Edge-Based Split-and-Merge Superpixel Segmentation

Li Li, Jian Yao[†], Jinge Tu, Xiaohu Lu, Kai Li, and Yahui Liu
School of Remote Sensing and Information Engineering
Wuhan University, Wuhan, Hubei, P.R. China
[†]Email: `jian.yao@whu.edu.cn` Web: `http://cvrs.whu.edu.cn/`

***Abstract*— Superpixels are an oversegmentation of an image and popularly used as a preprocessing in many computer vision applications. Many state-of-the-art superpixel segmentation algorithms rely either on minimizing special energy functions or on clustering pixels in the effective distance space. While in this paper, we introduce a novel algorithm to produce superpixels based on the edge map by utilizing a split-and-merge strategy. Firstly, we obtain the initial superpixels with uniform size and shape. Secondly, in the splitting stage, we find all possible splitting contours for each superpixel by overlapping the boundaries of this superpixel with the edge map, and then choose the best one to split it which ensure the superpixels produced by this splitting are dissimilarity in color and similarity in size. Thirdly, in the merging stage, the Bhattacharyya distance between two color histograms in the RGB space for each pair of adjacent superpixels is computed to evaluate their color similarity for merging superpixels. At last, we iterate the split-and-merge steps until no superpixels have changed. Experimental results on the Berkeley Segmentation Dataset (BSD) show that the proposed algorithm can achieve a good performance compared with the state-of-the-art superpixel segmentation algorithms.**

***Index Terms*— Superpixels, Edge Drawing, Split-and-Merge, Image Segmentation**

I. Introduction

Superpixel segmentation is an important and effective preprocessing step in many computer vision fields such as stereo matching [1], image segmentation [2], [3], high dynamic range image reconstruction [4] and image blending [5]. The major advantage of using superpixels is to improve the efficiency of the algorithms at the prerequisite of ensuring the high quality of results. Superpixel segmentation aims to generate many small and homogeneous patches belonging to the same physical world object. In general, pixels in one superpixel should be similar in color, texture, and so on. And, the superpixels should be compact and regular. Thus we can replace traditional pixels with superpixels in many image processing algorithms to reduce computational complexity and memory cost.

The concept and the first algorithm of superpixel was first proposed by Ren and Malik [6] based on the Normalized-Cuts (N-Cuts) [7]. In the N-Cuts algorithm, superpixel segmentation has been regarded as an energy optimization problem which can be minimized via graph cut. Despite of high accuracy and good regularity in both size and shape, the high computational cost, however, limits its application. Some algorithms, like Mean Shift [8], Graph-based Segmentation [9] and watershed [10] algorithms, are fast and high accuracy. However, due to the lack of compactness constraint, superpixels produced by these algorithms are usually arbitrary in both size and shape, hence no longer as comparable as 'pixel', as shown in Fig. 1(a),(b).

Recently, many efficient algorithms to generate superpixels have been proposed. The aim of all of those algorithms is to produce superpixels with high accuracy, low computation and good regularity in size and shape, as shown in Fig. 1(c),(d). Generally, the superpixel segmentation problem can be solved in two ways, namely, energy optimization and pixels clustering.

Some algorithms regarded this as an energy optimization problem, such as TurboPixels [11], SEEDS [12] and Graph-Cuts based algorithms [13], [14]. The SEEDS algorithm started from an intial superpixel partitioning and then continuously refined the superpixels by modifying the boundaries. The special objective function based on enforcing homogeneity of the color distribution of the superpixels, plus a term that encourages smooth boundary shapes are designed to guide the movement of boundaries, and the optimization is based on a hill-climbing algorithm. Veksler et al. [13] formulated the superpixel partitioning problem in the graph cuts energy minimization framework, and solved it via Expansion-Moves [15]. And the regularity was considered in the energy function. It assumed that the input image are covered by half overlapping square patches of the same size and each patch corresponds to a label. Then, it assigned each pixel to a unique label (patch) through minimizing the energy functions. After that, to improve the segmentation efficiency, Zhang et al. [14] proposed to model the superpixel problem with only two labels instead of 'very-many-label' in [13]. The key issue of these algorithms is to choose a suitable energy optimization methods to optimize a predefined objective function usually defined in the form of a graph.

In addition, some algorithms regarded it as a pixels clustering problem, such as SLIC [16] and VCells [17]. Those

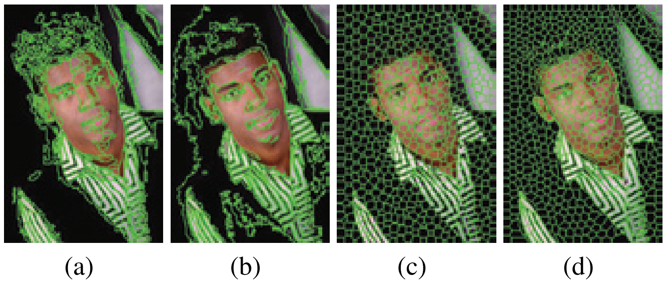


Fig. 1. Oversegmentation obtained with four algorithms:(a)Mean shift [8], (b) graph based [9], (c) superpixels via graph cuts [13], (d) VCells [17].

algorithms usually have two stages. In the first stage, input image should be initialized to many uniform segmentations with the same size and shape, at the same time, the initial clustering centers are obtained. In the second stage, algorithms would define a suitable distance measurement to refine the initial segmentation results iteratively, which is just similar to K-means clustering. SLIC clustered pixels based on color similarity in the CIELAB color space and the spatial distance in the image plane. Thus, the 5-D space is defined by the L, a and b values of the CIELAB color space and the x and y pixel coordinates. And a novel distance measurement which enforce color similarity as well as pixel proximity in this 5D space are introduced to generate compact and regular superpixels. VCells algorithm firstly divided the input image into small segments of uniform size and shape, and then iterated clustered pixels by evalute the edge-wight distance between the pixels and the clustering centers which update in each iteration.

In this paper, we proposed a novel edge-based superpixel segmentation algorithm by utilizing a split-and-merge strategy, which is different from the above-mentioned two categories of superpixel segmentation algorithms. The split-and-merge strategy is a traditional method popularly used in image segmentation and has been proved effective. Instead of designing an objective function or defining a new distance space, we start with initial patches with similar shapes and sizes just like the first step of VCells, and then iteratively refine it by splitting and merging it with the help of the edge map produced by the Edge Drawing (ED) algorithm [18].

II. Algorithm

Our proposed superpixel segmentation algorithm is comprised of two main stages. In the first stage, we segment an image into many small patches with regular shape and size (see Section II-A). In the second stage, we use the split-and-merge strategy to refine these patches iteratively based on the edge map, generated by the Edge Drawing (ED) [18] method, to obtain the final superpixels (see Section II-B).

Before giving a detailed description of our algorithm, we briefly review the ED edge detection approach. The aim of ED is to find a high-quality edge map which is widely used in many computer vision and pattern recognition applications. ED first suppresses noise of the grayscale image by Gaussian filtering. Then the anchor points with local gradient extreme are extracted based on the computed gradient magnitude and edge direction map. After that, the edge map is obtained through the anchors connection guided by the gradient magnitude and edge direction map. ED can produce a better edge map in most cases than the popularly used Canny edge detection algorithm as well as with lower computational costs.

A. Uniform Segmentation

The first stage of our algorithm can be regarded as an initialization process. The results of this stage are used as the initial superpixels in the next split-and-merge stage.

Generally speaking, we can simply initialize the superpixels in a grid, namely, dividing the image into uniform squares or rectangles. Many algorithms have applied this strategy, such as SEEDS [12]. However, in our paper, we applied the CfCVDT [19] algorithm which have been proved efficient in VCells algorithm to generate initial small segments with homogeneous shape and size, as shown in Fig. 4(b). The major advantage of CfCVDT is that its runtime is independent of the image size. The initial uniform segmentation time is fixed no matter what the size of the input image is.

B. Split-and-Merge

The split-and-merge strategy is adopted in our algorithm to produce superpixels, which has been widely used in image segmentation. The key issue of this strategy is to define an appropriate standard or construct a function to evaluate the cost during splitting and merging. Unlike the conventional split-and-merge strategy, the constraint of compactness should be also considered to ensure that the segmentation results have the regular shape and size just like the real 'pixel'.

1) Splitting Superpixels: In the first stage, we have obtained the initial patches with the regular size and shape. The edge map is produced efficiently and accurately by the ED algorithm, as shown in Fig.2(a). The initial patches and edge map can be regarded as the basis of our algorithm. In order to split superpixels, the problem we need to solve is how to obtain the best splitting contour for each surperpixel with the help of the edge map. This problem can be solved through two steps. The first step is to find all possible splitting contours from the edge map for each superpixel by overlapping its boundaries with the edge map. Then we select the best one to split it by evaluating the costs of all splitting contours, which ensure the superpixels produced by this splitting are dissimilarity in color and similarity in size.

In the first step, we first overlap the initial patches with edge map as shown in Fig. 2(b). All edge contours cross each superpixel can be easily found at one time. In this way, all possible splitting contours, as shown in Fig.2(c), for each superpixel can be further evaluated for selection.

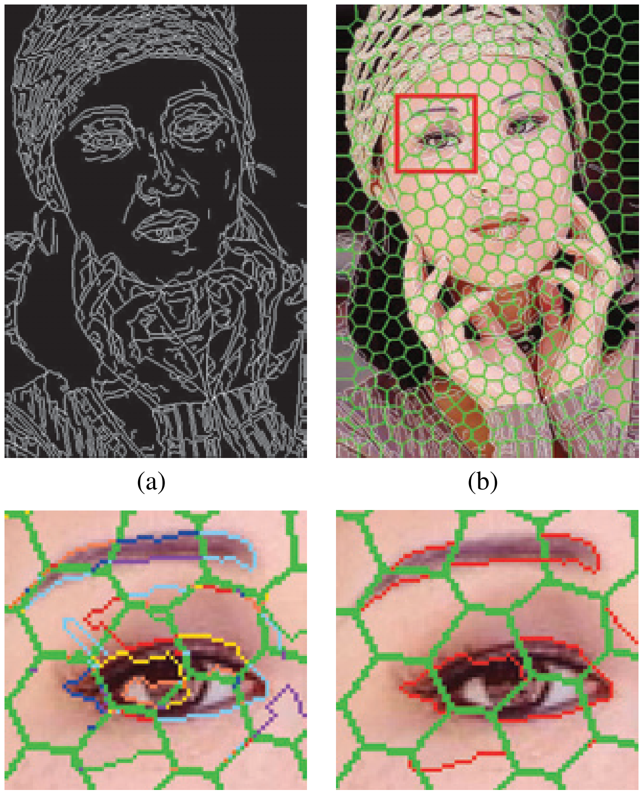


Fig. 2. (a) The edge map produced by ED algorithm; (b) The map visualization overlapped by the image, initial uniform segmentation (Green) and the edge map (White); (c) All possible splitting contours with different colors in each superpixel; (d) The best splitting contour in Red color for each superpixel in one iteration.

In the next step, the task is to find the best splitting contour for each superpixel, as shown in Fig.2(d). Two basic properties must be maintained by the best splitting contour. One is that the difference between two regions along this splitting contour should be the biggest and another is that this contour should split the superpixel as possible as equally. Thus we define an energy function to choose the best splitting contour based on the color histogram distance and the ratio between areas of two superpixels split by the contour. Let P be the superpixel to be split, and P_1 and P_2 be the split superpixels. The pixels belonging to P_1 or P_2 can be easily collected given a splitting contour. Let $\mathcal{S} = \{S_i\}_{i=1}^N$ be the N possible splitting contours for the superpixel P to be split. The following function is used to select the best splitting contour from $\mathcal{S}$ for the superpixel P:

$$C(S_i) = \lambda \frac{\min(|P_1|, |P_2|)}{\max(|P_1|, |P_2|)} + (1-\lambda) D_h(H_1, H_2), \quad (1)$$

where $|P_1|$ and $|P_2|$ denote the areas (i.e., the numbers of pixels) of P_1 and P_2 split by the contour S_i, H_1 and H_2 denote the color histogram in RGB color space of P_1 and P_2, and $D_h(H_1, H_2)$ denotes the color histogram Bhattacharyya

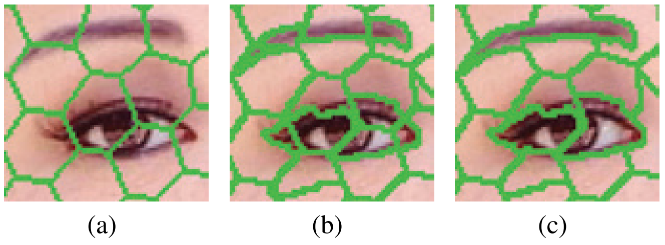


Fig. 3. A visual example of one iteration of split-and-merge step. (a) Initial segmentation results; (b) The results of splitting superpixels; (c) The results of merging superpixels.

distance calculated as follows:

$$D_h(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum_{k=1}^{N} \sqrt{H_1(k) H_2(k)}}, \quad (2)$$

where $\bar{H}_1$ and $\bar{H}_2$ denote the mean value of histograms H_1 and H_2 with N bins and $H(k)$ denotes the number of elements at the k-bin in the histogram H. The weight factor $\lambda \in [0, 1]$ is to balance the color histogram distance and the area ratio and it was fixed at 0.5 in this paper. The splitting contour with the biggest cost in Eq. (1) and $D_h(H_1, H_2) > \delta_p$ is selected as the best one, where δ_p is a pre-defined threshold to avoid splitting the superpixel with even and uniform texture.

2) Merging Superpixels: After splitting the superpixels, the same object in the real world has possibly been divided into many small superpixels such as eyes and eyebrows as shown in Fig. 3(b). As a result, there are many superpixels that are too small or too similar to adjacent superpixels. Obviously we need to merge them. Through merging superpixels, we can obtain the compact superpixels with regular size and shape. A typical example for the merging result is shown in Fig. 3(c). We should note that we only show the merging result of the first iteration in Fig. 3, and the final superpixels are shown in Fig. 4. Thus, it is normal that there are many differences between two figures in the loacl regions of eyes and eyebrows.

To merge adjacent superpixels, we need to evaluate their similarity. We choose to use the same color similarity measurement for merging as for splitting. Of course, anther region similarities (i.e.,texture similarity) can be integrated into our algorithm too, but we find that the simple RGB color histogram similarity can produce satisfying results. It also illustrate the effectiveness of split-and-merge strategy proposed by us to generate superpixels. Given a pair of adjacent superpixels (P_i, P_j), we compute the Bhattacharyya distance $D_h(H_i, H_j)$ where H_i and H_j denote the histogram in RGB color space for the superpixels P_i and P_j, respectively. The pair of adjacent superpixels (P_i, P_j) will be merged only when the following two conditions are satisfied. The first condition is that the area (i.e., the number of pixels) of one

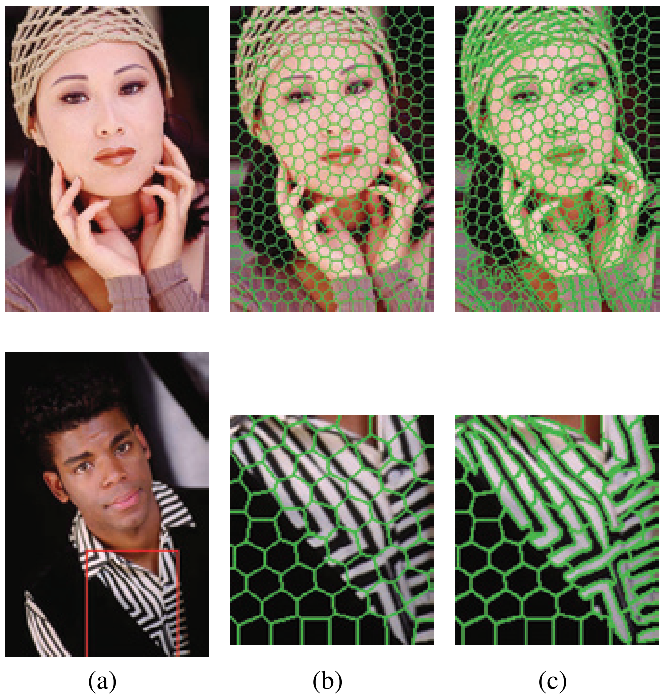


Fig. 4. (a) Initial color image; (b) Uniform segmentation; (c) Superpixels produced by our algorithm.

superpixel is small enough. The second one is that their color similarity $D_h(H_i, H_j) < \delta_m$ where $\delta_m \in (0, 1)$ is a predefined threshold.

Based on the former descriptions, our proposed superpixel segmentation algorithm can be summarized as follows:

- Step 1: Produce the edge map via ED.
- Step 2: Initialize the image into uniform segmentations.
- Step 3: Split superpixels: 1) Collecting all possible splitting contour for each superpixel from the edge map; 2) Choosing the best one to split it if possible by evaluating the splitting costs.
- Step 4: Merge superpixels: Merging the adjacent superpixels with similar color and small area.
- Step 5: If there is no split or merge happened for each superpixel in one iteration, return the current segments; otherwise, go to step 3 and repeat the split-and-merge iteration.

III. Experimental results

We condudcted our evaluation experiments on the Berkeley Segmentation Dataset (BSD) [20], which contains 200 training images and 100 test images with human-labeled ground truth segmentations. In this paper, we compared our algorithm with VCells algorithm which had be compared with many superpixel algorithms. We adopt the standard boundary recall measurement which computes the fraction of ground truth boundaries which fall within a small disk shaped neighborhood of the superpixels' boundaries in which

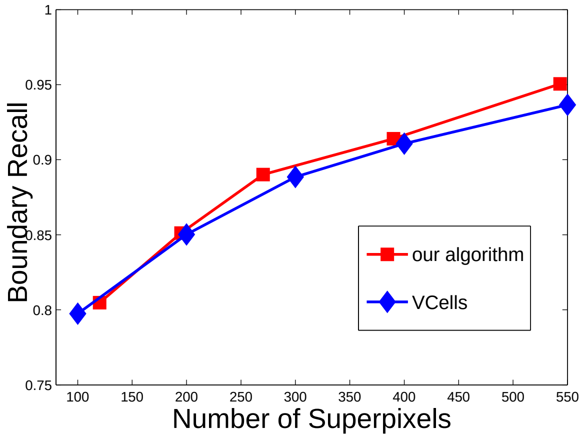


Fig. 6. Performance of VCells and our algorithm.

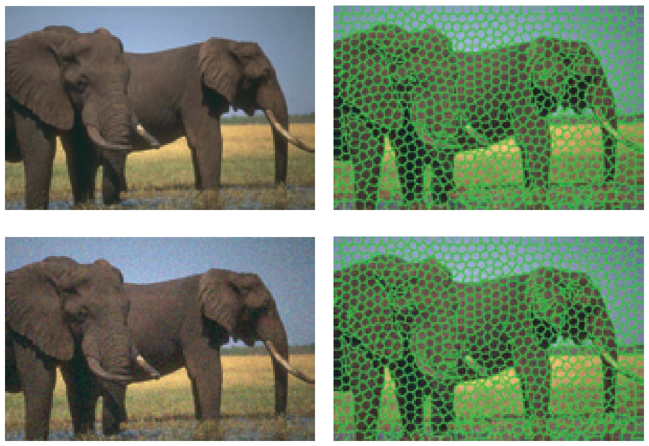
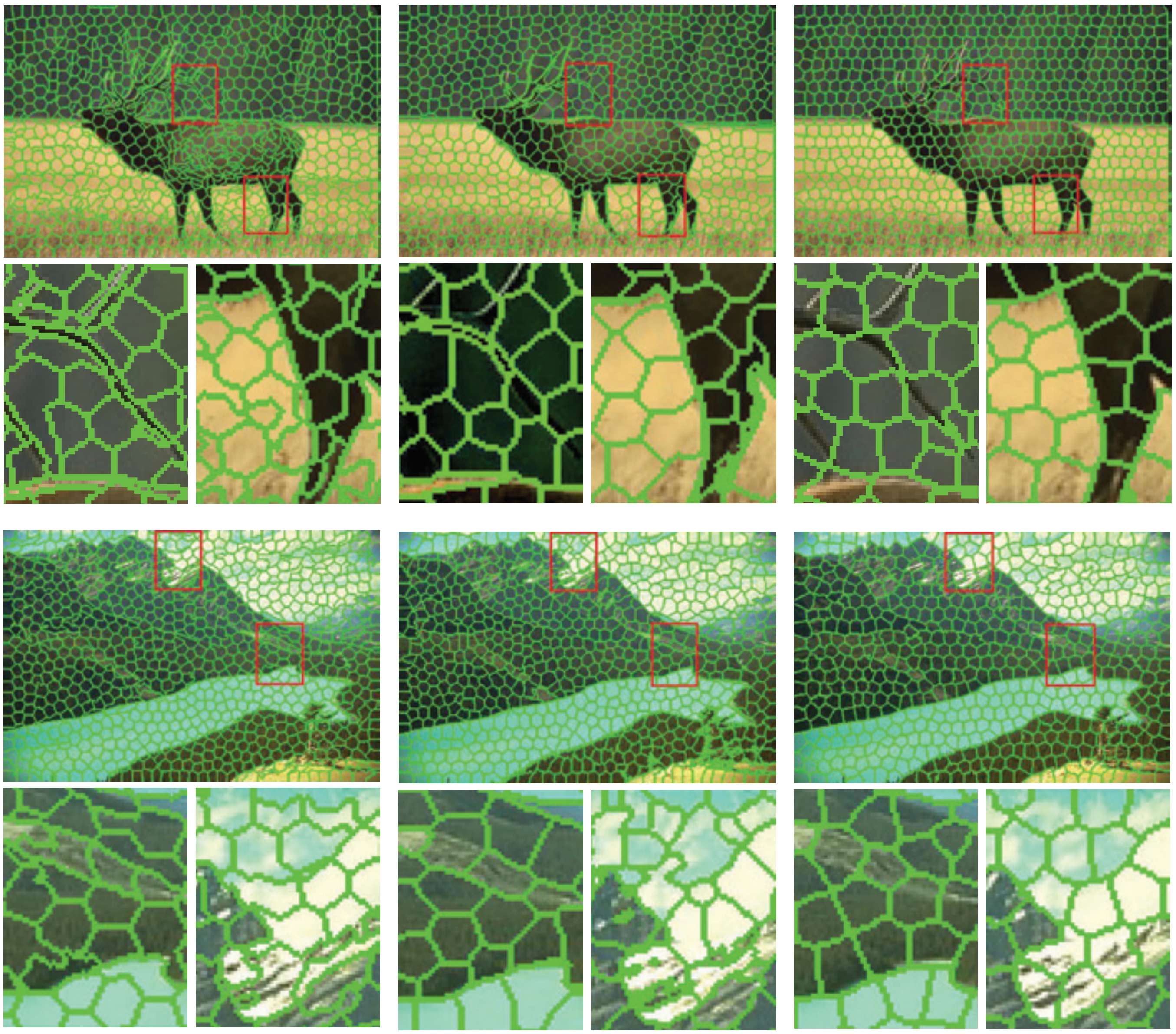


Fig. 7. Demonstration of the robustness of our algorithm on the image (in the second row) additionally noised by the original image (in the first row).

the radius is set to 2 pixels to evaluate the performance of superpixel segmentation algorithm. The training images were utilized to set a few parameters of our algorithm, which were fixed in all the experimental results. The default parameters for VCells were used for comparison. The boundary recalls of VCells and our algorithm were calculated averagely on the 100 test images. Superpixels in 5 different scales, ranging from 100 to 550 superpixels, were generated for comparison. The performance of two algorithms is shown in Fig. 6 from which we observed that our algorithm outperforms VCells in all the 5 different scales. And the comparison of VCells with another superpixel algorithms is shown in [17].

There exists two key parameters, δ_m and δ_p, in our algorithm. δ_m is the threshold for merging two adjacent superpixels. When the color histogram distance of two adjacent superpixels is smaller than δ_m, we regard them as one object and merge them, otherwise, we ignore it. In the splitting stage, the bigger the threshold δ_p for splitting the superpixels, the less superpixels will be split. In all experimental results shown in Fig. 6, we set $\delta_m = 0.5$ and $\delta_p = 0$.



(a) Our algorithm (b) VCells (c) SLIC

Fig. 5. Visual comparison of our algorithm, VCells and SLIC.

The superpixel segmentation results with the same number of superpixels of two images generated by our algorithm, VCells and SLIC are shown in Fig. 5. From the segmentation results in these two images and especially the detailed local regions, we observed that our algorithm obviously outperforms VCells and SLIC in some local regions where the edges are not obvious or not strong. For example, in the region of antelope horn which the color and texture are similar with the background, our algorithm successfully generated the superpixels which the boundaries are similarity with the ground truth, but VCells and SLIC algorithm failed in this region. The more regions similar with this example can be found in Fig. 5 easily.

Finally we tested the segmentation results of our algorithm on the image with noise, as shown in Fig. 7 where the left image of the second row were obtained by adding Gaussian noise to the original image (i.e., the left image of the one row). The two pictures in the right column are the superpixels produced by our algorithm for the corresponding left image. We found that the segmentation results on the original image and noised one are very similar, which proves that our algorithm is robust enough with respective to noise.

Some more examples of superpixel produced by our algorithm can be found in Fig. 8.

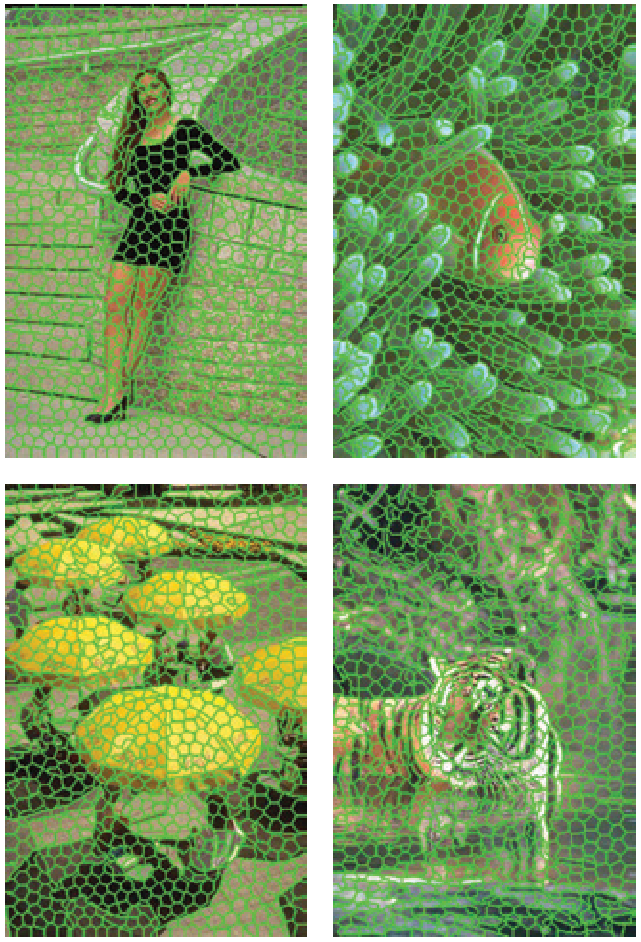


Fig. 8. More superpixels produced by our algorithm.

IV. Conclusion

In this paper, instead of regarding the superpixel as energy optimization or pixels clustering problem, we proposed an effective edge-based superpixel segmentation algorithm in a split-and-merge framework which has been widely and popularly used in many image segmentation applications. In the splitting stage, all possible splitting contours for each superpixel can be first collected from the edge map produced by the ED algorithm, then we choose the best one to split it. In the merging stage, each pair of adjacent superpixels will be merged when the Bhattacharyya distance between two color histograms in RGB space of these two superpixels is small enough. The compactness and regularity of superpixels are enforced in the step of split-and-merge. Experimental results sufficiently illustrated that our proposed superpixel segmentation algorithm achieved good quality segmentations.

References

[1] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N. Do, "Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[2] Pushmeet Kohli, Lubor Ladicky, and Philip H.S. Torr, "Robust higher order potentials for enforcing label consistency," *International Journal of Computer Vision*, vol. 82, pp. 302–324, 2009.

[3] Jifeng Ning, Lei Zhang, David Zhang, and Chengke Wu, "Interactive image segmentation by maximal similarity based region merging," *Pattern Recognition*, vol. 43, pp. 445–456, 2010.

[4] Shanmuganathan Raman and Subhasis Chaudhuri, "Reconstruction of high contrast images for dynamic scenes," *The Visual Computer*, vol. 27, pp. 1099–1114, 2011.

[5] Nuno Gracias, Mohammad Mahoor, Shahriar Negahdaripour, and Arthur Gleason, "Fast image blending using watersheds and graph cuts," *Image and Vision Computing*, vol. 27, pp. 597–607, 2009.

[6] Xiaofeng Ren and Jitendra Malik, "Learning a classification model for segmentation," in *IEEE International Conference on Computer Vision (CVPR)*, 2003.

[7] Jianbo Shi and Jitendra Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, 2000.

[8] Dorin Comaniciu and Peter Meer, "Mean Shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[9] Pedro F Felzenszwalb and Daniel P Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, pp. 167–181, 2004.

[10] Luc Vincent and Pierre Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 13, pp. 583–598, 1991.

[11] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 2290–2297, 2009.

[12] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool, "SEEDS: superpixels extracted via energy-driven sampling," in *European Conference on Computer Vision*. Springer, 2012.

[13] Olga Veksler, Yuri Boykov, and Paria Mehrani, "Superpixels and supervoxels in an energy optimization framework," in *European Conference on Computer Vision (ECCV)*, pp. 211–224. Springer, 2010.

[14] Yuhang Zhang, Richard Hartley, John Mashford, and Stewart Burn, "Superpixels via pseudo-boolean optimization," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[15] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1222–1239, 2001.

[16] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, "SLIC superpixels," *École Polytechnique Fédéral de Lausssanne (EPFL), Tech. Rep*, vol. 149300, 2010.

[17] Jie Wang and Xiaoqiang Wang, "VCells: Simple and efficient superpixels using edge-weighted centroidal voronoi tessellations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1241–1247, 2012.

[18] Cihan Topal and Cuneyt Akinlar, "Edge Drawing: A combined real-time edge and segment detector," *Journal of Visual Communication and Image Representation*, vol. 23, pp. 862–872, 2012.

[19] Lili Ju, "Conforming centroidal voronoi delaunay triangulation for quality mesh generation," *International Journal of Numerical Analysis and Modeling*, vol. 4, pp. 531–547, 2007.

[20] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *IEEE International Conference on Computer Vision (ICCV)*, 2001.