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

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

[a]*School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China*
[b]*Intelligent Control & Smart Energy (ICSE) Research Group, School of Engineering, University of Warwick, Coventry CV4 7AL, UK*
[c]*Lancaster Environment Centre, Lancaster University, Lancaster LA1 4YQ, UK*
[d]*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∼2.64ms, 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

*Shunyi Zheng
**Rui Li
URL: qiyuanzhang@whu.edu.cn (Qiyuan Zhang), syzheng@whu.edu.cn (Shunyi Zheng), c.zhang9@lancaster.ac.uk (Ce Zhang), wangxiqi@whu.edu.cn (Xiqi Wang), rui.li.4@warwick.ac.uk (Rui Li)

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 [1]. Precise camera poses are mandatory to utilize oblique images in 3D reconstruction, which can be obtained by the airborne global navigation satellite system and the inertial measurement unit (GNSS/IMU) system. However, limited by measurement accuracy, the image positioning and orientation system (POS) data obtained through the airborne GNSS/IM and installation angle cannot meet the requirement of direct image positioning and orientation accuracy. In the computer vision community, the structure from motion (SFM) can solve camera poses and 3D points automatically from overlapped images with high accuracy [2] [3]. In SFM technology, a key step is image matching, which occupies approximately half of the computational cost [4].

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 [5]. According to matching primitives, image matching technologies can be divided into three categories: point matching, line matching, and region matching [4]. Since the invention of scale-invariant feature transform (SIFT) [6], point matching methods have become the mainstream for oblique image pipelines thanks to their robustness to changes in scale, illumination and viewpoint [7]. Nevertheless, limited by the high time complexity of feature extraction and feature point matching, the time complexity of image matching methods based on point matching is relatively high [4]. There are two directions to increase the efficiency of point matching, including the improvement of the algorithm and the utilization of graphic processing unit (GPU) parallel computing. The former is related to design lightweight algorithms such as speeded-up robust features (SURF) [8], oriented fast and rotated brief (ORB) [9], while the latter aims to utilize the parallel computing capability of GPUs [10] [11].

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 open multiple view geometry (OpenMVG) [12], MicMac [13], Agisoft Metashape [14], and Pix4Dmapper [15]. However, 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 [16]. Although the utilization of GPU cards can accelerate the matching process of oblique images, it will create a significant amount of redundant transmission of match data between the disk and the graphics memory, leading to high time cost and insufficient use of computational resources. In addition to data scheduling, the feature point matching is also a bottleneck that limits the efficiency of the oblique image matching. Although cascade hashing algorithm has the highest efficiency for feature point matching, we find that the matching robustness of the cascade hashing algorithm is relatively low compared with the multi-random k-d trees algorithm [17].

When matching oblique images of large scenes, it is generally necessary to match hundreds of thousands of images. The limited memory and graphics memory capacity make the redundant transmissions of data unavoidable, leading to an unnecessarily high input and output (I/O) cost. After research, we found that the efficiency of large-scene oblique image matching is mainly affected by I/O cost and feature point matching efficiency. Xu et al. [10] proposed a data exchange strategy to solve the problem of high I/O cost. However, their data exchange strategy does not consider the adjacency relationship between the images, resulting in a situation where the I/O cost is too high and computing resources cannot be fully utilized when matching image sets with sparse matching relationships. In this paper, we propose 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 higher efficiency. Specifically, our method involves reducing the redundant transmission of match data and improves the robustness of cascade hashing matching to feature points. Our major contributions can be summarized as:

(1) We propose a three-level buffer data scheduling (TLBDS) algorithm that considers the adjacency between images to achieve efficient scheduling of match data from disk to graphics memory and improve the utilization of computing resources. The application of the TLBDS algorithm enables the matching of large-volume oblique images to be conducted on a mid-level computer, while the matching efficiency will not be affected by the changes in the number of images.

(2) We introduce the epipolar constraint calculated according to the rough POS into the cascade hashing (the cascade hashing with the epipolar constraint is called E-CasHash), thereby improving the robustness of the cascade hashing and reducing the cost of matching calculation.

(3) By fusing TLBDS and cascade hashing with epipolar constraint, we design a highly efficient matching method for large-scale oblique images, whose 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

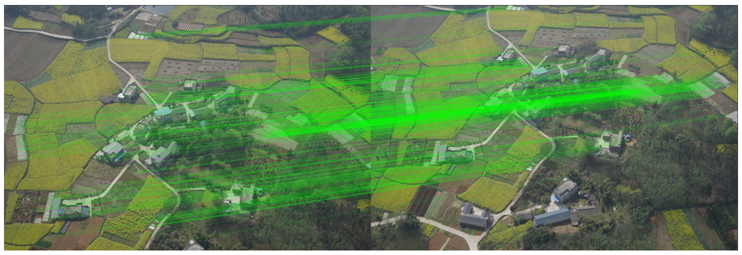
Image matching is a key step in SFM 3D reconstruction. Compared with other matching methods, the point feature-based matching technique has become a golden standard for aerial images, thanks to its invariance to translation, rotation, scale and tolerance to large deformations caused by changes in illumination and viewpoints [16]. So far, the mainstream matching methods used in oblique image matching pipelines are based on feature points [12]. 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 the corresponding feature points on two overlapping images. Therefore, 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.

For feature point extraction, Wu [18] harnessed GPU-aided hardware acceleration to increase the efficiency of SIFT feature point extraction algorithm to reduce the time cost of SFM. Herbert et al. [8] proposed a SURF (speeded-up robust feature) extraction algorithm. Rublee et al. [9] changed the feature descriptor to binary code and reduced the dimension of the descriptor. Li et al. [11] employed a different optimization and parallel computing to implement a high-performance SIFT as HartSift.

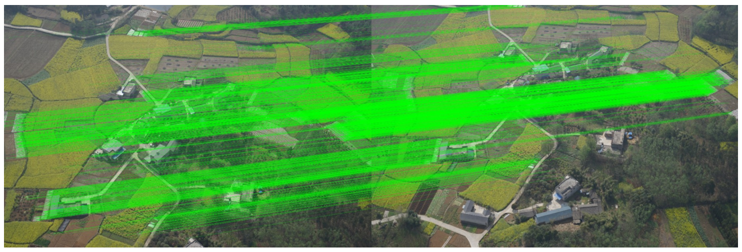
When it comes to feature point matching, the utilization of a simple exhaustive matching strategy will involve huge computational complexity. The reason is that oblique images have a higher degree of overlap with a large tilted angle and the number of oblique images collected by the drone at a test site is significantly large compared with traditional aerial images [16]. To address this issue, the selection of match pairs is the default strategy to accelerate image matching [19]. For example, Barazzetti et al. [20] implemented match pair selection using spatial overlap based on the intersection of footprints derived from rough POS to remove invalid match pairs. Jiang et al. [19] used the maximum spanning tree (MST) algorithm after selecting the matching pair to simplify the topological connection network (TCN) graph, so as to remove the redundant match pairs. Inspired by text retrieval, Agarwal et al. [21] proposed a vocabulary tree-based image retrieval method to select match pairs from unordered images (images without geographical labels and definite time series). In [22], the visual similarity of the image was quantified by the number of feature points matching. To be specific, after a small number of feature points were extracted from the down-sampled image, match pairs were then selected according to the matching rate of the feature points. Similarly, Wang et al. [23] quantified the visual similarity of images based on the number of feature matches, while the difference was that the multi-random k-d trees algorithm was used to accelerate the approximate nearest neighbor (ANN) search of feature points.

Apart from match pair selection, the feature point matching algorithm, as the subsequent procedure after feature point extraction, also has received wide attention over the past two decades. Normally, the feature point matching algorithm 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. For ANN algorithms, one of the most famous and frequently-used methods is the k-d trees [24]. For example, Silpa-Anan et al. [17] proposed a novel multi-random k-d trees algorithm based on the traditional k-d trees algorithm to accelerate the matching of SIFT descriptors. Muja and Lowe [25] performed a wide range of comparisons amongst k-d trees, PCA-tree [26], and RP-tree [27], showing that the multi-random k-d trees were one of the most effective methods for matching high dimensional SIFT descriptors. Muja and Lowe [28] proposed a new algorithm named the priority search k-means tree and released it as an open-source library called fast library for approximate nearest neighbors (FLANN) [29], which has been integrated into many open-source projects. As a well-performing nearest neighbor search on high-dimensional data, the k-d trees algorithm has also been widely used in the field of SFM 3D reconstruction [15]. Although being very effective in low dimensionality, the k-d trees' performance will decline rapidly for high-dimensional space.

Apart from the k-d trees algorithm, there are also many other types of research to speed up SIFT feature point matching. Inspired by linear discriminant analysis hash (LDAHash) [30], Cheng et al. [4] proposed a cascade hashing structure. Specifically, the cascade hashing was designed as a three-layer structure: hashing lookup, remapping, and ranking. Each layer leveraged different similarity measurements and filtering strategies to reduce the sensitivity to noise. Further, Xu et al. [10] implemented the cascade hashing algorithm on the GPU and optimized the implementation details, resulting in a 20-times faster approach compared with the original SIFT-GPU [18]. Xu et al. [31] calculated the fundamental matrix after matching a small number of feature points by brute force at a high scale and then introduced the epipolar constraint into the cascade hashing to reduce the cost of matching. However, as far as we know, in the region of repeated texture or repeated structure, the robustness of the cascade hashing algorithm for feature point matching is lower than that of the multi-random k-d trees algorithm as illustrated in Fig.1.



(a) Cascade hashing matching, 910 matches



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

Figure 1: Comparison of the matching results of (a) cascade hashing and (b) multi-random k-d trees. We can see that under the premise of extracting the same number of feature points, the robustness of the cascade hashing for feature point matching in the repeated texture region is lower than that of the multi-random k-d trees algorithm.

3. METHODS

A large-scale oblique image matching method is proposed by adopting efficient match data scheduling and feature point matching based on cascade hashing with epipolar constraint. The overall workflow of our method is illustrated in Fig.2. Although the cascade hashing algorithm is less robust in repeated texture regions, it has significant efficiency advantages [4]. Inspired by Xu et al. [31], after calculating the fundamental matrix using rough POS and camera intrinsic parameters, we introduce the epipolar constraint to improve the robustness of cascade hashing matching (this method is called E-CasHash), the details about the original cascade hashing can be found in [4]. Subsequently, a three-level buffer data scheduling algorithm (this method is called TLBDS) that considers the adjacency relationship between images is proposed to reduce the redundant match data transmission and enhance the utilization of computing resources. By doing so, the proposed method can match oblique images of a large scale on a medium configuration computer.

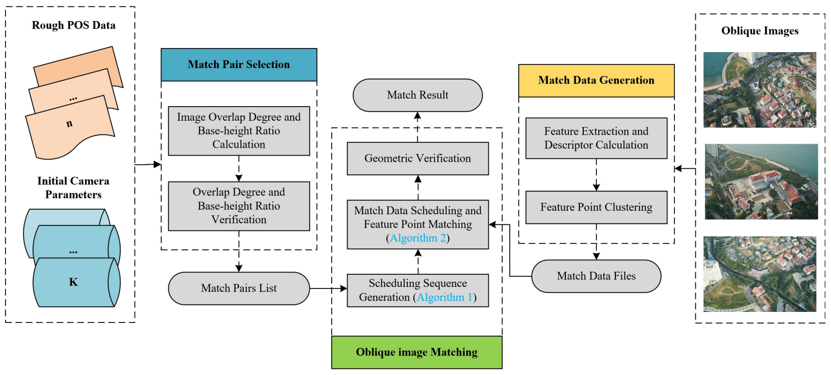

Rough POS Data
...
n
Initial Camera Parameters
...
K
Match Pair Selection
Image Overlap Degree and Base-height Ratio Calculation
Overlap Degree and Base-height Ratio Verification
Match Pairs List
Match Result
Geometric Verification
Match Data Scheduling and Feature Point Matching (Algorithm 2)
Scheduling Sequence Generation (Algorithm 1)
Oblique image Matching
Match Data Generation
Feature Extraction and Descriptor Calculation
Feature Point Clustering
Match Data Files
Oblique Images

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

3.1. Feature Point Clustering and Matching

We first harness SIFT-GPU [18] to extract the feature points of the image and calculate the 128-dimensional descriptor of each feature point [6]. Thereafter, the locality-sensitive hashing (LSH) algorithm [32] is employed to perform hashing remapping, which calculates the hashing code and bucket code of each feature point. According to the bucket code, we can cluster all feature points of each image into multiple hashing tables.

The specific process of feature point clustering consists of three steps. First, we choose the 1-dimensional Gaussian distribution $N(0,1)$ as the hash function to generate a set of L random matrices $Mat_b = \{M_1, M_2, ..., M_L\}$. The matrix

$M_i (i = 1, 2, \dots, L)$ contains m rows and 128 columns. Then, the dot product is conducted by each row $r_j (j = 1, 2, ..., m)$ of the random matrix M_i and the descriptor vector v according to formula (1) to obtain an m-bit bucket code (m is set as 10 in our method). After the dot product of L random matrices and descriptor vectors v, we obtain L m-bit bucket codes of descriptors. In order to facilitate the subsequent processing, we convert the L m-bit bucket codes from binary to decimal to obtain the L bucket ids of the descriptor. Finally, according to L bucket ids, the feature points can be clustered into L hash tables.

$$h_{r_j}(v) = \begin{cases} 1, & if \;\; r_j \cdot v > 0 \\ 0, & if \;\; r_j \cdot v \leq 0 \end{cases} \tag{1}$$

Feature point matching is achieved by E-CasHash executed on the GPU card. The E-CasHash algorithm is an improvement on the basis of the cascade hash [4] algorithm, while the process of E-CasHash can be divided into the following steps:

(1) Hashing lookup. After the feature point clustering, each image can establish L hash tables, and each hash table is composed of 2^m buckets [4]. Especially, the corresponding feature point between the images has the same bucket id. Therefore, according to the L bucket ids of the query point q in the image I_i, L candidate buckets can be queried in the hash table of the image I_j. There can be several points in the bucket is the candidate's corresponding point of q.

(2) Epipolar parameters calculation. According to the rough POS of oblique images, the relative pose parameters between images can be obtained. Besides, the initial camera intrinsic parameters are calibrated in advance. Hence, give the intrinsic matrics K_i, K_j, rotation matrics R_i, R_j, and translation vectors t_i, t_j, the fundamental matrix F_{ij} between stereo pair I_i and I_j can be computed as:

$$F_{ij} = {K_i}^{-T} R_i [{R_j}^T t_j - {R_i}^T t_i]_\times {R_j}^T {K_j}^{-1} \tag{2}$$

The epipolar parameters $l_q = \begin{bmatrix} a_q & b_q & c_q \end{bmatrix}^T$ for the point $q = \begin{bmatrix} u_q & v_q & 1 \end{bmatrix}^T$ in the image I_i corresponding to the image I_j can be computed as:

$$l_q = F_{ij} q \tag{3}$$

(3) Candidate feature point filter based on epipolar constraint. In E-CasHash, the epipolar constraint is described as the candidate point p and epipolar line l_q satisfying $dist(p, l_q) < d$. Therefore, after calculating the distance between the candidate point to the epipolar l_q according to equation (4), the candidate points in 1) can be filtered. In this paper, the value of d is the 95% confidence value of the 1-degree-of-freedom chi-square test where $d = 3.84$. The employment of epipolar constraints can not only reduce the matching cost but also enhance the robustness of cascade hashing matching.

$$dist(p, l_q) = \frac{|a_q u_p + b_q v_p + c_q|}{\sqrt{{a_q}^2 + {b_q}^2}} \tag{4}$$

(4) Top k nearest neighbors selection. After the epipolar filtering, the top k ($k = 2$ in our method) neighboring points can be selected by calculating the Hamming distance between hashing code of the query point q and the candidate point p.

(5) Euclidean distance calculation and Lowe ratio test [6]. By calculating the Euclidean distance between the SIFT descriptors of the top k neighboring points and query point q, the candidate point q' that satisfies the Lowe ratio test is the corresponding point of q. Since the Lowe ratio test is performed only among a subset of features close to the epipolar line, the E-CasHash algorithm can retain more correspondences on repetitive texture.

3.2. Efficient Three-level Buffer Data Scheduling: From Disk to Graphics Memory

During image matching, each image will form a match pair with one or more other images, so the match data of each image will inevitably be transferred from disk to graphics memory multiple times. To reduce the number of data transmissions, we propose a three-level buffer data scheduling (TLBDS) algorithm. The purpose of the data scheduling algorithm is to transfer the match data from the disk to the graphics memory in the optimal order without missing matches, thereby minimizing the total number of data transfers. Here, the data scheduling is similar to the postman delivery to deliver all letters and travel the least distance. Reducing redundant transmission can not only reduce the time cost of image matching but also make full use of the computing resources of the computer.

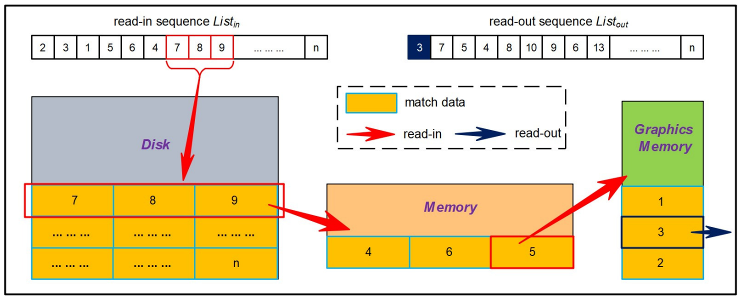
The TLBDS algorithm is composed of two parts, i.e., scheduling sequence generation and match data scheduling from disk to graphics memory. To be specific, the scheduling sequence generation is to generate an optimal match data reading order $List_{in}$ and match data clearing order $List_{out}$ based on the match pair information, the capacity of memory as well as graphics memory. The match data scheduling transfers the match data from the disk to the graphics memory based on the generated match data scheduling sequence and clears the matched data in the graphics memory. For more details on TLBDS, a few relevant terms are defined:

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. It should be noted that in this article each image has a separate match data file.

Definition 2: Assuming that there are M match data in Set_{inner} and K match data in Set_{outer} whose corresponding images exist matching relationships with the image I, the total match pairs $Pair_{all}$, the inner set match pairs $Pair_{inner}$, and the number of outer set match pairs $Pair_{outer}$ of the image I are $M+K$, M, and K, respectively. In the scheduling sequence generation procedure, the criterion for selecting the match data of the image I is $K_{select} = \omega * M - K$, 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 $List_{in}$ represents 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 available graphics memory is empty. 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 $Pairs_{outer}$ in the graphics memory will be pushed to the read-out sequence $List_{out}$. Repeating 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.



read-in sequence $List_{in}$
2 3 1 5 6 4 7 8 9 ... ... n
read-out sequence $List_{out}$
3 7 5 4 8 10 9 6 13 ... ... n
match data
read-in
read-out
Disk
7 8 9
Memory
4 6 5
Graphics Memory
1 3 2
n

Figure 3: 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 $List_{in}$ and $List_{out}$, the match data of each oblique image need to be loaded into the graphics memory according to the read-in sequence $List_{in}$ and matched data need be removed according to the read-out sequence $List_{out}$ (Fig.3), thereby reducing the redundant match data

Algorithm 1 Scheduling Sequence Generation

Input: an image id list $List_{img}$, a match pair list $List_{pairs}$, maximum number of files loaded in graphics memory N_{graMax}

Output: a read-in list $List_{in}$, a read-out list $List_{out}$

```
1:  $Set_{inner} \leftarrow \{\}$, $Set_{outer} \leftarrow \{List_{img}\}$
2:  calculate $Pairs_{inner}$ and $Pairs_{outer}$ of each image
3:  while $Set_{outer}$ is not empty do
4:      $idx \leftarrow 0$
5:      if $Set_{inner}.size() \geq N_{graMax}$ then
6:          search an image $idx$ with the minimum $Pairs_{outer}$ in $Set_{inner}$
7:          $List_{out}.add(idx)$
8:          $Set_{inner}.remove(idx)$
9:      else if $Set_{inner}$ is empty then
10:         search an image $idx$ with the largest $Pairs_{all}$ in $Set_{outer}$
11:         $List_{in}.add(idx)$
12:         $Set_{inner}.add(idx)$
13:         $Set_{outer}.remove(idx)$
14:     else
15:         search an image $idx$ with the largest $K_{select}$ in $Set_{outer}$
16:         $List_{in}.add(idx)$
17:         $Set_{inner}.add(idx)$
18:         $Set_{outer}.remove(idx)$
19:     end if
20:     update $Pairs_{inner}$ and $Pairs_{outer}$ of images related to the image $idx$
        according to $List_{pairs}$
21: end while
22: return $List_{in}$, $List_{out}$
```

transmission and enhancing the utilization of computing resources. When there exists free space in the graphics memory, the corresponding match data of the first id in $List_{in}$ will be transmitted from the memory to the graphics memory. When the available graphics memory is insufficient, the storage space occupied by the match data will be reclaimed according to $List_{out}$, in order to store new data in the vacated storage space. In the process of the TLBDS algorithm to perform match data scheduling, the E-CasHash algorithm is executed on the GPU card for feature point matching between oblique images. The details of oblique image matching based on TLBDS and the E-CasHash algorithm are summarized in Algorithm 2.

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 the computer unified device architecture (CUDA) stream [33] 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

The extensive experiments are conducted on three datasets captured in different sites and scales to evaluate the performance of the proposed method. First, the adjacency matrix is obtained from the match pair information to further validate the correctness of the scheduling sequence generated by our TLBDS algorithm. Then, we test the effectiveness of the proposed TLBDS algorithm by conducting match data scheduling experiments. Finally, we compare the performance of our method with four frequently-used pipelines, including OpenMVG-

Algorithm 2 Oblique Image Matching Based on TLBDS and E-CasHash

Input: match data set Set_{data}, read-in list $List_{in}$, read-out list $List_{out}$, maximum number of files loaded in memory and graphics memory N_{memMax}, N_{graMax}

Output: match result set Set_{result}

```
1:  $pos_{in} \leftarrow 0$, $pos_{outer} \leftarrow 0$, $N_{in} \leftarrow 0$, $N_{mem} \leftarrow 0$
2:  import $N_{memMax}$ match data files into memory according to $List_{in}$
3:  while $true$ do
4:    if $pos_{in}$ is $List_{in}.size() - 1$ then
5:      break
6:    end if
      // read-out process
7:    if $N_{in} \geq N_{graMax}$ then
8:      $id \leftarrow List_{out}[pos_{out}]$
9:      free $Set_{data}[id]$
10:     $N_{in} \leftarrow N_{in} - 1$
11:     $pos_{out} \leftarrow pos_{outer} + 1$
12:   else
13:     if $N_{mem} \geq N_{memMax}$ then
14:       import $N_{memMax}$ match data files into memory, according to $List_{in}$
15:       $N_{mem} \leftarrow 0$
16:     end if
        // read-in process
17:     $id \leftarrow List_{in}[pos_{in}]$
18:     load $Set_{data}[id]$ into graphics memory
19:     performing feature point matching on GPU according to E-CasHash, and get match result $r_{id}$
20:     $Set_{result}.add(r_{id})$
21:     $N_{in} \leftarrow N_{in} + 1$
22:     $pos_{in} \leftarrow pos_{in} + 1$
23:     $N_{mem} \leftarrow N_{mem} + 1$
24:   end if
25: end while
26: return $Set_{result}$
```

ANNL2 [12], OpenMVG-CasHash [12], COLMAP [34], Agisoft Metashape [14], and Pix4Dmapper [15]. 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), while the memory capacity is 48GB.

4.1. Test Sites and Datasets

Dataset-1: The ground covers of the first test site are presented in Fig.4(a). 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 a size of 6000 × 4000 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.

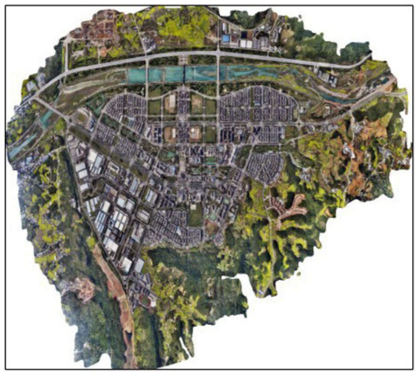
Dataset-2: The ground covers of the second test site are demonstrated in Fig.4(b). Dataset-2 is acquired in the second test site by the conventional five-camera oblique photogrammetric system equipped with SONY ILCE-5100 cameras. The overlap degree of 3,490 oblique images in Dataset-2 in the forward 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.4(c). 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, while 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.

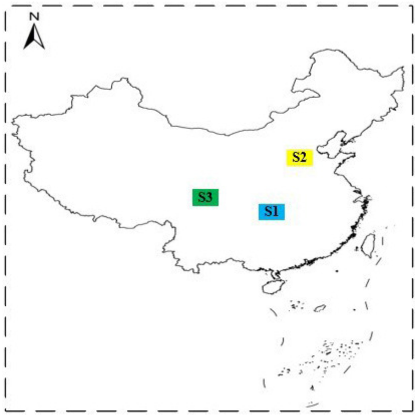
The detailed information for the flight configuration of the three datasets is presented in Table 1. During the data collection process, rough POS data of

(a) Test site1(S1)

(b) Test site2(S2)



(c) Test site3(S3)



(d) Geographical location of the test sites

Figure 4: The orthoimage of the three study sites to show the ground details

three datasets are measured using the GNSS/IMU device where the nominal accuracies in the horizontal and vertical directions are 4∼5cm.

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)	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)	6000 * 4000	6000 * 4000	7360 * 4912
GSD(cm/pixel)	2.85	1.8	3.1

4.2. Verification of Correctness and Effectiveness of TLBDS Algorithm

In this section, we investigate 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.

4.2.1. 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. [20] and generate the adjacency matrix, as shown in Fig. 5. 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) are 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.5, we can see that the adjacency matrices of the three datasets are all completely cleared after scheduling matching, which demonstrates the correctness of the scheduling sequence generated by the TLBDS algorithm.

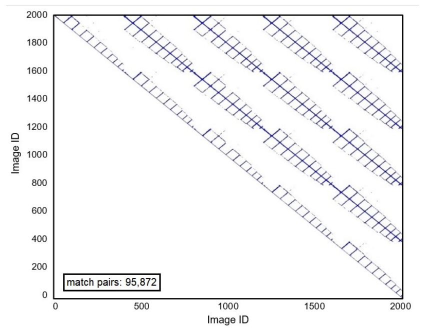
4.2.2. Effectiveness

In this section, we compare the matching efficiency of oblique images with and without data schedule to verify the effectiveness of the TLBDS algorithm. The only difference between oblique image matching with or without data scheduling is that the matching without data scheduling transfers the match data from the disk to the graphics memory according to the order of match pairs, while the matching with data scheduling is based on the data scheduling sequence generated by the TLBDS algorithm. In the above two cases, the image feature point matching is achieved by the E-CasHash algorithm on the GPU card. The details of matching the three datasets in the two cases are shown in Table 2.

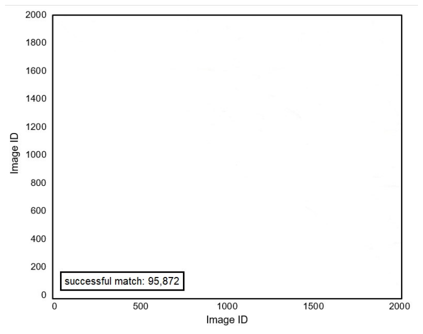
Table 2: Comparison of oblique image matching efficiency with or without data scheduling

Dataset	Match pairs	No scheduling		Scheduling		Speedup
		Time(s)	Speed (pairs/s)	Time(s)	Speed (pairs/s)	
Dataset-1	95,872	1,656.40	57.88	261.6	366.5	6.33×
Dataset-2	173,552	5,267.50	32.9	433.6	400.3	12.14×
Dataset-3	709,370	48,808.60	14.5	1,786.50	397.1	27.32×

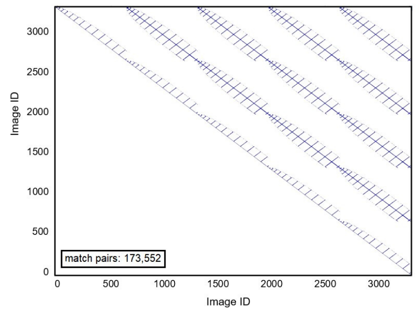
As can be seen from Table 2, for the relatively small dataset Dataset-2, the utilization of our TLBDS algorithm can yield about tenfold accelerations,



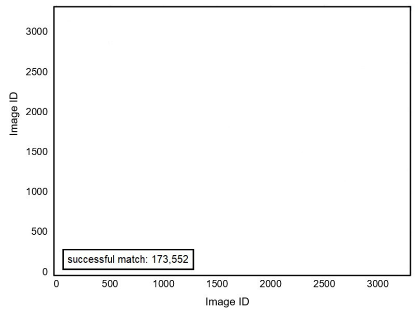
(a) Adjacency matrix of Dataset-1



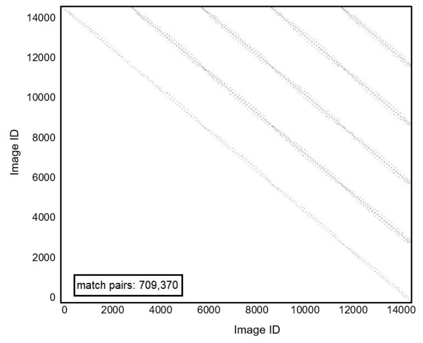
(b) Adjacency matrix after match data scheduling



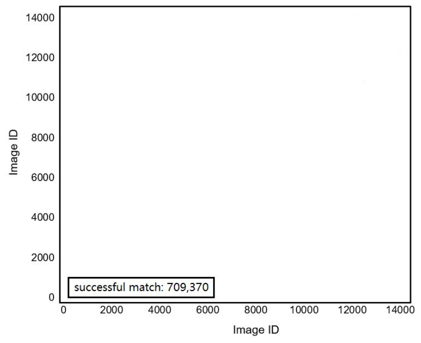
(c) Adjacency matrix of Dataset-2



(d) Adjacency matrix after match data scheduling



(e) Adjacency matrix of Dataset-3



(f) Adjacency matrix after match data scheduling

Figure 5: The adjacency matrix of match pairs of three datasets.

while the increase of efficiency for Dataset-3 is more than 27 times. Besides, the efficiency gap notably enlarges with the increase of images. The reason for this gap is that the reusability of the match data stored in graphics memory is not considered by those methods without scheduling, leading to the redundant transmissions of match data. Redundant match data transmission has a high I/O cost and will affect the full utilization of computing resources. In contrast, the proposed TLBDS first generates an optimal data scheduling sequence that rearranges the read-in and read-out order of match data, which reduces the number of data transmissions and increases the efficiency of oblique image matching.

4.3. 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. Four software packages, including OpenMVG, COLMAP, 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. Besides, OpenMVG provides seven feature point matching methods, including the most commonly used multi-random k-d trees and cascade hashing algorithm. COLMAP is a general-purpose SFM and multi-view stereo (MVS) pipeline with a graphical and command-line interface. In the feature point matching stage, COLMAP uses the algorithm in the open source library FLANN to perform the nearest neighbor search of feature points, that is, COLMAP also uses the multi-random k-d trees algorithm to search for corresponding feature points in candidate images. 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. 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
COLMAP	Internal generate	Multi-random k-d trees	no	no
Agisoft Metashape	Rough POS calculate	/	yes	yes
Pix4Dmapper	Rough POS calculate	/	yes	yes
Ours	Rough POS calculate	E-CasHash	yes	yes

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

4.3.1. 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 utilize the SIFT algorithm with default parameters for extracting feature points, and we adopt the identical match pair selection results based on the method proposed by Barazzetti et al [20] as the input of the pipelines of OpenMVG-ANNL2, OpenMVG-CasHash and Ours. When testing COLMAP, we count the number of generated match pairs and the corresponding time consumption to match. Besides, as the mature professional software does not provide the application programming interface of intermediate processes, we feed the same rough POS data into Agisoft Metashape and Pix4Dmapper for match pair selection.

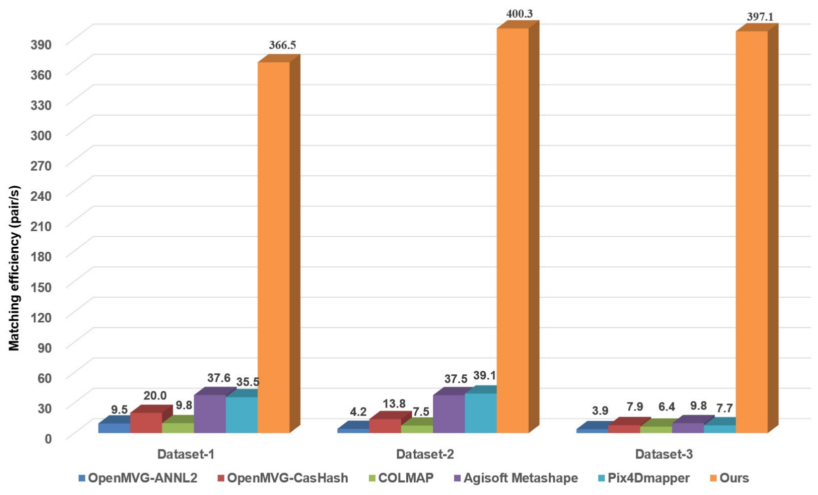


Figure 6: Comparison of matching efficiency between different pipelines.

The matching efficiency of each pipeline is reported in three datasets in Fig.6. The efficiency of the OpenMVG-ANNL2 pipeline, OpenMVG-CasHash pipeline and COLMAP 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. Although OpenMVG-ANNL2 and COLMAP pipeline uses the same match algorithm, the engineering optimization of COLMAP is more efficient.

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 left image (reference image) corresponding to each query image requires less memory and graphics memory. When matching smaller datasets, 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 redundant 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 independent 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 2.5∼2.64ms for our pipeline to matching a match pair. Compared with the state-of-the-art Agisoft Metashape and Pix4Dmapper pipelines, our pipeline can increase the matching efficiency of oblique images by 10 to 50 times. Generally, the I/O cost will be high when the matching relationship between images is relatively sparse. But, from the adjacency matrix in Fig.5, the proposed oblique image matching method still has the highest efficiency for three datasets with sparse relationships.

4.3.2. Accuracy

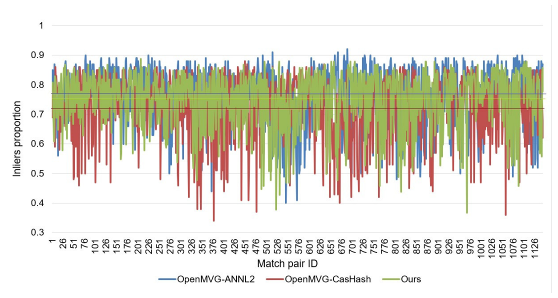
For accuracy analysis, we adopt inliers proportion as the assessment criteria. By the positively correlated relationship, the inliers proportion can directly measure the matching accuracy of the feature points in the oblique image. It should be pointed out that COLMAP and OpenMVG-ANNL2 use the same matching algorithm. So, the accuracy of OpenMVG-ANNL2, OpenMVG-CasHash and Ours are compared. Specially, we use the random sample consensus (RANSAC) [35] algorithm to estimate the fundamental matrix, then conduct geometric verification to remove the outliers. In each dataset, we count the inliers proportions of 1000 match pairs. As shown in Fig.7, we can see: (1) the inliers proportion of the OpenMVG-ANNL2 pipeline is higher than that of the 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 E-CasHash can obtain a competitive performance with the multiple-random k-d trees.

From 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.

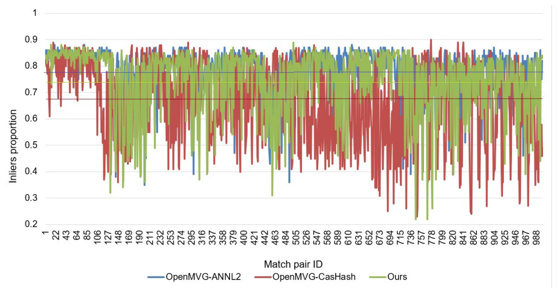
4.3.3. Completeness

For the completeness comparison, the BA (Bundle Adjustment) 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. The SFM reconstruction of our pipeline is executed on the G3D software developed by us, which adopts the method proposed in this paper for oblique image matching.

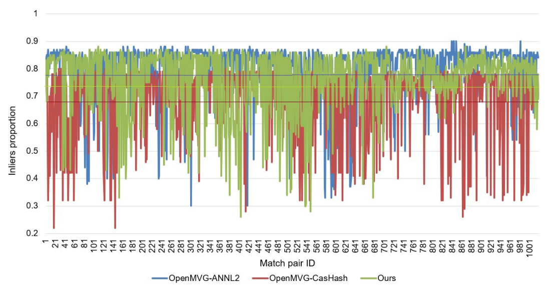
To evaluate the completeness, reprojection error and the number of con-



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



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



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

Figure 7: Comparison of the inliers proportion of the three pipelines matching. It can be seen that the matching accuracy of the proposed E-CasHash is better than that of cascade hashing and close to that of the multiple-random k-d trees algorithm.

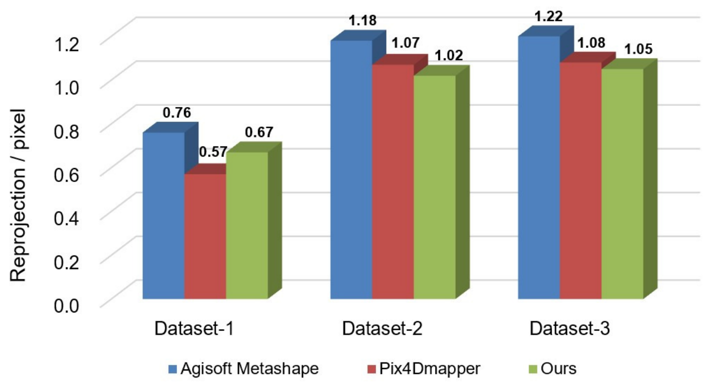
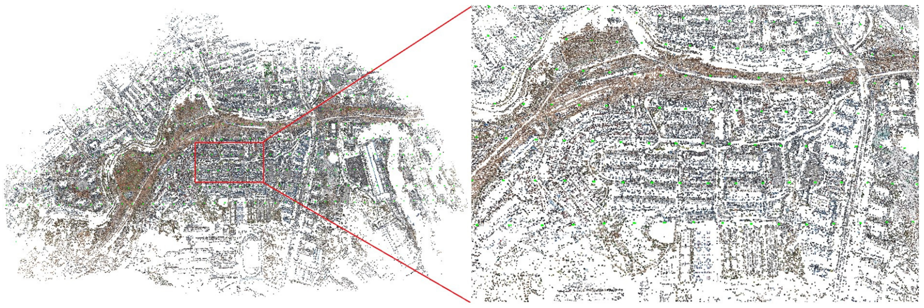


Figure 8: Average reprojection error comparison. The reprojection error can indirectly prove the accuracy of oblique image matching, we can find that the accuracy of our method is the highest on Dataset-2 and Dataset-3.

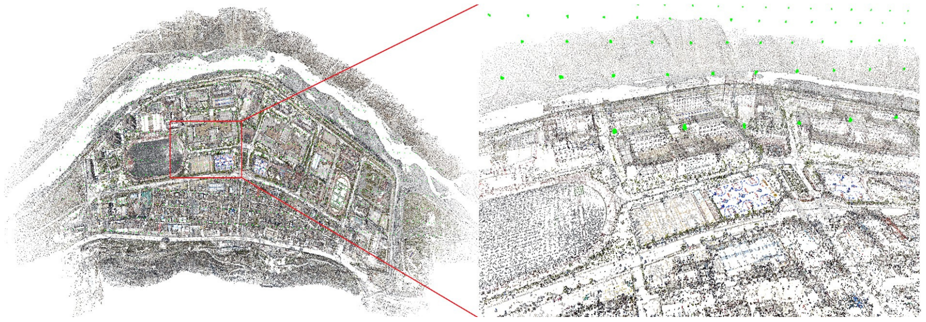
nected images and 3D tie points are counted. Fig.8 shows the reprojection error of the three pipelines after SFM reconstruction on the three data sets. Table 4 shows the information of the connected images and the generated 3D points, while the 3D point cloud generated by our pipeline and corresponding enlarged details are shown in Fig.9. We can see that: (1) from the analysis of reprojection error, our pipeline achieves a competitive accuracy; (2) the performance of our pipeline is similar to the Agisoft Metashape and Pix4Dmapper pipelines in terms of the number of connected images; (3) 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

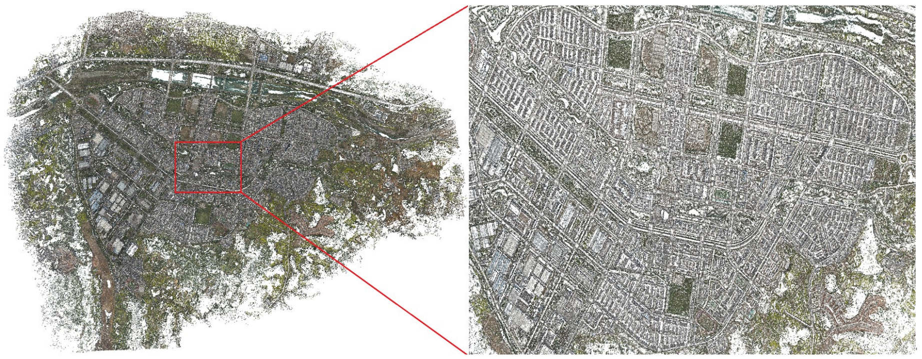
Dataset	Agisoft Metashape		Pix4Dmapper		Ours	
	Images	Points	Images	Points	Images	Points
1	1,914/1,914	293,490	1,914/1,914	102,310	1,914/1,914	259,614
2	3,489/3,490	473,759	3,490/3,490	423,158	3,490/3,490	693,083
3	14,212/14,225	4,631,650	14,214/14,225	1,904,063	14,216/14,225	3,073,284



(a) Dataset-1



(b) Dataset-2



(c) Dataset-3

Figure 9: SFM 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 increase the robustness 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 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 I/O cost. Combining the E-CasHash and the TLBDS algorithm, we design an efficient method for oblique image matching, and the efficiency of our method is 10∼50 times fast than the state-of-the-art.

In terms of the efficiency of oblique image matching, there are two reasons for our method's remarkable efficiency. First, our method uses epipolar constraint to reduce cascade hashing matching cost. From Section 4.3.1, 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 most importantly, we design an efficient TLBDS algorithm, which can significantly reduce the redundant transmission of match data and make full use of the computing resources of the GPU card. Specifically, redundant match data transmission has a high I/O cost and will cause insufficient data supply, making computing resources in a state of waiting for match data.

From the analysis of the inliers proportion (as shown in Section 4.3.2), 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 noise. By contrast, the hash remapping 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 repeat texture region is relatively high. Taking Euclidean distance between descriptors as the similarity measure will cause false matches during fine matching. 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 Fig.8.

6. CONCLUSION

In this paper, we proposed an efficient large-scale oblique image matching method. Our contributions were two-fold: the fast feature points matching based on cascade hashing with epipolar constraint and the efficient three-level buffer match data scheduling that considers the adjacency relationship between images. Extensive comparisons with state-of-the-art pipelines on oblique image matching on three datasets shown the efficiency as well as the accuracy of our pipeline for large-scale oblique image matching. Importantly, the application of the TLBDS algorithm allowed the matching of large-volume oblique images on a mid-level computer, and the matching efficiency of our method would 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

[1] Z. Xu, L. Wu, M. Gerke, R. Wang, H. Yang, Skeletal camera network embedded structure-from-motion for 3D scene reconstruction from UAV images, ISPRS journal of photogrammetry and remote sensing 121 (2016) 113–127.

[2] M. J. Westoby, J. Brasington, N. F. Glasser, M. J. Hambrey, J. M. Reynolds, 'structure-from-motion' photogrammetry: A low-cost, effective tool for geoscience applications, Geomorphology 179 (2012) 300–314.

[3] E. Rupnik, F. Nex, F. Remondino, Automatic orientation of large blocks of oblique images, The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 40 (Part 1) (2013) W1.

[4] J. Cheng, C. Leng, J. Wu, H. Cui, H. Lu, Fast and accurate image matching with cascade hashing for 3D reconstruction, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1–8.

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

[6] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

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

[8] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Computer vision and image understanding 110 (3) (2008) 346–359.

[9] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: An efficient alternative to SIFT or SURF, in: 2011 International conference on computer vision, Ieee, 2011, pp. 2564–2571.

[10] T. Xu, K. Sun, W. Tao, GPU accelerated image matching with cascade hashing, in: CCF Chinese Conference on Computer Vision, Springer, 2017, pp. 91–101.

[11] Z. Li, H. Jia, Y. Zhang, S. Liu, S. Li, X. Wang, H. Zhang, Efficient parallel optimizations of a high-performance SIFT on GPUs, Journal of Parallel and Distributed Computing 124 (2019) 78–91.

[12] P. Moulon, P. Monasse, R. Perrot, R. Marlet, OpenMVG: Open multiple view geometry, in: International Workshop on Reproducible Research in Pattern Recognition, Springer, 2016, pp. 60–74.

[13] E. Rupnik, M. Daakir, M. P. Deseilligny, Micmac–a free, open-source solution for photogrammetry, Open Geospatial Data, Software and Standards 2 (1) (2017) 1–9.

[14] A. Dascăl, M. Popa, Possibilities of 3D reconstruction of the vehicle collision scene in the photogrammetric environment Agisoft Metashape 1.6. 2, in: Journal of Physics: Conference Series, Vol. 1781, IOP Publishing, 2021, p. 012053.

[15] F. Alidoost, H. Arefi, Comparison of UAS-based photogrammetry software for 3D point cloud generation: A survey over a historical site., ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences 4.

[16] S. Jiang, C. Jiang, W. Jiang, 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 (2020) 230–251.

[17] C. Silpa-Anan, R. Hartley, Optimised KD-trees for fast image descriptor matching, in: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008, pp. 1–8.

[18] C. Wu, SiftGPU: A GPU implementation of david lowe's scale invariant feature transform (SIFT), GitHub, San Francisco, CA, USA.

[19] S. Jiang, W. Jiang, Efficient structure from motion for oblique UAV images based on maximal spanning tree expansion, ISPRS Journal of Photogrammetry and Remote Sensing 132 (2017) 140–161.

[20] L. Barazzetti, F. Remondino, M. Scaioni, R. Brumana, Fully automatic UAV image-based sensor orientation, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci 38 (Part 1) (2010) 6.

[21] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, R. Szeliski, Building rome in a day, Communications of the ACM 54 (10) (2011) 105–112.

[22] C. Wu, Towards linear-time incremental structure from motion, in: 2013 International Conference on 3D Vision-3DV 2013, IEEE, 2013, pp. 127–134.

[23] X. Wang, F. Rottensteiner, C. Heipke, Structure from motion for ordered and unordered image sets based on random kd forests and global pose estimation, ISPRS Journal of Photogrammetry and Remote Sensing 147 (2019) 19–41.

[24] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1) (1967) 21–27.

[25] M. Muja, D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration., VISAPP (1) 2 (331-340) (2009) 2.

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

[27] S. Dasgupta, Y. Freund, Random projection trees and low dimensional manifolds, in: Proceedings of the fortieth annual ACM symposium on Theory of computing, 2008, pp. 537–546.

[28] M. Muja, D. G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, IEEE transactions on pattern analysis and machine intelligence 36 (11) (2014) 2227–2240.

[29] M. Muja, D. Lowe, FLANN-Fast library for approximate nearest neighbors user manual, Computer Science Department, University of British Columbia, Vancouver, BC, Canada 5.

[30] C. Strecha, A. Bronstein, M. Bronstein, P. Fua, LDAHash: Improved matching with smaller descriptors, IEEE transactions on pattern analysis and machine intelligence 34 (1) (2011) 66–78.

[31] T. Xu, K. Sun, W. Tao, GPU accelerated cascade hashing image matching for large scale 3D reconstruction, arXiv preprint arXiv:1805.08995.

[32] M. S. Charikar, Similarity estimation techniques from rounding algorithms, in: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, 2002, pp. 380–388.

[33] D. Guide, CUDA C programming guide, NVIDIA, July 29 (2013) 31.

[34] J. L. Schonberger, J.-M. Frahm, Structure-from-motion revisited, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4104–4113.

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