Weakly supervised learning of image partitioning using decision trees with structured split criteria*

Christoph Straehle
christoph.straehle@iwr.uni-heidelberg.de
Ullrich Koethe
ullrich.koethe@iwr.uni-heidelberg.de
Fred A. Hamprecht
fred.hamprecht@iwr.uni-heidelberg.de

HCI, University of Heidelberg

Abstract

We propose a scheme that allows to partition an image into a previously unknown number of segments, using only minimal supervision in terms of a few must-link and cannot-link annotations. We make no use of regional data terms, learning instead what constitutes a likely boundary between segments. Since boundaries are only implicitly specified through cannot-link constraints, this is a hard and nonconvex latent variable problem. We address this problem in a greedy fashion using a randomized decision tree on features associated with interpixel edges. We use a *structured* purity criterion during tree construction and also show how a backtracking strategy can be used to prevent the greedy search from ending up in poor local optima. The proposed strategy is compared with prior art on natural images.

1 Introduction

This paper describes a new method to learn *edge* models from sparse user scribbles marking *regions*. Consider Figure 1 and assume that we want to segment each kiwi. Normally, scribbles as shown on the left would be used to learn region appearance models that can then serve as potential functions in energy-minimizing segmentation methods such as graph cuts. However, this does not work here because the individual objects are indistinguishable by region appearance. Another popular approach would use the scribbles as seeds for a suitable region growing algorithm such as a seeded watershed or random walker. However, such an approach would need a seed or scribble for each and every object.

*accepted at ICCV 2013

Figure 1: Segmentation example. Each connected component of the user scribbles is treated as an individual label and the decision tree is trained using must-link constraints inside each component and cannot-link constraints between label components. The resulting tree learns an edge model consistent with the user-provided constraints and successfully generalizes to the unlabeled part of the image, where it segments many objects successfully.

When region appearance alone is not informative, segmentation must be based on an edge model. Our ambition is to train such a model with minimal labeling effort on the user's part. Traditional learning methods require the user to place edge scribbles exactly on the desired edges. The required localization accuracy makes this a time consuming task. Section 1.1 describes recent proposals for a simplified edge labeling. In contrast, we strive to use cheap region scribbles like in Figure 1 to train edge models instead of the usual region models.

Clearly, region scribbles cannot be used for edge learning directly because they are typically located far away from edges. However, they provide a large number of constraints that can control edge learning indirectly:

- Each pair of pixels from the same scribble defines a *must-link* constraint, i.e. there must be at least one connecting path that does not cross any edge.
- Pixels from different scribbles define a *cannot-link* constraint, i.e. any connecting path must cross at least one edge.

We call this a "link-or-cut edge learning" problem. It turns out that these constraints contain sufficient information for successful training of an edge model, and we propose a structured decision tree-type algorithm to solve this problem.

Since the training data does not contain direct edge annotations, the training error cannot be defined as the fraction of mis-classified pixels or edges. However, such a simplistic criterion is unsuitable for segmentation quality assessment anyway [30, 13]: It cannot penalize the global consequences (big changes in the resulting connected components) that may be caused by local errors such as a fine gap in an object contour. Instead, we define the training error in terms of a clustering quality score similar to the Rand Index (eq. 1), which can be directly derived from the pairwise constraints. This has profound consequences for the learning algorithm: Local decisions should be conditioned on the state of the *entire* segmentation, and our new learning algorithm reflects this requirement.

The proposed algorithm recursively builds a decision tree that predicts the state of each edge of the image graph. During tree construction we use a non-local split criterion which takes into account the global connectivity consequences of local edge predictions.

To summarize, the proposed algorithm relies on cheap must-link and cannot-link annotations and has the following virtues:

- it requires no region appearance terms,
- the number of segments need not be specified in advance,
- weak supervision in terms of sparse annotations is sufficient and no explicit edge labels are needed,
- the training optimizes a global clustering score in a decision tree.

To the best of our knowledge, this is the first time a decision tree is trained using a non-local structured split criterion.

1.1 Related Work

Previous work on edge learning includes many approaches which are based on training data with exact boundary localization, e.g. [19, 34, 18, 23, 9]. The method presented in [26] learns an optimal edge labeling policy based on context and gestalt features, but also requires dense ground truth. An interesting approach using weaker supervision is the livewire method in [4] which snaps a path to the most probable boundary predicted by a classifier which is trained online. Another approach using weaker supervision is presented in [2]. The author learns an edge model from inaccurate boundary annotations.

Small errors in the boundary predictions have large global consequences when calculating the connected components. To avoid this some authors introduce higher order constraints (e.g. boundary closedness) to obtain a consistent segmentation [16, 1, 17]. A novel take at edge learning based on a non local clustering quality measure is used in [28] to learn a neural network. A non-local warping error that takes topological constraints into account is also proposed in [13] but the authors also use a dense labeling during neural network training.

The decision tree based edge learning algorithm that we propose is related to [25]. The authors learn edge on/off probabilities using a decision tree, but the method requires a dense labeling as input and does not take the effect on the connected components of the graph into account – it acts locally. Our special split criterion is inspired for example by [15] where a special loss function in the split nodes of a decision tree is optimized. But in contrast to our approach their objective is local, as is the case in [14]. The authors of [20] have introduced a decision tree algorithm that works on locally structured labels. We build on this idea and extend it to a structured loss function on a complete image graph. In [21] a decision tree is proposed whose intermediate learning state is used as a feature for further tree growing. This is the basis for our proposed algorithm which evaluates a structured split criterion with regard to the intermediate tree state.

The must-link and cannot-link constraints that we use to train our learning algorithm have also been used by [11, 32]. These algorithms partition an image graph into connected components using said constraints. Both partitioning algorithm use a single scalar value associated with each edge whereas our method can take a multitude of edge features into account and learns an edge model from the given constraints. The same type of partitioning constraints has been considered in [12] where the authors introduce must-link constraints in the context of the normalized cut algorithm. In the context of the image foresting transform cannot-link constraints have been investigated in [22]. To summarize, must-link and cannot-link constraints haven been investigated in the literature, but our approach is novel since we use these constraits for weakly supervised boundary learning.

2 Problem Definition and Objective function

We consider a segmentation problem defined on a graph $G(\mathcal{E}, \mathcal{V})$ in which the nodes $n_i \in \mathcal{V}$ correspond to the pixels of an image, and the edges $(i, j) \in \mathcal{E}$ correspond to

the pixel neighborhood of the image. We assume a suitable set of edge features w_{ijf} (such as color gradients or structure tensor eigenvalues on different scales) is available and can be attributed to each edge (i, j). In addition we are given a sparse constraint matrix $C \in \{-1, 0, 1\}^{N \times N}$ that defines whether a pair of pixels (i, j) must be in the same component ($C_{ij} = 1$), or in two different components ($C_{ij} = -1$). The decision variables $x_{ij} \in \{0, 1\}$ determine whether an edge (i, j) is removed from ($x_{ij} = 0$), or remains ($x_{ij} = 1$) in the graph G. The objective function $F(\mathbf{c}, \mathbf{x})$ which we seek to maximize depends on the set of constraints $\mathbf{c}$ and the connected components or partitions $\pi(\mathbf{x})$ implied by the binary edge indicator variables x_{ij}:

$$F(\mathbf{c}, \pi(\mathbf{x})) = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TP + FP)(TN + FN)}} \tag{1}$$

where $TP(\mathbf{c}, \pi(\mathbf{x}))$ is the number of pairs of pixels which are correctly (according to the constraints $\mathbf{c}$) assigned to the same component and $FP(\mathbf{c}, \pi(\mathbf{x}))$ is the number of pairs of pixels which are incorrectly assigned to the same component. $TN(\mathbf{c}, \pi(\mathbf{x}))$ and $FN(\mathbf{c}, \pi(\mathbf{x}))$ are the number of true negative and false negative pairs respectively. Equation 1 is known as Matthews correlation coefficient [24].

Thus, in the optimum $\hat{\mathbf{x}} = \operatorname{argmax}_{\mathbf{x}} F(\mathbf{c}, \pi(\mathbf{x}))$, the binary indicator vector $\hat{\mathbf{x}}$ corresponds to a partitioning of the graph into a set of connected components $\pi(\mathbf{x})$ that satisfy the constraint matrix. Note that the number of connected components defined by $\pi(\mathbf{x})$ can be larger than the number of components defined by the constraints $\mathbf{c}$: the unconstrained pixels of the image can be partitioned in many different ways.

2.1 Overview: global optimization using decision trees

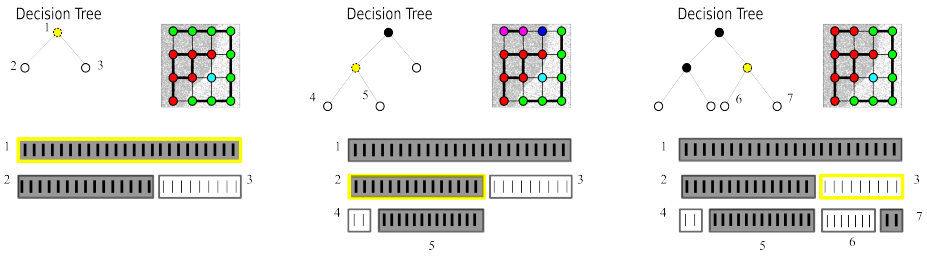

Figure 2: Illustration of our decision tree building process. Starting at the root node (1) the edges are partitioned into two sets, one of which is assigned $x_{ij} = 1$ (thick, edges stays in the graph) while the edges of the other partition are assigned $x_{ij} = 0$ (thin, edges are removed from the graph). In the next steps this initial decision on the edge states is revised by partitioning the two edge sets recursively further into an on and off set such that the objective function $F(\mathbf{x}, \mathbf{c})$ is maximized. The value of the objective function depends on the connected components of the graph G that are induced by the edge state defined in the leaf nodes. The connected components are distinguished by the node colors.

The objective function $F(\mathbf{c}, \pi(\mathbf{x}))$ defines a *global* objective which depends on the structure of the graph G. Maximizing F may be a simple task when the constraint set is very small: many different possible edge assignments $\mathbf{x}$ may partition the graph correctly. But depending on the size of the constraint set and the structure of the graph the problem can become computationally infeasible. The objective function is highly non-smooth and non-convex: switching a single binary indicator variable x_{ij} – for example on the boundary between two objects – may have a huge influence on the value of F.

In contrast to existing approaches which use classifiers to learn local pixel class probabilities or which learn pairwise boundary probabilities [25] from strong edge/no-edge examples, our aim is to *learn from weak labels*: the learner will be trained only from the *constraint set* $\mathbf{c}$ derived from sparse user scribbles and the *structure of the problem*, the pixel neighborhood graph G.

We have chosen to optimize Equation 1 using a greedy method inspired by traditional decision trees. Decision trees are constructed in the following fashion: Given a set of examples S_i associated with a decision tree node i, a tree is built starting at the root node 0 by partitioning the set of examples into two subsets S_L and S_R. The decision how the set S_i is partitioned depends on the parameters θ_i for node i and the features of the samples. These parameters are obtained by optimizing a split criterion function $\hat{\theta}_i = \text{argmax}_{\theta_i} CF_i(S_L, S_R, \theta_i; S_i)$. Examples of such split criteria CF include the *Gini-Impurity* or the *Information Gain*. Important is the *purely local dependency* of the split functions that are usually used – the function which is optimized only depends on the split parameters θ_i of the node i that is optimized and the set of training examples S_i over which this node optimizes and their associated features.

We however propose to optimize a *global* function at each split node. In other words, we condition the split criterion function CF_i on the parameters $\theta_0, ..., \theta_{i-1}$ of all other split nodes of the tree. Intuitively speaking, in each node we optimize the split parameters θ_i of that node, given all split decisions of all other nodes at the current state of the tree. Formally we find the parameters of node i:

$$\hat{\theta}_i = \text{argmax}_{\theta_i} CF_i(S_L, S_R, \theta_i; S_i, \theta_0, ..., \theta_{i-1})$$

In our case we seek to optimize the objective function F over the connected components of a graph G obeying constraints on pairs of nodes. The samples and features of the decision tree consist of edges and their associated features on this graph.

2.2 Decision Tree Building Algorithm

Our algorithm seeks to discriminate object boundaries by their features such that the boundaries satisfy a set of must-link and cannot-link constraints. Tree construction starts by trying to satisfy as many of the given constraints as possible by thresholding a single feature and thereby splitting the edges of the graph into one set that is removed from and another one that remains in the graph. The decision on the split parameters θ_i of node i is optimized by sorting the edges S_i associated with decision tree node i on m_{try} different feature values. For each of the m_{try} features all possible split points are evaluated by first removing all edges S_i associated with the decision tree node from

the graph G. It is important to realize that only the edges of the currently considered split node are removed, the presence and absence of all other edges, as defined by the current state of the decision tree, remains unchanged. In a second step, the edges associated with split node i are re-inserted into the graph one after the other in the order of increasing feature weight. After each edge insertion, its effect on the connected components of the graph is efficiently evaluated using a union find data structure. In addition, we count how many true positive and false positive pairs are generated with respect to the global constraint set $\mathbf{c}$ which specifies which nodes should be in the same component and which nodes should be in different components. Using the TP, FP, TN and FN counts we compute the value of the objective function F and remember the best split position that we encountered while inserting the edges into the graph. The same procedure is executed in descending sort order. After determining the feature and split position that yield the highest objective function value, two child nodes are added to the currently considered node i and the split parameters θ_i of the node are set accordingly. These two child nodes determine the new state of their associated edges until they are further refined in a recursive fashion. The recursive partioning continues until no further improvement in the objective function can be made.

Splitting a node and the associated edge set further thus re-optimizes the state of the edges associated with that node: the final leaf node with which an edge is associated defines the edge state within the tree. This process is illustrated in Figure 2.

It is important to see that during this recursive partitioning the optimization in each leaf node involves only the edges associated with that particular node. The state of the other edges is determined by the already existing leaf nodes and is assumed fixed. Thus each leaf node is optimized conditioned on the graph state given by the current complete decision tree.

While the insertion of edges and its effect on the connected components of the graph can be computed very efficiently, handling edge removal is more difficult. Handling edge removal requires either extremely intricate algorithms [27] or a linear scan over possibly *all* edges in G even though the removed edge set is very small. For this reason we compute the connected components of the graph a single time once all edges associated with a decision tree node are removed. We then trace all changes to the union find data structure and to the TP, FP, TN and FN pair counts caused by inserting an edge into the graph when we test for a split position. This allows us to unwind all changes once the objective function has been evaluated for all split points along one feature and to efficiently begin testing for better splits using the next feature without recomputing the connected component state from scratch.

2.3 Backtracking for greedy global decision trees

It is intuitively clear that the greedy tree building procedure that was outlined in the previous section may get stuck in local minima: when partitioning the edge set recursively, the sets assigned to the leaf nodes quickly get smaller and the edges associated to one decision tree leaf are not necessarily close to each other in the underlying graph. It becomes very likely that the edges in any single leaf are unable to form a linking path between two isolated connected components regardless of their labeling – this implies that it would be impossible to satisfy any constraint that would require linking

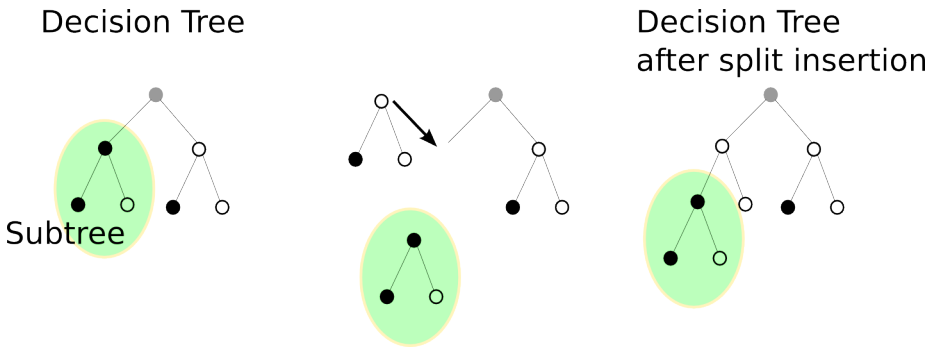

Figure 3: Illustration of split node insertion for backtracking. First a random subtree of the decision tree is selected (green overlay). Then a split node is inserted above this subtree whose split function is optimized under the assumption that the left partition below the inserted node is passed onto the existing subtree, while the right partition is passed to a new leaf node and is assigned either to 0 or 1. The insertion of a split at a higher tree level effectively optimizes over a larger set of samples compared to adding a split ad a leaf node. The insertion split takes away some samples from an existing part of a decision tree and overrides the existing subtree partially.

those components in the current state of the decision tree. Similarily it becomes more unlikely that the edges in small leaves are sufficient to build a cut across a connected component – thus it can become impossible to satisfy any constraint that would require splitting some component into two isolated parts because the edges that could form such a cut are distributed across different leaf nodes of the decision tree. For these reasons, we propose a novel *backtracking scheme* during decision tree building: we allow for split nodes to be *inserted at arbitrary positions* in the tree, not only at the leaves of the tree. Since we allow these nodes to be inserted at any tree level, these split nodes can optimize over a larger set of edges compared to a node at a decision tree leaf. In the extreme case of inserting such a node above the current root of the tree, the new node can reoptimize over all edges of the graph. This novel *insertion split* partitions the edge set arriving at an inserted node into two parts, such that the left partition is passed to the already existing subtree below the inserted split node as before while the right partition is either assigned to $x_{ij} = 0$ or $x_{ij} = 1$. Thus an existing learned combination of rules, defined by the subtree below the inserted node, is partially reused and the decision for a subset of the edges below/above a feature threshold is reconsidered.

The optimization of an inner node of the decision tree is executed in the same manner as already described for a leaf node. The only difference is that when scanning over the edges in the partition in increasing/decreasing feature weight order, not all edges are re-inserted into the graph. Instead, the current state of the edge x_{ij} which is determined by the subtree of the node currently under consideration is used as a mask. In a first trial only edges with current state $x_{ij} = 1$ are re-inserted. This corresponds to overriding the subtree for the edges right (increasing sort order) / left (decreasing sort order) of the split position with a $x_{ij} = 0$ assignment. In a second trial, only the

edges with $x_{ij} = 0$ are removed from the graph before inserting all edges in increasing/decreasing order. This corresponds to overriding the subtree for some edges with a $x_{ij} = 1$ assignment left or right of the split position.

2.4 Decision tree prediction algorithm

Once a decision tree has been built in the described manner, it can be used to determine a segmentation of the graph by predicting the binary indicator x_{ij} for all edges. Prediction proceeds as in any normal decision tree: samples (in our case edges (i, j)) are passed down the tree, starting from the root node 0 by comparing the value w_{ijf} of edge feature f with the split value that is stored in a tree node. Edges with a smaller (larger) feature value are passed to the left (right) child of the current node. Once a leaf node is reached, the x label of this leaf node is assigned to the edge. Now all edges with $x_{ij} = 1$ are switched on in the graph and its connected components are determined.

In an **unsupervised segmentation** setting, the resulting connected components of the graph are the final output.

In a **foreground/background segmentation** setting as shown in Figure 6, the number and type of user labels that are located inside each component are determined and the component is labeled with the winning label. Since not necessarily all induced components contain user given labels, the unlabeled components are assigned a label by determining the closest (node distance) labeled component in the adjacency graph.

3 Experiments

Unfortunately, at present there is no suitable benchmark for the sparse must-link/cannot-link edge learning problem that we propose. To indicate the usefulness of the proposed method we apply our method to a related benchmark dataset [31] and show that our method is applicable to a range of typical unsupervised segmentation problems. Examples of such problems are given in Figure 1 and Figure 4.

Edge features: a range of simple local filters such as Gaussian smoothing, Hessian eigenvalues and Gradient magnitude computed over several scales ($\sigma = 1.0, 1.3, 1.6, 2.5$) have been used as interpixel edge features. To obtain features that can be associated with an interpixel edge, these pixel features have been linearly interpolated from two neighboring pixels.

Postprocessing: When our algorithm satisfies a cannot-link constraint between nodes, it does so by introducing a closed boundary between these nodes. This boundary often consist of many isolated 1-pixel components, as can be observed in Figure 4. This thick boundary is a result of the ambiguity in the data. A simple way to obtain a visually more pleasing segmentation as in Figure 1 is to perform a seeded region growing from all large regions, and to reassign the 1-pixel components to the nearest larger component.

Analysis of training and test error is given in Figure 5. The scores were obtained by sparsely labeling all objects in the images of Figure 1 and Figure 4 and splitting the individual images into two parts. Training was done on the left half and testing on the right half and vice versa. The two examples with inhomogenous boundary appearance

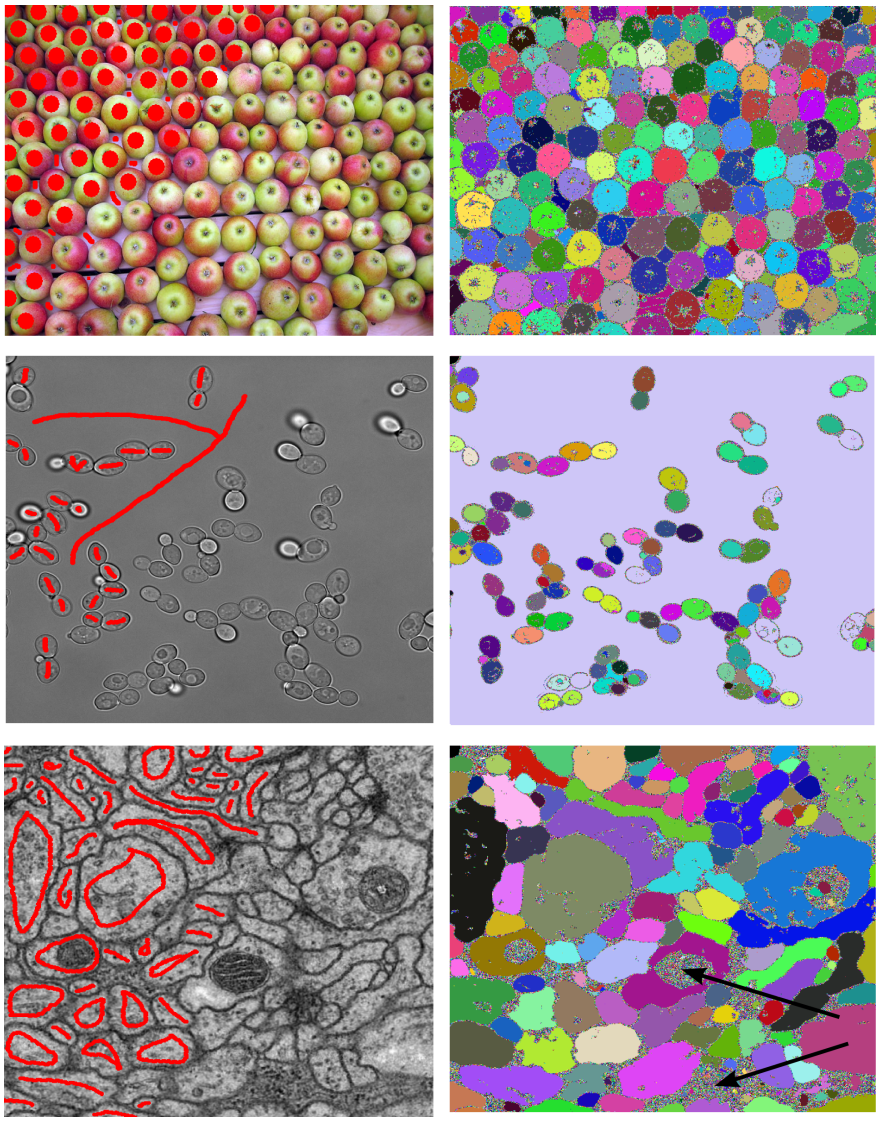

Figure 4: Segmentation examples of different characteristics, all segmented with the same parameter settings. The last two rows show yeast cells in light microscopy, and dendrites in electron microscopy [6, 7], respectively. Individual objects do not differ in appearance and can be discriminated via their boundaries only. These are learned in a weakly supervised fashion from the connectivity constraints that are implicit in the seeds. In contrast to seeded segmentation only a subset of objects need to be marked and the learned classifier can be applied to similar images. Region types that are not represented in the training set (no labels) receive an incoherent prediction (bottom example, arrows).

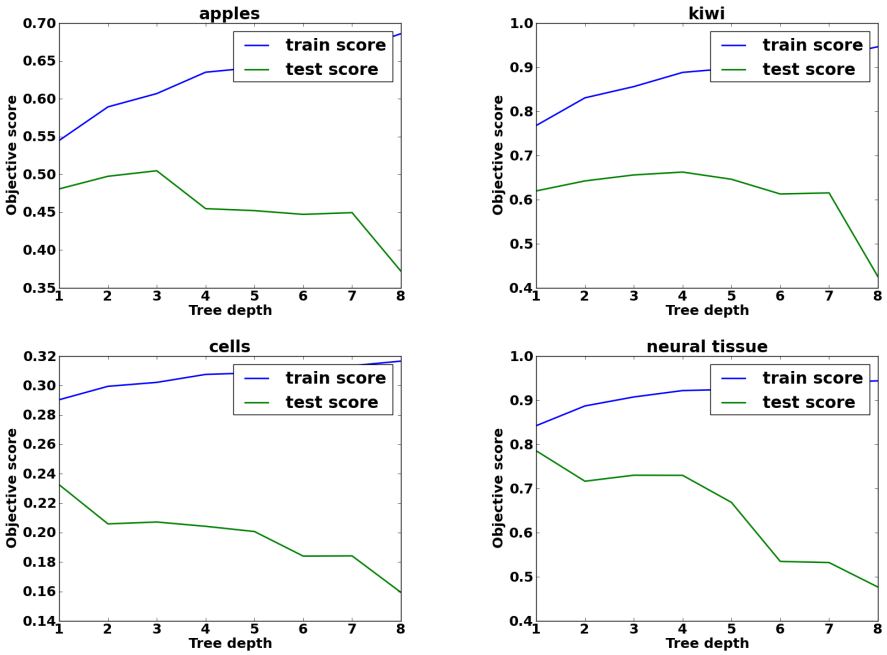

Figure 5: The plots show the training and testing score over the decision tree depth. The two examples with inhomogenous boundary appearance (apples, kiwis) profit from deeper trees, as can be seen from the test score which increases until a depth of 4. The examples with homogenous boundary appearance (cells, neural tissue) do not profit from more decision tree levels - the tree starts to overfit after the first level.

Bai et al. [3]	0.50
Gradi [10]	0.56
Couprie et al.[8]	0.58
our algorithm w/o. insertion splits	0.68
Boykov et al. [5]	0.69
our algorithm	0.71
Unger et al. [29]	0.73
Zhao et al.[33]	0.79

Table 1: Quantitative Evaluation on the LHI interactive segmentation dataset [31]. The dataset consist of several foreground-background segmentation tasks with varying seed quality (see Figure 6 for examples). Results for other algorithms were taken from [33].

(apples, kiwis) profit from deeper trees, as can be seen from the test score which increases until a depth of 4. The examples with homogenous boundary appearance (cells, neural tissue) do not profit from more decision tree levels - the tree starts to overfit after the first level. This ovefitting behaviour of single decision trees is well known. In the future we intend to remedy this problem by training an ensemble of randomized trees which are trained on different subsets of the training data.

Quantitative evaluation: In addition to the qualitative examples, we test our algorithm on the LHI [31] interactive segmentation benchmark. The Benchmark consists of several natural images and provides ground truth and three different types of foreground/background user scribbles with varying difficulty. We adapt the problem to our algorithm by introducing must-link constraints for all foreground-foreground label pairs and all background-background label pairs. In addition we introduce cannot-link constraints for all mixed foreground-background label pairs. We ran the benchmark on all three kinds of user scribbles and calculated the average foreground object precision ($S^+_{gt} \bigcap S^+_{res} / S^+_{gt} \bigcup S^+_{res}$) over all images.

The results show that our purely edge based decision tree achieves a segmentation quality that surpasses many established methods, without learning local class probabilities (unary potentials describing region appearance) from the user labels. These local class probabilities usually work very well on the benchmark images. In addition the other methods rely on a hand-crafted edge probability. We show experimentally that it is possible to achieve the same segmentation quality without relying on local class probabilities and without hand-crafting binary potentials for edges. Our method exclusively relies on the edge probabilities learned from sparse user scribbles.

4 Conclusion

We propose "link-or-cut edge learning", i.e. to learn an edge model from sparse region scribbles interpreted as must-link and cannot-link constraints. To solve this problem, a novel global structured learning scheme based on decision trees is introduced. We explain how decision trees can be trained using a global structured loss criterion and show how they can be used to learn an edge model on an image graph. In addition,

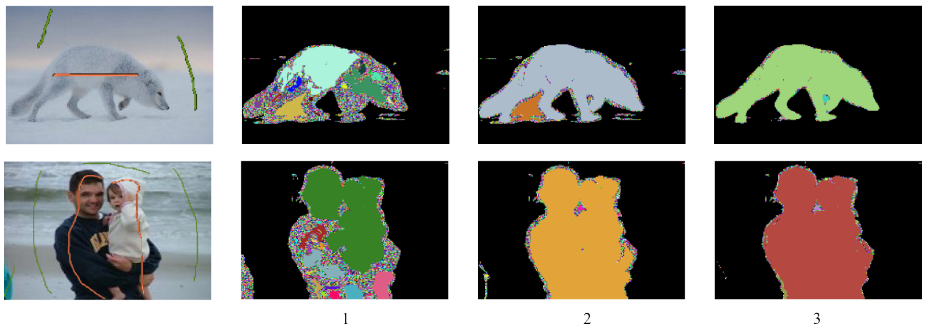

Figure 6: Supervised segmentation example. The images shown are two examples from the LHI interactive segmentation benchmark, also displayed are the benchmark provided scribbles. Our decision tree iteratively partitions the image graph into connected components (indicated by same color). In each tree level (1,2,3) the decisions are refined such that a global objective function over the image graph is optimized that enforces the pairwise connectivity and exclusion constraints that are implicitly defined by the labels.

we show how local minima during tree construction can be overcome by a split node insertion that reuses the already learned decision structure. When applied to interactive foreground/background segmentation problems on natural images, the proposed algorithm produces results comparable to other methods which do rely on local appearance models. The real strength of the proposed method, however, does not lie in the foreground/background segmentation, but in the discrimination of multiple, and possibly similarly-looking foreground objects. Unfortunately the presented approach does have some limitations which we want to discuss. A current limitation is the reliance on axis-orthogonal splits. If there is no single feature that can discriminate the edges around an object relatively well, the algorithm cannot construct a closed cut around this object and cannot escape from this situation – the objective function only improves, if an object is completely separated from its cannot-link partners. The same problem holds for the must-link constraint: if there is no single feature that can be used to build a linking path between two must-link partners, the objective function cannot be improved.

In future work we hope to remedy some of these problems, either by using a relaxed version of the objective function that allows to increase the objective value also by partially separating a node. In addition one could introduce oblique splits that may prevent some of the problems since the algorithm could construct a suitable linear combination of existing features. Another promising avenue is to extend the supervised segmentation learning algorithm using region homogeneity priors.

References

[1] B. Andres, J. H. Kappes, T. Beier, U. Köthe, and F. A. Hamprecht. Probabilistic image segmentation with closedness constraints. In *ICCV*, 2011. 4

[2] S. Bagon. Boundary driven interactive segmentation. In *Information Science and Applications (ICISA)*, 2012. 4

[3] X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV*, 2009. 12

[4] W. A. Barrett and E. N. Mortensen. Interactive live-wire boundary extraction. *Medical Image Analysis*, 1997. 4

[5] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, 2001. 12

[6] A. Cardona, S. Saalfeld, S. Preibisch, B. Schmid, A. Cheng, J. Pulokas, P. Tomancak, and V. Hartenstein. An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. *PLoS biology*, 2010. 10

[7] A. Cardona, S. Saalfeld, J. Schindelin, I. Arganda-Carreras, S. Preibisch, M. Longair, P. Tomancak, V. Hartenstein, and R. J. Douglas. Trakem2 software for neural circuit reconstruction. *PLoS One*, 2012. 10

[8] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In *ICCV*, 2009. 12

[9] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 4

[10] L. Grady. Random walks for image segmentation. *PAMI*, 2006. 12

[11] P. He, X. Xu, and L. Chen. Constrained clustering with local constraint propagation. In *ECCV. Workshops and Demonstrations*, 2012. 4

[12] M. Heiler, J. Keuchel, and C. Schnörr. Semidefinite clustering for image segmentation with a-priori knowledge. In *DAGM*, 2005. 4

[13] V. Jain, B. Bollmann, M. Richardson, D. R. Berger, M. N. Helmstaedter, K. L. Briggman, W. Denk, J. B. Bowden, J. M. Mendenhall, W. C. Abraham, et al. Boundary learning by optimization with topological constraints. In *CVPR*, 2010. 3, 4

[14] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *ICCV*, 2012. 4

[15] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression tree fieldsan efficient, non-parametric approach to image labeling problems. In *CVPR*, 2012. 4

[16] S. Kim, S. Nowozin, P. Kohli, and C. Yoo. Higher-order correlation clustering for image segmentation. NIPS, 2011. 4

[17] S. Kim, S. Nowozin, P. Kohli, and C. Yoo. Task-specific image partitioning. *Image Processing, IEEE Transactions on*, 22(2):488–500, 2013. 4

[18] I. Kokkinos, R. Deriche, O. Faugeras, and P. Maragos. Computational analysis and learning for a biologically motivated model of boundary detection. *Neurocomputing*, 2008. 4

[19] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *PAMI*, 2003. 4

[20] P. Kontschieder, S. Buló, H. Bischof, and M. Pelillo. Structured class-labels in random forests for semantic image labelling. In *ICCV*, 2011. 4

[21] P. Kontschieder, S. Rota, A. Criminisi, H. Bischof, P. Kohli, and Pelillo. Context-sensitive decision forests for object detection. In *NIPS*, 2012. 4

[22] F. Malmberg, R. Strand, and I. Nyström. Generalized hard constraints for graph segmentation. In *Image Analysis*. 2011. 4

[23] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 2004. 4

[24] B. W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 1975. 5

[25] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *ICCV*, 2011. 4, 6

[26] N. Payet and S. Todorovic. Sledge: Sequential labeling of image edges for boundary detection. *International Journal of Computer Vision*, 104(1):15–37, 2013. 4

[27] M. Thorup. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000. 7

[28] S. Turaga, K. Briggman, M. Helmstaedter, W. Denk, and H. Seung. Maximin affinity learning of image segmentation. *arXiv preprint arXiv:0911.5372*, 2009. 4

[29] M. Unger, T. Pock, W. Trobin, D. Cremers, and H. Bischof. Tvseg-interactive total variation based image segmentation. In *BMVC*, 2008. 12

[30] R. Unnikrishnan, C. Pantofaru, and M. Hebert. A measure for objective evaluation of image segmentation algorithms. In *CVPR Workshops*, 2005. 3

[31] B. Yao, X. Yang, and S. Zhu. Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2007. 9, 12

[32] S. Yu and J. Shi. Segmentation given partial grouping constraints. *PAMI*, 2004. 4

[33] Y. Zhao, S. Zhu, and S. Luo. Co3 for ultra-fast and accurate interactive segmentation. In *Proceedings of the international conference on Multimedia*, 2010. 12

[34] S. Zheng, A. Yuille, and Z. Tu. Detecting object boundaries using low-, mid-, and high-level information. *Computer Vision and Image Understanding*, 2010. 4