1$. Thus, we indeed arrive at a GeoMap with just one face. Since all intermediate GeoMaps are valid, the next step in the processing chain is always a legal operation.

Another way to understand this is by noting that Euler's equation defines a linear, 3-dimensional subspace of $\mathbb{Z}^4$, and the modification vectors (Δn, Δe, Δf, Δk) of the operators considered span this space. Thus, if we add the corresponding inverse operators (split face, insert bridge, split node), we can create any combination of n, e, f, and k that is permitted by Euler's equation. The set of Euler operators together with their inverses is complete in the sense that any GeoMap can be transformed in any other by just using these operators: First, find two reduction sequences that remove all cells (except the infinite face) from the source and target map. Then reverse this sequence for the target map and replace every operator with its inverse. The concatenation of the two sequences *source→infinite face only→target* performs the desired transformation, although probably in an inefficient way. However, we will not consider the inverse operators in this work because they are not needed in our context.

Since all Euler operators involve a single edge, we can parametrize the corresponding functions in the abstract data type of a GeoMap by simply passing a DARTTRAVERSER:

```
mergeFaces:      DartTraverser -> nil
removeBridge:    DartTraverser -> nil
contractEdge:    DartTraverser -> nil
```

4.4.2 Topology-preserving Manipulations

Since topology is concerned with properties that remain invariant under bending and morphing transforms, we have a lot of freedom to change a GeoMap's geometry without changing topology. Geometric modifications have generally two goals: we may want to simplify the geometric representation of the edges (e.g. use polygons with fewer knots), or we may want to improve the localization accuracy of the representation (e.g. by going from a pixel-based to a subpixel representation, as in Canny's algorithm 5.10). Sometimes, both goals can be achieved simultaneously: transforming a pixel chain into a polygon according to algorithm 4.7 below may result in a representation that is both more compact (fewer knots are required when the original object is representable by a simple polygon) and more accurate (because the round-off errors of pixel-based representations are reduced by averaging over longer edge segments).

Constructing a GeoMap only to simplify it immediately afterward may seem wasteful on first sight. However, there is a major motivation for working in this way: After construction of an initial GeoMap, we can compute non-local properties (such as average region colors, good continuations of edges across junctions, or global contour salience) that are not available before regions, edges, and vertices have been defined. Thanks to this additional information, a GeoMap derived indirectly from some initial GeoMap can potentially have much higher quality than a GeoMap that has been computed directly from a given boundary indicator. The *snake* approach to segmentation [Kass et al. 88] is a good case in point: by placing an initial closed contour (which is a basic kind of GeoMap) near the boundary of interest, one can adaptively compute statistical properties of the interior and exterior regions, as well as boundary length and curvature. These measures are not available without the definition of an initial contour, and play key roles during the iterative determination of the optimal contour. Finding this contour without the additional non-local information is much more difficult or impossible.

At this point of the discussion, we shall not be concerned with specific criteria and methods for geometric manipulations, but with two general problems: GeoMap simplification and topology preservation. Since the knots in the arcs of an initial GeoMap are usually placed very densely (with distance in the order of the pixel pitch or less), a simplified polygonal representation is generally possible:

Algorithm 4.6: Polygon simplification

Input: A polygonal GeoMap with high knot density.

1. For each arc that is not a self-loop perform the following recursive procedure:

    a) Compute the chord between the start and end knots of the current polygon segment and find the intermediate knot whose distance d from the chord is maximum. If d exceeds a threshold d_0, split the chord at the intermediate knot and repeat the procedure with the two sub-chords. The threshold d_0 should be chosen so that knot displacements exceeding d_0 are unlikely to be caused by noise. Noise analysis of edge point accuracy will be performed in section 7.2.2.

2. For each arc that is a self-loop: always split the arc at the knot with largest distance from the start knot, and continue with the above recursive procedure.

Other split criteria are also possible. For example, [Ren et al. 05a] propose to split at the knot which maximizes the orientation change of the two sub-chords, and stop the recursion when this angle is small enough. This has the advantage of being scale-independent. However, it is inferior in terms of error propagation: Measurement errors in orientation are much larger then errors in position, because orientation is essentially the derivative of position. Another consideration with polygonal GeoMap simplification is that the arcs of the coarser polygon may intersect, violating the topology constraint of a GeoMap. This can be checked by the line intersection test below (algorithm 4.8), and the recursion must possibly be continued until no intersection occurs. However, this rarely happens when the threshold d_0 is chosen according to the measurement error.

In case of a GeoMap that is represented on the grid, a different class of simplification algorithms can be applied. They create a polygon from a given pixel edge or crack edge, and preservation of topology is guaranteed. In fact, the original grid-based boundary can be reconstructed from the resulting polygonal arcs. These algorithms assume that the given polygon was created by a particular edge digitization method, namely subset digitization of the adjacent regions (definition 4.11) for crack edges, and grid-intersection digitization (definition 4.12) for thin 8-connected pixel boundaries. The set of pixels or cracks that are marked by the digitization of a single straight line segment is called a *digital straight line*. Thus, digital straight line algorithms split a given set of pixels or cracks into maximal digital straight lines and estimate the corresponding preimages of the digitization, i.e. the original Euclidean line segments. One can show that the digitization models mentioned imply an important property of digital straight lines: their convex hulls are always enclosed between two parallel Euclidean lines (called the *left* and *right supporting lines*) whose distance is narrowly bounded. The value of the bound depends on the line's direction and the digitization model [Andersen & Kim 85]. According to [Kovalevsky 97] (crack edges) and [Debled-Rennesson & Reveilles 95] (8-connected edges), the bounds can be expressed by the following inequalities:

$$0 \leq a\,x + b\,y + c \leq \begin{cases} |a| + |b| - 1 & \text{for crack edges} \\ \max(|a|\,, |b|) - 1 & \text{for 8-connected edges} \end{cases}$$

where a, b, c are suitably chosen integers that characterize the digital straight line. These inequalities allow the definition of so-called *linear online algorithms* which trace the pixels of the contour or line under consideration only once [Debled-Rennesson & Reveilles 95, Kovalevsky 97]. Starting from a point on the contour or line, the algorithm extends the current straight line segment as long as there exist a, b, c fulfilling the inequalities. Otherwise, the current segment is finished and a new segment initialized. We explain the algorithm for crack edges. To apply it to 8-connected edges, the expression $|a| + |b|$ must simply be replaced with $\max(|a|\,, |b|)$:

Algorithm 4.7: Digital Straight Line Detection

Input: The first two points $\vec{p}_1$ and $\vec{p}_2$ of a new line segment. Initialize four points representing the convex hull as $\vec{l}_1 = \vec{r}_1 = \vec{p}_1$ and $\vec{l}_2 = \vec{r}_2 = \vec{p}_2$ and let the current estimate of the line parameters be $(-b, a)^T = \vec{p}_2 - \vec{p}_1$ and $c = -(a, b)\,\vec{l}_2$. The line with normal $(a, b)^T$ through $\vec{l}_2$ is called the *left supporting line*, and the parallel line through $\vec{r}_2$ is the *right supporting line.* The line running in the middle of the two supporting lines is the *axis* of the current segment.

1. Repeat for points $\vec{p}_n$, $n = 3, 4, ..., N$:
   a) Check if the current segment (including point $\vec{p}_n$) contains at most two different increment vectors between consecutive points. If not, goto (h).
   b) Compute $v_n = (a, b)\,\vec{p}_n + c$
   c) If $0 < v_n < |a| + |b| - 1$: The new point lies between the supporting lines. Add it to the current segment.
   d) If $v_n = 0$: The point is on the left supporting line. Update $\vec{l}_2 = \vec{p}_n$ and add $\vec{p}_n$ to the current segment.
   e) If $v_n = |a| + |b| - 1$: The point is on the right supporting line. Update $\vec{r}_2 = \vec{p}_n$ and add $\vec{p}_n$ to the current segment.
   f) If $v_n = -1$: The point is just outside the left supporting line, but the line can be adjusted. Update $\vec{l}_2 = \vec{p}_n$; $\vec{r}_1 = \vec{r}_2$; $(-b, a)^T = \vec{l}_2 - \vec{l}_1$; $c = -(a, b)\,\vec{l}_2$ and add $\vec{p}_n$ to the current segment.
   g) If $v_n = |a| + |b|$: The point is just outside the right supporting line, but the line can be adjusted. Update $\vec{r}_2 = \vec{p}_n$; $\vec{l}_1 = \vec{l}_2$; $(-b, a)^T = \vec{r}_2 - \vec{r}_1$; $c = -(a, b)\,\vec{l}_2$ and add $\vec{p}_n$ to the current segment.
   h) Otherwise, the point $\vec{p}_n$ cannot belong to the current straight line. Store the parameters a, b and $c = -(a, b)\,\frac{\vec{l}_2 + \vec{r}_2}{2}$ for the axis of the segment just finished. Initialize a new line segment with $\vec{p}_{n-1}$ and $\vec{p}_n$ as its first two points and go to 1.
2. Assume that step 1 produced the segments $s_1, ..., s_K$.
   a) When the original polygon was closed (i.e. $\vec{p}_1 = \vec{p}_N$), return the point sequence [intersection(s_K, s_1), intersection(s_1, s_2), ..., intersection(s_K, s_1)], where the function intersection(.) computes the common point of the two segments' axes.
   b) When the original polygon was open (i.e. $\vec{p}_1 \neq \vec{p}_N$), return the point sequence [$\vec{p}_1$, intersection(s_1, s_2), ...,intersection(s_{K-1}, s_K), $\vec{p}_N$].

The above algorithm constructs a set of intersection points, and the edge is assumed to be straight between these points. These straight edges are constructed in such a way that their digitization reproduces the original polygon, when the same digitization method (subset or grid intersection digitization) is applied [Klette & Rosenfeld 04]. Therefore, when the original polygons of the GeoMap are replaced with their digital straight line approximations, the topology of the GeoMap is preserved! The disadvantage of this algorithm is that the resulting polygon depends on the starting point of the tracing. Figure 4.16 illustrates digital straight lines obtained for part of a circle.

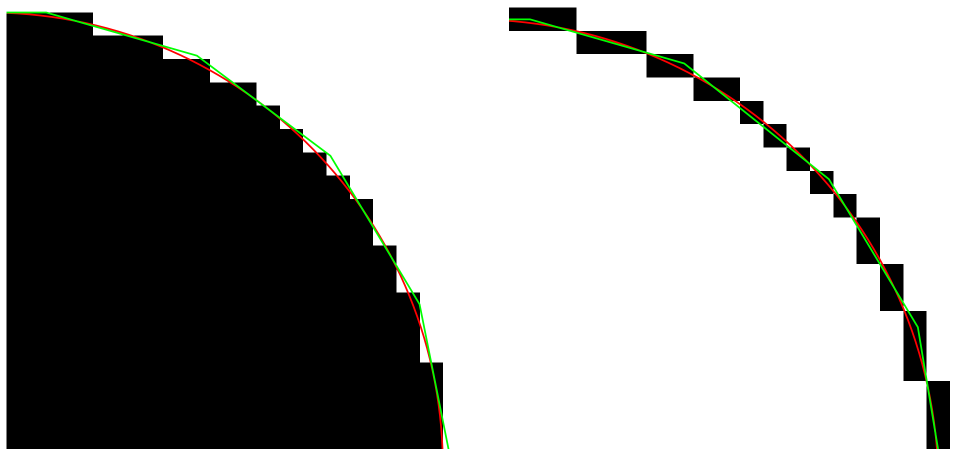


Figure 4.16: Digital straight line approximations (green) of a circle with radius 20 pixels (red) produced with algorithm 4.7. Left: Using subset digitization (black) and $|a| + |b|$. Right: Using grid intersection digitization (black) and $\max(|a|, |b|)$.

Unlike the digital straight line algorithm, many other geometric manipulations (including polygon simplification according to algorithm 4.6) run the inherent risk of violating topology: The modified arcs may have intersections at other points than the GeoMap's vertices. While it is possible to change the GeoMap's topology by introducing new vertices, this is not always the desired solution. Often, one simply wants to prevent such manipulations, see for example [Goudail & Réfrégier 04]. Whenever a point is to be translated, it is checked whether its adjacent lines would cross any of the existing contours afterward. Since the GeoMap topology was correct before the manipulation, it is sufficient to check this property against all contours of the adjacent regions. An interior edge point is adjacent to at most two different regions, while a vertex can have arbitrary many adjacent regions (although the number hardly ever exceeds four in practice). In case of a polygonal GeoMap, the algorithms is as follows:

Algorithm 4.8: Topology preservation under geometric manipulation of a polygonal GeoMap

Input: A polygonal GeoMap and a vertex or internal knot to be moved to a new location.

```
1. For each line segment starting at the given point:
   a) Compute the bounding box of the line segment for the new point position.
   b) For every arc bounding one of the regions adjacent to the point to be moved:
      i. Check whether the arc's bounding box intersects the line's bounding box.
         If not, the topology cannot be violated.
      ii. Otherwise, repeat the same check with the bounding boxes of the arcs
          internal line segments. If there is no intersection, the topology cannot be
          violated.
```

iii. Otherwise, perform a line intersection check. Let the two lines be bounded by the points $\vec{p}$, $\vec{q}$ and $\vec{r}$, $\vec{s}$ respectively. Then, the two lines intersect when $\vec{r}$ is to the left of the line $\vec{p}\vec{q}$, and $\vec{s}$ to its right (or vice versa), and $\vec{p}$ is to the left of line $\vec{r}\vec{s}$, and $\vec{q}$ to its right (or vice versa). Point $\vec{r}$ is to the left of line $\vec{p}\vec{q}$, when the sign of angle $\vec{p}\vec{q}\vec{r}$ is positive, which can be determined by

$$\operatorname{sign}(\angle\vec{p}\vec{q}\vec{r}) = \operatorname{sign}(\vec{p}_x(\vec{q}_y - \vec{r}_y) + \vec{q}_x(\vec{r}_y - \vec{p}_y) + \vec{r}_x(\vec{p}_y - \vec{q}_y))$$

When $\operatorname{sign}(\angle\vec{p}\vec{q}\vec{r})\operatorname{sign}(\angle\vec{p}\vec{q}\vec{s}) < 0$ and $\operatorname{sign}(\angle\vec{r}\vec{s}\vec{p})\operatorname{sign}(\angle\vec{r}\vec{s}\vec{q}) < 0$, the lines intersect.

When the translation would cause a topology violation, the operation must not be executed or the amount of translation must be reduced. This approach is especially suitable when the point to be moved and the magnitude of translation are determined by a randomized procedure, because convergence of the overall algorithm can still be ensured [Goudail & Réfrégier 04]. Instead of the bounding box, any other superset of arcs and lines can be used to speed up the intersection check. In pixel-based representations, a natural speed-up is to reject a modification if the new line would intersect or touch a pixel of an existing edge. This optimization has even proved useful in case of polygonal GeoMaps: Several kinds of queries (e.g. point-in-polygon queries) can be accelerated when a redundant pixel-based representation is kept in parallel to the polygonal representation. Another sufficient criterion for topology preservation can be derived when a triangulation of the points in a polygonal plane partition is available, for example when the GeoMap has been constructed by triangulation (see section 5.3). Then, every point is a corner of a number of triangles. The topology of the GeoMap will not be violated as long as the point is moved within the interior of the union of these triangles.

Examples for segmentation algorithms that have to check for topology preservation are *snakes* [Kass et al. 88] and their relatives. When a given contour is iteratively moved to optimally fit the given image data, topology violations have to be prohibited or must be handled by explicit modifications of the model (e.g. introduction of a new connected component). Since these explicit checks are rather expensive, many authors prefer not to fix the topology of the model during optimization. This is possible in a *level-set segmentation framework* [Osher & Paragios 03], where the topology is implicitly defined by the zero-contour of an auxiliary embedding function. The topology of the plane partition is then determined after convergence by simply thresholding the embedding function. In its original form, this approach is only applicable to binary plane partitions, but its extension to arbitrary non-binary topologies is an active area of research.

4.4.3 Interactive Segmentation in the GeoMap Framework

An over-segmentation of an image, encoded as a GeoMap, is a very good basis for interactive segmentation methods such as the *active paintbrush* [Maes 98] and *intelligent scissors / live-wire* [Mortensen & Barrett 98, Mortensen & Barrett 99]. In the active paintbrush approach, the oversegmentation of the scene is presented to the user. When the user

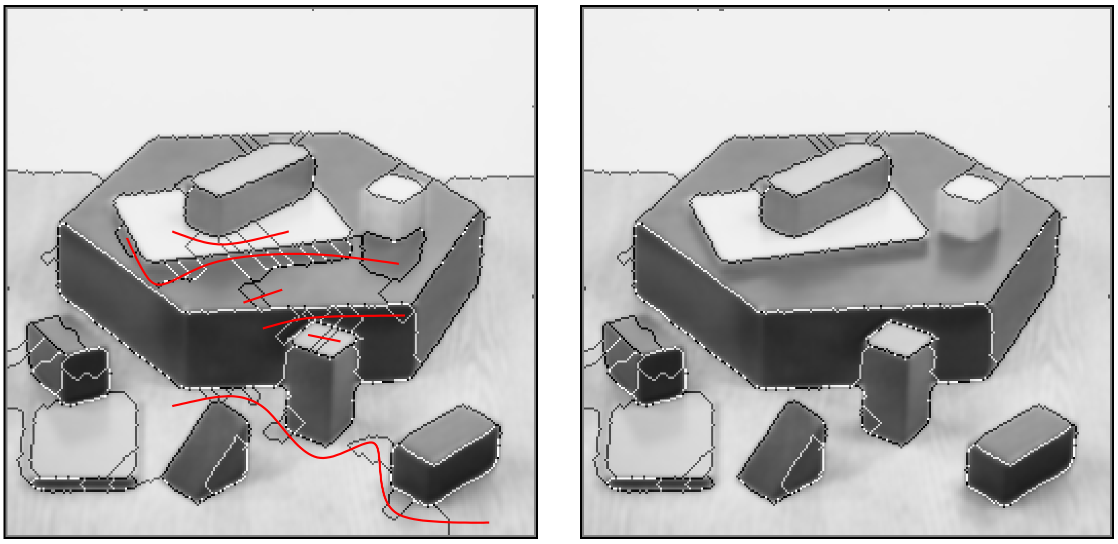


Figure 4.17: Interactive segmentation with the active paintbrush tool. Left: GeoMap with a number of false edge, which are "painted over" by the mouse cursor movements indicated by red lines. Right: Improved segmentation after removal of these edges. Illustration from [Meine & Köthe 05a].

crosses an edge with the mouse pointer while the mouse button is pressed, that edge is removed, and the adjacent regions are merged. In live-wire segmentation, the user selects an anchor point on a desired edge, and the system computes online the most salient edge between the anchor point and the current mouse position. By pressing the mouse button, the edge is finalized. Thus, relatively inaccurate mouse movements are sufficient to mark all important edges of the scene. Active paintbrush and live-wire segmentation are complementary in the sense that they allow relatively fast interactive segmentation by deleting and marking edges respectively.

To implement the active paintbrush tool by means of a GeoMap, it is only necessary to display the current GeoMap on the screen, and to provide an efficient mapping from screen coordinates onto darts:

```
dartCrossed: Line2D -> DartTraverser
```

The input `Line2D` is the result of a mouse movement on the screen. The function returns the first dart that is intersected by the straight line between two consecutive mouse positions. Figure 4.17 shows an example.

The implementation of intelligent scissors is more complicated because all edges must be augmented with a confidence that measures their salience. Given an anchor point and the current mouse position, the program automatically calculates the maximum salience path between these points by means of Dijkstra's algorithm. Intelligent scissors have originally been defined directly on the pixel grid [Mortensen & Barrett 98], but this implementation was too slow due to the large number of edges in the pixel adjacency graph. Therefore, the authors proposed an alternative algorithm in [Mortensen & Barrett 99]

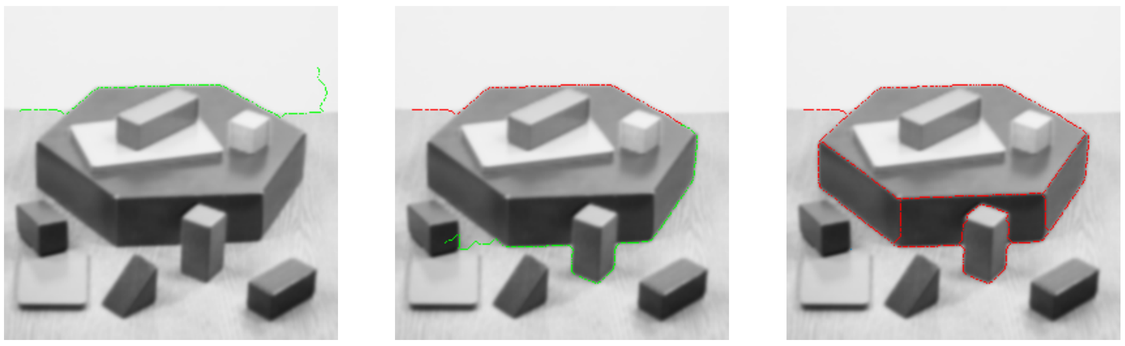


Figure 4.18: Interactive segmentation with intelligent scissors. The current path is drawn in green. Red paths have been finalized by double-clicking at a suitable end point. Illustration from [Meine 03].

which works on an edge graph derived from an oversegmentation of the image. Their so-called "tobogganing" algorithm for creating the initial oversegmentation is equivalent to the union-find watershed algorithm 5.8 to be introduced in section 5.2. Any other oversegmentation will do as well provided the true edges form a subset of the initial edge graph. To implement intelligent scissors on an arbitrary GeoMap, a function that maps cursor coordinates to the nearest node is required to determine the start and end point of the current path:

```
nearestNode: Point2D -> Node
```

In order to define edge cost measures, we need a method to obtain representative points along edges and within regions to compute relevant statistics:

```
scanEdge: Edge -> sequence_of(Point2D) // points along the edge
scanFace: Face -> sequence_of(Point2D) // points within a face
```

In a pixel based GeoMap representation, the points returned will usually be the pixel coordinates within these cells. In a polygonal GeoMap, the first function returns the knots of the edge's polygonal line, and the second a regular or randomized set of points in the region. The coordinates of these points can be used to collect region statistics (such as the average color) or edge statistics (such as the minimal gradient magnitude) from the corresponding locations in the original or gradient images. The edge cost may then be defined so that edges with high gradient and large color difference between the adjacent regions will be preferred. Figure 4.18 illustrates this approach using the same GeoMap as in figure 4.17.

Interactive segmentation on the basis of a subpixel-accurate initial over-segmentation is a big help for the definition of manual ground-truth, because these algorithms automatically take care of the geometric accuracy of the edges – the human operator only determines whether an edge is significant or not, which doesn't require very precise mouse positioning and is therefore quite fast. Geometric corrections are only necessary where the accuracy of the initial segmentation is unsatisfactory, e.g. near certain junctions.

5 Algorithms for GeoMap Creation

Abstract

If an ideal geometric image according to (2.1) were available to image analysis programs, GeoMap creation would be easy: We could simply place arcs along the lines of discontinuity, and vertices at the junctions of these lines. However, in practice we only have access to a blurred approximation of the ideal geometric image (e.g. a spline interpolation of the digital image, see section 3.3.1) which does not contain any discontinuities. Therefore, we must define boundaries by alternative criteria. Over the years, many possible criteria have been proposed. At the most fundamental level, these criteria can be classified into three groups: (i) criteria that apply to the analog reconstruction of the digital image, (ii) criteria that work directly on the digital image, and (iii) criteria that are based on a set of detected boundary points (and don't make direct use of the analog or digital image). In this chapter, we introduce important representatives for each of these approaches and present them in the unified GeoMap framework. Some of the algorithms considered are adaptations of well-known segmentation algorithms (e.g. the watershed transform and Canny's algorithm), while others are new. To illustrate the properties of these algorithms, we use the images in figure 5.1 as running examples.

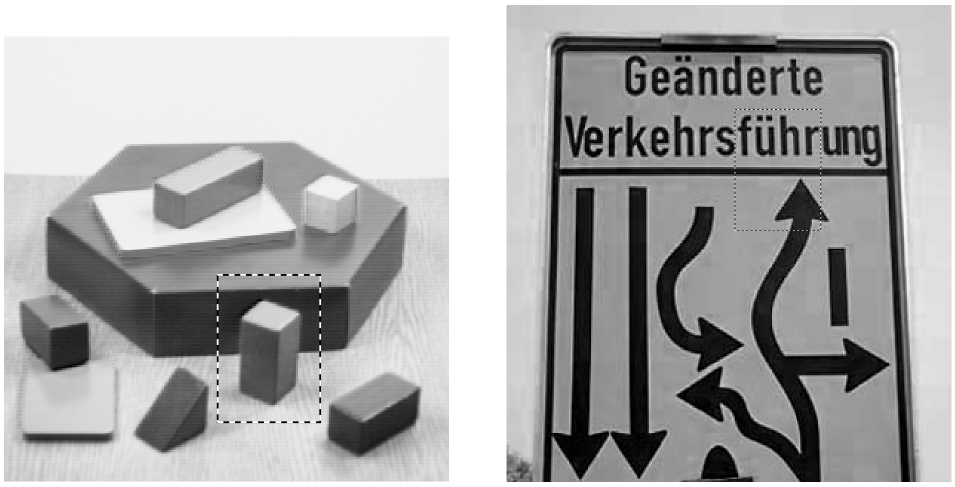


Figure 5.1: Two images ("blox" and "sign") that will be used as running examples in the present chapter. For the sake of clarity, these images are not too challenging, and experiments will be restricted to the regions shown.

5.1 Analog Boundary Definitions

After analog image reconstruction, we get a continuous image function, e.g. an interpolated spline surface. Our task is to recover the boundaries of the ideal geometric image from the reconstructed image. We had already seen in figure 1.1 and in chapter 3 that blurring of the ideal geometric image works to our advantage: It transforms geometric information (the location of a discontinuity) into intensity information (shades of gray that change according to the distance between a pixel and the discontinuity). In principle, the original discontinuities can be recovered with *subpixel accuracy* when the geometric information contained in the blurred intensities is put to good use.[1]

In order to recover boundaries from a *continuous* function, we must first define how boundaries are recognized, because the criterion of discontinuity is no longer applicable. When one looks at how boundary detection methods approach this problem, one finds that most of them are based on the notion of a *boundary indicator function*: The input image function is first transformed into a different function which simplifies the identification of boundaries. We have used the term "boundary indicator" (as opposed to "edge indicator" or "corner indicator") to emphasize the fact that, for the sake of topological consistency, we have to deal with all types of boundary features (i.e. edges, corners, junctions) simultaneously, and not just with edges or corners in isolation (cf. chapter 4).

There are many possibilities to create boundary indicator functions. But at their root, these methods rely on either of two fundamental principles:

Zero crossings: Boundaries are defined by the zero-level lines (or, by trivial intensity shifting, arbitrary level lines) of the boundary indicator function. Examples include thresholding, the Laplacian-of-Gaussian operator and level-set segmentation (where one looks for the zero crossings of an auxiliary distance function, cf. [Osher & Paragios 03]).

Ridges or valleys: Boundaries are defined as locations were the "boundary-ness" is higher (or, equivalently, the "non-boundary-ness" is lower) than in a suitably chosen neighborhood. Important examples are the image gradient magnitude and the SUSAN operator [Smith & Brady 97].

These principles are not completely unrelated: In some cases, it is possible to transform a ridge-based boundary indicator into a zero-crossing one by differentiating the boundary indicator along a suitably chosen local orientation. However, there are also genuinely ridge-based definitions, most importantly the *watershed ridges*, which cannot be reduced to zero-crossing-based descriptions [Koenderink & v. Doorn 93]. In order to objectively compare the quality of boundaries derived from alternative definitions we must implement these definitions with very high accuracy – we want to compare the true performance of

[1] It should be noted that *deconvolution* of the digital image (for example by means of Wiener filtering or iterative deconvolution, see e.g. [Wallace et al. 01]) improves the resolution of the digital image by shifting the effective band-limit to higher frequencies. But this will not eliminate the fundamental property of digitization that only a blurred version of the ideal geometric image can be reconstructed – one can reduce blurring, but never recover true discontinuities.

the models, not the artifacts created by suboptimal implementations. Therefore, we first concentrate on analog algorithms that indeed take advantage of the continuous nature of the boundary indicator. To this end, we assume that it is possible to access the values of the boundary indicator (and all required derivatives) at arbitrary real-valued coordinates. This is easily achieved by means of spline interpolation of a sampled boundary indicator, as explained in section 3.3.1. In our implementations, we mainly use splines of order 5 because they offer a very good trade-off between speed and accuracy.

The general approach to boundary detection in analog functions is *contour following with adaptive step-size control*. That is, starting at a point that is known to belong to the boundary, and a guess of the boundary's tangent, one iteratively detects new boundary points until a stopping criterion is met. This is repeated until no more starting points remain. During boundary tracing, the localization error is monitored and the step size adjusted so that a pre-defined error bound is not exceeded. We can therefore regard boundary tracing as a class of methods which *adaptively* sample the boundary at locations that are determined from the content of the data. This is in contrast to grid based methods, where the location of sampling points is determined by the sensor layout instead of the data content. Adaptive sampling can obviously achieve higher accuracy (significantly below pixel resolution). In addition, since boundary tracing methods work sequentially along a given contour, individual boundary points are automatically linked into arcs according to definition 2.2. To obtain a plane partition, it only remains to connect the arcs into a complete GeoMap as explained in section 4.3.1. Therefore, a separate edgel linking stage like in Canny's algorithm is not needed, and one source of topological inconsistencies is eliminated.

5.1.1 Subpixel-Accurate Tracing of the Zero-Contour

Suppose we are given a function $\psi(\vec{x})$ whose level-lines at level ψ_0 define the desired plane partition. Without loss of generality, we shall always assume $\psi_0 = 0$. The tangent unit vector $\vec{t}$ of a level-line is always perpendicular to the gradient direction: $\vec{t} = \nabla\psi^{\perp} / |\nabla\psi|$. Therefore, the points of a level-line can be traced by the following differential equation

$$\frac{\partial \vec{x}(\tau)}{\partial \tau} = \pm \vec{t}(\tau) = \pm \frac{\nabla\psi(\tau)^{\perp}}{|\nabla\psi(\tau)|} \tag{5.1}$$

with initial condition $\psi(\vec{x}_0) = 0$ and $\nabla\psi(\vec{x}_0) \neq 0$. The differential equation can be solved at all points where the gradient exists. Hence, the set $\{\vec{x} : \psi(\vec{x}) = 0\}$ must not contain plateaus (i.e. regions with constant intensity), i.e. it must be a set of measure zero. This condition is not a problem in practice, because the spline reconstruction of a smoothed, noisy image function has no regions with exactly constant intensity[2]. However, it is still possible for the set $\psi(\vec{x}) = 0$ to contain (isolated) saddle points where the contours will intersect. The tracing algorithm presented below is robust against this situation due to its inherent "inertia" (i.e. it traces the continuation with smallest change of tangent direction across the junction).

[2] In this context it is important to note that we always store intermediate results (e.g. images after preprocessing) with floating-point accuracy.

Suitable ψ-functions can be derived from the image in many ways. The most common are the *Laplacian* of the image (Marr-Hildreth operator [Marr 82])

$$\psi = f_{xx} + f_{yy} \tag{5.2}$$

and the second derivative along the gradient direction (Haralick operator [Haralick 84])

$$\psi = \frac{f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}}{f_x^2 + f_y^2} \tag{5.3}$$

The derivatives may either be computed directly from the spline representation of f (which is $(n-1)$-times continuously differentiable, with n denoting the spline order) or by means of discrete derivative filters followed by spline interpolation. Not all zero-crossings of these functions correspond to region boundaries, because both relative maxima and minima of the boundary strength give rise to zero-crossings. One selects the zero-crossings corresponding to maxima by additionally requiring the third derivative along the gradient to be negative

$$f_x^3 f_{xxx} + 3 f_x^2 f_y f_{xxy} + 3 f_x f_y^2 f_{xyy} + f_y^3 f_{yyy} < 0 \tag{5.4}$$

In addition, one usually requires the edge strength (gradient magnitude) to be above a threshold. The problem of threshold selection will be discussed in section 7.2.2.2.

The definitions above derive the ψ function directly from the original image. But in certain situations, only a boundary strength image is available, e.g. the result of the SUSAN operator [Smith & Brady 97]. Then, one can either apply a ridge-based boundary definition (see section 5.1.2), or one can transform the boundary strength image b into a zero-crossing image by means of the *height ridge* concept [Eberly 96]. Height ridges are the zero-crossings of the first derivative of b along the direction of maximum negative curvature:

$$\psi = \frac{\partial b}{\partial \vec{u}} \tag{5.5}$$

where $\vec{u}$ is the unit eigenvector associated with the small eigenvalue of the Hessian of b. However, this expression is insufficient for computing ψ because only the orientation of $\vec{u}$ is defined, not its direction. Thus, ψ is only determined up to its sign. Therefore, [Eberly 96] derives an alternative expression that has the same zero-crossings:

$$\psi = \nabla b^{\perp}\, \mathcal{H}b\, \nabla b$$

where $\mathcal{H}b$ denotes the Hessian of b. Again, an additional constraint is required to select relative maxima

$$\nabla b^{\perp} \mathcal{H}b\, \nabla b^{\perp} < \min\{0, \nabla b\, \mathcal{H}b\, \nabla b\}$$

or

$$\nabla \psi\, \mathcal{H}b\, \nabla \psi < 0$$

In practice, the second variant is numerically more stable.

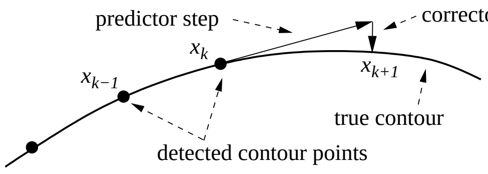


Figure 5.2: Principle of the predictor-corrector method for contour tracing (algorithm 5.1).

After computing ψ on a sufficiently dense grid[3], we represent it as a spline. If we use a 1st-order spline (i.e. linear interpolation), the equation $\psi(x, y) = 0$ can be solved analytically in every facet. Let the spline polynomial of the current facet be $f(u, v) = a\,u\,v + b\,u + c\,v + d$. Then the zero crossing curves are given by

$$\left(u + \frac{c}{a}\right)\left(v + \frac{b}{a}\right) = \frac{b\,c - a\,d}{a^2}$$

where u and v are restricted to the current facet, i.e. must be between zero and one. For higher order splines, the solution has to be found numerically. In principle, this can be done with standard methods (e.g. Runge-Kutta), but this is not the best approach, because it only uses the tangent direction but does not take advantage of the fact that the contour corresponds to a particular level-line, i.e. is constrained to a particular intensity. This constraint is used by a class of algorithms called *predictor-corrector methods* which significantly simplify level-line tracing. They use the current tangent (and possibly previous ones) to extrapolate a new candidate point of the current contour (predictor step), but these predictions need not be extremely accurate because the level-line constraint is subsequently used to adjust the new point's position until it lies exactly on the appropriate level-line (corrector step). see figure 5.2. In comparison to other methods, one can therefore use simpler predictors or larger steps. In addition, this approach allows to trace the contour over saddle points, where several level-lines cross: Thanks to its "inertia", the predictor will select an approximate continuation on the other side of the saddle, and the corrector will drag the candidate point onto the nearest actual continuation. The basic algorithm is as follows [Allgower & Georg 97]:

Algorithm 5.1: Predictor-Corrector Method for Contour Tracing

Input: a differentiable function $\psi(\vec{x})$ and a starting point $\vec{x}_0$ such that $\psi(\vec{x}_0) = 0$. Select an initial step size h and a bound ϵ_0 that specifies by how much $\psi(\vec{x})$ may deviate from the exact zero level along the contour.

1. While stopping criterion not fulfilled:

    a) Predict candidate point $\hat{\vec{x}}^{(0)}_{i+1} = h\,\vec{t}(\vec{x}_i)$ where $\vec{t}(\vec{x}_i) = \frac{\nabla\psi^{\perp}(\vec{x}_i)}{|\nabla\psi(\vec{x}_i)|}$ if $\vec{x}_i$ is not a saddle point of ψ, and $\vec{t}(\vec{x}_i) = \frac{\vec{x}_i - \vec{x}_{i-1}}{|\vec{x}_i - \vec{x}_{i-1}|}$ otherwise.

[3]Since the expressions for ψ involve products of band-limited functions, the band-limit of ψ is in general higher than that of the original image. To avoid aliasing, a denser grid is required to represent ψ, see section 7.1 for details

b) While $\left|\psi\left(\hat{\vec{x}}_{i+1}^{(k)}\right)\right| > \epsilon_0$:

i. Correct the candidate point by Newton iterations

$$\hat{\vec{x}}_{i+1}^{(k+1)} = \hat{\vec{x}}_{i+1}^{(k)} - \frac{\psi\left(\hat{\vec{x}}_{i+1}^{(k)}\right)}{\left|\nabla\psi\left(\hat{\vec{x}}_{i+1}^{(k)}\right)\right|^2}\nabla\psi\left(\hat{\vec{x}}_{i+1}^{(k)}\right)$$

c) If the total correction was small, accept $\hat{\vec{x}}_{i+1}^{(k+1)}$ as new point $\vec{x}_{i+1}$, set $i := i+1$, possibly increase h, and go to 2. Otherwise, reduce h and go to (a).

Several possibilities exist for the step size control in step (c). The simplest is to just bound the number of Newton iterations to 2 or 3: If more iterations are required, the step size must be reduced, if none or only one is needed, the step can be increased. More possibilities are described in the literature (e.g. [Allgower & Georg 97]). This article also describes more sophisticated predictor steps (e.g. by using several previous points on the contour) which serve to speed up processing by allowing larger step sizes. But in our application this is not necessarily an advantage, because we want to use the recovered contour polygons as approximations of the actual smooth contour. That is, we do not just require the points to lie on the contour, we also want their distance to be small enough so that the straight lines connecting consecutive knots stay close to the true contour. To ensure this in highly curved parts of the contour, the step size should not exceed 0.1 or 0.2 pixels. This small step size can already be achieved with the simplest predictor variant. The number of points in less curved parts of the contour can easily be reduced later by a standard polygon simplification algorithm (e.g. algorithm 4.6).

The stopping criterion is more problematic. Since level-lines form closed contours, one wants to stop the algorithm when it returns to the starting point. However, detecting this is not trivial because it is unlikely that the algorithm exactly hits the starting point again. Fortunately, there is a simple solution to this problem since $\psi(\vec{x})$ has been defined by spline interpolation. This solution also solves the problem of how to detect starting points. Consider the explicit polynomial representation (3.14) of a spline in a given facet and the locus of points where at least one of the local coordinates u or v is zero. These points form the *grid lines*, cf. definition 4.12. The squares enclosed by the grid lines are called the *dual pixels*. Now, by setting either u or v to zero, (3.14) simplifies to two 1-dimensional polynomials of order n in every facet. The roots of these polynomials can easily be computed by standard root finders. Each root that lies between 0 and 1 (if n is odd) or between $-1/2$ and $1/2$ (if n is even) corresponds to a point where the zero level-line crosses the corresponding grid line. This leads to the following algorithm:

Algorithm 5.2: Spline-Based Zero-Crossing Detection

Input: a spline function $\psi(\vec{x})$.

1. For every facet of the spline (i.e. every square $[i, i+1] \times [j, j+1]$ if the spline order is odd and every square $[i-0.5, i+0.5] \times [j-0.5, j+0.5]$ if the spline order is

even, where (i, j) are pixel coordinates, and $(u, v) = (x, y) - (i, j)$ are local facet coordinates, see section 3.3.1):

a) Detect all crossing points between the zero level-lines of ψ and the grid lines in the current facet:

$$\begin{aligned}\text{crossings}_{ij} \quad = \quad & \{(i+u, j) \mid f_{ij}(u, 0) = 0 \wedge u \in \text{facet}\} \\ & \cup \{(i, j+v) \mid f_{ij}(0, v) = 0 \wedge v \in \text{facet}\}\end{aligned}$$

where $f_{ij}(u, v)$ is the spline polygon in facet (i, j) according to (3.14), and $u, v \in$ facet means $u, v \in [0, 1]$ if the spline order is odd and $u, v \in [-0.5, 0.5]$ if the spline order is even.

2. For each dual pixel (i.e. each square $[i, i+1] \times [j, j+1]$):

   a) Determine whether the border of the current dual pixel is crossed by level lines by selecting the appropriate members of crossing_{ij}, $\text{crossing}_{(i+1)j}$, $\text{crossing}_{i(j+1)}$, and $\text{crossing}_{(i+1)(j+1)}$. If there are no crossings, skip the current dual pixel.

   b) Select one of the crossings as starting point and trace the corresponding level-line by means of the predictor-corrector method (algorithm 5.1) until it leaves the current dual pixel at another of the known crossings. Repeat this until all crossings on this dual pixel's border have been processed (as either start or end points).

A variant of this algorithm can also be used to replace contour tracing (algorithm 5.1) entirely by recursive subdivision: Whenever a dual pixel containing part of the level-line is detected, it is sub-divided into four new squares whose borders are checked for crossing points in the same way (i.e. by setting u and v equal to 1/2 instead of 0). Subdivisions are repeated recursively until the squares are so small that the crossing points at their borders may simply be connected with straight lines. For example, four subdivisions give an accuracy below 0.1 pixels, which should be sufficient for most applications.

Finally, we have to connect the line segments into a GeoMap. To do so, we first connect the contour pieces from individual dual pixels into complete connected contours. This is easy because we explicitly know the start and end points of all pieces, thanks to algorithm 5.2. It remains to identify the vertices. Since edges derived from zero-crossings always form closed contours, with possible self-intersections at saddle points, there are two kinds of vertices: If the curve self-intersects, the intersection point is always a vertex. If the curve does not self-intersect, an arbitrary point on the curve has to be selected as a vertex, which serves as both start and end point of the present curve (recall that the start and end points of an arc must not belong to the arc). If a curve leaves the image domain, a vertex is placed at that point as well. Due to constraint (5.4) which identifies zero-crossings corresponding to maxima of the boundary strength (or any other constraint), parts of the initial contour set will be deleted, resulting in new vertices at the terminations of the surviving contour segments. The maximally connected contour segments between vertices then become the arcs of the GeoMap.

Algorithm 5.3: Zero-Crossing GeoMap

Input: boundary segments according to algorithm 5.2 above.

1. Connect the contour pieces across the dual pixels' borders at the known crossing points.
2. Remove all contour parts that violate additional constraints, e.g equation (5.4).
3. Define vertices:
   a) Locate self-intersections and terminations and define vertices there.
   b) Identify closed contours without self-intersections and select an arbitrary point as vertex.
4. Define arcs as maximal connected contour pieces between vertices. Define darts as oriented arcs, and pair them into α-orbits.
5. At every vertex, sort the outgoing darts according to their angle of incidence to define the σ-orbits. Define faces according to the relation $\phi = \sigma^{-1}\alpha$.
6. Every connected component of the resulting graph is a combinatorial map.
   a) Identify the exterior face of every map: Perform contour following along each ϕ-orbit and compute the enclosed area according to the formula
   $$A = \frac{1}{2}\sum_i (x_i y_{i+1} - y_i x_{i+1})$$
   where (x_i, y_i) are the knots of the polygons representing the arcs of the current ϕ-orbit. If $A \leq 0$, the ϕ-orbit is an exterior face.
   b) Build the contains relation: Select an arbitrary point from every exterior ϕ-orbit. Use a standard polygon containment algorithm [Heckbert 94] to determine the ϕ-orbit that contains the point.

In practice, on can further speed-up step 6 by using the knowledge about which pixels are intersected by the boundary, but we won't go into detail here. The result of the algorithm is a GeoMap, where the topological relations have been made explicit via the α-, σ-, and ϕ-orbits and the exterior and contains relations. The geometric structure of the partition is defined by the coordinates of the vertices and by the polygonal lines forming the arcs. Figure 5.3 shows results of this algorithm.

The big advantage of zero-crossing based contours is the existence of a local criterion that tells us whether or not a point belongs to the contour. This opens up the possibility to replace contour following with recursive subdivision algorithms, e.g. [Bertram et al. 00, Stolte 05]. Subdivision methods are readily generalized to three dimensions, whereas contour following is not: since contour following requires a linear order of points, it can only be applied to linear structures, not to surfaces. While there exist generalizations

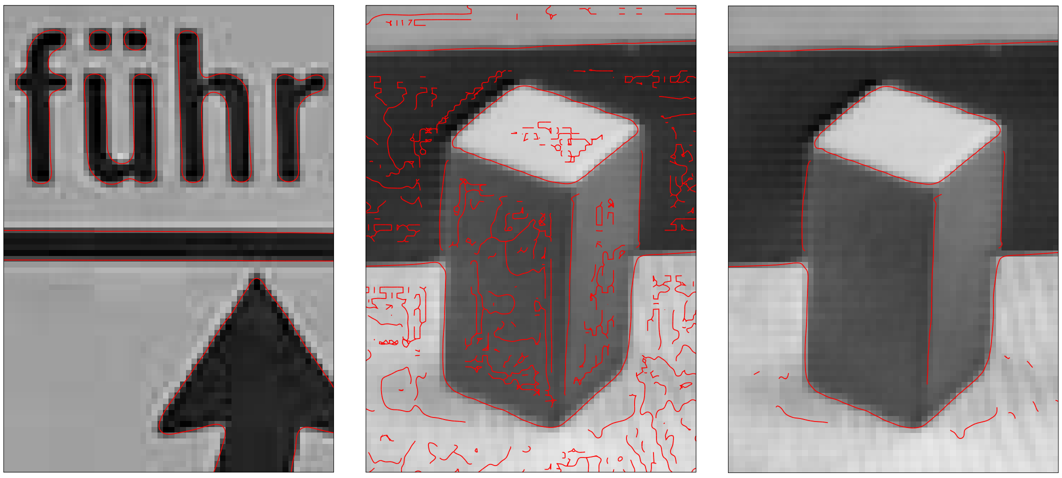


Figure 5.3: Left: Result of subpixel thresholding (threshold = 100) for the "sign" example. Center and right: Result of the subpixel Haralick edge detector without (center) and with (right) thresholding on the edge strength. That is, the center image contains all zero-crossings according to (5.3) that are indeed gradient maxima according to (5.4), whereas the right one is restricted to those where the gradient magnitude is above 32. Note that the boundary in the "blox" examples has gaps near junctions, which is a well-known property of Haralick's algorithm. Filter scale was $\sigma_{\text{filter}} = 1.2$.

of predictor-corrector methods to higher dimensions, e.g. [Brodzik 98, Henderson 02], subdivision methods seem to be much more popular in this context due to their simplicity.

On the other hand, zero-crossing contours also have a big disadvantage: Contour junctions can only occur at saddle points which are exactly at level zero (i.e. $\psi(x, y) = 0$ and $\nabla\psi(x, y) = 0$ must hold simultaneously within the accuracy of the algorithm), so junctions will be very rare and unstable. Therefore, it is not possible to represent arbitrary plane partitions in terms of zero contours. The method is essentially restricted to binary partitions (i.e. foreground and background only). Two solutions to this problem have been proposed: First, one can define several boundary indicator functions and build the union of their zero-contours. This method has been especially popular with variational segmentation in the level-set framework [Chan & Vese 01, Vese & Chan 02]. The other possibility is based on the observation that humans already perceive junctions when two contours or a contour and a termination are very close together, without actually intersecting each other. Algorithms that close these gaps have been proposed by [Beymer 91, Rothwell et al. 95, Ren et al. 05a]. But these algorithms contain several heuristics, and it is not clear whether or when they will result in well-defined and reliable solutions. We will come back to this problem in section 5.3.

5.1.2 Subpixel-Accurate Watershed Tracing

Many boundary indicators are based on the idea that boundaries are located at positions where some measure of boundary strength assumes a relative maximum. In the previous section, we showed that these boundary indicators can be transformed into descriptions in terms of level lines by means of the height ridge concept. Alternatively, one can detect the ridges constituting boundaries directly in the boundary strength function. A very useful method to do this is the *watershed algorithm*. Intuitively, watersheds are the locus of points where the path of a drop of water that runs downwards along the direction of steepest decent is not uniquely determined. Instead, the drop may end up in different valleys (also called catchment basins in this context), depending on arbitrary small fluctuations along the path. Watershed methods constitute a unique approach to boundary detection because watershed boundaries cannot be reduced to descriptions in terms of level lines [Koenderink & v. Doorn 93]. In general, no local criterion exists to decide whether a particular point (x, y) belongs to a watershed. Watersheds and height ridges coincide if and only if the ridge is a straight line. Otherwise, they may still be quite similar, but [Koenderink & v. Doorn 93] also show examples were the difference is big, and watershed ridges appear to be intuitively more appealing. The exact definition of the watersheds of a continuous function dates back to the 19$^{\text{th}}$ century works of A. Cayley and J.C. Maxwell [Cayley 1859, Maxwell 1870]:

Definition 5.1. *A* watershed *is a line of steepest ascent running from a saddle point to a local maximum.*

This definition requires critical points (saddles and extrema) to be isolated points, and a unique line of steepest ascent to exist in every non-critical point. This requirement is always fulfilled if the boundary indicator function b is a *Morse function*:

Definition 5.2. *A function* $b : \mathbb{R}^2 \to \mathbb{R}$ *is called a* Morse function *if it is twice differentiable, and the Hessian matrix has full rank at all* critical points *(minima, maxima, saddles), i.e. at all points where the gradient vanishes.*

Some authors also require the critical levels (i.e. the function values at the critical points) to be pairwise distinct, but we don't need this condition for our purposes. Non-degeneracy of the Hessian ensures that the critical points are isolated, i.e. that no plateaus and no horizontal ridges or valleys exist. In addition, there are no higher-order saddles (e.g. monkey saddles). Consequently, exactly two watersheds start at every saddle point, running in opposite directions along the eigenvector corresponding to the Hessian's positive eigenvalue.

Requiring boundary indicators to be Morse functions is not as big a constraint as it may seem because the set of Morse functions is dense in the space of differentiable functions. That is, to every non-Morse function there exists a Morse function with arbitrary small difference (in the L_2-norm sense). Moreover, it can be shown that plateaus (extended regions with degenerate Hessian) cannot occur in band-limited functions. Using spline interpolation, we have never encountered problems with the Morse function requirements in natural images, because noise, shading, and relatively complicated objects shapes make it very unlikely that a point has simultaneously a zero gradient and a degenerated Hessian. Problems occurred only in artificial test images consisting of highly regular objects without noise. We have found that the best way to turn a degenerate function into a Morse function is to add a very small amount of noise (with SNR $= 100 \ldots 200$). .

The watersheds define a plane partition whose vertices are the maxima and saddle points of the boundary indicator function b. The arcs connecting those vertices can be determined by means of (inverse) *flowline tracing*, i.e. by finding the curves that solve the differential equation (cf. [Najman & Schmitt 94])

$$\frac{\partial \vec{x}(\tau)}{\partial \tau} = \vec{t}(\tau) \qquad \vec{t}(\tau) = \frac{\nabla b(\vec{x}(\tau))}{|\nabla b(\vec{x}(\tau))|} \qquad (\tau > 0) \tag{5.6}$$

with initial condition

$$\vec{x}(0) = \vec{s}_k, \quad \vec{t}(0) = \pm \vec{u}_k$$

where $\vec{s}_k$ is the k^{th} saddle point and $\vec{u}_k$ is the eigenvector of the Hessian matrix corresponding to the positive eigenvalue at $\vec{s}_k$. The choice of $\pm\vec{u}_k$ as initial direction can be substantiated as follows: The gradient at the saddle point itself is zero, so it contains no information about the watershed direction. But in an infinitesimal neighborhood of the saddle, the shape of b is completely defined by a second-order Taylor polynomial. Therefore, the watershed is locally a straight line and coincides with the height ridge. The height ridge tangent at a saddle is equal to the eigenvector $\vec{u}$ which is thus the correct initial direction.

The flowline must be traced upwards because the solution of (5.6) is only stable in the direction from saddles to maxima: All flowlines in the neighborhood of the watershed converge to the same maximum, thus small tracing errors will not accumulate. In the opposite direction, neighboring flowlines do not converge to a saddle but to a minimum, so we would quickly loose track of the watershed.

Another nice feature of the algorithm is the possibility to check a predicate at each saddle point before flow-line tracing starts. In this way, weak boundaries can be recognized beforehand, and the effort of tracing can be saved. For example, we often use a threshold on the gradient magnitude to immediately discard boundaries that are likely caused by noise only. The selection of appropriate thresholds will be discussed in section 7.2.2.2.

Using splines, [Steger 99] was the first to demonstrate that the analog watershed algorithm can actually be implemented so that it runs reasonably fast on 2-dimensional array data (digital elevation models). When we adapted his algorithm to image segmentation, we found that his method for the identification of critical points did frequently miss some of them. Therefore, we replaced critical point detection with algorithm 3.1 described in section 3.3.2. Moreover, we found that a second-order Runge-Kutta procedure was sufficient to solve the differential equation (5.6). In comparison to a fifth-order algorithm, it traces the watersheds with smaller steps. This is desirable because many subsequent image analysis algorithms need very accurate polygonal approximations of the edge, where step sizes should not be larger than about 0.1 to 0.2 pixels. These step sizes can already be achieved with the second-order Runge-Kutta method. Subsequent polygon simplification can be applied when lower accuracy is sufficient in a particular application. Thus, our analog watershed algorithm works as follows [Meine & Köthe 05b]:

Algorithm 5.4: Sub-Pixel Watershed Algorithm

Input: The spline reconstruction of a boundary indicator function $b(\vec{x})$. Algorithm parameters: default step size h_0, desired localization accuracy ϵ_s

1. Use algorithm 3.1 to find all saddles and maxima.
2. For every saddle $\vec{s}$ (optionally only for those that fulfill some application specific predicate, such as sufficient edge strength):
   a) Define the starting point $\vec{x}_0 = \vec{s}$, the starting direction $\vec{t}_0 = \vec{u}$ as the eigenvector of the Hessian matrix corresponding to the positive eigenvalue at $\vec{s}$, and initial step step size $h = h_0$.
   b) While the distance of the current point to the nearest maximum is above h_0, i.e $\min_{\vec{m} \in \text{Maxima}} |\vec{x}_k - \vec{m}| > h_0$:
      i. Compute the candidate point $\vec{x}^{(1)}_{k+1}$ by a single Runge-Kutta step with step size h, and $\vec{x}^{(2)}_{k+1}$ by two consecutive Runge-Kutta steps with step size $h/2$, where a Runge-Kutta step of order 2 and step-size h is defined as
      $$\vec{x}\,' = \vec{x}_k + \frac{h}{2}\,\vec{t}_k, \qquad \vec{t}\,' = \left.\frac{\nabla b}{|\nabla b|}\right|_{\vec{x}=\vec{x}\,'}$$
      $$\vec{x}\,'' = \vec{x}_k + h\,\vec{t}\,', \qquad \vec{t}\,'' = \left.\frac{\nabla b}{|\nabla b|}\right|_{\vec{x}=\vec{x}\,''}$$
      ii. Estimate the step size that would have caused a localization error of at most ϵ_s in the current step
      $$h' = h\left(\frac{\epsilon_s}{\left|\vec{x}^{(1)}_{k+1} - \vec{x}^{(2)}_{k+1}\right|}\right)^{\frac{1}{3}}$$
      iii. If $h' \geq h/2$ accept $\vec{x}^{(2)}_{k+1}$ as new point $\vec{x}_{k+1}$, set $h = h'$ and goto (b). Otherwise retry the current step with $h = h'$.
   c) Repeat the same procedure for the starting point $\vec{x}_0 = \vec{s}$, but with starting direction $\vec{t}_0 = -\vec{u}$

In our experiments, we used $h_0 = 0.1$ and $\epsilon_s = 10^{-4}$. Larger values for ϵ_s cannot be used because tracing may then loose the watershed line. Smaller values wouldn't make sense either due to the noise in the original image data. The average step size selected by the step size control mechanism was about 0.05 pixels. Figure 5.4 illustrates the principle of the algorithm.

The result of this algorithm is a set of vertices and arcs, but the arcs must still be linked into a GeoMap. At a saddle point, this is trivial: the two arcs meeting there are joined into a single edge, and the two corresponding darts are obtained by tracing

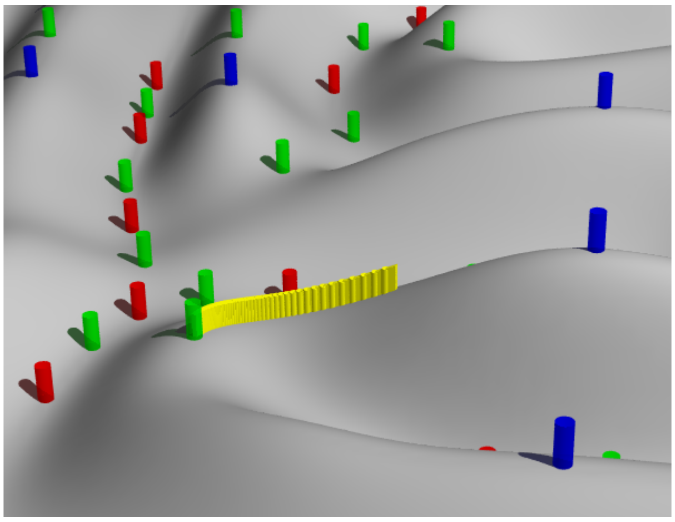


Figure 5.4: The principle of the subpixel watershed algorithm: The boundary strength function is depicted as a landscape, and the critical points are marked by cylinders: minima (red), maxima (blue), and saddle points (green). Starting from a saddle, watersheds are traced upwards along a flow-line until a minimum is reached (yellow). It can be seen how the step size of the Runge-Kutta algorithm varies according to the local properties of the boundary strength function. Image courtesy of Hans Meine.

the edge in either forward or backward direction. Recovering the σ-orbits at junctions is more difficult. In theory, one only needs to sort incoming arcs according to their tangent angles. However, in practice this is not possible because it would require infinite accuracy. This can be understood by observing that watersheds converge tangentially towards the maximum, cf. fig. 5.5. This is not an artifact of our implementation, but a well-known watershed property [Steger 99]: The surface in an infinitesimal neighborhood of a maximum can be described by $-\mu_1 u^2 - \mu_2 v^2$ with $\mu_2 \geq \mu_1 > 0$ and (u, v)-coordinates along the eigenvector directions, and it can be shown analytically that all flowlines (except the one parallel to $u = 0$) converge to the maximum tangentially to the u-direction [Koenderink & v. Doorn 93]. If the angle differences between incoming watersheds at distance h_0 from the maximum is too small, we cannot readily compute the correct σ-order. Due to the finite accuracy of flowline tracing, the computed flowlines may cross each other in practice (although this is not possible in theory), so the order derived from small angle differences can be incorrect. Therefore, we must follow parallel flowlines until they eventually diverge. The following algorithm recovers the correct σ-ordering at each vertex:

Algorithm 5.5: Watershed σ-Ordering

Initialization: To recover the σ-order at junction k located at $\vec{p}_k$: Set the reference point $\vec{p}_{\mathrm{ref}} = \vec{p}_k$, reference angle $\varphi_{\mathrm{ref}} = 0\,\mathrm{rad}$. Let g be the (unordered) set of half-edges starting at $\vec{p}_k$.

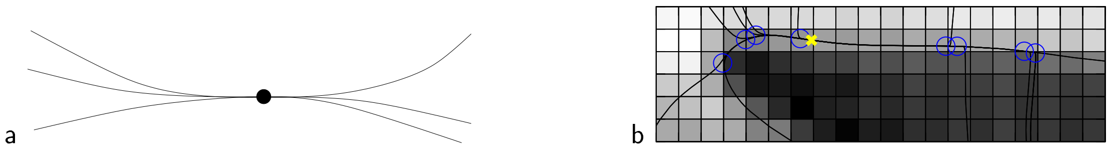


Figure 5.5: Tangential watershed convergence at a local maximum. Top: illustration; bottom: actual watersheds on an image – the single maximum of the boundary indicator in the ROI is marked yellow, and the blue circles mark locations where the half-edges converge (precisely, the r-circles where the σ-order is eventually found).

1. For each half-edge i in g find the intersection $\vec{p}_i$ of the corresponding polygonal arc with an r-circle around $\vec{p}_{\text{ref}}$ (we use $r = 0.5$). If there are several intersections, select the one whose arc length distance from the half-edge's start is maximum. If there is no intersection, use the half-edge's end point.

2. For each half-edge i in g calculate the angle φ_i between the vector $\vec{p}_i - \vec{p}_{\text{ref}}$ and φ_0, measured in the interval $-\pi < \varphi_i \leq \pi$. Sort g according to φ_i.

3. Detect tangential half-edges: Compute the difference angles $\Delta\varphi_i = \varphi_{i+1} - \varphi_i$. If $\Delta\varphi_i < \varphi_{\text{tang}} = 0.5\,\text{rad}$, half-edges i and $i+1$ are considered tangential. Group the half-edges of g into groups $\hat{g}_m$ such that each group contains a maximal sequence of tangential half-edges. If all groups have only one member, there are no tangential half-edges, and the algorithm stops. Otherwise repeat the procedure for each group with several members: Compute a new reference point $\hat{\vec{p}}_{\text{ref}}$ and reference angle $\hat{\varphi}_0$ as the average of the current group's members, and goto 1. (The choice of r and φ_{tang} is not critical.)

The points where converging watersheds first meet don't correspond to local maxima (hence are not represented by nodes of the GeoMap), yet they are clearly perceived as T-junctions by a human observer, see figure 5.5b. Therefore, we insert additional GeoMap nodes at these points.

Finally, the exterior and contains relations of the GeoMap are constructed like in algorithm 5.3 on page 138.[4] The advantage of the watershed algorithm is that it can naturally find junctions: Arbitrary many watershed lines can meet at a maximum, and any desired plane partition can be expressed in terms of the watersheds of a suitably defined boundary indicator function. Figure 5.6 shows results of the subpixel watershed algorithm.

It is also instructive to look at watersheds from an alternative point of view that was pioneered by [Nguyen et al. 03]. Consider the *topographic distance* between two points

$$d_{\text{topo}}(\vec{x}_1, \vec{x}_2) = \min_C \int_C \left|\nabla b(\vec{x}')\right| \, d\vec{x}'$$

where the minimum is taken over all possible paths C that connect $\vec{x}_1$ and $\vec{x}_2$. When the two points are on the same flowline, this flowline is the optimal path, and the topographic

[4]In practice, disconnected watershed graphs are encountered very rarely because they occur only under quite exceptional circumstances, see [Nackman 84, Rieger 97].

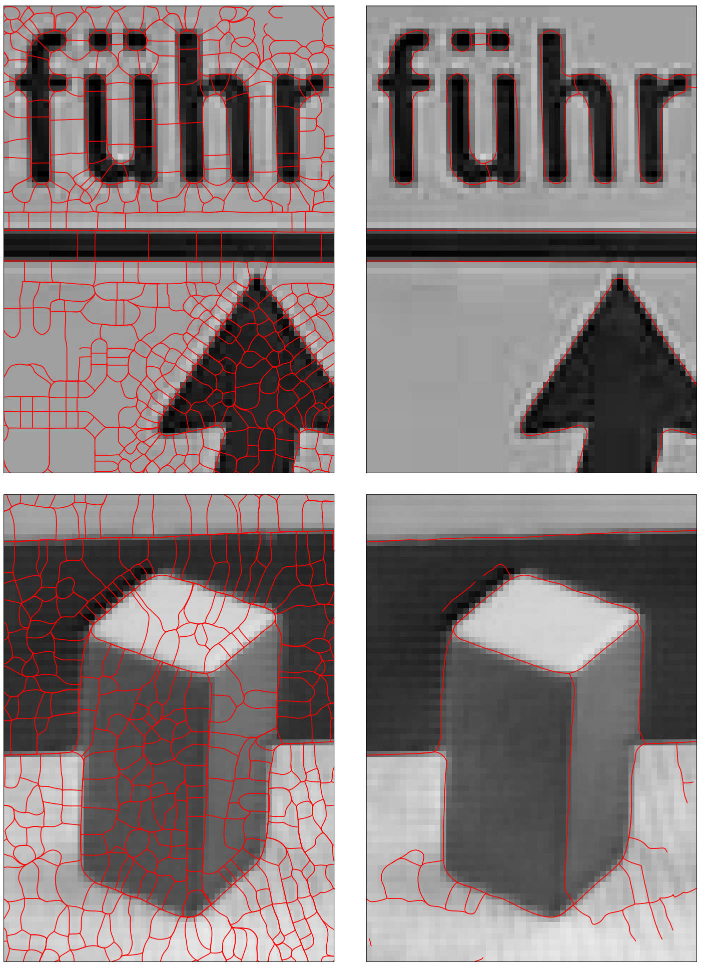


Figure 5.6: Result of the subpixel watershed algorithm 5.4 without (left) and with additional thresholding on the boundary strength. That is, the left images contain all watersheds, whereas the right ones only those where the gradient magnitude at the saddle point (i.e. the tracing start point) exceeds a value of 16 ("sign" example) or 4 ("blox" example). Without the threshold, we see the well-known oversegmentation of the watershed algorithm. The gradient filter scale was $\sigma_{\text{filter}} = 1$.

distance between $\vec{x}_1$ and $\vec{x}_2$ is equal to their absolute height difference:

$$d_{\text{topo}}(\vec{x}_1, \vec{x}_2) = |b(\vec{x}_1) - b(\vec{x}_2)|$$

Now consider a boundary indicator where all local minima have the same function value. This can always be achieved by changing a given boundary indicator so that the function surface in an arbitrary small neighborhood around each minimum is warped (like a rubber membrane) to the desired value. When done properly, this does not change the number, type, and location of the critical points, and the watersheds will remain unaltered. The functional

$$E_{\text{Watershed}}[b(\vec{x})] = \iint \min_{\vec{y} \in \text{Minima[b]}} d_{\text{topo}}(\vec{x}, \vec{y})\, d\vec{x} \tag{5.7}$$

is minimized when every point $\vec{x}$ is assigned to its nearest (in the sense of the topographic distance) minimum. The set of points assigned to the same minimum is called a “dale” or a “catchment basin”. It has been shown by [Nguyen et al. 03] that the watersheds are exactly the locus of points that have the same distance from more than one minimum.

This formulation of the watershed transform justifies the well-known flooding algorithm discussed in the next section. The watershed functional (5.7) is formally equivalent to a Voronoi partition where the Euclidean distance has been replaced with the topographic distance. In the same sense that pixels on a grid are defined as Voronoi regions around the sampling points (see definition 4.9), we may interpret the catchment basins of the watershed functional, i.e. the Voronoi regions according to topographic distance, as “superpixels”. In contrast to normal pixels, which are imposed on the image by the image acquisition device, superpixels arise in a natural way from the structure of the image itself.

5.2 Grid-Based Boundary Definitions

In image analysis, the original image data are given on a regular grid, and it is natural to take advantage of the special structure of the grid to simplify the representation and creation of GeoMaps. Due to the higher speed of these computations, one often tolerates the inevitable loss of accuracy relative to the analog methods described in the previous section. In section 4.3.2, we introduced two basic edge representations for raster-based GeoMap creation: well-composed crack edges and thin 8-connected boundaries (including mid-crack edges). A large part of the underlying algorithmic aspects have already been treated there in a general way. Here, we add the missing links between grid-based GeoMaps and various boundary indicator types.

From a topological point of view, the interpixel approach is the most popular because it is easy to implement and works in arbitrary dimensions [Ahronovitz et al. 95, Winter 95, Braquelaire & Brun 98, Braquelaire & Domenger 99, Damiand et al. 04]. A good review of the related theory and applications can be found in [Braquelaire 05]. It is straightforward to derive a GeoMap with inter-pixel edges from any 4-connected region image by means of the crack-insertion algorithm 4.1. This even works when the boundary is simply defined as a level-line of the image function, i.e. by thresholding:

Algorithm 5.6: Crack Insertion by Thresholding

Input: an image such that foreground and background can be distinguished by comparison with a threshold.

1. Use the threshold to create a binary image and label both the background and foreground according to 4-connectivity.

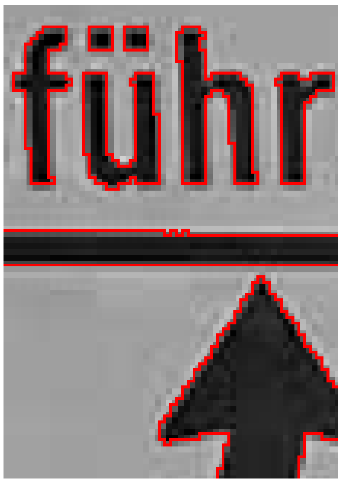


Figure 5.7: Pixel-accurate thresholding of the "sign" example using the crack-insertion algorithm 5.6 with threshold = 100. Compare with the subpixel result in figure 5.3.

2. Apply the crack insertion algorithm 4.1 to the labeled image and create a GeoMap as described in section 4.3.2.

An example is shown in figure 5.7.

A variant of this algorithm is required when the threshold is combined with an additional predicate that decides whether a particular transition between foreground and background is indeed significant. This applies, for example, to the pixel-accurate version of Haralick's edge detector, were the boundary indicator is the oriented second derivative of the image, but zero-crossings of this derivative must only be interpreted as edges (i.e. as local maxima of the contrast) when the third derivative is negative at this location, cf. equations (5.3) and (5.4). The modified crack insertion algorithm is as follows:

Algorithm 5.7: Crack Insertion with Constraint

Input: a discrete region image I_{zeros} on the integer domain $[0, ..., w-1] \times [0, ..., h-1]$, whose zero-crossings signal potential edges, and a predicate that decides for every zero-crossing whether it is significant.

1. Create an image I_{Crack} on the integer domain $[-1, ..., 2w-1] \times [-1, ..., 2h-1]$ and initially mark all pixels as "region".
   a) For all points of the form $I_{\text{Crack}}(2i+1, 2j)$: If there is a zero-crossing between the points $I_{\text{zeros}}(i, j)$ and $I_{\text{zeros}}(i+1, j)$ and the predicate is true for this transition, or one of the two points is outside the image, label $I_{\text{Crack}}(2i+1, 2j)$ as a boundary point.
   b) For all points of the form $I_{\text{Crack}}(2i, 2j+1)$: If there is a zero-crossing between the points $I_{\text{zeros}}(i, j)$ and $I_{\text{zeros}}(i, j+1)$ and the predicate is true for this transition, or one of the two points is outside the image, label $I_{\text{Crack}}(2i, 2j+1)$ as a boundary point.

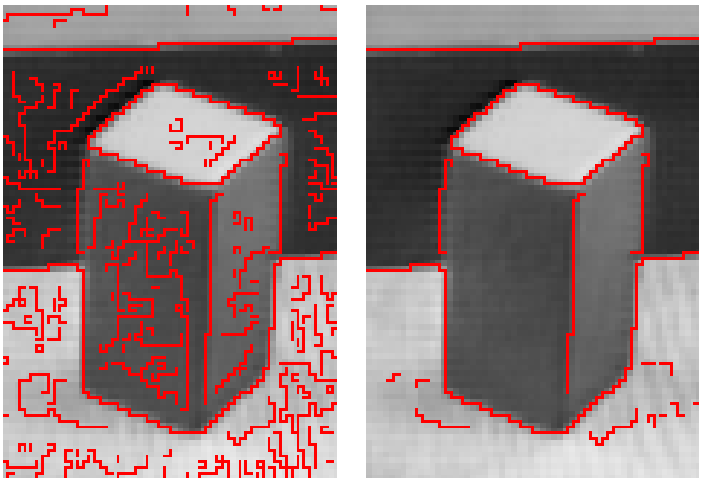


Figure 5.8: Pixel-accurate Haralick edges for the "blox" example using algorithm 5.7 without (left) and with gradient strength threshold (right, threshold = 32), for $\sigma_{\text{filter}} = 1$. Compare with the subpixel results in figure 5.3.

c) For all points of the form $I_{\text{Crack}}(2i+1, 2j+1)$: If any of the direct neighbors $I_{\text{Crack}}(2i, 2j+1)$, $I_{\text{Crack}}(2i, 2j-1)$, $I_{\text{Crack}}(2i+1, 2j)$, $I_{\text{Crack}}(2i-1, 2j)$ (where points outside the image are ignored) is marked as boundary, mark $I_{\text{Crack}}(2i+1, 2j+1)$ as a boundary point as well.

2. Create a GeoMap from the resulting boundary image.

An alternative way for computing labeled images suitable for GeoMap creation is by means of the discrete watershed transform. There are many variants of this algorithm, for example the popular flooding algorithm by [Vincent & Soille 91] which produces a 4-connected region image that can be further processed by the standard crack insertion algorithm. However, the so-called *union-find algorithm* [Roerdink & Meijster 00] is even more efficient, and it also helps to understand how discrete watershed transforms relate to analog ones. In the context of live-wire segmentation, the union-find algorithm is also known as *tobogganing* [Mortensen & Barrett 99]. Catchment basins can be determined by looking at where the downward flowline from each point ends. When the associated minimum of every sampling point is known, the discrete watershed transform is uniquely defined: All flowlines ending in the same point form a tree, and an image partition is obtained by simply assigning a unique label to the points in each tree. However, it is very expensive to follow each flowline exactly (i.e. with subpixel accuracy) until it converges. Therefore, one approximates exact flowline tracing by a discrete recursive procedure: It is assumed that each sampling point is associated to the same minimum as its lowest 4-neighbor. This leads to the following algorithm:

Algorithm 5.8: Union-find Algorithm for Watershed Detection

Input: A boundary indicator image whose watersheds are to be interpreted as boundaries.

1. For every pixel: Determine the 4-neighbor with lowest edge strength. If the strength of the current center is higher, remember the direction to that neighbor, otherwise mark the center as a local minimum.

2. Perform 4-connected components labeling on the thus marked image, where the equivalence classes are defined as follows: Two neighboring pixels belong to the same region when either is the lowest neighbor of the other.

3. Proceed on the resulting label image with the standard crack insertion algorithm. Optionally, apply thinning to the resulting crack edges to obtain a mid-crack representation.

Results can be seen in figure 5.9.

Since we used the 4-neighborhood in the definition of the flowlines, the algorithm indeed results in a valid GeoMap whose boundaries correspond to the interpixel or mid-crack boundary of the labeled image. The discrete GeoMap is topologically equivalent to a discretization of the corresponding analog watershed partition when two requirements are met:

1. There must be a 1-to-1 mapping between the root points of the trees and the true minima of the boundary indicator (as determined by critical point detection in the spline-interpolated indicator function).

2. The assumption that the lowest neighbor belongs to the same catchment basin must be true.

It is easy to see that both requirements can be violated: The roots of a flowline tree are always local minima in the 4-neighbor sense. We have shown in section 3.3.2 that the 4-neighborhood maxima can deviate dramatically from the true maxima of the continuous reconstruction of the same image. The same is true for 4-neighborhood minima as needed here. A condition for spurious minima (and therefore spurious regions) to occur follows from the geometric sampling theorem 6.2 to be discussed in section 6.1.1: sampled regions may become disconnected when the curvature of the true boundary exceeds $1/r$, where r is the pixel radius (i.e. $r = \sqrt{2}/2$ in a square grid with pixel distance 1). Therefore, if the level-line of b going through sampling point $\vec{p}$ has a smaller curvature radius than $\sqrt{2}/2$, it may happen that no point in the *4-neighborhood* of $\vec{p}$ lies in the lower subset of the *Euclidean neighborhood* of $\vec{p}$, see figure 5.10a. Then $\vec{p}$ is mistakenly classified as a minimum, and we end up with an extra root node in the watershed forest, i.e. a spurious region. Another error can happen when a sampling point is near a true watershed. Then its lowest 4-neighbor can actually lie in a different catchment basin, so that an entire subtree may become wrongly connected, figure 5.10b. This is also more likely to happen if the level-line through $\vec{p}$ has high curvature.

The union-find watershed algorithm always produces interpixel boundaries. A variant of the flooding-based watershed transform can be used to create thin 8-connected pixel boundaries. To guarantee 4-connected regions and thin boundaries, flooding is best

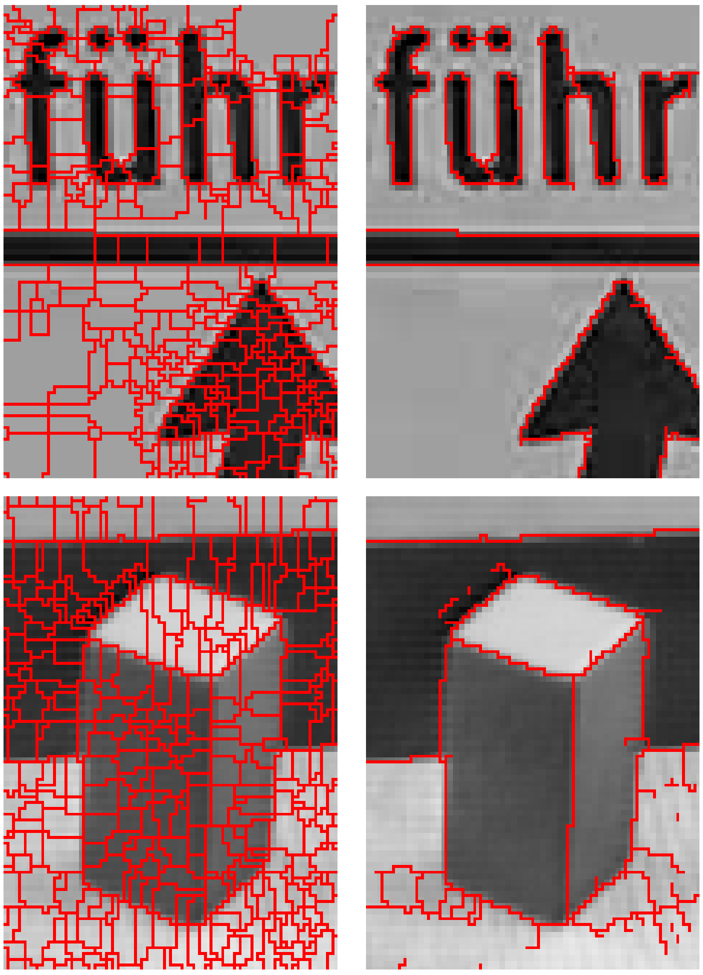


Figure 5.9: Result of the union-find watershed algorithm 5.8 without (left) and with additional thresholding on the boundary strength. That is, the left images contain all watersheds, whereas the right ones only those where the minimal gradient magnitude along the edge is below 16 ("sign" example) or 4 ("blox" example). Without the threshold, we again see oversegmentation. The gradient filter scale was $\sigma_{\text{filter}} = 1$. Compare with the subpixel results in figure 5.6.

realized by means of a topological thinning algorithm which starts in a state where all pixels belong to the boundary, with the exception of the local minima (in 4-neighborhood sense) that will become the root of the catchment basins. The boundary is then thinned whereby the boundary strength (value of the boundary indicator function) is used to prioritize the thinning steps:

Algorithm 5.9: Region Growing Algorithm for Watershed Detection

Input: A boundary indicator image whose watersheds are to be interpreted as boundaries.

1. Find all local minima in the 4-neighborhood sense and label each with a unique

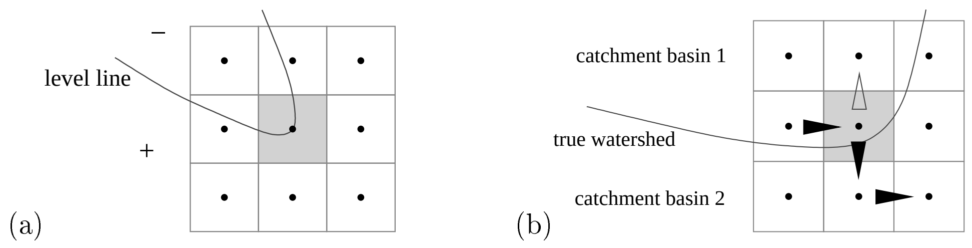


Figure 5.10: Errors of the pixel-based watershed algorithm: (a)When the level line trough the central sampling point has high curvature, none of the 4-neighbors may be located in the lower region of the points Euclidean neighborhood. A false minimum (spurious region) results. (b) When the lowest 4-neighbor of a point near the true watershed is in another catchment basin, the subtree belonging to this point will be assigned to the wrong region (black arrows). The true flowline runs more like the light arrow, but descends too slowly to make the corresponding neighbor the lowest one.

label. Label all other points with the boundary label.

2. Put all simple points of the current boundary region in a priority queue that is sorted according to increasing boundary strength. Recall that simple points are the ones whose removal from the boundary region would not change the number of 4-connected region components and 8-connected boundary components.

3. While the queue is not empty:

    a) Remove the first point from the queue and check if it is still simple. If yes, mark it with the unique label of the adjacent region, otherwise leave it in the boundary. If the point became a region point, look for new simple points in its 4-neighborhood and put them into the queue.

4. Classify the remaining boundary points into edge and node points according to definition 4.16 and create the GeoMap as explained in section 4.3.2.

Examples for this algorithm are shown in figure 5.11. This thinning approach obviously results in an image partition with thin 8-connected boundaries. When the initial seeds are arbitrary regions instead of the local minima, the algorithm still works and is called *marker-based watershed algorithm*.

Instead of the non-local watershed criterion, one can also define ridges by means of local criteria. In the continuous domain, such criteria can always be transformed into equivalent zero-crossing criteria, but this is not possible in the discrete domain. Therefore, this algorithm type deserves special attention, and we show how the most popular algorithm of this type, *Canny's algorithm* [Canny 86], is adapted to the GeoMap framework. Canny's algorithm differs from the algorithms mentioned so far in that it needs more input data: In addition to a boundary indicator encoding scalar boundary strength, the local orientation of the boundary is needed. The most common choice is the gradient direction, which is the natural choice when boundary strength is defined by the local gradient magnitude. However, many other possibilities exist, e.g. by using the eigen directions of local feature

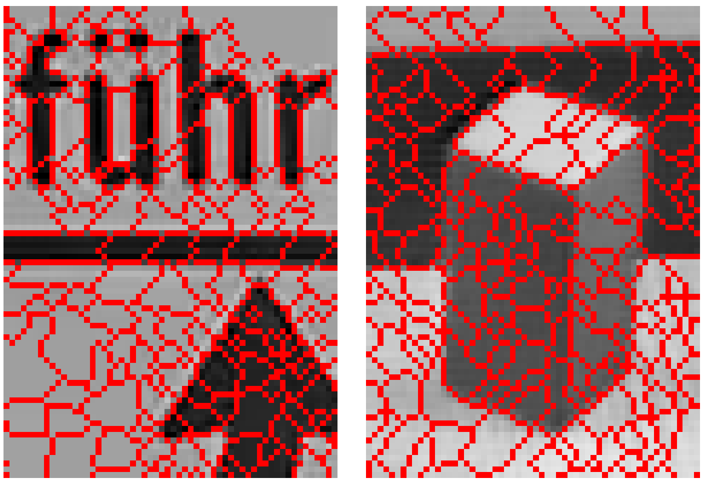


Figure 5.11: Result of the region-growing watershed algorithm 5.9 without additional thresholding on the boundary strength. The gradient filter scale was $\sigma_{\text{filter}} = 1$. Compare with the subpixel results in figure 5.6, and the union-find results in figure 5.9.

tensors (see section 9.1). When only a scalar boundary indicator is available (e.g. the SUSAN measure of local dissimilarity [Smith & Brady 97]), a local direction can still be defined by the direction maximizing the boundary indicator's negative curvature, like in the height-ridge equation (5.5). By convention, the orientation used in Canny's algorithm is normal to the boundary.

Algorithm 5.10: GeoMap creation by Canny's algorithm

Input: A scalar boundary indicator b and an orientation field $\vec{u} = (\cos\theta, \sin\theta)^T$, both defined on a square raster.

1. Mark all pixels as boundary candidates that are local maxima along the local orientation, that is where
$$b(\vec{x}) > b(\vec{x} \pm \vec{d})$$
with
$$\vec{d} = \left(\left\lfloor \frac{\cos\theta}{2\sin(\pi/8)} + \frac{1}{2} \right\rfloor, \left\lfloor \frac{\sin\theta}{2\sin(\pi/8)} + \frac{1}{2} \right\rfloor \right)^T$$
denoting the vector to the nearest-neighbor pixel of $\vec{x}$ in direction $\vec{u}$ ($\lfloor . \rfloor$ is the floor operation). Optionally apply a threshold t_1 on b to remove candidates caused by noise (see section 7.2.2.2 for selection of appropriate thresholds).

2. Perform topology preserving thinning with priority (algorithm 4.2) such that points with lower b are removed first (use the special rules that keep line endings and T-junctions).

3. Create a GeoMap from the resulting 8-connected boundary image (definitions 4.16 and 4.7). Transform it into a polygonal GeoMap according to theorem 4.2.

4. Optionally perform hysteresis thresholding: Delete all GeoMap edges where the maximum gradient magnitude along the edge is below a threshold t_2. (This step is significantly simplified by the GeoMap because finding edges and computing maximum gradient magnitudes are provided as standard operations in the GeoMap abstract data type.)

5. Translate the pixel-accurate GeoMap points to more accurate subpixel locations.

Figure 5.12 shows GeoMaps created by this algorithm. Our main modifications of Canny's original algorithm are the introduction of the thinning step and the use of a GeoMap representation (instead of the usual edgel chains, which lack the concepts of "region" and "junction" and may therefore pose topological problems).

The standard method for subpixel edge localization (step 5) is to fit a parabola to the three points $b(\vec{x})$, $b(\vec{x} \pm \vec{d})$ and place the edge point at its apex. Canny originally used linear interpolation to find the boundary strength of the points $b(\vec{x} \pm \vec{d})$, but we found that nearest-neighbor interpolation seems to improve accuracy [Bugl & Heinemeier 04], see figure 5.13 left. Alternatively, one can use all points in the 3×3-neighborhood of $\vec{x}$: The boundary indicator values from this neighborhood are orthogonally projected onto the line $\vec{u}(\vec{x})$, and a least-squares parabola is fitted through all nine points, figure 5.13 center. Its apex again defines the edge. This method is employed in the code available with [Baker & Nayar 99][5]. We show in section 7.2.1 that this method achieves higher accuracy than the simple 3-point method.

Yet higher accuracy can be achieved by replacing parabola fits with spline interpolation, figure 5.13 right. Let $\tilde{b}(\vec{x})$ be a continuous reconstruction of the boundary indicator according to the spline interpolation formula (3.10). Starting at $\vec{x}$, we perform Newton iterations to find the nearest local maximum of $\tilde{b}$ on the line $\vec{u}(\vec{x})$:

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} - \frac{\vec{u}^T \, \nabla\tilde{b}\left(\vec{x}^{(i)}\right)}{\vec{u}^T \, \mathcal{H}\tilde{b}\left(\vec{x}^{(i)}\right) \, \vec{u}} \, \vec{u} \tag{5.8}$$

where $\mathcal{H}\tilde{b}\left(\vec{x}^{(i)}\right)$ denotes the Hessian of $\tilde{b}$ at $\vec{x}^{(i)}$. Unfortunately, this method occasionally converges to a local minimum of $\tilde{b}$ instead of a maximum. This problem can be solved by replacing the simple Newton iteration with a more sophisticated line search algorithm, for example [Moré & Thuente 94].

It is important to ensure that subpixel point correction does not violate the topology of the GeoMap by moving a point across another edge. The simplest method to avoid this is to restrict the subpixel point coordinate to the current pixel. However, the optimal point according to the 9-point fit or spline interpolation is not always in the current pixel. Then, the topology check according to algorithm 4.8 should be used. Another alternative based on triangulation is described in section 5.3.

The main problem with Canny's algorithm is that its edge model breaks down near junctions: it is assumed that there is a unique boundary normal $\vec{u}$ at each pixel, but this is not true near junctions, where edges of different orientation meet. Orientation

[5]`http://www1.cs.columbia.edu/CAVE/`

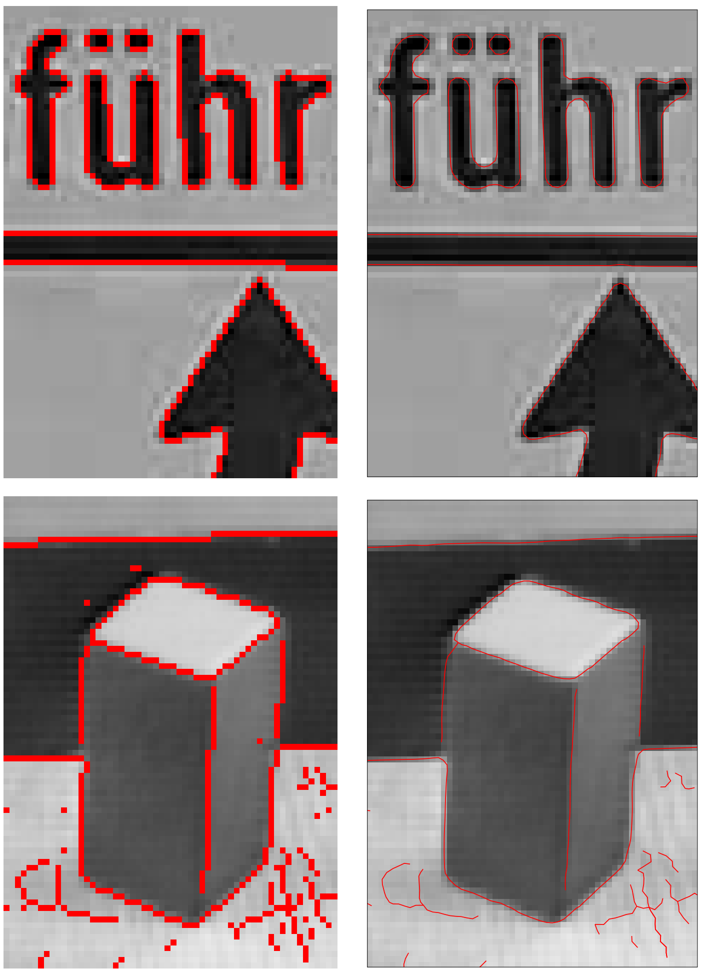


Figure 5.12: Result of Canny's algorithm 5.10 without subpixel correction (left) and with subpixel correction by means of a 3-point parabola fit. Note that the boundary has gaps near junctions, which is a well-known property of Canny's algorithm. The gradient filter scale was $\sigma_{\text{filter}} = 1$, gradient strength thresholds were 16 ("sign" example) and 4 ("blox" examples).

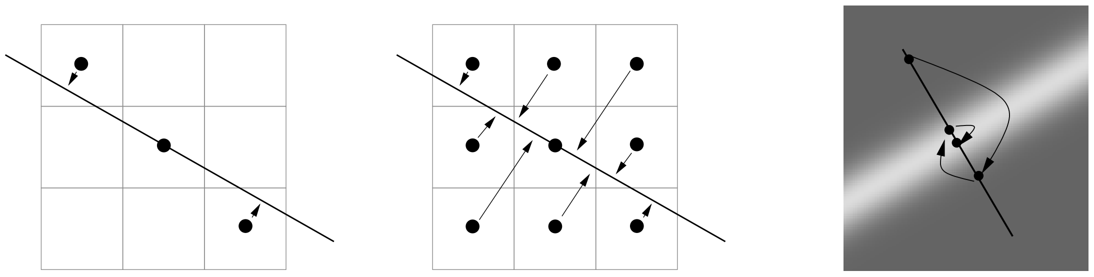


Figure 5.13: Subpixel edgel correction in Canny's algorithm. Left: Projection of three points onto the gradient line, followed by a parabola fit. Center: Likewise with 9 points. Right: Newton iterations along the gradient direction using a the spline-interpolated gradient image.

computation at a junction is dominated by the strongest edge. The pixels on weaker edges receive distorted orientation estimates, so that $b(\vec{x})$ is no longer a local maximum along $\vec{u}(\vec{x})$. Consequently, edges have gaps there. Several authors proposed heuristics to close these gaps. As an example, we report the method of [Rothwell et al. 95], another one is described in algorithm 5.14 in the next section. Rothwell's algorithm fits very well into our GeoMap framework – we only need to insert an additional step in our version of Canny's algorithm[6]:

Algorithm 5.11: GeoMap creation by Rothwell's algorithm

Perform algorithm 5.10 with the following additional step (to be executed between steps 1 and 2):

1a. Add more candidate edge pixels by adaptive thresholding of b:

   (a) Perform a forward Chamfer distance transform on the non-boundary pixels (i.e. those that have not been marked in step 1) to find the nearest boundary candidate in the causal neighborhood (i.e. among the pixels above and to the left). Likewise, perform a backward Chamfer distance transform to find the closest boundary candidate in the anti-causal neighborhood (i.e. among the pixels below and to the right).

   (b) Linearly interpolate the boundary indicator values from the two nearest boundary candidates, weighted according to their distance. Compute the local threshold by reducing the interpolated value to $\lambda\%$ (where $80 \leq \lambda \leq 95$).

   (c) Choose all pixels as additional boundary candidates where the local boundary strength exceeds the threshold.

In essence, this algorithm combines Canny's approach along edges with the region-growing-based watershed transform near junctions.

5.3 GeoMap Creation by Triangulation

So far, we created GeoMaps by means of contour following in the continuous domain, or by means of pixel linking/labeling on a grid. Another possibility is to start with a set of isolated points which mark the boundary but are not yet linked in any way. This approach is inspired by research on laser range scanning. A laser scanner samples the surface of a 3-dimensional object and returns a set of isolated surface points. It is necessary to reconstruct a connected surface from this set of points. This problem has been successfully solved with the concept of *α-shapes* [Edelsbrunner & Mücke 94]. The α-shape is essentially defined as a subset of the Delaunay triangulation of the points such that the radii of the Delaunay cells in the subset are below $\alpha \in \mathbb{R}^+$. Under certain

[6]Since the description in the paper is incomplete, the algorithm has been reverse engineered from an implementation available at `http://marathon.csee.usf.edu/edge/edge_detection.html`.

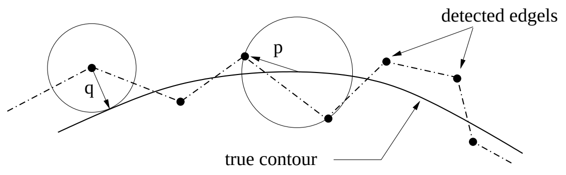


Figure 5.14: Illustration of a (p, q)-sampling: p is the maximum distance from any true contour point to the nearest edgel, and q is the maximum distance from any edgel to the true contour.

conditions, an α-shape is homeomorphic to the correct object surface, or at least of the same homotopy type.

To make clear what it means for the given points to "mark the boundary", we introduce the notion of a (p, q)-sampling of a plane partition:

Definition 5.3. *Let P be a plane partition with boundary B. A finite set of points $S = \left\{s_i \in \mathbb{R}^2\right\}$ is called a (p, q)-*sampling of the boundary B *when the distance of every boundary point $b \in B$ to the nearest point in S is at most p, and the distance of every point $s_i \in S$ to the nearest point in B is at most q, see figure 5.14. The points in a boundary sampling are called* edgels*. The sampling is said to be* strict *when all points are exactly on the boundary, i.e. $q = 0$.*

It should be noted that our definition of edgel is slightly more general than the usual definition because we do not require edgels to have any attributes (such as orientation or confidence) beyond their coordinates. Non-zero edgel displacements $q > 0$ are caused by systematic or statistical measurement errors.

Edgels may be determined in various ways. For example, we can use the points defined by any of the GeoMap algorithms introduced so far, and simply drop the connectivity information. This may not be the method of choice in practice, but it is very interesting and useful from a conceptional point of view because it connects these algorithms to the boundary sampling theorem 6.8 discussed in section 6.2. Another possibility is to detect candidate points by template matching or by just the first step of Canny's algorithm. Most importantly, the points can come from different sources simultaneously (e.g. edge and corner detectors), leading to a natural method for method combination. Since the points are not yet linked, they can also be freely shifted around towards improved subpixel positions without taking care of whether these coordinate shifts preserve topology. That is, instead of checking for topology preservation at every step according to algorithm 4.8, one can allow the points to move more freely and reconstruct the topology afterward.

For our present discussion, the method for computing edgels only matters in so far as it determines the accuracy of the sampling, i.e. the values of p and q. The first step in recovering a connected boundary from the edgels is Delaunay triangulation:

Definition 5.4. *The* Delaunay triangulation *D of a set of points S is the set of all triangles formed by triples $t \subset S$ such that the open circumcircle of every triangle does not contain any point of S. The Delaunay triangles define a simplicial complex, i.e. a polygonal plane partition where all edges are straight lines and all regions are triangles. The union of all cells $c \in D$ is called the* polytope *$|D|$ of D.*

If the points are in general position (i.e. no four points are on a common circle), the Delaunay triangulation is unique. It can be efficiently computed in $\mathcal{O}(n \log n)$ time, where n is the number of points in S. Advanced implementations are publicly available[7], and it takes about a second to triangulate 200000 points in a 3GHz Pentium machine. Since every Delaunay triangulation is a polygonal plane partition, it can be represented as a GeoMap[8]. Of course, the triangles are not the regions we want to reconstruct – Delaunay triangulation produces an even more dramatic over-segmentation than the watershed algorithm. However, under certain conditions the desired boundaries are a subset of the Delaunay edges: Since the distance between edgels along true edges is generally much smaller than the edgel distance across regions, it makes sense to simply remove large triangles and long edges:

Definition 5.5. *The* α-complex D_α *of a set* S *of points is defined as the subcomplex of the Delaunay triangulation* D *of* S *which contains all cells* c *such that*

1. *the radius of the open circumcircle of* c *is smaller than* α*, and this circle contains no point of* S*, or*

2. *an incident cell* c' *with higher dimension is already in* D_α*.*

The connected components of the complement $D_\alpha^C = \mathbb{R}^2 \setminus D_\alpha$ *are called* α-holes *of the triangulation. When the circumcircle radius of the largest triangle in some* α*-hole is at least* $\beta \geq \alpha$*, we speak of an* (α, β)-hole*. The polytope* $|D_\alpha|$ *of* D_α *(i.e. the union of all cells in* D_α*) is called the* α-shape *of the point set* S*.*

For simplicity, we also use the term “hole" for the component which contains the infinite region. It is an (α, β)-hole for arbitrary large β. The idea of the α-complex concept is that the holes approximate the regions of the original plane partition we are analyzing. This is indeed true when the (p, q)-boundary sampling fulfills certain conditions and α is correctly chosen, as we prove in theorem 6.8 in section 6.2. However, the theorem requires an additional algorithm step which removes a few spurious holes not corresponding to true regions. These holes are relatively small and can be recognized by looking at the largest triangle within each hole, as determined by β. This leads to the following algorithm:

Algorithm 5.12: (α, β)-boundary reconstruction

Input: A (p, q)-boundary sampling S of the plane partition of interest. Choose α such that $p < \alpha \leq r - q$ and $\beta = \alpha + p + q$, where r depends on the size of the ground-truth regions (detailed reasons for these parameter choices will be given in section 6.2).

1. Perform a Delaunay triangulation of S and represent it as a GeoMap.

[7] For example the Triangle software by J.R. Shewchuk, `http://www.cs.cmu.edu/~quake/triangle.html`.

[8] In fact, Delaunay triangulation was among the applications that initially motivated the invention of combinatorial maps, which are in turn the basis of our GeoMaps.

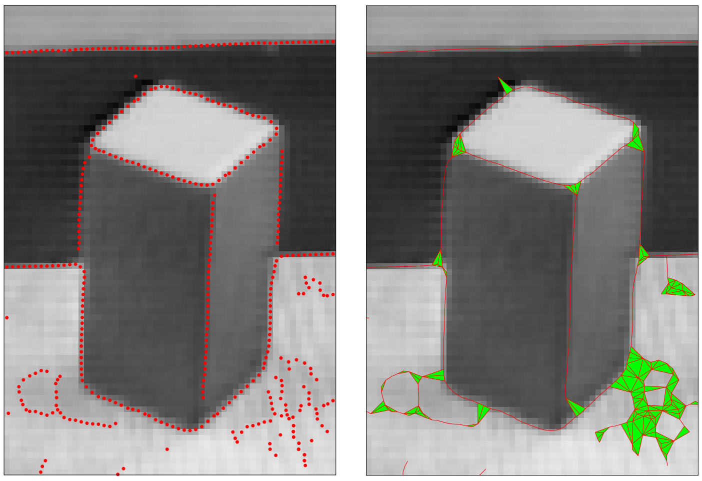


Figure 5.15: Left: Edgels defined by Canny's algorithm with subpixel correction (cf. figure 5.12). Right: Corresponding (α, β)-boundary reconstruction with $\alpha = 2.0$ and $\beta = 2.2$. Edges are depicted in red, triangles remaining in the thick boundary are green. Note that the gaps near junctions have been closed.

2. Mark all cells of the GeoMap that do *not* belong to the α-complex and determine the α-holes, i.e. the connect components of the marked cells.
3. For all α-holes that are not (α, β)-holes (i.e. do not contain a cell with circumradius of at least β), remove the mark from the contained cells.
4. Transform all remaining (α, β)-holes into single regions by merging the contained cells with the appropriate Euler operators.

In other words, spurious holes are simply "painted over" in step 3, and (α, β)-boundary reconstruction essentially amounts to hysteresis thresholding on the triangle size of a Delaunay triangulation. We are going to prove in theorem 6.8 that exactly the desired holes (regions) survive when α and β are chosen as specified. This result can be interpreted as a new sampling theorem for region boundaries: it guarantees that the reconstruction is structure preserving according to definition 2.9. Figure 5.15 shows the (α, β)-boundary reconstruction on the basis of subpixel Canny edgels for the "blox" example. It can be seen that the reconstruction is not everywhere thin: It contains triangles in the areas where the reconstruction is uncertain, according to the selected α and β, which in turn depend on the errors p and q. Figure 5.16 illustrates how different values for α affect the reconstruction: it contains more and more triangles, corresponding to increasing uncertainty. For this *self-diagnosis* of the algorithm to work properly, precise knowledge of the values of p and q is required. We will measure p and q for several important edge detectors in section 7.6.

Another unique feature of (α, β)-boundary reconstruction is its ability to integrate points from different sources: Since the algorithm treats all points equally and doesn't assume any relationship between them, the point set can be the union of the results of

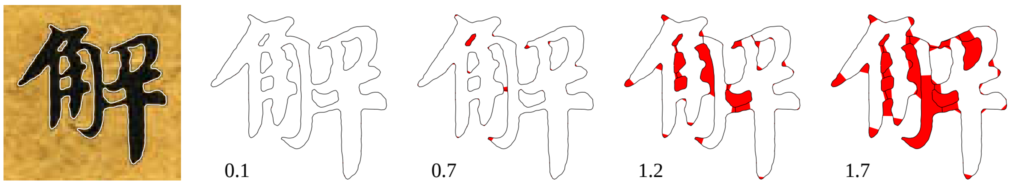


Figure 5.16: Left: Boundary detected for a Chinese character using the subpixel thresholding algorithm 5.2. Right: As α is increased, the (α, β)-boundary reconstruction becomes thicker, indicating increasing uncertainty about the true boundary location.

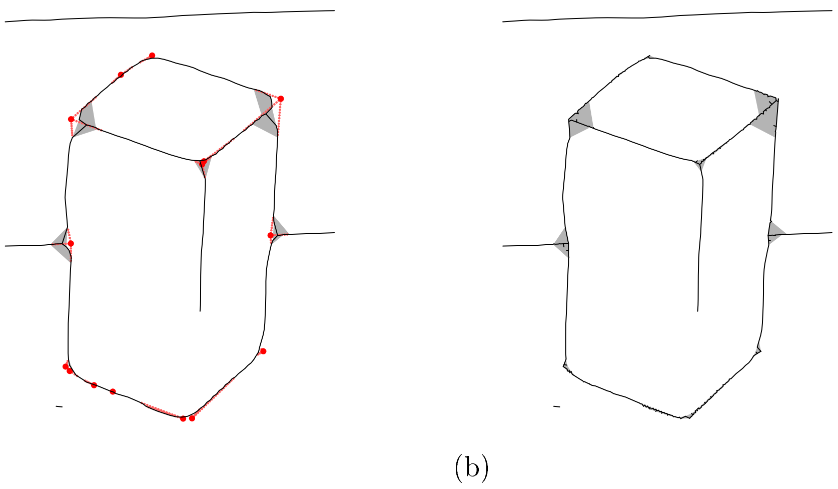


Figure 5.17: (a) Initial (α, β)-boundary reconstruction (black and gray) and maximum likelihood estimates of junctions points (red). (b) When the new points are included in the (α, β)-boundary reconstruction the result has significantly higher quality near junctions.

many boundary detectors. One particularly interesting application of this possibility is the improvement of the segmentation near junctions. We see in figure 5.17a that the (α, β)-boundary reconstruction exhibits significant distortions near junctions. Therefore, we append a second algorithm step that determines a maximum likelihood junction position from the gradient magnitudes and orientations in the neighborhood of connected thick areas, resulting in a set of additional edgels. These edgels are added to the original set, and (α, β)-boundary reconstruction is repeated. It can be seen in figure 5.17b that the junction quality is significantly improved in this way.

In many applications, thick boundary reconstructions are undesirable. Then, one can apply topology preserving thinning (possibly with priority) to the (α, β)-boundary in the same way as it is applied on a grid (cf. algorithm 4.2). An edge in the (α, β)-boundary is called *simple* when its removal does not change the topology of the reconstructed regions. Simple edges always bound an (α, β)-hole on one side and a triangle within the boundary on the other. Thinning removes simple edges until none are left:

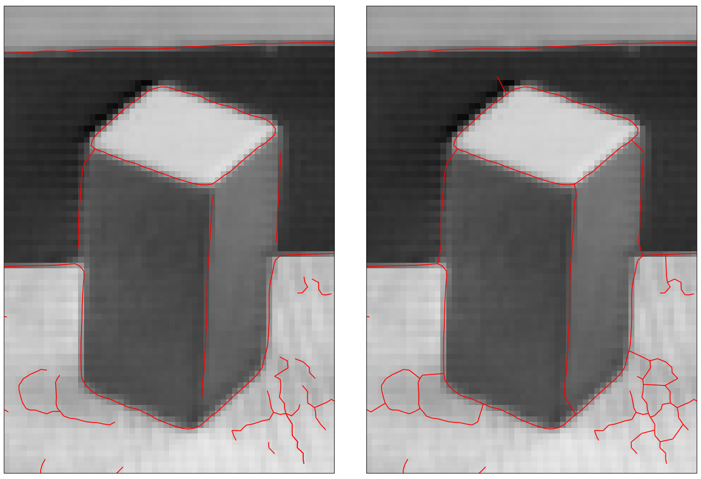


Figure 5.18: Left: The result of Canny's algorithm (same image as figure 5.12 right) has gaps near junctions. Right: (α, β)-boundary reconstruction closes these gaps (cf. figure 5.15), and subsequent thinning according to algorithm 5.13 recovers a thin boundary.

Algorithm 5.13: Minimal boundary reconstruction

Input: An (α, β)-boundary reconstruction.

1. Find all simple edges of the given reconstruction and put them in a priority queue so that long edges are removed first.

2. As long as the queue is not empty: Get the topmost edge from the queue and remove it from the reconstruction by means of merge faces when it is still simple (it may have lost this property after removal of other edges). Put the other edges in the triangle of the removed edge in the queue if they have now become simple.

As far as region topology is concerned, the ordering of the edgels in the priority queue is arbitrary. Instead of length, we can use any application specific importance criterion. For example, it is useful to measure edge contrast (average image gradient along the edge) and remove weak edges first. Figure 5.18 right shows the result of thinning on the basis of length.

Often, edgels are determined by an algorithm that already performs edgel linking, e.g. one of the edge detectors we discussed in the preceding sections. This linking information is ignored by the (α, β)-algorithm, which instead constructs new links from scratch. While this approach is well suited for the proof of the boundary sampling theorem, it might not be the optimal algorithmic solution. Instead, we can force the already known links (edges) to remain in the triangulation by means of *constrained Delaunay triangulation* [Chew 87, Bern & Eppstein 92]:

Definition 5.6. *Let P be a polygonal plane partition and G the planar straight line graph derived from it (i.e. the plane partition where all internal points of the polygonal arcs have* *been turned into vertices, so that all arcs become straight lines). A triangulation T of G is a* constrained Delaunay triangulation *(CDT) when*

1. each edge of G is in T, and

2. for all other edges in T there exists a circle through its end points v_1 and v_2 which does not contain any "visible" vertex of G. A vertex v is visible from the line v_1v_2, when neither of the two lines v_1v and v_2v crosses a line of G.

The constrained Delaunay triangulation is as close as possible to the standard Delaunay triangulation under the requirement that given edges must remain in the triangulation. When G does not contain any predetermined edges, the two triangulations are equivalent. The CDT can also be computed in $\mathcal{O}(n \log n)$ time, and the Triangle software mentioned above contains an implementation. In the context of image segmentation, use of the constrained Delaunay triangulation was proposed in [Ren et al. 05a]. Before computing the triangulation they reduce the number of points by means of polygon simplification:

Algorithm 5.14: Contour completion by constrained Delaunay triangulation

Input: A boundary indicator $b(x, y)$ along with local edge normals $\theta(x, y)$.

1. Create a polygonal GeoMap by Canny's algorithm 5.10.
2. Perform polygon simplification by means of algorithm 4.6. In order for the simplification to be scale-invariant, split point selection and recursion termination are defined in terms of the enclosed angle (unfortunately, the exact criteria are not described in [Ren et al. 05a]). Turn the simplified plane partition into a planar straight line graph G.
3. Compute a constrained Delaunay triangulation of G.
4. Remove edges from the CDT that are not relevant in terms of perceptual salience.

This algorithm is an alternative to Rothwell's heuristic (algorithm 5.11). The remaining vertices after polygon simplification are either edge terminations or corners and junctions. Thus, many edges inserted by the CDT connect an edge termination with the nearest corner or junction point on another arc. It is shown in [Ren et al. 05a] that good gap completions are often found among these additional edges.

The topological guarantees of (α, β)-reconstruction are not easily transformed to constrained Delaunay triangulation. However, this is not a very big problem because inclusion of given edgel links into the triangulation can also be enforced in another way, namely by means of *conforming Delaunay triangulation* [Bern & Eppstein 92]:

Definition 5.7. *Let G be a planar straight line graph. Then a* conforming Delaunay triangulation *is a standard Delaunay triangulation that covers all edges in G, without requiring any extra rules like in CDT. In general, it is necessary to find additional split points (called* Steiner points*) on the edges of G to obtain a conforming Delaunay triangulation.*

The number of additional split points can be large when G is an arbitrary planar straight line graph. But this is not so in the context of segmentation: Here, the distances between edgels *along* linked edges are in the order of the pixel diameter or lower, and thus generally much smaller than the edgel distances *across* regions (which are mostly above the pixel distance). The nearest neighbor of an edgel is usually its successor/predecessor in the same arc. Since nearest neighbor edges are always Delaunay edges, many of the predefined edges in G are automatically preserved. Only relatively few extra points are needed to ensure preservation of all edges in G.

Actually, we do not only want the original edgel links to be part of the Delaunay triangulation, but they should also remain in the (α, β)-reconstruction. A sufficient condition for an edge to belong to the (α, β)-reconstruction is that its length is less than 2α, and its circumcircle does not contain any other edgel. The circle is then called *conflict-free.* This leads to the following algorithm for Steiner point insertion into a polygonal plane partition:

Algorithm 5.15: Conforming (α, β)-reconstruction

Input: A polygonal plane partition P resulting from any segmentation algorithm described in this chapter.

1. Compute the planar straight line graph G by turning the internal knots of all polygons of P into vertices. Subdivide edges until all remaining edges are shorter than 2α.

2. For each vertex with degree ≥ 3: Let s be the junction point, and s_i the end point of dart i starting at s. Compute the length of the shortest dart starting at s: $\delta = \min_i \|s\, s_i\|$. Split all darts starting at s by inserting new vertices at distance $\delta/2$ from s.

3. Do likewise for each vertex with degree $= 2$, when the angle between the two incident darts is less than $90°$.

4. Compute the circumcircle of all edges which are incident to at least one vertex not yet treated in steps 2 and 3. When the circumcircle contains one or several other edgels, compute the minimum distance of these edgels from the current edge. Subdivide the current edge until all resulting segments are shorter than this minimum distance.

5. Perform (α, β)-reconstruction on the vertex set of the modified graph G'.

It is easy to see that this algorithm indeed ensures that the edges of the the resulting (α, β)-reconstruction cover the edges of P. This is immediately obvious for the edge subdivision in step 4, because edges are explicitly constructed so that their circumcircle does not contain any vertex. In the steps concerning vertices (steps 2 and 3), observe that the end points of the modified darts starting at vertex s are all located on a circle with radius $\delta/2$ around s. The circumcircles of these darts have radius $\delta/4$ and are completely

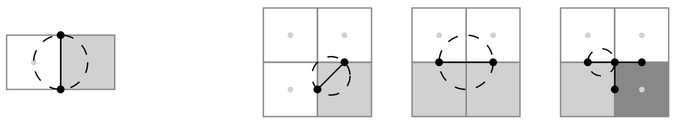


Figure 5.19: The circumcircles (dotted) of crack edges and mid-crack edges cannot contain any edgel in their interior. Left: Only one configuration (modulo rotation) exists in a crack edge. Right: In a mid-crack edge, there are three configurations (modulo rotation): diagonal segments, straight segments and segments adjacent to a junction.

contained in the $\delta/2$-circle around s. Thus, they cannot contain any other point of G', and the claim follows. It should be noted that additional split points are always inserted on existing edges, so that the geometry of the boundary of G (and therefore of P) is not modified.

We already mentioned that it is typically not necessary to subdivide many edges when conforming triangulation is to be performed in the context of segmentation. In fact, edges derived from subset digitization (i.e. crack edges and mid-crack edges according to definition 4.11) automatically lead to conforming triangulations. Recall that a crack edge consists of polygons whose knots are located at pixel corners, and a mid-crack edge consists of polygons whose knots are at the center points of the cracks (plus possibly junctions vertices at pixel corners), see figure 4.8. The boundary graph G is thus naturally defined by the knots and segments of these polygons. It is now easy to see that the edges of G are always contained in a standard Delaunay triangulation of the knots, because no knot can ever lie in the interior of any polygon segment's circumcircle. Figure 5.19 shows this for all configurations that can occur in a crack edge or mid-crack edge graph.

Conforming triangulations solve two problems simultaneously: They create true Delaunay triangulations where the conclusions of the boundary sampling theorem 6.8 apply, and they ensure that the original linked boundary remains in the triangulation, without enforcing this explicitly like in a CDT. Another advantage of conforming triangulations may be speed: When the boundaries to be kept in the conforming triangulation form closed contours (for example, because they were computed by a watershed transform), the triangulation can be computed in every region independently, because no Delaunay edge can ever cross a region boundary. Instead of a single triangulation problem with many points, the problem decomposes into a number of independent sub-problems with much fewer points. Since the complexity of Delaunay triangulation is $\mathcal{O}(n \log n)$, this may yield a noticeable speed-up.

6 Geometric Sampling Theorems

Abstract

We have stressed throughout this work that it is important to understand the relationship between the continuous world and its digital representation in the computer. We already discussed Shannon's sampling theorem in chapter 3, but it covers only one aspect of the image acquisition process: it states that a band-limited analog image can be reconstructed from the corresponding digital camera image. Two important sampling questions are thus still open:

- The analog camera image is created from the ideal geometric image by smoothing with the camera's point spread function. How much does this smoothing distort the geometry, and what remains visible at a given resolution (as determined by the band-limit of the PSF)?
- The reconstructed geometry must be represented with finite accuracy in a pixel-based or polygonal GeoMap. How complex can an original partition be so that a structure preserving reconstruction according to definition 2.9 is still possible?

Since these questions are not answered by Shannon's sampling theorem, we devote this chapter to a number of new sampling theorems that clarify the *topological* and *geometric* relationships between an ideal image and a real segmentation. These theorems state in a precise sense that the topology of the ground truth partition can be correctly reconstructed, and the geometric errors of the reconstruction are bounded if the true regions are sufficiently large (relative to the sampling grid), and the boundary detector is sufficiently accurate (considering all error sources).

6.1 Sampling Analysis of Grid-Based Region Representations

In this section we look at reconstructions where regions are defined as a union of pixels. That is, we consider subset digitization (definition 4.10) and interpixel boundaries (definition 4.11). It is evident that much of the geometric information in the analog or digital camera image is disregarded by such a representation, because edgel positions are restricted to the pixel corners. Nevertheless, enough information survives to make these representations useful, as was first shown by Gauss:

Theorem 6.1. *Let Q be a planar convex set and $\hat{Q}$ its reconstruction by means of subset digitization on a square grid (cf. definition 4.10). Then the area of $\hat{Q}$ converges linearly toward the true area of Q as the grid spacing h goes to zero* $\left| \|Q\| - \left\|\hat{Q}\right\| \right| = \mathcal{O}(h)$.

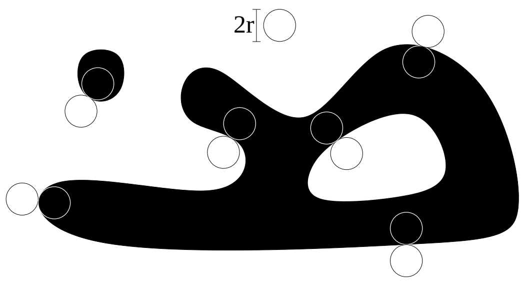


Figure 6.1: A r-regular set with some of the osculating disks shown. Illustration from [Stelldinger 03].

Area is, of course, only one important shape characteristic. In the context of image analysis, we are particularly interested in whether the topological structure of a given plane partition is preserved, i.e. whether the reconstruction is topological equivalent to the original or at least of the same homotopy type. In addition, we do not want to restrict ourselves to multigrid convergence statements like in theorem 6.1 – we also need bounds for given absolute grid spacings because the image resolution cannot be changed once the image has been taken[1].

The analysis in this section will be based on the terms *grid, pixel, adjacency*, and *path* as introduced in definition 4.9. Recall that pixels are defined as Voronoi regions of grid points. This is crucial for the following analysis because it implies a number of important pixel properties (e.g. convexity, neighborhood relations) without resorting to the actual pixel shape. In contrast to previous authors (e.g. [Serra 82, Pavlidis 82, Latecki et al. 98]) who restricted their attention to specific grids such as the square and hexagonal ones, this general pixel definition allows us to extend the analysis to arbitrary fixed grids, including irregular ones. It is only necessary to specify a bound for the maximum distance from a grid point to the border of its surrounding pixel:

Definition 6.1. *If the radius of the pixels in a grid S does not exceed r (i.e. $\vec{x} \in \mathrm{pixel}_S(\vec{s}) \Rightarrow |\vec{x} - \vec{s}| \leq r$), the grid is denoted an* r-grid. *The number r is also known as the grid's* fill distance *[Wendland 05].*

In case of a square grid with pixel distance d, the pixel radius is $d/\sqrt{2}$, hence we speak of a $(d/\sqrt{2})$-grid.

In order to facilitate formal proofs, the class of admissible plane partitions must be restricted to *r-regular shapes* (this restriction will be relaxed in section 6.2). The fundamental importance of this shape class was independently discovered by [Pavlidis 82] and [Serra 82] and probably others:

Definition 6.2. *A plane partition P with boundary B is called* r-regular *if at each point $b \in B$ there exist two osculating open disks with radius r that do not intersect B.*

An example is depicted in figure 6.1. The definition has three important consequences.

1. It requires the plane partition to be *binary*, i.e. the regions can be classified into foreground and background regions so that every point of B is in the closure of

[1]Unless we have a static scene and an active vision system that can take another image with improved resolution when necessary.

both the foreground and the background. When the boundary is assigned to the fore- or background in an arbitrary way, and the union of all foreground points is denoted as A, the background can be defined by the complement $A^C = \mathbb{R}^2 \setminus A$.

2. The definition ensures that the curvature of the boundary $B = \partial A = \partial \left(A^C\right)$ is everywhere at most $\frac{1}{r}$, and the partition cannot have corners and junctions.
3. The foreground objects as well as the background have to be wide enough for the r-disks to fit in, so that narrow isthmuses are prohibited, and objects may not be placed too close to each other.

The definition is equivalent to saying that the objects and background are invariant under morphological opening and closing with any r'-disk structuring element such that $r' \leq r$. This variant of the definition was used by [Serra 82].

6.1.1 Sampling without Blurring

In this section we simplify the problem further by assuming that the geometric image is digitized without blurring, i.e. that the camera PSF is a Dirac δ-function. The subset digitization is then equivalent to a nearest-neighbor interpolation of the digital image: all pixels whose sampling point is within A are completely filled with the foreground color (say, gray-level 1), all others with the background color (say, gray-level 0). The union of all foreground pixels is the reconstruction of A under this model and will be denoted $\hat{A}$. The reconstructed boundary $\partial\hat{A}$ is the union of all pixel boundaries that belong to both a fore- and a background pixel. The presentation in this section follows [Stelldinger & Köthe 05].

We first show that the reconstructed set has the same topology as the original both in the fore- and background when A is r-regular. The claim is established in a series of lemmas which are formulated and proved for the foreground A, but apply analogously to the background A^c.

Lemma 6.1. *Let A be a r-regular set and $\hat{A}$ its reconstruction on an arbitrary r'-grid such that $r' < r$. Then two sampling points lying in different components of A cannot lie in the same component of $\hat{A}$.*

Proof. The Hausdorff distance between any two components of A is at least $2r$ because otherwise no r-disk would fit between the components. Since the pixel radius is at most r', any component $\hat{A}'$ of the reconstruction is completely contained within the r'-dilation of some component A' of the original. The Hausdorff distance between two components of $\hat{A}$ that originate from different components of A is at least $2r - 2r' > 0$. Therefore, the reconstruction process will never merge two components of A. □

Lemma 6.2. *Let A' be a component of the set A, and $A'_\ominus = (A' \ominus B_{r''})^0$ the open interior of the erosion of A' with a disk structuring element $B_{r''}$ of radius $r'' < r$. Further, let S be an r'-grid with $r' < r''$ and $S_i := \{\vec{s} \in S : pixel(\vec{s}) \cap A'_\ominus \neq \emptyset\}$ the set of all sampling points whose pixels intersect $A'_\ominus$. Then at least one sampling point in S_i is in A'.*

Proof. Since A is r-regular, every component A' contains at least a disk of radius r. The center $\vec{m}$ of such a disk is necessarily in $A'_\ominus$. Let P be a pixel that contains $\vec{m}$ (there can be several such pixels if $\vec{m}$ happens to be on the pixel border), and $\vec{s}_P$ the sampling point of that pixel. Then $\vec{s}_P$ is in S_i, and the distance $|\vec{m} - \vec{s}_P|$ is at most $r' < r''$. Thus, $\vec{s}_P$ lies in A'. □

This lemma implies that no component of the original shape gets lost in the reconstruction. The next two lemmas show that all components of the reconstruction are connected via direct adjacency.

Lemma 6.3. *Let A, A', $A'_\ominus$, S, and S_i be defined as in lemma 6.2. Then any pair of sampling points in S_i is connected by a discrete path whose points all belong to S_i as well.*

Proof. Recall definition 4.9 for the meaning of the term discrete path. Let $\vec{s}_1$, $\vec{s}_2$ be two sampling points in S_i and P_1, P_2 the corresponding pixels. Due to the definition of S_i, the interior of the two pixels intersects $A'_\ominus$, and there exist points $\vec{s}_1{}' \in A'_\ominus \cap P_1$ and $\vec{s}_2{}' \in A'_\ominus \cap P_2$. Due to r-regularity, $A'_\ominus$ is a connected set, so we can connect the points $\vec{s}_1{}'$ and $\vec{s}_2{}'$ with a continuous path in the Euclidean plane which runs entirely in $A'_\ominus$. In fact, there are infinitely many such paths, and we can choose one that does not intersect any pixel corner. This path intersects a number of consecutive pixels, and the corresponding sampling points are all in S_i because the path is in $A'_\ominus$. An ordering on the continuous path induces an ordering of the sampling points, and consecutive points in this ordering are directly adjacent because the path does not intersect pixel corners. Hence, these sampling points form a discrete path whose points do all belong to S_i. □

Lemma 6.4. *Let A, A', $A'_\ominus$, S, and S_i be defined as in lemma 6.2, and let $S' \supseteq S_i$ be the set of all sampling points in A'. Then every point in S' is either also a member of S_i, or is connected to a member of S_i by a discrete path whose points all belong to S'.*

Proof. Let $\vec{s} \in S'$. Then there exists a disk of radius r that is entirely in A' and which contains $\vec{s}$. Let $\vec{m} \in A'_\ominus$ be the center of the disk. The half-line starting at $\vec{s}$ and going through $\vec{m}$ crosses the border of pixel $P(\vec{s})$ at exactly one point $\vec{c}$ (because pixels are Voronoi regions and thus convex). If $d(\vec{s}, \vec{m}) \leq d(\vec{s}, \vec{c})$, the point $\vec{m}$ lies in $P(\vec{s})$, and $\vec{s} \in S_i$. Otherwise, let G be the pixel edge that contains $\vec{c}$ (if $\vec{c}$ happens to be a pixel corner, arbitrarily select one of the two edges meeting at that corner). Due to the definition of Voronoi regions, the point $\vec{s}\,'$ constructed by mirroring $\vec{s}$ on G is also a sampling point of S, and $\vec{s}$ and $\vec{s}\,'$ are directly adjacent. This also implies that the distance from $\vec{c}$ to both points is equal: $s = d(\vec{s}, \vec{c}) = d(\vec{s}\,', \vec{c})$, so that $\vec{s}$ and $\vec{s}\,'$ are on a circle with radius s around $\vec{c}$. Since $\vec{s}$, $\vec{c}$ and $\vec{m}$ are collinear, $\vec{s}$ has the largest distance from $\vec{m}$ among all points on that circle, and in particular $d(\vec{s}\,', \vec{m}) < d(\vec{s}, \vec{m})$. Thus, $\vec{s}\,'$ is also in S', and closer to $\vec{m}$ than $\vec{s}$. When we repeat this construction iteratively, we eventually end up at a point $\vec{s}^{\,*}$ that is in S_i because S' contains only finitely many points. The sequence of directly adjacent points $\vec{s}, \vec{s}\,', \ldots, \vec{s}^{\,*}$ is a discrete path that lies entirely in S'. □

These lemmas are sufficient to prove if and when the topology of r-regular shapes will be preserved in the digital reconstruction. Interestingly, by allowing arbitrary (regular and irregular) grids, the theorem's conditions become both sufficient *and* necessary.

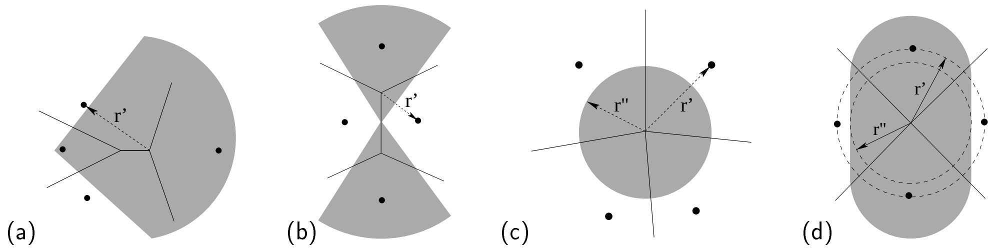


Figure 6.2: Examples where the topology of the reconstruction by an r'-grid differs from the topology of the original set because the set is not r'-regular. The grid is indicated by sampling points and pixel borders, whereas the set to be digitized is shown in gray. (a) The original set has a corner, its reconstruction is disconnected. (b) The original set has a junction, the reconstruction is again disconnected. (c) The original set is too small and gets lost completely. (d) The original set is too narrow, the reconstruction has a junction.

Theorem 6.2. [Stelldinger & Köthe 05]: *Let $A \subset \mathbb{R}^2$ be a shape and $\hat{A}$ its reconstruction by an arbitrary r'-grid S. Then A and $\hat{A}$ as well as their complements are guaranteed to be topologically equivalent and have identical homotopy trees if A is r-regular with $r > r'$ Conversely, if A is not r-regular, there always exists an r'-grid with $r' < r$ such that A and $\hat{A}$ (or A^c and $\hat{A}^c$) are not topologically equivalent.*

Proof. Due to lemma 6.2, there is a mapping from the foreground components of A onto foreground components of $\hat{A}$. Lemma 6.1 implies that the mapping is injective, and from lemmas 6.3 and 6.4 follows surjectivity. The same holds for the background components of A and $\hat{A}$. In other words, there is a injective mapping between the components of the original and the reconstruction. In addition, the lemmas imply that all components of $\hat{A}$ and $\hat{A}^c$ are connected by direct adjacency. Therefore, there is also a bijective mapping between the boundaries ∂A and $\partial \hat{A}$ which implies the identity of the homotopy trees of original and reconstruction. To establish topological equivalence, we must show that a homeomorphism exist for both the fore- and background. This is indeed the case: since there is an bijective mapping between the components in A and $\hat{A}$, and all their boundaries are Jordan curves, a homeomorphism between the boundaries exists. This homeomorphism can be extended to the interior of the components by standard means, giving the required $\mathbb{R}^2$-homeomorphism. This establishes that r-regularity of A is a sufficient condition.

To show that it is also a necessary condition, we proof that there always exists some grid which leads to a reconstruction that topologically differs from the original. We need to consider two cases: (1) A is not regular for any $r > 0$, and (2) A is r''-regular, but $r'' \leq r'$.

Case 1: In this case, the boundary ∂A contains at least one corner or junction. It is then trivial to place sampling points in a way that causes the reconstruction of a connected set to be disconnected, see figure 6.2 (a) and (b).

Case 2: Let A be r''-regular with $r'' < r' < r$. Then there exists a maximal inside or outside circle with radius r'' and center $\vec{p}$ that touches ∂A in at least two points. Draw a circle with radius r' around $\vec{p}$. *Case 2a*: If a component of ∂A coincides with the r''-circle, a component of A or A^c lies completely inside the r'-circle. We can place sampling points on this circle in a way that the component is lost in the reconstruction, figure 6.2 (c). *Case 2b*: Otherwise, we can choose r' so that part of the r'-circle is in A and part is in A^c. If these parts form more than two connected components we can place a sampling point in each of these components, and the reconstruction will contain a junction, whereas the original didn't, figure 6.2 (d). *Case 2c*: If there are exactly two components, we can move the r'-circle a little until either it no longer intersects ∂A (which brings us back to case 2a), or the number of components increases (which brings us back to case 2b). In no case will the topology be preserved. $\square$

The original sampling theorems of Pavlidis and Serra follow as corollaries of this more general theorem:

Corollary 6.1. [Pavlidis 82]: *Let S be a square grid with pixel spacing d, in arbitrary position (i.e. arbitrary rotation and translation with respect to the coordinate system). The pixel radius in such a grid is $d/\sqrt{2}$. The reconstruction of any r-regular shape with $r > d/\sqrt{2}$ will be topologically equivalent to the original. The same is true for the background.*

[Serra 82]: *Let S be a hexagonal grid with pixel spacing d, in arbitrary position. The pixel radius in such a grid is $d/\sqrt{3}$. The reconstruction of any r-regular shape with $r > d/\sqrt{3}$ has the same homotopy tree as the original.*

It should be noted that our proof provides a better bound than Serra's original one (he required $r > d$). In case of these special grids, r-regularity is only a sufficient, but no longer a necessary requirement. Certain shapes with corners are now allowed, provided that the angle at the corner is larger than 90° (square grid) or 60° (hexagonal grid). A more detailed treatment of these kind of shapes is currently being developed in the theory of *half-regular shapes* [Stelldinger 05], which we shall not pursue further in the present work.

However, we had seen in section 2.2 that topological equivalence is not sufficient for two shapes to be recognized as similar. It is also necessary to establish high geometric similarity. In the sequel, we are going to show that the reconstruction of any r-regular shape is even r'-homeomorphic to the original, where r' is the maximum pixel radius. But first, we give a simple proof for the shapes being weakly r-similar (see definition 2.13):

Lemma 6.5. *Let A be a r-regular shape, and $\hat{A}$ its reconstruction by some r'-grid S with $r' < r$. Then the Hausdorff distance between the boundaries ∂A and $\partial\hat{A}$ is at most r', i.e. the shapes are weakly r'-similar.*

Proof. Suppose to the contrary that $\partial\hat{A}$ contains a point $\vec{x}$ whose distance from ∂A exceeds r'. Let $Q := \{\vec{s} \in S : \vec{x} \in \text{pixel}(\vec{s})\}$ be the set of sampling points whose pixels contain $\vec{x}$. Due to the definition of an r'-grid, all sampling points in Q are located in a closed r'-disk around $\vec{x}$. Since the shape is r-regular and $r > r'$, this disk is either completely in A or completely in A^c. The same must be true for all points in Q. Therefore, $\vec{x}$ cannot belong to the boundary $\partial\hat{A}$ – contradiction. $\square$

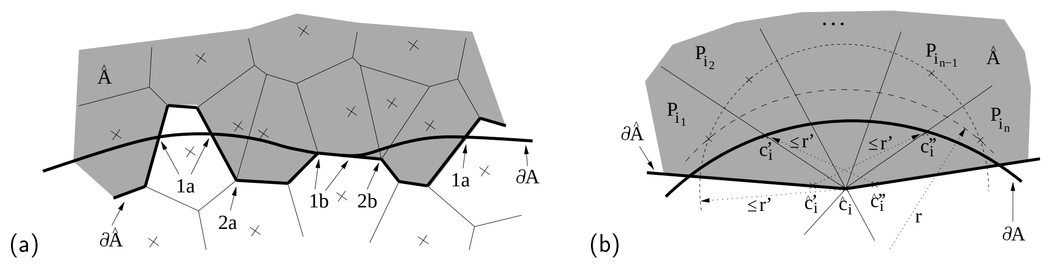


Figure 6.3: (a) Different cases for the definition of split points; (b) partition refinement and mapping for case 2b.

The analogous proof for strong r'-similarity is much more complicated.

Theorem 6.3. [Stelldinger & Köthe 05]: *Let A be a r-regular shape, and $\hat{A}$ its reconstruction by some r'-grid S with $r' < r$. Then there exists a r'-homeomorphism between A and $\hat{A}$. That is, there is a bijective function $f : \mathbb{R}^2 \to \mathbb{R}^2$ such that $f(\vec{x}) \in \hat{A} \Leftrightarrow \vec{x} \in A$ and $|f(\vec{x}) - \vec{x}| \leq r'$.*

Proof. According to theorem 6.2, we already know that A and $\hat{A}$ are $\mathbb{R}^2$-homeomorphic. So it remains to be shown that the homeomorphism moves no point by more than r'. This can always be guaranteed for the entire plane $\mathbb{R}^2$ if it is true for the boundaries ∂A and $\partial\hat{A}$. Due to r-regularity of A, no pixel can touch two components of ∂A. Therefore, we can treat each component $\partial A'$ of ∂A and its corresponding component $\partial\hat{A}'$ separately. The proof principle is to split $\partial A'$ and $\partial\hat{A}'$ into sequences of segments $\{C_i\}$ and $\{\hat{C}_i\}$, and show that, for all i, $\hat{C}_i$ can be mapped onto C_i with an r'-homeomorphism. The order of the segments in the sequences is determined by the orientation of the plane, and corresponding segments must have the same index. Then the existence of an r'-homeomorphism between each pair of segments implies the existence of the r'-homeomorphism for the entire boundary. We define initial split points $\hat{\vec{c}}_i$ of $\partial\hat{A}'$ as follows (see figure 6.3a):

Case 1: A split point is defined where $\partial\hat{A}'$ crosses or touches $\partial A'$. *Case 1a:* If this is a single point, it automatically defines a corresponding split point of $\partial A'$. *Case 1b:* If extended parts of the boundaries coincide, the first and last common points are chosen as split points.

Case 2: A pixel corner which is on $\partial\hat{A}'$ but not on $\partial A'$ becomes a split point if the corner point lies in A (A^c) and belongs to at least two pixels that are in $\hat{A}^c$ ($\hat{A}$). *Case 2a:* If there are exactly two such neighboring pixels, a corresponding split point is defined where $\partial A'$ crosses the common boundary of these pixels. *Case 2b:* Otherwise, the split point is treated specially.

In the course of the proof, the initial partition will be refined. The treatment of case 1b) is straightforward: Here, two segments C_i and $\hat{C}_i$ coincide, so we can define the r'-homeomorphism as the identity mapping.

Next, consider case 2b (figure 6.3b). Let the special split point $\hat{\vec{c}}_i \in A$ (A^c) be a corner of pixels $P_{i_1}, ..., P_{i_n} \in \hat{A}^c$ ($\hat{A}$). The orientation of the plane induces an order of these pixels. The pixels P_{i_2} to $P_{i_{n-1}}$ intersect $\partial\hat{A}'$ only at the single point $\hat{\vec{c}}_i$. We must avoid that an extended part of $\partial A'$ gets mapped onto the single point $\hat{\vec{c}}_i$. Thus, we change

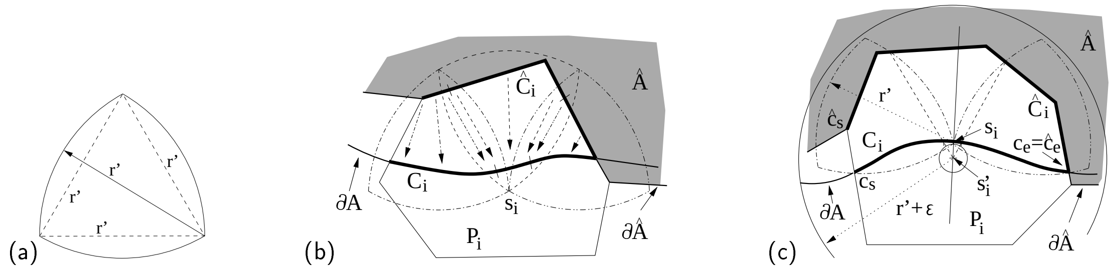


Figure 6.4: (a) Any two points in a Reuleaux triangle of size r' have a distance of at most r'; (b) Covering of corresponding segments with Reuleaux triangles; (c) Construction for sampling points lying on $\partial^{A'}$.

the initial partitioning: Replace $\hat{\vec{c}}_i$ with two new split points $\hat{\vec{c}}_i{}'$ and $\hat{\vec{c}}_i{}''$, lying on $\partial\hat{A}'$ to either side of $\hat{\vec{c}}_i$ at a distance ε. Define as their corresponding split points the points $\vec{c}_i{}'$ and $\vec{c}_i{}''$ where $\partial A'$ crosses the common border of P_{i_1}, P_{i_2} and $P_{i_{n-1}}, P_{i_n}$ respectively. Due to r-regularity, $|\overline{\vec{c}_i{}'\hat{\vec{c}}_i}| < r'$ and $|\overline{\vec{c}_i{}''\hat{\vec{c}}_i}| < r'$, and the same is true for all points between $\vec{c}_i{}'$ and $\vec{c}_i{}''$. Therefore, ε can always be chosen so that every point between $\vec{c}_i{}'$ and $\vec{c}_i{}''$ can be mapped onto every point between $\hat{\vec{c}}_i{}'$ and $\hat{\vec{c}}_i{}''$ with a displacement of at most r'. This implies the existence of an r'-homeomorphism between these segments.

After these modifications, the segments not yet treated have the following important properties: Each C_i is enclosed within one pixel P_i, and the corresponding segment $\hat{C}_i$ is a subset of P_i's boundary. To prove the theorem for these pairs, we use the property of Reuleaux triangles with diameter r' that no two points in such a triangle are farther apart than r' (figure 6.4a). Due to r-regularity, $\partial A'$ can cross the border of any r'-Reuleaux triangle at most two times. We refine the segments so that each pair is contained in a single triangle, which implies the existence of an r'-homeomorphism. Consider the pair C_i, $\hat{C}_i$ and let the sampling point of pixel P_i be $\vec{s}_i$. If this point is not on $\partial A'$ (figure 6.4b), C_i splits P_i into two parts, one containing $\hat{C}_i$ and the other containing $\vec{s}_i$. We now place r'-Reuleaux triangles as follows: a corner of every triangle is located at $\vec{s}_i$, every triangle intersects C_i and $\hat{C}_i$, and neighboring triangles are oriented at 60° of each other, so that no three triangles have a common overlap region. Since the pixel radius is at most r', this set of triangles completely covers both C_i and $\hat{C}_i$, and each consecutive pair of triangles shares at least one point of either segment. Thus, we can define additional split points among the shared points, so that corresponding pairs of the new segments lie entirely within one triangle. The existence of an r'-homeomorphism for the refined segments follows.

If the sampling point $\vec{s}_i$ of P_i is on $\partial A'$ (figure 6.4c), this Reuleaux construction does not generally work. In this case, we first place two r'-Reuleaux triangles such that both have $\vec{s}_i$ as a corner point, one contains the start points $\vec{c}_s, \hat{\vec{c}}_s$ of C_i and $\hat{C}_i$ respectively, the other the end points $\vec{c}_e, \hat{\vec{c}}_e$, and they overlap $\hat{C}_i$ as much as possible. If they cover $\hat{C}_i$ completely, the Reuleaux construction still works with $\vec{s}_i$ as split point. Otherwise $\hat{C}_i$ is partly outside of the triangles, and the normal of $\partial A'$ crosses $\hat{C}_i$ in this outside region. We choose a point $\vec{s}_i{}'$ on the opposite normal with distance ε from $\vec{s}_i$ and project each point $\vec{c}$ of $\hat{C}_i$ not covered by either triangle onto the point where the line $\overline{\vec{c}\,\vec{s}_i{}'}$ crosses C_i. It can be seen that this mapping is an r'-homeomorphism: Draw circles with radius ε and $r' + \varepsilon$ around $\vec{s}_i{}'$. C_i and $\hat{C}_i$ lie between these circles, so that each point is moved by at most r'. The extreme points of this construction define new split points, and the remaining parts of C_i and $\hat{C}_i$ can be mapped within the two triangles. Thus, there is an r'-homeomorphism in this case as well. □

This theorem stresses once more the fundamental role of r-regular shapes: Only r-regularity guarantees topological equivalence and geometric similarity under all circumstances, i.e. with all grid layouts, translations and rotations (see [Stelldinger & Köthe 05] for a proof of this statement). Unfortunately, the real world is not that simple – real shapes are often not regular. We will deal with this problem in section 6.2. Before, we will make our sampling analysis of regular shapes more realistic in another way, namely by considering blurring in a real camera.

6.1.2 Sampling of Blurred Images

In a real system, the geometric image is blurred before sampling, and a realistic geometric sampling theorem should incorporate a reasonable blurring model. This was first pointed out by [Latecki et al. 98]. They define *v-digitization* (with $0 \leq v \leq 1$) where a pixel belongs to the reconstructed set $\hat{A}$ if at least $v \times 100\%$ of the pixel area is covered by the shape to be digitized. For a square grid with pixel distance d they were able to prove that the topology of an r-regular shape is preserved for all v when $d < r/\sqrt{2}$. This result can easily be expressed and generalized in terms of the linear image acquisition model described in section 3.1. The presentation in this section follows again [Stelldinger & Köthe 05].

In the linear image acquisition model, blurring is generally described by a convolution of the ideal geometric image with a point spread function (PSF):

$$\begin{aligned} \tilde{f}_A(\vec{x}) &= PSF(\vec{x}) \star \chi_A(\vec{x}) \\ f_{kl} &= \tilde{f}_A\left(\vec{x} = (k \cdot d, l \cdot d)^T\right) \end{aligned}$$

where χ_A is the indicator function of the shape to be digitized, and d is the pixel pitch. The v-digitization is obtained by using a flat, square-shaped PSF which has exactly the same size as the pixels

$$PSF(\vec{x}) = \begin{cases} 1 & \text{if } \max\left(|x_1|, |x_2|\right) < \frac{d}{2} \\ 0 & \text{otherwise} \end{cases}$$

The choice of a particular value v is equivalent to applying the threshold v to the digital gray-scale image f_{kl}, followed by nearest-neighbor interpolation to get the binary shape reconstruction $\hat{A}$. The interpretation of v-digitization in terms of the linear camera model is much more general because we can now, in principle, choose the PSF arbitrarily. Since thresholding commutes with sampling and nearest-neighbor interpolation, we can simplify the theoretical analysis by changing the order of events: the threshold is now

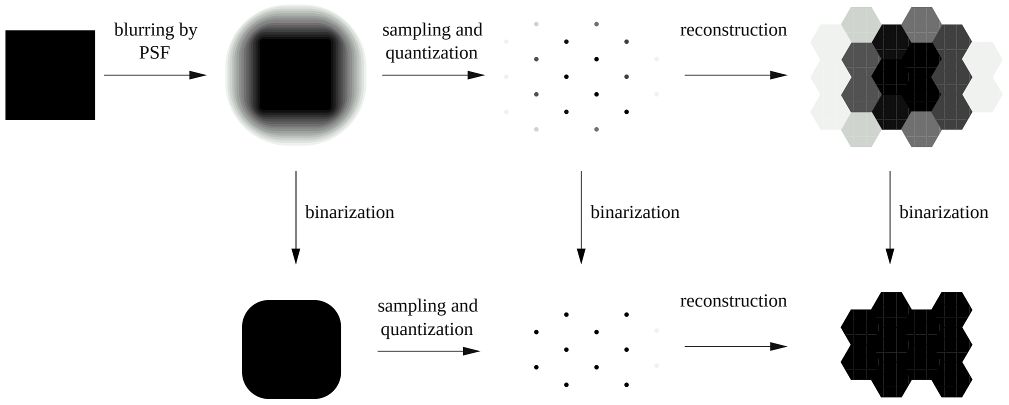


Figure 6.5: Discretization model of a real camera (top row). Since binarization commutes with sampling and reconstruction the binary reconstruction of a blurred image can be computed in several mathematically equivalent ways (bottom row). This property is independent of the grid layout (here, a hexagonal grid is used). Illustration from [Stelldinger 03].

taken in the continuous domain, directly after blurring, and the thresholded analog image is sampled and reconstructed. As figure 6.5 illustrates, the resulting reconstructions are identical in both cases. When we apply the transitions in this order, the analysis can be split into two independent parts: (1) blurring of the original image and binarization (= selection of a level set of $\tilde{f}_A$), and (2) sampling and reconstruction of the level set selected in (1). Provided that this level set fulfills the conditions of theorems 6.2 and 6.3, the results of the previous section directly apply to part (2), and only part (1) needs additional analysis.

We shall show in this section that smoothing with a flat circular PSF does not fundamentally change the statements of these theorems. A flat circular PSF of radius p is defined as

$$PSF(\vec{x}) = k_p(\vec{x}) = \begin{cases} 1 & \text{if } |\vec{x}| \leq p \\ 0 & \text{otherwise} \end{cases}$$

Flat PSFs have the advantage that the convolution with the original set can be reduced to measuring the area of overlap between the PSF and the shape A:

$$\tilde{f}_A(\vec{x}) = \frac{\|K_p(\vec{x}) \cap A\|}{\|K_p(\vec{x})\|}$$

where $K_p(\vec{x})$ is the support of the PSF, translated to point $\vec{x}$, and $\|.\|$ denotes area. Therefore, we can derive properties of the level sets of $\tilde{f}_A$ by purely geometric means. Obviously, only a $2p$-wide strip $A_p = \partial A \oplus K_p$ around the boundary ∂A (where $\oplus$ denotes morphological dilation) is relevant because the PSF does not overlap A outside this strip and $\tilde{f}_A$ will be identically 1 or 0 there. All level sets of $\tilde{f}_A$ have the following property:

Lemma 6.6. *Let $\vec{x}$ be a point on ∂A, and let $\vec{c}_1$ and $\vec{c}_2$ be the centers of the inside and outside osculating circles of radius r. Moreover, let $\vec{c}_3$ and $\vec{c}_4$ be the two points on the*

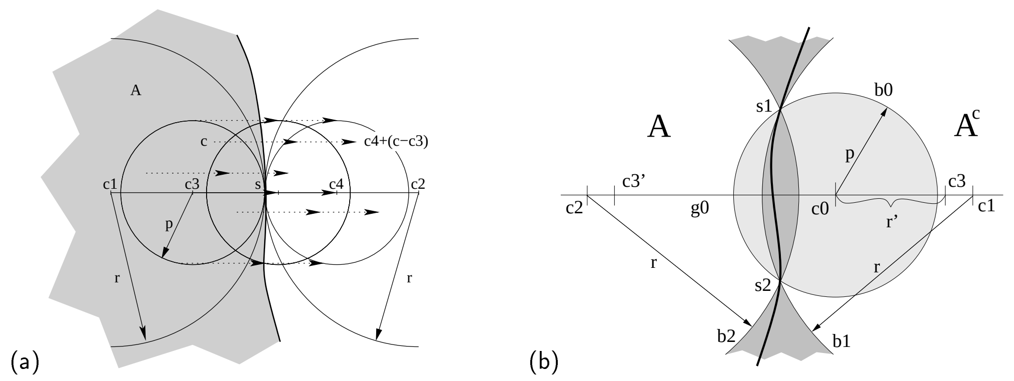


Figure 6.6: (a) If a p-disk is shifted orthogonally to the boundary ∂A from an inner osculating to an outer osculating position, its intersection area with A strictly decreases. (b) The boundary of the circle b_0 centered at point $\vec{c}_0$ (light gray) intersects the boundary ∂A (bold line) at the two points $\vec{s}_1$ and $\vec{s}_2$. Since A is r-regular, its boundary can only lie within the area marked with dark gray.

normal $\overline{\vec{c}_1\vec{c}_2}$ with distance p from $\vec{x}$. Then the boundary of every level set has exactly one point in common with $\overline{\vec{c}_3\vec{c}_4}$.

Proof. Consider a point $\vec{c}$ in $K_p(\vec{c}_3)$ and translate the line segment $\overline{\vec{c}_3\vec{c}_4}$ by $\vec{c}-\vec{c}_3$ (see figure 6.6a) . Because of the restricted curvature of ∂A, the translated line segment intersects ∂A at exactly one point. Thus, as $t \in [0,1]$ increases, the area of $\|K_p(\vec{c}_3+t\cdot(\vec{c}_4-\vec{c}_3))\cap A\|$ is strictly decreasing. This area is proportional to the result of the convolution, so the same holds for the gray values. Since the p-disk centered in $\vec{c}_3$ is an inside osculating disk of A, the gray value at $\vec{c}_3$ is $f(0) = 1$. Likewise, $f(1) = 0$. This implies the lemma. □

The curvature of the level set contours is bounded by the following lemma:

Lemma 6.7. *Let $\vec{c}_0 \in A_p$ be a point such that $\tilde{f}_A(\vec{c}_0) = (\chi_A \star k_p)(\vec{c}_0) = l$, $(0 < l < 1)$. Thus, $\vec{c}_0$ is on the boundary of level set $L_l = \{\vec{x} : \tilde{f}_A(\vec{x}) \geq l\}$. Then there exists a circle b_{out} of radius $r_o \geq r' = r - p$ that touches $\vec{c}_0$ but is otherwise completely outside of L_l. Likewise, there is a circle b_{in} with radius $r_i \geq r'$ that is completely within L_l.*

Proof. Consider the set $b_0 = K_p(\vec{c}_0)$ centered at $\vec{c}_0$. Let its boundary $\partial K_p(\vec{c}_0)$ intersect the boundary ∂A at the points $\vec{s}_1$ and $\vec{s}_2$ (see figure 6.6b). Let g_0 be the bisector of the line $\overline{\vec{s}_1\vec{s}_2}$. By construction, $\vec{c}_0$ is on g_0. Define $\vec{c}_1$ and $\vec{c}_2$ as the points on g_0 whose distance from $\vec{s}_1$ and $\vec{s}_2$ is r, and draw the circles b_1 and b_2 with radius r around them. Now, the boundary of A cannot lie inside either $b_1 \setminus b_2$ or $b_2 \setminus b_1$, because otherwise A could not be r-regular. The areas where ∂A may run are marked dark gray in figure 6.6b. Since $p < r$, there can be no further intersections between $\partial K_p(\vec{c}_0)$ and ∂A besides $\vec{s}_1$ and $\vec{s}_2$. On g_0, mark the points $\vec{c}_3$ between $\vec{c}_0$ and $\vec{c}_1$, and $\vec{c}_3{}'$ between $\vec{c}_0$ and $\vec{c}_2$, such that $|\overline{\vec{c}_1\vec{c}_3}| = |\overline{\vec{c}_2\vec{c}_3{}'}|$ and $\min(|\overline{\vec{c}_0\vec{c}_3}|, |\overline{\vec{c}_0\vec{c}_3{}'}|) = r' = r - p$. Due to the triangle inequality, and since $p < r$, such a configuration always exists. We prove the lemma for the circle b_{out} around $\vec{c}_3$, b_{in} around $\vec{c}_3{}'$ is treated analogously.

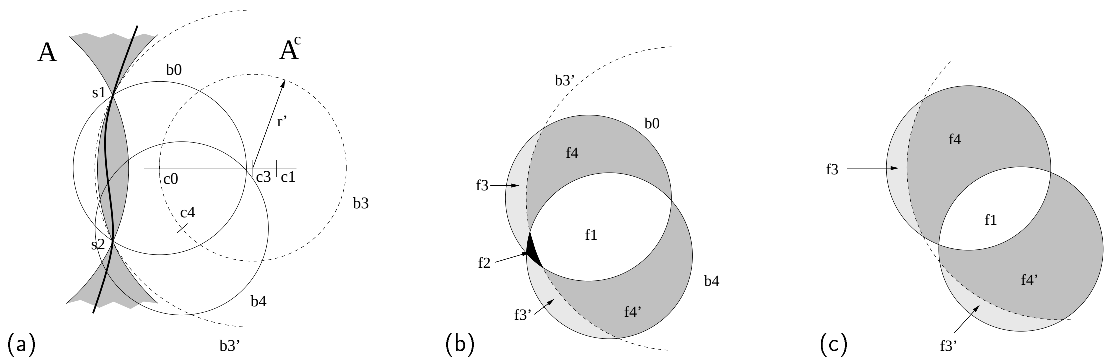


Figure 6.7: (a) The gray level at any point $\vec{c}_4 \neq \vec{c}_0$ on b_3 is smaller than the gray level at $\vec{c}_0$; (b) and (c) decomposition of the circles b_0 and b_4 into subsets (see text).

Let $b_3 = b_{\text{out}}$ be the circle around $\vec{c}_3$ with radius r', and b_3' the circle around $\vec{c}_3$ that touches $\vec{s}_1$ and $\vec{s}_2$ (figure 6.7a). Consider a point $\vec{c}_4$ on ∂b_3 and draw the circle b_4 with radius p around $\vec{c}_4$. This circle corresponds to the footprint of the PSF centered at $\vec{c}_4$. Now we would like to compare the result of the convolution $k_p \star \chi_A$ at $\vec{c}_0$ and $\vec{c}_4$. The convolution results are determined by the amount of overlap between A and $b_0 = K_p(\vec{c}_0)$ and $b_4 = K_p(\vec{c}_4)$ respectively. To compare $b_0 \cap A$ and $b_4 \cap A$, we split the two circles into subsets according to figure 6.7b (only b_0, b_4 and b_3' are shown in this figure). Circle b_0 consists of the subsets f_1, f_2, f_3, f_4, whereas b_4 consists of f_1, f_2, f_3', f_4'. The subsets f_1 and f_2 are shared by both circles, while due to symmetry f_3, f_3' and f_4, f_4' are mirror images of each other. In terms of the subsets, we can express the convolution results as follows:

$$(k_p \star \chi_A)(\vec{c}_0) = \frac{\|f_1 \cap A\| + \|f_2 \cap A\| + \|f_3 \cap A\| + \|f_4 \cap A\|}{\|K_p\|}$$
$$(k_p \star \chi_A)(\vec{c}_4) = \frac{\|f_1 \cap A\| + \|f_2 \cap A\| + \|f_3' \cap A\| + \|f_4' \cap A\|}{\|K_p\|}$$

By straightforward algebraic manipulation we get:

$$\|K_p\| \left((k_p \star \chi_A)(\vec{c}_0) - (k_p \star \chi_A)(\vec{c}_4)\right) = \|f_3 \cap A\| - \|f_3' \cap A\| + \|f_4 \cap A\| - \|f_4' \cap A\| \quad (6.1)$$

Since the radius of b_3' is smaller than r, and its center $\vec{c}_3$ is between $\vec{c}_0$ and $\vec{c}_1$, the boundary $\partial b_3'$ intersects ∂A only at s_1 and s_2. It follows that subset f_3 is completely inside of A, whereas f_4' is completely outside of A. Hence, we have $\|f_3 \cap A\| = \|f_3\| = \|f_3'\|$ and $\|f_4' \cap A\| = 0$. Inserting this into (6.1), we get

$$\|K_p\| \left((k_p \star \chi_A)(\vec{c}_0) - (k_p \star \chi_A)(\vec{c}_4)\right) = \|f_3'\| - \|f_3' \cap A\| + \|f_4 \cap A\| > 0 \quad (6.2)$$

Thus, the gray level at $\vec{c}_4$ is smaller than l. When $\vec{c}_4$ is moved further away from $\vec{c}_0$, the subset f_2 will eventually disappear from the configuration (fig. 6.7c). If $\vec{c}_3$ is outside of b_0, f_1 will finally disappear as well. It can easily be checked that (6.2) remains valid in either case. Due to the definition of $\vec{c}_3$, no other configurations are possible. Therefore, the gray values on the boundary ∂b_{out} are below l everywhere except at $\vec{c}_0$.

It remains to prove the same for the interior of b_{out}. Suppose the gray level at point $\vec{c} \in b^0_{\text{out}}$ were $l' \geq l$. By what we have already shown, the associated level line $\partial L'_l$ cannot cross the boundary ∂b_{out} (except at the single point c_0 if $l' = l$). So it must form a closed curve within b_{out}. However, this curve would cross some normal of ∂A twice, in contradiction to lemma 6.6. This implies the claim for outside circles. The proof for inside circles proceeds analogously. □

We conclude that all level sets of the blurred image $\tilde{f}_A$ are quite similar to the original shape A and fulfill the conditions of theorem 6.3:

Theorem 6.4. [Stelldinger & Köthe 05]: *Let A be an r-regular set, and L_l any level set of $k_p \star \chi_A$, where k_p is a flat disk-like point spread function with radius $p < r$. Then L_l is r'-regular (with $r' = r - p$) and strongly p-similar to A.*

Proof. The proof of r'-regularity follows directly from the definition of r-regularity and lemma 6.7. The required p-homeomorphism can be constructed as follows: Because of the restricted curvature of ∂A, the normals of ∂A cannot intersect within the p-strip A_p around ∂A (cf. [Latecki et al. 98, Latecki 98]). Therefore, due to 6.6, every point $\vec{s}$ on ∂A can be translated along its normal towards a unique point on the given level line ∂L_l and vice versa. The distance between s and its image is $\leq p$. This mapping can be extended to the entire $\mathbb{R}^2$-plane in the usual way, so that we get a p-homeomorphism with the desired properties. circles proceeds analogously. □

Combining this result with the results of the previous section concerning digitization and reconstruction of binary shapes, we finally get a sampling theorem for shapes that have been subjected to blurring with a flat disk PSF:

Theorem 6.5. [Stelldinger & Köthe 05]: *Let A be an r-regular set, L_l any level set of $k_p \star \chi_A$, where k_p is a flat disk-like point spread function with radius $p < r$, and S a grid with maximum pixel radius $r'' < r - p$. The S-reconstruction $\hat{L}_l$ of L_l is strongly $(p + r'')$-similar to A.*

Proof. By theorem 6.4, L_l is r'-regular and there exists a p-homeomorphism between A and L_l. By theorem 6.3, the S-reconstruction of an r'-regular set with an r''-grid ($r'' < r'$) is strongly r''-similar to the original set. In other words L_l and $\hat{L}_l$ are r''-homeomorphic. Consequently, there is at least a $(p + r'')$-homeomorphism between A and $\hat{L}_l$. □

This result is closely related to the findings of [Latecki et al. 98] regarding v-digitization. Recall that they proved that $d < r/\sqrt{2}$ must hold for the pixel spacing d of a square grid in order to preserve the topology of all r-regular shapes for all choices of v. If we express their result in terms of the pixel radius $r' = d/\sqrt{2}$ and the radius of the PSF $p = d/\sqrt{2}$, we get $r' + p < r$ which is exactly the same formula as we obtained for circular PSFs of radius p.

Unfortunately, the generalization of these results to non-flat PSFs is problematic. The analysis of level-line properties after smoothing becomes now much more difficult, because their geometry depends in complicated ways on exactly which part of the PSF overlaps the shape. We were able to derive partial results in [Stelldinger & Köthe 06] for so-called p-PSFs:

Definition 6.3. *A* p-point spread function $P_p(\vec{x})$ *is a function with the following properties:*

1. *It has unit integral* $\int_{\mathbb{R}^2} P_p(\vec{x})\, d\vec{x} = 1$ *in order to preserve signal energy.*
2. *It is radially symmetric* $P_p(\vec{x}) = P_p\left(|\vec{x}|\right)$.
3. *It is non-negative and compactly supported in a disc with radius* p*:* $P_p(\vec{x}) \geq 0$ *if* $|\vec{x}| < p$ *and* $P_p(\vec{x}) = 0$ *otherwise.*

p-PSFs are good approximations of real cameras as long as the error made by setting the PSF to zero outside a suitably chosen p-disk can be neglected. Using this concept we proved the following theorem:

Theorem 6.6. *Let* S *be a square, hexagonal, or triangular grid with pixel radius* r'. *Furthermore, let* A *be an* r*-regular shape,* $P_p(\vec{x})$ *an arbitrary* p*-PSF such that* $r' + p < r$ *and* $p < 1.1651r'$ *(square grid),* $p < 1.80191r'$ *(hexagonal grid), and* $p < 0.480269r'$ *(triangular grid). The blurred analog image* $\tilde{f}_A$ *is the convolution of the PSF with* A*'s indicator function* $\tilde{f}_A = P_p \star \chi_A$. *Then it holds for all thresholds* t *that the* S*-reconstruction* $\hat{A}_t$ *of the level-set* $\tilde{f}_A \geq t$ *is weakly* $(r'+p)$*-similar to* A*, that is* A *and* $\hat{A}_t$ *are topologically equivalent, and their Hausdorff distance is at most* $r' + p$.

The proof can be found in [Stelldinger & Köthe 06]. We conjecture that the restrictions on p relative to r' are not necessary, but we have not yet been able to prove this conjecture. But even without these restrictions the conditions of the theorem (r-regularity of the input shape, error-free images) limit its applicability to real problems. Therefore, we will pursue a different approach with different proof principles in the next section.

6.2 Sampling Analysis of Boundary Representations

In the previous section we have shown that pixel-based region reconstructions can only be guaranteed to be structure preserving when the original plane partition is r-regular. Unfortunately, this is a quite unrealistic condition in real scenes: most natural objects have corners, and even if they don't they may still occlude each other, so that the geometric projection contains T-junctions. These configurations cannot be handled by a model based on r-regularity. Moreover, the theory is not yet able to incorporate measurement errors.

It turns out that these problems are consequences of pixel-based region representations: When the reconstructed boundary $\partial\hat{A}$ is restricted to interpixel boundaries, undesirable effects caused by corners and junctions of the true shapes, or by measurement noise, are unavoidable. We can get rid of these restrictions by turning to more flexible concepts than interpixel boundaries. Suitable *adaptive* boundary representations (in the form of polygonal GeoMaps) have been introduced in sections 4.3.1, 5.1 and 5.3. For the purpose

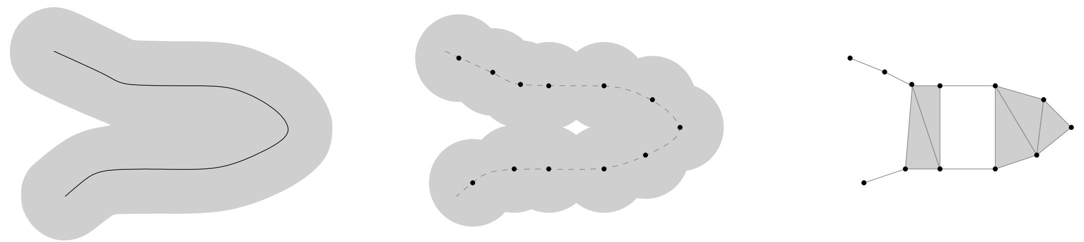


Figure 6.8: Left: The black curve and its dilation with an α-ball (gray region) have the same homotopy type, i.e. the curve is α-stable (definition 6.4). Center: When edgels (black) are sampled from the curve, the homotopy types of the curve and of the dilation of the edgels (gray region) may be different (note the hole). Right: The α-shape of the edgels is guaranteed to have the same homotopy type as the dilation of the edgels (theorem 6.7).

of sampling analysis, the notion of a (p, q)-*boundary sampling* (definition 5.3) is particularly suitable. Recall that a (p, q)-boundary sampling of a plane partition with boundary B is a set of edge points (edgels) such that the distance of the true boundary B to the nearest edgel does not exceed p, and the distance from each edgel to the nearest point of B does not exceed q, see figure 5.14 in section 5.3.

In contrast to representations where boundaries are restricted to pixel borders, adaptive boundary representations can recover subpixel boundary information because they can take advantage of higher order interpolation methods, whereas inter-pixel boundaries are implicitly based on nearest-neighbor interpolation. Adaptive boundaries utilize more information from the data and are therefore more robust against effects caused by accidental grid properties (such as grid anisotropy or connectivity paradoxes). On this basis, we are going to prove a much more general sampling theorem that applies to a much larger class of shapes (r-stable instead of r-regular, see below) and incorporates measurement errors. The presentation in this section follows [Stelldinger et al. 06, Köthe et al. 06].

Suppose a (p, q)-boundary sampling is given. Then we first compute the α-shape of the given points according to definition 5.5. The topology of the α-shape is related in a fundamental way to the dilation of the edgels:

Theorem 6.7. *Let S be a set of edgels (i.e. points near the edge of a shape to be reconstructed), and $|D_\alpha|$ the α-shape of S (i.e. the union of the cells in the α-complex). Then the union S_α of closed α-discs centered at the points $s_i \in S$ covers $|D_\alpha|$, and the two sets S_α and $|D_\alpha|$ are of the same homotopy type.*

The theorem is proved in [Edelsbrunner 95]. When the points in S are sampled exactly from the boundary B of a plane partition P (i.e. there are no measurement errors), the α-shape $|D_\alpha|$ is a structure preserving reconstruction of P if and only if the dilation of the edgels with α-discs is of the same homotopy type as B. This requirement is indeed fulfilled in certain situations: In [Bernardini & Bajaj 97] it is proved that $|D_\alpha|$ is even homeomorphic to B if P is an r-regular partition with $p < \alpha < r$ and $q = 0$. Unfortunately, this no longer applies when the original partition is not r-regular and/or the edgels are not exactly on the original boundary. Figure 6.8 shows an example where

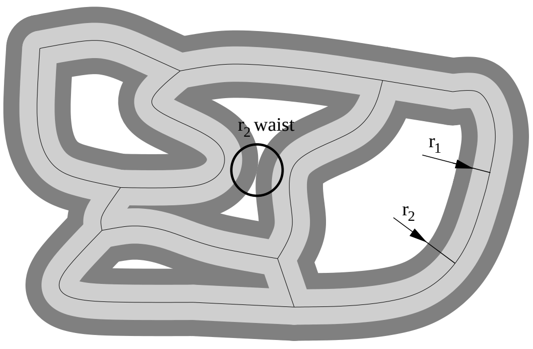


Figure 6.9: The homotopy type of an r-stable plane partition (black lines) will not change when its boundary is dilated with a disc of radius $r_1 \leq r$ (light gray), while dilations with a larger radius $r_2 > r$ (dark gray) may merge different arcs as marked by the circle. If this happens, we say that the region has an r_2-*waist*.

the r-dilation of B has the same homotopy type as B itself, but the r-dilation S_r of the edgels has different homotopy.

In order to relax the requirement of r-regularity and permit shapes with corners and junctions, the concept of r-*stability* is of fundamental importance:

Definition 6.4. *A plane partition P with boundary B is* r-stable *when B can be dilated with a closed disc of radius ρ without changing its homotopy type for any $\rho \leq r$.*

Figure 6.9 illustrates this shape characterization. Just like r-regularity, r-stability ensures that regions have a certain minimal size, because regions are not allowed to split up into several components by an r-dilation of the boundary. In addition, corners and junctions are now allowed, because only the homotopy type has to be preserved, whereas topological equivalence between the original boundary and its dilation is not required.

When S is a set of edgels which sample the boundary B sufficiently well (i.e. with sufficiently small p and q), theorem 6.7 ensures that the topology of the α-shape resulting from S (for some suitable α) will be very similar to the topology of B. But figure 6.8 makes it clear that the structure is not always completely preserved: the α-shape may contain spurious holes which do not correspond to any region of P. This is the reason for the introduction of (α, β)-holes in definition 5.5: for suitable β, the holes not containing a Delaunay triangle with circumradius of at least β are exactly the spurious holes. In order to prove this and to derive precise values for α and β, we need the following lemma:

Lemma 6.8. *An α-hole h is an (α, β)-hole if and only if it contains a point v whose distance from the nearest edgel is at least β.*

Proof. **I.** ($d_H(v \in h, S) \geq \beta \Rightarrow h$ is an (α, β) -hole): when v is in the infinite region, the claim follows immediately. Otherwise, v is contained in some Delaunay triangle. By assumption, the corners of this triangle must have distance $\geq \beta$ from v. Hence, the triangle's circumradius must be at least β and the claim follows.

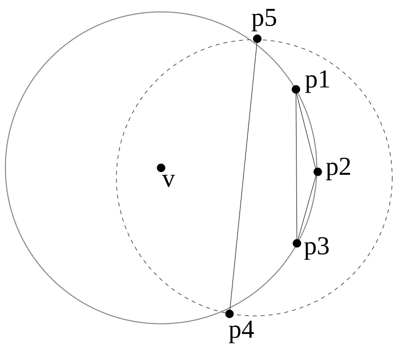


Figure 6.10: Illustration of the proof of lemma 6.8 (see text).

II. (h is an (α, β)-hole $\Rightarrow \exists\, v \in h$ with $d_H(v, S) \geq \beta$): by assumption, the closure of h contains a Delaunay triangle t with circumradius of at least β. Consider the center v of its circumcircle. If it is within the triangle t, it is also in h and the claim follows. Otherwise, it is at least in some (α, β)-hole, and we must prove that t is in the same hole. Suppose to the contrary that v and t are in different α-holes. Then there exists a Delaunay triangle t' or a single edge e between t and v whose smallest circumcircle is smaller than α. The corners of t' or e cannot be inside t's circumcircle since it is a Delaunay triangle. Neither t' nor e can contain v because their circumcircle radius would then be at least β. Now consider the illustrated triangle p_1, p_2, p_3 and its circumcircle (gray) with center v. The points p_4 and p_5 are the end points of e or of one side of t'. Their distance $|p_4p_5|$ must be greater than $|p_1p_3|$. Consequently, any circumcircle with radius $\leq \alpha$ (dashed) around p_4 and p_5 contains t, contrary to the condition (imposed by the definition of an α-complex) that it must not contain any other edgel. The claim follows from the contradiction. □

Now recall algorithm 5.12 for (α, β)-reconstruction. It constructs the α-complex from a given set S of edgels and then fills all (α, β)-holes, i.e. all holes not containing a triangle with circumradius of at least β. The remaining holes correspond to the regions of the original plane partition, and the resulting (α, β)-reconstruction has the same topological structure as the original plane partition. This is shown in the following theorem, which thus explains why algorithm 5.12 is defined in that particular way:

Theorem 6.8. *Let P be an r-stable plane partition with boundary B, and S a (p, q)-sampling of B. Then the (α, β)-boundary reconstruction $\hat{B}$ derived from S according to algorithm 5.12 has the same homotopy type as B, and the (α, β)-holes of $\hat{B}$ are topologically equivalent to the regions r_i of P, if*

1. $p < \alpha \leq r - q$
2. $\beta = \alpha + p + q$
3. *every region r_i contains an open γ-disc with $\gamma \geq \beta + q > 2(p + q)$.*

Proof. Let U be the union of open α-discs centered at the points of S. Furthermore, let $B^{\oplus} = B \oplus \mathcal{B}^o_{\alpha+q}$ be the dilation of B with an open $\alpha + q$-disc, and $r_i^{\ominus} = r_i \ominus \mathcal{B}_{\alpha+q}$ the erosion of region $r_i \in P$ with a closed $(\alpha + q)$-disc.

- According to the definition of a (p, q)-sampling, the dilation of B with a closed q-disc covers S. Consequently, $B^{\oplus}$ covers U. Therefore, U cannot have fewer connected components than $B^{\oplus}$. $B^{\oplus}$ has as many components as B due to r-stability of P. Conversely, since $\alpha > p$, every open α-disc around a point of S intersects B, and the union U of these discs covers B. It follows that U cannot have more components than B. The number of components of B and U is thus equal. Since U and $|D_\alpha|$ are of the same homotopy type (theorem 6.7), this also holds for the components of $|D_\alpha|$.

- Since P is r-stable with $r \geq \alpha+q$, each $r_i^{\ominus}$ is a connected set with the same topology as r_i. The intersection $r_i^{\ominus} \cap B^{\oplus}$ is empty, and $r_i^{\ominus}$ cannot intersect $|D_\alpha| \subset U \subset B^{\oplus}$. Hence, $r_i^{\ominus}$ is completely contained in a single α-hole of $|D_\alpha|$.

- Due to condition 3, r_i contains a point whose distance from B is at least $\gamma = \beta + q$. Its distance from S is therefore at least $\gamma - q = \beta$. Due to lemma 6.8, the α-hole which contains $r_i^{\ominus}$ is therefore also an (α, β)-hole.

- Since $B^{\oplus}$ covers U and U covers B, no (α, β)-hole can intersect both $r_i^{\ominus}$ and $r_j^{\ominus}$ $(i \neq j)$. It follows from this and the previous observation, that every region r_i can be mapped to exactly one (α, β)-hole which will be denoted h_i.

- An α-hole that does not intersect any region $r_i^{\ominus}$ must be completely contained within $B^{\oplus}$. Every point $v \in B^{\oplus}$ has a distance $d < \alpha + q$ to the nearest point of B. In turn, every point in B has a distance of at most p to the nearest point in S. Hence, the distance from v to the nearest point of S is $d' < \alpha+p+q = \beta$. According to lemma 6.8, this means that an α-hole contained in $B^{\oplus}$ cannot contain a triangle with circumradius β and cannot be an (α, β)-hole.

- The previous observation has two consequences: (i) All holes remaining in $\hat{B}$ intersect a region $r_i^{\ominus}$. Therefore, the correspondence between r_i and h_i is 1-to-1, and B and $\hat{B}$ enclose the same number of regions. (ii) All differences between $\hat{B}$ and D_α (i.e. all Delaunay cells re-inserted into $\hat{B}$) are confined within $B^{\oplus}$. This implies that $\hat{B}$ cannot have fewer components than $B^{\oplus}$ and B. Since all re-inserted cells are incident to D_α, $\hat{B}$ cannot have more components than $|D_\alpha|$, which has as many components as B (see first observation). Hence, B and $\hat{B}$ have the same number of components.

- Consider the components of the complement $(r_i^{\ominus})^C$ and recall that $r_i^{\ominus}$ is a subset of both r_i and h_i for any i. Since B and $\hat{B}$ have the same number of components, it is impossible for h_i^C to contain a cell that connects two components of $(r_i^{\ominus})^C$. This means that the sets r_i^C and h_i^C have the same number of components. This finally proves the topological equivalence of r_i and h_i, and implies that B and $\hat{B}$ have the same homotopy type.

□

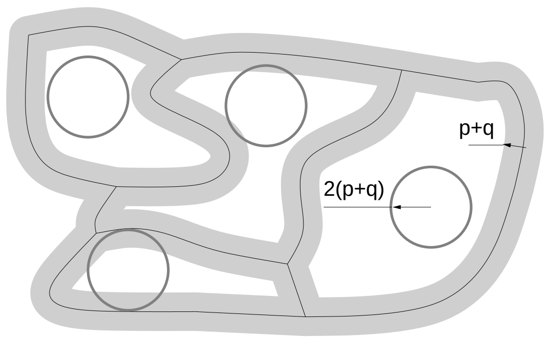


Figure 6.11: The intuitive meaning of the bounds in theorem 6.8: The boundary (black) must be $(p+q)$-stable (gray region), and every region must contain a circle with radius $2(p+q)$, where p and q are the maximum errors of the boundary sampling.

To state this result in a more intuitive way, we can say that topology-preserving (α, β)-reconstruction of a plane partition P is possible, when (i) the plane partition is r-stable with $r > p + q$ and (ii) each region of P is large enough to enclose a circle with radius $2(p + q)$, where p and q are the maximum errors of the given boundary sampling, see figure 6.11. Due to the requirements $r > p + q$ and $\alpha > p$, an edge detector with very different values for p and q is not very useful: The bigger of the two errors bounds will dominate the performance, and the high accuracy in the other will largely be wasted. This is especially a problem with Canny's edge detector, which may achieve very small q (below 0.2 pixels) whereas p is about 0.73 because the algorithm computes at most one edgel per pixel (see section 7.6).

If no r exists that meets all conditions of theorem 6.8 for a given plane partition (or if α is chosen too big), structure preservation is no longer guaranteed. Very small regions may get lost in the reconstruction. A region that gets split into two or more parts by an ρ-dilation of the boundary (i.e. has a *ρ-waist*) with $\rho < \alpha$ may also become disconnected in the reconstruction. In case of very small waists, i.e. when $\rho + 2p + 2q \leq \alpha$, this is even guaranteed to happen. Thus, we can still apply the new boundary sampling theorem: we modify the original plane partition by connecting the two sides of every small waist by a new arc. When the modified partition fulfills the requirements of the theorem, the modified topology is always preserved, and the difference between the modified reconstruction and the original plane partition is well defined. Thus, our theorem precisely predicts the topological errors being made by the reconstruction.

Figure 6.12 illustrates the principle of (α, β)-reconstruction and the meaning of its parameters. More examples can be found in chapter 7. The parameters α and β are essentially thresholds on the region size. Unlike many other thresholds, optimal values for these thresholds can be derived from an error analysis of the segmentation algorithm, as we will demonstrate below. There are two reasons why r-stable plane partitions are sufficient for topology preserving (α, β)-reconstruction, whereas r-regular partitions are required in the context of pixel-based region reconstruction:

- Edgels can be placed more freely than just at the pixel corners, so that the boundary

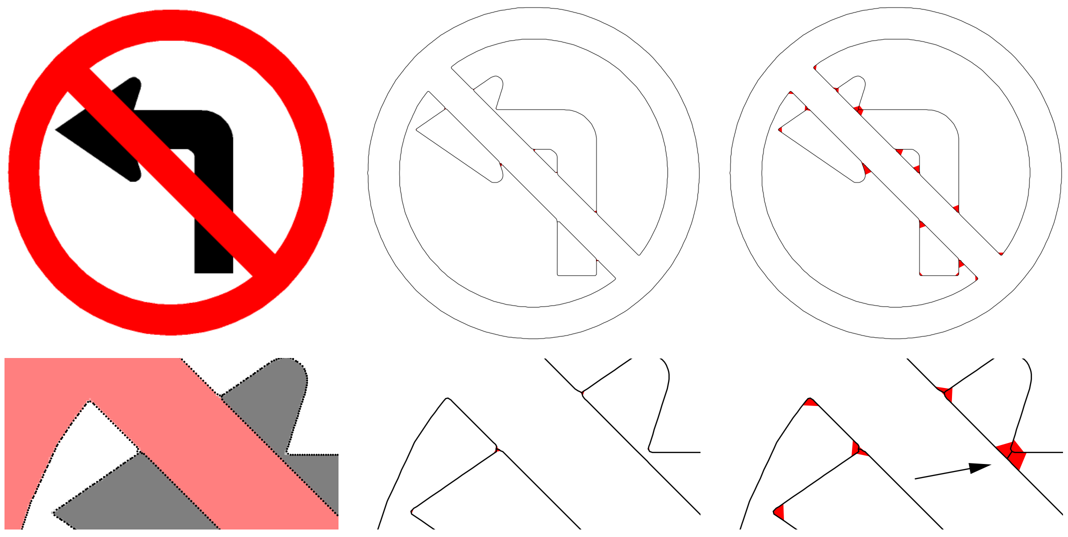


Figure 6.12: Illustration of (α, β)-reconstruction. Edgels are shown as black dots in the original image (left). Reconstructions have been done with and without subsequent boundary thinning according to algorithm 5.13 (black contours alone and black plus red areas respectively). The reconstructions use $\alpha = 1.6$, $\beta = 3.7$ (center) and $\alpha = 4.5$, $\beta = 6.7$ (right). The right images exhibit a connectivity error (indicated by the arrow) because α is too big here.

representation is more accurate to begin with.

- The (α, β)-reconstruction is not required to create thin boundaries (e.g. with vanishing area), but may contain triangles.

The latter relaxation has become possible by a corresponding relaxation in the definition of a "correct reconstruction": The (α, β)-reconstruction is required to be structure preserving (i.e. has a boundary of correct homotopy type), whereas a pixel-based digitization according to theorem 6.2 is topologically equivalent (homeomorphic) to the original plane partition which is a much stronger requirement. This leads to an interesting property of (α, β)-reconstruction: Whenever the resolution or accuracy of the edgels is locally insufficient for a crisp definition of the reconstructed boundary, reconstruction of an exact boundary location is not attempted because artifacts cannot be ruled out. Instead, artifacts are avoided by leaving a thick boundary, and thick boundaries clearly signal problems with the edgel accuracy. This ability for self-diagnosis is nicely illustrated by the red areas in figure 6.12 center and right.

When a thick boundary representation is undesirable and no special treatment of uncertain areas is required, one can apply topology-preserving thinning according to the minimal boundary reconstruction algorithm 5.13. Thinning results are illustrated in figure 6.12 (black lines in red areas) and in figure 6.13. Since the minimal boundary reconstruction is the shortest possible one with correct topology, surviving edges connect edgels closest to each other. Neighboring edgels align in an optimal way on the thinned boundary. The length $d_{\max}$ of the longest surviving edge is a measure of the density of

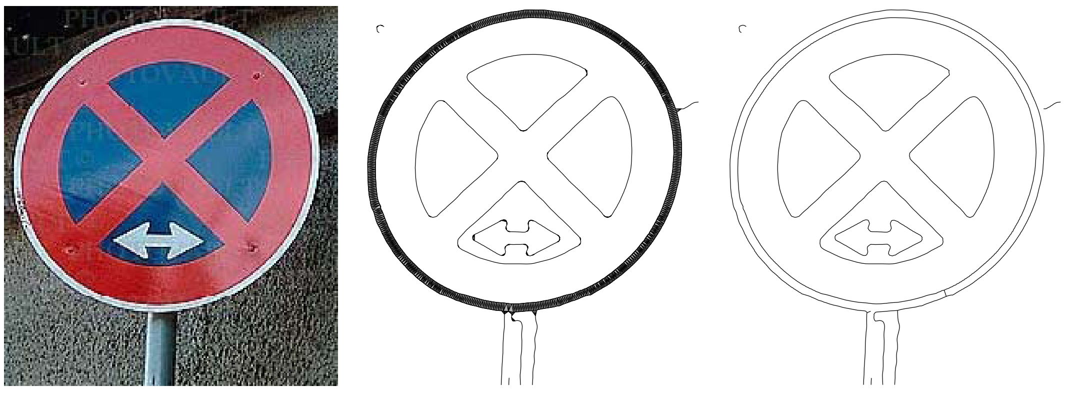


Figure 6.13: Left: real input image. Center: edgel detection with the subpixel Canny algorithm (9-point parabola fit) and its (α, β)-reconstruction. The outer white ring of the street sign cannot be resolved at the given value of α and appears as a single thick boundary. Right: Tolology-preserving thinning recovers the true boundaries.

the boundary sampling.

The maximum distance p from the true boundary to the nearest edgel may be much larger than $d_{\max}/2$ if the displacements of neighboring edgels are highly correlated. This often occurs in practice: For example, Canny or watershed edgels along a circular arc are consistently biased toward the concave side of the true curve. An (α', β)-reconstruction of the edgel set with $\alpha' = d_{\max}/2 + \epsilon < p$ and arbitrarily small ϵ is still correct in the sense of theorem 6.8: Since the minimal reconstruction is a subset of the (α', β)-reconstruction, no true regions can get merged. Since $\alpha' < \alpha$, no region can get lost, and since β remains unchanged, no additional holes can be created. In fact, $\beta' = \alpha' + p + q < 2p + q$ would have been sufficient.

We found experimentally that undesirable holes (α-holes that are not (α, β)-holes) are actually quite rare, and their largest triangles are hardly ever as large as the maximal possible circumradius β allows. Therefore, an (α', β')-boundary reconstruction with β' even smaller than $\alpha' + p + q$ often produces the correct region topology. We are currently investigating the conditions which permit weaker bounds. This is important, because a smaller β leads to a correspondingly reduced γ, i.e. the required size of the original regions is reduced, and more difficult segmentation problems can be solved correctly.

6.2.1 Application to Grid-Based Boundary Digitization Schemes

For further illustration, let us apply theorem 6.8 to the grid-based boundary digitization schemes introduced in section 4.3.2. In these methods, the grid imposes constraints on permitted edgel locations, so optimal error bounds are in general not achievable.[2] Nevertheless, analysis of these methods is instructive to understand how the error bounds p and q are derived, and how rounding to grid-based coordinates influences these errors.

Let's first look at grid intersection digitization according to definition 4.12. For sim-

[2]The localization errors of subpixel schemes will be analyzed in section 7.2.2.3.

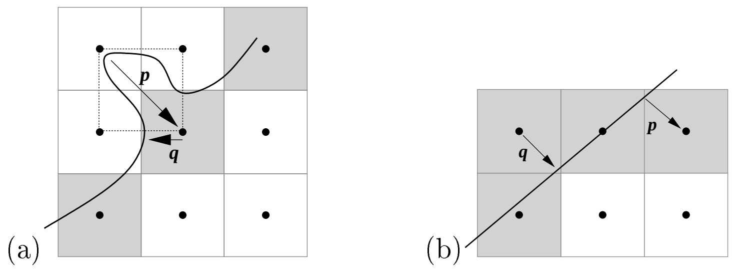


Figure 6.14: (a) The grid intersection digitization (defined by the centers of the gray pixels) of a curve whose error bounds p and q are close to the worst case $\sqrt{2}$ and $\frac{1}{2}$ respectively. (b) The same for supercover digitization with $p \approx \sqrt{2}/2$ and $q \approx \sqrt{2}/2$.

plicity, let the grid size (i.e. the smallest distance from one sampling point to another) be unity. When each component of the boundary B crosses at least one grid line (i.e. no region is so small that it fits in a single dual pixel), the distance p of any point of B to the nearest selected grid point is less than $\sqrt{2}$, and the distance q of any grid intersection to its rounded coordinate cannot exceed 1/2, see figure 6.14a. Inserting this into the conditions of theorem 6.8, we get $\alpha \geq \sqrt{2}$, $r \geq \sqrt{2} + \frac{1}{2}$, $\beta \geq 2\sqrt{2} + \frac{1}{2} \approx 3.3$, and $\gamma \geq 2\sqrt{2} + 1 \approx 3.8$. This means that every region must contain a circle with area of at least 46 pixels. However, the worst case configurations giving rise to the values of β and γ in the theorem cannot actually occur in a square grid because Delaunay edges between grid points cannot have arbitrary length. It can be shown that the largest circumradius in an undesirable α-hole is below $\sqrt{34} \approx 2.9$, so that $\gamma \approx 3.4$ (circle area 37 pixels) is sufficient.

The grid intersection digitization is a subset of the supercover digitization (definition 4.13). On a square grid, the latter turns any connected planar curve into a 4-connected digital curve. When pixels are Voronoi cells (cf. definition 4.9), the worst errors occur when a curve that just touches the corner of the biggest pixel. When the radius of this pixel is g, this implies that $p = g$ and $q \leq g$. Hence, $\alpha > g$, $r > 2g$, $\beta > 3g$ and $\gamma > 4g$ are required. On a unit square grid we have $g = \sqrt{2}/2$ and thus $q \leq p = \sqrt{2}/2$, $\gamma > 2\sqrt{2} \approx 2.8$, see figure 6.14b. Thus, the supercover digitization imposes weaker constraints on the original plane partition P than the grid-intersection digitization. This is mainly due to denser sampling of the boundary (smaller spacing of the edgels). As stated in [Ronse & Tajine 00], the supercover digitization is a Hausdorff discretization, i.e. it selects those grid points that minimize the Hausdorff distance $\max(p, q)$ between the set of edgels and the boundary B. Thus, the stated bounds for α, β and γ are sufficient conditions for any Hausdorff discretization.

Finally, it also possible to derive bounds for subset digitization where a region is represented by the union of the pixels whose sampling points are located in the region. In order to apply theorem 6.8, we need to use either the crack edges or mid-crack edges, see definition 4.11. The edgels are located at the pixel corners or at the intersections between cracks and grid lines respectively. Let the maximal pixel radius of the grid be g. Then, the maximal distance q of any edgel to the nearest boundary point cannot exceed

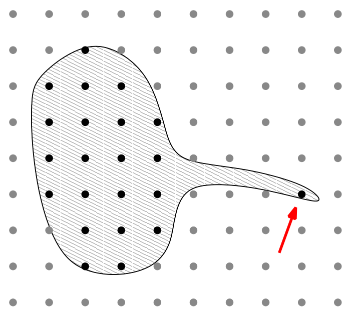


Figure 6.15: An r-stable region (hatched) may have a spike which causes its subset digitization (black circles) to be disconnected in the 4- or 8-neighborhood sense (note the isolated point marked with an arrow).

g in either the midcrack or endcrack digitizations, but the distance p from any boundary point to the nearest edgel can be arbitrary large: The grid reconstruction $\hat{R}$ of an r-stable region R is not in general topologically equivalent to the closure of R and may even be disconnected. The distance between the components of $\hat{R}$ may approach the diameter of R when R has a long narrow spike, see figure 6.15. Obviously, this is not a desirable bound for the value of p. We need a restriction that is stronger than r-stability, but weaker than r-regularity and which prevents these undesirable spikes:

Definition 6.5. *Let P be a plane partition with boundary B. We say two points $x1, x2 \in B$* delimit a (θ, d)-spike*, if the Euclidean distance from x_1 to x_2 is at most d and if every path on B from x_1 to x_2 contains at least one point with $\angle x_1 y x_2 < \theta$. We say that P is* free of (θ, d)-spikes *if for any pair of boundary points $x_1, x_2 \in B$ with distance of at most d, there exists a path $Y \subset B$ between x_1 and x_2 such that $\angle x_1 y x_2 \geq \theta$ for all points $y \in Y$.*

Intuitively, two points delimit a (θ, d)-spike, if the shortest boundary path between them does not differ too much from a straight line, i.e. it lies inside the shaded region in figure 6.16. Note that r-regular partitions have no (θ, d)-spikes for $d \leq r$ and $\theta = 2 \arctan\left(d/(2r - \sqrt{4r^2 - d^2})\right)$ (e.g. for $\theta = 90°, 60°$ we get $d = r$ and $d = \sqrt{3}r$ respectively). By sampling sufficiently densely one can enforce the angles to be arbitrarily flat. But in general, absence of (θ, d)-spikes does not imply r-stability, so we have to require both. Due to this requirement, the boundary will now remain within a cer-

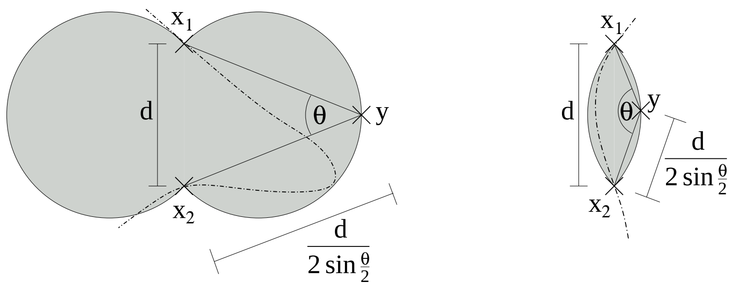

x_1
d
θ
y
x_2
$\frac{d}{2\sin\frac{\theta}{2}}$
x_1
d
θ
y
$\frac{d}{2\sin\frac{\theta}{2}}$
x_2

Figure 6.16: In order to preclude (θ, d)-spikes, the contour between the points x_1 and x_2 (dash-dotted) must remain entirely in the shaded region (left: $\theta < 90°$, right: $\theta > 90°$). It can be seen how the maximum size of spikes is thus bounded.

tain distance of the edgels, and we can now estimating p for midcrack and endcrack digitization:

Theorem 6.9. *Let G be a square grid with sample distance h (pixel radius $g = h/\sqrt{2}$), and let P be a plane partition such that every region $r_i \in P$ contains a closed g-disc and the boundary B has no (θ, d)-spikes. Then the endcrack digitization is a (p, q)-boundary sampling with $q = h/\sqrt{2}$ and $p = q + \left(\frac{h}{2} + q\right) / \sin\frac{\theta}{2}$, provided that $h \leq d/(1+\sqrt{2})$, and the midcrack digitization is a (p, q)-boundary sampling with $q = \frac{h}{2}$ and $p = q + \left(\frac{h}{2} + q\right) / \sin\frac{\theta}{2}$, provided that $h \leq \frac{d}{2}$.*

Proof. First, we prove the bounds on q. Let x, y be two 4-adjacent square grid points. Their common pixel edge is in the interpixel boundary if and only if x and y lie in different regions r_i and r_j, i.e. the grid line between x and y intersects the boundary B in at least one point v. The endcrack edgels are exactly the end points of these pixel edges, and their distance to v is at most $h/\sqrt{2}$. It follows that $q = h/\sqrt{2}$ for the endcrack digitization. The midcrack edgels are the center points between x and y, so their maximum distance to v is $\frac{h}{2}$. Hence, $q = \frac{h}{2}$ for the midcrack digitization. The maximum distance between neighboring edgels is h in both cases.

Now, we prove the bound on p given q. By definition $B = \bigcup \partial r_i$, where ∂r_i is the boundary of region r_i. Since every region contains a closed disc of radius $g = h/\sqrt{2}$, and every such disc contains at least one grid point, every region r_i contains a grid point, i.e. $\hat{r}_i$ is not empty, and there exist at least four edgels near ∂r_i. Due to the nonexistence of (θ, d)-spikes, any two components $(\partial r_i)_j$ and $(\partial r_i)_l$ of the boundary ∂r_i must have a distance of more than $d \geq 4q$. So, for every component there exists a set of edgels which are closer to $(\partial r_i)_j$ than to any other component. Obviously every component $(\partial r_i)_j$ is a closed curve. Thus by mapping every edgel to the nearest point of B, one gets a cyclic list of points $[b_k]^{(ij)}$ for every component $(\partial r_i)_j$, and each point b_k has a distance of at most $h+2q$ to its successor b_{k+1} in the list. For endcrack edgels, we have $h+2q = (1+\sqrt{2})h \leq d$, and for midcrack edgels $h + 2q = 2h \leq d$. Thus, the boundary part between b_k and b_{k+1} includes no point with an angle smaller than θ. As shown in figure 6.16, this implies that the distance from any boundary point between b_k and b_{k+1} to the nearer one of these two points is at most $\left(\frac{h}{2} + q\right) / \sin\frac{\theta}{2}$. Thus, the maximum distance to the nearest of the two edgels which are mapped onto b_k and b_{k+1} is $p = q + \left(\frac{h}{2} + q\right) / \sin\frac{\theta}{2}$. □

For example, if the grid spacing is $h = 1$ and the original plane partition has no $(60°, d)$-spikes for $d > 2.4$, we get $p = 3.12$ and $q = 0.71$ and for endcrack digitization and $p = 2.5$ and $q = \frac{1}{2}$ for midcrack digitization. It follows that midcrack digitization has slightly higher geometric accuracy than endcrack digitization and should be preferred, see figure 6.17.

6.2.2 Geometric Limitations of Pixel-Accurate Edges

We conclude the discussion of grid-based boundary representations by reporting additional results about geometric accuracy limits of subset digitization and crack edges in binary images. These limits have been studied by several authors, a comprehensive

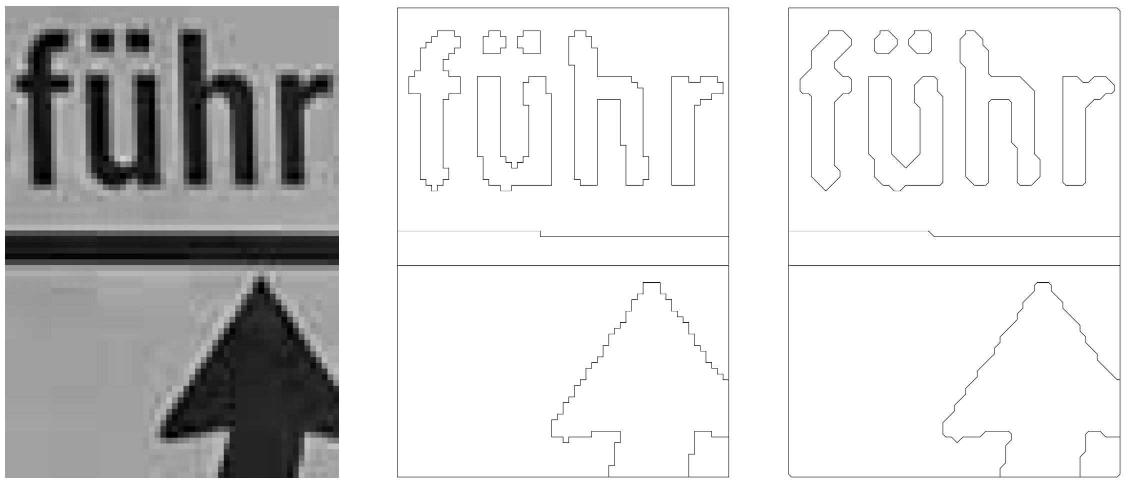


Figure 6.17: Left: original image. Center: thresholding segmentation (threshold = 100) with endcrack edges. Right: the same with midcrack edges. The latter is slightly more accurate (due to reduced staircasing), although not nearly as good as subpixel thresholding (figure 5.3 left).

overview is presented in [Klette & Rosenfeld 04]. We already observed in theorem 6.1 that one justification of subset digitization comes from the fact that the area of a digitized region converges to the true area as the sampling distance decreases. This property is called *multigrid convergence*. Unfortunately, many measurements involving the *boundary* of a digital region (i.e. the crack edges implied by the digitization) are *not* multigrid convergent.

Consider the computation of the perimeter of a reconstructed region $\hat{r}$ or, even simpler, the length of a straight line $\hat{l}$ from pixel-accurate data. First let us assume that $\hat{r}$ results from the subset digitization of a polygon. The boundary of $\hat{r}$ is an interpixel contour, i.e. consists of a sequence of horizontal and vertical lines. Therefore, the perimeter of the digitized shape will converge to the sum of L_1-distances $|\Delta x| + |\Delta y|$ between consecutive corners of the polygon, and not toward its true Euclidean perimeter. In the worst case (all polygon sides are oriented at multiples of 45°), the relative bias in the limit of infinite resolution will be $\sqrt{2} - 1 \approx 41\%$, see figure 6.18.

Defining lines by grid-intersection digitization does not fare much better. Let the reconstructed contour polygon be defined by connecting the centers of consecutive pixels in the digital contour. Since the angles between these connections and the x-axis are multiples of 45°, the bias is smaller than in the interpixel case (where only multiples of 90° occur), but still significant. In case of a straight line, the estimated length converges to $\hat{l} = \sqrt{2}\, n_d + n_i$, where $n_d = \min(|\Delta x|, |\Delta y|)$ is the number of diagonal steps between the two endpoints of the line, and $n_i = ||\Delta x| - |\Delta y||$ is the number of horizontal or vertical steps [Klette & Rosenfeld 04]. The maximum bias occurs for lines whose slope is an odd multiple of 22.5° and amounts to 8.2%. The average bias over all angles is 6.6%. The average bias can be minimized by weighting the two numbers n_i and n_d differently [Dorst & Smeulders 91]: When $\hat{l} = 0.945 n_i + 1.346 n_d$, the average over all angles becomes 2.6% in the limit. By also counting how often the slope direction

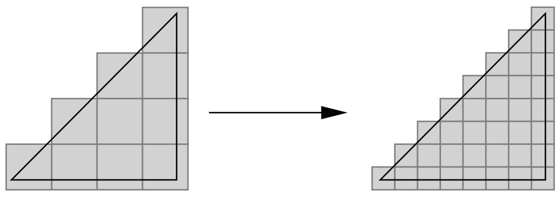


Figure 6.18: Geometric properties of the subset digitization (gray pixels) of a triangular shape (black lines): As the grid is refined, the area of the shaded region converges to the true area of the shape, but the perimeter does not converge (in this particular case, it even remains constant).

changes along the contour (call this number n_c), the estimate can be further refined as $\hat{l} = 0.980n_i + 1.406n_d - 0.091n_c$, but this still fails to converge to the true value and has an expected bias of 0.8%.

Refining boundary localization to subpixel accuracy is the natural solution for the convergence problem. If possible, subpixel refinement should be controlled by the original intensity or boundary strength data (like in Canny's algorithm 5.10) because the necessary geometric information has been recoded into intensities due to the action of the PSF (see figure 1.1 and section 3.1). But sometimes only a digital contour is available. Then the only possibility to reduce round-off errors of grid-based coordinates is by averaging over some part of the contour. In the absence of prior information about the form of the boundary (i.e. shape priors), the most popular approach is the construction of straight line segments by means of the *digital straight line* algorithm 4.7 or the *Euclidean path algorithm* [Braquelaire & Vialard 99]. These algorithms construct a set of subpixel accurate split points which define an approximate contour polygon. The length of this polygon is indeed a convergent estimate of the Euclidean length of the underlying contour or line (i.e. one sums over the Euclidean distance of consecutive split points). The speed of convergence is linear. According to experiments in [Klette & Rosenfeld 04], who investigated the error for a number of curved shapes at various resolutions, relative errors between 1% and 2% were observed at resolutions where the perimeter of the contour measured a few hundred pixels. It is even possible to achieve superlinear convergence ($\mathcal{O}\left(h^{1.5}\right)$) by using the *most probable original* length estimate $\hat{l} = n\sqrt{1 + (b/a)^2}$ where n is the number of pixels in a straight line segment, and b/a is the best rational approximation of its slope (as obtained by the above algorithm) [Dorst & Smeulders 91]. Since the line's end points do not in general coincide with pixel centers, n is a rounded value with variance 1/12 (the variance of the uniform distribution in the interval $[0, 1)$). Therefore, even if b/a were known exactly, 1% relative error would require $n \geq 29$, i.e. a line segment of almost 30 pixels. Higher accuracy requires to replace n with the exact arc length between the intersection of consecutive line segments, but apparently this possibility has not yet been investigated.

In summary, we are once again led to the conclusion that subpixel accurate boundary detectors are to be preferred. The accuracy of these detectors will be studied in detail in the next chapter.

7 The Gradient Magnitude: Detailed Error Analysis of a Boundary Indicator

Abstract

The image gradient is a very popular boundary indicator. For example, it is the basis of the Canny edge detector and of many watershed-based edge detectors. The oriented derivative of the gradient squared magnitude along the gradient direction is the Haralick edge detector. Gradients are also directly or indirectly involved in the energy functionals of many variational image segmentation methods. Therefore, this chapter is dedicated to a detailed error analysis of various aspects of gradient-based boundary detection, such as localization accuracy, orientation estimation, robustness against noise, behavior at more complex boundaries like parallel edges or junctions. Our most important finding is that theoretical error analysis correctly predicts the errors observed in experiments. Thanks to our careful modeling of all steps that eventually lead to a detected boundary, and to the powerful GeoMap framework, all major error sources are apparently accounted for. The analysis shows that the gradient is a very accurate and robust boundary indicator, but also highlights two important weaknesses: it has difficulties with junctions and with edges between shaded regions (where the step edge model is not applicable). The error analysis framework developed here can also be applied to alternative boundary detection methods, with the gradient serving as a good performance reference.

7.1 Sampling Analysis of the Gradient Magnitude

In section 3.1 we showed that a correspondence between discrete and continuous image representations can be achieved when the image acquisition process conforms to the requirements of Shannon's sampling theory. In particular, the analog camera image must be band-limited, at least effectively by making the aliasing energy significantly smaller than the noise energy. Naturally, the same requirement applies to all subsequent stages of the image analysis chain. Surprisingly, we found that this fact has been largely overlooked when gradient magnitudes are computed.

Suppose the analog image is band-limited, i.e. its spectrum is zero above a certain cut-off frequency ν_N (also called *Nyquist frequency*)

$$F(\vec{\nu}) = 0 \text{ if } |\vec{\nu}| \geq \nu_N$$

Then the analog image can be exactly reconstructed from the digital image when the

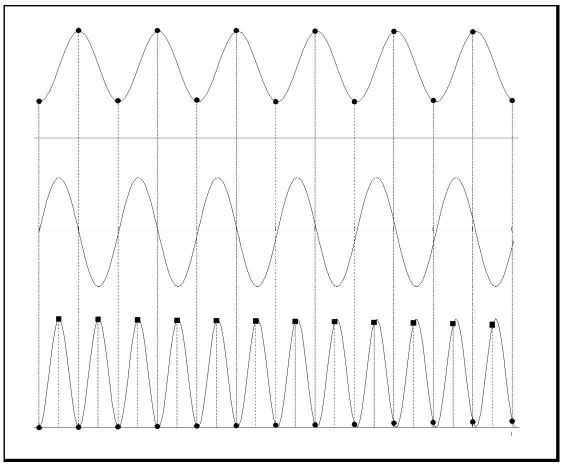


Figure 7.1: Illustration of the information loss when the squared derivative of a function is represented at the same sampling locations (circles) as the original signal. The true signal is lost unless additional samples (squares) are added. (top: sine wave $\sin(\nu)$ with $\nu = \nu_N - \varepsilon$, center: derivative $\cos(\nu)$, bottom: squared derivative $\cos^2(\nu) = \frac{1+\cos(2\nu)}{2}$)

sampling distance λ_{signal} is at most half the Nyquist wave length

$$\lambda_{\text{signal}} \leq \frac{1}{2\nu_N}$$

The original continuous function is (approximately) reconstructed by convolving the samples f_{kl} with an appropriate reconstruction filter such as the sinc or spline interpolator. The gradient of the analog image can be obtained by convolution with the first derivative of the reconstruction filter instead with the filter itself. Since computing a derivative corresponds to a multiplication in the Fourier domain, this operation does not change the band-limit of the analog image. However, when we compute the squared gradient magnitude b, we must square the responses of the linear gradient filter:

$$b(x, y) = |\nabla f(x, y)|^2$$

The Fourier domain equivalent of this operation is a convolution of the gradient's spectrum with itself. When two functions with finite support are convolved with each other, the resulting function's support is the Minkowski sum (morphological dilation) of the two original supports. This means that the cut-off frequency ν_{gradient} is *doubled*, and the sampling distance $\lambda_{\text{gradient}}$ must be halved accordingly in order to avoid information loss.

$$\nu_{\text{gradient}} = 2\nu_N$$
$$\lambda_{\text{gradient}} = \frac{\lambda_{\text{signal}}}{2}$$

This important sampling effect was first pointed out in [Köthe 03c]. A 1-dimensional illustration of the phenomenon is given in figure 7.1. Figure 7.2 demonstrates the phenomenon on a real image, more results can be found in section 7.7. Although aliasing

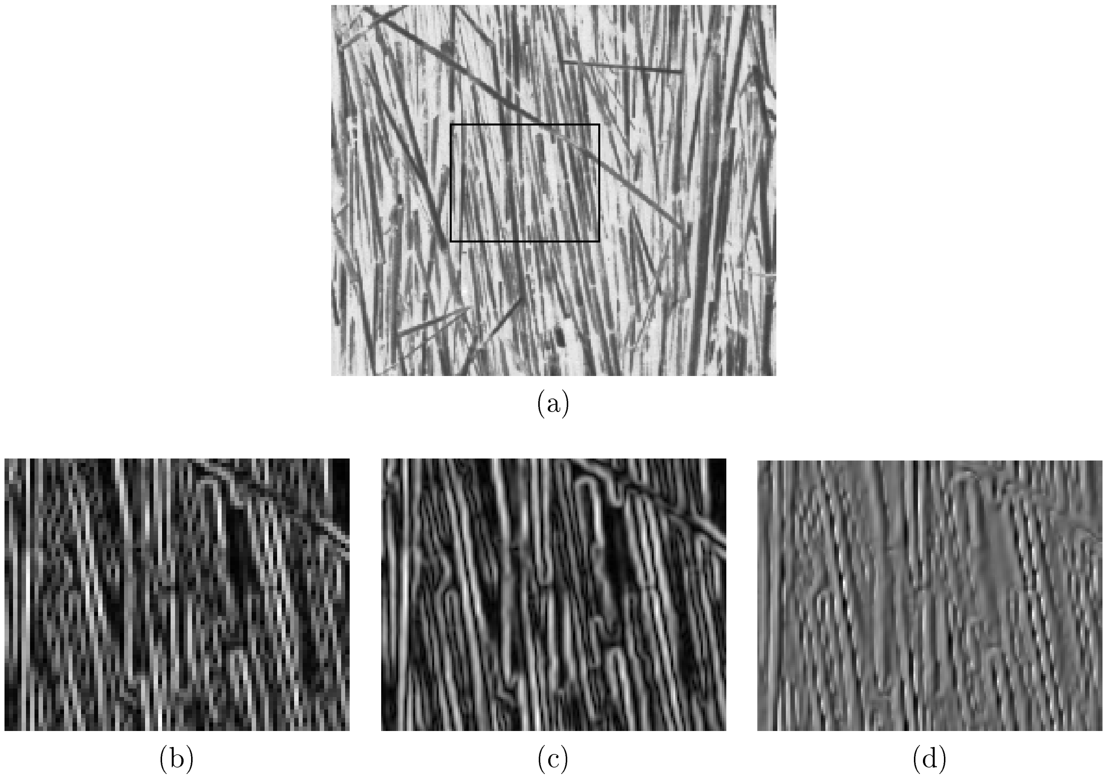


Figure 7.2: Gradient squared magnitude of a real image: (a) subregion of Brodatz texture D15 (with ROI rectangle marked); (b) gradient squared magnitude at original resolution; (c) gradient squared magnitude at doubled resolution. (d) The difference image clearly shows the high amount of aliasing noise present in the gradient magnitude at original resolution.

artifacts are not always as clearly visible as in this figure, they always degrade the results of subsequent analysis steps (such as edgel detection), unless the effective bandwidth of the original image was below half the Nyquist frequency in the first place. If the image contains small detail, we must represent the gradient squared magnitude on a raster with twice the resolution in both directions. In theory, the gradient magnitude (i.e. the square root of the squared magnitude) has an even greater band-width, but in practice the effective band-width of the square root operator is not significantly larger, so another increase in sampling density is not necessary.

If the zero-crossings of the oriented second derivative are used as a boundary indicator, the oversampling ratio must be even bigger. The numerator of the second derivative involves products of three band-limited functions

$$f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}$$

Therefore, the support of the spectra is dilated with itself two times, leading to a band-limit that is three times as large as the original band-limit. Accordingly, threefold over-sampling is required to avoid aliasing. Yet another example, namely the difference of the

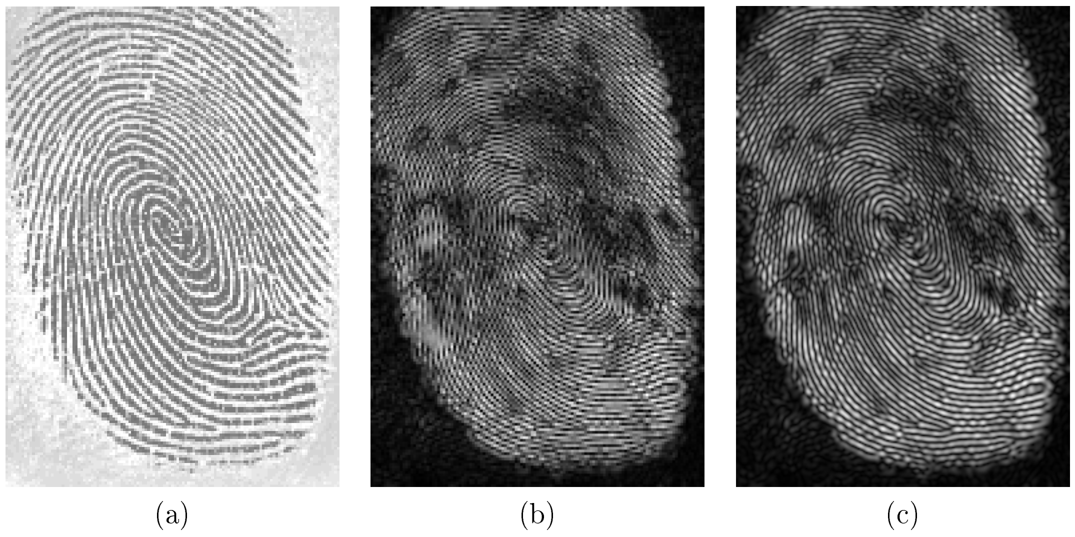


Figure 7.3: Ridge and valley detection by the difference between the eigenvalues of the Hessian of Gaussian matrix: (a) original fingerprint image; (b) eigenvalue difference at original resolution; (c) eigenvalue difference at doubled resolution. Aliasing artifacts are clearly visible in (b).

eigenvalues of the Hessian matrix

$$\lambda_1 - \lambda_2 = \sqrt{(f_{xx} - f_{yy})^2 + 4 f_{xy}^2}$$

is demonstrated in figure 7.3. This operator is a useful detector of ridges and valleys, which both appear bright, whereas edges remain dark. Aliasing artifacts are clearly visible in the center image, which has not been subjected to oversampling.

The important fact that oversampling is often necessary in order to fulfill the requirements of Shannon's sampling theorem has been overlooked in the image analysis literature so far. Given that the derivation of this conclusion is straightforward, we find this rather surprising. One reason may have been that artifacts are usually not as obvious as in figures 7.2 and 7.3, and are hard to distinguish from other sources of error. Moreover, image interpolation may appear to be useless at first sight because it doesn't reveal any new information. One has to understand that its effect is just the opposite here: it prevents *existing information from being lost* as a consequence of insufficient sampling. This is clearly an important factor in the empirically observed improvement of segmentation performance on interpolated images (see, for example, figures 1.4 and 1.5 in chapter 1).

The above analysis poses the question of how oversampling should be implemented in practice. The best method would be to perform it directly in the camera by placing sensors twice as densely relative to the effective cut-off frequency of the lens, similar to what happens in the human eye. Since such data are usually unavailable, the best solution is to interpolate the samples to a higher resolution before processing, as explained in section 3.3.1. Alternatively, one can integrate oversampling into the feature detection process itself: Since feature detection usually starts with linear filtering (e.g. derivative filters), we can design filters that also compute filter responses in between the original samples, e.g. at half integer coordinates. This is especially simple if filters are defined by a continuous function, e.g. a Gaussian derivative: The convolution of a discrete image f with a continuous filter g is written as

$$(f * g)(x, y) = \sum_{i,j} f_{i,j}\, g(x - i, y - j) \tag{7.1}$$

and we can obviously insert arbitrary real-valued coordinates on the left-hand side. This approach is slightly faster then the interpolation approach because it achieves two things in one step. On the other hand, its signal properties are slightly worse, because the sampling of the kernel in (7.1) may cause additional aliasing. This aliasing cannot occur in the interpolation approach because there the kernel is sampled after interpolation, i.e. at a higher sampling rate.

7.2 Analysis of Isolated Straight Step Edges

Recall from chapter 2 that we defined the ideal geometric image as a collection of regions r_i with indicator functions $\rho_i(x, y)$, where single continuous functions $f_i(x, y)$ describe the interior of each region, and their combination is discontinuous across region boundaries, cf. equation (2.1):

$$f_{\text{geometric}}(x, y) = \sum_i \rho_i(x, y)\, f_i(x, y)$$

If we could directly observe this ideal geometric image, the segmentation problem would be trivial because we only had to look for discontinuities. But the ideal image is always blurred by the continuous point spread function (PSF) of the optical system

$$\tilde{f} = \text{PSF} \star f_{\text{geometric}}$$

When the PSF is band-limited and there is no noise, this blurred image can be exactly reconstructed from the digitized image. However, we have to replace the discontinuity-based boundary definition with something else, because there are no discontinuities in the blurred image. A particularly important alternative is the definition of boundaries by the image gradient. Gradient-based detectors assume that boundaries are located at points where the gray-level variation has a relative maximum. In case of isolated straight step edges without noise, this assumption is exactly justified. In this context, "isolated" means that the separation between different edges is larger than the effective diameter of the PSF, so that every point in the image is influenced by at most one edge. Straight edges ensure that the 2-dimensional image function can be (locally) reduced to a one dimensional function

$$\tilde{f}(\vec{x}) = \tilde{f}_1\left(\vec{x}^T \vec{n}\right)$$

where $\vec{n}$ is the normal direction of the edge. Finally, the step edge assumption (i.e. the assumption that the f_i are constant functions) ensures – together with the assumption of a rotationally symmetric PSF – that the gradient is symmetric about the edge, so that a relative gradient maximum occurs exactly at the true edge position. The same applies to the zero-crossings of the second derivative in gradient direction. The term "gradient-based edge detector" shall refer to either of these edge definitions.

When a gradient-based edge detector is unable to find the true position of an isolated straight step edge, its implementation is apparently not an exact realization of the underlying theory. Deviations can, for example, be caused by replacing exact sinc interpolation with spline reconstruction, using discrete gradient filters instead of analog ones, or rounding results to integer coordinates. To quantify these deviations, it is natural to start with a comparison of edge detectors on artificial, noise-free test images. This is the topic of the next subsection. In later subsections, we investigate by how much edge detector performance degrades when the image becomes more complicated.

Throughout this chapter, we assume that the PSF is a Gaussian. We saw in section 3.2.3 that this is a good model for real cameras. Blurred step edges can than be described by the analytic model

$$\tilde{f}(\vec{x}) = \Phi_{\sigma_{\text{PSF}}}\left(\vec{x}^T\vec{n} + t\right) \tag{7.2}$$

where $\Phi_\sigma(u) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{u}{\sqrt{2}\sigma}\right)\right)$ is the integral of the Gaussian PSF (the so called "probit" function), $\vec{n}$ is a unit vector normal to the edge, and t is the *subpixel shift* of the edge relative to the origin of the local coordinate system. Test images for the experiments described in the present section were created analytically by equation (7.2), the coordinate origin was always in the image center, and error measurements are plotted against the angle ϕ between the true edge direction $\vec{n}$ and the x-axis.

For every edge detector, we search for the edgel nearest to the origin and test how well the value of t and the angle between $\vec{n}$ and the x-axis are reproduced at that point. It should be noted that the experiments described in this chapter only involve local measurements – we do not fit lines or other geometric primitives to collections of edgels in order to improve the estimates of the orientation or subpixel shift. Precise descriptions of the algorithms used in this chapter can be found in chapter 5.

7.2.1 Noise-Free Straight Edges

In principle, gradient-based operators should be able to detect the position and angle of noise-free isolated step edges perfectly. Here, "noise-free" does not only refer to the absence of statistical noise, but also to the absence of aliasing noise. Therefore, we use $\sigma_{\text{PSF}} = 0.9$ to ensure that the Gaussian PSF is effectively band-limited. We are interested in the question how well the (continuous) gradient-based edge model is realized by various algorithms, and start with an investigation of coordinate round-off errors in pixel-accurate edge detectors. Figure 7.4 shows experimental results. As expected, rounding leads to significant localization errors. When edges are restricted to pixel centers (pixel-accurate Canny algorithm, region growing-based watersheds) or to pixel corners (crack-edge watersheds), the expected error can easily be calculated: Round-off errors *along* the edge are undetectable, whereas round-off errors *perpendicular* to the edge are uniformly distributed within the interval of locations that get rounded to the same point. The half-width of these regions is ± 0.5 pixels for horizontal and vertical edges, and $\pm\sqrt{2}/2$ for diagonal ones. The variances of the corresponding uniform distributions are $1/12$ and

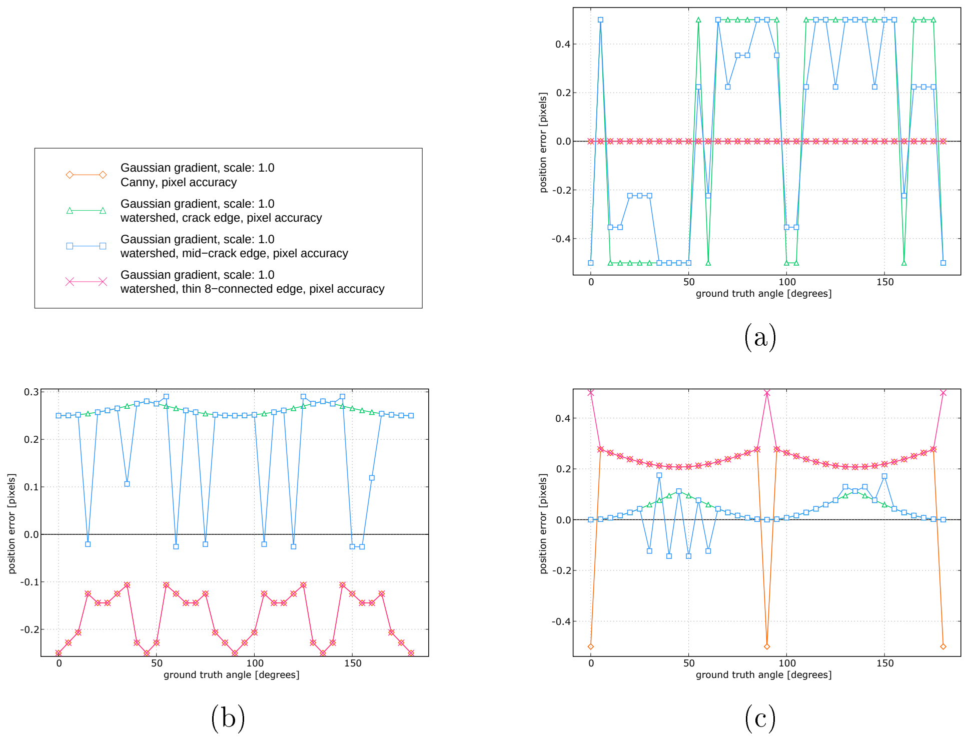


Figure 7.4: Geometric accuracy of pixel-based GeoMap creation algorithms (Canny's algorithm 5.10 without sub-pixel correction, end-crack and mid-crack edges from the watershed union-find algorithm 5.8, and thin 8-connected edges from the region-growing based watershed algorithm 5.9), with $\sigma_{\text{PSF}} = 0.9$ and $\sigma_{\text{filter}} = 1.0$. True subpixel positions are (a) $t = 0$, (b) $t = 0.25$, and (c) $t = 0.5$. Note that the red and orange curves almost always coincide because the first and last algorithms detect the same points.

1/6 respectively. In the general case of edges in direction ϕ with random subpixel shifts t, the root-mean-square (RMS) localization errors[1] are

$$\text{RMS}(x)_{\text{round off}} = \frac{\sqrt{1/12}}{\max\left(|\sin\phi|\,,|\cos\phi|\right)} \tag{7.3}$$

In the next experiment, we compare the errors of subpixel-accurate edge detectors, figure 7.5. It can be seen that the maximum localization error does never exceed 0.07 pixels, and the accuracy improves for more complex detection methods. In fact, the best methods (subpixel watersheds, and Canny's algorithm with spline-based subpixel correction) achieve errors below 0.003 pixels, i.e. are two orders of magnitude better than the pixel-accurate detectors. It is also remarkable that noise with an SNR of 200 is sufficient to turn a degenerate boundary indicator into a Morse function – recall that the

[1]The root-mean square-error of n measurements x_i with expectation $\bar{x}_i$ is defined as $\text{RMS}(x) = \sqrt{\frac{1}{n}\sum_i (x_i - \bar{x}_i)^2}$.

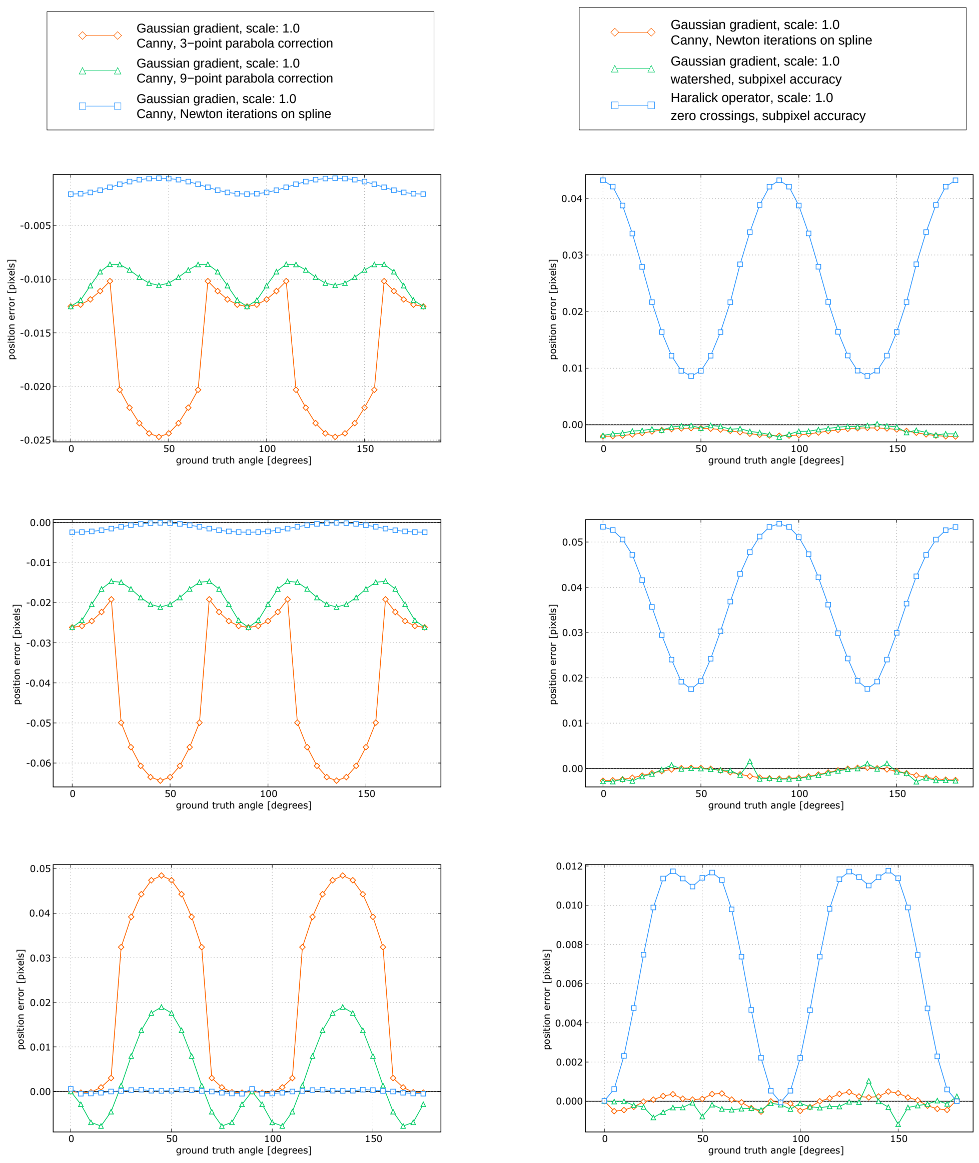


Figure 7.5: Edge position errors for subpixel-accurate edge detectors with $\sigma_{\text{PSF}} = 0.9$ and $\sigma_{\text{filter}} = 1.0$. True subpixel positions were $t = 0.1$ (top), $t = 0.3$ (center), $t = 0.5$ (bottom). Left column: Subpixel-correction methods in Canny's algorithm (3-point parabola fit, 9-point parabola fit, and Newton iterations on spline interpolation, cf. figure 5.13 in section 5.2) . Right column: Spline-based subpixel algorithms (Canny's algorithm with spline interpolation, subpixel watershed algorithm, Haralick's detector with subpixel zero crossings). Gaussian noise with signal-to-noise ratio 200 was added to these images in order to make the gradient magnitude fulfill the Morse property, definition 5.2.

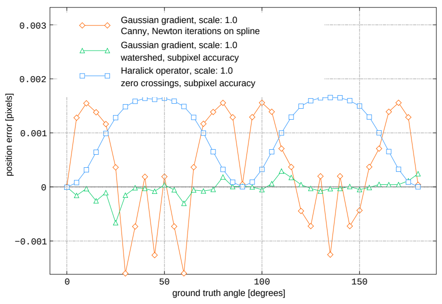


Figure 7.6: Same as figure 7.5 bottom right, but with two-fold oversampling of the original image. Note the dramatically reduced error of Haralick's detector.

Morse property (definition 5.2) is a prerequisite for the subpixel watershed algorithm to work.

Errors are largest for the 3-point parabola correction of Canny's algorithm. In addition, this operator exhibits much higher errors for diagonal edges, and the error reverses its sign as t increases from 0.3 to 0.5. This behavior is very undesirable because complicated systematic errors of that kind are hard to correct in subsequent analysis steps. The behavior of the 9-point parabola correction is much more uniform, and its error rarely exceeds 0.02 pixels. Relatively large errors are also found for the subpixel Haralick operator. This is a consequence of the fact that aliasing is introduced into the Gaussian second derivative filters (which form the basis of Haralick's operator) when these filters are sampled at $\sigma_{\text{filter}} = 1.0$ with unit sample spacing. To avoid this, two-fold oversampling should be applied to the original image, so that the derivative filters can be digitized with a sample spacing of 1/2. Significantly improved results on oversampled images are shown in figure 7.6.

Finally, we demonstrate results for edge orientation estimation in figure 7.7. We see that orientation errors are very small (below 0.004°), and that they are comparable for pixel-accurate and subpixel algorithms. This is due to the fact that the gradient direction of an isolated step edge is constant in a certain neighborhood of the edge, so that localization errors have only minor effects on orientation estimation.

We draw the following conclusions from the experiments so far: Straight edge localization on spline interpolated boundary indicators is extremely accurate when the images are free of noise and aliasing. A bias of as little as 0.003 pixels is possible, indicating that the continuous theory is very well implemented by these algorithms. Haralick's detector (when applied at the same filter scale) and the 9-point parabola fit are still reasonably accurate with a maximum error of about $0.02 - 0.03$ pixels. The 3-point parabola fit and the pixel-accurate methods are clearly ignoring much of the available information. The 9-point fit is by far the cheapest among the better algorithms, but it can create at most one edgel per pixel. Considering the boundary sampling theorem 6.8, higher edgel densities along the true edge are desirable because the maximum distance p from true boundary points to edgels (which is always at least half as big as the edgel spacing) should not significantly exceed the localization error q. When only one edgel per pixel is detected, p can never be smaller than $\sqrt{2}/2$ along diagonal edges, and one cannot draw significant advantages from the detector's high localization accuracy of $q < 0.03$.

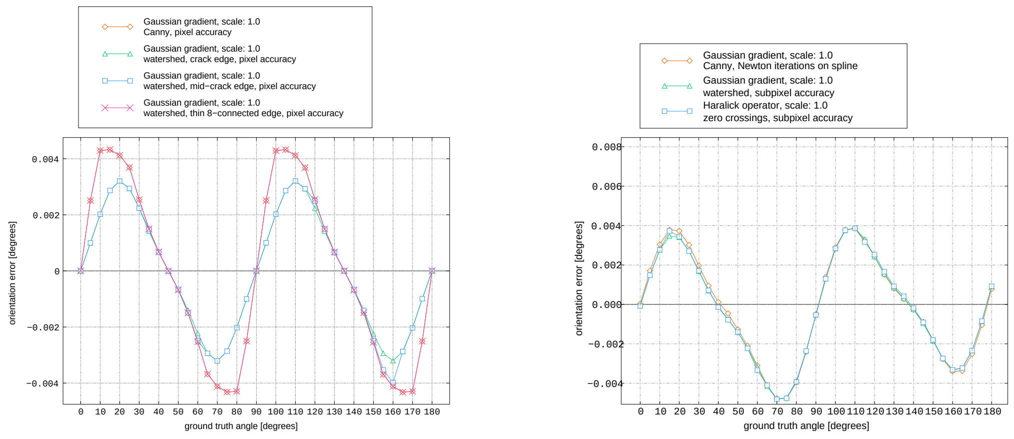


Figure 7.7: Edge orientation errors for pixel-accurate (left) and subpixel accurate (right) edge detectors at $\sigma_{\text{PSF}} = 0.9$, $\sigma_{\text{filter}} = 1.0$, $t = 0.3$ (the curves for other t are similar).

7.2.2 Noisy Images

We have seen that edge detection performance of subpixel-accurate algorithms in a noise-free situation is extremely good. We now extend our analysis to noisy images, but keep the constraint of isolated straight step edges. The same problem has been investigated by many authors, e.g. [Canny 86, Lyvers & Mitchell 88, Deriche 90, Kakarala & Hero 92, De Vriendt 95][2]. Canny introduced several metrics that quantify noise errors:

- the signal-to-noise ratio of the edge response as a measure for the probability of missed edges (false negatives),
- the expected localization error for true edges,
- the probability of false positives in the neighborhood of the true edge, and
- the expected distance between false positives in areas dominated by noise (i.e. far away from any true edges).

[Canny 86] himself and several subsequent authors, e.g. [Deriche 90, Shen & Castan 92], use these metrics mainly to identify optimal operators within certain classes of admissible edge detectors. In this context, it is sufficient to compute ratios between error metrics of different candidate detectors, whereas absolute error values are only of secondary interest. In contrast, our focus is on absolute theoretical error bounds for a given imaging situation and on the question whether real implementations of various edge detectors (gradient magnitude, oriented second derivatives) actually achieve these bounds in practice. To obtain realistic predictions, the theoretical analysis has to be conducted entirely in 2D (many other authors restrict their analysis to the 1D case) and it must include the effects of the camera PSF (signal blurring and possibly aliasing).

[2]See `http://iris.usc.edu/Vision-Notes/bibliography/edge235.html#KK1097` for a comprehensive list.

As before, we assume that the PSF is a Gaussian at scale σ_{PSF}, the signal is a straight step edge of height S, and the edge position is defined by the maxima of the Gaussian gradient at scale σ_{filter} or the zero-crossings of the second Gaussian derivative in gradient direction (both positions are identical for isolated straight step edges). Moreover, the noise shall be white Gaussian noise with standard deviation N, band-limited in the sampling pass-band.

The quotient S/N is the signal-to-noise ratio (SNR) of the digital image[3]. Since it is possible to reconstruct the blurred analog image by convolving the discrete image with the ideal interpolator (or a close approximation such as a quintic spline), we can perform error analysis in the continuous domain. When edge detection is based on Gaussian filters and their derivatives, we can define analog image functions by

$$f_{\text{filtered}}(x,y) = g_{\sigma_{\text{filter}}} \star \text{sinc} \star \left([g_{\sigma_{\text{PSF}}} \star \text{step}]_{ij} + n_{ij} \right) = s_{\text{filtered}}(x,y) + n_{\text{filtered}}(x,y) \quad (7.4)$$

where subscripts indicate sampled quantities. The smoothed edge $[g_{\sigma_{\text{PSF}}} \star \text{step}]$ is defined as in (7.2), and $g_{\sigma_{\text{filter}}}$ is a Gaussian filter at scale σ_{filter} that reduces the noise. Derivatives are defined as true infinitesimal operators acting on f_{filtered}, not by finite differences. This is easily implemented by replacing $g_{\sigma_{\text{filter}}}$ with the appropriate Gaussian derivative filters. When the PSF is effectively band-limited, sampling of the signal and sinc-reconstruction cancel each other, so that PSF and noise filter can be combined into a single convolution of the step with a Gaussian whose total scale is $\sigma = \sqrt{\sigma_{\text{PSF}}^2 + \sigma_{\text{filter}}^2}$:

$$s_{\text{filtered}}(x,y) = g_{\sigma} \star \text{step} \quad (7.5)$$

In contrast, the noise is convolved with the noise filter and the sinc (which ensures that it is band-limited), but not with the PSF. If the noise filter is itself effectively band-limited (i.e. $\sigma_{\text{filter}} \geq 0.9$), the sinc-interpolation can be dropped

$$n_{\text{filtered}}(x,y) = g_{\sigma_{\text{filter}}} \star \text{sinc} \star (n_{ij}) \approx g_{\sigma_{\text{filter}}} \star (n_{ij})$$

Without loss of generality, we shall assume in our theoretical analysis that the edge is vertical at $x = 0$. Then, all derivatives of the signal $s_{\text{filtered}}(x,y)$ along the y-axis are zero, whereas the derivatives of the noise $n_{\text{filtered}}(x,y)$ are independent of the edge direction. Experiments will be carried out at all edge orientations in order to check whether edge detectors are rotationally invariant.

7.2.2.1 Effects of Aliasing Noise

We first investigate how the term $s_{\text{filtered}}(x,y)$ deviates from equation (7.5) when the PSF is not band-limited. We had shown in section 3.2.1 that aliasing occurs when $\sigma_{\text{PSF}} < 0.9$. Since $\sigma_{\text{PSF}} > 0.4$ in all practically relevant cameras, the PSF is still strong enough to ensure that aliasing is only caused by the first spectrum replication outside the sampling

[3]Some authors prefer the term "contrast-to-noise ratio" in order to avoid confusion with SNR definitions in terms of signal power.

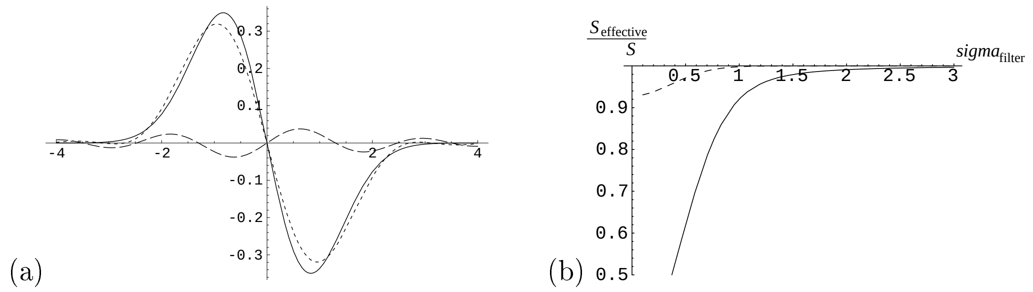


Figure 7.8: (a) The ideal response of the second Gaussian derivative of a step without sampling artifacts (solid), the actual response (dotted) and the aliasing component (dashed) for $\sigma_{\text{PSF}} = 0.45$ and $\sigma_{\text{filter}} = 0.7$. (b) Effective reduction in the height of the step due to aliasing at $\sigma_{\text{PSF}} = 0.45$ (solid) and $\sigma_{\text{PSF}} = 0.9$ (dotted) as a function of σ_{filter}.

pass-band. Then, the Fourier transform of the step after reconstruction can be written as

$$\mathcal{F}\left[\text{sinc} \star [g_{\sigma_{\text{PSF}}} \star \text{step}]_{ij}\right](\nu_1, \nu_2)$$
$$= \begin{cases} \frac{S}{2\pi \mathrm{i}\, \nu_1} e^{-2\pi^2(\nu_1^2+\nu_2^2)\sigma_{\text{PSF}}^2} + \\ \frac{S}{2\pi \mathrm{i}\,(\nu_1-1)} e^{-2\pi^2((\nu_1-1)^2+\nu_2^2)\sigma_{\text{PSF}}^2} + \frac{S}{2\pi \mathrm{i}\,(\nu_1+1)} e^{-2\pi^2((\nu_1+1)^2+\nu_2^2)\sigma_{\text{PSF}}^2} & \text{if } |\nu_1|, |\nu_2| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

where the first term corresponds to the contribution of the sampling pass-band, and the other two represent aliasing. This spectrum is multiplied with the transfer function of the Gaussian noise filter at scale σ_{filter}. The corresponding spatial domain functions cannot be expressed analytically. Numeric results are shown in figure 7.8. Figure 7.8a depicts the second Gaussian derivative. It can be seen that the signal and aliasing responses have opposite signs. Consequently, aliasing reduces the slope of the second derivative, which is equivalent to an effective reduction of the step height by as much as 22% at $\sigma_{\text{PSF}} = 0.45$ and $\sigma_{\text{filter}} = 0.7$. Figure 7.8b shows the amount of reduction for other parameter choices. We will see later that all error metrics are inversely proportional to the step height, so an effective step height reduction results in a proportional error increase. Thus, edge detection becomes significantly more difficult when aliasing is present.

On the other hand, aliasing noise of Gaussian PSFs is concentrated at high frequencies, so a Gaussian noise filter is able to remove some of the aliasing noise, cf. section 3.2.1. We see in 7.8b that for a PSF with $\sigma_{\text{PSF}} = 0.45$ (which is not band-limited), noise filters with $\sigma_{\text{filter}} = 1$ and $\sigma_{\text{filter}} = 2$ lead to effective step height reductions of only 8% and 1% respectively. This is still acceptable, especially when the image has low signal-to-noise ratio, so that large noise filters are required anyway. In contrast, the combination of a band-limited PSF with a small filter (for example, $\sigma_{\text{PSF}} = 0.9$, $\sigma_{\text{filter}} = 0.85$) should be preferred when the SNR high.

Experimental results on images containing aliasing due to a too narrow PSF confirm these findings, as can be seen in figure 7.9. In comparison to figure 7.6 (which reports

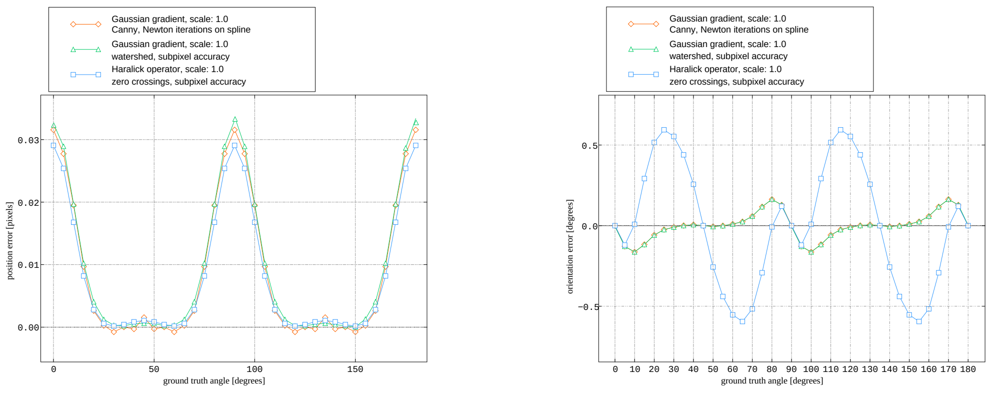


Figure 7.9: Errors of subpixel edge detectors at $\sigma_{\text{PSF}} = 0.45$ and $\sigma_{\text{filter}} = 1$ on 2-fold interpolated input and $t = 0.3$ (left: localization error, right: orientation error). These diagrams should be compared with figures 7.6 and 7.7 right.

the same experiment for $\sigma_{\text{PSF}} = 0.9$), the localization errors have increased by more than an order of magnitude. The increase in orientation errors is not much less (compare with figure 7.7 right). However, these errors exhibit some interesting behavior: they are much smaller for diagonal edges than for horizontal or vertical edges. This is easily understood when one recalls that the sampling passband is square-shaped. Consequently, the Nyquist limit for diagonal frequencies is $\sqrt{2}$-times as high as the one along the grid's principal directions, so there is only little aliasing for diagonal edges. On different reasons, the orientation error is also zero at edge directions which are exact multiples of 90°: Here, the gradient component along one direction is exactly zero, and no amount of aliasing in the other gradient component can cause an error in the orientation estimate. These experiments also show that aliasing can be neglected whenever the errors due to statistical noise are significantly larger than the maximal aliasing errors, i.e. when statistical position errors exceed 0.03 pixels, and orientation errors exceed 0.5°. Statistical errors are analyzed below.

7.2.2.2 Probability Distributions of Noisy Gradient Magnitudes and Optimal Thresholds

Let us now look at how the gradient magnitude changes due to statistical noise. The variance of Gaussian derivatives of the noise term $n_{\text{filtered}}(x, y)$ can be computed by autocorrelation. Due to Parseval's theorem, the autocorrelation of the $(k+l)$'s derivative is given in the Fourier domain as

$$\text{Var}\left[\frac{\partial^{k+l} n(x,y)}{\partial x^k \partial y^l}\right] = N^2 \int_{\frac{1}{2}}^{\frac{1}{2}} \int_{\frac{1}{2}}^{\frac{1}{2}} \left((2\pi\nu_1)^k (2\pi\nu_2)^l e^{-2\pi^2(\nu_1^2+\nu_2^2)\sigma_{\text{filter}}^2}\right)^2 d\nu_1 d\nu_2$$

where N^2 is the noise variance. Notice that the noise is only convolved with derivative filters at scale σ_{filter}, not with the PSF. The integration limits reflect the fact that the noise is band-limited within the sampling pass-band. If $\sigma_{\text{filter}} \gtrsim 0.9$ (which is usually the case in practice, especially when the image is oversampled as described in section 7.1), we can safely extend the integration over the entire Fourier domain to obtain simpler expressions. The variance of the first few derivatives of the noise are therefore

$$\begin{aligned} N_x^2 = N_y^2 &= \frac{N^2}{8\pi\sigma_{\text{filter}}^4} \\ N_{xx}^2 = N_{yy}^2 &= \frac{3N^2}{16\pi\sigma_{\text{filter}}^6} \\ N_{xy}^2 &= \frac{N^2}{16\pi\sigma_{\text{filter}}^6} \\ N_{xxx}^2 = N_{yyy}^2 &= \frac{15N^2}{32\pi\sigma_{\text{filter}}^8} \end{aligned}$$

In order to compute the probability of false positives/negatives due to noise, we must compare the gradient magnitude of a noisy step edge with the gradient magnitude of pure noise. The gradient magnitude of a *noise-free* vertical step edge located at $x = 0$ is

$$s = \frac{\partial}{\partial x}\left(g_\sigma \star \text{step}_S\right)\bigg|_{x=0} = \frac{S}{\sqrt{2\pi}\sigma}$$

where S is the step height and $\sigma = \sqrt{\sigma_{\text{PSF}}^2 + \sigma_{\text{filter}}^2}$ is the total scale. The derivative of the noisy step edge model (7.4) in x-direction

$$f_x = s_x + n_x$$

is the sum of a constant and a Gaussian random variable with zero mean and variance N_x^2. Hence, $f_x|_{x=0}$ is a Gaussian random variable with mean $s = \frac{S}{\sqrt{2\pi}\sigma}$ and variance N_x^2. The derivative f_y does not contain any signal component and is therefore just a Gaussian random variable with zero mean and variance $N_y^2 = N_x^2$. The squared gradient magnitude

$$g^2 = f_x^2 + f_y^2 = (s_x + n_x)^2 + n_y^2$$

is the sum of squares of two Gaussian random variables with equal variance but different means. Its probability is a *non-central χ^2-distribution* with two degrees of freedom. According to [Proakis 89], the general formula for a non-central χ^2-distribution with k degrees of freedom is

$$p(y) = \frac{1}{2\sigma_0^2}\left(\frac{y}{\mu^2}\right)^{(k-2)/4} e^{-\frac{y+\mu^2}{2\sigma_0^2}} I_{k/2-1}\left(\sqrt{y}\frac{\mu}{\sigma_0^2}\right)$$

where $I_n(.)$ is the nth-order modified Bessel function of the first kind, σ_0^2 is the variance of the original random variables, and μ^2 is the *noncentrality parameter* of the distribution

$$\mu^2 = \sum_i \mu_i^2$$

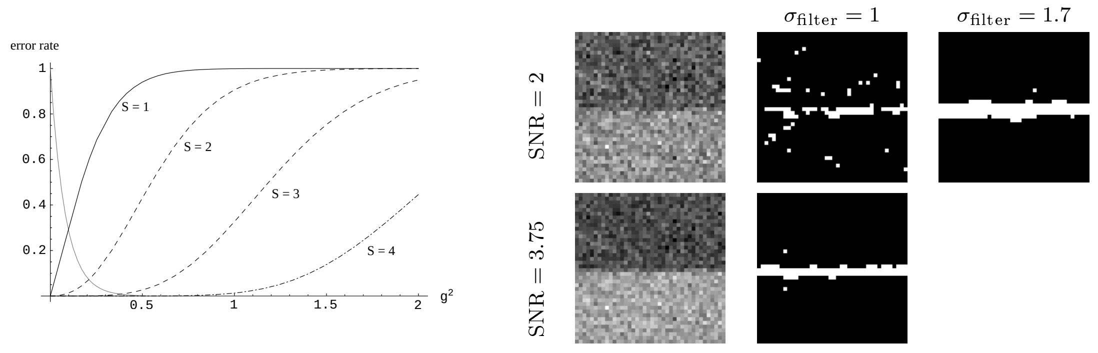


Figure 7.10: Left: False alarm rate (gray) and miss rates for step heights $S \in \{1, 2, 3, 4\}$ (black) as a function of the gradient squared magnitude with $N^2 = 1$, $\sigma_{\text{PSF}} = 0.5$ and $\sigma_{\text{filter}} = 1$. Right, first row: At SNR $= 2$, there is no good threshold on the gradient magnitude for $\sigma_{\text{filter}} = 1$, whereas a good threshold exists at $\sigma_{\text{filter}} = 1.7$; second row: SNR $= 3.75$ is required for a good threshold to exist for $\sigma_{\text{filter}} = 1$. At the given image size of 41×41, an error rate of 0.1% allows for one or two false pixels in the image domain. Thresholds were chosen according to table 7.1.

where μ_i denotes the mean of the i^{th} original variable. The non-central χ^2-distribution has mean $k\sigma_0^2 + \mu^2$ and variance $2\sigma_0^2(k\sigma_0^2 + 2\mu^2)$. In our case, $k = 2$, $\mu_1 = s$, $\mu_2 = 0$ and $\sigma_0^2 = N_x^2 = N_y^2$. The above expression then simplifies to

$$p(g^2) = \frac{1}{2N_x^2}\, e^{-\frac{g^2+s^2}{2N_x^2}}\, I_0\left(\frac{|g|\, s}{N_x^2}\right) \tag{7.6}$$

When the step height S is zero, the formula reduces to the standard χ^2-distribution with two degrees of freedom and characterizes the probability distribution of the gradient squared magnitude for pure noise:

$$p_{\text{noise}}(g^2) = \frac{1}{2N_x^2}\, e^{-\frac{g^2}{2N_x^2}}$$

The *false alarm rate* for some threshold g_0^2 on the squared magnitude is just the complement of the cumulative probability of the noise

$$\text{false alarm rate}(g_0^2) = 1 - \int_0^{g_0^2} p_{\text{noise}}(t)\, dt = e^{-\frac{g_0^2}{2N_x^2}}$$

The *miss rate* for true edges is the cumulative probability of $p(g^2)$ up to the threshold, but this integral has no closed-form solution. The total error rate is minimized for the *balanced* threshold where false alarm rate and miss rate are equal, i.e. where the two curves cross. Edge detection will work reliably when the error rates are low at this threshold. Figure 7.10 left shows these rates for various signal-to-noise ratios and standard scales $\sigma_{\text{PSF}} = 0.5$ and $\sigma_{\text{filter}} = 1$. It can be seen that the reliability increases rapidly with increasing SNR. At the optimal threshold, an error rate of 1% is achieved forSNR ≥ 2.9 (threshold $g_0^2 = 0.36$), and an error rate of 0.1% is achieved for SNR ≥ 3.75 (threshold

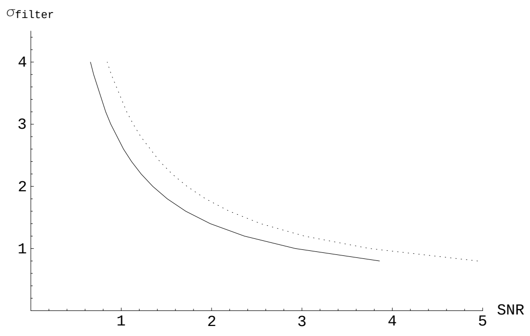


Figure 7.11: Required filter scale σ_{filter} for reliable edge detection as a function of the signal-to-noise ratio S/N for error rates 1% (solid) and 0.1% (dotted). The curves refer to $\sigma_{\text{PSF}} = 0.5$. Corresponding curves for $\sigma_{\text{PSF}} = 0.9$ run between these curves.

	error rate 1%			error rate 0.1%		
SNR	minimal σ_{filter}	g_0^2/N^2	$\lvert g_0\rvert/N$	minimal σ_{filter}	g_0^2/N^2	$\lvert g_0\rvert/N$
≥ 4	0.8	0.9	0.95	0.95	0.68	0.82
3	1.0	0.37	0.61	1.2	0.26	0.51
2	1.4	0.095	0.31	1.7	0.06	0.24
1	2.6	0.008	0.09	3.2	0.005	0.07

Table 7.1: Values for minimal required filter scale and corresponding normalized thresholds on the gradient squared magnitude for $\sigma_{\text{PSF}} = 0.5$. The noise variance N^2 must be made constant throughout the image, for example by means of a noise normalization transform (cf. section 3.4).

$g_0^2 = 0.55$). These predictions are confirmed by the images on the right of figure 7.10. A more detailed analysis of the relation between SNR and filter scale is given in figure 7.11 and table 7.1 which show the scales and thresholds required for achieving given error rates for given SNR.

A simple approximate formula for the required SNR at a given filter size can be derived by the following consideration due to Canny. A false positive near a true edge is unlikely when the slope of the signal's second derivative at the edge exceeds the maximum expected slope of the second derivative of the noise. The maximum expected slope of the second derivative of the noise is $3N_{xxx}$ (with 99.7% probability), whereas the slope of the second derivative of the signal equals the step height times the second derivative of a Gaussian. Inserting these values, we get

$$\left.\frac{|s_{xxx}|}{3N_{xxx}}\right|_{x=0} = \frac{S}{N}\,\frac{4}{3\sqrt{15}}\,\frac{\sigma_{\text{filter}}^4}{\left(\sigma_{\text{PSF}}^2 + \sigma_{\text{filter}}^2\right)^{3/2}} > 1$$

Therefore, the required SNR is

$$\text{SNR} > \frac{3\sqrt{15}}{4}\,\frac{\left(\sigma_{\text{PSF}}^2 + \sigma_{\text{filter}}^2\right)^{3/2}}{\sigma_{\text{filter}}^4} \tag{7.7}$$

For filter sizes $\sigma_{\text{filter}} = 1, 2, 3$ and $\sigma_{\text{PSF}} = 0.5$, this formula gives minimum required SNRs of 4.0, 1.6, and 1.0 respectively, in reasonable agreement with table 7.1.

We see that the gradient magnitude edge detector can tolerate a whole lot of noise when noise filters of moderate size are used. However, this is only true for isolated step edges.

When two or more edges run very closely to each other, even a moderately strong post-filter with $\sigma_{\text{filter}} = 3$ may cause these edges to blend into each other (this phenomenon will be analyzed in more detail in section 7.3.2). To keep them separate, smaller filters have to be used, which significantly reduces the tolerance of edge detection against noise. A similar effect can be observed at edges between shaded regions, see section 7.3.1.

7.2.2.3 Error Propagation for Edge Position and Orientation

To estimate the edge localization error due to noise, we use the same technique as [Canny 86], but apply it in the 2-dimensional image domain. That is, our error analysis is based on 2D derivatives and integrals throughout, whereas Canny restricted itself to the 1-dimensional case in most of his paper. Our analysis is similar to [De Vriendt 95], who first derived many of the results reported in this section. We assume without loss of generality that the true edge runs vertically at $x = 0$. Statistical expectations are then independent of y, and we can assume $y = 0$ as well. Furthermore, let us pretend for the moment that we already know the edge direction. Then we can detect the noisy edge position by just looking for zero crossings of the second derivative along the x-direction. The second x-derivative of the noise-free edge vanishes at $x = 0$, but the second derivative of the noisy edge vanishes at a displaced position Δx_Z:

$$f_{xx}(\Delta x_Z) = s_{xx}(\Delta x_Z) + n_{xx}(\Delta x_Z) = 0$$

We expand the signal term into its first order Taylor series around $x = 0$. Since s_{xx} is zero at this point, we get

$$f_{xx}(\Delta x_Z) \approx s_{xxx}(x)\big|_{x=0}\, \Delta x_Z + n_{xx}(\Delta x_Z) = 0 \tag{7.8}$$

The third derivative of the step is $s_{xxx}(x)\big|_{x=0} = -\frac{S}{\sqrt{2\pi}\sigma^3}$. After rearranging, we obtain the standard deviation of the edge position as

$$\operatorname{StdDev}[\Delta x_Z] = \left.\frac{N_{xx}}{|s_{xxx}|}\right|_{x=0} = \frac{N}{S}\frac{\sqrt{6}}{4}\left(1 + \frac{\sigma^2_{\text{PSF}}}{\sigma^2_{\text{filter}}}\right)^{3/2} \approx 0.61\frac{N}{S}\ (\text{if } \sigma_{\text{filter}} \gg \sigma_{\text{PSF}}) \tag{7.9}$$

However, in reality the true edge direction is unknown, so we cannot just search for zero crossings of the second x-derivative. Instead, we take the second derivative along the local gradient direction. This expression contains additional terms which potentially increase the variance of the result. When we still assume (w.l.o.g.) the edge to run vertically through the coordinate origin, the second directional derivative along the gradient direction will again vanish at some displaced location Δx_Z such that

$$\left.\frac{f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}}{f_x^2 + f_y^2}\right|_{x=\Delta x_Z} = 0$$

For $f_x \neq 0$ (which always holds near the true edge), this can be rewritten as

$$\left.\frac{1}{1+\left(\frac{f_y}{f_x}\right)^2} f_{xx} + 2\frac{\frac{f_y}{f_x}}{1+\left(\frac{f_y}{f_x}\right)^2} f_{xy} + \frac{\left(\frac{f_y}{f_x}\right)^2}{1+\left(\frac{f_y}{f_x}\right)^2} f_{yy}\right|_{x=\Delta x_Z} = 0$$

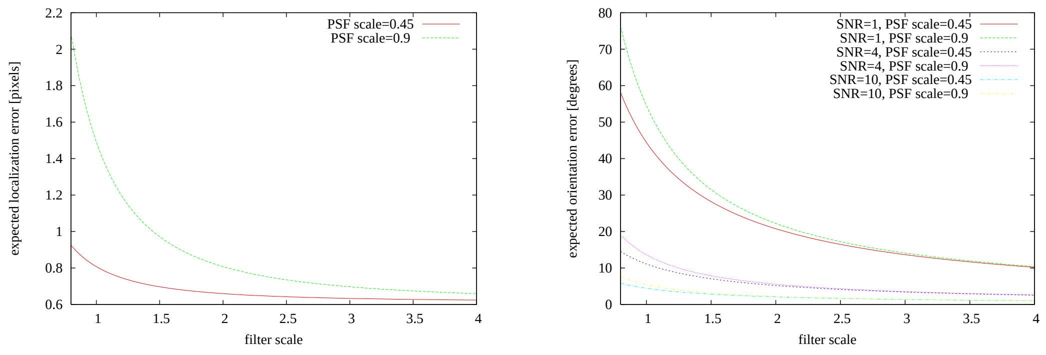


Figure 7.12: Left: The expected edge displacement due to noise as a function of the filter scale σ_{filter} for $S/N = 1$ and $\sigma_{\text{PSF}} \in \{0.45, 0.9\}$. Corresponding values for other signal-to-noise ratios are obtained by simply dividing the error by the SNR. In case of a pixel-accurate edge detector, the expected round-off error must be added. Right: Expected edge orientation error as a function of σ_{filter} for $S/N \in \{1, 4, 10\}$ and and $\sigma_{\text{PSF}} \in \{0.45, 0.9\}$. Errors due to aliasing (cf. figure 7.9) can be neglected at these noise levels.

Expanding the image into its signal and noise parts $f = s + n$ and recalling that all y-derivatives of the signal part are zero, we get

$$\left.\frac{1}{1+\left(\frac{n_y}{s_x+n_x}\right)^2}(s_{xx}+n_{xx}) + 2\frac{\frac{n_y}{s_x+n_x}}{1+\left(\frac{n_y}{s_x+n_x}\right)^2}n_{xy} + \frac{\left(\frac{n_y}{s_x+n_x}\right)^2}{1+\left(\frac{n_y}{s_x+n_x}\right)^2}n_{yy}\right|_{x=\Delta x_Z} = 0 \tag{7.10}$$

Near the true edge, we have $\operatorname{Var}[s_x + n_x] \approx s_x^2 = \frac{S^2}{2\pi(\sigma_{\text{PSF}}^2+\sigma_{\text{filter}}^2)}$ and

$$\operatorname{Var}\left[\frac{n_y}{s_x+n_x}\right] \approx \left(\frac{N}{S}\right)^2 \frac{\sigma_{\text{PSF}}^2+\sigma_{\text{filter}}^2}{2\sigma_{\text{filter}}^2}$$

For reasonable signal-to-noise ratios and filter scales, this is always $\ll 1$. Thus, the second and third terms in (7.10) are close to zero, whereas the first term is almost equal to $f_{xx} = s_{xx} + n_{xx}$. It follows that the localization error for unknown edge direction (7.10) is essentially the same as that for known edge direction (7.9). The error in the estimated edge orientation has no significant influence on the localization error. Figure 7.12 left depicts formula (7.9) as a function of the filter scale. When $S/N > 14$ (which is realistic in high quality images), we should be able to achieve localization errors below 1/20 of a pixel. This is a strong justification of sub-pixel accurate edge detection. We will see below that such high accuracy can actually be achieved in practice.

In this context, it is interesting to note that the difference between the known-edge-direction / unknown-edge-direction cases is not always small. For example, when edges are defined by zero-crossings of the Laplacian of Gaussian, the Taylor series reads

$$f_{xx} + f_{yy} = 0 = s_{xxx}(x)|_{x=0}\,\Delta x_{\text{Laplace}} + n_{xx}(\Delta x_{\text{Laplace}}) + n_{yy}(\Delta x_{\text{Laplace}})$$

The quantities n_{xx} and n_{yy} are correlated with a correlation coefficient of $\frac{1}{3}$, see [De Vriendt 95]. Therefore, we get

$$\mathrm{Var}\left[n_{xx}+n_{yy}\right]=\mathrm{Var}\left[n_{xx}\right]+\mathrm{Var}\left[n_{yy}\right]+\frac{2}{3}\sqrt{\mathrm{Var}\left[n_{xx}\right]\mathrm{Var}\left[n_{yy}\right]}=\frac{8}{3}\mathrm{Var}\left[n_{xx}\right]$$

In other words, the localization error of the Laplacian of Gaussian exceeds the one of the oriented second derivative by a factor of $\sqrt{8/3}\approx 1.63$:

$$\mathrm{StdDev}\left[\Delta x_{\mathrm{Laplace}}\right]=\sqrt{\frac{8}{3}}\,\mathrm{StdDev}\left[\Delta x_{Z}\right] \tag{7.11}$$

These results will be confirmed by our experiments, see sections 7.2.2.5 and 7.4.

Just like edge position estimates, edge orientation estimates suffer from noise. The edge orientation can be determined by the angle between the gradient vector and the x-axis

$$\phi=\arctan\left(\frac{f_y}{f_x}\right) \tag{7.12}$$

The variance of this expression can be derived from the variance of the gradient vector components by standard error propagation

$$\mathrm{Var}[\phi]=\mathbf{J}_\phi^T\,\mathbf{\Sigma}_{f_x,f_y}\,\mathbf{J}_\phi$$

where $\mathbf{J}_\phi=(\partial\phi/\partial f_x,\partial\phi/\partial f_y)^T$ is the Jacobian of the orientation, and $\mathbf{\Sigma}_{f_x,f_y}$ is the covariance matrix of the gradient components. Since the gradient components are orthogonal, the covariance matrix is a diagonal matrix, and we get

$$\mathrm{Var}[\phi]=\frac{\mathrm{Var}[f_y]\,f_x^2+\mathrm{Var}[f_x]\,f_y^2}{\left(f_x^2+f_y^2\right)^2}$$

As the gradient operator is isotropic, we can compute the expected angular error for an arbitrary edge orientation, e.g. vertical. Then we have $f_y^2=0$, $f_x^2=\frac{S^2}{2\pi\sigma^2}$ (with $\sigma^2=\sigma_{\mathrm{PSF}}^2+\sigma_{\mathrm{filter}}^2$ as before), and $\mathrm{Var}[f_y]=N_y^2=\frac{N^2}{8\pi\sigma_{\mathrm{filter}}^4}$. Inserting these values, we obtain

$$\mathrm{Var}[\phi]=\frac{N^2}{4S^2}\,\frac{\sigma_{\mathrm{PSF}}^2+\sigma_{\mathrm{filter}}^2}{\sigma_{\mathrm{filter}}^4}$$

and thus

$$\mathrm{StdDev}[\phi]=\frac{N}{S}\,\frac{\sqrt{\sigma_{\mathrm{PSF}}^2+\sigma_{\mathrm{filter}}^2}}{2\sigma_{\mathrm{filter}}^2} \tag{7.13}$$

This formula is illustrated in figure 7.12 right. Its predictions are also well confirmed by the experiments in sections 7.2.2.5 and 7.4. At SNR = 10 and standard scales, the expected orientation error is around 4.5°, which is still acceptable. At SNR = 1, orientation estimation becomes essentially impossible with small filters.

Curvature (and therefore the radii of curved edges) can be determined by taking the derivative of the tangent angle with respect to arc length. If the edge direction is defined by the vector perpendicular to the gradient direction, the curvature is simply the *isophote curvature*

$$\kappa = \frac{f_y^2 f_{xx} - 2 f_x f_y f_{xy} + f_x^2 f_{yy}}{\left(f_x^2 + f_y^2\right)^{3/2}} \tag{7.14}$$

The standard error propagation rule results in a quite complicated expression which depends on the detailed shape of the contour. If we assume that the curvature is small, the result simplifies to

$$\mathrm{StdDev}[\kappa] = \sqrt{\frac{3}{8}}\frac{N}{S}\,\frac{\sqrt{\sigma_{\mathrm{PSF}}^2 + \sigma_{\mathrm{filter}}^2}}{\sigma_{\mathrm{filter}}^3} \tag{7.15}$$

This expression is exact for straight lines, and a good approximation for slightly curved ones. When we set $\sigma_{\mathrm{PSF}} = 0.5$, $\sigma_{\mathrm{filter}} = 1$ and $\mathrm{SNR} = 10$, the expected curvature error is $\mathrm{StdDev}[\kappa] = 0.07$. Thus, the expected error exceeds the true curvature when the curve radius is above 15 pixels! This shows that precise curvature estimation is an extremely difficult problem in noisy images – when the curvature is small, not even its sign can be determined with any certainty, cf. [Utcke 03].

7.2.2.4 Error Correlation along the Edge

Another interesting question is how fast the different errors change along the edge. Consider a particular point on the edge where the position error happens to be zero and define a local coordinate system whose horizontal axis coincides with the tangent at that point, and which is oriented so that the sign of the displacement is negative to the left of the point, and positive to its right. As we traverse the edge to the right, the error will first raise, but eventually will become zero again, this time changing sign from positive to negative, and so on. We denote the expected distance between two consecutive error zero-crossings with the *same* polarity (both from negative to positive error or vice versa) as the average *wave length* of the error along the edge.

This wave length can be computed as follows: Without loss of generality, we assume that the edge runs vertically. Then the position error is zero whenever n_{xx} (the second horizontal derivative of the noise) is zero, cf. (7.8). Similarly, the orientation error is zero whenever $f_y = n_y$ (the first vertical derivative of the noise) is zero, cf. (7.12). The curvature error is dominated by the term $f_x^2 f_{yy}$, which is zero whenever n_{yy} is zero. Along the y-direction, i.e. along the true vertical edge, all these errors are 1-dimensional Gaussian random processes. According to [Rice 45] (see also [Canny 86]), the expected distance between two consecutive zero-crossings (with opposite polarity) of a noise process p with derivative p' is

$$d_z = \pi\sqrt{\frac{R_p(0)}{R_{p'}(0)}}$$

where $R_p(0)$ and $R_{p'}(0)$ denote the autocorrelations of p and p' at zero. Since our noise processes have zero mean, the autocorrelation equals the variance, and the expected wave length λ is twice the distance of consecutive zero crossings. Applying this formula to the position, orientation, and curvature error cases, we get the following wave-length estimates

$$\begin{aligned}
\text{position:} \quad \lambda_{\text{pos-error}} &= 2\pi\sqrt{\frac{N_{xx}^2}{N_{xxy}^2}} = \sqrt{8}\pi\sigma_{\text{filter}} \\
\text{orientation:} \quad \lambda_{\text{ori-error}} &= 2\pi\sqrt{\frac{N_{y}^2}{N_{yy}^2}} = \sqrt{\frac{8}{3}}\pi\sigma_{\text{filter}} \qquad (7.16) \\
\text{curvature:} \quad \lambda_{\text{curv-error}} &= 2\pi\sqrt{\frac{N_{yy}^2}{N_{yyy}^2}} = \sqrt{\frac{8}{5}}\pi\sigma_{\text{filter}}
\end{aligned}$$

For example, at $\sigma_{\text{filter}} = 1$, the position error can be expected to keeps its sign over an average boundary length of 4 pixels.

An even more detailed characterization of the noise along the boundary is possible by means of the power spectrum of the localization error. The power spectrum not only tells us how errors along the edge are correlated. It also allows us to predict how fast these errors decrease when the initial boundary polygon is subsequently smoothed. We will make use of this possibility in chapter 8. Due to the correlation of neighboring errors, the error power spectrum along the edge does not have the characteristics of Gaussian white noise (i.e. constant noise power up to some limit frequency), contrary to a popular model assumption.

To derive the power spectrum, recall that the noise was Gaussian distributed *before edge detection*. Thus, we have to investigate how noise characteristics change due to noise filtering. We have shown above that the localization error depends on the values of the second derivative perpendicular to the edge, see (7.9), and that slight errors in the edge orientation estimate do not have significant influence on the localization error. An oriented second derivative filter can be separated into its derivative component perpendicular to the edge, and its smoothing component along the edge. After filtering *perpendicular* to the edge, the noise spectrum *along* the edge is still Gaussian distributed, because the filter is linear. Thus, correlation between neighboring points along the edge is solely caused by the filter component acting along the edge. It follows that the power spectrum of the localization noise must be proportional to the squared transfer function of the noise filter along the edge:

$$\|\mathcal{F}[\Delta x_Z]\|^2 \sim \left(e^{-2\pi^2\nu^2\sigma_{\text{filter}}^2}\right)^2$$

(where $\|\mathcal{F}[\Delta x_Z]\|^2$ denotes the power spectrum of the localization error Δx_Z, and ν is the spatial frequency along the edge). On the other hand, it follows from Parseval's theorem that the integral over the power spectrum must be equal to the expectation of the squared localization error, which is simply the square of the standard deviation, equation (7.9). This condition defines the correct normalization of the power spectrum:

$$\begin{aligned}
\|\mathcal{F}[\Delta x_Z]\|^2 &= \frac{3\sqrt{\pi}}{4}\left(\frac{N}{S}\right)^2 \sigma_{\text{filter}}\left(1+\frac{\sigma_{\text{PSF}}^2}{\sigma_{\text{filter}}^2}\right)^3 e^{-4\pi^2\nu^2\sigma_{\text{filter}}^2} \\
&= 2\sqrt{\pi}\sigma_{\text{filter}}\epsilon^2\, e^{-4\pi^2\nu^2\sigma_{\text{filter}}^2} \qquad (7.17)
\end{aligned}$$

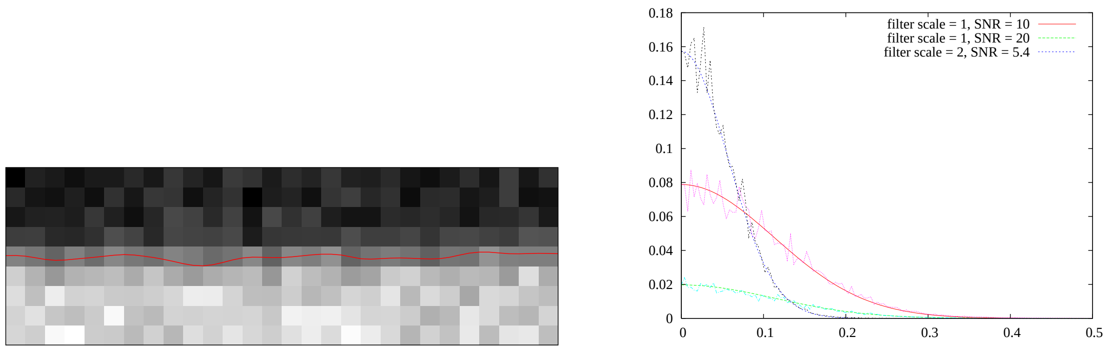


Figure 7.13: Localization noise power spectrum along a straight edge. Left: Subregion of a noisy image with detected edge (SNR = 10, $\sigma_{\text{filter}} = 1$). Right: Comparison of power spectrum measurements ($\sigma_{\text{PSF}} = 0.9$, edge length 256 pixels, edge detection by Canny's algorithm with spline-based subpixel correction according to equation (5.8), averaging over 80 noise realizations) with theoretical predictions for various values of SNR and σ_{filter}.

where $\epsilon = \text{StdDev}\left[\Delta x_Z\right]$ is the standard deviation of the localization error according to (7.9). This prediction is very well confirmed by experiment, as can be seen in figure 7.13.

7.2.2.5 Experimental Validation in Artificial Images

In order to validate our theoretical error analysis, we repeat the experiments from section 7.2.1 for noisy images. Figure 7.14 confirms for the subpixel-accurate versions of the watershed algorithm and Laplacian zero-crossings that the actual localization error is indeed independent of orientation and true edge position. In addition, it is verified that the error of the Laplacian zero-crossing operator is indeed $\sqrt{8/3}$-times as big as the one of the gradient-based operator. Similar diagrams are obtained for other algorithms.

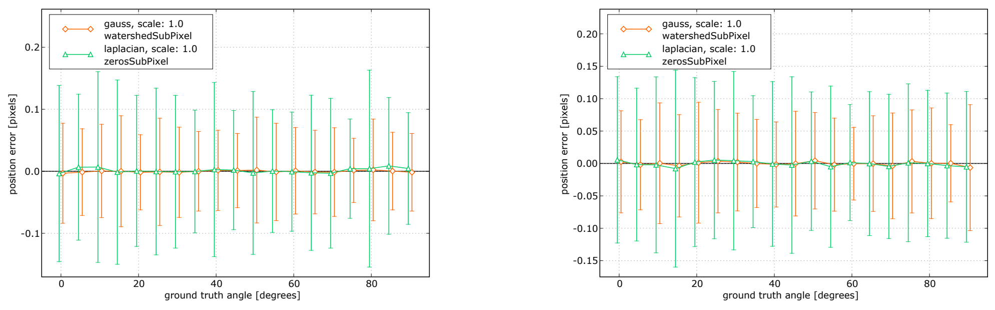


Figure 7.14: The mean localization error and its standard deviation as a function of the edge orientation for ground-truth subpixel shifts of $t = 0.1$ (left) and $t = 0.5$ (right), using the subpixel watershed algorithm and subpixel zero-crossings of the Laplacian operator at scales $\sigma_{\text{PSF}} = 0.9$, $\sigma_{\text{filter}} = 1$ and SNR = 20. The standard deviation predicted by theory is 0.075 pixels (watershed edges, equation (7.9)) and 0.12 pixels (Laplacian edges, equation (7.11)). Mean and standard deviation are computed from 40 different noise realizations for each angle.

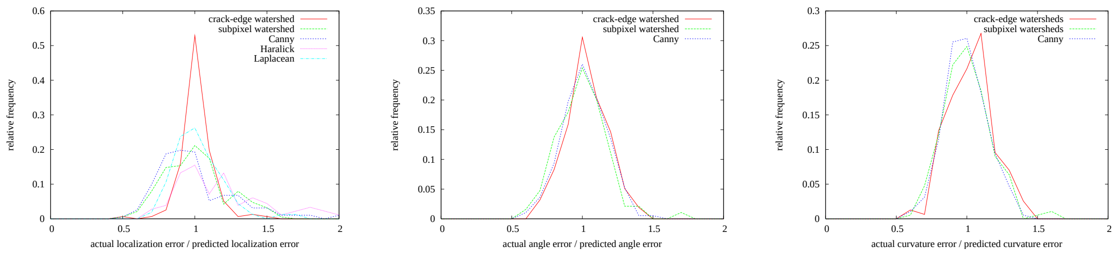


Figure 7.15: Comparison of measured and predicted errors. The curves represent histograms of the ratio between these errors for various algorithms. Histograms were computed over 200 artificial edge images generated according to (7.4) with additive Gaussian noise. Image parameters were randomly selected: edge orientation between 0 and 2π, subpixel shift t between -0.5 and 0.5, signal-to-noise ratio between 4 and 64, and the PSF scale in $\{0.5, 0.9\}$. Left: localization errors according to (7.3), (7.9), and (7.11). Center: tangent angle errors according to (7.13). Right: curvature errors according to (7.15).

These results are represented in compact form in figure 7.15. It shows histograms of the *ratio* between measured and predicted errors for various algorithms and features (left: position, center: orientation, right: curvature). In all cases, the distribution of this ratio has a marked peak at approximately 1, i.e. the theoretical predictions are confirmed very well. The good agreement is remarkable, given that absolute errors varied significantly (e.g. between 0.01 pixels and 0.4 pixels in case of localization) due to the wide range of image parameters tested. Interestingly, the systematic error of the 9-point fit in Canny's algorithm is negligible as soon as the localization error caused by noise exceeds about 0.02 pixels. Indeed, this algorithm is slightly more accurate than expected because the 9-point fit has the effect of a little additional regularization. Corresponding results for the tangent angle and curvature errors are shown in figure 7.15 center and right, and the predictions are again confirmed very well.

An interesting observation concerns a variant of gradient-based edge detection. Some applications require boundary indicators that signal edges by local minima rather than maxima. To adapt the gradient to these schemes, it is often replaced with the inverse gradient

$$f_{\text{inverse}} = \frac{1}{\epsilon + |\nabla g|}$$

for some small $\epsilon > 0$ that ensures f_{inverse} remains bounded. To test how this boundary indicator compares with the standard gradient, we repeat the localization experiment for straight edges. We apply the subpixel watershed algorithm with exactly the same parameters, except that we have to trace flowlines from saddle points *downwards* to minima now (for simplicity, we implement this by the standard watershed algorithm applied to $-f_{\text{inverse}}$). Figure 7.16 shows that the error of the inverse gradient is much higher than that of the standard gradient. Therefore, we cannot recommend the inverse gradient for edge detection or energy minimization. If a boundary indicator is needed which marks edges by minima, the negated gradient $-|\nabla g|$ should be preferred.

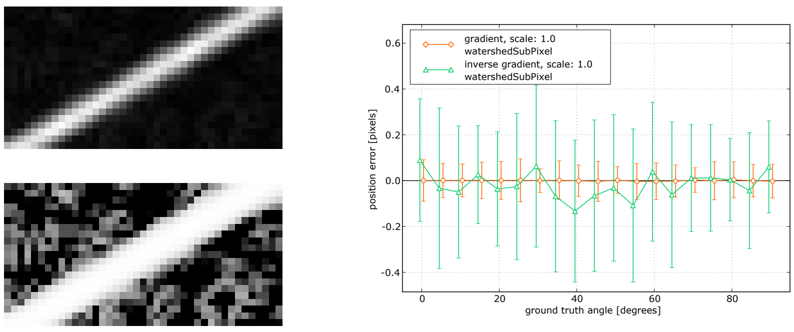


Figure 7.16: Left: Standard gradient $|\nabla g|$ and inverse gradient $\frac{-1}{10^{-6}+|\nabla g|}$ of a test image with SNR $= 10$. Inversion strongly amplifies the noise. Scales are equal to figure 7.14. Right: Comparison of the subpixel watershed algorithm on these boundary indicators. The standard deviation on the inverse gradient is 4 times as high, and its mean is much less accurate, indicating a distribution with long tails (outliers). That is, edge detection on the inverse gradient is much less stable.

7.3 Deviations from the Model of Isolated Straight Step Edges

Gradient-based boundary detectors can only be expected to give unbiased edge position estimates for isolated straight step edges. Since edges are not strictly conforming to this model in reality, we investigate by how much edge detection errors increase due to various deviations from the ideal model. In the sections to follow we consider shaded regions (deviation from the step edge requirement), parallel edges (deviation from the requirement of isolated edges), and curved edges. The behavior of gradient-based boundary detectors near corners and junctions is treated in section 7.5.

7.3.1 Shaded Regions

Equation (2.1) for ideal geometric images simplifies to the step edge model when the functions f_i (describing the interior of region r_i) are constant functions for all i. However, this is only an approximation of reality. In real images, shading and other effects cause the intensity in each region to change gradually. We model this by approximating f_i and f_j with their first order Taylor series around a boundary point $\vec{x}_0$. Higher order intensity variations are assumed to occur only at scales significantly beyond the effective diameters of the PSF and noise filter, and can thus be neglected. For simplicity, we also assume that the intensity does only change perpendicular to the edge, but is constant along the edge. Furthermore, we still use Gaussians for PSF and noise filters, so that the combined filter effect on the signal can be described by a single Gaussian convolution at total scale $\sigma = \sqrt{\sigma_{\text{PSF}}^2 + \sigma_{\text{filter}}^2}$.

Without loss of generality, the edge shall be oriented vertically so that the x-axis runs perpendicular to the edge, and the coordinate system shall be translated so that $\vec{x}_0$ is at the origin. Since shading is restricted to the direction perpendicular to the edge, all

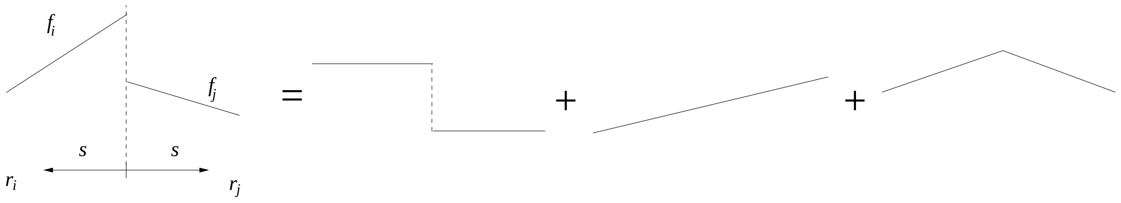


Figure 7.17: Left: Profile of the ideal geometric image perpendicular to the edge between shaded regions r_i and r_j, where f_i and f_j can be approximated as linear functions within the effective radius s of the PSF. Right: This function can be decomposed into a step of height b_d, a ramp with slope a_m and a symmetric roof with slope $\pm a_d/2$.

derivatives in y-direction are zero and have no influence on edge localization. The linear intensity profiles perpendicular to the edge is described by

$$f(x) = \begin{cases} a_l x + b_l & \text{if } x < 0 \\ a_r x + b_r & \text{if } x > 0 \end{cases}$$

When the edge profile changes slowly in y-direction, the parameters a_l, b_l, a_r, and b_r may be interpreted as averages along the edge. The profile $f(x)$ can be decomposed into a step (Θ denotes the unit step function), a ramp and a roof according to

$$f(x) = b_d\,\Theta(x) + (a_m x + b_l) + \frac{a_d}{2}\,|x|$$

with average slope $a_m = (a_r + a_l)/2$, slope difference $a_d = a_r - a_l$ and step height $b_d = b_r - b_l$, see figure 7.17. The decomposition makes quantitative analysis of the blurred gradient magnitude easy, because the derivatives of the individual transitions can be computed analytically – we get a delta function, a constant function, and a step function respectively. The gradient magnitude $b(x)$ of the blurred profile $f(x)$ is simply obtained by blurring the individual derivatives and computing the magnitude of the sum. In case of blurring with a Gaussian $g_\sigma(x)$, we get

$$b(x) = \left|\frac{\partial}{\partial x}\left(g_\sigma(x) \star f(x)\right)\right| = \left|g_\sigma(x) \star \frac{\partial}{\partial u} f(x)\right| = \left|b_d g_\sigma(x) + a_m + \frac{a_d}{2}\,\text{erf}\left(\frac{x}{\sqrt{2}\sigma}\right)\right| \tag{7.18}$$

where $\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2} du$ is the error function. Local extrema of this expression correspond to the zero crossings of its derivative:

$$\frac{\partial}{\partial u} b(x) = b_d g'(x) + a_d g_\sigma(x) = g_\sigma(x)\left(a_d - b_d \frac{x}{\sigma^2}\right) \overset{!}{=} 0$$

There is only one zero crossing at

$$\Delta x = \frac{a_d}{b_d}\,\sigma^2 \tag{7.19}$$

which always corresponds to a maximum of the gradient magnitude. We notice that the localization bias $|\Delta x|$ is zero if and only if the slope difference a_d between the left and

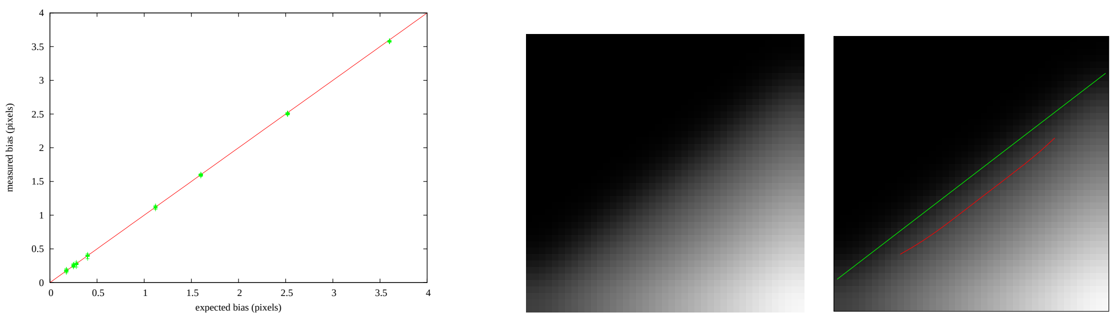


Figure 7.18: Left: Measured bias vs. bias predicted by (7.19) for 224 combinations of edge orientation, subpixel shift t, total scale σ and normalized slope difference $a_d/b_d > 0$ using the subpixel watershed algorithm. Predictions are so good that the spread of the measurements is hardly visible. Right: original image, ground truth edge (green) and detected edge (red) for $\sigma = 4.2$ and $a_d/b_d = 0.2$. The image size is 43×43 and the bias 3.6 pixels.

right region is zero. Step edges are a special case of this condition. Otherwise, the bias is proportional to the slope difference and to the *square* of the Gaussian's scale, i.e. it increases rapidly with scale. This prediction is confirmed by experiment, figure 7.18.

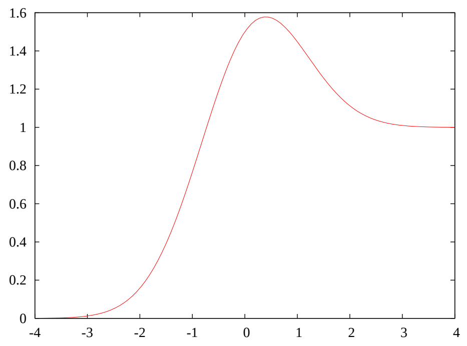
In addition to the bias, shaded region interiors have another undesirable consequence: they reduce the relative contrast of the edge response, and this effect also increases rapidly with scale. The reason is as follows: The gradient filter response of a linear function (i.e. the interior of a shaded region) is independent of the filter scale – it is always equal to the function's slope. The gradient magnitude of a step also increases when the slope difference between the adjacent regions increases, but at a lower rate. Consequently, the edge / non-edge responses become asymptotically equal with increasing slope difference, and this effect is amplified as the filter scale increases, see figure 7.19. If the image is noisy, the effective signal-to-noise ratio may be reduced to the point where the edge cannot be detected anymore: The gradient squared magnitude in the interior of a shaded region with slope a and additive Gaussian noise is a non-central χ^2-distribution with non-centrality parameter $\mu^2 = a^2$, in contrast to the plain χ^2-distribution we got for unshaded regions. This means that the probability curves for false and true edges in figure 7.10 will have increasing overlap, so that the error rate increases.

Edges between shaded regions are thus problematic both in terms of bias and statistical errors. We can take three measures to improve detection performance at these edges:

1. We can reduce the slope in relation to σ_{PSF} by taking images at higher resolution (i.e. smaller viewing distance or higher magnification).

2. When the noise level in the image is sufficiently low, we can use noise filters with small σ_{filter}.

3. We can use edge detectors that are also sensitive for changes in slope (i.e. maxima of the second derivative), for example the boundary tensor to be introduced in section 9.3.



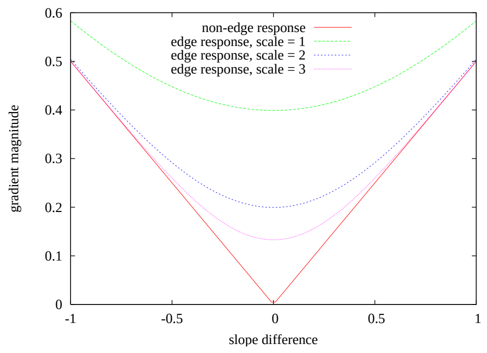


Figure 7.19: Left: Gradient magnitude along the x-axis for a step between shaded regions with $\sigma = 1$, $a_l = 0$, and $a_r = b_d = 1$. The difference between the edge response (the value of the gradient maximum) and the no-edge response in the right region is reduced due shading. Right: Comparison of edge and no-edge response as a function of the normalized slope difference a_d/b_d for various scales. It can be seen that the edge becomes effectively invisible when either the slope difference or filter scale are high enough.

Upon further analysis, it may also turn out that edges between shaded regions cannot be detected by purely low-level means with sufficient reliability under realistic conditions. One could then specifically look for additional (high-level) information that helps in the detection of these edges. Thus, shading certainly calls for more research.

7.3.2 Parallel and Approximately Parallel Edges

Edges frequently occur in pairs. This is obvious in artificial environments, where parallel edges abound due to the convenient tradition of designing objects in terms of rectangles and other basic shapes. But parallel or almost parallel edges are also common in natural environments: tree trunks and branches, animals' legs and bodies, and many fruit shapes are but a few examples. Even if the objects themselves are not characterized by parallel boundaries, the gap between adjacent objects (i.e. the background) may exhibit parallelism. Accordingly, explicit recognition of parallelism is a useful part of the image analysis toolbox, as was recently demonstrated by [Ren et al. 05b].

Error bounds for parallel edges will differ from those for isolated edges as soon as the separation between the two edges becomes less than the effective diameter of the blurring kernel. We will speak of *geometric interference* when the responses of neighboring features start to influence each other. Interference leads to reduced signal-to-noise ratio, higher localization errors and may even cause complete detection failure. In the context of parallelism, we need to distinguish two cases: edges with identical polarity (staircase edges) and with opposite polarity (bar patterns). The error behavior of the two cases is different. Let us first look at edges without noise. As before, we assume a Gaussian PSF and use Gaussian filters, so that the combined scale of the PSF and filter becomes $\sigma = \sqrt{\sigma^2_{\text{PSF}} + \sigma^2_{\text{filter}}}$. Without loss of generality, the edges shall run vertically, i.e. the

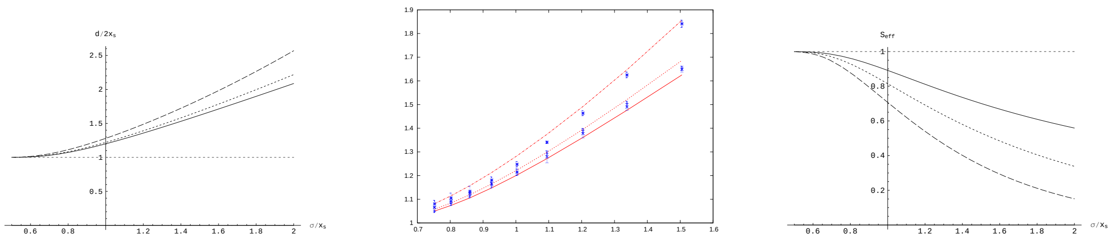


Figure 7.20: Left: Overestimation of the width of a narrow bar pattern: measured relative distance $d/2x_s$ between the edge pair of a bar pattern, as a function of σ/x_s for $a_2 = -a_1$ (solid), $a_2 = -2a_1$ (dotted), and $a_2 = -4a_1$ (dashed). Center: Same diagram with experimental results added (using the subpixel watershed algorithm at SNR=100, 10 different bar widths, and averaging over 20 angles and subpixel shifts at every bar width). Right: Reduction of the effective contrast of a narrow bar pattern: the effective gradient of the small step in a bar pattern (i.e. the one with step height a_1) relative to the gradient of a single isolated step with height a_1, as a function if σ/x_s for $a_2 = -a_1$ (solid), $a_2 = -2a_1$ (dotted), and $a_2 = -4a_1$ (dashed).

edge model is

$$f(x, y) = a_1\Theta(x + x_S) + a_2\Theta(x - x_S)$$

where $\Theta(.)$ is the unit step function, and a_1, a_2 are the contrasts of the first and second step. This function is convolved with a Gaussian with total scale σ, and the edge positions are determined by the zeros $x_{1,2}$ of the second horizontal derivative of the result at $y = 0$

$$\frac{\partial^2}{\partial x^2}(f \star g_\sigma)\bigg|_{y=0} = -\frac{1}{2\pi\sigma^4}\left(a_1\,(x + x_S)\,e^{-\frac{(x+x_S)^2}{2\sigma^2}} + a_2\,(x - x_S)\,e^{-\frac{(x-x_S)^2}{2\sigma^2}}\right) \tag{7.20}$$

When $x_S > 2\sigma$, the edges do not significantly interfere and are simply detected as two separate edges, irrespective of a_1 and a_2. When σ approaches x_S or even becomes bigger, the behavior of the edge detector critically depends on the values of a_1 and a_2.

We speak of a *bar pattern* when a_1 and a_2 have similar magnitudes, but opposite signs. In this case, two separate zero-crossings do always exist, no matter how large the ratio σ/x_S becomes. In the limit $\sigma/x_S \to \infty$ and $a_1 = -a_2$, the distance $x_2 - x_1$ between the zero-crossings approaches 2σ, because the bar is effectively an impuls, and (7.20) approaches the second derivative of a Gaussian. When the magnitudes of a_1 and a_2 differ, the distance between the two edges is overestimated even more when σ/x_s becomes large, see figure 7.20 left and center. When the image is not corrupted by noise, we can still detect the pair of edges correctly, provided that our image and edge data structures are able to represent two separate edges at a very small distance. For example, when $\sigma = 1$ pixel, $x_s = 0.5$ pixels and $a_1 = a_2$, the resulting edge distance in the analog image is about 2 pixels. If we rely on grid-based edge representations at the *original* image resolution (i.e. thin 8-connected pixel edges), correct representation of these edges will often be impossible. Resolution problems can be avoided when the image is interpolated to twice its original size before gradient computation (cf. section 7.1), or a polygonal edge representation is used.

Another problem with parallel edges is that the effective contrast is reduced relative to a single edge with identical step height, especially when the step heights differ on

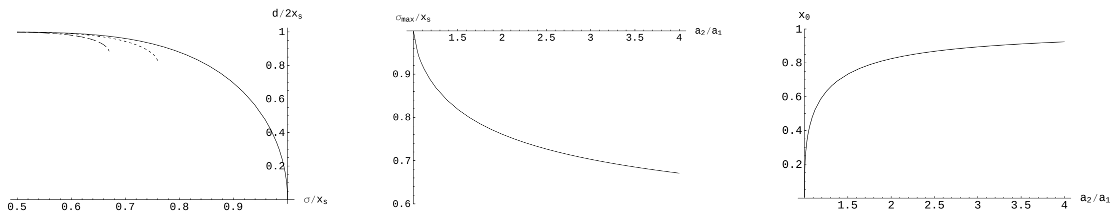


Figure 7.21: Left: Relative reduction in the measured distance between the edges of a staircase pattern as a function of the relative scale, for $a_2 = a_1$ (solid), $a_2 = 2a_1$ (dotted), $a_2 = 4a_1$ (dashed). The graphs end at the scale where one of the edges disappears. Center: Relative scale where a staircase edge pair turns into a single edge, as a function of the relative step height between the two edges. Right: The measured position of the surviving edge for $\sigma/x_s = 1$ moves towards the stronger edge in the pair as the relative step height between the two edges increases (the staircase center position is at $x_0 = 0$).

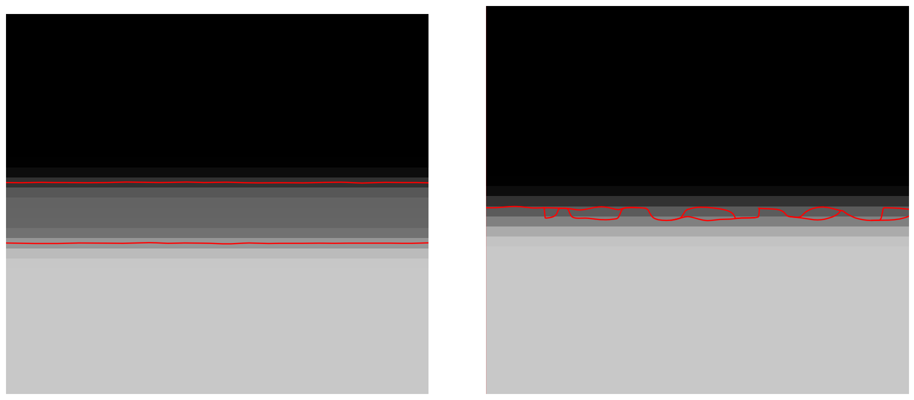


Figure 7.22: Left: A staircase pattern with sufficient distance between the edges (here, $\sigma/x_s = 0.4$) is reliably detected. Right: When $\sigma/x_s \approx 1$, it is no longer possible to reliably detect a pair of edges. However, humans interpret this image as a single edge too. ($a_2 = a_1$ in both images)

the two sides of the bar pattern. This can be seen in figure 7.20 right. For example, a contrast reduction to 50% is observed when $a_2 = -2a_1$ and $\sigma/x_s \approx 1.6$. This leads to a corresponding increase in all noise-related errors.

The staircase pattern occurs when a_1 and a_2 have the same sign. When σ/x_s is small, two separate edges are detectable at the correct distance $2x_s$. The measured distance between the two edges decreases as the ratio σ/x_s increases (figure 7.21 left). If $a_1 = a_2$, the two edges eventually meet when $\sigma/x_s = 1$, and only a single zero crossing (at $x = 0$) exists then. In other words, the staircase is smoothed to the point where it appears as a single boundary. When $a_1 \neq a_2$, the edge pair disappears even earlier (see figure 7.21 center), for example at $\sigma/x_s \approx 0.76$ when $a_1 = 2a_2$. Moreover, the measured position x_0 of the surviving edge moves toward the higher contrast step of the pair, figure 7.21 right. Then the staircase essentially appears as a single edge, see figure 7.22.

The situation is similar when two neighboring edges are only approximately parallel. One common case is encountered when two objects almost touch each other. Although the edges to the sides of the gap are rarely exactly parallel, the detectability limits are similar to those of exactly parallel edges. If the configuration is near the limits, it may happen that the two contours are no longer properly separated, leading to topological errors (falsely merged regions and/or spurious regions). This is especially likely when only pixel-accurate representations are used.

An even more important effect is what we call the *ladder phenomenon*. It occurs in the segmentation of long bar patterns, i.e. when two almost parallel edges with opposite polarity remain close together over a relatively large part of the image. Figure 7.23 illustrates the ladder phenomenon with idealized and real examples. Due to noise, the measured edge positions always oscillate around their true positions (cf. section 7.2.2.4) and don't run exactly parallel. Now consider the gradient at the center between the two edges of the bar. The gradient along the center line would only be exactly zero, if the step heights on either side of the bar were equal, and the oscillations were exactly synchronized. However, this never happens in real images. Instead, the real gradient has local minima where the distance to the two edges is maximal, and it has saddle points where this distance is minimal. Gradient maxima occur on the edges at points where the distance to the other edge is maximal. According to Maxwell's definition of watersheds (definition 5.1 in section 5.1.2), watershed edges run from saddle points to maxima. Consequently, almost parallel edges are always connected by spurious edges, similar to the rungs in a ladder. We have found that these ladder configurations are a major cause of the watershed algorithm's oversegmentation. Since the resulting regions are often very small, they are sometimes missed by pixel-accurate watershed algorithms, whereas they are found very reliably by the subpixel watershed algorithm. This may lead to the impression that the subpixel algorithm is inferior, but this impression is wrong because the ladder edges are artifacts of the boundary indicator image, not of the edge detector.

Detection and removal of undesirable ladder rungs is not easy. Simple thresholding often fails to remove the rungs, because the gradient in the space between two strong edges is not necessarily small. Some good edges are usually disappearing before the strongest bad edges. There is also no straightforward geometric criterion because the ladder pattern is usually very regular and cannot readily be distinguished from a genuine repetitive pattern. Interestingly, Canny's algorithm does not suffer from the ladder phenomenon (see figure 7.24 top): Since Canny's non-maxima suppression looks for local gradient maxima only along the gradient direction, it will not detect ladder rungs because their normal direction tends to be perpendicular to the local gradient direction. However, this advantage comes at the price of missed edges and gaps near junctions, as discussed in section 5.2. Gaps are very problematic because they deprive us of the possibility to compute meaningful region features (e.g. gray-level averages) which are important in many image analysis tasks.

Therefore, the question arises whether one can combine the ability of the watershed algorithm to produce closed contours with the robustness of Canny's algorithm against the ladder phenomenon. Indeed, we have found that the following modification to the subpixel watershed algorithm effectively removes the ladder effect. Recall that the subpixel watershed algorithm works by edge tracing in a boundary indicator function, starting at

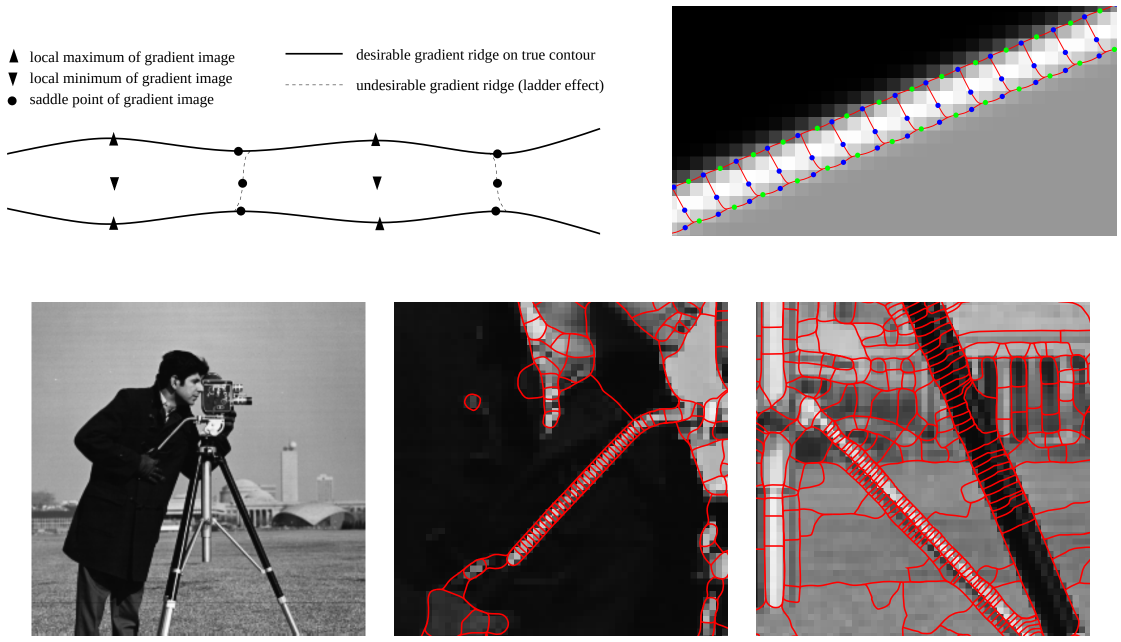

▲ local maximum of gradient image
▼ local minimum of gradient image
● saddle point of gradient image
desirable gradient ridge on true contour
undesirable gradient ridge (ladder effect)

Figure 7.23: Top left: Illustration of the ladder phenomenon: Due to unavoidable irregularities, the gradient image has additional saddle points between parallel edges, giving rise to undesirable ridges that look like rungs in a ladder. Top right: Demonstration of the ladder phenomenon in a test image (SNR $= 100$, true distance between the two parallel edges is 2.2 pixel) using the subpixel watershed algorithm (red: watersheds, green: local maxima of the gradient, blue: saddle points). Bottom: Demonstration in a real image. Due to its many parallel edges, the ladder phenomenon is occurring frequently in the well-known camera man image, as can be seen in the two subregions shown (edge detection with subpixel watershed algorithm, $\sigma_{\text{filter}} = 1.0$, 2-fold oversampling, gradient threshold $= 1.2$).

saddle points of the boundary indicator and moving upwards along the flowline (algorithm 5.4 in section 5.1.2). False positive edges resulting from noise are recognized by the boundary strength at the saddle point (which is always the point of lowest strength in any given edge): An edge is not traced when the strength at the saddle point is below a threshold which is derived from the image's noise characteristics (see section 7.2.2.2). This latter condition can be modified so that a candidate edge is also dropped when the saddle point is apparently located on a ladder edge. We determine whether the local edge normal is pointing in a direction with significant local gray-level change. In order to avoid that this decision has the unwanted side effect of creating edge gaps like in Canny's algorithm, we use a measure for directed edge strength that can represent multiple edge directions in a single point, namely the *structure tensor* [Jähne 02].

Algorithm 7.1: Subpixel watershed algorithm with ladder removal

Input: Image to be segmented.

1. Compute the image gradient by means of a Gaussian derivative filter at scale σ_{filter}:

$$\begin{pmatrix} f_x \\ f_y \end{pmatrix} = \begin{pmatrix} \frac{\partial g_{\sigma_{\text{filter}}}}{\partial x} \star f \\ \frac{\partial g_{\sigma_{\text{filter}}}}{\partial y} \star f \end{pmatrix}$$

2. Compute the gradient magnitude and the structure tensor

$$\|\nabla f\| = \sqrt{f_x^2 + f_y^2}, \qquad \mathbf{S} = g_{\sigma_{\text{filter}}} \star \begin{pmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{pmatrix} = \begin{pmatrix} s_{11} & s_{12} \\ s_{21} = s_{12} & s_{22} \end{pmatrix}$$

3. Use algorithm 3.1 to find the saddle points of the gradient magnitude $\|\nabla f\|$.

4. For each saddle point:

   a) Compute the Hessian matrix $\mathbf{H}$ of $\|\nabla f\|$ and define the initial edge tracing direction $\vec{t}$ by the eigenvector corresponding to the large eigenvalue of $\mathbf{H}$, and the initial edge normal $\vec{n} = \vec{t}^{\perp}$ by the eigenvector corresponding to the small eigenvalue of $\mathbf{H}$.

   b) Compute the large eigenvalue $s_{\max} = \frac{1}{2}\left(s_{11} + s_{22} + \sqrt{(s_{11} - s_{22})^2 + 4s_{21}^2}\right)$ of the structure tensor $\mathbf{S}$ and the tensor projection s onto the normal direction $\vec{n}$: $s = \vec{n}^T \mathbf{S}\, \vec{n}$.

   c) If the ratio $s/s_{\max}$ is above a threshold $s_0 = 0.2$ (meaning that there is significant relative edge strength along direction $\vec{n}$) and the adjusted edge strength $\|\nabla f\| \frac{s}{s_{\max}}$ is above the gradient threshold g_0 (meaning that there is significant absolute edge strength along direction $\vec{n}$)[4], perform edge tracing according to algorithm 5.4. Otherwise, ignore the present saddle point (i.e. drop the half-edge pair starting at this point).

This algorithm is not entirely satisfying because the criterion in step 4(c) is a heuristic which has not yet been justified by theoretical analysis. But the algorithm performs quite well in practice, as figure 7.24 bottom shows: All ladder edges have been correctly recognized, yet all detected contours remain closed. This is in contrast to the result of Canny's algorithm shown in figure 7.24 top, which doesn't exhibit the ladder effect, but has no closed contours.

[4]The threshold g_0 is computed according to table 7.1.

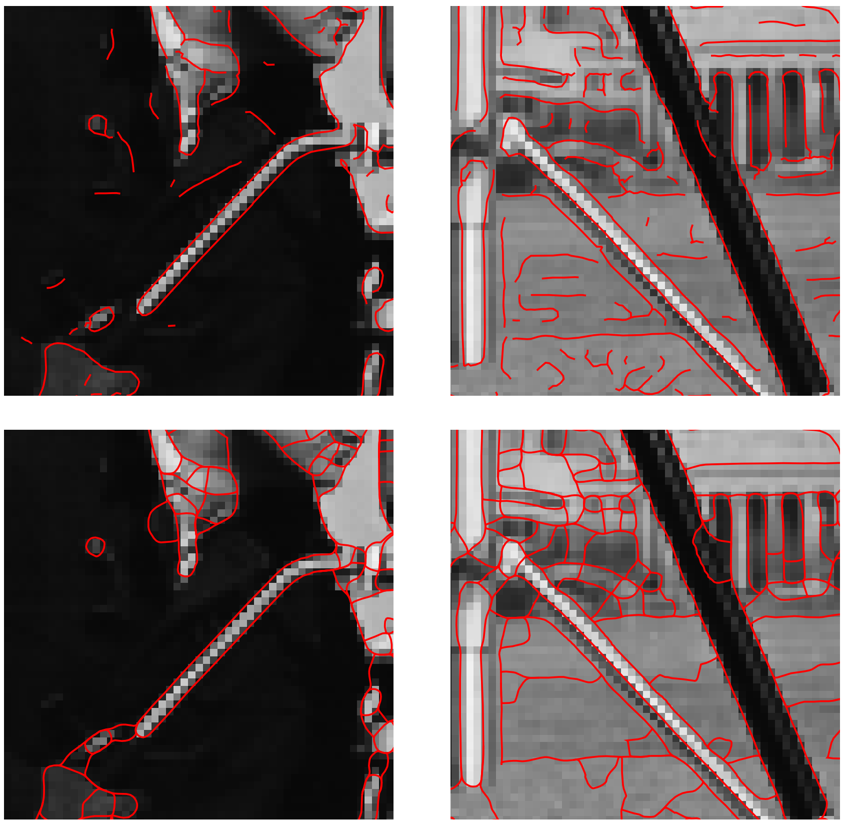


Figure 7.24: Top: Canny's algorithm does not suffer from the ladder phenomenon, but fails to produce closed contours. Bottom: Application of the modified subpixel watershed algorithm that recognizes and removes spurious ladder edges while keeping closed contours (algorithm parameters for both algorithms as in figure 7.23 bottom)

7.3.3 Curved Edges

In case of isolated straight step edges, the symmetry of the configuration guarantees that the relative maxima of the gradient magnitude and the zero-crossings of the second derivative are located precisely at the true edge positions. When edges are not straight, this symmetry is lost, and gradient-based edge detectors will return biased edge positions. The magnitudes of the bias, along with errors due to noise, are addressed in this section.

7.3.3.1 Noise-Free Curved Edges

Systematic localization errors are occurring when an edge is not straight. The prototypical configuration for this situation is a circular region with radius R and contrast a. When the PSF is band-limited, we can analyze this configuration in the continuous domain. The Gaussian PSF and Gaussian noise filters can be combined into a single Gaussian kernel with total scale $\sigma = \sqrt{\sigma_{\text{PSF}}^2 + \sigma_{\text{filter}}^2}$. A closed-form expression for the gradient

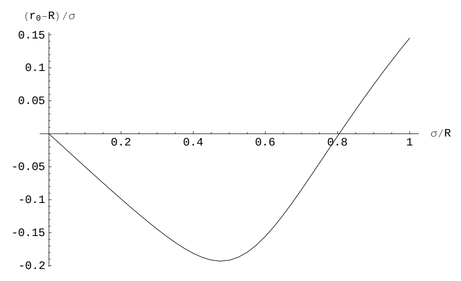


Figure 7.25: Normalized bias of gradient-based edge positions of curved edges, as a function of the ratio σ/R, computed by numeric maximization of (7.21) . An analytic approximation of this curve for $\sigma/R < 0.5$ is given in (7.23).

magnitude of a blurred circular region was derived by [Lim 03, Bouma et al. 05]

$$b(r) = |a| \, \frac{R}{\sigma^2} \, e^{-\frac{r^2+R^2}{2\sigma^2}} I_1\left(\frac{r\,R}{\sigma^2}\right) \tag{7.21}$$

where I_1 is the modified Bessel function of order 1, and r is the radial distance from the center of the region. The edge position is defined by the position of the relative maximum $r_0 = \arg\max_r b(r)$ of this expression along the radial direction. Since no closed-form expression for the position of the maximum is known, figure 7.25 shows numeric results. It plots the normalized displacement $(r_0 - R)/\sigma$ of the detected edge against the ratio σ/R between total scale and region radius. Thanks to these normalizations, the diagram applies to arbitrary scales and radii. When σ/R is large, the blurring kernel is much bigger than the circle. In the limit $\sigma/R \to \infty$, the circle is indistinguishable from an impulse, and (7.21) converges toward the gradient of the kernel itself.

More interesting behavior is observed when the kernel covers only part of the circle. Since the effective radius of a Gaussian kernel is about 2σ, this means we are mainly interested in the interval $\sigma/R < 0.5$. In this interval, the bias is always negative, i.e. the detected curve is displaced toward the circle's interior. For small σ/R, the magnitude of the relative displacement grows almost linearly

$$\frac{r_0 - R}{\sigma} \approx -\frac{\sigma}{2R} \tag{7.22}$$

(the maximum approximation error of this equation is 10^{-3} when $\sigma/R < 0.2$, and 10^{-2} when $\sigma/R < 0.3$). Hence, the *absolute* displacement $|r_0 - R|$ increases with the *square* of the scale, similar to what we found in case of shaded regions, equation (7.19). In other words, the bias grows faster than the scale, so that small PSFs and noise filters are again desirable. An approximation that is valid in the entire range $\sigma/R < 0.5$ is given by

$$\frac{r_0 - R}{\sigma} = 0.52\sqrt{0.12^2 + \left(\frac{\sigma}{R} - 0.476\right)^2} - 0.255 \tag{7.23}$$

(the error of this approximation does not exceed $3 \cdot 10^{-4}$). The maximum displacement of about -0.2σ occurs at $\sigma/R \approx 0.476$. When the relative scale exceeds this size, gradients from the entire contour of the circular region start to interfere, and the detected contour no longer shrinks, but grows again. This eventually leads to the impulse response limit mentioned above.

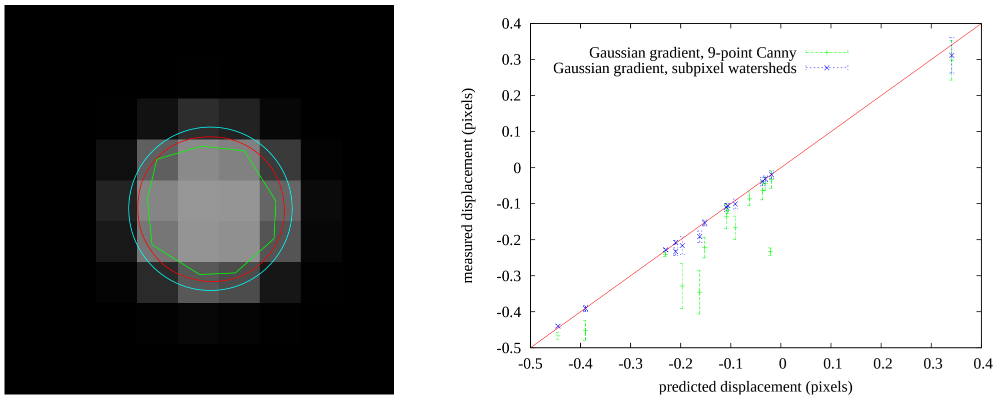


Figure 7.26: Left: Edge detection results for a circular region with $R = 2$ pixels and $\sigma_{\text{PSF}} = 0.5$: true boundary (cyan), subpixel watershed algorithm (red), and 9-point fit Canny algorithm (green), both with $\sigma_{\text{filter}} = 0.9$. Note that edgel computation is purely local, no circle fitting is performed. Right: Measured displacement $r_0 - R$ (average with error bars) for Canny's algorithm with 9-point subpixel correction (green) and the subpixel watershed algorithm (blue) vs. predicted bias according to numerical maximization of (7.21) for $R \in \{2, 4, 10, 20\}$, $\sigma_{\text{PSF}} = 0.5$ and $\sigma_{\text{filter}} \in \{0.7, 1.0, 2.0, 3.0\}$. (Each data point is obtained by averaging along the perimeter of 20 circles which have different subpixel center positions.)

7.3.3.2 Experimental Validation with Artificially Created Curved Edges

In this section we repeat some of the experiments from section 7.2 for curved edges. We first investigate whether the predicted bias is actually reproduced by real edge detectors. Since we use small PSFs and filters, challenging problems are especially posed by small circles. Figure 7.26 left demonstrates that the subpixel watershed algorithm (red) is able to find a very accurate boundary for a region as small as $R = 2$. In contrast, the result of Canny's algorithm (green) suffers from the limitation that at most one edge per pixel is found. The resulting contour is therefore a 9-gon, which is only a coarse approximation to a circle. In this example, the total scale of PSF and edge detection filter was $\sigma = 1.12$, so the configuration is close to the worst case with maximum normalized bias according to theory.

Figure 7.26 right shows a plot of measured displacements against those predicted by the numerical maximization of (7.21). Results of the subpixel watershed algorithm match theoretical predictions very well, whereas the Canny algorithm usually has larger displacements (by magnitude) than predicted. This is partly due to the additional regularization which is implicitly performed when the edge is localized by the least-squares fit in a 3×3 neighborhood (i.e. the 9-point Canny algorithm has a somewhat larger effective scale than σ).

Our next experiment concerns noisy curved edges. We are interested in the question whether the noise error analysis for straight edges (section 7.2.2.3) carries over to curved edges. We first repeat the experiment of figure 7.26 (circle with radius 2) with a noisy input image, see figure 7.27. It can be seen that the region is still correctly located, albeit with larger errors. Moreover, the noise causes the error to change its sign around the

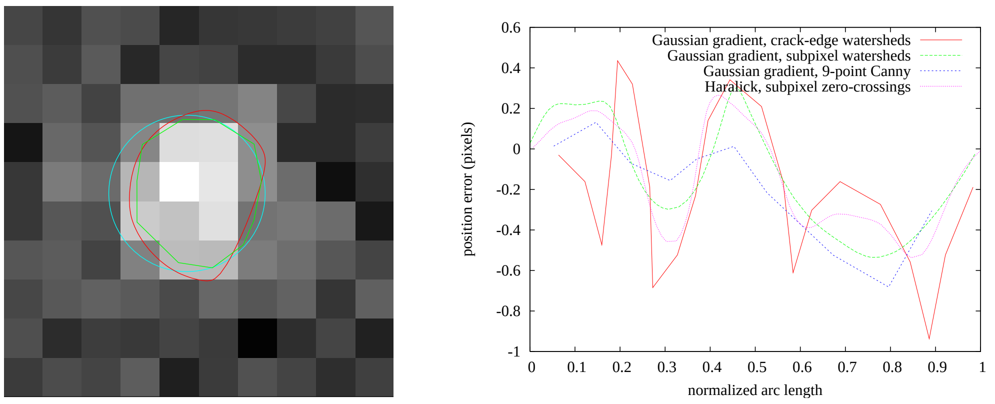


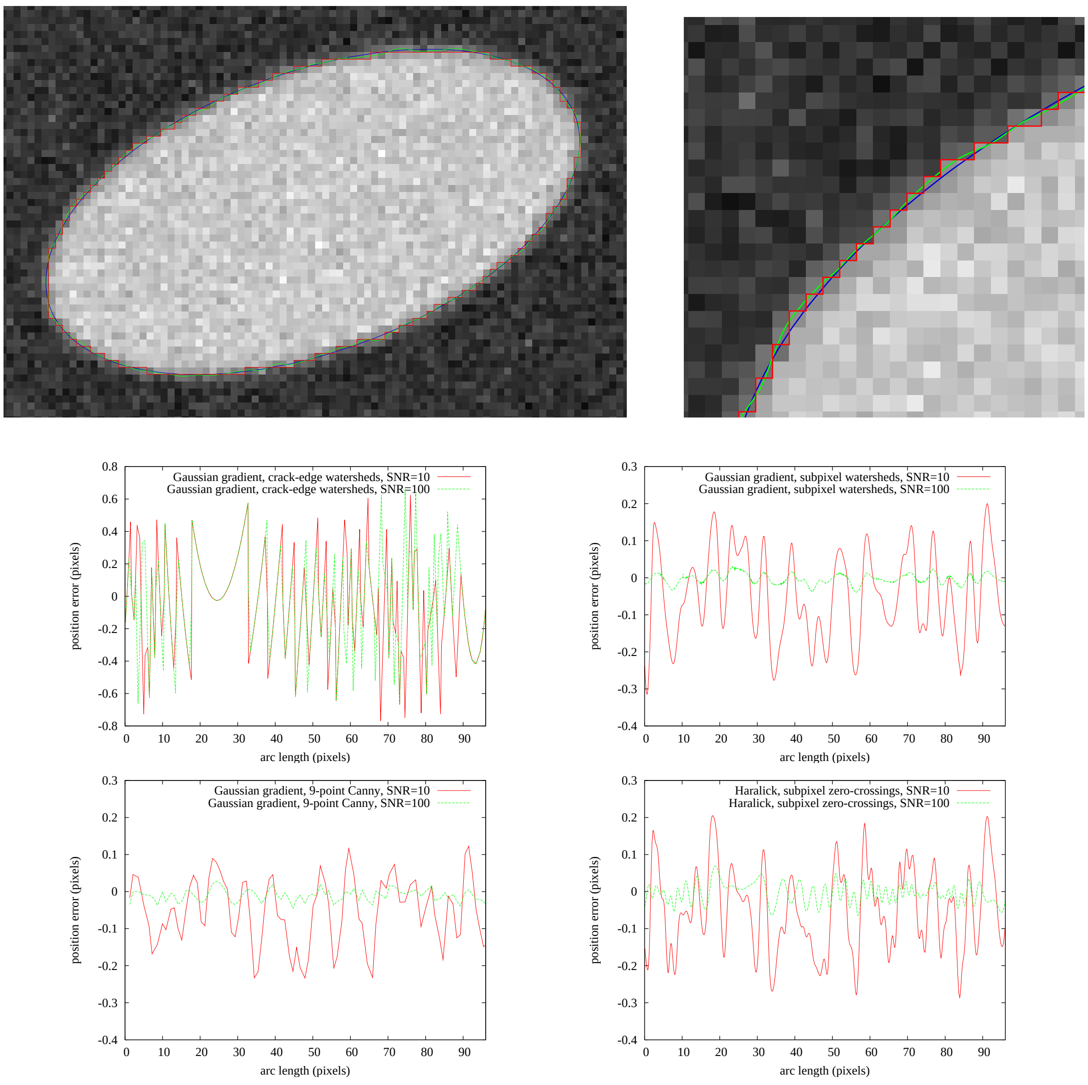
Figure 7.27: Edge detection on a small noisy circle (compare with the noise-free version of this experiment, figure 7.26). Left: Original image with true boundary (cyan), subpixel watershed algorithm (red), and 9-point Canny algorithm (green). Circle radius was 2 pixels, SNR = 10, and $\sigma_{\mathrm{PSF}} = \sigma_{\mathrm{filter}} = 1$. Right: Position error along the entire circle boundary for various algorithms.

method	bias (pixels)	st. dev. (pixels)
theoretical prediction	-0.11	0.17
crack-edge watershed	-0.22	0.35
subpixel watershed	-0.11	0.26
9-point Canny	-0.20	0.23
Haralick	-0.12	0.25

Table 7.2: Comparison of mean and standard deviation of position error with theoretical predictions.

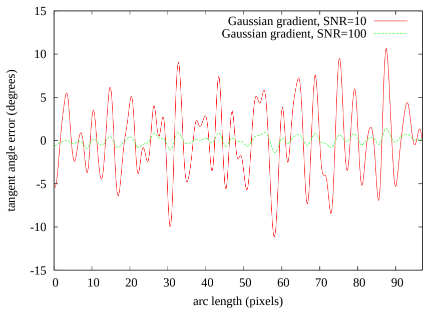
circle contour. Sign changes occur four to six times, in accordance with the prediction of equation (7.16). Table 7.2 compares the measured errors with theoretical predictions of the bias according to equation (7.23) and the standard deviation according to equation (7.9). The standard deviations of all algorithms are higher than predicted, which is no surprise, because the prediction of the standard deviation was taken from the straight edge model, whereas the actual edge is highly curved. Nonetheless, the predictions are good enough to be of practical value, especially as they become better when the edge has lower curvature (see below). The bias of the most accurate algorithms (subpixel watersheds and Haralick zero-crossings) conforms to the prediction. Like in the noise-free experiment, the 9-point Canny algorithm has slightly higher bias (but lower standard deviation) due to its additional regularization. Finally, the errors of the pixel-accurate watershed algorithm are significantly larger than predicted because the prediction doesn't include round-off to pixel coordinates.

Finally, we create test images with a noise-free or noisy ellipse in order to validate theoretical predictions on a contour with varying curvature. Position error results are shown in figure 7.28. Again, theoretical predictions are verified quite well. The accuracy of different subpixel methods is very similar, and the error decreases proportionally to the inverse SNR. In contrast, the accuracy of the crack-edge watershed is dominated by round-off errors to grid coordinates and does not profit from better SNR. All methods

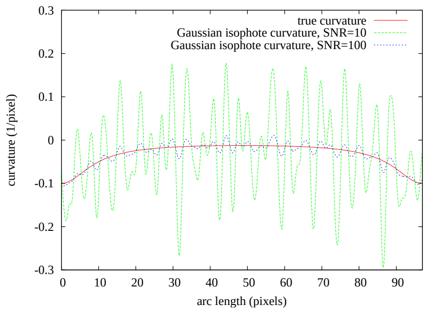


method	SNR=10		SNR=100	
	residual bias (pixels)	st. dev. (pixels)	residual bias (pixels)	st. dev. (pixels)
theoretical prediction	0.0	0.17	0.0	0.017
crack-edge watershed	-0.08	0.32	-0.05	0.30
subpixel watershed	-0.04	0.14	-0.003	0.017
9-point Canny	-0.05	0.11	-0.01	0.022
Haralick	-0.05	0.14	-0.005	0.027

Figure 7.28: Top: Test image (ellipse with radii 40 and 20 pixels, SNR = 10, $\sigma_{\text{PSF}} = \sigma_{\text{filter}} = 1$) with true contour (blue), result of crack-edge (red) and subpixel (green) watershed algorithms. Center: Edge position error for various methods at SNR = 10 and SNR = 100 (the same noise realization was used in both cases). Curves refer to the top half of the ellipse. The expected bias according to (7.23) has already been subtracted from the position error measurements, i.e. the diagrams show residual errors. Bottom: Comparison of residual position errors with theoretical predictions.



method	SNR=10		SNR=100	
	bias (degrees)	st. dev. (degrees)	bias (degrees)	st. dev. (degrees)
theoretical prediction	0°	4.1°	0°	0.41°
subpixel watershed	-0.01°	4.2°	-0.005°	0.46°
9-point Canny	-0.08°	4.2°	-0.01°	0.47°



method	SNR=10		SNR=100	
	bias (1/pixels)	st. dev. (1/pixels)	bias (1/pixels)	st. dev. (1/pixels)
theoretical prediction	0.0	0.087	0.0	0.0087
subpixel watershed	0.0008	0.092	$-6 \cdot 10^{-6}$	0.009
9-point Canny	0.0007	0.093	$3 \cdot 10^{-5}$	0.009

Figure 7.29: Detailed results for the ellipse experiment in figure 7.28 (continued). Curves correspond to the top half of the ellipse. Top: Errors of edge tangent angle and comparison with theoretical predictions. Bottom: Curvature and comparison of curvature errors with theoretical predictions.

exhibit a residual bias, i.e. the bias is larger than predicted by the purely curvature-dependent formula (7.22). This indicates that the localization bias of curved edges also depends on the SNR. However, the residual bias is smaller than the position standard deviation, so more accurate theoretical analysis of the bias may not be worthwhile. The average wave-length of the error along the subpixel watershed curve also conforms to the prediction of (7.16): For $\sigma_{\text{filter}} = 1$, the depicted interval with arc-length 96 pixels should contain 22 error zero-crossings, and the actual number in the diagram is 24.

Experimental results for tangent angles and curvature on the same ellipse are shown in figure 7.29. Again we find very good agreement between theory and experiment. Errors reduce proportionally to the inverse SNR. Formula (7.16) predicts the tangent error to have 38 zero-crossings in the depicted interval, and the curvature error to have 49. The actual values are 39 and 50 respectively. The experiment also highlights the difficulties in curvature estimation: The true curvature of the ellipse is between 0.1 and 0.012. Therefore, the relative error of the isophote curvature according to (7.14) is between 50% and 400%, when a filter of $\sigma_{\text{filter}} = 1$ is used on an image with SNR $= 10$!

7.4 Experimental Validation in Natural Images

In order to check whether our results on artificial edge images carry over to images taken by a real camera, we repeat our experiments on the test images by [Baker & Nayar 99]

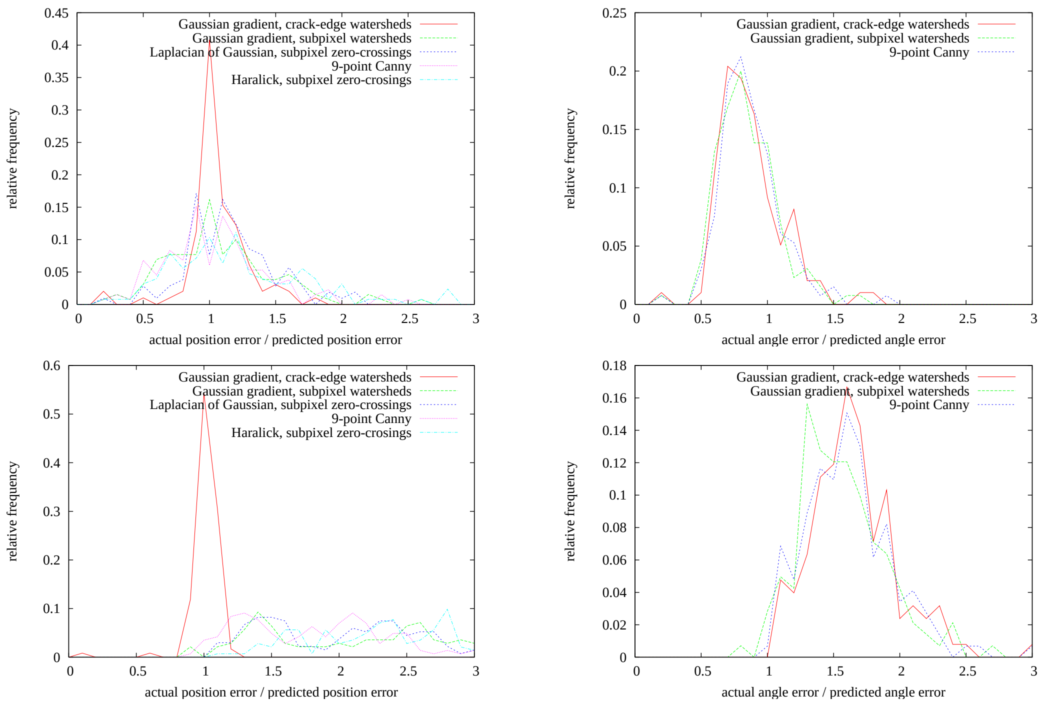


Figure 7.30: Comparison of measured position and angle error with theoretical predictions. Top row: set "benchmark2", bottom row: set "benchmark1".

which we discussed in section 2.2.2.2. In particular, we use the straight edge test sets "benchmark1" (150 images) and "benchmark2" (185 images) and the ellipse test set "benchmark7" (75 images). Results of edge detection algorithms were compared with consensus ground truth according to algorithm 2.1 (see figure 2.12 for two example images with our computed ground truth). Since the object type in each image is known, it was possible to compute the consensus ground truth by a robust global model fit (of an edge or ellipse). The ground truth is therefore significantly more accurate than the purely local measurements of individual edge detectors. Thus, we can expect error estimates to be close to what they would be when exact ground truth were available.

To prepare our experiments, the input images are first subjected to non-parametric noise normalization according to algorithm 3.2. Ground truth was computed afterward because noise normalization leads to a slight shift in the apparent edge position. The SNR after noise normalization was between 17 and 29 (set "benchmark1") and between 3 and 40 (set "benchmark2"). In order to compare the measured errors with theoretical predictions, we also estimated the PSF scale in the straight-edge images by means of the slanted edge technique described in [ISO 12233:2000]. Since many edges are slanted much more than the recommended 5°, we additionally corrected the estimated PSFs for the true slant angle. The PSFs were well approximated by Gaussians, so we computed σ_{PSF} according to the optimal Gaussian fit. It turned out that σ_{PSF} varied between 0.5 pixels and 1.2 pixels, where high values are probably due to imperfect focus.

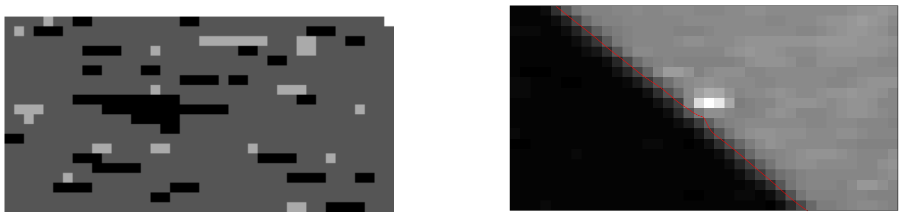
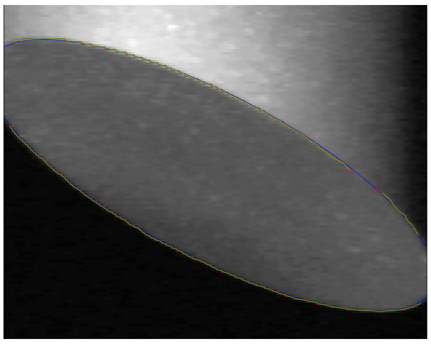
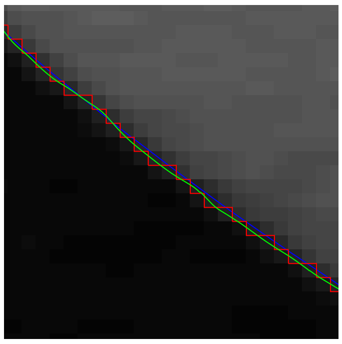


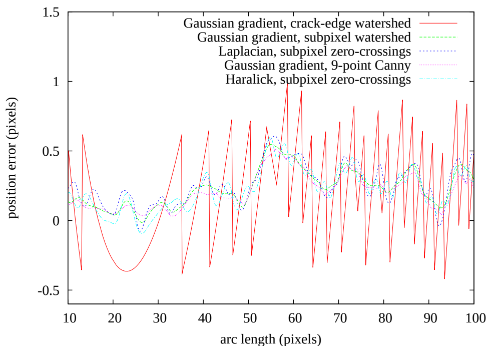
Figure 7.31: Examples for non-Gaussian noise in the "benchmark1" set (image "d2"). Left: Contrast enhancement in the dark image region reveals horizontally correlated quantization noise. Right: Outliers near the true contour cause distortions in the detected edge. Horizontal noise correlation is also visible in the bright image area.

Figure 7.30 shows histograms of the ratio between measured and predicted errors on the straight-edge data sets. It can be seen that the predictions are confirmed very well in the test set "benchmark2", whereas the actual errors are significantly higher than predicted in test set "benchmark1". This is surprising, because "benchmark2" contains the more difficult images, with higher noise levels and partly shaded regions. When we look at absolute results, we recognize that the absolute errors for the set "benchmark1" are indeed small (typical localization errors below 0.1 pixels), but nevertheless exceed the theoretical prediction by a factor of 2 to 3. We believe that the reason for this behavior lies in the noise characteristic of the "benchmark1" images: The noise is markedly non-Gaussian, as is demonstrated in figure 7.31. It is horizontally correlated, contains outliers and is dominated by quantization round-off in dark image areas. These noise characteristics cannot be corrected by point-wise noise normalization according to algorithm 3.2. Consequently, edge position errors are as large as they would have been under Gaussian noise with two or three times higher standard deviation. A more complicated noise normalization algorithm could possibly improve results, provided that the noise is accurately modeled, but the present simple experiment is probably not worth the effort. Since most image analysis algorithms are best suited for Gaussian noise, image acquisition devices with Gaussian noise characteristics should be preferred.

Results for an ellipse image are reported in figure 7.32. Agreement between theory and experiment is generally quite satisfactory, with the exception of the edge localization bias, which should be negative but is actually positive. This is probably due to inaccuracies in the computed ground-truth. Another interesting phenomenon are the somewhat lower errors in the first half of the diagram (up to an arc length of 50). This may be a consequence of horizontal noise correlation: the first part of the contour runs approximately horizontally, so that the negative influence of horizontal noise correlation is less noticeable. Once more, we observe that the curvature cannot be determined with filters of scale $\sigma_{\text{filter}} = 1$ on real images.

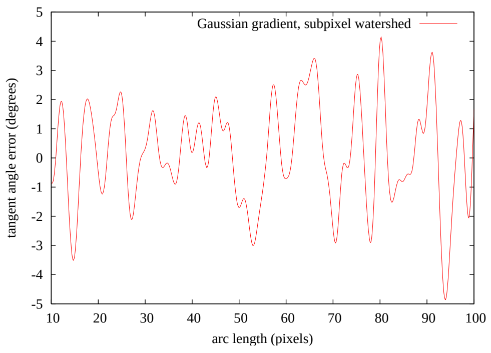




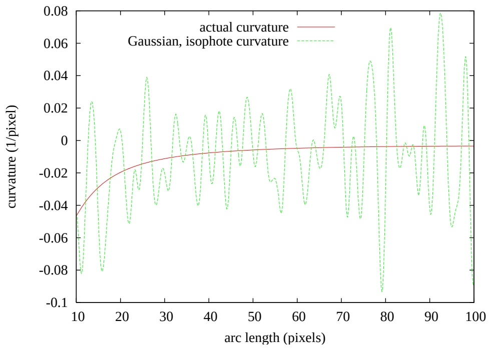


Gaussian gradient, crack-edge watershed
Gaussian gradient, subpixel watershed
Laplacian, subpixel zero-crossings
Gaussian gradient, 9-point Canny
Haralick, subpixel zero-crossings
position error (pixels)
arc length (pixels)

method	bias	st. dev.
	(pixels)	(pixels)
theoretical prediction	-0.01...-0.1	0.11
crack-edge watershed	0.17	0.34
subpixel watershed	0.22	0.13
Laplacian	0.25	0.14
9-point Canny	0.22	0.13
Haralick	0.22	0.15



method	bias	st. dev.
	(degrees)	(degrees)
theoretical prediction	0°	2.7°
subpixel watershed	0.17°	1.5°
9-point Canny	0.18°	1.5°



method	bias	st. dev.
	(1/pixels)	(1/pixels)
theoretical prediction	0.0	0.07
subpixel watershed	-0.044	0.027
9-point Canny	-0.044	0.027

Figure 7.32: Top: Noise-normalized test image benchmark7-a2 with true contour (blue), result of crack-edge (red) and subpixel (green) watershed algorithms ($\sigma_{\text{filter}} = 1$) . In the upper right part of the contour, the edges from local algorithms have a gap, because the edge contrast is lost due to a contrast inversion along the contour. At the point where the edges end, the SNR is about 3. Row 2: Error of edge position for the lower part of the contour, where the SNR decreases from 20 to 15 (the ground truth arc length increases from right to left in the lower part of the ellipse). Row 3: Error of edge tangent angle. Row 4: Comparison of the true curvature with the measured isophote curvature along the subpixel watershed.

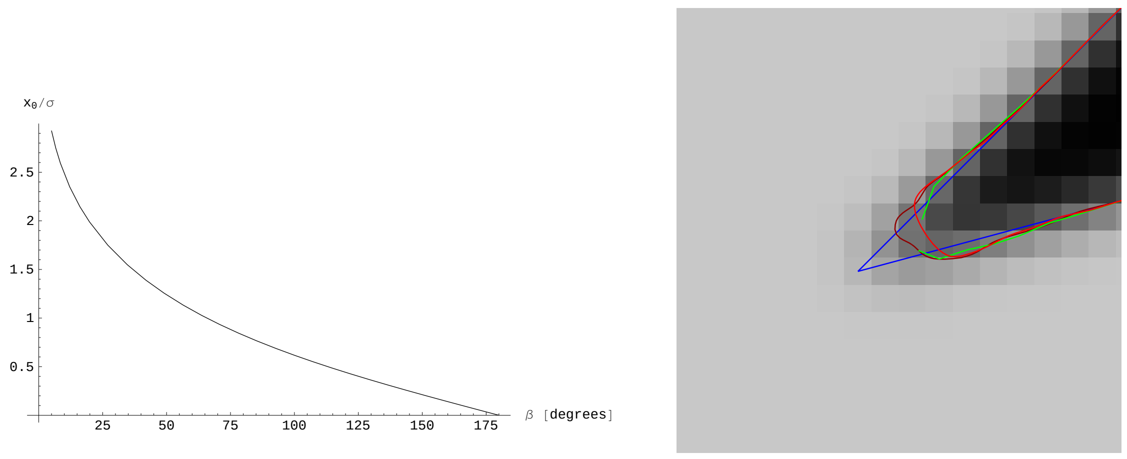


Figure 7.33: Left: Relative displacement x_0/σ along the bisector of an L-corner as a function of the corner's opening angle, according to (7.24). Right: Results of various subpixel edge detectors for a 30°-corner at SNR = 100: ground-truth (blue), subpixel watersheds (red), Canny with 9-point correction (green, with gap), and Haralick operator with subpixel zero-crossings (dark red).

7.5 Corners and Junctions

We have seen in section 7.3.3.1 that curved edges cause a bias in the detected edge positions. Even larger displacements are encountered at sharp corners (see figure 7.33 right). A detailed theoretical investigation of the errors for different boundary indicators was conducted by [Rohr 94]. He derived the following equations for the edge displacement x_0 along the corner bisector for boundary indicators on the basis of the Gaussian gradient. Let x' be the solution of the implicit equation

$$\frac{1}{\sqrt{2\pi}} e^{-x'^2/2} - \left(\tan\left(\frac{\beta}{2}\right)\right)^2 \frac{x'}{2} \left(1 + \operatorname{erf}\left(\frac{x'}{\sqrt{2}}\right)\right) = 0$$

where β is the opening angle of the corner. Then the displacement is

$$x_0 = \sigma\, x' \sqrt{1 + \left(\tan\left(\frac{\beta}{2}\right)\right)^2} \tag{7.24}$$

where $\sigma = \sqrt{\sigma^2_{\text{PSF}} + \sigma^2_{\text{filter}}}$ is the combined scale of the PSF and derivative filter. The edge is always displaced towards the acute angle of the corner, and the (theoretical) displacement becomes maximal on the bisector. Figure 7.33 left depicts the theoretically predicted displacement according to (7.24) as a function of the opening angle. It can be seen that the displacement increases without bound as the angle approaches 0°. Corners with small opening angles are therefore very difficult to detect with local operators. At 15°, the displacement is 2.2σ which can be considered as a practical limit in the sense that detection of smaller corners is not reliably possible with the edge operators considered here.

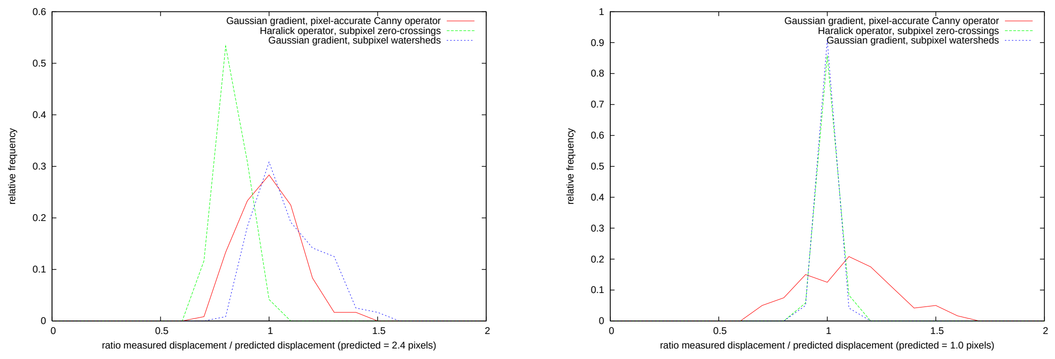


Figure 7.34: Comparison of the predicted displacement according to (7.24) with experimental data at $\sigma = \sqrt{2}$ and SNR $= 100$. Histograms are generated from 120 images with different orientations and subpixel positions of the corner. Left: Corner opening angle 30°. Right: Corner opening angle 90°.

We use the technique described in section 2.2.2.1, equation (2.3) to create test images for experimental validation of these results. Figure 7.33 right shows the actually detected edges of various subpixel-accurate detectors for a low-noise 30° corner image. Significant distortions relative to the ground truth can be noted. The Haralick operator performs slightly better than the others, whereas Canny's algorithm fails to produce a closed contour. 7.34 shows the ratio between predicted and actual errors in more detail. It can be noted that theory and experiment agree very well for subpixel-accurate detectors and 90° corners (fig. 7.34 right), whereas the results of pixel-accurate detectors exhibit high variability due to the additional grid round-off errors. These round-off errors are no longer very noticeable for 30° corners, because the bias is much larger than the pixel distance here (cf. figure 7.34 left). Consequently, the pixel-accurate Canny operator performs nearly as well as the subpixel watershed algorithm (as long as it produces a closed contour). As was already apparent in figure 7.33 right, subpixel zero-crossings computed from Haralick's operator are slightly better than expected for 30° corners. Moreover, the results of all operators exhibit noticeable variability (it is not yet clear whether this variability is caused by noise alone).

Extending this theoretical analysis towards junctions of degree 3 and higher is difficult because the number of degrees of freedom increases rapidly. For example, we can independently choose one angle and two gray levels for a T-junction, two angles and two gray levels for a Y-junction etc. Even more variables would be required if we moved from the step edge model to a piecewise planar gray level model. Then, a simple corner (degree 2) has already seven degrees of freedom: three for the planar intensity model in either of the two adjacent regions, and one for the opening angle of the corner.

There is no simple normalization that reduces the general properties of these configurations into an easily comprehensible form. Moreover, the errors of different detectors are no longer qualitatively similar, as they had been in case of straight edges and corners. It is therefore only possible to analyze a – hopefully representative – set of example junctions. This has been done by [Neumann 88, Deriche & Giraudon 93, Rohr 92, Beymer 91,

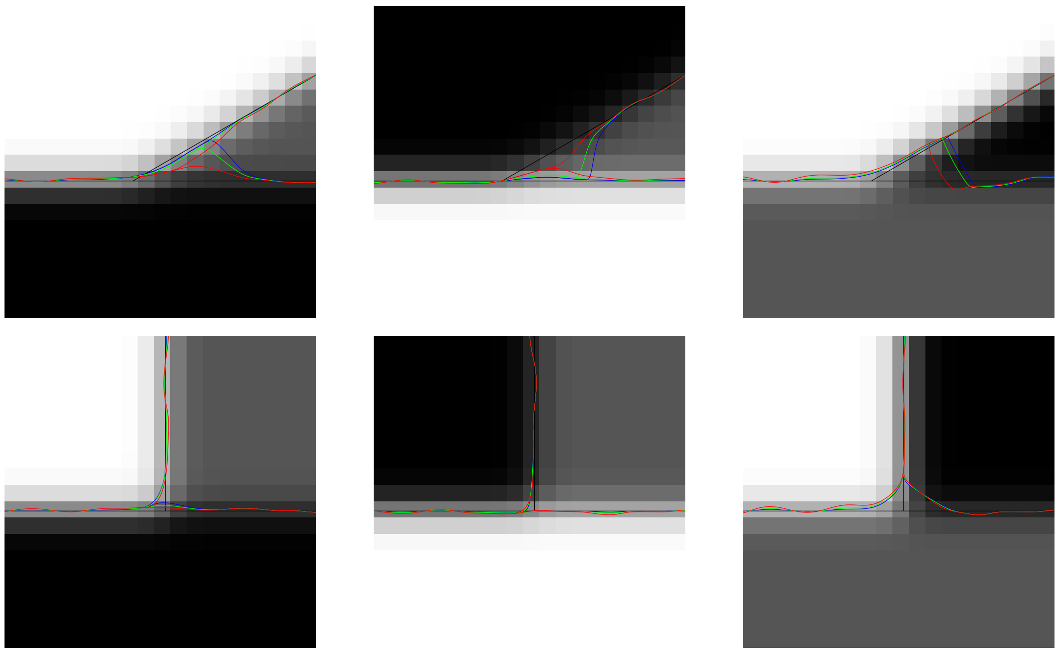


Figure 7.35: Edge distortions at T-junctions for the subpixel watershed algorithm ($\sigma = \sqrt{2}$, SNR $= 10$, background image is shown without noise). Black: ground truth; red: strongest edge (black→white) has twice the contrast of the weakest one (black→gray); green: ratio of strongest to weakest edge is 3 (this corresponds to the background image); blue: ratio is 4.

Rothwell et al. 95], who analyzed the performance of various edge detectors at junctions under the step edge model, and we reproduce some of their experiments with our edge detectors.

Test images are again generated by the technique described in section 2.2.2.1, equation (2.3). When edge detection is performed by means of Gaussian filters and their derivatives, PSF and filter scale add to the total scale $\sigma = \sqrt{\sigma^2_{\text{PSF}} + \sigma^2_{\text{filter}}}$ as usual. In practice, junctions of degree 3 are by far the most common. Since the number of degrees of freedom in these junctions is still relatively low, experiments can cover the parameter space fairly well. The experiments revealed the following typical edge detector behaviors:

1. All edges are distorted near the junction. The magnitude of the distortions depends on the relative strengths between the edges at the same junction (weaker edges have higher distortions), the angle enclosed by adjacent edges (smaller angles lead to larger distortions), and the order of intensity changes around the junction (permuting the intensities may completely change the distortion pattern). Figure 7.35 illustrates this for a number of T-junctions, and figure 7.36 for more complicated junctions of degree 3.

2. The results of subpixel edge detectors are largely rotationally invariant, as figure 7.37 shows. This is not the case for pixel-accurate detectors since round-off errors depend on the orientation of the edges relative to the grid.

3. The edge detectors of the watershed family are producing closed contours. In contrast, edges detected by Canny's algorithm or zero crossings (Haralick's operator, Laplace operator) often have gaps, see figure 7.38. Gaps always occur on the weakest edge, and their size is usually between σ and 2σ, but may exceed 3σ in some cases. These gaps result from the assumption of a single local edge direction, which is inherent to Canny's and Haralick's detectors, but clearly violated near junctions, cf. section 5.2. The gaps roughly correspond to the distorted part of the watershed boundary, and the tendency to form gaps and the size of the gaps increases somewhat when the edges at the junction differ significantly in strength (not shown).

The situation at junctions of higher degree is even more complex. In addition to the phenomena we already observed for junctions of degree 3 (distortions, lower reliability for small angles), the results of the subpixel watershed algorithm show a new kind of error: degree-4 junctions are often split up into several junctions of degree 3. This is illustrated in figure 7.39: Most degree-4 junctions are split into two degree-3 junctions, but if the gray-levels at the junction form a saddle point, we may even get four degree-3 junctions enclosing a small phantom region. In contrast, Canny's and Haralick's algorithms exhibit the gaps we already know from degree-3 junctions. Due to these gaps, we often get a single connected edge plus two dangling edges instead of a degree-4 junction. The ability of the watershed algorithm to maintain closed contours can be considered an advantage because it correctly reproduces region topology and allows the computation of region properties. Conversely, the gaps produced by the alternative algorithms often lead to the entire image to consist of only a single region, so that no useful region properties can be computed.

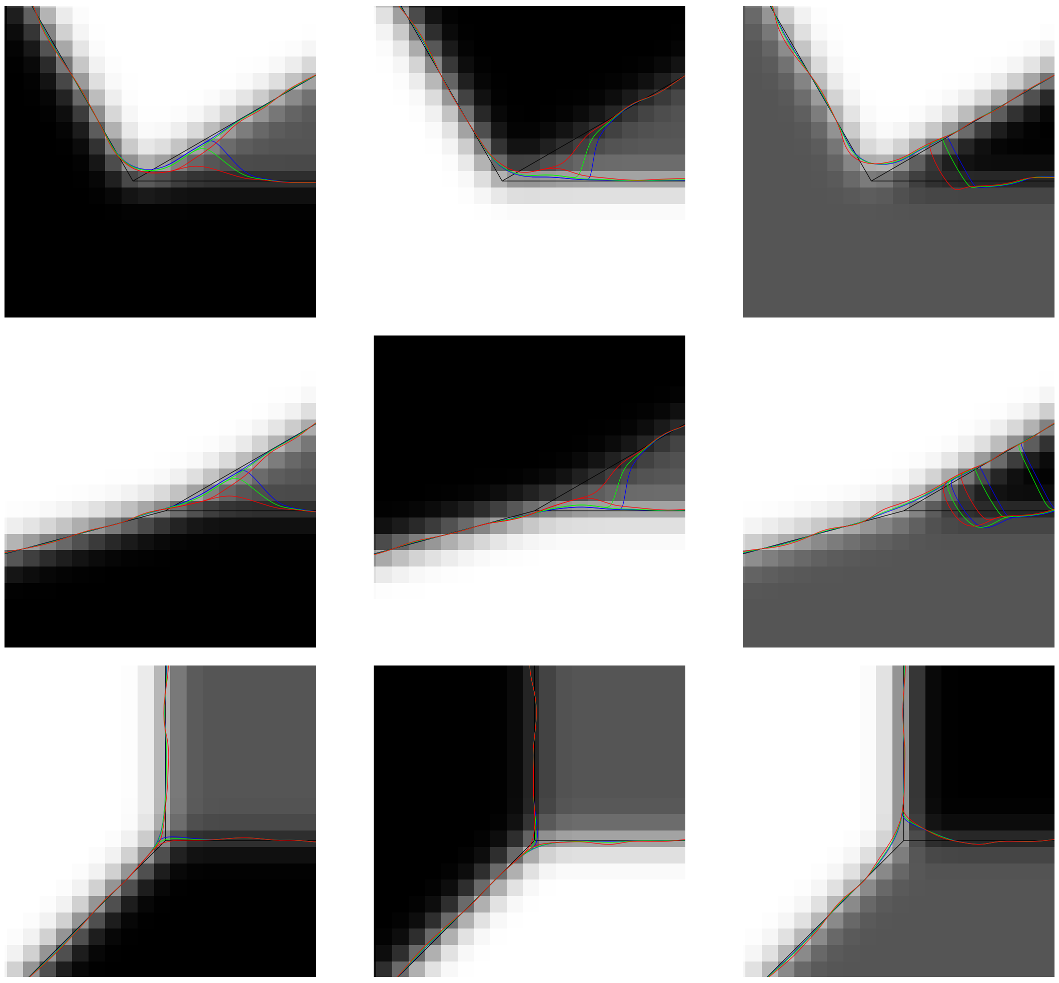


Figure 7.36: Edge distortions at degree-3 junctions for the subpixel watershed algorithm ($\sigma = \sqrt{2}$, SNR = 10, background image is shown without noise). Black: ground truth; red: strongest edge (black→white) has twice the contrast of the weakest one (black→gray); green: ratio of strongest to weakest edge is 3 (this corresponds to the background image); blue: ratio is 4.

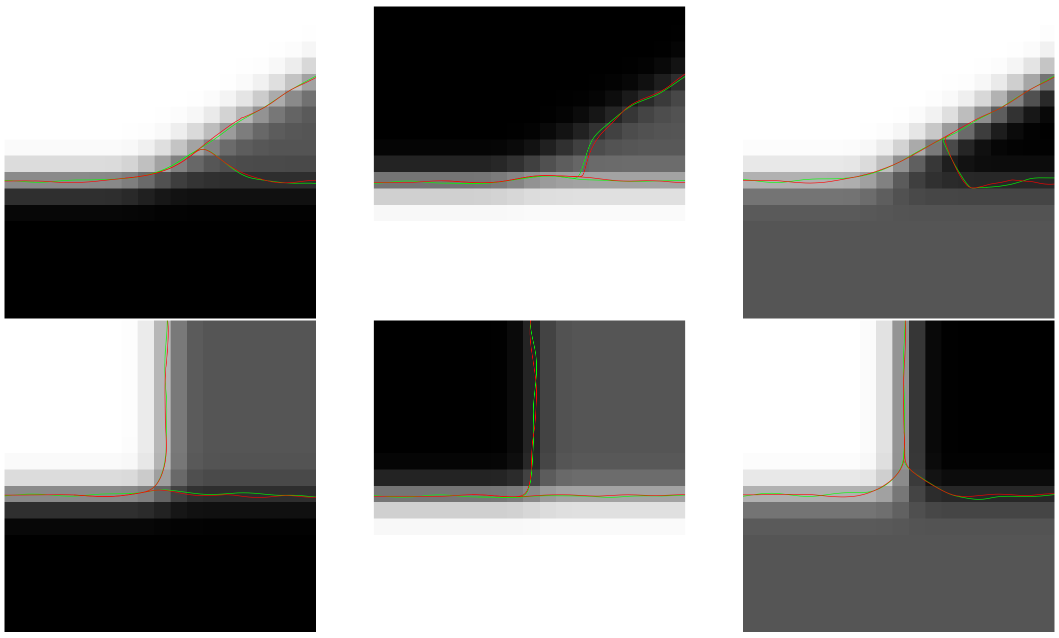


Figure 7.37: Rotational invariance of edge distortions at a junction for the subpixel watershed algorithm ($\sigma = \sqrt{2}$, SNR $= 10$, background image is shown without noise). Green: edges detected in the images shown; red: edges detected in 30° rotated images, aligned with the background image by inverse rotation. The two results should be equal. The small differences visible can be explained by the fact that the noise was different in the two (rotated and non-rotated) images.

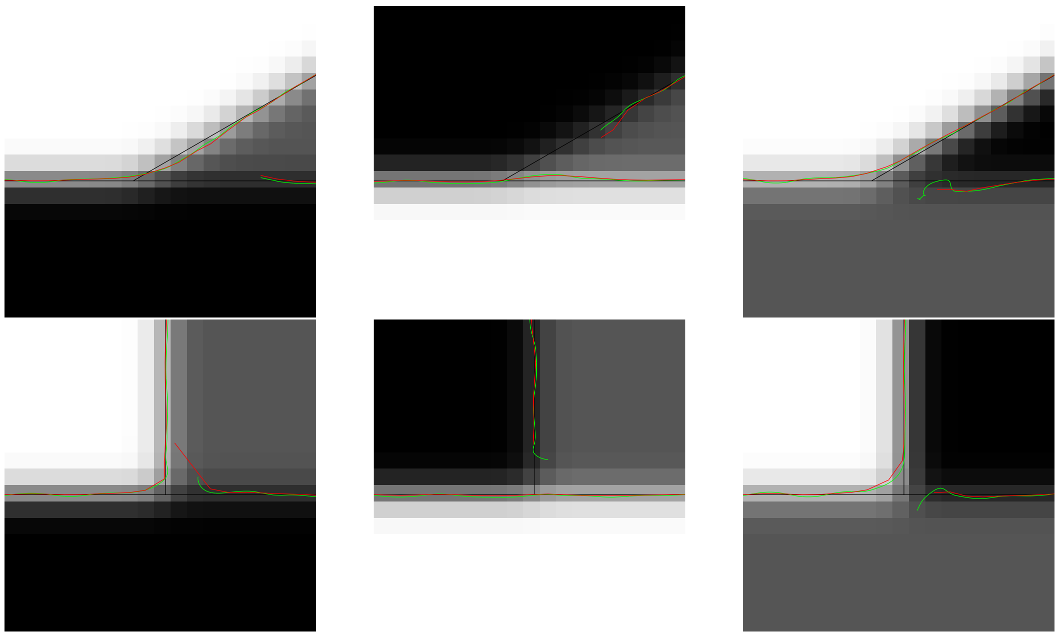


Figure 7.38: Edge gaps at T-junctions for the subpixel Canny algorithm (red) and the subpixel Haralick operator (green) against ground truth (black, background image is shown without noise). Parameters: $\sigma = \sqrt{2}$, SNR $= 10$, ratio of strongest to weakest edge is 3.

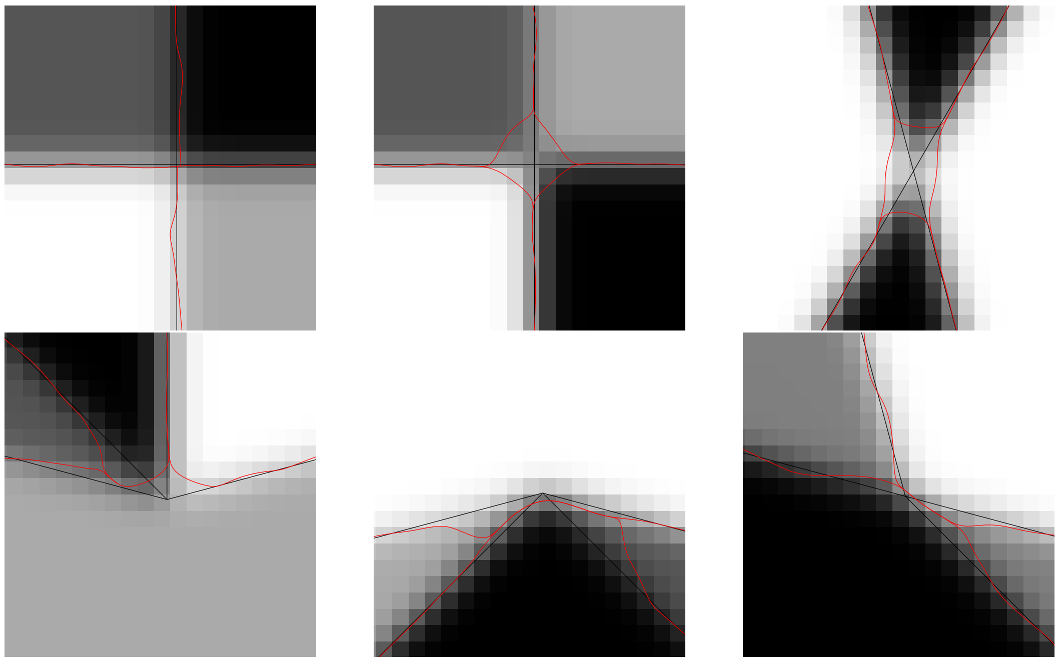


Figure 7.39: Edge distortions at degree-4 junctions for the subpixel watershed algorithm ($\sigma = \sqrt{2}$, SNR $= 10$, background image is shown without noise). Black: ground truth; red: result of the subpixel watershed algorithm.

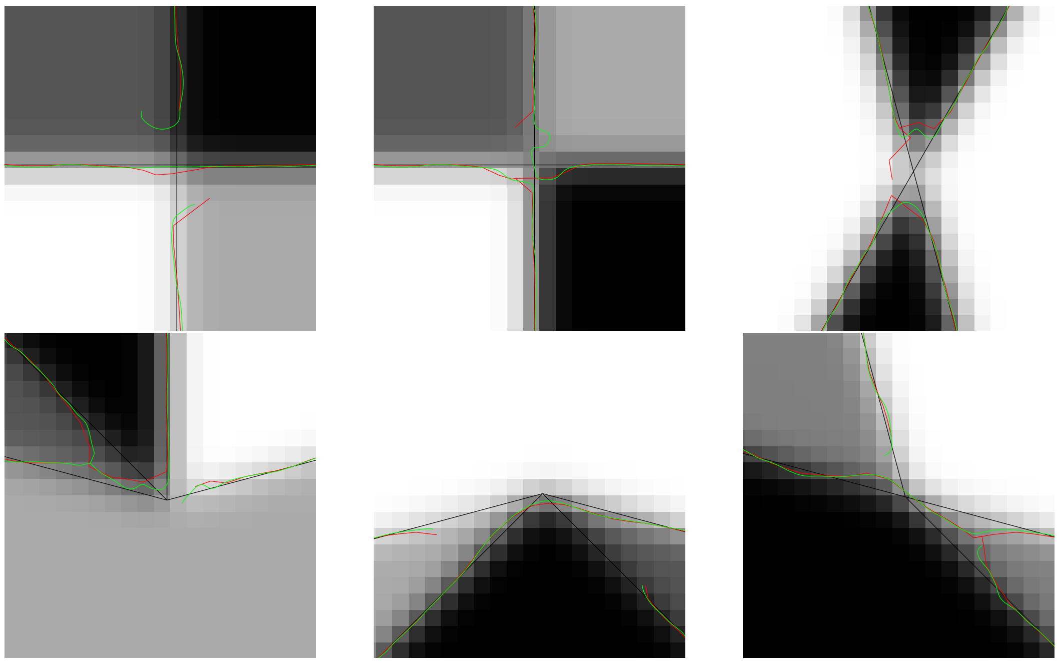


Figure 7.40: Edge gaps at degree-4 junctions for the subpixel Canny algorithm (red) and the subpixel Haralick operator (green) against ground truth (black, background image is shown without noise). Parameters: $\sigma = \sqrt{2}$, SNR $= 10$.

	crack-edge WS		subpixel WS		pixel Canny		subpixel Canny		mid-crack Haralick		subpixel Haralick	
	p	q	p	q	p	q	p	q	p	q	p	q
straight line SNR = 100	0.71 [0.71]	0.71 [0.71]	0.13 [0.11]	0.06 [0.05]	0.79 [0.79]	0.52 [0.5]	0.73 [0.72]	0.09 [0.05]	0.71 [0.71]	0.53 [0.5]	0.11 [0.07]	0.09 [0.05]
straight line SNR = 10	1.0	2.4 [1.4]	0.61	0.58 [0.52]	1.21	1.01 [1.0]	1.14	0.48 [0.52]	0.99	0.96 [1.0]	0.62	0.57 [0.52]
disc, $r = 4$ SNR = 100	0.78	1.0 [0.96]	0.30	0.29 [0.25]	0.96	0.74 [0.75]	0.79	0.34 [0.25]	0.87	0.72 [0.75]	0.34	0.33 [0.25]
corner 90° SNR = 100	1.34	1.55	1.06 [1.0]	0.74 [0.71]	1.17	0.75	1.29	0.73	1.43	0.96	1.06 [1.0]	0.74 [0.71]
corner 30° SNR = 100	2.84	1.84	3.00 [2.4]	0.94 [0.62]	2.69	0.99	2.85	0.60	2.34	0.84	2.42 [2.4]	0.55 [0.62]
T-junction SNR = 100	3.19	4.54	2.89	3.81	3.46	1.68	3.40	1.40	3.26	3.88	3.21	3.40
T-junction ($\geq 30°$) SNR = 100	2.30	4.54	2.00	3.81	2.89	1.32	2.80	1.40	2.61	3.88	2.60	3.40
X-junction SNR = 100	2.65	4.53	3.87	2.86	3.01	2.2	3.07	1.86	3.07	3.31	2.78	3.82

Table 7.3: Maximal errors p and q for various feature types and edge detectors. Theoretical predictions (if available) are given in brackets. Parameters of all algorithms: combined scale of PSF and filter $\sigma = \sqrt{2}$, gradient threshold $t = 1.2$. "T-junction ($\geq 30°$)" means that the minimum angle between adjacent edges was 30°. Otherwise, it was 15°.

7.6 Measurement Errors and the Boundary Sampling Theorem

While the analysis of junction errors by means of examples is interesting and instructive, it doesn't have direct practical consequences, because it cannot answer questions like "Is the resolution of this image sufficient?" or "How large will the error margin be?". Fortunately, the boundary sampling theorem 6.8 allows us to draw more general conclusions about the practical performance of different boundary detectors near junctions. Recall that the sampling theorem guarantees preservation of important topological properties provided that the error of the boundary sampling is sufficiently small relative to the region size. That is, we can analyze a large, representative sample of junctions and measure p (the maximal distance of any true boundary point from a detected edgel) and q (the maximal distance of any detected edgel from the true boundary). In order for the topology to be preserved despite of these errors, the correct (ground-truth) boundaries must be r-stable with $r > p + q$, and every region must contain a circle with radius $2r$. Table 7.3 lists the measured errors for various situations and various detectors.

The computation of theoretical predictions for p and q is simplified because these numbers represent maximum errors. Thus, the theoretical maximum errors from independent sources (e.g. from noise and from round-off to pixel coordinates) can just be added. Specifically, predictions have been obtained as follows:

- The predictions of p and q for pixel-accurate detectors in the low-noise case are taken from the rounding error analysis of the corresponding sampling schemes in section 6.2.1 (additional statistical errors can be neglected due to the high SNR).
- The predictions for pixel-accurate detectors in noisy images are the sum of the maximum statistical errors according to equation (7.9) and the rounding errors.
- The error q of subpixel-accurate detectors for straight lines is computed by means of equation (7.9). The corresponding error p is obtained as $p = \sqrt{d^2/4 + q^2}$, where d is the maximum distance between subsequent edgels. In case of the Canny algorithm, we have $d = \sqrt{2}$ because at most one edgel is returned per pixel. The other subpixel-accurate algorithms support adaptive step size control, and we adjusted the steps so that we got $d = 0.2$ for the subpixel watershed algorithm and $d = 0.1$ for the subpixel Haralick algorithm.
- The predictions of p for corners are the displacements according to equation (7.24), and the corresponding q is the distance of this displaced point from the ground truth contour, i.e. $q = p \sin(\phi/2)$, where ϕ is the corner's opening angle.

We can make a number of interesting observations in this table.

1. The subpixel algorithms are indeed more accurate than the corresponding pixel-accurate algorithm versions, usually by a large margin. Theoretical predictions are pretty close to the experimental data, with the exception of q for the pixel-accurate watershed algorithm, where some false positive edges (due to oversegmentation) cause larger experimental errors than expected.
2. The error at corners and junctions is much bigger than the error at straight and curved lines.
3. No algorithm is a clear winner.

The subpixel watershed algorithm has the lowest values for p since it is able to maintain closed boundaries (i.e. there are no false negatives due to missing boundary parts). In contrast, it has relatively high values for q since the well-known oversegmentation of the watershed transform results in extra edges which are often located far from the true boundary. It is unclear whether the advantage of closed contours can compensate for the disadvantage of oversegmentation, especially in the context of (α, β)-boundary reconstruction which only considers edgels as points and ignores their original links. In terms of the sum $p+q$ (which is the most relevant characteristic because the ground-truth plane partition is required to be $(p+q)$-stable), the subpixel Canny algorithm is best. Its disadvantage of having large p for straight and curved edges can be compensated if it is applied to 4- or 8-fold oversampled versions of the original image – then the point density along the edge becomes comparable to that of the other subpixel algorithms. Figure 7.41 contrasts the original segmentation of two example junctions with (α, β)-reconstruction (algorithm 5.12) on the basis of parameters selected according to table 7.3. It can be seen that the differences between edge detectors are smaller after (α, β)-reconstruction than they were before. In the upper example, all algorithms make a topological error (the watershed algorithm creates a spurious region, the others leave a gap and merge two regions) which is corrected by (α, β)-reconstruction.

The large discrepancy between algorithm performance for edges and for junctions is a big problem. We had seen earlier that subpixel edge detectors can readily segment parallel edges with a separation as small as two pixels or circular regions with a radius smaller than 2, and $\alpha < 1$ can be used for (α, β)-reconstruction of isolated straight or curved edges. However, these α-values are not nearly large enough to close gaps at junctions or to detect uncertain junction configurations. But when we choose larger α-values according to the maximum errors at junctions, we will no longer be able to resolve small regions.

In practice, the problem is not quite as bad, because table 7.3 represents maximum errors that guarantee topological correctness in all circumstances. If we only require junctions to be correct with a certain high probability, we can get good results with much smaller α-values, because very problematic junctions occur relatively infrequently in actual images. Nevertheless, it remains an important question whether boundary indicators with better performance near junctions can be developed. We will come back to this question in chapter 9.

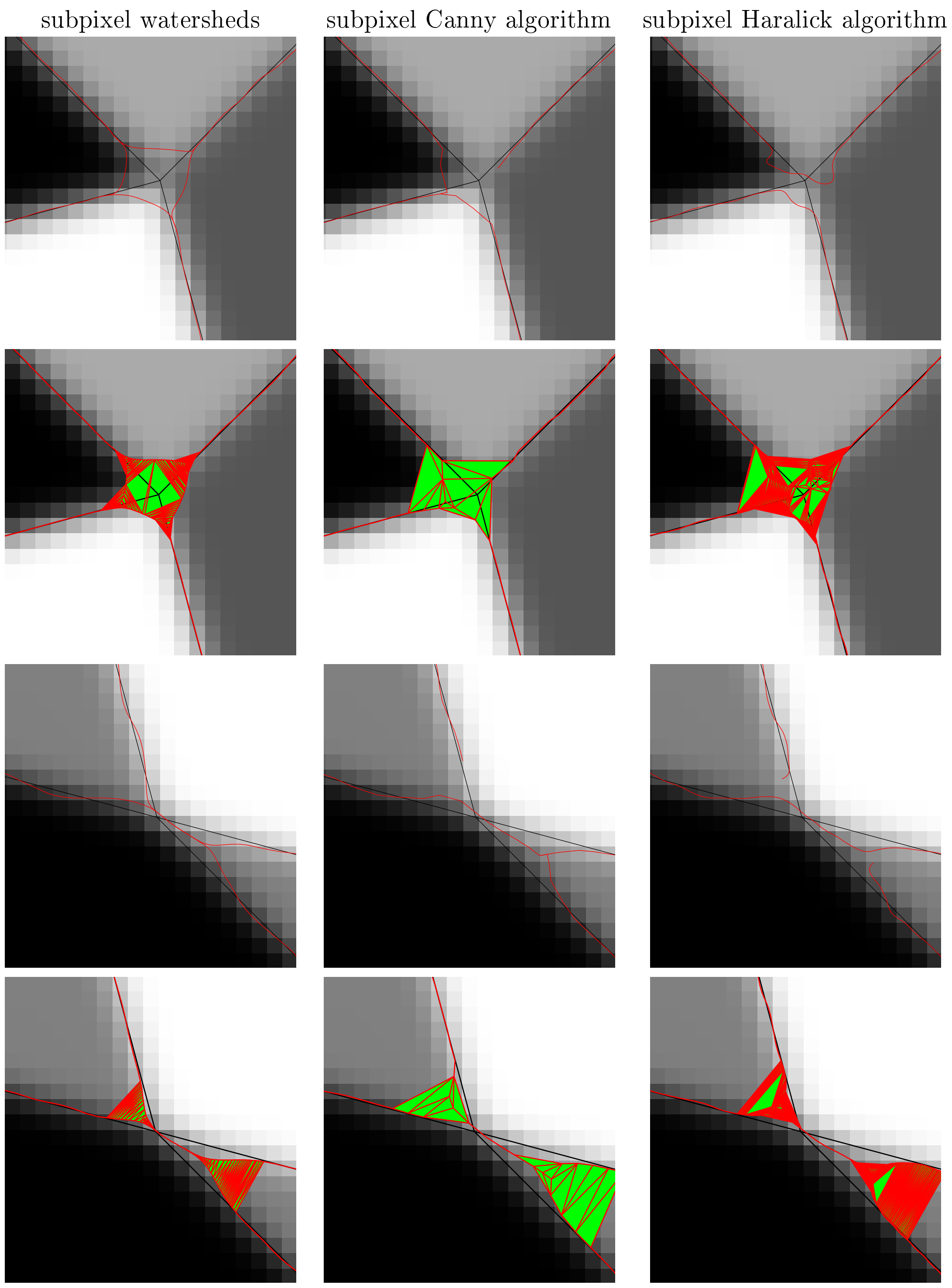


Figure 7.41: Comparison of (α, β)-reconstruction (rows 2 and 4) with the original segmentations (rows 1 and 3). Parameters: SNR $= 100$, $\sigma_{\text{PSF}} = \sigma_{\text{filter}} = 1$, $\alpha = 2.0$ (subpixel watershed), $\alpha = 2.8$ (subpixel Canny), $\alpha = 2.6$ (subpixel Haralick), $\beta = \alpha$.

7.7 Examples

A few examples shall demonstrate that the methods we presented also work well on real images. First, consider figure 7.42. It shows a texture that is subjected to significant perspective foreshortening. Thus, the image resolution reduces rapidly toward the top image border. We compare the performance of a standard pixel-accurate Canny detector at original resolution with a subpixel watershed algorithm on an oversampled image. It is clearly seen that the watershed segmentation maintains fidelity almost to the top border (although with slight oversegmentation), whereas the traditional Canny edge image has already many errors in the lower half of the image.

The good performance of the subpixel watershed algorithm is also demonstrated in figure 7.43, where it is able to resolve individual planks on the building's wall, instead of just interpreting the wall as a texture.

Figure 7.44 illustrates the application of edge detectors to a color image. Here, we replaced the gray-level gradient with a color gradient defined as

$$g(x,y) = \sqrt{\lambda_1 - \lambda_2} \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} \tag{7.25}$$

where $\lambda_1 \geq \lambda_2$ are the eigenvalues of the color gradient tensor

$$\mathbf{C} = \begin{pmatrix} r_x^2 + g_x^2 + b_x^2 & r_x r_y + g_x g_y + b_x b_y \\ r_x r_y + g_x g_y + b_x b_y & r_y^2 + g_y^2 + b_y^2 \end{pmatrix}$$

and $(\cos\phi, \sin\phi)^T$ is the eigenvector corresponding to λ_1. Despite the high color contrast, the standard Canny operator cannot segment the smaller features (especially the window)

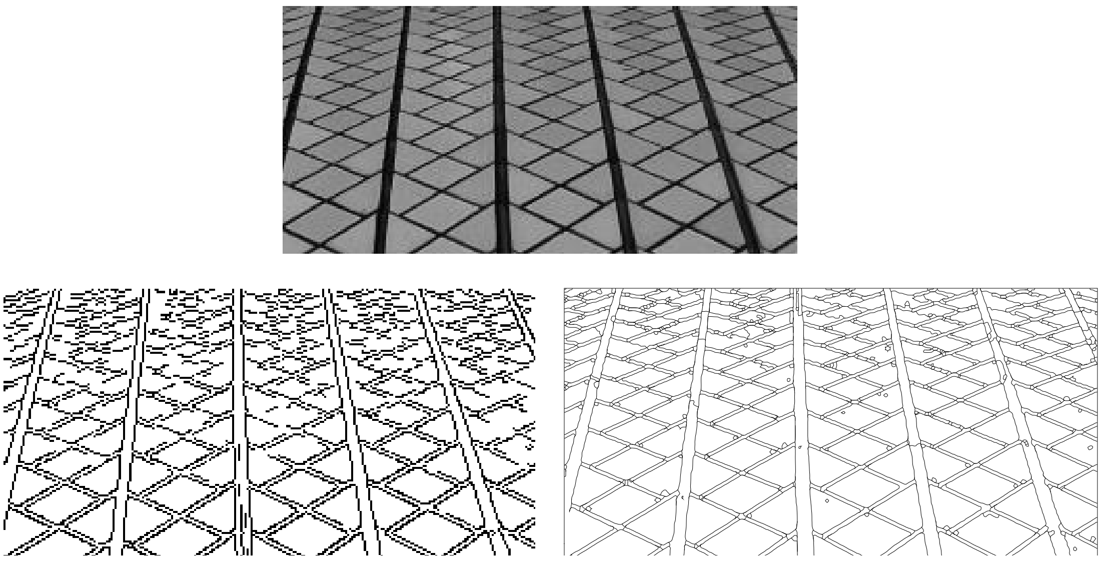


Figure 7.42: Texture foreshortening example. Bottom left: pixel-accurate Canny algorithm ($\sigma_{\text{filter}} = 0.8$, gradient threshold 10). Bottom right: subpixel watershed algorithm (two-fold oversampling, $\sigma_{\text{filter}} = 1.6$ in the oversampled coordinate system, gradient threshold 5).

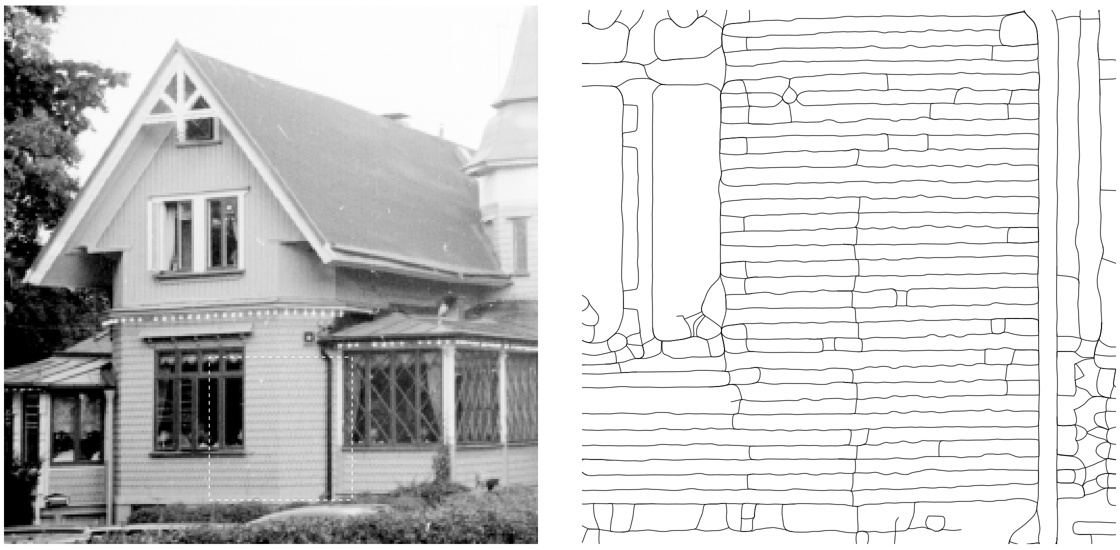


Figure 7.43: Result of the subpixel watershed algorithm with ladder removal (algorithm 7.1) on a low-resolution region (two-fold oversampling, $\sigma_{\text{filter}} = 1.6$ in the oversampled coordinate system, gradient threshold 0.4).

and has many false positives (spurious edges) and false negatives (gaps). In contrast, the subpixel watershed algorithm (which only makes use of the magnitude of $g(x, y)$, but on a two-fold oversampled image) resolves small details correctly, has no gaps, and only slight oversegmentation.

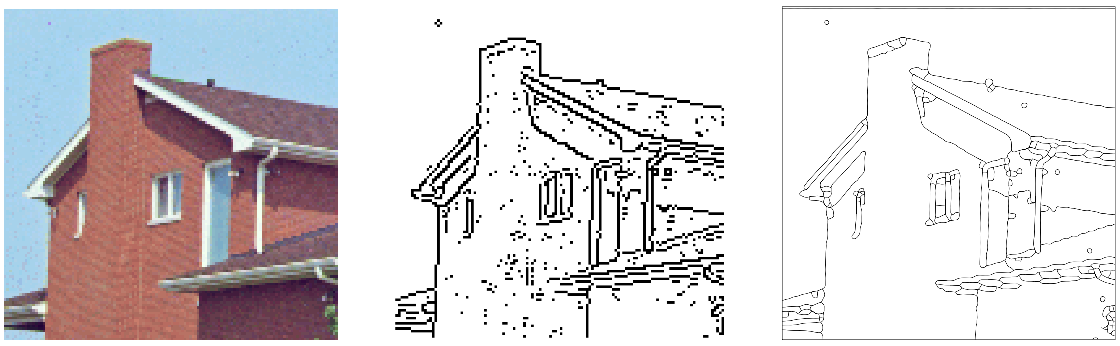


Figure 7.44: Left: original color image. Center: result of pixel-accurate Canny algorithm on color gradient ($\sigma_{\text{filter}} = 0.7$, gradient threshold 12). Right: result of subpixel watershed algorithm (two-fold oversampling, $\sigma_{\text{filter}} = 1.4$ in the oversampled coordinate system, gradient threshold 6)

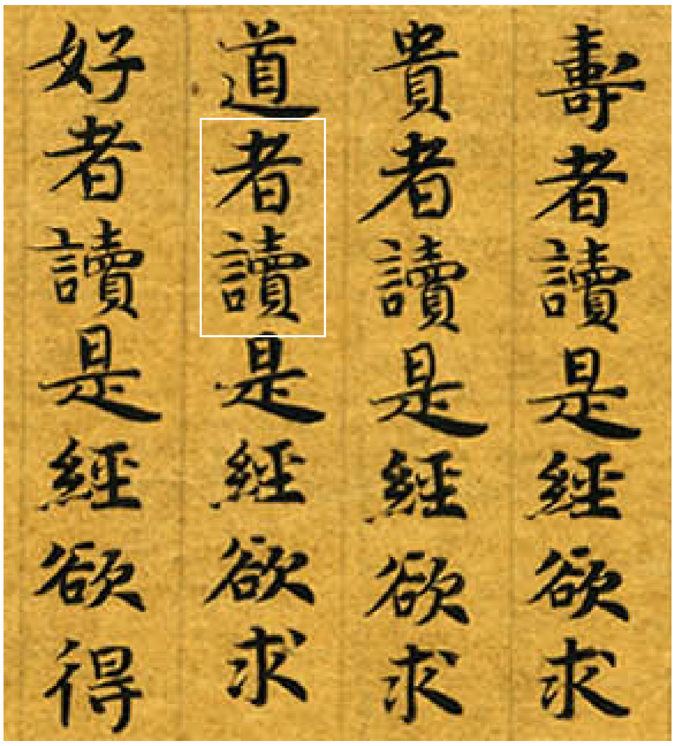

壽者讀是經欲求
貴者讀是經欲求
道者讀是經欲求
好者讀是經欲得

Figure 7.45: Part of an ancient Chinese manuscript. This fragment gives an impression of the shape variations as the text happens to be very similar in four consecutive columns. The white rectangle indicates the characters to be shown in figure 7.46.

A Case Study

To conclude this section, we present a small case study that highlights particularly well how the methods outlined in this chapter help in obtaining good segmentation results. The task in this project is the analysis of ancient Chinese manuscripts which have recently been excavated and sometimes date back as far as 300 B.C. (figure 7.45). The specialists for these manuscripts want to analyze various writing characteristics in order to identify scribes and their schools. This knowledge would be a tremendous help in various tasks such as the correct grouping of fragments into entire works, their precise dating, and their assignment to different thought traditions. Decisions about these questions should be based on objective text characteristics (such as shape features of the calligraphy), rather than content-based text analysis, because the latter may be biased by the interpreter's modern background: Since interpretation of ancient manuscripts is still a relatively new field, there is a certain risk that knowledge about modern interpretations of a single character or an entire text passages might impose some unconscious prejudice on attempts at semantic text analysis. Purely geometric text features are not susceptible to this risk.

Regardless of which geometric features will eventually turn out to be useful, segmentation of individual characters is a useful (perhaps necessary) processing step before such characteristics are computed. Since differences between scribes may by very subtle (such as tiny variations in stroke widths and angles), these segmentations should be as accurate as possible. Inspection of the input data reveals that the width of some important strokes is below 2 pixels, sometimes even only 1 pixel. Likewise, the size of certain stroke characteristics that reveal a stroke's direction is only in the order of single pixels. Furthermore, stroke angles must be measured with an accuracy below 5 degrees to be useful. These requirements clearly suggest that a subpixel-accurate segmentation algorithm has to be used[5]. Moreover, twofold oversampling is required to avoid aliasing in gradient-based algorithms.

Since dark characters on bright background are binary shapes, thresholding comes immediately to mind as a method of choice, but edge detection should also give reasonable results. (Image analysis is performed on the red channel only because it has the highest contrast.) Figure 7.46 shows results for three segmentation methods. It can be seen that pixel-accurate thresholding recovers neither the geometry nor the topology with sufficient accuracy to be useful for the present application (figure 7.46 left). In contrast, the subpixel thresholding algorithm 5.2 does indeed give very good results (figure 7.46 center). The subpixel watershed algorithm (figure 7.46 right) is also very good, but exhibits some oversegmentation, especially in the lower character, which is more difficult due to its very narrow strokes and very small holes. Upon closer inspection one recognizes that some of these additional edges are indeed supported by significant intensity gradients, which remain invisible to the thresholding segmentation.

The manuscript example is of particular interest because it allows us to predict the quality of the segmentation by means of the theoretical tools described in this work. In order to apply these tools, we must first analyze the basic properties of the data – the size of the features to be detected, the noise characteristic of the sensor, and its PSF. Measurement of these characteristics should be an intrinsic part of the image acquisition process. Since this has not been done here, we must estimate them as good as possible from the data itself. By visual inspection, we find that the width of the smallest features of interest is in the order of the pixel diameter. To determine the PSF (e.g. by means of the slanted edge technique described in [ISO 12233:2000], cf. section 3.2.3), a straight edge with sufficiently large homogeneous regions on both sides or a similar well-defined structure would be needed. Unfortunately, the image does not contain such a structure, so we can only guess the PSF. From the values of typical detectors and visual comparison with images whose PSF is known, we use a probable value of $\sigma_{\text{PSF}} \approx 0.6$. Mean and noise standard deviation in the bright background region are $v_{\text{bg}} = 215$ and $s_{\text{bg}} = 9.0$, whereas in the dark foreground we have $v_{\text{fg}} = 14.9$ and $s_{\text{fg}} = 9.0$. We can therefore perform a linear noise normalization according to equation (3.20), which in this particular case simply amounts to a division by the noise standard deviation. After noise normalization, the noise variance becomes unity, and the average intensities of background and foreground are $v_{\text{bg}} = 24.0$ and $v_{\text{fg}} = 1.65$. The signal-to-noise ratio is thus SNR ≈ 22, and a graylevel threshold of $t = 13$ suggests itself (figure 7.46 left and center).

[5]The obvious solution to take images at higher resolution is not applicable because access to the original documents is extremely restricted. To make things worse, existing images are often corrupted by non-Gaussian noise (probably JPEG compression noise).

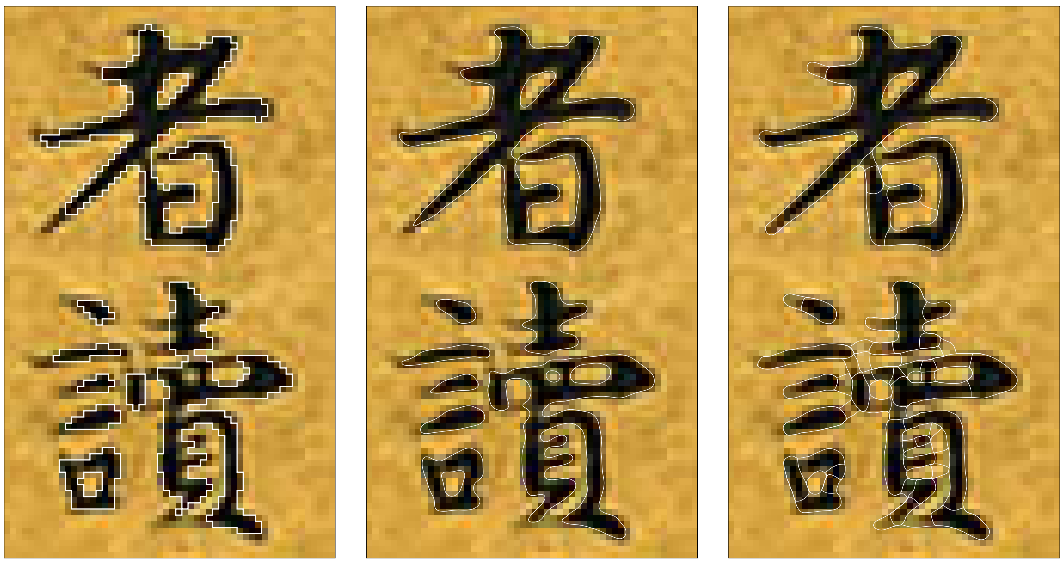


Figure 7.46: Segmentation results on two characters from the fragment shown in figure 7.45 (computed on the red channel of the original color image, and using two-fold oversampling in case of the subpixel-accurate detectors). Left: Pixel-accurate thresholding (threshold $t = 13$). Center: Subpixel-accurate thresholding (threshold $t = 13$). Right: Subpixel watersheds of the gradient magnitude ($\sigma_{\text{filter}} = 0.8$, gradient threshold $g_0^2 = 1.4$). Algorithm parameters apply to the noise normalized red channel.

Since many strokes in the character images are quite thin, we want to use small filters with $\sigma_{\text{filter}} = 0.8 \ldots 1.0$. Equation (7.7) tells us that these filter scales require signal-to-noise ratios above 5 to 7 to be applicable. Since the actual SNR is 22, this condition is clearly fulfilled. However, the effective SNR of narrow bar patterns is lower than that of step edges (cf. section 7.3.2), so this number has to be adjusted. The diagram in figure 7.20 right tells us that the effective height of a bar with equal steps on its two sides is reduced to 60% when $\sigma/x_s = 2$, where σ is the combined scale of PSF and filter (i.e. $\sigma = 1.0 \ldots 1.17$ in the present case), and x_s is the half-width of the bar. Thus, the required effective SNR is still achieved for stroke widths of only 1 to 1.2 pixels. Assuming an effective signal-to-noise ratio of SNR $= 10$, and $\sigma = 1.0 \ldots 1.17$, the optimal threshold on the gradient squared magnitude in the noise-normalized images should be chosen as $g_0^2 \approx 1.4$ (figure 7.46 right). It can be seen that this threshold leads to some oversegmentation. However, the oversegmentation is not a consequence of noise, but of the complicated geometry (similar to the ladder effect described in section 7.3.2).

We can get rid of the oversegmentation when we base our decision whether or not an edge is significant on more sophisticated criteria than simple gradient thresholds. New criteria would be especially useful when they were based on information beyond what has already been utilized in the algorithm. Recall that the gradient magnitude is a purely local measure of boundary strength. Therefore, more sophisticated criteria should make use of non-local properties of the segmentation, such as region statistics and boundary continuity. The computation of such properties has become possible because the regions of

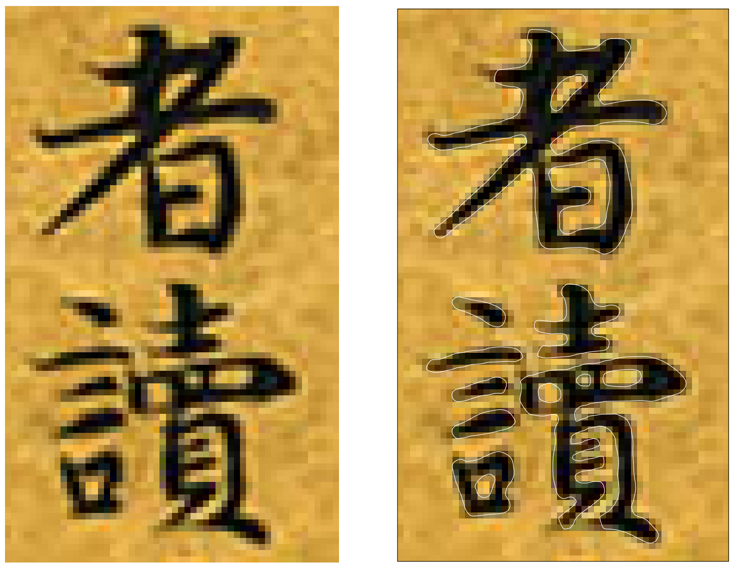


Figure 7.47: Left: Original image. Right: Improvement of the watershed segmentation by means of region averages (details see text).

the oversegmentation can be interpreted as *superpixels,* cf. [Ren & Malik 03]. Superpixels are "adaptive pixels" that reflect the structure of the data rather than the structure of the imaging device. Therefore, they carry much more information than the pixels of the grid. We investigated superpixel-based criteria for reduction of oversegmentation in two diploma theses [Boetius 06, Kaynig 06]. The details of these methods are beyond the scope of the present work, but the application to the Chinese character images is quite simple. Here, we compute the average gray-level of each superpixel, and keep only those edges that separate a bright region from a dark one. The threshold $t = 13$ is the same as the one used in the direct thresholding segmentation, but figure 7.47 shows that the result is now superior to both the original watershed segmentation and the direct threshold segmentation.

8 Tangent Direction Estimation

Abstract

The tangent orientation is an important and useful shape feature. Several more or less sophisticated tangent estimation methods have been proposed in the literature, but existing method evaluation studies have only had limited scope (some were limited to tangents derived from pixel-accurate boundaries, e.g. [Klette & Rosenfeld 04, Vialard 96], others have only considered a single algorithm, e.g. [Kovalevsky 01b]). In this chapter, we show how the results of the boundary error analysis from chapter 7 can be used to make theoretical predictions about a derived measurement such as the tangent angle. These predictions are well confirmed by subsequent experiments. Our analysis suggests that sophisticated tangent estimation methods are only superior to the (trivially computable) direction perpendicular to the image gradient, when long boundaries with slowly varying curvature have to be analyzed in noisy images.

8.1 Introduction

The local tangent direction is among the most important characteristics of a boundary. At least four method classes are commonly used for tangent estimation:

1. One can derive the tangent angle directly from a local direction provided by the boundary indicator function, e.g. from the direction perpendicular to the gradient.
2. When the boundary is represented by a polygonal arc, the tangent at a knot of the polygon can be approximated by the average direction of the adjacent line segments.
3. When the boundary is represented as a function $\vec{x}(t)$ of arc length t, one can apply suitable derivative filters to estimate the tangent direction.
4. One can fit a parametric model to a piece of the boundary (e.g. a straight edge or a circle) and determine the tangent direction from the model parameters.

The first possibility is only applicable when the boundary indicator actually provides a local preference direction. This is not always the case. A purely scalar boundary strength is, for example, computed by the SUSAN operator [Smith & Brady 97]. Moreover, even if a single preference direction is available, it cannot account for the tangent of all adjacent boundaries near a junction.

The second method is unsuitable for grid-based boundary representations because the polygons derived from grid-based coordinates have only a few different orientations (e.g. two for interpixel edges). The resulting tangent estimates would thus be extremely inaccurate and useless. Methods 3 and 4 require sufficiently long pieces of boundary to be applicable. This is problematic when an edge is short or when the tangent is to be determined near an edge's end point. Problems also arise with these methods when the curvature of the boundary changes within the neighborhood considered, because this leads to biased orientation estimates.

We already determined the accuracy of the gradient direction (method 1) in section 7.2.2.3, so we will use it as a reference and concentrate on the other possibilities in this chapter.

8.2 Direct Tangent Estimation from Polygon Segment Directions

If applicable, determining the tangent direction at a knot as the average of the two adjacent edge segments is the simplest method (method 2 in the list). In the limit of zero knot distance along the polygon, this definition is equivalent to computing tangents by means of derivatives

$$\phi_{\text{tangent}} = \arctan\left(\frac{\frac{\partial C}{\partial y}}{\frac{\partial C}{\partial x}}\right) \tag{8.1}$$

where C is the boundary of interest. Obviously, this approach will not yield good results for pixel-accurate boundary representations: Since a 4-connected boundary has only two different segment orientations, and an 8-connected one has four, only 4 respectively 8 different tangent orientations are possible. This is too inaccurate for virtually any conceivable application. In contrast, the polygons resulting from subpixel-accurate edge detectors are quite accurate, so this simple method may yield reasonable results. In fact, if the image were not corrupted by noise, the polygon tangent would approach the true tangent as the knot distance along the polygon approached zero. Since the subpixel watershed algorithm 5.4 and the subpixel zero-crossing algorithm 5.3 sample the boundary quite densely (with typical vertex distances in the order of 1/10 of a pixel), the tangent accuracy in low-noise images can be expected to be quite good.

We can quantify the expected error of the polygon tangent by looking at the limiting case, the true boundary derivative (8.1). For small angles, the arc-tangent can be replaced with its argument[1]. Without loss of generality, we assume that the true tangent direction is parallel to the x-axis, i.e. we can represent the curve as a function $y = f(x)$, and $\phi_{\text{tangent}} = df/dx = 0$ would hold at the point of interest when there were no noise. The standard deviation of the tangent orientation is therefore approximately equal to the standard deviation of the derivative df/dx along the boundary. This standard deviation can be computed by means of the power spectrum (7.17) of the localization error along the boundary. Multiplying the power spectrum with the power spectrum $(2\pi\nu)^2$ of the

[1]The error of this linearization is below 10% for angles up to 34°. Since we are using the approximation only for the estimation of the *angle error*, 10% accuracy are certainly sufficient.

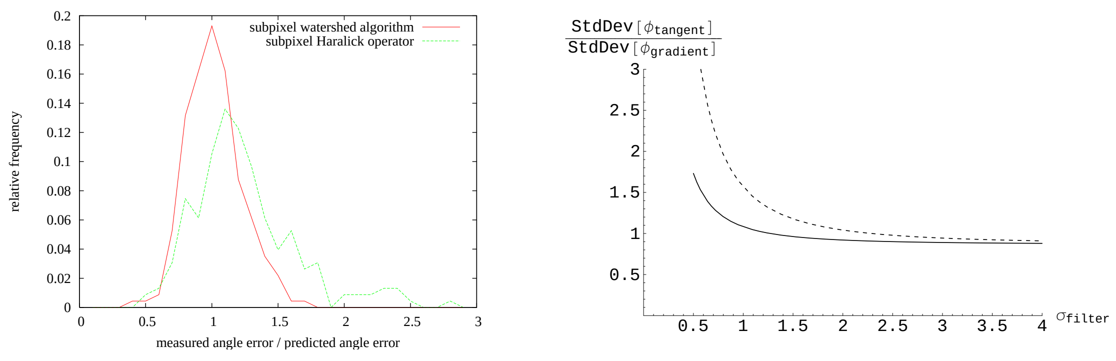


Figure 8.1: Left: Histogram of the ratio between the measured and predicted tangent angle error for the subpixel watershed and Haralick algorithms. Measurements were taken over 228 images with edge orientations between 0° and 90°, SNR between 5 and 20, $\sigma_{\text{PSF}} \in \{0.5, 0.9\}$ and $\sigma_{\text{filter}} \in \{1.0, 2.0\}$. Absolute errors ranged from 1° to 15°. Right: Ratio between the expected errors of the polygon-based tangent estimator and the gradient orientation for small σ_{filter} with $\sigma_{\text{PSF}} = 0.5$ (solid) and $\sigma_{\text{PSF}} = 0.9$ (dashed). The ratio approaches $\sqrt{3}/2 \approx 0.866$ as $\sigma_{\text{filter}} \to \infty$.

derivative operator and integrating over all frequencies gives the variance of the angle:

$$\text{Var}[\phi_{\text{tangent}}] = 2\sqrt{\pi}\sigma_{\text{filter}}\epsilon^2 \int_{-\infty}^{\infty} (2\pi\nu)^2 \, e^{-4\pi^2\nu^2\sigma^2_{\text{filter}}} \, d\nu$$

Note that this is a 1-dimensional integral, because we integrate along a 1-dimensional curve. The standard deviation is just the square root of the variance:

$$\begin{aligned} \text{StdDev}[\phi_{\text{tangent}}] &= \sqrt{\text{Var}[\phi_{\text{tangent}}]} = \frac{\epsilon}{\sqrt{2}\sigma_{\text{filter}}} \\ &= \frac{\sqrt{3}}{4}\frac{N}{S}\frac{\left(\sigma^2_{\text{PSF}} + \sigma^2_{\text{filter}}\right)^{3/2}}{\sigma^4_{\text{filter}}} \end{aligned} \tag{8.2}$$

where $\epsilon = \sqrt{\frac{3}{8}}\frac{N}{S}\left(1 + \left(\frac{\sigma_{\text{PSF}}}{\sigma_{\text{filter}}}\right)^2\right)^{3/2}$ is the standard deviation of the localization error along the boundary according to (7.9), and σ_{filter} is the scale of the Gaussian gradient operator used to detect the edge. Figure 8.1 left shows a comparison of actually measured errors with the errors predicted by this formula, and we see generally quite good agreement. The error of the tangent obtained from the polygon may be compared with the expected error of the gradient direction according to (7.13), i.e. method 1 in the list:

$$\text{StdDev}[\phi_{\text{gradient}}] = \frac{N}{S}\frac{\sqrt{\sigma^2_{\text{PSF}} + \sigma^2_{\text{filter}}}}{2\sigma^2_{\text{filter}}}$$

In the limit $\sigma_{\text{filter}} \to \infty$, the ratio $\frac{\text{StdDev}[\phi_{\text{tangent}}]}{\text{StdDev}[\phi_{\text{gradinet}}]}$ is independent of σ_{PSF} and approaches $\frac{\sqrt{3}}{2} \approx 0.866$, i.e. the polygon-based tangent estimator is better in the limit. Figure 8.1 right depicts the ratio between the two errors for small values of σ_{filter}, where the gradient orientation is superior. For straight edges, the two methods become equally accurate at $\sigma_{\text{filter}} = \sqrt{3+2\sqrt{3}}\,\sigma_{\text{PSF}} \approx 2.5\,\sigma_{\text{PSF}}$.

8.3 Filter-Based Tangent Estimation

Instead of using the polygon segments directly for tangent estimation, we may also employ a more complicated method in order to achieve higher accuracy. This approach is especially popular for pixel-accurate boundaries, where direct tangent estimation according to the previous section is too inaccurate. In filter-based tangent estimation, the derivatives $\partial C/\partial x$ and $\partial C/\partial y$ are computed by means of suitable derivative filters. Let the curve be represented as $C = \vec{p}(t) = (x(t), y(t))^T$, where t is an arc length parameter. Then, the derivatives are determined by convolution of $\vec{p}$ with the derivative filter $g'_{\sigma_C}(t)$:

$$\begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} = \vec{p}(t) \star g'_{\sigma_C}(t)$$

where σ_C is a measure of the filter size. Two filters are in common use: the symmetric difference

$$g'_{\sigma_C}(t) = \frac{\delta(t-\sigma_C) - \delta(t+\sigma_C)}{2\sigma_C}$$

and the Gaussian derivative

$$g'_{\sigma_C}(t) = \frac{-t}{\sqrt{2\pi}\sigma_C^3} e^{-\frac{t^2}{2\sigma_C^2}}$$

The former approximates the tangent at point $\vec{p}(t)$ by the chord between the points $\vec{p}(t-\sigma_C)$ and $\vec{p}(t+\sigma_C)$, while the latter computes weighted centroids of the polygon points to the left and right of $\vec{p}(t)$ and uses the direction between these centroids as tangent direction. As σ_C increases, the estimated tangent direction is less and less influenced by noise. The expected error can again be computed by integrating over the product of the power spectra of the noise and the filter. We get the following expressions

$$\text{StdDev}[\phi_{\text{sym-diff}}] = \text{StdDev}[\phi_{\text{tangent}}]\,\frac{\sigma_{\text{filter}}}{\sigma_C}\sqrt{1 - e^{-\frac{\sigma_C^2}{\sigma_{\text{filter}}^2}}} \tag{8.3}$$

and

$$\text{StdDev}[\phi_{\text{Gd}}] = \text{StdDev}[\phi_{\text{tangent}}]\left(1 + \frac{\sigma_C^2}{\sigma_{\text{filter}}^2}\right)^{-3/4} \tag{8.4}$$

where $\text{StdDev}[\phi_{\text{tangent}}]$ is the expected error of the tangent without filtering according to (8.2), and σ_{filter} is the scale of the filter that was used to compute the underlying boundary indicator. As expected, the error with filtering approaches the error without filtering when $\sigma_C \to 0$. For large σ_C, the error decreases asymptotically as σ_C^{-1} and $\sigma_C^{-3/2}$ respectively. However, while the Gaussian derivative filter is superior in the limit

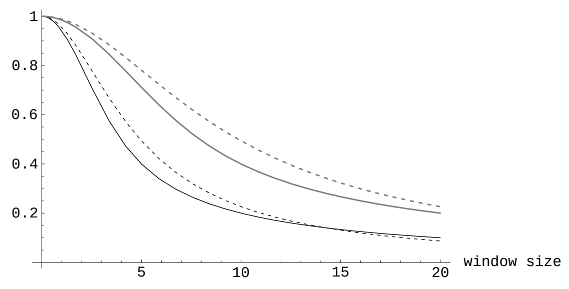


Figure 8.2: Reduction in the tangent angle standard deviation when the angle is determined by a filter with the given window size: symmetric difference (solid) and Gaussian derivative (dashed) at $\sigma_{\text{filter}} = 1$ (black) and $\sigma_{\text{filter}} = 2$ (gray).

of infinitely large filter size, this is not necessarily the case at practically relevant values of σ_C, where only a finite piece of the boundary is available.

Each filter requires a certain minimal length of boundary to be applicable: The symmetric difference filter is only applicable when the boundary continues for at least σ_C on either side of the point of interest. That is, we need a "window" of length $l_{\text{sym-diff}} = 2\sigma_C$ along the polygon. The Gaussian derivative filter is infinitely large, so we must clip it at a finite window size. In order to avoid systematic errors, we should at least extend the window to $2\sigma_C$ on either side of the point of interest, so the total required window size is $l_{\text{Gd}} = 4\sigma_C$.

Figure 8.2 shows the error reduction relative to the unfiltered tangent as a function of the required window size l. We observe two interesting facts:

1. The window must be quite large ($\approx 4\sigma_{\text{filter}}$) before the reduction of the error becomes noticeable, and

2. when we use identical windows for the symmetric difference and Gaussian derivatives, the Gaussian filter is not superior until the window size becomes quite large ($l \approx 14$ for $\sigma_{\text{filter}} = 1$).

Contrary to our intuition, Gaussian filters are not superior to symmetric differences in this application unless the shapes we are interested in are very large.

Irrespective of the filter, one would like to choose σ_C as large as possible in order to minimize the error due to noise. However, the estimated tangent will be biased when the polygon is asymmetric around the point $\vec{p}$, see figure 8.3. The bias due to curve asymmetry increases with increasing filter size, so one wants to choose small σ_C. Consequently, there is a trade-off in the selection of σ_C that optimally balances noise errors against bias. [Kovalevsky 01b] proposes a method for estimating the optimal σ_C for the symmetric difference filter, and we generalize his method to arbitrary filters. To simplify the derivation, we rotate the coordinate system so that $\vec{p}(t)$ is at the origin and the x-axis is parallel to the tangent at $\vec{p}(t)$. As long as $|x|$ is not too large, the polygonal curve can be

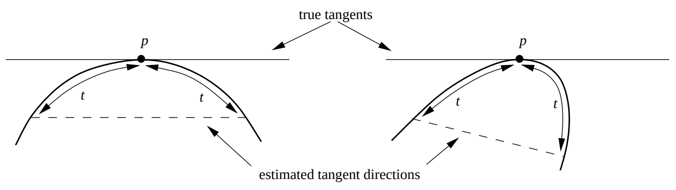


Figure 8.3: Tangent estimation by symmetric differences. When the curvature of the boundary is symmetric around the point $\vec{p}$, the estimated tangent direction is unbiased (left), otherwise it is biased (right). This applies likewise to tangent estimation with Gaussian filters.

approximated in the new coordinate system by a function $y(x) = q(x) + n(x)$ where $q(x)$ is the true curve and $n(x)$ is additive localization noise with power spectrum according to (7.17). We expand $q(x)$ into a Taylor series around $x = 0$:

$$y(x) \approx q(0) + q'(0)\,x + \frac{q''(0)}{2}\,x^2 + \frac{q'''(x)}{6}\,x^3 + n(x)$$

The constant and linear terms on the right hand side vanish by definition of the local coordinate system. The convolution of the Taylor series with a derivative filter thus gives

$$y \star g'_\sigma\big|_{x=0} \approx \frac{q''(0)}{2}\,(x^2 \star g'_{\sigma_C}) + \frac{q'''(x)}{6}\,(x^3 \star g'_{\sigma_C}) + (n \star g'_{\sigma_C})$$

where all convolutions are evaluated at $x = 0$. The convolution of the even function x^2 with a derivative filter is zero because derivatives are odd functions. Hence, the term $x^2 \star g'_{\sigma_C}$ vanishes as well. Therefore, we have

$$y \star g'_{\sigma_C}\big|_{x=0} \approx \frac{q'''(x)}{6}\,(x^3 \star g'_{\sigma_C}) + (n \star g'_{\sigma_C})$$

Since we want to approximate $q'(0) = 0$ as well as possible, the filter response should be close to zero. That is, we want to minimize the square of the right hand side

$$\sigma_{C\text{-optimal}} = \arg\min_{\sigma_C} \left(\frac{q'''(x)}{6}\,(x^3 \star g'_{\sigma_C}) + (n \star g'_{\sigma_C})\right)^2$$

The expression to be minimized is a non-central χ^2-distributed random variable with mean $\mu = \left(\frac{q'''(x)}{6}\,(x^3 \star g'_{\sigma_C})\right)^2 + \mathrm{Var}\left[n \star g'_{\sigma_C}\right]$, and σ_C should be chosen so that μ is minimized. We already derived expressions for the variance (actually its square root, the standard deviation) in equations (8.3) and (8.4). The systematic error is easy to compute as well. In case of the symmetric difference filter, we get

$$\left(\frac{q'''(x)}{6}\,(x^3 \star g'_{\sigma_C})\right)^2 = \left(\frac{q'''(x)\,\sigma_C^2}{6}\right)^2$$

and for a Gaussian derivative

$$\left(\frac{q'''(x)}{6}\,(x^3 \star g'_{\sigma_C})\right)^2 = \left(\frac{q'''(x)\,\sigma_C^2}{2}\right)^2$$

Inserting these expressions into μ and setting the derivative with respect to σ_C to zero, we get a closed form solution for the optimal width $\sigma_{\text{sym-diff}}$ of the symmetric difference filter

$$\sigma_{\text{sym-diff}} = \left(\frac{3\epsilon}{|q'''|}\right)^{1/3} \tag{8.5}$$

where ϵ is again the standard deviation of the localization noise according to (7.9). In contrast, there is no closed form solution for the optimal scale σ_{Gd} of the Gaussian derivative. The optimal σ_{Gd}^2 is a root of the polynomial $4\,|q'''|^4\, s^2(\sigma_{\text{filter}}^2 + s)^5 - 9\epsilon^4\sigma_{\text{filter}}^2 = 0$. In order to obtain an approximate closed form expression for the root we simplify the polynomial by noting that $\sigma_{\text{Gd}} \gg \sigma_{\text{filter}}$ in all practically relevant cases. Then $s \approx s + \sigma_{\text{filter}}^2$, and thus

$$\sigma_{\text{Gd}} \approx \sqrt{\left(\frac{9\epsilon^4\sigma_{\text{filter}}^2}{4\,|q'''|^4}\right)^{1/7} - \sigma_{\text{filter}}^2} \tag{8.6}$$

The expression under the square root is positive when $\frac{\epsilon}{|q'''|} > \frac{2^{1/4}4}{3^{1/2}}\,\sigma_{\text{filter}}^3$. Otherwise, the condition $\sigma_{\text{Gd}} \gg \sigma_{\text{filter}}$ is violated, and Gaussian filtering wouldn't make sense anyway. It turns out that

$$\sigma_{\text{sym-diff}} \approx 2\sigma_{\text{Gd}}$$

so that the required window sizes $l_{\text{sym-diff}} = 2\sigma_{\text{sym-diff}}$ and $l_{\text{Gd}} = 4\sigma_{\text{Gd}}$ are indeed approximately equal, i.e. both filters use the same piece of the polygon when their sizes are optimally chosen. When optimal intervals are used, the expected error in tangent angle (in radians) for the two filter types is

$$\begin{aligned}
\text{StdDev}[\phi_{\text{sym-diff}}] &= \frac{1}{2}\left(\sqrt{3}\epsilon^2 q'''\right)^{1/3} \\
\text{StdDev}[\phi_{\text{Gd}}] &= \frac{1}{2}\left(\left(\frac{32}{\sqrt{27}}\epsilon^4\,|q'''|^3\,\sigma_{\text{filter}}^2\right)^{1/7} + |q'''|\left(\sigma_{\text{filter}}^2 - \left(\frac{9\epsilon^4\sigma_{\text{filter}}^2}{4\,|q'''|^4}\right)^{1/7}\right)\right)
\end{aligned}$$

(note that ϵ is also a function of σ_{filter}).

A natural question arising from these derivations is whether it is possible to compute the optimal interval in practice. After all, the *third* derivative q''' of the function is needed in order to obtain an optimal estimate of the *first* derivative q'. [Kovalevsky 01b] notes that the expressions for optimal $\sigma_{\text{sym-diff}}$ and σ_{Gd} represent minima and depend on $|q'''|$ only as very low powers, so that rather coarse estimates of $|q'''|$ should be sufficient for reasonable $\sigma_{\text{sym-diff}}$ and σ_{Gd}. We have found experimentally, that a sufficiently accurate $|q'''|$ can be computed by means of a finite difference filter according to

$$|q'''(t)| = \left\|\frac{-\vec{p}(t-2d) + 2\vec{p}(t-d) - 2\vec{p}(t+d) + \vec{p}(t+2d)}{2d^3}\right\| \tag{8.7}$$

where t is arc-length, and $d \geq 8\sigma_{\text{filter}}$ (this bound corresponds to the average wavelength of the localization error along the polygon according to (7.16)). In other words, a window of

$$l = 32\sigma_{\text{filter}}$$

is required in order to compute a sufficiently accurate third derivative! This is more than will be available in many applications, so that filtering with optimal filter sizes will often be impossible.

Regardless of whether optimal filter sizes are used or not, we must deal with another difficulty: The knots of the polygons resulting from our edge detectors are not in general equidistant, whereas discrete convolution requires equidistant points. There are three possibilities to solve this problem:

1. One can resample the polygon at (approximately) equidistant intervals by using linearly interpolated points. In case of a pixel-accurate boundary polygon, an improved variant of this approach was proposed by [Vialard 96]. They first use the digital straight line algorithm 4.7 to transform the original (pixel-based) polygon into a polygon with longer (subpixel-accurate) segments. The new polygon is then resampled by linear interpolation. This works better than direct resampling of pixel-accurate coordinates.
2. One can compute an equidistant polygon from the original one by means of a least squares fit [Eberly 03].
3. The original knots can be used without resampling when the convolution is implemented as a *normalized convolution* [Knutsson & Westin 93].

Only the first two methods make sense for a symmetric difference filter, whereas all three may be applied in connection with Gaussian derivative filters. Normalized convolution of a polygon with a smoothing filter is defined as

$$\vec{p} \star g_\sigma = \frac{\sum_i \vec{p}_i g_\sigma(t - t_i)}{\sum_i g_\sigma(t - t_i)}$$

where t_i is the arc-length parameter of point $\vec{p}_i$. The numerator is a discrete convolution with non-equidistant points, and the denominator adjusts the normalization of the numerator for non-uniform sampling. When normalized convolution is to be applied with a derivative filter, the right hand side must be differentiated according to the quotient rule

$$\vec{p} \star g'_\sigma = \frac{\left(\sum_i \vec{p}_i g'_\sigma(t - t_i)\right)\left(\sum_i g_\sigma(t - t_i)\right) - \left(\sum_i \vec{p}_i g_\sigma(t - t_i)\right)\left(\sum_i g'_\sigma(t - t_i)\right)}{\left(\sum_i g_\sigma(t - t_i)\right)^2}$$

In our experiments, we employ least-squares polygon resampling (method 2). We use reflective boundary conditions for open polygons and cyclic boundary conditions for closed ones.

8.4 Model-Based Tangent Estimation

Alternatively, the tangent can be determined by fitting a geometric model to a piece of the polygon, and then estimating the angle from the model. A least squares fit of a model to the point sequence $\vec{p}_{i-k}, ..., \vec{p}_{i+l} = \vec{p}(t_{i-k}), ..., \vec{p}(t_{i+l})$ around the point of interest

$\vec{p}_i = \vec{p}(t_i)$ minimizes the sum of squared distances d_j^2 between the given points and the model, the so-called *geometric residual*

$$\mathcal{R} = \min_m \sum_j d_j^2 = \min_m \sum_j \|\vec{p}_j - m(\vec{p}_j)\|^2$$

where the minimization extends over some set of feasible models (i.e. over the parameter space of a specific model class), and $m(\vec{p}_j)$ is the point on the model curve m which is closest to $\vec{p}_j$. To solve the fitting problem, it is useful to translate the points into a coordinate system whose origin coincides with the centroid of the given points

$$\vec{p}_j{}' = \vec{p}_j - \vec{p}_S \text{ with } \vec{p}_S = \frac{1}{n}\sum_j \vec{p}_j$$

This not only simplifies many equations, but also improves numerical stability and accuracy because errors due to finite number representations are significantly reduced. On the other hand, centering of the points reduces algorithm speed because one can no longer compute geometric moments of the point set incrementally[2].

The simplest possible model is a straight line, where $d_j = A\,x_j' + B\,y_j' + C$ with constraint $A^2 + B^2 = 1$. It is well known that the optimal solution to this problem is determined by the eigenvector corresponding to the large eigenvalue of the scatter matrix

$$\mathbf{T} = \begin{pmatrix} t_{xx} & t_{xy} \\ t_{xy} & t_{yy} \end{pmatrix} = \begin{pmatrix} \frac{\sum x_j'^2}{n} & \frac{\sum x_j' y_j'}{n} \\ \frac{\sum x_j' y_j'}{n} & \frac{\sum y_j'^2}{n} \end{pmatrix}$$

where $n = l + k + 1$ is the number of points in the interval, and $\vec{p}_j{}' = (x_j', y_j')^T$ are the centered point coordinates. The angle of the optimal line, and therefore the estimated tangent orientation, is

$$\phi = \frac{1}{2}\arctan\frac{2t_{xy}}{t_{xx} - t_{yy}}$$

The optimal interval size can be chosen according to a χ^2-test of the residual: k and l should be increased as long as the residual $\mathcal{R}$ does not exceed a given confidence level for a χ^2-distribution with $n-2$ degrees of freedom and variance ϵ^2, where ϵ is the expected localization error of the edge detector used.

Natural contours can be better approximated when not only straight lines, but also circular arcs are permitted as candidate models. Under this more general model, the minimum distance between a point and the model can be computed as

$$d_j = \frac{2P_j}{1 + \sqrt{1 + A\,P_j}}$$

[2]In principle, it is possible to compute centered moments incrementally (recall, for example, the well-known formula $\frac{1}{n}\sum_j x_j'^2 = \frac{1}{n}\sum_j x_j^2 - \left(\frac{1}{n}\sum_j x\right)^2$). But for higher order moments, e.g. $\frac{1}{n}\sum_j \left(x_j'^2 + y_j'^2\right)^2$ as needed for circle fitting, these formulas are numerically very problematic and therefore unsuitable for practical computations.

(cf. [Chernov & Lesort 05]) with

$$P_j = A\left(x_j'^2 + y_j'^2\right) + B\,x_j' + C\,y_j' + D$$

subject to the constraint

$$B^2 + C^2 - 4\,A\,D = 1$$

It can be seen that these definitions reduce to the straight line case when $A = 0$. If $A \neq 0$, the circle radius is given by $r = \left|\frac{1}{2A}\right|$, and the coordinates of the circle's center are $x_C' = -\frac{B}{2A}$ and $y_C' = -\frac{C}{2A}$. Unfortunately, the model instance minimizing the geometric residual $\mathcal{R}$ can no longer be determined analytically, and an iterative minimization method is required. However, this is too expensive in our context, because we have to perform the fit thousands or even millions of times per image. A popular alternative is to replace the geometric residual equation with an *algebraic* one

$$\mathcal{R}_a = \min_m \sum_j P_j^2$$

with constraint $A = 1$. This minimization problem is easy to solve because it leads to a linear system in the parameters B, C, and D, but the resulting solutions are not always close to the optimal geometric fits one is actually interested in. Therefore, [Pratt 87] proposed to minimize a *gradient-weighted algebraic residual* defined as

$$\mathcal{R}_g = \min_m \sum_j \frac{P_j^2}{B^2 + C^2 - 4\,A\,D}$$

and demonstrated the superiority of this approach over the naive algebraic fit. Subsequently, [Chernov & Lesort 05] observed that minimizing $\mathcal{R}_g$ is equivalent to minimizing the algebraic residual $\mathcal{R}_a$, subject to the constraint $B^2 + C^2 - 4A\,D = 1$ (instead of $A = 1$). This allowed them to introduce a very efficient new algorithm for finding an optimal circle. Consider the moment matrix

$$\mathbf{T} = \begin{pmatrix} t_{zz} & t_{xz} & t_{yz} & t_z \\ t_{xz} & t_{xx} & t_{xy} & 0 \\ t_{yz} & t_{xy} & t_{yy} & 0 \\ t_z & 0 & 0 & 1 \end{pmatrix}$$

where

$$t_{xx} = \tfrac{1}{n}\textstyle\sum_j x_j'^2, \qquad t_{yy} = \tfrac{1}{n}\sum_j y_j'^2, \qquad t_{zz} = \tfrac{1}{n}\sum_j z_j'^4$$

$$t_{xz} = \tfrac{1}{n}\textstyle\sum_j x_j' z_j'^2, \qquad t_{yz} = \tfrac{1}{n}\sum_j y_j' z_j'^2, \qquad t_z = \tfrac{1}{n}\sum_j z_j'^2 = t_{xx} + t_{yy}$$

and $z_j'^2 = x_j'^2 + y_j'^2$. The matrix entries for t_x and t_y are zero because the centroid of the given points lies in the origin of the primed coordinate system. This is crucial for the following algorithm to work. The algebraic residual can now be written as $\mathcal{R}_a = \mathbf{a}^T\,\mathbf{T}\,\mathbf{a}$ with $\mathbf{a} = (A, B, C, D)^T$. Likewise, the constraint is written as $\mathbf{a}^T\,\mathbf{C}\,\mathbf{a} = 1$ with

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{pmatrix}$$

The optimal solution $\mathbf{a}$ can be found as the eigenvector corresponding to the smallest non-negative eigenvalue of the generalized eigenvalue problem

$$\mathbf{T}\,\mathbf{a} = \lambda\,\mathbf{C}\,\mathbf{a}$$

The key of the algorithm of [Chernov & Lesort 05] is the observation that the characteristic polynomial of this eigenvalue problem has a particularly simple form

$$\det\left(\mathbf{T} - \lambda\,\mathbf{C}\right) = 4\lambda^4 + c_2\lambda^2 + c_1\lambda + c_0$$

with

$$\begin{aligned} c_0 &= t_{xz}^2 t_{yy} + t_{yz}^2 t_{xx} - 2t_{xy}t_{xz}t_{yz} + \left(t_z^2 - t_{zz}\right)\left(t_{xx}t_{yy} - t_{xy}^2\right) \\ c_1 &= t_z\left(t_{zz} - t_z^2 + 4t_{xx}t_{yy} - 4t_{xy}^2\right) - t_{xz}^2 - t_{yz}^2 \\ c_2 &= 4t_{xx}t_{yy} - 4t_{xy}^2 - 3t_z^2 - t_{zz} \end{aligned}$$

Consequently, it is not necessary to solve the eigenvalue problem by means of expensive matrix methods. Instead, the desired root λ_o of the characteristic polynomial can be determined by Newton's algorithm, which is guaranteed to converge quickly to the correct solution when iterations are started at an initial guess of $\lambda = 0$. Finally, the center of the optimal circle (in the coordinate system whose origin coincides with the centroid of the given points) is

$$\begin{aligned} x'_C &= \frac{t_{xz}\left(t_{yy} - \lambda_o\right) - t_{xy}t_{yz}}{2\left(\lambda_o^2 - \lambda_o t_z + t_{xx}t_{yy} - t_{xy}^2\right)} \\ y'_C &= \frac{t_{yz}\left(t_{xx} - \lambda_o\right) - t_{xy}t_{xz}}{2\left(\lambda_o^2 - \lambda_o t_z + t_{xx}t_{yy} - t_{xy}^2\right)} \end{aligned}$$

and its radius is

$$r = \sqrt{x_C'^2 + y_C'^2 + 2\lambda_o + t_z}$$

From the coordinates of the circle center, the desired tangent angle at point $\vec{p_i}'$ is defined as

$$\phi = \arctan\frac{x'_C - x'_i}{y'_i - y'_C}$$

Other models beyond lines and circles (e.g. ellipses and parabolas) are also possible, but we found their performance to be inferior to the circle fitting method and didn't consider them further.

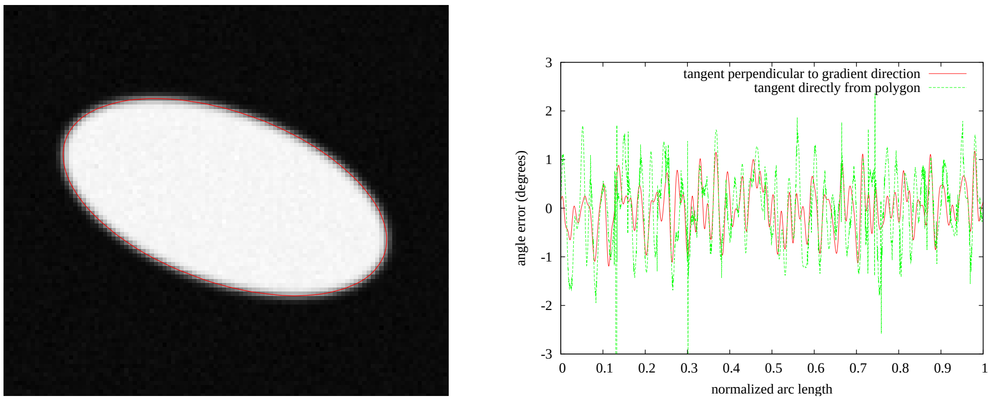


Figure 8.4: Left: Original image with overlayed subpixel watershed boundary. Ellipse radii are $r_1 = 40$, $r_2 = 20$ (i.e. perimeter is ≈ 194 pixels, minimum curvature radius 10 pixels), SNR = 100, and $\sigma_{\text{PSF}} = \sigma_{\text{filter}} = 1$. Right: Angle errors for tangents taken perpendicular to the gradient direction, and by the average of two consecutive polygon segments at every knot of the polygon.

8.5 Experimental Comparison

We performed a large number of experiments on generated noisy ellipses (with $\sigma_{\text{PSF}} = 1$) to validate and compare the performance of the tangent estimation algorithms outlined above. Unless otherwise noted, edge detection was performed with the subpixel watershed algorithm at scale $\sigma_{\text{filter}} = 1$. Figure 8.4 left shows the original image (with SNR = 100) and the detected contour, which looks perfect in an image of such high quality. In all diagrams, we plot the error of the tangent direction relative to the ground truth tangent angle for all detected points around the ellipse. Figure 8.4 right shows these measurements for the gradient-based tangent (which will act as a reference in all experiments) and the tangent estimated directly from the direction of the polygon segments. Theory predicts that the error of the polygonal tangents should exceed the one of the gradient by a factor of $\sqrt{3} \approx 1.7$ at the algorithms parameters chosen. Actual error standard deviations are listed in table 8.6. It can be seen that the tangent directions are measured quite accurately – the angle error of the gradient is below 0.5°. The errors of the polygon tangent are somewhat higher than expected. In figure 8.4 right, we can see that this is caused by outlier points, which deviate slightly (invisible to the naked eye) from the true contour, causing exceptionally high angle errors (up to about 4°) at a few points. These outliers are a numerical problem in the subpixel watershed algorithm that is caused by the image function being very close to degenerate (e.g. close to not being a Morse function) due to large homogeneous regions and low noise. Natural images do not exhibit this problem.

Figure 8.5 reports results for filter-based and model based tangent estimation within small windows along the boundary. All algorithms compute the tangent angle by using points from the same interval with radius 2 pixels to either side of the current contour point. The resulting errors are remarkably similar for all these methods, and we don't get significantly better results than with the much simpler gradient method. The error

	SNR = 100		SNR = 10	
method	bias (degrees)	std. dev. (degrees)	bias (degrees)	std. dev. (degrees)
gradient	-0.0002	**0.48** [0.41]	0.006	4.2 [4.1]
simple tangent	-0.01	0.94 [0.71]	-0.1	7.0 [7.1]
Gauss ($\sigma_C = 1$)	-0.0003	0.49 [0.42]	-0.0001	4.3 [4.2]
Gauss ($\sigma_C = 2$)	-0.0003	0.50 [0.21]	0.0003	2.3 [2.1]
Gauss ($\sigma_C = 4$)	-0.0007	1.14 [0.08]	-0.002	**1.6** [0.85]
Gauss (scale selection)	0.036	**0.42**	0.06	1.6
symmetric difference ($\sigma_C = 2$)	-0.0003	**0.45** [0.35]	0.0004	3.5 [3.5]
symmetric difference ($\sigma_C = 4$)	-0.0003	0.60 [0.18]	0.0004	1.8 [1.8]
symmetric difference ($\sigma_C = 8$)	-0.0008	1.60 [0.09]	-0.004	2.0 [0.9]
symmetric difference (scale selection)	-0.003	0.45	0.23	1.7
line fit (window radius = 2)	-0.0004	0.50	0.0005	4.3
line fit (window radius = 4)	-0.0004	0.48	0.001	2.3
line fit (window radius = 8)	-0.0003	1.05	-0.004	**1.5**
circle fit (window radius = 2)	-0.0003	0.50	-0.0003	4.3
circle fit (window radius = 4)	$-3 \cdot 10^{-5}$	0.48	-0.0007	2.3
circle fit (window radius = 8)	0.0002	1.10	0.003	**1.6**

Table 8.1: Bias and standard deviations of various algorithms for the image in figure 8.4 (center columns) and figure 8.9 (right columns). Theoretical predictions, if available, are given in brackets, best results are printed in bold face.

values in table 8.1 confirm that this behavior is also predicted by theory. In order to improve the accuracy, filters and model fit must be applied in larger windows. Figure 8.6 shows results for window radii 4 and 8. However, table 8.1 shows that the average errors are not reducing, in contrast to the predictions of theory. This is due to the fact that theoretical predictions apply to straight lines, whereas the ellipse is curved. The bias due to curved boundaries is clearly visible in figure 8.6: The (absolute) errors have marked maxima near the apexes of the ellipse (i.e. at arc lengths near 0 and near 0.5), whereas the errors along the low-curvature part of the ellipse (arc lengths around 0.25 and .75) are indeed significantly reduced relative to the gradient method. This bias is even more pronounced for more elongated ellipses – see figure 8.7.

In order to keep the bias at a minimum, we can apply the optimal scale selection method proposed by [Kovalevsky 01b], see equations (8.7) and (8.6). Results of filter-based tangent estimation with optimal scales for the two ellipses are shown in figure 8.8. Scale selection is successful in ensuring that the error does never increase with respect to the gradient direction, and that it is significantly *reduced* whenever the boundary is sufficiently symmetric, i.e. especially along its low-curvature part. Scale selection determines the optimal window size to have radius 2 near the apexes and radius 8 to 16 along the symmetric part of the boundary. Unfortunately, the third derivative according to equation (8.7), which is required to compute the optimal scale, must be determined on a

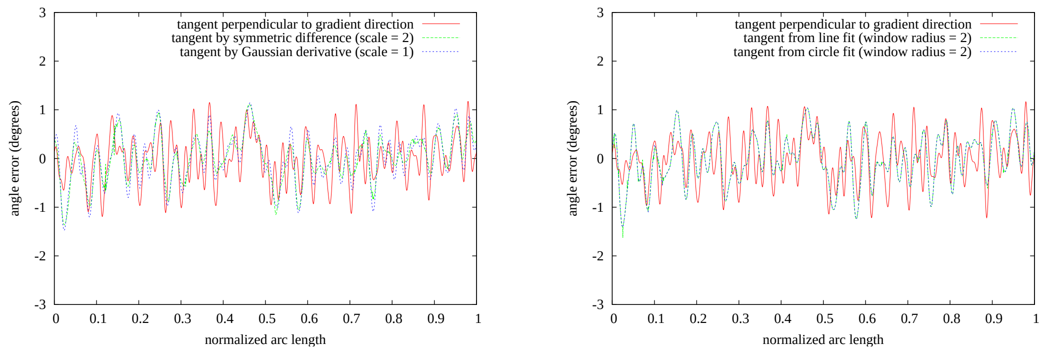


Figure 8.5: Angle errors of filtering and model fitting for the image from figure 8.4. The stated scale and radii mean that all algorithms use the same window (of length 4 pixels – recall that the Gaussian window has radius $2 \cdot$ scale). This window is too small for the results being superior to the gradient direction, due to error correlation along the polygon.

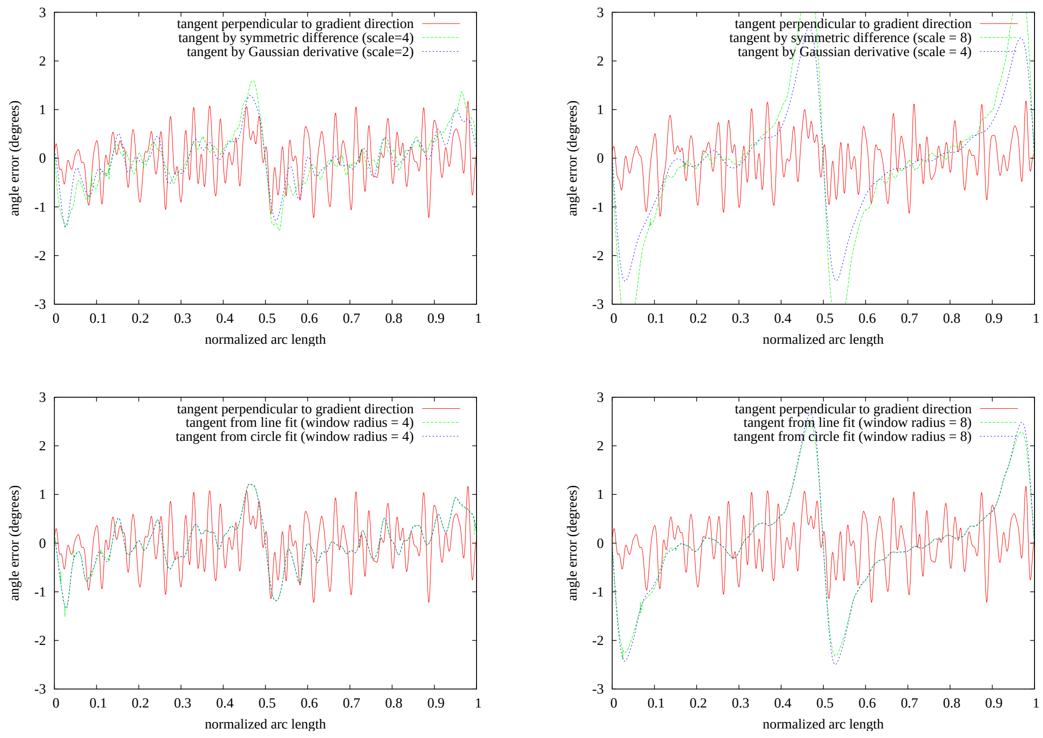


Figure 8.6: Same as figure 8.5, but for larger windows. Now the statistical errors are significantly reduced, but near the apexes of the ellipse we get strongly biased angle estimates due to curve asymmetry.

very large window (radius 16) in order to be sufficiently accurate for scale computation. Thus, the scale selection method can only be applied when the objects in the image are quite large. Moreover, this method cannot be applied near the ends of open polygons (see below).

In figure 8.9, we repeated the same experiment with a noisy ellipse. As table 8.1 (right columns) shows, the measured errors also conform to our theoretical predictions. In fact, the agreement is even better than in the low-noise case because the bias caused by the curved boundary is now much lower than the statistical error. This also means that the trade-off between bias and window size for filter and model-fit methods is now different. Here, window radii up to 8 pixels can be used without problem, because the bias introduced by the large window size is still below the statistical error without filtering. In fact, in noisy images it is preferable to just use larger filter sizes instead of optimal scale selection: Due to noise, the estimates of the third derivative (8.7) become too inaccurate. Consequently, the selected scales will be noisy, which translates into noisy estimates of the tangent angle.

Another kind of noisy boundary is encountered in the context of pixel-accurate edge detectors. Here, the noise is due to round-off to grid-based coordinates. We already mentioned that the tangents of the polygon are of no use here, because only a few different directions are possible. Therefore, filtering is absolutely necessary. Results are shown in figure 8.10. It can be seen that small filter windows are insufficient, but even with a large window (of radius 8) the accuracy of the gradient is not nearly achieved.

Finally, we investigate the accuracy near the ends of open polygons. Here, a symmetric filters or symmetric model fits cannot be applied because the remaining part of the curve is too short on one side of the current point. One could try to apply reflective boundary conditions to elongate the curve, but this is not helpful because the computed tangent directions will then simply converge to the direction of the curve's last polygon segment when the current point moves closer to the end. Therefore, we need an asymmetric tangent estimation method. Taking the direction of the curve's last segment (instead of the average between two consecutive polygon segments) is the simplest possibility. Alternatively, one can apply model fit methods in asymmetric windows. In our experiments on ellipses, we simulate open polygons by forcing the use of asymmetric windows. That is, an asymmetric window size of 4 pixels contains as many points as a symmetric window with radius 2, but all points in the window are on the same side of the current point. Figure 8.11 shows the results. It can be seen that the segment-based and line fit methods are not working well in the asymmetric case whereas the curve fit method works quite well, due to its ability to extrapolate a curved boundary better than is possible with straight lines. However, it is unclear whether its performance matches the one of the gradient, because we did not simulate by how much the gradient direction will be distorted near the end of a boundary (i.e. near a junction).

We draw the following conclusions from our experiments:

1. The tangent directions estimated from gradient directions are very accurate.[3]

[3]This is especially true when one considers that we did not yet take any measures for improving raw gradient directions, comparable to using larger windows in polygon-based tangent estimation.

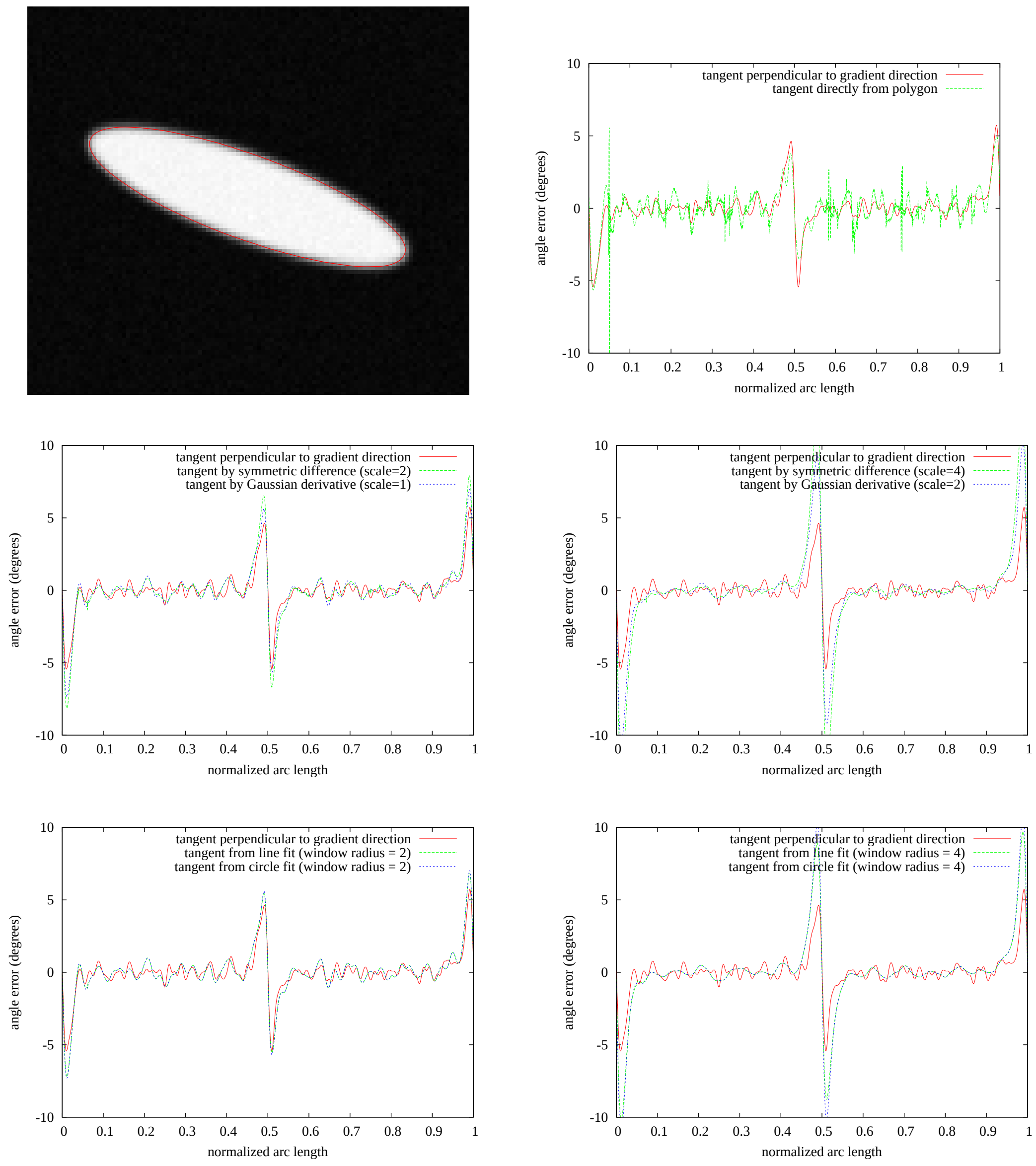


Figure 8.7: Results for a more elongated ellipse with overlayed subpixel watershed boundary. Ellipse radii are $r_1 = 40$, $r_2 = 10$ (i.e. perimeter is ≈ 172 pixels, minimum curvature radius 2.5 pixels), SNR $= 100$, and $\sigma_{\text{PSF}} = \sigma_{\text{filter}} = 1$. The bias of the angle estimates near the ellipse's apexes is even more pronounced, especially in the filtering and model fitting methods.

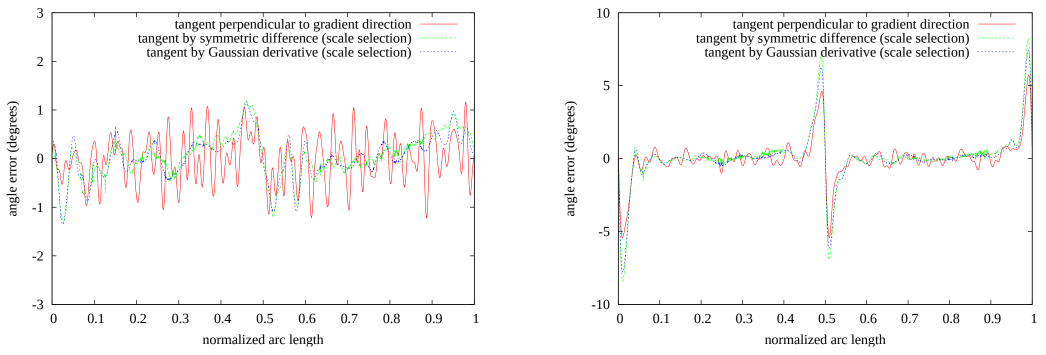


Figure 8.8: Results of filtering with optimal scale selection on the images figure 8.4 (left) and figure 8.7 (right). The third derivative (8.7) was computed with $d = 8\sigma_{\text{filter}}$, i.e. in a window of length 32 pixels. The estimated optimal scales were between 2 and 16 pixel (symmetric difference) and 1 and 8 pixels (Gaussian derivative).

2. Due to correlated localization errors in neighboring polygon points, filter-based and model-fitting method are only superior when relatively large filter windows (stretching over at least 4 pixels to either side of the current point when $\sigma_{\text{filter}} = 1$) are used. However, this requires long boundaries to be available and carries the risk of getting biased angle estimates is much higher for large windows. In low-noise images, this problem can be addressed by automatic scale selection, whereas the bias can often be neglected in high-noise images.

3. Tangent angle estimation from pixel-accurate boundaries requires filtering. But even with large filters, the tangents are less accurate than the ones obtained from the gradient direction.

4. Near the end points of open polygons, only the circle fitting method achieves satisfactory performance, besides possibly the gradient direction.

In summary, considering the much higher cost of polygon filtering or model fitting, the method of choice in most situations will be the gradient direction approach.

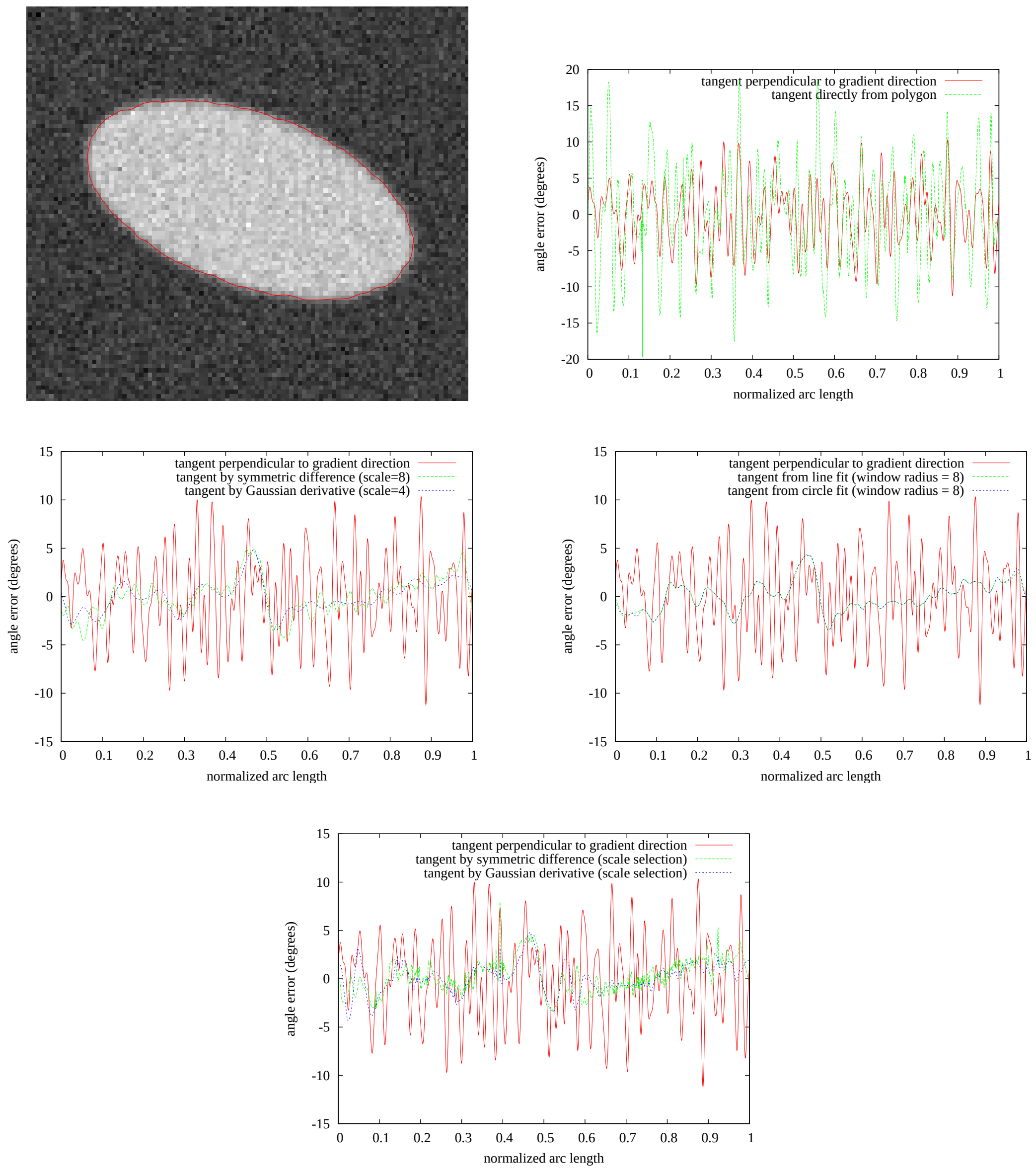
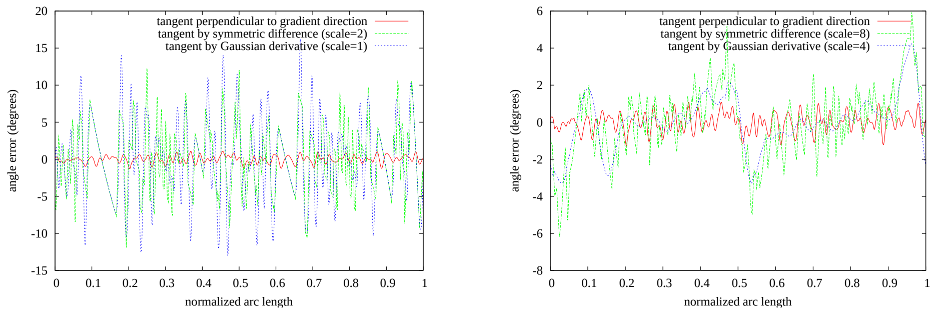
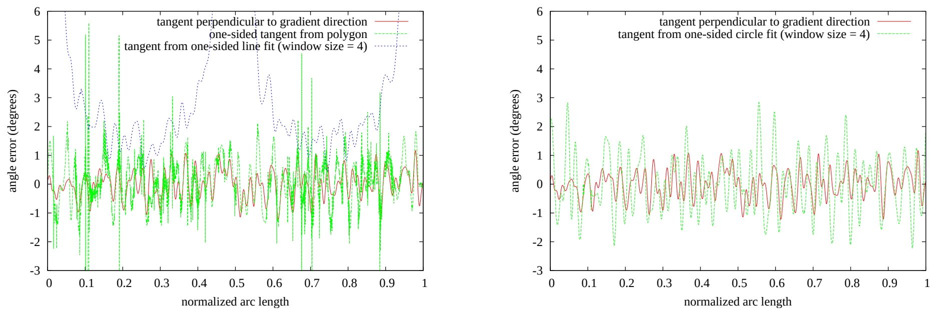


Figure 8.9: Same as figures 8.4 to 8.6, but with SNR = 10. The bias caused by filtering is now smaller than the statistical error without filtering, i.e. filtering is beneficial. Note also that the filtering result with scale selection is now quite noisy due to noisy estimates of the underlying third derivative.



method	average (degrees)	standard deviation (degrees)
gradient	-0.0002	**0.48**
Gauss ($\sigma_C = 1$)	0.017	5.51
Gauss ($\sigma_C = 4$)	0.004	1.44
symmetric difference ($\sigma_C = 2$)	0.012	4.32
symmetric difference ($\sigma_C = 8$)	0.004	1.90

Figure 8.10: Tangent angle errors for the filtering method applied to pixel-accurate Canny edges (image as in figure 8.4). Note that not even the largest windows are sufficient to match the performance of the gradient direction.



method	average (degrees)	std. dev. (degrees)
one-sided simple tangent	0.08	1.24
one-sided line fit (window width = 4)	3.71	2.98
one-sided line fit (window width = 8)	7.43	5.8
one-sided circle fit (window width = 4)	-0.0002	**1.00**
one-sided circle fit (window width = 8)	0.0004	1.06

Figure 8.11: Tangent angle errors for one-sided model fit (image and edge detector as in figure 8.4). Only the circle fit is able to handle the asymmetry reasonably well (note that the gradient data are only shown for reference and do not contain an asymmetry).

9 Improving the Junction Response

Abstract

We had seen in section 7.6 that the localization errors of gradient-based detectors near corners and junctions are quite large (in the order of several pixel diameters). Many alternative boundary detection methods attempt to improve the boundary quality near junctions, but only a few of them manage without heavy heuristics. Tensor-based methods are important representatives of the latter category. In this chapter we describe two approaches to tensor computation: (i) by means of spatial integration of a vectorial boundary indicator (i.e. isotropic and anisotropic structure tensors of the gradient), and (ii) by means of 2-dimensional quadrature filters (the boundary tensor). While the theoretical justifications of these methods are extremely promising, their objective experimental evaluation is somewhat disappointing: none of the tensor methods is consistently outperforming the simple image gradient.

9.1 Tensors

The quality of gradient-based boundaries is very different along edges (a fraction of a pixel diameter) and near corners and junctions (several pixel diameters). When we work conservatively and adjust the parameters of subsequent algorithms (such as (α, β)-boundary reconstruction, algorithm 5.12) according to the accuracy limits suggested by the junction errors, we will waste a lot of resolution near edges. It is therefore an important question whether the junction response can be improved, if possible to the point where it matches the accuracy of the edge response.

One reason for the unsatisfactory performance near junctions is the inability of gradient-based boundary detectors to represent several orientations at a single location: There is only one gradient direction in every point, but corners and junctions are defined as points where edges with two or more different orientations meet. Therefore, a promising way toward better junction response is the introduction of boundary indicators that are not restricted to a single orientation. One approach to these boundary indicators is the concept of *tensors*, see e.g. [Granlund & Knutsson 95].

Tensors are collections of measurements (called *tensor elements*) whose values change in a certain, well-defined way when the underlying coordinate system is rotated. Each tensor element is referred to by an ordered tuple of indices, and a tensor whose elements are defined over p different indices is called a p^{th}-*order tensor*. Each index takes values between 1 and the dimension N of the underlying space, i.e. $N = 2$ in case of 2-dimensional images. A tensor with no indices ($p = 0$) is referred to as a 0^{th}-order tensor, and it contains only a single number, called a *scalar*. A tensor with one index in two dimensions is a set of two numbers (T_1, T_2) and referred to as a 1st-order tensor or a *vector*. Likewise, a 2nd-order tensor consists of a 2 by 2 matrix of numbers, and so on. In case of Cartesian tensors (i.e. those defined in a Cartesian coordinate system) the relationship between tensor elements in a given coordinate system and a rotated one is particularly simple: The tensor elements in the rotated system can be calculated as linear combinations of the tensor elements in the original system:

$$\tilde{T}_{i_1 \dots i_p} = \sum_{l_1=1}^{N} \cdots \sum_{l_p=1}^{N} r_{i_1 l_1} \dots r_{i_p l_p} T_{l_1 \dots l_p} \tag{9.1}$$

where $T_{l_1 \dots l_p}$ are the elements of a p^{th}-order tensor, and r_{il} are the elements of the N-dimensional rotation matrix. These transformation rules ensure that the properties represented by the tensor as a whole remain invariant under Euclidean transformations (rotation and translation) of the underlying space, although the values of the individual tensor elements change. In the special case of a 0th-order tensor, we have $\tilde{T} = T$, i.e. T is a rotationally invariant quantity. For example, the intensity values of an analog image can be interpreted as 0th-order tensors because they are independent of the rotation of the camera around the optical axis.

New tensors can be created from existing ones by a number of simple operations:

- Linear combinations: When $\mathbf{T}_1$ and $\mathbf{T}_2$ are tensors of the same order, then $\mathbf{T} = \alpha_1 \mathbf{T}_1 + \alpha_2 \mathbf{T}_2$ is also a tensor of the same order, i.e. equation (9.1) remains valid.

- Cartesian product (also called outer product): A tensor $\mathbf{T}$ of order $p + q$ can be obtained from two given tensors $\mathbf{P}$ and $\mathbf{Q}$ of orders p and q respectively, when the individual tensor elements are defined as:

$$T_{i_1 \dots i_p j_1 \dots j_q} = P_{i_1 \dots i_p} Q_{j_1 \dots j_q}$$

- Contraction: A tensor $\mathbf{P}$ of order $p - 2$ can be obtained from a tensor $\mathbf{T}$ of order p by taking the sum over a pair of indices, say the indices numbered k and l

$$Q_{i_1 \dots i_{k-1} i_{k+1} \dots i_{l-1} i_{l+1} \dots i_p} = \sum_{j=1}^{n} T_{i_1 \dots i_{k-1} j \, i_{k+1} \dots i_{l-1} j \, i_{l+1} \dots i_p}$$

  The notation of this operation is often simplified by Einstein's summation convention, where summation is automatically implied when a tensor element has two like-named indices. For example, $Q_{im} = T_{ijjm}$ denotes the contraction of a 4th-order tensor to a 2nd-order one, i.e. $Q_{im} = T_{i11m} + T_{i22m} + \dots T_{iNNm}$ where N is the dimension of the space.

Obviously, arbitrary powers of rotationally invariant quantities (i.e. 0th-order tensors) remain rotationally invariant. Likewise, it is easily shown that matrix products of 2nd-order tensors are still 2nd-order tensors.

New rotationally invariant quantities can be defined by contracting *all* indices of a given even order tensor. For example, in a 2-dimensional space, there are two independent possibilities to obtain rotationally invariant numbers from a 2nd-order tensor, namely the *trace*

$$\operatorname{tr}[\mathbf{T}] = T_{11} + T_{22} = T_{ii}$$

and the *determinant*

$$\det[\mathbf{T}] = T_{11}T_{22} - T_{12}T_{21} = \frac{1}{2}(T_{ii})^2 - \frac{1}{2}T_{ij}T_{ji}$$

where the alternative definitions (which make use of the summation convention) emphasize that trace and determinant can indeed be computed in terms of the three elementary tensor operations (for example, $T_{ij}T_{ji}$ denotes two-fold contraction of the outer product of $\mathbf{T}$ with itself). An equivalent pair of rotationally invariant numbers is given by the tensor's *eigenvalues*

$$\begin{aligned} \lambda_{1,2} &= \frac{1}{2}\left(T_{11} + T_{22} \pm \sqrt{(T_{11} - T_{22})^2 + 4T_{12}T_{21}}\right) \\ &= \frac{1}{2}\left(T_{ii} \pm \sqrt{2T_{ij}T_{ji} - (T_{ii})^2}\right) \end{aligned}$$

where the second definition again emphasizes that eigenvalues can be computed in terms of elementary tensor operations. It is easily verified that $\operatorname{tr}[\mathbf{T}] = \lambda_1 + \lambda_2$ and $\det[\mathbf{T}] = \lambda_1\lambda_2$. The significance of 2nd-order tensors for the junction problem arises from the fact that these tensor can distinguish locally 1-dimensional structures (i.e. edges) from locally 2-dimensional ones (i.e. corners and junctions): Tensors representing the first case have only one non-zero eigenvalue, whereas those representing the second case have two non-zero eigenvalues. Thus, 2-dimensional second order tensors provide more powerful boundary representations than scalar or vector-valued measurements (such as the ones we discussed in chapter 7 – directed second derivative, Laplacian operator, gradient vector and magnitude), although they still have an important limitation: While they are able to distinguish locally 2-dimensional structures from 1-dimensional ones, they cannot represent two *independent* local directions. The eigenvectors associated with the two eigenvalues are always pointing into orthogonal directions, i.e. only one direction can be chosen independently. This is insufficient for certain applications. Therefore, some authors proposed even more powerful tensor representations. For example, [Aach et al. 06] compute 3-dimensional vectors from the second derivatives (f_{xx}, f_{xy}, f_{yy}) of the image and define 2nd-order tensors by the outer product of these vectors with themselves, followed by spatial integration over a neighborhood. The resulting tensors can encode two independent directions. An even more powerful representation is proposed by [Nordberg 04], who computes a 4th-order tensor in every point of the image. Multiple local orientations are then determined by a generalization of eigenvector decomposition to 4th-order tensors. However, these approaches are rather new, and their properties in terms of accuracy and reliability are not yet known in detail. Therefore, in the sections to follow we will restrict our attention to the well-known case of 2-dimensional second order tensors.

9.2 Tensor Definition by Spatial Integration of Gradients

Corners and junctions are characterized by the fact that the gradients in a neighborhood have different directions. Therefore, it should be possible to find corners and junctions by integrating (i.e. averaging) the gradient directions over the neighborhood. However, direct integration of the gradient vectors does not give useful results, because gradients with opposite directions cancel out. It is necessary to first transform the gradients into a representation where this cannot happen. This is achieved by taking the outer product of the gradient vector with itself, resulting in the *gradient tensor*:

$$\begin{aligned}\mathbf{G} &= \nabla f \, \nabla f^T \\ &= \begin{pmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{pmatrix}\end{aligned} \tag{9.2}$$

The gradient tensor is always rank-deficient, i.e. it has only one non-zero eigenvalue, and the associated eigenvector points in the gradient direction or in the opposite direction (in other words, the tensor represents only an orientation, not a direction). Now, spatial integration can no longer cause differently oriented structures to cancel each other. Instead, we obtain tensors with two non-zero eigenvalues, and the magnitude of the smaller eigenvalue indicates the degree to which the present point is locally 2-dimensional. The tensor obtained after spatial integration is called *structure tensor*

$$\begin{aligned}\mathbf{S} &= g_{\sigma_i} \star \mathbf{G} \\ &= \begin{pmatrix} g_{\sigma_i} \star f_x^2 & g_{\sigma_i} \star f_x f_y \\ g_{\sigma_i} \star f_x f_y & g_{\sigma_i} \star f_y^2 \end{pmatrix}\end{aligned} \tag{9.3}$$

where g_{σ_i} denotes an integration filter at scale σ_i, usually a Gaussian. This approach first appeared in [Bigün et al. 91, Nagel 85].

Unfortunately, the advantages of spatial integration come at a price: Spatial integration results in a loss of resolution. In addition to the filters computing the derivatives, a second smoothing is imposed by the spatial integration step. For example, when we consider a step edge blurred with a Gaussian PSF, the gradient magnitude along the gradient direction has the form of a Gaussian with width $\sqrt{\sigma_{\text{PSF}}^2 + \sigma_{\text{filter}}^2}$, but the width after integration becomes $\sqrt{\sigma_{\text{PSF}}^2 + \sigma_{\text{filter}}^2 + 2\sigma_i^2}$ (where the edge strength has been defined as the square root of $\operatorname{tr}[\mathbf{S}]$, which is the analogue of the gradient magnitude in the context of the structure tensor). In practice, one usually sets σ_i to a value between $2\sigma_{\text{filter}}$ and $3\sigma_{\text{filter}}$. Thus, with standard values of $\sigma_{\text{PSF}} = 0.5$, $\sigma_{\text{filter}} = 0.9$, the width of the edge response increases three- to four-fold after integration. In order to remain distinguishable, the separation between two neighboring edges must be increased accordingly.

This loss of resolution is often unacceptable. Blurring can be reduced when the integration is not performed uniformly in all directions, but is restricted to directions where it actually has a desirable effect, namely along edges. The idea is that reliable gradient information is only available up to a certain distance from a junction, whereas gradients in the immediate neighborhood of a junction are distorted. Therefore, edge information should be extrapolated toward junctions from the locations where it is still reliable.

This idea is very similar to the ability of the human visual system to group features according to the rules of *good continuation*. This ability is, for example, used to group the individual segments of a dashed line into a coherent whole. Edge extrapolation according to the rules of good continuation has been explored by [Williams & Jacobs 97] who proposed *stochastic completion fields* that simulate contour continuation by means of random walks. The walk starts at a line ending and proceeds along the previous tangent direction and with the previous curvature, but direction and curvature are allowed to change according to a certain probability distribution. As the distance from the starting point increases, these probability distributions are becoming less and less peaked. The stochastic completion field is now defined as the probability for a random walker starting at the line ending to reach a particular point in the plane. Each completion field has the shape of a fan oriented according to the line ending direction, and the superposition of all completion fields in the image has exactly the desired effect of anisotropic edge extrapolation.

[Williams & Jacobs 97] determine completion fields by means of random walk simulation, whereas [August & Zucker 03] compute them as solutions of suitably defined differential equations. Since these solutions are quite expensive, [Medioni et al. 00] propose a much simpler parametrization of the completion fields. In their approach, line endings are represented by tensors encoding feature strength, isotropy, and orientation. The tensor at a given image point influences all other points in a neighborhood: It casts votes as to how the tensors at neighboring points should look like if they were good continuations of the present tensor. Thus, votes are strong when the two points are close together, and the orientations of the two tensors are consistent with a circular arc connecting the two points (i.e. the orientation of voting tensors is adjusted according to the curvature of the connecting arc). Each voting field has thus the shape of a butterfly oriented along the tensor direction at the field's center, and the superposition of all voting fields gives the final completion field. Accordingly, the method is called *tensor voting*.

A further simplification was proposed by the *hourglass filter* method [Köthe 03c]. Here, tensors vote in their neighborhood like in standard tensor voting, but they only cast votes according to their own orientation. This simplification is possible because we use relatively small voting kernels (scales in the kernel's major direction are about 2 to 3 pixels, corresponding to the size of a junction's neighborhood were the gradient information is distorted). The voting strength is thus defined as

$$h\left(r, \Delta\phi\right) = \frac{1}{p} \exp\left(-\frac{r^2}{2\sigma_i^2}\right) \exp\left(-\frac{\left(\tan\,\Delta\phi\right)^2}{2\sigma_\phi^2}\right) \tag{9.4}$$

where $r = |\vec{x} - \vec{x}_0|$ is the distance between the point $\vec{x}_0$ casting the vote and the point $\vec{x}$ receiving the vote, $\Delta\phi = \arccos\left(\vec{t}_0^T \cdot \vec{t}\right)$ is the angle difference between the minor axis $\vec{t}_0$ of the tensor (i.e. the eigenvector corresponding to the small eigenvalue, which points along the edge when $\mathbf{T}$ is a gradient tensor) and the vector $\vec{t} = \frac{1}{r}\left(\vec{x} - \vec{x}_0\right)$, σ_i and σ_ϕ are the radial and angular scales of the kernel, and the constant p normalizes the

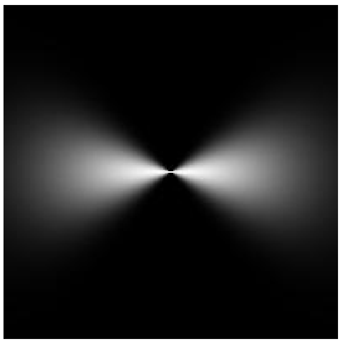


Figure 9.1: Hourglass kernel according to (9.4) with $\vec{t}_0$ along the horizontal axis, $\sigma_\phi = 0.4$.

kernel to unit integral. Figure 9.1 depicts the kernel for the case that $\vec{t}_0$ points along the horizontal axis. When $\vec{t}_0$ is different (i.e. the edge runs in another direction), the voting kernel is rotated accordingly. The scale σ_ϕ controls the opening angle of the hourglass, and we found $\sigma_\phi = 0.4$ to be a good choice for most applications. This corresponds to an hourglass opening angle of about 50°, i.e. the filter amplitude at $\Delta\phi = \pm 25°$ is half the maximal amplitude at $\Delta\phi = 0$. Filter results are not very sensitive to the choice of σ_ϕ – values between 0.3 and 0.7 give essentially the same results. However, for $\sigma_\phi < 0.3$, the filter becomes susceptible to noise in the estimated direction $\vec{t}_0$. For $\rho > 0.7$, the anisotropy of the kernel is lost, and undesirable blurring becomes again visible.

When an anisotropic kernel is used, the structure tensor is no longer obtained from the gradient tensor by means of convolution. Instead, it must be computed according to

$$\mathbf{S}_h(\vec{x}) = \sum_{\vec{y}} h\left(|\vec{x}-\vec{y}|, \arccos\left[\vec{t}_0(\vec{y})^T \cdot (\vec{x}-\vec{y}) / |\vec{x}-\vec{y}|\right]\right) \mathbf{G}(\vec{y}) \tag{9.5}$$

Note that it is crucial to determine the reference direction $\vec{t}_0$ according to the eigendirections of $\mathbf{G}$ at the point $\vec{y}$, because votes must be casted along the edge direction of the voting point[1]. We will refer to the result $\mathbf{S}_h$ of anisotropic integration according to (9.5) as the *hourglass tensor*. Figure 9.2 compares the gradient magnitude with the trace of the isotropic and anisotropic structure tensors for an example image.

Another approach toward anisotropic structure tensor integration is a statistical one. Suppose point $\vec{x}_0$ is located near a corner or junction. Then all edges in a neighborhood of $\vec{x}_0$ will meet approximately at $\vec{x}_0$. The orientations of these edges are determined by the direction perpendicular to the gradient direction, and the importance of each edge is determined by the gradient magnitude, both measured at a point on the edge. It is now possible to compute a least-squares estimate of the most likely crossing point of all edges in the given neighborhood. Let the gradient at point $\vec{x}$ be $\vec{g}(\vec{x})$. Then the normal of the edge through $\vec{x}$ is parallel to the gradient, i.e. $\vec{n}(\vec{x}) = \vec{g}(\vec{x}) / |\vec{g}(\vec{x})|$. The distance between

[1]If the direction $\vec{t}_0$ could be estimated at point $\vec{x}$, a slightly modified version of the anisotropic filter could be efficiently implemented by means of steerable filters [Freeman & Adelson 91]. However, the requirement to use $\vec{t}_0(\vec{y})$ implies that the complete 2-dimensional sum has to be evaluated at every point.

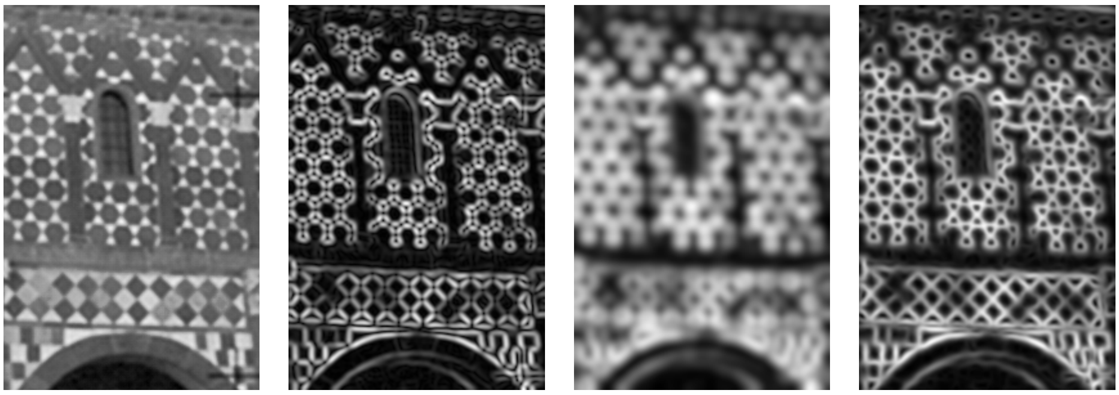


Figure 9.2: Left: The facade of the entrance hall of Lorsch monastery is decorated with delicate tilings. Center left: The gradient magnitude correctly represents the edges of the tiles, but is zero at their corners (because these points happen to be saddle points, i.e. the gradient vanishes). Center right: Due to excessive blurring, the tiling structure is lost in the square root of the structure tensor trace. Right: The square root of the trace of the *anisotropic* structure tensor (i.e. integration is done with the hourglass filter) represents both edges and corners correctly. Parameters were $\sigma_{\text{filter}} = 0.8$, $\sigma_i = 1.6$, $\sigma_\phi = 0.4$. All computations were performed in a two-fold oversampled image to avoid aliasing artifacts (cf. section 7.1), and the given filter sizes have been scaled accordingly.

this edge and the point $\vec{x}_0$ is given by the scalar product

$$d\left(\vec{x}, \vec{x}_0\right) = \vec{n}^T\left(\vec{x} - \vec{x}_0\right) = \frac{\vec{g}(\vec{x})^T\left(\vec{x} - \vec{x}_0\right)}{|\vec{g}(\vec{x})|}$$

Now, the most likely edge crossing is the point $\vec{x}_0$ which minimizes the squared sum over all distance d is minimized:

$$\vec{x}_{0,\text{opt}} = \arg\min_{\vec{x}_0} \sum_{\vec{x}} w\left(|\vec{x} - \vec{x}_0|\right) |\vec{g}(\vec{x})|^2 \; d\left(\vec{x}, \vec{x}_0\right)^2$$

where $w\left(|\vec{x} - \vec{x}_0|\right)$ is a windowing function (usually a Gaussian), and $|\vec{g}(\vec{x})|^2$ weights point $\vec{x}$ according to its edge strength. Note that this factor cancels with the denominator of d^2. The right hand side can therefore be written as

$$\vec{x}_{0,\text{opt}} = \arg\min_{\vec{x}_0} \sum_{\vec{x}} w\left(|\vec{x} - \vec{x}_0|\right) \cos\left(\Delta\psi\right)^2 \; \mathbf{G}(\vec{x})$$

where $\Delta\psi$ is the angle between the edge normal $\vec{n}(\vec{x})$ and the vector $\vec{x} - \vec{x}_0$, and $\mathbf{G}(\vec{x}) = \vec{g}(\vec{x})\,\vec{g}(\vec{x})^T$ is the gradient tensor at point $\vec{x}$. Since $\cos\left(\Delta\psi\right)^2 = 1 - \sin\left(\Delta\psi\right)^2$, and $\Delta\psi = \pi/2 - \Delta\phi$ (where $\Delta\phi$ is the angle between the edge direction and the vector $\vec{x} - \vec{x}_0$), this is equivalent to

$$\vec{x}_{0,\text{opt}} = \arg\min_{\vec{x}_0} \left[\sum_{\vec{x}} w\left(|\vec{x} - \vec{x}_0|\right) \mathbf{G}(\vec{x}) - \sum_{\vec{x}} w\left(|\vec{x} - \vec{x}_0|\right) \cos\left(\Delta\phi\right)^2 \; \mathbf{G}(\vec{x})\right]$$

When we assume that $\vec{x}_0$ is close to the true junction point $\vec{x}_{0,\mathrm{opt}}$, the weights $w\left(|\vec{x}-\vec{x}_0|\right)$ will not change significantly when we move from $\vec{x}_0$ to $\vec{x}_{0,\mathrm{opt}}$. Therefore, the first term in brackets is approximately constant, and we can also write

$$\vec{x}_{0,\mathrm{opt}} = \arg\max_{\vec{x}_0} \sum_{\vec{x}} w\left(|\vec{x}-\vec{x}_0|\right)\, \cos\left(\Delta\phi\right)^2\ \mathbf{G}(\vec{x})$$

The expression to be maximized can be interpreted as another anisotropic filter conforming to (9.5). Its response at junctions is a local *maximum*, in agreement with the desired behavior of anisotropic integration. The kernel of this version of anisotropic integration is

$$h_2(r,\Delta\phi) = \frac{1}{p}\exp\left(-\frac{r^2}{2\sigma_i^2}\right)\cos\left(\Delta\phi\right)^2 \tag{9.6}$$

where p is again a normalization constant. This kernel looks quite similar to the hourglass kernel with $\sigma_\phi = 0.85$ and still exhibits considerable blurring perpendicular to the edges. More pronounced anisotropy (smaller opening angle) is achieved when the cosine is raised to a higher power. A kernel similar to the hourglass kernel with $\sigma_\phi = 0.4$ is obtained by using $\cos^6$

$$h_6(r,\Delta\phi) = \frac{1}{p}\exp\left(-\frac{r^2}{2\sigma_i^2}\right)\cos\left(\Delta\phi\right)^6 \tag{9.7}$$

Results of anisotropic integration with the kernels (9.6) and (9.7) are shown in figure 9.3, where it can be seen that the $\cos^6$ kernel does indeed perform extremely similar to the hourglass kernel.

Yet another approach to anisotropic structure tensor integration is suggested by structure-preserving diffusion. In contrast to isotropic diffusion, where the diffusivity is a scalar, the diffusivity in anisotropic diffusion is a tensor. That is, the diffusivity is represented by an ellipse whose axes are oriented along the directions of maximum and minimum diffusivity. Under this interpretation, we get an anisotropic Gaussian integration kernel

$$h_e(r,\Delta\phi) = \frac{1}{p}\exp\left(-\frac{\left(r\,\cos\left(\Delta\phi\right)\right)^2}{2\sigma_1^2}\right)\exp\left(-\frac{\left(r\,\sin\left(\Delta\phi\right)\right)^2}{2\sigma_2^2}\right) \tag{9.8}$$

where r and $\Delta\phi$ are defined as before, p is a normalization constant, and σ_1 and $\sigma_2 < \sigma_1$ are the scales of the kernel along the edge and perpendicular to it, respectively. This kernel is inserted into (9.5) instead of the hourglass kernel[2]. We obtained good results with $\sigma_1 \approx 2\sigma_{\mathrm{filter}}$ and $\sigma_2 \lesssim \sigma_{\mathrm{filter}}$. Although the shapes of the anisotropic Gaussian kernel and the hourglass kernel are quite different, the filter results are remarkably similar, see figure 9.3.

[2]Note that the result of this filter is not the same as the result of anisotropic diffusion, because the latter requires a sequence of infinitesimally small smoothing steps, whereas we use a single step with a relatively large ellipse.

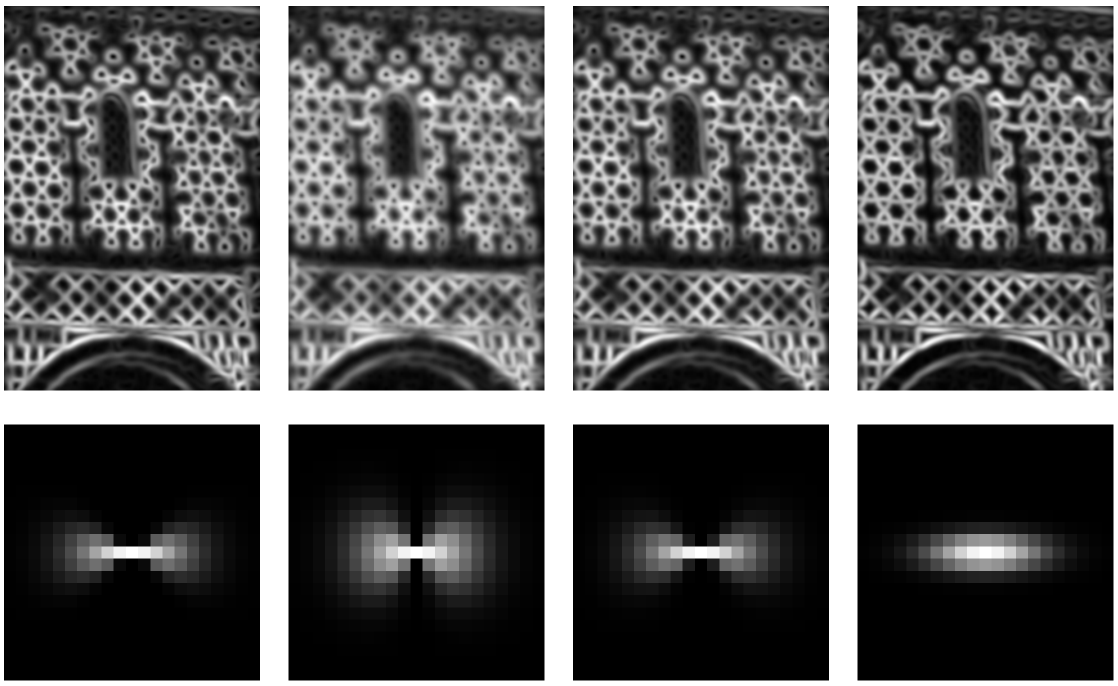


Figure 9.3: Comparison of different variants of anisotropic tensor integration of the gradient tensor (top row) and the corresponding integration kernels (bottom row). Left: Result of the hourglass filter (9.4) for the Lorsch example (repeated from figure 9.2). Center left: $\cos^2$ kernel according to (9.6). Center right: $\cos^6$ kernel according to (9.7). Right: Anisotropic Gaussian kernel according to (9.8). Filter parameters were $\sigma_{\text{filter}} = 0.8$, $\sigma_i = \sigma_1 = 1.6$, $\sigma_2 = 0.5$. Filtering was performed in two-fold oversampled images (with filter sizes adjusted accordingly), and images are shown for this resolution.

9.3 Tensor Definition by Combination of Even and Odd Filters

An alternative to spatial tensor integration was proposed in [Granlund & Knutsson 95] and earlier works by the same authors. Originally, they had been interested in the extension of edge detection to other locally 1-dimensional image features, in particular lines (as found in line drawings). Formally, 1-dimensional features are defined by the fact that the image is locally reduced to a 1-dimensional function that varies only along a certain direction $\vec{n}$ and is constant perpendicular to that direction:

$$f_2(\vec{x}) \approx f_1(\vec{x}^T \vec{n})$$

(We write $\approx$ instead of $=$ because the condition is only fulfilled locally in real images.) Then the local signal energy and orientation can be represented by an *orientation tensor* which is the outer product of the local direction vector times a factor λ encoding signal strength:

$$\mathbf{T} = \lambda \vec{n}\,\vec{n}^T \tag{9.9}$$

The value of this tensor should now be equal for edges and lines of equal strength, regardless of their different profiles. This property is called *phase invariance* because edges and lines can be understood as superpositions of trigonometric (complex exponential) basis functions at different phase (namely phase 0 or π for lines and $\pm\pi/2$ for edges). Phase invariance can be achieved by means of oriented quadrature filters [Granlund & Knutsson 95] or local polynomial approximations [Farnebäck 02]. *Quadrature filter pairs* have been invented to estimate the instantaneous energy and phase of 1-dimensional signals. A quadrature pair $(h_{\text{even}}, h_{\text{odd}})$ consists of an even and an odd symmetric filter, and the instantaneous (edge or line) energy can be calculated as the sum of squares of the filter responses:

$$E(x) = (h_{\text{even}} \star f_1)^2 + (h_{\text{odd}} \star f_1)^2 \tag{9.10}$$

To actually form a quadrature pair, the filters must be related by the Hilbert transform $\mathcal{H}$, which is defined in the Fourier domain by

$$H_{\text{odd}}(u) = \mathcal{H}[H_{\text{even}}(u)] = \mathbb{i}\frac{u}{|u|} H_{\text{even}}(u) = \mathbb{i}\ \text{sign}(u) H_{\text{even}}(u) \tag{9.11}$$

(slanted capitals denote the Fourier transforms of the corresponding lower-case functions). To apply these filters in 2D, it is conventional to rotate them into some orientation of interest. In order to estimate $\mathbf{T}$ in a 2D image, at least 3 orientations are necessary [Granlund & Knutsson 95]. When the local image structure is indeed 1-dimensional and the orientations $\theta_i = [0, \pi/3, 2\pi/3]$ are used, one gets

$$\mathbf{T} = \sum_i (\mathbf{m}_i \mathbf{m}_i^T - \mathbf{I}/4)\, E_i \tag{9.12}$$

where E_i is the energy computed for orientation i, $\mathbf{m}_i = (\cos\theta_i, \sin\theta_i)^T$ and $\mathbf{I}$ is the unit tensor.

Another possibility to define a tensor representation which is sensitive to both step edges and lines is the formula

$$\mathbf{T} = \mathbf{a}\mathbf{a}^T + \gamma \mathbf{B}\mathbf{B}^T \tag{9.13}$$

where **a** and **B** are first- and second-order tensors which represent odd-symmetric (i.e. step edge-like) and even symmetric (e.g. line-like) image structures respectively. An in-depth discussion of how to estimate suitable **a** and **B** can be found in [Farnebäck 02]. Possibilities include local polynomial fits, facet models, moment filters, and Gaussian derivative filters. However, when **a** and **B** are estimated according to one of these methods, **T** is only phase invariant for a single feature scale depending on γ (γ is therefore considered as an algorithm tuning parameter). This is undesirable because most images contain features at different scales, and phase invariance cannot be achieved for all of them simultaneously by a fixed value of γ.

Fortunately, there is a method for defining **a** and **B** that does not suffer from these shortcomings. It is based on the *Riesz transform,* a generalization of the Hilbert transform (9.11) to arbitrary dimensional spaces. According to [Felsberg & Sommer 01], the *Riesz transform* is defined in the Fourier domain by

$$\mathcal{H}_N[H(\vec{u})] = \mathbb{i}\frac{\vec{u}}{|\vec{u}|}H(\vec{u}) \tag{9.14}$$

In contrast to the Hilbert transform, the argument $\vec{u}$ is now a vector in an N-dimensional space rather than a scalar. In the spatial domain, the Riesz transform of some function $h(\vec{x})$ can be written as a convolution

$$\mathcal{H}_N[h(\vec{x})] = -\frac{\Gamma((N+1)/2)}{\pi^{(N+1)/2}}\left(\frac{\vec{x}}{|\vec{x}|^{N+1}} \star h(\vec{x})\right) \tag{9.15}$$

where $\Gamma(.)$ is the gamma function.

We see that the Riesz transform is very similar to the gradient (which is expressed as $2\pi\mathbb{i}\,\vec{u}\,H(\vec{u})$ in the Fourier domain). The only difference is the factor $(2\pi\,|\vec{u}|)^{-1}$ in the Riesz transform expression. The meaning of this difference is best understood when the two operators are written in polar coordinates. In the 2-dimensional Fourier domain with polar coordinates (ρ, ϕ), the gradient becomes $2\pi\mathbb{i}\,(\rho\cos\phi, \rho\sin\phi)^T$, whereas the expression for the Riesz transform reads $\mathbb{i}\,(\cos\phi, \sin\phi)^T$. That is, both operators modify the angular component of a function $H\,(\rho, \phi)$ in the same way, but the gradient does also modify the radial component, which the Riesz transform leaves alone. It turns out that this is the key property for making filters phase invariant at all scales simultaneously: When all filters involved in the construction of the tensor have the same radial spectrum, and differ only in their angular spectra, phase invariance at a single scale automatically implies phase invariance at all scales.

Given any radial symmetric band-pass with transfer function $H\,(|\vec{u}|)$, the corresponding first-order Riesz kernel in the spatial domain is a vector-valued function of the form

$$h^{\text{odd}}(\vec{x}) = \mathcal{H}\left[h\left(|\vec{x}|\right)\right] = \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} \tilde{h}\left(|\vec{x}|\right)$$

where $\tilde{h}\left(|\vec{x}|\right)$ is the first order Hankel transform of $H\left(|\vec{u}|\right)$:

$$\tilde{h}\left(r\right) = 2\pi \int_0^\infty H(\rho)\, J_1(2\pi\, r\, \rho)\, \rho\, d\rho$$

and $J_1(.)$ is the first-order Bessel function of the first kind. The Hankel transform is essentially a 2-dimensional Fourier transform for polar separable functions where the angular integration has already been carried out. The second order Riesz kernel is a matrix-valued function

$$h^{\text{even}}(\vec{x}) = \mathcal{H}^2\left[h\left(|\vec{x}|\right)\right] = \begin{pmatrix} \cos^2\phi & \sin\phi\cos\phi \\ \sin\phi\cos\phi & \sin^2\phi \end{pmatrix} \tilde{\tilde{h}}\left(|\vec{x}|\right)$$

with $\tilde{\tilde{h}}\left(|\vec{x}|\right)$ being the second order Hankel transform of $H\left(|\vec{u}|\right)$

$$\tilde{\tilde{h}}\left(r\right) = 2\pi \int_0^\infty H(\rho)\, J_2(2\pi\, r\, \rho)\, \rho\, d\rho$$

The kernels h^{odd} and h^{even} play the same role as the even and odd filters in (9.10) and generalize them to 2D. It is easily shown that h^{odd} and h^{even} always fulfill the requirements (9.1) of first- and second-order tensors, independent of the particular form of $H\left(|\vec{u}|\right)$. In order to obtain a boundary detector, we simply have to choose $H\left(|\vec{u}|\right)$ suitably, for example as the Laplacian of Gaussian (see section 9.3.2).

The convolution of a scalar-valued image $f\left(\vec{x}\right)$ (i.e. of a function which is a 0^{th}-order tensor in every point) with h^{odd} and h^{even} results in a vector- and matrix-valued image respectively. Thus, the filtered images are functions whose values are 1^{st}- and 2^{nd}-order tensors in every point, i.e. we can write

$$\mathbf{a}_h\left(\vec{x}\right) = \left[\begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} \tilde{h}\left(|\vec{x}|\right)\right] \star f\left(\vec{x}\right) \tag{9.16}$$

$$\mathbf{B}_h\left(\vec{x}\right) = \left[\begin{pmatrix} \cos^2\phi & \sin\phi\cos\phi \\ \sin\phi\cos\phi & \sin^2\phi \end{pmatrix} \tilde{\tilde{h}}\left(|\vec{x}|\right)\right] \star f\left(\vec{x}\right) \tag{9.17}$$

Using the thus obtained $\mathbf{a}_h$ and $\mathbf{B}_h$, we can now define $\mathbf{T}$ in every image point according to (9.13) as

$$\mathbf{T}_h = \mathbf{a}_h\mathbf{a}_h^T + \mathbf{B}_h\mathbf{B}_h^T = \mathbf{T}_B \tag{9.18}$$

This tensor was first introduced in [Köthe 03a] (by means of a different derivation) and will be referred to as the *boundary tensor*. As we will show below, this name is justified because $\mathbf{T}_B$ does not only react to step-like and line-like 1-dimensional features but also gives reasonable responses at 2-dimensional feature points such as corners and junctions. It is also easily seen that the boundary tensor is phase invariant. Unlike the tensors in (9.13), there is no need for a tuning parameter γ to achieve this. This follows from the fact that $\mathbf{a}_h$ and $\mathbf{B}_h$ are related by the Riesz transform. Consider an intrinsically 1-dimensional image $f\left(\vec{x}\right) = f_1\left(\vec{x}^T\vec{n}\right)$, where we assume without loss of generality that

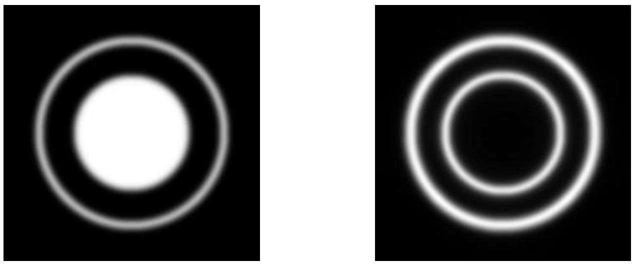


Figure 9.4: Left: A test image with a circular step edge and a circular line. Right: the corresponding boundary energy. The boundary energy is rotationally invariant and phase invariant, i.e. it produces the same response for odd-symmetric features such as edges and even-symmetric features such as lines.

$\vec{n}$ is parallel to the horizontal axis. Then, all terms in $\mathbf{a}_h$ and $\mathbf{B}_h$ containing $\sin(\phi)$ are zero, and the trace of $\mathbf{T}_B$ reduces to

$$\operatorname{tr}[\mathbf{T}_B] = \left(\tilde{h} \star f_1\right)^2 + \left(\tilde{\tilde{h}} \star f_1\right)^2$$

which is precisely the same as the quadrature energy (9.10) in the 1-dimensional case[3]. The trace $\operatorname{tr}[\mathbf{T}_B]$ is called the *boundary energy* because it generalizes the 1D energy computation (9.10) to two dimensions. Figure 9.4 demonstrates phase invariance and rotational invariance of the boundary energy for a simple test image. The boundary tensor is also closely related to the orientation tensors according to (9.12): It can be shown that the latter tensors converge to the boundary tensor when the energies E_i are computed by means of steerable filters [Freeman & Adelson 91], and the number of orientations approaches infinity.

The downside of phase invariance is a loss in resolution in comparison to applying just an odd or even filter. Consider, for example, a single straight step edge. Both the boundary energy and the squared magnitude of the Gaussian gradient have Gaussian-like response profiles perpendicular to the edge. However, when the scales of the two operators are equal, the gradient profile is only 70% as wide as the boundary energy, cf. figure 9.5. In effect, two edges must have 1.4 times the distance from each other to remain distinguishable in the boundary energy. This is caused by the fact that the even filter part does not contribute useful information at odd-symmetric features like edges. This property of quadrature filters also has a negative effect on the signal-to-noise-ratio of the boundary energy.

9.3.1 Analysis of the Boundary Tensor as a Quadratic Filter

In order to understand the behavior of the boundary tensor near corners and junctions, we adopt the proposal of [Nordberg & Farnebäck 03] and formulate the tensor as a quadratic filter [Sicuranza 92]. This is necessary because the boundary tensor is defined in terms

[3]A more detailed proof that $\mathbf{T}_B$ is phase invariant and generalizes quadrature filters to 2D can be found in [Köthe 06a].

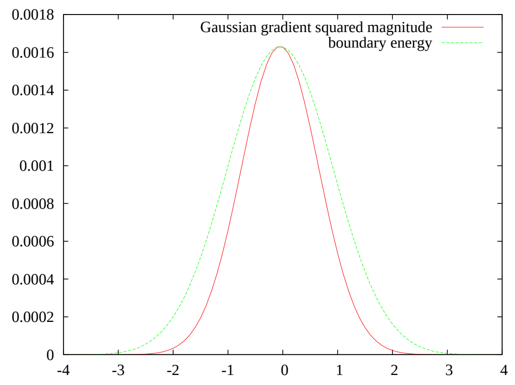


Figure 9.5: Profile of the operator response to a step edge. The squared magnitude of the Gaussian gradient (red) is only 70% as wide as the boundary energy (green). Operator scale was $\sigma = 1$ in both cases.

of *squared* filter responses, so an analysis on the basis of linear filters is insufficient. Quadratic convolution is defined as

$$\tilde{f}(\vec{x}) = \iint h(\vec{x}-\vec{x}_1, \vec{x}-\vec{x}_2) f(\vec{x}_1) f(\vec{x}_2)\, d\vec{x}_1\, d\vec{x}_2$$

where $h(.,.)$ is the kernel, and the method is termed "quadratic" because the original image f appears twice in the integral. Let $h_i^{\text{odd}}(\vec{x})$ denote the i^{th} component ($i = 1, 2$) of the first order kernel $h^{\text{odd}}(\vec{x})$. Then

$$\begin{aligned}
(\mathbf{a}\,\mathbf{a}^T)_{il} &= \mathbf{a}_i \mathbf{a}_l = (h_i^{\text{odd}} \star f)(h_l^{\text{odd}} \star f) \\
&= \int h_i^{\text{odd}}(\vec{x}-\vec{x}_1) f(\vec{x}_1)\, d\vec{x}_1 \int h_l^{\text{odd}}(\vec{x}-\vec{x}_2) f(\vec{x}_2)\, d\vec{x}_2 \\
&= \iint \left(h_i^{\text{odd}}(\vec{x}-\vec{x}_1) h_l^{\text{odd}}(\vec{x}-\vec{x}_2)\right) f(\vec{x}_1) f(\vec{x}_2)\, d\vec{x}_1\, d\vec{x}_2
\end{aligned}$$

Similarly, let $h_{il}^{\text{even}}(\vec{x})$ represent component il ($i, l = 1, 2$) of the kernel for the second order kernel $h^{\text{even}}(\vec{x})$. This leads to

$$\begin{aligned}
(\mathbf{B}\,\mathbf{B}^T)_{il} &= \sum_k \mathbf{B}_{ik} \mathbf{B}_{kl} = \sum_k (h_{ik}^{\text{even}} \star f)(h_{kl}^{\text{even}} \star f) \\
&= \iint \left(\sum_k h_{ik}^{\text{even}}(\vec{x}-\vec{x}_1) h_{kl}^{\text{even}}(\vec{x}-\vec{x}_2) \right) f(\vec{x}_1) f(\vec{x}_2)\, d\vec{x}_1\, d\vec{x}_2
\end{aligned}$$

We can combine both equations into a single quadratic convolution with kernel

$$h_{il}(\vec{x}_1, \vec{x}_2) = h_i^{\text{odd}}(\vec{x}_1) h_l^{\text{odd}}(\vec{x}_2) + \sum_k h_{ik}^{\text{even}}(\vec{x}_1) h_{kl}^{\text{even}}(\vec{x}_2)$$

Then the components of the boundary tensor can be written as

$$\mathbf{T}_{B,il}(\vec{x}) = \iint h_{il}(\vec{x}-\vec{x}_1, \vec{x}-\vec{x}_2) f(\vec{x}_1) f(\vec{x}_2)\, d\vec{x}_1\, d\vec{x}_2$$

Due to Parseval's theorem, the filter response at the origin $\vec{x} = 0$ is equal to the integral over the Fourier transform of the convolution integral, i.e.

$$\mathbf{T}_{B,il}(\vec{x} = 0) = \iint H_{il}(\vec{u}, \vec{v}) F(\vec{u}) F(\vec{v}) \, d\vec{u} \, d\vec{v} \tag{9.19}$$

where F is the 2-dimensional Fourier transform of f, H_{il} is the (2×2)-dimensional Fourier transform of $h_{il}(\vec{x}_1, \vec{x}_2)$. Inserting the Fourier representation of the Riesz transform, H_{il} gets a simple functional form:

$$\begin{aligned} H_{il}(\vec{u}, \vec{v}) &= -\frac{\vec{u}_i}{|\vec{u}|} \frac{\vec{v}_l}{|\vec{v}|} H(|\vec{u}|) H(|\vec{v}|) + \sum_k \left(\frac{\vec{u}_i \vec{u}_k}{|\vec{u}|^2} \frac{\vec{v}_k \vec{v}_l}{|\vec{v}|^2} \right) H(|\vec{u}|) H(|\vec{v}|) \\ &= \frac{\vec{u}_i}{|\vec{u}|} \frac{\vec{v}_l}{|\vec{v}|} \left(-1 + \frac{\vec{u}^T \vec{v}}{|\vec{u}||\vec{v}|} \right) H(|\vec{u}|) H(|\vec{v}|) \end{aligned} \tag{9.20}$$

where $H\left(|\vec{u}|\right)$ is the given radially symmetric bandpass filter. When we base the boundary tensor on the Laplacian of Gaussian, we have $H(t) = -4\pi^2 t^2 \exp\left(-2\pi^2 t^2 \sigma^2\right)$.

In this form, it is now possible to see how H_{il} responds to intrinsically 2-dimensional image features. To simplify matters, we assume that the coordinate system has been shifted so that the point of interest is the coordinate origin. Moreover, we assume the resulting image spectrum $F(\vec{u})$ to be polar separable, i.e. can be written as the product of a radial and an angular function

$$F(\vec{u}) = F(|\vec{u}|, \phi) = F_r(|\vec{u}|) F_a(\phi)$$

If the point of interest is at the center of an intrinsically 2-dimensional feature (i.e. a corner or junction), and the scale of the bandpass filter $H(|\vec{u}|)$ matches the feature scale, this assumption is at least approximately fulfilled in the pass-band of $H(|\vec{u}|)$. The product $H(|\vec{u}|)F(\vec{u})$ then becomes $H(|\vec{u}|)F_r(|\vec{u}|)F_a(\phi)$. After inserting this and (9.20) into the Fourier domain expression (9.19), the integrals over $\vec{u} = \rho_1(\cos(\phi), \sin(\phi))^T$ and $\vec{v} = \rho_2(\cos(\psi), \sin(\psi))^T$ can be expressed in polar coordinates and factor into a product of two integrals:

$$\begin{aligned} \mathbf{T}_B\left(\vec{x} = 0\right) &= \iint \frac{\vec{u}}{|\vec{u}|} \frac{\vec{v}^T}{|\vec{v}|} \left(-1 + \frac{\vec{u}^T \vec{v}}{|\vec{u}||\vec{v}|} \right) H(|\vec{u}|) H(|\vec{v}|) F(\vec{u}) F(\vec{v}) \, d\vec{u} \, d\vec{v} \\ &= \left(\iint \begin{pmatrix} \cos\phi \cos\psi & \cos\phi \sin\psi \\ \sin\phi \cos\psi & \sin\phi \sin\psi \end{pmatrix} \left(-1 + \cos\phi\cos\psi + \sin\phi\sin\psi\right) F_a(\phi) F_a(\psi) \, d\phi \, d\psi \right) \\ &\qquad \times \left(\iint H(\rho_1) H(\rho_2) F_r(\rho_1) F_r(\rho_2) \, \rho_1 \, d\rho_1 \, \rho_2 \, d\rho_2 \right) \\ &= \mathbf{T}_{B,a} \times \mathbf{T}_{B,r} \end{aligned} \tag{9.21}$$

In other words, the boundary tensor factors into the product of an angular component (a 2×2 matrix) and a radial component (a scalar). It should be noted that this decomposition is only possible because the boundary tensor is phase invariant. The angular integral $\mathbf{T}_{B,a}$ in (9.21) can be further simplified in terms of the Fourier coefficients of F_a:

$$\alpha_n = \int_0^{2\pi} \cos(n\phi) F_a(\phi)\, d\phi \qquad \beta_n = \int_0^{2\pi} \sin(n\phi) F_a(\phi)\, d\phi \tag{9.22}$$

It turns out that only the Fourier coefficients up to second order are relevant (the others drop out due to orthogonality of trigonometric functions), and the boundary tensor components can be written as (see [Köthe 06a] for details):

$$\begin{aligned} \mathbf{T}_{B,11} &= (\alpha_1^2 + \frac{1}{4}(\alpha_0 + \alpha_2)^2 + \frac{1}{4}\beta_2^2)\, \mathbf{T}_{B,r} \\ \mathbf{T}_{B,22} &= (\beta_1^2 + \frac{1}{4}(\alpha_0 - \alpha_2)^2 + \frac{1}{4}\beta_2^2)\, \mathbf{T}_{B,r} \\ \mathbf{T}_{B,21} = \mathbf{T}_{B,12} &= (\alpha_1\beta_1 + \frac{1}{2}\alpha_0\beta_2)\, \mathbf{T}_{B,r} \end{aligned} \tag{9.23}$$

where $\mathbf{T}_{B,r}$ is the (scalar) radial part of (9.21). These equations show what type of features the boundary tensor is sensitive to: The radial part $\mathbf{T}_{B,r}$ assumes high values if the image contrast is high near the point of interest, and the angular components $\mathbf{T}_{B,a}$ are high if the neighborhood around that point is well described by the first five Fourier coefficients. That is, when we draw a circle around the point of interest whose radius corresponds to the scale of the bandpass filter $H(|\vec{u}|)$, the intensity variations along this circle should be well described by the Fourier coefficients α_0 to β_2. This makes it immediately clear why the boundary tensor is a richer feature descriptor than the gradient magnitude: The latter is only sensitive to the two Fourier coefficients α_1 and β_1 (corresponding to step edge-like image structures, i.e. features of odd symmetry).

Figure 9.6 shows example for the performance of the boundary tensor on some junction configurations. The depicted intensity profiles demonstrate that five Fourier coefficients are sufficient for coarse approximation of the true intensity, but details (such as narrow regions and staircase-like profiles) are lost. Consequently, the boundary energy represents the structure very well (center right). The small eigenvalue of the boundary tensor (right), which encodes to what degree the structure is intrinsically 2-dimensional, has maxima in the neighborhood of the junctions, but not necessarily at the precise junction point. This reflects the limitations imposed by the five-coefficient approximation of the profile.

9.3.2 Efficient Computation of the Boundary Tensor

Thanks to the convolution theorem of Fourier theory, the filter responses h^{odd} and h^{even} can always be computed in the Fourier domain, and the boundary tensor is then obtained by point-wise combinations of the spatial domain responses after inverse Fourier transform. However, for small filters this may not be the most efficient method since it requires one forward and 5 inverse Fourier transforms. An algorithm working purely in the spatial domain may be much faster then. Let the bandpass filter $H(|\vec{u}|)$ be the Laplacian of Gaussian. Its transfer function is

$$H(|\vec{u}|, \sigma) = -4\pi^2 |\vec{u}|^2 e^{-2\pi^2 |\vec{u}|^2 \sigma^2} \tag{9.24}$$

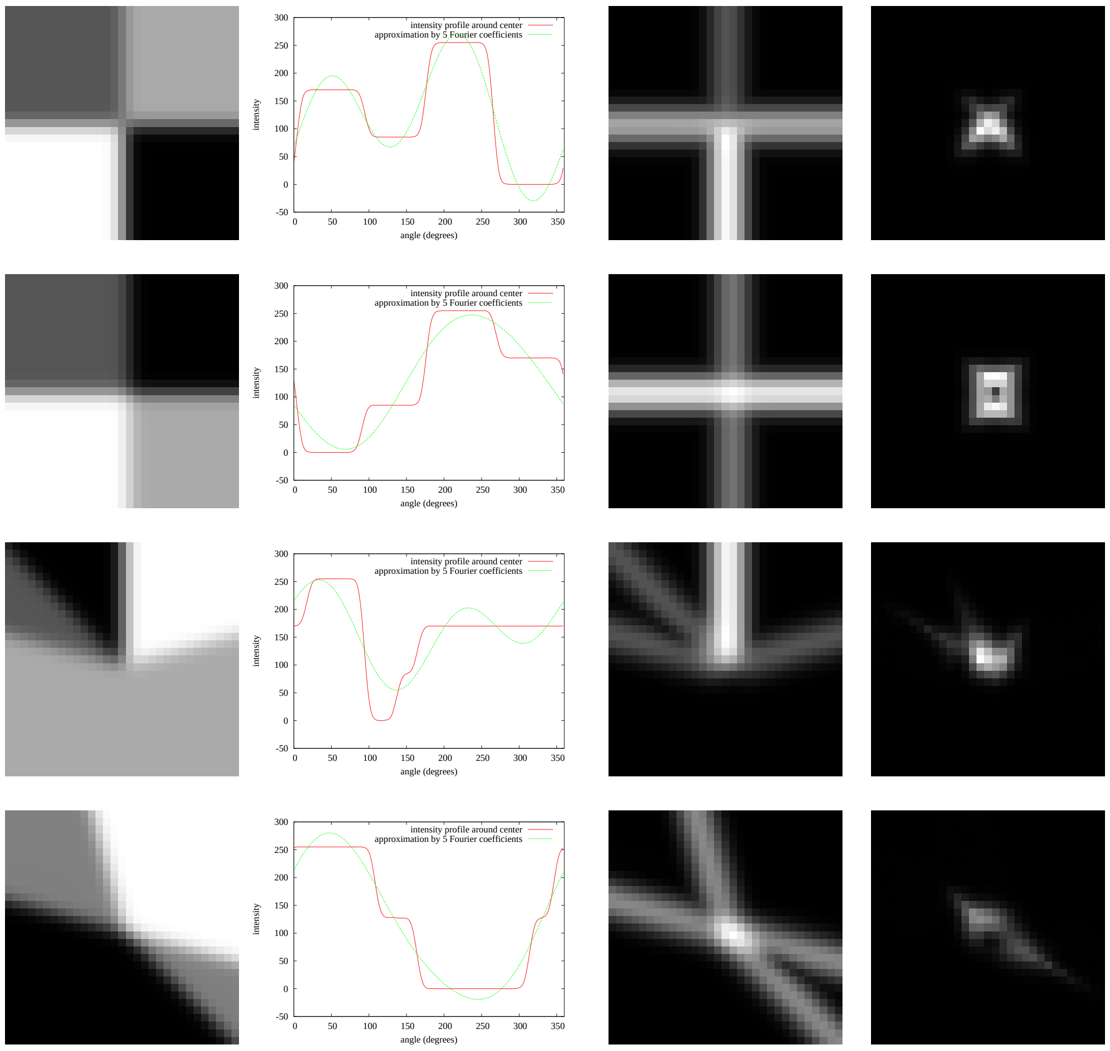
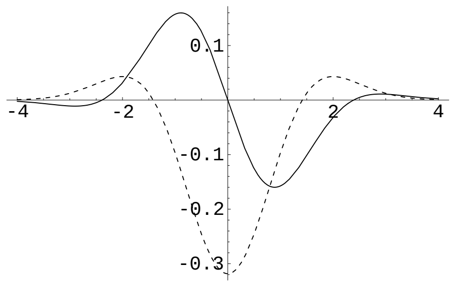


Figure 9.6: Boundary tensor response for some junction configurations (left). Center left: the intensity profile around the center of the configuration (red) and its approximation by the five Fourier coefficients that the boundary tensor actually sees (green). Center right and right: Boundary energy and small eigenvalue of the boundary tensor.



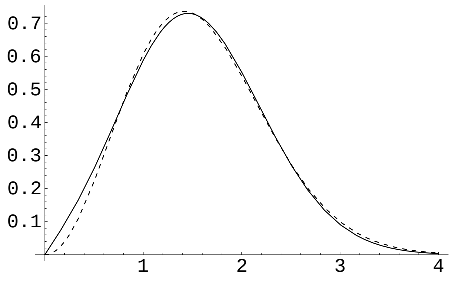


Figure 9.7: Left: h_1^{odd} (solid) and h_{11}^{even} (dashed) along the x_1 axis when the band-pass is the Laplacian of Gaussian at $\sigma = 1$ (the approximation $\tilde{h}_1^{\mathrm{odd}}$ is indistinguishable form the exact function h_1^{odd} in the depicted 4σ interval, it only has longer tails). Right: transfer functions of H (dashed) and its approximation $\tilde{H}_1^{\mathrm{odd}} = \mathcal{F}[\tilde{h}_1^{\mathrm{odd}}]$ (solid) at $\sigma = 1$.

We have found experimentally that this band-pass gives very good feature resolution (good separation of nearby features) and is better than other choices in this respect, which is probably due to the Gaussian's optimal localization in both the spatial and frequency domains. Moreover, the resulting spatial domain filters can be efficiently computed by means of separable Gaussian-like filters. This is immediately obvious for the second-order Riesz transform of this filter, whose transfer function reads

$$\begin{aligned} H^{\mathrm{even}}(\vec{u},\sigma) &= -4\pi^2 \begin{pmatrix} \cos^2\phi & \sin\phi\cos\phi \\ \sin\phi\cos\phi & \sin^2\phi \end{pmatrix} |\vec{u}|^2 e^{-2\pi^2|\vec{u}|^2\sigma^2} \\ &= -4\pi^2 \begin{pmatrix} u_1^2 & u_1u_2 \\ u_1u_2 & u_2^2 \end{pmatrix} e^{-2\pi^2\left(u_1^2+u_2^2\right)\sigma^2} \end{aligned}$$

which we readily recognize as the Fourier transform of the Hessian matrix of the Gaussian. The corresponding spatial domain filters are therefore simply the second derivatives of the Gaussian:

$$h_{il}^{\mathrm{even}}(\vec{x},\sigma) = \frac{x_i x_l - \sigma^2\delta_{il}}{2\pi\sigma^6} \exp\left(-\frac{x_1^2+x_2^2}{2\sigma^2}\right)$$

The transfer function of the odd filters (first order Riesz transform) becomes

$$H^{\mathrm{odd}}(\vec{u},\sigma) = -4\pi^2 \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} |\vec{u}|^2 e^{-2\pi^2|\vec{u}|^2\sigma^2}$$

The corresponding spatial domain expression is not so simple

$$h_i^{\mathrm{odd}}(\vec{x},\sigma) = \frac{x_i}{4\sqrt{2\pi}\sigma^7} \exp\left(-\frac{|\vec{x}|^2}{4\sigma^2}\right) \left((|\vec{x}|^2-3\sigma^2) I_0\left(\frac{|\vec{x}|^2}{4\sigma^2}\right) - (|\vec{x}|^2-\sigma^2) I_1\left(\frac{|\vec{x}|^2}{4\sigma^2}\right)\right)$$

where I_0 and I_1 are modified Bessel functions of the first kind. Fig. 9.7 left depicts the shape of h_1^{odd} and h_{11}^{even} along the x_1 axis. Unfortunately, the exact spatial domain expressions of the odd kernels are unsuitable for practical applications because their asymptotic decay is only of order $\mathcal{O}(|\vec{x}|^{-4})$, and the filters are not Cartesian separable. This means that large 2-dimensional filter masks are needed, which makes computation of h_i^{odd} very slow. Therefore, we apply a design technique similar to the one used for steerable quadrature filters [Freeman & Adelson 91] to approximate h_i^{odd} with filters $\tilde{h}_i^{\text{odd}}$ that can be computed separably and decay exponentially. The idea is to realize $\tilde{h}_i^{\text{odd}}$ as sums of filters that are third order polynomials times a Gaussian. The polynomials-times-Gaussian are defined so that they together form a supersymmetric third order tensor filter $\tilde{h}_{ikl}$ (a supersymmetric tensor has the property that its components don't change under permutation of indices, i.e $\tilde{h}_{112} = \tilde{h}_{121} = \tilde{h}_{211}$ etc.). Then the first order tensor filters can be obtained by contraction over any pair of indices, i.e. $\tilde{h}_i^{\text{odd}} = \sum_k \tilde{h}_{ikk}$. We make the following ansatz:

$$\begin{aligned}
\tilde{h}_{iii}(\vec{x},\sigma') &= \left(\frac{s\,x_i^3}{\sigma'^5} + \frac{t\,x_i}{\sigma'^3}\right)\frac{1}{2\pi\sigma'^2}\exp\left(-\frac{x_1^2+x_2^2}{2\sigma'^2}\right) \\
\tilde{h}_{ill}(\vec{x},\sigma') &= \frac{x_i}{\sigma'^2}\left(\frac{s\,x_l^2}{\sigma'^3} + \frac{t}{3\sigma'}\right)\frac{1}{2\pi\sigma'^2}\exp\left(-\frac{x_1^2+x_2^2}{2\sigma'^2}\right) \qquad (i \neq l)
\end{aligned}$$

By expressing these functions in a rotated coordinate system, it is easy (if tedious) to verify that the tensor requirements (9.1) are fulfilled with $p = 3$. The transfer function of the resulting filters $\tilde{h}_i^{\text{odd}}$ is

$$\tilde{h}_i^{\text{odd}} = \sum_k \tilde{h}_{ikk} \;\circ\!\!-\!\!\bullet\; \tilde{H}_i(\vec{u},\sigma') = 8\pi^3\frac{u_i}{\sigma'}\left(s(4-|\vec{u}|^2\sigma'^2) + \frac{4t}{3}\right)e^{-2\pi^2|\vec{u}|^2\sigma'^2} \qquad (9.25)$$

We now formulate a least squares problem to choose s, t, and σ' so that the radial part of $\tilde{H}_i(\vec{u},\sigma')$ becomes as similar as possible to the desired $H(\vec{u},\sigma)$:

$$\text{minimize w.r.t. } s,t,\sigma' : \quad \int\left(\tilde{H}(|\vec{u}|,\sigma') - H(|\vec{u}|,\sigma)\right)^2 d\vec{u}$$

where $\tilde{H}(|\vec{u}|,\sigma')$ is obtained from $\tilde{H}_i(\vec{u},\sigma')$ by replacing u_i with $|\vec{u}|$. This optimization problem is similar to the one in [Freeman & Adelson 91], but we compute the solution in the 2-dimensional Fourier domain, and include σ' in the optimization. Therefore, our approximation will be significantly better. The solution is

$$s = -0.5589, \qquad t = 2.0425, \qquad \sigma' = 1.0818\,\sigma \qquad (9.26)$$

Fig. 9.7 right depicts $\tilde{H}$ and H. It can be seen that the approximation is very good. Thus, the boundary tensor can be computed by means of 7 separable, Gaussian derivative-like filters. This can be compared with the structure tensor, where 2 filters are needed to compute the gradient tensor, but then 3 filters at a larger (typically doubled) scale are applied to integrate the gradient tensors over a neighborhood. Thus, while the number of filters to compute the structure tensor is lower (5 vs. 7), larger windows are required, making the overall computational effort about equal.

An even simpler approximation of the boundary tensor in terms of Gaussian derivative filters is possible by computing the *gradient energy tensor* or *GET operator* according to [Köthe & Felsberg 06, Felsberg & Köthe 05]. Let

$$\mathbf{a} = \nabla g_\sigma \star f$$

be the Gaussian gradient of the image f at scale $\sigma_1 = \sigma$. Then,

$$\mathbf{B} = \left(\nabla \nabla^T\right) g_\sigma \star g_\sigma \star f$$

is the Hessian matrix of the image at scale $\sigma_2 = \sqrt{2}\sigma$. Finally,

$$\mathbf{c} = \nabla \left(\left(\nabla^T \nabla\right) g_\sigma \star g_\sigma \star g_\sigma \star f\right)$$

is the gradient of the Laplacian of Gaussian at scale $\sigma_3 = \sqrt{3}\sigma$. Note that **a**, **B**, and **c** can be efficiently computed by repeated application of a first derivative of Gaussian filter at a fixed scale σ. It is even possible to approximate the first derivative by a simple Sobel filter. Then, the gradient energy tensor is defined as

$$\mathbf{T}_{GET} = \mathbf{B}\,\mathbf{B}^T - \frac{1}{2}\left(\mathbf{a}\,\mathbf{c}^T + \mathbf{c}\,\mathbf{a}^T\right) \tag{9.27}$$

It was shown in [Köthe & Felsberg 06] that $\mathbf{T}_{GET}$ is a good approximation of the boundary tensor $\mathbf{T}_B$ when the latter is computed on the basis of the Laplacian of Gaussian bandpass at scale $\sqrt{2}\sigma$. The main difference is that $\mathbf{T}_B$ is guaranteed to be positive semi-definite, whereas no such guarantee can be given for $\mathbf{T}_{GET}$. However, in practice this doesn't seem to be a big problem – when $\mathbf{T}_{GET}$ has negative eigenvalues, they can simply be clamped at zero, because they indicate that there is only weak image structure in one or all directions.

9.4 Experimental Evaluation

In order to find out whether the proposed tensor methods are indeed consistently better near junctions we apply them to the same test images as have been used in section 7.6, that is to artificial junction images of degree 3 and 4. The boundary detector in these experiments is the subpixel watershed algorithm, applied to the trace of the tensor response of the different filter methods. This choice has the advantage that it facilitates direct comparison of the resulting boundaries. On the other hand, it doesn't make use of the complete tensor information – it only uses the sum of the eigenvalues, but not their difference or the eigenvector orientation. However, it is unclear whether this information would improve results significantly, and convincing boundary detection algorithms that incorporate complete tensor information don't seem to exist as yet. One reason for this may be the fact that the tensor eigenvalues encode only one independent direction, but advanced junction representations would need at least two independent directions. We conjecture that 2×2 tensors will not be able to achieve significantly better results than will be presented below.

Figure 9.8 demonstrates that it is possible to get better results from tensor-based methods than are obtained from the gradient magnitude. However, it can also be seen that no single algorithm is consistently better than the gradient: When we sort results by increasing error, the order is different for each junction configuration, and there are even cases where the plain gradient magnitude wins by a large margin (figure 9.8 right).

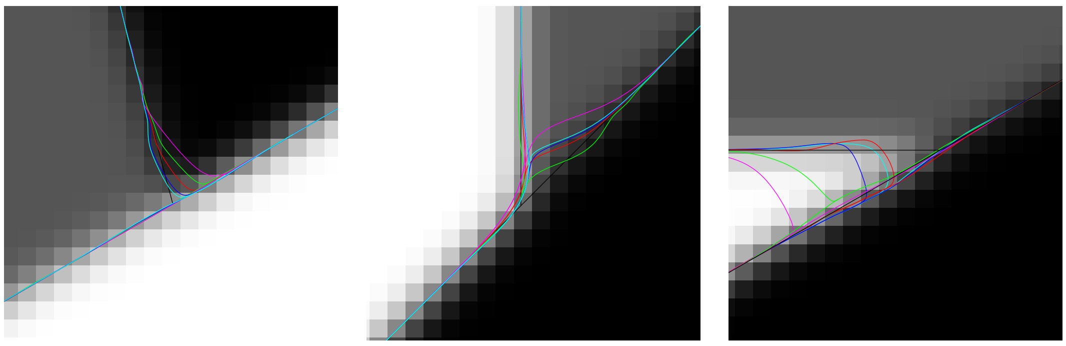


Figure 9.8: Comparison of tensor-based boundary detection with the gradient magnitude at T-junctions (black: ground truth, red: gradient magnitude, magenta: structure tensor, green: boundary tensor, blue: anisotropic structure tensor, cyan: hourglass tensor). Parameters of images: $\sigma_{\mathrm{PSF}} = 1$, $\mathrm{SNR} = 100$; of gradient magnitude: $\sigma_{\mathrm{filter}} = 1.2$; of structure tensor, equation (9.3): $\sigma_{\mathrm{filter}} = 1$, $\sigma_i = 2$; of boundary tensor (9.24): $\sigma = 1.2$; of anisotropic structure tensor (9.8): $\sigma_{\mathrm{filter}} = 1$, $\sigma_1 = 2$, $\sigma_2 = 0.7$; of hourglass kernel (9.4): $\sigma_{\mathrm{filter}} = 1$, $\sigma_i = 2$, $\sigma_\phi = 0.4$. The results of filters according to (9.6) and (9.7) are very similar to the hourglass result and therefore not shown.

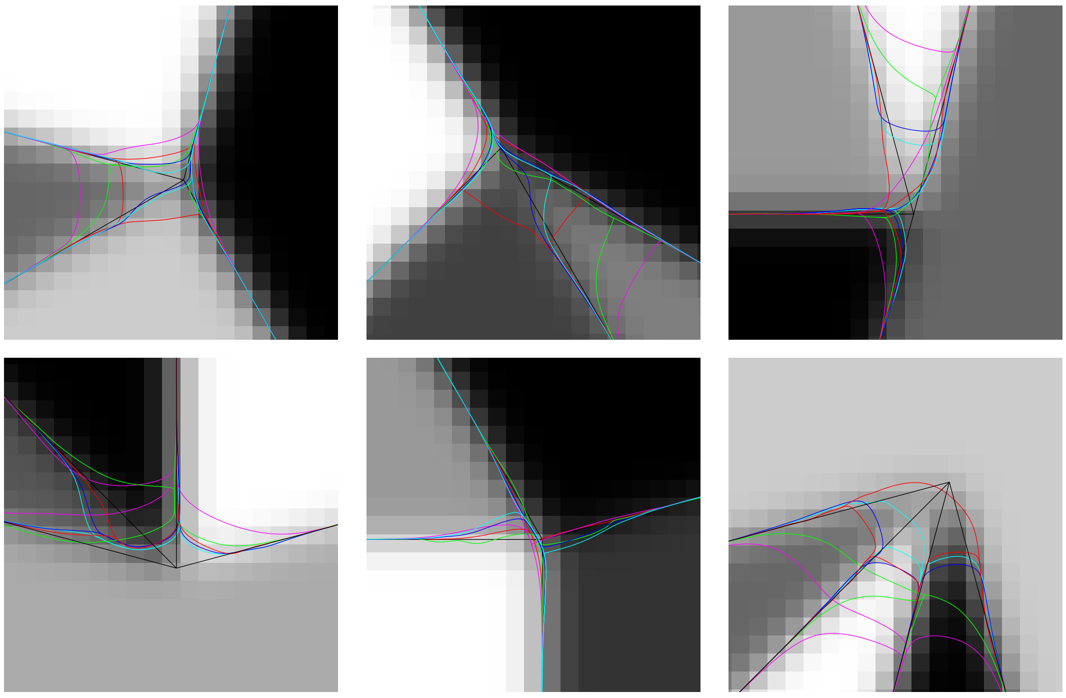


Figure 9.9: Comparison of tensor-based boundary detection with the gradient magnitude at junctions of degree 4 (black: ground truth, red: gradient magnitude, magenta: structure tensor, green: boundary tensor, blue: anisotropic structure tensor, cyan: hourglass tensor). Parameters of images and operators are as in figure 9.8.

	gradient magnitude		structure tensor		boundary tensor		anisotropic structure tensor		hourglass tensor	
	p $[\bar{p}]$	q $[\bar{q}]$	p $[\bar{p}]$	q $[\bar{q}]$	p $[\bar{p}]$	q $[\bar{q}]$	p $[\bar{p}]$	q $[\bar{q}]$	p $[\bar{p}]$	q $[\bar{q}]$
straight lines SNR = 100	0.21 [0.024]	0.046 [0.011]	0.22 [0.026]	0.18 [0.010]	0.31 [0.027]	0.12 [0.013]	0.24 [0.028]	0.16 [0.014]	0.30 [0.03]	0.20 [0.014]
straight lines SNR = 10	0.65 [0.14]	0.65 [0.14]	0.67 [0.092]	0.34 [0.070]	1.4 [0.25]	1.5 [0.40]	0.49 [0.11]	0.49 [0.10]	0.50 [0.12]	0.50 [0.11]
T-junctions SNR = 100	2.5 [0.10]	2.3 [0.04]	5.3 [0.32]	3.2 [0.06]	4.3 [0.17]	2.6 [0.05]	2.8 [0.15]	2.0 [0.06]	2.9 [0.15]	1.8 [0.05]
X-junctions SNR = 100	3.5 [0.17]	2.5 [0.08]	8.6 [0.47]	3.3 [0.13]	5.2 [0.30]	3.3 [0.11]	3.6 [0.19]	2.3 [0.09]	4.5 [0.19]	2.0 [0.08]

Table 9.1: Maximum errors p and q for various boundary detectors and image features. Corresponding average errors are given in brackets. The minimum angle between adjacent edges in the junction configurations was 30°.

It is also not uncommon for the different detectors to produce very similar results (not shown). Figure 9.9 shows that the situation is the same for junctions of degree 4.

In order to get insight into the quality of tensor-based results beyond mere examples, we make again use of the boundary sampling theorem 6.8. It states that the quality of a boundary detector should be judged by the maximum errors p (maximum distance of a ground truth point to the nearest detected boundary point) and q (maximum distance of a detected boundary point to the ground truth). In table 7.3 we reported that gradient-based boundary detectors have maximum errors in the order of 3 pixels at junctions. On the other hand, the step edge response of these detectors is very accurate, depending on the noise level. So it is of interest whether tensor-based methods improve the junction response without worsening the edge response. Since theoretical error bounds similar to the ones we derived for gradient-based methods are not yet available for tensor-based methods, we performed this analysis experimentally on a set of several hundred test images generated according to equation (2.3) with randomly selected parameters.

The results are shown in table 9.1. Tensor methods can indeed achieve lower errors. For example, the q-value of isotropic and anisotropic structure tensors (including the hourglass tensor) is significantly lower for noisy straight edges, because the tensor integration effectively suppresses noise. As one would expect, this effect is biggest for isotropic integration. Near junctions, the anisotropic structure tensor and the hourglass tensor perform slightly better than the gradient magnitude (at least in terms of $p+q$), but the improvement is small and may not always be worth the much higher computational effort. The errors of the boundary tensor for noisy edges are very high which turned out to be a consequence of the fact that it was impossible to find a threshold that removed all false positives without removing true positives. The boundary tensor is therefore more susceptible to noise than the gradient. This is easy to understand: At the location of the step edge, the signal response of the even filters is zero due to the edges anti-symmetry. Therefore, the even filters only contribute noise in this case. If we assume that the noise behavior of the odd filter part is similar to that of the gradient magnitude, the total noise content of the boundary tensor trace must clearly be higher.

10 Conclusions and Outlook

In this work we explored whether it is possible to derive low-level segmentation methods with predictable performance. We can summarize our findings as follows:

1. We conducted a very careful error analysis for low-level segmentation and achieved remarkably good agreement between theoretical predictions and actually observed performance.

2. A number of factors critically contribute to this success:

   a) We defined low-level image segmentation as a subsystem whose goal has to be well distinguished from that of high-level segmentation: Low-level segmentation is concerned with the reconstruction of properties of an ideal geometric image from a real digital image. This point of view facilitates the definition of realistic measures of success. In contrast, recovery of actual object boundaries would ask too much from a low-level subsystem, and corresponding performance targets would be impossible to achieve.

   b) It is insufficient to work in either the continuous or discrete domain alone. Rather, it is necessary to treat these domains as complements of each other, and to switch between corresponding representations as need arises. This fact must be reflected by image acquisition and boundary detection models. When algorithms are genuinely discrete, their relationship to the (analog) real world must be explicitly established. Likewise, when algorithms are defined in the continuous domain, discretisation must be an intrinsic element of the algorithm design and analysis. Oversampling of the boundary indicator (section 7.1) and subpixel-accurate boundary detection (section 5.1) have been identified as highly effective methods arising from this point of view.

   c) Algorithms and data structures must be powerful enough to take advantage of the complete information contained in the image data. This includes topological relations and subpixel-accurate geometric information. Spline interpolation and the GeoMap framework have proved to be highly powerful tools in this respect.

3. Many well-known image segmentation algorithms have successfully been transformed into our proposed framework, often by replacing certain heuristics with concepts that serve the same purpose but lend themselves to formal performance analysis. Examples include edgel linking in the GeoMap framework instead of traditional heuristic edgel linking, and thresholding of gradient magnitudes only after noise-normalization.

4. Understanding the relationships between sensor resolution, lens blurring, noise and target object shape/size is critical to the success of low-level segmentation. We were able to formalize these relationships in a number of new geometric sampling theorems which can be considered as key results of this work. These theorems formulate sufficient conditions which guarantee that ground truth topology and geometry are correctly recovered within given error bounds. If an image analysis task does not meet these conditions, higher resolution or additional information (e.g. shape constraints) are required for reliable segmentation.

5. By means of very careful experimentation we found that complicated methods do not consistently outperform simple methods. This applies to both initial boundary detection, and derived measures such as the tangent angle.

The last observation is especially interesting, because it matches the practical experience of many application developers, whereas publications on new algorithms regularly suggest the opposite. In part, this contradiction can be attributed to the fact that different applications pose different requirements on algorithms. But our experience during the preparation of this work also points out another reason: It is extremely difficult to conduct really objective comparisons of different low-level approaches. This is not only a technical problem, but also a psychological one. On the technical side, there are two difficulties:

1. The definition of representative ground truth for low-level image analysis is difficult, because the geometric image (which is the reference of comparison) is generally unknown. We have discussed this problem in section 2.2.2, but do not consider it solved. In this work, we based algorithm comparisons on relatively simple natural images with known properties, and on realistic artificial images, and found quite good agreement between these image types.

2. Methods to be compared must be implemented so that they only differ in a single respect, whereas everything else should be equal. For example, when we compare the quality of non-maxima suppression, we have to make sure that the algorithms apply the same filters for boundary indicator computation (e.g. the Gaussian gradient at a particular scale), use the same precision for intermediate results (e.g. single-precision floating point numbers), and so on. Otherwise, it would never be exactly clear which algorithm detail caused the observed performance differences. To fulfill these requirements, we did not use algorithm implementations from different sources, but implemented everything from scratch in our unified framework, so that algorithms share all code except for the one implementing the feature to be tested.

On the psychological side, the problems are even more subtle: It is all too easy for our unconscious brain to slightly bias the experimental design so that our favorite algorithm (e.g. the one we just invented, or the one that took the most work to implement) is given some hidden advantage: The set of test images may slightly favor that algorithm, a little less care may be used in the implementation and testing of competing algorithms, certain negative results may not be presented in the paper, and so on. We have done the utmost to avoid this in the present work, but the best strategy for the future might be to assign the roles of algorithm designer/implementer and experimenter to different persons, similar to what is done in other fields like physics.

Our results suggest that two algorithms (which also appear to be the most popular among practitioners) offer the best performance: the watershed transform, and Canny's algorithm. Both work well on the Gaussian gradient, but can easily be adapted to other boundary indicators. The watershed transform is slightly more flexible, because it only needs a scalar boundary indicator, whereas Canny's algorithm also requires local orientations. The geometric accuracy of both algorithms is similar. Moreover, both algorithms are available in subpixel-accurate versions, which makes them suitable for low-resolution imagery. Again, the subpixel watershed algorithm has a slight advantage here, because it is able to place points along the contour with any desired spacing, whereas Canny's algorithm is restricted to one boundary point per pixel. Here, research should be directed towards a subdivision scheme that allows insertion of additional points into the Canny boundary without loosing geometric accuracy.

In terms of topological correctness, the two algorithms exhibit opposite behavior: The watershed transform tends to produce oversegmentation, whereas Canny's algorithm produces undersegmentation. In other words, Canny's algorithm returns few false positives, but edges have gaps, especially near junctions. We presented (α, β)-reconstruction (cf. section 5.3) as a well-defined way to close these gaps. Unfortunately, the values for α and β required for reliable gap closure are relatively large, so that resolution is lost at other locations in the image where the accuracy is significantly higher (e.g. at parallel edges). Therefore, the development of boundary detectors with improved junction behavior should be a high research priority. The proposals discussed in chapter 9, while promising from a theoretical perspective, have not lived up to expectations in experiments. The same can be said about other ideas from the literature that are not reviewed in detail here. Obviously, the junction problem is more difficult than one would think, and further research might just as well discover a formal proof that a purely low-level solution of this problem is impossible.

In the context of GeoMap creation we found it highly desirable that the underlying boundary detector produces closed boundaries. Closed boundaries allow us to compute region properties that can be used in subsequent computations to improve the segmentation or derive higher-level object properties. In contrast to boundary indicator on the basis of local filtering, region properties are adapted to the geometry of the image partition. The watershed transform has again a slight advantage here, because it creates closed boundaries, although at the price of oversegmentation. In contrast, the edge gaps typical for Canny's algorithm lead to undersegmentations without meaningful regions. We are convinced that it is easier to merge regions in an oversegmentation than to close gaps in an undersegmentation, and methods taking advantage of region properties are currently under active development.

We would like to point out the following open problems that might be the focus of future research:

1. We already mentioned the necessity to improve the junction response, or to prove the impossibility of this endeavor. This problem can be approached either by means of an improved boundary indicator, or by the combination of several boundary indicators. A major difficulty is to ensure that advances near junctions are not achieved at the cost of reduced boundary fidelity away from junctions.

2. False positives or negatives are often caused by image features that are beyond the boundary indicator's capabilities. This especially applies to shaded and textured regions. We had seen in chapter 7 that smooth shading tends to cause undersegmentation, whereas textures will usually lead to oversegmentation. Boundary indicators dealing with these region properties have been proposed, but have not yet reached the same level of understanding as the more basic boundary indicators.

3. The reliability analysis should be extended to other boundary definitions, especially those based on the minimization of certain energy functionals. While these methods are guaranteed to find a local or even global optimum of the energy functional, the relationship between this optimum and the ground truth segmentation is unclear. Formal proofs showing when an energy optimum will reproduce the desired ground truth segmentation do not yet seem to exist.

4. The representation of initial segmentation results in a GeoMap allows the application of post-processing methods that take advantages of the graph-like GeoMap structure. These methods can make use of non-local image features such as the ones suggested by gestalt theory, or statistics of extended, irregularly shaped regions, which cannot be computed by local boundary indicators. While certain results on such methods exist (e.g. in the context of the waterfall algorithm and irregular pyramids), the full potential of this approach has not yet been realized.

5. We believe that the scope of geometric sampling theorems is not restricted to boundary detection. It would be interesting to develop similar theorems for other analysis modalities such as motion and object recognition. These theorems could, for example, tell us whether the resolution of an image and the accuracy of the extracted features are sufficient for determining which object (out of a given set of candidates) is visible in an image. Results like this would be very helpful in deciding which feature set should be applied in a given object recognition task.

6. If the performance of purely bottom-up methods is insufficient for reliable image analysis, additional information (such as shape priors or hypotheses about the objects to be seen) has to be utilized. The additional information should compensate for the deficits of low-level image analysis, but should never override valid stimulus data. Further research is required to determine the optimal balance between visual input data and information outside the stimulus.

7. Many of the ideas presented in this work can be generalized to 3- and higher-dimensional problems. However, this generalization is not straightforward for subpixel-accurate boundary tracing methods (subpixel watersheds and subpixel zero-crossings),

because these methods rely on a linear ordering of the boundary points. A surface in 3-dimensional space has to be traced along two independent directions, which is much more difficult, especially when the surface has an irregular shape. Here, subdivision methods are to be preferred, but a detailed formal performance analysis of these methods seems to be an open problem.

11 Acknowledgments

This thesis could not have been written without the help of many people, and I gratefully acknowledge their support. First and foremost I'd like to thank Prof. Hans Siegfried Stiehl for his continued encouragement and interest in the topics and problems of this thesis, and for pointing out shortcomings of existing approaches which motivated many of the proposed solutions. I'm also most grateful to Prof. Bernd Neumann, the head of the Cognitive Systems Group, for creating and maintaining the creative atmosphere that made this work possible, and for giving me the freedom and resources required to successfully pursue my research.

Peer Stelldinger and Hans Meine have always been close collaborators and good friends, and I'm convinced that this thesis could not exist in its present form without them. Our discussions were always very inspiring and great fun. The collaboration was often so tight that it was hard to tell precisely which idea had been who's. At any rate, I admire Peer's ability to find relevant literature from seemingly unrelated fields and to come up with novel definitions that later give rise to beautiful theorems. Likewise, Hans used his outstanding software design skills to craft much of the infrastructure that allowed me to conduct experiments in whose results I can trust.

I'd like to thank all students – in particular Matthias Bock, Nils Boetius, Alexander Bugl, Joshua Buttkus, Jörn Heinemeier, Florian Heinrich, Rainer Herzog, Verena Kaynig, Gunnar Kedenburg, Yevgen Reznichenko, Benjamin Seppke, Leonid Tcherniavski, and Andreas Tyart – who have been adventurous enough to try out some of my ideas. I always learned a great deal from their work, and their results significantly shaped my own thinking. Teaching is the best way to learn, as the saying goes. They and many others, especially Kasim Terzic, helped implementing improvements and extensions to the VIGRA framework which were instrumental to my research.

I'm grateful to Prof. Tony Lindeberg and Prof. Gösta Granlund for their hospitality, support and valuable discussions during my visits to their labs. They and their coworkers – most notably Michael Felsberg and Klas Nordberg – introduced me to important new concepts like Riesz transforms, energy operators and channel representations, and I had the honor of many fruitful discussions that resulted in a number of joint publications.

Finally, I'm deeply indebted to my family for their support (not the least in proof-reading the manuscript) and for their patience during countless hours of thinking, experimenting, and writing.

Bibliography

[Aach et al. 06] T. Aach, C. Mota, I. Stuke, M. Mühlich, E. Barth: *"Analysis of superimposed oriented patterns"*, IEEE Trans. on Image Processing, 15(12):3690-3700, 2006

[Abramowitz & Stegun 72] M. Abramowitz, I. Stegun: *"Handbook of Mathematical Functions"*, Dover, 1972

[Ahronovitz et al. 95] E. Ahronovitz, J.P. Aubert, C. Fiorio: *"The Star Topology: a Topology for Image Analsysis"*, in: Proc. 5th Intl. Conf. Discrete Geometry for Computer Imagery (DGCI 1995), pp. 101-116, 1995

[Allebach 05] J.P. Allebach: *"Image Scanning, Sampling, and Interpolation"*, in: [Bovik 05], pp. 629-644, 2005

[Allgower & Georg 97] E.L. Allgower, K. Georg: *"Numerical path following"*, In: P.G. Ciarlet, J.L. Lions (Eds.), Handbook of Numerical Analysis, volume 5, pp. 3-207, North-Holland, 1997

[Andersen & Kim 85] T.A. Andersen, C.E. Kim: *"Representation of digital line segments and their pre-images"*. Computer Vision, Graphics, and Image Processing (CVGIP), 30:279-288, 1985

[Artal & Navarro 94] P. Artal, R. Navarro: *"Monochromatic modulation transfer function of the human eye for different pupil diameters: an analytical expression"*, J. Optical Society of America A, 11(1):246-249, 1994

[August & Zucker 03] J. August, S. Zucker: *"Sketches with Curvature: The Curve Indicator Random Field and Markov Processes"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 25(4):387-400, 2003

[Baddeley 92] A.J. Baddeley: *"An error metric for binary images"*, in: W. Förstner, S. Ruwiedel (Eds.): Robust Computer Vision: Quality of Vision Algorithms, pp. 59-78, Karlsruhe: Wichmann Verlag, 1992

[Baker & Nayar 99] S. Baker, S. Nayar: *"Global Measures of Coherence for Edge Detector Evaluation"*, in: CVPR'99, Proc. of IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, pp. 373-379, Los Alamitos: IEEE Computer Society Press, 1999

[Beckmann & Legge 02] P. Beckmann, G. Legge: *"Preneural limitations on letter identification in central and peripheral vision"*, J. Optical Society of America A, 19(12):2349-2362 , 2002

[Belongie et al. 02] S. Belongie, J. Malik, J. Puzicha: *"Shape Matching and Object Recognition Using Shape Contexts"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 24(4):509-522, 2002

[Bern & Eppstein 92] M. Bern, D. Eppstein: *"Mesh Generation and Optimal Triangulation"*, in: D.-Z. Du, F. Hwang (Eds.): Computing in Euclidean Geometry, Lecture Notes Series on Computing, vol. 1, pp. 23-90, Singapore: World Scientific, 1992

[Bernardini & Bajaj 97] F. Bernardini, C.L. Bajaj: *"Sampling and Reconstructing Manifolds Using Alpha-Shapes"*, Proc. 9th Canadian Conf. Computational Geometry, 1997

[Bertram et al. 00] M. Bertram, M.A.Duchaineau, B. Hamann, K.I. Joy: *"Bicubic Subdivision-Surface Wavelets for Large-Scale Isosurface Representation and Visualization"*, in: Proc. IEEE Visualization 2000, pp. 389-396, Los Alamitos: IEEE Computer Society Press, 2000.

[Bertrand et al. 99] Y. Bertrand, C. Fiorio, Y. Pennaneach: *"Border Map: a Topological Representation for nD Image Analysis"*, in: G. Bertrand, M. Couprie, L. Perroton (Eds.): Proc. 8th Intl. Conf. Discrete Geometry for Computer Imagery (DGCI 1999), Lecture Notes in Computer Science 1568, pp. 242-257, Berlin: Springer, 1999

[Beymer 91] D. Beymer: *"Finding Junctions Using the Image Gradient"*, in: Proc. CVPR'91, IEEE Conf. on Computer Vision and Pattern Recognition, pp. 720-721, 1991. Long version: MIT AI Lab Memo No. 1266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1991

[Bigün et al. 91] J. Bigün, G. Granlund, J. Wiklund: *"Multidimensional Orientation Estimation with Applications to Texture Analysis and Optic Flow"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 13(8):775-790, 1991

[Boetius 06] N. Boetius: *"Detektion salienter Kanten mit modernen Lernverfahren"*, Diploma thesis, Computer Science Department, University of Hamburg, 2006

[Boomgaard & Weijer 03] R. v.d. Boomgaard, J. v.d. Weijer: *"Least Squares and Robust Estimation of Local Image Structure"*, in: L. Griffin, M. Lillholm (Eds.): Scale Space Methods in Computer Vision, Proc. ScaleSpace 2003, pp. 237-254, Lecture Notes in Computer Science 2695, Heidelberg: Springer, 2003

[Bouma et al. 05] H. Bouma, A. Vilanova, L.J. van Vliet, F.A. Gerritsen: *"Correction for the Dislocation of Curved Surfaces Caused by the PSF in 2D and 3D CT Images"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 27(9):1501–1507, 2005

[Bovik 05] A. Bovik: (Ed.): *"Handbook of Image and Video Processing"*, Second Edition, Amsterdam: Elsevier, 2005

[Bracewell 78] R.N. Bracewell: *"The Fourier Transform and its Applications"*, New York: McGraw-Hill, 1978

[Braquelaire & Brun 98] J.-P. Braquelaire, L. Brun: *"Image segmentation with topological maps and interpixel representation"*, J. Visual Communication and Image Representation **9**(1):62-79, 1998

[Braquelaire & Domenger 99] J.-P. Braquelaire, J.-P. Domenger: *"Representation of Segmented Images with Discrete Geometric Maps"*, Image and Vision Computing, 17:715-735, 1999

[Braquelaire & Vialard 99] J.-P. Braquelaire, A. Vialard: *"Euclidean paths : A new representation of boundary of discrete regions"*, Graphical Models and Image Processing 61(1):16-43, 1999

[Braquelaire 05] A. Braquelaire: *"Representing and Segmenting 2D Images by Means of Planar Maps with Discrete Embeddings: From Model to Application"*, in: L. Brun, M. Vento (Eds.): Graph-Based Representations in Pattern Recognition, Proc. 5th IAPR Workshop GbRPR '05, Lecture Notes in Computer Science 3434, pp. 92-121, Springer, 2005

[Brice & Fennema 70] C. Brice, C. Fennema: *"Scene Analysis Using Regions"*, Artificial Intelligence 1(3), 205-226, 1970

[Brodzik 98] M.L. Brodzik: *"The Computation of Simplicial Approximations of Implicitly Defined p-Manifolds"*, Computers and Mathematics with Applications, 36(6):93-113, 1998.

[Brun & Domenger 97] L. Brun, J.-P. Domenger: *"A new split and merge algorithm with topological maps and inter-pixel boundaries"*, In: Proc. 5th Intl. Conf. in Central Europe on Computer Graphics and Visualization, 1997

[Brun et al. 98] L. Brun, J.-P. Domenger, J.-P. Braquelaire: *"Discrete maps: a framework for region segmentation algorithms"*, In: Proc. WS Graph-Based Representations in Pattern Recognition, pp. 83-92, Berlin: Springer, 1998

[Brun & Kropatsch 01] L. Brun, W. Kropatsch: *"Introduction to Combinatorial Pyramids"*, in: G. Bertrand, A. Imiya, R. Klette (Eds.): Digital and Image Geometry - Advanced Lectures, Lecture Notes in Computer Science 2243, pp. 17-37, Berlin: Springer, 2001

[Brun et al. 03] L. Brun, M. Mokhtari, J.-P. Domenger: *"Incremental modifications of segmented images defined by discrete maps"*, J. Visual Communication and Image Representation, 14(3):251-290, 2003

[Bugl & Heinemeier 04] A. Bugl, J. Heinemeier: *"Ein Rahmenwerk zur Evaluation von Kantendetektoren"*, Diploma thesis, Department of Informatics, University of Hamburg, 2004

[Campbell & Green 65] F.W. Campbell, D.G. Green: *"Optical and retinal factors affecting visual resolution"*, J. of Physiology, 181:576-593, 1965

[Campbell & Gubisch 66] F.W. Campbell, R.W. Gubisch: *"Optical Quality of the Human Eye"*, J. of Physiology, 186:558-578, 1966

[Canny 86] J. Canny: *"A Computational Approach to Edge Detection"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679-698, 1986

[Catmull & Rom 74] E.E. Catmull, R.J. Rom: *"A class of local interpolating splines",* in: R.E. Barnhill, R.F. Riesenfeld (Eds.): Computer Aided Geometric Design, pp. 317-326, Orlando: Academic Press, 1974

[Cayley 1859] A. Cayley: *"On contour and slope lines"*, The London, Edinghburgh and Dublin Philosophical Magazine and J. of Science, vol. 18, no. 120, pp. 264-268, 1859.

[Chan & Vese 01] T. Chan, L. Vese: *"Active contours without edges"*, IEEE Trans. Image Processing, 10(2):266-277, 2001.

[Chernov & Lesort 05] N. Chernov, C. Lesort: *"Least squares fitting of circles"*, J. of Mathematical Imaging and Vision, 23(3):239-252, 2005

[Chew 87] L.P. Chew: *"Constrained Delaunay triangulations"*, in: Proc. 3$^{\mathrm{rd}}$ Annual Symposium on Computational Geometry, pp. 215-222, New York: ACM Press, 1987

[Cormack 05] L.K. Cormack: *"Computational Models of Early Human Vision"*, in: [Bovik 05], pp. 325-346, 2005

[Curcio et al. 90] C.A. Curcio, K.R. Sloan, R.E. Kalina, A.E. Hendrickson: *"Human Photoreceptor Topography"*, J. Comparative Neurology, 292:497-523, 1990

[Damiand et al. 04] G. Damiand, Y. Bertrand, C. Fiorio: *"Topological model for two-dimensional image representation: definition and optimal extraction algorithm"*, Computer Vision and Image Understanding, 93(2):111-154, 2004

[Debled-Rennesson & Reveilles 95] I. Debled-Rennesson, J.-P. Reveilles: *"A linear algorithm for segmentation of digital curves"*, Intl. J. Pattern Recognition and Artificial Intelligence, 9(6):635-662, 1995

[Deriche 90] R. Deriche: *"Fast algorithms for low-level vision"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 1(12):78-88, 1990

[Deriche & Giraudon 93] R. Deriche, G. Giraudon: *"A computational approach for corner and vertex detection"*, Intl. Journal of Computer Vision, 10(2):101-124, 1993

[De Vriendt 95] J. De Vriendt: *"Effect of Sampling, Quantization, and Noise on the Performance of the Second Directional Derivative Edge Detector",* Multidimensional Systems and Signal Processing, **6:**37-68, 1995

[Di Battista et al. 99] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis: *"Graph Drawing"*, Prentice Hall, 1999

[Dibos & Knopfler 00] F. Dibos, G. Knopfler: *"Global total variation minimization"*, SIAM J. Numerical Analysis, 37(2):646-664, 2000

[Dorst & Smeulders 91] L. Dorst, A. Smeulders: *"Straight Line Segments: Parameters, Primitives and Properties"*, Contemporary Mathematics, **119:**45-62, 1991

[Duffieux 46] P.M. Duffieux: *"L'Intégral de Fourier et ses Applications à l'Optique"*, Rennes: Societé Anonyme des Imprimeries Oberthur, 1946. English translation: "The Fourier Transform and its Applications to Optics", Second Edition, New York: John Wiley & Sons, 1983

[Duford & Puitg 00] J.-F. Dufourd, F. Puitg: *"Functional specification and prototyping with oriented combinatorial maps"*, Computational Geometry 16:129-156, 2000

[Eberly 96] D. Eberly: *"Ridges in Image and Data Analysis"*, Dordrecht: Kluwer Academic Publishers, 1996

[Eberly 03] D. Eberly: *"Least-Squares Reduction of B-Spline Curves"*, `http://www.geometrictools.com/Documentation/BSplineReduction.pdf` (link checked 5.5.07), 2003

[Edelsbrunner & Mücke 94] H. Edelsbrunner, E.P. Mücke: *"Three-dimensional alpha shapes"*, ACM Trans. Graphics, **13**:43-72, 1994

[Edelsbrunner 95] H. Edelsbrunner: *"The union of balls and its dual shape"*, Discrete Comput. Geom., **13**:415-440, 1995

[Erhardt et al. 84] H.G. Erhardt, J. Kane, L.S. O'Hara: *"Silicon cylindrical lens arrays for improved photoresponse in focal plane arrays"*, in: SPIE vol. 501, Proc. State of the Art Imaging Arrays and Their Applications, pp. 165-172, 1984

[Farnebäck 02] G. Farnebäck: *"Polynomial Expansion for Orientation and Motion Estimation"*, PhD thesis, Linköping University, Dissertation No. 790, 2002

[Faugeras & Luong 01] O. Faugeras, Q.-T. Luong: *"The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications"*, MIT Press, 2001

[Felsberg & Sommer 01] M. Felsberg, G. Sommer: *"The Monogenic Signal"*, IEEE Trans. Image Processing, 49(12):3136-3144, 2001

[Felsberg & Köthe 05] M. Felsberg, U. Köthe: *"GET: The Connection Between Monogenic Scale-Space and Gaussian Derivatives"*, in: R. Kimmel, N. Sochen, J. Weickert (Eds.): Scale Space and PDE Methods in Computer Vision, Proc. of Scale-Space 2005, Lecture Notes in Computer Science 3459, pp. 192-203, Heidelberg: Springer, 2005

[Fiete 04] R.D. Fiete: *"Elements of Photogrammetric Optics"*, in: [McGlone 04], Chapter 4, pp. 317-398, 2004

[Fiorio 96] C. Fiorio: *"A topologically consistent representation for image analysis: the topological graph of frontiers"*, in: S. Miguet, A. Montanvert, S. Ubéda (Eds.): Proc. 6th International Conference on Discrete Geometry for Computer Imagery (DGCI 1996), Lecture Notes in Computer Science 1176, pp. 151-162, Berlin: Springer, 1996

[Fischler & Bolles 81] M.A. Fischler, R.C. Bolles: *"Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography"*, Comm. of the ACM, **24**:381-395, 1981

[Förstner 86] W. Förstner: *"A Feature Based Correspondence Algorithm for Image Matching"*, Intl. Arch. of Photogrammetry and Remote Sensing, **26**:150-166, 1986

[Förstner 99] W. Förstner: *"Image Preprocessing for Feature Extraction in Digital Intensity, Color and Range Images"*, Proc. Summer School on Data Analysis and the Statistical Foundations of Geomatics, Lecture Notes in Earth Science, Berlin: Springer, 1999

[Förstner 05] W. Förstner: *"Uncertainty and Projective Geometry"*, in: E. Bayro Corrochano (Ed.): Handbook of Geometric Computing, pp. 493-534, Berlin: Springer, 2005

[Freeman & Adelson 91] W. Freeman, E. Adelson: *"The design and use of steerable filters"*, IEEE Trans. Pattern Analysis Machine Intelligence, 13(9):891-906, 1991

[Georgeson & Freeman 96] M.A. Georgeson, T. Freeman: *"Perveived Location of Bars and Edges in One-dimensional Images: Computational Models and Human Vision"*, Vision Research, 37(1):127-142, 1997

[Georgeson 98] M.A. Georgeson: *"Edge-finding in human vision: a multi-stage model based on the perceived structure of plaids"*, Image and Vision Computing **16**:389-405, 1998

[Geusebroek 05] J.-M. Geusebroek: *"The Stochastic Structure of Images"*, in: R. Kimmel, N. Sochen, J. Weickert (Eds.): Scale-Space and PDE Methods in Computer Vision, Proc. of ScaleSpace 05, Lecture Notes in Computer Science 3459, pp. 327-338, Berlin: Springer, 2005

[Goodman 05] W. Goodman: *"Introduction to Fourier Optics"*, Third Edition, Englewood: Roberts & Company, 2005

[Goudail & Réfrégier 04] F. Goudail, P. Réfrégier: *"Statistical Image Processing Techniques for Noisy Images"*, New York: Kluwer Academic / Plenum Publishers, 2004

[Graham & Hood 92] N. Graham, D.C. Hood: *"Modeling the dynamics of light adaptation: The merging of two tradtions"*, Vision Research, **25:**1373-1393, 1992

[Granlund & Knutsson 95] G. Granlund, H. Knutsson: *"Signal Processing for Computer Vision"*, Dordrecht: Kluwer Academic Publishers, 1995

[Granlund & Moe 04] G. Granlund, A. Moe, *"Unrestricted Recognition of 3-D Objects for Robotics Using Multi-Level Triplet Invariants"*, Artificial Intelligence Magazine 25(2):51-67, 2004

[Greivenkamp 90] J.E. Greivenkamp: *"Color dependent optical prefilter for the suppression of aliasing artifacts"*, Applied Optics, 29(5):676-684, 1990

[Haralick 84] R. Haralick: *"Digital Step Edges from Zero Crossings of Second Directional Derivatives"*, IEEE Trans. Pattern Analysis Machine Intelligence, **6:**58-64, 1984

[Haralick & Shapiro 92] R. Haralick, L. Shapiro: *"Computer and Robot Vision"*, vol. 1, Addison Wesley, 1992

[Harris & Stevens 88] C.G. Harris, M.J. Stevens: *"A Combined Corner and Edge Detector"*, Proc. of 4th Alvey Vision Conference, pp. 147-151, 1988

[Hartley & Zisserman 04] R.I. Hartley, A. Zisserman: *"Multiple View Geometry in Computer Vision"*, Second Edition, Cambridge University Press, 2004

[Hatcher 02] A. Hatcher: *"Algebraic Topology"*, Cambridge: Cambridge Univerity Press, 2002

[Heckbert 94] P.S. Heckbert (Ed.): *"Graphics Gems IV"*, San Diego: Morgan Kaufmann, 1994

[Healey & Kondepudy 94] G. Healey, R. Kondepudy: "Radiometric CCD Camera Calibration and Noise Estimation", IEEE Trans. Pattern Analysis Machine Intelligence, 16(3):267-276, 1994

[Heath et al. 97] M. Heath, S. Sarkar, T. Sanocki, K.W. Bowyer: *"A Robust Visual Method for Assessing the Relative Performance of Edge-Detection Algorithms"*, IEEE Trans. Pattern Analysis Machine Intelligence, 19(12):1338-1359, 1997

[Henderson 02] M.E. Henderson: *"Multiple Parameter Continuation: Computing Implicitly Defined k-Manifolds"*, Int. J. Bifurcation and Chaos, 12(3):451-476, 2002

[Hirsch & Curcio 89] J. Hirsch, C.A. Curcio: *"The Spatial Resolution Capacity of Human Foveal Retina"*, Vision Research, 29(9):1095-1101, 1989

[Hofer et al. 05a] H. Hofer, B. Singer, D.R. Williams: *"Different sensations from cones with the same photopigment"*, J. of Vision, vol. 5, pp. 444-454, 2005

[Hofer et al. 05b] H. Hofer, J. Carroll, J. Neitz, M. Neitz, D.R. Williams: *"Organization of the Human Trichromatic Cone Mosaic"*, J. of Neuroscience, 25(42):9669-9679, 2005

[Huck et al. 99] F.O. Huck, C.L. Fales, R. Alter-Gartenberg, S.K. Park, Z. Rahman: *"Information-theoretic assessment of sampled imaging systems"*, Optical Engineering 38(5):742-762, 1999

[Iglesias et al. 98] I. Iglesias, N. Lopez-Gil, P. Artal: *"Reconstruction of the Point Spread Function of the Human Eye from Two Double-Pass Retinal Images Using Phase Retrieval Algorithms"*, J. Optical Society of America A, vol. 15, pp. 326-339, 1998

[ISO 12233:2000] ISO Standard No. 12233: *"Photography - Electronic still picture cameras - Resolution measurements"*, 2000

[Jähne 97] B. Jähne: *"Practical Handbook on Image Processing for Scientific Applications"*, Boca Raton: CRC Press, 1997

[Jähne 02] B. Jähne: *"Digitale Bildverarbeitung"*, 5. Auflage, Berlin: Springer, 2002

[Johansson & Moe 05] B. Johansson, A. Moe: *"Patch-Duplets for Object Recognition and Pose Estimation"*, in: 2nd Canadian Conference on Robot Vision, pp. 9-16, IEEE Computer Society Press, 2005

[Julesz 81] B. Julesz: *"Textons, the Elements of Texture Perception, and their Interaction"*, Nature, 290:91-97, 1981

[Kakarala & Hero 92] R. Kakarala, A.O. Hero: *"On Achievable Accuracy in Edge Localization"*, IEEE Trans. Pattern Analysis Machine Intelligence, 14(7):777-781, 1992

[Kass et al. 88] E. Kass, A. Witkin, D. Terzopoulos: *"Snakes: active contour models"*, Intl. Journal for Computer Vision, 1(4):321-331, 1988

[Kaynig 06] V. Kaynig: *"Perceptual Criteria for Edge Relevance"*, Diploma thesis, Computer Science Department, University of Hamburg, 2006

[Kettner 98] L. Kettner: *"Designing a Data Structure for Polyhedral Surfaces"*, Proc. 14th ACM Symp. on Computational Geometry, New York: ACM Press, 1998

[Khalimsky et al. 90] E. Khalimsky, R. Kopperman, P. Meyer: *"Computer Graphics and Connected Topologies on Finite Ordered Sets"*, J. Topology and its Applications, vol. 36, pp. 1-27, 1990

[Klette & Rosenfeld 04] R. Klette, A. Rosenfeld: *"Digital Geometry"*, Amsterdam: Elsevier, 2004

[Knutsson & Westin 93] H. Knutsson, C.-F. Westin: *"Normalized and differential convolution"*, in: Proc. CVPR 93, Computer Vision and Pattern Recognition, pp. 515-523, 1993

[Koenderink & v. Doorn 93] J. Koenderink, A. v. Doorn: *"Local Features of Smooth Shapes: Ridges and Courses"*, SPIE vol. 2031: Proc. Geometric Methods in Computer Vision, pp. 2-13, 1993

[Köthe 00] U. Köthe: *"Generische Programmierung für die Bildverarbeitung"*, PhD thesis, Department of Informatics, University of Hamburg, 2000

[Köthe 01] U. Köthe: *"Generic Programming Techniques that Make Planar Cell Complexes Easy to Use"*, in: G. Bertrand, A. Imiya, R. Klette (Eds.): Digital and Image Geometry - Advanced Lectures (Proc. of a Dagstuhl Seminar), Lecture Notes in Computer Science 2243, pp. 17-37, Berlin: Springer, 2001

[Köthe 02] U. Köthe: *"XPMaps and Topological Segmentation - a Unified Approach to Finite Topologies in the Plane"*, in: A. Braquelaire, J.-O. Lachaud, A. Vialard (Eds.): Proc. of 10th International Conference on Discrete Geometry for Computer Imagery (DGCI 2002), Lecture Notes in Computer Science 2301, pp. 22-33, Berlin: Springer, 2002

[Köthe 03a] U. Köthe: *"Integrated Edge and Junction Detection with the Boundary Tensor"*, in: ICCV 03, Proc. of 9th Intl. Conf. on Computer Vision, Nice 2003, vol. 1, pp. 424-431, Los Alamitos: IEEE Computer Society, 2003

[Köthe 03b] U. Köthe: *"Deriving Topological Representations from Edge Images"*, in: T. Asano, R. Klette, C. Ronse (Eds.): Geometry, Morphology, and Computational Imaging, 11th Intl. Workshop on Theoretical Foundations of Computer Vision, Lecture Notes in Computer Science 2616, pp. 320-334, Berlin: Springer, 2003

[Köthe 03c] U. Köthe: *"Edge and Junction Detection with an Improved Structure Tensor"*, in: B. Michaelis, G. Krell (Eds.): Pattern Recognition, Proc. of 25th DAGM Symposium, Magdeburg 2003, Lecture Notes in Computer Science 2781, pp. 25-32, Berlin: Springer, 2003

[Köthe & Stelldinger 03] U. Köthe, P. Stelldinger: *"Shape Preserving Digitization of Ideal and Blurred Binary Images"*, in: I. Nyström, G. Sanniti di Baja, S. Svensson (Eds.): Discrete Geometry for Computer Imagery, Proc. of 11th DGCI Conference, Naples 2003, Lecture Notes in Computer Science 2886, pp. 82-91, Berlin: Springer, 2003

[Köthe 04] U. Köthe: *"Accurate and Efficient Approximation of the Continuous Gaussian Scale-Space"*, in: C.E. Rasmussen, H. Bülthoff, M. Giese, B. Schölkopf (Eds.): Pattern Recognition, Proc. of 26th DAGM Symposium, Lecture Notes in Computer Science 3175, pp. 350-358, Berlin: Springer, 2004

[Köthe & Felsberg 06] U. Köthe, M. Felsberg: *"Riesz-Transforms Versus Derivatives: On the Relationship Between the Boundary Tensor and the Energy Tensor"*, in: R. Kimmel, N. Sochen, J. Weickert (Eds.): Scale Space and PDE Methods in Computer Vision, Proc. of Scale-Space 2005, Lecture Notes in Computer Science 3459, pp. 179-191, Berlin: Springer, 2005

[Köthe 06a] U. Köthe: *"Low-level Feature Detection Using the Boundary Tensor"*, in: J. Weickert, H. Hagen (Eds.): Visualization and Processing of Tensor Fields, pp. 63-79, Berlin: Springer, 2006

[Köthe 06b] U. Köthe: *"VIGRA Reference Documentation"*, version 1.5.0, 2006
`http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/`

[Köthe 06c] U. Köthe: *"Boundary Characterization within the Wedge-Channel Representation"*, in: B. Jähne, E. Barth, R. Mester, H. Scharr (Eds.): 1st Intl. Workshop on Complex Motion, IWCM 2004, Lecture Notes in Computer Science 3417, pp. 42-53, Berlin: Springer, 2006

[Köthe et al. 06] U. Köthe, P. Stelldinger, H. Meine: *"Provably Correct Edgel Linking and Subpixel Boundary Reconstruction"*, in: K. Franke, K.-R. Müller, B. Nikolay, R. Schäfer (Eds.): Pattern Recognition, Proc. DAGM 2006, Lecture Notes in Computer Science 4174, pp. 81-90. Berlin: Springer, 2006

[Kovalevsky 89] V. Kovalevsky: *"Finite Topology as Applied to Image Analysis"*, Computer Vision, Graphics, and Image Processing, 46(2):141-161, 1989

[Kovalevsky 97] V. Kovalevsky: *"Applications of Digital Straight Segments to Economical Image Encoding"*, in: E. Ahronovitz, C. Fiorio (Eds): Discrete Geometry for Computer Imagery, Proc. of 7th DGCI Conference, Lecture Notes in Computer Science 1347, pp. 49-62, Berlin: Springer,1997

[Kovalevsky 01a] V. Kovalevsky: *"Algorithms and Data Structures for Computer Topology"*, In: G. Bertrand, A. Imiya, R. Klette (Eds.): Digital and Image Geometry - Advanced Lectures (Proc. of a Dagstuhl Seminar), Lecture Notes in Computer Science 2243, pp. 37-58, Berlin: Springer, 2001

[Kovalevsky 01b] V. Kovalevsky: *"Curvature in Digital 2D Images"*, Intl. J. Pattern Recognition and Artificial Intelligence, 15(7):1183-1200, 2001

[Kropatsch 95] W. Kropatsch: *"Building Irregular Pyramids by Dual Graph Contraction"*, IEE Proceedings Vision, Image and Signal Processing, 142(6):366-374, 1995

[Latecki 98] L.J. Latecki: *"Discrete Representation of Spatial Objects in Computer Vision"*, Dordrecht: Kluwer Academic Publishers, 1998

[Latecki et al. 98] L.J. Latecki, C. Conrad, A. Gross: *"Preserving Topology by a Digitization Process"*, Journal of Mathematical Imaging and Vision **8**:131-159, 1998

[Liang & Williams 97] J. Liang, D.R. Williams: *"Aberrations and retinal image quality of the normal human eye"*, J. Optical Society of America A, 14(11):2873-2883, 1997

[Lienhardt 91] P. Lienhardt: *"Topological models for boundary representation: a comparison with n-dimensional generalized maps"*, Computer Aided Design, 23(1):59-82, 1991

[Lim 03] J.-Y. Lim: *"Discrete Scale-Space Formulation and Multiscale Edge Extraction toward Higher Dimensions"*, PhD Thesis, Department of Informatics, University of Hamburg, Berlin: Akademische Verlagsgesellschaft, 2003

[Lindeberg 94] T. Lindeberg: *"Scale-Space Theory in Computer Vision"*, Dordrecht: Kluwer Academic Publishers, 1994

[Lowe 04] D.G. Lowe, *"Distinctive image features from scale-invariant keypoints"*, International Journal of Computer Vision, 60(2):91-110, 2004

[Lyon & Hubel 02] R. Lyon, P. Hubel: *"Eying the Camera: Into the Next Century"*, in: Proc. of IS&T/TSID 10th Color Imaging Conference, pp. 349-355, 2002

[Lyvers & Mitchell 88] E.P. Lyvers, O.R. Mitchell: *"Precision Edge Contrast and Orientation Estimation"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 10(6):927-937, 1988

[MacLoad et al. 92] D. MacLoad, D.R. Williams, W. Makous: *"A Visual Nonlinearity Fed by Single Cones"*, Vision Research, 32(2):347-363, 1992

[Mäntylä 88] M. Mäntylä: *"An Introduction to Solid Modeling"*, Computer Science Press, 1988

[Maes 98] F. Maes: *"Segmentation and Registration of Multimodal Images: From Theory, Implementation and Validation to a Useful Tool in Clinical Practice"*, PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, May 1998.

[Marcos 03] S. Marcos: *"Image Quality of the Human Eye"*, Int. Ophthalmol. Clin., 43(2):42-62, 2003

[Marr 82] D. Marr: *"Vision – A Computational Investigation into the Human Representation and Processing of Visual Information"*, San Francisco: W.H. Freeman and Company, 1982

[Martin et al. 01] D. Martin, C. Fowlkes, D. Tal, J. Malik: *"A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics"*, in: Proc. 8th Intl. Conf. on Computer Vision, vol. 2, pp. 416-425, 2001

[Maxwell 1870] J.C. Maxwell: *"On Hills and Dales"*, London, Edinburgh, and Dublin Philosophical Mag. and J. of Sci., 40:421-425, 1870; reprinted in: W. Niven (Ed.): The Scientific Papers of James Clark Maxwell, vol. II, Dover, 1965

[McGlone 04] J. McGlone (Ed.): *"Manual of Photogrammetry"*, Fifth Edition, Bethesda: American Society for Photogrammetry and Remote Sensing, 2004

[Medioni et al. 00] G. Medioni, M.-S. Lee, C.-K. Tang: *"A Computational Framework for Segmentation and Grouping"*, Elsevier, 2000

[Meijering et al. 99] E.H.W. Meijering, K.J. Zuiderveld, M.A. Viergever: *"Image Reconstruction by Convolution with Symmetrical Piecewise nth-Order Polynomial Kernels"*, IEEE Trans. Image Processing, 8(2):192-201, 1999

[Meine 03] H. Meine: *"XPMap-Based Irregular Pyramids for Image Segmentation"*, Diploma thesis, Department of Informatics, University of Hamburg, 2003

[Meine et al. 04] H. Meine, U. Köthe, H.S. Stiehl: *"Fast and Accurate Interactive Image Segmentation in the GeoMap Framework"*, in: T. Tolxdorff, J. Braun, H. Handels, A. Horsch, H.-P. Meinzer (Eds.): Proc. Bildverarbeitung für die Medizin 2004, pp. 60-64, Berlin: Springer, 2004

[Meine & Köthe 05a] H. Meine, U. Köthe: *"The GeoMap: A Unified Representation for Topology and Geometry"*, in: L. Brun, M. Vento (Eds.): Graph-Based Representations in Pattern Recognition, Proc. 5th IAPR Workshop GbRPR '05, Lecture Notes in Computer Science 3434, pp. 132-141, Springer, 2005

[Meine & Köthe 05b] H. Meine, U. Köthe: *"Image Segmentation with the Exact Watershed Transform"*, in: J.J. Villanueva (Ed.): VIIP 05, Proc. 5th IASTED Intl. Conf. Visualization, Imaging, and Image Processing, pp. 400-405, ACTA Press, 2005.

[Meine 08] H. Meine: PhD thesis, Department of Informatics, University of Hamburg, forthcoming

[Meyer 94] F. Meyer: *"Topographic Distance and Watershed Lines"*, Signal Processing 38(1):113-125, 1994

[Michailovich & Tannenbaum 06] O. Michailovich, A. Tannenbaum: *"Despeckling of Medical Ultrasound Images"*, IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control, 53(1):64-78, 2006

[Mikolajczyk & Schmid 04] K. Mikolajczyk, C. Schmid: *"Scale and affine invariant interest point detectors"*, International Journal of Computer Vision, 60(1):63-86, 2004

[Montanvert et al. 91] A. Montanvert, P. Meer, A. Rosenfeld: *"Hierarchical Image Analysis Using Irregular Tessellations"*, IEEE Trans. Pattern Anal. and Machine Intelligence, 13(4):307-316, 1991

[Moré & Thuente 94] J. Moré, D. Thuente: *"Line Search Algorithms with Guaranteed Sufficient Decrease"*, ACM Trans. Math. Software 20, 286-307, 1994.

[Mortensen & Barrett 98] E. Mortensen, W. Barrett: *"Interactive segmentation with intelligent scissors"*, Graphical Models and Image Processing, 60(5):349-384, 1998

[Mortensen & Barrett 99] E. Mortensen, W. Barrett: *"Toboggan-based intelligent scissors with a four parameter edge model"*, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition, CVPR '99, vol. 2, p. 452-458, 1999

[Nackman 84] L.R. Nackman: *"Two-dimensional critical point configuration graphs"*, IEEE Trans. Pattern Anal. and Machine Intelligence 6(4):442-450, 1984

[Nagel 85] H.H. Nagel: *"Analyse und Interpretation von Bildfolgen II"*, Informatik-Spektrum, 8(6):312-327, 1985

[Najman & Schmitt 94] L. Najman, M. Schmitt: *"Watershed of a continuous function"*, Signal Processing 38:99-112, 1994

[Neumann 88] H. Neumann: *"Theoretische Untersuchungen zur Extraktion monokularer Tiefenhinweise (Konturen und Schattierung) und ihre partielle methodische Evaluierung in einem rechnergestützten Perzeptionslabor"*, PhD Thesis, Fachbereich Informatik, Universität Hamburg, 1988

[Nguyen et al. 03] H.T. Nguyen, M. Worring, R. v.d. Boomgaard: *"Watersnakes: Energy-Driven Watershed Segmentation"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 25(3):330-342, 2003

[Nordberg & Farnebäck 03] K. Nordberg, G. Farnebäck: *"A Framework for Estimation of Orientation and Velocity"*, Proc. IEEE Intl. Conf. on Image Processing, vol. 3, pp. 57-60, 2003

[Nordberg 04] K. Nordberg: *"A fourth order tensor for representation of orientation and position of oriented segments"*, Technical Report LiTH-ISY-R-2587, Dept. of Electrical Engineering, Linköping University, 2004

[Olsen & Nielsen 97] O.F. Olsen, M. Nielsen, *"Multi-scale gradient magnitude watershed segmentation"*, in: A. Del Bimbo (Ed.): Image Analysis and Processing , 9th Int. Conf. ICIAP'97, Lecture Notes in Computer Science 1310, pp. 6-13, Berlin: Springer, 1997

[Ortiz & Oliver 06] A. Ortiz, G. Oliver: *"Radiometric Calibration of Vision Cameras and Intensity Uncertainty Estimation"*, Image and Vision Computing, 24(10):1137-1145, 2006

[Osher & Paragios 03] S. Osher, N. Paragios: *"Geometric Level Set Methods in Imaging, Vision, and Graphics"*, Berlin: Springer, 2003.

[Osorio et al. 98] D. Osorio, D.L. Ruderman, T.W. Cronin: *"Estimation of errors in luminance signals encoded by primate retina resulting from sampling of natural images with red and green cones"*, J. Optical Society of America A, 15(1):16-22, 1998

[Overington 92] I. Overington: *"Computer Vision: a unified, biologically-motivated approach"*, Amsterdam: Elsevier, 1992

[Park & Rahman 99] S.K. Park, Z. Rahman: *"Fidelity analysis of sampled imaging systems"*, Optical Engineering 38(5):786-800, 1999

[Pattanaik et al. 98] S. Pattanaik, J.A. Ferwerda, M.D. Fairchild, D.P. Greenberg: *"A multiscale model of adaptation and spatial vision for realistic image display"*, in: M.F. Cohen (ed.): Proc. of SIGGRAPH 98, pp. 287-298, Addison Wesley, 1998.

[Pavlidis 77] T. Pavlidis: *"Structural Pattern Recognition"*, Berlin: Springer, 1977

[Pavlidis 82] T. Pavlidis: *"Algorithms for Graphics and Image Processing"*, Rockville: Computer Science Press, 1982

[Poularikas 96] A.D. Poularikas (Ed.): *"The Transforms and Applications Handbook"*, CRC Press, Boca Raton, 1996

[Poynton 96] C. Poynton: *"A Technical Introduction to Digital Video"*, New York: Wiley, 1996

[Pratt 87] W. Pratt, *"Direct least-squares fitting of algebraic surfaces"*, Computer Graphics **21**:145-152, 1987

[Pratt 01] W. Pratt: *"Digital Image Processing"*, Third Edition, New York: Wiley, 2001

[Pritchard 73] D.H. Pritchard: *"Stripe-Color-Encoded Single Tube Color-Television Camera Systems"*, RCA Rev., vol. 34, pp. 217-266, 1973

[Proakis 89] J.G. Proakis: *"Digital Communications"*, Second Edition, New York: McGraw Hill, 1989

[Rahman & Jobson 03] Z. Rahman, D.J. Jobson: *"Information Theoretic Analysis of Noise Sources in Image Formation"*, In: SPIE vol. 5108, Proc. Visual Information Processing XII, 2003

[Ren & Malik 03] X. Ren, J. Malik: *"Learning a classification model for segmentation"*, in: Proc. 9th Int. Conf. Computer Vision, ICCV'03, vol. 1, pp. 10-17, 2003

[Ren et al. 05a] X. Ren, C. Fowlkes, J. Malik: *"Scale-invariant contour completion using conditional random fields"*, in: Proc. 10th Intl. Conf. Computer Vision, ICCV'05, vol. 2, pp. 1214-1221, 2005

[Ren et al. 05b] X. Ren, A. Berg, J. Malik: *"Recovering Human Body Configurations using Pairwise Constraints between Parts"*, in: Proc. 10th Intl. Conf. Computer Vision, ICCV'05, vol. 1, pp. 824-831, 2005

[Rice 45] S.O. Rice: *"Mathematical analysis of Random Noise"*, Bell System Technical Journal, 24:46-156, 1945

[Rieger 97] J. Rieger: *"Topographical properties of generic images"*, Intl. Journal of Computer Vision, 23(1):79-92, 1997

[Roerdink & Meijster 00] J. Roerdink, A. Meijster: *"The watershed transform: definitions, algorithms, and parallelization strategies"*, Fundamenta Informaticae 41:187-228, 2000

[Rohr 92] K. Rohr: *"Modelling and Identification of Characteristic Intensity Variations"*, Image and Vision Computing 10(2):66-76, 1992

[Rohr 94] K. Rohr: *"Localization Properties of Direct Corner Detectors"*, Journal of Mathematical Imaging and Vision **4**:139-150, 1994

[Ronse & Tajine 00] C. Ronse, M. Tajine: *"Discretization in Hausdorff Space"*, Journal of Mathematical Imaging and Vision **12**:219-242, 2000

[Roorda & Williams 99] A. Roorda, D.R. Williams: *"The arrangement of the three cone classes in the living human eye"*, Nature, vol. 397, pp. 520-522, 1999

[Rosenfeld 70] A. Rosenfeld: *"Connectivity in digital pictures"*, J. of the Association for Computing Machinery, 17(1):146-160, 1970

[Rosin et al. 92] P.L. Rosin, A.C.F. Colchester, D.. Hawkes: *"Early image representation using regions defined by maximum gradient paths between singular points"*, Pattern Recognition, 25(7):695-711, 1992

[Rothwell et al. 95] C. Rothwell, J. Mundy, W. Hoffman, V.-D. Nguyen: *"Driving Vision by Topology"*, in: Proc. of IEEE Intl. Symposium on Computer Vision, pp. 395-400, 1995

[Rovamo & Kukkonen 98] J. Rovamo, H. Kukkonen: *"Foveal optical modulation transfer function of the human eye at various pupil sizes"*, J. Opt. Soc. Am. A, 15(9):2504-2513, 1998

[Rudin et al. 92] L.I. Rudin, S. Osher, E. Fatemi: *"Non-linear total variation based noise removal algorithms"*, Physica D, 60:259-268, 1992

[Ryan & Schwartz 56] T.A. Ryan, C.B. Schwartz: *"Speed of perception as a function of mode of representation"*, American Journal of Psychology, 69:60-69, 1956

[Schade 48] O.H. Schade: *"Electro-optical characteristics of television systems"*, RCA Review 9(1):5-37, 1948

[Schoenberg 64] J.J. Schoenberg: *"Spline functions and the problem of graduation"*, Proc. Nat. Acad. of Science **52**:947-950, 1994

[Serra 82] J. Serra: *"Image Analysis and Mathematical Morphology"*, New York: Academic Press, 1982

[Serre et al. 05] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, T. Poggio: *"A Theory of Object Recognition: Computations and Circuits in the Feedforward Path of the Ventral Stream in Primate Visual Cortex"*, AI Memo 2005-036/CBCL Memo 259, Massachusetts Inst. of Technology, Cambridge, 2005.

[Shen & Castan 92] J. Shen, S. Castan: *"An Optimal Linear Operator for Step Edge Detection"*, CVGIP: Graphical Models and Image Processing, 54(2):112-133, 1992

[Sicuranza 92] G. Sicuranza: *"Quadratic Filters for Signal Processing"*, Proc. of the IEEE, 80(8):1263-1285, 1992

[Smith & Brady 97] S. Smith, M. Brady: *"SUSAN – A New Approach to Low Level Image Processing"*, Intl. Journal of Computer Vision, 23(1):45-78, 1997

[Smith et al. 03] C. Smith, F. Shu, L. Ion, M. Cowan: *"Image Resolution of the One-CCD Palomar Motion Picture Camera"*, 37th Advanced Motion Imaging Conference, 2003

[Spies & Johansson 03] H. Spies, B. Johansson: *"Directional Channel Representation for Multiple Lines Endings and Intensity Levels",* in: ICIP 03, Proc. IEEE Intl. Conf. on Image Processing, 2003

[Sporring et al. 97] J. Sporring, M. Nielsen, L. Florack, P. Johansen (Eds.): *"Gaussian Scale-Space Theory"*, Dordrecht: Kluwer Academic Publishers, 1997

[Steger 99] C. Steger: *"Subpixel-Precise Extraction of Watersheds"*, in: ICCV '99, Proc. 7th Intl. Conf. Computer Vision, vol. II, pp. 884-890, 1999

[Stelldinger 03] P. Stelldinger: *"Theoretische Grenzen der Bilddigitalisierung"*, Diploma thesis, Department of Informatics, University of Hamburg, 2003

[Stelldinger & Köthe 03] P. Stelldinger, U. Köthe: *"Shape Preservation During Digitization: Tight Bounds Based on the Morphing Distance"*, in: B. Michaelis, G. Krell (Eds.): Pattern Recognition, Proc. of 25th DAGM Symposium, Magdeburg 2003, Lecture Notes in Computer Science 2781, pp. 108-115, Berlin: Springer, 2003

[Stelldinger & Köthe 05] P. Stelldinger, U. Köthe: *"Towards a general sampling theory for shape preservation"*, Image and Vision Computing, Special Issue on Discrete Geometry for Computer Vision, 23(2):237-248, 2005

[Stelldinger 05] P. Stelldinger: *"Digitization of Non-regular Shapes"*, in: C. Ronse, L. Najman, E. Decenciere (Eds.): Mathematical Morphology, Proc. of ISMM '05, Dordrecht: Springer, 2005

[Stelldinger & Köthe 06] P. Stelldinger, U. Köthe: *"Connectivity preserving digitization of blurred binary images in 2D and 3D"*, Computers & Graphics, 30(1):70-76, 2006.

[Stelldinger et al. 06] P. Stelldinger, U. Köthe, H. Meine: *"Topologically Correct Image Segmentation Using Alpha Shapes"*, in: DGCI'06, Lecture Notes in Computer Science, Berlin: Springer, 2006

[Stolte 05] N. Stolte: *"Arbitrary 3D Resolution Discrete Ray Tracing of Implicit Surfaces"*, in: E. Andres, G. Damiand, P. Lienhardt (Eds.): Discrete Geometry for Computer Imagery, Proc. of 12th Intl. Conf. DGCI 2005, Lecture Notes in Computer Science 3429, pp. 414-426, Berlin: Springer, 2005.

[Thibos et al. 87] L.N. Thibos, F.E. Cheney, D.J. Walsh: *"Retinal limits to the detection and resolution of gratings"*, J. Optical Society of America A, 4(8):1524-1529, 1987

[Tutte 84] W.T. Tutte: *"Graph Theory"*, Cambridge University Press, 1984

[Unnikrishnan et al. 07] R. Unnikrishnan, C. Pantofaru, M. Hebert: *"Toward Objective Evaluation of Image Segmentation Algorithms"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 29(6):929-944, 2007

[Unser et al. 93] M. Unser, A. Aldroubi, M. Eden: *"B-Spline Signal Processing"*, IEEE Trans. Signal Processing, 41(2), pp. 821-833 (part I), 834-848 (part II), 1993

[Utcke 03] S. Utcke: *"Error-Bounds on Curvature Estimation"*, in: L. Griffin, M. Lillholm (Eds.): Scale Space Methods in Computer Vision, Proc. 4th ScaleSpace Conf., Lecture Notes in Computer Science 2695, pp. 657-666, Berlin: Springer, 2003

[Utcke 06] S. Utcke: *"Error propagation in geometry-based grouping"*, PhD thesis, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, 2006

[VanRullen & Thorpe 01] R. VanRullen, S.J. Thorpe: *"Is it a bird? Is it a Plane? Ultra-rapid visual categorization of natural and artificial categories"*, Perception, 30(6):655-688, 2001

[Vese & Chan 02] L. Vese, T. Chan: *"A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model"*, Intl. J. of Computer Vision, 50(3):271-293, 2002

[Vialard 96] A. Vialard: *"Geometric Parameter Extraction from Digital Paths"*, in: S. Miguet, A. Montanvert, S. Ubéda (Eds.): Discrete Geometry for Computer Imagery, Proc. of 6th DGCI Conference, Lecture Notes in Computer Science 1176, pp. 24-35, Berlin: Springer, 1996

[Vimal et al. 89] R.P. Vimal, J. Pokorny, V. Smith, S. Shevel: *"Foveal Cone Thresholds"*, Vision Research 29(1):61-78, 1989

[Vincent & Soille 91] L. Vincent, P. Soille: *"Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 13(6):583-598, 1991

[Wachtler et al. 96] T. Wachtler, C. Wehrhahn, B. Lee: *"A Simple Model of Human Foveal Ganglion Cell Response to Hyperacuity Stimuli"*, J. Computational Neuroscience, vol. 3, pp. 73-82, 1996

[Wallace et al. 01] W. Wallace, L.H. Schaefer, J.R. Swedlow. *"A working persons guide to deconvolution in light microscopy"*, BioTechniques 31(5):1076-1097, 2001

[Weiss 94] I. Weiss: *"High-Order Differentiation Filters That Work"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 16(7):734-739, 1994

[Wendland 05] H. Wendland: *"Scattered Data Approximation"*, Cambridge University Press, 2005

[Westheimer 86] G. Westheimer: *"The Eye as an Opical Instrument"*, in: K.R. Boff, L. Kaufman, J.P. Thomas (Eds.): Handbook of Perception and Human Performance, vol. 1, pp. 4.1-4.20, New York: Wiley and Sons, 1986

[Williams & Burns 01] D.R. Williams, P.D. Burns: *"Diagnostics for Digital Capture using MTF"*, in: Proc. of IS&T PICS Conference, pp. 227-232, 2001

[Williams 85a] D.R. Williams: *"Aliasing in Human Foveal Vision"*, Vision Research, 25(2):195-205, 1985

[Williams 85b] D.R. Williams: *"Visibility of inerference fringes near the resolution limit"*, J. Optical Society of America A, 2(7):1087-1093,1985

[Williams et al. 94] D.R. Williams, D.H. Brainard, M. McMahon, R. Navarro: *"Double-pass and interferometric measures of the optical quality of the eye"*, J. Optical Society of America A, 11(12):3123-3135, 1994

[Williams & Hofer 03] D.R. Williams, H. Hofer: *"Formation and Acquisition of the Retinal Image"*, in: L.M. Chalupa, J.S. Werner (Eds.): The Visual Neuro Sciences, vol. 1, pp. 795-810, MIT Press, 2003

[Williams & Jacobs 97] L. Williams, D. Jacobs: *"Stochastic Completion Fields: A Neural Model of Illusory Contour Shape and Salience"*, Neural Computation 9(4):837-858, 1997

[Winkler 99] S. Winkler: *"Issues in vision modeling for perceptual video quality assessment"*, Signal Processing, 78(2):231-252, 1999

[Winter 95] S. Winter: *"Topological Relations Between Discrete Regions"*, in: M. Egenhofer, J. Herring (Eds.): Advances in Spatial Databases, Lecture Notes in Computer Science 951, Berlin: Springer, 1995

[Worring & Smeulders 93] M. Worring, A. Smeulders: *"Digital Curvature Estimation"*, CVGIP: Image Understanding, 58(3):366-382, 1993

[Worring & Smeulders 95] M. Worring, A. Smeulders: *"Digital Circular Arcs: Characterization and Parameter Estimation"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 17(6):587-598, 1995

[Yitzhaky & Peli 03] Y. Yitzhaky, E. Peli: *"Method for Objective Edge Detection Evaluation and Detector Parameter Selection"*, IEEE Trans. Pattern Analysis and Machine Intelligence, 25(8):1027-1033, 2003

[Zomorodian 05] A. Zomorodian: *"Topology for Computing"*, Cambridge University Press, 2005