Cluster Sensing Superpixel and Grouping

Rui Li* Lu Fang†

Abstract

*Superpixel algorithms have shown significant potential in computer vision applications since they can be used to accelerate other computationally demanding algorithms. However, in contrast to the original purpose of superpixels, many upper layer methods still suffer from computational problems when incorporating superpixel for speedup. In this paper, we present a cluster sensing superpixel (CSS) method to efficiently generate superpixel bricks. Based on the insight of pixel density, cluster centers generally have properties of representativeness (*i.e.*, local maximal pixel density) and isolation (*i.e.*, large distance from other cluster centers). Our CSS method efficiently identifies ideal cluster centers via utilizing pixel density. We also integrate superpixel cues into a bipartite graph segmentation framework and apply it to microscopy image segmentation. Extensive experiments show that our CSS method achieves impressive efficiency, being approximately five times faster than the state-of-the-art methods and having comparable performance in terms of the standard metrics. Application on microscopy image segmentation also benefits our efficient implementation.*

1. Introduction

Superpixel representation is becoming increasingly popular in various computer vision applications, *e.g.*, segmentation [2, 13], stereo matching [21, 22], saliency detection [4, 20, 26], tracking [23, 25] *etc*. There are two primary reasons for the performance gains of superpixel representation. The first is superpixels that reduce the computational primitives significantly without obvious information loss, and the second is that regular shape bricks preserve the main connection relationships of images. Many computationally expensive tasks benefit from these attractive properties, and improve the efficiency of algorithm[11] or achieve performance improvements[5].

To further improve superpixel performances, many novel methods have been proposed, for example, graph-based methods [7, 12], spectral clustering [17], mean shift[6], k-means clustering [1, 10] and energy maximization [19]. Although these previous arts achieve satisfactory results, most of them have difficulty striking a good balance between real-time level efficiency and regular shape. Since reducing the computational burden is essential to superpixel, the efficiency is extremely critical for superpixels, and irregular shape for segmentation will lead to vagueness for the original spatial relationship.

Inspired by the idea in[16], the idea of cluster centers can be described as high similarity to local others (*i.e.*, representativeness) and as having a relatively large distance from higher density points (*i.e.*, isolation). We denote pixel density as the level of pixel aggregation or the similarity of the center to its neighboring pixels. In contrast to the pixel values, pixel density can reflect the pixel concentration in the local region and can be utilized as the metric to identify image components. For example, the pixels near image boundaries generally have the comparatively low density value since boundaries separate two semantic regions with very different color distribution, leading to the a low-density trajectory in the density channel; pixels in the smooth regions are similar to the local others with comparatively high and smooth density values. Therefore, density channel can be utilized as the a metric to identify the cluster centers, boundaries, smooth regions, noise pixels *etc*. Since nature image is piecewise smooth, the conventional RGB or Lab color space cannot distinguish the spatial location of pixel (*e.g.*, close to the boundary pixels, central region of smooth patch) or the relationship to local others (*e.g.*, similarity). However, these information is crucial to superpixels for selecting the seeds or guiding the searching clusters. Fortunately, the pixel density provides a new way to discover the spatial-dependant information.

In this paper, our main technical contributions mainly are the following: 1) We reveal the insight of the pixel density and show that it can be utilized as a powerful metric to discover the low-level components. 2) We propose a cluster sensing superpixel (CSS) method that efficiently searches local optimal cluster centers by pixel density and aggregates pixels via kernelized distance metrics. 3) We also integrate the superpixel cues into the bipartite graph framework and apply it to the microscopy image segmentation. Moreover,

[1]alr@mail.ustc.edu.cn, University of Science and Technology of China, Hefei, Anhui, China
[2]eefang@ust.hk, correspondence author, Hong Kong University of Science and Technology, Hong Kong, China

our CSS not only achieves regular shaped superpixels and comparable performance with previous arts, but also can run at approximated 60 fps (five times faster than the previous methods) for modest-size images.

2. Related Works

In this section, we review the state-of-the-art superpixel methods, more detailed surveys can refer to [1, 2].

Graph-based methods address superpixel or segmentation problems as minimizing cut or object function on graph [8]. Early work [17] proposed the normalized cut to find the global optimum for segmentation via spectral clustering. Since normalize cut involves the computationally expensive eigen decomposition, many accelerating methods have been proposed for improving this novel but computationally demanding algorithm, *e.g.*, [3, 18, 13]. [7] proposed an efficient graph cut method (known as FH) to solve over segmentation problem. Despite its efficiency and impressive boundary adherence, FH lacks spatial constraint and tends to overlap with multiple objects. For more visually pleasing superpixel algorithms, ERS[12] integrates an entropy rate term and a balanced term for encouraging smoothness and boundary adherence, but also this leads to a high computational time ($\approx 2s$).

Cluster-based methods treat superpixel problem as finding clusters in the feature space. Due to the introduction of spatial constraints for clustering process, cluster-based methods generally produce regular and visual pleasing segmentation, *e.g.*, SLIC[1] and LSC[10], which are widely-used for supporting upper layer algorithms. SLIC is one of the most popular superpixel methods due to its simplicity and effectiveness; however, SLIC is not efficient enough to accelerate real-time applications. LSC[10] maps traditional pixel values to a specific feature space via kernel function and yields a globally optimal solution, but the kernel function also increases the computational burden. Bergh *et al.* [19] proposed the very efficient SEEDS that adopts a hill-climbing method to optimize the proposed energy function as well as generate homogeneous segmentations. However, SEEDS generally produce superpixels with irregular shapes, which have difficulty becoming semantic bricks to represent images compactly. For overcoming the drawback of SEEDS, [24] added a spatial regular term for generating more regular superpixels. Despite the board family of superpixel methods that solve problem from different angles, *e.g.*, NCut[17], temporal consistency [15], lattices[14] and geometric information [9], most of them still do not well handle the computational problem.

In contrast to most previous arts, our CSS method addresses superpixel problem as searching cluster centers in image space by exploring density peaks. Compared to the most widely-used superpixel methods SLIC[1] and SEEDS[19], our CSS method achieves regular and homogeneous superpixels, with an impressive efficiency of 60 fps (20 times faster than SLIC and five times faster than SEEDS), and can be used as an efficient replacement for the state-of-the-art superpixel methods.

3. Cluster Sensing Superpixel Method

Clustering-based superpixel methods address superpixel problems as searching representative pixels as centers[1, 10]. Rodriguez and Laio [16] introduced the novel idea that cluster centers are highly similar to local others (representativeness) and have relatively large distance from other higher density points (isolation). Thus, representativeness and isolation can be utilized as robust metrics to identify cluster centers and suppress outliers. Based on the idea above, we first introduce our density channel, and then detail our CSS.

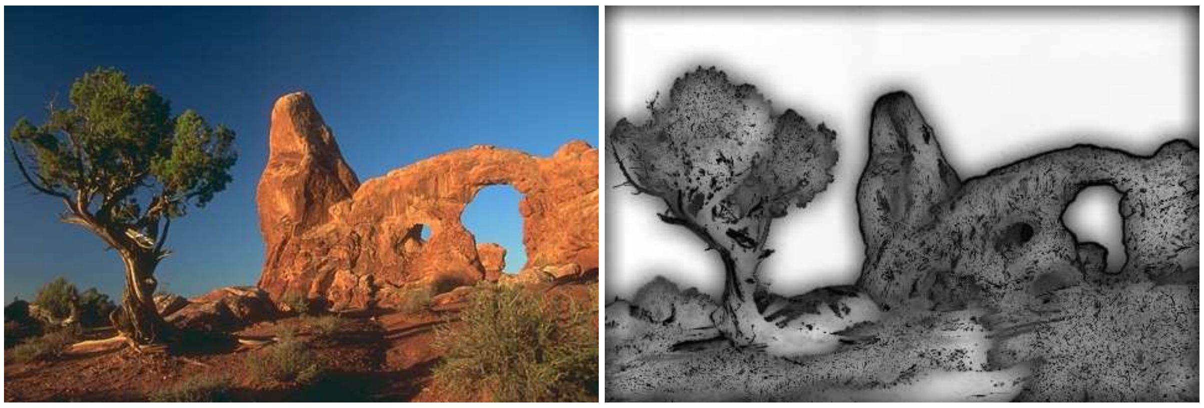
3.1. Density Channel

We define pixel density as the number of pixels that are close to a given pixel in the feature space around the local region. As illustrated Fig. 1, pixel density can reflect many important image structures (*e.g.* edge, smooth regions, texture *etc.*) and removes irrelevant information, *e.g.*, noise or color. Many perceptual grouping methods, *e.g.*, superpixel, edge detection, involve in dividing an image into semantic parts such as edges or object components. However, traditional simple pixel-level features can not effectively distinguish these components since nature images are complicated and generally have many patterns for different types of regions. Fortunately, the density channel is a powerful means to identify different patterns, even when those patterns are diverse on a simple color space.

Fig. 1(b) shows that smooth regions generally have comparatively high density values with smooth distribution. As the illustration in Fig. 1(d) shows density channels have comparatively low-density values on both edges and image noise pixels since edges generally have distinct color distribution on two different sides of boundary and noise pixels are isolated points with a large color distance from neighboring region. For different texture regions, the color distributions generally are various. But, as the illustration in Fig. 1(c) shows the texture in the density channel shows similar characteristics to rough distribution even when the color distribution is significantly different. Thus, good cluster centers for superpixels should satisfy several requirements: 1) highly represent the local region, 2) isolate to other cluster centers and 3) avoid locating in boundaries or noise pixels. We adopt the density channel for searching those possible candidates that satisfy the above three requirements.

We formulate the pixel density mathematically as:

$$\rho(p) = \sum_{q \in \mathcal{N}_p} \delta(p, q), \tag{1}$$


(a)

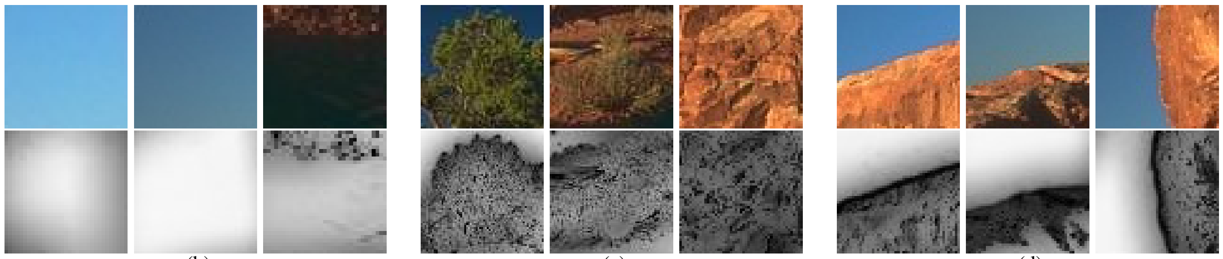

(b) (c) (d)

Figure 1. The examples of density channel and the region patterns. (a) shows the example image (left) and corresponding density channel (right). (b), (c), (d) illustrate the examples of smooth region, texture and edge patterns with corresponding density channel respectively.

where $\rho(p)$ is the density value of p, $\mathcal{N}_p$ is the neighboring pixel set around p and $\delta(\cdot,\cdot)$ is the similarity function.

We measure the similarity of two pixels by mapping the Euclidean distance to the kernel space (Gaussian). Our pixel distance consists of a spatial weighted term and a color weighted term. The color weighted term considers the color similarity of two pixels and the spatial weighted term enhances the influence of nearby pixels and suppress the weights of farther pixels. The gaussian kernel maps two terms to $[0, 1]$, which is consistent with the practical meaning of pixel density. Thus, the spatial weighted term d_s and the color weighted term d_c are:

$$d_s(p,q) = \mathrm{e}^{-\frac{D_s(p,q)^2}{2\delta_s^2}}, \quad (2)$$

$$d_c(p,q) = \mathrm{e}^{-\frac{D_c(p,q)^2}{2\delta_c^2}}, \quad (3)$$

where D_s and D_c denote the Euclidean distance on the spatial and color feature space respectively. δ_s and δ_c are the variance of Gaussian kernel and control the influence of the spatial and color weight. Thus, Eqn. 1 can be formulated as:

$$\rho_p = \sum_{q \in \mathcal{N}_p} d_s(p,q) d_c(p,q). \quad (4)$$

The computational complexity of the density channel is $\mathcal{O}(|\mathcal{N}_p|N)$, where $|\mathcal{N}_p|$ is the number of pixels in $\mathcal{N}_p$ and N is the total pixel in the image. For computational efficiency, we set $\mathcal{N}_p$ to be the small size (*e.g.*, $(7, 7)$), but a large-size $\mathcal{N}_p$ does not significantly affect the density value due to the large spatial distance pixels having less influence for density computing.

3.2. Cluster Center Searching and Aggregation

We compute the density value for each pixel by Eqn. 4 and generate the density channel. We randomly sample some pixels as initial candidates, and then iteratively move the candidates to the highest density value position on search region (*e.g.*, 3×3). The iterations repeat until all the cluster centers converge to local maximums. As illustrated in Fig. 2, cluster center candidates start moving from the initial red points and iteratively find and move to the maximal positions in the local search region, which are blue points, at the neighboring region on the density channel. Green points are the local maximums on the density channel, it is possible to move to the same local maximum point to form a cluster, such as point 1 and point 2 in Fig. 2. Point 3 has different trajectory that forms another new cluster.

THe above searching scheme finds the cluster centers based on the density channel, and also satisfies the requirements of the cluster centers. First, the candidate pixels will move to the local optimum in density map, which has

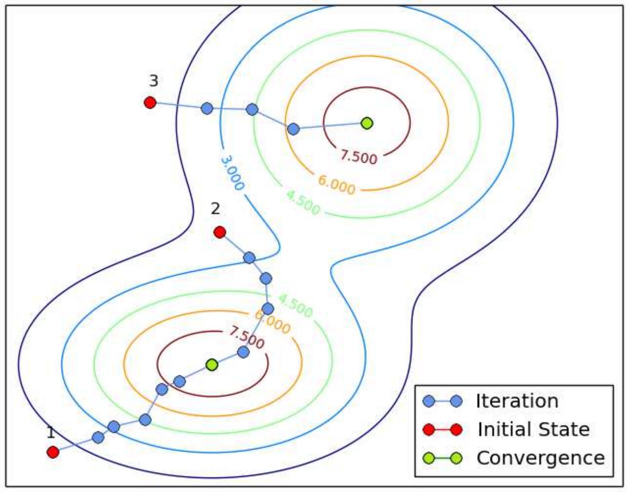


Figure 2. The illustration of searching scheme on density channel. 1,2,3 are the cluster center candidates that start from initial point (red) and iteratively update current position (Blue) until reach the local optimal position (Green).

the highest density (most representative) in the local regions. Second, density peaks in the nature image generally are isolated pixels, so our cluster centers will have a comparatively large distance from each others. Third, the boundary and noise pixels have low density, and our density search scheme can avoid falling into outlier pixels. Moreover, our cluster center searching strategy is efficient, in that only a small number of candidates are taken into consideration.

Similar to SLIC[1], we aggregate pixels based on the similarity between cluster centers and neighboring pixels. We define our similarity metric by adopting the spatial weighed term d_s and color weighed term d_c in Eqn. 2,

$$d(p,q) = d_c(p,q) + \lambda d_s(p,q), \tag{5}$$

where $d(p,q)$ is the overall similarity of p and q and λ adjusts the influence between d_c and d_s for controlling compactness. But, in contrast to SLIC, our pixel aggregation scheme does not require the adjustment of cluster centers like k-means. Like other superpixel methods [1, 7] that do not enforce the connectivity in superpixel, a post-processing procedure is used that merges the small discontinued components into adjacent large superpixels.

3.3. Segmentation Control

Like other superpixel methods require controlling the number of superpixel, our CSS method accept the user input of desired segmentations. We first divide the image into grids, and the number of grids is approximate to the desired number of superpixels. Then, we randomly sample the pixels for each grid as the initial position of the cluster center candidates. Finally, the cluster centers will iteratively update to the local optimal positions via proposing density channel. In general, the total number of cluster centers will be slightly less than the original input sampling pixels, since some cluster center candidates will convergence to same local optimal positions. Thus, the number of cluster center can be controlled by the user input. Since the image space may contain high frequency context, the density channel also contains many local optimal density peaks. But, in the initial sampling procedure, only one candidate pixel in the local grid is sampled. Thus, for each local region, only one cluster center is preserved, and other local optimal peaks will be ignored. In smooth region, the density channel will be also smooth, but the position of cluster centers are not as critical as clusters that near object boundary or object components.

3.4. Bipartite Graph Grouping

Given a set of superpixels and pixels, grouping is done to divide superpixel or pixel sets into k subsets that include superpixels or pixels. Following [11], superpixels can be utilized as powerful cues for grouping problems and improve the performance of segmentation in terms of accuracy and efficiency. But different superpixel algorithms have different segmentation appetites; *e.g.*, [7] favors generating highly flexible superpixels, but tends to overlap with multiple objects and the distribution of superpixel size is unbalanced. In contrast to [7], for superpixel methods with strong spatial and regular constraints (*e.g.*, CSS), superpixels are regular and have similar size to each other. The pros and cons are also obvious, the former approach tends to discover global shapes but fails to identify small components, and the latter can easily discover local small components but fails to capture global shapes. The combination of the two different superpixel methods will have the complementary effects. We integrate our CSS method into multiple superpixel grouping frameworks in [11]. We test our segmentation method for both nature image segmentation for the BSDS500 dataset[2] is shown in Fig. 3 and microscopy image segmentation as shown in Fig. 5.

4. Experiments and Applications

We evaluate our superpixel method based on widely-used metrics: boundary recall (BR), corrected under-segmentation error (CUE), achievable segmentation accuracy (ASA) and frame per seconds (FPS). BR evaluates the ability of superpixel discovering object boundaries. ASA measures the upper bound performance of superpixel adopting output superpixels as the basic processing unit for segmentation. CUE measures the ability of superpixel overlapping only on one object component. Several state-of-the-art methods are selected for comparisons: SLIC[1], LSC[10], TP[9], SEEDS[19].

In the experiment, we set $\sigma_c = 150$, $\sigma_s = 60$, $\lambda = 0.1$ as fixed parameters to compare performance. We compute

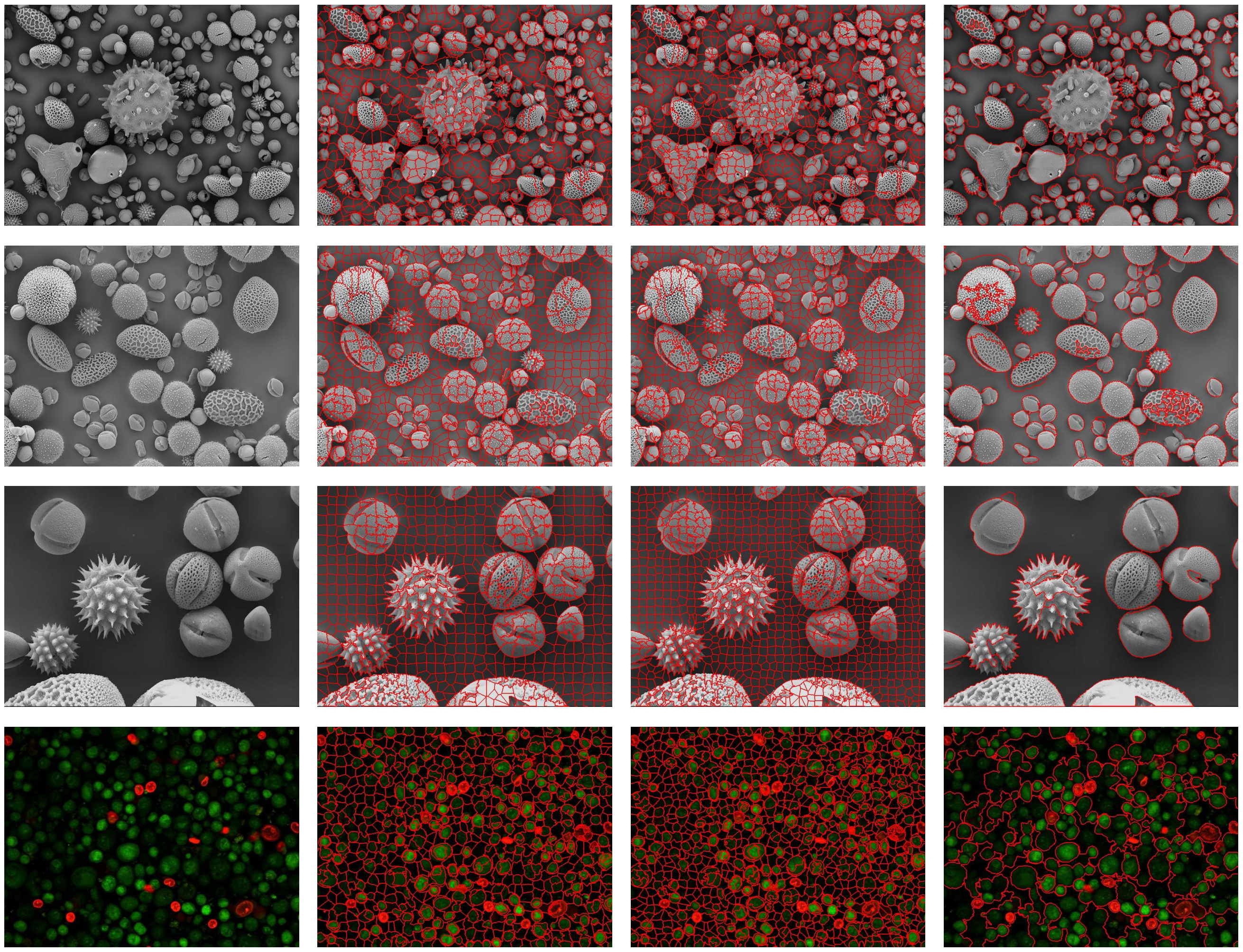


Figure 3. Segmentation results for microscopy images. We illustrate superpixel segmentation in the second and the third columns, and grouping results based on superpixel cues are shown in the fourth column. We adopt our CSS method for bipartite graph grouping [11].

the density channel in a $(7, 7)$ patch for balancing efficiency and performance. All the experiments are conducted on an i3-2130 @ 3.4GHz CPU.

4.1. Comparison with the State-of-the-art Methods

We compare our CSS method with other state-of-the-art methods from various aspects. Fig. 4 shows the comparison in terms of BR, CUE, ASA and FPS between the state-of-the-art methods. Fig. 6 shows a visual comparison between several competing methods and CSS.

TP[9] achieves regular and smooth superpixel segmentation with a clear boundary (Fig. 6(a)); however, TP achieves almost the worst boundary adherence in terms of BR (Fig. 4(a)) since it does not fully utilize the color information for superpixel generation. Moreover, TP is the slowest among all compared methods. SLIC[1] is one of the most widely-used superpixel methods due to its simple implementation and regular shape superpixel with comparable segmentation performance. While SLIC is popular, it rarely support real-time applications because it generally provides $3 \sim 5$ fps to generate superpixels for a single image. SEEDS[19] is fast and achieves state-of-the-art efficiency. However, while SEEDS achieves good segmentation performance, it meets the drawback that it generates complicated boundaries and diverse shapes, which favor an adhesive object boundary but fail to represent objects uniformly (see Fig. 6(c)). Moreover, SEEDS cannot provide explicit control over granularity of superpixel leading to various parameters for images with different resolution[1]. LSC[10] provides uniform su-

[1]As illustration of Fig. 4(d), SEEDS crash when image resolution of 1920×1080 since appropriate parameters is difficult to find.

perpixel and a good adherence to image boundary. While LSC is similar to SLIC, it differs in the kernel function, LSC achieves similar superpixel shapes and behavior in the smooth region, but also sensitive to texture region that provide a complicated superpixel boundary. Moreover, LSC is much slower than SLIC since kernel function is used, and runs for approximately $1s$ for a modest-size image.

As Fig. 4 illustrates, our CSS achieves comparable performance in terms of several metrics and significant efficiency gain compared with other previous arts. In contrast to most state-of-the-art methods that aggressively adhere to object boundary [12, 19] and fail to generate regular shape superpixel, our CSS is able to generate visually pleasing superpixels that generally have a regular shape and clear boundary, and are also insensitive to the texture region.

Efficiency Comparison In this section, we compare our CSS method with previous arts in terms of efficiency as shown in Fig. 4(d). SEEDS[19] is the most efficient superpixel method among the previous arts and achieves approximately 40∼50 ms (nearly real-time) for a single image in the BSDS500 dataset. And the most widely-used SLIC[1] achieves approximate 0.5s for a single image and its kernelized version implementation, LSC[10], requires more than 1s to process a similar-size image. Our CSS outperforms these previous arts in terms of efficiency and can process modest-size images in real-time, which provides solution for supporting other computationally demanding problems, *e.g.*, real-time tracking or real-time detection. For testing images in the BSDS500[2], our CSS takes approximately 15 ms for a single image, 5∼10 ms for generating the density channel and 4∼6 ms for pixel aggregation. Thus, our CSS can be a good replacement for generating superpixel efficiently.

We also compare the computational complexity of the-state-of-the-art methods. In fact, SEEDS, SLIC, and LSC have a computational complexity of $\mathcal{O}(N)$, where N is the number of pixels. The computational complexity of our CSS is also $\mathcal{O}(N)$. For some early methods, the computational complexity is comparatively high, *e.g.*, for ERS[12], it is $\mathcal{O}(N^2 \lg N)$.

4.2. Robustness Evaluation

As illustrated in Fig. 7, we test the robustness of our CSS by adjusting σ_s, σ_c and λ. σ_c and σ_s adjust the measurements of the color similarity and spatial similarity. Large σ_c or σ_s indicates small strictness of distance for similarity measurement and vice versa. λ is the tradeoff parameter for adjusting the weight of color term and spatial term, and it controls the compactness of superpixels, as a large λ encourages compact superpixel and a small λ encourages better color homogeneity. Fig. 7 shows that our CSS achieves stable segmentation when σ_c, σ_s and λ change significantly, which is critical for the practical applications.

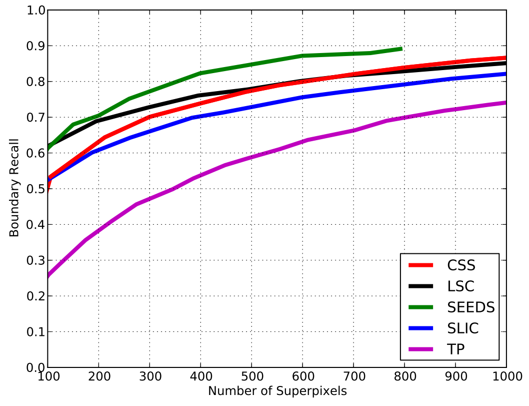
Bipartite Graph Grouping We compare the grouping results when using different superpixels algorithms in Fig. 5. We adopt the bipartite graph segmentation method[11] to utilize superpixel for grouping tasks. We test different types of superpixels as initial inputs, namely, CSS, Mean Shift [6] and the combination of two methods CSS + Mean Shift. As illustrated in Fig. 5, when integrating CSS and Mean Shift together as the input of the bipartite graph grouping, the two different superpixels achieve the effect of complementarity and yield better visual segmentation results. Moreover, due to the introduction of superpixel cues, the segmentation methods are also efficient, and only take approximately 5s for a single image of 800×600.

Microscopy Image Segmentation Many popular superpixel and segmentation methods become increasingly expensive in segmentation tasks, especially in large resolution images. For microscopy images, *e.g.*, mitochondria, cell images from electron micrographs (EM), the resolution is comparatively large and directly downsampled EM images will affect accuracy for further applications and analysis. We adopt CSS to generate a set of superpixels for reducing the computational primitives, and then utilize superpixel cues for grouping via the approach in [11]. In Fig. 3, we illustrate the example results of superpixels and segmentations including the original image, superpixel segmentation results with different numbers of superpixels and final segmentation results.

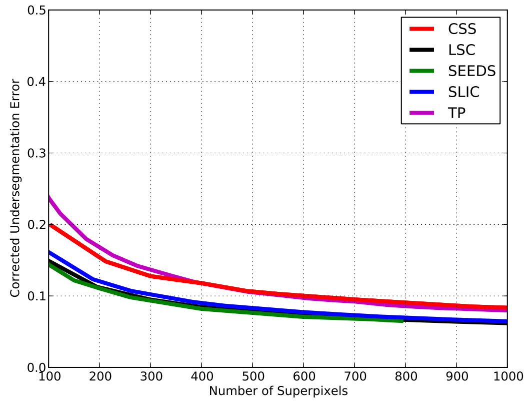
Failure Cases Our CSS achieves competing performance in comparatively large number of superpixel. However, when the desired number of superpixel is comparatively small (*e.g.*, $<$100), our CSS suffers degradation since the distance metrics is less effective when spatial distance is large. Moreover, our CSS favors regular shape of superpixels, but when desired segmentations are irregular, superpixel segmentation will be also less effective.

5. Conclusion

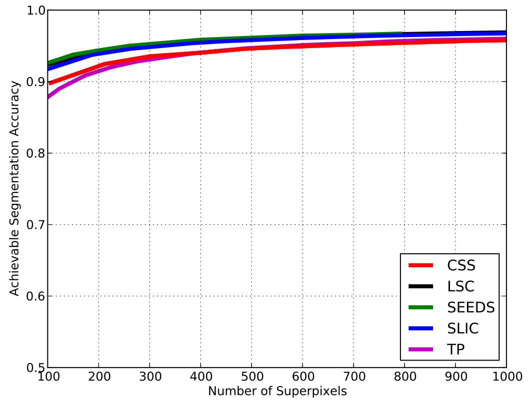
In this paper, we propose an efficient cluster sensing superpixel (CSS) method that addresses the superpixel problem as searching the cluster centers via the proposed density channel and aggregating pixels via kernelized distance metrics. We also empirically compare several kernel functions for generating the density channel and measuring the similarity between two pixels. Extensive experiments show that our method achieves competitive performance with several state-of-the-arts methods. Despite the effectiveness of our CSS, we also shows that our CSS method is able to run in real-time in modest-size images and outperforms other



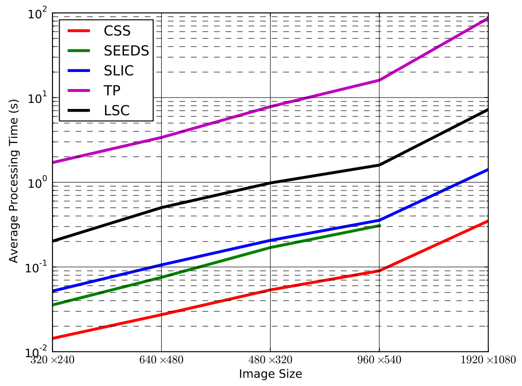
(a) BR



(b) CUE

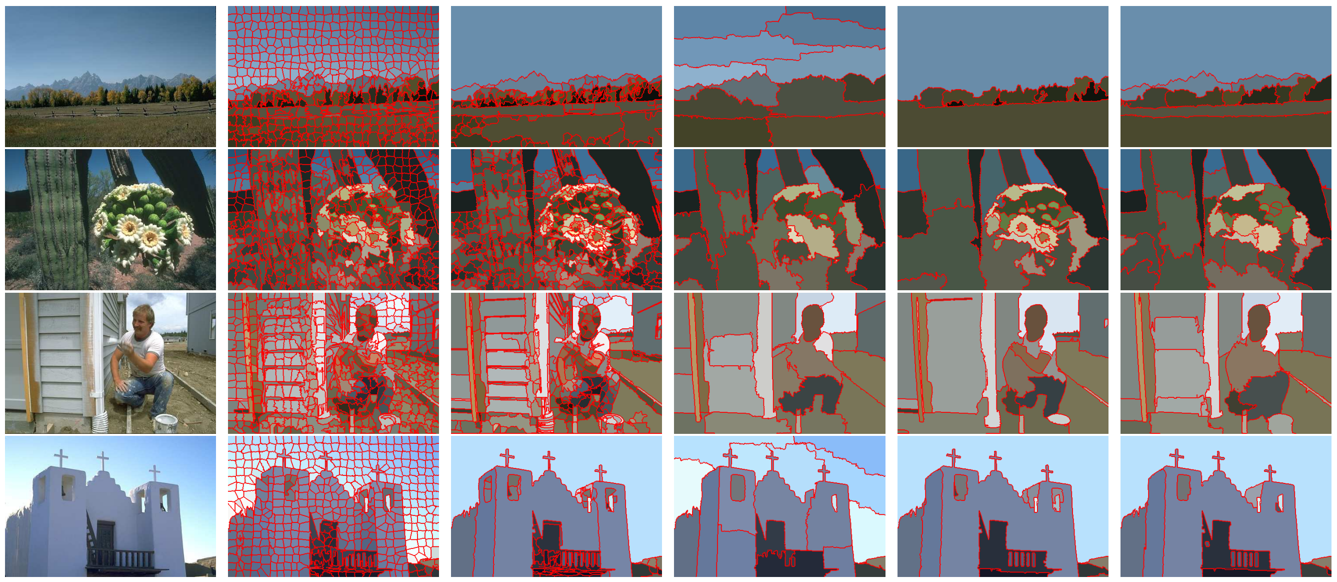


(c) ASA



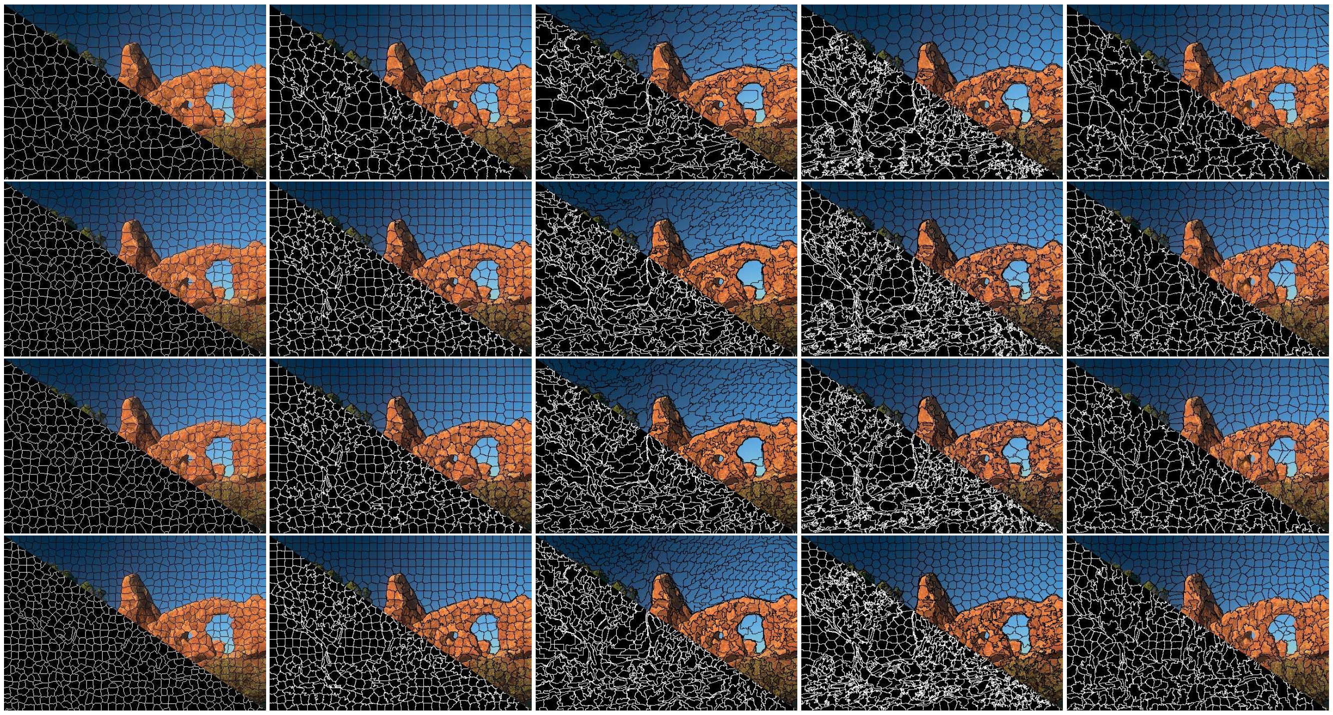
(d) FPS

Figure 4. Performance benchmark on BSDS500[2] in terms of BR, CUE, ASA and FPS.



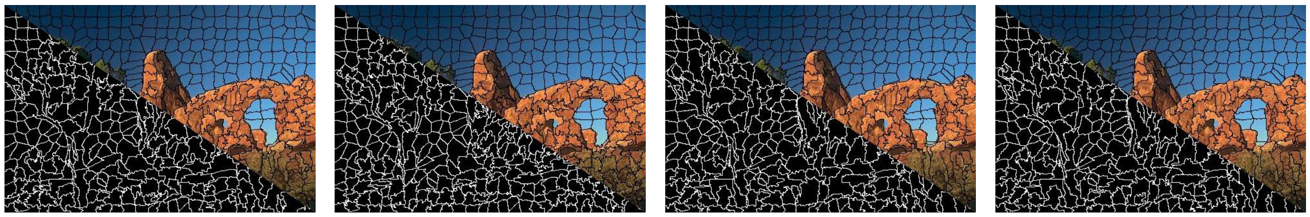
(a) Original Image (b) CSS (c) Mean Shift (d) Grouping (CSS) (e) Grouping (MS) (f) Grouping(CSS+MS)

Figure 5. The visual results of adopting the bipartite graph grouping for segmentation. (a) original image, (b) the superpixel results of CSS, (c) Mean Shift results [6], (d) the segmentation results based on CSS, (e) segmentation results based on Mean Shift, (f) segmentation results integrated CSS and Mean Shift.

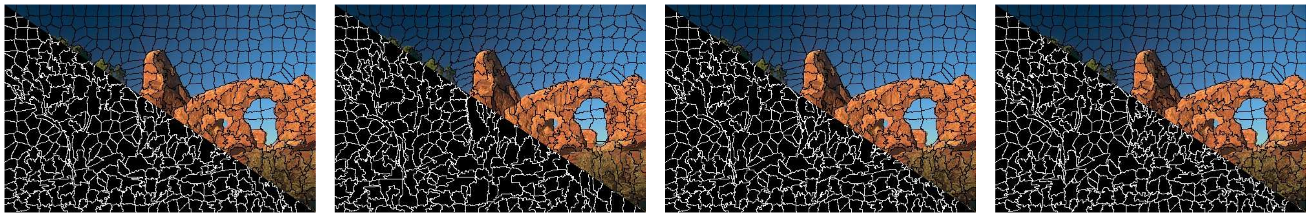


(a) TP[9] (b) SLIC[1] (c) SEEDS[19] (d) LSC[10] (e) CSS

Figure 6. Visual comparison between several state-of-the-art methods: TP[9], SLIC[1], SEEDS[19] and LSC[10] at the superpixel number of 400, 600, 800, 1000 respectively (from top to bottom).



(a) $\sigma_c = 150, \sigma_s = 60, \lambda = 0.1$ (b) $\sigma_c = 150, \sigma_s = 40, \lambda = 0.1$ (c) $\sigma_c = 150, \sigma_s = 80, \lambda = 0.1$ (d) $\sigma_c = 120, \sigma_s = 60, \lambda = 0.1$



(e) $\sigma_c = 180, \sigma_s = 60, \lambda = 0.1$ (f) $\sigma_c = 150, \sigma_s = 60, \lambda = 0.05$ (g) $\sigma_c = 150, \sigma_s = 60, \lambda = 0.15$ (h) $\sigma_c = 150, \sigma_s = 60, \lambda = 0.2$

Figure 7. Superpixel results for different σ_c, σ_s and λ at 500 superpixels.

state-of-the-art methods in different resolutions. Moreover, we also integrate our CSS into the bipartite graph segmentation framework for microscope image segmentation.

Acknowledgements This work is supported in part by Natural Science Foundation of China (NSFC) under contract No. 61303151, in part by the GRF 16211615.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012. 1, 2, 4, 5, 6, 8

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 2011. 1, 2, 4, 6, 7

[3] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 2

[4] M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S. Hu. Global contrast based salient region detection. *PAMI*, 2015. 1

[5] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. S. Torr, and S.-M. Hu. Global contrast based salient region detection. *PAMI*, 2014. 1

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 2002. 1, 6, 7

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004. 1, 2, 4

[8] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *PAMI*, 2004. 2

[9] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *PAMI*, 2009. 2, 4, 5, 8

[10] Z. Li and J. Chen. Superpixel segmentation using linear spectral clustering. In *CVPR*, 2015. 1, 2, 4, 5, 6, 8

[11] Z. Li, X.-M. Wu, and S.-F. Chang. Segmentation using superpixels: A bipartite graph partitioning approach. In *CVPR*, 2012. 1, 4, 5, 6

[12] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy-rate clustering: Cluster analysis via maximizing a submodular function subject to a matroid constraint. *PAMI*, 2014. 1, 2, 6

[13] M. Maire and S. X. Yu. Progressive multigrid eigensolvers for multiscale spectral segmentation. In *ICCV*, 2013. 1, 2

[14] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *CVPR*, 2008. 2

[15] M. Reso, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Temporally consistent superpixels. In *ICCV*, 2013. 2

[16] A. Rodriguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 2014. 1, 2

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 1997. 1, 2

[18] C. J. Taylor. Towards fast and accurate segmentation. In *CVPR*, 2013. 2

[19] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool. Seeds: Superpixels extracted via energy-driven sampling. *IJCV*, 2015. 1, 2, 4, 5, 6, 8

[20] Y. Xie, H. Lu, and M.-H. Yang. Bayesian saliency via low and mid level cues. *TIP*, 2013. 1

[21] K. Yamaguchi, D. McAllester, and R. Urtasun. Robust monocular epipolar flow estimation. In *CVPR*, 2013. 1

[22] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*. 2014. 1

[23] F. Yang, H. Lu, and M.-H. Yang. Robust superpixel tracking. *TIP*, 2014. 1

[24] J. Yao, M. Boben, S. Fidler, and R. Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In *CVPR*, 2015. 2

[25] Y. Yuan, J. Fang, and Q. Wang. Robust superpixel tracking via depth fusion. *TCSVT*, 2014. 1

[26] W. Zhu, S. Liang, Y. Wei, and J. Sun. Saliency optimization from robust background detection. In *CVPR*, 2014. 1