A Parallel Drone Image Mosaic Method Based on Apache Spark

Yirui Wu, Lanbo Ge, Yuchi Luo, Deqiang Teng, and Jun Feng(✉)

College of Computer and Information, Hohai University, Nanjing, China
{wuyirui,junfeng}@hhu.edu.cn, lanbo_cs@163.com, hhu_lyc@126.com, 3247592726@qq.com

Abstract. MapReduce has been widely used to process large-scale data in the past decade. Among the quantity of such cloud computing applications, we pay special attention to distributed mosaic methods based on numerous drone images, which suffers from costly processing time. In this paper, a novel computing framework called Apache Spark is introduced to pursue instant responses for the quantity of drone image mosaic requests. To assure high performance of Spark-based algorithms in a complex cloud computing environment, we specially design a distributed and parallel drone image mosaic method. By modifying to be fit for fast and parallel running, all steps of the proposed mosaic method can be executed in an efficient and parallel manner. We implement the proposed method on Apache Spark platform and apply it to a few self-collected datasets. Experiments indicate that our Spark-based parallel algorithm is of great efficiency and is robust to process low-quality drone aerial images.

Keywords: Distributed mosaic method · Parallel processing · Apache Spark · Big data · Drone aerial image

1 Introduction

In recent years, drones have shown significant potential to satisfy the demands of outdoor aerial exploration in different fields, such as aerial photography, agricultural, disaster observation and military purposes [7]. During a variety of applications, one of the key problems is to provide a high-resolution image with an aerial view for large areas, which could offer sufficient and abundant information for further analysis.

Generating a large and high-resolution aerial image with a traditional and single-threaded mosaicking method is essentially low in running speed since it requires two key and time-consuming steps, i.e., image registration and image stitching, for quantity of high-resolution drone images. The former step is designed to register the image in the same coordinate system to discover the correspondence relationships among images with varying degrees of overlap, meanwhile, the latter step is to generate a high-resolution image by stitching

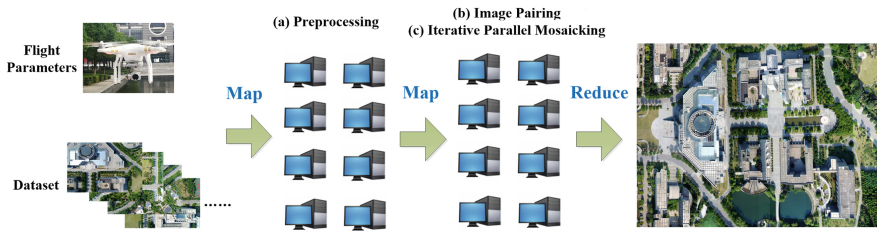

Fig. 1. Framework of the proposed method to mosaic drone aerial images.

the registered images. Image stitching algorithms take alignment estimates produced by such registration algorithms as input. Afterwards, they are responsible to blend the images seamlessly.

With the significant development of cloud computing for big data processing, we try to utilize distributed computing idea and appropriate framework to save computing time and efforts for drone image mosaicking. More precisely, we proposed to accelerate the efficiency of the traditional mosaicking method by modifying its structure with parallel and distributed computing idea, and utilize Spark framework to implement drone image mosaic processing for instant running speed. Supposing hundreds of drones have uploaded different sequences of aerial images to the proposed mosaicking system, we show the framework of the proposed system in Fig. 1, including steps of (a) preprocessing, (b) roughly drone image pairing and (c) iterative parallel mosaicking with PCA-SIFT features.

Specifically, corrections of perspective distortions and extraction of PCA-SIFT features are performed during step (a). During step (b), we roughly pair images based on image similarity before mosaicking, since it not only saves time-consuming mosaicking trials but also helps improve traditional mosaicking method to fit for distributed and parallel computing with such a light-weight pre-processing part. Finally, a novel and distributed mosaicking algorithm based on PCA-SIFT feature is carried out on Spark framework during step (c). It's noted that mosaicking process of step (c) is performed iteratively until all input drone images are integrated to generate a single and final aerial image. Moreover, we utilize two operations of Map and one operation of Reduce to construct the proposed "MapReduce" system. The reason that we could design two separate Map operations lies in the fact that steps of preprocessing and rough pairing could be executed without interactions among different input drone images.

The main contribution of this paper is to propose a parallel and distributed drone images mosaicking method, which has been successfully implemented on Spark platform. Essentially, how to design task-specified and cloud-based systems to perform image processing tasks with high efficiency is a major concern in the domain of cloud computing for big data processing. We try to complete such tasks by design a novel drone images mosaicking method based on the traditional mosaic method, which is further improved with thoughts of parallel and distributed running. After implementation on Spark with in-memory representation, results on several self-collected datasets with relatively cheap drones have proved that the proposed system is of great efficiency and accuracy for drone image mosaicking tasks.

2 Related Work

The existing methods related to our work can be categorized into image mosaicking methods and the development of cloud computing.

Mosaicking Methods. Essentially, mosaicking is well-developed in the domain of image processing. Generally speaking, it consists of five steps [13]: 1) image pre-processing, including operations of image denoising, creating image matching templates and so on; 2) image matching, which is the key step to find the exact position of feature points between the original and reference image; 3) calculate transformation parameters of two images in the geometric model based on matching feature points; 4) uniform coordinate transformation, which converts the image to be stitched into the coordinate system of the reference image; 5) fusion reconstruction, where the overlapping regions of both images are fused to reconstruct a large image. It's noted that the second and third steps refer to image registration procedure, meanwhile, the fourth and fifth steps represent the procedure of image mosaicking.

With similar structure explained above, quantity of mosaicking methods is proposed to deal with different situations in a variety of applications. For example, Wang et al. [20] convert a surveillance video clip into one abstraction image with SIFT-based image aligning and integrating. Zhang et al. [23] propose an image mosaic method based on Speed Up Robust Feature (SURF) to achieve rapid image splicing in UAV low-altitude remote sensing field. To obtain the morphology of the whole microstructure, Wang et al. [18] proposes a SIFT-based mosaic algorithm by using multiple images from a microgroove structure processed by femtosecond laser. Afterwards, a stitched image of the whole groove structure could be studied experimentally and realized.

Based on former successful implements of image mosaicking methods, we can notice it's useful to adopt the classic SIFT algorithm for step 2, i.e., image matching. To solve problems of translation, rotation and scale variances between original and reference images, the SIFT algorithm realizes the matching of the extracted feature points and forms a high quality panoramic image by matching several feature points [2]. However, directly utilizing SIFT for large-scale mosaicking is impossible, due to the large amount of computation brought by quantity trials of image matching between each pair of input images. Therefore, drone image data with the huge volume feature cannot be directly mosaicked by a single computing node with a traditional mosaicking algorithm.

Development of Cloud Computing. Faced with "big data" challenge [10], adopting a parallel computing architecture such as MapReduce [4] is a proper choice to mosaic quantity of drone images. MapReduce is carefully designed to store and parallel process large volumes of data in a distributed network. By separating available disk, computing resource and scheduling jobs, its successor platforms, such as Hadoop Standalone and Hadoop YARN [17], have become popular for big data operating. Recently, Apache Spark (Spark), an innovative framework for real-time processing, has been proposed by the University of California, Berkeley to perform in-memory computing and big data analysis [22]. In Spark, Berkeley proposed to utilize internal memory rather than disk to process in MapReduce, considering the fact that internal memory channels have higher bandwidth than disks and other PCI devices. Based on the Spark platform, Amazon.com constructed a general-purpose and scalable in-memory computing framework to process large-scale data at extremely high speed (reported 100-TB data sort in 23 min on 207 nodes) [21]. It's great potential to use the Spark framework on Hadoop platform to mosaic large-scale drone images. However, How to design a task-specified mosaic algorithm to fit parallel programming and utilize features of drone images, and what the performance gain is remain to be explored.

Most related to our project, Huang et al. [5] propose a strip-oriented parallel programming model to facilitate the design of generic remote-sensing (RS) algorithms using an Apache Spark on Hadoop YARN. They report a multitasking algorithm took less than 4 h to process more than half a terabyte of RS data on a small YARN cluster, which proved their proposed algorithms are of great efficiency. However, they pay special attention to generic parallel RS algorithms other than a complicated and task-specified algorithm, such as mosaic, detection, classification and so on. Plaza et al. [14] introduce an open-source segmentation framework that runs efficiently on large electron microscopy (EM) image datasets. Their proposed framework is implemented in Apache Spark and laid over DVID [1], a volume data-service for accessing and versioning volume data. It's reported that ingestion and segmentation with the proposed system on 453 GB EM data only took 1 h, which demonstrates the scalability and efficiency by applying image processing algorithms on the Spark platform.

3 Methodology

In this section, we describe the proposed system by preprocessing, Drone Images Pairing based on Similarity and Iterative Parallel Mosaicking with PCA-SIFT features.

3.1 Preprocessing

In this subsection, we first utilize flight parameters as input to correct perspective distortions and then extract PCA-SIFT features for further mosaicking.

Essentially, adopting a cheap consumer drone to collect data would bring several matters to quality of the inputting images. Among these matters, perspective distortion caused by unsteady flight would harm visual effect of the final mosaicking image without preprocessing. Therefore, the core of the preprocessing step is to design suitable Correlation filters (CF), which acts as a measure of similarity

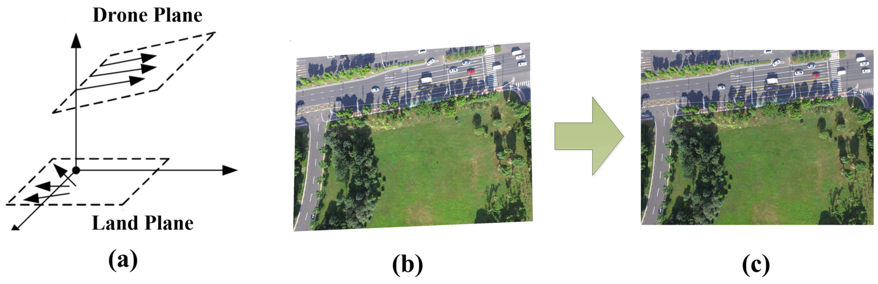

Fig. 2. Illustration of perspective distortion model. (a) Perspective distortion is caused by the angle between the camera of drone and land planes, (b) input image with perspective distortion, and (c) the image after correction of perspective distortion.

between two functions. Many CF designs can be interpreted as optimizing a distance metric between an ideal desired correlation output for an input image and the correlation output of the training images with the filter template [15].

We model the discussed problem as shown in Fig. 2. As a result, the projections from land plane to the imaging plane, i.e., drone plane should be a group of divergent trapezoid shapes. Therefore, perspective distortion correction for the drone is actually a reverse trapezoid-based compensation problem. According to the above analysis, we use trapezoid-based reverse transformations for perspective distortion correction. Suppose (x, y) is a pixel from a captured aerial image, while (x', y') is its corresponding corrected point, then (x, y) and (x', y') obey the following linear transformation model:

$$\begin{bmatrix} x' \\ y' \\ \omega' \end{bmatrix} = \begin{bmatrix} S_u & 0 & 0 \\ 0 & S_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \omega \end{bmatrix} \tag{1}$$

where (ω, ω') are Homogeneous coordinates and S_u , S_v are flight parameters in X-axis and Y-axis achieved directly from the drone we use in the experiments. We show an example result in Fig. 2, where (b) and (c) represent the image with perspective distortion and after correction, respectively.

Due to camera auto exposure and white balance affected by ambient light, there exists a difference in brightness and color between the images. This is contrary to the brightness constancy, so we should adjust the brightness of the images. We can use the sample point to adjust image brightness. The three image channels RGB are handled separately, and the difference in brightness and color can be approximated by the following linear equation:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} a_r & 0 & 0 \\ 0 & a_g & 0 \\ 0 & 0 & g_b \end{bmatrix} \cdot \begin{bmatrix} R_1 \\ G_1 \\ B_1 \end{bmatrix} + \begin{bmatrix} b_r \\ b_g \\ b_b \end{bmatrix} \tag{2}$$

where a and b are linear transformation parameters. The sample points are chosen from the SIFT match previous step and its neighbor points. MSAC [3] was used to refine the sample points. Regression analysis is used to fetch the linear transform parameters of the three channels.

After correction, we extract PCA-SIFT features to prepare for further mosaicking. SIFT is proved to be a robust and stable descriptor in image matching or content analysis [8,9]. However, the extracted 128-dimension SIFT feature makes matching quite inefficient, especially for instant drone image mosaic. We thus utilize PCA-SIFT [6], a PCA (Principal Components Analysis) version of original SIFT features, to decrease the dimensions of feature space and fasten the process of image mosaic. Besides, for images containing a large number of straight lines (such as buildings, etc.), the different angles' collection of the same object by the drone will make the contour not parallel. To increase the quality of image mosaic, we made some small rotation transformation.

3.2 Drone Images Pairing Based on Similarity

In this subsection, we mainly describe the proposed algorithm to roughly pair drone images based on the similarity between two images.

Calculating the similarity between two images is a hot topic in computer vision [19,25]. Former methods adopt elementary image processing to calculate, such as histogram, Average Hash and Perceptual Hash [24]. These methods are not only fast in speed, but also simple and straightforward in implementation. However, they are not effective in calculating category-level image similarity [16]. Inspired by the significant power of deep learning architectures, researchers propose a deep ranking model to calculate and sort similarity among category-level images. For example, Wang et al. [19] integrate deep learning techniques and fine-grained ranking model to learn the image similarity ranking model directly from images.

For the proposed method, we aim to utilize a paring algorithm to avoid time-consuming mosaicking trials and convert the following parallel mosaic algorithm to fit for parallel running. Considering the feature of suitability for small computation tasks of the Spark-based system, we adopt histogram, a light-level but highly effective for resemble images similarity calculating algorithm, to be the pairing algorithm. In fact, we utilize histogram to search and pair two aerial images by calculating the similarity between the subareas of two images captured in a disorderly flying video sequence, which could be written as following:

$$Sim(G,S) = \max_{k=1,2} \sum_{i=1}^{N} \sum_{i=1}^{255} (1 - \frac{|g_{i,j,k} - s_{i,j,k}|}{\max(g_{i,j,k}, s_{i,j,k})}) \tag{3}$$

where g and s refer to the histograms of G and S respectively, N represents the blocks we split for each subarea and k represents the number of subareas. Note that k is settled with 2, which represents we utilize the left and right parts of aerial images with preset areas to calculate. In that way, we aim to search horizontal neighboring images and pair them up. Splitting blocks are used to improve the robustness of the histogram method, preventing similar color of

G, S results in high similarity. After calculating similarity, we roughly pair two images by

$$P(I_i) = \arg\max_{j} Sim(I_i, I_j), where\ I_i, I_j \in \psi \tag{4}$$

where ψ refers to the set of unpaired images. We show two successful mosaicked samples in Fig. 3 to illustrate the calculation of similarities, where we can notice the right part of (a),(e) and the left part of (b),(f) are similar in both appearance and corresponding histograms.

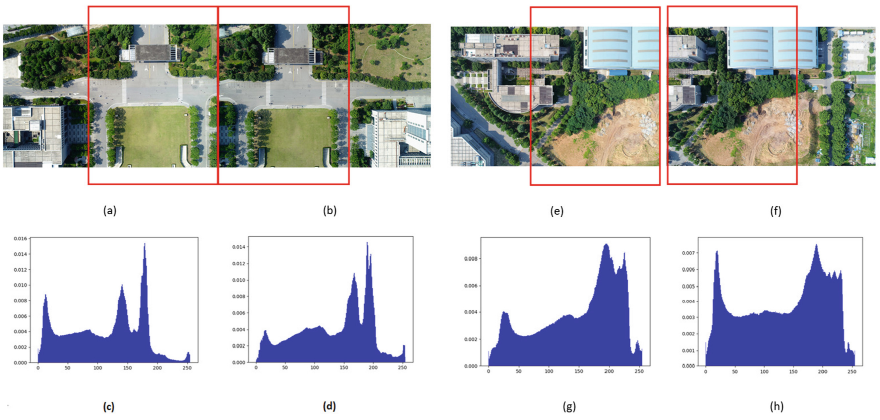

Fig. 3. Sampling images and their corresponding histograms.

The importance of such roughly pairing operation lies in the fact that we cannot achieve a parallel workflow without pre-arrangement. More precisely, the collected images are often disordered and will not succeed if mosaicked directly because they share few same features. We need to do similarity matching to adjust these images to a sequence that can be mosaicked correctly, that is, two images that can be mosaicked put together. Specifically, for an image, another image that is most likely to be mosaicked with it is selected through similarity comparison. Thus, the image sequence that can be mosaicked correctly can be generated through multiple operations.

3.3 Iterative Parallel Mosaicking with PCA-SIFT

In this subsection, we describe the proposed iterative parallel mosaicking algorithm.

We show the workflow of the proposed mosaic algorithm in Fig. 4 . We can observe the input is a pair of drone images $P_{m,t,r} = \{I_{i,t,r}, I_{j_t,r}\}$ computed by former subsection, where t and r represents the index of turn and iteration, respectively. We firstly judge whether $t < \eta$ to prevent cycle running, where η equals 4 in experiments. Then, we follow [20] to try mosaicking between paired

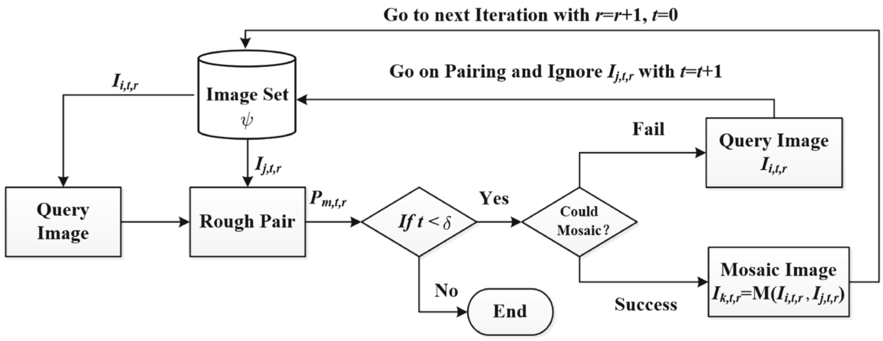

Fig. 4. Workflow of the proposed Iterative Parallel Mosaicking algorithm for disorderly fly model.

subareas of $I_{i,t,r}$ and $I_{j_t,r}$, which first utilize PCA-SIFT features to calculate distance and then properly integrate similar parts between two images. Specifically, we need to establish the mathematical relationships that map pixel coordinates from one image to another. For standard lens images, the mathematical model is usually established by affine transformation, and the least squares method is used to optimize the parameters, which can be expressed as $I_{k,t,r} = M(I_{i,t,r}, I_{j_t,r})$.

Due to the rough pairing results, we could get wrong trials especially for location with homogeneous visual appearances. With a wrong trial, the proposed algorithm goes on pairing in the next turn with ignoring $I_{j_t,r}$. With a successful mosaic trial, the proposed algorithm outputs the mosaic image $I_{k,t,r}$ and puts it in dataset ψ for next iteration of computation by setting $r = r + 1$ and $t = 0$.

Essentially, the proposed iterative mosaicking algorithm could be parallel performed. We could utilize spark nodes to execute the workflow presented in Fig. 4 at the same time, where the number of the nodes could be defined as half of the size of ψ for each iteration and we only need a lock scheme on acquiring permissions of the image set to guarantee the successfully parallel running. The highly parallel performance of the proposed mosaicking algorithm is actually ensured by a simple but effective pair algorithm and a short pipeline length, *i.e.*, the maximal turn number η. It's noted that several corner images may result in unpaired images with $t \geq \eta$, which will be regarded as images in ψ for the next iteration.

3.4 Implementation Details

We adopt spark to be the implementation framework and believe Spark to be an ideal framework for several reasons:

- An in-memory representation for large label data will empower future algorithms to use high-level, global context to improve mosaic quality.
- Spark expands the widely-used MapReduce model by supporting many computational models, such as interactive query and stream processing, and supporting to work with other famous big data tools, such as Hadoop clusters.

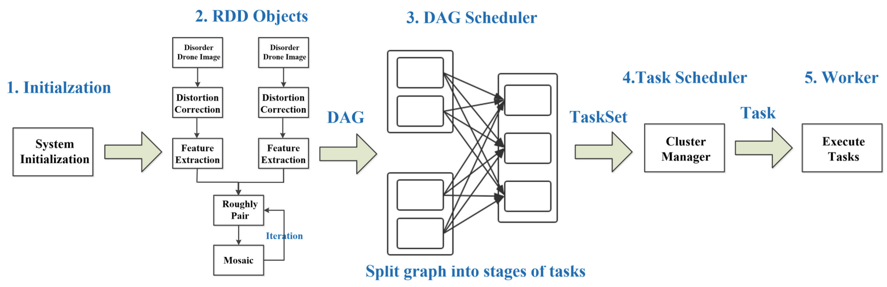

Fig. 5. Execution workflow of the proposed mosaic algorithm implemented on Spark.

– The core component of Spark is fast and versatile so that it supports to be designed for a variety of application scenarios, such as machine learning and image processing.

The implementation is built on the core component of Spark since it supports fast response for requests. The Core component consists of different modules, such as task scheduling, memory management, error recovery, interaction with the storage system and so on. It also includes definitions of APIs for Resilient Distributed Datasets (RDDs), which can be operated parallel on multiple compute nodes. As shown in Fig. 5, we describe detailed steps as below:

1. Read dataset of drone images and set initial parameters based on data size and node configuration. We specially define a reasonable partition scale parameter to maximize the usage of cluster resources by guidance of [5]. Apply for Executor resource to Hadoop Standalone and start the Standalone Executor.
2. According to the total workflow of the proposed mosaicking algorithm, perform MAP operation to define the corresponding RDD objects, which further construct a directed acyclic graph (DAG). From Fig. 5, we can notice steps of distortion correction and feature extraction could be parallel executed, since such RDD objects for each image is irrelevant with other images. Meanwhile, steps of pair and mosaic could be parallel performed for every two images if designing the proper lock scheme.
3. Through the operation of Reduce, DAG is translated into a physical execution plan by splitting DAG into stages of tasks and tracking parent nodes.
4. With the physical execution plan, the Spark task scheduler would submit a job to perform these operations. Note that each job contains one or more procedures, which consist of a series of parallel computing tasks.
5. Every procedure, corresponding to one or more steps in the DAG, would be performed with multiple parallel pipelines inside one worker.

The experiment environment consists of six nodes. We used 2 Amax PSC-HB1X servers. Both of them have 24 cores, 12 GB main memory and 2 TB disk. We employ a VMware Workstation Pro 12.1.0 to create 4 more virtual nodes and all of them hosted on Amax PSC-HB1X servers. Therefore, each node owns 4 cores and 2 GB in total. All nodes adapt as OS 14.04.2 LTS. The Spark version is Spark 2.1.3. The JDK version is JDK 1.8.0. The python interpreter version is python 3.5.2. Daemons of the Hadoop master and the Spark master are running on the same nodes, with other nodes being used for storage and computing. Therefore, we have one spark master and five spark slaves in total.

4 Experiments

In this section, we show the effectiveness and efficiency of the proposed system for drone image mosaicking tasks. We first introduce dataset and measurements, and then describe performance analysis on both time and quality.

4.1 Dataset and Measurement

We apply the proposed drone image mosaic method on a self-collected dataset, which contains 5840 images with 31.38 GB size. All these data are captured at six locations named Campus, GeneralMount, CowheadMount, XiaolongBridge, JinzhiFactory, BaijiaLake, and the corresponding captured image numbers are 540, 810, 620, 1000, 1540 and 1330. We use a DJI Phantom 3 drone to collect data. Samples of data and the drone are shown in Fig. 2. Note that we achieve aerial images by firstly taking videos and then equally sampling videos. For the video captured at Campus, we fly the drone at 140 m to get aerial images at 4700 * 3200 resolution. For other videos, we fly the drone at 90 m to get images at 4000 * 3000 resolution. In fact, DJI Phantom 3 drone is developed for consumer utilization, so that it's cheap in cost, *i.e.*, only around 600 dollars. Therefore, the proposed method could be further developed as an online service to instantly offer mosaicking aerial images, even if users upload low quality sequences with consumer level drones.

To evaluate the performance of the proposed system for mosaicking, we utilize two popular methods, i.e., NIQE [12] and BRISQUE [11], to measure the quality of generated aerial image, where both measurements are image quality estimation algorithm without reference, NIQE refers to Natural Image Quality Evaluator, and BRISQUE represents Blind/Referenceless Image Spatial Quality Evaluator. The detail of these two measurements can be found in [12] and [11].

4.2 Performance Analysis

To evaluate the proposed mosaic method on Spark, we compare it on six sets of collected drone images with the single process (not using Spark on a single node) and using Spark Local mode. Spark Local mode refers to local operating mode with the single node, which simulates Spark distributed computing environment with multiple threads of a single node. Note that methods running on the single process and Spark Local mode are the same as that on P[2] and P[5]. Table 1 gives the time comparison, where number in "Campus(540)" refers to the drone image number, L[N] represents Spark Local Mode, N means that N threads are used and each thread occupies a core, P[M] and P[M]' refers to the proposed

Table 1. Comparison of running time on collected dataset with the proposed method, Spark Local mode and single process.

Sequence	Order fly mode(s)				Disorder fly mode(s)				
	P[2]	P[5]	Local[4]	Single	P[2]	P[5]	P[5]'	Local[4]	Single
Campus(540)	5248.0	**3913.3**	7195.3	9496.4	5401.7	**3845.2**	5441.0	7392.2	9855.9
GMount(810)	4534.7	**2766.2**	5895.2	7841.2	4718.0	**2810.1**	11042.1	6128.7	8263.0
CMount(620)	3943.2	**2029.4**	4832.1	6185.5	4094.2	**2058.9**	6444.1	5022.7	6514.3
XBridge(1000)	5294.8	**2536.2**	7140.8	9923.3	5544.4	**2617.9**	16723.1	7456.8	10485.7
JFactory(1540)	7812.9	**3554.7**	10694.1	15017.3	8305.2	**3803.4**	39249.8	11309.8	16041.7
BLake (1330)	6762.9	**3017.7**	8735.0	12927.3	7130.4	**3210.7**	29523.0	9197.6	13717.8
Average	5599.4	**2969.6**	7415.4	10231.8	5865.7	**3057.7**	20817.7	7751.3	10813.1

method running with M computation nodes with and without parallel design, respectively. Moreover, we utilize brute force to try matching drone images with either left or right parts, which is the same as the proposed pair algorithm. The reason to set N and M as 4 and 5 lies in the fact that each node owns 4 cores and the spark cluster has 5 nodes, respectively. We also design two modes for flying, i.e., orderly fly and disorderly fly. The former implies upload drones offer images in consist orders, while the latter means drones have flown in a free way. By comparing these two modes, we can notice how Spark platform and parallel mosaicking algorithm help perform tasks with higher efficiency. To offer a more intuitive sense on how Spark-based mosaic method help, we show a comparison graph with ratio values on disorder fly mode in Fig. 6.

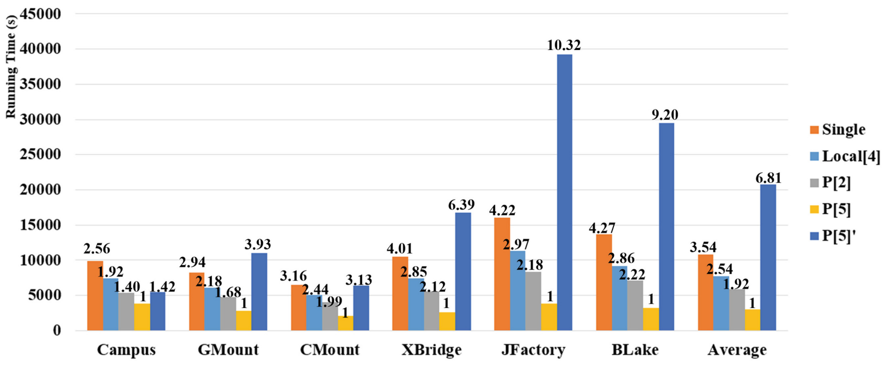

Fig. 6. Comparison graph and ratio values of running time on collected dataset under disorder fly mode. Note that numbers above refer to ratio values and the basis number is running time achieved by P[5].

As shown in Table 1, P[5] achieves the lowest running time in all 6 drone image sets. This is reasonable since P[5] utilizes the larger amount of computation resources than other comparative methods. By introducing parallel modifications, *i.e.*, roughly pair and iterative parallel mosaicking, we could notice the large amount of computation time is saved by comparing running time of P[5] and P[5]' under disorder fly mode. It's noted the times of saving time are unstable, since brute force adopted by P[5]' is not stable in performance and running time of P[5] is highly related to visual layout of scene. Moreover, we could find consuming time in disorder fly and order fly modes are nearly same with parallel modifications in all datasets. All these facts prove that parallel modifications are robust, efficient and fit for parallel executing.

We pay special interest to running time values under disorder fly mode, since it's more suitable than order fly mode for situations of how users use consumer drones. As shown in Fig. 6, we could notice that the running time of P[2] is nearly twice as large as that of P[5]. However, the computation resources of P[5] is 2.5 times larger than that of P[2], which proves that there is a loss in computation resources with more nodes. Drawing a map to analyze trends of computation loss with nodes requires more computation resource and is included in our future work. By comparing Local[4] and P[2], we could find a decrease in running time. However, it's not linearly related to the computation resource, which is due to the computation loss as well. But the extendibility of P[2] is far larger than Spark local mode, since we can easily involve more distributed computation nodes in the Spark cluster. With Spark system to MapReduce tasks, we can observe a quite large decrease in running time between Local[4] and Single using the same computation resource, which proves that the Spark system is much more appropriate for fast and parallel computation than multithread scheme.

Sample qualitative results of the proposed method are shown in Fig. 7, where original images of (a) are captured under order fly mode, and images of (b) and (c) are captured under disorder fly mode. From Fig. 7, we can see the proposed method could successfully mosaic whole aerial images with desirable visual effects. However, we can still view some artifacts, especially for regions with no obvious objects, such as trees, grass and so on. This is due to the reason that such objects are lack robust PCA-SIFT features to mosaic and quality of aerial images captured by consumer drones is not good enough.

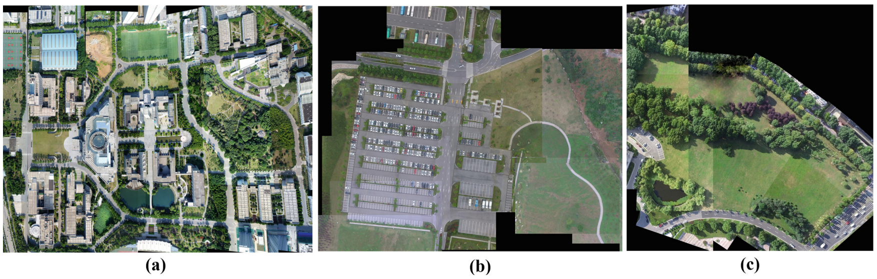

Fig. 7. Mosaic Results of Drone aerial images with the proposed method on (a) Campus, (b) CowheadMount and (c) GeneralMount dataset.

Finally, we use two measurements, i.e., NIQE and BRISQUE, to evaluate quality of generated aerial image, where we show exact results on three sets of self-collected data in Table 2. From the table, we can notice the decrease in calculation values before and after mosaic processing, which helps to verify the accuracy and robustness of the proposed mosaicking algorithm for drone images.

Table 2. The quality measures before and after the mosaic.

Data	Metric	Before process	After process
Campus	BRISQUE	42.86	41.51
	NIQE	16.38	15.85
GMount	BRISQUE	42.75	42.20
	NIQE	16.37	15.13
CMount	BRISQUE	43.24	42.77
	NIQE	16.62	16.18

5 Conclusions

In this paper, we propose a parallel drone image mosaic method to assure its high performance in a Spark-based cloud computing environment. The proposed method makes its steps to be fit for fast and parallel running to run efficiently on Spark-based system. The proposed method is implemented and tested on six self-collected dataset. Comparative results show that the proposed method is of great efficiency and could achieve visual desirable images even with low-quality input images. Our future work includes applying the proposed mosaic method on a large spark-based cluster to offer instant on-line mosaicking service and improve traditional image processing methods by transforming them to be fit for the Spark-based system.

References

1. Distributed, versioned, image-oriented dataservice (dvid). http://github.com/janelia-flyem/dvid
2. Brown, M., Lowe, D.G.: Automatic panoramic image stitching using invariant features. Int. J. Comput. Vision **74**(1), 59–73 (2007). https://doi.org/10.1007/s11263-006-0002-3
3. Capel, D.: Image mosaicing and super resolution. Ph.D. thesis, University of Oxford (2004)
4. Dean, J., Ghemawat, S.: Mapreduce: a flexible data processing tool. Commun. ACM **53**(1), 72–77 (2010)

5. Huang, W., Meng, L., Zhang, D., Zhang, W.: In-memory parallel processing of massive remotely sensed data using an Apache Spark on Hadoop YARN model. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **10**(1), 3–19 (2017)
6. Ke, Y., Sukthankar, R.: PCA-SIFT: a more distinctive representation for local image descriptors. In: Proceedings of Computer Vision and Pattern Recognition, vol. 2, p. II. IEEE (2004)
7. Lee, J.N., Kwak, K.C.: A trends analysis of image processing in unmanned aerial vehicle. Int. J. Comput. Inf. Sci. Eng. **8**(2), 2–5 (2014)
8. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94
9. Lyu, M.R., Song, J., Cai, M.: A comprehensive method for multilingual video text detection, localization, and extraction. IEEE Trans. Circuits Syst. Video Techn. **15**(2), 243–255 (2005)
10. Ma, Y., et al.: Remote sensing big data computing: challenges and opportunities. Future Gener. Comput. Syst. **51**, 47–60 (2015)
11. Mittal, A., Moorthy, A.K., Bovik, A.C.: No-reference image quality assessment in the spatial domain. IEEE Trans. Image Process. **21**(12), 4695–4708 (2012)
12. Mittal, A., Soundararajan, R., Bovik, A.C.: Making a "completely blind" image quality analyzer. IEEE Signal Process. Lett. **20**(3), 209–212 (2013)
13. Moravec, H.P.: Rover visual obstacle avoidance. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp. 785–790 (1981)
14. Plaza, S.M., Berg, S.E.: Large-scale electron microscopy image segmentation in spark. arXiv preprint arXiv:1604.00385 (2016)
15. Rodriguez, A., Boddeti, V.N., Kumar, B.V.K.V., Mahalanobis, A.: Maximum margin correlation filter: a new approach for localization and classification. IEEE Trans. Image Process. **22**(2), 631–643 (2013)
16. Taylor, G.W., Spiro, I., Bregler, C., Fergus, R.: Learning invariance through imitation. In: Proceedings of Conference on Computer Vision and Pattern Recognition, pp. 2729–2736 (2011)
17. Vavilapalli, V.K., et al.: Apache Hadoop YARN: yet another resource negotiator. In: Proceedings of ACM Symposium on Cloud Computing, pp. 5:1–5:16 (2013)
18. Wang, F.B., Tu, P., Wu, C., Chen, L., Feng, D.: Multi-image mosaic with SIFT and vision measurement for microscale structures processed by femtosecond laser. Opt. Lasers Eng. **100**, 124–130 (2018)
19. Wang, J., et al.: Learning fine-grained image similarity with deep ranking. In: Proceedings of Conference on Computer Vision and Pattern Recognition, pp. 1386–1393 (2014)
20. Wang, L.M., Wu, Y., Tian, Z., Sun, Z., Lu, T.: A novel approach for robust surveillance video content abstraction. In: Qiu, G., Lam, K.M., Kiya, H., Xue, X.-Y., Kuo, C.-C.J., Lew, M.S. (eds.) PCM 2010. LNCS, vol. 6298, pp. 660–671. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15696-0_61
21. Xin, R., Deyhim, P., Ghodsi, A., Meng, X., Zaharia, M.: Graysort on apache spark by databricks. GraySort Competition (2014)
22. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of 2nd USENIX Workshop on Hot Topics in Cloud Computing (2010)
23. Zhang, W., Li, X., Yu, J., Kumar, M., Mao, Y.: Remote sensing image mosaic technology based on SURF algorithm in agriculture. EURASIP J. Image Video Process. **2018**(1), 1–9 (2018). https://doi.org/10.1186/s13640-018-0323-5

24. Zhou, G., et al.: Paper infrared image retrieval of power equipment based on perceptual hash and surf. In: Proceedings of International Conference on Advanced Infocomm Technology (ICAIT), pp. 387–392. IEEE (2017)
25. Zhou, Z., Wang, Y., Wu, Q.J., Yang, C.N., Sun, X.: Effective and efficient global context verification for image copy detection. IEEE Trans. Inf. Forensics Secur. **12**(1), 48–63 (2017)