SUPERPIXEL SEGMENTATION BASED ON MULTIPLE SEED GROWTH

Zhicheng Wang, Xiaopeng Guo, Xiang Wu, Zhiheng Wang

CAD Research Center, Cao'an Highway No.4800 Tongji University, Shanghai, China
Zhichengwang@tongji.edu.cn gxiaop211@163.com 276743142@qq.com 1531707@tongji.edu.cn

ABSTRACT

The purpose of the superpixel algorithm is to find an over-segmented set of images. In this paper, we transform the superpixel segmentation problem into the problem of minimizing the cost function on the graph. We propose a new minimum spanning tree cost function based on graph theory. In order to achieve this cost function, we propose a greedy algorithm based on multiple seed growth. Compared to other superpixel segmentation algorithms, our proposed algorithm not only ensures superpixel segmentation quality but also has linear execution time and is easy to implement. Experiments on the BSD benchmark dataset show that superpixel segmentation algorithm in this paper is superior to the current most of the superpixel segmentation algorithm in Boundary recall and Under-segmentation error, and the algorithm is more efficient than the current most of the superpixel segmentation algorithm.

Index Terms—Superpixel segmentation, MST, graph theory, greedy strategy, multiple seed growth

1. INTRODUCTION

Superpixel segmentation is a popular image preprocessing method, which is widely used in computer vision, including single-view image segmentation[1,3,5,7], 3D reconstruction [2,6], target detection[10] and so on. Fig.1 shows the results of the algorithm proposed by this paper for seed numbers at 200,500, 1000 respectively, It can be seen the superpixel segmentation divide the pixels with similar visual characteristics into an atomic region. All the superpixels form a division of the whole image, requiring each pixel in the image only belong to one specific superpixel. To meet the requirements of image post-processing, the superpixel obtained by algorithm should be closely arranged and its size should be basically the same. In addtion, the edge should be close to the object boundary.

The concept of superpixel and the implementation of the first superpixel segmentation algorithm are done by Ren and Malik[4], who proposed a segmentation algorithm based on Normalized-cut theory. That achieves high processing accuracy, but its time cost is also high and usually takes several minutes. By treating the superpixel segmentation as an image segmentation problem, people have proposed some smaller time-cost segmentation methods, such as the Mean shift[8] method and the graph-based method[9]. However, the size and shape of superpixel are not consistent by using these methods, hence it is difficult to generate the superpixel equivalent as "pixels" when processing image. In recent years, the researchers have proposed several algorithms that are even less time-cost and better in quality, such as Super Lattices[9], SLIC (Single Linear Iterative Clustering)[12], Turbo Pixels[13]. These algorithms can shorten superpixel segmentation time cost from a few minutes to a few seconds.

We summarize the various design criteria of superpixel presented by previous superpixel extraction algorithms, and list those widely accept:

1. The algorithm is simple and efficient.
2. A single superpixel must cover only one object, and superpixels can not intersect each other.
3. The superpixel generated by the algorithm should fit the target edge as much as possible; the size and shape of superpixel should be consistent as much as possible
4. The final number of superpixels can be explicitly or implicitly determined by the user.

Combining with the above algorithm design criteria and the advantages and disadvantages of the existing algorithms, we believe that it is necessary to reduce the time complexity of the segmentation algorithm under the premise of ensuring certain segmentation precision. So far, many superpixel segmentation algorithms have been proposed based on the above criteria[8,9,11-14,17,18,20]. However, many of these implementations base on complex mathematical theory, with the need to build complex mathematical model. Now, this trend is becoming more and more obvious, which eventually lead to higher computational complexity. In order to achieve the balance of time complexity and the high quality of superpixel, we design a new energy function based on the minimum spanning tree (MST) method, and then we can solve the superpixel segmentation problems as problems of finding the minimum energy function. The new energy function consists of two parts: the term of mandatory superpixel for color uniform and the term of edge shape smoothing. The graph-based MST can well gather similar pixels and discard the pixels with large gaps. First, we should determine the number of generated superpixels that set the number of seed points K manually; and then solve the entire energy function optimal solution through the iterative method. Although the MST-based energy function looks complicated, the optimization algorithm we designed is very simple. Theoretical analysis and experimental results show that our energy function and the solution process are accurate and effective.

978-1-5386-2159-2/17/$31.00 ©2017 IEEE

Fig.1 Superpixels generated by our algorithm

2. CONSTRUCT ENERGY FUNCTION

2.1. Construct Graph

In this paper, we use the MST method in graph theory to carry out superpixel division, denoting undirected graph with $G = (V, E)$, where V is the vertex set, vertex $v_i \in V$, E is edge set, edge $(v_i, v_j) \in E$ represents the edge connecting the vertex v_i and v_j, and each edge corresponds to a weight value $w(v_i, v_j)$. We define the neighbor V^τ of vertex v as a set of points that are directly adjacent to$v$, where $V^\tau := \{\omega \in V: \omega, v \in E\}$. Assuming that there are vertices $v_1, v_2, \ldots, v_m \in V$, and $(v_i, v_{i+1}) \in E$, $i = 1,2, \ldots, m-1$, then there is a path from vertex v_1 to vertex v_m. We define the connected component c of the undirected graph as the subgraph of graph G, and there is at least one path between any vertices in the subgraph. The spanning tree T of the undirected graph G is a tree, and all the vertices of G are in T. We define the sum of the weights of the spanning tree as the weight of the edges contained in the tree, and the minimum spanning tree MST of graph G is the smallest of the weights in the spanning trees of G.

From the perspective of graph theory, we can consider the problem of superpixel segmentation as the partitioning problem of graph $G = (V, E)$, which divides the graph into smaller parts according to the specified characteristics. Each of these smaller parts can regard as a subgraph of the original image, and there is no common node between each subgraph. The nodes in all the subgraphs form the set of nodes of the original image. We will use the new representation of these smaller parts $G' = (V, E')$, where $V = V$, $E' \in E$. Each connected component in graph G' is equivalent to a superpixel in the image, and graph G' is equivalent to a superpixel segmentation of the image.

2.2. Energy Function

At present, there are many criteria for quality evaluation of superpixel segmentation, but almost all of the criteria contain the same segmentation criteria: the pixels within the same superpixel are as consistent as possible, and the gap between pixels within different superpixels is as large as possible. It is equivalent to that the weight of the edge in E' is the smaller the better, the same time, can also be seen as that the weight of the edge in $E - E'$ is the greater the better. Thus, there are two way to divide the graphs: one to achieve the division of the graph is obtaining $E - E'$, that is, removing the selected edges from the graph, and the typical algorithm using that division method is the graph-cut algorithm[25]. Another to achieve the division of the graph is obtaining E', which similar to the strategy of regional growth. From the principle, the algorithm is the concrete realization of this segmentation strategy.

First, for the connected component $c \in G'$, we define the internal dissimilarity of the connected component as the weight of the minimum spanning tree:

$$Dis(c) = \sum_{e \in MST(c,E)} w(e) \tag{1}$$

Where E is the set of edges contained in the graph, e is an edge in edge set E, and $w(e)$ is the weight corresponding to edge e.

Inspired by SEEDS[14] and SLIC[12] defining superpixel shape term, we also introduce the edge shape stimulus d after equation (1). The new energy function forms are as follows:

$$Dis(c_t) = \sum_{e_{i,j} \in MST(c_t,E)} (w(e_{i,j}) + \alpha d_{t,j}) \tag{2}$$

Where c_t is the t-th connected component, $e_{i,j}$ is the edge between the nodes v_i and v_j in the connected component c_t, α is the balance factor, $d_{t,j}$ represents the distance between the spanning tree root node and the node v_j. The introduction of the balance factor α allows us to control the shape and size of the generated superpixels. The larger the value of α, the greater the proportion of the shape factor occupied, the formation of superpixel more closely. Here we give the value range of α between [1, 15] by experiments.

Secondly, according to the definition of the connected components, we define the dissimilarity of the whole segment G' of the graph as the sum of the dissimilarity of the connected components:

$$Int(G') = \sum_{c_t \in G'} Dis(c_t) = \sum_{c_t \in G'} \sum_{e_{i,j} \in MST(c_t,E)} (w(e_{i,j}) + \alpha d_{t,j}) \tag{3}$$

2.3. Spanning Tree Weight

2.3.1. The Dissimilarity between Pixels

Using the definition of G and G' in Section 2.1, the superpixel segmentation method proposed in this paper achieves the division of superpixels by minimizing the energy function $Int(G')$, so the weight in the graph G becomes critical. The lower weight values reflect the more similarities between the pixels, and the higher weights reflect a larger gap between the pixels. The method proposed in this paper defines the dissimilarity between pixels in the RGB color space, and the color distance is defined by the Euclidian distance. Color distance formula defined as follow:

$$w(v_i, v_j) = dis_{RGB}(i, j) = \sqrt{(R_i - R_j)^2 + (G_i - G_j)^2 + (B_i - B_j)^2} \quad (4)$$

Where R, G, and B represent the RGB color components, respectively.

2.3.2. The Distance between Pixels

For the definition of $d_{t,j}$ in equation (2), we propose two distance definitions: one is the tree-based path length and the other is the distance definition in the image plane using pixel coordinates.

First, we assume that v_t is the root of the tree, and v_j is the current pixel, and we can define the distance between the two points in the spanning tree as the number of edges in the path connecting the two points.

$$d_{t,j} = \sum_{v_m, v_n \in C} D(e_{m,n}) \quad (5)$$

Where C is the connected component of the graph, $v_m$and v_n are the internal nodes of the connected component C, $e_{m,n}$ is the edge connecting v_m and v_m, $D(e_{m,n})$ is the binary function, which value is 0 or 1.

$$D(e_{m,n}) = \begin{cases} 0 & e_{m,n} \notin path(v_t, v_j) \, and \, v_m, v_n \in C \\ 1 & otherwise \end{cases} \quad (6)$$

Where $path(t,j)$ is the shortest path between v_t and v_j.

The second definition of the distance needs to be combined with the actual image plane. We define the distance between two nodes in the spanning tree as the distance between two pixels in the image. Thus, we can use the Euclidian distance to define $d_{t,j}$ in our energy function.

$$d_{t,j} = \sqrt{(x_t - x_j)^2 + (y_t - y_t)^2} \quad (7)$$

Where (x_t, y_t) is the physical position of the pixel $I(x_t, y_t)$ in the image, and the same (x_i, y_i) is the physical position of the pixel $I(x_i, y_i)$ in the image. In the actual implementation, we use this distance to define the way.

3. IMPLEMENTATION AND ANALYSIS

In our algorithm, we first gather K seed points in the image plane, where the seed points avoid appearing at the edges of object to ensure the maximum similarity within the superpixels obtained by them. In this paper, we adopt the greedy method to generate the superpixels and regard it as an execution that all the pixels are assigned to the corresponding supeerpixels. Then, we recalculate the positions of seed points in terms of the generated superpixels and repeat it until algorithm converges.

3.1. Initialization

We first select K seed points in the image, which are equivalent to the root nodes of spanning tree. We define the number of all pixels as image area. Suppose that there are N pixels in image, and thus image area is N. We suppose to generate K superpixels with regular shapes, and define the distance between two superpixels as $2d = \sqrt{N/K}$. We can treat the center of superpixels as the extracted seed points, so d is the distance between the center and edge of a superpixel. Theoretically, this setting way of seed points is feasible, and the experimental results also prove that this way is accurate. But this way may set the object edge pixels or noise pixels as seed points, we adopt a currently widely used way to adjust the positions of seed points. Specifically, the position which has the minimum gradient value within 3×3 pixels neighboring to each seed point is selected as final position of seed point. This adjustment strategy well avoids putting seed points at the edges, and reduces the chances that the noise pixels are selected as seed points[12].

3.2. Superpixel Generation

To illustrate superpixel generation, we first define the following symbols: G denotes the complete image, G_i is i-th superpixel, V and E respectively denote the vertexes and edges on G, V_i and E_i respectively denote the vertexes and edges in i-th superpixel. In the beginning, V_i is the selected seed point in i-th superpixel, $E_i = \emptyset$. There is the relation between G and G_i:

$$G = \sum_{i=1}^{k} G_i + (V - \sum_{i=1}^{k} V_i) + (E - \sum_{i=1}^{k} E_i) \quad (8)$$

$\bar{V}$ and $\bar{E}$ respectively denote the vertex and edge set which are not in any superpixel, so $\bar{V} = V - \sum_{i=1}^{k} V_i$, $\bar{E} = E - \sum_{i=1}^{k} E_i$. We select one vertex from $\bar{V}$ each time and add it to V_i, that is, when $v_a \in V_i$ and $v_b \in \bar{V}$, we add the edge (v_a, v_b) with minimum weight to E_i, and add the corresponding v_b to V_i. Repeat the above process until $\bar{V} = \emptyset$. After the whole process, we obtain a superpixel segment result, and we take the whole process as atomic operation. Then recalculate the position of superpixels based on last segment result, and iterate until convergence. Pseudo code of a single superpixel generation algorithm is described in Algorithm 1, and Algorithm 2 shows pseudo code of minimizing the objective function and its results correspond to superpixels which final generated.

3.3. Complexity Analysis

The procedure of our proposed superpixel generation algorithm is outlined in Algorithm 1. For a fixed number of superpixels, time complexity of our proposed algorithm is roughly linear. On the one hand, overall, our algorithm is very alike with k-means algorithm, the update of superpixel seed points is similar to determination of cluster center in k-means. On the other hand, the procedure of superpixel generation in our algorithm is similar to region growing.

Combined with the initialization process in Sec.3.1, we can know that in Algorithm 2, step1 has a time complexity of $O(1)$ to initialize superpixel seed points, and step 2 is a

common operation in image processing which has a time complexity of O(N). During the actual implementation of our algorithm, we use a priority queue to maintain the edge weights, so the time complexity of each iteration is O(Nlog(N)).

Algorithm 1: Superpixel Generation Algorithm
1: i = 0;
2: Initialize G_i,V_i,E_i,$\overline{V}$;
for i = 0 to $N-K$:
Seek out the edge (v_a, v_b) with minimum weight in $\overline{V}$, where $v_a \in V_i$, $v_b \in \overline{V}$;
Add edge (v_a, v_b) to edge set E_i, and add vertex v_b to vertex set V_i;
end for

Algorithm 2: Minimization of Objective Function
1: Initialize K superpixel seed points and the deviation E = ∞;
2: Calculate similarity matrix between four-neighborhood pixels of each pixel;
3: Fine-tuning seed position;
while (E <= threshhold):
Divide image into superpixels by Algorithm 1;
Recalculate superpixel seed points, and update the deviation E (L1 distance between new seed points and the previous ones);
end while

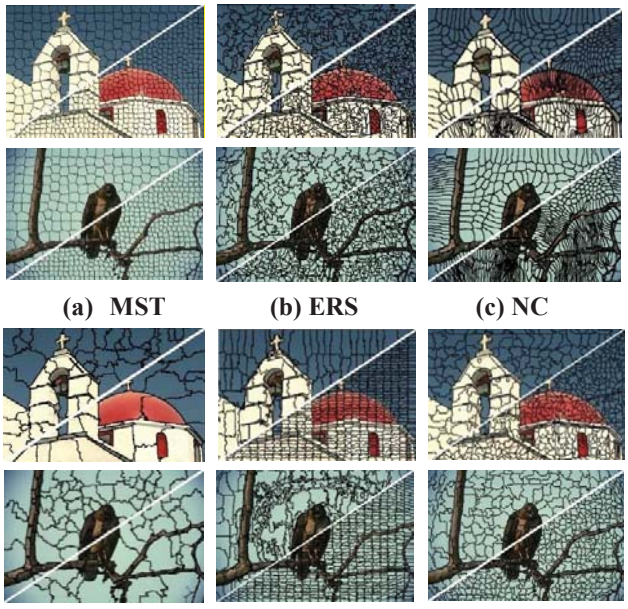
4. EXPERIMENTAL RESULTS

We compare the superpixel segmentation algorithm proposed in this paper with the existing seven segmentation algorithms. Experiments include ERS [17], NCuts [15], QS [24], EAMS [8], SEEDS [14], Watershed [21], and SLIC [12], by means of the existing superpixel segmentation algorithm library [22], where ERS, NCuts are based on graph theory, and EAMS, QS, Watershed, SEEDS are gradient-based methods.

Fig. 2 shows the comparison between the algorithm and the other algorithms in terms of visual effects. In order to make our experiment more convincing. We use the two criteria proposed in [23] to evaluate the effect of the superpixel segmentation algorithm. Meanwhile, in order to analyze the time complexity of the method more objectively, we also carry out the comparative experiment on the algorithm execution time. The experiments involved in this paper are based on the Matlab toolbox [16] using the Berkeley Segmentation Dataset (BSDS500) data set [19], and run on dual-core CPU (2.8 GHz i7) experimental platform.

4.1. Evaluation criteria

Due to the complexity of the visual task, it is difficult to find a way to meet all the needs of the division. For the superpixel segmentation we usually want to split each superpixel to be covered only on an object and the superpixel boundary to fit the edge of the object closely. In order to assess the quality of the superpixel segmentation algorithm, the researchers have



(a) MST (b) ERS (c) NC

(d) QS (e) SEEDs (f)WaterShed

Fig. 2 Each shows the results of six superpixel segmentation algorithms. Each picture contains two seed points that are 300 and 1000, respectively.

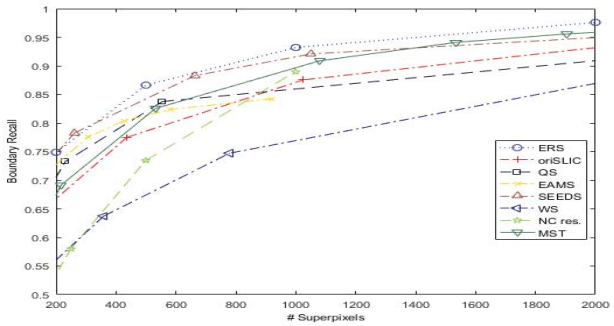
designed a variety of criteria for testing the quality of the segmentation. The most commonly used criteria are the boundary recall and under-segmentation error.

Boundary Recall(BR): Boundary recall is part of the segmentation precision detection framework introduced in [6], which quantifies the accuracy of the boundary pixels captured by the superpixel segmentation. The definition of boundary recall in this paper follows the definition of Ming-Yu Liu in [17]: boundary recall is the fraction of ground-truth boundaries that have a nearest superpixel boundary within an ε pixels distance of at least one superpixel boundary. Boundary-recall define as:

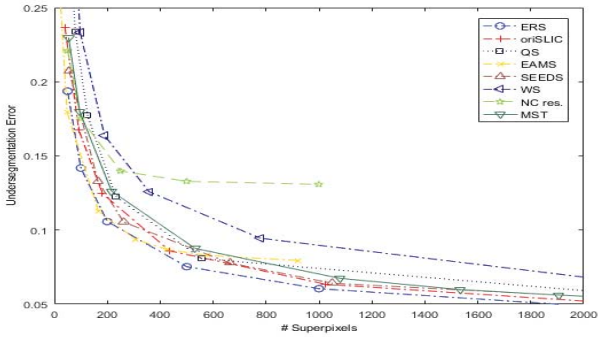
$$BR_{\Re}(\kappa) = \frac{\sum_{p\in\Re}\Gamma(\min_{q\in\kappa}\| p-q \| < \varepsilon)}{|\Re|} \qquad (9)$$

We use $\Re$ to denote the set of ground truth boundaries and κ to denote the set of superpixel boundaries. The indicator function Γ checks if the nearest pixel is within ε pixels distance. In our experiment we set $\varepsilon = 2$. Fig. 3 (a) shows the relationship between the number of superpixels generated by the algorithm and the boundary recall. At the same time, it also gives an experimental comparison between the algorithm and the other seven algorithms in the boundary recall.

Under-segmentation error(UE)[13]: measures fraction of pixel leak across ground truth boundaries. It evaluates the quality of segmentation based on the requirement that one superpixel should overlap with only one object in the image. Given a ground-truth segmentation into segments $G = \{G_1, G_2, G_3, \dots, G_n\}$ and a superpixel segmentation into



(a)



(b)

Figure 3 (a) The relationship between the number of superpixels and boundary recall. (b) The relationship between the number of superpixels and the under-segmentation error.

superpixels $S = \{S_1, S_2, S_3, \ldots, S_k\}$. We use

$$UE_G(S) = \frac{\sum_{i=1}^{n}\left(\sum_{j=1:S_j \cap G_i \neq \varnothing}^{k} |S_j|\right) - \sum_{i=1}^{n}|G_i|}{\sum_{i=1}^{n}|G_i|} \tag{10}$$

to quantify the under-segmentation error for segments $G = \{G_1, G_2, G_3, \ldots, G_n\}$. Where: $|\cdot|$ find the number of pixels contained in the segment. Fig. 3 (b) shows that the relationship between the number of superpixels and the under-segmentation error, and it also compares the algorithm with other seven algorithms in under-segmentation error.

Run time(RT): The introduction of time as an evaluation of the superpixel segmentation algorithm is because we believe that time complexity is another very important factor. Under the premise of a better segmentation effect, it determines whether the algorithm has practical application value. Fig. 4 compares the segmentation algorithm with the other seven algorithms in terms of runtime.

4.2. Segmentation quality discussion

It is generally believed that a good superpixel segmentation algorithm should have both lower under-segmentation error and higher boundary recall, while the divided superpixels should be as uniform as possible and the distribution should

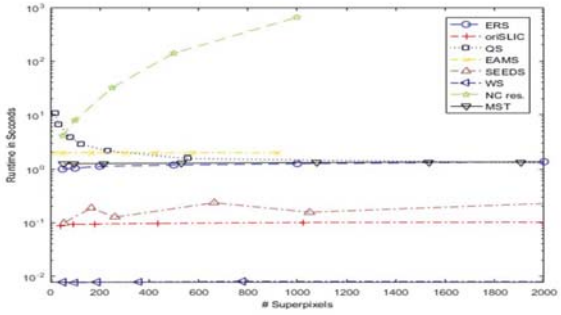


Figure 4 algorithm runtime

be as close as possible. At the same time, we believe that a good algorithm should reduce the time complexity under the premise of good segmentation effect. The algorithm is ultimately intended for practical use, and therefore must have a lower time complexity. Therefore, we did a comparison from the following four aspects: under-segmentation error, boundary recall, runtime and the actual segmentation.

By observing Figs. 3 (a) and (b), it can be seen that the ERS algorithm has the highest boundary recall and the lowest under-segmentation error in the eight super-pixel segmentation algorithms. If we only consider these two criteria, the ERS algorithm is the best, the ERS algorithm is the best, but the size and shape of the superpixels obtained by the ERS algorithm are quite different, and the same color region may be divided into different superpixels. This segmentation result is very detrimental to post-processing. In contrast, our algorithm is not as good as the ERS in boundary recall and under-segmentation error, but the superpixels generated by our algorithm tend to be more consistent in size and shape, which is very helpful for post-processing.

As shown in Fig.3, the algorithm proposed in this paper has higher boundary recall than the SEEDS algorithm when the number of superpixel seed points is 1400, and the under-segmentation error is basically the same as SEEDS. At the same time, compared with the algorithm segmentation effect shown in Fig.2, it can be seen that the segmentation of SEEDS algorithm appears to be in disorder when the number of seed points selected is small, while our algorithm still remain consistent. Considering the number of seed points in practical application should not be too small, that is, superpixel area should not be too large, so our algorithm is a good choice when seed selection is more.

Fig.3(a) shows that the proposed algorithm has higher boundary recall than other algorithms when the superpixel selected is bigger than 500, and the obtained superpixel is still regular. Fig.3(b) shows that the proposed algorithm also has lower under-segmentation error than other algorithms.

Although, in run-time, our algorithm is not as fast as the Watershed, SLIC, SEEDS algorithm, but the execution speed is the same as the ERS algorithm, about 1S or so. By observing Fig.3(c), it can be seen that the algorithm proposed in this paper is basically linear at the execution time, and the execution time varies little with the number of seed points.

5. CONCLUSION

In this paper, we propose an algorithm that can generate any number of superpixels in linear time segmentation, which can produce the nearly same size and close distribution super-pixels. Based on the minimum spanning tree in the graph theory, the algorithm divides the superpixels by region growth according to the pre-determined multiple seed points. The energy function constructed in this paper contains both the color difference and the distance factor, so it can generate superpixels with the same color and size. Whether it is from the actual segmentation results or algorithm evaluation criteria, the proposed algorithm can be comparable to most of the existing superpixel segmentation algorithm.

6. ACKNOWLEDGMENTS

The authors would like to express their gratitude to the anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper. The work is supported by the National Science and Technology Support Project of China under Grant 2015BAF04B01 and Science and Technology Innovation Project of Shanghai under Grant 16111107602.

7. REFERENCES

[1] Lim J, Han B. Generalized background subtraction using superpixels with label integrated motion estimation[C] //European Conference on Computer Vision. Springer International Publishing, 2014: 173-187.

[2] Ramalingam S, Kohli P, Alahari K, et al. Exact inference in multi-label CRFs with higher order cliques[C] //Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008: 1-8.

[3] Arbelaez P, Maire M, Fowlkes C, et al. Contour detection and hierarchical image segmentation[J] //IEEE transactions on pattern analysis and machine intelligence, 2011, 33(5): 898-916.

[4] Ren X, Malik J. Learning a Classification Model for Segmentation[C] //ICCV. 2003, 1: 10-17.

[5] Trulls E, Tsogkas S, Kokkinos I, et al. Segmentation-aware deformable part models[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 168-175.

[6] Kohli, L. Ladicky, and P. Torr. Robust higher order potentials for enforcing label consistency[C] //In IJCV 2009, 82:302–324.

[7] Kohli P, Torr P H S. Robust higher order potentials for enforcing label consistency[J]. //International Journal of Computer Vision, 2009, 82(3): 302-324.

[8] Comaniciu D, Meer P. Mean shift: A robust approach toward feature space analysis[J]. //IEEE Transactions on pattern analysis and machine intelligence, 2002, 24(5): 603-619.

[9] Moore A P, Prince S J D, Warrell J, et al. Superpixel lattices[C] //Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008: 1-8.

[10] Mori G, Ren X, Efros A A, et al. Recovering human body configurations: Combining segmentation and recognition[C] //CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. IEEE, 2004, 2: II-II.

[11] Shen J, Du Y, Wang W, et al. Lazy random walks for superpixel segmentation[J] //IEEE Transactions on Image Processing, 2014, 23(4): 1451-1462.

[12] Achanta R, Shaji A, Smith K, et al. SLIC superpixels compared to state-of-the-art superpixel methods[J]. //IEEE transactions on pattern analysis and machine intelligence, 2012, 34(11): 2274-2282.

[13] Levinshtein A, Stere A, Kutulakos K N, et al. Turbopixels: Fast superpixels using geometric flows[J] //IEEE transactions on pattern analysis and machine intelligence, 2009, 31(12): 2290-2297.

[14] Van den Bergh M, Boix X, Roig G, et al. SEEDS: Superpixels extracted via energy-driven sampling[C] //European conference on computer vision. Springer Berlin Heidelberg, 2012: 13-26.

[15] Shi J, Malik J. Normalized cuts and image segmentation[J] //IEEE Transactions on pattern analysis and machine intelligence, 2000, 22(8): 888-905.

[16] Neubert P. Superpixels and their Application for Visual Place Recognition in Changing Environments[J] //2015.

[17] Liu M Y, Tuzel O, Ramalingam S, et al. Entropy rate superpixel segmentation[C] //Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011: 2097-2104.

[18] Zhang Y, Hartley R, Mashford J, et al. Superpixels via pseudo-boolean optimization[C] //Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011: 1387-1394.

[19] Martin D, Fowlkes C, Tal D, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics[C] //Computer Vision, 2001. ICCV 2001, 2: 416-423.

[20] Neubert P, Protzel P. Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms[C] //Pattern Recognition (ICPR), 2014 22nd International Conference on. IEEE, 2014: 996-1001.

[21] Vincent L, Soille P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations[J]. IEEE transactions on pattern analysis and machine intelligence, 1991, 13(6): 583-598.

[22] Stutz D, Hermans A, Leibe B. Superpixels: An evaluation of the state-of-the-art[J] //Computer Vision and Image Understanding, 2017.

[23] Martin D R, Fowlkes C C, Malik J. Learning to detect natural image boundaries using local brightness, color, and texture cues[J]. IEEE transactions on pattern analysis and machine intelligence, 2004, 26(5): 530-549.

[24] Vedaldi A, Soatto S. Quick shift and kernel methods for mode seeking[J]. //Computer vision–ECCV 2008, 2008: 705-718.

[25] Boykov Y Y, Jolly M P. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images[C] //Computer Vision. ICCV 2001, 2001, 1: 105-112.