EFFICIENT STRUCTURE-PRESERVING SUPERPIXEL SEGMENTATION BASED ON MINIMUM SPANNING TREE

Yu Bai, Xuejin Chen

University of Science and Technology of China.

ABSTRACT

We propose a novel superpixel algorithm based on Minimum Spanning Tree (MST), to generate superpixels efficiently while strictly adhere to object boundaries. The MST, which built by gradually removing strong edges of the image graph extracted from the image, is more sensitive to image local structures. Therefore, an efficient hierarchical clustering strategy is basically employed in our algorithm to segment the input image into superpixels based on the tree distance. To gradually merge the image pixels and remove texture noises, a multi-layer scheme with different resolutions of superpixels is proposed. In each layer, the graph is constructed from the lower layer and segmented into superpixels in a linear complexity with the node number in the graph. Because the node number in each layer is exponentially reduced, the computational time of our method mainly concentrates on the first few layers, which is linear with the number of image pixels. The experimental results conducted on the Berkeley Segmentation Dataset demonstrate that our method outperforms state-of-the-art methods both in terms of structure preservation and computational efficiency.

Index Terms— Image Segmentation

1. INTRODUCTION

Most previous superpixel algorithms can be typically divided into *graph-based algorithms* and *grow-based algorithms*. The *graph-based algorithms* start with Normalized Cuts [15], which results segments of similar size, and then improved by Felzenszwalb and Huttenlocher [7], which encourage pixels of similar color to assemble together in greedy way. Veksler and Boykov [18] put initial labeling on image and optimize the regional term of graph-cut energy function, which is further simplified by Zhang et al. [21] by dividing the labeling problem into vertical and horizontal directions. Superpixel Lattice [12] and Lattice Cut [13] both generate regular shape superpixels and preserve the Cartesian grid topology. It is beneficial to understand the high order representation and high level relation among segments. Liu et al. [9] propose a new energy function based on entropy rate of the random work on graph. They have the best performance in homogeneous clusters but time-consuming. Overall, the energy functions defined by graph-based algorithms are always solved globally and fail to catch image local structures. Moreover, the graph-based optimization is NP-hard, which results in a extremely high computational complexity.

The *grow-based algorithms* usually begin with uniform cluster centers, and iteratively update the superpixels for ensuring each pixel is assigned to its most similar cluster center. However, no matter Euclidean distance of color and ordinate space used by Mean-shift [5] and SLIC [1], or Geometric distance used by Levinshtein et al. [8] and Zeng [19], is not structure-aware enough to generate good structure-preserving superpixels. SEEDS [16] is not grow-based algorithm in strict sense. It starts from regular grids, and recursively exchanges the pixels between neighboring patches. It focuses on encouraging the color homogeneity in each superpixel but less successful in protecting image structures.

Recently, a novel measurement tree distance is emerged from Bao et al. [2] because of its excellent structure-sense and efficiency. Firstly, edges of large weights will be removed when MST is built, thus the tree distance defined in the MST is a fine edge-aware metric for describing the similarities between pixels. Secondly, if a graph has Cartesian grid topology, an MST can be extracted from a 1-megapixel 8-bit image in about 70 ms on an Intel 3.4 GHz Core i7 CPU [2].

Therefore, an MST-based structure-preserving superpixel algorithm is proposed in this paper. The proposed algorithm consists of single layer and multi-layer segmentation. With extracted MST from image, we can use a simple and efficient strategy to obtain a good partition of tree in single layer segmentation using tree distance. A sampling approach is proposed to uniformly select cluster centers in the MST based on Poisson disc sampling [3], and a nearest neighbor approach is then used to categorize the rest of nodes in the graph.

However, the superpixels generated from the single layer segmentation are easily influenced by texture accumulation. Therefore, a multi-layer segmentation scheme is proposed to smooth the image textures. Each superpixel generated from the previous layer is averaged and treated as a node of current layer. A graph is extracted from current layer and is partitioned into coarser superpixels based on the single layer segmentation. The nodes number in each layer reduces exponentially, and thus the computational cost mainly resides in the first few layers and is linear to the image size. The pro-

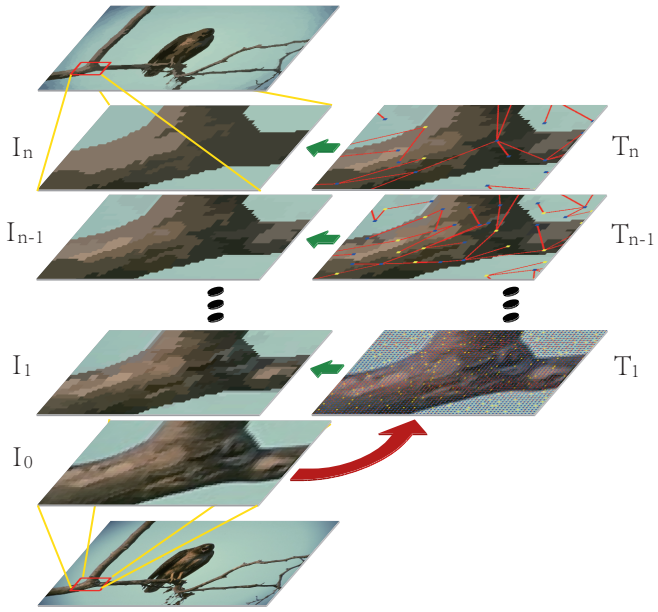


Fig. 1. Algorithm overview: I_0 is input image and I_n is output superpixels with desired number, see Sec. 2 in details.

posed superpixel extraction algorithm runs at around 26FPS on an Intel I7 4.0GHz CPU. It is faster than state-of-the-art algorithms and better preserves object boundaries.

2. OVERVIEW

Our algorithm generates superpixels by gradually merging the image pixels based on the multi-layer strategy, as showed in Figure 1. Given an input image I_0, we firstly build an MST T_1 for layer L_1 (is described in Sec. 3). Based on T_1 and a specified seed distance $\hat{D}_{seed}$, we cluster the pixels into K_1 superpixels based on our single layer segmentation (as will be detailed in Sec. 4). We next average the color in each superpixel to generate I_1. For any layer L_i ($i > 2$), an undirected graph G_i can be extracted from the avaraged image I_{i-1} given by L_{i-1}. After that an MST T_i is generated from the graph and an approach similar to the single layer segmentation is performed to generate K_i superpixels and the averaged image I_i (see more details in Sec. 5). We adopt an consistently increasing seed distance $\hat{D}_{seed}$ in each layer to iteratively reduce the superpixels size. A bisection scheme is used in the final layer L_n to choose an appropriate $\hat{D}_{seed}$ to produce the desired number of superpixels I_n.

3. TREE DISTANCE

The superpixel algorithm proposed in this paper relies on two types of distance measurements defined on the MST. Before we introduce the details of our clustering method, we firstly explain the definitions of MST and the two distances.

Minimum Spanning Tree from Image Given an input image I of N pixels, an undirected graph $G = (V, E)$ can be obtained directly from the image by treating the image pixels as nodes $V = \{p_i\}_{i=1,\ldots,N}$ and the edges between the 4-connected neighboring pixels as the edges E. The edge weights are computed from the maximum color differences between two neighboring pixels p and q: $e(p,q) = \max_{c\in\{R,G,B\}} \|I_c(p) - I_c(q)\|$. From the undirected image graph G, a minimum spanning tree T can be extracted efficiently and linear to image size [6].

Spatial distance in MST The *spatial distance* $D_S(p,q)$, firstly introduced in [2], is defined as the path length between two nodes (p,q) on the MST. $D_S(p,q)$ = 1 if p and q are adjacent nodes.

Weighted distance in MST The *weighted distance* $D_W(p,q)$ between two nodes p and q in an MST is defined as the sum of weights of the connected edges on the path between p and q. $D_W(p,q) = e(p,q)$ if p and q are adjacent pixels.

Spatial distance and weighted distance in the MST are both edge-aware measurement, and can be computed efficiently in recursive manner in linear complexity to node numbers of the MST (proposed by [2]). Figure 2 (b)-(c) present the spatial distances of all the image pixels to the red and green pixels, respectively. Figure 2 (d)-(e) present the corresponding weighted distances. For pixels in a homogeneous region, the weighted distance does not consider the spatial relation in image. As shown in Figure 2(e), a large amount of pixels in the sky region will have the same distance to the green pixel. In contrast, the spatial distance in Figure 2(c) has a higher performance in homogeneous region. Nevertheless, the spatial distance is not sensitive to boundaries enough. Compared to Figure 2 (b), the weighted distance in Figure 2 (d) better presents the color difference between the branch and sky.

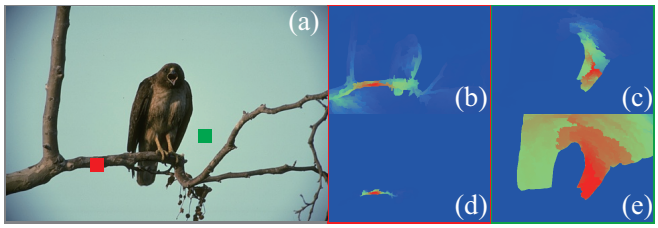


Fig. 2. Comparison between two types of distance. (a) Input image with two marked pixels. The red pixel is located on a thin and long branch. The green pixel is located on a homogeneous region. (b) (c): spatial distance maps from all the image pixels to the red and green pixels, respectively. (d) (e): weighted distance maps from all the image pixels to the red and green pixels, respectively. In the distance maps, red color indicates a smaller distance while the blue color indicates a larger distance, and the green color is in-between.

4. SINGLE LAYER SUPERPIXELS

Given an MST, the single layer segmentation can be divided into two steps: *seed sampling* and *node clustering*. In the seed sampling step, we uniformly select a set of seeds on the MST as the cluster centers based on MST spatial distances. In the clustering step, the remaining nodes are clustered to their nearest seeds based on both MST spatial and weighted distances.

4.1. Seed sampling

In order to make the initial seeds uniformly distribute on the MST, a similar method with Poisson disc sampling [3] is implemented. From Figure 2, the MST spatial distance is more reasonable to measure the spatial uniformity, thus it is used as the uniform measurement to sample seeds on MST.

Denote $\hat{D}_{seed}$ as the seed distance (similar to the sampling radius in Poisson sampling process), the seed nodes can be selected on the MST through an iterative searching

1. **Initialization**: Two empty lists $\mathbb{L}_{seed}$ and $\mathbb{L}_{candidate}$ will be created for storing the sampled seeds and candidate nodes respectively. The root node of the MST will be firstly pushed to $\mathbb{L}_{candidate}$.

2. **Seed sampling**: Pop a node from $\mathbb{L}_{candidate}$ and let p denote this node. Check if p is whin distance $\hat{D}_{seed}$ of existing seeds when $\mathbb{L}_{seed}$ is not empty. If not, insert p (regard as a new seed) into $\mathbb{L}_{seed}$ and do a *depth-first search process* from p to find the new candidate nodes.

   For each node q in the searching path, we progressively compute the MST spatial distance $D_S(p,q)$ between p and q. Once $D_S(p,q) > \hat{D}_{seed}$, push q into $\mathbb{L}_{candidate}$ and stop searching on this branch. When the depth-first search ends, all the potential candidates related to p will be pushed into $\mathbb{L}_{candidate}$.

3. **Termination**: Repeat the seed sampling step until $\mathbb{L}_{candidate} = \emptyset$. All the sampled seeds will be stored in $\mathbb{L}_{seed}$.

4.2. Node Clustering

In the clustering step, we firstly give each sampled seed a label, valued by its index in $\mathbb{L}_{seed}$, then cluster the remaining nodes on the MST based on a nearest neighbor strategy measured by a clustering distance $D_{cluster}(p,q)$. For each node p, we assign it the same label with its closest seed as

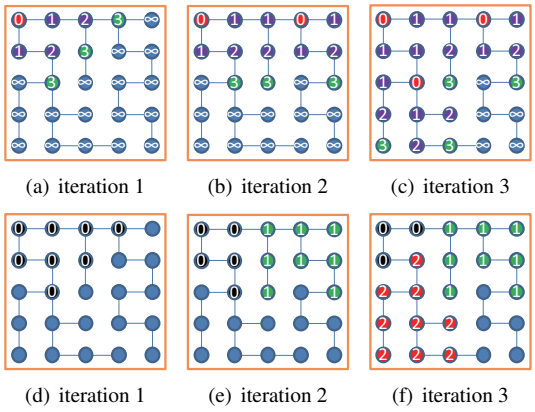
$$L(p) = L(\text{argmin}_{q \in \mathbb{L}_{seed}} D_{cluster}(p,q)), \tag{1}$$

Based on the advantages of the MST spatial distance (considers spatial relation) and weighted distance (presents color difference) described in Sec. 3, we combine them for measuring the clustering distance as follows:

$$D_{cluster}(p,q) = \sum_{<p_a,p_b> \in Path(p,q)} \max(D_W(p_a,p_b), 1), \tag{2}$$

where $< p_a, p_b >$ is a pair of connecting nodes in the path $Path(p,q)$ on the MST.

We use an array $D_{min}(q)$ to record the $D_{cluster}(p,q)$ between each node q and its nearest seed p, and another array $L(q)$ to record the label of p. Since both the spatial distance and the weighted distance between any pair of nodes are computed progressively on the same path, we can compute both the $D_S(p,q)$ and $D_{cluster}(p,q)$ in *depth-first search process* from seed p and gradually update $D_{min}(q)$ and $L(q)$, as showed in Figure 3. Because only the distances from each seed to its nearby pixels are computed, the complexity of distance estimation is linear to number of nodes in MST.



(a) iteration 1 (b) iteration 2 (c) iteration 3

(d) iteration 1 (e) iteration 2 (f) iteration 3

Fig. 3. Sampling and clustering in three iterations when $\hat{D}_{seed} = 2$. (a)-(c): The sampled seeds are marked in red. The seed candidates are marked in green. The purple nodes are those have been visited during the depth-first searching, $D_{min}(q)$ are showed inside the nodes. (d)-(f): The label values $L(q)$ of nodes update in each iteration.

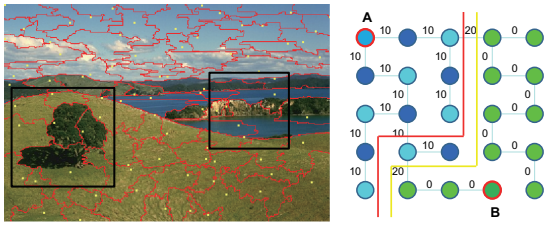
4.3. Bisection method

The number of sampled cluster centers monotonously depends on the $\hat{D}_{seed}$ in the seed sampling process. We cannot perfectly control the generated cluster numbers by analytically computing an exact $\hat{D}_{seed}$. A *bisection method* is thus employed to find the appropriate $\hat{D}_{seed}$ to generate the desired $\hat{K}$ clusters.

Let $\hat{D}_{seed} \in \{D_{min}, D_{max}\}$. We iteratively set $\hat{D}_{seed} = \frac{D_{min}+D_{max}}{2}$ and run the segmentation process that generates K^* clusters. If $K^* < \hat{K}$, we set $D_{max} = \hat{D}_{seed}$; otherwise, $D_{min} = \hat{D}_{seed} + 1$. The iteration terminates when $D_{min} = D_{max}$ or $K^* = \hat{K}$. D_{min} is initialized to 0 and D_{max} should be large enough and thus is initialized to image size.

4.4. Discussion

However, the image textures sometimes mislead the nearest neighbor strategy and cause undesirable results, as showed in black squares of Figure 4 (a). Because the MST weighted distance will be significantly accumulated at texture regions and is very likely to be much larger than the color difference of the real boundary. The problem is illustrated in Figure 4 (b), the superpixel boundary (red line) generated by nearest neighbor strategy is deviated from the real boundary (yellow line). Therefore, an multi-layer approach is proposed in Sec. 5 to gradually cluster the pixels and smooth the texture regions.



(a) (b)

Fig. 4. (a): False segmentation caused by textures noises (100 superpixels), the yellow points are sampled seeds. (b): A, B are two seeds, red line is superpixel boundary generated by nearest neighbor strategy, and yellow line is the real boundary.

5. MULTI-LAYER SEGMENTATION

Our multi-layer segmentation mainly consists of two step: *connecting neighboring layers* and *segmenting each layer*. The connecting step generates a graph from the averaged image given by previous layer and extracts the MST. The segmenting step clusters the nodes of the MST and produces an averaged image in current layer.

5.1. Connecting neighboring layers

Given an N pixels image I, assumed that I has been segmented into K_{r-1} superpixels in the previous layer L_{r-1}, $\mathcal{N}_{r-1}(p)$ is the number of pixels belong to each superpixel p ($\mathcal{N}_0(p) = 1$ and $K_0 = N$ to original image). For the current layer L_r, we build a new undirected graph G by setting each superpixel of L_{r-1} as a node p. The edges are constructed according to the spatial neighborhood relations between the superpixels.

The edge weight between two neighboring nodes p and q in graph G is defined as

$$e_r(p,q) = \|\bar{I}_{r-1}(p) - \bar{I}_{r-1}(q)\|, \tag{3}$$

where $\bar{I}_{r-1}(p)$ is the averaged color of all the pixels in superpixel p in previous layer L_{r-1}. Finally an MST is extracted from the graph G and is used as the input to L_r.

5.2. Segmentation at each layer

In multi-layer scheme, the segmentation conducted in each layer L_r takes the same method as single layer segmentation except for the use of MST spatial distance $D_S^r(p,q)$. For encouraging to sample seeds and produce superpixels uniformly in image, $D_S^r(p,q)$ is not longer equals to the length of the path $P_r(p,q)$ in the MST, but considers the number of pixels in each node (each superpixel in previous layer) on the path $P_r(p,q)$. So that the MST spatial distance is redefined as

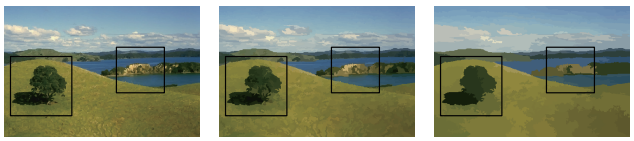
$$D_S^r(p,q) = \sum_{v \in P_r(p,q)} \mathcal{N}_{r-1}(v), \tag{4}$$

v is any node on the path $P_r(p,q)$. Based on the new MST spatial distance, the segmentation method conducted in layer L_r will generate K_r superpixels.

5.3. Parameters

The size of the output clusters monotonically decreases when the seed distance $\hat{D}_{seed}$ increases. In order to converge the number of superpixels exponentially, we set $\hat{D}_{seed} = k \times d^{r-1}$ for each layer L_r, where d controls the growing speed of $\hat{D}_{seed}$ (the decreasing speed of the number of superpixels), and k controls the initial value of $\hat{D}_{seed}$ in the first layer. Let $\hat{K}$ denote the desired number of superpixels. As the layer r increases, there must be a layer l that satisfy $K_l > \hat{K}$ and $K_{l+1} < \hat{K}$ because of the monotonic property. Then we use the bisection method to generate $\hat{K}$ superpixels in L_{l+1}, which is regarded as the final layer in our algorithm.

Since the color of each superpixel has been averaged in each layer, the texture regions are gradually smoothed, as showed in Figure 5. And the false segmentation problem (Figure 4 (a)) is efficiently solved by multi-layer strategy (Figure 5 (c)) with the same superpixels size.



(a) (b) (c)

Fig. 5. Hierachical results for generating 100 superpixels.

Because the number of nodes decreases exponentially w.r.t the increase of r, the computational cost of proposed multi-layer method mainly resides in the first few layers and is linear to the image size. The small d meticulously smooth the image texture, but reduce the nodes number slowly and raise the runtime. The large k favor to reduce the nodes number quickly, but dampen the segmentation performance in first layer. We balance the parameters and adopt $d = 3, k = 9$ for experiments to compare our superpixel method with the state-of-the-art methods.

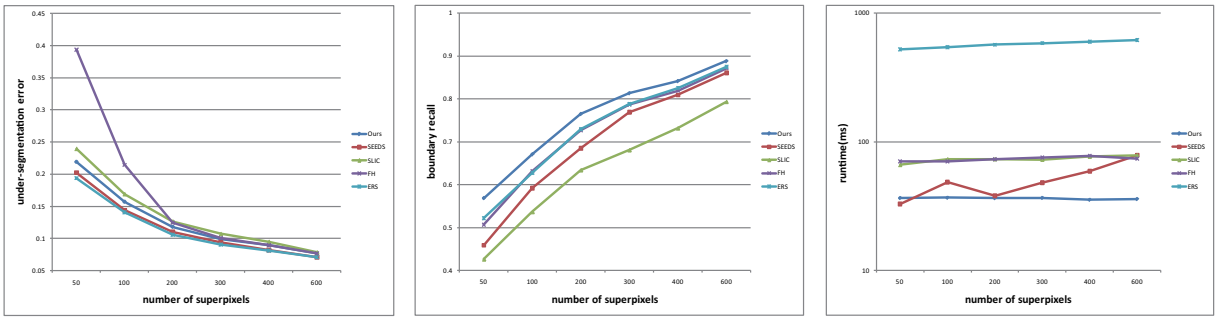


Fig. 6. Quantitative evaluation using BSD data sets. The experiments were conducted on an Intel I7 4.0GHz CPU.

6. EXPERMENTS

In this section, the evaluation benchmark of superpixel algorithm is firstly introduced. Then, four recent state-of-the-art superpixel segmentation methods including ERS [9], SLIC [1], SEEDS [16] and FH [7] are numerically and visually compared with the proposed method.

6.1. Benchmark

The 200 test images of Berkeley Segmentation Dataset (BSD) [10] are used in our experiments. Two standard measurements, boundary recall (BR) and under-segmentation error (UE), are used to evaluate the performance of different superpixel algorithms. Neubert and Protzel [14] propose a standard implementation of boundary recall for eliminating the confusion of different definition of parameters. They also summarize the previous definition of under-segmentation error and propose a new version for reducing serious penalty by the large superpixels overlap the ground truth segment with a small region. This paper uses the open source code provided by Neubert and Protzel to compute the BR and UR for each method for quantitative evaluation.

6.2. Comparison

Figure 6 presents the boundary recall, under-segmentation error and running time of each method computed from 200 test images. It shows that our superpixel algorithm outperforms all the other state-of-the-art algorithms on boundary recall and time efficiency. Our algorithm concentrates on catching the local detail of image, which the manual segmentation benchmark pay less attention to. Thus our segmentation error is slightly higher than ERS and SEEDS (no more than 0.02), but our boundary recall is higher than ERS and SEEDS (more than 0.04) and our runtime is the most efficient.

Figure 7 presents visual comparison with the state-of-the-art. It shows that the proposed method has a higher performance on preserving image structures. FH is a good structure-preserving superpixel algorithm as well, but it can easily merge large area of background pixels into a single superpixel if they have similar color (c). SEEDS might produce strip-like error (the sky area in (a)). In contrast, the proposed method has advantage on preserving the local detail in image such as human face (e) and water (g), and preserving long-thin structures (i).

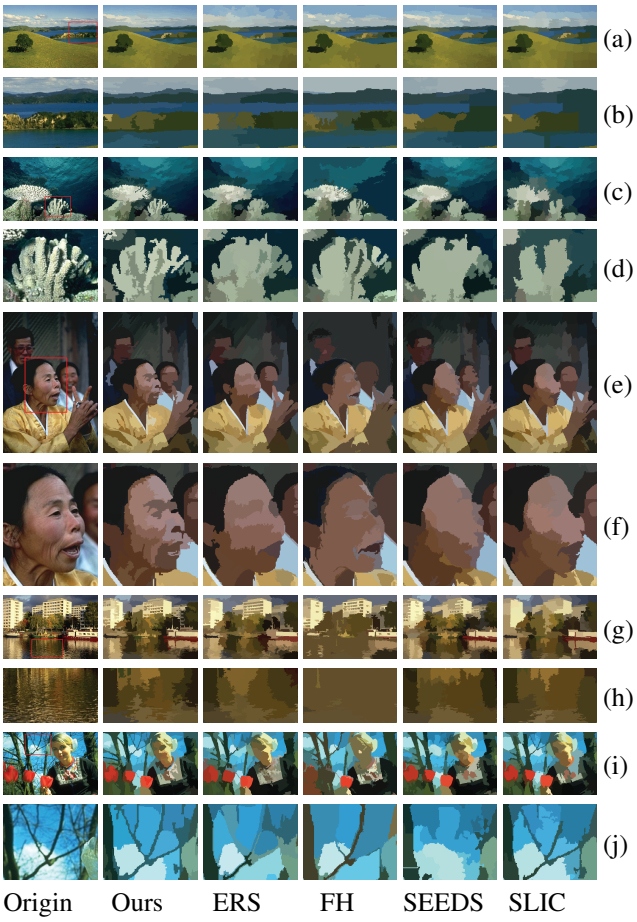


Fig. 7. Visual comparison with four state-of-the-art algorithms by the averaged color of 100 superpixels

7. CONCLUSION

We present an efficient structure-preserving superpixel algorithm based on MST. The presented algorithm takes structure-sensitive advantage of the MST and achieves a high performance both on structure-preserving and efficiency. Also, our algorithm proposes a multi-layer scheme for gradually merging the image pixels and dampening the texture noises. The nodes number is exponentially reduced in each layer so that our algorithm stay in linear complexity. The experimental results show that the proposed algorithm outperforms current state-of-the-art algorithms both on accuracy (boundary recall) and time efficiency. However, our algorithm is not robust enough for video superpixels, which will be considered in our future works.

8. THANKS

We would like to thank the anonymous reviewers. This work was supported by the National Natural Science Foundation of China (NSFC) under Nos. 61472377 and 61331017, and the Fundamental Research Funds for the Central Universities under No.WK2100060011.

9. REFERENCES

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on PAMI*, 34(11):2274–2282, 2012.

[2] L. Bao, Y. Song, Q. Yang, H. Yuan, and G. Wang. Tree filtering: Efficient image smoothing with a minimum spanning tree. *IEEE TIP*, 23(2):555–569, 2014.

[3] R. Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[4] M. Chen, Q. Yang, Q. Li, G. Wang, and M.-H. Yang. Spatiotemporal background subtraction using minimum spanning tree and optical flow. In *Computer Vision–ECCV*, pages 521–534. Springer, 2014.

[5] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on PAMI*, 24(5):603–619, 2002.

[6] T. Matsui The minimum spanning tree problem on a planar graph. *Discrete Applied Mathematics*, 58(1):91-94, 2002.

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[8] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on PAMI*, 31(12):2290–2297, 2009.

[9] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *CVPR*, pages 2097–2104. IEEE, 2011.

[10] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, volume 2, pages 416–423. IEEE, 2001.

[11] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on PAMI*, 26(5):530–549, 2004.

[12] A. P. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[13] A. P. Moore, S. J. Prince, and J. Warrell. lattice cut-constructing superpixels using layer constraints. In *CVPR*, pages 2117–2124. IEEE, 2010.

[14] P. Neubert and P. Protzel. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, 2012.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, 2000.

[16] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *Computer Vision–ECCV*, pages 13–26. Springer, 2012.

[17] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Computer Vision–ECCV*, pages 705–718. Springer, 2008.

[18] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *ECCV*, pages 211–224. Springer, 2010.

[19] P. Wang, G. Zeng, R. Gan, J. Wang, and H. Zha. Structure-sensitive superpixels via geodesic distance. *IJCV*, 103(1):1–21, 2013.

[20] Q. Yang. A non-local cost aggregation method for stereo matching. In *CVPR*, pages 1402–1409. IEEE, 2012.

[21] Y. Zhang, R. Hartley, J. Mashford, and S. Burn. Superpixels via pseudo-boolean optimization. In *ICCV*, pages 1387–1394. IEEE, 2011.