High quality superpixel generation through regional decomposition

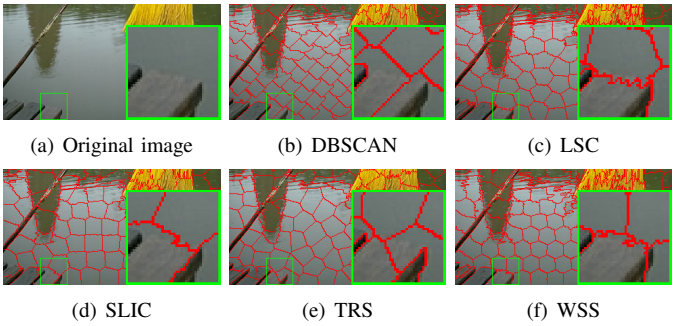
Yunyang Xu, Xifeng Gao, Caiming Zhang, Jianchao Tan, Xuemei Li

***Abstract*—Superpixel generation is increasingly an important area for computer vision tasks. While superpixels with highly regular shapes are preferred to make the subsequent processing easier, the accuracy of the superpixel boundaries is also necessary. Previous methods usually depend on a distance function considering both spatial and color coherency regularization on the whole image, which however is hard to balance between shape regularity and boundary adherence, especially when the desired number of superpixels is small. In addition, non-adaptive parameters and insufficient contour information also affect the performance of segmentation. To mitigate these problems, we propose a robust divide-and-conquer superpixel segmentation method, of which the core idea is that we apply a new contour information extraction and a pixel clustering to separate the input image into flat and non-flat regions, where the former targets shape regularity and the latter emphasizes boundary adherence, followed by an efficient hierarchical merging to clean up tiny and dangling superpixels. Our algorithm requires no additional parameter tuning except the desired number of superpixels since our internal parameters are self-adaptive to the image contents. Experimental results demonstrate that for public benchmark datasets, our algorithm consistently generates more regular superpixels with stronger boundary adherence than state-of-the-art methods while maintaining a competitive efficiency. We will release our code upon acceptance.**

***Index Terms*—Regional partition, Superpixel, Self-adaption, Saliency detection.**

I. Introduction

SUPERPIXELS are clusters of pixels with similar properties, such as brightness, grayscale, color, or texture. The idea of superpixel generation was firstly introduced in [1], and its concept of over-segmentation is an increasingly important pre-processing step that can reduce redundant information of image and computational cost for downstream computer vision and image processing tasks, such as image segmentation [2]–[4], target tracking [5]–[7], object recognition [8], [9], classification [10], [11], saliency detection [12]–[14], stereo matching [15] and so on. In the past two decades, many superpixel segmentation algorithms have been proposed.



(a) Original image (b) DBSCAN (c) LSC

(d) SLIC (e) TRS (f) WSS

Fig. 1. The region with similar colors.

A superpixel segmentation algorithm may have diverse advantages and evaluation metrics in different tasks. However, a good superpixel segmentation algorithm should generally meet all the following requirements:

- Regularity: the shape of superpixels should be as regular as possible.
- Pixels similarity: pixels inside a superpixel should have similar colors and brightness.
- Boundary adherence: the boundaries of superpixels should adhere well to the non-trivial boundaries of the image.
- Efficiency: the generation of superpixels should be fast enough to speed up downstream applications.

To generate superpixels that have compact shape and similar visual perceptions, existing methods utilize features derived from image colors and pixel positions in Euclidean space (e.g. SLIC [16], LSC [17], TRS [18], DBSCAN [19] and WSS [20]). However, relying purely on these properties is not enough to distinguish pixels with similar colors but different semantic regions, which often leads to incorrect segmentation (Fig. 1). Moreover, different regions of an image may be treated adaptively to better satisfy the requirements, such as regularity and boundary adherence. Previous methods typically adopt unified processing for the whole image, which may generate undesired results. As shown in Fig. 2, although SLIC [16] and LSC [17] produce superpixels with high regularity, the boundary adherence is not satisfied well in many regions. On the contrary,ERS [21] and GMM [22] align the superpixels with image boundaries excellently, but suffer from low shape regularity. In addition, using the same set of parameters and objective functions for general images usually leads to

This work was supported partly by the National Natural Science Foundation of China under Grant Nos. 62072281, 62007017.

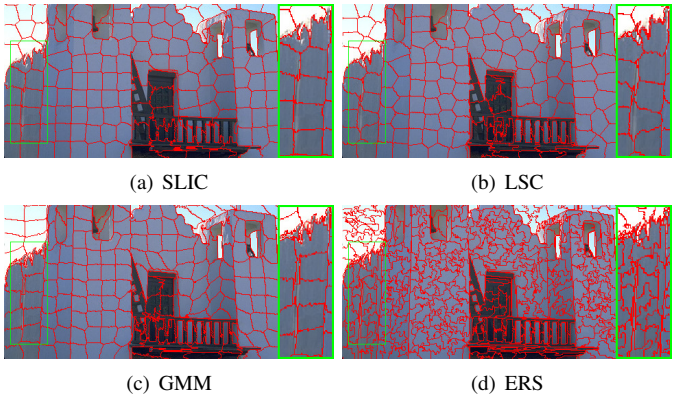
Yunyang Xu is with the School of Software, Shandong University, Jinan, China, 250101, e-mail: (xuyunyang@mail.sdu.edu.cn).

Xifeng Gao is with the Computer Science Department, Florida State University, Tallahassee, Florida, e-mail: (gxf.xisha@gmail.com).

Caiming Zhang is with the School of Software, Shandong University, Jinan 250101, China Shandong Co-Innovation Center of Future Intelligent Computing, Yantai, China, 264025, e-mail:(czhang@sdu.edu.cn).

Jianchao Tan is currently a Staff Research Scientist in Kuaishou Technology, working for Auto-mated Machine Learning, Computer Graphics and Vision, e-mail: (tanjianchaoustc@gmail.com).

Xuemei Li(✉) is with the School of Software, Shandong University, Jinan, China, 250101 e-mail: (xmli@sdu.edu.cn).



(a) SLIC (b) LSC

(c) GMM (d) ERS

Fig. 2. Visualization of different superpixel segmentation algorithms.

inconsistent and sub-optimal performance since the content of images may vary dramatically.

We solve the contradiction issue of achieving regularity and maintaining boundary adherence of superpixels by first grouping pixels (Section V) into flat and non-flat regions, and then generating the corresponding superpixels for them, respectively (Section VII). In this case, the regularity of superpixels in flat regions is ensured and the boundary adherence in non-flat regions is achieved. We also design self-adaptive parameters for extracting superpixels in non-flat regions that account for images with varying scenes. Finally, we perform a hierarchical merging operation to clean up tiny and isolated pixels. The effectiveness of our approach has been verified by comparing our method with 16 state-of-the-art methods (Section VIII) on three datasets with ground truth. We also apply our generated superpixels to the saliency detection application, demonstrating the advantages of our generated superpixels.

The main contributions are summarized below:

- In addition to the traditional color and position properties of a pixel, we propose to include the contour information into the similarity metric for pixel clustering.
- We propose to decompose an image into flat and non-flat regions so that different segmentation strategies can be adopted to achieve the balance between the high regularity and the strong boundary adherence of superpixels.
- We propose a self-adaptive weight tuning approach to combine the color, position, and contour information of pixels to align superpixels in non-flat regions. As a result, our approach can consistently produce high-quality superpixels for images with varying contents.

II. Related Work

In the past two decades, many algorithms have been proposed to generate suitable superpixels. These algorithms can be roughly classified into three categories: graph-based, clustering-based, and deep learning-based.

A. Graph-based Methods

The graph-based superpixel generation methods treat each pixel as a node in the graph and the edge weight between two nodes is proportional to the similarity between adjacent pixels. Superpixels are produced by minimizing the cost function defined over the graph. ERS [21] uses an energy function that includes a random walking entropy rate of an image and an equilibrium term. SOHOE [23] optimizes the k-means segmentation result by using a higher-order energy function that can adaptively adjust the energy terms based on texture measurements in different local regions of the image. ANRW [24] first employs non-local random walk to obtain the initial superpixel segmentation result, after merging small superpixels, compact and regular superpixels will be obtained. SH [25] adopts Boruvka's minimum spanning tree (MST) method to generate superpixels. This efficient algorithm has high segmentation accuracy, but the produced superpixels are very irregular. HSPDM [26] adopts a hierarchical superpixel-to-pixel approach to determine the correspondence between two images and uses superpixel-level pairing to drive pixel-level matching to obtain finer texture details.

B. Clustering-based Methods

Various clustering methods are applied to cluster pixels into superpixels. LSC [17] applies linear spectral clustering to generate superpixels, which produce superpixels with good shape regularity and high boundary adherence in a relatively short time. WSGL [27] adopts a new strategy with two distinct criteria for global and local refinement of the boundary pixels. This algorithm can produce regular superpixels, and the boundary adherence is good. Zhang et al. [28] propose a fast density-based noise-applied spatial clustering (DBSCAN) and superpixel binning with edge penalty to segment SAR images adaptively. This method can quickly generate compact and regular superpixels. Jing et al. [29] propose a shrink-expand search strategy (CES) that explicitly exploits the continuity information contained in neighboring pixels and enforces the connectivity of superpixels without any post-processing steps. This method can generate superpixels with high boundary adherence. Our method introduces contour information into pixel clustering to divide the image into flat and non-flat regions and solve them with different strategies, achieving a better balance between regularity and adherence.

C. Deep Learning-based Methods

Deep Neural Networks (DNN) is a powerful tool to perform image tasks like classification, segmentation, and so on. SSN [30] proposes a DNN model for superpixel sampling, which can learn superpixel segmentation through the deep network end-to-end. SSFCN [31] proposes a simple fully convolutional network to predict superpixels on a regular image grid quickly. LNSnet [32] proposes an unsupervised method based on CNN, with a re-adjusting of the weight gradient based on the channel and spatial context. FGSLT [33] proposes a local similarity loss function to improve segmentation accuracy. The generated superpixels are characterized by topological consistency. AINet [34] proposes a novel association implant (AI) module that allows the network to accurately capture the relationship between a pixel and its surrounding grid, which has good segmentation accuracy. However, these methods will

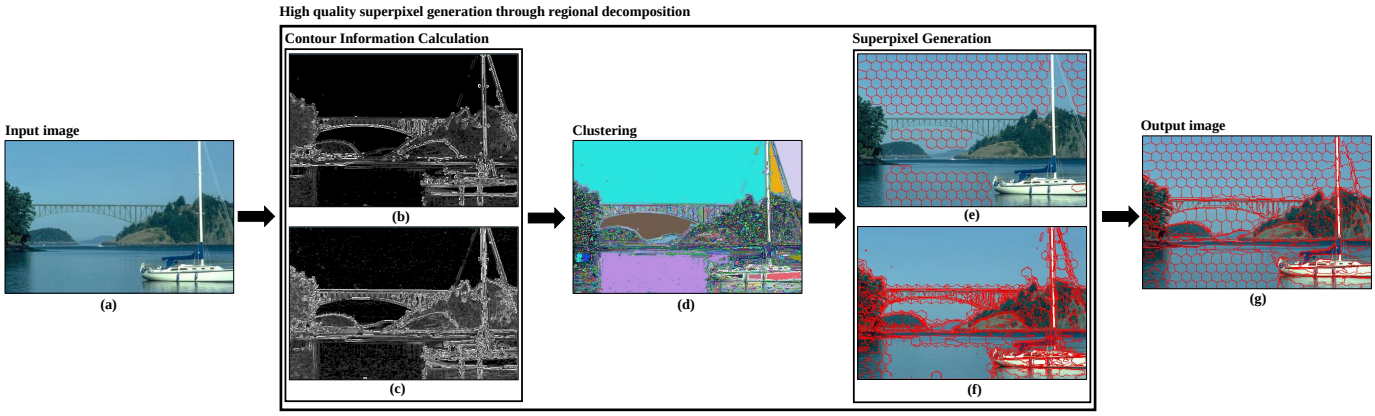


Fig. 3. Algorithm flow. (a) Input. (b) and (c) are 4-neighborhood and 8-neighborhood contour information, respectively. (d) Clustering. (e) Superpixels in flat regions. (f) Superpixels in non-flat regions. (g) Output.

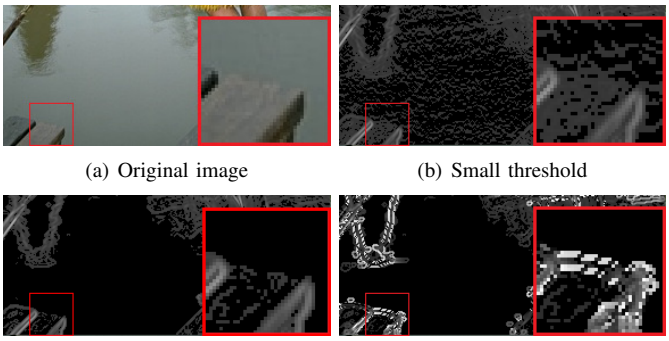
suffer from the performance drop when the input image is out of the distribution of the training dataset, as described in [35].

III. Overview

In order to obtain a satisfactory superpixel segmentation result, we propose a robust approach by solving three critical issues: how to distinguish pixels reasonably; how to balance between keeping regularity and maintaining tight boundary adherence; how to choose appropriate parameters. Our algorithm consists of four steps: contour information calculation, clustering, seed point generation, and superpixel generation. Given an input image (Fig. 3(a)), First, we perform a new contour extraction step, i.e. 4 or 8 neighborhood contour information (Fig. 3(b) and Fig. 3(c)), on the edge-enhanced image. The purpose is to combine the extracted contour information with the color and spatial properties to better distinguish pixels with similar colors. Second, we decompose the image into flat and non-flat regions by introducing a new clustering method (Fig. 3(d)), where the former targets the shape regularity and the latter emphasizes boundary adherence. Third, we compute the seed points based on the clustering. While the superpixels in flat regions can be generated in advance (Fig. 3(e)), the marking of superpixels in non-flat regions requires special care. We design a new distance function with adaptive parameters according to features in non-flat regions. The corresponding superpixels are generated in an iterative process (Fig. 3(f)). Finally, The superpixels (Fig. 3(g)) are obtained after merging scattered and isolated pixels. Compared with global approaches RSS [36], TRS [18] and SLIC [16], our divide-and-conquer strategy generates superpixels with more regularity in flat regions and tighter boundary adherence in non-flat regions. Meanwhile, since our parameters are set adaptively based on the image content of non-flat regions, our approach can produce high-quality results consistently for images with varying scenes.

IV. Contour Information Calculation

When the pixels near the image edges have similar colors, it is difficult to distinguish them only based on color information and simple spatial distance information (see an example in Fig. 1). We propose to incorporate the contours of these pixels to enhance the ability of our algorithm to distinguish such pixels. Contour information has been used widely for segmenting images. While a small threshold leads to false edges (Fig. 4(b)), a large threshold often results missing edges (Fig. 4(c)). To obtain contour information more reliably, we calculate the contour information on the edge-enhanced image, and the result is shown in Fig. 4(d).



(a) Original image (b) Small threshold

(c) Large threshold (d) Result on edge-enhanced images

Fig. 4. Different contour information

Since the gradient of the boundary pixels is larger than that of pixels in smooth regions, we estimate pixels' contour information using their gradients. Firstly, an edge detection algorithm (i.e. RCF [37]) is used to obtain the edge image, which is further processed by non-maximum suppression. Then each edge has only one or two pixels width and some boundary points are lost in certain areas, as shown in Fig. 5(b). However, contour information of pixels on both sides of the boundary is needed for superpixel generation, thus the partially lost boundary points are complemented to meet the needs of contour information calculation.

For boundary completion, we assign the value of local-regional (15 shifted neighborhoods, as shown in the Fig. 7(a)) of each boundary point to be the value of each boundary point, i.e. Set the value of $j_1, j_2, ..., j_{15}$ to the value of boundary point $\hat{p}$. The local-regional of boundary point can cover the pixels on both sides of the boundary in most cases, partially deleted

boundaries can be completed by modifying the value of the local-regional of boundary points, as shown in Fig. 5(c). In CIELAB color space, since human eyes are more sensitive to brightness, contour information of pixels is calculated at the edge-enhanced L channel of the image. Fig. 6(e) is the edge-enhanced L channel of the input image, which is obtained by adding Fig. 6(d) to Fig. 6(c).

(a) (b) (c)

Fig. 5. (a) Original image. (b) Edge image obtained by RCF and non-maximum suppression. (c) The result of boundary point completion.

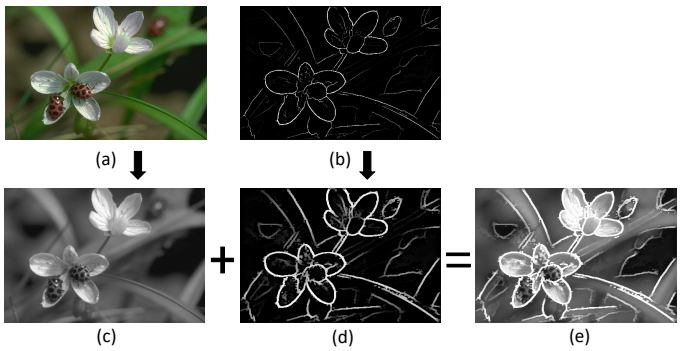


Fig. 6. (a) Original image. (b) Edge image obtained by RCF and non-maximum suppression. (c) L channel of the image. (d) The result of boundary point completion. (e) Edge-enhanced L channel of the image.

A. 4-Neighborhood Contour Information

Let $\hat{t}$ denote the 4-neighborhood gradient value of pixel p of the edge-enhanced L channel of the image. The neighborhood of the pixel p is shown in Fig. 7(b).

j_1	j_2	j_3	j_4
j_5	j_6	j_7	j_8
j_9	j_{10}	$\hat{p}$	j_{11}
j_{12}	j_{13}	j_{14}	j_{15}

(a)

p_1	p_2	p_3
p_4	p	p_5
p_6	p_7	p_8

(b)

Fig. 7. (a) The local-regional of boundary point $\hat{p}$. (b) Pixel p and its neighbor pixels.

$\hat{t}$ is defined as:

$$\hat{t} = \sqrt{t_h^2 + t_v^2} \tag{1}$$

Where, $t_h = p_4 - p_5$ and $t_v = p_2 - p_7$ represent horizontal and vertical change of the gray scale value of the edge-enhanced L channel of the image, respectively.

4-neighborhood contour information t is defined as:

$$t = \begin{cases} 0, & \hat{t} < T \\ \hat{t}, & otherwise \end{cases} \tag{2}$$

Where T is the threshold, if $\hat{t} < T$, the pixel is considered to be in a flat region, and its 4-neighborhood contour information is set to 0. In order to eliminate most of the false edges while retaining the contour information of the object, we set T to 5 through experiments.

B. 8-Neighborhood Contour Information

Prewitt operator is applied to calculate 8-neighborhood gradient values $\hat{g}$ of the pixel p in the edge-enhanced L channel of the image:

$$\hat{g} = \sqrt{g_h^2 + g_v^2} \tag{3}$$

Where:

$$g_h = (p_1 + p_4 + p_6) - (p_3 + p_5 + p_8) \tag{4}$$

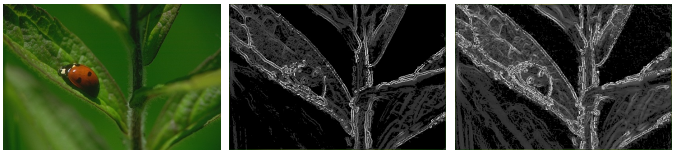
$$g_v = (p_1 + p_2 + p_3) - (p_6 + p_7 + p_8) \tag{5}$$

8-neighborhood contour information g is defined as:

$$g = \begin{cases} 0, & \hat{g} < G \\ \hat{g}, & otherwise \end{cases} \tag{6}$$

Where G is the threshold. When $\hat{g} < G$, the pixel is considered to be in a flat region, and its 8-neighborhood contour information is set to 0. In order to obtain more contour information than the 4-neighborhood contour, G is set to 5 through experiments.

Fig. 8 demonstrates examples of the extracted 4-neighborhood and 8-neighborhood contour information. It can be seen that 8-neighborhood contour information has a higher value than 4-neighborhood contour information and has more details, but also contains more noise information.



(a) (b) (c)

Fig. 8. (a) Original images. (b) The 4-neighborhood contour information. (c) The 8-neighborhood contour information.

V. Clustering

We propose a new neighborhood distance for clustering pixels to decompose the image into flat and non-flat regions. Since the image has color space continuity, there is a certain similarity between pixels in the same area without considering the noise effect. Therefore, the similarity between pixels can be judged by measuring the color distance of adjacent pixels to cluster some pixels. However, in regions with low contrast, only color similarity is often insufficient for a satisfactory clustering result. It is necessary to define contour information for each pixel p to determine whether p is on the flat regions. Our neighborhood distance is obtained by aggregating both the Euclidean distance in CIELAB color space and the contour information. The contour information of pixel p should be defined by its neighbors (4- and 8-neighborhood) pixels for better locality. Next, we analyze how the contour information

of p should be defined theoretically. p can be regarded as obtained by sampling from a function $P(x, y)$. According to Taylor expansion formula and numerical approximation theory, $P(x, y)$ can be approximated by a relatively low-order polynomial in flat regions, whereas $P(x, y)$ needs to be approximated by a relatively high-order polynomial in non-flat regions. 4-neighborhood (five pixels) and 8-neighborhood (nine pixels) can be fitted with bilinear and biquadratic polynomial functions, respectively. The contour information describes the shape of the fitting function. Consequently, the contour information in the flat regions and the non-flat regions is defined by the 4- and 8-neighborhood, that is, we use Eq. (2) and (6) to define the contour information. In order to obtain a better flat regions result and make the region division more accurate, we use contour information defined by 4-neighborhood. As can be seen from Fig. 7 (Supporting Document), using 4-neighborhood contour information produces better clusters than 8-neighborhood contour information. Better clustering means a more accurate division of regions. When segmenting non-flat regions, 8-neighborhoods are used to define the contour information of pixels in the region, which will be described in Section VII.

A. Neighborhood Distance

For each pixel point p, we calculate the neighborhood distance between pixel p and its neighboring pixel $p_m (m = 1, 2, \ldots, 8)$ to measure the similarity between this pixel and its surrounding pixels. Let $d(p, p_m)$ be the distance between pixel p and its neighbor pixel p_m. $d(p, p_m)$ is computed as:

$$d(p, p_m) = t_m + \sqrt{(l - l_m)^2 + (a - a_m)^2 + (b - b_m)^2} \quad (7)$$

Where l, a, b and l_m, a_m, b_m are the values (l, a, b) of the pixel p and the neighboring pixel p_m in the CIELAB space, and t_m is the 4-neighborhood contour information of pixel p_m. Obviously, eight neighborhood distances can be used to measure the similarity between p and p_m. If the value of $d(p, p_m)$ is small, the similarity between pixels p and p_m is large. Based on the above analysis, we propose a new clustering algorithm based on the similarity between pixels. At the beginning of clustering, we need to define an initial similarity set $C(p)$ for each pixel p. The elements in the set $C(p)$ are deemed to be similar pixels of the pixel p. We rank all the neighboring pixels of p according to $d(p, p_m)$ and put the pixels with the distance smaller than the third smallest distance value into a set $C(p)$. For example, for eight distances (0,0,2,2,2,3,3,4), the smallest two sets are (0,0,2,2,2). Then find all the most similar pixels of each pixel according to the neighborhood distance to complete the pixel clustering. Neighborhood distances are coerced to integers so that the smallest two sets of distances can be found.

B. Clustering

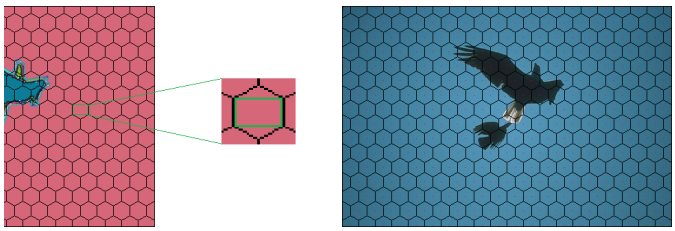
Based on the similar set of pixels obtained in previous step, for any pixel p_m in the similar pixel set $C(p)$ of the pixel p, if pixel p is also in the similar pixel set $C(p_m)$ of p_m, pixel p and p_m are considered to be in the same class. In addition, if pixels p_1 and p_2 are in the same class and p_3 and p_2 have the same class label, then they all belong to the same class. The clustering of pixels is achieved by repeating this recursive grouping. As shown in Fig. 9, large cluster regions can be generated in flat regions, while small clusters tend to appear in non-flat regions. The pseudo code of the clustering is in Algorithm 1. The threshold used in contour information computation and the way of similar pixels selection for clustering in flat regions are discussed in the Supporting Document.

(a) Original images (b) Our clustering results

Fig. 9. The same color represents the same clustering class.

VI. Seed Point Generation

The generated clusters can be considered as an initial superpixel segmentation result. In this way, the probability of finding suitable superpixel seed points in the cluster is significantly increased, and the number of iterations required for generating superpixels will be greatly reduced. Like other methods, we initialize the image with a regular hexagonal discretization to enhance superpixel regularity. We first generate K hexagons over the clustering image, K is the number of user-specified seed points. Two constraints should be met when selecting seed points: one is that the seed points cannot be boundary pixels, and the other is that they should be placed in the middle of the hexagon as far as possible so that superpixels are as large and symmetrical as possible and the number of iterations can be reduced. To reduce the search space, we only look for the seed point within a rectangular box bounded by four vertices of the hexagon (the red rectangle in Fig. 10(a)).



(a) (b)

Fig. 10. (a)K initial hexagons. (b) The red point is the initial seed point.

In order to prevent the seed point from becoming a boundary pixel, our strategy is that if a pixel p in the rectangular box is consistent with the class label of its eight neighboring pixels p_m, p is excluded as a boundary pixel and considered as a candidate seed point. To make the seed point as close to the hexagon's center z as possible, we use the following strategy:

For each candidate seed point p, calculate its distance $ds(p, z)$ and choose the pixel with the lowest distance as the seed point. The distance $ds(p, z)$ is defined as:

$$ds(p, z) = \sqrt{(x - x_z)^2 + (y - y_z)^2} \tag{8}$$

Where (x, y) and (x_z, y_z) are spatial coordinates of the pixel p and the center z of hexagon, respectively.

If no pixels located in the above rectangular search box meet the criteria, then it implies the color information in this hexagon is complicated. At this time, seed points should be generated in areas where color information varies little. The second derivative describes the acceleration, which is used to describe the change of particle motion along the orbit. The second derivative is proportional to the orbital curvature and is generally approximated using the second-order difference quotient. The relatively small second-order difference quotient indicates that little color information has changed. Therefore, we use the second-order difference quotient to measure the complexity of color information in the hexagon. We select the pixel that makes the formula (10) smallest as the seed point. The computation is as follows:

Let the second-order difference quotient of the four directions of the pixel p be $\Delta_1, \Delta_2, \Delta_3, \Delta_4$. In the CIELAB color space, $\Delta_1, \Delta_2, \Delta_3, \Delta_4$ are the values of the second-order difference quotient of four directions on l, a, b channel, respectively. And the second-order difference quotient of the four directions of L channel is defined as:

$$\begin{cases} l_{\Delta_1} = \frac{(l_1 - l) - (l - l_8)}{2} \\ l_{\Delta_2} = \frac{(l_2 - l) - (l - l_7)}{2} \\ l_{\Delta_3} = \frac{(l_3 - l) - (l - l_6)}{2} \\ l_{\Delta_4} = \frac{(l_4 - l) - (l - l_5)}{2} \end{cases} \tag{9}$$

Likewise, the second-order difference quotient of a and b channels are calculated. $l_\Delta, a_\Delta, b_\Delta$ are the mean value of the second-order difference quotient in four directions of each l, a, b channel. We measure whether pixel point p is the seed point by:

$$seed =_p \{l_\Delta + a_\Delta + b_\Delta + ds(p, z)\} \tag{10}$$

The smallest $seed$ among all pixels in the current search box is the seed point of this hexagon Fig. 10(b).

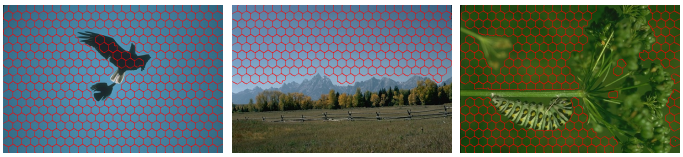
VII. Superpixel Generation

In order to ensure the superpixel regularity and improve the boundary adherence, our method first generates superpixels for flat regions based on the clustering result and the initial seed points. Then we design a distance function with adaptive parameters to partition pixels in non-flat regions only. By treating regions differently, we can generate superpixels with high regularity in flat regions and with tight boundary adherence for non-flat regions.

A. Flat Regions

As shown in Fig. 9, large clusters are usually generated in flat regions of the image. We can complete the labeling of superpixels for large clusters in advance. For pixel p in each hexagon, if its class label is equal to that of the seed point of the same hexagon, we consider pixel p and the seed point to belong to the same superpixel and let the superpixel label of the pixel p be equal to the superpixel label of the seed point. We perform this procedure for all the hexagons of the clustering.

Let the number of a hexagon containing pixels that have completed superpixel labeling be h. A hexagon is classified to be in a flat region if h is larger than $0.8 \times R$ ($R = N/K$ is the average superpixel size and N is the number of pixels of the image). Otherwise, hexagon belongs to non-flat regions. Note that, we generate superpixels only in flat regions by now. Since we perform comparison operations in the previous step on all hexagons and some superpixel labels may appear in hexagons of non-flat regions, we erase these superpixel labels and postpone the superpixel labels generation of the entire non-flat regions to the next step. After this step, flat regions have superpixels with regular shapes, while pixels of non-flat regions have not been labeled yet, as shown in Fig. 11.



(a) (b) (c)

Fig. 11. Superpixels in flat regions.

B. Non-Flat Regions

To complete the superpixel labeling of non-flat regions, we propose a distance function using parameters that are adaptive to the content of non-flat regions.

1) Distance Function: Considering the pixel q in non-flat regions and the seed point s. The distance between pixel q and seed point s is defined as $D(q, s)$, where q and s are two vectors with five components, i.e. $[l, a, b, c, g]^T$ where l is lightness information, a and b are color component information, c is spatial information represented by vector $[x, y]^T$ and g is 8-neighborhood contour information. The weighted average distance measurement is calculated as:

$$D(q, s) = \sum_{j \in I} w_j d_j^2(q, s) \tag{11}$$

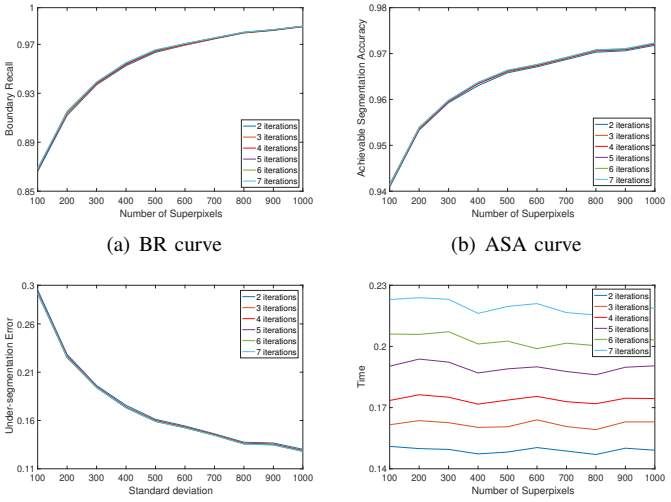
Where j indexes each component of the vector, w_j is weight parameter of the identification information j; $d_j(q, s)$ is the distance between pixel q and seed point s in the component j. In Eq. (11), we introduce the weight parameter w_j to make the distance between the components of two vectors play a different role in calculating $D(q, s)$. In theory, the similarity between two vectors is determined by the similarity between each component, but each component plays a different role for different applications. In superpixel segmentation, we use

the principle of "small distance and great weight" to calculate similarity, i.e., to give a relatively large weight w_j to a relatively small distance $d_j(q,s)$. Our experimental results show that adjusting the weights can achieve better superpixel segmentation results than fixing the weights. Superpixel segmentation is achieved by minimizing the following objective function F:

$$F = \sum_{s\in S}\sum_{q\in U} D(q,s) \tag{12}$$

Where S is the set of seed points and U is the set of non-flat regions pixels. We use Euclidean distance for the spatial distance and adopt Manhattan Distance for both the color distance and contour distance. Based on the theoretical analysis at the beginning of Section V, 8-neighborhood is used to define the contour information of pixels in non-flat regions.

By calculating the distance between pixel q and seed points around q, the superpixel label of the seed point with the minimum distance $D(q,s)$ is chosen as the superpixel label of q. We set the search scope of superpixel seed points to be within a circle where the center is the current seed point and the radius is 2.5 times the edge length of the hexagon. Once each pixel q is assigned with a superpixel label, we update the weights and seed points. Our superpixel segmentation will converge after several iterations. Through experimental results (Fig. 12), we find that when the iterations is 4, the numerical results (VIII-B) begin to converge. As the number of iterations increases, the numerical result tends to be constant. To trade off the quality and efficiency, the iterations of our algorithm are set to 4.



(a) BR curve (b) ASA curve

(c) UE curve (d) Time curve

Fig. 12. When the number of iterations is 2, 3,..., 7, our algorithm's numerical results on the BSD500 dataset.

2) Weights Adjustment: The principle of "small distance and great weight" is used to calculate similarity. That is, a larger weight w_j is assigned to a smaller distance $d_j(q,s)$.

Let $\hat{d}_j = d_j(q,s)$. V_j is used to represent the average intra-class distance of certain identification information j:

$$V_j = \frac{1}{K} sum_j \tag{13}$$

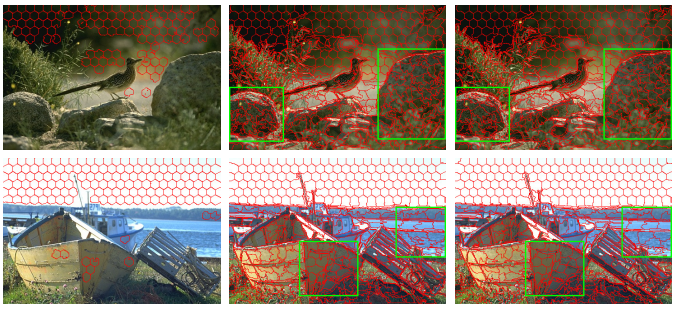
Where $j \in \{l,a,b,c,g\}$, sum_j is the sum of $\hat{d}_j^2$ of all pixels that belong to non-flat regions in the component j. The adjustment of w_l is computed as:

$$w_l = Q^\gamma / [(V_l+\sigma)^\gamma (\sum_{j\in I} Q^\gamma/(V_j+\sigma)^\gamma)] \tag{14}$$

$$Q = (V_l+\sigma)(V_a+\sigma)(V_b+\sigma)(V_c+\sigma)(V_g+\sigma) \tag{15}$$

Where $I = \{l,a,b,c,g\}$, σ is a very small constant for preventing the divisor being 0 when V_j is 0, and γ is a harmonic parameter. In our experiment, γ is set to 1.3. Similarly w_a, w_b, w_c and w_g can be calculated. The smaller V_j is, the larger w_j is, according to Eq. (14). In this way, the proportion of $d_j(q,s)$ in the calculation of $D(q,s)$ will be greater. Which realizes the mechanism that the components with the short distances play a more significant role when determining the similarity of two vectors.

Fig. 13 shows examples of superpixel generation for non-flat regions and compares fixed and adaptive weights. Compared with the fixed weighting (when all w_j are 0.2), the results by adaptive weights are better in terms of regularity, and the numerical results (Fig. 15) are better than the existing methods. In general, for non-flat regions of different images, the weights of the identification information generated by the algorithm are different. Such as, the final weights (w_l, w_a, w_b, w_c, w_g) of the two images in Fig. 13 are (0.0742, 0.1736, 0.2955, 0.3892, 0.0675) and (0.0682, 0.3246, 0.2209, 0.3684, 0.0178), respectively. These verify the effectiveness of adjusting the weight to meet the similarity calculation principle. And the weight of different identification information is adaptive to the nature of image non-flat regions.



(a) (b) (c)

Fig. 13. (a) Non-flat regions of images that are not segmented, (b) segmented with $w_j = 0.2$, and (c) adaptive w.

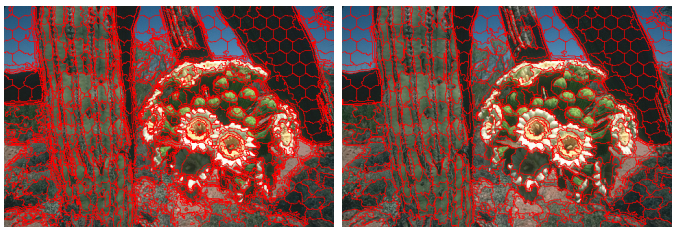
C. Seed Point Updating

The normalization of each identification information can produce irrational numbers, but the computer's accuracy is limited. In order to prevent the accumulation of errors caused by the normalization of each identification information of pixel in seed point calculation, for the un-normalized data, we calculate the mean value of 8-neighborhood contour information, spatial information, and color information of all pixels contained in a superpixel. Then these mean values are normalized as a new seed point. We do this for all superpixels. The seed points are updated globally on the image to avoid

a large number of seed points moving to the non-flat regions, which may reduce the regularity of the generated superpixels and affect the connectivity within superpixels.

D. Superpixel Combination

After the superpixel subdivision of the non-flat regions is completed, to improve the regularity of superpixels and eliminate some of the scattered pixels, we clean up these isolated tiny superpixels by iteratively merging them with adjacent superpixels that have the closest properties, based on the PriorityQueue implementation, and one result is shown in Fig. 14. Algorithm 2 summarizes the selection of seed points and the steps of superpixels generation. Algorithm 1 and Algorithm 2 are shown below.



(a) (b)

Fig. 14. (a) The result before the superpixel merge. (b) The result after the superpixel merge.

Algorithm 1 Contour information extraction and clustering

Require: Image I

Ensure: Class label $C_l(p)$ and contour information

RCF is used to obtain the edge image of image I and then perform non-maximum suppression on the edge image.
Complete part of the boundary points by setting the values of the local-regional of each boundary point to be the value of this boundary point.
Gaussian filtering is performed on image I in CIELAB space.
Calculate 4-neighborhood and 8-neighborhood contour information of pixel by Eq. (2) and Eq. (6), respectively.
Calculate neighborhood distance $d(p, p_m)$ between pixel p and its neighbor pixels p_m and generating similar pixels set $C(p)$.
Let $C_l(p)$ be the class label of pixel p.
for each pixel p **do**
  if pixel p and p_m are similar **then**
    $C_l(p_m) = C_l(p)$
  end if
  Extend the number of elements in class.
end for

VIII. Experiment

We compare our method with state-of-the-art methods in superpixel segmentation, including SLIC [16], ERS [21], SSBC [38], LSC [17], GMM [22], SH [25], TRS [18], SSN [30], WSS [20], ECCPD [39], WSGL [27], SSFCN [31], LNSnet [32], RSS [36], FGSLT [33], and AINet [34]. The

Algorithm 2 Seed selection and superpixel generation

Require: Image I, expected number of superpixels K, Class label $C_l(p)$, contour information

Ensure: Superpixel label $S_l(p)$ for image

Generate K hexagons on the clustering image.
Let $S_l(p)$ be the superpixel label of pixel p and $S_l(s)$ be the superpixel label of seed point s.
Set superpixel label $S_l(p) = 0$ for each pixel p.
Set superpixel label $S_l(s) = u$ for each seed point s, u is the order of the seed points.
Generate seed points s in each hexagon.
Generate superpixels in the flat regions.
Pixel unlabeled in the non-flat region is defined as pixel q.
Using five dimensions $[l, a, b, s, g]^T$ to represent q and s, normalize q and s.
Set the distance $D(q) = \infty$ for each pixel q.
Set weight parameters $w_j = 0.2, j \in \{l, a, b, s, g\}$.
Set the iteration number $k = 1$.
repeat
  for each seed point s **do**
    for each pixel q belonging to the circle search range of seed point s **do**
      Calculate the distance $D(q, s)$ between pixel q and seed point s.
      if $D(q, s) < D(q)$ **then**
        $D(q) = D(q, s)$
        $S_l(q) = S_l(s)$
      end if
    end for
  end for
  for each identification information j **do**
    Adapt the weight parameter w_j for identification information j by using Eq. (14).
  end for
  Update seed points.
  $k = k + 1$
until $k > 4$
Re-divide the superpixel label of all pixels.
Put each superpixel into the Priority Queue with the number of pixels contained in the superpixel as the priority.
Merge isolated superpixels iteratively.

results of these algorithms are obtained by running the source codes published online. Our implementation is in C++. All experiments are conducted on a personal computer with Intel Core i7 at 3.7 GHz and Nvidia graphics card GTX-1060.

A. Dataset

The standard Berkeley Segmentation Dataset (BSD500 [40]), the Fashionista Dataset (Fash [41]) and the PASCAL-S [42] are used for evaluating the effectiveness of our approach and performing comparisons. The BSD500 contains 200 training, 100 validation, and 200 test images. The Fash dataset contains 685 fashion clothing images with semantic ground truth segmentations. The PASCAL-S dataset contains 850

TABLE I
NUMERICAL RESULTS ON THE PASCAL-S DATASET WHEN THE NUMBER OF SUPERPIXELS IS 300.

	SSBC	WSS	LSC	ERS	SH	SLIC	GMM	TRS	RSS	ECCPD	WSGL	SSN	SSFCN	LNSnet	AINet	FGSLT	Our algorithm
Boundary recall (BR)	0.9056	0.9357	0.9329	0.9315	0.9466	0.8820	0.9315	0.8650	0.9386	0.7560	0.9476	0.8173	0.9081	0.8928	0.9040	0.9267	**0.9654**
Achievable segmentation accuracy (ASA)	0.9644	0.9703	0.9712	0.9685	0.9707	0.9625	0.9711	0.9639	0.9630	0.9568	0.9678	0.9511	0.9738	0.9629	0.9728	0.9670	**0.9758**
Under segmentation error (UE)	0.2216	0.1975	0.1957	0.1980	0.1902	0.2320	0.1900	0.2224	0.2313	0.2492	0.2106	0.2700	0.1692	0.2373	0.1751	0.2089	**0.1608**
Average time per image	1.1906	0.2104	0.3595	0.8550	0.0566	0.0715	0.1723	4.2835	**0.0259**	8.1812	0.0547	0.3294	0.0680	0.6215	0.0410	0.5672	0.1967

Bold indicates the best result

TABLE II
NUMERICAL RESULTS ON THE BSD500 DATASET WHEN THE NUMBER OF SUPERPIXELS IS 300.

	SSBC	WSS	LSC	ERS	SH	SLIC	GMM	TRS	RSS	ECCPD	WSGL	Our algorithm
Boundary recall (BR)	0.8802	0.9161	0.9086	0.9017	0.9305	0.8360	0.9159	0.8081	0.9083	0.6742	0.9365	**0.9382**
Achievable segmentation accuracy (ASA)	0.9497	0.9547	0.9559	0.9544	0.9559	0.9471	0.9582	0.9482	0.9428	0.9394	0.9533	**0.9596**
Under segmentation error (UE)	0.2434	0.2226	0.2212	0.2189	0.2186	0.2566	0.2035	0.2445	0.2795	0.2752	0.2326	**0.1947**
Average time per image	1.0942	0.2050	0.3008	0.6002	0.0385	0.0621	0.1603	4.2926	**0.0213**	5.4120	0.0560	0.1751

Bold indicates the best result

images of various object categories and sizes. Three datasets are assigned with human-annotated ground-truth labels. In addition, we used the images in the DIV2K [43] dataset to verify the time complexity of our algorithm.

B. Evaluation Metrics

We use standard superpixel evaluation metrics, including Boundary Recall (BR), Under-segmentation Error (UE), Achievable Segmentation Accuracy (ASA), and Explained Variation (EV). Let the standard segmentation region of an image be $G_i(i = 1, 2, \cdots, M)$ and $S_l(l = 1, 2, \cdots, K)$ represents each superpixel. Each pixel is designated p, with $1 \leq n \leq N$ (N is the number of pixels of the image I).

1) Boundary Recall: BR is used for evaluating superpixel boundary adherence. It calculates the percentage of the artificially labeled boundaries that fall between at least two superpixel boundary pixels. High BR means strong edge preservation.

2) Achievable Segmentation Accuracy: ASA is used to determine whether an object in the image is correctly recognized. The highest recognition rate is calculated by labeling each superpixel with a standard segmentation that has the largest overlap area. The larger the ASA value is, the more objects are correctly identified. We compute our method's ASA value by averaging the ASA values of all the images in the dataset.

$$ASA = \frac{\sum_l \arg_i \max |S_l \cap G_i|}{\sum_i |G_i|} \tag{16}$$

3) Under-segmentation Error: Based on the requirement that each superpixel belongs to only one object, UE measures the percentage of pixels inside the superpixel that is leaked from the ground truth segmentation. If a superpixel effectively overlaps with more than one true contour, the UE will increase accordingly. The UE of the entire image is calculated as follows:

$$UE = \frac{1}{N}\left[\sum_{i=1}^{M}\left(\sum_{\{S_l||S_l - G_i|>B\}} Area(S_l)\right) - N\right] \tag{17}$$

Where, N is the number of all pixels, $Area(S_l)$ is the area of superpixel S_l, and B is the area of the overlap region of the minimum number. We set B to be 5% of $Area(S_l)$. A smaller UE value means that more objects are identified in an image.

4) Explained Variation: The explained variation (EV) [44] quantifies the color variation in the superpixels. Since image boundaries tend to show great changes in color and structure, EV assesses boundary adherence independent of human annotations. EV is defined as

$$\mathrm{EV} = \frac{\sum_{S_l} |S_l| \left(\mu\left(S_l\right) - \mu(I)\right)^2}{\sum_{p_n} \left(p_n - \mu(I)\right)^2} \tag{18}$$

Where $\mu(S_l)$ and $\mu(I)$ are the mean colors of superpixel S_l and image I, respectively. The higher the value is, the better the quality is.

C. Numerical Comparisons

1) Results on PASCAL-S: As shown in Fig. 15, on the PASCAL-S dataset, the BR of our algorithm is better than other algorithms. When the number of superpixels is between 100 and 500, our algorithm is the best among all the algorithms in terms of the ASA and UE metrics. As the number of superpixels increases, the UE and ASA of our algorithm are slightly weaker than SSFCN but better than other algorithms and our BR and EV are better than SSFCN. Note that our computation efficiency is competitive compared with WSS, SLIC, LSC, GMM, LNSnet, and SSFCN, and much faster than ERS, WSS, LSC, LNSnet, SSN and FGSLT, as shown in Fig. 19.

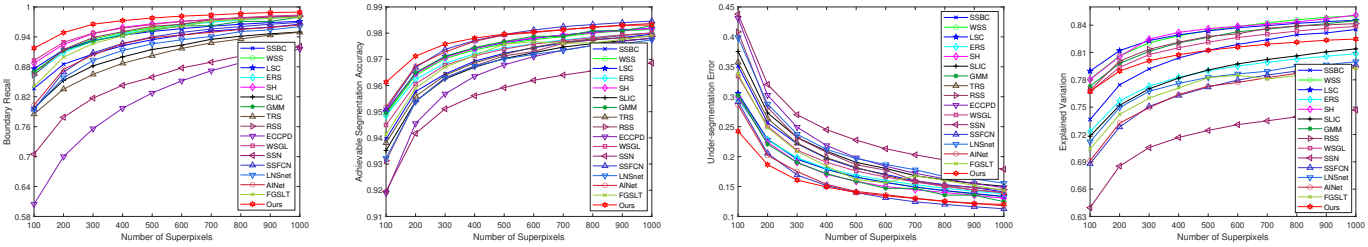
2) Results on BSDS500: BSDS500 contains 200 training, 100 validation, and 200 test images. We run the SSFCN, SSN, LSNet, FGSLT and AINet algorithms on the test images. For other algorithms, we run them on the entire dataset. As shown in Fig. 16, on the entire dataset of BSD500, when the number of superpixels is between 100 and 800, our algorithm is the best among all the algorithms, in terms of the BR, ASA, and UE metrics. Similarly, our method achieved the best results on the test images under EV and BR metrics, as shown in Fig. 17(a) and 17(d).

3) Results on FASH: The numerical results of our algorithm on the FASH dataset are shown in Fig. 18. When the number of superpixels is between 100 and 400, our algorithm is the best among all the algorithms, in terms of the BR and ASA metrics. As the number of superpixels increases, the BR of our algorithm is the same as the algorithms SH and is better than other algorithms. Similarly, when the number of superpixels

TABLE III
NUMERICAL RESULTS ON THE FASH DATASET WHEN THE NUMBER OF SUPERPIXELS IS 300.

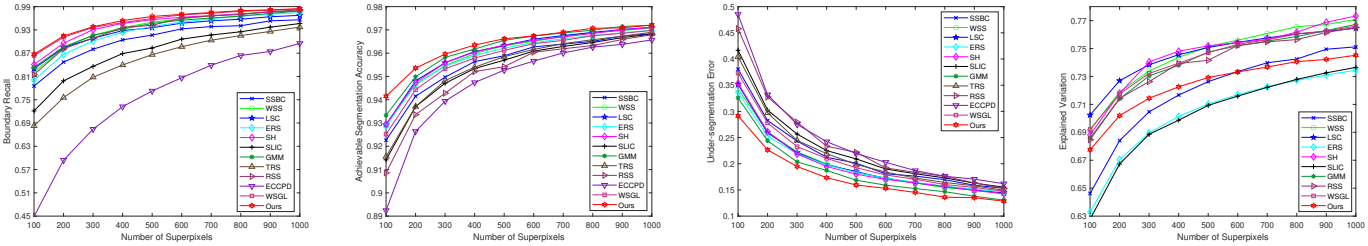
	SSBC	WSS	LSC	ERS	SH	SLIC	GMM	TRS	RSS	ECCPD	WSGL	SSN	SSFCN	LNSnet	AINet	FGSLT	Our algorithm
Boundary recall (BR)	0.9391	0.9586	0.9647	0.9495	0.9659	0.9234	0.9508	0.9006	0.9576	0.6581	0.9570	0.7713	0.8933	0.9287	0.8744	0.9524	**0.9720**
Achievable segmentation accuracy (ASA)	0.9740	0.9760	0.9780	0.9762	0.9767	0.9728	0.9756	0.9717	0.9685	0.9545	0.9724	0.9543	0.9747	0.9663	0.9718	0.9751	**0.9789**
Under segmentation error (UE)	0.1289	0.1209	**0.1086**	0.1129	0.1151	0.1344	0.1214	0.1333	0.1557	0.1745	0.1381	0.2071	0.1189	0.1287	0.1305	0.1221	0.1109
Average time per image	1.5858	0.2740	0.4628	1.0315	0.0603	0.0975	0.2159	6.0202	0.0310	11.912	0.1375	0.4043	0.0510	0.9511	**0.0260**	0.7662	0.2610

Bold indicates the best result



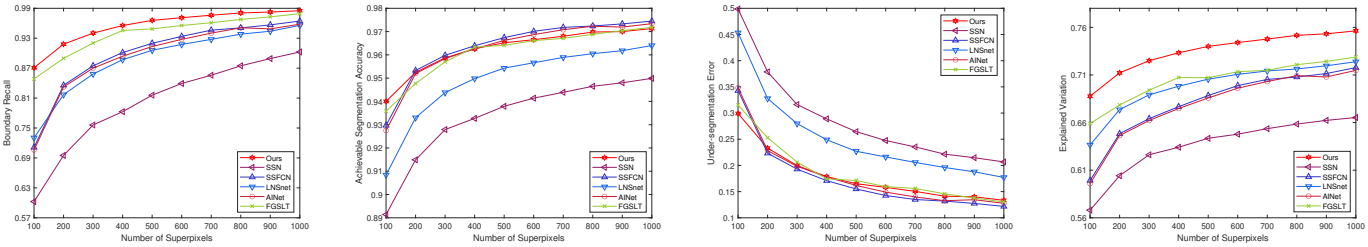
(a) BR curve (b) ASA curve (c) UE curve (d) EV curve

Fig. 15. The results of numerical comparison of different algorithms on the PASCAL-S dataset.



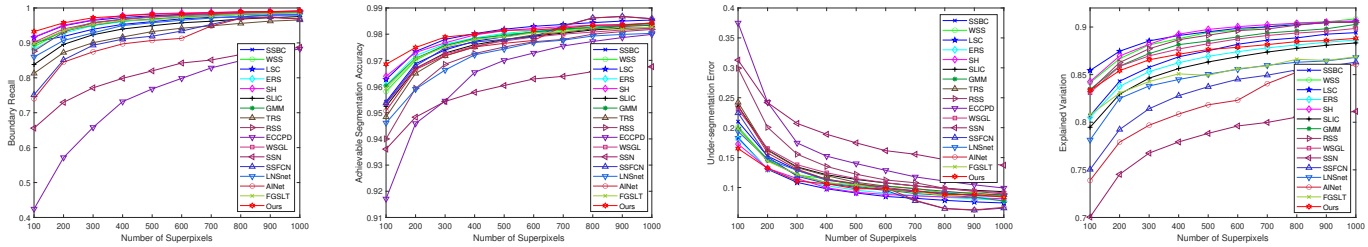
(a) BR curve (b) ASA curve (c) UE curve (d) EV curve

Fig. 16. The results of numerical comparison of different algorithms on the BSD500 dataset.



(a) BR curve (b) ASA curve (c) UE curve (d) EV curve

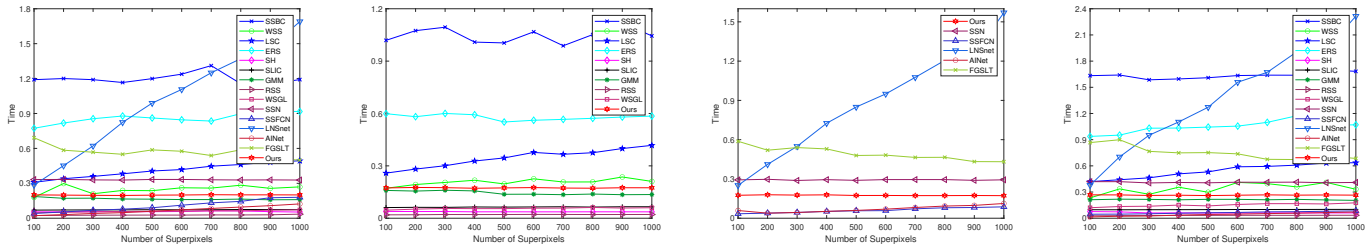
Fig. 17. The results of numerical comparison of different algorithms on the test images (BSD500 dataset).



(a) BR curve (b) ASA curve (c) UE curve (d) EV curve

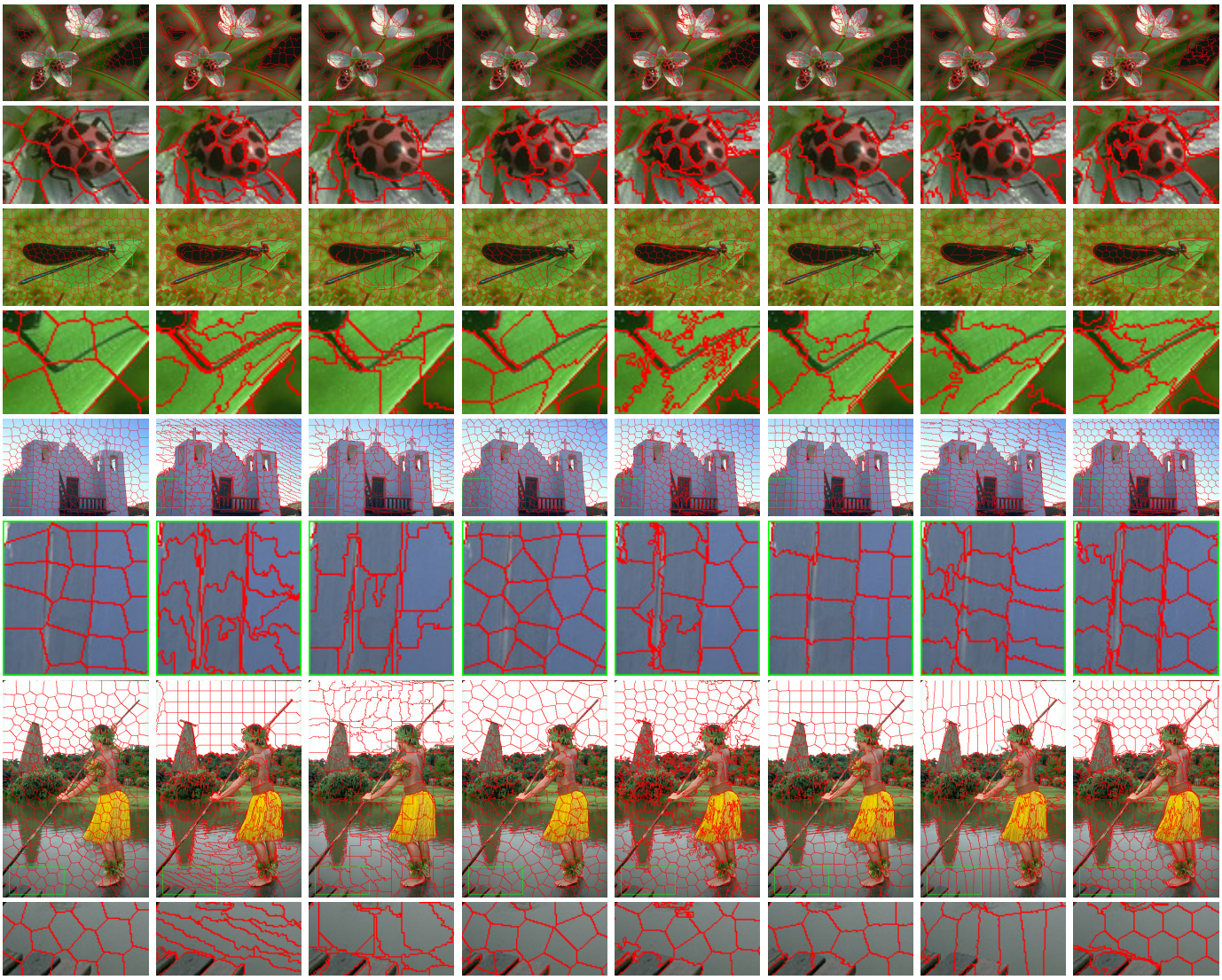
Fig. 18. The results of numerical comparison of different algorithms on the FASH dataset.

is between 100 and 200, the UE of our algorithm is better than other algorithms. When the superpixel is between 200 and 500, our UE is slightly weaker than ERS, SH, and LSC, and better than other algorithms.



(a) PASCAL-S dataset (b) BSDS500 dataset (c) test images (BSD500 dataset) (d) FASH dataset

Fig. 19. The time of different algorithms.



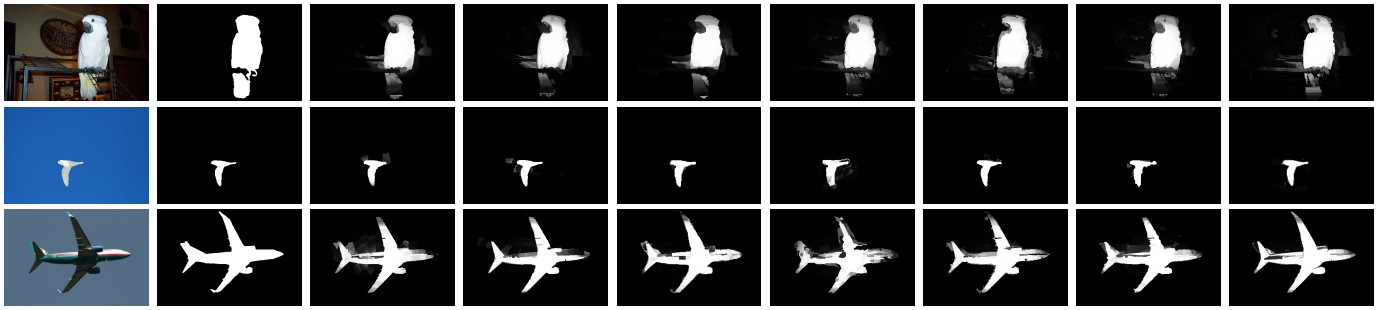
(a) ECCPD (b) RSS (c) SSFCN (d) TRS (e) WSGL (f) SLIC (g) LNSnet (h) Ours

Fig. 20. Visualization of different superpixel segmentation algorithms when the number of superpixels is 300.

To summarize briefly from the Fig. 15, 16 and 18, we claim that our algorithm can achieve SOTA performance in terms of the balance between the shape regularity and the boundary adherence, especially when the superpixel number is small. When superpixels number are large, many algorithms can work well, since they can have enough freedom to balance the regularity and boundary adherence. However, when the number is small, those methods will have an obvious performance drop. Our algorithm has obviously better balance instead, due to its divide-and-conquer processing and self-adaptive parameters.

D. Visual Comparisons

Fig. 20 shows more representative results from ECCPD, RSS, SSFCN, TRS, WSGL, SLIC, LNSnet, and our algorithm.



(a) Original (b) GT (c) SLIC (d) RSS (e) SSFCN (f) SSN (g) WSGL (h) LNSnet (i) Ours

Fig. 21. Examples of the saliency detection on the PASCAL-S database by using different superpixel segmentation algorithms for pre-processing. Our algorithm consistently generates saliency maps close to the ground truth.

TABLE IV
QUANTITATIVE RESULTS ON FIG. 20 WHEN THE NUMBER OF SUPERPIXELS IS 300.

	ECCPD	RSS	SSFCN	TRS	WSGL	SLIC	LNSnet	Our
BR	0.7217	0.9456	0.9072	0.8595	0.9424	0.8804	0.8841	**0.9618**
ASA	0.9433	0.9601	0.9644	0.9573	0.9611	0.9597	0.9512	**0.9667**
UE	0.2717	0.2175	0.1778	0.2081	0.2011	0.2111	0.2407	**0.1723**
EV	———	**0.6493**	0.6217	———	0.6480	0.6409	0.6277	0.6403

Bold indicates the best result

Compared with other algorithms, our method generates the best results in boundary adherence and is good at handling weak boundaries and small objects. This is because we use the pixels' contour information for calculating parameters of superpixel generation for non-flat regions of the image and adaptively adjusting the parameters to fit the image contents. Our algorithm produces regular and compact superpixels. The SSFCN and RSS adhere well to the boundary of the image, but the regularity of their superpixels is not good enough. WSGL, ECCPD, SLIC, and TRS can generate superpixels with regular shapes. In addition, we also made five visual comparison GIF files with the number of superpixels from 100 to 1000 in the Multimedia folder. Finally, we calculated the BR, ASA, UE, and EV of each image in Fig. 20, and then averaged these values across all images as the final numerical result of the algorithm. The quantitative results are shown in Table IV. The results in Table IV agree with those reported in VIII-C.

E. Saliency Detection

The task of saliency detection is to identify the most important and informative part of a scene. To illustrate the effectiveness of our method on real vision tasks, we employ a saliency detection application [45], which adopts superpixel segmentation as a pre-processing step. We replaced the originally used superpixel segmentation (i.e. SLIC) with SSN, RSS, WSGL, SSFCN, LNSnet, and our method, respectively. Using the default parameter settings in [45], on PASCAL-S [42] dataset (a well-known dataset for saliency detection), our algorithm consistently highlights salient regions and preserves better object boundaries than other methods, as shown in Fig. 21. Meanwhile, precision and recall are used to evaluate the mentioned superpixel algorithms. The precision value is the ratio of salient pixels correctly assigned to all the pixels of extracted regions, while the recall value corresponds to the percentage of detected salient pixels in relation to the ground-truth number. We normalize the saliency map obtained by the mentioned superpixel methods from 0 to 255 and use G as the binarized ground truth. Then, we utilize the fixed threshold changing among 0∼255 to binary the saliency map to get the binarized map S. For each fixed threshold among 0∼255, a pair of precision/recall values will be obtained by formula (19) to form the PR curves.

$$Precision = \frac{|S \cap G|}{|S|}, Recall = \frac{|S \cap G|}{|G|} \tag{19}$$

Where $|*|$ denotes the non-zero input value for a binary image. Standard precision-recall curves are shown in Fig. 22. When the recall is between 0.4 and 1, our precision is better than other algorithms.

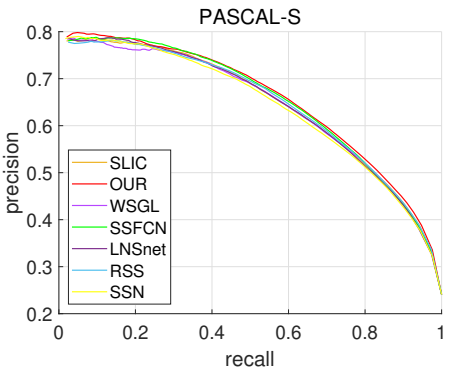


Fig. 22. Precision-recall curves.

F. Time Complexity Analysis

Suppose an image contains N pixels, and it is desired to generate K superpixels. The complexity of computing the contour information and the neighborhood distance is $O(N)$, respectively. The time complexity of clustering is $O(N)$. The time complexity of calculating the seed point is $O(K * S_n)$, where S_n is the average number of pixels to calculate in each hexagon. The time complexity of generating superpixels in the flat regions is $O(N)$. The time complexity of each iteration is $O(K * S_f)$, where S_f is the average number of pixels to

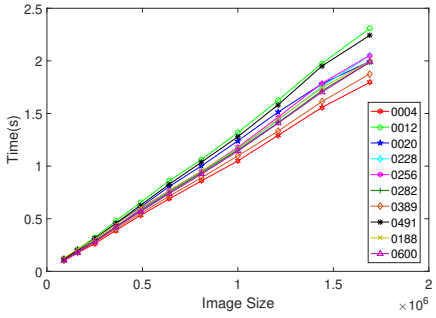


Fig. 23. The average running time of our algorithm on ten images where we sample each image with an increased resolution, i.e. 300*300, 400*400... 1300 * 1300. The "Image size" is the number of pixels contained in the current image.

be calculated for each seed point. Updating the seed points' time complexity is $O(N)$. The time complexity of adjusting the weights is $O(n_f)$, and n_f is the number of pixels in non-flat regions. Then, the time complexity of the step of merging small superpixels is $O(zn)$, where z represents the total number of small superpixels that need to be merged during the entire merging process, and n is the average number of superpixels adjacent to them. So, the time complexity of the algorithm is $O((4+d)N+K*S_n+d*K*S_f+d*n_f+zn)$, where d is the number of iterations. Since (zn) is much smaller than $O((4+d)N+K*S_n+d*K*S_f+d*n_f)$, when considering the worst case, e.g. $n_f = N$, $K*S_n = N$, $K*S_f = 6.25N$, the time complexity of our algorithm is $O((5+8.25d)N)$. So the time complexity of our algorithm is $O(N)$. For further verification, ten images in the DIV2K dataset are used to evaluate our algorithm's time complexity, including landscapes, people, buildings, animals, and vehicles. We ran our method on the same pictures of different sizes, and the running time of the algorithm is shown in Fig. 23. It can be seen from Fig. 23 and Fig. 19 that the running time of our algorithm only increases linearly with the increase of the image size.

IX. Conclusion

In this paper, we propose a high-quality superpixel generation through regional decomposition. The core idea is that we separate the input image into flat and non-flat regions, where different segmentation strategies were adopted to meet the balance between shape regularity and boundary adherence. Compared with state-of-the-art methods, our algorithm consistently generates more regular superpixels with stronger boundary adherence while maintaining a competitive efficiency. Although we have achieved higher numerical performance, superpixels generated by our algorithm are sometimes still not very regular in some boundary regions. In the future, we consider using a new non-flat regions superpixel generation method to further improve the superpixel regularity for those boundary regions while maintaining strong boundary adherence. For example, further adaptively divide non-flat regions, and adopt different strategies for different regions. We also want to consider more elegant ways to extract high-quality superpixels for the videos with intensive sharp changes between frames.

References

[1] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 1, Oct. 2003, pp. 10–17.
[2] Y. Liang, J. Shen, X. Dong, H. Sun, and X. Li, "Video supervoxels using partially absorbing random walks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 5, pp. 928–938, 2016.
[3] J. Shen, J. Peng, and L. Shao, "Submodular trajectories for better motion segmentation in videos," *IEEE Transactions on Image Processing*, vol. 27, no. 6, pp. 2688–2700, 2018.
[4] X. Dong, J. Shen, L. Shao, and L. Van Gool, "Sub-markov random walk for image segmentation," *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 516–527, 2016.
[5] F. Yang, H. Lu, and M.-H. Yang, "Robust superpixel tracking," *IEEE Transactions on Image Processing*, vol. 23, no. 4, pp. 1639–1651, 2014.
[6] Y. Yuan, J. Fang, and Q. Wang, "Robust superpixel tracking via depth fusion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 15–26, 2014.
[7] W. Wang, C. Wang, S. Liu, T. Zhang, and X. Cao, "Robust target tracking by online random forests and superpixels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 7, pp. 1609–1622, 2017.
[8] G. Mori, X. Ren, A. A. Efros, and J. Malik, "Recovering human body configurations: Combining segmentation and recognition," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2. Citeseer, 2004, pp. II–II.
[9] T. Cour and J. Shi, "Recognizing objects by piecing together the segmentation puzzle," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2007, pp. 1–8.
[10] J. Cheng, J. Liu, Y. Xu, F. Yin, D. W. K. Wong, N.-M. Tan, D. Tao, C.-Y. Cheng, T. Aung, and T. Y. Wong, "Superpixel classification based optic disc and optic cup segmentation for glaucoma screening," *IEEE Trans. on Medical Imaging*, vol. 32, no. 6, pp. 1019–1032, 2013.
[11] B. Liu, H. Hu, H. Wang, K. Wang, X. Liu, and W. Yu, "Superpixel-based classification with an adaptive number of classes for polarimetric sar images," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 907–924, 2013.
[12] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung, "Saliency filters: Contrast based filtering for salient region detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2012, pp. 733–740.
[13] W. Wang, J. Shen, and F. Porikli, "Saliency-aware geodesic video object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2015, pp. 3395–3402.
[14] Z. Liu, X. Zhang, S. Luo, and O. Le Meur, "Superpixel-based spatiotemporal saliency detection," *IEEE transactions on circuits and systems for video technology*, vol. 24, no. 9, pp. 1522–1540, 2014.
[15] L. Li, S. Zhang, X. Yu, and L. Zhang, "Pmsc: Patchmatch-based superpixel cut for accurate stereo matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 3, pp. 679–692, 2018.
[16] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.
[17] J. Chen, Z. Li, and B. Huang, "Linear spectral clustering superpixel," *IEEE Trans. on Image Processing*, vol. 26, no. 7, pp. 3317–3330, 2017.
[18] X. Pan, Y. Zhou, Z. Chen, and C. Zhang, "Texture relative superpixel generation with adaptive parameters," *IEEE Transactions on Multimedia*, pp. 1–1, 2019.
[19] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real-time superpixel segmentation by dbscan clustering algorithm," *IEEE Trans. on Image Processing*, vol. 25, no. 12, pp. 5933–5942, 2016.
[20] X. Qian, X. Li, and C. Zhang, "Weighted superpixel segmentation," *The Visual Computer*, vol. 35, no. 11, 2019.
[21] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2011, pp. 2097–2104.
[22] Z. Ban, J. Liu, and L. Cao, "Superpixel segmentation using gaussian mixture model," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4105–4117, 2018.
[23] J. Peng, J. Shen, A. Yao, and X. Li, "Superpixel optimization using higher order energy," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 5, pp. 917–927, 2016.
[24] H. Wang, J. Shen, J. Yin, X. Dong, H. Sun, and L. Shao, "Adaptive nonlocal random walks for image superpixel segmentation," *IEEE Transactions on Circuits & Systems for Video Technology*, pp. 1–1, 2019.

[25] X. Wei, Q. Yang, Y. Gong, N. Ahuja, and M. H. Yang, "Superpixel hierarchy," *IEEE Transactions on Image Processing*, pp. 1–1, 2018.

[26] X. Dong, J. Shen, and L. Shao, "Hierarchical superpixel-to-pixel dense matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2518–2526, 2017.

[27] Y. Yuan, Z. Zhu, H. Yu, and W. Zhang, "Watershed-based superpixels with global and local boundary marching," *IEEE Transactions on Image Processing*, vol. 29, pp. 7375–7388, 2020.

[28] L. Zhang, S. Lu, C. Hu, D. Xiang, T. Liu, and Y. Su, "Superpixel generation for sar imagery based on fast dbscan clustering with edge penalty," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 804–819, 2022.

[29] W. Jing, T. Jin, and D. Xiang, "Content-sensitive superpixel generation for sar images with edge penalty and contraction–expansion search strategy," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.

[30] V. Jampani, D. Sun, M.-Y. Liu, M.-H. Yang, and J. Kautz, "Superpixel sampling networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[31] F. Yang, Q. Sun, H. Jin, and Z. Zhou, "Superpixel segmentation with fully convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2020*, 2020.

[32] L. Zhu, Q. She, B. Zhang, Y. Lu, Z. Lu, D. Li, and J. Hu, "Learning the superpixel in a non-iterative and lifelong manner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1225–1234.

[33] X. Pan, Y. Zhou, Y. Zhang, and C. Zhang, "Fast generation of superpixels with lattice topology," *IEEE Transactions on Image Processing*, vol. 31, pp. 4828–4841, 2022.

[34] Y. Wang, Y. Wei, X. Qian, L. Zhu, and Y. Yang, "Ainet: Association implantation for superpixel segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7078–7087.

[35] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[36] D. Chai, "Rooted spanning superpixels," *International Journal of Computer Vision*, vol. 128, 12 2020.

[37] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3000–3009.

[38] Y. Zhang, X. Li, X. Gao, and C. Zhang, "A simple algorithm of superpixel segmentation with boundary constraint," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 7, pp. 1502–1514, 2016.

[39] D. Ma, Y. Zhou, S. Xin, and W. Wang, "Convex and compact superpixels by edge-constrained centroidal power diagram," *IEEE Transactions on Image Processing*, vol. PP, pp. 1–1, 12 2020.

[40] 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 *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, 2001, pp. 416–423.

[41] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg, "Parsing clothing in fashion photographs," in *2012 IEEE Conference on Computer vision and pattern recognition*. IEEE, 2012, pp. 3570–3577.

[42] Y. Li, X. Hou, C. Koch, J. Rehg, and A. Yuille, "The secrets of salient object segmentation," in *CVPR, 2014, IEEE Conference on*, 2014.

[43] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 126–135.

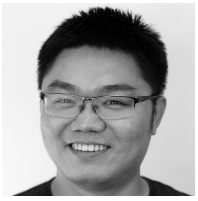
[44] A. P. Moore, S. J. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[45] X. Li, H. Lu, L. Zhang, X. Ruan, and M.-H. Yang, "Saliency detection via dense and sparse reconstruction," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2976–2983.

Yunyang Xu received a B.E. degree from the Department of Information Engineering, Hefei University of Technology, Xuancheng, China, in 2017. He is currently working toward his M.E. degree in the School of Software, Shandong University, Jinan, China. His research interests include image segmentation and computer graphics.

Dr. Xifeng Gao is an assistant professor of the Computer Science Department at Florida State University. Dr. Gao was a PostDoc for two years at the Courant Institute of Mathematical Sciences of New York University. He received his Ph.D. degree in 2016 and won the best Ph.D. dissertation award from the Department of Computer Science at the University of Houston. Dr. Gao has wide research interests that are related to geometry, such as Computer Graphics, Visualization, Multimedia Processing, Robotics, and Digital Fabrication.

Caiming Zhang received B.S. and M.S. degrees in computer science from Shandong University, Jinan, China, in 1982 and 1984, respectively, and a Ph.D. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 1994. From 1998 to 1999, he was a Post-Doctoral Fellow with the University of Kentucky, Lexington, USA. He is currently a Professor with the School of Software, Shandong University, and a Distinguished Professor with the School of Computer Science and Technology, Shandong University of Finance and Economics. His research interests include CAGD, Information Visualization and Medical Image Processing.



Jianchao Tan is currently a Staff Research Scientist in Kuaishou Technology, working for Automated Machine Learning, Computer Graphics and Vision. He obtained Ph.D. degree from the Computer Science Department at George Mason University in 2019 and B.S. degree in 2013 from the Electronic Engineering and Information Science Department at the University of Science and Technology of China.

Xuemei Li received the Master's and Doctor's degrees from Shandong University, Jinan, China, in 2004 and 2010, respectively. From 2013 to 2014, she was a visiting scholar at the University of Houston, USA. She is currently an associate professor in the School of Software, Shandong University, and a member of the GD&IV lab. She is engaged in research on Geometric Modeling, CAGD, Medical Image Processing and Information Visualization.