AUGMENTED TREE PARTITIONING FOR INTERACTIVE IMAGE SEGMENTATION

Yangqing Jia[†] *Jingdong Wang*[‡] *Changshui Zhang*[†] *Xian-Sheng Hua*[‡]

[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Automation, Tsinghua University, Beijing 100084, China
[‡]Internet Media Group, Microsoft Research Asia, Beijing 100080, China

ABSTRACT

In this paper, we propose a new fast semi-supervised image segmentation method based on augmented tree partitioning. Unlike many existing methods that use a graph structure to model the image, we use a tree-based structure called the augmented tree, which is built up by augmenting several abstract label nodes to the minimum spanning tree of the original graph. We then model image segmentation as the partitioning problem on the augmented tree. Dynamic programming is used to efficiently solve the optimization problem. Experimental results show that our method gives competitive segmentation results, and the speed is much faster than graph-based methods.

Index Terms— Image segmentation, augmented tree, dynamic programming

1. INTRODUCTION

Image segmentation is one of the fundamental problems in many image processing applications. One wants to to cut out the area of interest, or decompose the image to several blobs for further analysis. Recently, fully automated segmentation techniques have seen a great development, but the results are not as satisfactory as people desire. Thus, in practice, a more popular way is to perform semi-supervised (also called interactive) image segmentation methods, which adopts human aid to perform segmentation. For instance, the user labels parts of the image by marking several pixels or strokes in the image, and the algorithm then segments the image into two or more parts.

One of the representative works on semi-supervised image segmentation is the s/t graph cuts approach [1], which uses a graph structure to model the image and performs segmentation on the graph. There are also some later studies such as [2, 3] that follows the graph cuts thought. The graph cuts approach can get the global optimum for two-label segmentation. But it can only be used to approximately solve multi-label segmentation [1]. Some statistical inference approaches are alternatively used to solve multi-label problems, such as generalized Swendsen-Wang cuts in [4] and generalized belief propagation in [5]. Some other works mainly focus on providing a convenient interactivity interface, such as Lazy snapping [6] and grabcut [7].

However, the shortcoming of the graph-based algorithms is that it involves a polynomial time complexity, often no less than $\Theta(n^2)$, where n is the number of pixels. Thus segmentation of a large image is prohibitive, and even with the implementation of superpixels such as the one in [8], it may still require several seconds to produce the result. This prevents it from giving instant feedback to the user in a interactive image segmentation procedure.

In this paper, we focus on providing a new image segmentation algorithm that has a fast computational speed for real-time use as well as competitive performance against its graph-based counterpart. We first use the minimum spanning tree to approximate the graph structure of the image, then introduce the notion of augmented tree to incorporate label information. An efficient dynamic programming approach is used to find label for all the nodes of the tree. To demonstrate the efficiency of our method, we evaluate our method based on both visual performance and quantitative measures. The results are very promising.

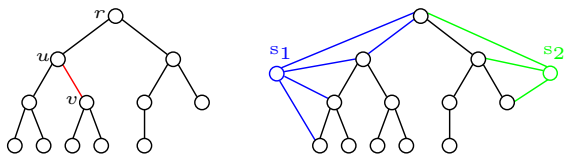
The remainder of this paper is organized as follows. Sec. 2 presents the proposed approach, augmented tree partitioning. Sec. 3 gives the experiments on interactive image segmentation. Sec. 4 concludes this paper.

2. AUGMENTED TREE PARTITIONING

In this section, we first introduce the augmented tree structure to model an image, then discuss the segmentation based on it using dynamic programming.

2.1. Constructing Augmented Tree

A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ defined on an image contains all the pixels (or superpixels, as we will discuss later) as its vertices. Each pair of pixels that are spatial neighbors has an edge connecting them. The length of the edge is computed as the dis-



(a) A tree (b) An augmented tree

Fig. 1. The tree structure and the augmented tree structure.

tance between its corresponding two vertices u and v as:

$$g(u,v) = \|f_u - f_v\|, \tag{1}$$

where f_u and f_v are the RGB value of the pixels.

A general graph may be cyclic and methods based on it often involve a high time complexity. Thus, we use a tree to model the image. A tree structure $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is an acyclic and connected graph that has one root node, and each node other than the root node has a unique parent node. An example of the tree structure is shown in Fig. 1(a). We use the minimum spanning tree criterion to convert the graph to the tree. The Prim's algorithm or Kruskal's algorithm can be implemented to efficiently solve the problem. In a tree, we define $pa(v)$ as the parent node of v, and define $\mathcal{T}_v$ as the subtree rooted from node v. For example, in Fig. 1(a), $\mathcal{T}_v$ is formed of v and its two child nodes. Let $r \in \mathcal{V}$ be the root node, the depth of all other $v \in \mathcal{V}$, denoted as d_v, is the number of the edges on the shortest path from r to v. It is obvious that $d_v = d_{pa(v)} + 1$, which can be easily verified in Fig. 1(a). Also, we define the root node's depth to be 0 by default.

For a k-way segmentation problem, we form an augmented tree by adding several abstract nodes $\{s_i\}_{i=1}^k$ and connecting them with all nodes $\mathcal{V}$ in the tree. Each abstract node can be seen as indicating the kth possible labels. We denote the augmented tree by $\mathcal{T}' = (\mathcal{V} \cup \{s_i\}_{i=1}^k, \mathcal{E} \cup \mathcal{E}_a)$ where $(\mathcal{E}_a = \{(v,s)\})$, $v \in \mathcal{V}$ and $s \in \{s_i\}_{i=1}^k$. An example of the augmented tree is shown in Fig. 1(b). For convenience, we just draw a few augmented edges.

2.2. Augmented Tree Partitioning

A partitioning on an augmented tree is defined as separating the nodes into k disjoint subsets, $\{\mathcal{V}_i \cup \{s_i\}\}_{i=1}^k$, such that $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$, $\cup_{i=1}^k \mathcal{V}_i = \mathcal{V}$, and there are no edges between $\mathcal{V}_i$ and $\mathcal{V}_j$. This is solved by simply removing some edges. To incorporate prior information given by the user, we introduce an additional constraint that the augmented nodes $\{s_i\}_{i=1}^k$ must lie in different subsets. Denote a possible labeling on $\mathcal{V}$ as $L = \{l_v\}$ where l_v is the subset that v belongs to, we aim to find an optimum partition that maximizes the following probability measure:

$$P(L) = \prod\nolimits_v P(s_{l_v}, l_v) \prod\nolimits_v T(l_v | l_{pa(v)}), \tag{2}$$

where $P(s_{l_v}, l_v)$ encodes the likelihood that node $v \in \mathcal{V}$ is connected to s_{l_v} (note that every node is connected to one and only one of the abstract nodes). In our experiment, this likelihood is evaluated by learning a Gaussian mixture model (GMM) in the RGB color space from the labeled pixels. $T(l_v|l_{pa(v)})$ encodes the likelihood of l_v given the label of its parent node, which represents the tree structure. In our method, it is modeled as a Potts model as follows:

$$T(l_v|l_{pa(v)}) = \frac{1}{Z}\begin{cases} 1, & l_v = l_{pa(v)} \\ 1 - \exp(-\lambda g(v, pa(v))), & l_v \neq l_{pa(v)}, \end{cases} \tag{3}$$

where $g(v, pa(v))$ is the distance measure of v and $pa(v)$ as we defined in the prior subsection. Z is the normalization parameter, and λ controls the steepness of the exponential function. In our experiments, λ is set to 1 by default.

An efficient dynamic programming procedure can be adopted to maximize Eqn. (2), and we follow the way similar with [9]. We describe the algorithm in detail.

Consider subtree T_v rooted from node v, we define a function $q_v(l_v)$ with node v's label l_v as the argument:

$$q_v(l_v) = \max_{l_*} p(l_v, l_*), \tag{4}$$

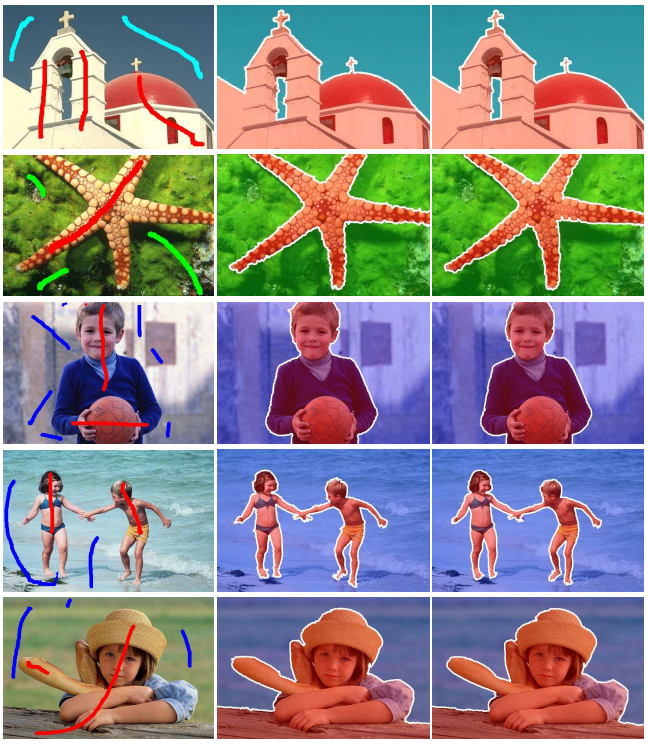
where l_* represents the possible labels of all the nodes in subtree $\mathcal{T}_v$ except node v, and $p(l_v, l_*) = P_{\mathcal{T}_v}(L_{\mathcal{T}_v})$ is the probability measure in subtree $\mathcal{T}_v$. For the internal nodes of the tree, from the Markov and acyclic properties, we obtain the recursive calculation of the function value as:

$$\begin{aligned} q_v(l_v) &= \max_{\{l_w, w \in C_w\}} P(s_{l_v}, l_v) \prod\nolimits_{w \in C_v} T(l_w|l_v) q_w(l_w) \\ &= P(s_{l_v}, l_v) \prod\nolimits_{w \in C_v} \max_{l_w} T(l_w|l_v) q_w(l_w). \end{aligned} \tag{5}$$

It is easy to see that for a leaf v, $q_v(l_v)$ can be evaluated directly as $q_v(l_v) = p(l_v) = P(s_{l_v}, l_v)$. Thus, $q_v(l_v)$ for all the internal nodes and the root node can be evaluated in a recursive bottom-up way. Suppose the maximum depth of the tree is D, the nodes with depth D are just leaves, and their posterior probabilities $q_v(l_v)$ can be directly evaluated as discussed above. Then, we can evaluate $q_v(l_v)$ for all the nodes with depth $D-1$ using Eqn. (5). Similarly, the process is repeated in a decreasing depth order until we reach the root node.

The optimal labeling can be then found in a top-down way from the root node to leaf nodes. The optimal label assignment for root node r can be written as $l_r^* = \arg\max_{l_r} q_r(l_r)$. Then we use the optimal value at root r to find the labels of its children $w \in C_r$ by replacing $\max$ with $\arg\max$ in Eqn. (5). This $\arg\max$ can be recorded in the process of bottom-up posterior probability evaluation. Then we can go down the tree in order of increasing depth to compute the optimal label assignment of each child node w, using the precomputed $\arg\max_{l_w}$.

In summary, our method includes two passes on the tree: the *Bottom-up pass* evaluates the posterior probabilities in a



(a) Original Image (b) ATP (c) GC

Fig. 2. Interactive segmentation for figure-ground separation.

depth decreasing order starting from the leaf nodes, and the *Top-down pass* assigns the optimal labels in a depth increasing order starting from the root node. The dynamic programming procedure to maximize Eqn. (2) takes only $\Theta(nk^2)$ time and consumes only $\Theta(nk)$ memories, which is superior over graph-based methods.

Dynamic programming on a tree has been proved equivalent to belief propagation on the same tree. Recently, several inference methods such as loopy belief propagation and tree reweighted belief propagation [10] generalized the belief propagation for a loopy (cyclic) graph to obtain an approximate local-optimum solution. In our method, the dynamic programming procedure always reaches the global optimum even in the multi-label cases, and the minimum tree fitting step may introduce some approximation error on finding the solution of the original loopy graph. However, although both our method and prior methods do not guarantee global optimality, our advantage is that we have circumvented the difficult and time-consuming graph-based problem by the simple tree structure. This enables our method to bring immediate feedbacks to the user and produce competitive performance in practice, thus is quite suitable for interactive image segmentation. This also indicates that a good solution of the loopy graph model may not be completely equivalent to a good practicable segmentation algorithm.

(a) Original Image (b) ATP (c) α-GC

Fig. 3. Interactive segmentation for multiple object extraction.

2.3. Prior Model on Superpixels

To make tree partitioning more practical, a graph coarsening step can be performed before tree fitting. Particularly, the image graph can be coarsened by building the graph on the superpixels of the image. This will bring at least two advantages: 1) The memory complexity of the graph is reduced, and 2) The time complexities of tree construction and inference on the tree are reduced. However, this poses a new problem of defining the distance g between two superpixels.

We borrow the idea of pairwise predicate presented in [8], and define the distance g between two superpixels C_1 and C_2 based on the external and internal differences as:

$$g(C_1, C_2) = \max\left[d(C_1, C_2)/\mathrm{Int}(C_1), d(C_1, C_2)/\mathrm{Int}(C_2)\right], \tag{6}$$

where the external difference d is defined to be the minimum distance among spatial neighboring pixels:

$$d(C_1, C_2) = \min_{u \in C_1, v \in C_2, (u,v) \in \mathcal{E}} g(u, v), \tag{7}$$

and the internal difference $\mathrm{Int}(C)$ is defined as:

$$\mathrm{Int}(C) = \max_{(u,v) \in MST(C)} g(u, v), \tag{8}$$

size	470 × 318	481 × 321	574 × 384		
ATP	0.02	0.03	0.03	0.03	0.02
GC	0.24	0.32	0.37	0.45	0.46

size	425 × 521	600 × 603	535 × 513	800 × 533
ATP	0.04	0.07	0.04	0.05
α-GC	1.22	1.84	1.41	4.173

Table 1. A quantitative comparison of optimization times. The left table shows image sizes and optimization times (in seconds) of ATP and GC for the images shown in Fig. 2 in the same order, and the right table corresponds to Fig. 3.

where the maximization is done over the edges in the minimum spanning tree $MST(C)$ of the superpixel C.

3. EXPERIMENTAL RESULTS

We demonstrate the proposed method on general-purpose image segmentation. The results based on tree partitioning are all obtained by segmenting the superpixels generated by the watershed algorithm proposed in [11]. The graph structure is constructed by setting the superpixels as the nodes and connecting two superpixels iff they are spatial neighbors. Then, the minimum spanning tree is constructed to approximate the graph. The ranges for RGB values are all normalized to [0, 1]. The running times are given based on an 1.9 GHz Pentium 4 desktop PC without special optimization.

For interactive image segmentation, we follow the user interactivity similar in [1], and let users draw several scribbles to mask the pixels as different objects. Similarly, we set the masked pixels as hard constraints. To impose this, we set $P(i_v|l_v) = 0$ if l_v is not as the label indicated by the user, and otherwise $P(i_v|l_v) = 1$. Figs. 2 and 3 show our results and graph cut (GC) based results both with the same scribbles as shown in (a). In multi-object cases, the iterative α-expansion graph cuts (α-GC) algorithm is adopted for comparison. We have also tested other algorithms such as random walk[12], the results and computation time are similar with graph cut, thus are omitted due to space constraints.

Segmentation results indicate that our approach can get satisfactory performance, and the time consumed in the optimization of ATP and graph cut (the image preprocessing time is not included since it just needs to be performed once on load) demonstrates that augmented tree partitioning is more efficient. For all the images, ATP can return the result in less than 0.1 seconds using dynamic programming and gives almost instant feedback to the user, while graph cuts needs much more time. For multi-object cases, the α-expansion graph cuts algorithm usually takes more than 1 second to compute (note that it still does not guarantee global optimality), which results in a noticeable lag. Thus, we believe that our method can give a more smooth interaction in application.

4. CONCLUSION

In this paper, we presented a new semi-supervised image segmentation algorithm based on augmented tree partitioning and dynamic programming. By using a tree structure to model an image graph, the time complexity of our algorithm is linear to the node numbers. Experimental results have shown that our method produces competitive performance in practice, requiring cheap computational cost.

5. ACKNOWLEDGEMENT

This work is supported by the National 863 Project (No. 2006AA10Z210) of China.

6. REFERENCES

[1] Yuri Boykov and Marie-Pierre Jolly, "Graph Cuts and Efficient N-D Image Segmentation," *Int. J. Comput. Vision*, vol. 70, no. 2, pp. 109–131, 2006. 1, 4

[2] Olivier Juan and Yuri Boykov, "Active Graph Cuts," in *CVPR*, 2006, pp. 1023–1029. 1

[3] Pushmeet Kohli and Philip H. S. Torr, "Effciently Solving Dynamic Markov Random Fields Using Graph Cuts," in *ICCV*, 2005, pp. 922–929. 1

[4] Adrian Barbu and Song-Chun Zhu, "Generalizing Swendsen-Wang to Sampling Arbitrary Posterior Probabilities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1239–1253, 2005. 1

[5] Noam Shental, Assaf Zomet, Tomer Hertz, and Yair Weiss, "Learning and Inferring Image Segmentations using the GBP Typical Cut Algorithm," in *ICCV*, 2003, pp. 1243–1250. 1

[6] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum, "Lazy Snapping," in *ACM SIGGRAPH*, 2004, pp. 303–308. 1

[7] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts," in *Proceedings of ACM SIGGRAPH.*, 2004, pp. 309–314. 1

[8] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, "Efficient Graph-Based Image Segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, 2004. 1, 3

[9] Olga Veksler, "Stereo Correspondence by Dynamic Programming on a Tree," in *CVPR*, 2005, pp. 384–390. 2

[10] Vladimir Kolmogorov, "Convergent Tree-Reweighted Message Passing for Energy Minimization," in *AISTATS*, 2005. 3

[11] L. Vincent and P. Soille, "Watersheds in Digital Spaces: an Efficient Algorithm Based on Immersion Simulations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 6, pp. 583–598, June 1991. 4

[12] Leo Grady, "Random Walks for Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1768–1783, 2006. 4