Journal of Real-Time Image Processing (2021) 18:143–156
https://doi.org/10.1007/s11554-020-00959-y

ORIGINAL RESEARCH PAPER

Parallel hashing-based matching for real-time aerial image mosaicing

Roberto de Lima[1] · Aldrich A. Cabrera-Ponce[2] · Jose Martinez-Carranza[2,3]

Received: 2 September 2019 / Accepted: 9 March 2020 / Published online: 19 March 2020

Abstract

This paper presents a GPU-based real-time approach for generating high-definition (HD) aerial image mosaics. The cumbersome process of registering HD images is addressed by a parallel scheme that rapidly matches binary features. The proposed feature matcher takes advantage of the fast ORB (oriented FAST and rotated BRIEF) descriptor and its attainable arrangement into hash tables. By exploiting the best functionalities of binary descriptors and hashing-based data structures, the process of creating HD mosaics is accelerated. On average, real-time performance of 14.5 ms is achieved in a frame-to-frame process, for input images of 2.7 K resolution (2704 × 1521). For evaluation purposes in terms of robustness and speed, we selected two image registration methods for comparison. The first method uses the feature extractor and matcher modules of the well-known ORB-SLAM. The second comparison is carried out against the standard KNN-based matcher of OpenCV. The experiments were conducted under different conditions and scenarios, and the proposed approach exhibits a speed-up of 10.5 times compared to ORB-SLAM-based approach and 36.5 times compared to the OpenCV matcher. Therefore, this research widens the range of applications for aerial mosaicing, since the proposed system is capable of creating high-detail panoramas of large sites while acquiring data.

Keywords Image stitching · Feature matching · CUDA · Binary descriptors

1 Introduction

In the recent decade, with the increase in unmanned aerial vehicles, the task of aerial mapping through image mosaicing has become the core of several applications. For example, construction progress monitoring [1], post-disaster assessment [2], 3D reconstruction of earth-moving construction sites [3], etc. The performance of these applications heavily relies on both the quality of the obtained image and the processing time. On the one hand, high-resolution mosaics enable experts to infer high-level information from the obtained data. On the other hand, real-time processing allows the operator to carry out on-site inspections and to make decisions while capturing data.

To overcome the mentioned constraints, numerous researches have focused on achieving a good trade-off between HD mosaics and real-time processing by enhancing the image registration process. More specifically, researchers have proposed methods to speed up the process of aligning aerial images while maintaining stitching accuracy. Feature-based image registration is the cornerstone of these approaches since there exist various visual descriptors that offer both robustness to image deformations and registration accuracy. The well-known SIFT algorithm [12] has been widely used for image stitching. However, the construction of this floating-point descriptor is computationally expensive, preventing aerial mosaics systems from achieving real-time performance.

Binary descriptors [4–6] have gained popularity over the last years as they offer simple computations while performing similarly to floating-point descriptors. Nevertheless, since these fast descriptors have constructed from pairwise comparisons of pixel properties, the matching process becomes sensitive to affine transformations. This constraint negatively affects the alignment of the aerial images that

✉ Jose Martinez-Carranza
carranza@inaoep.mx

Roberto de Lima
roberto.delimahernandez@kuleuven.be

Aldrich A. Cabrera-Ponce
aldrichcabrera@inaoep.mx

1 KU Leuven, Leuven, Belgium

2 INAOE, Puebla, Mexico

3 University of Bristol, Bristol, UK

compose the mosaic. Therefore, a denser number of features is required to mitigate the registration error. However, this time-consuming process prevents the system to operate at high frame rates, especially when dealing with HD images (more than 1280×720 resolution).

In order to produce real-time HD aerial mosaics (Fig. 1), we propose a methodology from a GPU-based binary feature matching approach. This parallel scheme relies on the robust ORB [6] descriptor and on the hashing-based data structure to approximate nearest neighbour search [7]. The organisation of the binary vectors within the hash tables notably reduces the computational burden of the feature matching process, allowing for generating aerial mosaics at frame rates up to 35 ms [8]. However, the higher image resolution, the greater the number of tables to store the binary vectors. Hence, the real-time performance of the aerial mosaic system speed drastically drops bellow 5 fps when processing images with resolution higher than 800×480. By parallelising the process of storing and searching the binary descriptors in the hash tables, the creation of aerial mosaics is performed at 65 fps, considering ultra-HD images.

Fig. 1 Aerial mosaic performed on a 100 m high flight, using hash table in a GPU architecture. The distance travelled by the drone for the generation of the mosaic was 1.7 km. A video of this work for reviewing purposes is found at https://youtu.be/CRT3v1n6xtQ

The main contribution of this extended paper is a novel data arrangement suitable to match dense sets of binary descriptors based on the fundamental aspects of hashing. Unlike the sequential process that entails building and searching within hash tables, dynamic lists are avoided when constructing the tables so that descriptors are efficiently allocated into GPU's memories. This distribution is fundamental for computational cores to compare binary strings in parallel massively. As a result, the system is capable of processing thousands of key points, which is a critical requirement to register HD images in real-time accurately. Moreover, the experimental framework is extended by testing the methodology for different terrains, drone velocities, paths and image resolutions, thereby showing the robustness of the system against multiple conditions.

In order to present our work, this paper is organised as follows. Section 2 provides a literature study of the feature-based approaches to generate an aerial image mosaics. Section 3 describes the step-by-step process to generate aerial mosaics. The parallel architecture to store and arrange the binary vectors is conveyed in detail in Sect. 4. Section 5 describes the experimental design and the comparison of our approach against fast registration implementations. Finally, conclusions and future work are outlined in Sect. 6.

2 Related work

Image mosaicing is a challenging area of study since it involves various image processing steps: visual descriptors, registration, stitching and blending. The registration stage plays a crucial role in the creation of seamless aerial mosaics. This computational module computes a matrix that numerically relates a pair of images. This is a relation in the form of a rigid transformation, meaning that the images are related based solely on rotation and translation. This geometric relation is typically computed on the basis of correspondences of image characteristics extracted from either frequency domain or spatial domain. The proposed research fall within the latter category, due to aerial images are registered by estimating correspondences between pixel properties. Seeking to draw the differences and advantages of our system over the state of the art, the related work focuses on spatial domain-based registration techniques.

2.1 Spatial domain-based aerial image mosaicing approaches

In general terms, spatial domain-based approaches aim to compute a geometric relationship between a pair of images based on the information provided by their pixels. Frequently, the correspondence problem is tackled by comparing templates or local features. The former approach finds correspondences by calculating similarity metrics [9, 10] between grouped pixels, generally known as windows or templates. The latter approach computes numerical representations of key points in the form of floating-point or binary vectors, to later find matches between them. Feature-based aerial mosaicing systems are described in this section.

2.1.1 Feature-based image mosaicing approaches

Feature-based image registration methods have been widely used to be the main core of several computer vision applications. Their popularity lies in the robustness against affine transformations, such as rotation, translation, illumination and scale. Over the last years, the computational burden of these algorithms has been significantly reduced. As a result, these approaches have been broadly used to perform real-time applications. More concretely, image mosaicing systems have leveraged feature-based registration not only to improve the image alignment accuracy but also to accelerate the processing time [11]. The general workflow of these approaches is listed as follows. First, local descriptors are extracted and matched. Afterwards, the set of correspondences are used to find a rigid transformation between the images. Finally, the images are aligned into a standard reference frame. This process is repeated every key frame, obtaining like a result an aerial mosaic.

SIFT [12] is a floating-point descriptor that has played a seminal role in feature-based applications, in such degree that some studies [13] consider it as one of the most powerful descriptors. As a consequence, the SIFT descriptor has been used to be the central computational module for image mosaicing systems [14, 15]. Given the fact that SIFT offers robustness at the cost of a heavy computational burden, these approaches exhibit high accuracy in terms of image alignment, but a slow frame rate.

From the need of speeding-up the feature extraction and matching process, binary descriptors [16] emerged as an alternative to floating-point features, offering low memory footprint, lighter computational burden, and fast descriptors comparison. ORB [6] is an example of which that derived from the seminal binary descriptor BRIEF [5]. Given its robustness and fast matching, this binary descriptor has been recently used for real-time aerial image mosaicing. In [17], ORB descriptors are extracted and matched by using a brute force approach. Accordingly, once outliers are removed using RANSAC, a perspective transformation in the form of either a homography or a rigid transformation matrix is computed to stitch the images. In [18], RANSAC is substituted by the spatial–temporal coherent filter method to accelerate the outliers removal stage, generating aerial mosaics in real time. In [19], the process from extracting matches to finding correspondences is naively parallelised, by splitting the images in the number of CPU threads available. More recently, in [8] the feature matching process is accelerated by arranging ORB descriptors into hash tables, achieving a real-time performance up to 30fps. However, the computational burden of the presented approaches becomes a bottleneck as the aerial image is resolution is higher since the registration process requires a dense matching.

Appropriate to cope with the correspondence problem of thousands of descriptors, several works have focused on speeding the matching process itself for a wide range of applications. For instance, in [20] epipolar geometry and hash tables are leveraged to match BRIEF vectors and to perform a dense stereo reconstruction. A remarkable robust matching approach is reported in [21], which exploits the vocabulary tree data structure to notably speed up the matching of binary vectors. This fast approach has drawn the attention of numerous researchers, in a way that it is a key component of the feature matching stage of ORB-SLAM [22], the well-known navigation and mapping system. With the advent of sophisticated hardware architectures, researches have turned their focus to parallel-based feature matching implementations. In [23], a novel GPU-based architecture was proposed to accelerate the brute force matching without taking into consideration epipolar geometry assumptions. By using a lower-level design, in [24], a low-cost FPGA implementation of the ORB descriptor is proposed, notably speeding up the extraction and matching process. Even though different application domains have benefited from these algorithmic and hardware-based alternatives, image mosaicing systems have not fully exploited them.

Therefore, in this work, the generation of HD image mosaics is conducted by employing a massive parallel GPU-based architecture. This quick scheme takes the advantages of arranging the binary descriptors into hash tables to maximise the real-time performance of the feature matching process. For the sake of a fair comparison, the aerial mosaicing system is also implemented with the feature extractor and matcher of the ORB-SLAM. Although the OBB-SLAM-based aerial mosaicing approach exhibits promising results, the proposed system is capable of processing 2.7K resolution images at 60 fps, which is the standard frame rate of modern cameras.

3 Aerial image mosaic approach

As mentioned in Sect. 1, this work is an extension of the proposed methodology presented in [8], where we underscored the impact of arranging binary image descriptors into hash tables to accelerate the feature matching process. For a clearer understanding of the whole approach, this section

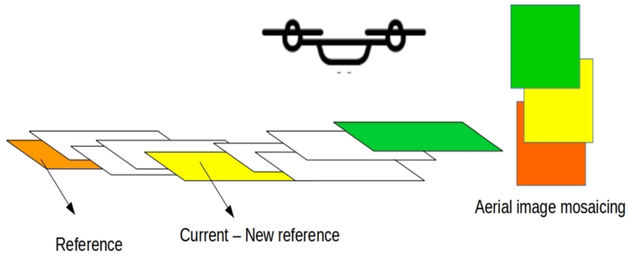

Fig. 2 General overview of the proposed approach to generate an aerial image mosaicing (image taken from [8])

describes the workflow to generate an aerial mosaic, from reading the image frames, the hash-based search approach, to the image registration process.

The general overview of the proposed system is shown in Fig. 2. This process is summarised in the flow diagram depicted in Fig. 3. This scheme describes the frame-to-frame computational process to generate aerial mosaic. The baseline of the proposed strategy is composed of twofold: i) feature extraction and matching, ii) image registration.

3.1 Feature extraction and matching

As it was mentioned, binary descriptors are well suited for real-time systems. The ORB descriptor is an example of which that has gained its place as the most used binary feature since it offers: ease to compute, fast comparison, low memory footprint, and robustness to affine transformations. The feature extraction module of the proposed approach relies on this robust descriptor, given the aforementioned characteristics. The ORB binary vector is constructed from pairwise pixel values tests. The BRIEF descriptor [5] introduced this way of describing features. However, ORB differs from BRIEF in the sense that the former descriptor is capable of handling in-plane rotations. Besides, a learning algorithm is incorporated in the construction of the binary vectors to enhance their distinctiveness. This characteristic, in particular, makes the ORB descriptors well suited to be organised into hashing tables. The benefits of arranging the binary vectors into this fast data structure are twofold. First, the binary representation of ORB involves a low memory footprint, dramatically reducing memory resources. Second, as feature searching lies in the Hamming space, the vectors are fast to compare by using hardware instructions such as *popcount*, which are intrinsically implemented on Nvidia and Intel processors. These properties are fundamental to reduce the computational burden of the feature matching stage. Moreover, in [6], authors have tested the feature matching performance when the binary vectors are arranged into a Local Sensitivity Hashing data structure LSH [7]. The obtained results are comparable to the performance of SURF and SIFT descriptors. Hence this fast binary descriptor has been widely used for ample applications that entail image registration.

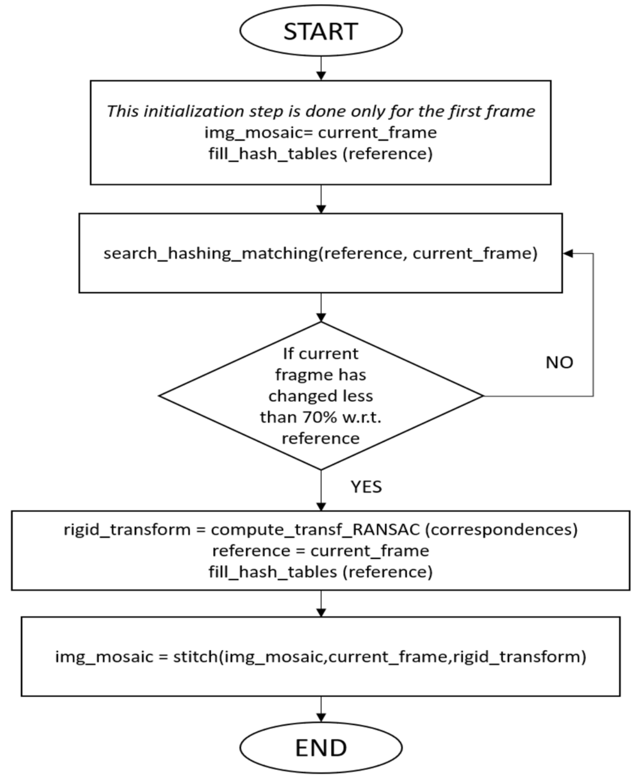

Fig. 3 Flow diagram of the proposed approach (image taken from [8]). In the last step, the current frame turns into the reference frame, and the process is repeated

3.2 Image transformation estimation and stitching

Following the typical pipeline for image stitching, the next step after finding matches between two images consists of estimating a rigid transformation. The computed matrix represents a non-singular relationship between image planes [25]. That is to say, the rigid transformation matrix describes the transformation that relates the set of matched points in terms of rotation and translation. This matrix is computed based on the relationship of at least four correspondences. However, the more correspondences estimated, the more accurate the transformation is. RANSAC [26] is used to compute the best transformation that relates the matched points of the current frame to the reference frame. Finally, the images are stitched and loaded into a canvas for visualisation.

As indicated in [8], binary descriptors and hash-based search notably reduce the computational burden of the image registration stage. Nevertheless, the processing time increases exponentially as the resolution becomes higher. The bottleneck has to do with the sequential construction of hash tables since the number of buckets needed to build up the tables relies on the quantity of extracted features.

4 Parallel architecture to store and retrieve ORB descriptors into multiple hash tables

In order to speed up the hash-based feature search, a novel way of structuring and searching by means of a parallel approach is presented in this section. The feature matching approach is described in two parts. Firstly, the organisation of the binary vectors into the hash tables, including the storing criteria, number of tables and searching space. Secondly, the CUDA-based architecture to rapidly compute correspondences. This description includes not only the algorithmic components of the storing and searching processes but also the data arrangement into the different memories of the GPU.

4.1 Registration based on hashing techniques

Hashing techniques include two main algorithmic processes, filling and searching tables. These are carried out based on a hashing function, which indicates the index address of the tables where the data is either stored or searched. The filling stage is performed for the reference frame, and the searching process is carried out for the current frame, as shown in Fig. 3.

The data organisation is performed following an LSH strategy. As the ORB descriptors are already in the Hamming space, the hashing function consists of a subset of 8 bits out of 256-bits vector, selected by taking a consecutive subset of 8 bits. Once the tables are filled, the process to match a single feature of the current frame w.r.t. the reference frame is listed as follows:

1. Find the hashing keys for the different tables.
2. For each table, find the Hamming distance against all the descriptors in the bucket:
   - Compute the XOR operation between bits.
   - Count the number of one-bits by using the *pop count* processor instruction.
3. Find the minimum distance provided for each table.
4. If the minimum distance is less than a threshold, then consider it as a match.

4.2 Parallel hashing-based binary matching

This subsection describes the parallel CUDA-based implementation of two crucial processes: filling and searching tables. The image mosaicing workflow remains as presented in Fig. 3, meaning that only those functions *FillHashTables* and *SearchHashingMatching* are parallelised. The latter function is an embarrassingly parallel workload since a single processor can process each hash table at the same time. However, the task of arranging binary descriptors into hash tables requires a strictly sequential process. Therefore, the parallel computational logic differs considerably from the sequential approach. The parallel design of the mentioned processes are illustrated in Figs. 4 and 5 respectively. The description of both diagrams is detailed in the following paragraphs.

Filling tables function In order to alleviate the lack of dynamic data structures presented on GPU's ecosystems, the binary descriptors are stored into a large vector of binary strings and sorted based on their hashing key value. This way of arranging the data differs from the index-vector data structure but helps to process the data independently. The step-by-step process for the binary arrangement organisation is enumerated as follows:

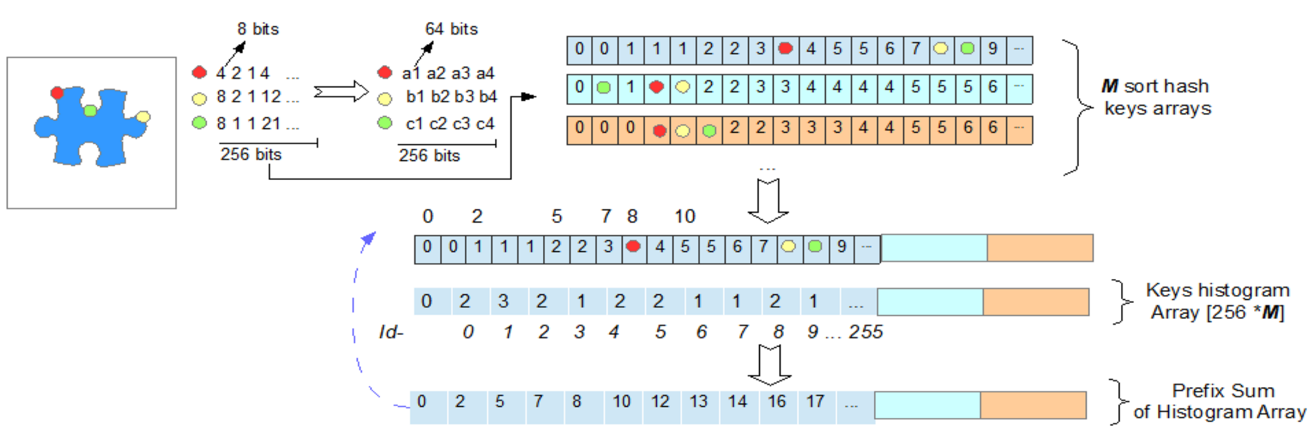

Fig. 4 Process of arranging binary descriptors suitably to perform parallel searching. This is done every time the reference frame is updated

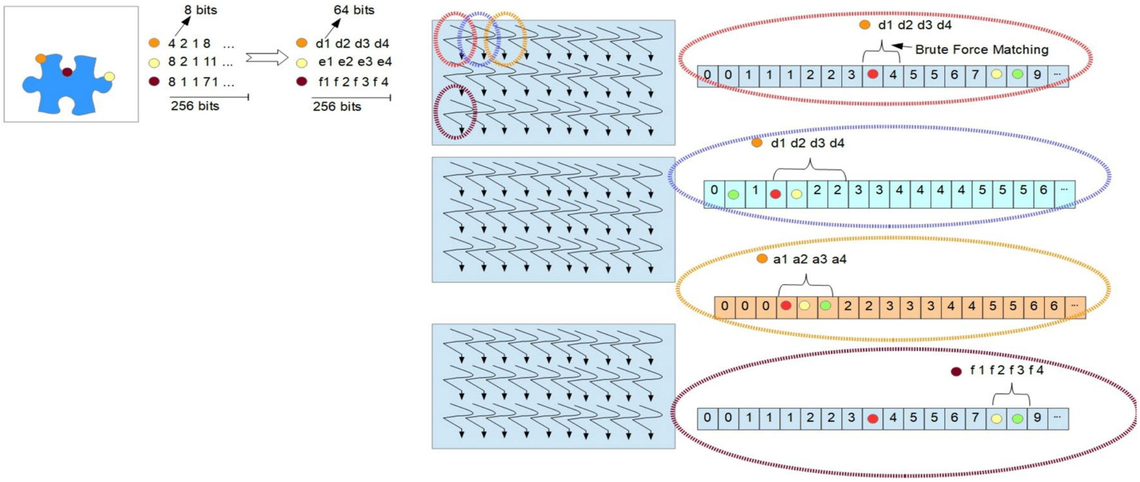

Fig. 5 Massively parallel approach to retrieve NN descriptors from the hash tables. The rectangles and curly narrows represent the blocks and threads that compose the GPU architecture. This is an example to retrieve the NN descriptors from the tables created in Fig. 4

1. ORB descriptors are computed.
2. 256-bit binary vectors are split into 64-bits arrays.
3. *M* 8-bits hashing keys are selected and sorted for each descriptor (M is the number of hash tables).
4. *M* histograms of hashing keys are computed.
5. A prefix-sum vector is calculated for each histogram.

For the sake of clarity, Fig. 4 depicts the above process for three descriptors and hash tables, respectively. Firstly, the extraction of binary descriptors is implemented on the GPU by using OpenCV. These distinctive features are marked in red, green and yellow in Fig. 4 extreme top-left. The numeric representation of the ORB descriptor is given in the form of an 8-bits unsigned integer array of 32 elements. However, to fully exploit the *popcount* processor function, descriptors are converted into a 64-bits format, resulting in a binary descriptor in the form of an integer array of 4 elements. The *popcount* instruction of the GPU yields the number of bits that are set to 1 in a 64-bit integer, allowing for rapidly computing the hamming distance between two binary vectors in the matching stage. The next step consists of arranging the set of vectors on the basis of a hashing structure. This means that the vectors are grouped by hashing keys so that the searching process is carried out among the elements of a bucket instead of the whole set. In order to avoid a sequential memory arrangement and search, we propose to construct two arrays to store and retrieve the binary bins. One long vector in which the descriptors are sorted by their respective key, and a string that contains the prefix sum of the histogram of keys. The former vector alone represents the hashing structure, while the later array supports the data access. The prefix-sum vector P indicates the initial index of every key k so that the range within which a descriptor might be found is delimited by the index $P(k)$ and $P(k+1)-1$. An example of how this vector may be constructed and deployed is depicted in Fig. 4 button-right. For instance, the descriptors that belong to the key 1 are found from the index 2 and 4 of the first hashing array (top-right). The operations to build up these arrays are embarrassingly parallel. The descriptor keys are sorted by means of the radix sort parallel algorithm, and the histogram and prefix-sum are computed by deploying atomic operations to avoid race conditions.

Searching tables function The hashing-based matching parallelisation relies on the fact that each table can be processed by an independent computational core. Figure 5 shows a matching example between the reference frame descriptors shown in Fig. 4 and a new set of descriptors arranged into 10 hashing tables which are extracted from the current frame. Figure 5 top-left represents the current frame, and its descriptors. Next to it, the rectangles and curly arrows showcase the threads and blocks of the GPU, respectively. Lastly, the same figure top-right illustrates the kernel process for four CUDA threads. As noted, every created hash table is processed by each column of the grid of threads. For example, the threads marked with dotted lines in red, blue and green compute their corresponding Nearest Descriptor (ND) from the first, second and third tables, respectively. Therefore the GPU architecture is designed as follows: number of blocks = $\#Descriptors/64$, number of threads per block = $(64, \#Tables)$ threads per block. The algorithmic description of this process is depicted in Algorithm 1. This computational module requires as input the vectors obtained in the previous step and the set of current descriptors. This parallel strategy focuses on arranging the descriptors in such a manner that shared memory is exploited, accelerating the process of fetching data from memory. First, the computed descriptors of the current frame are converted into a 64-bits format. After that, the binary vectors are temporarily stored in shared memory. Next (line 10–15), the "bucket" keys are selected, and a brute force matching based on Hamming distance is carried out to find the closest feature between the current descriptors and the reference descriptors. The obtained output is a list of Hamming distances that indicates the M potential matching candidates (M is the number of tables). The final set of correspondences is obtained through a Hamming-based comparison between a defined threshold and the distance of matching candidates obtained from different hash tables. Based on this criterion, the best match is estimated while avoiding conflicts among hash table results.

Algorithm 1: kernel, Parallel Hashing-based Feature Matching

```
   Input: SortedKeysArray
   Input: HistArray
   Input: PrefixSumArray
   Input: NoPrev ← Number of previous descriptor
   Input: PrevDesc ← Previous descriptors in 64-bits format
   Result: MatchingVector
   Result: HammingDistVector
1  ImgDescriptors ← New descriptors array;
2  ORB-GPU(ImgDescriptors);
3  Dsc[64] ← 64-bits array in shared memory;
4  ty ← threads per block y-axis;
5  tx← threads per block x-axis;
6  j ← total of threads x-axis;
7  From8to64(Desc[tx]);
8  Syncthreads();
9  t = 256*ty ← Start index of each table;
10 key=ImgDescriptors[j*32+ty]← Hashing Keys ;
11 for i=0;i<HistArray[t+key]; i++) do
12     id= SortedKeysArray[NoPrev * ty + PrefixSumArray[t + key]+ i];
13     HammingDistance(PrevDesc[id],Dsc[tx]);
14     Compute the minimum Hamming distance of each table
15 end
16 MinDistance(MinDist) ← Compute the minimum Hamming distances
     among the hash tables;
17 if MinDist < Threshold then
18     MatchingDistVector← MinDist;
19     MatchingVector← MinDist Index;
20 end
```

5 Experiments and results

The carried out experiments focus on assessing the impact of hash tables on the feature matching process and the image mosaicing system as a whole. Therefore, the experimental design is divided into two sections: an evaluation of the number of tables used for storing/searching and the real-time performance of the system when compared to similar methods. The first evaluation is crucial to measure how the nosiness of binary features impacts on the matching process. The second experiment allows us to locate the proposed approach in comparison to other fast matching approaches. To this end, different feature extractors and matches are incorporated into our workflow, more specifically, the KNN matcher of OpenCV and the feature extractor and matcher modules of ORB-SLAM.

All the experiments were performed on the following system: Intel Core i7-6700 at 2.6 GHz, 32 GB RAM, Graphic card Nvidia GTX1070, 1056 MHz GPU clock, 1920 CUDA cores, 8 GB memory. For the experimental setup (see Fig. 6), the drone Matrice 100 with the 4 K monocular camera Zenmuse X3 on-board is used to acquire the data. The frames are transferred directly to the remote control and then to the workstation connecting the HDMI output to Frame Grabber AV.io 4K where is compressed and decoded there to grabbed it to the workstation. This way, we were able to decode the data and to convert the drone's images into a compatible format, like a video streaming of an USB camera. Besides, this acquiring data system enabled us to operate with different resolution images while preserving the aspect ratio of the original image.

The synchronisation of the algorithmic modules was conducted by the Robotic Operating System (ROS) [27]. As depicted in Fig. 6, two nodes of ROS are required to generate the aerial image mosaic. The first node performs the following tasks: (1) to read the image from the workstation, (2) to extract and to estimate correspondences of the binary descriptors, (3) to compute the rigid transformation matrix. The output of this node is the key-frame along with the rigid transformation w.r.t. the previous keyframe. This information is published to the second node, whose primary task consists of stitching the images and generating the mosaic. Given the flexibility of this structure, the system is easily adapted, so that different algorithmic modules could be loaded into the nodes—for example, the modules of feature extractor and matcher of ORB-SLAM.

5.1 Impact of the number of tables on feature matching

As the number of tables to arrange the binary descriptors increases, the quantity of potential matches becomes higher. Computational complexity wise, increasing the number of tables does not impact the real-time performance of searching potential binary descriptor correspondences from hash tables, as this process is carried out in parallel. However, in terms of data storage, the system performance might be affected due to the fact that fast data stream storage units have memory footprint constraints. Consequently, it is vital to determine the optimal number of tables needed to retrieve discriminative features. To this end, the feature searching approach is tested for different numbers of hashing tables. Figure 7 shows the contrast between the quantity of obtained correspondences against the number of tables. As noted, it is clear that from 10 tables on, the number of true correspondences slightly changes. This plot confirms the distinctiveness of the ORB descriptors and provide us with an insight into

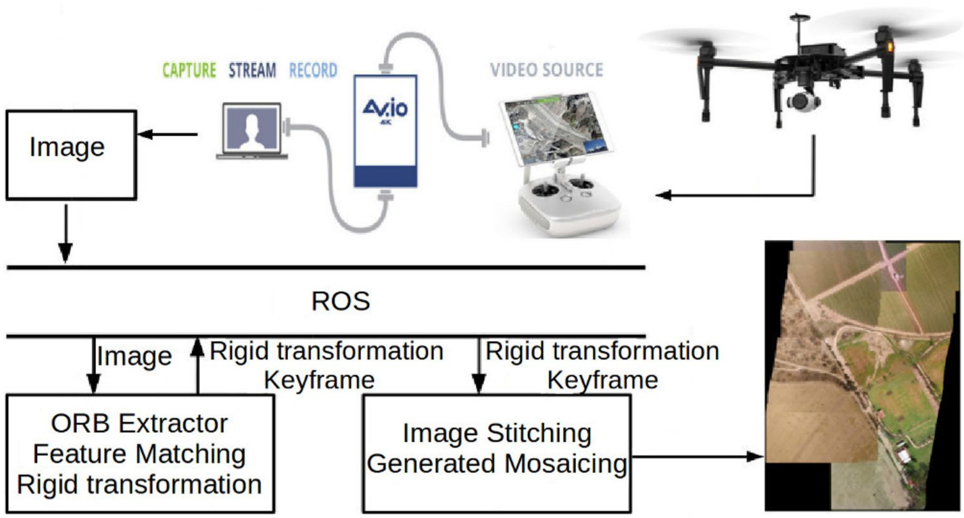

Fig. 6 Setup for the real-time image mosaicing approach: digital acquisition system and algorithmic modules synchronised by the Robot Operating System (ROS) [27]. The components performed on the GPU are both the ORB extractor and Feature Matching. The latter encompasses the proposed GPU architecture

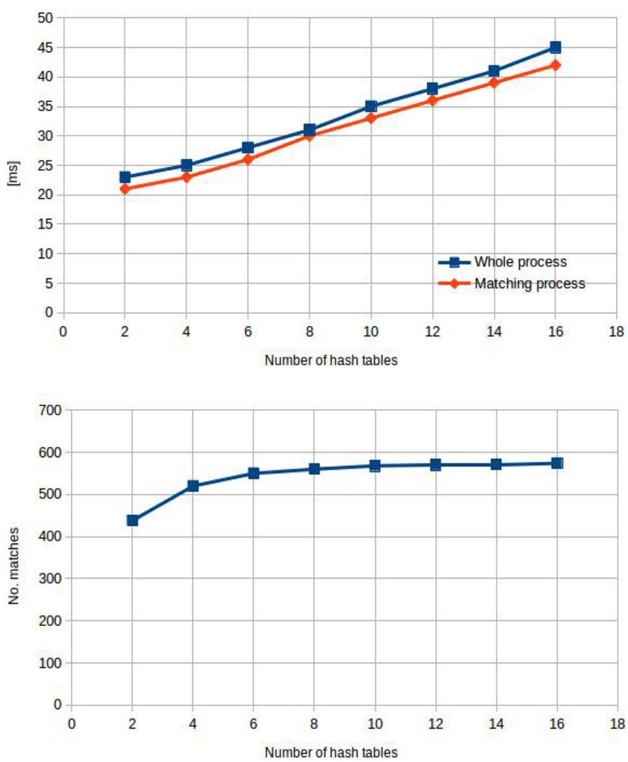

Fig. 7 Top image: mean processing time results for different number of hash tables; Bottom image: mean number of found matches, considering 2200 extracted features (image taken from [8])

the tendency of the number of matches w.r.t. the number of hashing tables.

5.2 GPU results in different environments

The proposed method is tested in different scenarios considering different image resolutions: WVGA (853×480), HD (1280×720), UHD (1920×1080) and 2.7K (2704×1521). To avoid bias, the extension of the mapped area as well as the drone's altitude is different for each scenario. For the sake of comparison, the frames of the recorded trajectories are processed by different image registration approaches. These include: OpenCV ORB extractor + OpenCV matcher, ORB-SLAM feature extractor + ORB-SLAM matcher, OpenCV ORB extractor + Hash-Table matcher (CPU), OpenCV ORB extractor + Hash-Table matcher (GPU).

First scenario: Instituto Nacional de Astrofisica, Optica y Electronica (INAOE) Figure 8 shows the traversed path along with the generated panoramas. The flight was performed at 100 m height at 4.0 m/s. Table 1 shows the real-time performance for different resolution images, processing time (feature matching + rigid transformation) and stitching time that is the average time that takes to perform the entire process outlined in the flow diagram of Fig. 3. As noted, when dealing with WVGA images, both the sequential and parallel hashing-based approaches exhibit a high frame rate. When HD images are processed, the ORB-SLAM and the sequential hashing approach perform fairly similar. However, the impact of the GPU-based approach becomes noticeable when dealing with 2.7 K resolution images. The most remarkable out of this experiment is the ability of the system to deal with a fast drone's speed, a 577 m distance was mapped in 2.4 min roughly.

Second scenario: Tlalancaleca This flight was carried out on an archaeological site in Puebla, Mexico, at 90 m height. Following the same scheme as the previous experiment, Fig. 9 shows the set of generated mosaics. As noted in Table 2, the real-time performance follows the same pattern as the previous flight. Approximately, when processing 2.7 K resolution images, the proposed system is 1:6 time

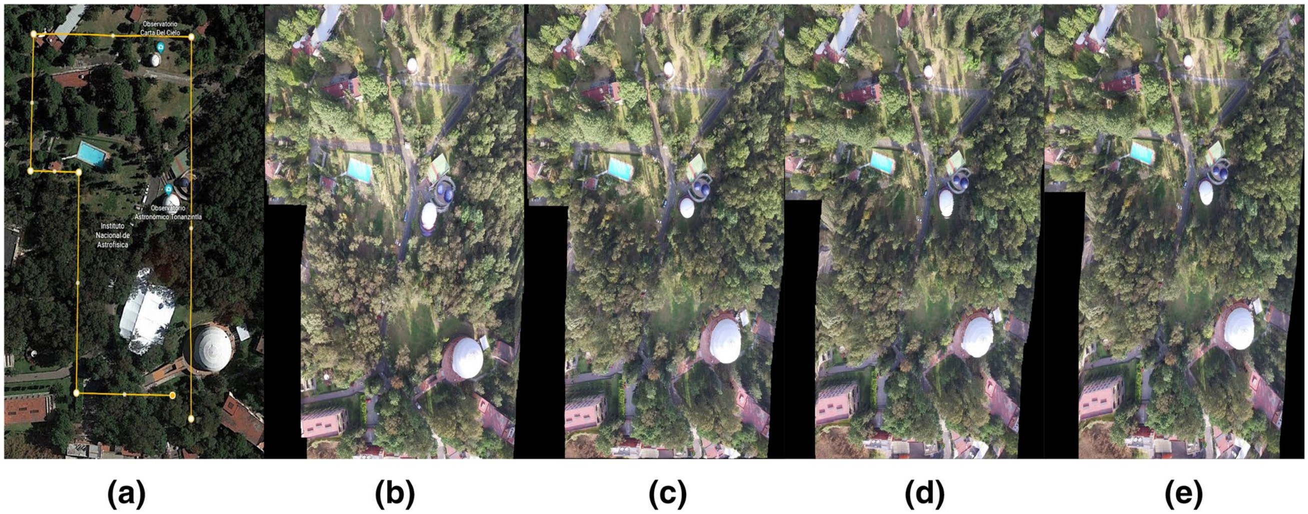

Fig. 8 From left to right. **a** Satellite image and drone's path (Latitude: 19.032416, Longitude: − 98.314594), travelled distance 576.95 m. **b** Mosaic generated with the OpenCV matcher, **c** Mosaic generated with ORB-SLAM feature extractor and matcher, **d** Mosaic generated with our approach CPU, **e** Mosaic generated with our approach GPU (1210 × 1920 resolution for each mosaic)

Table 1 GPU results in INAOE to 576.95 m of distance travelled

Method	Matches	Frames	Keyframes	Average (fps)	Processing (ms)	Stitching (ms)
Resolution of WVGA (853 × 480)						
OpenCV	448	4536	318	1.7159	582.771	625.129
ORB-SLAM	432	4536	51	13.1874	75.8298	48.6579
Hash table CPU	647	4536	173	23.9595	41.7370	72.8210
Hash table GPU	706	4536	391	91.984	**10.8714**	49.0051
Resolution of HD (1280 × 720)						
OpenCV	383	4536	300	1.6463	607.407	670.002
ORB-SLAM	461	4536	67	15.8730	63.0000	71.0002
Hash table CPU	725	4536	161	15.6180	64.0285	105.718
Hash table GPU	1051	4536	865	81.833	**12.2221**	64.7929
Resolution of UHD (1920 × 1080)						
OpenCV	334	4536	299	1.7414	574.240	884.510
ORB-SLAM	281	4536	66	8.8870	112.523	216.8701
Hash table CPU	626	4536	129	8.4569	118.526	284.133
Hash table GPU	568	4536	577	67.167	**14.8882**	183.435
Resolution of 2.7 K (2704 × 1521)						
OpenCV	241	4536	220	1.3412	745.562	1030.762
ORB-SLAM	89	4536	121	11.622	86.0410	265.723
Hash table CPU	401	4536	75	4.3437	230.216	539.503
Hash table GPU	647	4536	443	54.549	**18.3320**	262.580

The numbers in bold highlight that our proposed method, the Hash-Table GPU, achieved the least processing time w.r.t to the other methods, in all the cases

faster than the ORB-SLAM-based approach. It is important to note that both flights do not present significant drifting errors. However, the image shows that white construction has a small delay due to the stitching time for each method, being more noticeable with OpenCV.

Third scenario: open field In order to test the system on larger distances, we conducted two open field flights (see Figs. 10, 11). The drone's path covered distances of 1.17 Km and 2 Km, respectively, at 100 m height. As expected, for both flights, the real-time performance tendency remains

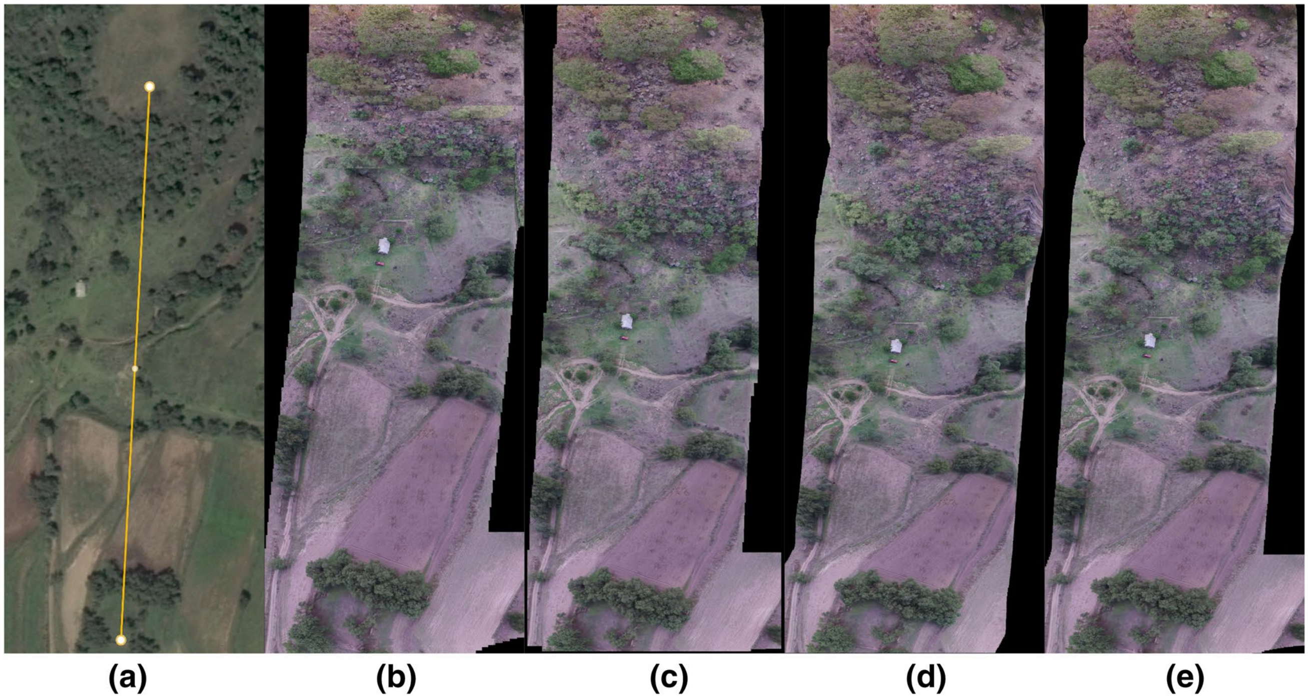

Fig. 9 From left to right. **a** Satellite image and drone's path (Latitude: 19.3147168, Longitude: − 98.5227814), travelled distance 340.45 m. **b** Mosaic generated with the OpenCV matcher, **c** Mosaic generated with ORB-SLAM feature extractor and matcher, **d** Mosaic generated with our approach CPU, **e** Mosaic generated with our approach GPU (683 × 1920 resolution for each mosaic)

Table 2 GPU results in Tlalancaleca to 340.45 m of distance travelled

Method	Matches	Frames	Keyframes	Average (fps)	Processing (ms)	Stitching (ms)
Resolution of WVGA (853 × 480)						
OpenCV	118	2664	264	2.0956	477.183	553.025
ORB-SLAM	348	2664	58	10.2590	97.4751	52.5785
Hash table CPU	562	2664	223	29.9943	33.3396	73.5690
Hash table GPU	348	2664	955	134.136	**7.4551**	59.8288
Resolution of HD (1280 × 720)						
OpenCV	42	2664	274	2.6293	380.323	570.394
ORB-SLAM	273	2664	71	8.1311	122.984	121.891
Hash table CPU	538	2664	177	18.0802	55.3090	167.537
Hash table GPU	420	2664	487	89.838	**11.1311**	162.961
Resolution of UHD (1920 × 1080)						
OpenCV	40	2664	217	1.7551	569.758	777.495
ORB-SLAM	160	2664	102	7.7197	129.538	265.518
Hash table CPU	291	2664	102	10.6442	93.9470	423.633
Hash table GPU	481	2664	473	71.193	**14.0463**	182.393
Resolution of 2.7K (2704 × 1521)						
OpenCV	18	2664	180	1.5150	660.057	1127.482
ORB-SLAM	39	2664	120	12.391	80.7032	630.019
Hash table CPU	218	2664	72	6.2028	161.215	452.409
Hash table GPU	510	2664	249	62.391	**16.0278**	292.619

The numbers in bold highlight that our proposed method, the Hash-Table GPU, achieved the least processing time w.r.t to the other methods, in all the cases similar to previous examples. However, from Fig. 10, it is noticeable that mosaics exhibit minor drift errors. The aerial image misplacement is prominent when using the OpenCV matcher and less visible in the mosaic produced by the ORB-SLAM approach. This error is caused by the terrain similarity, which negatively influences the distinctiveness of the descriptors (Table 3). As a consequence, the number of false-positives rises, preventing the system from computing accurate image transformations. Finally, Fig. 11 shows the results for a 2 Km distance flight. As well as the previous flight, imperceptible drifting errors are exhibited. However, the average frame rate of the GPU approach exceeds the other methods more than ten times, considering 2.7 K resolution images (Table 4).

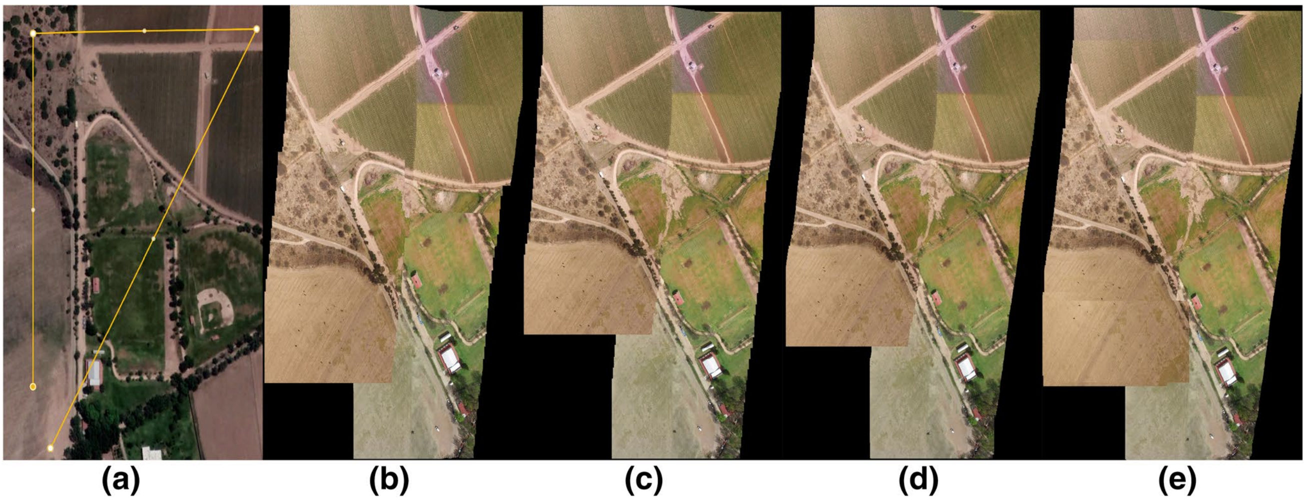

Fig. 10 From left to right. **a** Satellite image and drone's path (Latitude: 21.9869028, Longitude: – 102.2804471), travelled distance 1.17 Km. **b** Mosaic generated with the OpenCV matcher, **c** Mosaic generated with ORB-SLAM feature extractor and matcher, **d** Mosaic generated with our approach CPU, **e** Mosaic generated with our approach GPU (1210 × 1920 resolution for each mosaic)

Table 3 GPU results in an open field to 1.17 km of distance travelled

Method	Matches	Frames	Keyframes	Average (fps)	Processing (ms)	Stitching (ms)
Resolution of WVGA (853 × 480)						
OpenCV	389	4896	399	3.6461	274.115	688.614
ORB-SLAM	234	4896	115	12.3361	81.0628	55.888
Hash table CPU	547	4896	303	34.4131	29.0587	88.750
Hash table GPU	347	4896	840	127.250	**7.8585**	84.196
Resolution of HD (1280 × 720)						
OpenCV	548	4896	385	1.5710	636.537	746.893
ORB-SLAM	267	4896	76	11.1233	89.9008	221.284
Hash table CPU	660	4896	184	19.2777	51.8733	190.231
Hash table GPU	597	4896	631	100.687	**9.9317**	170.634
Resolution of UHD (1920 × 1080)						
OpenCV	353	4896	355	1.4140	707.167	868.692
ORB-SLAM	272	4896	118	8.9678	111.510	347.217
Hash table CPU	631	4896	126	10.8272	92.3598	428.107
Hash table GPU	1252	4896	356	65.435	**15.2822**	350.792
Resolution of 2.7K (2704 × 1521)						
OpenCV	216	4896	302	1.2163	822.120	1122.80
ORB-SLAM	69	4896	147	6.9946	142.967	754.490
Hash table CPU	461	4896	98	5.9307	168.612	550.424
Hash table GPU	1541	4896	356	65.619	**15.2394**	383.070

The numbers in bold highlight that our proposed method, the Hash-Table GPU, achieved the least processing time w.r.t to the other methods, in all the cases

Fig. 11 From left to right. **a** Satellite image and drone's path (Latitude: 21.9869028, Longitude: − 102.2804471), travelled distance 2.0 Km. **b** Mosaic generated with the OpenCV matcher, **c** Mosaic generated with ORB-SLAM feature extractor and matcher, **d** Mosaic generated with our approach CPU, **e** Mosaic generated with our approach GPU (1210 × 1920 resolution for each mosaic)

Table 4 GPU results in an open field to 2.0 km of distance travelled

Method	Matches	Frames	Keyframes	Average (fps)	Processing (ms)	Stitching (ms)
Resolution of WVGA (853 × 480)						
OpenCV	193	8640	749	2.1195	471.809	530.083
ORB-SLAM	425	8640	180	9.0495	110.502	161.881
Hash table CPU	469	8640	455	37.2345	26.8568	174.293
Hash table GPU	364	8640	1033	130.696	**7.6513**	173.208
Resolution of HD (1280 × 720)						
OpenCV	117	8640	984	1.6599	602.441	746.624
ORB-SLAM	254	8640	201	5.9579	167.844	523.698
Hash table CPU	508	8640	283	20.7923	48.0946	378.101
Hash table GPU	495	8640	1112	122.375	**8.1716**	246.020
Resolution of UHD (1920 × 1080)						
OpenCV	72	8640	568	1.58000	632.909	813.134
ORB-SLAM	114	8640	269	3.2545	307.261	978.207
Hash table CPU	481	8640	221	11.4935	87.0055	480.629
Hash table GPU	666	8640	704	86.419	**11.5714**	345.700
Resolution of 2.7K (2704 × 1521)						
OpenCV	28	8640	673	1.8763	532.959	817.358
ORB-SLAM	44	8640	401	6.5755	152.078	1023.894
Hash table CPU	293	8640	188	6.7586	147.958	495.826
Hash table GPU	1005	8640	673	68.528	**14.5925**	361.736

The numbers in bold highlight that our proposed method, the Hash-Table GPU, achieved the least processing time w.r.t to the other methods, in all the cases

6 Conclusions

A real-time parallel-based approach for creating aerial image mosaics was proposed. This rapid system can generate high-definition aerial panoramas at high frame rates up to 122 fps, considering 1280 × 720 resolution images. The main contribution of this approach relies on the fast registration method from binary descriptors and a GPU-hashing-based matcher. More concretely, a CUDA-based scheme is designed to rapidly store and search the binary vectors within several hashing tables. Based on this parallel image registration, the process of creating aerial mosaics is completed by stitching the images and organising them into a canvas for real-time visualisation. To show the effectiveness of this approach, various high-resolution mosaics were generated under different scenarios and conditions.

In order to situate the real-time performance of the proposed approach in comparison with fast broadly used registration methods, these computational components are instantiated into the system's pipeline. The evaluation results show that the presented method is suitable to process image resolutions from 1280×720 onward. For example, the obtained processing time is 7.6 times faster than the ORB-SLAM-based approach when pondering results for input images of 2704×1521 resolution. Besides, the real-time matching performance highly benefits the seamless creation of aerial image mosaics since it makes the system robust against turbulence caused by either wind conditions or flight control inputs.

To envisage the potential applications that become attainable with the proposed research, different types of terrains were recorded. For all the tested scenarios, the resulted aerial mosaics are comparable to their respective satellite images. The optimised process of generating aerial image mosaics allows for incorporating sophisticated algorithms into the pipeline, i.e. object detection, segmentation, anomaly monitoring and so on. However, in order to rely on these mosaics for surveying applications, alignment accuracy is a factor to consider for future work. To this end, image registration might be enhanced by incorporating GPS coordinates to mitigate the drift error.

Acknowledgements This work has been partially funded by the Royal Society through the Newton Advanced Fellowship with reference NA-140454 and by the CONACYT-INEGI research project 268528.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Ham, Y., Han, K.K., Lin, J.J., Golparvar-Fard, M.: Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (UAVs): a review of related works. Vis. Eng. **4**(1), 1 (2016)
2. Ezequiel, C.A.F., Cua, M., Libatique, N.C.,Tangonan, G.L., Alampay, R., Labuguen, R.T., Favila, C.M., Honrado, J.L.E., V. Canos, Devaney, C. et al.: UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development. In: International Conference on Unmanned Aircraft Systems (ICUAS), pp. 274–283. IEEE (2014)
3. Siebert, S., Teizer, J.: Mobile 3D mapping for surveying earthwork projects using an unmanned aerial vehicle (UAV) system. Autom. Constr. **41**, 1–14 (2014)
4. Leutenegger, S., Chli, M., Siegwart, R.Y.: Brisk: Binary robust invariant scalable keypoints. In: IEEE International Conference on Computer Vision (ICCV), pp. 2548–2555. IEEE (2011)
5. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: binary robust independent elementary features. Comput. Vis. ECCV **2010**, 778–792 (2010)
6. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571. IEEE (2011)
7. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, pp. 253–262. ACM (2004)
8. de Lima, R., Martinez-Carranza, J.: Real-time aerial image mosaicing using hashing-based matching. In: Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), pp. 144–149. IEEE (2017)
9. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, New York (2012)
10. Lewis, J.P.: Fast normalized cross-correlation. Vis. Interface **10**, 120–123 (1995)
11. Kokate, M.D., Wankhede, V.A., Patil, R.S.: Survey: image mosaicing based on feature extraction. Int. J Comput. Appl. **165**(1), 26–30 (2017). https://doi.org/10.5120/ijca2017913776
12. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
13. Khan, N., McCane, B., Mills, S.: Better than SIFT? Mach. Vis. Appl. **26**(6), 819–836 (2015)
14. Liqian, D., Yuehui, J.: Moon landform images fusion and mosaic based on SIFT method. In: International Conference on Computer and Information Application (ICCIA), pp. 29–32. IEEE (2010)
15. Nemra, A., Aouf, N.: Robust invariant automatic image mosaicing and super resolution for UAV mapping. In: ISMA'09. 6th International Symposium on Mechatronics and its Applications, pp. 1–7. IEEE (2009)
16. Bekele, D., Teutsch, M., Schuchert, T.: Evaluation of binary keypoint descriptors. In: 20th IEEE International Conference on Image Processing (ICIP), pp. 3652–3656. IEEE (2013)
17. Kern, A., Bobbe, M., Bestmann, U.: Towards a real-time aerial image mosaicing solution. In: International Micro-Air Vehicle Conference and Competition (IMAV) (2016)
18. Li, J., Yang, T., Yu, J., Lu, Z., Lu, P., Jia, X., Chen, W.: Fast aerial video stitching. Int. J. Adv. Robot. Syst. **11**(10), 167 (2014)
19. Wang, G., Zhai, Z., Xu, B., Cheng, Y.: A parallel method for aerial image stitching using ORB feature points. In: IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), pp. 769–773. IEEE (2017)
20. Heise, P., Jensen, B., Klose, S., Knoll, A.: Fast dense stereo correspondences by binary locality sensitive hashing. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 105–110. IEEE (2015)
21. Gálvez-López, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Trans. Robot. **28**(5), 1188–1197 (2012)
22. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)
23. de Lima, R., Martinez-Carranza, J., Morales-Reyes, A., Mayol-Cuevas, W.: Toward a smart camera for fast high-level structure extraction. J. Real-Time Image Proc **14**, 685–699 (2018). https://doi.org/10.1007/s11554-017-0704-5
24. Weberruss, J., Kleeman, L., Drummond, T.: ORB feature extraction and matching in hardware. In: Proceedings of the Australasian

Conference on Robotics and Automation, pp. 2–4. The Australian National University, Canberra (2015)
25. Agarwal, A., Jawahar, C., Narayanan, P.: A survey of planar homography estimation techniques. Technical Report IIIT/TR/2005/12. Centre for Visual Information Technology (2005)
26. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
27. Quigley, M., Conley, K., Gerkey, B.P., Faust, J.: ROS: an open-source robot operating system. ICRA Workshop Open Source Softw. **3**(3.2), 5 (2009)

Roberto de Lima is a Ph.D. candidate in the Geomatics Research Group at KU Leuven, in Belgium. He obtained a BSc in Electronic Engineering from the Benemerita Universidad Autonoma de Puebla and a M.Sc. in Computer Science from the Instituto Nacional de Astrofisica Optica y Electronica, in Mexico. At the latter institute, he also worked as a research assistant in the Computer Science Department. His research interests include high-performance computing, computer vision and graphical user interfaces.

Aldrich A. Cabrera-Ponce is Master Student in the Computer Science Department at the Instituto Nacional de Astrofisica, Optica y Electronica (INAOE). He obtained a B.Eng. in Mechatronics from the Instituto Tecnologico Superior de Atlixco, Puebla, Mexico. He is part of the Mexican team QuetzalC++ that has obtained awards in International Drone Competitions: 1st Place in the IROS 2017 Autonomous Drone Racing competition; 2nd Place in the International Micro Air Vehicle competition (IMAV) 2016 and ranked 4th in the IMAV 2017.

Jose Martinez-Carranza is Associate Professor in the Computer Science Department at the Instituto Nacional de Astrofisica, Optica y Electronica (INAOE) and Honorary Senior Research Fellow at the Computer Science Department in the University of Bristol. He obtained a BSc in Computer Science (Cum Laude) from the Benemerita Universidad Autonoma de Puebla in 2004, and an M.Sc. in Computer Science (Best Student) from INAOE in 2007, both institutions in Mexico. In 2012, he received his Ph.D. from the University of Bristol in the UK, where he also worked as Postdoctoral Researcher from 2012 to 2014. He received the highly prestigious Newton Advanced Fellowship (2015–2018), granted by the Royal Society in the UK to work with autonomous drones in GPS-denied environments. He also leads a Mexican team that has achieved outstanding performance in International Drone Competitions: 1st Place in the IROS 2017 Autonomous Drone Racing competition; 2nd Place in the International Micro Air Vehicle competition (IMAV) 2016; and ranked 4th in the IMAV 2017.