


Computationally Efficient Mean-Shift Parallel Segmentation Algorithm for High-Resolution Remote Sensing Images

Tianjun Wu^{1,2,5}  · Liegang Xia³ · Jiancheng Luo⁴ · Xiaocheng Zhou² · Xiaodong Hu⁴ · Jianghong Ma¹ · Xueli Song¹

Received: 5 March 2017 / Accepted: 21 August 2018 / Published online: 6 September 2018
© Indian Society of Remote Sensing 2018

Abstract

In high-resolution remote sensing image processing, segmentation is a crucial step that extracts information within the object-based image analysis framework. Because of its robustness, mean-shift segmentation algorithms are widely used in the field of image segmentation. However, the traditional implementation of these methods cannot process large volumes of images rapidly under limited computing resources. Currently, parallel computing models are generally employed for segmentation tasks with massive remote sensing images. This paper presents a parallel implementation of the mean-shift segmentation algorithm based on an analysis of the principle and characteristics of this technique. To avoid the inconsistency on the boundaries of adjacent data chunks, we propose a novel buffer-zone-based data-partitioning strategy. Employing the proposed data-partitioning strategy, two intensively computation steps are performed in parallel on different data chunks. The experimental results show that the proposed algorithm effectively improves the computing efficiency of image segmentation in a parallel computing environment. Furthermore, they demonstrate the practicality of massive image segmentation when computer resources are limited.

Keywords High-resolution remote sensing images · Image segmentation · Mean-shift · Parallel computation · Data-partitioning

Introduction

With the development of high-resolution remote sensing technology, massive remote sensing images are constantly appearing, which are mainly reflected in the following aspects. First, the continuous production of massive data. Remote sensing data can obtain ground surface phenomena in real time by space borne or no-load sensors. They can be continuously transmitted to ground stations as long as storage space and transmission bandwidth are allowed. For example, the Chinese domestic satellites, ZY-3 with 2.1 m resolution and 5 days return cycle, generates an average of more than 0.5 TB of data per day. Second, the rapid growth of the amount of single view image. Because of the enhancement of the sensor technology, the spatial resolution of the data is improved continuously. While the quality of the remote sensing image is improved, the data volume of a single scene is greatly increased. For instance, the original ZY-3 multi-spectral image can exceed the 500 MB, and its fusion image can even exceed 5 GB.

✉ Tianjun Wu
tjwu@chd.edu.cn

¹ Department of Mathematics and Information Science, College of Science, Chang'an University, Xi'an 710064, People's Republic of China

² Key Laboratory of Spatial Data Mining and Information Sharing of Ministry of Education, Fuzhou University, Fuzhou 350002, People's Republic of China

³ College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, People's Republic of China

⁴ State Key Laboratory of Remote Sensing Sciences, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100101, People's Republic of China

⁵ State Key Laboratory of Geo-Information Engineering, Xi'an 710054, People's Republic of China

Third, the enhancement of the quantitative level of data. The development of the sensor also improves the spectral resolution of the image. The data of the new satellites, such as ZY-3 in China and Landsat 8 in the USA, have reached 10 bits of quantization, which increases the capacity of the distribution data more than one time. These situations determine the eventual generation of massive high-resolution remote sensing data, and thus, how to process them effectively and quickly has become an important problem in practical engineering applications.

Among the processing tasks of high-resolution remote sensing images, segmentation is an important step for object-based image analysis (Dikshit and Behl 2009; Blaschke 2010; Wu et al. 2015; Zoleikani et al. 2017). The segmentation performance directly affects the accuracy and efficiency of object-based information extraction (Zhou and Luo 2009; Kavzoglu et al. 2018). Among various segmentation methods, the watershed algorithm, mean-shift algorithm, and Definiens' multi-resolution image algorithm are widely used in the segmenting high-resolution remote sensing images (Vamsee et al. 2018). In particular, the image segmentation algorithm from Definiens is integrated into eCognition software, and its implementation is computationally efficient but not fully open source at present (Baatz et al. 2004). As a statistical approach, mean-shift segmentation algorithm with open source has been proven to converge efficiently and provide a robust segmentation (Li et al. 2005). This method has been intensively investigated in recent years, and applied widely in the fields of cluster analysis, target tracking, image segmentation, image smoothing, and image edge extraction (Huang et al. 2014; Zalik and Zalik 2009; Mukherjee et al. 2009). However, its traditional implementation cannot process massive images rapidly under limited computing resources (Michel et al. 2014, 2015) as the algorithm needs to transfer all the data to the computer's internal memory for a one-time process. Inevitably, the speed of segmentation substantially decreases, which makes it difficult to rapidly process massive amounts of data. Moreover, the algorithm may behave unpredictably when computing resources are limited and the amount of data increases (Grizonnet et al. 2017; Su and Zhang 2017), while the data volumes of widely used high-resolution remote sensing images are increasing because of the development of sensors and storage ability. Therefore, mean-shift segmentation algorithm needs to be improved to accommodate the processing requirements of massive remote sensing images.

In most of the published literatures, the improvements in mean-shift algorithms have been mainly carried out in the algorithm itself. A variety of research approaches for better segmentation such as the selection of different kernel functions (Wang et al. 2008) and parameter optimization

(Mukherjee et al. 2009) have been investigated deeply. However, few studies have focused on improving the algorithm from the perspective of its computational efficiency. In fact, researchers within the field of image processing often adopt parallel methods to perform computing tasks on large amount of data (Shen et al. 2007; Innocenti et al. 2009; Huang and Guo 2001; Shen et al. 2006). The processing speed can be increased by using parallel computation, which is always implemented through multi-threaded computing on a single computer or collaborative computing on multiple processors connected by an internal local area network. The ability to process large amounts of data can be significantly increased using these schemes. For instance, in an agricultural survey of land-cover types, the generation of the final maps often needs to segment a whole scene of a Systeme Probatoire d'Observation de la Terre 5 (SPOT5) fused image. Generally speaking, the internal memory requirements of the mean-shift segmentations of such a massive image are not realistic for an ordinary computer. Therefore, a major concern today is how to improve the processing performance and then reduce the hardware requirements. Although much effort has been made on these improvements, an efficient and effective segmentation has yet to be developed based on mean-shift algorithms. A parallel computation with appropriate data-partitioning may overcome their limitations on computing resource. Hence, in this paper, we propose a parallel implementation of several computing-intensive steps within the mean-shift segmentation procedure. In this implementation, we mainly focus on the data-partitioning methods in the parallel computation. Crucially, a novel buffer-zone-based data-partitioning strategy is proposed to avoid the inconsistency on the boundaries of adjacent data chunks when parallel computation is employed. The performance analysis demonstrates that the proposed approach can effectively improve the processing efficiency and make the segmentation more rapid when computer resources are limited.

The rest of this paper is structured as follows. “[Method Implementation](#)” section presents the principles of mean-shift parallel segmentation algorithm. In “[Experiments and Results](#)” section, experiments are performed to evaluate the effectiveness of the implementation. Conclusions and future research works are given in final last section.

Method Implementation

Mean-Shift Segmentation

In this paper, mean-shift algorithm is focused owing to its robustness in image segmentation. As a kind of region-based statistical segmentation method, this algorithm

performs a nonparametric density function estimation and automatic clustering using means that are iteratively shifted toward the local maxima of the density functions in the feature space (Comaniciu and Meer 2002). Recently, because of its accuracy and stability, mean-shift algorithm has been increasingly applied to the field of remote sensing image segmentation (Huang and Zhang 2008).

Remote sensing images are typically represented as a spatial range in a joint feature space. The dimensionality of the joint domain is $d = 2 + p$ (two for the spatial domain and p for the spectral domain). That is, for a point $x = (x^s, x^r)$, spatial domain $x^s = (l_x, l_y)^T$ denotes the coordinate of a pixel, and range domain $x^r = (r_1, \dots, r_p)^T$ represents the spectral signals for its channels. The multi-variate kernel is then defined using joint density estimation,

$$K_{h_s, h_r}(x) = \frac{C}{h_s^2 h_r^p} k\left(\left\|\frac{x^s}{h_s}\right\|^2\right) k\left(\left\|\frac{x^r}{h_r}\right\|^2\right), \quad (1)$$

where C is a normalization parameter, $k(\cdot)$ is the kernel profile, h_s and h_r are the kernel bandwidths for the spatial and range sub-domains. After the kernel function and its bandwidths have been determined, image clustering can be achieved through mean-shift filtering. Regions are then merged according to minimum merging parameter M . Finally, the vectorization technique is employed to extract the boundaries of objects through area marking. A more detailed description of the algorithm can be found in (Comaniciu and Meer 2002; Huang and Zhang 2008).

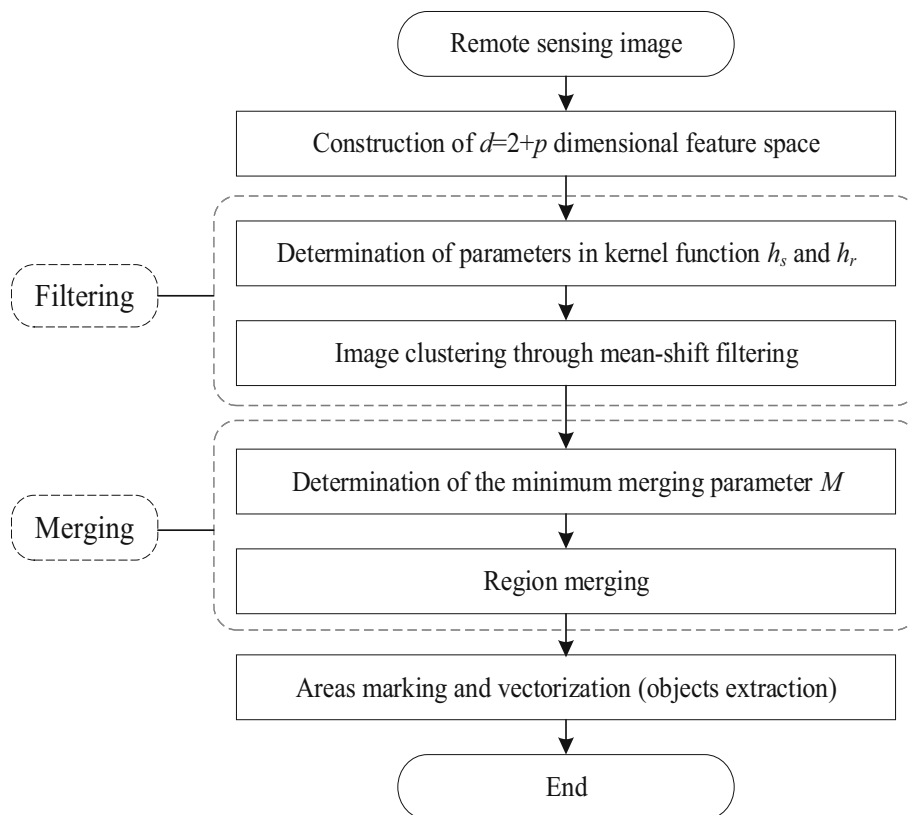
The implementation of mean-shift segmentation is summarized in Fig. 1. As described in this figure, two main steps are carried out in the $(2 + p)$ -dimensional feature space, namely mean-shift filtering and region merging. First, the image is clustered through the mean-shift filtering step, which consists of many iterations. In the iterative computing process, it is necessary to map the values of the image pixel into a high-dimensional feature space. Hence, high memory capacity is required during this one-time process. Therefore, single-core computers cannot handle the processing tasks of a large amount of data. Parallel computing is thus necessary to deal with the massive calculations needed in such cases. In addition, each pixel must be calculated to approximate its “local mean point” in the mean-shift process. Computational complexity substantially increases along with the increase in pixel volume, and this becomes a bottleneck that prevents the speed of segmentation from increasing. Hence, we first consider a parallel implementation of the computationally intensive steps in the mean-shift segmentation algorithm.

Implementation of Mean-Shift Parallel Segmentation

For parallel computation pattern in the field of massive remote sensing image processing, there are three kinds of models according to the characteristics of images, namely pipeline parallelism, functional parallelism, and data parallelism. Pipeline parallelism successively moves different lines in an image into various functional modules in the procedure, while functional parallelism moves an image into the various functional modules simultaneously and performs the calculation at the same time. Data parallelism divides an image into several sub-blocks and performs the same operations on each chunk (Zhou 2003). Note that different processes in pipeline parallelism accomplish different functions and deal with different data. Pipeline parallelism takes the characteristics of functional parallelism and data parallelism both into account (Shen et al. 2012). If properly designed, this kind of parallelism can obtain a higher efficiency. However, it is unsuitable for the current mainstream structure of parallel processing because of its high hardware requirements. Furthermore, because of the correlative relationships between various steps in an algorithm, it is also difficult to neatly make use of functional parallelism for image processing. In fact, owing to the strong regularity and consistency of image contents, the pattern of data parallelism is more appropriate for meeting the tasks of parallel segmentation for high-resolution remote sensing images. It is also more suitable for current mainstream parallel computing systems, such as massively parallel processing (MPP) and cluster systems. Consequently, this study adopts data parallelism to achieve parallel segmentation.

Considering these, we collect distributed computing nodes in the parallel implementation and compute the computationally intensive steps in the process of mean-shift segmentation, i.e., the filtering and merging steps, in parallel to seek better performance. The procedure is set as follows: (1) Set uniform segmentation parameters in each computing node to obtain uniform segmentation results. (2) The master computing node separates the image into n_1 chunks and assigns the chunks to different computing nodes (such as different threads and cluster nodes) to perform the mean-shift filtering separately. (3) The master computing node merges the filtered blocks back together to form a whole clustered image. (4) The main computing node then splits the filtered result into n_2 chunks and assigns them to different computing nodes to merge the regions separately. (5) Finally, the master computing node assembles the merged chunks into areas and marks them as image objects.

Fig. 1 Workflow of the mean-shift segmentation algorithm



This process is illustrated in Fig. 2, where the inputs of the filtering and merging steps are both partitioned into different data chunks by the master node. After performing the data partition, the chunks are distributed to different computing nodes. To speed up processing, different data chunks are simultaneously filtered and merged in a parallel computing environment. This parallel computation can be achieved through multi-threaded computing or multi-point interface (MPI)-based cluster computing. Therefore, using this workflow, the key part of the parallel segmentation algorithm is the data-partitioning strategy, which directly affects the effectiveness and efficiency of the parallelization.

Figure 3 gives a specific example of a commonly employed data-partitioning strategy (strategy I). In this strategy, divided chunks are, respectively, filtered without considering their effects on each other, and thus, each has different global statistical features. After combining the filtering results, there will be a clear “merging line” on the final segmentation map (see Fig. 3c). The upper and lower areas that touch the data partition border do not correspond to each other, which leads to an inconsistent segmentation around the boundary.

To avoid this inconsistency, we propose a novel buffer-zone-based data-partitioning strategy (strategy II) in this paper, as shown in Fig. 4. In this approach, the original data block is divided into more than the conventional

number of chunks. Data chunks in the buffer zone are assigned to assure filtering consistency in the regions close to the boundary. The divided filtering results (clustered regions) on each chunk are fused together on the master node for subsequent region merging. Note that, the height of buffer zone is set as $2h_s$. When the filtered chunks are merged in this step, the pixels in the upper part of buffer area (i.e., 50% pixels of buffer area) are taken from the above chunk, while the other 50% pixels in the lower part are taken from the next chunk. In this way, we can get the overlaps between the chunk and the dotted buffer in the second chunk and there is a smooth combination of the filtering results when the filtering steps are performed on the boundaries of adjacent data chunks. Then, in the following merging processing, several chunks are also extracted from the combined filtering result. In this stage, as no global statistics need to be calculated, a simple data-partitioning can be employed. However, considering the integrity of the edges between the partitions, we further propose a two-step data-partitioning for the merging step. First, as shown in Fig. 4, several regular data chunks of the filtered images are partitioned like a chessboard. Gaps are left between the regular chunks, and they are thus not related to parallel merge computing. After the chunk merge, the data in the irregular chunk area (i.e., gap area) with dotted lines including the intersection area is read for further processing. Merging step is single-handed on this

Fig. 2 Workflow of the mean-shift parallel segmentation algorithm

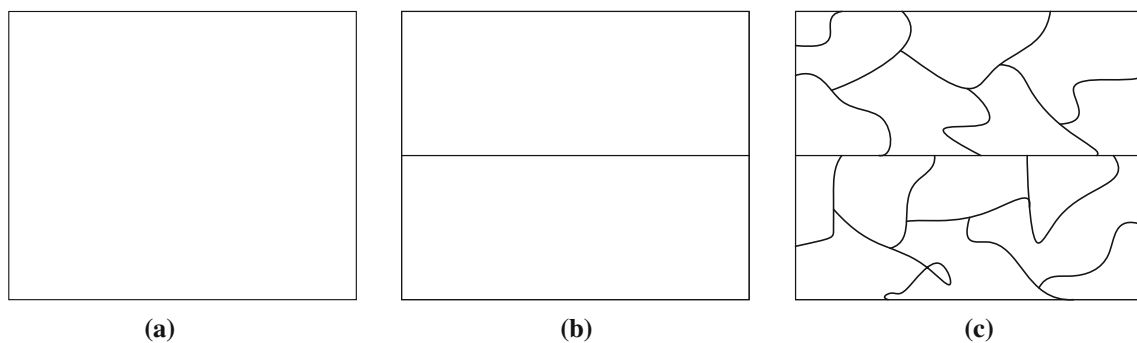
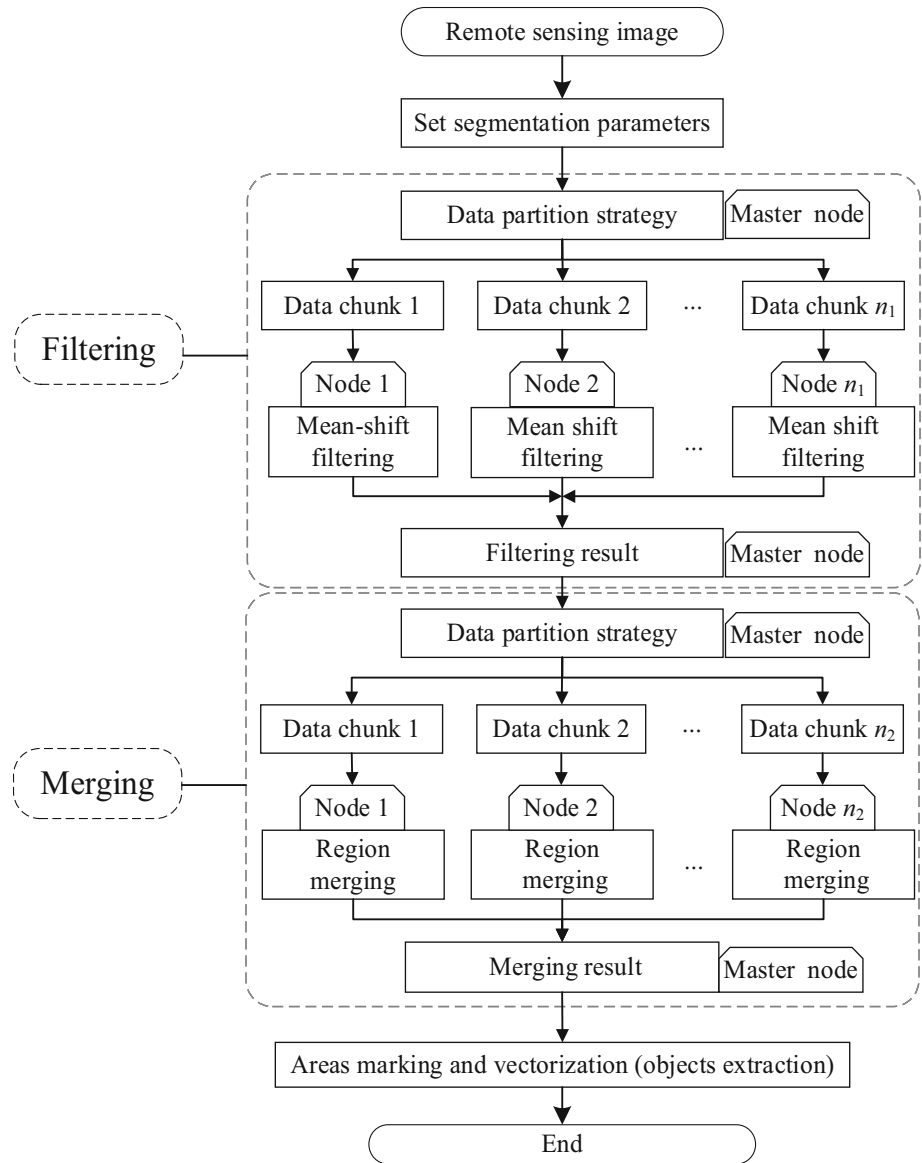


Fig. 3 Simple data-partitioning strategy (strategy I) for parallel segmentation: **a** original image, **b** data chunks, and **c** final segmentation result

chunk adjacent with the above regular chunks. To avoid the merging line, only the results on the regions does not intersect to the dotted borders are written into the final

map. After the merging has been performed on each chunk, the final segmentation map can be obtained by combining the divided results.

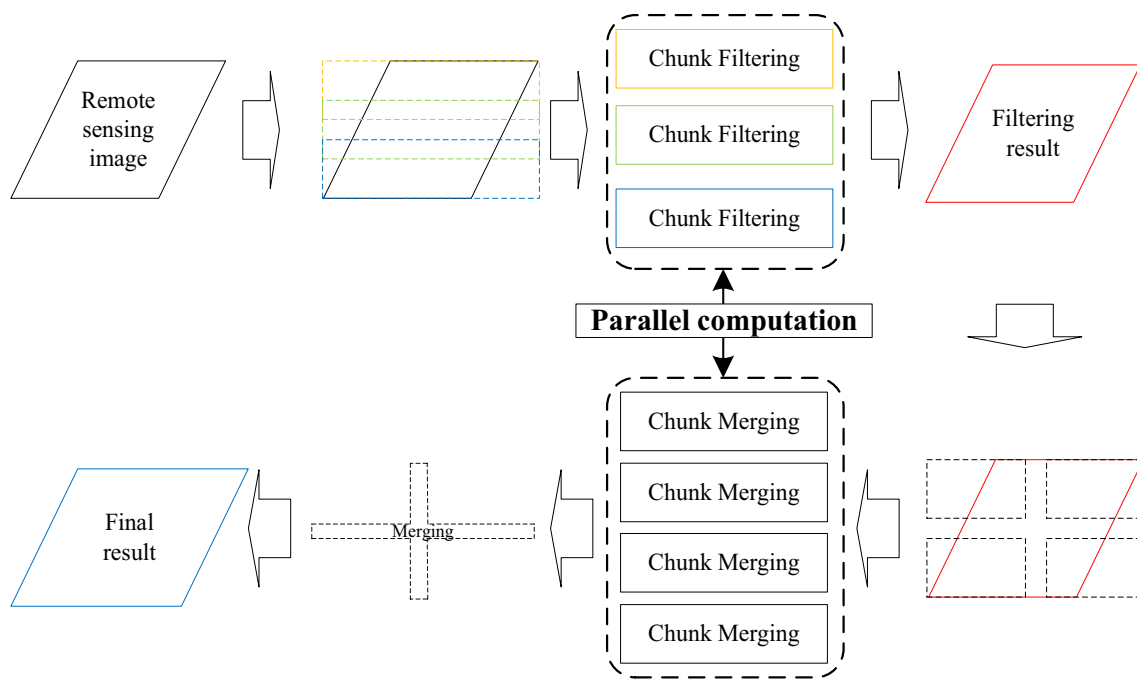


Fig. 4 Data-partitioning strategy (strategy II) in the mean-shift parallel segmentation algorithm

Following questions are further clarified. In the merging process, only the distance between each region and its adjacent regions is computed. If the distance is less than a given threshold, their neighborhood label will be re-labeled after merging. So all regions are merged once, and then the region adjacency table is re-established according to the region label to prepare for the next round of merging. Because the homogeneity of the region will change after the merging, the distance between adjacent regions will also change greatly, which makes the threshold-based merging step stopping after several iterations. From the above regional merging strategy, it can be seen that the regional distance actually plays a role of merging standard, and therefore plays a decisive role in the merging accuracy.

Since the phenomenon of different objects with the similar spectra characteristics is common in multi-spectral remote sensing images, an appropriate distance measure is important for improving the final segmentation accuracy. Thus, for each data chunk in the merging step, the following spectral distance is applied to determine the similarity between adjacent regions, calculated as,

$$d(s_1, s_2) = \sqrt{[Ed_{s_1, s_2}]_{\text{Normalized}}^2 + (1 - r_{s_1, s_2})^2}, \quad (2)$$

where s_1 and s_2 are the mean pixel values of spectral signal vectors in two adjacent regions. The first term in the Eq. (2), $[Ed_{s_1, s_2}]_{\text{Normalized}}$, is the normalized Euclidean distance of two spectral vectors. This term is lower for similar vectors and is higher for dissimilar vectors. Further, r_{s_1, s_2} in the second term is the correlation coefficient that exploits

the overall shape difference between the reflectance curves. Combining these two metrics, the resulting measure in Eq. (2) matches the total difference between two adjacent regions in the aspects of shape and spectrum (Thenkabail et al. 2007). Smaller values of $d(s_1, s_2)$ indicate that two adjacent regions are more similar and there is a larger probability of merging with a given threshold. For remote sensing images with abundant spectral information, this kind of measure can achieve better merging effect by making full use of the distance measurement of spectral information in the variation and direction of each band (van der Meer 2001).

Experiments and Results

Data Set A

First, several experiments are carried out to illustrate the effect in Fig. 3 by comparing the results from different data-partitioning strategies. A multi-threaded programming technique is adopted on data set A to implement parallel computation in these experiments. A subset of a panchromatic image from IKONOS satellite over Beijing City, China, is used as the experimental data. As shown in Fig. 5a, the spatial resolution of the image is 1 m, and the original image size is 600×600 pixels. Figure 5b shows the parallel segmentation result using data-partitioning strategy I in the filtering step, where the merging line is clearly visible in the dotted box. Figure 5c shows the

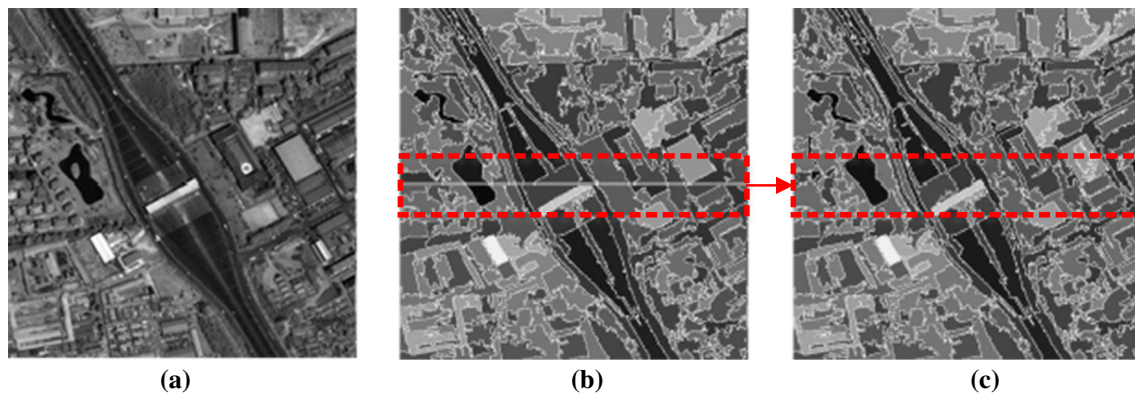


Fig. 5 Comparison of parallel segmentation results using different data-partitioning strategies for the filtering step: **a** original image, **b** segmentation result using data-partitioning strategy I and **c** segmentation result using data-partitioning strategy II

parallel segmentation result using strategy II in the filtering step. The use of the buffer-zone data-partitioning in the filtering step eliminates significant “merging line” on the final segmentation map. The result is nearly the same as the result obtained using a mean-shift without data-partitioning in segmentation, which improves its credibility.

Data Set B

The experiment on data set B aims to further verify the performance of the proposed algorithm. A SPOT5 fused image with 2.5 m spatial resolution and $18,000 \times 12,000$ pixels (the total amount of this remote sensing image is close to 1 GB) is used to assess the efficiency of the parallel implementation. During the experiment, data-partitioning strategy II is employed in the filtering and merging steps, respectively.

Figure 6 presents intermediate filtering results in a local region near the block line with different data-partitioning strategies. It can be clearly found that the results of chunk filtering have obvious errors when they are not buffered, and the effect of filtering result without data-partitioning can be basically achieved using our buffer-zone-based data-partitioning strategy. For this local sub-region, the quantitative analysis of the homogenization degree near the “merging line” is further carried out by calculating the Euclidean distances E_d of the pixels near the “merging line” in the filtering results. The statistical results with 87 pixels near the “merging line” are shown in the following Table 1. The above experimental results reflect the advantages of buffer-zone-based data-partitioning strategy in dealing with image segmentation via synergetic consideration of efficiency and accuracy. It provides a good guarantee for the implementation of subsequent merging step, and can meet the performance needs of practical applications.

Figure 7 presents the following parallel merging process. That is, five data chunks of this image are extracted from the filtering result for the region merging steps. Chunks 1, 2, and 3 are then computed in parallel on different threads. After that, serial computing rather than parallel computing is used for the intersecting chunks. That is, region merging is carried out in order on chunks 4 and 5. The excellent merging result in the final map further illustrates the effectiveness of the developed data-partitioning strategies for mean-shift parallel segmentation.

Figure 8 shows a local sub-region’s merging results (i.e., segmented image objects) with different data-partitioning strategies. For this local sub-region, we further calculate the time consumption of merging and the number of segmented image objects after merging step. From Fig. 8 and Table 2, we find that the data partition (i.e., the methods with strategy I and II) has a significant improvement in the speed of operation. In addition, the buffer-zone-based data-partitioning strategy (i.e., strategy II) sacrificed some computation efficiency by comparing strategy I. However, it basically obtains the precision of the method without data-partitioning whether from visual comparison or the number of segmented image objects, while the precision of the method without buffer (i.e., strategy I) is the lowest among them, and thus, it is obviously not suitable for the application of massive image segmentation. Therefore, it is not difficult to find that the parallel method in this paper obviously improves the speed while guaranteeing the accuracy by properly using the data-partitioning strategy.

To further show the compute performance and acceleration effect on massive image segmentation, we compare our method with the multi-resolution image segmentation algorithm implemented in Definiens’ eCognition software. Two indexes are employed in this

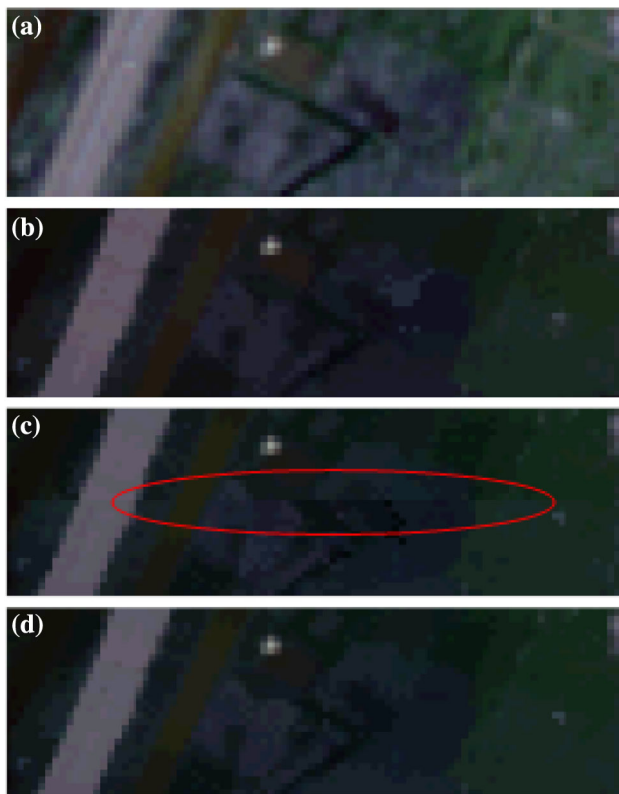


Fig. 6 Comparison of filtering results (true color bands) using different data-partitioning strategies in a local sub-region. **a** original sub-image, **b** filtering result without data-partitioning, **c** filtering result with data-partitioning strategy I, **d** filtering result with data-partitioning strategy II

comparison, namely the computing time and consumption of internal memory as the focus of this paper is how to improve the computational performance of segmentation algorithms. The computer in this experiment is equipped with an Intel(R) Core(TM) i7-2760QM, 2.40 GHz frequency 4-core (four threads) CPU, 8 GB memory and a 64-bit operating system. The experimental results in Table 3 show that the total computing time and maximum internal memory needed during the overall parallel segmentation process for routine mean-shift algorithm are both more than those of eCognition's algorithm, while a significant improvement in the performance of these two aspects is achieved by applying our proposed strategy in mean-shift algorithm (about a third of the processing time is saved and memory utilization is higher). This indicates that our approach has a high speed-up ability and processing stability for the segmentation of large-scale data. Furthermore, because less necessary internal memory is required during the processing, our parallel algorithm makes it possible to segment massive images under limited memory constraints. Consequently, the proposed data-partitioning strategy avoids the limitations of data size and the problem of insufficient computer memory. Hence, the proposed technique can be employed as a robust and rapid segmentation algorithm for massive high-resolution remote sensing images.

Table 1 Quantitative comparison of the homogenization degree near the “merging line” and time consumption of different data-partitioning strategy (for sub-image in Fig. 6)

| Data-partitioning setup | Without data-partitioning | Data-partitioning strategy I with two chunks | Data-partitioning strategy II with two chunks |
|-----------------------------------|---------------------------|--|---|
| Time consumption of filtering (s) | 54.2 | 29.4 | 30.1 |
| Number of pixels ($E_d > 2$) | 19 | 73 | 35 |
| Number of pixels ($E_d > 10$) | 8 | 63 | 17 |

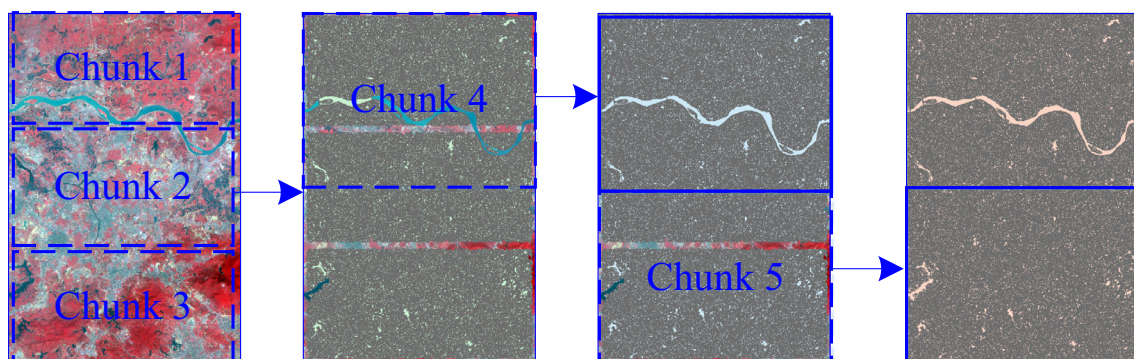


Fig. 7 Data-partitioning strategy in the merging step for parallel segmentation of data set B

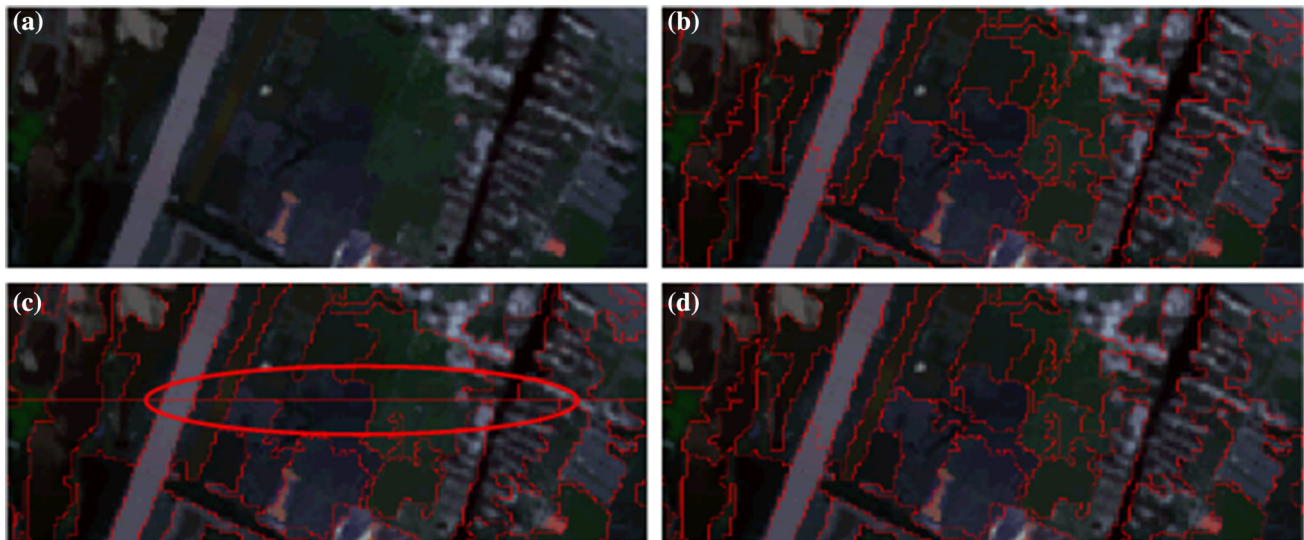


Fig. 8 Comparison of merging results (true color bands) using different data-partitioning strategies in a local sub-region. **a** filtered sub-image, **b** filtering result without data-partitioning, **c** merging result with data-partitioning strategy I, **d** merging result with data-partitioning strategy II

Table 2 Quantitative comparison of the homogenization degree and time consumption of different data-partitioning strategy (for sub-image in Fig. 8)

| Data-partitioning setup | Without data-partitioning | Data-partitioning strategy I with two chunks | Data-partitioning strategy II with two chunks |
|-----------------------------------|---------------------------|--|---|
| Time consumption of merging (s) | 13.7 | 8.3 | 8.8 |
| Number of segmented image objects | 11,451 | 11,474 | 11,459 |

Table 3 Comparison of the computational efficiency of our segmentation algorithm and eCognition's algorithm

| Method | Overall time of parallel segmentation (s) | Maximum necessary internal memory (GB) |
|---|---|--|
| eCognition's algorithm | 1505 | 2.7 |
| Routine mean-shift algorithm | 1791 | 3.4 |
| Mean-shift algorithm using our strategy | 1027 | 2 |

Conclusions

To increase the speed of segmentation for massive remote sensing images, this paper proposed a parallel mean-shift segmentation algorithm. Because the process of the mean-shift algorithm requires global data analysis, conventional simple data-partitioning methods will lead to visible merging lines between different data chunks. According to the characteristics of conventional mean-shift algorithms, we presented a buffer-zone-based data-partitioning strategy in this paper for the parallel computing in the computationally intensive steps. The experiments showed the proposed data-partitioning strategy can effectively solve the

problem of visible merging lines. It was also demonstrated to be a feasible and efficient segmentation algorithm for massive images when computer resources are limited. Therefore, to a certain extent, the proposed strategy can be extended to other similar algorithms for image parallel processing.

Although this paper demonstrated the feasibility of the proposed method from the perspective of algorithm implementation, we did not evaluate how to determine the optimal sizes of the buffer zones and the number of chunks. Moreover, the performance of our algorithm should be further evaluated when implemented using MPI-based cluster parallel computing rather than multi-threaded

computing. All these issues are challenging problems worthy of further investigation.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (Grant Nos: 41631179, 41601437); National Key Research and Development Program (Grant No: 2017YFB0503600); Natural Science Basic Research Plan in Shaanxi Province of China (Grant No: 2017JQ4002); Open Projects of Key Laboratory of Spatial Data Mining and Information Sharing of Ministry of Education, Fuzhou University (Grant No: 2018LSDMIS03) and State Key Laboratory of Geo-information Engineering (No. SKLGIE2017-Z-4-3); Special Fund for Basic Scientific Research of Central Colleges in Chang'an University (Grant No: 310812163504).

References

- Baatz, M., Benz, U., Dehghani, S., Heynen, M., Höltje, A., Hofmann, P., et al. (2004). *eCognition professional user guide 4*. Munich: Definiens Imaging.
- Blaschke, T. (2010). Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(2010), 2–16.
- Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 603–619.
- Dikshit, O., & Behl, V. (2009). Segmentation-assisted classification for IKONOS imagery. *Journal of the Indian Society of Remote Sensing*, 37(4), 551–564.
- Grizonnet, M., Michel, J., Poughon, V., Inglada, J., Savinaud, M., & Cresson, R. (2017). Orfeo ToolBox: Open source processing of remote sensing images. *Open Geospatial Data, Software and Standards*, 2(1), 15.
- Huang, G. M., & Guo, J. F. (2001). Data-partition for distributed-parallel processing of remote sensing imagery. *Remote Sensing Informatics*, 43(2), 9–12.
- Huang, X., & Zhang, L. P. (2008). An adaptive mean-shift analysis approach for object extraction and classification from urban hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 46(12), 4173–4185.
- Huang, Z. J., Zhang, J. F., Li, X., & Zhang, H. (2014). Remote sensing image segmentation based on dynamic statistical region merging. *International Journal for Light and Electron Optics*, 125, 870–875.
- Innocenti, E., Silvani, X., Muzy, A., & Hill, D. (2009). A software framework for fine grain parallelization of cellular models with OpenMP: Application to fire spread. *Environmental Modelling and Software*, 24(7), 819–831.
- Kavzoglu, T., Tonbul, H., Erdemir, M. Y., & Colkesen, I. (2018). Dimensionality reduction and classification of hyperspectral images using object-based image analysis. *Journal of the Indian Society of Remote Sensing*, 8, 1–10.
- Li, X. R., Wu, F. C., Hu, Z., & Hu, Y. (2005). Convergence of a mean shift algorithm. *Journal of Software*, 16(3), 365–374.
- Michel, J., Grizonnet, M., & Youssefi, D. (2014). Mean shift based segmentation of full VHR imagery with limited resources: An exact solution. In *Proceedings of the 2014 IEEE international conference on geoscience and remote sensing symposium (IGARSS)* (pp. 2842–2845). Quebec City, Canada.
- Michel, J., Youssefi, D., & Grizonnet, M. (2015). Stable mean-shift algorithm and its application to the segmentation of arbitrarily large remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 53(2), 952–964.
- Mukherjee, D. P., Levner, Y. P., & Zhang, H. (2009). Ore image segmentation by learning image and shape features. *Pattern Recognition Letters*, 30(6), 615–622.
- Shen, Z. F., Luo, J. C., Chen, Q. X., Huang, G. Y., & Sheng, H. (2006). Data-partition policy of high-resolution remotely sensed image parallel processing. *Journal of Harbin Institute of Technology*, 38(11), 1968–1971.
- Shen, Z. F., Luo, J. C., Huang, G. Y., Ming, D. P., Ma, W. F., & Sheng, H. (2007). Distributed computing model for processing remotely sensed images based on grid computing. *Information Sciences*, 177(2), 504–518.
- Shen, Z. F., Luo, J. C., Wu, W., & Hu, X. D. (2012). A new approach to improve the cluster-based parallel processing efficiency of high-resolution remotely sensed image. *Journal of the Indian Society of Remote Sensing*, 40(3), 357–370.
- Su, T. F., & Zhang, S. W. (2017). Local and global evaluation for remote sensing image segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130, 256–276.
- Thenkabail, P. S., Gangadhara Rao, P., Biggs, T., Krishna, M., & Turrall, H. (2007). Spectral matching techniques to determine historical land use/land cover (LULC) and irrigated areas using time-series 0.1 degree AVHRR pathfinder datasets. *Photogrammetric Engineering and Remote Sensing*, 73(9), 1029–1040.
- Vamsee, A. M., Kamala, P., Martha, T. R., Kumar, K. V., Sankar, G. J., & Amminedu, E. (2018). A tool assessing optimal multi-scale image segmentation. *Journal of the Indian Society of Remote Sensing*, 46(1), 31–41.
- van der Meer, F. (2001). Spectral matching using pixel cross-correlograms for the analysis of LANDSAT TM data. *International Journal of Applied Earth Observation and Geoinformation*, 3(2), 197–202.
- Wang, Y. Z., Liang, Y., Zhao, C. H., & Pan, Q. (2008). Kernel-based tracking based on adaptive fusion of multiple cues. *Acta Automatica Sinica*, 34(5), 393–399.
- Wu, T. J., Luo, J. C., Xia, L. G., Shen, Z. F., & Hu, X. D. (2015). Prior knowledge-based automatic object-oriented hierarchical classification for updating detailed land cover maps. *Journal of the Indian Society of Remote Sensing*, 43(4), 653–669.
- Zalik, K. R., & Zalik, B. (2009). A sweep-line algorithm for spatial clustering. *Advances in Engineering Software*, 40(6), 445–451.
- Zhou, H. F. (2003). *Study and implementation of parallel algorithms for remote sensing image processing*. Beijing: National University of Defense Technology.
- Zhou, C. H., & Luo, J. C. (2009). *Geo-computing for high resolution remote sensing images*. Beijing: Science Press.
- Zoleikani, R., Valadan Zoej, M. J., & Mokhtarzadeh, M. (2017). Comparison of pixel and object oriented based classification of hyperspectral pansharpened images. *Journal of the Indian Society of Remote Sensing*, 45(1), 25–33.