Efficient large-scale oblique image matching based on cascade hashing and match data scheduling

Qiyuan Zhang [a], Shunyi Zheng [a], Ce Zhang [b,c], Rui Li [a], Xiqi Wang [a]

[a]School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China.

[b]Lancaster Environment Centre, Lancaster University, Lancaster LA1 4YQ, UK.

[c]UK Centre for Ecology & Hydrology, Library Avenue, Lancaster LA1 4AP, UK.

ABSTRACT

In this paper, we design an efficient large-scale oblique image matching method. First, to reduce the number of redundant transmissions of match data, we propose a novel three-level buffer data scheduling (TLBDS) algorithm that considers the adjacency between images for match data scheduling from disk to graphics memory. Second, we adopt the epipolar constraint to filter the initial candidate points of cascade hashing matching, thereby significantly increasing the robustness of matching feature points. Comprehensive experiments are conducted on three oblique image datasets to test the efficiency and effectiveness of the proposed method. The experimental results show that our method can complete a match pair within 2.50~2.64 ms, which not only is much faster than two open benchmark pipelines (i.e., OpenMVG and COLMAP) by 20.4~97.0 times but also have higher efficiency than two state-of-the-art commercial software (i.e., Agisoft Metashape and Pix4Dmapper) by 10.4~50.0 times.

Keywords: Oblique image matching; Feature point matching; SIFT; Cascade hashing; Match data scheduling; Structure from motion.

1. Introduction

With the continuous development of unmanned aerial vehicles (UAVs) and oblique imaging technology, oblique images have been employed for surface 3D reconstruction of large-scale scenes such as cities (Xu et al., 2016). Precise camera poses are mandatory to utilize oblique images in 3D reconstruction, which can be obtained by the airborne GNSS/IMU (Global Navigation Satellite System/Inertial Measurement Unit) system. However, limited by measurement accuracy, the image POS (position and orientation system) data obtained through the airborne GNSS/IMU, and installation angle cannot meet the requirement of direct image positioning and orientation accuracy. In the computer vision community, Structure from Motion (SfM) is able to solve camera poses and 3D points automatically from overlapped images with high accuracy (Snavely et al., 2008; Westoby et al., 2012; Rupnik et al., 2013; Schönberger et al., 2014). In SfM technology, a key step is image matching, which occupies approximately half of the computational cost (Cheng et al., 2014).

Image matching aims to find corresponding points automatically between overlapping images based on a specific similarity measure, which is an important research topic in the field of photogrammetry and computer vision (Gruen et al., 2012). According to matching primitives, image matching technologies can be divided into three categories：

point matching, line matching, and region matching (Cheng et al., 2014). Since the invention of scale-invariant feature transform (SIFT) (Lowe, 2004), point matching methods have become the mainstream for oblique image pipelines thanks to their robustness to changes in scale, illumination, and viewpoint (Jiang et al., 2017b). Nevertheless, the time complexity of point matching is high. There are two directions to increse the efficiency of point matching, including the improvement of the algorithm and the utilization of GPU computing. The former is related to design lightweight algorithms such as speeded-up robust features (SURF) (Herbert et al., 2008), oriented fast and rotated brief (ORB) (Rublee et al., 2012), while the latter aims to utilize the parallel computing capability of GPUs (Wu, 2007; Xu et al., 2017; Li et al, 2019).

Within the field of oblique image matching, several open-source libraries and commercial software have been developed and released over the past decade, such as OpenMVG (Open Multiple View Geometry) (Moulon et al., 2016), MicMac (MicMac, 2018), Agisoft Metashape (Agisoft, 2020) and Pix4Dmapper (Pix4Dmapper, 2020). The state-of-art matching pipeline such as Pix4Dmapper (Pix4Dmapper, 2020), however, still requires 1.75 hours to match the oblique image with 14,255 images. As a consequence, a solution to increase the matching efficiency of oblique images is urgently needed for 3D reconstruction over large-scale scenes.

Oblique images are characterized by a large amount of data and a high degree of spatial overlap between images, resulting in huge complexity in the combination of match pairs (Jiang et al., 2020). When GPU card is used to accelerate the matching process of oblique images, this will create a significant amount of redundant transmission of match data between the disk and the graphic memory, leading to high time cost and insufficient use of computational resources. Alternatively, the cascade hashing algorithm has high efficiency for feature point matching, but the matching accuracy and reliability are low compared with multi-random k-d trees algorithm (Silpa-Anan and Hartley, 2008). Given these gaps in either accuracy or efficiency, this paper proposed a novel efficient large-scale oblique image matching method that can achieve a competitive accuracy compared with state-of-the-art methods but with much high efficiency. Specifically, our method involves reducing the redundant transmission of match data and increasing the accuracy of cascade hashing matching to feature points. Our major contribution can be summarized as:

(1) We proposed a three-level buffer data scheduling (TLBDS) algorithm to achieve efficient scheduling of match data from disk to graphic memory. The application of the TLBDS algorithm enables the matching of large-volume oblique images on a mid-level computer, and the matching efficiency will not be affected by the changes in the number of images.
(2) The epipolar constraint is introduced in the cascade hashing to filter the initial candidate points, which increases the accuracy of the cascade hashing algorithm for matching feature points.
(3) By fusing TLBDS and cascade hashing with epipolar constraint, we design a highly efficient matching method for large-scale oblique images, where the accuracy and efficiency are tested across different scales, platforms, and environments.

The remainder of this paper is structured as follows. Section 2 reviews the related work. The proposed oblique image matching method is described in Section 3, and experiment results and

analyses are provided in Section 4. Some discussions about our method can be seen in Section 5, and Section 6 draws the conclusion together with our further work.

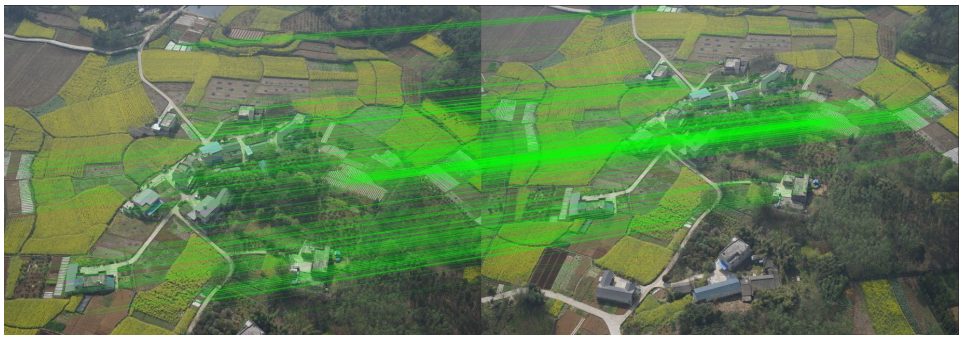
2. Related work

Image matching is a key step in SfM 3D reconstruction. Compared with other matching methods, the point feature-based matching technique has become a golden standard for aerial images, thanks to its invariance to translation, rotation, and scale, and tolerance to large deformations caused by changes in illumination and viewpoints (Jiang et al., 2020). So far, the mainstream matching methods used in oblique image matching pipelines are based on feature points (e.g. Moulon et al., 2016; MicMac, 2018; Agisoft, 2020; Pix4Dmapper, 2020). Oblique image matching based on point feature involves two steps: 1) feature point extraction and 2) feature point matching. The time complexity of feature extraction has a linear relationship with the number of images. Feature point matching refers to searching of the corresponding feature points on two overlapping images. Specifically, the cost of time in feature matching relates to the number of match pairs, the number of feature points on images, and the time complexity of the feature point search algorithm. To reduce the time cost of feature point extraction in SfM, Wu (2007) harnessed GPU-aided hardware acceleration to increase the efficiency of SIFT feature point extraction algorithm. Herbert et al. (2008) proposed a speeded-up robust feature (SURF) extraction algorithm. Rublee et al. (2012) changed the feature descriptor to binary code and reduced the dimension of the descriptor. He et al. (2018) enhanced the method of scale-space pyramids as well as descriptors, and utilized GPU acceleration to speed up the feature extraction procedure. Similarly, Li et al. (2019) employed a different optimization and parallel computing to implement a high-performance SIFT as HartSift.

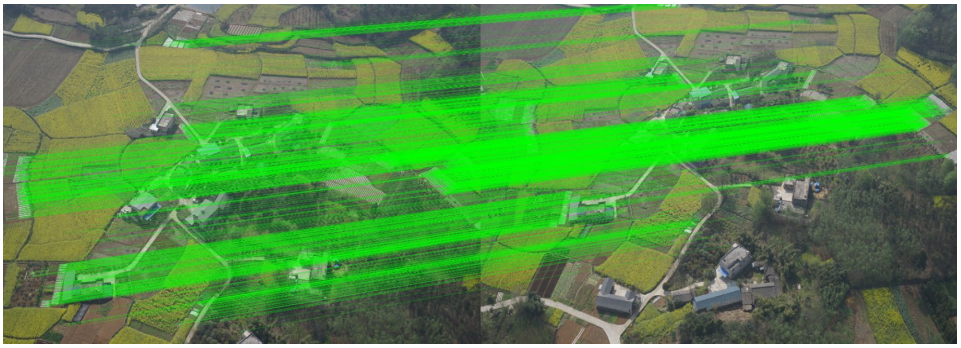
Compared with traditional aerial images, the oblique image has a high degree of overlap, and the number of oblique images collected by the drone at a test site is significantly high (Jiang et al., 2020). Feature point matching of the oblique image could involve huge computational complexity when simple exhaustive matching strategy is adopted. The selection of match pairs is the default strategy to accelerate image matching (Jiang and Jiang, 2017a). To remove invalid match pairs, Barazzetti et al. (2010) implemented match pair selection using spatial overlap based on the intersection of footprints derived from rough POS. Jiang and Jiang (2017a) use the maximum spanning tree (MST) algorithm after selecting the match pair to simplify the topological connection network (TCN) graph, so as to remove the redundant match pairs. Inspired by text retrieval, Agarwal et al. (2009) used a vocabulary tree-based image retrieval method to select match pairs from unordered images (images without geographical labels and definite time series). In Wu (2013), the visual similarity of the image is quantified by the number of feature points matching. After a small number of feature points are extracted from the down-sampled image, match pairs are selected according to the matching rate of the feature points. Similarly, Wang et al. (2019) quantified the visual similarity of images based on the number of feature matches. The difference is that the multi-random k-d trees algorithm is used to accelerate the approximate nearest neighbor (ANN) search of feature points.

Apart from match pair selection, the feature point matching, as the subsequent procedure after feature point extraction, received wide attention over the past two decades. The feature point matching takes the Euclidean distance or Hamming distance between the descriptor vectors as the similarity measurement and leverages the ANN algorithm to find the corresponding feature points. The k-d trees (Cover and Hart, 1967) are one of the most famous ANN algorithms. Although it is very effective in low dimensionality, its performance will decline rapidly for high-dimensional

space. Silpa-Anan and Hartley (2008) proposed a novel multi-random k-d trees algorithm based on the traditional k-d trees algorithm to accelerate the matching of SIFT descriptors. Since the well-perfomed nearest neighbor search on high-dimensional data, it has been widely used in the field of SfM 3D reconstruction (Moulon et al., 2016; MicMac, 2018). Muja and Lowe, (2009) performed a wide range of comparison amongst k-d trees, PCA-tree (Sproull, 1991) and RP-tree (Dasgupta et al., 2008), showing that the multi-random k-d trees are one of the most effective methods for matching high dimensional SIFT descriptors. Muja and Lowe (2014) proposed a new algorithm named the priority search k-means tree and released as an open-source library called fast library for approximate nearest neighbors (FLANN), which has been integrated into many open-source projects. Inspired by linear discriminant analysis hash (LDAHash) (Strecha et al., 2011), Cheng et al., (2014) proposed a cascade hashing structure to speed up SIFT feature point matching. Specifically, the cascade hashing is designed as a three-layer structure: hashing lookup, remapping, and ranking. Each layer leverages different similarity measurements and filtering strategies to reduce the sensitivity to noise. Further, Xu et al., (2017) implemented the cascade hashing algorithm on the GPU and optimized the implementation details, resulting in 20-times faster approach compared with original SIFT-GPU (Wu 2007). However, the accuracy of cascade hashing for feature point matching is affected by hash mapping, which is low compared with the multi-random k-d trees method as illustrated in Fig. 1.



(a) Cascade hashing matching, 910 matches



(b) Multi-random k-d trees matching, 2289 matches

Fig. 1 Comparison of the matching results of (a) cascade hashing, (b) multi-random k-d trees

3. Methods

A large-scale oblique image matching method is proposed by adopting efficient match data scheduling and feature point matching with low time complexity. The overall workflow of the method is illustrated in Fig. 2. Specifically, the cascade hashing algorithm is used to perform clustering and matching considering its low time complexity (Cheng et al., 2014). We further

introduce the epipolar constraint to increase the accuracy of cascade hashing matching. Subsequently, three-level buffer data scheduling algorithm (TLBDS) is proposed to reduce the redundant match data transmission and enhance the efficiency of oblique image matching. In doing so, the matching efficiency is increased and the trade-off between the size of the match data and the available storage of the computer is balanced effectively.

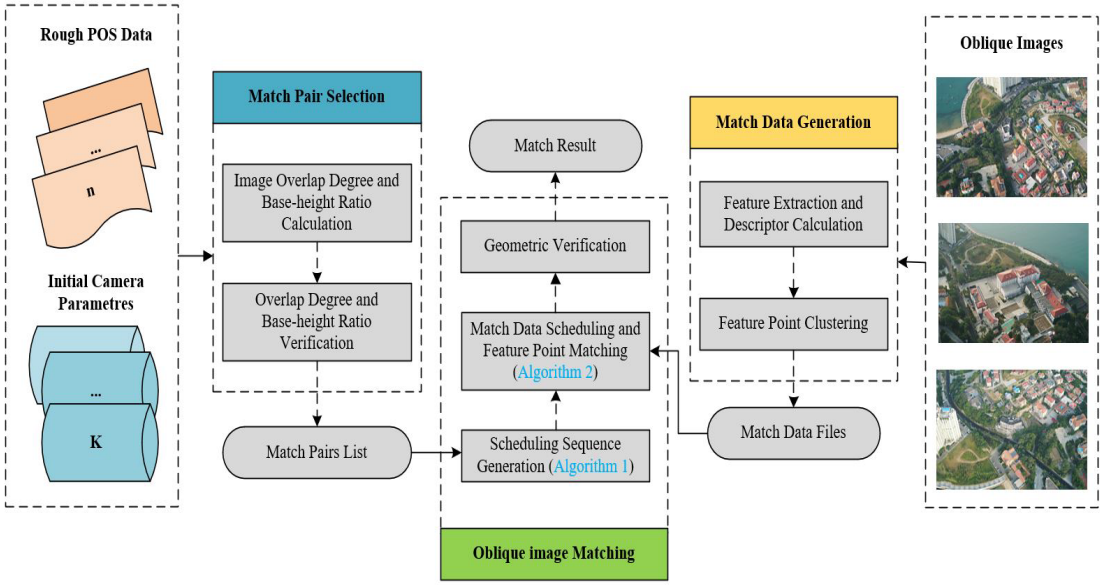


Fig. 2. The overall workflow of the proposed oblique image matching method

3.1 Feature point clustering and matching

We first use SIFT-GPU (Wu, 2007) to extract the feature points of the image and calculate the 128-dimensional descriptor of each feature point (Lowe, 2004). Thereafter, the locality sensitive hashing (LSH) algorithm (Charikar, 2002) is employed to calculate the bucket code of each feature point to cluster the feature points. Feature point matching is achieved by the cascade hashing algorithm with epipolar constraint that executes on the GPU card. The process of cascade hashing can be divided into three steps, i.e., feature point clustering, initial candidate point selection, and fine matching.

First, according to the L bucket indexes (IDs), the feature point clustering is mapping all feature points onto the hashing table of the corresponding image. The hashing table comprises L groups of buckets and each group contains 2^m buckets, where m is the bits of the bucket code. The initial candidate points selection is to index the L candidate buckets in the hashing table of the search image J based on the L bucket IDs of the query point q. The Hamming distance between q and feature points in each candidate bucket is then calculated. For each candidate bucket, the k feature points with the smallest Hamming distance are retained as initial candidate points (i.e., the coarse matching result). Finally, based on the Euclidean distance as the similarity measure and the Lowe ratio test (Lowe, 2004), accurate matching points are selected from the initial candidate points. The cascade hashing algorithm including clustering and initial selection is illustrated in Fig. 3.

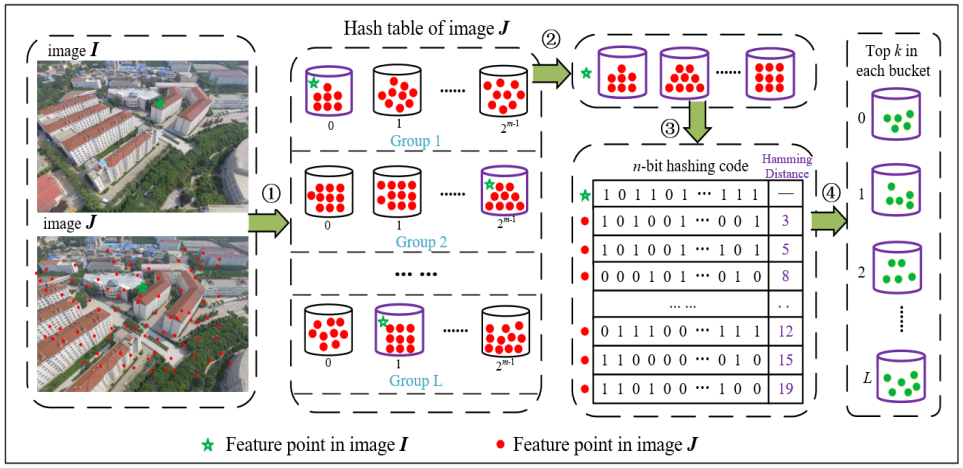


Fig. 3. The flowchart of the cascade hashing algorithm.

In feature point clustering, the cascade hashing employs the LSH algorithm (Charikar, 2002) to generate the hashing function, which is then utilized to calculate the hashing code and bucket ID of each feature point. Here, we use 1-dimensional Gaussian distribution $N(0,1)$ as the hashing function to generate N random vectors r_i with the length of 128, forming a random matrix Mat_{hash} with the size of $n \times 128$. The n-bit hashing code can be obtained by multiplying Mat_{hash} with 128-dimensional descriptor vector q. Taking the product of the element r_i of the i-th row of Mat_{hash} and the descriptor vector q, the hashing code is calculated as follows:

$$h_{r_i}(q) = \begin{cases} 1, & if\ r_i \cdot q > 0 \\ 0, & if\ r_i \cdot q \le 0 \end{cases} \tag{1}$$

Similarly, when calculating L m-bit bucket codes, random matrices Mat_{bucket} with m rows and 128 columns are first generated by 1-dimensional Gaussian distribution $N(0,1)$ and L m-bit bucket codes of the descriptor are obtained by the dot product between Mat_{bucket} and descriptor vector q thereafter. Finally, L bucket IDs can be generated by converting the bucket codes from binary to decimal.

To increase the accuracy of feature point matching based on cascade hashing, we introduce the rough POS data and initial camera intrinsic parameters to calculate the epipolar between the corresponding points in stereo images. The initial candidate points of the cascade hashing matching are then filtered to leverage the epipolar constraint, and finally, the Euclidean distance is used as the similarity measurement to match the feature points accurately.

3.2 Efficient three-level buffer data scheduling: from disk to graphics memory

To avoid the redundant transmission of match data and increase the utilization rate of computional resources, we proposed a three-level buffer data scheduling (TLBDS) algorithm composed of two parts, i.e., scheduling sequence generation and match data scheduling from disk to graphics memory. To be specific, the scheduling sequence generation is to generate an optimal match data reading order $List_{in}$ and match data clearing order $List_{out}$ based on the match pair information available memory and graphics memory sizes. The match data scheduling transfer the match data from the disk to the graphics memory based on the generated match data scheduling sequence and clear the matched data in the graphics memory. For more details on TLBDS, a few relevant terms are defined as:

Definition 1: Supposing that the set Set_{all} containing N images, the match data of M images stored in the graphics memory are called the inner sets Set_{inner} while the match data of $N - M$ images stored in the disk or memory are called the outer sets Set_{outer}. Before scheduling, all match data are stored on the disk.

Definition 2: Assuming that there are M match data in Set_{inner} and N match data in Set_{outer} whose corresponding images exist matching relationships with the image I, the total match pairs $Pairs_{all}$, the inner set match pairs $Pair_{inner}$, and the number of outer set match pairs $Pair_{outer}$ of the image I are $M + N$, M, and N, respectively. In the scheduling sequence generation procedure, the criterion for selecting the match data of the image I is $K_{select} = \omega * M - N$, where ω is the weighting factor and is set as 2.3 in this paper.

Definition 3: Memory indicates the Random Access Memory (RAM) of the computer, while the graphics memory denotes the memory of the graphics card. Based on the above definitions, the read-in sequence $List_{in}$ represent the sequence of match data that is waiting to be transferred from memory to graphics memory, while the read-out sequence $List_{out}$ signifies the sequence of matched data that is waiting to be cleared in graphics memory.

Three situations exist during the scheduling sequence generation: no match data in the graphics memory, part of the match data in the graphics memory but the available graphics memory has not been used up, and no available graphics memory. For the first case, the id of the image with the largest $Pairs_{all}$ in Set_{outer} will be pushed into the read-in sequence $List_{in}$. For the second case, the id of the image with the largest K_{select} in Set_{outer} will be pushed into the read-in sequence $List_{in}$. For the third case, the id of the image with the smallest $Pair_{outer}$ in the graphics memory will be pushed to the read-out sequence $List_{out}$. Repeat the above process, the optimal data scheduling sequence $List_{in}$ and $List_{out}$ can be gradually obtained. The details of the scheduling sequence generated by the TLBDS algorithm are summarized in Algorithm 1.

Algorithm 1. Scheduling sequence generation

Input:
Image name list:
$List_{img} = \{(imgName_1, 0), \dots, (imgName_N, N-1)\}_1^N$
Match pairs list:
$List_{pairs} = \{(imgId_1, imgId_2), \dots, (imgId_N, imgId_{N-1})\}_1^M$

Output:
Read-in sequence: $List_{in}$, read-out sequence: $List_{out}$

1 Initialize: inner sets $Set_{inner} \coloneqq empty()$, outer sets $Set_{outer} \coloneqq empty()$, $ratio_g = 0.7$
2 Gets the size of the available memory and available graphics memory, V_{memory}, $V_{graphic}$
3 Gets the size of a single match file $V_{matchFile}$
5 $count_{maxGraphic} \coloneqq ratio_g * V_{graphic} \,/\, V_{matchFile}$
6 **procedure** UPDATEIMAGEMATCHESNUMBER
7 Calculate $Pairs_{inner}$ and $Pairs_{outer}$ of each image
8 **end procedure**
9 **while** true **do**
10 **if** $Set_{inner}.size() \geq count_{maxGraphic}$ **then**
11 Search an image idi with the minimum $Pairs_{outer}$ in Set_{inner}
12 $List_{out}.add(idi)$
13 **procedure** UPDATEIMAGEMATCHES

Update $Pairs_{inner}$ and $Pairs_{outer}$ of images related to the image idi
end procedure
else if $Set_{inner}.size()$ *is* 0 **then**
Search an image idj with the largest $Pairs_{all}$ in $List_{img}$
$List_{in}.add(idj)$
Mark image idj
procedure UPDATEIMAGEMATCHES
Update $Pairs_{inner}$ and $Pairs_{outer}$ of images related to the image idi
end procedure
else
Search an image idk with the largest K_{select} in Set_{outer}
$List_{in}.add(idk)$
Mark image idk
procedure UPDATEIMAGEMATCHES
Update $Pairs_{inner}$ and $Pairs_{outer}$ of images related to the image idi
end procedure
end if
if all images have been marked **then**
break
end if
end while
return $List_{in}$, $List_{out}$

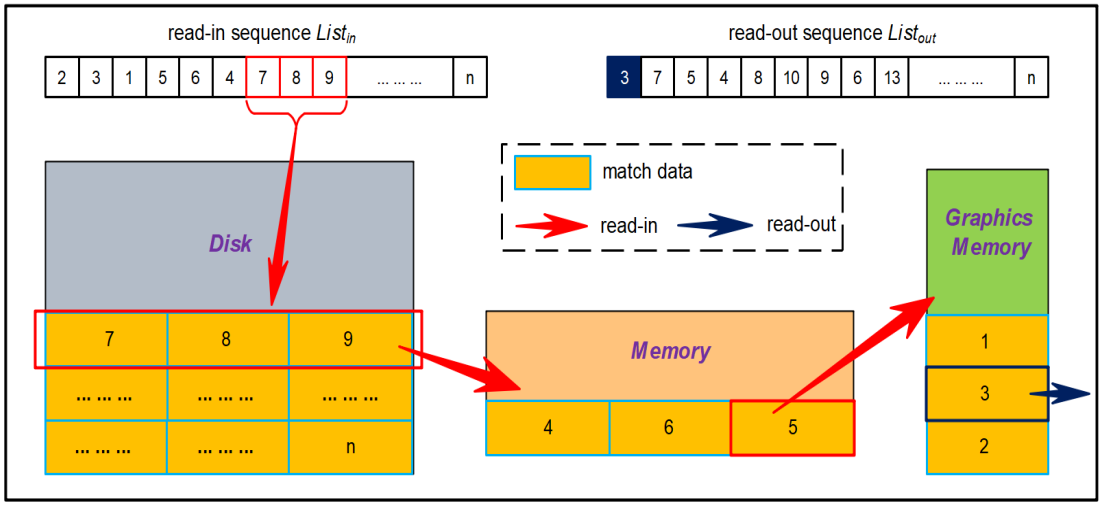


Fig. 4 Match data scheduling schematic diagram. Note that the read-in sequence determines the transferred order of match data both from disk to memory in batch and from memory to graphics memory one by one. The read-out sequence determines the cleared order of matched data in graphics memory.

After acquiring the scheduling sequence of match data transmission, the match data of each oblique image needs to be loaded into the graphics memory according to the read-in sequence $List_{in}$ and remove the matched data according to the read-out sequence $List_{out}$ (Fig. 4), thereby avoiding the underutilization of GPU card computing resources and reducing the redundant match data transmission. When there is free space in the graphics memory, the corresponding match data of the first id in $List_{in}$ is transmitted from the memory to the graphics memory. When the available graphics memory is insufficient, we reclaim the storage space occupied by the match data according

to $List_{out}$. Finally, the cascade hashing algorithm with epipolar constraint is implemented on the GPU card, and the TLBDS algorithm is used to perform match data scheduling, which can efficiently achieve the matching of oblique images. The details of oblique image matching based on match data scheduling and cascade hashing with epipolar constraint are summarized in Algorithm 2.

Algorithm 2. Oblique image matching based on match data scheduling and cascade hashing with epipolar constraint

Input:
Match data $D \coloneqq \{file_1, file_2, \dots, file_N\}$
Read-in sequence $List_{in}$, read-out sequence $List_{out}$,
Maximum number of files loaded in memory and graphics memory $count_{maxMem}$, $count_{maxGraphic}$
Match file byte size $V_{matchFile}$
Output:
Match result $R \coloneqq \{outFile_1, outFile_2, \dots, outFile_N\}$

Initialize:
1 $pos_{in} \coloneqq 0,\ pos_{out} \coloneqq 0, count_{in} \coloneqq 0, count_{mem} \coloneqq 0,$ $V_{maxG} \coloneqq count_{maxGraphic} * V_{matchFile},$ $V_{maxM} \coloneqq count_{maxMem} * V_{matchFile}$
2 Allocate byte size V_{maxG} in memory
3 Allocate byte size V_{maxG} in graphics memory
4 **procedure** LOADMATCHDATA
5 Import $count_{maxMem}$ match data files into memory, according to $List_{in}$
6 **end procedure**
7 **while** true **do**
8 **if** pos_{in} is $List_{in}.size() - 1$ **then**
9 **break**
10 **end if**
11 **if** $count_{in} \geq count_{maxGraphic}$ **then**
12 **procedure** READOUT
13 $id \coloneqq List_{out}[pos_{out}]$
14 Free $D[id]$ and take back byte size $V_{matchFile}$
15 **end procedure**
16 $count_{in} \coloneqq count_{in} - 1$
17 $pos_{out} \coloneqq pos_{out} + 1$
18 **else**
19 **if** $count_{mem} \geq count_{maxMem}$ **then**
20 **procedure** LOADMATCHDATA
21 Import $count_{maxMem}$ match data files into memory, according to $List_{in}$
22 **end procedure**
23 $count_{mem} \coloneqq 0$
24 **end if**
25 **procedure** READIN
26 $id \coloneqq List_{in}[pos_{in}]$
27 Load $D[id]$ into graphics memory
28 **end procedure**
29 **procedure** FEATUREPOINTMATCHING
30 Performing feature point matching on GPU according to cascade hashing with epipolar constraint.
31 **end procedure**
32 $count_{in} \coloneqq count_{in} + 1$

33	$pos_{in} \coloneqq pos_{in} + 1$
34	$count_{mem} \coloneqq count_{mem} + 1$
35	return R

To avoid memory fragmentation and extra time consumption caused by memory application-release, we stipulate that the match data file of each image has the same size. Therefore, the storage space is allocated in memory and graphics memory all at once in the initialization phase. Meanwhile, in the process of match data scheduling, the new match data fed into the graphics memory directly cover the storage space occupied by cleared data. Hence, the accelerated computing power of the GPU card can be fully exploited to perform oblique image matching. To be specific, the multi-thread and CUDA Stream (Nvidia, 2010) technology are adopted to concurrent execution of match data transmission and feature point matching, improving the efficiency of oblique image matching significantly.

4. Experiment and results

In the experiments, we use three datasets captured in different sites and scales to evaluate the performance of the proposed method. First, we test the effectiveness of the proposed TLBDS algorithm by conducting match data scheduling experiments. Then, the adjacency matrix is obtained from the match pair information to further validate the correctness of the scheduling sequence generated by our TLBDS algorithm. For assessing the impact on the matching performance of cascade hashing caused by the group number of buckets L, the number of bits of hash code n, the number of bits of bucket code m, and the number of candidates k in each candidate bucket, we perform comparative feature point matching experiments using cascade hashing algorithm under different parameter settings. Finally, we compare the performance of our method with four frequently-used pipelines, including OpenMVG-ANNL2 (Moulon et al., 2016), OpenMVG-CasHash (Moulon et al., 2016), Agisoft Metashape (Agisoft, 2020), and Pix4Dmapper (Pix4Dmapper, 2020). We evaluate the merits of each method from three aspects including efficiency, accuracy, and completeness. The proposed method is implemented using the C++ programming language and all experiments are executed on the Windows 10 platform with an Intel Core i7-7820X CPU (3.60 GHz) and a TITAN Xp graphics card (12GB).

4.1 Test sites and datasets

Dataset-1: The ground covers of the first test site are presented in Fig. 5a. Dataset-1 is obtained in the first test site by the oblique photography system with five SONY ILCE-5100 cameras. There are 1, 914 oblique images in Dataset-1 with the size of 7592 × 5304 pixels. The camera mounting angles in nadir and oblique directions are 0 °, 45°/- 45°, respectively. The altitude of this flight is 230m, the overlap degrees of images in the forward and side directions are 85% and 75%, and the average GSD is 2.85 cm/pixel.

Dataset-2: The ground covers of the second test site are demonstrated in Fig. 5b. Dataset-2 is acquired in the second test site by the conventional five-camera oblique photogrammetric system equipped with SONY ILCE-5100 cameras. The overlap degree of 3, 490 oblique images in Dataset-2 in the forward and side directions are 75% and 55%, respectively. The altitude of this flight is 140m, and the average GSD of Dataset-3 is 1.8 cm/pixel.

Dataset-3: The details of the third test site are illustrated in Fig. 5c. As a large-volume oblique image dataset, Dataset-3 is gathered in the third test site by a conventional five-camera oblique

photogrammetric system with SONY ILCE-7R cameras. This photogrammetric system is equipped with one nadir camera and four oblique cameras, with the four oblique cameras rotated by 45 ° with the inspection to the nadir camera. There are 14, 225 oblique images in the size of 7360 × 4921 pixels, and the altitude of UAV flight is 300m.

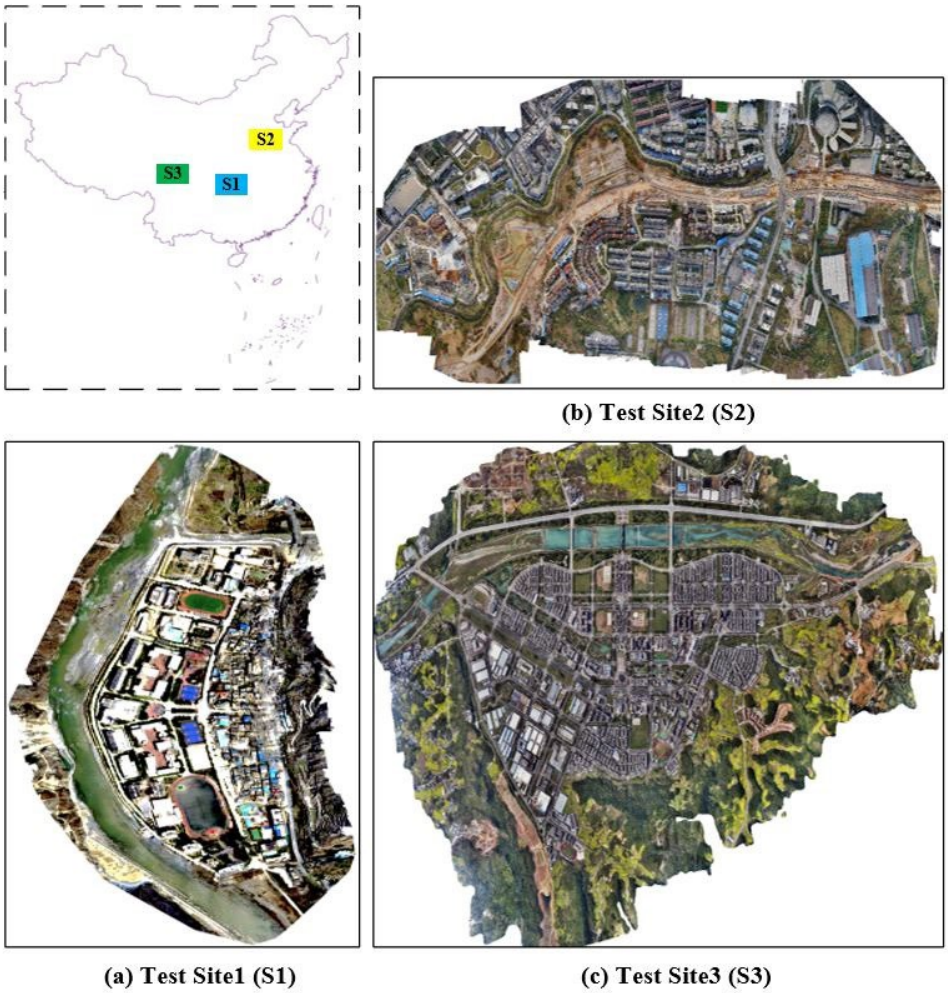


Fig. 5. The orthoimage of the three study sites to show the ground details

The detailed information for the flight configuration of the three datasets is presented in Table 1. During the data collection process, rough POS data of three datasets are measured using the GNSS/IMU device where the nominal accuracies in the horizontal and vertical directions are 4~5 cm.

Table 1

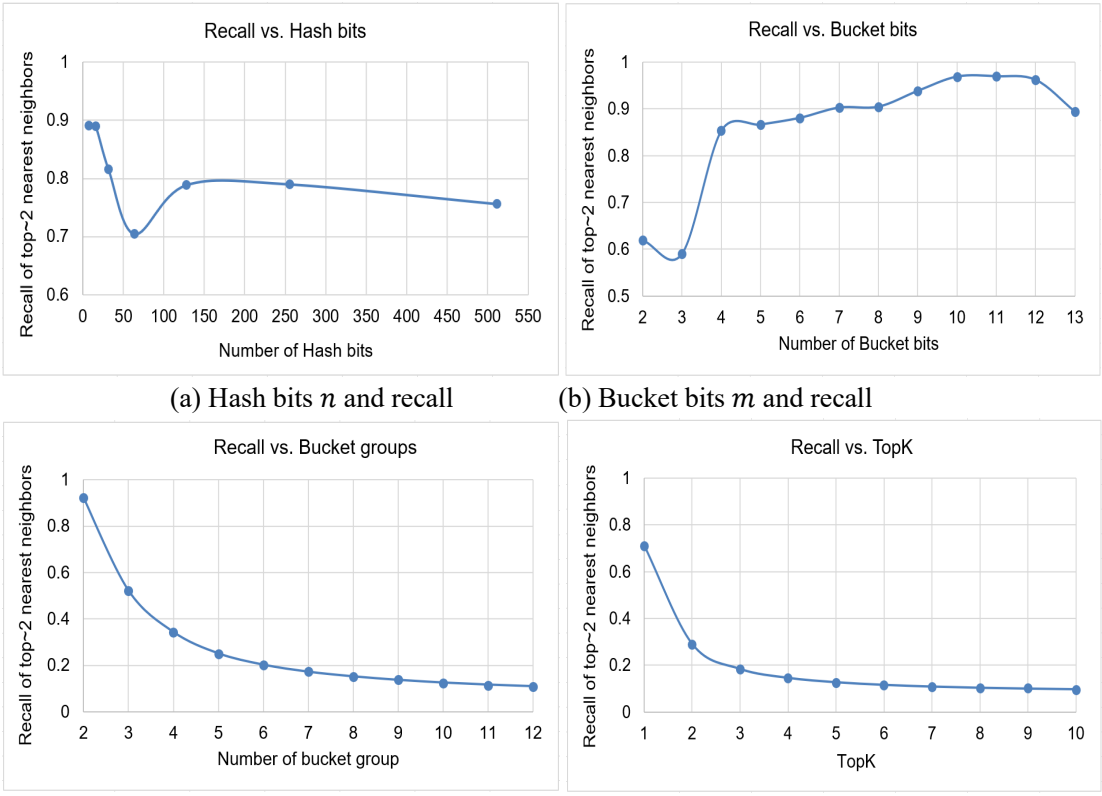
Detailed information for flight configuration of the three datasets.

Item name	Dataset-1	Dataset-2	Dataset-3
Flight height (m)	230	140	300
Forward / side overlap (%)	85 / 75	75 / 55	75 / 55
Camera mode	SONY ILCE-5100	SONY ILCE-5100	SONY ILCE-7R
Number of cameras	5	5	5
Sensor size (mm×mm)	23.4×15.6	23.4×15.6	35.9×23.9

Focal length (mm)	nadir: 20 oblique: 35	nadir: 20 oblique: 35	nadir: 35 oblique: 50
Camera mount angle	nadir: 0 oblique: 45 / -45	nadir: 0 oblique: 45 / -45	nadir: 0 oblique: 45 / -45
Number of images	1, 914	3, 490	14, 225
Image size (pixel×pixel)	6000×4000	6000×4000	7360×4921
GSD(cm/pixel)	2.85	1.8	3.1

4.2 Analysis of key parameters of cascade hashing

The impact of the hash code bits n, the bucket bits m, the bucket groups L and the number of candidate points k is analyzed by the performance of the cascade hashing using the controlled variable method. To be specific, we calculate and report the recall of the last two candidate points that meet the Lowes ratio test in Dataset-1. From Fig.6a, we can see that when hashing bits $n = 8$, cascade hashing has the highest recall for feature point matching, and when $n = 128$, the recall will also be close to 0.8. Considering the coupling between cascade hashing and SIFT, we set n as 128 in this paper. According to Fig. 6b, we set $m = 10$ in the subsequent experiments as the recall of cascade hashing matching feature points will obtain the largest figure. Similarly, we set $L = 2$ and $k = 1$ based on the experimental results shown in Fig. 6c and Fig. 6d.



(a) Hash bits n and recall (b) Bucket bits m and recall

(c) Bucket group L and recall (d) Top k nearest neighbors and recall

Fig. 6. The recall change, when hashing bits n, bucket bits m, bucket groups L, and k change, the nearest neighbor's two points are searched by cascade hashing to satisfy the Lowes ratio test.

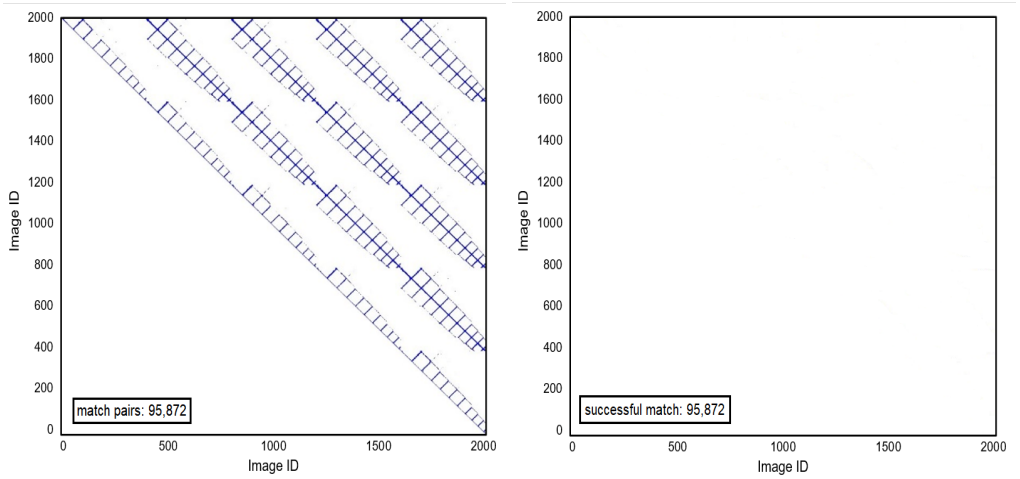
4.3 Verification of correctness and effectiveness of TLBDS algorithm

In this section, we investigated the correctness and effectiveness of the TLBDS algorithm. Firstly, the proposed TLBDS algorithm is employed to generate the data scheduling sequence, and then the

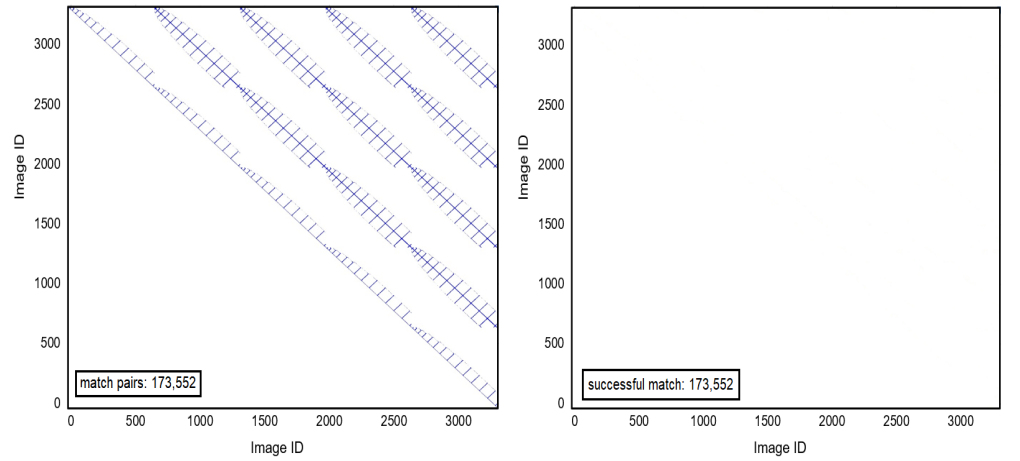
match data scheduling and matching are performed. The correctness of the TLBDS algorithm is verified by checking whether there are missing match pairs after match data scheduling. Finally, by comparing the number of match data transmissions before and after match data scheduling, the effectiveness of the TLBDS algorithm is tested.

4.3.1 Correctness

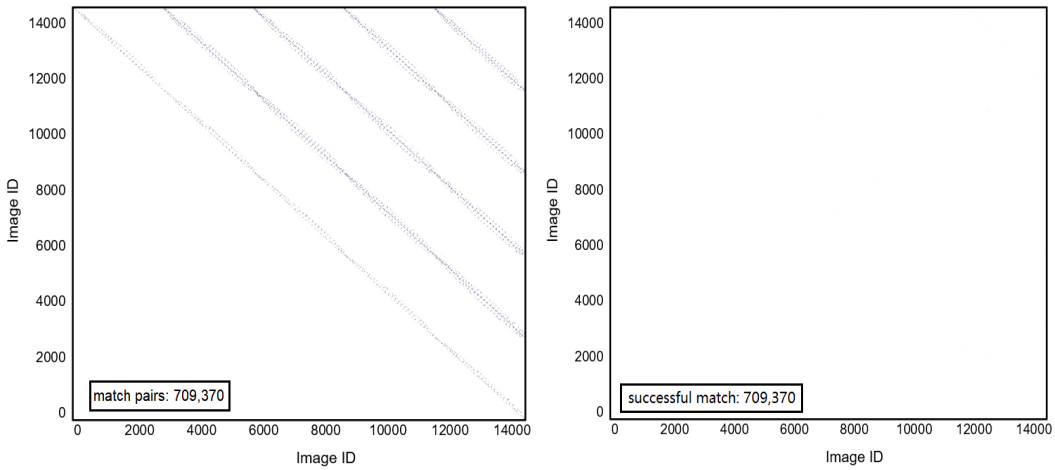
To verify the correctness of the scheduling sequence generated by the TLBDS algorithm, we perform match pair selection based on the pipeline proposed by (Barazzetti et al., 2010) and generate the adjacency matrix, as shown in Fig. 7. In the adjacency matrix graph, if the position (i, j) is blue, the image I_a with id i and image I_b with id j are a match pair. During the verification process, the value of position (i, j) in the adjacency matrix will be set as white after the match data of match pairs (i, j) is read into the graphics memory. If the scheduling sequence is correct, all the elements in the adjacency matrix will be white after match data scheduling according to the scheduling sequence. From Fig. 7, we can see that the adjacency matrices of the three datasets are all completely cleared after scheduling matching, demonstrating the reliability of the proposed TLBDS algorithm strongly.



(a) Adjacent matrix of Dataset-1 (b) Adjacency matrix after match data scheduling



(c) Adjacent matrix of Dataset-2 (d) Adjacency matrix after match data scheduling



(e) Adjacent matrix of Dataset-3 (f) Adjacency matrix after match data scheduling

Fig. 7. The adjacency matrix of match pairs of three datasets.

4.3.2 Effectiveness

For the TLBDS algorithm, the first step is to calculate an optimal match data scheduling sequence according to match pairs information, memory size, and graphics memory size. The second step is to load match data from disk to memory and then to graphics memory according to the data scheduling sequence. In this section, we compare the number of match data transmissions before and after the data schedule to verify the effectiveness of our TLBDS algorithm.

Table 2

The number of match data transmissions from the disk to memory and then to graphics memory.

Dataset	Before data scheduling	After data scheduling (TLBDS)	
		Storage size 1	Storage size 2
Dataset-1	95,872	5,168	2,895
Dataset-2	173,552	9,238	5,487
Dataset-3	709,370	39,820	24,189

We set two storage conditions: 1) the available memory size $V_{memory} = 1$Gb, available graphics memory size $V_{graphic} = 4$Gb; 2) $V_{memory} = 2$Gb, $V_{graphic} = 8$Gb, and count the number of match data transmissions from the disk to memory and then to graphics memory. Without data scheduling, the match data will be loaded by the order of the match pair which does not consider the reusability of the match data stored in graphics memory, leading the redundant data transmissions. As shown in Table 2, before data scheduling, there will be 95872, 173552, and 709370 transmissions in three datasets. In contrast, the proposed TLBDS first generates an optimal data scheduling sequence that rearranges the read-in and read-out order of match data. Specifically, if a match data A has a match relationship with the match data B and B will be read into the graphics memory later, then A will be scheduled at the relatively back position in the read-out sequence. The more matching relationships A has with the subsequent match data to be read, the later position in the read-out sequence A is. On the contrary, the match data with fewer match relationships with the subsequent match data to be read will be scheduled at the front of the read-out sequence that will be released early. As shown in Table 2, our TLBDS will save 10× to 20× transmissions and the advantages are more obvious under larger storage conditions.

4.4 Comparison of efficiency, accuracy, and completeness with state-of-the-art methods

We comprehensively evaluate our proposed method in efficiency, accuracy, and completeness with state-of-the-art methods. Three software packages, including OpenMVG, Agisoft Metashape, and Pix4Dmapper, are taken as comparative approaches. As a library for computer vision scientists and the multi-view geometry community, OpenMVG is designed to provide an SfM solution from feature extraction to sparse reconstruction. In the image feature point extraction stage, OpenMVG integrates two algorithms: SIFT and AKAZE (Alcantarilla, 2013). Besides, OpenMVG provides seven feature point matching methods, including the most commonly used multi-random k-d trees and cascade hashing algorithm. The Agisoft Metashape combines the most advanced image feature point matching algorithm and employs multi-core processing and GPU card parallel computing technology, providing a robust and efficient oblique image matching function. As the world's leading professional photogrammetric data processing software, Pix4Dmapper delivers a complete solution from image matching to orthoimage generation. Similarly, Pix4Dmapper also uses multi-core processing and GPU acceleration technology to improve the efficiency of digital image matching. In the pipelines of Agisoft Metashape and Pix4Dmapper, brute force pair-wise matching is replaced by rough POS data for match pair selection to reduce the time consumption.

For a fair comparison, we adopt the identical match pair selection results based on the method proposed by (Barazzetti et al., 2010) as the input of the pipelines of OpenMVG-ANNL2 OpenMVG-CasHash and ours. Besides, as mature professional software does not provide the application programming interface of intermediate processes, we fed the same rough POS data into Agisoft Metashape and Pix4Dmapper for match pair selection. The detailed information of the five pipelines for oblique image matching is shown in Table 3.

Table 3

The configure information of five oblique image matching pipelines

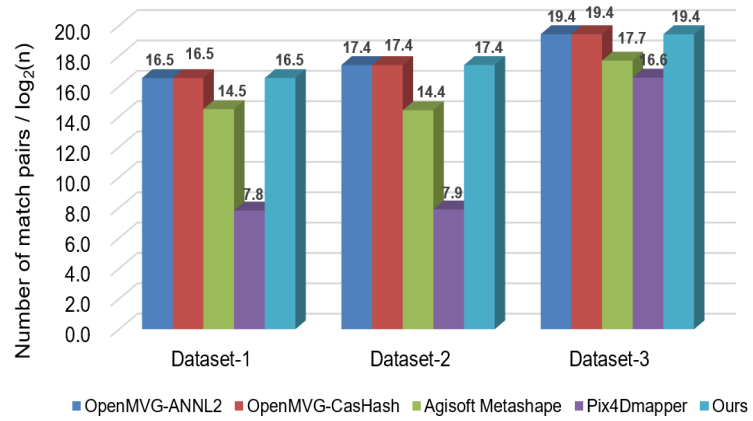
Pipeline	Match pairs selection method	Feature points matching algorithm	Match data scheduling	GPU card acceleration
OpenMVG-ANNL2	External input	Multi-random k-d trees	no	no
OpenMVG-CasHash	External input	Cascade hashing	no	no
Agisoft Metashape	Rough POS calculate	/	yes	yes
Pix4Dmapper	Rough POS calculate	/	yes	yes
Ours	Rough POS calculate	Improved Cascade hashing	yes	yes

Note: due to the confidentiality of commercial software, unknown information in the table is indicated by '/'.

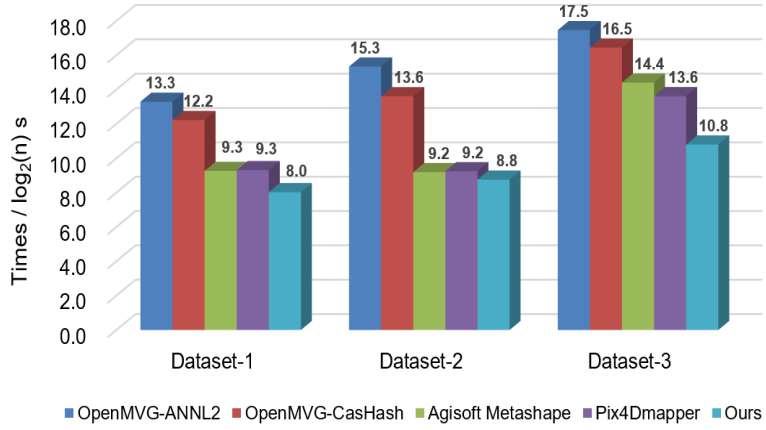
4.4.1 Efficiency

We compare the number of match pairs matched per second to evaluate the matching efficiency. To achieve the impartial comparison tests, we set all pipelines in Table 3 to use the same SIFT

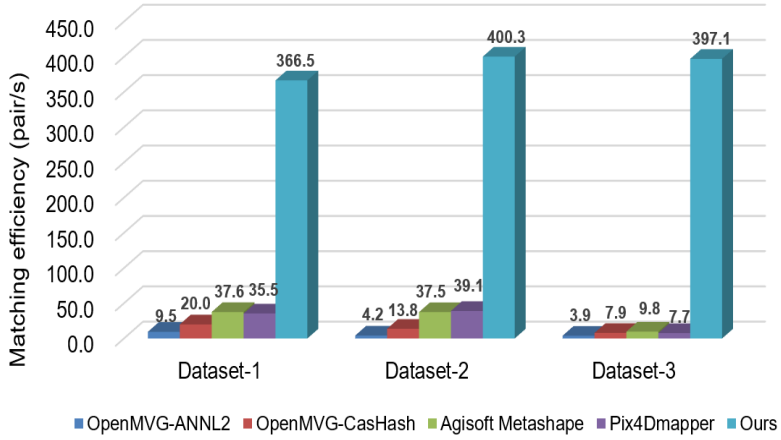
algorithm with default parameters for extracting feature points. The match pairs generated by Agisoft Metashape and Pix4Dmapper are based on the rough POS of oblique images. In our pipeline, we employ the method proposed by Barazzetti et al., (2010) to select match pairs, while the selected pairs are then fed into the OpenMVG-ANNL2 and OpenMVG-CasHash pipelines.



(a) Comparison of the number of match pairs



(b) Comparison of the time elapsed in the matching



(c) Comparison of matching efficiency.

Fig. 8. Comparison of (a) match pairs, (b) matching time, and (c) matching efficiency between different pipelines.

Figure 8a shows the number of match pairs in the five pipelines. Since OpenMVG-ANNL2, OpenMVG-CasHash, and our pipeline use the same match pairs, the number of match pairs in the above three pipelines is the same. Meanwhile, the Agisoft Metashape and Pix4Dmapper pipelines employ a more effective match pair selection method that reduces the number of redundant match pairs. Therefore, the number of match pairs is smaller than that of our pipeline. Figure 8b shows the time consumed by five pipelines matching three datasets, which can be seen that our pipeline consumes the least time.

The matching efficiency of each pipeline is reported in three datasets in Fig. 8c. The efficiency of the OpenMVG-ANNL2 pipeline and OpenMVG-CasHash pipeline is relatively low as they only utilize the multi-core CPU for matching. Specifically, the matching efficiency of OpenMVG-ANNL2 is 9.5 pairs/s for Dataset-1 and is 4.2 pairs/s for Dataset-2, while the figures for OpenMVG-CasHash are 20 pairs/s and 13.8 pairs/s, respectively. Besides, without match data scheduling, the efficiency (3.9 pairs/s for OpenMVG-ANNL2 and 7.9 pairs/s for OpenMVG-CasHash) becomes lower for the larger Dataset-3. Please note that the only difference is that the OpenMVG-ANNL2 pipeline uses the multiple-random k-d trees algorithm to matching feature points, while the OpenMVG-CasHash pipeline uses a cascade hashing algorithm. Through the above comparison, we can find that the efficiency of the cascade hashing algorithm is about 2~3 times faster than the multiple-random k-d trees algorithm.

We further compare the efficiency of the most advanced commercial software Agisoft Metashape and Pix4Dmapper. The matching efficiency of Agisoft Metashape for the three datasets is 37.69 pairs/s, 37.5 pairs/s, and 9.8 pairs/s, while the figures for the Pix4Dmapper pipeline are 35.5 pairs/s, 39.1 pairs/s, and 7.7 pairs/s, respectively. As the Agisoft Metashape and Pix4Dmapper pipelines generate fewer redundant match pairs, the match data of the search image corresponding to each query image requires less memory and graphics memory. Considering the size of memory and graphics memory, the match data can be transmitted into the memory and graphics memory within finite times, leading to a relatively stable matching efficiency. However, the number of data transmissions will increase with the growth of the match data volume, resulting in the underutilization of computing resources and the reduction of matching efficiency. It can be found that when matching Dataset-3, although accelerated by the GPU card, the efficiency of the Agisoft Metashape pipeline and Pix4Dmapper pipeline is only equivalent to the cascade hashing performing on a multi-core CPU.

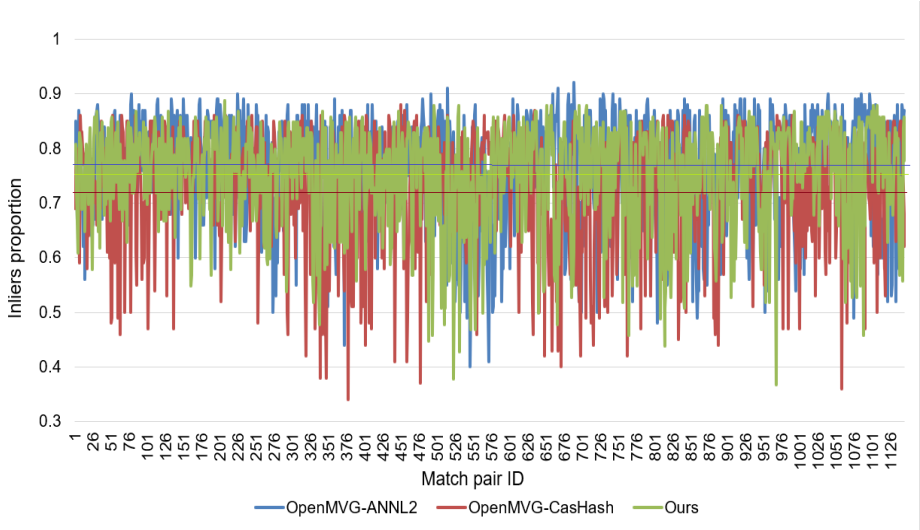
By contrast, the efficiency of our pipeline is unacted on the match data volume since the optimal match data scheduling sequence with minimum redundant data transmission is obtained by the proposed TLBDS algorithm. Specifically, the efficiency of our pipeline on three datasets is 366.5 pairs/s, 400.3 pairs/s, and 397.1 pairs/s, respectively. In other words, it only takes approximately 2.50 ~ 2.64ms for our pipeline to matching a match pair. Compared with the state-of-the-art Agisoft Metashape and Pix4Dmapper pipelines, the efficiency of oblique image matching has been significantly improved by 10 to 50 times. More importantly, the efficiency of our method is hardly affected by the change in the number of oblique images. As be seen from Fig. 8c, the efficiency gap between our pipeline and the Agisoft Metashape as well as Pix4Dmapper pipelines is observably widened for the large-volume Dataset-3.

4.4.2 Accuracy

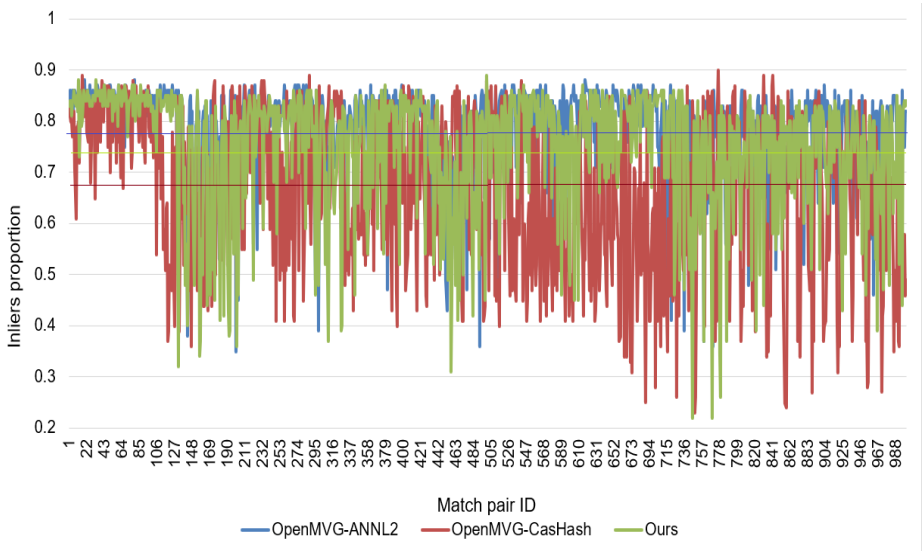
For accuracy analysis, we adopt inliers proportion and the reprojection error after bundle adjustment (BA) as the assessment criteria. By the positively correlated relationship, the inliers proportion can directly measure the matching accuracy of the feature points in the oblique image. The reprojection error can be used to comprehensively measure the accuracy of oblique image matching. First, considering that the intermediate results cannot be obtained from the two commercial pipelines (Agisoft Metashape and Pix4Dmapper), here we only compare the inliers proportion with the OpenMVG-ANNL2 and OpenMVG-CasHash pipelines. Specially, we use the RANSAC (Fischler et al., 1981) algorithm to estimate the fundamental matrix, then do geometric verification and to removing the outliers. In each dataset, we count the inliers proportions of 1, 000 match pairs. As shown in Fig. 9, we can see: (1) The inliers proportion of OpenMVG-ANNL2 pipeline is higher than that of OpenMVG-CasHash pipeline, demonstrating the multiple-random k-d trees algorithm has higher matching accuracy than the cascade hashing; (2) The inliers proportion of our pipeline is significantly higher than that of the OpenMVG-CasHash pipeline but is similar to the OpenMVG-ANNL2 pipeline, illustrating that the accuracy of proposed cascade hashing with epipolar constraint can obtain a competitive performance with the multiple-random k-d trees.

Second, for evaluating the accuracy between our pipeline and the Agisoft Metashape as well as Pix4Dmapper, we perform SfM reconstruction on the three datasets to obtain the accurate intrinsic and camera pose of each image and 3D points. Thereafter, each 3D point is reprojected onto its corresponding views image to calculate the average residual error of the image point. As shown in Fig. 10, our pipeline achieves competitive performance in three datasets.

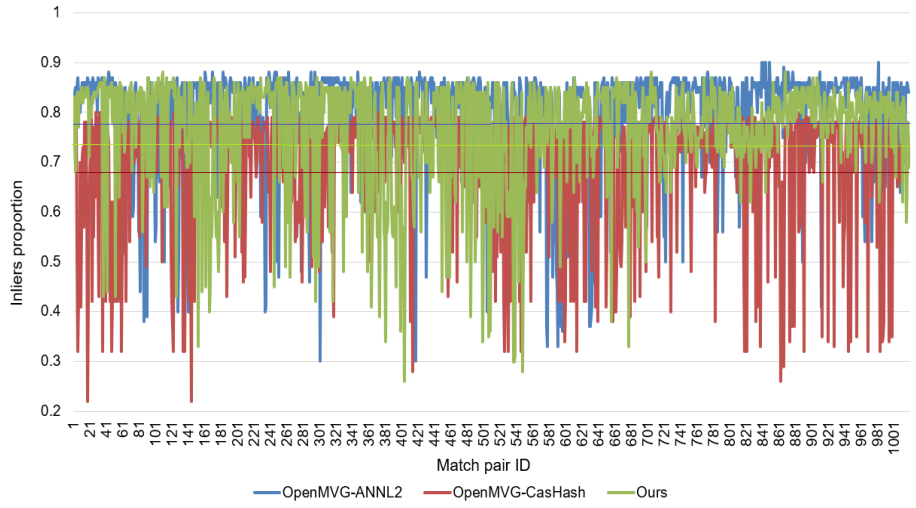
Comprehensive analysis of efficiency and accuracy, we can find that the proposed oblique image matching method not only robustly delivers reliable accuracy but also significantly improves the efficiency of large-scale oblique image matching.



(a) Dataset-1 inliers proportion. The average inliers proportion of OpenMVG-ANNL2, OpenMVG-CasHash, and our pipeline was 0.76, 0.73, 0.75.



(b) Dataset-2 inliers proportion. The average inliers proportion of OpenMVG-ANNL2, OpenMVG-CasHash, and our pipeline was 0.78, 0.68, 0.74.



(c) Dataset-3 inliers proportion. The average inliers proportion of OpenMVG-ANNL2, OpenMVG-CasHash, and our pipeline was 0.79, 0.69, 0.76.

Fig. 9. Comparison of the inliers proportion of the three pipelines matching. It can be seen that the matching accuracy of our pipeline is better than that of cascade hashing and close to that of the multiple-random k-d trees algorithm.

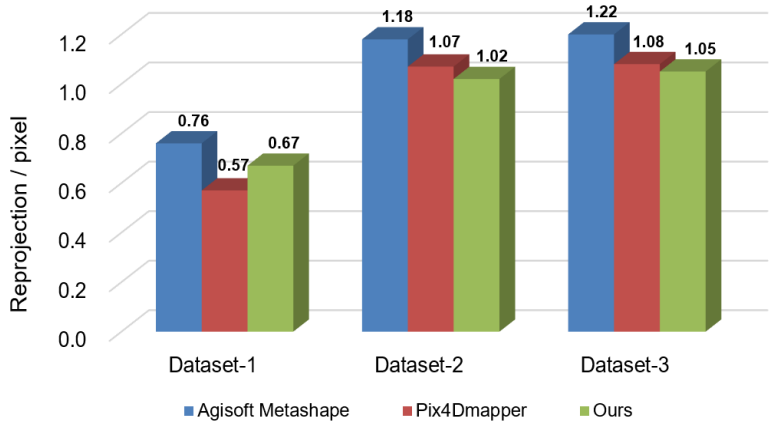


Fig. 10. Average reprojection error comparison. The reprojection error can indirectly prove the accuracy of oblique image matching, we can find that the accuracy of our method is the highest on Dataset-2 and Dataset-3.

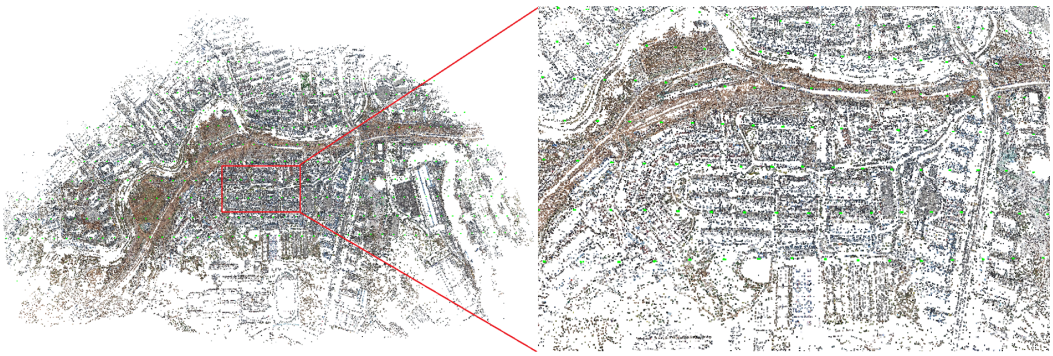
4.4.3 Completeness

For the completeness comparison, the BA experiment is employed to perform SfM reconstruction. Specifically, the incremental SfM strategy is adopted in the Agisoft Metashape, Pix4Dmapper, and our pipelines. In the BA stage, two images with a large enough intersection angle and a sufficient number of well-distributed features are chosen as the seed to operate the scene recovery. Thereafter, camera poses and 3D points are obtained by continuously adding a well-conditioned image to the reconstructed scene. In the process of image registration and triangulation, local BA is executed to reduce accumulated errors along newly added images. Thereafter, global BA is used to accurately calculate all camera poses and 3D points (Snavely. 2008). To evaluate the completeness, the number of connected images and 3D tie points is counted. Table 4 shows the information of the connected images and the generated 3D points, while the 3D point cloud generated by our pipeline and corresponding enlarged details are shown in Fig. 11. We can see that (1) the performance of our pipeline is similar to the Agisoft Metashape and Pix4Dmapper pipelines in terms of the number of connected images; (2) our reconstruction pipeline can obtain 3D point clouds with sufficient completeness and detailed information.

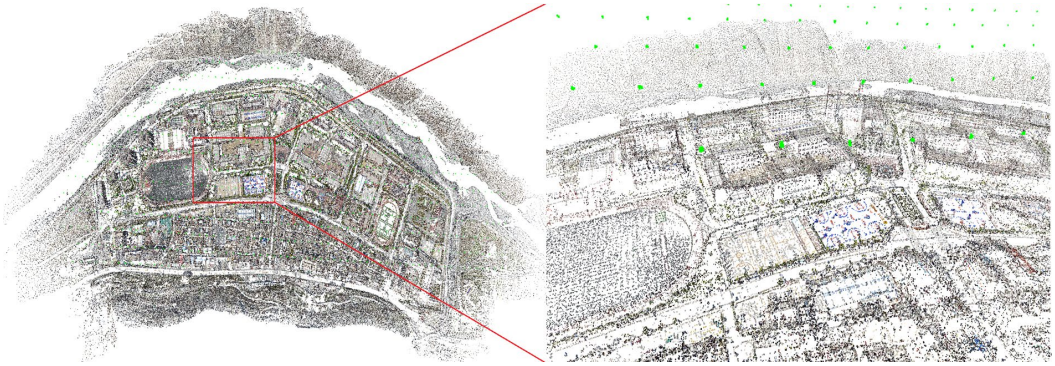
Table 4

The numbers of connected images and 3D points for completeness comparison

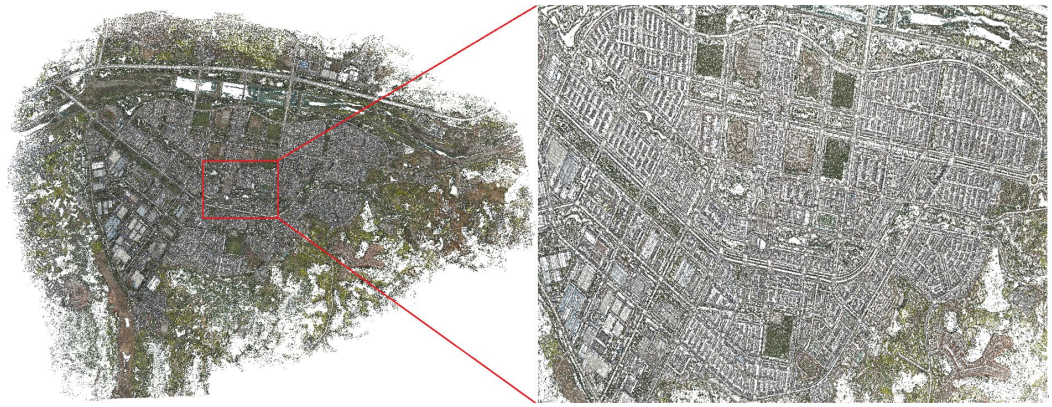
Dataset	Agisoft Metashape		Pix4Dmapper		Ours	
	Images	Points	Images	Points	Images	Points
1	1,914/1,914	293,490	1,914/1,914	102,310	1,914/1,914	259,614
2	3,489/3,490	473,759	3,490/3,490	423,158	3,490/3,490	693,083
3	14,212/14,225	4,631,650	14,214/14,225	1,904,063	14,216/14,225	3,073,284



(a) Dataset-1



(b) Dataset-2



(c) Dataset-3

Fig. 11. SfM re reconstruction result by our pipeline. The right image is the detail of the red rectangle in the left image.

5. Discussion

In this paper, we propose an efficient large-scale oblique image matching method with two major contributions. First, to improve the accuracy of the cascade hashing algorithm, the rough POS of the oblique image is leveraged to calculate the epipolar between the corresponding points. The epipolar is applied to the filter of the initial candidate points of the cascade hashing matching (as shown in Section 3.1). Second, an algorithm called TLBDS is designed and employed for the scheduling of match data from disk to graphics memory (as shown in Section 3.2), thereby reducing the redundant transmission of data. Combining the cascade hashing with epipolar constraint and the TLBDS algorithm, we have obtained an efficient method for oblique image matching, and the efficiency of our method is 10 ~ 50 times that of the state-of-the-art.

From the analysis of the inliers proportion (as shown in Section 4.4.2), the accuracy of our method is higher than the cascade hashing but slightly lower than the multiple-random k-d trees algorithm. Compared with cascade hashing, the reason why our method has higher accuracy is that we use epipolar constraints to filter the initial candidate points, thereby reducing the impact of hash mapping and similar textures on fine matching. Specifically, the hash mapping error will cause more noise in the candidate points obtained by using the Hamming distance as the similarity measure. Meanwhile, the descriptor similarity of the feature points in the texture-similar region is relatively high, during fine matching with the Euclidean distance between descriptors as the similarity measure which will cause false matches. Besides, compared with the most correct corresponding points, the influence of a small number of wrong points on the relative orientation results can be eliminated in the subsequent BA optimization. Therefore, the difference in accuracy between our method and the multiple-random k-d trees algorithm is negligible to the oblique image matching employed in SfM. This is also confirmed in Figure 10.

In terms of the efficiency of oblique image matching, there are two reasons for our method's remarkable efficiency. First, our method uses the cascade hashing with epipolar constraint to matching the feature points. From Section 4.4.1, it can be found that the time complexity of this algorithm is lower than that of the multiple-random k-d trees algorithm. Second, and the most important is, we designed an efficient TLBDS algorithm, which can significantly reduce the redundant transmission of match data(as shown in Section 4.3.2), and make full use of computing resources of the GPU card. Specifically, redundant match data transmission has a high time cost and will cause insufficient data supply, making computing resources in a state of waiting for match data.

6. Conclusions

In this paper, we propose an efficient large-scale oblique image matching method. Our contributions are two-fold: fast feature points matching based on cascade hashing with epipolar constraint and efficient three-level buffer match data scheduling. The extensive comparisons with state-of-art pipelines on oblique image matching and SfM reconstructions on three datasets prove the efficiency as well as the accuracy of our pipeline for large-scale oblique image matching. Importantly, the application of the TLBDS algorithm allows to matching large-volume oblique images on a mid-level computer, and the matching efficiency of our method will not be affected by changes in the number of images. However, the accuracy of rough POS data will affect the matching accuracy of our method. Therefore, reducing the dependence of our method on external conditions is the direction that needs to be explored in our further research.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank Jian Cheng who has made their algorithms of Cascade hashing free and open-source, which is helpful to the research in this paper. Meanwhile, heartfelt thanks to the anonymous reviewers and the editors, whose comments and advice improved the quality of the work. This work was supported by the National Natural Science Foundation of China (41671452).

References

Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R., 2009. Building rome in a day. In: 2009 IEEE 12th international conference on computer vision. IEEE, pp. 72–79.

Agisoft, 2020. Agisoft metashape homepage. http://www.agisoft.com, accessed: 2020-2-24.

Alcantarilla, P.F, 2013. Fast explicit diffusion for accelerated features in nonlinear scale spaces, in British Machine Vision Conference, Bristol, British Machine Vision Conference (BMVC).

Barazzetti, L., Remondino, F., Scaioni, M., Brumana, R., 2010. Fully automatic UAV image-based sensor orientation. International Archives of Photogrammetry. Remote Sensing and Spatial Information Sciences XXXVIII-1/C22, 25–31.

Charikar, M.S., 2002. Similarity estimation techniques from rounding algorithms. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (pp. 380-388).

Cheng, J., Leng, C., Wu, J., Cui, H., Lu, H.: Fast and accurate image matching with cascade hashing for 3D reconstruction. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE, June 2014

Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. IEEE Trans. Inf. Theory 13 (1), 21–27.

Dasgupta., Sanjoy., Freund., Yoav., 2008. Random projection trees and low dimensional manifolds. Proceedings of the Annual Acm Symposium on Theory of Computing, 124(4S), 537-546.

Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24 (6), 381–395.

Gruen, A., 2012. Development and status of image matching in photogrammetry. The Photogramm. Record 27 (137), 36–57.

Herbert, Bay., Andreas, Ess., et al., 2008. Speeded-up robust features (surf). Computer Vision & Image Understanding.

Jiang, S., Jiang, W., 2017a. Efficient structure from motion for oblique uav images based on maximal spanning tree expansion. ISPRS J. Photogramm. Remote Sens. 132, 140–161.

Jiang, S., Jiang, W., 2017b. On-board GNSS/IMU assisted feature extraction and matching for oblique UAV images. Remote Sensing, 9(8), 813.

Jiang, S., Jiang, C., Jiang, W., 2020. Efficient structure from motion for large-scale UAV images: A review and a comparison of SfM tools. ISPRS Journal of Photogrammetry and Remote Sensing, 167, 230-251.

Li, Z., Jia, H., Zhang, Y., Liu, S., Li, S., Wang, X., et al. 2019. Efficient parallel optimizations of a high-performance sift on gpus. Journal of Parallel and Distributed Computing, 124(FEB.), 78-91.

Lowe, D.G., 2004. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision. 60(2):91-110.

MicMac. Available online: http://www.tapenade.gamsau.archi.fr/TAPEnADe/Tools.html accessed on 19 March 2018).

Moulon, P., Monasse, P., Perrot, R., Marlet, R., 2016. OpenMVG: Open multiple view geometry. In International Workshop on Reproducible Research in Pattern Recognition (pp. 60-74). Springer, Cham.

Muja, M., Lowe, D.G., 2009. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. International Conference on Computer Vision Theory & Application Vissapp.

Muja, M., Lowe, D.G., 2014. Scalable nearest neighbor algorithms for high dimensional data. Pattern Analysis & Machine Intelligence IEEE Transactions on, 36(11), 2227-2240.

Nvidia, C.: Programming Guide (2010).

Pix4Dmapper, 2020. Pix4dmapper homepage. https://www.pix4d.com, accessed: 2020- 2-24.

Rublee, E., Rabaud, V., Konolige, K., Bradski, G., 2012. ORB: An efficient alternative to SIFT or SURF. International Conference on Computer Vision. IEEE.

Rupnik, E., Nex, F., Remondino, F., 2013. Automatic orientation of large blocks of oblique images. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XL-1/W1, 299–304.

Schönberger, J.L., Fraundorfer, F., Frahm, J.-M., 2014. Structure-from-motion for UAV image sequence analysis with photogrammetric applications. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XL-3 (3), 305–312.

Silpa-Anan, C., Hartley, R., 2008. Optimised K-d treess for fast image descriptor matching. IEEE Conference on Computer Vision & Pattern Recognition, pp. 1-8.

Snavely, N., Seitz, S. M., Szeliski, R., 2008. Skeletal graphs for efficient structure from motion. In 2008 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-8). IEEE.

Sproull, R.F., 1991. Refinements to nearest-neighbor searching in k-dimensional trees. Algorithmica, 6(1-6), 579-589.

Strecha, C., Bronstein, A.M., Bronstein, M.M., Fua, P., 2011. Ldahash: improved matching with smaller descriptors. IEEE Transactions on Pattern Analysis & Machine Intelligence, 34(1).

Wang, X., Rottensteiner, F., Heipke, C., 2019. Structure from motion for ordered and unordered image sets based on random kd forests and global pose estimation. ISPRS J. Photogramm. Remote Sens. 147, 19–41.

Westoby, M.J., Brasington, J., Glasser, N.F., Hambrey, M.J., Reynolds, J.M., 2012. 'Structure-from-Motion'photogrammetry: a low-cost, effective tool for geoscience applications. Geomorphology 179, 300–314.

Wu, C., 2007. SiftGPU: A GPU Implementation of David Lowe's Scale Invariant Feature Transform (SIFT). <https://github.com/pitzer/SiftGPU> (accessed: 2017- 06-19).

Wu, C., 2013. Towards Linear-Time Incremental Structure from Motion. In: International Conference on 3d Vision, pp. 127–134.

Xu, T., Sun, K., Tao, W., 2017. GPU Accelerated Image Matching with Cascade hashing. CCF Chinese Conference on Computer Vision. Springer, Singapore.

Xu, Z., Wu, L., Gerke, M., Wang, R., Yang, H., 2016. Skeletal camera network embedded structure-from-motion for 3D scene reconstruction from UAV images. ISPRS J. Photogramm. Remote Sens. 121, 113–127.

Yang, H.C., Zhang, S.B., Wang, Y.B., 2012. Robust and precise registration of oblique images based on scale-invariant feature transformation algorithm. IEEE Geosci. Remote Sens. Lett. 9 (4), 783–787.