Published in final edited form as:
IEEE Trans Image Process. 2016 October ; 25(10): 4596–4607. doi:10.1109/TIP.2016.2592704.

Image Segmentation Using Hierarchical Merge Tree

Ting Liu,
Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA

School of Computing, University of Utah

Mojtaba Seyedhosseini, and
Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA

Department of Electrical and Computer Engineering, University of Utah

Tolga Tasdizen, IEEE [Senior Member]
Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA

Department of Electrical and Computer Engineering, University of Utah

Abstract

This paper investigates one of the most fundamental computer vision problems: image segmentation. We propose a supervised hierarchical approach to object-independent image segmentation. Starting with over-segmenting superpixels, we use a tree structure to represent the hierarchy of region merging, by which we reduce the problem of segmenting image regions to finding a set of label assignment to tree nodes. We formulate the tree structure as a constrained conditional model to associate region merging with likelihoods predicted using an ensemble boundary classifier. Final segmentations can then be inferred by finding globally optimal solutions to the model efficiently. We also present an iterative training and testing algorithm that generates various tree structures and combines them to emphasize accurate boundaries by segmentation accumulation. Experiment results and comparisons with other recent methods on six public data sets demonstrate that our approach achieves state-of-the-art region accuracy and is competitive in image segmentation without semantic priors.

Index Terms

Image segmentation; hierarchical merge tree; constrained conditional model; supervised classification; object-independent; ensemble model

I. Introduction

Image segmentation is an important mid-level computer vision problem that has been studied for a long time yet remains challenging. General image segmentation is used as a pre-processing step for solving high-level vision problems, such as object recognition and image classification. In many inter-disciplinary areas, e.g., biological and medical imaging, image segmentation also plays a significant role in helping scientists quantify and analyze image data. While a lot of research has been done to achieve high segmentation accuracy for specific types of images, the quality of image segmentation for general scenes is still less than satisfactory.

In this paper, we introduce a supervised learning based image segmentation framework, namely, the hierarchical merge tree model. Starting with over-segmenting superpixels, we propose to represent the region merging hierarchy with a tree-like constrained conditional model. An ensemble boundary classifier is trained to score each factor in the graphical model. A globally optimal label assignment to the model is computed by minimizing the total energy under the region consistency constraint, and a final segmentation is recovered from the labeling. We also propose an iterative approach that generates various region merging hierarchies and combines them to improve the overall performance via segmentation accumulation. We conduct extensive experiments for empirical validation. Comparisons with other very recent methods on six public data sets show that our proposed method produces state-of-the-art region segmentation results.

We begin with a review of previous related work on general image segmentation methods in Section II. In Section III, we illustrate our hierarchical merge tree as a constrained conditional model and introduce the ensemble boundary classifier. In Section IV, we describe a modification to the hierarchical merge tree model with iterative segmentation accumulation. Experimental results are shown in Section V, in which we compare the segmentation performance of our method with different settings, as well as with other recent state-of-the-art methods. In Section VI, we summarize our current work and discuss the possible improvement for the future.

II. Previous work

There are two different perspectives of image segmentation [1]. One is edge detection, which aims at finding edges between different perceptual pixel groups. The other one is region segmentation, which partitions an image into disjoint regions. Usually, edge detection focuses on assigning a binary label to each pixel with certain confidence indicating if it belongs to an edge or not and does not guarantee closed object contours. Though closed contours and thus regions they encircle can be recovered from edges, such transformation with high accuracy is usually non-trivial. On the other hand, region segmentation seeks to find the cluster membership of each pixel, and closed contours of an object can be trivially generated as the outmost points of a region. Many region segmentation methods also take advantage of the edge detection outputs as boundary cues to help with the search for correct partitioning. Our method belongs to the region segmentation category, and in this section we emphasize reviewing previous related works in this category.

First, we briefly summarize related edge detection works. Early edge detections are mostly based on image derivatives [2], [3] or filter banks responses [4], [5]. More recent works utilize richer information such as colors and textures. One of the most notable works, gPb [1], combines multi-scale local cues and globalized cues via spectral clustering and sets up a benchmark for edge detection and region segmentation research. Taking advantage of supervised learning techniques has also become the recent trend in edge detection. Ren and Bo [6] train a classifier with sparse codes on local neighborhood information and improve the edge detection performance. Dollar and Zitnick [7] propose a structured learning framework using modified random decision forest for efficient edge detection. Seyedhosseini and Tasdizen [8] propose a hierarchical model to capture multi-scale contextual information and achieve state-of-the-art edge detection performance.

Early works on region segmentation seek to directly group image pixels in an unsupervised manner. Belongie *et al.* [9] fit Gaussian mixture models to cluster pixels based on six-dimensional color and texture features. Mean shift [10] and its variant [11] consider region segmentation as a density mode searching problem. A number of works belong to graph partitioning category, which regards an image as a graph with pixels being nodes and edge weights indicating dissimilarity between neighbor pixels. Normalized cuts [12] takes the image affinity matrix and partitions an image by solving eigenvalue problems. Felzenszwalb and Huttenlocher [13] propose to greedily merge two connected components if there exists an inter-component edge weight that is less than the largest edge weights in the minimum spanning trees of both components. Arbelaez *et al.* [1] propose a variant of watershed transform to generate a hierarchy of closed contours. We refer readers to [14] for a comprehensive review of existing methods.

As in edge detection, supervised learning based methods for region segmentation have gained increased popularity in recent years. This trend leads to and is further promoted by a number of publicly available computer vision data sets with human-labeled ground truth [15], [1], [16], [17], [18], [19]. Though unsupervised methods, such as [20], [21], are shown to generate perceptually coherent segmentations, learning segmentation models from supervised data enables much more capability and flexibility of incorporating preference from human observers and leads to many more interesting works.

Following the classic foreground/background segmentation, object-independent segmentation methods seek to partition an image based only on its appearance and do not utilize underlying semantics about the scene or specific information about target objects. Kim *et al.* propose a hypergraph-based correlation clustering framework [22] that uses structured SVM for learning the structural information from training data. Arbelaez *et al.* develop the multi-scale combinatorial grouping (MCG) framework [23] that exploits multi-scale information and uses a fast normalized cuts algorithm for region segmentation. Yu *et al.* [24] present a piecewise flat embedding learning algorithm and report the best published results so far on Berkeley Segmentation Data Set using the MCG framework. Two other recent superpixel-merging approaches are ISCRA [25] and GALA [26]. Starting with a fine superpixel over-segmentation, ISCRA adaptively divides the whole region merging process into different cascaded stages and trains a respective logistic regression model at each stage to determine the greedy merging. Meanwhile, GALA improves the boundary classifier training by augmenting the training set via repeatedly iterating through the merging process. Moreover, impressive results in the extensive evaluations on six public segmentation data sets are reported in [25].

Object-dependent or semantic segmentation is another branch of region segmentation. Object-dependent prior knowledge is exploited to guide or improve the segmentation process. Borenstein and Ullman [27] formulate object segmentation as a joint model that uses both low-level visual cues and high-level object class information. Some other object segmentation methods first generate object segmentation hypotheses using low-/mid-level features and then rank segments with high-level prior knowledge [28], [29]. A recent work, SCALPEL [30], incorporates high-level information in the segmentation process and can generate object proposals more efficiently and accurately. There are also a group of methods, called co-segmentation, that utilizes the homogeneity between different target objects and jointly segments multiple images simultaneously [31], [32], [33].

Our method falls into the object-independent hierarchical segmentation category. A preliminary version of our method with the merge tree model and a greedy inference algorithm appeared in [34], [35] and was only applied to segmenting electron microscopy images, apart from which the contributions of this paper include:

- Reformulation of the hierarchical merge tree as a constrained conditional model with globally optimal solutions defined and an efficient inference algorithm developed, instead of the greedy tree model in [34], [35].
- An iterative approach to diversify merge tree generation and improve results via segmentation accumulation.
- Experiments that extensively compare different variants and settings of the hierarchical merge tree model and show the robustness of the proposed approach against image noise at testing time.
- Experiments with state-of-the-art results on six public data sets for general image segmentation.

Compared with recent competitive hierarchical segmentation methods, ISCRA [25] and GALA [26], which use a threshold-based greedy region merging strategy, our hierarchical merge tree model has two major advantages. First, the tree structure enables the incorporation of higher order image information into segmentation. The merge/split decisions are made together in a globally optimal manner instead of by looking only at local region pairs. Second, our method does not require the threshold parameter to determine when to stop merging as in ISCRA and GALA, which may be so important to the results that needs carefully tuning. Furthermore, our method is almost parameter-free given the initial superpixel over-segmentation. The only parameter is the number of iterations, which can be fixed as shown in the experiments on all the data sets.

III. Hierarchical merge tree model

Given an image I consisting of pixels $\mathscr{P}$, a segmentation is a partition of $\mathscr{P}$, denoted as $S = \{s_i \in 2^{\mathscr{P}} \mid \cup_i s_i = \mathscr{P};\ \forall i \neq j,\ s_i \cap s_j = \varnothing\}$, where $2^{\mathscr{P}}$ is the power set of $\mathscr{P}$. A segmentation assigns every pixel an integer label that is unique for each image object. Each s_i, which is a connected subset of pixels in $\mathscr{P}$, is called a segment or region. All possible partitions form a segmentation space $\mathscr{S}_{\mathscr{P}}$. A ground truth segmentation $S_g \in \mathscr{S}_{\mathscr{P}}$ is usually generated by humans and considered as the gold standard. The accuracy of a segmentation S is measured based on its agreement with S_g. In a probabilistic setting, solving a segmentation problem is formulated as finding a segmentation that maximizes its posterior probability given the image as

$$S^* = \underset{S \in \mathcal{S}_{\mathcal{P}}}{\arg\max}\, P(S \mid I)\,. \quad (1)$$

The current trend to alleviate the difficulty in pixelwise search for S^* is to start with a set of over-segmenting superpixels. A superpixel is an image segment consisting of pixels that have similar visual characteristics. A number of algorithms [12], [13], [36], [37], [38] can be used to generate superpixels. In this paper, we use the watershed algorithm [39] over the output of the boundary detector gPb [1].

Let S_o be the initial over-segmentation given by the superpixels, the final segmentation consisting only of merged superpixels in S_o can be represented as $S = \{s_i \in 2^{\mathcal{P}} \mid \cup_i s_i = \mathcal{P}; \forall i \neq j, s_i \cap s_j = \varnothing; \forall i, \exists S' \in 2^{S_o}, \text{s.t. } s_i = \cup_{s'_j \in S'} s'_j\}$. Therefore, the search space for S is largely reduced to $\mathcal{S} \subseteq \mathcal{S}_{\mathcal{P}}$. Even so, however, exhaustive search is still intractable, and some kind of heuristic has to be injected. We propose to further limit $\mathcal{S}$ to a set of segmentations induced by tree structures and make the optimum search feasible.

A. Hierarchical merge tree

Consider a graph, in which each node corresponds to a superpixel, and an edge is defined between two nodes that share boundary pixels with each other. Starting with the initial over-segmentation S_o, finding a final segmentation, which is essentially the merging of initial superpixels, can be considered as combining nodes and removing edges between them. This superpixel merging can be done in an iterative fashion: each time a pair of neighboring nodes are combined in the graph, and corresponding edges are updated. To represent the order of such merging, we use a full binary tree structure, which we call the hierarchical merge tree (or merge tree for short) throughout this paper. In a merge tree $Tr = (\mathcal{V}, \mathcal{E})$, a node $v_i^d \in \mathcal{V}$ represents an image segment $s_i \in 2^{\mathcal{P}}$, where d denotes the depth in Tr at which this node occurs. Leaf nodes correspond to initial superpixels in S_o. A non-leaf node corresponds to an image region formed by merging superpixels, and the root node corresponds to the whole image as one single region. An edge $e_{ij} \in \mathcal{E}$ between node v_i^d and its child v_i^{d+1} exists when $s_j \subset s_i$, and a local structure $(\{v_i^d, v_j^{d+1}, v_k^{d+1}\}, \{e_{ij}, e_{ik}\})$ represents $s_i = s_j \cup s_k$. In this way, finding a final segmentation becomes finding a subset of nodes in Tr. Fig. 1c shows a merge tree example with initial superpixels shown in Fig. 1a corresponding to the leaf nodes. The non-leaf nodes represent image regions as combinations of initial superpixels. Fig. 1b shows a final segmentation formed by a subset of tree nodes. It is noteworthy that a merge tree defined here can be seen as a dendrogram in hierarchical clustering [40] with each cluster being an image region.

In order to determine the merging priority, we define a merging saliency function $f_{ms} : S^2 \to \mathbb{R}$ that assigns a real number to each pair of regions in S as a measurement of their merging likelihood. For any pair of regions s_i and s_j that are not neighbors, we define $f_{ms}(s_i, s_j) =$

$-\infty$. Then starting from a set of initial superpixels $S = S_o$ as leaf nodes, a merge tree is constructed by iteratively merging $(s_i^*, s_j^*) = \arg\max_{s_i, s_j \in S, i \neq j} f_{ms}(s_i, s_j)$ to a parent node, until only one region remains in S corresponding to the root node. Statistics over the strengths of boundary pixels between two merging regions from boundary detection probability maps may be used as f_{ms}. Following [35], we use negated median

$$f_{ms}(s_i, s_j) = 1 - \mathrm{median}(\{Pb(k) \mid k \in \mathscr{B}(s_i, s_j)\}), \quad (2)$$

where $Pb(k)$ is the value of the k-th pixel on some boundary detection probability map Pb, and $\mathscr{B}(s_i, s_j)$ is the set of boundary pixels between s_i and s_j. $\mathscr{B}$ can be different on implementation. In our work, we define

$$\mathscr{B}(s_i, s_j) = (s_i \cap \mathscr{N}(s_j)) \cup (s_j \cap \mathscr{N}(s_i)), \quad (3)$$

where $\mathscr{N}(s.)$ is the set of neighbor pixels of $s.$. We also propose to learn the merging saliency function from data in Section IV.

B. Constrained conditional model

In order to select a subset of nodes that forms an optimal segmentation, we formulate the merge tree as a constrained conditional model. It is essentially a factor graph for the merge tree, in which the node set aligns identically with $\mathscr{V}$, and each merge in the merge tree that involves three nodes $(\{v_i^d, v_j^{d+1}, v_k^{d+1}\}, \{e_{ij}, e_{ik}\})$is considered as a clique p_i in the factor graph. A label $y_i = +1$ or $y_i = -1$ is assigned to each node indicating whether its children merge or not. All leaf nodes must be labeled +1. A complete label assignment $Y = \{y_i\}_i$ of all nodes must also be subject to the region consistency constraint that if a node is labeled +1, all of its descendants must be labeled +1 as well. Then the nodes whose labels are +1 and parents are labeled −1 are selected as segments in the final segmentation. Fig. 1d is the factor graph for the constrained conditional model derived from the merge tree in Fig. 1c. The red box shows a clique, and a set of consistent labeling is shown.

We train a classifier (Section III-C) to predict the probability $P(y_i)$ for each merge $(\{v_i^d, v_j^{d+1}, v_k^{d+1}\}, \{e_{ij}, e_{ik}\})$. Then we score each clique p_i by associating it with energy with respect to its label

$$E_i(y_i) = -\log P(y_i), y_i = \pm 1 . \quad (4)$$

Under the Markov assumption, we formulate our labeling problem as a constrained optimization problem

$$\min_Y \sum_{y_i \in Y} E_i(y_i), y_i = \pm 1,$$

$$\text{s.t.} \quad y_i = +1, \forall i, v_i^d \text{ is a leaf}, \quad (5)$$

$$y_i \leq y_j, \forall i, j, v_i^d \text{ is parent to } v_j^{d+1},$$

for which an inference algorithm will be introduced in Section III-D.

C. Boundary classifier

To score each clique, we train a boundary classifier to predict the probability of each merge. To generate training labels that indicate whether the boundary between two regions exists or not, we compare both the merge and the split case against the ground truth under certain error metric, such as the Rand error [41] and the variation of information (VI) [42], [43] (See Section V-B for details). The case with smaller error deviates less from the ground truth and is adopted. In practice, we choose VI for its robustness to size rescaling [26].

Boundary features and region features are extracted for classification. For a pair of merging regions, boundary features provide direct cues about how it is likely the boundary truly exists, and regional features measure geometric and textural similarities between the two regions, which can both be informative to boundary classification. We choose features following [25] for comparison purposes. A summary of features is provided in Appendix A. The boundary classifier is not limited to any specific supervised classification model. We use random forest [44] in our experiments. The parameter setting for our random forest is summarized in Appendix B.

The boundary classification problem is highly non-linear, and learning one universally good boundary classifier for all merging cases is essentially difficult. The size of merging regions affects the feature representativeness in classification. For instance, textural features in the form of averaged histograms among patches may not be informative when the merging regions are too small, because textural features can be extracted from only a very limited number of image patches and is thus noisy. On the other hand, when two regions are so big that they contain under-segmentation from different perceptual groups, the features again may not be meaningful, but for a different reason, that is, the histogram averaging is not able to represent the variation of textures. It is worth noting that for the same reason, different classifiers have to be learned at different merging stages in [25].

We categorize the classification problem into sub-problems, train a separate sub-classifier for each sub-problem, and form the boundary classifier as an ensemble of sub-classifiers. We compute the median size $|s|_{\text{med}}$ of all regions observed in the training set and assign a category label to a training sample that involves regions s_i and s_j based on their sizes as in (6). Three sub-classifiers are then trained respectively using only samples with identical category labels.

$$c(s_i, s_j) = \begin{cases} 1, & \text{if } \max(|s_i|, |s_j|) < |s|_{\text{med}}, \\ 2, & \text{if } \min(|s_i|, |s_j|) < |s|_{\text{med}} \le \max(|s_i|, |s_j|), \\ 3, & \text{otherwise.} \end{cases} \quad (6)$$

At testing time, a sample is categorized based on its region sizes and assigned to the corresponding sub-classifier for prediction. Since all the sub-classifiers are always used adjointly, we refer to the set of all sub-classifiers as the boundary classifier in the rest of this paper.

D. Inference

Exhaustive search to solve (5) has exponential complexity. Given the tree structure, however, we can use a bottom-up/top-down algorithm to efficiently find the exact optimal solution under the region consistency constraint. The fundamental idea of the bottom-up/top-down algorithm is dynamic programming: in the bottom-up step, the minimum energies for both decisions (merge/split) under the constraint are kept and propagated from leaves to the root, based on which the set of best consistent decisions is made from the root to leaves in the top-down step. It is noteworthy that our bottom-up/top-down algorithm is only for inference and conceptually different from the top-down/bottom-up framework in [27], which seeks to combine high-level semantic information and low-level image features. On the other hand, the two-way message passing algorithm used in [27] and our algorithm both belong to the Pearl's belief propagation [45], [46] category, except that our inference algorithm explicitly incorporates the consistency constraint into the optimization procedure.

In the bottom-up step, a pair of energy sums are kept track of for each node v_i^d with children v_j^{d+1} and v_k^{d+1}: the merging energy E_i^m of node v_i^d and its descendants all being labeled +1 (merge), the splitting energy E_i^s of it that v_i^d is labeled −1 (split), and its descendants are labeled optimally subject to the constraint. Then the energies can be computed bottom-up recursively as

$$E_i^m = E_j^m + E_k^m + E_i(y_i = +1), \quad (7)$$

$$E_i^s = \min(E_j^m, E_j^s) + \min(E_k^m, E_k^s) + E_i(y_i = -1). \quad (8)$$

For leaf nodes, we assign $E_i^m = 0$ and $E_i^s = \infty$ to enforce their being labeled +1. Fig. 2 illustrates the bottom-up algorithm in pseudocode.

In the top-down step, we start from the root and do a depth-first search: if the merging energy of a node is lower than its splitting energy, label this node and all its descendants +1;

otherwise, label this node −1 and search its children. Fig. 3 illustrates the top-down algorithm in pseudocode.

Eventually, we select the set of the nodes, such that its label is +1 and its parent is labeled −1, to form an optimal final segmentation. In both algorithms, each node is visited exactly once with constant operations, and we need only linear space proportional to the number nodes for $\mathscr{T}_E$ and Y, so the time and space complexity are both $O(|\mathscr{V}|)$.

IV. Iterative hierarchical merge tree model

The performance upper bound of the hierarchical merge tree model is determined by the quality of the tree structure. If all true segments exist as nodes in the tree, they may be picked out by the inference algorithm using predictions from well-trained boundary classifiers. However, if a desirable segment is not represented by any node in the tree, the model is not able to recover the segment. Hence, the merging saliency function, which is used to determine merging priorities, is critical to the entire performance. With a good merging saliency function, we can push the upper bound of performance and thus improve segmentation accuracy.

Statistics over the boundary strengths can be used to indicate merging saliency. We use the negated median of boundary pixel strengths as the initial representation of saliency, as mentioned in Section III-A. Since a boundary classifier is essentially designed to measure region merging likelihood, and it has advantages over simple boundary statistics because it takes various features from both boundary and regions, we propose to use the merging probabilities predicted by boundary classifiers as the merging saliency to construct a merge tree.

As described in Section III-C, the training of a boundary classifier requires samples generated from a merge tree, but we would like to use a boundary classifier to construct a merge tree. Therefore, we propose an iterative approach that alternately collects training samples from a merge tree for the training of boundary classifiers and constructs a merge tree with the trained classifier. As illustrated in Fig. 4a, we initially use the negated median of boundary strengths to construct a merge tree, collect region merging samples, and train a boundary classifier f_b^0. Then, the boundary classifier f_b^0 is used to generate a new merge tree from the same initial superpixels S_o, from which new training samples are generated. We next combine the samples from the current iteration and from the previous iterations, remove duplicates, and train the next classifier f_b^1. This process is repeated for T iterations or until the segmentation accuracy on a validation set no longer improves. In practice, we fix the iteration number to $T = 10$ for all data sets. Eventually, we have a series of boundary classifiers $\{f_b^t\}_{t=0}^T$ from each training iteration. The training algorithm is illustrated in Fig. 5.

At testing time, we take the series of trained classifiers and iterate in a way similar to the training process, as shown in Fig. 4b: at each iteration t, we take the previous boundary classifier f_b^{t-1} to construct a merge tree over the same initial superpixels S_o and use the current classifier f_b^t to predict each merge score in the merge tree, based on which a final segmentation S^t is inferred. Finally, we transform each segmentation into a binary closed contour map by assigning boundary pixels 1 and others 0 and average them for each image over all iterations to generate a segmentation hierarchy in the form a real-valued contour map. The testing algorithm is illustrated in Fig. 6.

The explanation for the iterative approach is two-fold. First, by collecting samples that were not seen in previous iterations, we can explore the merge sample space and in turn explore the space of merge trees generated by the classifiers trained using the augmented sample set towards the "correct" merge tree. Second, like a bagging algorithm, segmentation averaging through iterations tends to emphasize accurate boundaries by phasing out non-systematic errors due to incorrect tree structures or classifier mispredictions. The segmentation accumulation alleviates the difficulty of training one accurate classifier to generate segmentations by improving via averaging.

V. EXPERIMENTS

We conduct experiments with two validation goals. First, we evaluate the performance of our hierarchical merge tree model with different combinations of settings. Second, we compare our method with other state-of-the-art methods.

A. Setting

We experiment with six publicly available data sets for image segmentation:

1. Berkeley Segmentation Data Set 300 (BSDS300) [15]: 200 training and 100 testing natural images of size 481 × 321 pixels. Multiple ground truth segmentations are provided with different labeling of details.
2. Berkeley Segmentation Data Set 500 (BSDS500) [1]: an extension of BSDS300 with 200 new testing images of the same size, with multiple ground truth segmentations for each image.
3. MSRC Object Recognition Data Set (MSRC) [16]: 591 320 × 213 natural images with one ground truth per image. A cleaned-up version [47] is used, in which "void" regions are removed, and disconnected regions that belong to the same object class are assigned different labels in a single image.
4. PASCAL Visual Object Classes Data Set (VOC12) [18]: 1449 validation images with one ground truth per image for PASCAL VOC 2012 Challenge. The average image size is 496 × 360. We use the ground truth for object segmentation and treat the object boundary pixels as background.
5. Stanford Background Data Set (SBD) [17]: 715 approximately 320 × 240 images of outdoor scenes with one ground truth per image.
6. NYU Depth Data Set v2 (NYU) [19]: 1449 indoor scene images with one ground truth per image. Down-sampled versions (320 × 240) [6] are used with frame

pixels cropped. Only RGB channels are used in our experiment; the depth maps are not used.

In order to compare with the other state-of-the-art methods, we follow [25] and train our boundary classifiers with the 200 training images in BSDS300. Five ground truth segmentations are selected for each image in the order of increasing details as indicated by the number of true segments. The training and the testing are done for each detail level, and the results are combined into a segmentation hierarchy. In our performance evaluation of different configurations of the merge tree model, we test on the testing images in BSDS500. For comparisons with other methods, we test on all six data sets.

Appendix A summarizes the features used for boundary classification, most of which follows [25]. Appendix B provides the parameters that we use in our hierarchical merge tree model experiments.

B. Evaluation metrics

Following [1], we use the segmentation covering [18], the probabilistic Rand index [48], and the variation of information [42], [43] for segmentation accuracy evaluation. Here, we summarize the three evaluation metrics. For more details, please refer to [1].

The segmentation covering measures averaged matching between proposed segments with a ground truth labeling, defined as

$$SC(S, S_g) = \sum_{s_i \in S} \frac{|s_i|}{|\mathscr{P}|} \max_{s_j \in S_g} \frac{|s_i \cap s_j|}{|s_i \cup s_j|}, \quad (9)$$

where $\mathscr{P}$ is the set of all pixels in an image. It matches each proposed segment to a true segment, with which the proposed segment has the largest overlapping ratio, and computes the sum of such optimal overlapping ratios weighted by relative segment sizes.

The Rand index, originally proposed in [41], measures pairwise similarity between two multi-label clusterings. It is defined as the ratio of the number of pixel pairs that have identical labels in S and S_g or have different labels in S and S_g, over the number of all pixel pairs.

$$RI(S, S_g) = \frac{1}{\binom{|\mathscr{P}|}{2}} \sum_{i<j} \mathbb{I}(S(i) = S(j) \wedge S_g(i) = S_g(j)), \quad (10)$$

where $S(i)$ is the label of the ith pixel in S, and $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the input condition is met or 0 otherwise. The Rand error is sometimes used to refer $1 - RI$. The probabilistic Rand index is the Rand index averaged over multiple ground truth labelings if available.

The variation of information measures the relative entropy between a proposed segmentation and a ground truth labeling, defined as

$$VI(S, S_g) = H(S|S_g) + H(S_g|S), \quad (11)$$

where $H(S \mid S_g)$ and $H(S_g \mid S)$ are conditional image entropies. Denote the set of all labels in S as $\mathscr{L}_S$ and the set of all labels in S_g as $\mathscr{L}_{S_g}$, we have

$$H(S|S_g) = \sum_{l \in \mathscr{L}_S, l_g \in \mathscr{L}_{S_g}} P(l, l_g) \log \frac{P(l_g)}{P(l, l_g)}, \quad (12)$$

where $P(l_g)$ is the probability that a pixel in S_g receives label l_g, and $P(l, l_g)$ is the joint probability that a pixel receives label l in S and label l_g in S_g. $H(S_g \mid S)$ can be defined similarly by switching S and S_g in (12).

For each data set, segmentation results are evaluated at a universal fixed scale (ODS) for the entire data set and at a fixed scale per testing image (OIS), following [1]. The evaluated numbers are averaged over all available ground truth labelings. As pointed out in [25], since we focus on region segmentation, the pixelwise boundary-based evaluations for contour detection results [1] are not relevant, and we use only the region-based metrics.

C. Ensemble vs. single boundary classifier and constrained conditional model vs. greedy tree model

We evaluate the performance of using single ("SC") or ensemble boundary classifiers ("EC") (Section III-C) with our hierarchical merge tree model. We also compare the proposed constrained conditional model ("CCM") formulation and greedy tree model ("Greedy") previously proposed in [34], [35]. The greedy tree model shares the same hierarchical merge tree structure and scores each tree node only based on local merges the node is involved with, based on which a subset of highest-scored nodes that conform with the region consistency constraint are greedily selected. The training is done using the 200 training images in BSDS300 as described in Section V-A, and we show the testing results on the 200 testing images in BSDS500 in Table I.

A comparison between the first two rows in Table I shows that using ensemble boundary classifiers outperforms using only a single boundary classifier among all metrics, which supports our claim that the classifier ensemble is better able to capture underlying merging characteristics of regions at different size scales.

Comparing the first and the third row, we can see that CCM significantly outperforms the greedy model in terms of VI, which is preferred over the other metrics for segmentation quality evaluation [26]. It appears that CCM is outperformed by the greedy tree model in terms of PRI, but this is because both models are trained using the labels determined based on VI (Section III-C). We perform another experiment where both are trained using the labels determined based on the Rand index, and CCM outperforms the greedy model 0.829 vs. 0.826 in terms of ODS PRI and 0.855 vs. 0.848 in terms of OIS PRI.

The fourth row shows the results using our previous work [35]. It is clear that the proposed constrained conditional model and ensemble boundary classifier are an improvement over our previous approach without including the iterative segmentation accumulation.

D. Non-iterative vs. iterative segmentation accumulation

We evaluate the performance of our hierarchical merge tree model with or without iterative segmentation accumulation (Section IV). The experimental setting follows the previous experiments in Section V-C. The constrained conditional model formulation and ensemble boundary classifiers are adopted. Results at each iteration are shown in Table II.

We can see that despite occasional oscillations, the results are improved through iterations. The rate of improvement slows down as more iterations are included in the averaging process. More sophisticated ways of choosing segmentations to average over can be used, such as to average segmentations only from the iterations that achieve the top accuracy on some validation set. In our experiment, since we would like to compare our method with other methods, we keep the same setting for training and testing data sets and do not use a separate validation set. We fix the iteration number to $T = 10$ and only report the results from averaging all the segmentations.

We also test how the iteration influences the robustness of our method to image noise. Gaussian white noise is added to the BSDS500 testing images. We experiment with different large noise variances $\sigma_n^2 = 0.001$ and $\sigma_n^2 = 0.01$, so that the noise is clearly observable, and the input images are considerably corrupted. The previous model learned with noise-free BSDS300 training images is then used for testing. We observe significant decrease in the strength of gPb boundary detection, so we lower the initial water level to 0.005 from 0.01 (Appendix B) for superpixel generation. We keep all other settings identical to the previous experiment and run the iterative testing (Fig. 6) for $T = 10$ iterations. The results for the first iteration and the last iteration are shown in Table III. Comparing Table III and the corresponding entries in Table II, we can see that when the input images are noisy, the performance from HMT that uses gPb boundary saliency to generate the merge trees are severely degraded. However, with the iterative approach, the HMT performance is significantly improved. This is because the iterative approach enables the use of boundary classifiers that utilize different cues for better merge tree generation than using only boundary detection saliency under the noisy setting. In addition, the iterative segmentation accumulation stabilizes the HMT performance for noisy inputs by smoothing out non-systematic errors.

E. Comparisons with other methods

In this section, we compare our proposed iterative hierarchical merge tree method (CCM + ensemble boundary classifier + iteration, under name “HMT”) with various other state-of-the-art region segmentation methods and benchmarks [1], [25], [26], [23], [49], [22], [24] in very recent years on the public data sets. The results are shown in Table IV. Note that [22] generates a single segmentation instead of contour hierarchies for each image. The OIS evaluations are therefore essentially the same as the ODS results, so we exclude the OIS entries for the sake of clarity. Fig. 7 shows sample testing segmentation results for each data set.

From Table IV we can see that our method is highly competitive and outperforms very recent state-of-the-art methods on some data sets, including BSDS500, which is the most used data set for image segmentation evaluation. It is noteworthy that the generalization of our method is almost as good as ISCRA [25] by being trained only on BSDS (general natural photos) and achieving competitive results on the NYU data set (indoor scene photos). It is also worth pointing out that our hierarchical segmentation framework can be used in combination with other features that can better guide the boundary classification. For example, using the most recent piecewise flat embedding (PFE) [24], we expect the results to be further improved in a manner similar to the results from “MCG” to “PFE-MCG” on BSDS500 in Table IV.

VI. Conclusion

Exhaustive search for optimal superpixel merging in image region segmentation is intractable. We propose a hierarchical image segmentation framework, namely the hierarchical merge tree model, that limits the search space to one that is induced by tree structures and thus linear with respect to the number of initial superpixels. The framework allows the use of various merging saliency heuristics and features, and its supervised nature grants its capability of learning complex conditions for merging decisions from training data without the need for parameter tuning or the dependency on any classification model. Globally optimal solutions can be efficiently found under constraints to generate final segmentations thanks to the tree structure.

We also introduce a modification to the hierarchical merge tree model that iteratively trains a new boundary classifier with accumulated samples for merge tree construction and merging probability prediction and accumulates segmentation to generate contour maps.

For further improvement, the combination of merge trees from each iteration as one single model and its global resolution can be investigated. Furthermore, it would be interesting to study the application of our method to semantic segmentation with the introduction of object-dependent prior knowledge.

Acknowledgments

This work was supported by NSF IIS-1149299 and NIH 1R01NS075314-01. We thank Zhile Ren at Brown University for providing the ISCRA results on the testing data sets. We also thank the editor and the reviewers whose comments greatly helped improved this paper.

Appendix A

Summary of boundary classifier features

We use 55 features from region pairs to train the boundary classifiers, including:

1. Geometry (5-dimensional): Areas of two regions normalized by image area and perimeters and boundary length of two regions normalized by length of the image diagonal.
2. Boundary (4-dimensional): Means and medians of boundary pixel intensities from gPb and UCM [1]. Boundary detector gPb generates probability maps that describe how likely each pixel belong to an image boundary. UCM is the result from post-processing gPb probability maps that depicts how boundary pixels contribute to contour hierarchies in images. The boundary pixels follow the definition in (3).
3. Color (24-dimensional): Absolute mean differences, L_1 and χ^2 distances and absolute entropy differences between histograms (10-bin) of LAB and HSV components of original images.
4. Texture (8-dimensional): L_1 and χ^2 distances between histograms of texton [50] (64-bin) and SIFT [51] dictionary of 256 words. The SIFT descriptors are computed densely, and 8 × 8 patches are used on gray, A, and B channel of original images.
5. Geometric context (14-dimensional): L_1 and χ^2 distances between histograms (32-bin) of the probability maps of each of the seven geometric context labels. The geometric context labels indicate orientations of the surfaces in the images, which are predicted by a fixed pre-trained model provided by [52].

Appendix B

Summary of parameters

We use the watershed algorithm for superpixel generation, for which the water level needs to be specified. In general, lowering the water level reduces under-segmentation by producing more superpixels, which gives us sets of high-precision superpixels to start with, but also increases the computation cost. We fixed the water level at 0.01 for all five datasets (BSDS300/500, MSRC, SBD, and VOC12), except the NYU data set. For the NYU data set of indoor scene images, we observe the decrease in gPb boundary detection strength, so we lower the water level to 0.001. We also pre-merge regions smaller than 20 pixels to their neighboring regions with the lowest boundary barrier, i.e. the median of boundary detection probabilities on the boundary pixels between the two regions.

We train 255 fully grown decision trees for the random forest boundary classifier. To train each decision tree, 70% of training samples are randomly drawn and used. The number of features examined at each node is the square root of the total number of features $(\lfloor\sqrt{55}\rfloor = 7)$. In the experiments, the training data are usually imbalanced. The ratios between the number of positive and negative samples are sometimes considerably greater than 1. Therefore, we assign to each class a weight reciprocal to the number of samples in the class to balance the training.

We fix the number of iterations to $T = 10$ for all data sets for our iterative hierarchical merge tree model.

References

1. Arbeláez P, Maire M, Fowlkes C, Malik J. Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2011; 33(5):898–916. [PubMed: 20733228]
2. Marr D, Hildreth E. Theory of edge detection. Proc. Roy. Soc. London, Series B, Biological Sciences. 1980
3. Canny J. A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 1986; (6):679–698. [PubMed: 21869365]
4. Morrone MC, Owens RA. Feature detection from local energy. Pattern Recognition Lett. 1987; 6(5): 303–313.
5. Freeman WT, Adelson EH. The design and use of steerable filters. IEEE Trans. Pattern Anal. Mach. Intell. 1991; (9):891–906.
6. Ren X, Bo L. Discriminatively trained sparse code gradients for contour detection. Adv. Neural Inform. Process. Syst. 2012:584–592.
7. Dollár P, Zitnick C. Structured forests for fast edge detection; Proc. Int. Conf. Comput. Vision; 2013. 1841–1848.
8. Seyedhosseini M, Tasdizen T. Semantic image segmentation with contextual hierarchical models. IEEE Trans. Pattern Anal. Mach. Intell. 2015; 38(5):951–964. [PubMed: 26336116]
9. Belongie S, Carson C, Greenspan H, Malik J. Color- and texture-based image segmentation using EM and its application to content-based image retrieval; Proc. Int. Conf. Comput. Vision; 1998. 675–682.
10. Comaniciu D, Meer P. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. 2002; 24(5):603–619.
11. Vedaldi A, Soatto S. Proc. Eur. Conf. Comput. Vision. Springer; 2008. Quick shift and kernel methods for mode seeking; 705–718.
12. Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2000; 22(8):888–905.
13. Felzenszwalb PF, Huttenlocher DP. Efficient graph-based image segmentation. Int. J. Comput. Vision. 2004; 59(2):167–181.
14. Zhu H, Meng F, Cai J, Lu S. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. J. Visual Commun. Image Representation. 2016; 34:12–27.
15. Martin D, Fowlkes C, Tal D, Malik J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Proc. Int. Conf. Comput. Vision. 2001; 2:423, 416.
16. Shotton J, Winn J, Rother C, Criminisi A. Proc. Eur. Conf. Comput. Vision. Springer; 2006. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation; 1–15.
17. Gould S, Fulton R, Koller D. Decomposing a scene into geometric and semantically consistent regions; Proc. Int. Conf. Comput. Vision; 2009. 1–8.
18. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascalnetwork.org/challenges/VOC/voc2012/workshop/index.html.
19. Silberman N, Hoiem D, Kohli P, Fergus R. Proc. Eur. Conf. Comput. Vision. Springer; 2012. Indoor segmentation and support inference from RGBD images; 746–760.
20. Cheng B, Liu G, Wang J, Huang Z, Yan S. Proc. Int. Conf. Comput. Vision. IEEE; 2011. Multi-task low-rank affinity pursuit for image segmentation; 2439–2446.
21. Zhu H, Zheng J, Cai J, Thalmann NM. Object-level image segmentation using low level cues. IEEE Trans. Image Process. 2013; 22(10):4019–4027. [PubMed: 23782812]

22. Kim S, Yoo CD, Nowozin S, Kohli P. Image segmentation using higher-order correlation clustering. IEEE Trans. Pattern Anal. Mach. Intell. 2014; 36(9):1761–1774. [PubMed: 26352230]
23. Arbeláez P, Pont-Tuset J, Barron J, Marques F, Malik J. Multi-scale combinatorial grouping; Proc. IEEE Conf. Comput. Vision and Pattern Recognition; 2014. 328–335.
24. Yu Y, Fang C, Liao Z. Piecewise flat embedding for image segmentation; Proc. Int. Conf. Comput. Vision; 2015. 1368–1376.
25. Ren Z, Shakhnarovich G. Image segmentation by cascaded region agglomeration; Proc. IEEE Conf. Comput. Vision and Pattern Recognition; 2013. 2011–2018.
26. Nunez-Iglesias J, Kennedy R, Parag T, Shi J, Chklovskii DB. Machine learning of hierarchical clustering to segment 2D and 3D images. PLoS One. 2013; 8(8):e71715. [PubMed: 23977123]
27. Borenstein E, Ullman S. Combined top-down/bottom-up segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2008; 30(12):2109–2125. [PubMed: 18988946]
28. Carreira J, Sminchisescu C. CPMC: Automatic object segmentation using constrained parametric min-cuts. IEEE Trans. Pattern Anal. Mach. Intell. 2012; 34(7):1312–1328. [PubMed: 22144523]
29. Arbeláez P, Hariharan B, Gu C, Gupta S, Bourdev L, Malik J. Semantic segmentation using regions and parts; Proc. IEEE Conf. Comput. Vision and Pattern Recognition; 2012. 3378–3385.
30. Weiss D, Taskar B. SCALPEL: Segmentation cascades with localized priors and efficient learning; Proc. IEEE Conf. Comput. Vision and Pattern Recognition; 2013. 2035–2042.
31. Joulin A, Bach F, Ponce J. Discriminative clustering for image co-segmentation; Proc. IEEE Conf. Comput. Vision and Pattern Recognition; 2010. 1943–1950.
32. Vicente S, Rother C, Kolmogorov V. Object cosegmentation; Proc. IEEE Conf. Comput. Vision and Pattern Recognition; 2011. 2217–2224.
33. Kim G, Xing EP, Fei-Fei L, Kanade T. Distributed cosegmentation via submodular optimization on anisotropic diffusion; Proc. Int. Conf. Comput. Vision; 2011. 169–176.
34. Liu T, Jurrus E, Seyedhosseini M, Ellisman M, Tasdizen T. Watershed merge tree classification for electron microscopy image segmentation; Proc. Int. Conf. Pattern Recognition; 2012. 133–137.
35. Liu T, Jones C, Seyedhosseini M, Tasdizen T. A modular hierarchical approach to 3D electron microscopy image segmentation. J. Neurosci. Methods. 2014; 226:88–102. [PubMed: 24491638]
36. Levinshtein A, Stere A, Kutulakos KN, Fleet DJ, Dickinson SJ, Siddiqi K. Turbopixels: Fast superpixels using geometric flows. IEEE Trans. Pattern Anal. Mach. Intell. 2009; 31(12):2290–2297. [PubMed: 19834148]
37. Veksler O, Boykov Y, Mehrani P. Proc. Eur. Conf. Comput. Vision. Springer; 2010. Superpixels and supervoxels in an energy optimization framework; 211–224.
38. Achanta R, Shaji A, Smith K, Lucchi A, Fua P, Susstrunk S. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern Anal. Mach. Intell. 2012; 34(11):2274–2282. [PubMed: 22641706]
39. Beucher S, Lantuejoul C. Use of watersheds in contour detection. Int. Workshop Image Process.: Real-time Edge and Motion Detection/Estimation. 1979
40. Duda RO, Hart PE. Pattern classification and scene analysis. Vol. 3. New York: Wiley; 1973.
41. Rand WM. Objective criteria for the evaluation of clustering methods. J. Amer. Stat. Assoc. 1971; 66(336):846–850.
42. Meil M. Comparing clusterings: an axiomatic view; Proc. Int. Conf. Mach. Learn; 2005. 577–584.
43. Yang AY, Wright J, Ma Y, Sastry SS. Unsupervised segmentation of natural images via lossy data compression. Comput. Vision Image Understanding. 2008; 110(2):212–225.
44. Breiman L. Random forests. Mach. Learn. 2001; 45(1):5–32.
45. Pearl J. Reverend bayes on inference engines: A distributed hierarchical approach; Proc. AAAI Conf. Artif. Intell; 1982. 133–136.
46. Kschischang FR, Frey BJ, Loeliger H-A. Factor graphs and the sum-product algorithm. IEEE Trans. Inform. Theory. 2001; 47(2):498–519.
47. Malisiewicz T, Efros AA. Improving spatial support for objects via multiple segmentations; Proc. Brit. Mach. Vision Conf; 2007.
48. Unnikrishnan R, Pantofaru C, Hebert M. Toward objective evaluation of image segmentation algorithms. IEEE Trans. Pattern Anal. Mach. Intell. 2007; 29(6):929–944. [PubMed: 17431294]

49. Donoser M, Schmalstieg D. Discrete-continuous gradient orientation estimation for faster image segmentation; Proc. IEEE Conf. Comput. Vision and Pattern Recognition; 2014. 3158–3165.

50. Malik J, Belongie S, Leung T, Shi J. Contour and texture analysis for image segmentation. Int. J. Comput. Vision. 2001; 43(1):7–27.

51. Lowe DG. Object recognition from local scale-invariant features. Proc. Int. Conf. Comput. Vision. 1999; 2:1150–1157.

52. Hoiem D, Efros AA, Hebert M. Geometric context from a single image. Proc. Int. Conf. Comput. Vision. 2005; 1:661, 654.

Biographies

Ting Liu received his B.S. degree with honors from Northeastern University, China, in 2010 and his Ph.D. degree from the University of Utah in 2016. During his Ph.D. years, he was with the School of Computing and the Scientific Computing and Imaging Institute, University of Utah. He is currently with Google, Inc. His research interests include computer vision, machine learning, and their applications in biological image analysis.

Mojtaba Seyedhosseini received the B.S. degree in electrical engineering from the University of Tehran, Iran, in 2007, and the M.S. degree in electrical engineering from the Sharif University of Technology, Iran, in 2009. He received his Ph.D. from the Scientific Computing and Imaging Institute, University of Utah in 2014. He is now with Google, Inc. His research interests include computer vision, machine learning, statistical pattern recognition, and image analysis.

Tolga Tasdizen (M'98–SM'09) received the B.S. degree in electrical and electronics engineering from Bogazici University, Turkey, in 1995. He received his M.S. and Ph.D. degrees in engineering from Brown University in 1997 and 2001, respectively. He held Postdoctoral researcher and Research Assistant Professor positions with the Scientific Computing and Imaging (SCI) Institute, University of Utah, 2001–4 and 2004–8, respectively. Since 2008, he has been with the Department of Electrical and Computer Engineering, University of Utah, where he is currently an Associate Professor. Dr. Tasdizen is a Utah Science Technology and Research Initiative faculty member with the SCI Institute. His research interests are in image processing, computer vision, and pattern recognition with a focus on applications in biological and medical image analysis. He is a recipient of the National Science Foundation CAREER Award. He is a member of Bio Imaging and Signal Processing Technical Committee of the IEEE Signal Processing Society and serves as an associate editor for the IEEE Signal Processing Letters and BMC Bioinformatics.

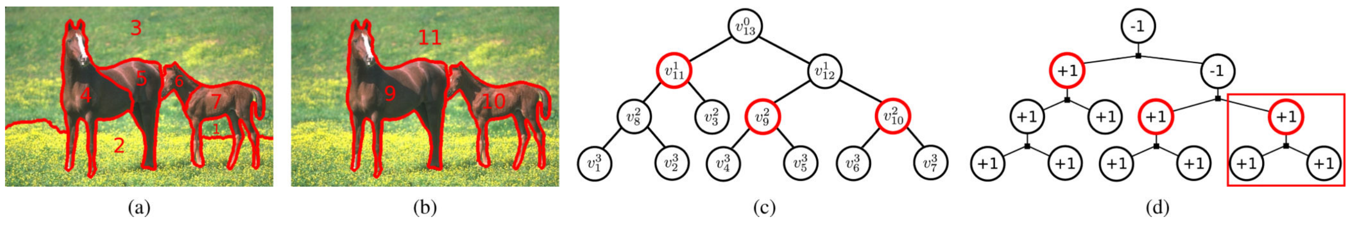


Fig. 1.

Example of (a) an initial segmentation, (b) a consistent final segmentation, (c) a merge tree, and (d) the corresponding conditional model factor graph (Section III-B) with correct labeling. In (c), the leaf nodes have labels identical to those of the initial regions. The red nodes correspond to regions in the final segmentation. The red box in (d) indicates a clique in the model.

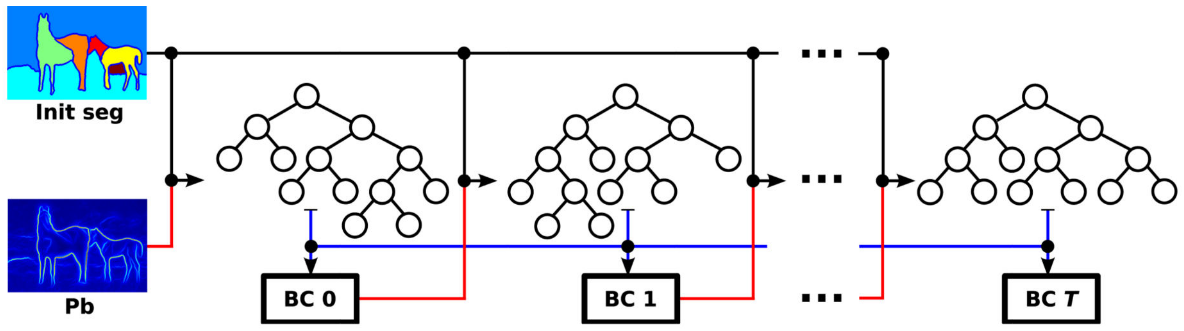
```
Input: A list of energy $\{E_i(y_i)\}_{i=1}^{|\mathcal{V}|}$ for each clique $p_i$
Output: A list of energy tuples $\mathcal{T}_E = \{(E_i^m, E_i^s)\}_{i=1}^{|\mathcal{V}|}$
 1: $\mathcal{T}_E \leftarrow \{\}$
 2: ComputeEnergyTuples($v_r^0$), where $v_r^0$ is the root node
 3: /* Helper function that recursively computes energies */
 4: function ComputeEnergyTuples($v_i^d$):
 5:     if $v_i^d$ is a leaf node then
 6:         $\mathcal{T}_E \leftarrow \mathcal{T}_E \cup \{(0, \infty)\}$
 7:         return $(0, \infty)$
 8:     end if
 9:     $(E_j^m, E_j^s) \leftarrow$ ComputeEnergyTuples($v_j^{d+1}$)
10:     $(E_k^m, E_k^s) \leftarrow$ ComputeEnergyTuples($v_k^{d+1}$)
11:     $E_i^m \leftarrow E_j^m + E_k^m + E_i(y_i = +1)$
12:     $E_i^s \leftarrow \min(E_j^m, E_j^s) + \min(E_k^m, E_k^s) + E_i(y_i = -1)$
13:     return $(E_i^m, E_i^s)$
14: end function
```

Fig. 2.
Pseudocode of the bottom-up energy computation algorithm.

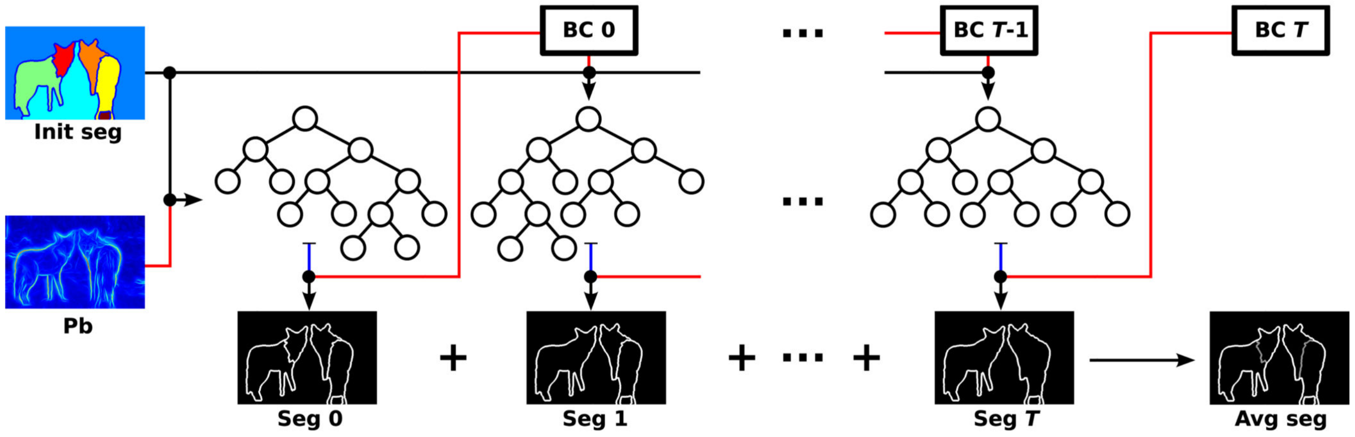
```
Input: A list of energy tuples $\mathcal{T}_E = \{(E_i^m, E_i^s)\}_{i=1}^{|\mathcal{V}|}$
Output: A complete label assignment $Y = \{y_i\}_{i=1}^{|\mathcal{V}|}$
 1: $Y \leftarrow \{\}$
 2: AssignNodeLabels($v_r^0$), where $v_r^0$ is the root node
 3: /* Helper function that recursively decides node labels */
 4: function AssignNodeLabels($v_i^d$):
 5:     if $E_i^m < E_i^s$ then
 6:         $Y \leftarrow Y \cup \{y_i = +1\} \cup \{y_{i'} = +1 \,|\, \forall i' \in \mathcal{D}(i)\}$,
    where $\mathcal{D}(i)$ is set of indices of descendants of $v_i^d$
 7:     else
 8:         $Y \leftarrow Y \cup \{y_i = -1\}$
 9:         AssignNodeLabels($v_j^{d+1}$)
10:         AssignNodeLabels($v_k^{d+1}$)
11:     end if
12: end function
```

Fig. 3.

Pseudocode of the top-down label assignment algorithm.



(a)



(b)

Fig. 4.

Illustrations of (a) the training and (b) the testing procedure of the iterative hierarchical merge tree model. Starting with the fixed initial superpixels (“Init Seg”), the first iteration uses boundary probability (“Pb”) statistics for merge tree generation, and the training procedure iteratively augments the training set by incorporating new samples from merge trees and trains a new boundary classifier (“BC”), which is used for merge tree generation from the same initial superpixels in the next iteration. At testing time, boundary probability statistics and boundary classifiers learned at each iteration are used to generate merge trees from the same initial superpixels, and each boundary classifier is used to score merge cliques in the previous iteration; segmentations are generated from each merge tree and accumulated to generate the final contour hierarchy. The black lines show the use of initial superpixels, the red lines show the use of boundary classifiers, and the blue lines show the flow of sample data collected from tree structures.

```
Input: Original images $\{I_i\}_{i=1}^{N_{\text{tr}}}$, boundary maps $\{Pb_i\}_{i=1}^{N_{\text{tr}}}$,
    and iteration number $T$
Output: Boundary classifiers $\{f_b^t\}_{t=0}^{T}$
 1: Generate initial superpixels $\{S_{oi}\}_{i=1}^{N_{\text{tr}}}$
 2: for $t : 0, 1, \ldots, T$ do
 3:     if $t == 0$ then
 4:         Generate $\{Tr_i^0\}_{i=1}^{N_{\text{tr}}}$ from $\{S_{oi}\}_{i=1}^{N_{\text{tr}}}$ using $\{Pb_i\}_{i=1}^{N_{\text{tr}}}$
 5:     else
 6:         Generate $\{Tr_i^t\}_{i=1}^{N_{\text{tr}}}$ from $\{S_{oi}\}_{i=1}^{N_{\text{tr}}}$ using $f_b^{t-1}$
 7:     end if
 8:     Generate samples $\{(X_i^t, Y_i^t)\}_{i=1}^{N_{\text{tr}}}$ from $\{Tr_i^t\}_{i=1}^{N_{\text{tr}}}$
 9:     Train $f_b^t$ using $\cup_{t'=1}^{t}\{(X_i^{t'}, Y_i^{t'})\}_{i=1}^{N_{\text{tr}}}$
10: end for
```

Fig. 5.
Pseudocode of the iterative training algorithm.

```
Input: Original images $\{I_i\}_{i=1}^{N_{\text{te}}}$, boundary maps $\{Pb_i\}_{i=1}^{N_{\text{te}}}$,
    and boundary classifiers $\{f_b^t\}_{t=0}^{T}$
Output: Hierarchical segmentation contour map $\{C_i\}_{i=1}^{N_{\text{te}}}$
 1: Generate initial superpixels $\{S_{oi}\}_{i=1}^{N_{\text{te}}}$
 2: for $t : 0, 1, \ldots, T$ do
 3:     if $t == 0$ then
 4:         Generate $\{Tr_i^0\}_{i=1}^{N_{\text{te}}}$ from $\{S_{oi}\}_{i=1}^{N_{\text{te}}}$ using $\{Pb_i\}_{i=1}^{N_{\text{te}}}$
 5:     else
 6:         Generate $\{Tr_i^t\}_{i=1}^{N_{\text{te}}}$ from $\{S_{oi}\}_{i=1}^{N_{\text{te}}}$ using $f_b^{t-1}$
 7:     end if
 8:     Score merges with $f_b^t$ and infer segmentations $\{S_i^t\}_{i=1}^{N_{\text{te}}}$
 9:     Binarize $\{S_i^t\}_{i=1}^{N_{\text{te}}}$ to contour maps $\{C_i^t\}_{i=1}^{N_{\text{te}}}$
10: end for
11: $\{C_i\}_{i=1}^{N_{\text{te}}} = \{\sum_{t=0}^{T} C_i^t/(T+1)\}_{i=1}^{N_{\text{te}}}$
```

Fig. 6.
Pseudocode of the iterative testing algorithm.

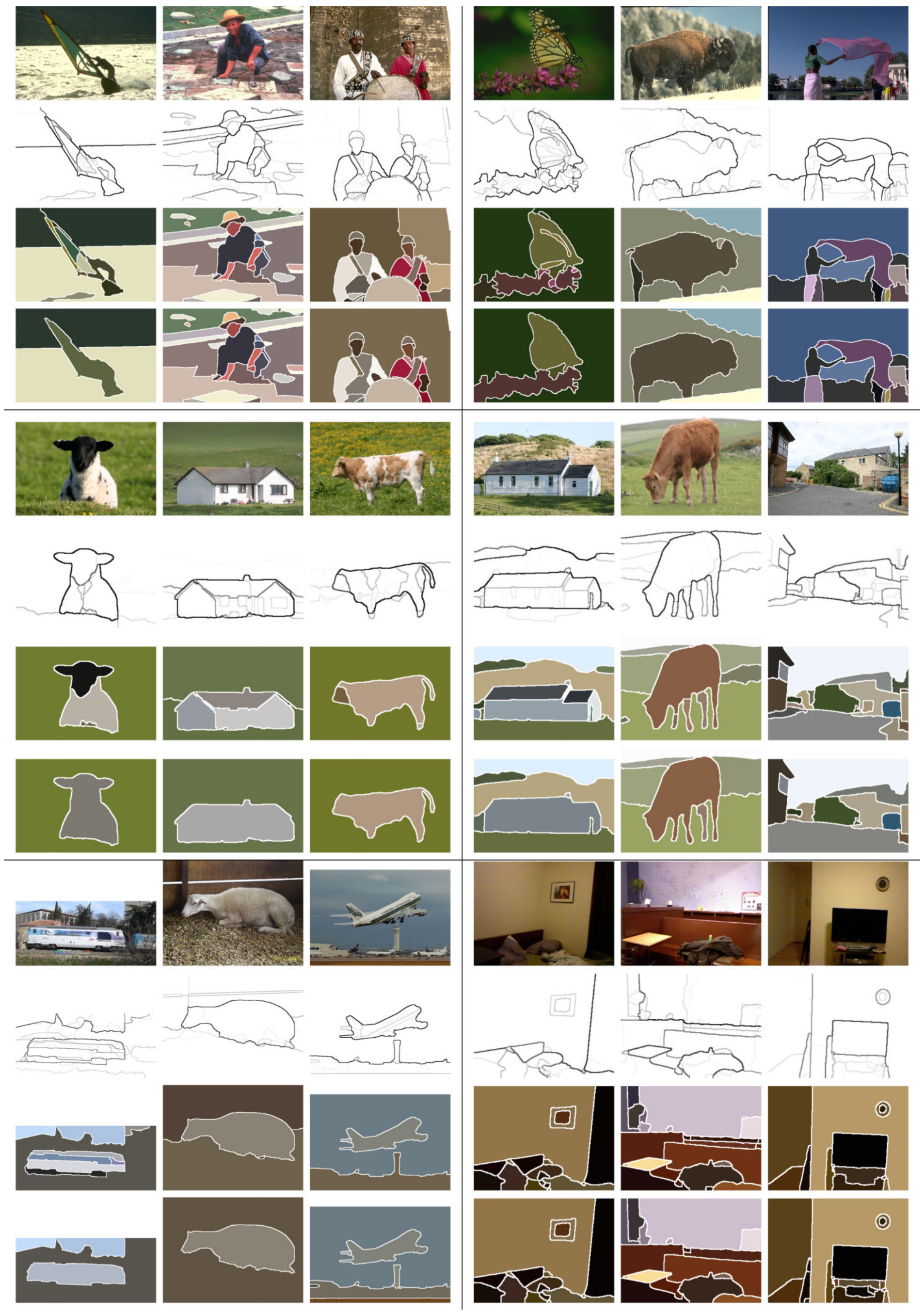


Fig. 7.
Testing segmentation results on BSDS300 (top-left), BSDS500 (top-right), MSRC (middle-left), SBD (middle-right), VOC12 (bottom-left), and NYU (bottom-right) data set. For each image, from top to bottom: the original image, the hierarchical contour map, the ODS covering segmentation, and the OIS covering segmentation. The training uses BSDS300 training images.

TABLE I

Results on BSDS500 using the constrained conditional model (CCM) formulation or greedy tree model (Greedy) [34], [35] in combination with the ensemble boundary classifier (EC) or single boundary classifier (SC). The segmentation covering (Covering), the probabilistic Rand index (PRI), and the variation of information (VI) are reported for optimal data set scale (ODS) and optimal image scale (OIS).

	Covering		**PRI**		**VI**	
HMT variant	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
CCM+EC	0.594	0.607	0.804	0.809	1.682	1.556
CCM+SC	0.573	0.581	0.779	0.781	1.690	1.617
Greedy+EC	0.587	0.620	0.821	0.834	1.737	1.589
Greedy+SC [35]	0.582	0.601	0.805	0.812	1.748	1.639

TABLE II

Results on BSDS500 of hierarchical merge tree model with iterative segmentation accumulation. The segmentation covering (Covering), the probabilistic Rand index (PRI), and the variation of information (VI) are reported for optimal data set scale (ODS) and optimal image scale (OIS).

	Covering		PRI		VI	
Iteration	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
0	0.594	0.607	0.804	0.809	1.682	1.556
1	0.601	0.637	0.825	0.841	1.661	1.498
2	0.612	0.654	0.829	0.853	1.596	1.432
3	0.618	0.666	0.834	0.860	1.564	1.407
4	0.624	0.671	0.834	0.864	1.545	1.391
5	0.624	0.676	0.836	0.865	1.544	1.378
6	0.626	0.678	0.835	0.867	1.539	1.374
7	0.628	0.679	0.835	0.868	1.532	1.373
8	0.628	0.679	0.835	0.869	1.534	1.370
9	0.628	0.680	0.835	0.869	1.530	1.371
10	0.629	0.679	0.835	0.869	1.526	1.375

TABLE III

Results on BSDS500 with Gaussian white noise with different variances σ_n^2 of hierarchical merge tree model with iterative segmentation accumulation. The training uses noise-free BSDS300 training images. The segmentation covering (Covering), the probabilistic Rand index (PRI), and the variation of information (VI) are reported for optimal data set scale (ODS) and optimal image scale (OIS).

		Covering		PRI		VI	
σ_n^2	**Iter.**	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
0.001	0	0.457	0.459	0.587	0.587	1.929	1.917
	10	0.617	0.665	0.836	0.860	1.576	1.411
0.01	0	0.343	0.344	0.394	0.394	2.278	2.271
	10	0.574	0.606	0.805	0.815	1.754	1.581

TABLE IV

Results of different methods on (a) BSDS300, (b) BSDS500, (c) MSRC, (d) VOC12, (e) SBD, and (f) NYU data set. The segmentation covering (Covering), the probabilistic Rand index (PRI), and the variation of information (VI) are reported for optimal data set scale (ODS) and optimal image scale (OIS).

(a)

	BSDS300					
	Covering		**PRI**		**VI**	
Method	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
gPb-OWT-UCM [1]	0.59	0.65	0.81	0.85	1.65	1.47
ISCRA [25]	0.60	**0.67**	0.81	**0.86**	1.61	1.40
HOCC [22]	0.60	-	0.81	-	1.74	-
MCG [23]	**0.61**	**0.67**	0.81	**0.86**	**1.55**	**1.37**
HMT	**0.61**	**0.67**	**0.82**	**0.86**	1.58	1.40

(b)

	BSDS500					
	Covering		**PRI**		**VI**	
Method	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
gPb-OWT-UCM [1]	0.59	0.65	0.83	0.86	1.69	1.48
ISCRA [25]	0.59	0.66	0.82	0.86	1.60	1.42
GALA [26]	0.61	0.67	**0.84**	0.86	1.56	**1.36**
HOCC [22]	0.60	-	0.83	-	1.79	-
DC [49]	0.59	0.64	0.82	0.85	1.68	1.54
MCG [23]	0.61	0.66	0.83	0.86	1.57	1.39
PFE-mPb [24]	0.62	0.67	**0.84**	0.86	1.61	1.43
PFE-MCG [24]	0.62	**0.68**	**0.84**	**0.87**	1.56	**1.36**
HMT	**0.63**	**0.68**	**0.84**	**0.87**	**1.53**	1.38

(c)

	MSRC					
	Covering		PRI		VI	
Method	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
gPb-OWT-UCM [1]	0.65	0.75	0.78	0.85	1.28	0.99
ISCRA [25]	**0.67**	0.75	0.77	0.85	**1.18**	1.02
HMT	**0.67**	**0.77**	**0.79**	**0.86**	1.23	**0.93**

(d)

	VOC12					
	Covering		PRI		VI	
Method	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
gPb-OWT-UCM [1]	0.46	0.59	0.76	0.88	0.65	0.50
ISCRA [25]	**0.50**	0.58	0.69	0.75	1.01	0.93
HMT	0.49	**0.63**	**0.77**	**0.91**	**0.60**	**0.44**

(e)

	SBD					
	Covering		PRI		VI	
Method	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
gPb-OWT-UCM [1]	0.58	0.64	0.86	0.89	1.88	1.62
ISCRA [25]	**0.62**	**0.68**	**0.87**	**0.90**	1.73	1.49
HMT	0.61	0.67	0.86	**0.90**	**1.72**	**1.48**

(f)

	NYU					
	Covering		**PRI**		**VI**	
Method	**ODS**	**OIS**	**ODS**	**OIS**	**ODS**	**OIS**
gPb-OWT-UCM [1]	0.55	0.60	**0.90**	**0.92**	1.89	1.89
ISCRA [25]	**0.57**	**0.62**	**0.90**	**0.92**	**1.82**	**1.63**
HMT	**0.57**	0.61	**0.90**	**0.92**	1.83	1.66