EUROGRAPHICS 2020 / E. Eisemann, A. Jacobson, and F.-L Zhang
(Guest Editors)

Volume 39 (2020), Number 7

Generating High-quality Superpixels in Textured Images

Paper ID 1163

Abstract

Superpixel segmentation is important for promoting various image processing tasks. However, existing methods still have difficulties in generating high-quality superpixels in textured images, because they cannot separate textures from structures well. Though texture filtering can be adopted for smoothing textures before superpixel segmentation, the filtering would also smooth the object boundaries, and thus weaken the quality of generated superpixels. In this paper, we propose to use the adaptive scale box smoothing instead of the texture filtering to obtain more high-quality texture and boundary information. Furthermore, we design a novel distance metric to measure the distance between different pixels, which considers boundary, color and Euclidean distance simultaneously. As a result, our method can achieve high-quality superpixel segmentation in textured images without texture filtering. The experimental results demonstrate the superiority of our method over existing methods, even the learning-based methods. Benefited from using boundaries to guide superpixel segmentation, our method can also suppress noise to generate high-quality superpixels in non-textured images.

CCS Concepts

• ***Computing methodologies*** → ***Image processing; Texturing;*** *Image segmentation;*

1. Introduction

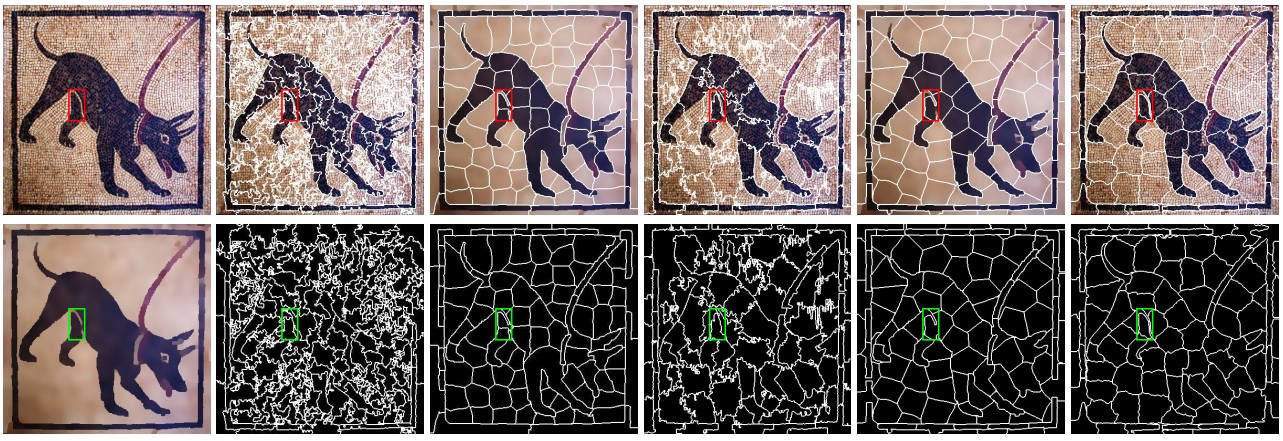
Superpixel segmentation can effectively capture image features and greatly reduce the complexity of subsequent image processing, which plays an important role in many applications, such as image segmentation [RM03, PAB*17, CLH*16, RS13, LJP*18], object tracking [WLYY11], salient object detection [YZL*13], 3D reconstruction [MK10], contour detection [MPAVG18, AMFM11] and edge detection [LCH*17]. Thus, many methods have been proposed for superpixel segmentation, including graph-based methods [SM00,FH04,LTRC11], clustering-based methods [DBBR*12, ASS*12, CLH17, AS17, LYYH16, PZCZ19], and learning-based methods [JSL*18, TLJ*18, URF19].

Though much progress has been achieved, the existing traditional methods [LTRC11, ASS*12] are mainly relied on the color difference between pixels to derive edge information and perform superpixel segmentation. This prevents them from obtaining high-quality results in textured images since textures may also contain edge information, as illustrated in Fig.1 (b) and (d), the resulting superpixels from existing methods are cluttered, and their edges significantly deviate from the boundaries between different texture regions. The learning-based methods [JSL*18, TLJ*18] usually require a large amount of labeled data to train the model and the training is time-consuming. Besides, the learning method cannot yield compact superpixels in textured images, though they can obtain boundary-preserved superpixels in non-textured images, as shown in Fig. 6.

A possible way is to smooth the textures before superpixel segmentation. Unfortunately, texture filtering is always expensive [CLKL14]. Even worse, texture smoothing is done by averaging the color of the pixels in a region, which would unavoidably decrease the contrast around the boundaries. Hence, the boundaries between different textured regions are difficult to detect, thus preventing generating boundary-preserved superpixels, as illustrated in Fig. 1 (c), (e), where some of the boundaries between textured regions are broken.

In this paper, we address this challenge by developing a novel algorithm to generate uniform and boundary-preserved superpixels in textured images. Specifically, we forgo texture filtering and instead estimate the probability of pixels on boundaries using the adaptive scale box. In this way, the textures are well-considered while the shortcomings of employing texture filtering are avoided. Besides, we developed a novel distance measure that utilizes the boundary probability information, color and Euclidean distance at the same time. As illustrated in Fig. 1 (f), our produced superpixels are compact and well boundary-preserved, while this cannot be achieved with existing methods, even when they use the filtered images as input. In addition, our approach can run quickly, as shown in Fig. 9. When the superpixels are required in different numbers, our method is also able to produce high-quality results, as illustrated in Fig. 2.

In the experiments, we compared our method with ten state-of-the-art methods, including a graph-based method [LTRC11], three learning-based methods [JSL*18, TLJ*18, URF19] and six



(a) Input and filtered image Time: BTF takes 3s | (b) ERS Time: 2.5s | (c) ERS on filtered image Time: 5.5s | (d) SLIC Time: 0.22s | (e) SLIC on filtered image Time: 3.22s | (f) Ours Time: 0.52s

Figure 1: *Comparison between our method and some existing methods for the superpixel segmentation in a textured image (100 superpixels). Our results are compact and the boundaries are well preserved, while those of other methods are not, as shown in the enlarged red/green rectangles. (a) The input image (top) and its filtered one (bottom) using BTF [CLKL14] with 4 iterations. (b) The resulting superpixels by applying ERS [LTRC11] to the input image, which are cluttered and the boundaries are not preserved. (c) The resulting superpixels by applying ERS to the filtered image, which contain some broken boundaries. (d) The resulting superpixels by applying SLIC [ASS*12] to the input image, which are not compact and do not have well-preserved boundaries. (e) The resulting superpixels by applying SLIC to the filtered image, which have some broken boundaries. (f) Our results from the input image are compact and have well-preserved boundaries.*

clustering-based methods [DBBR*12, ASS*12, CLH17, AS17, LYYH16, GTP18]. The results show that our method can considerably promote superpixel segmentation in textured images. Besides, our method can also obtain high-quality superpixels in non-textured images, since our algorithm can treat the noises and details as textures to suppress their interferences on superpixel segmentation. The experiments demonstrate that our method can achieve comparable efficiency to the state-of-the-art methods.

Our contributions are summarized as follows:

- We propose an adaptive scale box that can measure the texture/boundary probability information and reduce the influence of textures in superpixel segmentation.
- We design a novel boundary metric to measure the distance between different pixels, which makes use of boundary information, color and Euclidean distance simultaneously. Therefore, our method can generate boundary-preserved superpixels in textured images.
- We compare the proposed method with ten approaches on two datasets. The experimental results demonstrate that the proposed method can achieve the significant performance under different evaluation metrics.

The rest of this paper is organized as follows. A brief discussion of the related work is given in Section 2. Then, our boundary estimation and distance measure are described in Section 3 and 4, respectively. The detailed algorithm for superpixel segmentation is presented in Section 5. Afterwards, the experimental results are discussed in Section 6. Finally, the conclusion is drawn in Section 7.

Figure 2: *Images are segmented into 100/200/400 high-quality superpixels using our proposed algorithm.*

2. Related Work

2.1. Superpixel Segmentation

Numerous superpixel segmentation methods have been developed. They are mostly based on graphs or clustering techniques. Re-

cently, learning-based methods were studied and have achieved significant progress. They are discussed in the following.

Graph-based algorithms consider an image as a planar graph and partition the graph into subgraphs to generate the superpixels. Here, the pixels are regarded as graph nodes and they are connected by the affinities between them. Until now, various partition methods using merging or splitting techniques have been proposed. For example, the NCUTS algorithm [SM00] recursively computes normalized cuts on a graph for partitioning, and the bottom-up method (FH) [FH04] progressively merges sets of pixels based on the internal variation of the pixel affinities. To preserve the boundaries in superpixel segmentation well, the ERS method [LTRC11] merges disjointed sets of pixels by maximizing the entropy rate of the pixel affinities. However, the irregular shape of its superpixels may become a potential drawback in the following tasks [YZL*13]. In general, graph-based methods are always time-consuming and their superpixels are not very uniform, as their splitting or merging techniques are not easy to execute on graphical structures.

Clustering-based algorithms are more preferred due to their efficiency and easy implementation. They progressively refine an initial clustering of pixels until they converge. The initial clustering is always performed by evenly distributing the seeds in the image, which is then refined by using various clustering techniques. Some popular methods are introduced as follows. The SEEDS [DBBR*12] algorithm uses uniform blocks as the initial approximation of superpixels and iteratively exchanges the neighboring blocks in a coarse-to-fine manner. However, this approach suffers from shape irregularity and has difficulties in controlling the number of superpixels. The SLIC [ASS*12] method places the initial cluster centers on a uniform grid and performs k-means clustering in the five-dimensional space of the CIELab color and position feature space to achieve fast superpixel segmentation. The LSC [CLH17] method uses a kernel function to map pixels to a high dimensional feature space to capture perceptually important global image properties, which allow it to produce compact and regular shaped superpixels with linear time complexity and high memory efficiency. The MSLIC [LYYH16] maps an image to a 2-dimensional manifold space to more effectively capture the boundary information and improve superpixel segmentation. The SNIC [AS17] proposed a non-iterative clustering method using progressive propagation from seeds in a synchronous manner. Though much progress has been made with these methods, they cannot handle textured images since they do not consider the effects of textures on superpixel segmentation.

Learning-based algorithms have been recently proposed for superpixel segmentation. The SEAL-ERS [TLJ*18] proposed a segmentation-aware affinity learning approach to enhance boundary detection and promote graph-based superpixel segmentation. Based on the SLIC method, The SSN [JSL*18] proposed a differentiable algorithm and construct an end-to-end learning system for superpixel segmentation. The BASS [URF19] proposed a bayesian nonparametric mixture model for superpixel segmentation, which also respects topology and favors spatial coherence. In learning-based methods, there is the challenge of collecting sufficient labeled images. Otherwise, high-quality results cannot be achieved. In addition, the learning-based methods always rely on the color differences in pixels, which would focus too much on the details and so ineffective at handling textures. Thus, their resulting superpixels are not regular and they are not effective at handling textured images, which will be discussed in Section 6.

There are several other approaches such as the watershed transform [MFCP*15] and geometric flows [LSK*09]. Interested readers can refer to a recent survey paper [SHL18]. To the best of our knowledge, there is a lack of existing studies on the effects of textures on superpixel segmentation. This prevents these approaches from preserving boundaries between textured regions, which impair superpixel segmentation.

2.2. Texture Filtering

Texture filtering is designed to smooth textures while preserving the prominent structures of an image. For texture measurements, Xu et al. [XYXJ12] offered a relative total variation measure within a window for effectively distinguishing textures and structures. However, textures of various scales can coexist in an image, and small structures may be mistaken as textures within a large window. Thus, many methods have been proposed to promote texture measurement to improve texture filtering. Cho et al. [CLKL14] suggested patch-shifting to have the window placed inside textured regions as much as possible. Lin et al. [LWS*16] used smaller patches to preserve pixels at structured edges to avoid overly smoothing small structured edges while using larger patches to smooth the texture regions. Jeon et al. [JLKL16] presented a scale-aware method for finding the optimal per-pixel smoothing scales to filter textures adaptively. Xu et al. [XW18] provided long and narrow edge-aware windows for adaptive texture measurements to preserve small structures. Unfortunately, texture filtering methods unavoidably smooth boundaries, thereby preventing boundary detection and preservation in superpixel segmentation, as illustrated in Fig. 1 (c) and (e). In addition, texture filtering methods are generally time-consuming because of their iterative smoothing.

Based on these works for texture measurement and detection, we developed a novel measure to effectively detect the boundaries between textured regions. With the estimated boundary information, our method can achieve high-quality superpixel segmentation.

3. Boundary Estimation

To well estimate the boundaries between different textured regions, we need to measure textures to know their respective covered regions. As we know, existing texture filtering methods have made many efforts to this target. Unfortunately, due to the various scales of textures and the small structures in an image, high-quality estimates of the boundaries is still a difficult problem. As illustrated in Fig. 3 (b) and (f), the method [CLKL14] missed or merged some boundaries into small structures.

Therefore, we present a measure for well detecting multi-scale textures and preserving small structures, thereby improving boundary estimation. This algorithm includes the following steps:

1. Texture measurement with a large box is applied to each pixel in the image.

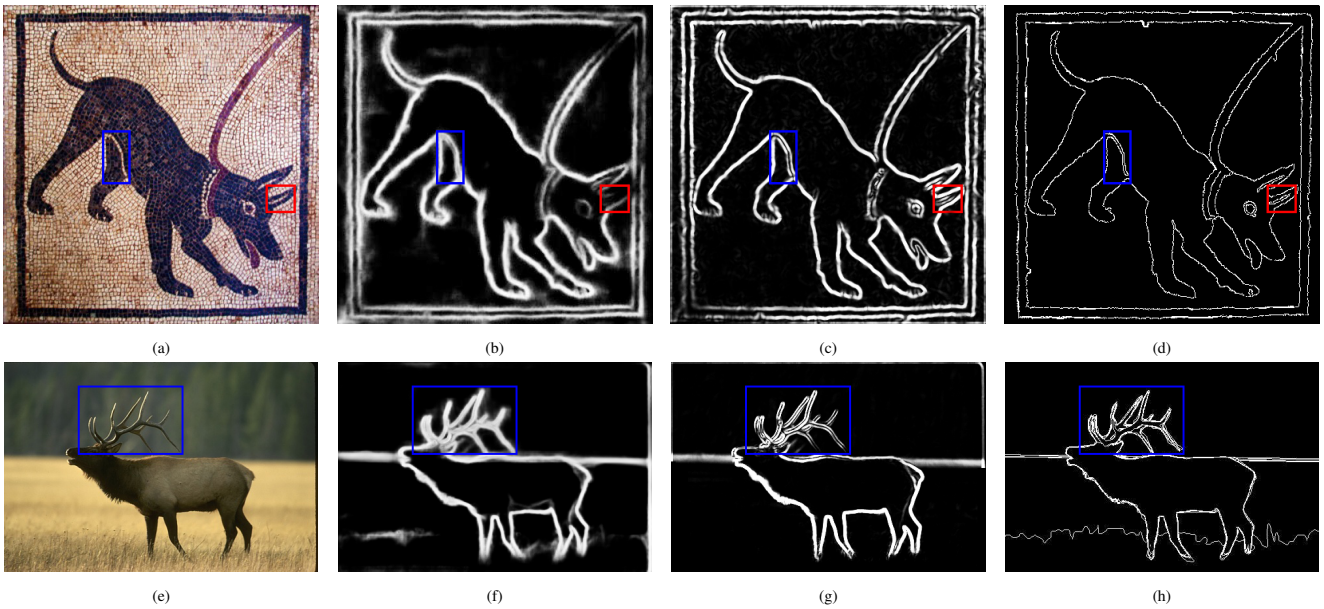


Figure 3: *Comparison of the estimated boundaries in textured images using the method from [CLKL14] and ours. (a) and (e) are the input images. (d) and (h) are the ground truths. In (b) and (f), the method from [CLKL14] resulted in some missed boundaries (in red rectangles), some boundaries in small structures being merged (in blue rectangles), and blurred estimated boundaries. For our results in (c) and (g), the boundaries are well detected, even in small structures, and they are very clean. Compared with the ground truth, our estimation can well capture the prominent object boundaries in textured images.*

2. Each pixel is assigned an adaptive scale box for adaptive mean smoothing of the image, in which the size of the pixel's adaptive scale box is determined by its measured texture value from step 1).
3. Texture measurement with a small box is applied to each pixel in the smoothed image of step 2).
4. The results from step 3) can well approximate the probability of pixels at boundaries, and they form the map for boundary estimation, which is called T_{map}.

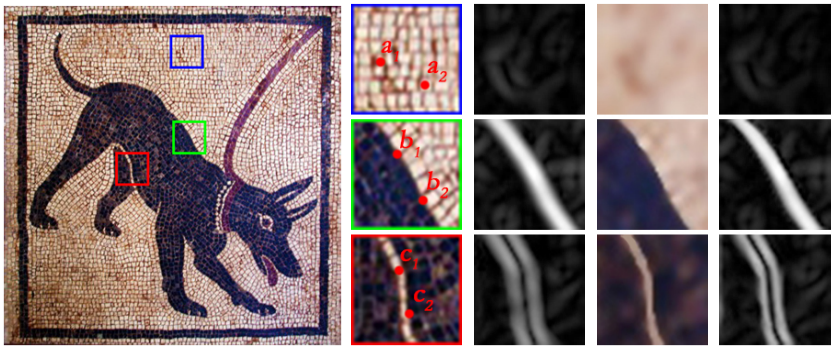
Now, we discuss our measure for high-quality boundary estimation using the illustration in Fig. 4. In step 1), using a large box for texture measurement is helpful for detecting both large-scale textures and small-scale textures. Of course, some small structures will be mistaken as textures. Fortunately, the pixels in the textured regions tend to have lower values in this texture measurement, and the pixels on boundaries would have higher values, even when the pixels are in small structures, which will be explained at the end of this section. Benefited from these differences, we can determine the adaptive boxes to smooth the image in step 2), through which the pixels in the texture regions would be smoothed using larger boxes, while the pixels at the boundaries would be smoothed using smaller boxes. Thus, the color differences between these two kinds of pixels would be enlarged. As a result, by using a small box for texture measurement in step 3), the values of the pixels on the boundaries and the pixels in textured regions can be well separated, as illustrated in the table in Fig. 4. Therefore, our boundary estimation in step 4) can be high-quality, as illustrated in Fig. 4. The experimental results in Section 6 will show that our results have much higher boundary recalls than existing methods, even the learning-based methods. This means that our measure is very effective for estimating the boundaries in textured images.

In our implementation, we used a large box with a size of 9×9 pixels for texture measurement in step 1) since the largest scale of textures in images is seldom larger than this size according to the previous literature. For the texture measurement in step 3), we used a small box of 3×3 pixels to enhance the individual characteristics of the pixels, and well separate the pixels on the boundaries from the pixels in textured regions. With regard to the adaptive scale box for smoothing a pixel p in step 2), it is computed by using the following formula:

$$K(p) = \zeta(10 \times (1 - T(p))) \tag{1}$$

where $T(p)$ is the value of the texture measurement of p in step 1), and ζ is a function that takes the nearest odd integer number. Clearly, if $T(p)$ is smaller, its $K(p)$ will be larger, and vice versa. Since $T(p)$ is generally in [0,1], $K(p)$ will be an odd number in [1, 9], meaning that the box for smoothing would not be larger than our used large box for texture measurement in step 1). In Fig. 4, for the pixels a_1 and a_2 in the textured region, their smoothing boxes are 9×9 pixels, and for the pixels b_1 and b_2 on the boundaries, their smoothing boxes are 3×3 pixels. This shows the effectiveness of our method for adaptive smoothing of pixels.

With respect to texture measurement, we adopted the measure in [LWS*16] due to its ease of use, which employs two kinds of



Pixels	First T	Final T_{map}
a_1	0.18	0.13
a_2	0.08	0.05
b_1	0.76	0.84
b_2	0.78	0.85
c_1	0.32	0.18
c_2	0.56	0.72

(a) Source image (b) Different regions (c) First T (d) Smoothed image (e) T_{map} (f) Measured values of the corresponding pixels

Figure 4: *Our texture measurement for boundary estimation. The first texture measurement (First T) with large boxes of* 9×9 *pixels can make the pixels* a_1 *and* a_2 *inside texture regions have low values, and the pixels* b_1 *and* b_2 *at boundaries have high values. As for the pixels* c_1 *and* c_2 *in small structures, their values are in the middle, and* c_1 *has a lower value than* c_2*, because* c_1 *is inside a texture region while* c_2 *at the boundary. After an adaptive smoothing, our second texture measurement (Final* T_{map}*) with small boxes of* 3×3 *pixels can make* a_1 *and* a_2 *assigned much lower values, and* b_1*,* b_2 *assigned much higher values. As for* c_1*, its value is decreased to be very low. With regard to* c_2*, its value is increased to be very high. This shows our potential for effective boundary estimation.*

boxes to handle pixels located in textured regions or in small structures. This measure is computed by the following formula:

$$\begin{aligned} T_x(p) &= |\sum_{q \in R(p)} g_{p,q} \cdot (\partial xI)_q| \\ T_y(p) &= |\sum_{q \in R(p)} g_{p,q} \cdot (\partial yI)_q| \\ g_{p,q} &\propto \exp(-\frac{(x_p - x_q)^2 + (y_p - y_q)^2}{2\sigma_1^2}) \\ T(p) &= Norm(T_x(p) + T_y(p)) \end{aligned} \tag{2}$$

where I is the input image, ∂xI and ∂yI are the gradients of the input image along the x and y directions, respectively. $R(p)$ is the box centered at pixel p and $g_{p,q}$ is a weighting function. $T_x(p)$ and $T_y(p)$ are called *windowed inherent variation*, which calculates the absolute sum of partial derivative's value. *Norm* is for normalizing the T_{map} values into the range of [0,1].

As discussed in [XYXJ12], if pixel p is inside a texture region, its $T(p)$ value would be very low since the partial derivative's value may be positive and negative, when summing the values up, the positive and negative values may be counteracted by each other. In contrast, if pixel p is on a boundary, it would have a high $T(p)$ value.

4. Distance Measure

Our method for superpixel segmentation also clusters pixels using seeds set beforehand. Here, an important task is to measure the distances between pixels and seeds. To suppress the impairment of the textures in superpixel segmentation. Inspired by [ZLGZ17], we develop a distance measure to well preserve the boundaries between textured regions when generating superpixels. It is given as:

$$D(p,s) = w_b \times B(p,s) + w_i \times I'(p,s) + \alpha \times w_c \times C(p,s) \tag{3}$$

$B(p,s)$, $I'(p,s)$ and $C(p,s)$, called the *boundary term*, the *color term* and the *compactness term*, respectively, are used for measuring the differences between the textures, colors and positions of two pixels in textured regions, and they each have their own weights that adjust their effects in superpixel segmentation. Noticed that $I'(p,s)$ is the smoothed image introduced in step 2), Section 3 and Algorithm 1. α is a parameter to control the compactness of the resulting superpixels. When α is higher, the resulting superpixels would be more compact, but their boundary adherence would be reduced, and vice versa. Generally, α can be set as a value in $[0.1, 5]$. In this paper, α is set to 1.0 to achieve a good balance between compactness and boundary adherence. To make the three terms have similar effects on superpixel segmentation, we compute their weights using the following formula:

$$\begin{aligned} w_b &= \frac{I'(p,s) + C(p,s)}{J} \\ w_i &= \frac{B(p,s) + C(p,s)}{J} \\ w_c &= \frac{I'(p,s) + B(p,s)}{J} \\ J &= 2(B(p,s) + I'(p,s) + C(p,s)) \end{aligned} \tag{4}$$

Clearly, if the boundary term has a higher value, its weight will be lower to reduce its effects on the distance measure, and vice versa. In this type of balancing treatment, the boundary term can well prevent the color term from having larger effects, thereby suppressing the influences of textures in superpixel generation. Thus, high-quality superpixels can be expected in textured images.

Algorithm 1 Our superpixel segmentation algorithm

Input: Input image I, desired superpixel number M and compactness parameter α.
Output: The resulted superpixels RS.

Compute $T(p)$ for each pixel p in I with Eq(2), where a large box of 9×9 pixels is used.
Compute $K(p)$ for each pixel p with Eq(1).
for each pixel p **do**
    $I'(p) \leftarrow$ adaptive mean filtering of $I(p)$.
end for
Compute $T_{map}(p)$ for each pixel p in I' with Eq(2), where a small box of 3×3 pixels is used..
Initialize clustering seeds C_M by sampling pixels evenly with the interval S between them.
Each initial seed is optimized to be at the pixel with the lowest T_{map} value in a box of 3×3 pixels that is centered at the seed.
for $t = 0 : n_{iter} - 1$ **do**
    for each clustering seed C_M **do**
        Collect the best matching pixels from its $2S \times 2S$ square neighborhood by the distance measure with Eq(3).
    end for
end for
Enforce connectivity.

As done in previous works [ASS*12, AS17], our color term and compactness term are computed using the following formulas:

$$I'(p,s) = \sqrt{(L_p - L_s)^2 + (a_p - a_s)^2 + (b_p - b_s)^2} \tag{5}$$

where $I'(p,s)$ is the smoothed image, (L_p, a_p, b_p) and (L_s, a_s, b_s) are respectively the colors of pixel p and seed s in the CIELab space.

$$C(p,s) = \sqrt{(x_p - x_s)^2 + (y_p - y_s)^2} \tag{6}$$

where (x_p, y_p) and (x_s, y_s) are the coordinates of pixel p and seed s in the image, respectively.

For the computation and the effects of the boundary term for superpixel segmentation, we conduct a detailed discussion in the following subsection.

4.1. Boundary Term

To avoid superpixels from straddling the boundaries of different texture regions, we use the boundary term to measure the probability of the line segment from pixel p to seed s straddling boundaries. Here, we use the Line Iterator in the OpenCV library to obtain the line pixels because it is simple and fast. For these pixels, their values of T_{map} are accumulated to compute the boundary term. If the boundary term has a higher value, it is more likely that the line segment straddles the boundaries between textured regions. Here, we use a clipping operation to avoid the accumulation of small T_{map} values that can mistake boundary information along with a long line segment. This is helpful for generating high-quality results when fewer superpixels are required because each of them tends to be much larger. The clipping operation forces the values that are smaller than a threshold to be zero. In our tests, we set the mean value of T_{map} as the threshold and it is able to obtain very good results. Finally, our boundary term is computed using the following formula:

(a) (b) (c)

Figure 5: *Comparison of the generated superpixels with different boundary estimations using our method and the method from [CLKL14]. (a) is the input image. (b) is the result by the method [CLKL14] and (c) is the result by our method. Clearly, our resulting superpixels are much better, e.g., those in the red rectangles.*

$$\begin{aligned} B(p,s) &= \sum_{q \in U} T_{map}(q), U \text{ is the line from } p \text{ to } s \\ T_{map}(q) &= \begin{cases} T_{map}(q), \text{ if } T_{map}(q) > \widetilde{T}_{map} \\ 0, \text{ otherwise} \end{cases} \end{aligned} \tag{7}$$

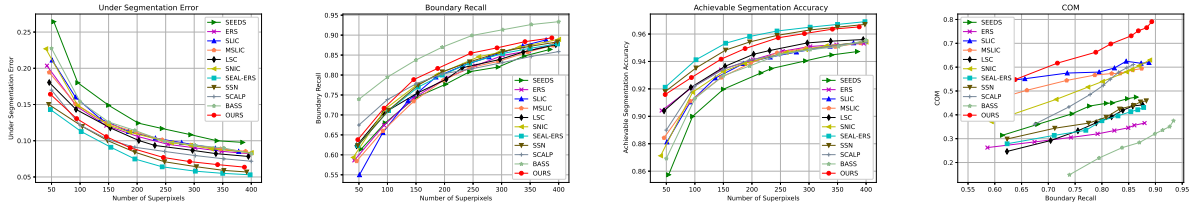
where U is the line from pixel p to seed s. $\widetilde{T}_{map}$ is the mean value of T_{map}.

Here, we have an analysis of our boundary term. Let's do not consider the color I' and the position C information first. If there are object boundaries between the pixel p and the seed s, the value of the boundary term and distance measure will be larger, so the pixel p is unlikely to belong to the superpixel of the seed s. In contrast, if there is no object boundary between the pixel p and the seed s, the boundary term is small or even zero, then the pixel p may belong to the superpixel of the seed s. Of course, distance measure also needs to consider color and position information. If the pixel p is in the texture region, the boundary term may be zero. But because the texture region in the smoothed image I' is more uniform than the source image I, the results of our method are more compact than other methods, as shown in Fig. 6.

5. Our Algorithm

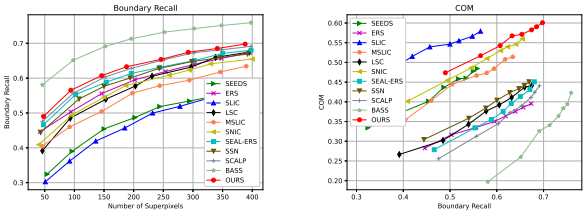
Our algorithm is composed of two parts. First, we construct T_{map} using the measures in Section 3. Then, we follow the pipeline of SLIC [ASS*12] for superpixel segmentation. For an image with N pixels, the neighboring superpixels should be positioned at a regular grid interval $S = \sqrt{N/M}$. The pixels joining the cluster of a seed are from a $2S \times 2S$ area centered at the seed. Afterwards, we use our new distance measure described in Section 4 to select the

Figure 6: *Visual comparison of our method against other state-of-the-art algorithms, including SEEDS [DBBR*12], ERS [LTRC11], SLIC [ASS*12], LSC [CLH17], MSLIC [LYYH16], SNIC [AS17], SEAL-ERS [TLJ*18], SSN [JSL*18], SCALP [GTP18] and BASS [URF19]. As illustrated in the zoomed-in regions, our superpixels can well and compactly adhere to the boundaries, while the superpixels of the other methods cannot.*



(a) UE (b) BR (c) ASA (d) COM

Figure 7: *Quantitative evaluation of the results by the tested methods on BSD500.*



(a) BR (b) COM

Figure 8: *Quantitative evaluation of the results by the tested methods on the textured images.*

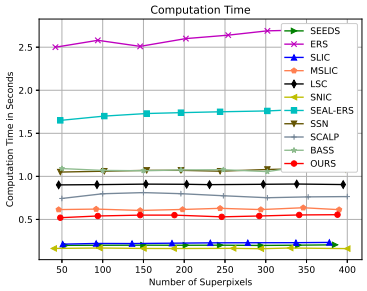


Figure 9: *Efficiency comparison.*

pixels for generating superpixels. Our algorithm is listed in Algorithm 1. Our T_{map} is constructed from Line 1 to Line 6, and our superpixel segmentation proceeds from Line 7 to Line 14.

In our algorithm, T_{map} plays an important role. If the boundaries in textured images are high-quality estimates, the superpixels can be well generated, and vice versa. As illustrated in Fig. 5, with the estimated boundaries using the method from [CLKL14], the generated superpixels cannot well preserve boundaries, while with our method, the generated superpixels are very good.

6. Experiments

We compared our method with ten existing methods, including the graph-based method ERS [LTRC11], the learning-based method SEAL-ERS [TLJ*18], SSN [JSL*18], BASS [URF19] and six clustering-based methods SEEDS [DBBR*12], SLIC [ASS*12], LSC [CLH17], MSLIC [LYYH16], SNIC [AS17], SCALP [GTP18]. We implemented our method with C++, and ran the implementation of the compared methods from their websites respectively. The datasets used for the comparison are the conventional images from the BSD500 [MFTM01] and the 40 textured images from [XW18]. The criteria for the quantitative evaluation are as follows:

- **Undersegmentation Error (UE)**. UE measures the deviation of superpixels from the ground truth segments. It is computed by the overlap of superpixels with multiple, nearby ground truth segments [NP12]. A lower UE indicates that more objects in an image are well recognized.
- **Boundary Recall (BR)**. BR is an important metric for evaluating the boundary adherence of superpixels by measuring the fraction of the ground truth boundaries that fall within at least two pixels of the resulting superpixel boundaries [MFTM01]. A high BR indicates that very few true boundaries are missed.
- **Achievable Segmentation Accuracy (ASA)**. ASA measures whether the objects in images are correctly recognized, by labeling each superpixel with the ground truth segments of the largest overlapping area [LTRC11]. Higher ASA values are expected when more correct objects are recognized.
- **Compactness (COM)**. COM cares for whether each superpixel has a regular shape and size [SFS12]. It is computed by comparing the area of each superpixel with the area of a circle (the most compact 2-dimensional shape) with the same perimeter, i.e., higher is better.

Fig. 6 displays some resulting superpixels from these methods, showing the superiority of our method over existing methods with respect to boundary preservation and compactness.

Fig. 7 plots the quantitative evaluation results for these four criteria for the resulting superpixels when using these methods on the BSD500 images [MFTM01]. Because SEAL-ERS and SSN use the segmentation ground truth of BSD500 as the training data, it behaves better than ours in terms of UE and ASA. Except for this, our method has the best performance among the compared methods. From the results in Fig. 6, it is clear that the generated superpixels with the learning-based methods SEAL-ERS, SSN and BASS are significantly cluttered.

Fig. 8 shows the evaluation results of BR and COM for these methods with respect to the textured images in [XW18]. We do not compute their UE and ASA values since only the ground truth of the boundaries are provided for these textured images. Note that BASS has the highest BR and the lowest COM values. This is be-

cause BASS cannot well distinguish the edge information in textures from the object boundaries, and so its results are much cluttered, as shown in Fig. 6. Our method makes considerable improvements in terms of BR and COM metrics in textured images.

We also compared these methods with respect to their efficiency in treating these images. The time costs were collected using a personal computer installed with an Intel Core i7, 3.40 GHz CPU, where the images are kept as 321×481 pixels. The average time for them to generate superpixels of different numbers in an image is plotted in Fig. 9. Since our method needs to construct T_{map} for boundary estimation, which requires approximately 0.2 seconds for an image in general, our method is a little slower than SEEDS, SLIC and SNIC. Even so, our method still have comparable efficiency to the state-of-the-art methods.

7. Conclusion

In this paper, we proposed a new superpixel segmentation algorithm that forgoes texture filtering and instead estimates the probability of pixels on boundaries in textured regions using the adaptive scale box, then we employed the probability information to design a novel distance measure for well considering the boundary, color, and Euclidean distance simultaneously. In this way, our approach can produce boundary-preserved superpixels. The experimental results show the superiority of our method over existing methods. Besides, our method is simple and fast, and it could be helpful for other vision tasks.

References

[AMFM11] ARBELAEZ P., MAIRE M., FOWLKES C. C., MALIK J.: Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 5 (2011), 898–916. 1

[AS17] ACHANTA R., SUSSTRUNK S.: Superpixels and polygons using simple non-iterative clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), 4895–4904. 1, 2, 3, 6, 7, 8

[ASS*12] ACHANTA R., SHAJI A., SMITH K., LUCCHI A., FUA P., SUSSTRUNK S.: Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 11 (2012), 2274–2282. 1, 2, 3, 6, 7, 8

[CLH*16] CHENG M., LIU Y., HOU Q., BIAN J., TORR P. H. S., HU S., TU Z.: Hfs: Hierarchical feature selection for efficient image segmentation. *Proceedings of the Conference on European Conference on Computer Vision* (2016), 867–882. 1

[CLH17] CHEN J., LI Z., HUANG B.: Linear spectral clustering superpixel. *IEEE Transactions on Image Processing 26*, 7 (2017), 3317–3330. 1, 2, 3, 7, 8

[CLKL14] CHO H., LEE H., KANG H., LEE S.: Bilateral texture filtering. *ACM Transactions on Graphics 33*, 4 (2014), 128. 1, 2, 3, 4, 6, 8

[DBBR*12] DEN BERGH M. V., BOIX X., ROIG G., DE CAPITANI B., VAN GOOL L.: Seeds: superpixels extracted via energy-driven sampling. *Proceedings of the Conference on European Conference on Computer Vision 7578* (2012), 13–26. 1, 2, 3, 7, 8

[FH04] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Efficient graph-based image segmentation. *International Journal of Computer Vision 59*, 2 (2004), 167–181. 1, 3

[GTP18] GIRAUD R., TA V., PAPADAKIS N.: Robust superpixels using color and contour features along linear path. *Computer Vision and Image Understanding 170* (2018), 1–13. 2, 7, 8

[JLKL16] JEON J., LEE H., KANG H., LEE S.: Scale-aware structure-preserving texture filtering. *Computer Graphics Forum 35*, 7 (2016), 77–86. 3

[JSL*18] JAMPANI V., SUN D., LIU M., YANG M., KAUTZ J.: Superpixel sampling networks. *Proceedings of the Conference on European Conference on Computer Vision* (2018), 363–380. 1, 3, 7, 8

[LCH*17] LIU Y., CHENG M., HU X., BIAN J., ZHANG L., BAI X., TANG J.: Richer convolutional features for edge detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 41*, 8 (2017), 1939–1946. 1

[LJP*18] LIU Y., JIANG P., PETROSYAN V., LI S., BIAN J., ZHANG L., CHENG M.: Del: Deep embedding learning for efficient image segmentation. *International Joint Conference on Artificial Intelligence* (2018), 864–870. 1

[LSK*09] LEVINSHTEIN A., STERE A., KUTULAKOS K. N., FLEET D. J., DICKINSON S., SIDDIQI K.: Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 12 (2009), 2290–2297. 3

[LTRC11] LIU M., TUZEL O., RAMALINGAM S., CHELLAPPA R.: Entropy rate superpixel segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2011), 2097–2104. 1, 2, 3, 7, 8

[LWS*16] LIN T., WAY D., SHIH Z., TAI W., CHANG C.: An efficient structure-aware bilateral texture filtering for image smoothing. *Computer Graphics Forum 35*, 7 (2016), 57–66. 3, 4

[LYYH16] LIU Y., YU C., YU M., HE Y.: Manifold slic: A fast method to compute content-sensitive superpixels. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 651–659. 1, 2, 3, 7, 8

[MFCP*15] MACHAIRAS V., FAESSEL M., CARDENAS-PENA D., CHABARDES T., WALTER T., DECENCIERE E.: Waterpixels. *IEEE Transactions on Image Processing 24*, 11 (2015), 3707–3716. 3

[MFTM01] MARTIN D., FOWLKES C. C., TAL D., MALIK J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proceedings of the IEEE Conference on International Conference on Computer Vision 2* (2001), 416–423. 8

[MK10] MICUSIK B., KOSECKA J.: Multi-view superpixel stereo in urban environments. *International Journal of Computer Vision 89*, 1 (2010), 106–119. 1

[MPAVG18] MANINIS K., PONTTUSET J., ARBELAEZ P., VAN GOOL L.: Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 40*, 4 (2018), 819–833. 1

[NP12] NEUBERT P., PROTZEL P.: Superpixel benchmark and comparison. *Forum Bildverarbeitung* (2012). 8

[PAB*17] PONTTUSET J., ARBELAEZ P., BARRON J. T., MARQUES F., MALIK J.: Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39*, 1 (2017), 128–140. 1

[PZCZ19] PAN X., ZHOU Y., CHEN Z., ZHANG C.: Texture relative superpixel generation with adaptive parameters. *IEEE Transactions on Multimedia 21*, 8 (2019), 1997–2011. 1

[RM03] REN, MALIK: Learning a classification model for segmentation. *Proceedings of the IEEE Conference on International Conference on Computer Vision* (2003), 10–17. 1

[RS13] REN Z., SHAKHNAROVICH G.: Image segmentation by cascaded region agglomeration. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), 2011–2018. 1

[SFS12] SCHICK A., FISCHER M., STIEFELHAGEN R.: Measuring and evaluating the compactness of superpixels. *Proceedings of the IEEE Conference on International Conference on Pattern Recognition* (2012), 930–934. 8

[SHL18] STUTZ D., HERMANS A., LEIBE B.: Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding 166* (2018), 1–27. 3

[SM00] SHI J., MALIK J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8 (2000), 888–905. 1, 3

[TLJ*18] TU W., LIU M., JAMPANI V., SUN D., CHIEN S., YANG M., KAUTZ J.: Learning superpixels with segmentation-aware affinity loss. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 568–576. 1, 3, 7, 8

[URF19] UZIEL R., RONEN M., FREIFELD O.: Bayesian adaptive superpixel segmentation. *Proceedings of the IEEE Conference on International Conference on Computer Vision* (2019), 8470–8479. 1, 3, 7, 8

[WLYY11] WANG S., LU H., YANG F., YANG M.: Superpixel tracking. *Proceedings of the IEEE Conference on International Conference on Computer Vision* (2011), 1323–1330. 1

[XW18] XU P., WANG W.: Improved bilateral texture filtering with edge-aware measurement. *IEEE Transactions on Image Processing 27*, 7 (2018), 3621–3630. 3, 8

[XYXJ12] XU L., YAN Q., XIA Y., JIA J.: Structure extraction from texture via relative total variation. *ACM Transactions on Graphics 31*, 6 (2012), 1. 3, 5

[YZL*13] YANG C., ZHANG L., LU H., RUAN X., YANG M.: Saliency detection via graph-based manifold ranking. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), 3166–3173. 1, 3

[ZLGZ17] ZHANG Y., LI X., GAO X., ZHANG C.: A simple algorithm of superpixel segmentation with boundary constraint. *IEEE Transactions on Circuits and Systems for Video Technology 27*, 7 (2017), 1502–1514. 5