Computational Visual Media
https://doi.org/10.1007/s41095-018-0123-y Vol. 4, No. 4, December 2018, 333–348

Research Article

FLIC: Fast linear iterative clustering with active search

Jiaxing Zhao[1], Ren Bo[1] (✉), Qibin Hou[1], Ming-Ming Cheng[1], and Paul Rosin[2]

Abstract In this paper, we reconsider the clustering problem for image over-segmentation from a new perspective. We propose a novel search algorithm called "active search" which explicitly considers neighbor 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 methods, such as simple linear iterative clustering (SLIC) and its variants, which use fixed search regions and perform the assignment and the update steps separately, our novel scheme reduces the number of iterations required for convergence, and also provides better boundaries in the over-segmentation results. Extensive evaluation using the Berkeley segmentation benchmark verifies that our method outperforms competing methods under various evaluation metrics. In particular, our method is fastest, achieving approximately 30 fps for a 481×321 image on a single CPU core. To facilitate further research, our code is made publicly available.

Keywords image over-segmentation; SLIC; neighbor continuity; back-and-forth traversal

1 Introduction

Superpixels, generated by image over-segmentation, can take the place of pixels as the fundamental units in various computer vision tasks, including image segmentation [1], image classification [2], 3D reconstruction [3], object tracking [4], etc. Such a technique can greatly reduce computational costs, avoid under-segmentation, and reduce the influence of noise. Clearly, efficiently generating superpixels plays an important role in many vision and image processing applications.

Many classical methods have been developed for superpixel generation, including FH [5], Mean Shift [6], and Watershed [7] methods. The lack of compactness and the irregularity of the resulting superpixels restrict their application, especially when contrast is poor or shadows are present. To solve the above-mentioned problems, Shi and Malik proposed the normalized cuts (NC) method [8] which generates compact superpixels. However, this method does not follow image boundaries very well, and the computational cost is high. The GraphCut method [9, 10] regards the segmentation problem as an energy optimization process. It solves the compactness problem by using min-cut/max-flow algorithms [11, 12], but their parameters are hard to control. The Turbopixel method [13] provides another approach to solving the compactness problem. However, the inefficiency of the underlying level-set method [14] restricts its application. Van den Bergh et al. [15] proposed an energy-driven algorithm, SEEDS, whose results follow image boundaries well, but unfortunately it suffers from irregularity and the number of superpixels output is hard to determine. The ERS method [16] performs well on the Berkeley segmentation benchmark, but has a high computational cost that limits its practical use.

Achanta et al. [17] proposed a linear clustering based algorithm called SLIC; it generates superpixels based on Lloyd's algorithm [18] (also known as Voronoi iteration or the k-means method). For speed, in the assignment step of SLIC, each pixel p is associated with those cluster seeds whose search regions overlap its location. This strategy is also adopted by most subsequent works based on SLIC.

1 Nankai University, Tianjin 300350, China. E-mail: J. Zhao, zhaojiaxing@mail.nankai.edu.cn; R. Bo, 015158@nankai.edu.cn (✉); Q. Hou, andrewhoux@gmail.com; M.-M. Cheng, cmm@nankai.edu.cn.

2 Cardiff University, Wales, United Kingdom. E-mail: rosinpl@cardiff.ac.uk.

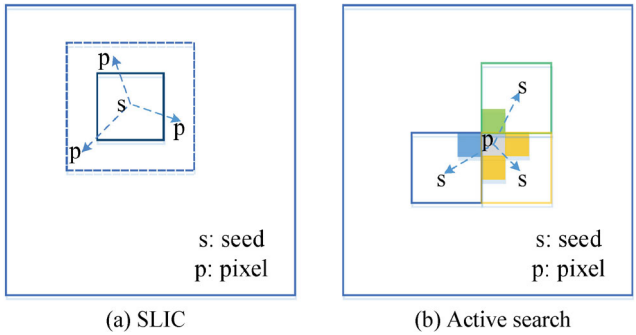
Manuscript received: 2018-05-15; accepted: 2018-08-31

SLIC is widely used in various applications [4] because of its high efficiency and good performance. Inspired by SLIC, Wang et al. [19] implemented an algorithm called SSS that considers the structural information within images. It uses geodesic distance [20] computed by geometric flows instead of the simple Euclidean distance. However, its efficiency is poor because of the high computational cost of measuring geodesic distances. Very recently, Liu et al. [21] proposed the Manifold SLIC method that generates content-sensitive superpixels by computing a centroidal Voronoi tessellation (CVT) [22] in a special feature space. This advanced technique is much faster than SSS but still slower than SLIC due to the cost of its mapping, splitting, and merging processes. In summary, the above-mentioned methods improve the results by either using more complicated distance measurements or by providing more suitable transformations of the feature space. However, the assignment and update steps within these methods are performed separately, leading to a low convergence rate.

2 Proposed approach

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

- Good awareness of neighboring-pixel continuity, leading to results with good boundary sensitivity regardless of image complexity and contrast.
- Joint performance of the assignment and update steps, providing rapid convergence (in just two iterations). Our method has the highest speed of any superpixel segmentation approach, with better performance according to a variety of metrics evaluated on the Berkeley segmentation benchmark.



(a) SLIC (b) Active search

Fig. 1 (a) Search method used in SLIC. Each seed only searches over a limited region to reduce computational cost. (b) Our active search method. Each pixel decides its own label by searching its surroundings.

3 Preliminaries

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

Let $\{I_i\}_{i=1}^{N}$ be a color image, where I_i represents some pixel. Given a set of evenly distributed seeds $\{S_k\}_{k=1}^{K}$, SLIC simplifies Lloyd's algorithm to get the centroidal Voronoi tessellation (CVT) [22] that will be discussed in Section 3.4. In the assignment step, each pixel I_i is associated with those cluster seeds whose search regions overlap the pixel's location: see Fig. 1(a). The area of a search region is given by $2T \times 2T$, where $T = \sqrt{N/K}$. In detail, SLIC considers I_i to lie in a five dimensional space that includes 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, computed by

$$D(I_i, I_j) = \sqrt{d_{\mathrm{c}}^2 + \left(\frac{d_{\mathrm{s}} m}{N_{\mathrm{s}}}\right)^2} \tag{1}$$

where m is a variable that controls the weight of the spatial term, and $N_{\mathrm{s}} = T$. Variables d_{s} and d_{c} are respectively the spatial and color distances, given by

$$d_{\mathrm{s}} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

and

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

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

centers. Overall, it obtains the over-segmentation results by iteratively performing the assignment and update steps.

Variants of SLIC use a similar approach. They improve upon the performance of SLIC by using better distance measures or more suitable transformation functions between color space and spatial space. However, in these algorithms, each search region is fixed during the assignment step within each loop, and the relationships between neighboring pixels are largely ignored when allocating pixels to superpixels. Separately performing the assignment step and the update step causes delayed incorporation of pixel label changes.

Since superpixel computation is typically used as the first step of other vision applications, the rapid generation of superpixels with good boundaries is a crucial problem. Here, unlike previous algorithms [17, 21], we consider this problem from a new point of view, in which only 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 determination of over-segmentation regions. Moreover, the assignment step and the update step are performed jointly. Very few iterations are required for our approach to converge. An overview of our algorithm is provided in Algorithm 1.

Algorithm 1 FLIC

Input: Image I with N pixels, the desired number of superpixels K, the maximum number of iterations $itr_{\max}$ and the spatial distance weight m.
Output: K superpixels
Divide the image into regular grid cells $\{G_k\}_{k=1}^{K}$ with grid spacing $\upsilon = \sqrt{N/K}$.
Initialize labels $\{L_k\}_{k=1}^{K}$ for pixels according to their locations.
Move each seed to the lowest gradient position in its 3×3 neighborhood.
Initialize seeds $\{S_k\}_{k=1}^{K}$.
Regard pixels sharing the same label as a superpixel ζ.
Initialize distance $d_{(i)} = \infty$ for each pixel and $itr = 0$.
while $itr < itr_{\max}$ **do**
  for each superpixel ζ_k **do**
    Use back-and-forth scan to traverse superpixel ζ_k to determine the pixel processing sequence (§ 3.3).
    for each pixel I_i in the sequence **do**
      Set $d_{(i)} = D(I_i, S_{L_i})$ using Eq. (1)
      for I_j in the four-neighborhood of I_i **do**
        if $L_j \neq L_i$ **then**
          Compute $D = D(I_i, S_{L_j})$ using Eq. (1)
          if $D < d_{(i)}$ **then**
            $d_{(i)} = D$; $L_i = L_j$.
          end if
        end if
      end for
      if L_i was changed to L_j **then**
        Use Eq. (10) to update ζ_{L_i};
        Use Eq. (11) to update ζ_{L_j};
        Update the bounding box of ζ_{L_j} (§ 3.4).
      end if
    end for
  end for
  itr++;
end while

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 [17], the original RGB color space is transformed to the more useful CIELAB color space. 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 $\upsilon = \sqrt{N/K}$ as in Ref. [17], and the initial label for each pixel I_i is assigned as

$$L_i = k, \quad \text{when} \quad I_i \in G_k \tag{5}$$

We initialize the seed S_k in G_k at its centroid. Therefore, S_k can also be defined as being 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 to leverage as much of this a priori information as possible. In our method, unlike most previous approaches [17, 21], the label of each pixel is only determined by its neighbors. We compute the distances between the current pixel and the seeds of its four or eight adjacent pixels—see Fig. 1. Specifically, for a pixel I_i, our assignment principle is

$$L_i = \operatorname*{argmin}_{L_j} D(I_i, S_{L_j}), \quad I_j \in A_i \tag{7}$$

where A_i consists of I_i and its four neighboring pixels, and S_{L_j} is I_j's corresponding superpixel seed. We

use Eq. (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, local pixel continuity has a stronger effect in our 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 clear: firstly, the nearby assignment principle can avoid the occurrence of too many isolated regions, indirectly preserving the desired number of superpixels. Secondly, this assignment operation is not limited by a fixed range in space, resulting in better superpixel boundary compliance even when very complicated content leads to irregular superpixel shapes. Furthermore, during the assignment process, the superpixel centers are also self-adaptively modified, leading to faster convergence. Detailed demonstration and analysis are given in Section 4.5. It is worth mentioning that the neighbors of the internal pixels in a superpixel normally share the same labels, so it is unnecessary to process them further, allowing us to process each superpixel extremely quickly.

3.3 Traversal order

The traversal order plays a very important role in our approach: an appropriate scanning order may lead to a visually better segmentation. As explained in Section 3.2, the label of each pixel only depends on the seeds of its surrounding pixels. Thus, in a superpixel, the label of the current pixel is directly or indirectly related to those pixels that have already been processed. To better take advantage of this avalanche effect, we adopt a back-and-forth traversal order as in PatchMatch [23], in which the pixels that are processed later will benefit from updates to previously processed pixels. Figure 2 illustrates this process. In the forward pass, the label decision for 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.

While an arbitrary superpixel may have an irregular shape instead of a simple rectangle or square, we use a simple strategy to traverse the whole superpixel. We first find a minimum bounding box within which all its pixels are enclosed, as shown in Fig. 2. We then perform the scanning process for all 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

It is common in existing methods, such as SLIC [17], that the assignment and update steps are performed separately, leading to delayed feedback from pixel label changes to superpixel seeds. An obvious problem of such a strategy is that many (normally more than five) iterations are required before convergence. In our approach, based on the assignment principle in Eq. (7), we use a joint assignment and update strategy which performs these two steps at a finer granularity. This approach is able to adjust the superpixel seed center position on the fly, significantly reducing the number of iterations needed for convergence. Since most clustering-based superpixel methods use the centroidal Voronoi tessellation (CVT), we first briefly introduce the CVT 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

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

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

$$\mathcal{V}_{\mathrm{D}}(S) = \{\mathcal{V}_{S_k} \neq \emptyset \mid \forall S_k \in S\} \quad (9)$$

A CVT is then defined as a Voronoi diagram whose generator point of each Voronoi cell is also its center of mass. The 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 Eq. (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, say, 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 accordingly.

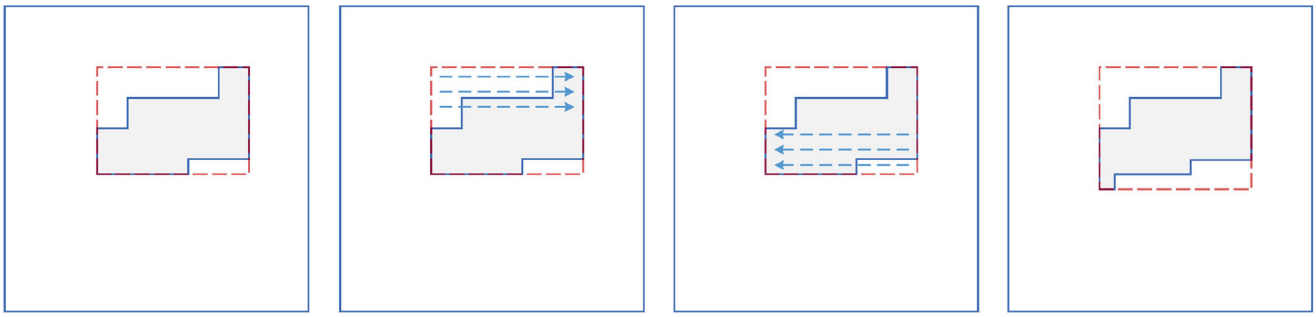


Fig. 2 Scanning order for each superpixel. Gray regions enclosed by blue lines represent superpixels. Red dashed rectangles denote their corresponding bounding boxes. We first scan the bounding box from left to right and top to bottom (b) and then in the opposite direction (c). The shape of each superpixel may change, whereupon we update the bounding box (d) if necessary.

As the above updates only contain very simple arithmetic operations, they can be performed very quickly. Such an immediate update will help later pixels make a better choice during assignment, leading to better convergence. Figure 10 given later illustrates the speed of convergence of our approach.

3.5 Superpixel processing order

In our method, superpixels are processed independently. Thus, the superpixel processing order will affect the performance of the method; label changes affect the surrounding pixels. However, we do not want current pixel changes to affect the superpixels we have already processed. Therefore, we give priority to those complex superpixels in which many pixels labels will be changed, processing them first and then simple superpixels later. Superpixels which have a uniform color are simple, and almost no pixels labels need to change.

We use color entropy to define the complexity of superpixels. We calculate it using the following formulation (15):

$$\text{Entropy}_l = \frac{\sum_{p\in\zeta_{L_j}}(p_l-\hat{p_l})^2}{|\zeta_{L_j}|} \tag{12}$$

$$\text{Entropy}_a = \frac{\sum_{p\in\zeta_{L_j}}(p_a-\hat{p_a})^2}{|\zeta_{L_j}|} \tag{13}$$

$$\text{Entropy}_b = \frac{\sum_{p\in\zeta_{L_j}}(p_b-\hat{p_b})^2}{|\zeta_{L_j}|} \tag{14}$$

$$\text{Entropy}_{\zeta_{L_j}} = \frac{\text{Entropy}_l+\text{Entropy}_a+\text{Entropy}_b}{3} \tag{15}$$

where $|\zeta_{L_i}|$ is the number of pixels in superpixel ζ_{L_i}, p is a three-dimensional vector in lab color space, and $\hat{p}$ is the mean of p in superpixel ζ_{L_i}.

After determining the entropy of all superpixels, we process them in descending order of entropy. We later demonstrate the benefits of this processing order.

4 Experiments

Our method has been implemented in C++ on a PC with a 4.0 GHz Intel Core i7-4790K CPU with 32 GB RAM, and 64 bit operating system. We compare our method to various previous and state-of-the-art works, including FH [5], SLIC [17], Manifold SLIC [21], SEEDS [15], and ERS [16], using the BSDS500 benchmark and the evaluation methods proposed in Refs. [24, 25]. As the source codes used in evaluation of the above works may not be the same as the reported versions, we may observe performance differences from the original during evaluation. For fair comparison, we uniformly use publicly available source code [24, 25] for all methods. As in previous research in the literature [19, 21], we mainly evaluate all algorithms on 200 randomly selected images of resolution 481×321 from the Berkeley dataset [24]. In addition, we compared our method with other state-of-the-art methods on the Pascal Context dataset [26].

4.1 Datasets

The Berkeley dataset is the most common dataset in the area of image segmentation and boundary detection. The Pascal Context dataset includes a set of additional annotations to the PASCAL VOC 2010 dataset [27]. It goes beyond the original PASCAL semantic segmentation task by providing annotations for the whole scene and is mainly intended for semantic segmentation and object detection. Hence, we only perform an ablation study on the Berkeley dataset and compare our method with other methods on both datasets. We randomly

selected six images and corresponding ground-truths from the BSDS (Fig. 3) and Pascal Context datasets (Fig. 4).

As we can see, the images in the Pascal Context are more realistic, while the annotations for the BSDS dataset are more detailed.

4.2 Parameters

Our approach requires three parameters to be set. The first is the number of superpixels K. One of the common advantages of clustering-based algorithms is that the desired number of superpixels can be directly chosen by setting the clustering parameter K. The second parameter is the spatial distance weight m. It has a large effect on the smoothness and compactness of superpixels. We show that performance increases as m decreases. However, making m too small can also lead to irregularity of superpixels. To achieve a good trade-off between compactness and speed, in the following experiments, we set $m = 5$ as default. The last parameter is the maximum number of iterations itr. Here we set $itr = 2$ by default to balance time taken and result quality. We emphasise that, to make a fair comparison, we optimize the parameters for each method to maximize its recall value for the BSDS500 benchmark.

Fig. 3 Images and ground-truths for the BSDS dataset. Rows 1,3: original images. Rows 2,4: corresponding annotated edges.

Fig. 4 Images and ground-truths for the Pascal Context dataset. Rows 1,3: original images. Rows 2,4: corresponding annotated edges.

4.3 Comparison using BSDS dataset

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.

4.3.1 Boundary recall

Boundary recall (BR) [17] is a measurement which assesses how well superpixel boundaries represent ground-truth boundaries. It computes the fraction of ground-truth edges that fall within ε-pixel length from at least one superpixel boundary. BR can be computed by

$$BR_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_{p\in\xi_{\mathcal{G}}} \Pi(\min_{q\in\xi_{\mathcal{S}}} \|p-q\| < \varepsilon)}{|\xi_{\mathcal{G}}|} \quad (16)$$

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 is within ε distance. Here we

follow Refs. [17, 21] and set $\varepsilon = 2$ in our experiment. The boundary recall curves for different methods are plotted in Fig. 5(a). One can easily observe that our FLIC method outperforms all other methods.

4.3.2 Undersegment error

The undersegment error (UE) [28] reflects the extent to which superpixels do not exactly overlap the ground-truth segmentation. As for BR, UE also reflects boundary adherence, but UE uses segmentation regions instead of boundaries in the measurement. Mathematically, UE can be computed by

$$UE_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_{G\in\varrho_{\mathcal{G}}}(\sum_{S:S\cap G\neq\phi}\min(S_{\text{in}}, S_{\text{out}}))}{N} \quad (17)$$

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 overlap between superpixel S and ground-truth segment G, and S_{out} denotes the rest of the superpixel S. As shown in Fig. 5(b), our results are nearly the same as those of the best approach ERS [16], while our method is significantly faster.

4.3.3 Achievable segmentation accuracy

Achievable segmentation accuracy (ASA) [16] gives the highest accuracy achievable for object segmentation that utilizes superpixels as units. As for UE, ASA utilizes segments instead of boundaries. It can be computed by

$$ASA_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_k \max_i |S_k \cap G_i|}{\sum_i G_i} \quad (18)$$

where S_k represents the superpixel and G_i represents the ground-truth segment. A better superpixel segmentation will have a larger ASA value. Figure 5(c) shows that, compared to the ERS method [16], the quality of results provided by our approach is competitive, and our method achieves

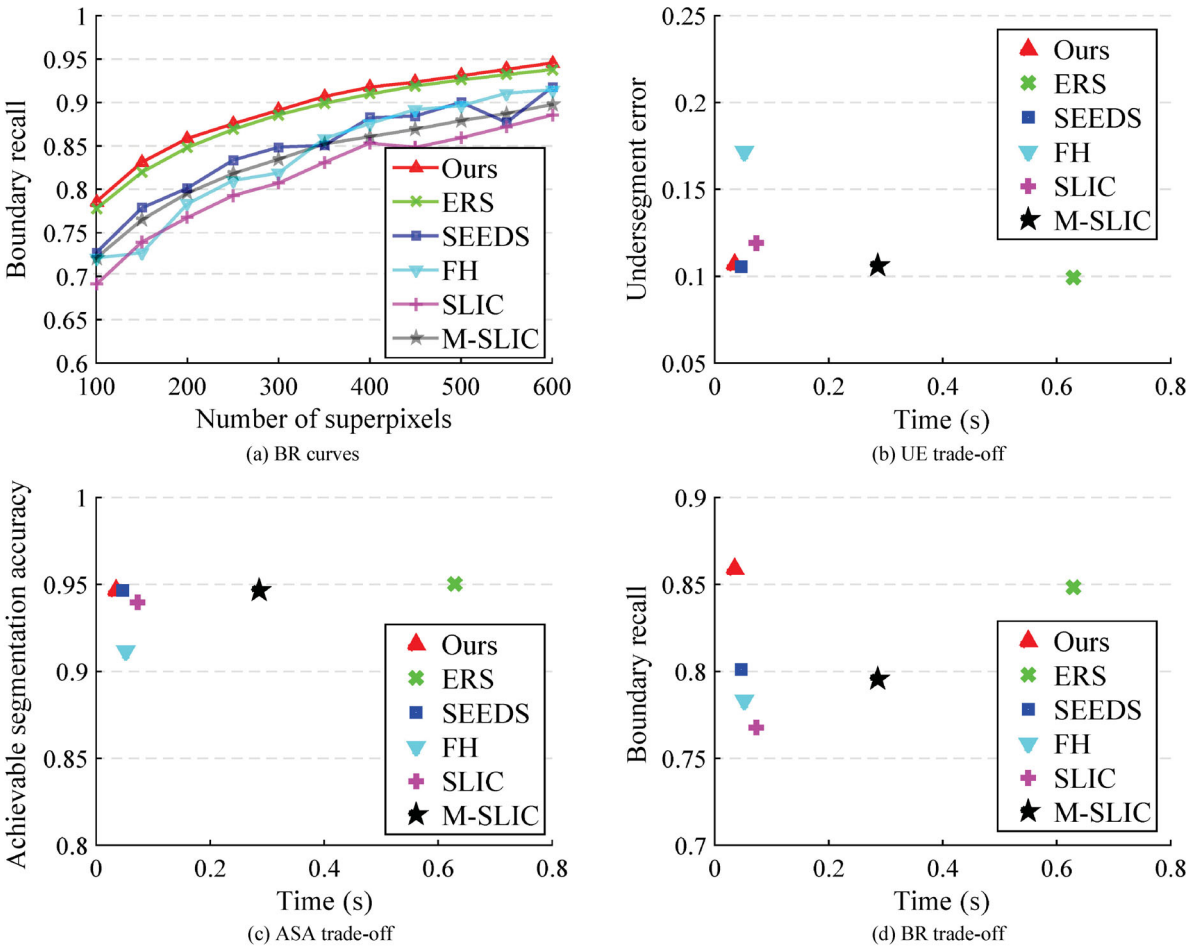


Fig. 5 Comparisons between state-of-the-art methods and our approach on the BSDS500 benchmark. In (b)–(d), K is fixed to 200 for the best trade-off between result quality and speed. In terms of boundary recall, our strategy significantly outperforms methods that take similar time. Furthermore, competitive results are also achieved compared to slower methods (e.g., the state-of-the-art ERS method [16]) according to all evaluation metrics, but at an order of magnitude faster speed.

the best trade-off between quality and speed.

4.3.4 Time

Similar to SLIC, our method has $O(N)$ time complexity. Speed is one of the most important issues for using superpixels as elementary units. Many approaches are limited by their speeds, such as SSS [19] and ERS [16]. As shown in Fig. 5, the average time taken by our FLIC method using two iterations to process an image is 0.035 s, while the time needed by ERS, Manifold SLIC, SLIC, and FH is 0.625 s, 0.281 s, 0.072 s, and 0.047 s, respectively. FLIC has the lowest time cost of all these methods and is nearly 20 times faster than ERS whilst providing comparable result quality.

4.3.5 Visual results and analysis

Figure 6 show several superpixel segmentation results using different algorithms. It can be seen that our approach is more sensitive to image boundaries, especially when there is poor contrast between the foreground and background. Compared to the SLIC method, our approach follows boundaries very well and runs twice as quickly. Compared to the ERS method, our superpixels are much more regular and the mean execution speed of our approach is 20 times greater.

The above facts and Fig. 5 show that our approach achieves an excellent compromise between *boundary adherence*, *compactness*, and *time*.

4.4 Comparison using Pascal Context dataset

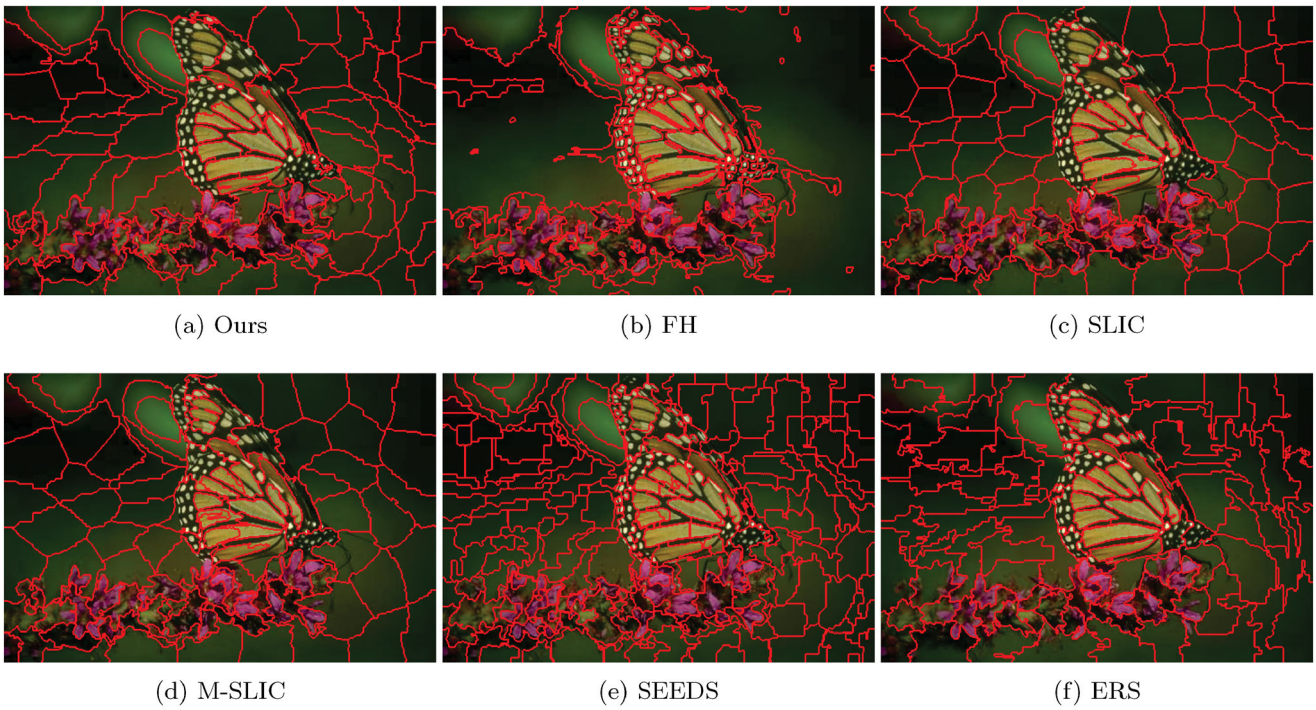
Additional, We evaluate the over-segmentation methods on Pascal Context dataset, which is larger and more realistic. We use the same metrics to evaluate the over-segmentation method.

4.4.1 Boundary recall

The boundary recall curves for different methods are plotted in Fig. 7(a). We can see that almost all over-segmentation methods achieve quite high recall values. Our method is no longer the best but still achieves competitive performance. Figure 7(d) shows that our method achieves 0.974 in terms of boundary recall while the state-of-the-art method M-SLIC [21] achieves 0.978. However, our method runs ten times faster than M-SLIC. We think this difference may be because that there are fewer edges in the ground-truth in the Pascal Context dataset.

4.4.2 Undersegment error

Figure 7(b) shows that our results are nearly the same as those of the best approach, SEEDS [15], but run faster.



(a) Ours (b) FH (c) SLIC

(d) M-SLIC (e) SEEDS (f) ERS

Fig. 6 Visual comparison of superpixel segmentation results using various algorithms, with 100 superpixels and $m = 10$. Our approach follows boundaries very well and at the same time produces compact superpixels.

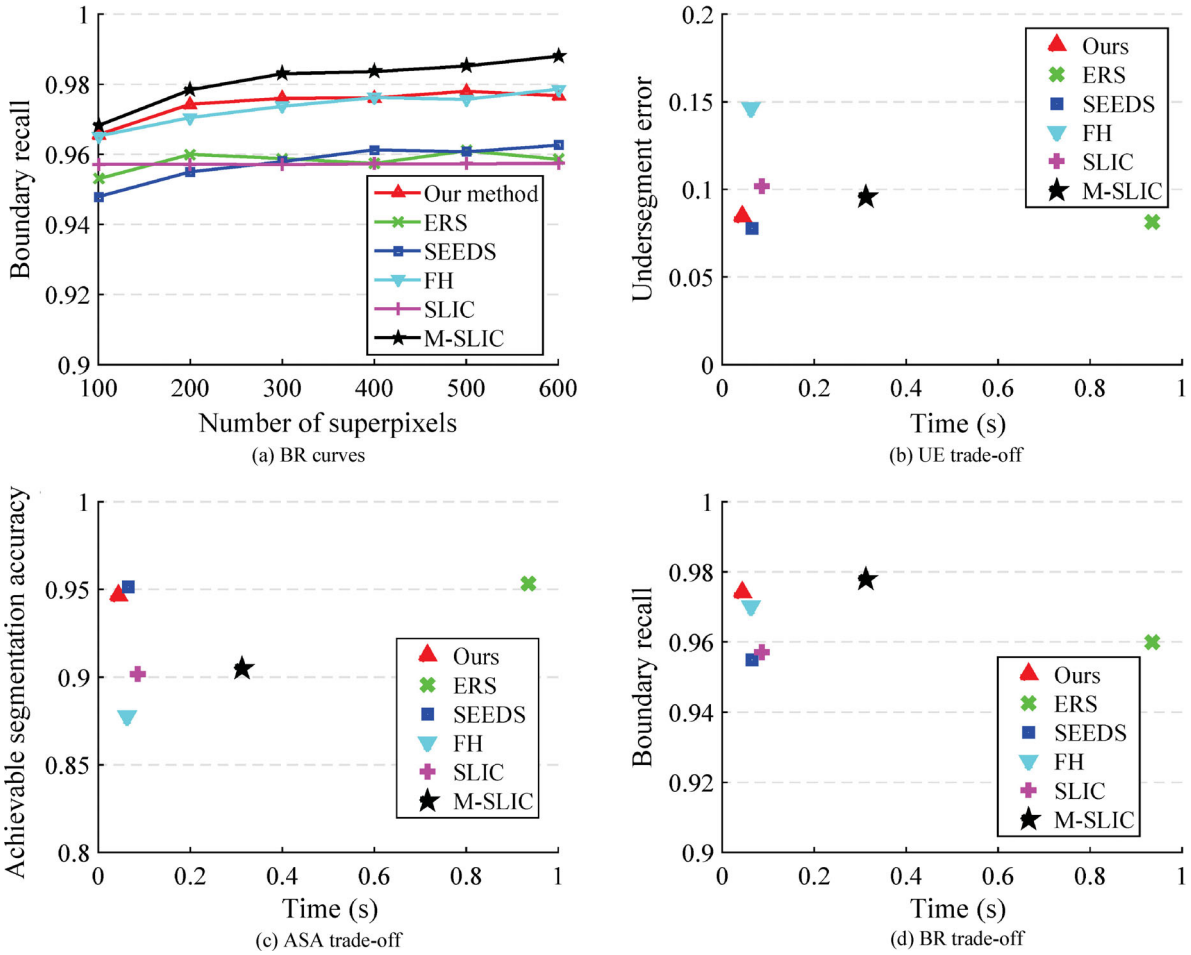


Fig. 7 Comparisons between state-of-the-art methods and our approach on the Pascal Context dataset. Due to the simplicity of this dataset, almost all over-segmentation methods yield good boundary recall. The proposed method achieves a competitive trade-off on this dataset.

4.4.3 Achievable segmentation accuracy

Figure 7(c) shows that, compared to other over-segmentation methods, our approach provides competitive result quality, giving the best trade-off between quality and time.

4.4.4 Time

The size of images in the BSDS dataset is fixed, either 480 ×320 or 320 × 480, while in the Pascal Context dataset, the image size is arbitrary. Figure 7 shows that our proposed method still takes least time amongst all over-segmentation methods.

4.5 Algorithm analysis

4.5.1 Efficacy of back-and-forth traversal

As shown in Fig. 2, we adopt a back-and-forth traversal order to scan the whole enclosing bounding box for each superpixel. Actually, using two forward scans can also perform very well for our method. Figure 8 provides a quantitative comparison between two strategies: using four iterations of purely forward scanning, and using the proposed back-and-forth scan order twice (which also results in four iterations). The blue line indicates results using normal forward scan order while the red line indicates results using our method. The red curve significantly outperforms the blue curve while taking similar time: our back-and-forth scan order considers more information about the regions outside the bounding box, leading to more reliable boundaries.

4.5.2 Role of spatial distance weight

Figure 9(a) shows that, unlike for SLIC [17], the BR curve monotonically decreases with respect to the spatial distance weight m in our approach. This is because 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 results in less regular superpixels, so we choose $m = 5$ for our comparison

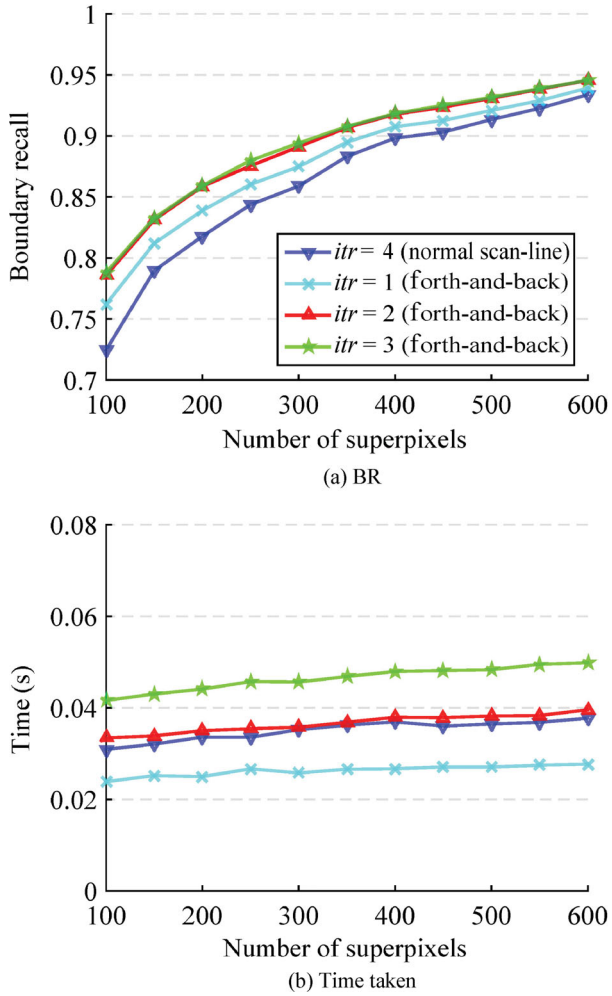


Fig. 8 Partial sensitivity analysis for standard evaluation metrics and time taken.

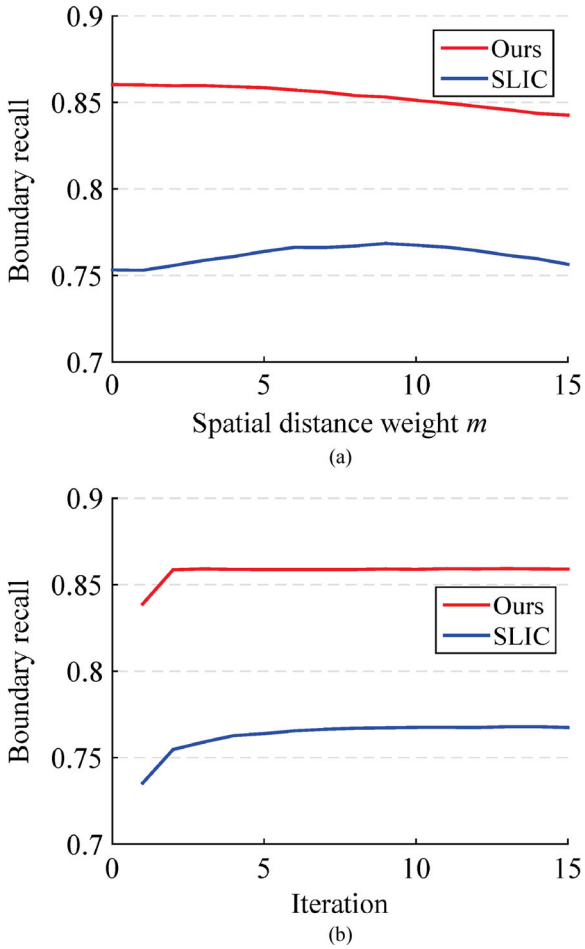


Fig. 9 (a) BR–m curves; m is the spatial distance weight in Eq. (1). Our results are far better than those from SLIC for all values of m. (b) BR–iteration curves. Our method converges within 2 iterations, much fewer than SLIC.

with previous works. Superpixels are normally used as a first step in vision tasks and these vision tasks often favor superpixel methods with good boundaries. Our approach allows users to select a reasonable value for m according to their specific requirements. In any case, our overall performance is significantly better for all m values.

4.5.3 Convergence rate

FLIC significantly accelerates the evolution so that we only need a few iterations before convergence. We compare the result quality for different numbers of iterations using the Berkeley benchmark. It can be easily seen from Fig. 9(b) that our algorithm quickly converges after two iterations and further iterations only bring marginal benefits to the results. For example, when K is set to 200, the boundary recall of the superpixels with only one iteration is 0.835; after two iterations it is 0.859 and after three iterations it is 0.860. The undersegment error values are 0.115, 0.108, and 0.107, respectively, while the achievable segmentation accuracy values are 0.941, 0.945, and 0.946, respectively. As can be seen in Fig. 9(b), our algorithm not only converges much more quickly than SLIC (which requires ten iterations to converge), but also obtains better quality results.

4.5.4 Role of joint assignment and update

Our algorithm jointly performs the assignment and update steps. Figure 10 show the convergence rates for both our joint approach and for separately performing assignment and update steps. Clearly, our joint approach converges very quickly as only two iterations are needed, while the separate approach needs another two iterations to reach the same BR value. This demonstrates that our joint approach is

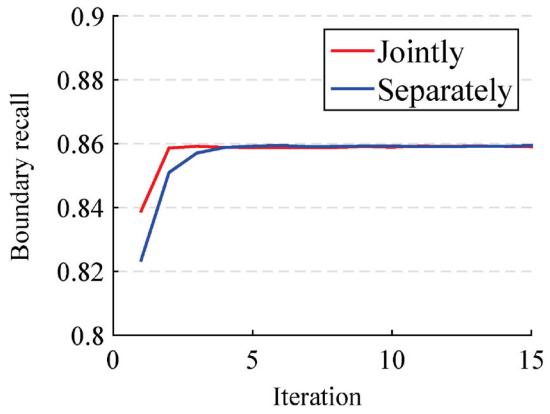


Fig. 10 Comparison of convergence rate for joint and separate assignment and update steps.

efficient while having no negative effect on the final results.

4.5.5 Size of neighborhoods

In our method, the label of the current pixel relies on its four neighboring pixels. Using eight neighborhoods is also plausible, as more neighbors should definitely further more useful information. In Table 1, we briefly compare the results for these two approaches. As might be expected, larger neighborhoods lead to an increase in result quality but at the cost of reducing speed. In real applications, users can select either approach to suit their own requirements.

4.5.6 Traversal order

In our approach, we adopt the same traversal order as used in PatchMatch [23]. Actually, there are many other possible traversal orders. Below we list a few simple ones:

- In PatchMatch, the horizontal axis is the major axis. Instead, we could first scan the vertical axis: forward scan first from top to bottom and then from left to right. Backward scan first from bottom to top and then from right to left.
- In PatchMatch, superpixels are scanned from the top left corner to the right bottom corner. Instead we may scan the superpixels from right bottom to top left.
- We also could combine the horizontal and vertical scans: first adopt the default traversal method, a horizontal scan, and then use a vertical scan, and so on. In our setting, we set the number of iterations to ten, with five horizontal and five vertical scans.

We compare results using different traversal orders in Table 2. The default traversal order achieves the best results amongst all possible traversal orders, but the difference is not great. In addition, the default traversal order is most straightforward to implement, hence we adopt it, as does PatchMatch [23].

4.5.7 Superpixel processing order

We further compared the results with and without sorting superpixels according to their entropies. If we process the superpixels in descending order of entropy, we get the results mentioned in previous experiments, with a boundary recall value of 85.9. However, the boundary recall is 84.8 if we process the superpixels according to their spatial order. As we can see, we get a minor improvement by sorting.

4.5.8 Qualitative results

Figure 11 shows some segmentation results for the BSDS dataset produced by our approach with $m = 20$ and the number of superpixels set to 1000, 400, and 200, respectively. It is seen that, in each case, the edges of the resulting superpixels are always very close to the boundaries. This is especially obvious in the first and third images. We also show some segmentation results for different values of m in Fig. 12. When m is smaller, 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.

In Figs. 13 and 14 we show segmentation results for the Pascal Context dataset. Most images in the Pascal Context dataset come from real scenes, and

Table 1 Boundary recall versus time for 4-neighborhoods and 8-neighborhoods with different superpixel counts: 100, 200, 300, 400

	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 2 Boundary recall for different traversal orders. TO1 represents the default traversal order which is first from left to right and then from top to bottom. TO2 represents traversing the pixels in a superpixel first from top to bottom and then from left to right. TO3 represents traversing the pixels first from right to left and then from bottom to top. TO4 represents the combined scheme

	100	200	300	400	500	600
TO1	78.6	85.9	89.1	91.8	93.1	94.6
TO2	77.6	85.1	88.5	91.4	92.8	94.3
TO3	77.9	85.3	88.8	91.4	92.9	94.4
TO4	78.1	85.3	88.7	91.5	93.2	94.4

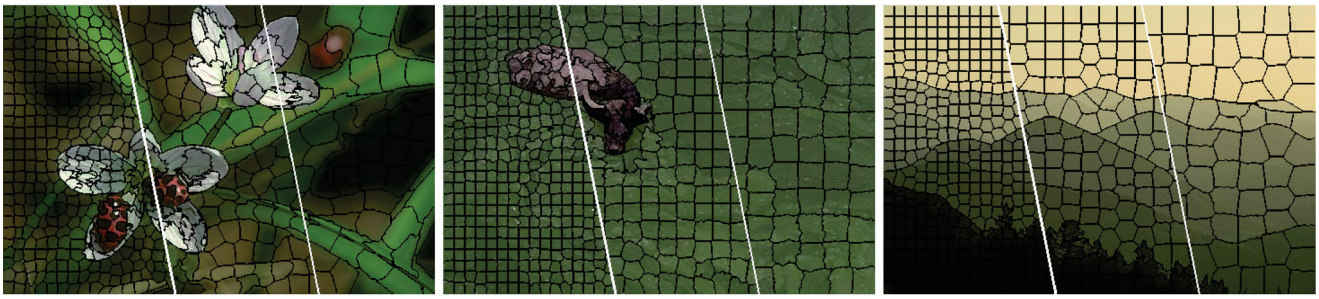


Fig. 11 Images from the Pascal Context dataset segmented by our approach with $m = 20$ and the number of superpixels set to 1000, 400, and 200, respectively. The resulting superpixels follow region boundaries very well.

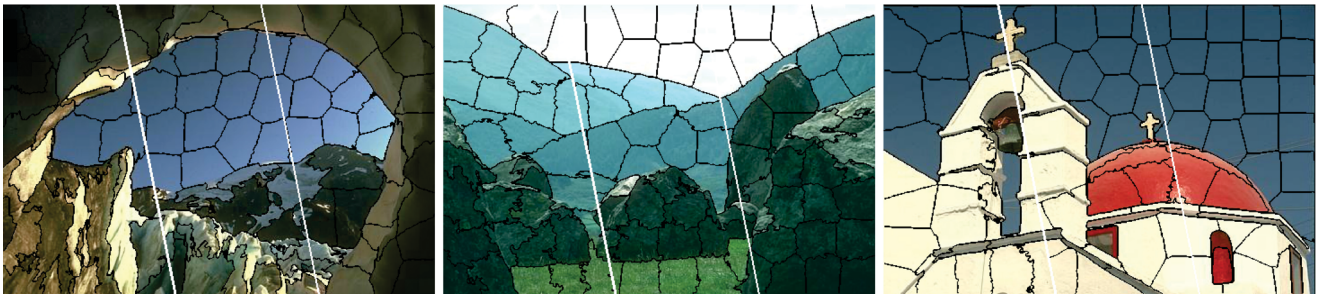


Fig. 12 Images from the BSDS dataset segmented by our proposed approach with $m = 10$, 20, and 30, respectively. When m is smaller, the superpixels follow boundaries well. When m is larger, the superpixels are more compact.

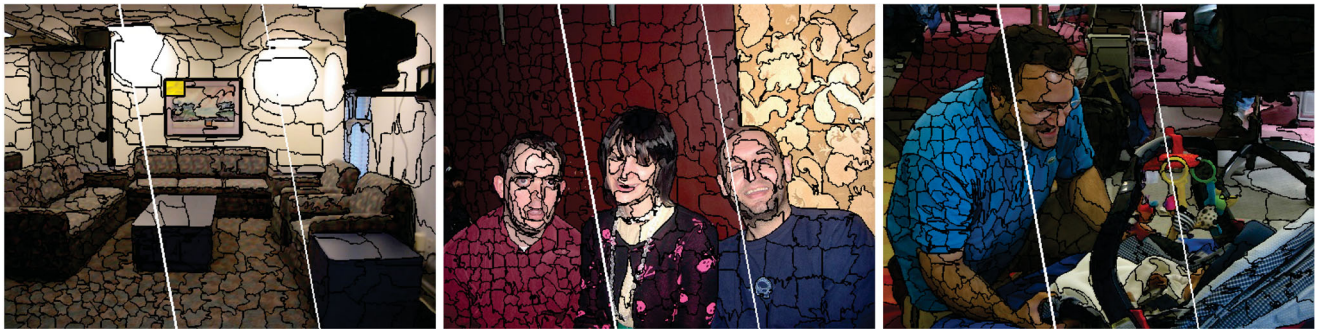


Fig. 13 Images from the BSDS dataset segmented by our approach with $m = 20$ and the number of superpixels set to 1000, 400, and 200, respectively. The resulting superpixels follow region boundaries very well.

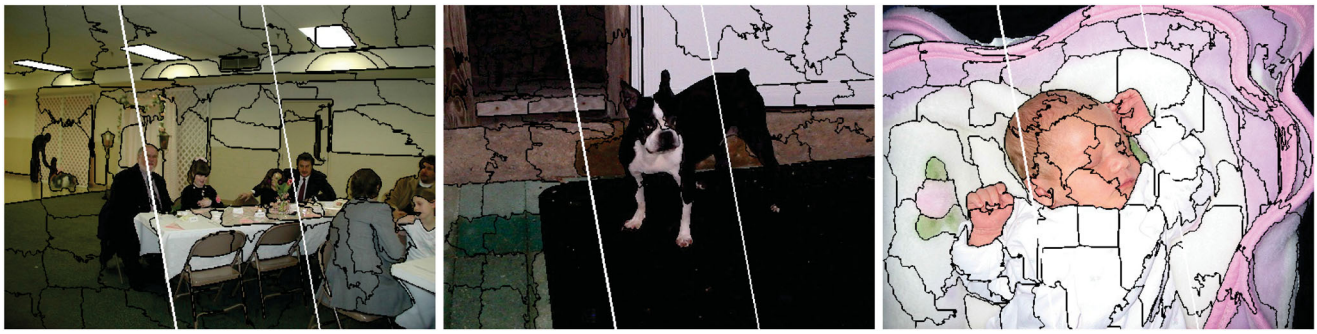


Fig. 14 Images from the Pascal Context dataset segmented by our proposed approach with $m = 10$, 20, and 30, respectively. When m is smaller, the superpixels follow boundaries well. When m is larger, the superpixels are more compact.

are more complex than those in the BSDS dataset. As a result, the segmentation results are more cluttered.

5 Applications

In this section, we applied our method to a higher level application image segmentation problem [1]. In hfs [1], image over-segmentation is used as an initial step: its results are taken as the input to the algorithm to generate a final segmentation result. In the original method, SLIC [17] is used to generate the image over-segmentation result. Here we simply replace SLIC with our proposed method and use the same image segmentation evaluation metric. However, we only use the optimal dataset scale (ODS) during training as the proposed method has few parameters. We use the F-measure of precision and recall on the whole dataset to evaluate the boundary performance; the following evaluation metrics are used to assess region performance:

- Variation of Information (VI), which measures the distance between ground-truth and the segmentation result. Lower VI is better.
- Probabilistic Rand Index (PRI), which measures the pairwise compatibility of element assignment between ground-truth and the proposed segmentation. A large value is better.
- Segmentation Covering (Covering), which measures the average overlap between ground-truth and the proposed segmentation. Again, a large value is better.

The results can be seen in Table 3. The segmentation results are closer when the number of superpixels is large, e.g., 1000 or 2000: as the number of superpixels increases, the quality of different over-segmentation methods becomes closer. When the number of superpixels is small, e.g., 100, segmentation results using FLIC as the initial step perform better than those using SLIC. Hence, it is reasonable to say that our method is better than SLIC in practical applications.

Table 3 Comparison of segmentation results generated by SLIC and FLIC for different numbers of superpixels. E.g., SLIC-100 indicates use of SLIC to generate 100 superpixels which were then input to a downstream algorithm to achieve the final segmentation result

	F-measure	Covering	PRI	VI
SLIC-100	0.559	0.502	0.711	2.031
FLIC-100	0.573	0.508	0.718	2.024
SLIC-1000	0.638	0.534	0.791	2.135
FLIC-1000	0.635	0.536	0.785	2.128
SLIC-2000	0.645	0.531	0.794	2.187
FLIC-2000	0.642	0.535	0.796	2.134

6 Conclusions

This paper has presented a novel algorithm using active search, which is able to improve the result quality and significantly reduce the time taken when producing 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 images. Moreover, it is able to converge in only two iterations, making it faster than previous methods, providing result quality comparable to the state-of-the-art ERS method in 1/20th of the time. We have used various evaluation metrics on the Berkeley segmentation benchmark dataset to demonstrate the speed and high quality results of our approach.

Acknowledgements

This research was sponsored by National Natural Science Foundation of China (Nos. 61620106008 and 61572264), Huawei Innovation Research Program (HIRP), and IBM Global SUR Award.

References

[1] Cheng, M.-M.; Liu, Y.; Hou, Q.; Bian, J.; Torr, P.; Hu, S.-M.; Tu, Z. HFS: Hierarchical feature selection for efficientimage segmentation. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9907*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 867–882, 2016.

[2] Wang, Z.; Feng, J.; Yan, S.; Xi, H. Image classificationvia object-aware holistic superpixel selection. *IEEE Transactions on Image Processing* Vol. 22, No. 11, 4341–4352, 2013.

[3] Hoiem, D.; Efros, A. A.; Hebert, M. Automatic photopop-up. *ACM Transactions on Graphics* Vol. 24, No. 3, 577–584, 2005.

[4] Wang, S.; Lu, H.; Yang, F.; Yang, M.-H. Superpixeltracking. In: Proceedings of the IEEE International Conference on ComputerVision, 1323–1330, 2011.

[5] Felzenszwalb, P. F.; Huttenlocher, D. P. Efficientgraph-based image segmentation. *International Journal of Computer Vision* Vol. 59, No. 2, 167–181, 2004.

[6] Comaniciu, D.; Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 24, No. 5, 603–619, 2002.

[7] Vincent, L.; Soille, P. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 13, No. 6, 583–598, 1991.

[8] Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 22, No. 8, 888–905, 2000.

[9] Boykov, Y.; Veksler, O.; Zabih, R. Fast approxi-mate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 23, No. 11, 1222–1239, 2001.

[10] Veksler, O.; Boykov, Y.; Mehrani, P. Superpixels and supervoxels in an energy optimization framework. In: *Computer Vision – ECCV 2010. Lecture Notes in Computer Science, Vol. 6315.* Daniilidis, K.; Maragos, P.; Paragios, N. Eds. Springer Berlin Heidelberg, 211–224, 2010.

[11] Boykov, Y.; Kolmogorov, V. An experimental-comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 26, No. 9, 1124–1137, 2004.

[12] Kolmogorov, V.; Zabin, R. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 26, No. 2, 147–159, 2004.

[13] Levinshtein, A.; Stere, A.; Kutulakos, K. N.; Fleet, D. J.; Dickinson, S. J.; Siddiqi, K. TurboPixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 31, No. 12, 2290–2297, 2009.

[14] Osher, S.; Sethian, J. A. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics* Vol. 79, No. 1, 12–49, 1988.

[15] Van den Bergh, M.; Boix, X.; Roig, G.; de Capitani, B.; Van Gool, L. SEEDS: Superpixels extracted via energy-drivensampling. In: *Computer Vision – ECCV 2012. Lecture Notes in Computer Science, Vol. 7578.* Fitzgibbon, A.; Lazebnik, S.; Perona, P.; Sato, Y.; Schmid, C. Eds. Springer Berlin Heidelberg, 13–26, 2012.

[16] Liu, M.-Y.; Tuzel, O.; Ramalingam, S.; Chellappa, R. Entropy rate superpixel segmentation. In: Proceedings of the IEEE Conference on Computer Visionand Pattern Recognition, 2097–2104, 2011.

[17] Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 34, No. 11, 2274–2282, 2012.

[18] Lloyd, S. P. Least squares quantization in PCM. *IEEE Transactions on Information Theory* Vol. 28, No. 2, 129–137, 1982.

[19] Wang, P.; Zeng, G.; Gan, R.; Wang, J.; Zha, H. Structure-sensitive superpixels via geodesic distance. *International Journal of Computer Vision* Vol. 103, No. 1, 1–21, 2013.

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

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

[22] Du, Q.; Faber, V.; Gunzburger, M. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review* Vol. 41, No. 4, 637–676, 1999.

[23] Barnes, C.; Shechtman, E.; Finkelstein, A.; Goldman, D. B. PatchMatch: A randomized correspondence algorithm forstructural image editing. *ACM Transactions on Graphics* Vol. 28, No. 3, Article No. 24, 2009.

[24] Arbelaez, P.; Maire, M.; Fowlkes, C.; Malik, J. Contourdetection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 33, No. 5, 898–916, 2011.

[25] Stutz, D. Superpixelsegmentation: An evaluation. In: *Pattern Recognition. Lecture Notes in Computer Science, Vol. 9358.* Gall, J.; Gehler, P.; Leibe, B. Eds. Springer Cham, 555–562, 2015.

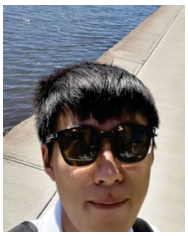
[26] Mottaghi, R.; Chen, X.; Liu, X.; Cho, N.-G.; Lee, S.-W.; Fidler, S.; Urtasun, R.; Yuille, A. The role of contextfor object detection and semantic segmentation in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 891–898, 2014.

[27] Everingham, M.; Van Gool, L.; Williams, C. K. I.; Winn, J.; Zisserman, A. The PASCAL visual object classes challenge 2010 (VOC2010) results. 2010. Available at `http://www.pascalnetwork.org/challenges/VOC/voc2010/workshop/index.html`.

[28] Neubert, P.; Protzel, P. Superpixel benchmark and comparison. In: Proceedings of the Forum Bildverarbeitung, 1–12, 2012.

Jiaxing Zhao is a master student at CCCE&CS, Nankai University (Tianjin, China). He received his bachelor degree from Nankai University in 2017. His research interest includes computer vision and machine learning (especially deep learning).

Bo Ren is a lecturer at CCCE&CS, Nankai University (Tianjin, China). He received his B.S. and Ph.D. degrees from Tsinghua University (Beijing, China) in 2010 and 2015, respectively. His main research interest is in physically-based simulation and rendering, and geometry for computer graphics.



Qibin Hou is at present a first-year Ph.D. student at CCCE&CS, Nankai University (Tianjin, China). Before joining in the media group at Nankai University, he was a machine learning engineer in Baidu. His research interests include low-level vision, deep learning, and multimedia applications.

Ming-Ming Cheng is a professor with College of Computer Science, Nankai University, leading the Media Computing Lab. He received his Ph.D. degree from Tsinghua University in 2012. Then he worked as a research fellow for 2 years, working with Prof. Philip Torr in Oxford. Dr. Cheng's research primarily centers on algorithmic issues in image understanding and processing, including image segmentation, editing, retrieval, etc. He has published over 30 papers in leading journals and conferences, such as IEEE TPAMI, ACM TOG, ACM SIGGRAPH, IEEE CVPR, and IEEE ICCV. He has designed a series of popular methods and novel systems, indicated by 9000+ paper citations (2000+ citations to his first author paper on salient object detection). He received several research awards, including ACM China Rising Star Award, the IBM Global SUR award, and the CCF-Intel Young Faculty Award. His work has been reported by several famous international media, such as BBC, UK Telegraph, Der Spiegel, and Huffington Post.

Paul Rosin gained his B.S. degree in computer science and microprocessor systems in 1984 from Strathclyde University, Glasgow, and Ph.D. degree in information engineering from City University, London, in 1988. He was a research fellow at City University, developing a prototype system for the home office to detect and classify intruders in image sequences. He worked on the Alvey project "Model-Based Interpretation of Radiological Images" at Guy's Hospital, London, before becoming a lecturer at the Department of Computer Science, Curtin University of Technology, Perth, Australia. He moved to Italy to work at the Institute for Remote Sensing Applications at the Joint Research Centre, followed by a return to the UK, becoming lecturer at the Department of Information Systems and Computing, Brunel University, London. In 2000, he moved to the School of Computer Science & Informatics, Cardiff University.