Real-Time Incremental and Geo-Referenced Mosaicking by Small-Scale UAVs

Danilo Avola[1], Gian Luca Foresti[1](✉), Niki Martinel[1], Christian Micheloni[1], Daniele Pannone[2], and Claudio Piciarelli[1]

[1] Department of Mathematics, Computer Science, and Physics, University of Udine, Via delle Scienze 206, 33100 Udine, Italy
{danilo.avola,ginaluca.foresti,niki.martinel,christian.micheloni, claudio.piciarelli}@uniud.it

[2] Department of Computer Science, Sapienza University, Via Salaria 113, 00198 Rome, Italy
pannone@di.uniroma1.it

Abstract. In the last decade, the use of small-scale Unmanned Aerial Vehicles (UAVs) is increased considerably to support a wide range of tasks, such as vehicle tracking, object recognition, and land monitoring. A prerequisite of many of these systems is the construction of a comprehensive view of an area of interest. This paper proposes a small-scale UAV based system for real-time creation of incremental and geo-referenced mosaics of video streams acquired at low-altitude. The system presents several innovative contributions, including the use of A-KAZE feature extractor in aerial images, a Region Of Interest (ROI) to speed-up the stitching stage, as well as the use of the rigid transformation to build a mosaic at low-altitude mitigating in part the artifacts due to the parallax error. To prove the correctness of the proposed system at low-altitude, the public UMCD dataset and a simple metric based on the difference between image regions are presented. Instead, to show the overall effectiveness of the system, the public NPU Drone-Map dataset and a correlation measure are used. The latter metric evaluates the similarity between mosaics generated by the proposed method and those provided by a reference work of the current literature. Finally, the performance of the system compared with that of different modern solutions is also discussed.

Keywords: UAVs · Incremental mosaicking · Real-time mosaicking · ROI · UMCD dataset · NPU dataset · Rigid transformation · A-KAZE

1 Introduction

In recent years, the use of small-scale UAV based systems to support a wide range of application domains has increased considerably. In particular, these systems are especially useful in all those tasks in which a frequent, or even continuous, monitoring of an area of interest is required [16]. In military field, for example, a typical use of these systems regards the land monitoring for security purposes [14]. In fact, in many operative contexts, a large number of interesting areas has to be continually checked to detect the unexpected presence of people and vehicles, which can represent a possible danger [19,22]. Another typical example regards the frequent monitoring of strategic areas near military bases, refugee camps, and connecting roads, to detect the presence of objects (e.g., Improvised Explosive Devices, IEDs) that can threaten the crossing of humanitarian and military convoys. In civilian field, these systems can be suitably used for monitoring restricted areas after catastrophic events, such as earthquakes, tsunamis, damage to nuclear power plants, and so on. A prerequisite of many of the above introduced systems is the construction of a comprehensive view of an area of interest. The video sequence that represents the area is often acquired at low-altitude for several reasons, including the need to have a high spatial resolution for classifying objects [8,15], to safeguard the UAV, to hide the UAV, and many others.

This paper presents a small-scale UAV based system for real-time creation of incremental and geo-referenced mosaics of areas of interest acquired at low-altitude. The only input required by the system is a set of GPS coordinates that specifies one or more areas that have to be mosaicked. The proposed mosaicking algorithm presents several innovative contributions compared to the current state-of-the-art. First, to speed-up the feature extraction and matching processes, it adopts the A-KAZE extractor [1]. The recent literature [1–3] has shown that A-KAZE features are faster to compute than SIFT [13] and SURF [4], moreover they exhibit much better performance in detection and description than ORB [24]. Second, the mosaicking algorithm implements an automatic method to optimize the acquisition rate of the RGB camera based on the telemetry (i.e., speed and height). Third, to speed-up all steps involved in the stitching process, the mosaicking algorithm implements a ROI through which the computation required for the stitching of each new frame on the mosaic is reduced. Fourth, unlike the majority of the mosaicking algorithms known in literature that use RANSAC [7] to perform the geometric transformation stage, the proposed algorithm adopts the rigid transformation [18] that allows the building of mosaics at low-altitude mitigating in part the artifacts due to the parallax error [9]. Currently, public datasets for testing mosaicking algorithms contain video sequences acquired at high-altitude, for this reason we have implemented and made available the UAV Mosaicking and Change Detection (UMCD) dataset[1]. Instead, to test the algorithm at high-altitudes we have used the NPU Drone-Map dataset[2].

The rest of the paper is structured as follows. Section 2 presents some selected works near to that proposed. Section 3 introduces the architecture of the proposed mosaicking algorithm and discusses the different algorithmic choices, including the feature extraction by the A-KAZE extractor, the stitching process by the rigid transformation, and the implementation of the ROI. Section 4 reports

[1] http://www.umcd-dataset.net/.
[2] http://zhaoyong.adv-ci.com/downloads/npu-dronemap-dataset/.

the experimental results obtained by using both UMCD and NPU Drone-Map datasets. Finally, Sect. 5 concludes the paper.

2 Related Work

Regardless the specific size of the UAVs (e.g., small, medium, large), in the last years a wide range of tasks has been supported by their use, such as urban monitoring [23], vegetation analysis [17], surveillance [10], and others. Anyway, the pipeline of these systems is similar and includes specific main stages: extraction of salient points from frames (i.e., feature extraction), find image transformation values (i.e., rotation, scale, and translation), and merge frames together.

Several works in the literature produce a mosaic in off-line mode, i.e., when all the frames are available for the processing. Two examples are reported in [20] and [11], respectively, where the authors present a robust system that uses SIFT extractor and homography transformation based on RANSAC. Similar steps are used in [27], where the authors, first, utilize a transformation between frames based on an iterative threshold to find the edges and, subsequently, apply a correlation phase to merge them. From a performance point of view, the mosaicking of a high number of frames is a time-consuming duty that requires a wide availability of resources. A possible solution to this issue is reported in [21], where a fast algorithm using little amount of resources is presented. In particular, the proposed algorithm works by doing pairwise image registration, then it projects the resulting points to the ground and produces a new set of control points by moving these points closer to each other. Then, it fits image parameters to these new control points and repeats the process to convergence. Regarding the real-time processing, in [12] the authors use ORB as feature extractor and provide a spatial and temporal filter for removing the majority of the outlier points. In [28], the authors use SIFT as feature extractor and Euclidean distance with a threshold for matching the frames. Finally, in [26], the authors adopt an incremental technique and more UAVs to cover an area of interest and to build a qualitative mosaic. Inspired by several of these works, but unlike them, the proposed mosaicking algorithm uses A-KAZE and ROI to speed-up the stitching process. Moreover, the use of the rigid transformation allows to obtain mosaics whose video sequences are acquired at low-altitude.

3 The Mosaicking Algorithm

The logical architecture of the small-scale UAV based system and the pipeline of the proposed mosaicking algorithm are shown in Fig. 1. The algorithm consists of four main stages each of which is discussed below. The system is designed to work with standalone and client-server architectures. However, the latter is used to explain properly how the system works.

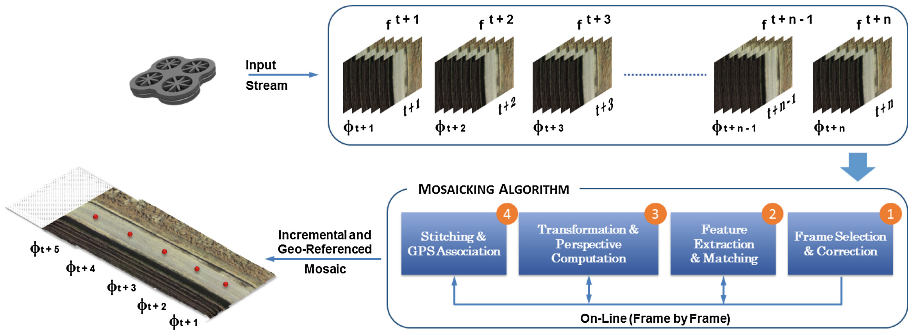

Fig. 1. The proposed mosaicking algorithm. f^t and ϕ_t are the frames and the linked GPS coordinates provided to the algorithm at each second t, respectively.

3.1 Background

In the following, let:

$$UAV_{path} = \{\phi_{t+i}(x_{t+i}, y_{t+i}) \mid t \in \mathbb{N} \wedge i \in [1, \ldots, n] \subset \mathbb{N}\} \tag{1}$$

be the set of GPS coordinates that defines the area of interest that needs to be mosaicked, where, t is the amount of seconds required by the UAV to reach the area, and n is the seconds of flight duration within the area. Besides, for each $i \in [1, \ldots, n]$, $\phi_{t+i}(x_{t+i}, y_{t+i})$ is the $t + i^{th}$ coordinate and (x_{t+i}, y_{t+i}) is the pair $(latitude, longitude)$. Without loss of generality, we can define ϕ_{start} and ϕ_{end} when $i = t + 1$ and $i = t + n$, respectively. In addition, let:

$$F_{TRS} = \{f^{t+i} \mid t \in \mathbb{N} \wedge i \in [1, \ldots, n] \subset \mathbb{N}\} \tag{2}$$

be the set of frames transmitted from the UAV to the processing unit (local or remote) within the UAV_{path}, where t and n are defined as above. For each $i \in [1, \ldots, n]$, $f^{t+i} = \{f_1^{t+i}, f_2^{t+i}, \ldots, f_{FPS}^{t+i}\}$ is the set of frames transmitted by the UAV at the second i. The set depends on frame per second (FPS) of the RGB camera. The UAV starts the transmission to the processing unit from the take-off up to the landing. In general, each second $k \in \mathbb{N}$ of transmission is composed of a GPS coordinate, $\phi_k(x_k, y_k)$, and a set of frames, $f^k = \{f_1^k, f_2^k, \ldots, f_{FPS}^k\}$.

3.2 Frame Selection and Correction

Since the aim of the algorithm is to build the mosaic of the area of interest defined by the UAV_{path}, all the frames transmitted outside of this path (i.e., $f^k \notin F_{TRS}$ for each $k \in \mathbb{N}$) are discarded by the processing unit. The rest of the frames transmitted by the UAV (i.e., $f^k \in F_{TRS}$ for each $k \in [t + 1, \ldots, t + n] \subset \mathbb{N}$) are used in part to create the mosaic, while the remaining are discarded again. This is due to the fact that at each second the UAV tends to transmit more frames than ones necessary to create a proper mosaic. The proposed algorithm implements a two-step approach to select a suitable number of frames. In the first, the system adopts the telemetry of the UAV. The main idea is that hight and speed of the UAV can derive the number of frames required to construct a mosaic without disjunctions (F_{STEP_1}). This step can be defined as follows:

$$F_{STEP_1} = f_{MAX} \frac{1}{(1+c)} \ with\ c = \frac{FoV * h}{v} \tag{3}$$

where, f_{MAX} is the FPS of the sensor, FoV (i.e., Field of View) is the width of the angle of view of the sensor expressed in degree, h is the flight height of the UAV expressed in meters and, finally, v is the speed of the UAV expressed in meters per second. The frames selected by the first step can be further thinned out by means of a user parameter (F_{STEP_2}) that defines the amount of overlap between a current frame and the linked part of the mosaic. Although it is not a focus of the paper, it should be observed that the system implements also a layer in which the calibration parameters of the camera can be stored to correct the possible lens-distortion introduced by the RGB camera.

3.3 Feature Extraction and Matching

Let M_j be the mosaic built up to the second j and let $\hat{f}_s^{j+1}$, with $j+1 \in [t+1, \ldots, t+n] \subset \mathbb{N}$ and $s \in [1, \ldots, FPS] \subset \mathbb{N}$, the current selected frame, at the second $j+1$, to be added to the mosaic. The main steps to built the new mosaic, $M_j \cup \hat{f}_s^{j+1}$, are the feature extraction and matching processes. In general, the features extracted from each current frame should be compared with those extracted from the whole mosaic to establish where the current frame has to be placed. Since the size of the mosaic grows over time, the comparison stage tends to become unmanageable after a certain period of time. With the aim to avoid such a issue, the proposed system uses a ROI to extract the features from the mosaic. The ROI tracks the last frame added to the mosaic and delimits, to a region surrounding it, the feature extraction process. A ROI centred on the last frame and sized three times than the size of a frame is sufficient to ensure the proper execution of the mosaicking algorithm. By the ROI the adding of a new frame takes a constant-time, no more dependent on the increasing size of the mosaic. Notice that the ROI concept is not new, but it is worth describing it due to the its effectiveness in increasing the system performance. The proposed algorithm uses A-KAZE, instead of the most popular extractors, such as SIFT, SURF or ORB. This is due to the fact that A-KAZE adopts both the Fast Explicit Diffusion (FED) embedded in a pyramidal framework and the Modified-Local Difference Binary (M-LDB) descriptor in order to speed-up feature detection in non-linear scale space and to exploit gradient information from the non-linear scale space, respectively. These aspects make A-KAZE an optimal compromise between speed and performance with respect to the current literature [1].

The keypoints extracted from M_j and $\hat{f}_s^{j+1}$ are used to detect the overlapping region between them. Let $\mathcal{X}_{M_j} = \{\alpha_1, \ldots, \alpha_h\}$ and $\mathcal{X}_{\hat{f}_s^{j+1}} = \{\beta_1, \ldots, \beta_t\}$ be the set of keypoints extracted by A-KAZE from M_j and $\hat{f}_s^{j+1}$, respectively. With the aim of finding the correspondence between the keypoints in $\mathcal{X}_{M_j}$ with those in $\mathcal{X}_{\hat{f}_s^{j+1}}$ a simple Brute Force Matcher (BFM) algorithm is used [25]. This algorithm performs an exhaustive search between the two sets of keypoints and matches only those keypoints that have an identical pattern (i.e., local structure of the pixels). Formally, at the end of the process, the algorithm generates two sub-sets $\hat{\mathcal{X}}_{M_j} = \{\alpha_{h_1}, \ldots, \alpha_{h_m}\} \subseteq \mathcal{X}_{M_j}$ and $\hat{\mathcal{X}}_{\hat{f}_s^{j+1}} = \{\beta_{t_1}, \ldots, \beta_{t_m}\} \subseteq \mathcal{X}_{\hat{f}_s^{j+1}}$ where for each $k \in \{h_1, \ldots, h_m\}$ exists a single $j \in \{t_1, \ldots, t_m\}$ such that $\alpha_k \equiv \beta_j$. As well-known, the two sub-sets have the same cardinality.

3.4 Transformation and Perspective Computation

Once obtained the corresponding keypoints (i.e., $\hat{\mathcal{X}}_{M_j}$ and $\hat{\mathcal{X}}_{\hat{f}_s^{j+1}}$) between the two frames, the system must compute the geometrical transformation by which the keypoints of the current frame, $\hat{f}_s^{j+1}$, are collimated with ones of the mosaic, $\mathcal{X}_{M_j}$, within the reference system of the latter. This transformation is subsequently used on each pixel of the frame to stitch it over the mosaic. In literature, the RANSAC algorithm to calculate the homography transformation is considered the reference approach. It consists in using the corresponding keypoints to iteratively estimate the parameters of a mathematical model by which to perform the geometric projection of each pixel between the two images. Despite this, as shown in Fig. 2a, the homography transformation can produce a high level of distortions especially when it is applied on images acquired a low-altitude. In particular, the mosaic can present an unreal curvature. This is due to the fact that the homography transformation matrix has 8 degrees of freedom, hence at least 4 corrected correspondences are required to build a proper mosaic. In the proposed mosaicking algorithm, the acquired images can be considered as a linear scanning of the ground surface, therefore a transformation with less degrees of freedom can be adopted. For this reason, the rigid transformation matrix that has only 4 degrees of freedom is implemented [18]. The reference example reported in Fig. 2b shows the goodness of the obtained results. The majority of the UAV based systems treat video sequences acquired at high altitude, or propose a orthorectification pre-processing step at the expense of the real-time

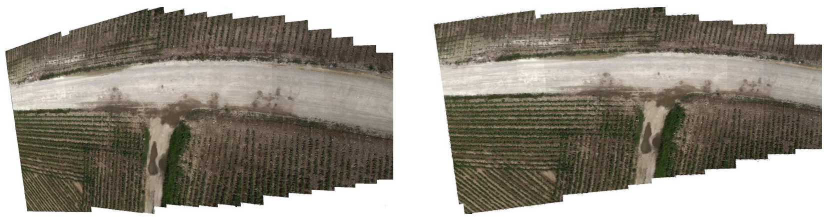
(a) (b)

Fig. 2. Geometric transformation: (a) homography transformation by RANSAC algorithm, (b) rigid transformation.

processing [29] thus avoiding this type of issue. The last step of the module is to merge the pixels of the mosaic, $\mathcal{X}_{M_j}$, with the transformed pixels of the frame, $\Gamma(\hat{f}_1^{j+s})$, to obtain a new pixel matrix, $\mathcal{X}_{M_j} \cup \Gamma(\hat{f}_s^{j+1})$.

3.5 Stitching and GPS Association

The acquisition of the GPS coordinates is performed following the NMEA[3] format, one of the most widespread standards for the transmission of position data. Current commercial GPS transmitters provide one or more position data per second, however in the latter case a good practice is to derive a single information per second to reduce the intrinsic error due to the acquisition process. Since the construction of a mosaic can require more frames per second, this means that only the first of the n frames for second acquired by the RGB camera is associated to a GPS coordinate, the rest of the $n-1$ frames, if added to the mosaic, has to be associated to coordinates inferred by ones previously acquired. Actually, once obtained two coordinates of the first frame of two consecutive seconds, then the coordinates of the remaining frames of the first second can be derived by adopting a simple linear interpolation. Let $\phi_j(x_j, y_j)$ and $\phi_{j+1}(x_{j+1}, y_{j+1})$ be the GPS coordinates acquired and associated with the frames $\hat{f}_1^j$ and $\hat{f}_1^{j+1}$, respectively ($s = 1$ in both cases since they are the first frames of each second). In addition, considering $\hat{f}_1^j$ belonging to the mosaic M_j and $\hat{f}_1^{j+1}$ the current frame. Then, the coordinate of any frame added to the mosaic between them can be derived as follows:

$$x_k = x_j + \frac{k}{FPS}(x_{j+1} - x_j), \; y_k = y_j + \frac{k}{FPS}(y_{j+1} - y_j) \tag{4}$$

where, x_k and y_k are the interpolated *latitude* and *longitude*, respectively, of the new GPS coordinate $\phi_k(x_k, y_k)$ associated to the frame $\hat{f}_k^j$. Moreover, k specifies the coordinate of which frame needs to be computed, finally, FPS is the frames per second of the sensor. The current version of the system performs the mosaicking algorithm in on-line mode. This means that when the system acquires a new GPS coordinate, it also considers the previous acquired one, computes the interpolation process and associates the interpolated coordinates to the linked frames within the mosaic. Each GPS coordinate (acquired or interpolated) is anchored to the barycentre of the linked frame. This last is a main aspect to enable the system with a wide range of tasks. Once that the GPS coordinate has been linked to the new frame, the gain compensation between this latter and the mosaic is performed by using the multi-band blending [5]. This assures that there will be no seams when the new frame is added to the current mosaic.

4 Experimental Results and Discussion

For testing the mosaicking algorithm, two recent public datasets were used. The first is the UMCD dataset, that contains a collection of aerial video sequences

[3] http://www.nmea.org/.

acquired at low-altitudes. The second is the NPU Drone-Map dataset, that contains a collection of aerial video sequences acquired at high-altitude. In both cases, the sequences are acquired by small-scale UAVs. Regarding the first dataset, we tested the algorithm on 40 challenging video sequences and measured the quality of the obtained mosaics by a simple metric based on the difference between image regions. Regarding the second dataset, we compared the proposed mosaicking algorithm with that presented in [6]. The latter is one of the few works in the literature that makes available source code, video sequences (i.e., the NPU Drone-Map dataset), and obtained mosaics to support a concrete comparison with other approaches. In particular, 4 challenging video sequences were selected from the second dataset and a correlation measure was adopted to quantify the similarity between mosaics pairs.

4.1 Low-Altitude and High-Altitude Mosaicking

In this sub-section, key considerations about the quality of the obtained mosaics are reported and discussed. Regarding the low-altitude, the adopted 40 video sequences had an average acquisition height of about 15 meters. In Fig. 3a an example is shown. In order to measure the quality of the mosaics derived by these video sequences the image difference process presented in [3] is adopted. The main idea is that each part of the mosaic must have the same spatial and colour resolution with respect to the original frames that have generated it. For

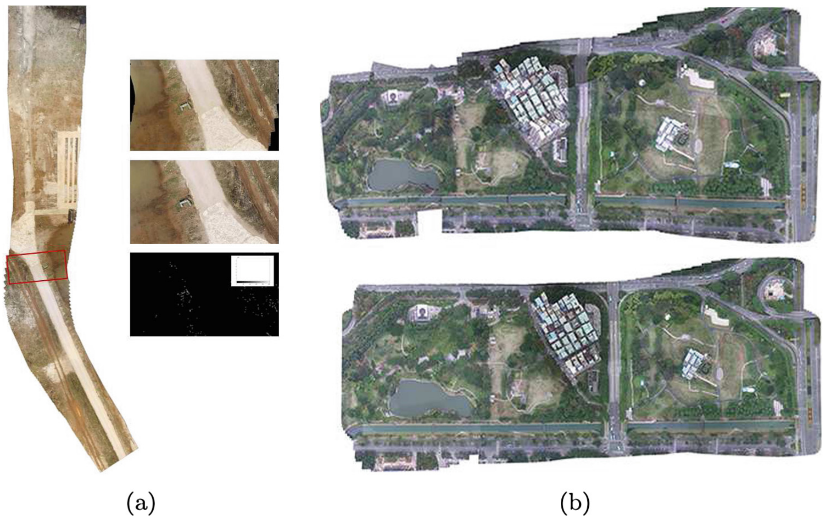

Fig. 3. Experimental results: (a) example of mosaic at low-altitude. The three miniatures are the frame extracted from the mosaic (up), one of the original frames used to build the mosaic (middle), the difference between the overlapped regions (bottom), (b) examples of mosaics at high-altitude by the proposed method (up), the method proposed in [6] (bottom).

this reason, the difference between each portion of the mosaic and the linked original frames is computed. Subsequently, a simple histogram is calculated on each image difference to evaluate the degree of deviation. Anyway, this simple but effective process has shown that each part of the mosaic generated by the proposed method is quite similar to the linked frames. On average, the difference images shown a deviation of about 15%. This can be considered a real good result taking into account all the geometrical distortion and error propagations that occur during the complex mosaicking process. Moreover, it should be considered that the incremental real-time mosaicking process at low-altitude is a topic that needs to be further investigated. By the implemented UMCD dataset and the provided results, the aim is to provide a concrete first contribute for the comparison of these algorithms. In Fig. 3b, examples of high-altitude mosaics are shown. In particular, the mosaic on the top of the Fig. 3b is generated with the proposed approach, while the mosaic on the bottom is generated with the method proposed in [6]. Both mosaics were created by using the same video sequence contained in the NPU Drone-Map dataset (named: phantom3-centralPark). How it is possible to observe, some visual differences are present. This is due to the fact that the proposed method applies only basic transformations, such as translation, rotation, and scale change, while the method with which we compare performs the orthorectification of the frames. Despite this, the degree of correlation between the two types of mosaic is impressive. To verify the similarity between them the following metric was adopted:

$$corr = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \tag{5}$$

where A, B are the two mosaics, and $\bar{A}$, $\bar{B}$ are the means of the mosaics pixels. On average, considering all the 4 video sequences reported in Table 1, we obtained a correlation value of about 80% among the mosaics. It should be considered that due to the different image processing, such as geometric transformation, orthorectification, stitching, and so on, it is not possible to obtain a perfect overlap between the mosaics. In particular, the different perspectives of the obtained mosaics are seen as significant differences by the metric. In any case, the degree of correlation can be considered a very high value.

Table 1. Time needed for generating the mosaics. The unit is in minutes.

Sequence	Frames	KFs	Proposed	Bu et al. [6]	Pix4D	Photoscan
phantom3-npu	19,983	457	7.2	9.32	140.08	538.38
phantom3-centralPark	12,744	471	6.01	8.49	127.73	563.57
phantom3-village	16,969	406	10.4	11.31	132.07	360.70
phantom3-huangqi	14,776	393	8	10.36	102.83	462.32

4.2 Mosaicking Performance

In this sub-section, the performance of the proposed method is presented. All the experiments were performed on a laptop equipped with an Intel i7 6700HQ CPU, 16 GB DDR3 RAM and a nVidia GTX960 GPU. In Table 1, the time needed for generating the mosaics is reported. More specifically, we compared the proposed method with the algorithm reported in [6] and with two commercial software, Pix4D[4] and Photoscan[5], also reported in the same work. The proposed method stitches 1 frame per second, while the method proposed in [6] requires the stitching of 10 frames per second. Both Pix4D and Photoscan, instead, use only the keyframes to produce the final mosaic (i.e., similar to the proposed algorithm). Since all methods, with the exception of that proposed, use the GPU, a resize to the half of HD resolution (i.e., the original size of the frames) to be stitched is performed. In Table 1, the comparison is shown. As it is possible to observe, both the proposed and [6] algorithms take much less time than the commercial software. The proposed method show low processing times even with respect to the work proposed [6] and the generated mosaics by the two approaches result quite similar. Anyway, we are currently developing an approach to perform the orthorectification frame by frame.

5 Conclusions

This paper propose a small-scale UAV based system for the real-time creation of incremental and geo-referenced mosaics of video streams acquired at low-altitude. The system presents several innovative contributions, including the use of A-KAZE feature extractor in aerial images, a ROI to speed-up the stitching stage, as well as the use of the rigid transformation to build a mosaic at low-altitude mitigating in part the artifacts due to the parallax error. We implemented the UMCD dataset and used the NPU Drone-Map dataset to test the algorithm at low-altitude and high-altitude, respectively. The adopted metrics have shown remarkable results, in time and quality, compared with selected solutions of the current state-of-the-art.

Acknowledgments. This work was partially supported by both the “Proactive Vision for advanced UAV systems for the protection of mobile units, control of territory and environmental prevention (SUPReME)” FVG L.R. 20/2015 project and the “Augmented Reality for Mobile Applications: advanced visualization of points of interest in touristic areas and intelligent recognition of people and vehicles in complex areas (RA2M)” project.

References

1. Alcantarilla, P., Nuevo, J., Bartoli, A.: Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: BMVC, pp. 1–11 (2013)

[4] https://pix4d.com/.
[5] http://www.agisoft.com/.

2. Avola, D., Foresti, G.L., Cinque, L., Massaroni, C., Vitale, G., Lombardi, L.: A multipurpose autonomous robot for target recognition in unknown environments. In: INDIN, pp. 766–771 (2016)
3. Avola, D., Cinque, L., Foresti, G.L., Massaroni, C., Pannone, D.: A keypoint-based method for background modeling and foreground detection using a PTZ camera. In: PRL (2016)
4. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (SURF). CVIU **110**(3), 346–359 (2008)
5. Brown, M., Lowe, D.G.: Automatic panoramic image stitching using invariant features. IJCV **74**(1), 59–73 (2007)
6. Bu, S., Zhao, Y., Wan, G., Liu, Z.: Map2dfusion: real-time incremental UAV image mosaicing based on monocular slam. In: IROS, pp. 4564–4571 (2016)
7. 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)
8. García, J., Martinel, N., Gardel, A., Bravo, I., Foresti, G.L., Micheloni, C.: Modeling feature distances by orientation driven classifiers for person re-identification. JVCIR **38**, 115–129 (2016)
9. He, B., Yu, S.: Parallax-robust surveillance video stitching. Sensors **16**(1), 1–12 (2015)
10. Heikkilä, M., Pietikäinen, M.: An image mosaicing module for wide-area surveillance. In: VSSN, pp. 11–18 (2005)
11. Javadi, M.S., Kadim, Z., Woon, H.H.: Design and implementation of automatic aerial mapping system using unmanned aerial vehicle imagery. In: CGMIP, pp. 91–100 (2014)
12. Li, J., Yang, T., Yu, J., Lu, Z., Lu, P., Jia, X., Chen, W.: Fast aerial video stitching. IJARS **11**, 1–11 (2014)
13. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)
14. Martinel, N., Avola, D., Piciarelli, C., Micheloni, C., Vernier, M., Cinque, L., Foresti, G.L.: Selection of temporal features for event detection in smart security. In: Murino, V., Puppo, E. (eds.) ICIAP 2015. LNCS, vol. 9280, pp. 609–619. Springer, Cham (2015). doi:10.1007/978-3-319-23234-8_56
15. Martinel, N., Micheloni, C., Foresti, G.L.: The evolution of neural learning systems: a novel architecture combining the strengths of NTs, CNNs, and ELMs. IEEE Syst. Man Cybern. Mag. **1**(3), 17–26 (2015)
16. Martinel, N., Micheloni, C., Piciarelli, C.: Pre-emptive camera activation for video surveillance HCI. In: ICIAP, pp. 189–198 (2011)
17. Michener, W.K., Houhoulis, P.F.: Detection of vegetation changes associated with extensive flooding in a forested ecosystem. Photogram. Eng. Remote Sens. **63**(12), 1363–1374 (1997)
18. Ngo, P., Passat, N., Kenmochi, Y., Talbot, H.: Topology-preserving rigid transformation of 2D digital images. IEEE TIP **23**(2), 885–897 (2014)
19. Piciarelli, C., Micheloni, C., Foresti, G.L.: Occlusion-aware multiple camera reconfiguration. In: ICDSC, pp. 88–94 (2010)
20. Piciarelli, C., Micheloni, C., Martinel, N., Vernier, M., Foresti, G.L.: Outdoor environment monitoring with unmanned aerial vehicles. In: Petrosino, A. (ed.) ICIAP 2013. LNCS, vol. 8157, pp. 279–287. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41184-7_29
21. Pritt, M.: Fast orthorectified mosaics of thousands of aerial photographs from small UAVs. In: AIPR, pp. 1–8 (2014)

22. Remagnino, P., Velastin, S.A., Foresti, G.L., Trivedi, M.: Novel concepts and challenges for the next generation of video surveillance systems. MVA **18**(3), 135–137 (2007)
23. Ridd, M.K., Liu, J.: A comparison of four algorithms for change detection in an urban environment. Remote Sens. Environ. **63**(2), 95–100 (1998)
24. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: ICCV, pp. 2564–2571 (2011)
25. Song, B.C., Kim, M.J., Ra, J.B.: A fast multiresolution feature matching algorithm for exhaustive search in large image databases. IEEE TCSVT **11**(5), 673–678 (2001)
26. Wischounig-Strucl, D., Rinner, B.: Resource aware and incremental mosaics of wide areas from small-scale uavs. MVA **26**(7), 885–904 (2015)
27. Yang, X.H., Xin, S.X., Jie, Z.Q., Jing, X., Dan, Z.D.: The UAV image mosaic method based on phase correlation. In: ICCA, pp. 1387–1392 (2014)
28. Yang, Y., Sun, G., Zhao, D., Peng, B.: A real time mosaic method for remote sensing video images from UAV. J. Sig. Inf. Process. **4**(3B), 168–172 (2013)
29. Zhou, G.: Near real-time orthorectification and mosaic of small UAV video flow for time-critical event response. IEEE TGRS **47**(3), 739–747 (2009)