Journal of Intelligent & Fuzzy Systems 43 (2022) 4409–4430
DOI:10.3233/JIFS-212967
IOS Press

CMSuG: Competitive mechanism-based superpixel generation method for image segmentation

Qianna Cui[a], Haiwei Pan[a,*], Xiaokun Li[b,*], Kejia Zhang[a] and Weipeng Chen[a]
[a]*School of Computer Science and Technology, Harbin Engineering University, Harbin, China*
[b]*School of Computer Science and Technology, Heilongjiang University, Harbin, China*

Abstract. During the last years, object-based image segmentation (OBIA) has seen a considerable increase in the image segmentation. OBIA is generally based on superpixel methods, in which the clustering-based method plays an increasingly important role. Most clustering methods for generating superpixels suffer from inaccurate classification points with inappropriate cluster centers. To solve the problem, we propose a competitive mechanism-based superpixel generation (CMSuG) method, which both accelerates convergence and promotes robustness for noise sensitivity. Then, image segmentation results will be obtained by a region adjacent graph (RAG)-based merging algorithm after constructing an RAG. However, high segmentation accuracy is customarily accompanied by expensive time-consuming costs. To improve computational efficiency, we address a parallel CMSuG algorithm, the time of which is much less than the CMSuG method. In addition, we present a parallel RAG method to decrease the expensive time-consuming cost in serial RAG construction. By leveraging parallel techniques, the running time of the whole image segmentation method decline with the time complexity from $O(N) + O(K^2)$ to $O(N/K)$ or $O(K^2)$, in which N is the size of an input image and K is the given number of the superpixel. In the experiments, both nature image and remote sensing image segmentation results demonstrate that our CMSuG method outperforms the state-of-the-art superpixel generation methods, and then performs well for image segmentation in turn. Compared with the serial segmentation method, our parallel techniques gain more than four times acceleration in both remote sensing image dataset and nature image dataset.

Keywords: Superpixels, competitive mechanism, image segmentation, parallel, graph-based

1. Introduction

Object-based image analysis (OBIA) with image objects instead of individual pixels, has attracted considerable attention in the computer vision field [1]. Image objects are more likely to extract the information of homogeneous regions, thus reduce the burden of redundant information extracted with single pixels. OBIA is usually composed of three parts: (1) image pre-segmentation; (2) feature extraction based on objects; (3) objects classification and image merging [1]. Good superpixels with higher homogeneity are suitable for image objects.

Superpixel is generated by grouping local adjacent similar pixels into a cluster [2]. The principle of grouping pixels contains intra-region similarity and inter-region dissimilarity, which lies in the colour and other properties of each pixel. It means that the pixels are similar in the same region and dissimilar in different parts. In contrast with single-pixel,

*Corresponding authors. Haiwei Pan, School of Computer Science and Technology, Harbin Engineering University, Nan-Tong Street, Harbin, China. E-mail: panhaiwei@hrbeu.edu.cn and Xiaokun Li, School of Computer Science and Technology, Heilongjiang University, Xuefu Road, Harbin, China. E-mail: li.xiaokun@163.com.

superpixel significantly improves the computational efficiency and resists the sensitivity of noise in the image segmentation field [2, 3]. Due to the above advantages, superpixel plays a key role in computer vision applications, such as image segmentation and classification [4, 5], pattern recognition [6], object detection [7], and short video processing [8].

Recently, various images are segmented with superpixel, such as remote sensing image [9, 10] and natural image [11]. The widely used methods for superpixel generation are generally classified into two kinds: graph-based and non-graph-based algorithms [2]. In these methods, the simple linear iterative clustering (SLIC) [2] method is widely applied to image segmentation due to its lower computational complexity [12, 13]. SLIC and the improved SLIC methods depend on the initial clusters strongly, which suffers from dead points or underutilization points. These pixels are assigned to a cluster that is not suitable for them. The sensitivity to original cluster centres results in the wrong categorization of noise points.

Image segmentation based on superpixels is to classify all superpixels into a set of regions with the minimum number. These regions satisfy two conditions simultaneously: maximum correlation within a region and maximum dissimilarity between regions according to some properties of superpixels such as colours. Compared with the seeded region growing and mean-shift methods [1], graph methods implicitly arrange superpixels into the mathematical structure and obtain the segmentation result with more efficient computation. In all graph methods, the minimum spanning tree (MST) is widely used by the reason of the higher computational efficiency [14]. The MST algorithm is based on the construction of a region adjacency graph (RAG), where each vertex is denoted as a superpixel and each edge links the adjacency vertexes. The weight of each edge is computed with the dissimilarity of adjacent vertexes, which is calculated by the features of adjacent superpixels [14].

In this paper, we propose a competitive mechanism-based image segmentation method to improve above all problems. To tackle the problem in superpixel generation, we introduce competitive mechanism to weaken the dependence of initial clusters in fuzzy simple linear iterative clustering (FSLIC) [15]. Inspired by [16, 17], we propose a ***C**ompetitive **M**echanism-based **Su**perpixel **G**eneration* (CMSuG) algorithm by joining the possibility of non-initial clusters. An example superpixel result by CMSuG algorithm is shown in Fig. 1(a) and 1(c). Then we construct an RAG based superpixels and do merging operation by RAG-based algorithm.

In some actual image applications, real-time processing is an urgent request, such as search and rescue mission, real-time target detection and so on. However, the time complexity of CMSuG algorithm and superpixel-based RAG construction are related to the size of an input image. Then the time-consuming cost is too expensive in the whole image segmentation process. In the case of limited resources, to meet the need of real-time image segmentation, we use the graphical processing units (GPU) platform to improve the above-proposed image segmentation method by the advantage of low cost, lightweight and portability [18]. Because of the multi-thread parallel processing of GPU, we propose parallel CMSuG and parallel RAG construction methods to decrease computation time by assigning each pixel to a single thread in the Compute Device Unified Architecture (CUDA) of NVidia. Figure 1(b) and (d) display superpixel generation results by parallel CMSuG algorithm.

The main contributions of this paper is as follows:

(1) We propose a CMSuG algorithm by introducing competitive mechanism, which is able to cope with the absence of non-initial clusters in FSLIC.

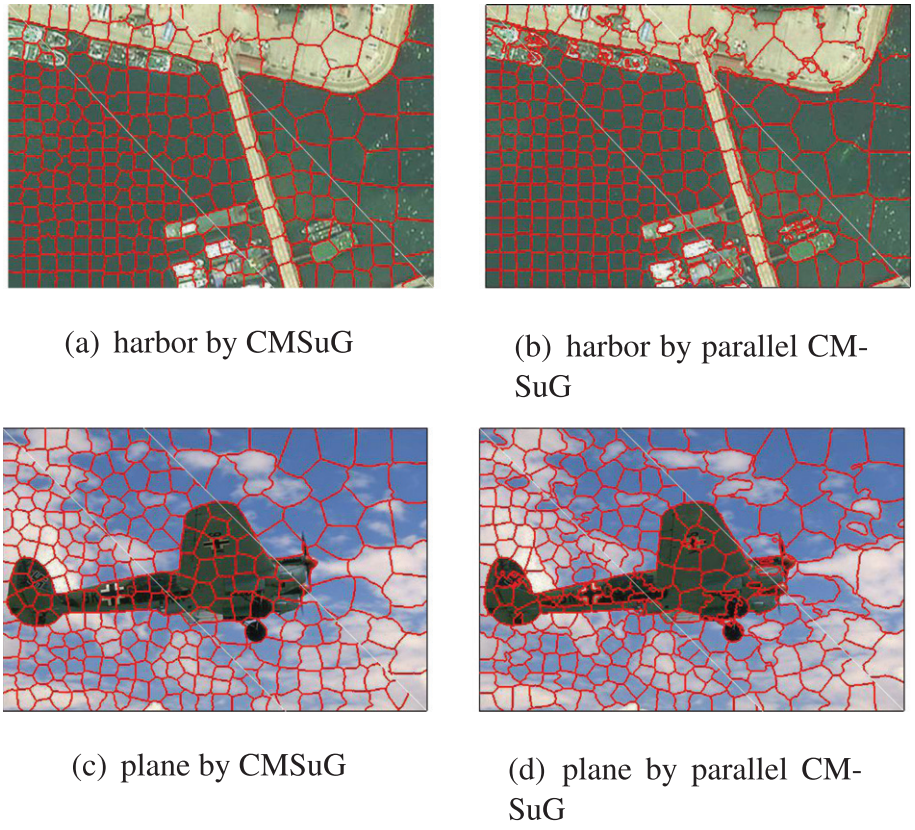
(2) To complete the image segmentation task, we propose a post-merging algorithm with superpixel by using the RAG-based merging method.

(3) To overcome the time-consuming in CMSuG method and RAG construction, we address the parallel CMSuG method and RAG construction method. For doing that, image segmentation has a great efficiency, which in turn satisfying the request of real-time image processing.

2. Related work

Superpixel is a homogeneous group that clustering similar pixels in local adjacent areas. Due to that advantage, superpixel-based instead of pixel-based methods used in image segmentation can be likely to connect contextual structures. High-efficient computation is distinct in superpixel-based image segmentation method with fewer relationship construction.

In the superpixel-based image segmentation field, it is mainly composed of two phases: (1) superpixel generation; (2) post-merging based on the feature of superpixels. Generating superpixel can be accelerated by the support of parallel computation device



(a) harbor by CMSuG

(b) harbor by parallel CMSuG

(c) plane by CMSuG

(d) plane by parallel CMSuG

Fig. 1. Examples of superpixels are shown on 100/300/500 scale by the CMSuG algorithm and parallel CMSuG algorithm, in which habor is a remote sensing image and plane is a nature image.

GPU. With the extracted feature of each superpixel, post-merging is to classify all local adjacent superpixels with similar features into a category. An intuitive method is that the relationship between adjacent superpixels can be constructed in a graph, which is used to complete post-merging.

2.1. Superpixel generation methods

Superpixel generation methods include graph-based and non-graph-based superpixel generation algorithms.

2.1.1. Graph-based algorithms

An image is described as an undirected graph, which considers each pixel as a vertex and connects a pair of adjacent vertices with an edge [19]. The weight of edges is usually calculated with the dissimilarity between pixel and its eight neighbors [19]. Classical graph-based method mainly contains Felzenszwalb and Huttenlocher (FH) [19], normalized cut (NC) [20] and entropy rate superpixels(ERS) [3]. In fact, FH is an algorithm of looking for a minimal spanning tree (MST) with $O(N \log N)$ complexity, where N is the total number of image pixels. NC algorithm generates superpixels by minimizing the cost function of cut between different parties, the time complexity of which is $O(N^{\frac{3}{2}})$. Good superpixels have high compactness and homogeneity, because of this, ERS constructs a maximum objective function that contains the two properties. And, a greedy algorithm with $O(N \log N)$ complexity was proposed in ERS [3]. Another graph-based superpixel generation algorithm is to find optimal paths with $O(N^{\frac{3}{2}} \log N)$ complexity in paper [21]. These graph-based methods are all short of controlling compactness function for superpixels and have high time-consuming cost [15].

2.1.2. Non-graph-based algorithms

The non-graph-based algorithm is usually used with gradient ascent and clustering-based methods. Gradient ascent starts with an initial pixel and then update clusters in the direction of gradient descent

[2], which mainly incorporates mean-shift (MS) [22], watershed (WS) [23] and Turbopixel (TP) [24] methods. Both MS and WS generate irregularly shaped superpixels due to the absence of a controlling compactness factor. The complexity of the two algorithms is $O(N^2)$ and $O(N \log N)$, respectively. In addition, superpixels generated by TP have weak boundary adherence with $O(N)$ complexity.

Another commonly used superpixel generation method adopts the clustering theory. The SLIC method was firstly proposed in 2012 [2]. It is a modified clustering method based on the k-means clustering algorithm. The performance of SLIC outperforms the above two categories in compactness controlling, and it is faster than them with $O(N)$ complexity. However, the algorithm is short of robustness [15]. Subsequently, to improve the performance of SLIC, certain clustering algorithms are proposed based on SLIC. Linear spectral clustering (LSC) [25] has achieved both boundary adherence and global image properties by combining NC formulation and computational similarity of SLIC, but without robustness in ambiguous pixels. Simple non-iterative clustering (SNIC) [26] updates the cluster centers online with lower memory requests, but the superpixel is irregular. All above superpixel generation algorithms based on SLIC falls into weak regularity and wrong pixel allocation with the initial pre-set cluster, the time complexity of which is $O(N)$. Wu et al. [15] developed FSLIC by adopting multiple classifications for each pixel with fuzzy membership. The complexity of FSLIC is $O(11N)$. Compared with the above SLIC, LSC, SNIC algorithms, FSLIC achieves a high-performance improvement. However, FSLIC segmentation still has weak robustness for blurry and noise images without considering pixels that do not belong to existing clusters.

2.2. Parallel superpixel generation method

The growing of GPU platform is a parallel computing architecture [27], by which the running time of image processing reduces sharply [28]. High-speed superpixel generation methods based on GPU improve the expensive time cost of previous superpixel generation. A modified MS algorithm was proposed under the CUDA framework [29], which worked more than 20 times faster than an improved MS algorithm based on CPU. The parallel TP algorithm [30] is a parallel level-set evolution process via superpixel expansion in each thread. It has improved 15 times faster than the TP algorithm. Carl et al. [31] introduced a parallel SLIC method based on GPU by assigning a pixel into a thread, which was 83 times faster than SLIC. Similar to the assignment of pixels in parallel SLIC, a modified LSC based on GPU has obtained impressive results [32], which has achieved 83 times faster than LSC. However, the lower time cost is accompanied by lower segmentation properties. Good performance of improved algorithms based on GPU is up to the basic algorithms on CPU.

2.3. Post-merging method based on graph

The superpixel-based merging method is performed well on a graph constructed on pre-segmentation [33]. Graph model includes RAG [19] and nearest neighbor graph (NNG) [34]. Although NNG accelerates the merging procedure by searching for the best neighbor, NNG is built with the dependence of RAG [35]. It is time-consuming for searching NNG in each iterative merging step. RAG Construction is to extract road [14, 36] based on superpixels generated by the superpixel generation method. In the RAG-based merging algorithm, it starts with a vertex, then merge it and a corresponding adjacent vertex until not satisfy the merging condition. However, RAG construction relies on travelling all pixels in each superpixel, which takes up time.

3. Methodology

Our competitive mechanism-based superpixel generation method for image segmentation contains the following steps: (1) CMSuG method. (2) Post-merging with superpixels. This procedure, two phrases included, is mainly about RAG construction with superpixels and RAG-based merging. The whole processing chain of the proposed image segmentation is illustrated in Fig. 2.

3.1. CMSuG method

In clustering-based superpixel generation methods, initial clusters limit pixel classification. If a pixel belongs to an external cluster, we call that the pixel is assigned to the competitor of existing clusters. By introducing competitive mechanism in FSLIC, we propose a CMSuG method.

Superpixels are image blocks that are composed of adjacent pixels with similar features. So, each cluster is expected to be updated along the direction that contains the largest number of similar pixels. At the same

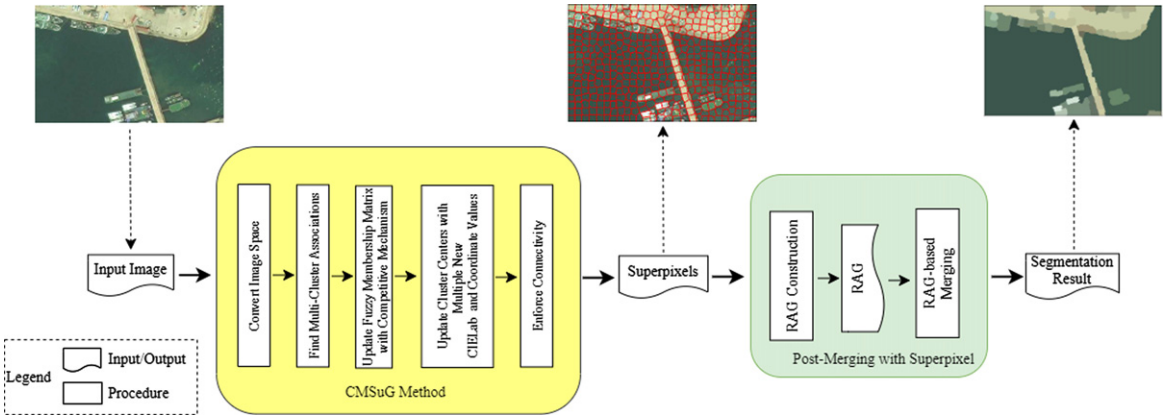


Fig. 2. Flowchart of the proposed image segmentation method.

time, the objective of the expectation is also to suppress the growth of clusters in a direction that deviates from the expected direction, which is competitors. In the CMSuG algorithm, not only each initial cluster is modified with a certain learning rate but also the competitors of which are learning with a smaller factor. The CMSuG algorithm is expressed as follows:

(1) Colour space conversion. The input image data is usually an RGB image, which is converted to CIELab colour space $[l_i, a_i, b_i]^T$. Each image point is represented by a five-dimensional vector composed of the color vector and XY coordinates $[x_i, y_i]^T$, which is denoted as $\boldsymbol{p}_i = [l_i, a_i, b_i, x_i, y_i]^T$.

(2) Cluster centroids initialization. Given superpixel number K, the size of each superpixel can be computed by N/K, where N is the total number of image pixels. Initial clustering centers are sampled at regular grid steps $S = \sqrt{N/K}$. Then move each cluster center to the lowest gradient positions at its corresponding 3×3 neighbor windows.

(3) Multi-cluster associations foundation. For each pixel i in a $2S \times 2S$ searching scope around the center $\boldsymbol{c}_j$, the distance between each cluster center and all pixels that hit in the searching scope can be calculated by the following equation:

$$\begin{cases} d_c = \sqrt{(l_i - l_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2} \\ d_s = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \\ dist(\boldsymbol{p}_i, \boldsymbol{c}_j) = \sqrt{(\frac{d_c}{m^c})^2 + (\frac{d_s}{S})^2}. \end{cases} \tag{1}$$

where d_c is the colour distance, d_s describes the spacial distance, m^c is the normalization factor of colour distance. An expression $dist(\boldsymbol{p}_i, \boldsymbol{c}_j)$ measures the distance between pixel $\boldsymbol{p}_i$ and cluster centers $\boldsymbol{c}_j$. The maximum possibility that the ith pixel belongs to the jth cluster means that the $dist(\boldsymbol{p}_i, \boldsymbol{c}_j)$ is the minimum in all distance $(dist(\boldsymbol{p}_i, \boldsymbol{c}_{j_1}), \cdots, dist(\boldsymbol{p}_i, \boldsymbol{c}_{j_t}))$. The searching scope of cluster center $\boldsymbol{c}_{j^*}(j^* \in [1, j_t])$ concludes pixel $\boldsymbol{p}_i$. Because of that, multiple possible clusters that associate each pixel can be found. Thus, it is practical that clusters with larger possibilities among all possible clusters are selected with the maximum number t^* for all pixels. After that operations, there are t^* cluster centers arranged on each pixel, and then storage these centers as classification labels.

(4) Fuzzy membership matrix updating with competitive mechanism. The objective of updating clusters with competitive mechanism is to ensure each maximum cluster with minimum competitors. For the first, it is feasible to minimize the distances between all pixels and the corresponding cluster centers. Motivated by the theory of competitive learning [16], the second target is to suppress the growth of competitors by introducing novel membership constraint function $f(u_{ij}) = \sum_{j=1}^{z} u_{ij}(1 - u_{ij}^{m-1})$, the minimum objective function of CMSuG is as follows,

$$\begin{cases} J_{\text{CMSuG}} = \sum_{j=1}^{z} \sum_{i=1}^{N_j} u_{ij}^m \|\boldsymbol{p}_i - \boldsymbol{c}_j\|^2 \\ \qquad\qquad + \sum_{i=1}^{N_j} a_i \sum_{j=1}^{z} u_{ij}(1 - u_{ij}^{m-1}) \\ s.t. \quad \sum_{j=1}^{z} u_{ij} = 1, \qquad \forall i \end{cases} \tag{2}$$

where $z(z \leq K)$ is the total number of clusters after initialization, N_j is the number of pixel in the

searching region around cluster center $\boldsymbol{c}_j$, u_{ij} is the membership degree of the ith pixel that belongs to the jth cluster, $\boldsymbol{p}_i$ is the ith pixel, and the operation $\|\cdot\|^2$ is the square of the norm, m is the fuzzy partition factor for controlling the degree of fuzzy overlap.

Let objective optimization problem (2) convert to an unconstrained optimization problem by introducing Lagrange multipliers, which describes as follows,

$$F(\lambda, u_{ij}) = J_{\text{CMSuG}} + \lambda(\sum_{j=1}^{z} u_{ij} - 1). \tag{3}$$

Then, the membership matrix $\boldsymbol{U}$ can be computed by following equation,

$$u_{ij} = (\sum_{j=1}^{t_i} \left(\frac{(\|\boldsymbol{p}_i - \boldsymbol{c}_j\|)^2 - a_i}{(\|\boldsymbol{p}_i - \boldsymbol{c}_j\|)^2 - a_i} \right)^{\frac{1}{m-1}})^{-1} \tag{4}$$

where t_i is the number of possible clusters that the ith pixel pertains, $a_i = \alpha \cdot \min\{\|\boldsymbol{p}_i - \boldsymbol{c}_j\|^2\}$, the parameter $0 < \alpha < 1$ is the controlling factor for suppressing the growth of competitors. It is obvious that so small membership degree means that pixel $\boldsymbol{p}_i$ belongs to cluster $\boldsymbol{c}_j$ next to nothing. In CMSuG method, the first t^* maximum possibility is considered, that is to say the maximum value of each t_i is t^*.

Neighboring pixels usually belong to the same cluster. To improve the robustness of pixels classification, the degree of membership u_{ij} companies with the spatial information of 3×3 neighbors of each pixel [15]. Thus, an improving fuzzy membership matrix $U^{'}$ in the CMSuG method is computed with $u_{ij}^{'} = \frac{h_{ij}^2}{\sum_{k=1}^{n_i} h_{ik}^2}$. In the above expression, $h_{ij} = \sum_{i^* \in W_i} u_{i^* j}$ is the sum of the fuzzy partition degree of the ith pixel's neighbors i^* classified into the jth cluster, in which W_i is the 3×3 neighborhood set of pixel $\boldsymbol{p}_i$.

(5) Cluster centers updating with new CIELab and coordinate values. Each cluster center is updated by the weighted linear combination of pixel set in which the ith pixel belongs to the jth cluster, as shown in Equation (5),

$$\boldsymbol{c}_j = \frac{\sum_{i=1}^{n_j} (u_{ij}^{'})^m \boldsymbol{p}_i}{\sum_{i=1}^{n_j} (u_{ij}^{'})^m} \tag{5}$$

where n_j is the total number of pixels that have the assignment of cluster $\boldsymbol{c}_j$.

(6) Labels assignment and enforce connectivity. Each pixel is assigned to the cluster with the greatest fuzzy partition value. There may be some isolated pixels after labels assignment, these pixels are assigned to the nearest cluster center.

The detailed process of the CMSuG algorithm is shown in Algorithm1.

Algorithm1: CMSuG

Input: Initial image I, number of superpixel K, maximum number of iterations N^*, maximum number of possible clusters t^*, threshold ϵ.
Output: Superpixel set $\mathbb{R} = R_1, \ldots R_n$ $(n \leq K)$.
Convert the initial image to CIELab colour space.
Initialize the cluster center sets $\mathbb{C}$ by sampling pixels at regular grid step S $(S = \sqrt{N/K}$, N is the pixel number of $I)$ and adjust these centers to the lowest gradient within the 3×3 neighbors.
Set label matrix $L(i) \leftarrow -1$, storing possible label matrix $G(i,l) \leftarrow -1$, storing possible cluster's number matrix $F(i,1) \leftarrow 0$, distance matrix $D(i,l) \leftarrow \infty$ for each pixel $\boldsymbol{p}_i$. $(l \in [1 : t^*])$
repeat
  $U(i,l) = 0, H(i,l) = 0, U^{'}(i,l) = 0, F(i,1) = 0.$
  for *each cluster center* $\boldsymbol{c}_j$ **do**
    for *each pixel* $\boldsymbol{p}_i$ *in search region around* $\boldsymbol{c}_j$ **do**
      Compute distance d_{ij} between $\boldsymbol{p}_i$ and $\boldsymbol{c}_j$ by (1)
      if $F(i,1) < t^*$ **then**
        $D(i, n_i) = d_{ij}, G(i, n_i) = i, F(i,1)++$
      end
      else if $d_{ij} < \max_l D(i,l)$ **then**
        $l_\alpha = index(\max_l D(i,l))$, $D(i, l_\alpha) = d_{ij}$, $G(i, l_\alpha) = j$
      end
    end
  end
  for *each pixel* $\boldsymbol{p}_i$ **do**
    Find minimum distance value $d_{\min} = \min_l D(i,:)$
    Compute U by $U(i,l) = (\sum_{k=1}^{l_\beta} \left(\frac{(D(i,l))^2 - \alpha \cdot d_{\min}^2}{(D(i,k))^2 - \alpha \cdot d_{\min}^2} \right)^{\frac{1}{m-1}})^{-1}$,
    where $l_\beta = F(i,1)$.
  end
  for *each pixel* $\boldsymbol{p}_i$ **do**
    for *each neighbor* $\boldsymbol{p}_{i^*}$ *of pixel* $\boldsymbol{p}_i$ **do**
      if $G(i,l) = G(i^*, l^*)$ **then**
        $H(i,l) += U(i^*, l^*)$
      end
    end
    $U^{'}(i,l) = \frac{(H(i,l))^2}{\sum_{k=1}^{F(i,1)} (H(i,k))^2}$
  end
  for *each cluster center* $\boldsymbol{c}_j$ **do**
    Update $\boldsymbol{c}_j$ with CIELab and coordinate values and matrix $U^{'}$ by equation (5)
  end
  Compute residual error E, which is L_2 norm between the new cluster center position and the previous cluster center position.
until $E \leq \epsilon$ *or reaching* N^*;
for *each pixel* $\boldsymbol{p}_i$ **do**
  $L(i) = G(\text{argmax } U(i,:))$
end
for *each pixel* $\boldsymbol{p}_i$ **do**
  Enforce pixel $\boldsymbol{p}_i$ and its its 2×2 neighbors $\boldsymbol{p}_{i^*}$ with the same label.
end
Superpixel $R_{j'}$ is the set of pixels with label value $L = j^{'}$.

3.2. Post-merging with superpixels

Superpixel-based image segmentation is depicted mathematically as follows [37]: I is an image with superpixel sets $\{R_i\}$, final segment sets of I are $\{S_k\}$, $\mathscr{H}(\cdot)$ is the homogeneity criteria, then (i) $S_k \cap S_l = \varnothing$ and $R_i \cap R_j = \varnothing$ (ii) $I = \bigcup_k S_k = \bigcup_i R_i$, (iii) $\mathscr{H}(S_k \cup S_l) = False$, when $S_k = R_i \cup \ldots \cup R_j$ and $S_l = R_x \cup \ldots \cap R_y$. As known in the theory of image

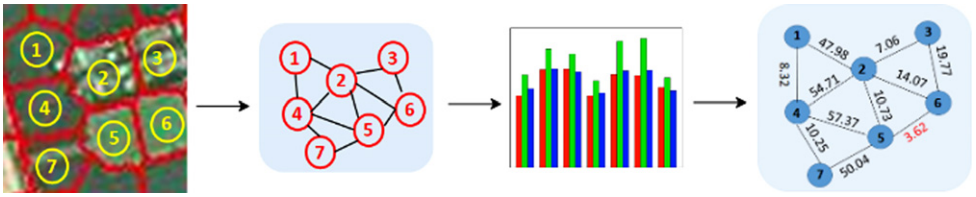


Fig. 3. An example of RAG construction based on superpixels.

segmentation with superpixels [38], adjacent superpixels is more likely to merge into a same segment, and vice versa.

RAG $= (V, E)$ can express adjacent relationships between segments obtained by the CMSuG method. Each vertex $v_i \in V$ in RAG represents an object, each edge $e_{ij} \in E$ connects a pair of adjacency vertex v_i, v_j, where $i \neq j$. The weight w_{ij} of edge e_{ij} is the dissimilarity of adjacent superpixels. With the relationship built in an RAG, post-merging can be done to gain the final segmentation results.

3.2.1. RAG construction

An example of RAG construction with superpixels generated with CMSuG method is shown in Fig. 3, which vividly express the relationship between superpixel and RAG. As seen in Fig. 3, a graph constructed with superpixels mainly contains two phrases, building edges between adjacency superpixels and computing the weight of each edge. Each edge reflects the adjacent relationship between two adjacent superpixels, the weight of each edge is the difference between the two neighborhoods.

Weights of edges measure the difference of two adjacent superpixels, which depend on the features of all superpixels. Color features of images are likely to catch dissimilarity for its generalization in all fields. Three color attributes of superpixels are computed by the mean of all pixels' attributions in a superpixel. Let $S_i(r_i, g_i, b_i)$ represent the feature vector of the region i, which is calculated by the following formulas:

$$r_i^* = \frac{1}{n_i}\sum_{k=1}^{n_i} r_k,\ \ g_i^* = \frac{1}{n_i}\sum_{k=1}^{n_i} g_k,\ \ b_i^* = \frac{1}{n_i}\sum_{k=1}^{n_i} b_k \tag{6}$$

where n_i is the total number of region i, the attributes of pixel k in the region are r_k, g_k, b_k. Then, w_{ij} is obtained by:

$$w_{ij} = \sqrt{(r_i^* - r_j^*)^2 + (g_i^* - g_j^*)^2 + (b_i^* - b_j^*)^2}. \tag{7}$$

We set 3-dimension matrix M_{col}^S, 1-dimension matrix M_{num}^S and 1-dimension matrix A to storage feature values of each superpixel, the total number of pixels that each superpixel contains, and the adjacent relationship between a pair of superpixels, respectively. The construction algorithm is as follows.

Algorithm2: Serial RAG construction

Input: Initial image I, Label matrix L of superpixel, number of superpixel K.
Output: RAG.
Create storage matrix A, M_{col}^S, M_{num}^S and initialize these matrix with all elements zero, where A is the size of $K * K$, M^S is $K * 3$ and M_{num}^S is $K * 1$.
for *each pixel* $\boldsymbol{p}_i$ **do**
  if $L(i) = l$, $l \in [0, K]$ **then**
    $M_{col}^S(l) += I(i)$, $M_{num}^S(l) += 1$
  end
  for *each neighbor pixel* $\boldsymbol{p}_{i*}$ *of pixel* $\boldsymbol{p}_i$ **do**
    if $L(i) \neq L(i^*)$ **then**
      $A(L(i), L(i^*)) = 1$
    end
  end
end
for $k = 1$ *to* K **do**
  $M_{col}^S(i) /= M_{num}^S(i)$
end
for $i = 1$ *to* K, $j = 1$ *to* K **do**
  if $A(i,j) = 1$ *and* $A(j,i) = 1$ **then**
    edge e_{ij} between vertex v_i and vertex v_j is constructed with difference between $M_{col}^S(i)$ and $M_{col}^S(j)$.
  end
end

3.2.2. RAG-based merging method

RAG-based merging algorithm is to obtain the segmentation result by some merging criterion. It means that two segments are merged into a new component by the way of measuring the difference of the two and a given criterion. The final segmentation result requires heterogeneous enough between the two adjacent segments, so do merging operation for two components $\mathcal{C}_i, \mathcal{C}_j$ if and only if $Diff(\mathcal{C}_i, \mathcal{C}_j) \leq MInt(\mathcal{C}_i, \mathcal{C}_j)$.

In the criterion, there is at least one vertex in $\mathcal{C}_i$ and $\mathcal{C}_j$ respectively, and the degree is greater than zero, $Diff(\mathcal{C}_i, \mathcal{C}_j) = \min_{v_i \in \mathcal{C}_i, v_j \in \mathcal{C}_j, e_{ij} \in E} w_{ij}$ is the minimum weight edge connecting a pair of vertexes concluded in $\mathcal{C}_i$, $\mathcal{C}_j$ respectively. $Diff(\mathcal{C}_i, \mathcal{C}_j) = \infty$ when there is no edge connecting $\mathcal{C}_i$ and $\mathcal{C}_j$. The minimum inter-difference in single component is defined as, $MInt(\mathcal{C}_i, \mathcal{C}_j) = \min(Int(\mathcal{C}_i) + \tau(\mathcal{C}_i),\ \ Int(\mathcal{C}_j) + \tau(\mathcal{C}_j))$, in which $Int(\mathcal{C}_i) = \max_{v_{i1}, v_{i2} \in \mathcal{C}_i} w(e_{i1,i2})$ and

$\tau(\mathcal{C}_i) = k/|\mathcal{C}_i|$. The threshold function $\tau(\mathcal{C}_i)$ is to suppress the generation of too large merging regions by adjusting the inter-difference, where k is a constant parameter and $|\mathcal{C}_i|$ denotes the number of component $\mathcal{C}_i$. In addition, $Int(\mathcal{C}_i) = 0$ if $|\mathcal{C}_i| = 1$. So, the criterion is that do merging if the difference between $\mathcal{C}_i$ and $\mathcal{C}_j$ is less than the minimum weight in the two internal components.

Initial component $\mathcal{C}_i$ is a single superpixel, which is a vertex in RAG. Merging procedure runs with all units $\mathcal{C}_i$, it is as follows,

Algorithm3: RAG-based merging

Input: RAG $= (V, E, W)$, where $v_i \in V, e_{ij} \in E, w_{ij} \in W$.
Output: Segmentation results $\mathbb{R}^* = R_1^*, \cdots, R_m^*$.
Set initial segmentation $\mathcal{S}_0 = (\mathcal{C}_1, \cdots, \mathcal{C}_n)$, where $\mathcal{C}_i = \{v_i\}$ represents the ith superpixel.
Sort the edges in non-decreasing order of weight w_{ij}.
while $E \neq \emptyset$ **do**
  if $w_{ij} \leq MInt(\mathcal{C}_i, \mathcal{C}_j)$ **then**
    Merge $\mathcal{C}_i$ and $\mathcal{C}_j$ to a new component $\mathcal{C}_{i'}$. Then insert $\mathcal{C}_{i'}$ to segmentation $\mathcal{S}_i$, and remove $\mathcal{C}_i$, $\mathcal{C}_j$ from $\mathcal{S}_i$, update $\mathcal{S}_i$ to $\mathcal{S}_{i'}$.
  end
  Remove e_{ij} from E.
end
Segment R_k^* is all superpixels in the kth component of final segmentation $\mathcal{S}_m$.

3.3. Complexity analysis

As the fuzzy membership degree is updated with 8-neighbors for each pixel, and the fuzzy membership matrix is three dimensions, the computational time of the CMSuG algorithm is $O(11N) + O(z)$, where N is the total number of image pixels, z is the initial cluster number. Actually, CMSuG algorithm needs few times iterations to reach termination condition. Without considering the maximum iteration number, the CMSuG algorithm is $O(N)$ on account of $z \ll N$.

Serial RAG construction requires sequentially travelling all pixels and all superpixels to extract features, the time complexity of which is $O(N) + O(K)$. The time complexity of adjacent matrix extraction is $O(N * 8)$, which is because each pixel has 8-neighbors. At last, the time complexity of RAG output requests $O(K^2)$. It means that all superpixels need to travel. Then, the time complexity of serial RAG construction is $O(N) + O(K^2)$.

RAG-based merging algorithm travels all edges with $O(K)$ by traversing all vertex pairs, the total number of which is at most $8 * K$. The time complexity of the sorting operation is at least with $O(|E| \log(|E|))$, where $|E|$ is the total number of vertex pairs. Thus, the complexity of RAG-based algorithm with RAG is $O(K \log K)$ at most.

In the whole image segmentation method by using CMSuG method, the time complexity is $O(N) + O(K^2)$.

4. Parallel optimization

Because the time complexity of the whole image segmentation is related to an input image size N and the superpixel number K, the time-consuming cost is too expensive in image segmentation tasks. It is unbeneficial for real-time image processing. In the whole segmentation method, the CMSuG method and serial RAG construction algorithm are the processes that need much time.

Traveling all pixels in CMSuG and RAG construction can be modified with parallel thread on GPU. With the support of the parallel techniques, they are done with a thread per pixel.

4.1. Parallel CMSuG method

Each step, which is in the CMSuG method, runs in a way that processes all pixels sequentially. The expensive time cost is adverse for the application of real-time image processing. The troublesome computation problem becomes easier to move the image to GPU by assigning parallel threads. For the acceleration advantage of parallel threads of GPU, we propose a parallel CMSuG algorithm to modify the CMSuG algorithm. In the whole algorithm expected updating cluster centers, we set the thread number to be $16 * 16$ in a block, so the block number in a grid is $\lceil (h^I/16) \rceil * \lceil (w^I/16) \rceil$, where h^I is the height and w^I is the width of an input image I.

For clearly depicted, the main parallel procedures on GPU platform are as follows:

(1) Image space conversion. Transfer I from CPU to GPU with CUDA function *cudaMemcpyAsync*(). Convert image to CIELab color space by using one thread per pixel, and store the result into I^T.

(2) Cluster centers initialization. Use one thread for each cluster center to initialize cluster centers. For convenience, the block dimension is the same as image space conversion. Each cluster center contains center coordinates, total membership degree values of associated pixels, and color information. As seen that, the number of cluster centers is superpixel numbers in CMSuG method. Thus, a new map representing all cluster centers is created as M^C, which size is $h^C * w^C$, where $h^C = \lceil h^I/S \rceil$ and $w^C = \lceil w^I/S \rceil$ are the total number of clusters in horizontal and vertical

direction, and $S = \sqrt{N/K}$ is depicted in CMSuG method. In the new map, 3-dimension color vector, 2-dimension position vector, and 1-dimension membership degree information are contained.

(3) Finding multi-cluster associations. For each pixel in I^T, compute the distance from the current pixel to its closest nine clusters instead of using search space in the CMSuG method. The information of each cluster is storied in matrix M^C. The reason for doing above operations is that each pixel is visited by its nearest 3×3 cluster centers at most. Find the first t^* minimum distance values for each pixel and storage them in matrix D while calculating the distance. At the same time, save these clusters' labels that are the index of clusters into matrix G. The implication of D and G are the same as the CMSuG algorithm, the size of which is $h^I * w^I * t^*$. After that, set a small enough value for D to avoid being zero. Do them operations on one thread per pixel, and the block size is the same as the step (2). The block size also applies to the following step (4) and step (5).

(4) Updating membership matrix with competitive mechanism. Update t^*-dimension membership matrix U in dimensional order. Calculate H by combining each pixel with its 8-neighborhood. Update matrix $U^{'}$ with H. All above matrix is updated with one thread per pixel. The same definitions of U, H, $U^{'}$ are shown in the CMSuG method.

(5) Calculating membership matrix-based CIELab images and coordinate matrices. Computing them by combining original CIELab I^T and a membership matrix $U_l^{'}(U_l^{'} = U^{'}(:, l), l \in [1, t^*]$. In the calculated operation, each pixel is calculated with a thread, and new CIELab images are stored in $I_1^m, \cdots, I_{t^*}^m$. The same calculation is done to obtain new coordinate matrices by combining original coordinate and membership matrix $U_l^{'}$, and the results are stored in $X_1^m, \cdots, X_{t^*}^m$. The size of all matrices in I_l^m and X_l^m is $h^I * w^I * 3$ and $h^I * w^I * 2$, where $l \in [1, t^*]$.

(6) Updating centers $\boldsymbol{c}$ with multiple new CIELab and coordinate values. Update each cluster center using the pixels assigned to it. Two steps in this process are done in different thread assignment. Firstly, assign each pixel into the corresponding cluster block with 256-dimensional shared threads. Each cluster needs $9 * S * S$ storage memory to accommodate all pixels associate to it because that each pixel accesses to nine cluster centers. So, in this step, the block dimension is $h^C * w^C * \lceil (9 * S * S)/(16 * 16) \rceil$ as the total number of clusters is $h^C * w^C$. Sum all shared thread in the same block at the same time, and store the value into the first thread of each shared block. This step runs with t^* times since the above t^* new CIELab images $I_1^m, \cdots, I_{t^*}^m$ and possible clusters $X_1^m, \cdots, X_{t^*}^m$ need to be done. These sum results are written to three new maps $M_1^R, \cdots, M_{t^*}^R$ of size $h^R * w^R$, where $h^R = h^C * \lceil (9 * S^2)/(16^2) \rceil$, $w^R = w^C$. These maps take along three kinds of information, which is the same as M^C.

Secondly, the values that are storied temporarily in $M_1^R, \cdots, M_{t^*}^R$ are reduced to the corresponding center and then an updated M^C is obtained with the summation of $M_1^R, \cdots, M_{t^*}^R$. It means that all values with a size of $h^C * w^C * \lceil (9 * S^2)/(16^2) \rceil$ in three maps are reduced to one map of size $h^C * w^C$. In this step, the block number in a grid is $h^C * w^C$.

(7) Selection of optimal cluster association. Select the maximum membership degree value $U_l^{'}$ from multiple values in $U^{'}$ and then put the value in G_l into final label matrix L by using one thread per pixel. The size of L is $h^I * w^I * 1$. The blocks in this this process and the following two steps are the same as that in step (2).

(8) Enforce connectivity. Eliminate isolated pixels with one thread per pixel to prompt the connectivity of each superpixel. If the label of each pixel and its surrounding pixels are all different, change it as its neighborhood. At last, transfer label L from GPU to CPU by CUDA function *cudaMemcpy*().

4.2. Parallel RAG construction

Two steps in RAG construction, feature extraction, and adjacent matrix extraction, require traveling all pixels, which is time-consuming in the whole image segmentation. So, we propose a parallel RAG construction method with the advantage of GPU to improve the two processes. Our parallel RAG construction method consists parallel feature extraction and parallel adjacent matrix extraction based superpixels. The flowchart of parallel RAG construction is shown in Fig. 4.

First, transfer I and L from CPU to GPU with CUDA function *cudaMemcpyAsync*(), where I is the initial image and L is obtained with parallel CMSuG method. In the whole process, the number of threads in each block is $16 * 16$.

One step is computing adjacent matrix, adjacent edges are found from each pixel and its 4-neighborhood with a thread per pixel. If the label of current pixel is different with its all neighbors, the value of current position in adjacent matrix is set

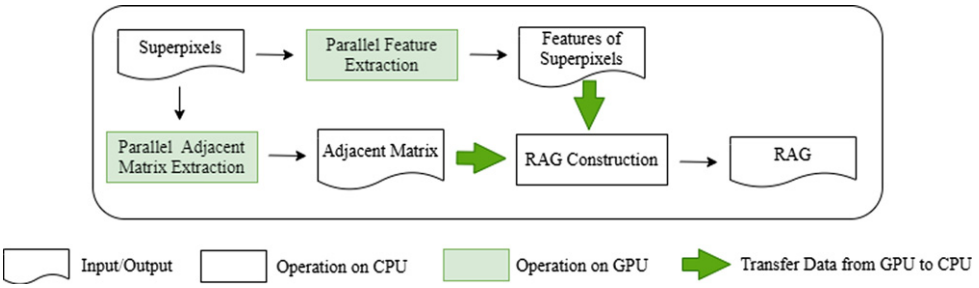


Fig. 4. Flowchart of parallel RAG construction.

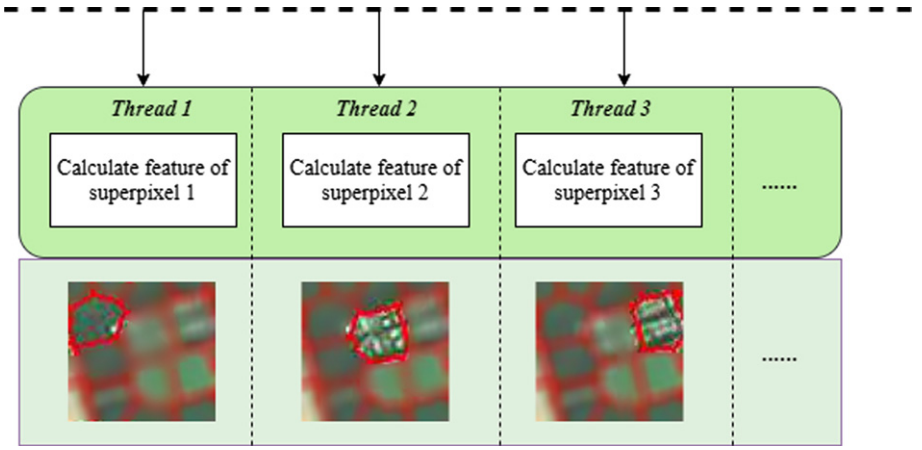


Fig. 5. An example of parallel feature extraction process based on superpixels.

as 1. Otherwise, it is set as 0. The block number is $\lceil (h^I/16) * (w^I/16) \rceil$ in this step, where h^I, w^I, I is the same as parallel CMSuG method. And the adjacent edges are storied in matrix A. For convenience, let the size of matrix A be $K * K$ with the pre-set number K of superpixel.

The other step is extracting color features of each superpixel by by using a similar operation as updating cluster centers in parallel CMSuG algorithm. The purpose of parallel feature extraction is to compute features of all vertexes by assigning a superpixel to only one thread. The description is shown vividly in Fig. 5. Construct a new map M^S to storage superpixel information, the map contains color information M^S_{col} and the total number of pixels M^S_{num} in each superpixel. Before the reduction operation of gaining M^S, each cluster mush search $9S * S$ pixels. Sum all the pixels that belong to a superpixel at the first time and then story them to a map M^{R^*}, which contains the same information as M^S. The block number and thread assignment is set as cluster center updating in step (6) of parallel CMSuG method. In addition, the size of M^S and M^{R^*} are the size of M^C and M^R respectively.

At last, transfer adjacent matrix A and feature matrix of superpixels M^{R^*} from GPU to CPU by CUDA function *cudaMemcpy*().

4.3. Complexity analysis

Although, data transform between CPU and GPU is not be ignored in parallel computing. This process is improving with the optimizing CUDA function continuously. Without considering transform tasks, the time complexity of the parallel processing is analysed.

In the parallel CMSuG algorithm, recurrence in updating centers except for computing fuzzy partition matrix. The time complexity of parallel reduction is $O(\lceil 9 * (N/K)/16^2 \rceil * t^*)$ with the need for three maps being reduced, where K is the pre-set number of superpixels, t^* is the number of multi-clusters. With visiting the 8-neighborhood in computing fuzzy

partition membership, so the time complexity of the parallel CMSuG algorithm is $O(N/K)$ by the constant t^*.

The adjacent matrix extracted only need four times cycle, and the time complexity of feature extraction is $O(\lceil 9 * (N/K)/16^2 \rceil)$ in parallel RAG construction. However, the RAG construction travels all elements in an adjacent matrix with size $K * K$. Compared with series RAG construction, the time complexity of parallel RAG construction is reduced from $O(N) + O(K^2)$ to $O(K^2)$.

By using the parallel technique in the whole image segmentation method, the time complexity is either $O(N/K)$ or $O(K^2)$, which depends on the value of $\sqrt{N/K}$ and K. In either case, the time complexity is far less than the complexity without parallel method $O(N) + O(K^2)$.

5. Experiments

5.1. Experiment settings

Research in images with undisclosed data is becoming more and more popular, the universality of the algorithm is particularly important. For that, we used the remote sensing data set (RSD) of the sea surface from Google Earth [39], which includes 50 images with 500×312 size. The data set are produced by our group, and the ground truth is made by three professionals. Then, we selected 50 nature images from BSD500 as the verified data sets, the size is 321×481.

There are certain parameters in our image segmentation method, the color initialization factor is 10 according to experience in previous papers [2]. Let fuzzy partition matrix exponent m be 2 by the introduction in FSLIC [15]. Most pixels do not have more than 3 labels that have been proved in FSLIC, so we adopted the maximum cluster number $t^* = 3$. Let ξ be 16 in our experiments motivated by some experiments in previous papers [31]. Similar to FSLIC in [15], we set the maximum iterative number $N^* = 10$. We selected parameter α as 0.9 according to experience and experiments.

All algorithms are programmed by Visual Studio 2019 with OpenCv 4.5 library and tested on a Windows 10 64-bit with Intel CPU Xeon E5-2689 at 2.6GHz, an NVIDIA GeForce GTX 1070Ti 8GB and 16GB RAM.

5.2. Benchmark metrics

Each step in our whole method, which contains superpixel generation, RAG construction, and graph-based image merging, influences the performance of final segmentation results. Thus, we compared the performance of these procedures individually. We evaluated the state-of-the-art superpixel generation algorithms FSLIC [15] and our CMSuG method, parallel FSLIC designed by us and our parallel CMSuG method by: boundary recall (BR) [40], boundary precision (BP) [26], Under-segmentation error (UE) [2, 26] and Compactness (CO) [41]. To test the segmentation results with a graph-based merging algorithm, we computed the BR, BP, and UE to depict the influence of different superpixel generation methods. The running time of these algorithms, which consist of superpixel generation, RAG construction, RAG-based merging algorithms, are displayed.

5.2.1. Boundary recall

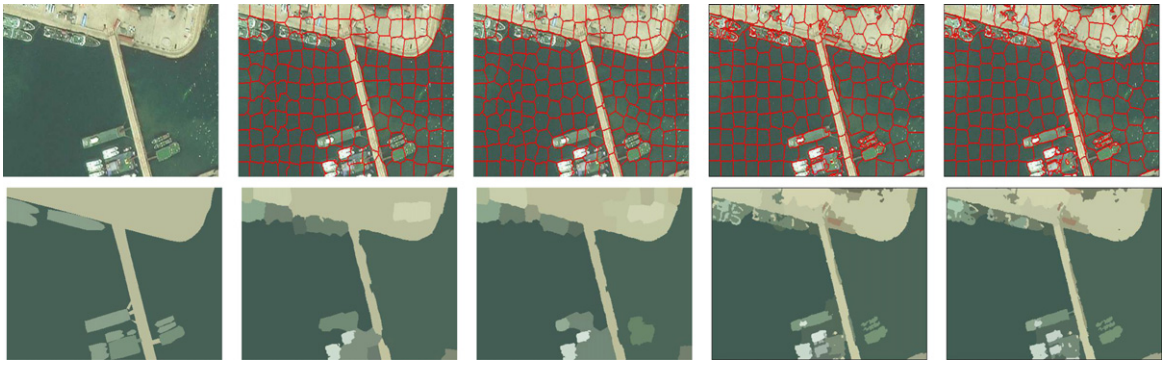
Boundary adherence of superpixel is the most property of measuring segmentation methods [2]. Boundary recall(BR) measures the amount of boundary pixels coverage on superpixels and ground truth. A high BR indicates that a very high correct boundary hit rate can be shown. Usually, minor errors are allowed between superpixels and ground truth. Then, BR is calculated by:

$$BR = \frac{1}{N_g}\left(\sum_{j=1}^{N_g} \tau\left(\min_{(x_i, y_i) \in B_s} \left\|(x_i, y_i) - (x_j, y_j)\right\| < \gamma\right)\right), \tag{8}$$

where $\tau(\chi) = 1$ when χ is true and $\tau(\chi) = 0$ otherwise, N_g is the pixel number of ground truth located in the boundary. The position of boundary pixel i in superpixels and pixel j ground truth are denoted as (x_i, y_i), (x_j, y_j), and B_s is the set of pixels of superpixels boundary. And $\gamma = 2$ is allowed in BR.

5.2.2. Boundary precision

BR only considers the true positive boundary rate for superpixels, while boundary precision (BP) adds the performance of false-positive ratio for judging accuracy. Poor BP score is possibly accompanied by extremely high BR. BP is a criterion for judging the positive superpixel boundaries in all superpixel boundaries, then we calculate the BP by



(a) Ground Truth (b) FSLIC (c) CMSuG (d) Parallel FSLIC (e) Parallel CMSuG

Fig. 6. The first line images are original RSD image and superpixels with FSLIC, CMSuG, Paralle FSLIC, Parallel CMSuG on 200 scale. The second line are ground truth and segmentation results with merging threshold $\tau = 40$.

$$BP = \frac{1}{N_s}\left(\sum_{j=1}^{N_g} \tau\left(\min_{(x_i, y_i)\in B_s} \left\|(x_i, y_i) - (x_j, y_j)\right\| < \gamma\right)\right), \tag{9}$$

where N_s is the amount of pixels located in the superpixel boundary.

5.2.3. Under-segmentation error

Under-segmentation error was proposed to evaluate the overlap error that computing the leakage pixel from superpixels across the boundary of ground truth [2]. Recent state-of-the-art works tend to use improved UE in [42], which is proposes to overcome both sides penalizes happening in a superpixel crosses a ground truth boundary. As below, the formula of UE is

$$UE = \frac{1}{N}\sum_{S_i}\sum_{G_j} \min\{|S_i \cap G_j|, |S_i - G_j|\}, \tag{10}$$

where S_i is the ith superpixel region, G_j is the jth segment of ground truth. The intersection of superpixel i and ground truth segment j is depicted with $S_i \cap G_j$, the left of region S_i after removing the intersection part can be indicated as $S_i - G_j$ and $|\cdot|$ is the absolute value operator. A high value of UE means that superpixels fit the ground truth with a higher inaccuracy.

5.2.4. Compactness

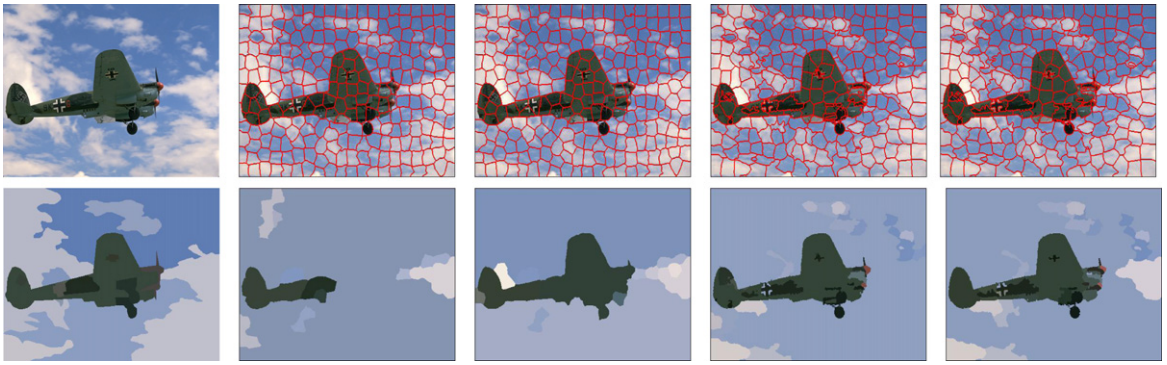
Compactness (CO) of shape is the important index for superpixel generation, it is considered to produce the superpixel with regular shape and smooth boundaries. It is likely to capture information from compact superpixel boundaries while not easily extract many and diverse boundary information with non-compact superpixels. It is denoted as:

$$CO = \frac{1}{N}\sum_{S_i} |S_i| \frac{4\pi A(S_i)}{(P(S_i))^2}, \tag{11}$$

where $A(S_i)$ and $P(S_i)$ are the area and perimeter of region S_i. The ability to produce a circle region is the regularity of a superpixel, that is why a higher CO value is better.

5.3. Qualitative analysis

To evaluate the influence of different superpixel generation methods in image segmentation results, we have done some experiments in two images from different data sets. We present a visual comparison of the two segmentation results in Figs. 6 and 7. As shown in the two figures, the segmentation results with FSLIC, CMSuG, parallel FSLIC, and parallel CMSuG superpixel generation methods are displayed clearly. We selected 200 as the pre-segmentation scale in the two example images from RSD and BSD. In order to avoid the threshold is too large and cause under-segmentation, let $\tau = 40$. The performance of these segmentation results is depicted in Tables 1 and 2. BR, BP, UE, and CO measure the boundary recall, boundary precision, under-segmentation error, and compactness of all superpixels generated by different superpixel generation methods. Correspondingly, BR_M, BP_M, and UE_M are boundary recall, boundary precision and under-segmentation error for testing image segmentation results after running graph-based merging algorithm. Running time reflects the efficiency of each algorithm. We set TIME_S, TIME_R, TIME_{PR}, and TIME_M as the time of running superpixel generation algorithm,



(a) Ground Truth (b) FSLIC (c) CMSuG (d) Parallel FSLIC (e) Parallel CMSuG

Fig. 7. The first line images are original BSD image and superpixels with FSLIC, CMSuG, Parallel FSLIC, Parallel CMSuG on 300 scale. The second line are ground truth and segmentation results with merging threshold $\tau = 40$.

Table 1
Performance metrics of superpixel segmentation algorithms at K=200 and merging threshold with $\tau = 40$ on an RSD image

Algorithm	BR	BP	UE	CO	BR_M	BP_M	UE_M	$TIME_S$/s	$TIME_R$/s	$TIME_{PR}$/s	$TIME_M$/s
F-SLIC	0.5110	0.1354	0.0579	1.1304	0.4000	0.4535	0.0684	7.6240	0.8631	0.0035	**0.8040**
CMSuG	0.5516	0.1494	0.0519	**1.1839**	0.4833	**0.4690**	0.0659	6.7809	0.9212	**0.0019**	1.2421
Parallel FSLIC	0.7558	0.1759	0.0417	0.9886	0.7218	0.4045	0.0496	0.0874	**0.6009**	0.0040	1.2164
Parallel CMSuG	**0.7602**	**0.1792**	**0.0373**	1.0069	**0.7321**	0.4300	**0.0420**	**0.0564**	0.8661	0.0022	1.0074

RAG construction method, parallel RAG construction method, and post-merging method respectively. In the table, all values with bold font indicate the best performance of all algorithms.

5.3.1. Analysis of RSD

Table 1 depicts the performance of different image segmentation methods for an RSD image. Obviously, all performance expect running time of our image segmentation with CMSuG method is better than that of image segmentation with FSLIC method. Superpixels generated with our parallel CMSuG method perform best in BR, BP, and UE, while the best CO value is obtained by using our CMSuG method. As demonstrated in Table 1, superpixels generated by CMSuG algorithm exceed that by FSLIC. And we can see the same phenomenon that superpixels generated by parallel CMSuG and parallel FSLIC algorithms. That is to say, our superpixel generation method with competitive mechanism performs better than that without competitive mechanism method both in CPU and GPU platforms.

After merging an image with superpixels generated by the above algorithms, the BR_M and UE_M of parallel CMGuS method outperforms the other methods. The results in the better BR_M, BP_M, and UE_M value for CMSuG when comparing with FSLIC. Highest BP_M value of segmentation result with CMSuG method demonstrates that positive boundary is the highest. This value of parallel CMSuG method is also better than that of parallel FSLIC method.

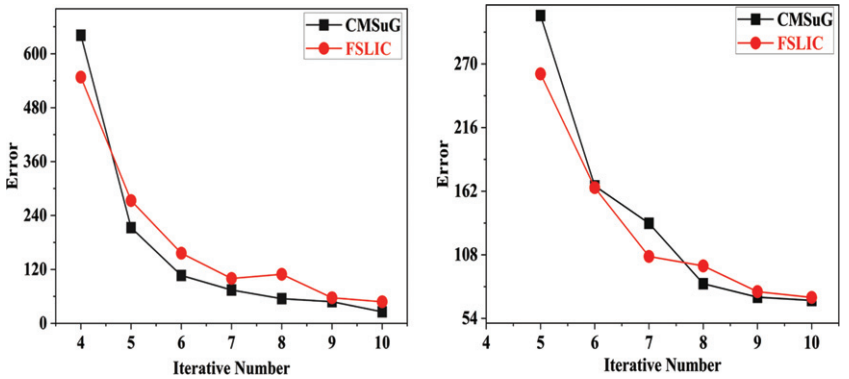
We can see that superpixels generated by parallel CMSuG method are done with the highest efficiency. And an obvious phenomenon is that the parallel superpixel generation method is faster than the ordinary method by about one hundred times. Running time of parallel RAG construction method is far less than RAG construction on CPU platform, in which CMSuG method is than FSLIC method whether to take parallel idea. However, the merging running time of our CMSuG method is the highest while FSLIC is the lowest. In general, we can see that the performance of our image segmentation method with the CMSuG method is better than that without competitive mechanism in both CPU and GPU platforms.

5.3.2. Analysis of BSD

In Table 2, boundary recall and compactness of superpixels produced by CMSuG method outperform other superpixel generation methods. And under segmentation error of CMSuG method is lower than FSLIC algorithm with 0.002. Compared with the

Table 2
Performance metrics of superpixel segmentation algorithms at K=300 and merging threshold with $\tau = 40$ on an BSD image

Algorithm	BR	BP	UE	CO	BR_M	BP_M	UE_M	$TIME_S$/s	$TIME_R$/s	$TIME_{PR}$/s	$TIME_M$/s
FSLIC	0.8557	**0.0395**	0.0837	1.1641	0.4147	0.1234	0.6246	7.1294	1.3735	0.0019	2.4076
CMSuG	**0.9434**	0.0340	0.0813	**1.2064**	0.8224	**0.2300**	**0.4086**	7.2244	1.3466	**0.0018**	**2.2534**
Parallel FSLIC	0.9364	0.0392	**0.0718**	1.0442	**0.8698**	0.1120	0.5385	0.1245	**1.1007**	0.0022	2.3813
Parallel CMSuG	0.9142	0.0385	0.0739	1.0615	0.8657	0.1119	0.5148	**0.1182**	1.2805	0.0024	2.3740



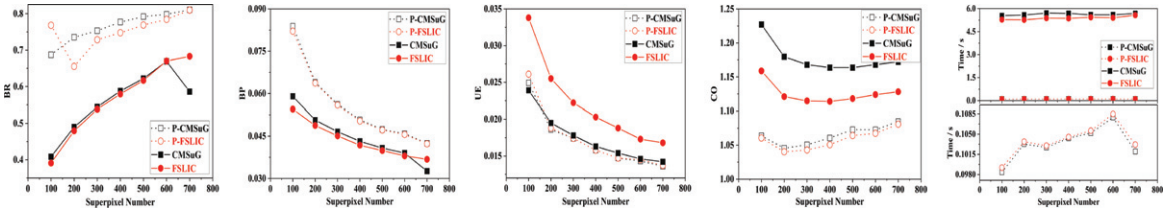
(a) image from RSD (b) image from BSD

Fig. 8. Convergence of CMSuG and FSLIC superpixel generation algorithms running on an example image, in which the image in left is obtained with an image from RSD and the right is from BSD.

parallel FSLIC method in CO performance with 1.0442, superpixels with a higher compactness value of 1.0615 are obtained by the parallel CMSuG method. Although, the other performance of parallel CMSuG method is slightly lower than parallel FSLIC method, the time cost is the lowest. As illustrated Fig. 7, the segmentation result accompanied with obvious wrong segments happens in image segmentation with FSLIC method by the large area of fuselage gone. Thus, all performance gained by FSLIC method is the worst. CMSuG performs best in boundary precision and under segmentation error. The value of BR_M is the highest of parallel FSLIC method, and the same measuring value along with the other two values of parallel CMSuG is slightly lower than it. We can see that the RAG-based image segmentation method with CMSuG algorithm obtains high-performance segmentation results in lower running time cost than FSLIC. Compared to image segmentation with parallel FSLIC, the running time of parallel CMSuG algorithm is slightly less than parallel FSLIC. Generally speaking, the image segmentation method by using competitive mechanism exceeds the non-competitive mechanism image segmentation method in CPU platform. In the GPU platform, the two parallel methods are about the same with a tiny difference. Our parallel RAG construction method is faster than the previous serial method by 700 times, which is beneficial for real-time image segmentation.

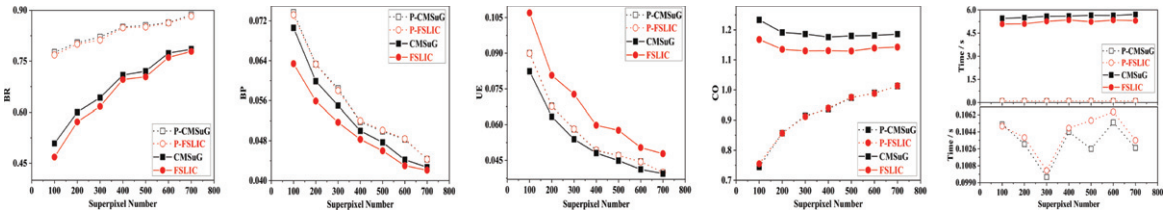
5.3.3. *Convergence analysis*

In the above two examples, the segmentation results with competitive mechanism achieve significantly superior results. Our CMSuG method generates superpixels by suppressing the growth of competitors. This technique helps all clusters group into robust associations rapidly. When the iterative number increases, two periods from all convergence processes are plotted. With a big change of error values, a picture does not show us overall changes significantly. Figure 8 shows the contrast of error values of the two superpixel generation techniques both on the remote sensing image of Fig. 6 and nature image of Fig. 7. We can see that errors decline with a far-fast time by the values of CMSuG being below that of FSLIC. Our CMSuG method reaches convergence with lower errors at the iterative time 10. For a remote sensing image, the error value is 25.71 obtained by CMSuG method while that is 48.17 by FSLIC. Error value is 69.31 gained with the CMSuG method while the other is 71.85 at the 10*th* iteration on an image from BSD.



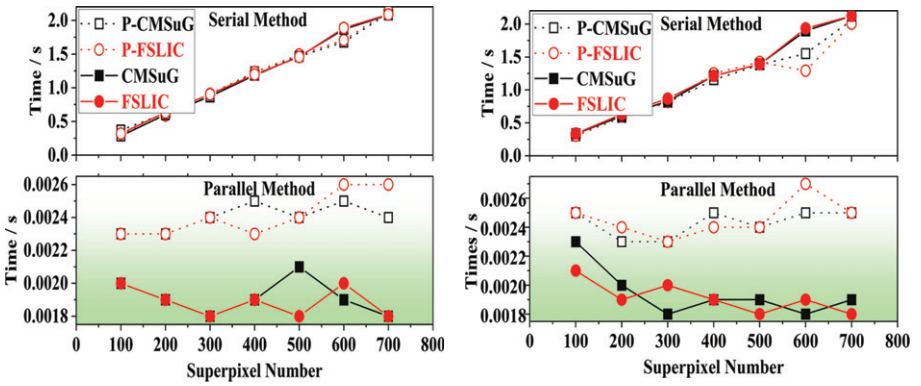
(a) BR of RSD (b) BP of RSD (c) UE of RSD (d) CO of RSD (e) Time of RSD

Fig. 9. Superpixels performance comparison of all methods in RSD.



(a) BR of BSD (b) BP of BSD (c) UE of BSD (d) CO of BSD (e) Time of BSD

Fig. 10. Superpixel performance comparison of all methods in BSD500.



(a) Time comparison of RAG construction on RSD

(b) Time comparison of RAG construction on BSD

Fig. 11. Time comparison of RAG construction based on superpixels generated with different superpixel generation methods, in which the graph in the left is obtained on RSD and the right is on BSD.

So, the competitive mechanism technique is beneficial for fastening convergence in the superpixel generation method.

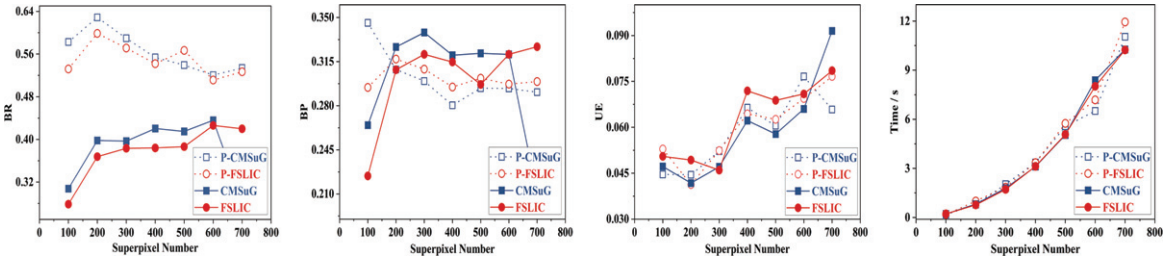
5.4. Quantitative analysis

Our image segmentation method consists of two steps: superpixel generation and post-merging with superpixels. The comparison of superpixel-based segmentation results is presented as certain line charts lie in Figs. 9 to 14. Similar to qualitative analysis, we use P-CMSuG and P-FSLIC as simplified forms of parallel CMSuG and parallel FSLIC in all pictures.

5.4.1. Analysis of superpixel generation methods

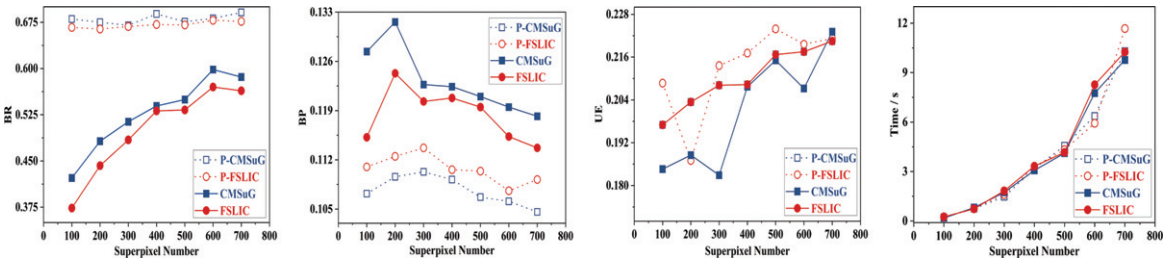
In this part, we analyzed 50 images from RSD with the above-mentioned metrics. The comparison of superpixels is presented as several line charts lie in Fig. 9. To verify the generalization of our algorithms, we selected 50 images from BSD500 and the performance is shown in Fig. 10.

It can be seen from Figs. 9 and 10 that the BR value of parallel CMSuG method is the highest in



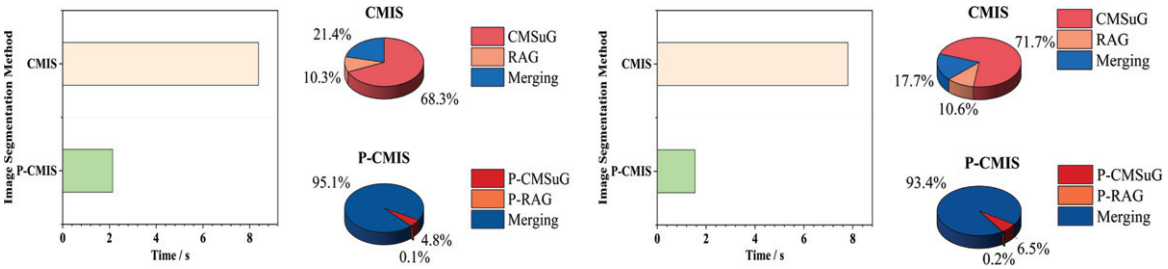
(a) BR_M of RSD (b) BP_M of RSD (c) UE_M of RSD (d) $Time_M$ of RSD

Fig. 12. The post-merging results analysis with different superpixels in RSD.



(a) BR_M of BSD (b) BP_M of BSD (c) UE_M of BSD (d) $Time_M$ of BSD

Fig. 13. The post-merging results analysis with different superpixels in BSD.



(a) Total time on RSD (b) Time division on RSD (c) Total time on BSD (d) Time division on BSD

Fig. 14. The whole time analysis with serial and parallel techniques in both RSD and BSD.

two dataset, and that of CMSuG is constantly higher than FSLIC in the whole sight. That is to say, superpixels generated by parallel CMSuG algorithm hit the boundary of ground truth with the maximum probability in the two data sets. And CMSuG also gains great probability of hitting the truth boundary by comparing the superpixel generation method without competitive mechanism. The same phenomenon is found in BP values, which is that BP values of parallel CMSuG is the highest and that of CMSuG is higher than FSLIC. The higher the BP values are, the more the positive boundaries. However, When the superpixel number grows to 700, BR and BP values are the lowest in RSD abnormally. An obvious conclusion is that competitive mechanism-based superpixel generation method obtains superpixels with perfect hitting possibility of boundaries and large number of positive boundary.

There is no significant difference of UE value between parallel CMSuG and parallel FSLIC. Under segmentation errors of parallel CMSuG is lower than CMSuG method in RSD while the opposite situation occurs in BSD. Whatever, FSLIC method obviously falls into a large under segmentation error with higher UE values. This indicates that competitive mechanism helps to achieve better superpixels.

The values of metric CO of the proposed method CMSuG are the highest, which shows that the CMSuG method is better than the FSLIC method for both remote sensing images and nature images. We can also see that CO values obtained by parallel CMSuG method is higher than parallel FSLIC

algorithm on RSD. The higher the CO value, the better the superpixels. It means that, competitive mechanism-based superpixel generation algorithm obtains much more regular and compact superpixels than that without competitive mechanism.

Because of the computation of finding the biggest competitor, the CMSuG algorithm need a more expensive time cost than FSLIC algorithm. While in the parallel CMSuG algorithm, it is improved by GPU with parallel threads. Using parallel technique for CMSuG method achieves 55 times faster than serial CMSuG method in the two dataset. Coincidentally, compared with FSLIC, parallel FSLIC algorithm also gets speed up 50 times. And the time cost of the parallel CMSuG superpixel generation method is shorter than parallel FSLIC. These results are plotted in Figs. 9(e) and 10(e), in which line charts in the small box are enlarged displayed the running time of parallel algorithms.

The above results demonstrated that our competitive mechanism-based superpixel generation algorithm in generating superpixels exceeds the-state-of-art algorithm FSLIC. For all the above analysis of superpixel generation methods, a notable conclusion is that our parallel CMSuG method could play an important role in real-time image processing.

5.4.2. Analysis of post-merging method

Our post-merging method consists of RAG construction and RAG-based merging with superpixels. The parallel RAG construction method is designed for decreasing time cost in the whole image segmentation. We make a comparison for the running time of serial and parallel RAG construction in Fig. 11. Each line represents the RAG construction time after obtaining superpixels in the two subgraphs. Serial construction is displayed at the top and parallel construction at the bottom. Superpixels generated with the same generation method are diagrammed with the same symbols.

It can be seen that parallel RAG construction time is significantly less than the serial algorithm by the maximum value of the ordinate. Time cost by running serial RAG construction method increases with the growth of superpixel numbers, while the time cost line oscillates at the same level by the parallel method. Parallel RAG construction method based on superpixels generated with parallel superpixel generation method takes 0.2 milliseconds more. Parallel RAG construction is faster than the serial method with at least 161 and 121 times on RSD and BSD respectively. No matter what scale for the two datasets, parallel RAG construction gains high-efficient improvement. So, in real-time image processing field, our parallel RAG construction can make image segmentation tasks more efficiently be completed.

For analysing the performance of the RAG-based merging method, we selected threshold $\tau = 20$ in the following experiments. Post-merging results construction of RSD and BSD is manifested in Figs. 12 and 13. In the two displayed results, we can see that the BP value of all merging results based on parallel CMSuG algorithm is the best. That value obtained with CMSuG method, meantime, is higher than that with FSLIC except for an outlier at size of 700 in RSD. High boundary precious of segments based on CMSuG method catches our sights with its highest values in BSD. However, these values after parallel CMSuG method are at the lowest in BSD. In RSD, BP_M of merging results with CMSuG method is higher than that of FSLIC except an outlier happens in 700. The results shown in 700 demonstrate that the segmentation results are related to the performance of superpixels. Under segmentation error of all segments after running merging algorithm with superpixels generated by using CMSuG method achieves the greatest success on both RSD and BSD data, which are shown in Figs. 12 and 13 clearly.

The running time of the graph-based merging algorithm is related to the number of superpixels, the more number the higher the time cost. The experiments show that merging time-based on all superpixel generation methods has no obvious difference. Actually, a large number of superpixels is unnecessary for image segmentation, which has great performance when set the superpixel number less than 400.

5.4.3. Analysis of whole segmentation method

We compare the segmentation results by our whole segmentation method and other three pixel-based segmentation methods, which contain MS [22] and FH [19] methods. For simplicity, we use CMIS as an image segmentation method with CMSuG and serial RAG construction. Similarly, the proposed image segmentation method with parallel CMSuG and parallel RAG techniques is simplified as P-CMIS. Tables 3 and 4 summary the performance of all segmentation methods, in which the BR*, BP*, UE* and TIME* represent boundary recall, boundary precision, under-segmentation error and running time with the whole segmentation methods respectively.

Although the BR value obtained by MS algorithm is higher, the BP values are the lowest and the

Table 3
Segmentation results comparison on RSD set

Algorithm	BR*	BP*	UE*	TIME*
MS	**0.7207**	0.0899	0.0463	233.1018
FH	0.3656	0.1058	0.0475	7.6699
CMIS	0.3980	**0.3266**	0.0445	7.0060
P-CMIS	0.6286	0.3087	**0.0418**	**0.8977**

Table 4
Segmentation results comparison on BSD set

Algorithm	BR*	BP*	UE*	TIME*
MS	**0.8728**	0.0465	0.1796	219.3879
FH	0.5521	0.1178	0.1814	8.3569
CMIS	0.4820	**0.1316**	**0.1785**	6.9187
P-CMIS	0.6750	0.1096	0.1934	**0.8195**

computation time is the highest in the two dataset. Our CMIS method can achive the best BP values, and the time is lower than that of MS and FH. As shown in the two tables, we can see that the values of the metrics TIME* of the proposed parallel method is the best. It demonstrates that our P-CMIS algorithm can quickly obtain the segmentation results. In addition, our CMIS method is faster than the MS and FH methods. In a word, both of our CMIS and P-CMIS methods have their own significant advantages. Our segmentation algorithm, not only parallel but serial methods, outperforms than the other two methods with its own advantages.

The running time of our two segmentation method has a huge difference. For that, we compared the total time of our competitive mechanism-based image segmentation method with parallel technique and without it. Figure 14 shows the total time of two image segmentation methods and its corresponding time assignment of each step running on two datasets with superpixel number 300.

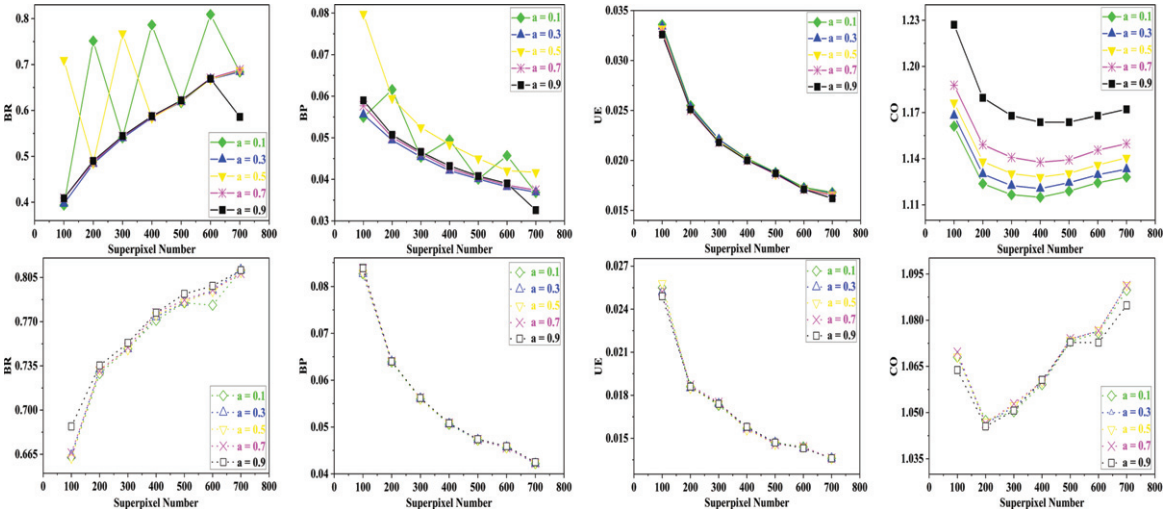
A fact that can be seen in the figure is that the total time of CMIS is four times of P-CMIS when run on two different datasets. In the whole improved image segmentation, parallel technique is mainly used to generate superpixels and build RAG, and because of this, we achieve the high-efficient image segmentation method.

5.5. Parameter analysis

There are two important parameters in our CMIS method, controlling factor α and merging threshold τ. Parameter α is used to control the growth of rivals, which impacts on generating superpixels. Threshold τ controls the scale of all segments, which in turn affects the segmentation results. We analyze the changes of superpixel and image segments with the two parameters respectively.

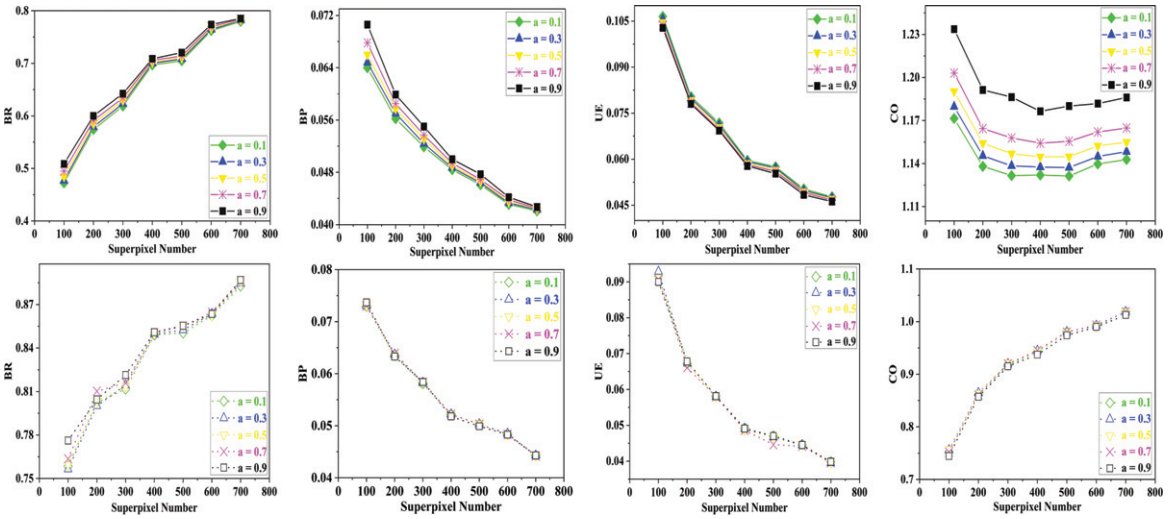
5.5.1. Analysis of α

For CMSuG method, 0.1/0.3/0.5/0.7/0.9 were selected to product superpixels. We analyze the influence of α on the generation of superpixels based on CMSuG and parallel CMSuG method in two dataset RSD and BSD, which are shown in Figs. 15 and 16.



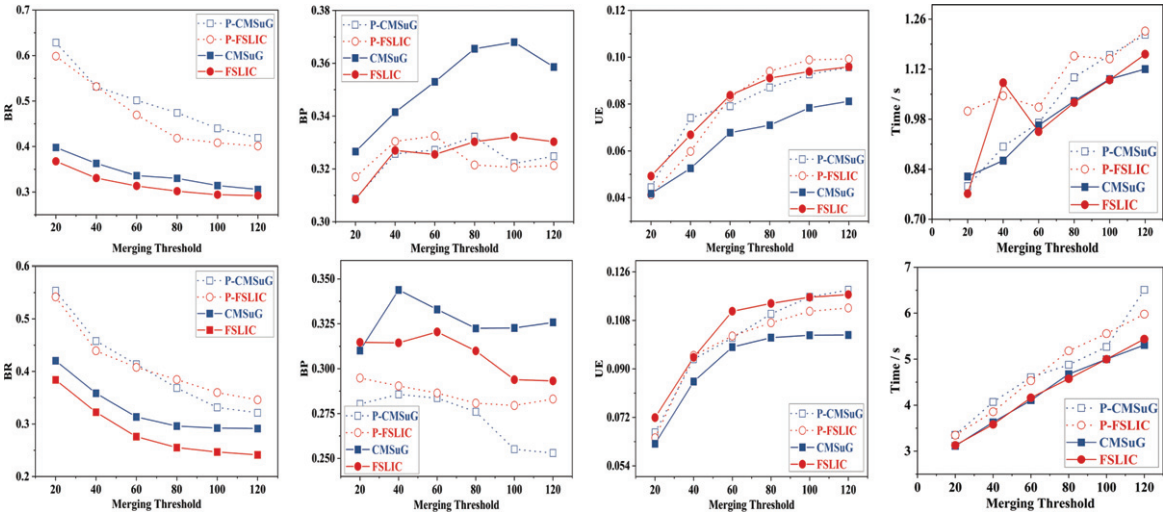
(a) BR of RSD (b) BP of RSD (c) UE of RSD (d) CO of RSD

Fig. 15. The performance comparisons of CMSuG and parallel CMSuG methods with different parameter α in RSD.



(a) BR of BSD (b) BP of BSD (c) UE of BSD (d) CO of BSD

Fig. 16. The performance comparisons of CMSuG and parallel CMSuG methods with different parameter α in BSD.

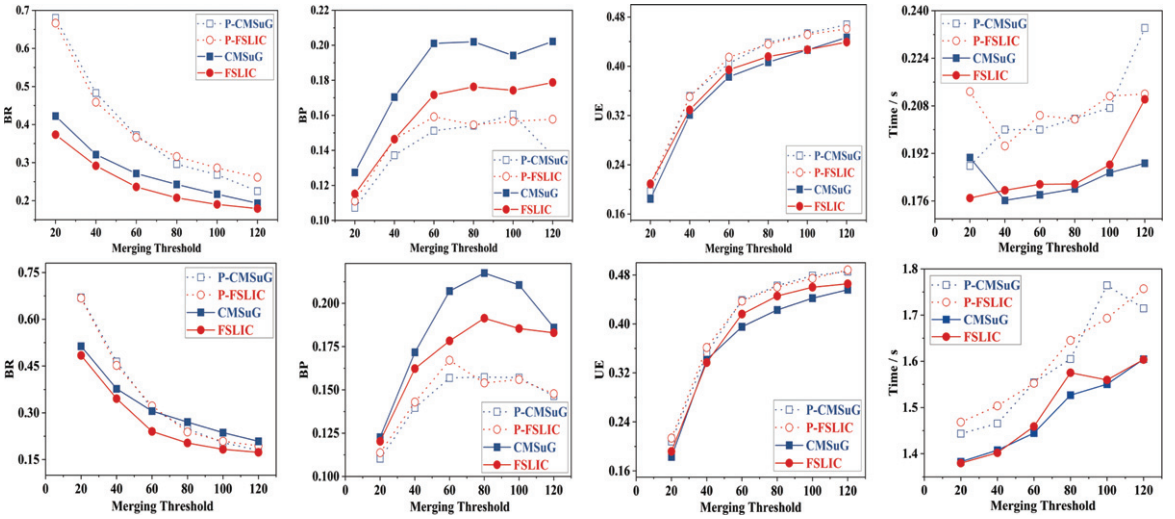


(a) BR_M of RSD (b) BP_M of RSD (c) UE_M of RSD (d) $Time_M$ of RSD

Fig. 17. The post-merging results analysis with 200 superpixels in RSD is shown in first line. And the second line is the results with 400 superpixels.

CMSuG algorithm fluctuates irregularly in boundary performance for RSD when $\alpha = 0.1$ and $\alpha = 0.5$. In BSD, the BR values increase with the development of α, the best performance is gained with $\alpha = 0.9$. Good performance of BR values obtained by parallel CMSuG method are illustrated in both RSD and BSD when $\alpha = 0.9$. As can be seen in Fig. 16, the highest BP value is also obtained with $\alpha = 0.9$ by CMSuG method on BSD dataset. In Fig. 15, expect the outliers $\alpha = 0.1/0.5$, CMSuG performs well with $\alpha = 0.9$. Both superpixels of RSD and BSD, which are generated by parallel CMSuG method, have no difference in charts of BP values with different α.

Different UE values obtained by the CMSuG method are shown in Fig. 16. We can see that under segmentation errors are the lowest when $\alpha = 0.9$. There is no obvious difference in all UE values generated by parallel CMSuG method whether in RSD or BSD. With $\alpha = 0.9$, superpixels generated with CMSuG method are the most compact and regular.



(a) BR_M of BSD (b) BP_M of BSD (c) UE_M of BSD (d) $Time_M$ of BSD

Fig. 18. The post-merging results analysis with 100 superpixels in BSD is shown in first line. And the second line is the results with 300 superpixels.

As the parameter increases, the degree of restraint on competitors strengthens. Thus, superior superpixels happen to competitors with a low probability of becoming a cluster.

5.5.2. Analysis of τ

In order to evaluate the image segmentation results by different merging threshold τ selected as 20/40/60/80/100/120, we used the fixed variable of superpixel number as 200/400 in RSD and 100/300 in BSD.

After generating superpixels, image merging results are analysed in Figs. 17 and 18. With increasing merging threshold, the performance of boundary recall values decreases on both RSD and BSD. BP value chart shows an upward trend followed by a downward trend. When merging threshold grows, under segmentation error increases significantly. Although, running time of merging step in CMIS increases with the growth of threshold τ. For merging time cost, the influence of threshold is less than superpixel number.

And an interesting phenomenon can be seen, merging operation with CMSuG superpixel generation method is always outstanding than FSLIC. Our experiments explain that the well-behaved merging result is CMSuG even with small numbers of objects. But selecting threshold with an adaptive method is necessary, which means that the same merging threshold is unsuitable for all number of superpixels.

6. Conclusion

Based on the FSLIC superpixel generation algorithm, we proposed a more robust superpixel generation method CMSuG by introducing competitive mechanism, which converges at a faster rate. The excellent performance of our method has been displayed by certain experiments compared to the state-of-the-art superpixel generation method in remote sensing dataset and nature dataset. However, good superpixels depend on superpixel scale, too fine pre-segmentation has a high number of superpixels. It will be more effective if a pending image has a suitable superpixel number by an adaptive method. Subsequently, outperforming segmentation results can be obtained by using a proposed superpixel-based post-merging algorithm with great superpixels. However, the merging result is affected by the threshold. Thus, adaptive threshold selection method could be of future interest.

To improve the speed of our segmentation method, a high-speed superpixel generation method, parallel CMSuG, was proposed based on GPU. And the superiority of its improvement greatly reduced the running time of CMSuG. Experiments in this paper showed the superiority of parallel CMSuG compared with parallel FSLIC. In addition, by using parallel techniques, a parallel RAG construction method was proposed. The two steps, parallel CMSuG and parallel RAG construction, speed up the overall segmentation algorithm fourfold. Because of the parallel

technique, real-time image processing becomes true with high performance. In our parallel method, computer resources are not being fully utilized, and more improved resource assignments will be our future research.

Acknowledgment

The authors would like to thank the google company for the favor of remote sensing images. The paper is supported by the National Natural Science Foundation of China under Grant No. 62072135. The paper is also supported by the Postdoctoral Program of Heilongjiang Hengxun Technology Co., Ltd.

References

[1] M. Hossain and D. Chen, Segmentation for object-based image analysis (obia): A review of algorithms and challenges from remote sensing perspective, *ISPRS Journal of Photogrammetry and Remote Sensing* **150**(APR) (2019), 115–134.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. SuSstrunk, Slic superpixels compared to state-of-the-art superpixel methods, *IEEE Transactions on Pattern Analysis Machine Intelligence* **34**(11) (2012), 2274–2282.

[3] M.Y. Liu, O. Tuzel, S. Ramalingam and R. Chellappa, Entropy rate superpixel segmentation, *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR* 2011, pp. 20–25.

[4] Fachao, Qin, Jiming, Guo, Fengkai and Lang, Superpixel segmentation for polarimetric sar imagery using local iterative clustering, *IEEE Geoscience and Remote Sensing Letters* **12**(1) (2015), 13–17.

[5] P. Avelar, A.R. Tavares, T. Silveira, C.R. Jung and L.C. Lamb, Superpixel image classification with graph attention networks, 2020.

[6] C. Peng, X. Gao, N. Wang and J. Li, Superpixel-based face sketch-photo synthesis, *IEEE Transactions on Circuits Systems for Video Technology* **27**(2) (2017), 288–299.

[7] M.M. Isaac, M. Wilscy, S.M. Thampi, E. El-Alfy, S. Mitra and L. Trajkovic, Image forgery detection using region – based rotation invariant co-occurrences among adjacent lbps, *Journal of Intelligent Fuzzy Systems* **34**(3) (2018), 1679–1690.

[8] H. Jin, Y. Yu, Y. Li and Z. Xiao, Network video summarization based on key frame extraction via superpixel segmentation, *Transactions on Emerging Telecommunications Technologies* **1** (2020), e3940.

[9] L. Zhang, S. Lu, C. Hu, D. Xiang, T. Liu and Y. Su, Superpixel generation for SAR imagery based on fast DBSCAN clustering with edge penalty, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **15** (2022), 804–819.

[10] W. Zhang, D. Xiang and Y. Su, Fast multiscale superpixel segmentation for SAR imagery, *IEEE Geoscience and Remote Sensing Letters* **19**, 2022.

[11] Q. Lin, W. Zhong and J. Lu, Deep superpixel cut for unsupervised image segmentation, *25th International Conference on Pattern Recognition* 2021, pp. 8870–8876.

[12] Q. Shi, S. Yin, K. Wang, L. Teng and H. Li, Multichannel convolutional neural network-based fuzzy active contour model for medical image segmentation, *Evolving Systems*, 2021.

[13] S. Di, M. Liao, Y. Zhao, Y. Li and Y. Zeng, Image superpixel segmentation based on hierarchical multi-level li-slic, *Optics Laser Technology* **135** (2021), 1–13.

[14] R. Alshehhi and P.R. Marpu, Hierarchical graph-based segmentation for extracting road networks from high-resolution satellite images, *Isprs Journal of Photogrammetry Remote Sensing* **126** (2017), 245–260.

[15] C. Wu, L. Zhang, H. Zhang and H. Yan, Fuzzy slic: Fuzzy simple linear iterative clustering, *IEEE Transactions on Circuits and Systems for Video Technology* **31**(6) (2021), 2114–2124.

[16] Z. Lin, F.L. Chung and S. Wang, Generalized fuzzy c-means clustering algorithm with improved fuzzy partitions, *IEEE Transactions on Cybernetics* **39**(3) (2009), 578–591.

[17] L. Xu and A. Krzyzak, Rival penalized competitive learning for clustering analysis, rbf net, and curve detection, *IEEE Transactions on Neural Network* **4**(4) (1993), 636–649.

[18] M.A. Hossam, H.M. Ebied, M.H. Abdel-Aziz and M.F. Tolba, Accelerated hyperspectral image recursive hierarchical segmentation using gpus, multicore cpus, and hybrid cpu/gpu cluster, *Journal of Real-Time Image Processing* **14**(2) (2018), 413–432.

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

[20] J. Shi and J.M. Malik, Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[21] S. Bu, Z. Liu, J. Han and J. Wu, Superpixel segmentation based structural scene recognition, *in Proceedings of the 21st ACM international conference on Multimedia*, 2013.

[22] Comaniciu, Dorin, Meer and Peter, Mean shift: A robust approach toward feature space analysis, *IEEE Transactions on Pattern Analysis Machine Intelligence*, 2002.

[23] L. Vincent and P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(6) (1991), 583–598.

[24] A. Levinshtein, A. Stere, K.N. Kutulakos, D.J. Fleet, S.J. Dickinson and K. Siddiqi, Turbopixels: Fast superpixels using geometric flows, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**(12) (2009), 2290–2297.

[25] Z. Li and J. Chen, Superpixel segmentation using linear spectral clustering, *in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[26] R. Achanta and S. Susstrunk, Superpixels and polygons using simple non-iterative clustering, *in IEEE Conference on Computer Vision Pattern Recognition*, 2017.

[27] R. Shams and N. Barnes, Speeding up mutual information computation using nvidia cuda hardware, *in 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications*, 2008.

[28] D. Reska and M. Kretowski, Gpu-accelerated image segmentation based on level sets and multiple texture features, *Multimedia Tools and Applications* **80**(1) (2021), 1–23.

[29] X.G. Sun, M.C. Li, Y.X. Liu, T. Lu and L. Wei, Accelerated segmentation approach with cuda for high spatial resolution

remotely sensed imagery based on improved mean shift, *in Proceedings of the Ninth National Member Congress of the Chinese Society of Surveying and Mapping and the 50th Anniversary of the Founding of the Society*, 2009.

[30] J. Xu, H. Yuan, S. Wu and H. Yu, Cuda-based parallel implementation of turbopix algorithm, *Microcomputer Its Applications*, 2013.

[31] C.Y. Ren, V.A. Prisacariu and I.D. Reid, gslicr: Slic superpixels at over 250hz, *Computer Science*, 2015.

[32] Z. Ban, J. Liu and J. Fouriaux, Glsc: Lsc superpixels at over 130fps, *Journal of Real-Time Image Processing* **14**(Special Issue) (2018), 605–616.

[33] X. Zhang, P. Xiao, X. Feng and G. He, Another look on region merging procedure from seed region shift for high-resolution remote sensing image segmentation, *ISPRS Journal of Photogrammetry and Remote Sensing* **148** (2019), 197–207.

[34] H. Liu, Q. Guo, M. Xu and I.F. Shen, Fast image segmentation using region merging with a k-nearest neighbor graph, *in IEEE Conference on Cybernetics Intelligent Systems*, 2008.

[35] Z. Zhao, B. Li, X. Kang, L. Chen and M. Xin, Hybrid image segmentation method based on anisotropic gaussian kernels and adjacent graph region merging, *Review of Scientific Instruments* **91**(015104) (2020).

[36] J. Li, Q. Hu and M. Ai, Unsupervised road extraction via a gaussian mixture model with object-based features, *International Journal of Remote Sensing* **39**(8) (2018), 2421–2440.

[37] H.D. Cheng, X.H. Jiang, Y. Sun and J. Wang, Color image segmentation: Advances and prospects, *Pattern Recognition* **34**(12) (2001), 2259–2281.

[38] Z. Fu, Y. Sun, L. Fan and Y. Han, Multiscale and multifeature segmentation of high-spatial resolution remote sensing images using superpixels with mutual optimal strategy, *Remote Sensing* **10**(8) (2018), 1–30.

[39] Google earth software. *http://earth.google.com/*, 2008.

[40] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik, Contour detection and hierarchical image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**(5) (2011), 898–916.

[41] A. Schick, M. Fischer and R. Stiefelhagen, Measuring and evaluating the compactness of superpixels, *21st International Conference on Pattern Recognition*, 2012, pp. 930–934.

[42] P. Neubert and P. Protzel, Superpixel benchmark and comparison, *Forum Bildverarbeitung*, 2012.