Remote Sens. **2013**, *5*, 3259-3279; doi:10.3390/rs5073259

OPEN ACCESS

Remote Sensing

ISSN 2072-4292

www.mdpi.com/journal/remotesensing

Article

Segmentation for High-Resolution Optical Remote Sensing Imagery Using Improved Quadtree and Region Adjacency Graph Technique

Gang Fu, Hongrui Zhao *, Cong Li and Limei Shi

Institute of Geomatics, Department of Civil Engineering, Tsinghua University, Chengfu Road, Beijing 100084, China; E-Mails: gangfu2008@hotmail.com (G.F.); yimi121072@gmail.com (C.L.); shilimi8@hotmail.com (L.S.)

* Author to whom correspondence should be addressed; E-Mail: zhr@tsinghua.edu.cn; Tel.: +86-010-6279-4967.

Received: 28 April 2013; in revised form: 21 June 2013 / Accepted: 23 June 2013 / Published: 5 July 2013

Abstract: An approach based on the improved quadtree structure and region adjacency graph for the segmentation of a high-resolution remote sensing image is proposed in this paper. In order to obtain the initial segmentation results of the image, the image is first iteratively split into quarter sections and the quadtree structure is constructed. In this process, an improved fast calculation method for standard deviation of image is proposed, which significantly increases the speed of quadtree segmentation with standard deviation criterion. A spatial indexing structure was built using improved Morton encoding based on this structure, which provides the merging process with data structure for neighborhood queries. Then, in order to obtain the final segmentation result, we constructed a feature vector using both spectral and texture factors, and proposed an algorithm for region merging based on the region adjacency graph technique. Finally, to validate the method, experiments were performed on GeoEye-1 and IKONOS color images, and the segmentation results were compared with two typical algorithms: multi-resolution segmentation and Mean-Shift segmentation. The experimental results showed that: (1) Compared with multi-resolution and Mean-Shift segmentation, our method increased efficiency by 3–5 times and 10 times, respectively; (2) Compared with the typical algorithms, the new method significantly improved the accuracy of segmentation.

Keywords: image segmentation; remote sensing imagery; quadtree; spatial indexing; region adjacency graph

1. Introduction

Image segmentation divides images into partitions, which is typically used to recognize objects or other relevant information in digital images [1]. For processing of remote sensing image, especially for high-resolution image, image segmentation is a primary step in classification or other analysis. For instance, object-oriented classification is a basic process for many applications including change detection [2] and land cover investigation [3]. High-resolution remote sensing images express more information of ground objects, and show great diversity of texture features.

The using of traditional pixel-based segmentation methods, such as K-Means and ISODATA, will obtain a large number of trivial segments. However, a lot of these segments are actually noise, causing serious impact to subsequent analysis. The main difficulty of image segmentation lies in efficient region generation and merging [4]. At present, there exist several studies on object-oriented image segmentation. The most representative studies include Mean-Shift (MS) algorithm [5], a method based on kernel density estimation, which is widely used in data clustering analysis. MS algorithm breaks the limitation of requiring knowledge of the prior cluster number. It has been successfully applied to feature space analysis [6] and texture image segmentation [7]. However, the drawback of the MS algorithm is computational complexity for local maxima searching in the feature space. Therefore, some researchers propose using a speedup mechanism such as Locality-Sensitive Hashing, K-D Tree [7–9]. Watershed Transform [10] is a data analysis method based on morphological technique. The algorithm treats each local minimum and surrounding area as a basin for catchments, and the basin watershed as a boundary of image segment. Based on this theory, a deeper analysis can be discussed on merging technique [11,12]. Multi-Resolution (MR) segmentation algorithm considers both spectrum and geometry factors as region growing criteria [13], which is implemented with the commercial software eCognition [14]. Graph-based technique is another class of image segmentation method. By abstracting pixels and their similar relationship as nodes and edges in a graph, this method splits objects by cutting their graph edge under certain optimization criteria [15]. Commercial software ENVI [16] adopts the graph-based Full Lambda-Schedule algorithm as its basic procedure of object-oriented information extraction [17].

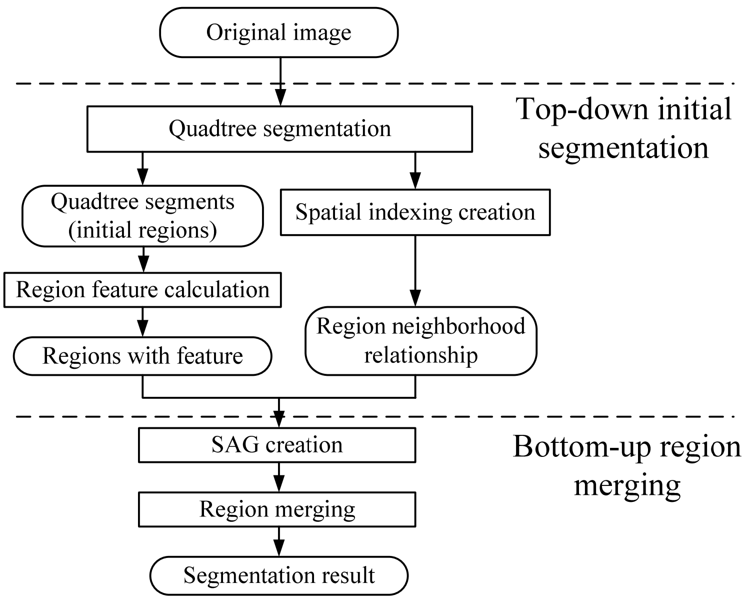
Quadtree structure method was proposed in 1974 by Raphael Finkel and J.L. Bentley [18]. It was originally used in creating vector data spatial indexing. Because image wavelet transform has a natural hierarchy quarter structure, many scholars applied quadtree-based method to image segmentation in the wavelet domain [19]. In spatial domain, for its simple form, the quadtree technique is used in several fields such as image coding and raster compression. In the image segmentation field, the core difficulties with the quadtree method are image splitting and region merging. To resolve these issues, splitting judgment by calculating boundary smoothness, contrast and edge information between sub regions [20], or uniformity of Gaussian-Markov random field of adjacent regions [21] have been proposed.

Currently, the typical bottom-up approaches (such as MS algorithm and Full Lambda-Schedule) always process from pixels, and are more likely to result in lots of trivial segments, causing great difficulty to post-processing. At the same time, remote sensing images are always huge in size. So a direct merging process based on pixels is time-consuming. Actually, for many existed algorithms, high complexity is one of the bottlenecks of their practical application. Top-down approaches such as quadtree decomposition always have a high efficiency. However, it will obtain regions of inconsistent sizes, making it difficult to conduct neighborhood searching for region merging. To overcome this difficulty and improve the efficiency of the neighborhood searching, we used the quadtree as the basic structure for analysis, and added the spatial indexing mechanism based on improved Morton coding. Finally, we conducted a region merging process based on Region Adjacency Graph (RAG).

2. Methodology

In general, our method has two major steps: a top-down initial segmentation step and a bottom-up region merging step. In the top-down step, a quadtree initial segmentation is performed first, providing the follow-up merging process with basic region elements. In this step, we also conduct a spatial indexing creation and region feature calculation, in order to provide the RAG creation with similarity foundation and neighborhood information. In the second bottom-up step, we use RAG to express the relationship between regions. In RAG, regions are represented by nodes. If two nodes are not adjacent (judged by region neighborhood relationship obtained from the previous step), there is no edge connected. Otherwise, they are connected by an edge, representing their similarity (in RAG creation procedure, similarities are calculated by region features calculated from the previous step). Region merging is performed between the most similar and adjacent regions in RAG. Region merging will stop until the smallest similarity in RAG is larger than a given threshold. After the merging process, the final segmentation result is obtained. Figure 1 is the general procedure of our method.

Figure 1. Flowchart of proposed method (use rectangular and round rectangular block to denote processing and input/output, respectively).


Original image
Top-down initial segmentation
Quadtree segmentation
Quadtree segments (initial regions)
Spatial indexing creation
Region feature calculation
Region neighborhood relationship
Regions with feature
SAG creation
Bottom-up region merging
Region merging
Segmentation result

2.1. Quadtree Initial Segmentation

Quadtree segmentation is a top-down approach, treating the whole image as the root node at the beginning, before the region is split into four rectangular sub-regions under a certain splitting criterion. The same determination is then conducted in each sub-region until all regions have satisfied the given criterion. Standard deviation is a statistical value, which can effectively measure the dispersion of a random sequence. It can characterize the variation of grey values of an image. Therefore, we use the local standard deviation as the consistency criterion for the splitting judgment. Since quadtree segmentation is an iterative algorithm, it requires great computation on standard deviation for each layer and each sub-region. So we propose a fast algorithm for standard deviation computation based on the integral image [22] and squared integral image. Then, a spatial indexing structure was built using improved Morton encoding based on this structure, which provides the merging process with data structure for neighborhood queries.

2.1.1. Fast Calculation of Standard Deviation Criterion

Quadtree segmentation based on the standard deviation splitting criterion needs to calculate the standard deviation of the sub-regions repeatedly. As the tree depth increases, the computation will also increase dramatically. In recent years, the Haar-Like feature has proven to be the most effective technique of face detection, and has been successfully applied to real-time face detection systems [22]. The Haar-Like feature requires repeated calculations of the summation of pixel values of any rectangular region. The use of the integral image makes it possible for this calculation to be done in constant time regardless of the size of the region. The integral image is defined as:

$$II(x,y) = \sum_{\substack{x'<x \\ y'<y}} I(x',y') \tag{1}$$

where $I(x, y)$ is the pixel value of the original image at location (x, y). It can be seen that the value of (x, y) in integral image represents the sum of pixels in the upper left area of the original image. By using the integral image, sum of pixels of any rectangular sub-region in the original image can be calculated in constant time, by using:

$$[II(x_2,y_2) + II(x_1,y_1)] - [II(x_1,y_2) + II(x_2,y_1)] \tag{2}$$

where (x_1, y_1), (x_2, y_2) are the upper left and lower right coordinates of the region, respectively. Avoiding pixel traversal, the integral image is of great significance for time sensitive applications. In addition, we constructed the squared integral image, which is defined as:

$$SII(x,y) = \sum_{\substack{x'<x \\ y'<y}} [I(x',y')]^2 \tag{3}$$

Through the combined use of the integral image and the squared integral image, we can calculate the standard deviation of any region in constant time with the following equation:

$$\sigma = \sqrt{\frac{D'_{2,1} - 2m \cdot D_{2,1} + N \cdot m^2}{N}} \tag{4}$$

where $N = (x_2 - x_1 + 1) \times (y_2 - y_1 + 1)$ is the pixel count of the region; $D_{2,1}$ is the pixel sum of the region calculated by Equation (2); $D'_{2,1} = [S\text{II}(x_2, y_2) + S\text{II}(x_1, y_1)] - [S\text{II}(x_1, y_2) + S\text{II}(x_2, y_1)]$ is the pixel squared sum and $m = D_{2,1}/N$ is the mean pixel value of the region.

Consider for example an image 512 × 512 in size. Quadtree segmentation was performed on the structure using the standard deviation criterion and the result is shown in Figure 2. We use number of elementary operations and pixel accesses to evaluate the time complexity. Elementary operations include addition, subtraction, multiplication, and division operations. Pixel accesses refer to obtaining pixel data from memory. The comparison is listed in Table 1.

Figure 2. An example of quadtree segmentation with four layers.

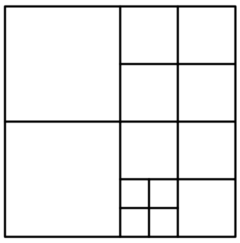


Table 1. Complexity comparison between traditional method and our method for quadtree segmentation using standard deviation criterion (using example shown in Figure 2).

Method	Addition and Subtraction	Multiplication and Division	Pixel Accesses
Traditional method	2015198	671778	1572864
Our method	524422	102	524228

From the comparison, we can see that improvement for efficiency is obvious. When a larger image is processed and the quadtree structure is more complex, this advantage will be more apparent.

2.1.2. Quadtree Segmentation and Spatial Indexing Creation

Quadtree Segmentation

Using average of standard deviation for all bands as splitting criterion, the entire image is treated as quadtree root, and then the iterative quadtree segmentation is conducted for each sub-region until all regions have satisfied the given criterion threshold.

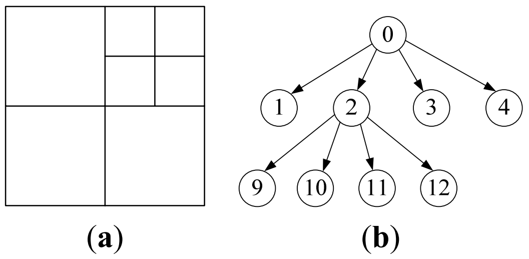
Quadtree segmentation splits images into four parts of the same size. However, for the image with a large length-width ratio (e.g., image strip along an imaging orbit), this method will produce narrow strip initial segments. So, before the segmentation, we will judge the image length-width ratio. According to experience, we use 1.5 as the length-width ratio threshold. When the ratio is larger than 1.5, the image will be cut along the longer edge direction, in order to make the length-width ratio for each slide less than 1.5. Then, we perform quadtree segmentation on each image slide independently, evading narrow strip segmentation. The following study focuses on images with a length-width ratio of less than 1.5.

Spatial Indexing Based on Improved Morton Coding

Region merging is performed between neighbouring nodes. Because segments generated from quadtree segmentation have different sizes, it is difficult to perform a quadtree neighbourhood searching between segments. Direct node traversal requires a great computation. Hence, an indexing mechanism for efficient neighbourhood searching needs to be constructed. We use Morton coding [23] as our basic method, and improve its efficiency, in order to make it more effective when applied to huge image segmentation.

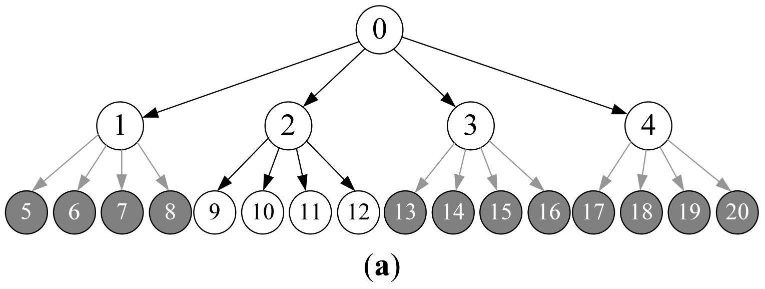
Morton coding assumes that the code of the root node (represents the whole image) is 0. For each of the inner nodes encoding as $i (i \geq 0)$. Codes of its 4 children are $4i + k$ (k = 1,2,3,4). Through this coding mechanism, the layer index of node i can be obtained by $[\log_4(3i + 1)]$ $(i \geq 0)$ and its parent code can be obtained by $[(i - 1)/4]$ $(i > 0)$. Figure 3 is an example of quadtree segmentation and its Morton coding.

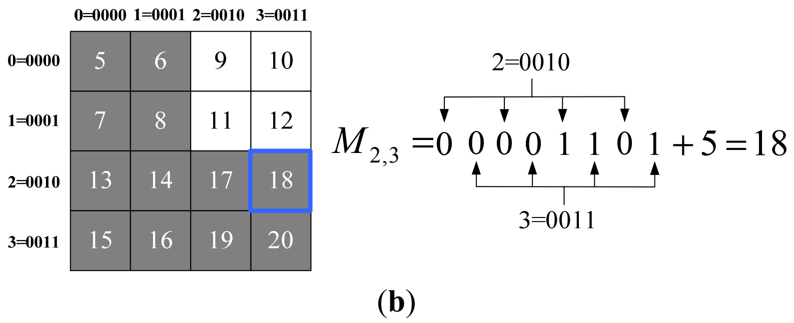
Figure 3. Quadtree segmentation and Morton coding (**a**) Image quadtree segmentation; (**b**) Corresponding quadtree with Morton coding.



Neighborhood searching based on Morton coding is completed through the construction of the Virtual Complete Quadtree (VCQ). VCQ is constructed by adding virtual nodes to the original incomplete quadtree. In VCQ, all leaves are at the same layer, and in which every parent has four children. Figure 4(a) is a VCQ example for quadtree shown in Figure 3(b). An incomplete quadtree corresponds to an inconsistent grid (original segmentation). VCQ corresponds to a uniform grid. By using the VCQ, the difficult neighborhood searching on inconsistent grid is transformed into simple neighborhood searching on uniform grid. Coding each cell in the uniform grid by using Morton code $M_{i,j} = M'_{i,j} + \sum_{i=0}^{i<n} 4^i$, where $n \geq 0$ is the total layer of the quadtree. $M'_{i,j}$ is the binary interleave coding of row index i and column index j, in the decimal representation. Take location at row 2 and column 3 as an example. Binary expressions of row and column are 2 = 0010, and 3 = 0011, respectively. So, the binary interleave is 00001101, and its decimal value is 13. Therefore, its Morton code is $13 + \sum_{i=0}^{i\leq 2} 4^i = 18$. See the right schematic in Figure 4(b) for detail of constructing binary interleaves.

Figure 4. Virtual Complete Quadtree (VCQ), uniform grid and the Morton coding of the example shown in Figure 3 (the grey part is the virtual structure). (**a**) VCQ structure of the example shown in Figure 3 and (**b**) the corresponding uniform grid and Morton coding.





A State Lookup Table (SLT) with size $\sum_{i=0}^{i\leq n} 4^i$ also needed for recording the state of each node. The states denote the position and attribute of nodes in the VCQ, including:

RI: Real inner nodes in VCQ. Also include the real nodes having virtual offspring.
RL: Real leaf nodes in VCQ (real nodes at deepest layer).
VN: Virtual nodes in VCQ. Also include virtual inner nodes.
Table 2 shows the SLT for above example.

Table 2. A State Lookup Table (SLT) for above quadtree example.

Node code	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
State	RI	RI	RI	RI	RI	VN	VN	VN	VN	RL	RL
Node code	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
State	RL	RL	VN	VN	VN	VN	VN	VN	VN	VN	

By using the VCQ and SLT, the neighborhood searching problem on quadtree is transformed into neighborhood searching on a uniform grid.

For a given node of code M_D (hereinafter abbreviated as M_D) in a n layered quadtree, procedure of its neighborhood searching includes two steps:

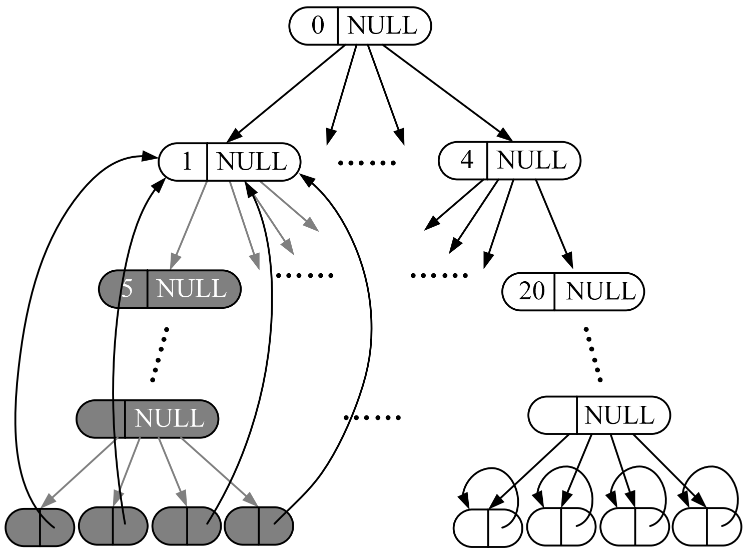
Step 1-Uniform grid mapping: If the layer of M_D is n, directly map it to a single cell in uniform grid. Otherwise, if the layer is less than n, map it to a rectangular region consisting of multiple cells by down-traversing to all its virtual nodes at layer n. For example, M_D = 9 in Figure 4 will be mapped to a single cell with code 9, and M_D = 1 will be mapped to a rectangular region containing cells with codes 5, 6, 7 and 8 in the uniform grid, respectively.

Step 2-Neighborhood searching: Perform normal neighborhood searching on uniform grid, finding all the cells neighboring to the mapped cell(s). Then judge the states of them one by one via SLT: If the state is RL, then add this node to the neighbor list. If the state of node is VN, then up-traverse to all its ancestor nodes until the ancestor node is found which state is RI. If the ancestor node is not in the neighbor list, then add it to the neighbor list.

However, the Morton algorithm has two main shortcomings: First, for each node in neighborhood judgment, its code needed to be obtained first, and then be used as an index to query its state in SLT. This mechanism requires maintenance of both the VCQ and SLT, and also requires frequent interactions between the VCQ and SLT. The operation is complicated. Secondly, it requires traversing up the parental node when processing nodes of a VN-state. When the quadtree depth is great, this traversal is inefficient.

To solve the above problem, we improved the Morton algorithm by adding a pointer field to each node structure. Figure 5 depicts an example of quadtree structure with pointer fields.

Figure 5. Quadtree structure with pointer field (grey part is the virtual structure).



The pointer is a quadtree node typical of a three-valued case:

Case 1: If the current node state is RI, then the pointer is set to NULL;

Case 2: If the current node state is RL, then the pointer points to itself;

Case 3: If the current node state is VN and at the deepest layer, then the pointer points to its nearest ancestor node with state RI. If the node is not at the deepest layer, then the pointer is set to NULL.

Value assignment of the pointer can be completed in the quadtree construction. By using the pointer value from Case 3, we can obtain the ancestor node with state RI from the current node in one step, evading multi-step (at least one step) up-traversals; thus, greatly improving the search speed. Also, a unified node is needed in Cases 2 and 3 for neighborhood discrimination. At the same time, the 3-valued pointer can completely replace the role of the SLT without increasing storage space, thereby evading the need for additional maintenance work.

2.1.3. Region Feature Calculation

In our method, region merging is performed on adjacent and similar regions. Adjacency can be judged by using the spatial indexing mechanism discussed in the above section. Similarity, it is calculated based on region features. In this section, we discuss the calculation of region features.

We use both spectral and texture features to construct feature vector for each region. We use pixel mean value and entropy of region to represent spectral feature, and use directionality and line-likeness of the Tamura [24] feature as texture features. So, there are four features in feature vector for single-band image. For multi-band image, we calculate above features for each band, so the feature vector contents $4 \cdot nBands$ features, where $nBands$ is the total number of bands.

In the spectral aspect, the pixel mean value and entropy are used for representing regional grey scale and spectral homogeneity, respectively. The entropy is defined as:

$$Ent = -\sum_{i=0}^{G} p_i \log p_i \tag{5}$$

where G is the maximum possible grey level (for 8-bits image data, G = 255), and p_i is the pixel frequency with grey value i.

In the spectral aspect, considering that edge direction and intensity are the most important textural features, we use directionality and line-likeness as features to characterize the region texture. The directionality F_{dir} is defined as:

$$F_{dir} = \sum_{p}^{n_p} \sum_{\phi \in w_p} (\phi - \phi_p)^2 H_D(\phi) \tag{6}$$

where $H_D(\varphi)$ is the angle histogram calculated from edge map obtained by edge detection. p is the peak in the histogram. n_p is number of peaks. For a peak p, w_p is range of the p th peak between valleys [24]. φ_p is the p th peak position in the histogram.

The line-likeness F_{lin} is defined as:

$$F_{lin} = \frac{\sum_{i}^{n} \sum_{j}^{n} P_{Dd}(i,j) \cos[(i-j)2\pi/n]}{\sum_{i}^{n} \sum_{j}^{n} P_{Dd}(i,j)} \tag{7}$$

where P_{Dd} is the $n \times n$ local directional co-occurrence matrix.

2.2. Region Merging Based on RAG

Region merging is a bottom-up process in which large regions are obtained by combining small regions under certain criteria. Regions involved are required to be spatially adjacent and similar in features. The difficulty of region merging lies in the adjacency judgment and similarity measurement. Currently, most existing methods treat each pixel as the initial region with which to discuss the merging issue. However, the merging process requires a large amount of computation at the pixel level, and requires huge storage space. Therefore, we take segments generated by quadtree segmentation as initial regions, and obtain the final segmentation result by merging these regions. The advantage of this method lies in evading pixel level region merging and having lower computation and storage size. We characterize the relationship of the area that can be combined by constructing RAG.

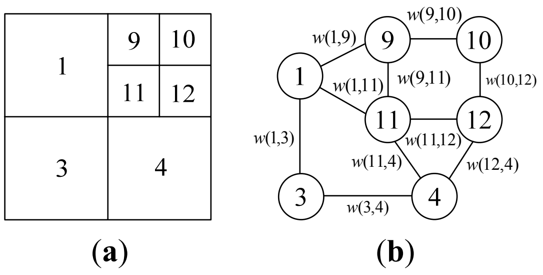
2.2.1. RAG Criterion

RAG is defined as a weighted undirected graph, in which $G = (V, E)$, where V is the vertex set, representing regions generated by the merging process. E is the edge set, with the element $w(i,j) = Merge(v_i,v_j)$ defined as the merge cost function between vertex v_i and v_j. Full Lambda-Schedule [17] is a simple algorithm, which can express well the relationship between regions, so we reference their ideas and define the merge cost function between two regions as:

$$Merge(v_i, v_j) = \frac{O_i \cdot O_j}{(O_i + O_j) \cdot \ell(v_i, v_j)} \left\| u_i - u_j \right\|^2 \tag{8}$$

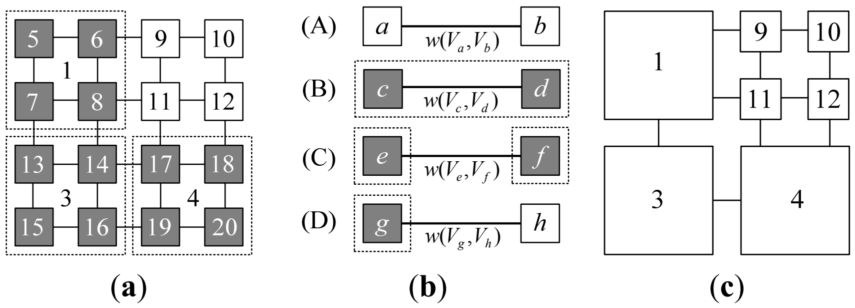
where O_i and O_j are the areas of regions that two vertices corresponds to. u_i and u_j are feature vectors of the regions. $\lambda(v_i, v_j)$ is the common boundary length between the two regions. There is no edge connection between nonadjacent regions. Figure 6 shows an example of the two-layered quadtree segmentation and its corresponding RAG.

Figure 6. A two-layered quadtree segmentation and its corresponding Region Adjacency Graph (RAG). (**a**) Two-layered quadtree segmentation and (**b**) its RAG.



In [17], the merging process was started from pixel level. The initial vertices in the graph therefore correspond to the original pixels, and the initial edges were created by the traversal of neighbouring pixels from horizontal and vertical directions. However, quadtree segmentation generates non-uniform regions. To solve this problem, we constructed initial RAG using the virtual grid. In Figure 7, the dashed boxes represent the ancestor nodes of the virtual leaf nodes inside them. Their state is RI.

Figure 7. Initial RAG construction using the virtual grid (the virtual structure is represented in grey). (**a**) Virtual grid and virtual edge; (**b**) Four cases of virtual edge; (**c**) Completed RAG.



Procedure for RAG construction has two main steps:

Step 1. Virtual edge construction: Virtual edges are constructed on the uniform grid mapped from the VCQ in horizontal and vertical directions. For a quadtree with depth n, $2^{2n+1} - 2^{n+1}$ virtual edges need to be constructed. There are 24 virtual edges in the example depicted in Figure 7a;

Step 2. Graph reformation: Traverse all virtual edges, and each edge is processed by its case. Suppose the current virtual edge is *e*, if:

Case A: The states of two nodes connected by e are all RL. See case (A) in Figure 7b. Then add *e* to edge set *E*, and add the 2 nodes to vertex set *V*. In the example in Figure 7a, the satisfied edges are *w*(9,10), *w*(9,11), *w*(9,12), and *w*(11,12);

Case B: The states of two nodes connected by e are all VN, and they are descendants of the same ancestor. See case (B) in Figure 7b. It does not need to be processed. In the example in Figure 7a, the satisfied edges are *w*(5,6), *w*(5,7), *w*(6,8), *w*(7,8), *w*(13,14), *w*(13,15), *w*(14,16), *w*(15,16), *w*(17,18), *w*(17,19), *w*(18,20) and *w*(19,20);

Case C: The states of two nodes connected by *e* are VN, but they are descendants of different ancestors. See case (C) in Figure 7b. Reform it to e' by connecting their respective ancestor nodes. Check the existence of e'. If it does not exist, then add e' to edge set *E*, and the two ancestor nodes to vertex set *V*. In the example in Figure 7a, the edges satisfied include: *w*(1,3) reformed from *w*(7,13) and *w*(8,14) and *w*(3,4) reformed from *w*(14,17) and *w*(16,19).

Case D: The states of two nodes connected by e are VN and RL. See case (D) in Figure 7b. Reform it to e', which is a connection between the ancestor of VN node and the original RL node. Then add e' to edge set *E*, and the two nodes after reformation to vertex set *V*. In the example shown in Figure 7a, the satisfied edges include: *w*(1,9) reformed from *w*(6,9), *w*(1,11) reformed from *w*(8,11), *w*(11,4) reformed from *w*(11,17), and *w*(12,4) reformed from *w*(12,18).

After above processing, the initial RAG is created, as shown in Figure 7c.

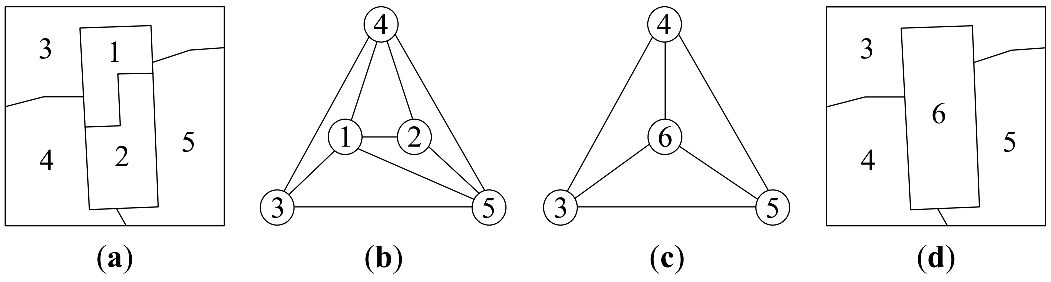
2.2.2. Region Merging

After the initial construction, region merging is performed on the RAG. The topological structure of the RAG will be changed in merging process. The merging process always occurred between the most similar and adjacent regions. After merging of two regions, a new ID is given to the new region and the feature vector is updated. Feature vector of the new merged region is calculated as the weighted average of the area by the pre-merging regions:

$$u_k = \frac{O_i \cdot u_i + O_j \cdot u_j}{O_i + O_j} \tag{9}$$

where u_k is the feature vector of new merged region. Other variables have the same meaning as they do in Equation (8). Suppose the region merging is processed as shown in Figure 8a. There are five regions numbered as 1–5. Regions 1 and 2 are homogeneous regions that can be merged. Figure 8b shows the corresponding RAG. Regions 1 and 2 are merged to form region 6. The feature vector for region 6 can be calculated using Equation (9). The edges can then be reconstructed: remove edge *w*(1,2) connect vertices 1 and 2; use edges *w*(6,3), *w*(6,4) and *w*(6,5) as new edges update *w*(1,3), *w*(1,4) and *w*(1,5), respectively as shown in Figure 8b; All the edges connect to vertex 2, *w*(2,4) equivalent to *w*(1,4), and *w*(2,5) equivalent to *w*(1,5), so remove these two edges. The completed RAG and region map are presented in Figure 8b,c.

Figure 8. Region merging schematic diagram. (**a**) Region map before merging; (**b**) RAG corresponding to (a); (**c**) RAG after merging; (**d**) Region map after merging.



3. Algorithm Experiment and Analysis

In order to examine the method proposed by this paper, we perform our method on GeoEye-1 (0.41 m resolution, Experiment A) and IKONOS (1.0 m resolution, Experiment B) images, which are two common high-resolution satellite remote sensing images. Image sizes are 961 × 747 and 945 × 734 pixels respectively. Experiment is implemented by Visual C++ 9.0, and performed on Windows 7 operating system. The hardware configuration is 2G memory and 2.10 GHz CPU.

3.1. Step 1: Initial Segmentation Based on Quadtree

Images used in our experiment are all true color fusion images. So, standard deviation averages of the red, green and blue bands, T_s, are taken as the quadtree segmentation threshold. Figures 9 and 10 show the original image and quadtree segmentation results using three different T_s values Experiments A and B, respectively. It can be seen from the results listed in Figure 9c–e and Figure 10c–e that with an increasing of the threshold, the size of segments also increased, while the detail of segmentation scale is reduced.

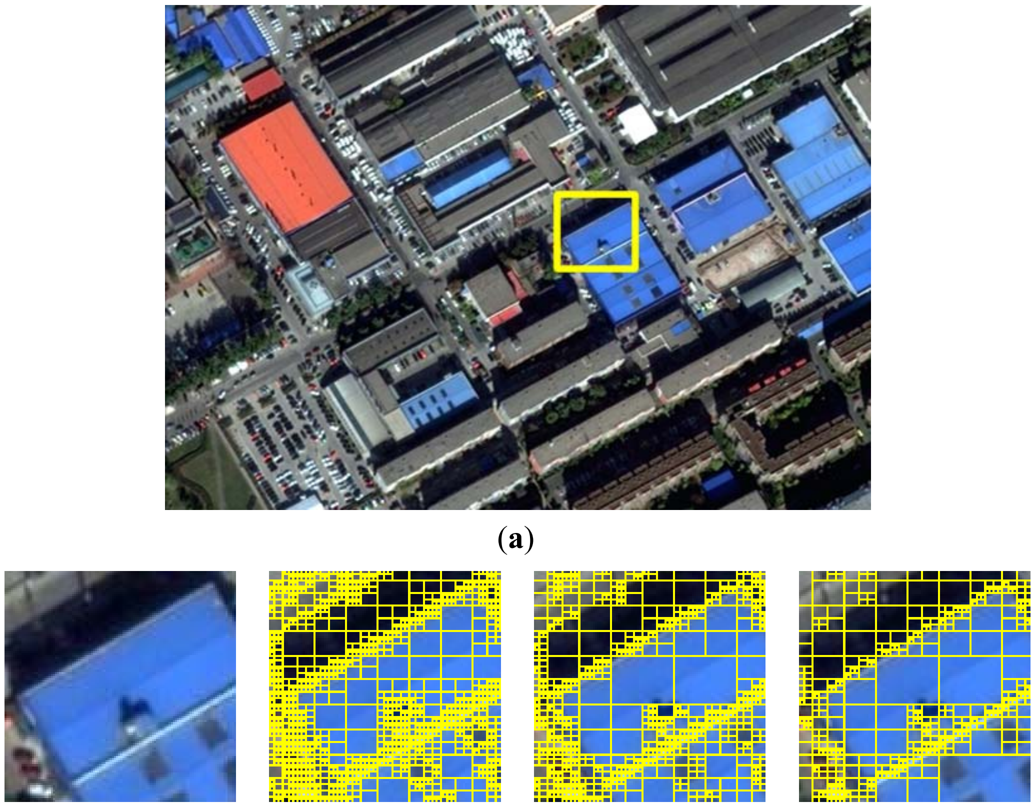
Comparison of the above quadtree initial segmentation is given in the following Table 3:

Table 3. Comparison of quadtree initial segmentation with different T_s.

Experiment	T_s	Time of Quadtree Segmentation (s)	Time of Spatial Indexing Creation (s)	Total Time (s)	Quadtree Depth	Segment Count
Exp. A	3	2.3872	0.5140	2.9012	9	147586
	10	1.2793	0.3252	1.6045	8	98185
	30	0.5634	0.1173	0.6807	8	18147
Exp. B	4	2.0366	0.6102	2.6468	9	186034
	12	1.4622	0.5615	2.0237	9	97336
	25	0.4621	0.2614	0.7235	8	33859

In the above experiment, we proposed to use the standard deviation criterion fast calculation as a method to determine the initial quadtree segmentation. In order to test the efficiency of the proposed calculation, we compared our calculation with the traditional standard deviation calculation method, using the same image and parameter settings as depicted in Figures 9 and 10. In Figure 9, time costs of the traditional standard deviation calculation are about: 10.7 s, 6.6 s and 3.4 s for 3 scales. In Figure 10, time costs of the traditional standard deviation calculation are: 12.5 s, 5.2 s and 4.1 s for three scales. It can be seen that the proposed fast calculation method improved efficiency 4–6 times.

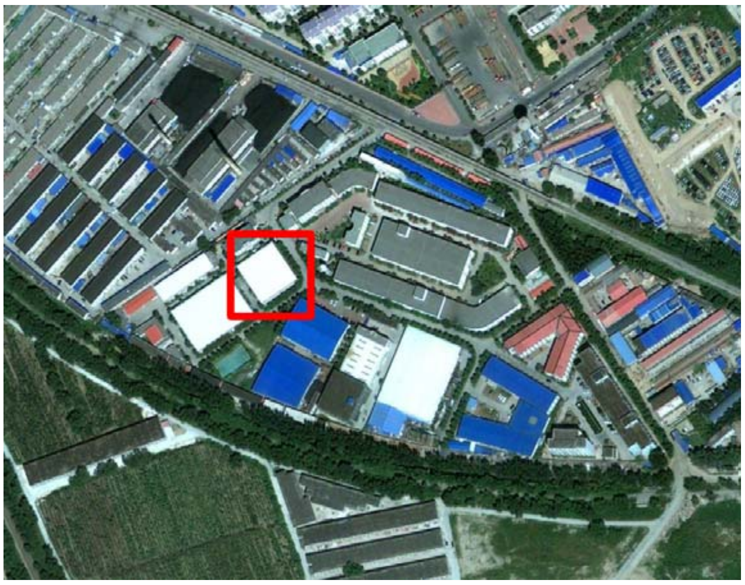
Figure 9. Quadtree initial segmentation results for GeoEye-1 image in Experiment A. (**a**) Original image; (**b**) Local zoomed area of original image; (**c**–**e**) Quadtree initial segmentation and local area with $T_s = 3$, $T_s = 10$ and $T_s = 30$.



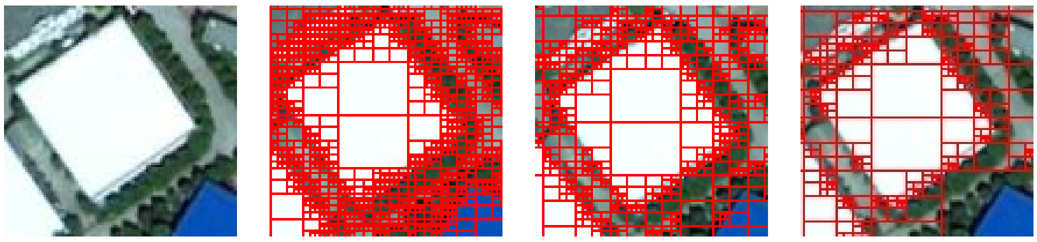
(**b**) (**c**) (**d**) (**e**)

Initial segmentation completed with a too small T_s value will lead to trivial and broken segments, increasing the number of useless merge operations. At the same time, using a large threshold value will lead to an ineffective separation of objects and the generation of lots of jagged edges in the final results. Therefore, we selected 10 classes of ROI of typical ground target: water, grass, asphalt road, cement road, shrubs, close-planted forest, bare land and roof, and then analyzed the consistency for each class. The results of the analysis show that water has the strongest consistency. The range of its standard deviation average of red, green and blue bands is 7–10 in the true color image. Therefore, we used $T_s = 10$ for the quadtree segmentation threshold value, as shown in Figure 9d and Figure 10d.

Figure 10. Quadtree initial segmentation results for IKONOS image in Experiment B. (**a**) Original image; (**b**) Local zoomed area of original image; (**c**–**e**) Quadtree initial segmentation and local area with $T_s = 4$, $T_s = 12$ and $T_s = 25$.



(**a**)



(**b**) (**c**) (**d**) (**e**)

3.2. Step 2: Region Merging Based on RAG

Region merging is performed after the quadtree initial segmentation. The merging scale can be controlled by using different merging thresholds T_m. Figures 11 shows the region merging results of different merging thresholds after the quadtree initial segmentation with $T_s = 10$. With the increase of the merging threshold, the scale of region merging also increases.

In Experiment A (Figure 11a,b), the number of regions obtained with $T_m = 2$ and $T_m = 3.5$ was 1,158 and 722, and time costs were 2.6166 and 1.516 s, respectively. In Experiment B (Figure 11c,d), number of regions obtained with $T_m = 2.5$ and $T_m = 3.5$ was 1,543 and 1,109, and time costs were 3.201 and 2.457 s, respectively.

Two merging results of building roofs in Experiment A were used as typical examples, and their quadtree nodes and merging results are shown in Figure 12.

Figure 11. Region merging results with different threshold values. (**a**,**b**) Region merging result with T_m = 2 and T_m = 3.5 respectively for GeoEye-1 image in Experiment A; (**c**,**d**) Region merging result with T_m = 2.5 and T_m = 3.5 respectively for IKONOS image in Experiment B.

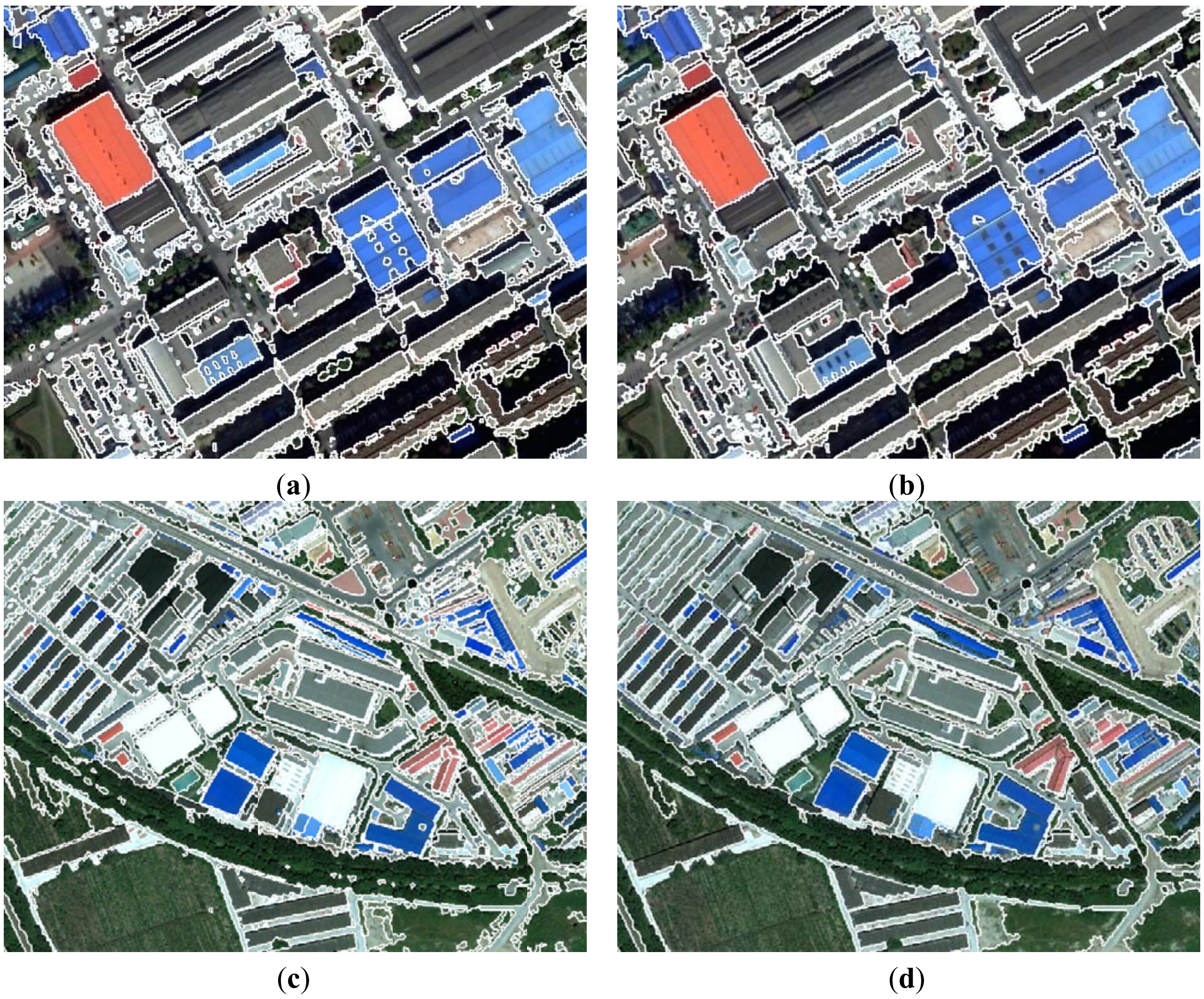
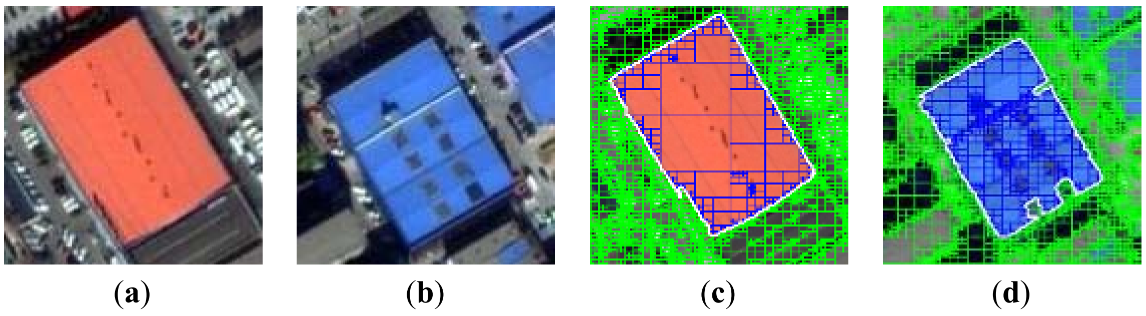


Figure 12. Quadtree nodes and region merging of two building roofs in Experiment A. (**a**,**b**) Original images of building roofs; (**c**,**d**) Quadtree nodes and corresponding merging results under T_m = 3.5.



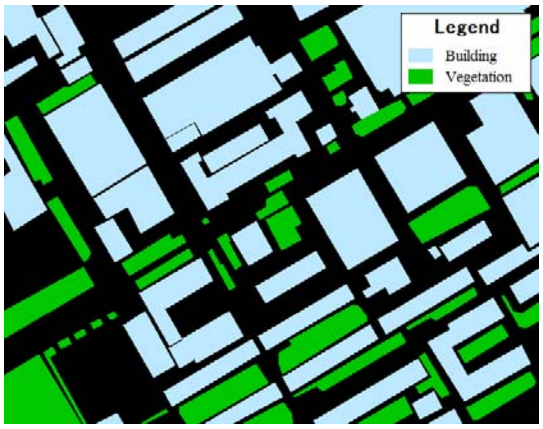
3.3. Method Comparison and Discussion

3.3.1. Method Comparison

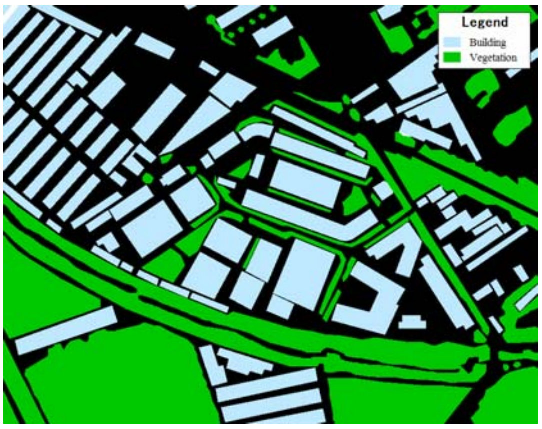
To validate the method, our method was compared with the MR and MS algorithms for precision and speed analysis. The MR segmentation algorithm was performed with the commercial image analysis software, eCognition, using software recommended settings for urban image segmentation. Figure 13b shows the segmentation result. The MS algorithm was performed in image segmentation software EDISON developed by Bogdan Georgescu [7]. The MR segmentation parameters are: scale factor $\lambda_{Scale} = 80$, color $\lambda_{Scale} = 0.5$. The MS segmentation parameters are: shape bandwidth $\lambda_{Scale} = 12$, color bandwidth $\lambda_{Scale} = 8$. Parameters of our algorithm: $T_s = 10$, $T_m = 4$.

Number of regions obtained and time consumption of three segmentation algorithms are shown in Table 4.

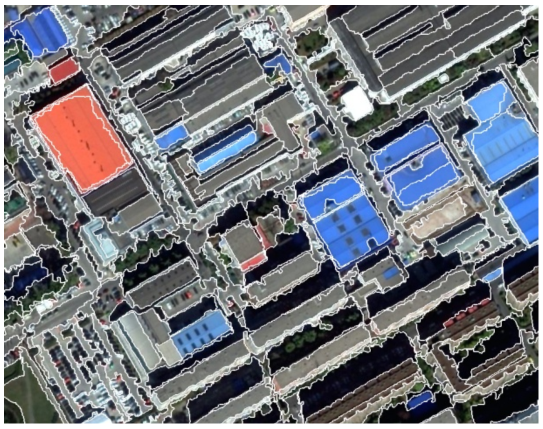
Figure 13. Segmentation results of MR, MS method and our method. (**a**,**b**) Ground truth map for Experiments A and B; (**c**,**d**) Segmentation results of MR algorithm for Experiments A and B; (**e**,**f**) Segmentation results of MS algorithm for Experiments A and B; (**g**,**h**) Segmentation results of our algorithm for Experiments A and B.



(**a**)



(**b**)


(**c**)

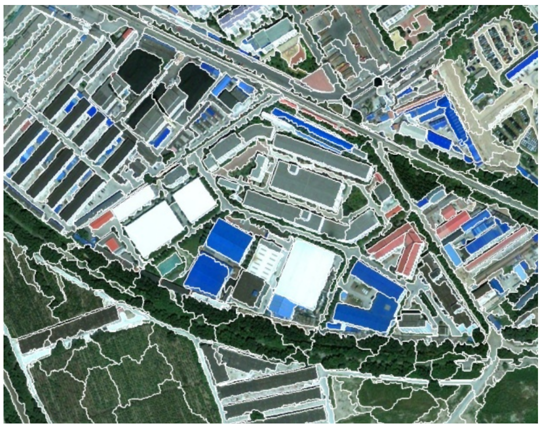
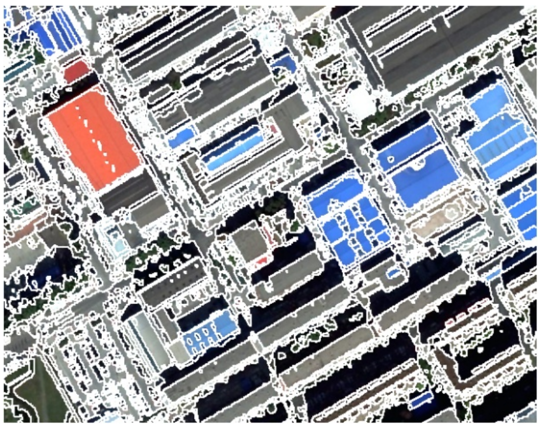
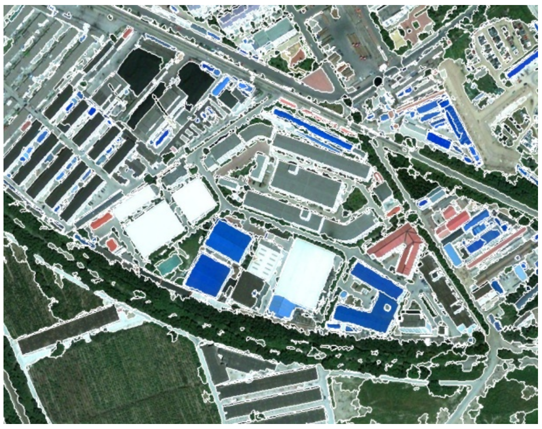

(**d**)

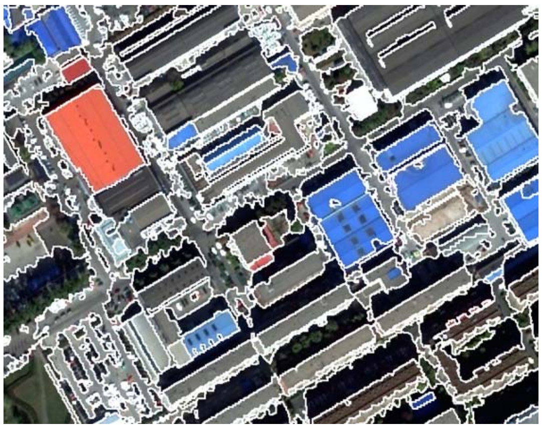
Figure 13. *Cont.*



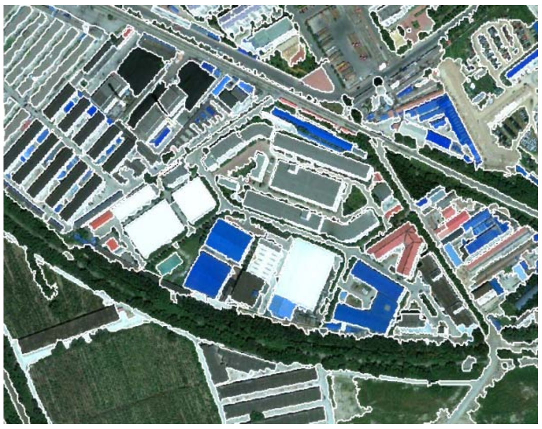
(e)



(f)



(g)



(h)

Table 4. Region number and time consuming comparison of MR, MS, and our method.

Experiment	Method	Region Count	Time (s)
Exp. A	MR method	1487	about 8
	MS method	1853	29.77
	Our method	657	3.7530
Exp. B	MR method	1659	about 10
	MS method	1968	22.13
	Our method	711	4.57

To test the precision of the algorithm proposed in this paper, we used a vector map produced by a professional image interpreter as ground truth data, containing building and vegetation as two classes of ground targets. The ground truth map includes 47 building objects and 38 vegetation objects for Experiment A, and 98 building objects and 53 vegetation objects for Experiment B. Each region generated by segmentation was marked as building or vegetation category through human-computer interaction for the three methods. The segmentation results with the category mark were used for precision analysis and comparison.

We defined segmentation precision from two aspects: segmentation accuracy and object integrity. Segmentation accuracy is defined as: for a type of ground target, the area of the correct region (regions generated by segmentation process) proportional to the total area of the ground targets:

$$Accuracy = \frac{Area(Rgn_{gt} \cap Rgn_{seg})}{Area(Rgn_{seg})} \times 100\% \tag{10}$$

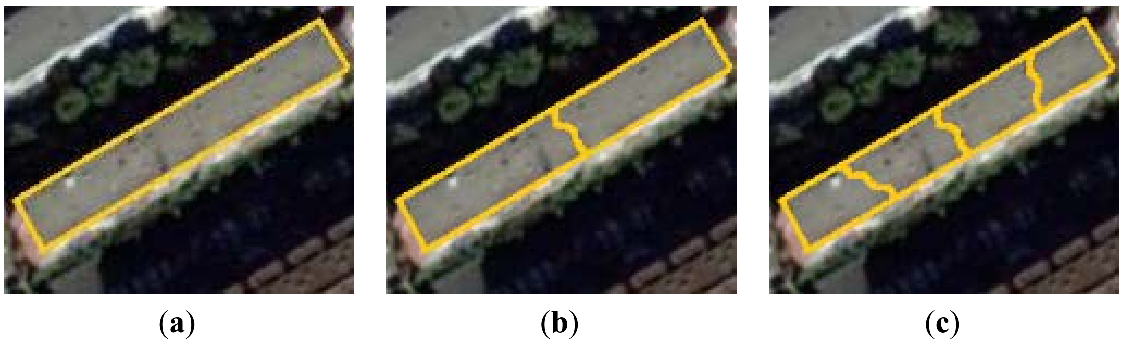
where Rgn_{gt} is ground truth region set. Rgn_{seg} is the region set of the segmentation result within the same category. $Area(\cdot)$ is area (in pixel) calculation function.

Whether regions obtained by segmentation can express ground true objects is a key question for following object-oriented information extraction. We defined object integrity for a type of ground targets as: ratio of ground truth object count and correct region (regions generated by segmentation process) count:

$$Completeness = \frac{Count(Rgn_{gt})}{Count(Rgn_{gt} \cap Rgn_{seg})} \times 100\% \tag{11}$$

where $Count(\cdot)$ region count statistic function. Consider the building in Figure 14 as an example of object integrity calculation for a single object, $Count(Rgn_{gt}) = 1$.

Figure 14. Object integrity calculation illustration (**a**) $Count(Rgn_{gt} \cap Rgn_{seg} = 1)$, $Completeness = 1$; (**b**) $Count(Rgn_{gt} \cap Rgn_{seg} = 2$, $Completeness = 2.5$; (**c**) $Count(Rgn_{gt} \cap Rgn_{seg} = 4)$, $Completeness = 0.25$.



The precision analysis results of two experiments are shown in Table 5:

Table 5. Comparison of segmentation precision of MR, MS, and our algorithm.

Experiment	Method	Segmentation Accuracy	Object Integrity
Exp. A	MR method	90.50%	31.79%
	MS method	90.77%	29.35%
	Our method	92.45%	51.63%
Exp. B	MR method	91.36%	27.92%
	MS method	89.32%	25.11%
	Our method	95.74%	48.44%

It can be seen from the precision comparison that the three algorithms all have high segmentation accuracy levels. However, in the case of object integrity, our algorithm has more obvious advantages. It has stronger object-oriented features.

3.3.2. Discussion

Both MR and MS segmentation are bottom-up algorithms, started from pixels. Therefore, it is unavoidable to produce inadequate region merging phenomenon as can be seen from Figure 13c–f. The MR segmentation algorithm uses a single discriminate function as a regional merging criterion although the size of the ground object and spectral features have great differences. The use of a single discriminate function cannot implement segmentation for different scale objects, causing over merging for some objects and subsequent merging for other objects. It will bring great difficulties in subsequent region operations. The MS algorithm adds pixel coordinates to the feature vector, expressing spatial information to participate in segmentation operation. However, the spatial information is limited for segmentation. At the same time, the MS algorithm has high computational complexity, $O(n^2)$. It is difficult to effectively apply to large remote sensing image segmentation. The first step of our algorithm is an up-bottom procedure that evades subsequent pixel level region merging. To some extent, this mechanism will inhibit the over merging phenomenon. It is also more conducive to improving the efficiency of the region merging.

4. Conclusions

This paper presents a high-resolution remote sensing image segmentation algorithm based on improved quadtree structure and RAG technique. Our algorithm includes two steps: up-bottom initial segmentation and bottom-up region merging. In the first step, we proposed a fast method for standard deviation calculation method. By using this improved the segmentation efficiency of quadtree is improved for 4–6 times. This improvement has significance for huge remote sensing image processing. After the creation of quadtree structure, a spatial index mechanism based on improved Morton coding was added to quadtree structure, providing a fast neighborhood data access mechanism. In the second step, we use a region merging algorithm based on RAG to obtain the final segmentation result. Our algorithm is tested on a true color fusion GeoEye-1 image. We use both segmentation accuracy and object integrity as indicators for segmentation result evaluation. The result shows that the accuracy of our method is higher than 90%, and the object integrity is higher than 50%. Our segmentation method will provide subsequent process such as target objects classification and other applications with more accurate data.

Acknowledgments

This work is partially funded by the National High Technology Research and Development Program of China (863 program, Grant Number 2012AA121303) and the National Natural Science Foundation of China (Grant Number 71273154).

Conflict of Interest

The authors declare no conflict of interest.

References

1. Shapiro, L.G.; Stockman, G.C. *Computer Vision*; Prentice-Hall: Upper Saddle River, NJ, USA, 2011; pp. 279–325.
2. Willhauck, G.; Schneider, T.; de Kok, R.; Ammer, U. Comparison of Object Oriented Classification Techniques and Standard Image Analysis for the Use of Change Detection between SPOT Multispectral Satellite Images and Aerial Photos. In Proceedings of XIXth ISPRS Congress—Technical Commission III: Systems for Data Processing, Analysis and Representation, Amsterdam, The Netherlands, 16–23 July 2000; pp. 35–42.
3. Geneletti, D.; Gorte, B. A method for object-oriented land cover classification combining Landsat TM data and aerial photographs. *Int. J. Remote Sens.* **2003**, *24*, 1273–1286.
4. Tan, Y.M.; Huai, J.Z.; Tang, Z.S. Edge-guided segmentation method for multiscale and high resolution remote sensing image. *J. Infrared Millim. Waves* **2010**, *29*, 312–315.
5. Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE Trans. Pat. Anal. Mach. Intell.* **1995**, *17*, 790–799.
6. Comaniciu, D.; Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pat. Anal. Mach. Intell.* **2002**, *24*, 603–619.
7. Georgescu, B.; Shimshoni, I.; Meer, P. Mean Shift based Clustering in High Dimensions: A Texture Classification Example. In Proceedings of the Ninth IEEE International Conference on Computer Vision, Nice, France, 13–16 October 2003; pp. 456–463.
8. Xiao, C.; Liu, M. Efficient mean-shift clustering using gaussian KD-tree. *Comput. Graph. Forum* **2010**, *29*, 2065–2073.
9. Wang, L.C.; Zheng, L.; Lin, R.; Chen, T.; Mei, T.C. Fast segmentation algorithm of high resolution remote sensing image based on multiscale mean shift. *Spectrosc. Spectr. Anal.* **2011**, *31*, 177.
10. Roerdink, J.B.; Meijster, A. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundam. Informa.* **2000**, *41*, 187–228.
11. Bleau, A.; Leon, L.J. Watershed-based segmentation and region merging. *Comput. Vis. Image Underst.* **2000**, *77*, 317–370.
12. Pun, C.M.; An, N.Y.; Chen, C.L.P. Region-based image segmentation by watershed partition and DCT energy compaction. *Int. J. Comput. Intell. Syst.* **2012**, *5*, 53–64.
13. Baatz, M.; Schäpe, A. Multiresolution Segmentation: An Optimization Approach for High Quality Multi-scale Image Segmentation. In Proceedings of the Angewandte Geographische Information Sverarbeitung XII, Heidelberg, Germany, 5–7 July 2000; pp. 12–23.
14. Definients Image. *eCognition User's Guide 4*; Definients Image: Bernhard, Germany, 2004.
15. Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pat. Anal. Mach. Intell.* **2000**, *22*, 888–905.
16. Feature Extraction Module Version 4.6. In *ENVI Feature Extraction Module User's Guide*, December, 2008th ed.; ITT Corporation: Boulder, Colorado, USA, 2008.
17. Robinson, D.J.; Redding, N.J.; Crisp, D.J. *Implementation of a Fast Algorithm for Segmenting SAR Imagery*; DSTO Electronics and Surveillance Research Laboratory: Edinburgh, SA, Australia, 2002.

18. Finkel, R.; Bentley, J.L. Quad trees: A data structure for retrieval on composite keys. *Acta Inform.* **1974**, *4*, 1–9.
19. Choi, H.; Baraniuk, R.G. Multiscale image segmentation using wavelet-domain hidden Markov models. *IEEE Trans. Image Process.* **2001**, *10*, 1309–1321.
20. Pavlidis, T.; Liowm Y.T. Integrating region growing and edge detection. *IEEE Trans. Pat. Anal. Mach. Intell.* **1990**, *12*, 225–233.
21. Kelkar, D.; Gupta, S. Improved Quadtree Method for Split Merge Image Segmentation. In Proceedings of the 1st International Conference on Emerging Trends in Engineering and Technology, (ICETET'08), Nagpur, India, 16–18 July 2008; pp. 44–47.
22. Viola, P.; Jones, M.J. Robust real-time face detection. *Int. J. Comput. Vis.* **2004**, *57*, 137–154.
23. Abel, D.J.; Smith, J. A data structure and algorithm based on a linear key for a rectangle retrieval problem. *Comput. Vis. Graph. Image Process.* **1983**, *24*, 1–13.
24. Tamura, H.; Mori, S.; Yamawaki, T. Texture features corresponding to visual perception. *IEEE Trans. Syst. Man Cybern.* **1978**, *8*, 202–215.

© 2013 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).