An Accelerated Superpixel Generation Algorithm Based on 4-Labeled-Neighbors

Hongwei Feng, Fang Xiao, Qirong Bu(✉), Feihong Liu, Lei Cui, and Jun Feng(✉)

School of Information and Technology, Northwest University, Xi'an 710127, China
fenghw@foxmail.com, {boqirong,fengjun}@nwu.edu.cn, {xiaofang,child,chrislei}@stumail.nwu.edu.cn

Abstract. Superpixels are perceptually meaningful atomic regions that could effectively improve efficiency of subsequent image processing tasks. Simple linear iterative clustering (SLIC) has been widely used for superpixel calculation due to outstanding performance. In this paper, we propose an accelerated SLIC superpixel generation algorithm using 4-labeled neighbor pixels called 4L-SLIC. The main idea of 4L-SLIC is that the labels are assigned to a portion of the pixels while the others that associated with certain cluster are restrained by adjacent four labeled pixels. In this way, the average number of distance calculated times of pixels are effectively reduced and the similarity between adjacent pixels ensures a better segmentation effect. The experimental results confirm that 4L-SLIC achieved a speed up of 25%–30% without declining accuracy sharply compared to SLIC. In contrast to the method published on CVIU 2016, 4L-SLIC has an acceptable increase in the cost of time, in the mean time, there is a significant ascension to the accuracy of the segmentation.

Keywords: Superpixels · SLIC · Acceleration · 4-labeled neighbors

1 Introduction

Superpixel algorithms [1] can effectively extract the features of perceptually meaningful regions in images. Simple linear iterative clustering (SLIC) [2] is the most popular superpixel framework. Follow the pipeline of SLIC, there are many state-of-the-art superpixel algorithms which are widely used in various computer vision applications, such as image segmentation [1], object recognition [3], motion segmentation [4], 3D reconstruction [5] and so on.

However, the superpixel algorithms can't be applied to some tasks since the time consumption is very high. In order to satisfy the requirement of real-time processing, Choi and Oh proposed an accelerative strategy by introducing 2-labeled neighbors (2L-SLIC) verification which can reduce the processing time as less as a half [6]. But this method suffers from the accuracy since all four neighbors are similar to their central pixel with equal probability, and the usage of only 2-labeled neighbors may lose half part of information.

In this paper, we propose an accelerated version of superpixel algorithm in SLIC framework based on 4-labeled neighbors which is called 4L-SLIC. Using the similarity in neighbors, 4L-SLIC only searches the half pixels directly, and neglects the remaining half which labels are estimated by their neighbors. In this way, 4L-SLIC reduces the maximum search times near halfly while keeps the accuracy of general SLIC. Our algorithm is verified on the public Berkeley segmentation dataset and benchmark. The experiments shows that the speed of 4L-SLIC is increased almost 25%–30% than that of SLIC and slightly slower than 2L-SLIC. In the mean time, the proposed algorithm has a competitive accuracy with SLIC and is more accurate than 2L-SLIC.

2 A New SLIC Algorithm Based on 4-Labeled Neighbors

In this section, we firstly review the basic SLIC algorithms and analysis of computational redundancy in the algorithm. Afterwards, we propose a novel superpixel algorithm called 4L-SLIC based on 4-labeled neighbors to more efficiently generate superpixels. Through this algorithm, half of the pixels in the image are calculated by a drastically decrease. In the final, we analysis for how much the 4L-SLIC reduces the number of cluster searches has made compared to SLIC.

2.1 The Analysis of Computational Redundancy in SLIC

In general, SLIC can be roughly divided into four steps: initialization, cluster assignment, update and postprocessing. Given an input image T of size $R \times C = N$ and the only default parameter of SLIC is K which represent the number of superpixels.

In the initialization step, K initial cluster centers $f_{C_k} = [l_k, a_k, b_k, x_k, y_k]^T$ are sampled on a regular grid spaced S regions, $S = \sqrt{N/K}$ represent the distance between the adjacent initial clusters. f_{C_k} is a feature vector consisting of both three colors in the CIELAB color space and position. Similarly, a pixel i could be expressed as $f_i = [l_i, a_i, b_i, x_i, y_i]^T$.

In the clustering step, each pixel i is associated with the nearest cluster center whose search region overlaps its location [7]. The size of the search region was set to $2S \times 2S$ around the cluster center like Fig. 1(a). Achanta et al. use $d_5(i, C_k)$ denotes the similarity between the pixel i and cluster C_k in the *labxy* color-image plane space [2], which is defined as :

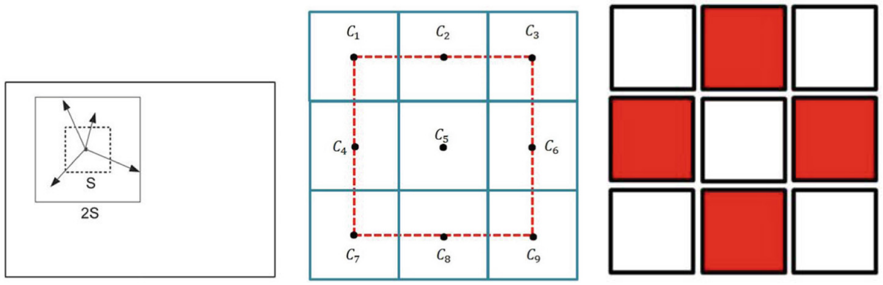
$$d_5(i, C_k) = \sqrt{d_3(i, C_k)^2 + \lambda^2 \cdot d_2(i, C_k)^2} \tag{1}$$

$$d_3(i, C_k) = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \tag{2}$$

$$d_2(i, C_k) = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \tag{3}$$

Which $d_3(i, C_k)$ and $d_2(i, C_k)$ respectively indicate normalized Euclidean distances in color and spatial space. And $\lambda = m/S$, to combine the two distances into a single measure, m represents the normalization factor to control the compactness.

When all the pixels are labeled, the update step is executed. Each f_{C_k} is updated in accordance with their average of pixels in the cluster. The iteration is continued until the residual error E between the cluster center of this iteration and the previous cluster center is less than the threshold T_e.



(a) SLIC searchs a limited region. (b) The coverage of clustering center in SLIC. (c) 4-labeled neighbor pixels.

Fig. 1. Different search regions between SLIC and 4L-SLIC. (Color figure online)

Although SLIC through limiting the size of search region to significantly reduce the number of distance calculations, we found that there still has computational redundancy [8]. For example, In Fig. 1(b), each blue region represents the initial superpixel block, while the red region represents the search range of the $2S \times 2S$ of C_5. For a pixel i belonging to C_5, it is likely to be covered by the search range of many clusters. However, the most extreme case is when i and C_5 are coincide, i will be covered by the search of nine clusters C_1 to C_9. In this case, i requires nine distance calculations. Thus, in SLIC, the number of calculations per pixel is $1 \sim 9$ times. However, the interpixel correlation is not considered in the SLIC algorithm.

Generally, there is a considerable spatial correlation between adjacent pixels, so-called interpixel correlation, which is one of the important characteristic of images that is often overlooked. The interpixel correlation can be expressed as if a pixel i is associated with a certain cluster, then its neighboring pixels have a highly trend of belonging to this cluster. For example, in Fig. 1(c), the block indicate a pixel. If the four red pixels are already labeled, and the middle white pixel are likely to be the same as one of the four red pixels according to the interpixel correlation. If this is done, the number of calculations per pixel will be reduced to a maximum of four times. So our algorithm could increase the speed of generating superpixels.

2.2 SLIC Acceleration with 4-Labeled Neighbor Pixels

In this subsection, we propose a more efficient labeling strategy in the clustering step based on interpixel correlation. In our algorithm, half of the pixels greatly reduced the number of distance calculations, resulting in faster generation of superpixel.

Before describing the proposed algorithm in detail, we first declare a few notations. According to a certain rule, divide all the pixels in the image into two parts as α and β respectively. Let θ_i, ω_i, $L(i)$ and $P(C_k)$ denote a set of cluster centers whose search domains contain i, the set of 8-connected neighbor pixels around i, a label associated with i, and the set of all cluster centers, respectively.

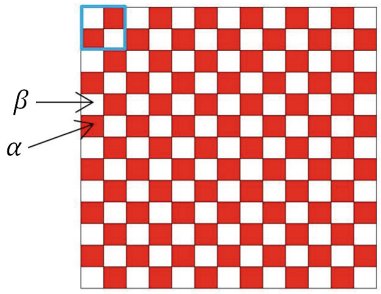
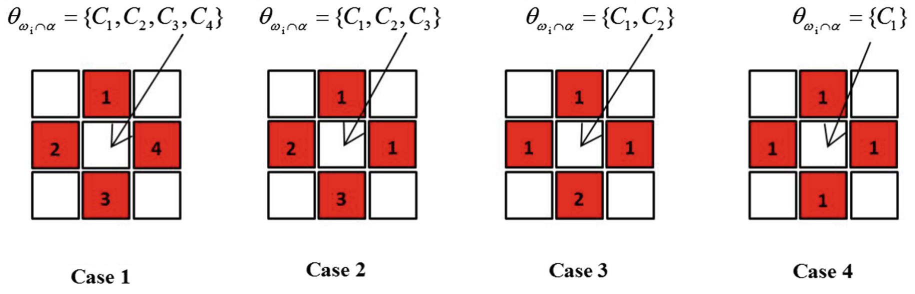


Fig. 2. α:red blocks, β:white blocks, and α:$\beta = 2$:2 for each four blocks e.g. the blue region. (Color figure online)

The process and accelerated theory are described below:

(1) At first, as demonstrated in Fig. 2, where a block indicate a pixel, from both left to right or top to bottom, we subsampled every second pixel from the first pixel as β, and the remaining pixels as α.
(2) Then, compared to the SLIC algorithm which labeled all pixel in the image, our algorithm only label α to their closest clusters by calculating distance of $d_5(i, C_k)$, but at this time β has no label yet. Therefore, except for the first row and column, the last row and column, the others pixels in β have four neighbor pixels that are already labeled. It is noteworthy that the number of



$\theta_{\omega_i \cap \alpha} = \{C_1, C_2, C_3, C_4\}$
1
2
4
3
Case 1
$\theta_{\omega_i \cap \alpha} = \{C_1, C_2, C_3\}$
1
2
1
3
Case 2
$\theta_{\omega_i \cap \alpha} = \{C_1, C_2\}$
1
1
1
2
Case 3
$\theta_{\omega_i \cap \alpha} = \{C_1\}$
1
1
1
1
Case 4

Fig. 3. Case 1: the four-connected pixel labels of i are not all the same. Case 2: three different labels around i. Case 3: only two different labels around i. Case 4: four labels are the same.

label $|L(\omega_i \cap \alpha)|$ is usually less than $|\theta_i|$ in which $i \in \beta$. Because the number of searches for θ_i is usually $1 \sim 9$ times and $|L(\omega_i \cap \alpha)|$ is 4 at most.

(3) Subsequently, the label is assigned to β using 4-labeled neighbor pixels as show in Fig. 3. It is guaranteed that, for each pixel $i \in \beta$, some of its four-connected neighboring pixels have been associated with a cluster. The figures in the red block indicates the label of this pixel. Then, for the in-between white pixel i, the label of its four-connected neighboring pixels $\theta_{\omega_i \cap \alpha}$ can have four different cases. In the Case 1, the four-connected pixels of i have four different labels. The distance metric is only need to calculate four times between i and cluster centers $\{C_1, C_2, C_3, C_4\}$ in which i is assigned to the nearest cluster center. It can be found that the number of distance calculations for each pixel is changed from 9 times in SLIC algorithm to 4 times at most. Likewise, Case 2 and Case 3 only need to calculate distance three times and twice. This has achieved drastically computation reduction. However, Case 4 always occurs inside a superpixel and all the pixels have the identical label in α. In this case, i is associated with the certain cluster without any distance calculation. It is notable that this situation usually occurs and can further significantly reduce execution time.

(4) The first cluster has been completed when all the pixels belonging to β are labeled. Executing the same update and iteration as SLIC until the convergence conditions are satisfied T_e.

2.3 Complexity Analysis of 4L-SLIC

In the previous section, we point out that our algorithm reduces the number of pixels calculation. In this subsection, we analyze how much the 4L-SLIC reduces the number of cluster searches compared to SLIC.

We use mathematical expectation to represent the number of distance calculations per pixel in each algorithm. In SLIC, the cluster search is performed the same number of times as $|\theta_i|$, even $|\theta_i| = 1$ is available. However, in the 2L-SLIC and 4L-SLIC, only one quarter and one half of the pixels are calculated by the same number times as SLIC while the other pixels only need to calculate $0 \sim 4$ times. The expectation of SLIC, 2L-SLIC and 4L-SLIC is simply expressed as Eqs. (4), (5) and (6) respectively :

$$E_{SLIC} = E[|\theta_i|] = \sum_{c=1}^{9} p(|\theta_i| = c) \cdot c \tag{4}$$

$$\begin{aligned} E_{2L-SLIC} &= \frac{|\alpha|}{|N|} E_{SLIC} + \frac{|\beta|}{|N|} E[|\theta_{\omega_i \cap \alpha}|] \\ &= \frac{1}{4} E[|\theta_i|] + \frac{3}{4} E[|\theta_{\omega_i \cap \alpha}|] \\ &= \frac{1}{4} \sum_{c=1}^{9} p(|\theta_i| = c) \cdot c + \frac{3}{4} \sum_{c=2}^{4} p(|\theta_{\omega_i \cap \alpha}| = c) \cdot c \end{aligned} \tag{5}$$

$$
\begin{aligned}
E_{4L-SLIC} &= \frac{|\alpha|}{|N|}E_{SLIC} + \frac{|\beta|}{|N|}E[|\theta_{\omega_i \cap \alpha}|] \\
&= \frac{1}{2}E[|\theta_i|] + \frac{1}{2}E[|\theta_{\omega_i \cap \alpha}|] \\
&= \frac{1}{2}\sum_{c=1}^{9} p(|\theta_i| = c) \cdot c + \frac{1}{2}\sum_{c=2}^{4} p(|\theta_{\omega_i \cap \alpha}| = c) \cdot c
\end{aligned} \tag{6}
$$

Based on these three formulas, we can compute the average number of calculations for each pixel in a different algorithm. The complexity of three algorithms is linear in the number of pixels, which can be expressed as $O(NIE_{SLIC})$, $O(NIE_{2L-SLIC})$, $O(NIE_{4L-SLIC})$ respectively, where I is the number of iterations required for convergence [9]. From the complexity we can seen, the smaller expectation, the more efficient of algorithm.

The proposed 4L-SLIC algorithm is summarized in Algorithm 1.

3 Experimental Results

We implemented 4L-SLIC in Matlab and tested it on a PC with an Intel I5-4590 CPU (3.60 GHz) and 8 GB RAM. We tested with the Berkeley database which containing three-hundred 321×481 images data with ground truth segmentation. Due to the literature [2] has demonstrated that SLIC already overcomed other all conventional algorithms. So in this paper, we mainly compared 4L-SLIC with SLIC and 2L-SLIC algorithm.

3.1 The Number of Pixels in Clustering Calculation

In order to demonstrate how much computing redundancy that 4L-SLIC has reduced, statistics of the calculations number for each pixel in SLIC and 4L-SLIC were summarized at Tables 1 and 2 respectively. Each row showed the ratio of between the number of pixels counted to c and the total number of pixels with K block. The last column E showed the expectation calculated by Eqs. (4) and (6), which representing the average number of calculations for each pixel in the image.

According to the statistics in Table 1, when the traditional cluster search is used, an average of 81.57% pixels needed to be calculated at $4 \sim 6$ times. More clusters should be inspected for some pixels within complex patterned regions, especially for pixels located along the image boundary. Finally, the average number of calculations per pixel in the SLIC is $E = 4.37$.

The statistical information for the number of pixels are calculated in 4L-SLIC was shown in Table 2. It was notable that about 92.92% of the pixels only need to be calculated by 0 or 2 times which drastically reduces the execution times. Therefore, in the 4L-SLIC, the cluster inspection for each pixel $i \in \beta$ is only performed $E = 0.965$.

Algorithm 1. 4L-SLIC

```
Input: image $T$,initial cluster interval $S$,regularization factor $m$

  /*Initialization*/
  1. Initialize cluster centers $f_{C_k} = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps $S$.
  2. Move cluster centers to the lowest gradient position within their 3*3 neighborhood.
  3. Set label $L(i) = -1$ for each pixel $i$.

  /*Cluster Assignment*/
  for $i \in \alpha$, do
      4. $L(i) \leftarrow C_{k^*} \leftarrow \underset{C_k \in \theta_i}{\operatorname{argmin}} d_5(i, C_k)$
  end for
  for $i \in \beta$, do
      if $|\theta_{\omega_i \cap \alpha}|$ is greater than 1, then
          for $unique(\theta_{\omega_i \cap \alpha})$, do
              $5-1$. $L(i) \leftarrow C_{k^*} \leftarrow \underset{C_k \in \theta_{\omega_i \cap \alpha}}{\operatorname{argmin}} d_5(i, C_k)$
          end for
      else
          $5-2$. $L(i) \leftarrow L(j)$, where $j \in \omega_i \cap \alpha$
      end if
  end for

  /*Update*/
  6. Update $f_{C_k}$ with $mean(P(C_k)), k = 1, \cdots, K$
  7. Compute residual error $E$.
  8. Repeat Steps 4, 5 until $E \leq T_e$. // $T_e$ is a threshold for iteration

  /*Postprocessing*/
  9. Merge small clusters into one of the adjacent clusters. termination.

  Output: superpixel results $P(C_k), k = 1, \cdots, K$
```

Table 1. Numbers of computation for different superpixel scales in SLIC.

c	1	2	3	4	5	6	7	8	9	E
$K = 100$	0.018	0.130	0.177	0.405	0.166	0.099	0.005	0	1.5E−05	3.888
$K = 300$	0.006	0.073	0.113	0.464	0.184	0.147	0.011	0.002	0	4.242
$K = 500$	0.003	0.054	0.099	0.465	0.183	0.181	0.012	0.003	0	4.377
$K = 800$	0.002	0.040	0.070	0.459	0.198	0.203	0.022	0.006	0	4.538
$K = 1000$	0.002	0.042	0.075	0.520	0.172	0.175	0.011	0.003	4.5E−06	4.402
$K = 1500$	0.001	0.027	0.045	0.418	0.213	0.242	0.039	0.015	0	4.772
Avg	0.005	0.061	0.097	0.455	0.186	0.175	0.017	0.005	3.3E−06	4.370

Table 2. Numbers of computation for different superpixel scales in 4L-SLIC.

c	1	2	3	4	E
$K = 100$	0.699	0.268	0.032	0.001	0.636
$K = 300$	0.602	0.344	0.052	0.002	0.852
$K = 500$	0.557	0.377	0.063	0.003	0.955
$K = 800$	0.513	0.407	0.077	0.003	1.057
$K = 1000$	0.502	0.415	0.081	0.004	1.083
$K = 1500$	0.453	0.442	0.100	0.005	1.204
Avg	0.554	0.376	0.067	0.003	0.965

According to Eqs. (4), (5) and (6), we can conclude the number of operations in these three algorithms as follows.

$$E_{SLIC} = 4.37$$

$$E_{2L-SLIC} = \frac{1}{4} \times 4.37 + \frac{3}{4} \times 0.965 = 1.8163$$

$$E_{4L-SLIC} = \frac{1}{2} \times 4.37 + \frac{1}{2} \times 0.965 = 2.6675$$

We can see that $E_{2L-SLIC}$ and $E_{4L-SLIC}$ have a great improvement compared to E_{SLIC}. Although $E_{4L-SLIC}$ was more than $E_{2L-SLIC}$, the two algorithms was still on the same order of magnitude in computation time. Moreover, 4L-SLIC segmentation effect was much better than 2L-SLIC which will be confirmed in the following experimental results.

We demonstrated the validity of the algorithm from a visual view point in Fig. 4. We used four different colors (Yellow: 1, Green: 2, Red: 3, Blue: 4) to depict the number of pixels are calculated that belong to β on the image after ten iterations running the 4L-SLIC with setting $K = 100$ and $m = 10$. From Fig. 4, we can clearly observe that almost all pixels inside superpixels are represented in Yellow, whereas it was hard to find Blue ones. Green and Red pixels usually appeared along superpixel boundaries and on junctions with three branches, respectively.

3.2 Computational Efficiency of 4L-SLIC

Figure 5 showed the 4L-SLIC and 2L-SLIC contrast to SLIC in terms of segmentation speed improvement. In order to verify the pure algorithm performance, all algorithms runed without any parallel hardware and compiler optimization.

Since three algorithms were all based on iterative clustering algorithm, the execution time was related to the number of iterations, and both quantities changed with similar tendency. As shown in Fig. 5, although the 2L-SLIC was

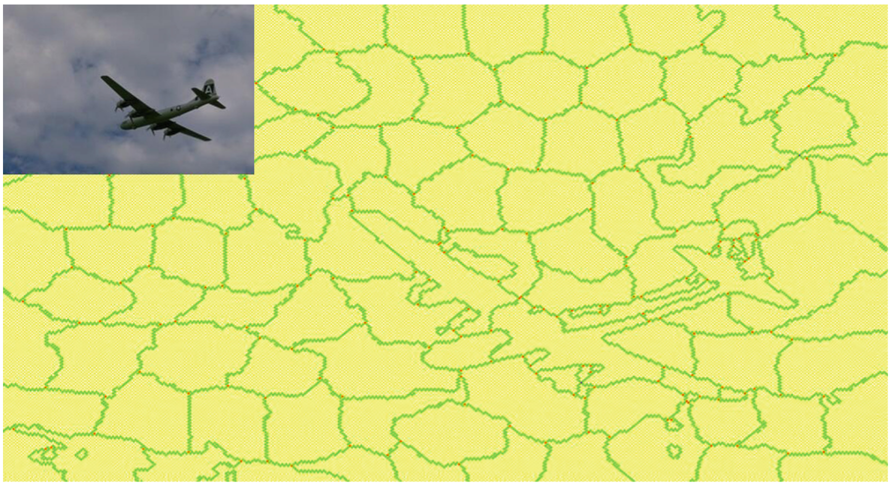


Fig. 4. Visual effect of the number of operations in 4L-SLIC. (Color figure online)

faster than the 4L-SLIC, these two algorithms were still in the same order of magnitude and the acceleration ratio was both about 30% compared to SLIC.

We can found further that with the increase in the number of superpixels K, the execution time of 4L-SLIC was getting closer to 2L-SLIC. The reason was that the more the number of superpixels, the more boundaries will be separated. That was to say there were more pixels need to be calculated many times. Therefore, the calculating time of the two algorithms was getting closer.

3.3 Segmentation Performance of 4L-SLIC

Superpixels were commonly used as a preprocessing step in image segmentation. A good superpixel algorithm should improve the performance of segmentation. So boundary adherence was also an important factor for evaluating the performance of superpixel segmentation algorithms.

Boundary Recall measured the fraction that ground truth boundaries correctly recovered by the superpixel boundaries. A high BR indicated that very few true boundaries are missed. Figure 6 showed the boundary recall curve. When $K < 1000$, 4L-SLIC was slightly lower than SLIC, whereas when $K > 1000$, 4L-SLIC has a better scores. However, compared with these two algorithms, the segmentation effect of 2L-SLIC was not as good as ours.

As shown in Figs. 7 and 8 where visual comparison of superpixels generated by all three algorithms. Figure 7 showed the result of segmentation on a simple image and Fig. 8 on a complex image. It was hard to find any difference result between SLIC and 4L-SLIC. But the boundary adherence of the regions that were circled in red is relatively poor at Fig. 7(b).

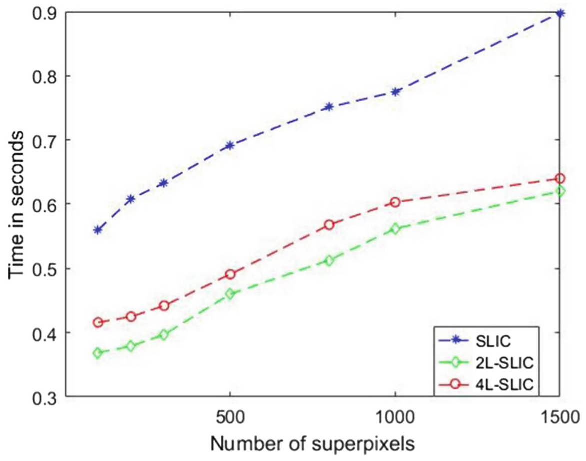


Fig. 5. Comparison of three algorithmic execution times.

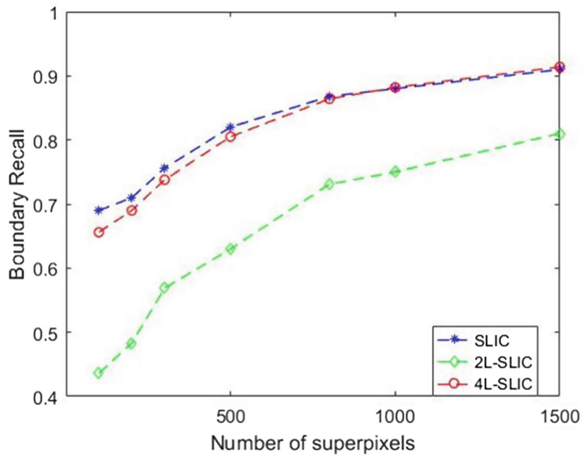
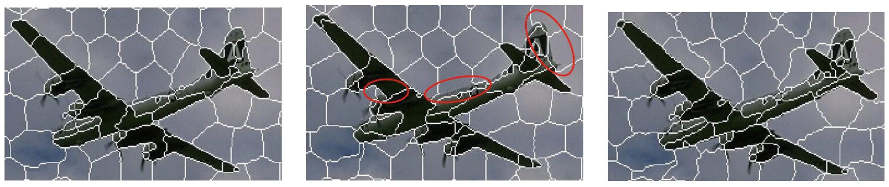


Fig. 6. Comparison of boundary recall of three algorithm.



(a) SLIC (b) 2L-SLIC (c) 4L-SLIC

Fig. 7. Segmentation effect in simple image.

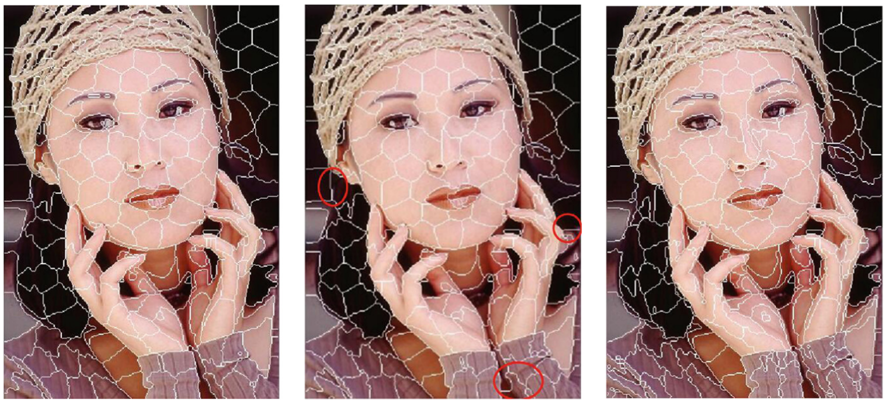

(a) SLIC (b) 2L-SLIC (c) 4L-SLIC

Fig. 8. Segmentation effect in complex image.

4 Conclusion

We proposed an accelerated version of superpixel algorithm which is named 4L-SLIC. 4L-SLIC reduces half search times without reducing any accuracy. We evaluated the experiments on the Berkeley public dataset. The time consumption results show that the speed of 4L-SLIC ups of 25%–30% than SLIC and 3%–4% to 2L-SLIC. As for the accuracy results, they show that our algorithm has a competitive performance with SLIC and is more accurate than 2L-SLIC. Our future work will focus on the accelerative vision of 4L-SLIC by parallel implementation.

Acknowledgments. This work was supported by National Natural Science Foundation of China (No. 61602380).

References

1. Malik, J.: Learning a classification model for segmentation. In: Proceedings of International Conference on Computer Vision, vol. 1, pp. 10–17 (2003)
2. Achanta, R., Shaji, A., Smith, K., et al.: SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern Anal. Mach. Intell. **34**(11), 2274 (2012)
3. Li, Z., Wu, X.M., Chang, S.F.: Segmentation using superpixels: a bipartite graph partitioning approach. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 789–796. IEEE Computer Society (2012)
4. Ayvaci, A., Soatto, S.: Motion segmentation with occlusions on the superpixel graph. In: IEEE International Conference on Computer Vision Workshops, pp. 727–734. IEEE (2009)
5. Mičušík, B., Košecká, J.: Multi-view superpixel stereo in urban environments. Int. J. Comput. Vis. **89**(1), 106–119 (2010)

6. Choi, K.S., Oh, K.W.: Subsampling-based acceleration of simple linear iterative clustering for superpixel segmentation. Comput. Vis. Image Underst. **146**, 1–8 (2016)
7. Liu, Y.J., Yu, C.C., Yu, M.J., et al.: Manifold SLIC: a fast method to compute content-sensitive superpixels. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 651–659. IEEE Computer Society (2016)
8. Lv, J.: An Improved SLIC superpixels using reciprocal nearest neighbor clustering. Int. J. Signal Process. Image Process. Pattern Recognit. 8 (2015)
9. Li, Z., Chen, J.: Superpixel segmentation using linear spectral clustering. In: Computer Vision and Pattern Recognition, pp. 1356–1363. IEEE (2015)