FLIC: Fast Linear Iterative Clustering with Active Search

Jiaxing Zhao[1] **Bo Ren**[1*] **Qibin Hou**[1] **Ming-Ming Cheng**[1] **Paul L. Rosin**[2]
[1]Nankai University [2]Cardiff University

Abstract

In this paper, we reconsider the clustering problem for image over-segmentation from a new perspective. We propose a novel search algorithm named "active search" which explicitly considers neighboring continuity. Based on this search method, we design a back-and-forth traversal strategy and a "joint" assignment and update step to speed up the algorithm. Compared to earlier works, such as Simple Linear Iterative Clustering (SLIC) and its follow-ups, who use fixed search regions and perform the assignment and the update step separately, our novel scheme reduces the number of iterations required for convergence, and also improves the boundary sensitivity of the over-segmentation results. Extensive evaluations on the Berkeley segmentation benchmark verify that our method outperforms competing methods under various evaluation metrics. In particular, lowest time cost is reported among existing methods (approximately 30 fps for a 481×321 image on a single CPU core). To facilitate the development of over-segmentation, the code will be *publicly available*.

1 Introduction

Superpixels, generated by image over-segmentation, take the place of pixels to become the fundamental units in various computer vision tasks, including image segmentation (Cheng et al. 2016), image classification (Wang et al. 2013b), 3D reconstruction (Hoiem, Efros, and Hebert 2005), object tracking (Wang et al. 2011), *etc*. Such a technique can greatly reduce computational complexity, avoid under-segmentation, and reduce the influence caused by noise. Therefore, how to generate superpixels with high efficiency plays an important role in many vision and image processing applications.

Generating superpixels has been an important research issue, and a group of classical methods have been developed, including FH (Felzenszwalb and Huttenlocher 2004), Mean Shift (Comaniciu and Meer 2002), Watershed (Vincent and Soille 1991), *etc*. The lack of compactness and the irregularity of superpixels restrict their applications, especially when contrast is poor or shadows are present. To solve the above-mentioned problems, Shi and Malik proposed Normalized Cuts (NC) (Shi and Malik 2000) that generated compact superpixels. However, this method does not adhere to image boundaries very well, and the complexity is high. GraphCut (Boykov, Veksler, and Zabih 2001; Veksler, Boykov, and Mehrani 2010) regarded the segmentation problem as an energy optimization process. It solved the compactness problem by using min-cut/max-flow algorithms (Boykov and Kolmogorov 2004; Kolmogorov and Zabin 2004), but their parameters are hard to control. Turbopixel (Levinshtein et al. 2009) is another method that is proposed to solve the compactness problem. However, the inefficiency of the underlying level-set method (Osher and Sethian 1988) restricts its applications. Bergh *et al.* (2012) proposed an energy-driven algorithm SEEDS whose results adhered to the boundaries well, but unfortunately it suffers from irregularity and the number of superpixels is uncertain. ERS (Liu et al. 2011), although it performs well on the Berkeley segmentation benchmark, has a high computational cost that limits its practical use.

Achanta *et al.* (2012) proposed a linear clustering based algorithm SLIC. It generates superpixels based on Lloyd's algorithm (Lloyd 1982) (also known as Voronoi iteration or k-means). In the assignment step of SLIC, as a key point to speed up the algorithm, each pixel p is associated with those cluster seeds whose search regions overlap its location. Such a strategy is also adopted by most subsequent works based on SLIC. SLIC is widely used in various applications (Wang et al. 2011) because of its high efficiency and good performance. Inspired by SLIC, Wang *et al.* (2013a) implemented an algorithm SSS that considered the structural information within images. It uses the geodesic distance (Peyré et al. 2010) computed by the geometric flows instead of the simple Euclidean distance. However, efficiency is poor because of the bottleneck caused by the high computational cost of measuring geodesic distances. Very recently, Liu *et al.* proposed Manifold SLIC (2016) that generated content-sensitive superpixels by computing Centroidal Voronoi Tessellation (CVT) (Du, Faber, and Gunzburger 1999) in a special feature space. Such an advanced technique makes it much faster than SSS but still slower than SLIC owing to the cost of its mapping, splitting and merging processes. From the aforementioned descriptions, we see that the above-mentioned methods improve the results by either using more complicated distance measurements or providing more suitable transformations of the feature space. However, the assignment and update steps within these methods

[*]Bo Ren is the corresponding author: rb@nankai.edu.cn.

are performed separately, leading to low convergence rate.

In this paper, we consider the over-segmentation problem from a new perspective. Each pixel in our algorithm is allowed to actively search which superpixel it should belong to, according to its neighboring pixels as shown in Figure 1. In the meantime, the seeds of the superpixels can be adaptively changed during this process, which allows our assignment and update steps to be performed jointly. This property enables our approach to converge rapidly. To sum up, our main advantages are:

- Our algorithm features good awareness of neighboring-pixel continuity, and produces results with good boundary sensitivity regardless of image complexity and contrast.
- Our algorithm performs the assignment step and the update step in a joint manner, and has a high convergence rate as well as the lowest time cost among all superpixel segmentation approaches. Experiments show that our approach is able to converge in two scan loops, with better performance measured under a variety of evaluation metrics on the Berkeley segmentation benchmark.

2 Preliminaries

Before introducing our approach that allows adaptive search regions and joint assignment and update steps, we first briefly recap the standard previous scheme with fixed search regions and separate steps. A typical one is the SLIC algorithm who improves Lloyd's algorithm, reducing the time complexity from $O(KN)$ to $O(N)$, where K is the number of the superpixels and N is the number of pixels.

Let $\{I_i\}_{i=1}^{N}$ be a color image, where I_i represents the corresponding variable of each pixel. Given a set of evenly distributed seeds $\{S_k\}_{k=1}^{K}$, SLIC simplifies the Lloyd's algorithm to get the Centroidal Voronoi Tessellation (CVT) (Du, Faber, and Gunzburger 1999) that will be introduced in Section 3.4. In the assignment step, each pixel I_i is associated with those cluster seeds whose search regions overlap its location as shown in Figure 1(a). The area of a search region can be denoted by $2T \times 2T$, where $T = \sqrt{N/K}$. Specifically, SLIC considers I_i to lie in a five dimensional space that contains a three dimensional CIELAB color space (l_i, a_i, b_i) and a two dimensional spatial space (x_i, y_i). SLIC measures the distance between two points using a weighted Euclidean distance, which can be computed by

$$D(I_i, I_j) = \sqrt{d_c^2 + \left(\frac{d_s * m}{N_s}\right)^2}, \tag{1}$$

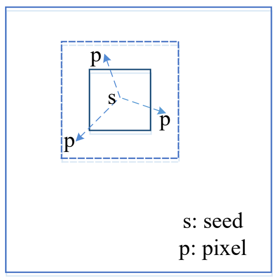
where m is a variable that controls the weight of the spatial term, and $N_s = T$. Variables d_s and d_c are respectively the spatial and color distances, which can be expressed as

$$d_s = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{2}$$

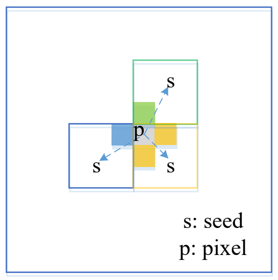
and

$$d_c = \sqrt{(l_i - l_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2}. \tag{3}$$

In the update step, SLIC recomputes the center of each superpixel and moves the seeds to these new centers. Then it



(a) SLIC



(b) Active search

Figure 1: (a) The search method used in SLIC. Each seed only searches a limited region to reduce computation complexity. (b) Our proposed active search. Each pixel is able to decide its own label by searching its surroundings.

obtains the over-segmentation results by iteratively performing the assignment and update steps.

The follow-up works of SLIC also use a similar procedure as SLIC. They improve the performance of SLIC using better distance measures or more suitable transformation function between color space and spatial space. However, in these algorithms, each search region is fixed in the assignment step of a single loop, and the relationship among neighboring pixels is largely ignored when allocating pixels to superpixels. Separately performing the assignment step and the update step also leads to a delayed feedback of pixel label change.

3 The Proposed Approach

Since superpixels normally serve as the first step of other vision related applications, how to generate superpixels with good boundaries and very fast speed is a crucial problem. Here, unlike previous algorithms (Achanta et al. 2012; Liu et al. 2016), we consider this problem from a new aspect, in which only the surrounding pixels are considered for determining the label of the current pixel. Each pixel actively selects which superpixel it should belong to in a back-and-forth order to provide better estimation of over-segmentation regions. Moreover, the assignment step and the update step are performed jointly. Very few iterations are required for our approach to reach convergence. An overview of our algorithm can be found in Alg. 1.

3.1 Problem Setup

Given the desired number of superpixels K and an input image $I = \{I_i\}_{i=1}^{N}$, where N is the number of pixels, our goal is to produce a series of disjoint small regions (or superpixels). Following most previous works (Achanta et al. 2012), the original RGB color space is transformed to the CIELAB color space (which has been proven useful). Thus, each pixel I_i in an image I can be represented in a five dimensional space,

$$I_i = (l_i, a_i, b_i, x_i, y_i). \tag{4}$$

We first divide the original image into a regular grid containing K elements $\{G_k\}_{k=1}^{K}$ with step length $v = \sqrt{N/K}$ as

Algorithm 1 FLIC

```
Require: Image $I$ with $N$ pixels, the desired number of superpixels $K$, the maximal iteration numbers $itr_{max}$ and the spatial distance weight $m$.
Ensure: $K$ superpixels
  Divide the original image into regular grids $\{G_k\}_{k=1}^K$ with step length $v = \sqrt{N/K}$.
  Initialize labels $\{L_k\}_{k=1}^K$ for pixels according to their locations.
  Move each seed to the lowest gradient position in $3\times3$ neighborhoods.
  Initialize seeds $\{S_k\}_{k=1}^K$.
  Regard pixels sharing the same label as a superpixel $\zeta$.
  Initialize distance $d_{(i)} = \infty$ for each pixel and $itr = 0$.
  while itr < $itr_{max}$ do
      for each superpixel $\zeta_k$ do
          Use back-and-forth scan to traverse superpixel $\zeta_k$ to get the pixels processing sequence (§3.3).
          for each pixel $I_i$ in the sequence do
              Set $d_{(i)} = D(I_i, S_{L_i})$ by Eqn. 1
              for $I_j$ in the four-neighborhood of $I_i$ do
                  if $L_j \neq L_i$ then
                      Compute $D = D(I_i, S_{L_j})$. (Eqn. 1)
                      if $D < d_{(i)}$ then
                          $d_{(i)} = D;\ L_i = L_j$.
                      end if
                  end if
              end for
              if $L_i$ is changed to $L_j$ then
                  Use Eqn. 10 to update $\zeta_{L_i}$;
                  Use Eqn. 11 to update $\zeta_{L_j}$;
                  Update the bounding box of $\zeta_{L_j}$ (§3.4).
              end if
          end for
      end for
      itr++;
  end while
```

in (Achanta et al. 2012), and the initial label for each pixel I_i is assigned as:

$$L_i = k,\ if\, I_i \in G_k. \tag{5}$$

We initialize the seed S_k in G_k as the centroid. Therefore, S_k can also be defined in the same five dimensional space

$$S_k = \{l_k, a_k, b_k, x_k, y_k\}. \tag{6}$$

3.2 Label Decision

In most natural images adjacent pixels tend to share the same labels, *i.e.* neighboring pixels have natural continuity. Thus, we propose an active search method that is able to leverage as much of this a priori information as possible. In our method, unlike most previous works (Achanta et al. 2012; Liu et al. 2016), the label of the current pixel is only determined by its neighbors. We will compute the distances between the current pixel and the seeds of its four or eight adjacent pixels. Figure 1 provides a more intuitive illustration. Specifically, for a pixel I_i, our assignment principle is

$$L_i = \underset{L_j}{argmin}\, D(I_i, S_{L_j}), I_j \in A_i, \tag{7}$$

where A_i consists of I_i and its four neighboring pixels, S_{L_j} is I_j's corresponding superpixel seed. We use Eqn. 1 to measure the distance $D(I_i, S_{L_j})$.

Since each pixel can only be assigned to a superpixel containing at least one of its neighbors, the local pixel continuity has a stronger effect in the proposed strategy, allowing each pixel to actively assign itself to one of its surrounding closely connected superpixel regions. The advantages of such a strategy are obvious. First, the nearby assignment principle can avoid the occurrence of too many isolated regions, indirectly preserving the desired number of superpixels. Second, such an assignment operation is not limited by a fixed range in space, resulting in better boundary adherence despite some irregular shapes of those superpixels with very complicated content. Furthermore, in the assignment process, the superpixel centers are also self-adaptively modified, leading to faster convergence. Detailed demonstration and analysis can be found in Section 4.3. It is worth mentioning that the neighbors of the internal pixels in a superpixels normally share the same labels, so it is unnecessary to process them any more. This fact allows us to process each superpixel extremely quickly.

3.3 Traversal Order

The traversal order plays a very important role in our approach in that an appropriate scanning order may lead to a visually better segmentation. As demonstrated in Section 3.2, the label of each pixel only depends on the seeds of its surrounding pixels. This indicates that, in a superpixel, the label of the current pixel is directly or indirectly related to those pixels that have already been dealt with. To better take advantage of this avalanche effect, we adopt a back-and-forth traversal order as in PatchMatch (Barnes et al. 2009), in which the pixels that are processed later will benefit from the previously processed pixels. Figure 2 makes this process clear. In the forward pass, the label decision of each pixel considers the information from the top surrounding pixels of the superpixel, and similarly, the backward pass will provide the information from the bottom surrounding pixels of the superpixel. With such a scanning order, all the surrounding information can be taken into consideration, yielding better segments.

Considering that an arbitrary superpixel might have an irregular shape instead of a simple rectangle or square, we actually use a simplified strategy to traverse the whole superpixel. For any superpixel, we first find a minimum bounding box within which all its pixels are enclosed, as shown in Figure 2. We then perform the scanning process for all the pixels in the corresponding minimum bounding box and only deal with those pixels that are within the superpixel.

3.4 Joint assignment and update step

A common phenomenon in existing methods, such as SLIC (Achanta et al. 2012), is that the assignment step and the update step are performed separately, leading to delayed feed-

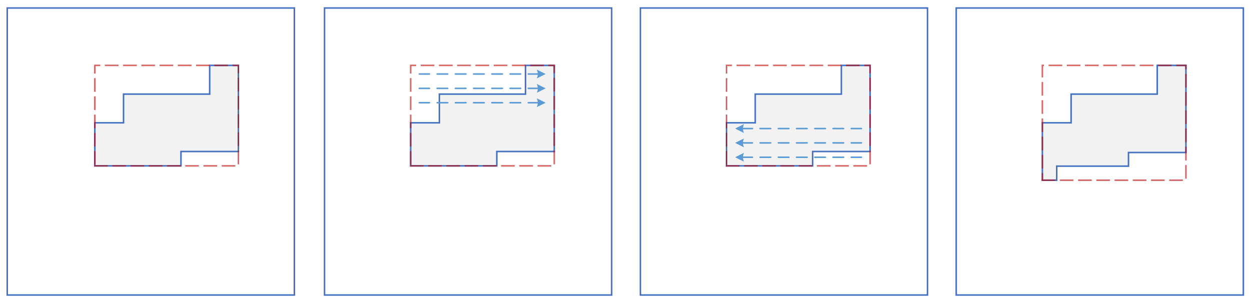

(a) A region and its bounding box (b) Forward scan order (c) Backward scan order (d) Bounding box after updating

Figure 2: Illustration of our scanning order for each superpixel. We use gray regions enclosed by blue lines to represent superpixels and use red dashed rectangles to denote their corresponding bounding boxes. As shown in (b) and (c), we first scan the bounding box from left to right and top to bottom and then in the opposite direction. The shape of each superpixel might change, and so we update the bounding box if this occurs, as in (d), and leave it unchanged if there are no changes to the superpixel shape.

back from pixel label changes to superpixel seeds. An obvious problem of such a strategy is that many (normally more than five) iterations are required which becomes the bottleneck of fast convergence. In our approach, based on the assignment principle Eqn. 7, we design a "joint" assignment and update strategy which operate these two steps at a finer granularity. The approximately joint step is able to adjust the superpixel seed center position on the fly, drastically reducing the number of iterations needed for convergence. Since most clustering-based superpixel methods use the Centroidal Voronoi Tessellation (CVT), we will briefly introduce the CVT first and then describe our method.

Let $S = \{S_k\}_{k=1}^{K}$ be the set of seeds in the image, where K is the expected number of superpixels. The Voronoi cell $\mathcal{V}_{(S_k)}$ of a seed S_k is denoted by:

$$\mathcal{V}_{S_k} = \{I_i \in I \mid d(I_i, S_k) \leq d(I_i, S_j), \forall S_j \in S\}, \quad (8)$$

where $d(I_i, S_k)$ is an arbitrary distance measure from pixel I_i to the seed S_k. The Voronoi Diagram $\mathcal{V}_D(S)$ is defined by

$$\mathcal{V}_D(S) = \{\mathcal{V}_{S_k} \neq \phi \mid \forall S_k \in S\}. \quad (9)$$

A CVT is then defined as a Voronoi Diagram whose generator point of each Voronoi cell is also its center of mass. As mentioned above, the traditional CVT is usually obtained by heuristic algorithms, such as Lloyd's algorithm, iteratively performing updates after each assignment step until convergence is reached.

In our approach, on account of our novel label decision strategy as shown in Eqn. 7, we are able to jointly perform the update step and the assignment step instead of separately. More specifically, after pixel I_i is processed, if its label is changed to, for instance, L_j, we immediately update the current seed S_{L_i} using the following equation:

$$S_{L_i} = \frac{S_{L_i} * |\zeta_{L_i}| - I_i}{|\zeta_{L_i}| - 1}, \quad (10)$$

where $|\zeta_{L_i}|$ is the number of pixels in superpixel ζ_{L_i}, and update S_{L_j} using the following equation

$$S_{L_j} = \frac{S_{L_j} * |\zeta_{L_j}| + I_i}{|\zeta_{L_j}| + 1}. \quad (11)$$

The bounding box of ζ_{L_j} is also updated thereafter.

It is noteworthy to mention that the above updates only contain very simple arithmetic operations and hence can be performed very efficiently. Such an immediate update will help later pixels make a better choice during assignment, leading to better convergence. Figure 6 shows the convergence speed of our approach.

4 Experiments

Our method is implemented in C++ and runs on a PC with an Intel Core i7-4790K CPU with 4.0GHz, 32GB RAM, and 64 bit operating system. We compare our method with many previous and current state-of-the-art works, including FH (Felzenszwalb and Huttenlocher 2004), SLIC (Achanta et al. 2012), Manifold SLIC (Liu et al. 2016), SEEDS (Van den Bergh et al. 2012), and ERS (Liu et al. 2011) on the BSDS500 benchmark, using the evaluation methods proposed in (Arbelaez et al. 2011; Stutz, Hermans, and Leibe 2014). Note that the source codes used in evaluation of the above works may be of different versions, and we find this leads to performance difference from the original reports when a different implementation of evaluation code is applied. To give a fair comparison, we uniformly use publicly available source code (Arbelaez et al. 2011; Stutz, Hermans, and Leibe 2014) for all the methods. As in previous research in the literature (Liu et al. 2016; Wang et al. 2013a), we evaluate all algorithms on 200 randomly selected images of resolution 481×321 from the Berkeley dataset.

4.1 Parameters

In our approach, three parameters need to be set. The first one is the number of superpixels K. One of the common advantages of clustering-based algorithms is that the expected number of superpixels can be directly obtained by setting the clustering parameter K. The second one is the spatial distance weight m. Parameter m has a large effect on the smoothness and compactness of superpixels. We shall show

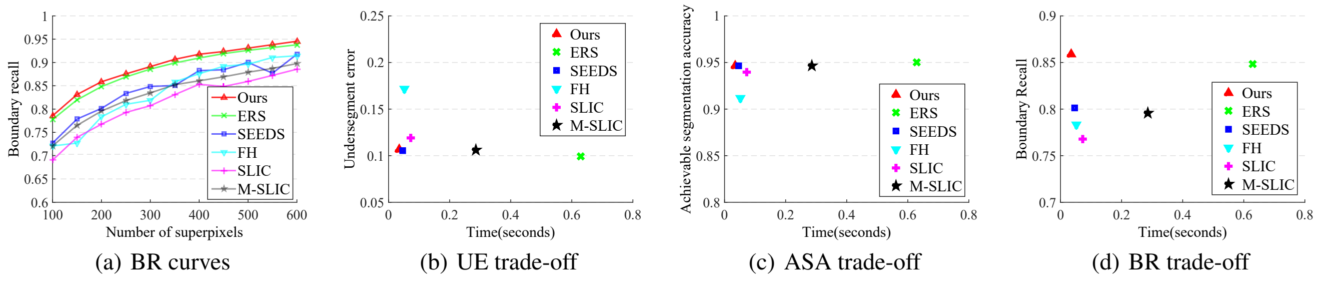


Figure 3: Comparisons between existing state-of-the-art methods and our approach (FLIC) on the BSDS500 benchmark. In (b)-(d), K is fixed to 200 to demonstrate our best trade-off between performance and efficiency between competing methods. As can be seen, our strategy significantly outperforms methods that have similar time cost in boundary recall. At least competitive results are also achieved compared to slower methods (e.g. the state-of-the-art method ERS (Liu et al. 2011)) according to all the evaluation metrics, but at an order of magnitude faster speed.

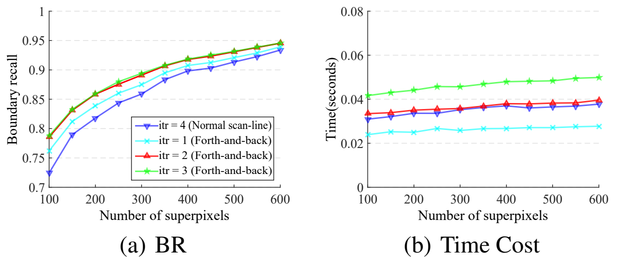


Figure 4: Part of sensitivity analysis under the standard evaluation metrics and time cost.

that our performance will increase as m decreases. However, a small m can also lead to the irregularity of superpixels. To achieve a good trade-off between compactness and performance, in the following experiments, we set $m = 5$ as default. The last parameter is the number of iterations itr. Here we set $itr = 2$ in default to get the balance between time cost and performance. What should be stressed out here is that to compare with other methods *in a fair way*, for each method we optimize its parameters to maximize the recall value computed on the BSDS500 benchmark.

4.2 Comparison with Existing Methods

Our approach outperforms previous methods that have similar computational efficiency , and achieve at least comparable results compared to slower algorithms with an order of magnitude faster speed. Details are discussed below.

Boundary Recall (BR). Boundary recall is a measurement which denotes the adherence to the boundaries. It computes what fraction of the ground truth edges falls within ε-pixel length from at least one superpixel boundary. The BR (Achanta et al. 2012) can be computed by

$$BR_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_{p\in\xi_{\mathcal{G}}} \Pi(\min_{q\in\xi_{\mathcal{S}}} \|p-q\| < \varepsilon)}{|\xi_{\mathcal{G}}|}, \qquad (12)$$

where $\xi_{\mathcal{S}}$ and $\xi_{\mathcal{G}}$ respectively denote the union set of superpixel boundaries and the union set of ground truth boundaries. The indicator function Π checks if the nearest pixel

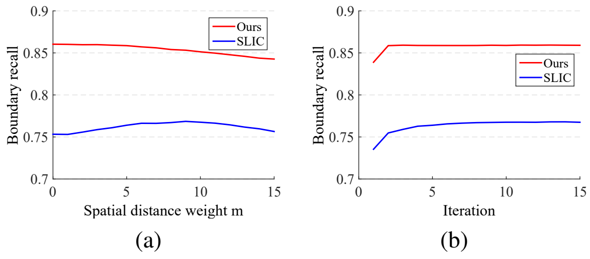


Figure 5: (a) The BR-m curves, where m is the spatial distance weight in Eqn. 1. Our overall performance is far better than SLIC for all the tested m. (b) The BR-Iteration curves. Our method converges within 2 iterations, which is much faster than SLIC.

is within ε distance. Here we follow (Achanta et al. 2012; Liu et al. 2016) and set $\varepsilon = 2$ in our experiment. The boundary recall curves of different methods are plotted in Figure 3(a). One can easily observe that our FLIC method outperforms all other methods.

Undersegment Error (UE). The undersegment error reflects the extent that superpixels do not exactly overlap the ground truth segmentation. Similar to BR, UE can also reflect the boundary adherence. The difference is that UE uses segmentation regions instead of boundaries in the measurement. Mathematically, the UE (Neubert and Protzel 2012) can be computed by

$$UE_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_{G\in\varrho_{\mathcal{G}}} (\sum_{S:S\cap G\neq\phi} \min(S_{in}, S_{out}))}{N}, \qquad (13)$$

where $\varrho_{\mathcal{S}}$ is the union set of superpixels, $\varrho_{\mathcal{G}}$ is the union set of the segments of the ground truth, S_{in} denotes the overlapping of the superpixel S and the ground truth segment G, and S_{out} denotes the rest of the superpixel S. As shown in Figure 3(b), our results are nearly the same as those of the best approach ERS (Liu et al. 2011) and run significantly faster.

Achievable Segmentation Accuracy (ASA). ASA gives the highest accuracy achievable for object segmentation that utilizes superpixels as units. Similar to UE, ASA utilizes

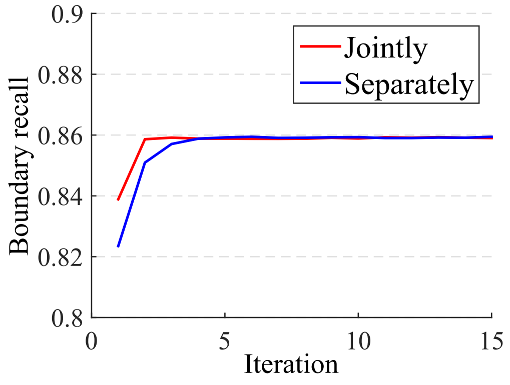


Figure 6: Comparison of the convergence rate between joint and separate assignment and update steps.

	100		200		300		400	
	BR	Time	BR	Time	BR	Time	BR	Time
4-N	78.6	34	85.9	35	89.1	36	91.8	38
8-N	80.5	54	87.4	56	90.5	59	92.7	61

Table 1: Boundary recall and time cost comparisons between 4-neighborhood and 8-neighborhood with different superpixel counts: 100, 200, 300, 400.

segments instead of the boundaries, which can be computed by (Liu et al. 2011)

$$ASA_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_k \max_i |S_k \cap G_i|}{\sum_i G_i}, \tag{14}$$

where S_k represents the superpixel and G_i represents the ground truth segment. A better superpixel segmentation will have a larger ASA value. As shown in Figure 3(c), compared to the ERS (Liu et al. 2011), the performance of our approach is competitive and our method achieves the best trade-off between the performance and time cost.

Time Cost (TC). Similar to SLIC, our method also achieves an $O(N)$ time complexity. We know that computation efficiency is one of the most important points for using superpixels as elementary units. Many approaches are limited by their speed, such as SSS (Wang et al. 2013a) and ERS (Liu et al. 2011). As shown in Figure 3, the average time cost of FLIC with two iterations when processing an image is 0.035s, while the time costs of ERS, Manifold SLIC, SLIC, and FH are 0.625, 0.281s, 0.072s, and 0.047s, respectively. It is obvious that FLIC has the lowest time cost among all methods and it runs nearly 20 times faster than ERS with comparable result quality.

Visual Results and Analysis. In Figure 7, we show several superpixel segmentation results using different algorithms. As can be seen, our approach is more sensitive to image boundaries, especially when there is poor contrast between the foreground and background. Compared to SLIC method, our approach adheres to boundaries very well and runs twice as fast. Compared to ERS method, our resulting superpixels are much more regular and the mean execution time of our approach is 20 times shorter.

All the above facts and Figure 3 reflect that our approach achieves an excellent compromise among *adherence*, *compactness*, and *time cost*.

4.3 Algorithm Analysis

Efficacy of the back-and-forth Traverse Order. As shown in Figure 2, we adopt a back-and-forth traverse order to scan the whole region enclosed by a bounding box for each superpixel. Actually, a couple of forward scans can also perform very well for our method. We provide a comparison between two strategies: using pure forward scan order for four iterations versus using the proposed back-and-forth scan order twice (which is also four iterations). Figure 4 shows quantitative comparisons between these two strategies. The blue line represents the results using normal forward scan order while the red line stands for the results using our method. It can be seen that the red curve significantly outperforms the blue one and achieves competitive time cost compared to the blue curve. This fact reflects that our back-and-forth scan order considers more information about the regions outside the bounding box, leading to more reliable boundaries.

The Role of the Spatial Distance Weight. As shown in Figure 5(a), unlike SLIC (Achanta et al. 2012), the BR curve with respect to the spatial distance weight m is monotonically decreasing in our approach. The reason for this phenomenon is that in our method local region continuity is mostly ensured by the active search algorithm, and color boundaries are less well preserved for larger m. On the other hand, small m will result in less regular superpixels, so we choose $m = 5$ for our comparison with previous works. It is noteworthy to mention that superpixels are normally considered as the first step of most vision tasks and these vision tasks often favor those superpixel methods with good boundaries. Therefore, users can select a reasonable value for m according to their specific conditions. In any case, our overall performance is significantly better for all m values.

Convergence Rate. FLIC significantly accelerates the evolution so that we only need a few iterations before convergence. We compare the performance curves with different iterations on the Berkeley benchmark. It can be easily found in Figure 5(b) that our algorithm quickly converges within only two iterations and more iterations only bring marginal benefits to the results. Numerically, the boundary recall of the superpixels with only one iteration is 0.835 when K is set to 200. The value after two iterations is 0.859 and after three iterations it is 0.860 when generating the same number of superpixels. The undersegment error values are 0.115, 0.108, and 0.107, respectively. The achievable segmentation accuracy values are 0.941, 0.945, and 0.946, respectively. As can be seen in Figure 5(b), our algorithm not only converges much faster than SLIC (which requires ten iterations to converge), but also obtains better performance.

The Role of the Joint Assignment and Update. Our algorithm jointly performs the assignment and update steps. In Figure 6, we show the convergence rates of both our joint approach and that of separately performing assignment and update steps. One can observe that our joint approach

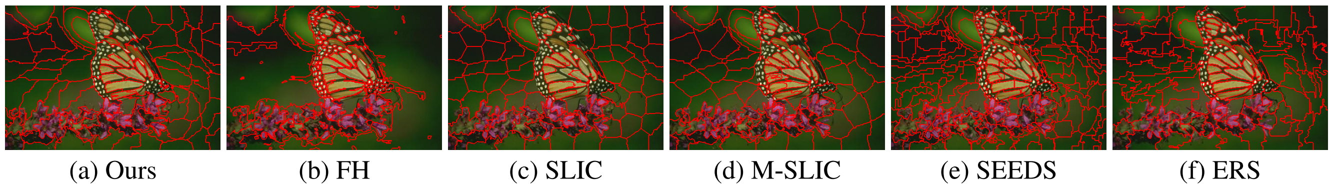


Figure 7: Visual comparison of superpixel segmentation results using different existing algorithm with 100 superpixels and $m = 10$. Our approach adheres to boundaries very well and at the same time produces compact superpixels.

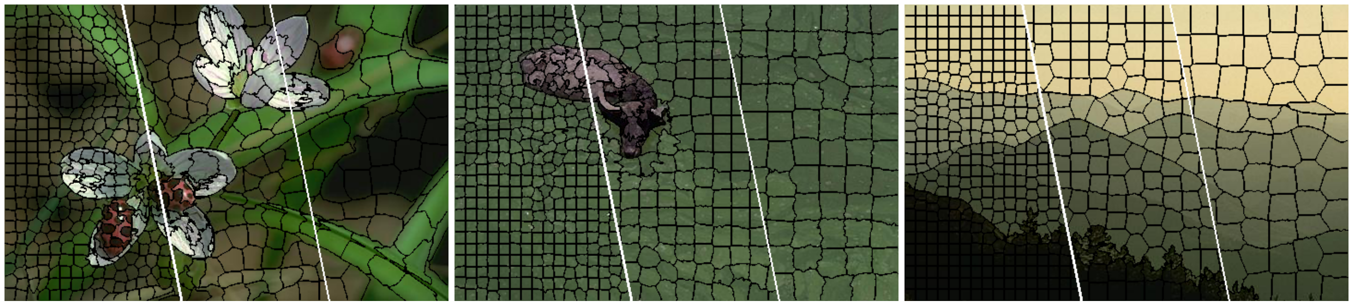


Figure 8: Images segmented by our proposed approach with $m = 20$ and the number of superpixels set to 1000, 400, and 200, respectively. The resulting superpixels adhere to region boundaries very well.

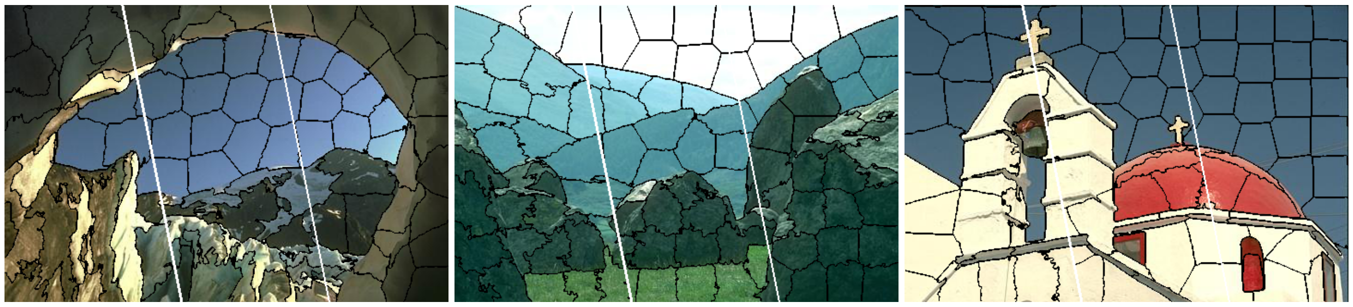


Figure 9: Images segmented by our proposed approach with $m = 10, 20$, and 30, respectively. When m tends to be a smaller value, the superpixels adhere well to boundaries. When m becomes larger, the superpixels become more compact.

converges very quickly and only two iterations are needed, while the separate approach needs another two iterations to reach the same BR value. This phenomenon demonstrates that our joint approach is efficient and without any negative effect on our final results.

Effect on the Size of Neighborhoods. As mentioned in Section 3, in our implementation, the label of the current pixel relies on its four neighborhood pixels. Actually, using eight neighborhood pixels is also reasonable as more neighbors will definitely provide more useful information. In Table 1, we briefly compare the results for these two cases. A natural observation is that using larger neighborhoods leads to an increase in performance but at the cost of reducing running speed. With regard to real applications, users can select either case to suit their own preferences.

Qualitative Results. In Figure 8 we show some segmentation results produced by our approach with $m = 20$ and the number of superpixels set to 1000, 400, and 200, respectively. It is seen that, over the range of K value, the edges of the resulting superpixels are always very close to the boundaries. This phenomenon is especially obvious in the first image and the third image. We also show some segmentation results with different values of m in Figure 9. When m tends to smaller values, for example 10, the shapes of the resulting superpixels become less regular. When m is larger, for example 30, the resulting superpixels become more compact.

5 Conclusions

In this paper we present a novel algorithm using active search, which is able to improve the performance and significantly reduce the time cost for using superpixels to over-segment an image. Taking advantage of local continuity, our algorithm provides results with good boundary sensitivity even for complex and low contrast image. Moreover, it is able to converge in only two iterations, achieving the lowest time cost compared to previous methods while obtaining performance comparable to the state-of-the-art method ERS with 1/20th of its running time. We have used various evaluation metrics on the Berkeley segmentation benchmark dataset to demonstrate the high efficiency and high performance of our approach.

Acknowledgments

This research was sponsored by NSFC (61620106008, 61572264), CAST (YESS20150117), Huawei Innovation Research Program (HIRP), and IBM Global SUR award.

References

Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; and Süsstrunk, S. 2012. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. PAMI* 34(11):2274–2282.

Arbelaez, P.; Maire, M.; Fowlkes, C.; and Malik, J. 2011. Contour detection and hierarchical image segmentation. *IEEE Trans. PAMI* 33(5):898–916.

Barnes, C.; Shechtman, E.; Finkelstein, A.; and Goldman, D. 2009. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG* 28(3):24.

Boykov, Y., and Kolmogorov, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. PAMI* 26(9):1124–1137.

Boykov, Y.; Veksler, O.; and Zabih, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. PAMI* 23(11):1222–1239.

Cheng, M.-M.; Liu, Y.; Hou, Q.; Bian, J.; Torr, P.; Hu, S.-M.; and Tu, Z. 2016. HFS: Hierarchical feature selection for efficient image segmentation. In *ECCV*, 867–882. Springer.

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

Du, Q.; Faber, V.; and Gunzburger, M. 1999. Centroidal Voronoi tessellations: applications and algorithms. *SIAM review* 41(4):637–676.

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

Hoiem, D.; Efros, A. A.; and Hebert, M. 2005. Automatic photo pop-up. *ACM transactions on graphics (TOG)* 24(3):577–584.

Kolmogorov, V., and Zabin, R. 2004. What energy functions can be minimized via graph cuts? *IEEE Trans. PAMI* 26(2):147–159.

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

Liu, M.-Y.; Tuzel, O.; Ramalingam, S.; and Chellappa, R. 2011. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2097–2104. IEEE.

Liu, Y.-J.; Yu, C.-C.; Yu, M.-J.; and He, Y. 2016. Manifold SLIC: A fast method to compute content-sensitive superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 651–659.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28(2):129–137.

Neubert, P., and Protzel, P. 2012. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, 1–12.

Osher, S., and Sethian, J. A. 1988. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics* 79(1):12–49.

Peyré, G.; Péchaud, M.; Keriven, R.; and Cohen, L. D. 2010. Geodesic methods in computer vision and graphics. *Foundations and Trends® in Computer Graphics and Vision* 5(3–4):197–397.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. PAMI* 22(8):888–905.

Stutz, D.; Hermans, A.; and Leibe, B. 2014. Superpixel segmentation using depth information. *RWTH Aachen University, Aachen, Germany*.

Van den Bergh, M.; Boix, X.; Roig, G.; de Capitani, B.; and Van Gool, L. 2012. Seeds: Superpixels extracted via energy-driven sampling. In *European conference on computer vision*, 13–26. Springer.

Veksler, O.; Boykov, Y.; and Mehrani, P. 2010. Superpixels and supervoxels in an energy optimization framework. In *European conference on Computer vision*, 211–224. Springer.

Vincent, L., and Soille, P. 1991. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. PAMI* 13(6):583–598.

Wang, S.; Lu, H.; Yang, F.; and Yang, M.-H. 2011. Superpixel tracking. In *2011 International Conference on Computer Vision*, 1323–1330. IEEE.

Wang, P.; Zeng, G.; Gan, R.; Wang, J.; and Zha, H. 2013a. Structure-sensitive superpixels via geodesic distance. *International journal of computer vision* 103(1):1–21.

Wang, Z.; Feng, J.; Yan, S.; and Xi, H. 2013b. Image classification via object-aware holistic superpixel selection. *IEEE Transactions on Image Processing* 22(11):4341–4352.