Map2DFusion: Real-time Incremental UAV Image Mosaicing based on Monocular SLAM

Shuhui Bu[1], Yong Zhao[1], Gang Wan[2], and Zhenbao Liu[1]

***Abstract*— In this paper we present a real-time approach to stitch large-scale aerial images incrementally. A monocular SLAM system is used to estimate camera position and attitude, and meanwhile 3D point cloud map is generated. When GPS information is available, the estimated trajectory is transformed to WGS84 coordinates after time synchronized automatically. Therefore, the output orthoimage retains global coordinates without ground control points. The final image is fused and visualized instantaneously with a proposed adaptive weighted multiband algorithm. To evaluate the effectiveness of the proposed method, we create a publicly available aerial image dataset with sequences of different environments. The experimental results demonstrate that our system is able to achieve high efficiency and quality compared to state-of-the-art methods. In addition, we share the code on the website with detailed introduction and results.**

***Index Terms*— UAVs, real-time, SLAM, stitching, adaptive weighted multiband**

I. Introduction

In recent years, Unmanned Aerial Vehicles (UAVs) have been more and more popular because they are able to accomplish aeronautic tasks at a low price. They are now used for coordinated urban surveillance [1], forest fire localization during monitoring [2], [3], object-based change detection [4], [5], and so on. A real-time global orthoimage can help missions such as fire monitoring and urban surveillance where both efficiency and accuracy are required.

A mature choice for quick orthoimage generation is Structure from Motion (SfM), which is able to explore the most information from the multi-view images. There are lots of implementations of SfM algorithm such as open-source projects OpenMVG [6], openDroneMap, and commercial software Pix4DMapper [7], [8], Photoscan [9]. They share almost the same pipeline: feature detection and matching, image alignment and sparse point cloud generation with bundle algorithm [10], dense point-cloud and mesh generation, texture and orthoimage mosaicing. While SfM methods always take hours to generate the final orthoimage and all images need to be prepared before computation, they are not suitable for real-time and incremental usage.

Since the expensive computation cost of SfM, several 2D methods are proposed in the past years for orthoimage. Niethammer et al. [5] show a straightforward approach for landslide surface displacements evaluation where images are warped onto the plane by matching the ground control points (GCPs) and the final result is blended with software OrthoVista[1]. Hirokawa et al. [11] use GCPs to calibrate the position and attitude of the UAV, while pictures are compensated to a spatial temporal GIS with UAV attitude information later. Homography transformation based on feature matching is also used for image registration [12], [13], where images are warped to the reference frame instead of the GCPs plane. GCPs methods seem tedious for quick response circumstances and 2D homography methods are hard to meet large-scale mosaicing requirements.

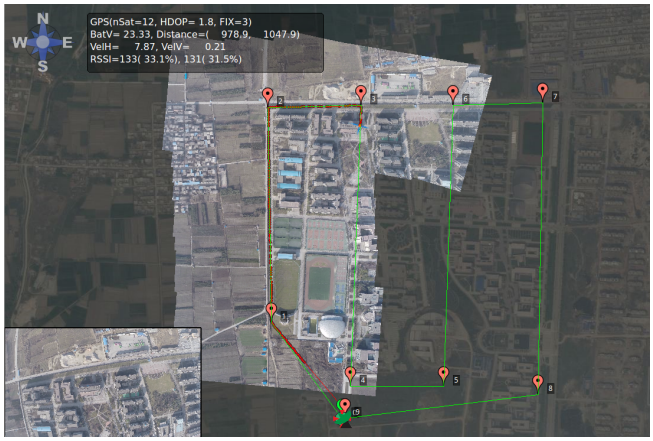


Fig. 1. We integrate Map2DFusion into the self-developed ground control station, where the orthoimage is generated and visualized online during the flight. The figure shows that the orthoimage is well aligned to Bing satellite map automatically after GPS trajectory fitted.

Simultaneous localization and mapping (SLAM), one of the hot researches in robotics and computer vision community, is considered to be the key technique for automatic navigation in unknown environments. The SLAM approach has the advantage of real-time performance due to the well designed processing flow. Some researches are conducted to utilize SLAM for real-time spherical mosaic [14] and underwater sonar image stitching [15]. Since [14] is limited to pure rotation situation and [15] lacks a proper blending algorithm which leads to poor visualization, they both do not meet the demand of aerial images stitching. In this work, a visual SLAM and an adaptive weighted multiband algorithm are designed to stitch the captured images online in real-time as demonstrated in Fig. 1. Because the proposed method has merits of real-time and fully automatic properties, it has a good prospect on applications such as real-time surveillance, GPS denied environment navigation

[1] Shuhui Bu, Yong Zhao, and Zhenbao Liu are with Northwestern Polytechnical University, 710072 Xi'an, China bushuhui@nwpu.edu.cn, zd5945@126.com

[2]Gang Wan is with Information Engineering University, Zhengzhou, China

[1]http://www.orthovista.com/

for UAV or drone cluster, forest fire localization during monitoring, and so on. In brief, the main contributions are concluded as follows:

1) **Automatic GPS and video synchronization**: As most video streams are not synchronized with GPS, a graph based optimization is proposed to synchronize video time with GPS time from coarse to fine. Consequently, our method is able to obtain WGS84 coordinates and conquer accumulated drift by fusing GPS information into SLAM processing without GCPs required.
2) **Real-time orthoimage blender**: We propose an adaptive weighted multi-band method to blend and visualize images incrementally in real-time. Compared to SfM methods, the approach obtains higher efficiency and satisfactory results without holes and twists caused by limited viewing angles.
3) **A public aerial image dataset**: We create an aerial image dataset with several different sequences for visual SLAM, orthoimage, and 3D reconstruction evaluation.

For better understanding of our work, not only the source code and dataset, but also introduction video, detailed results, and evaluations are publicly available on the project website[2].

II. Related Work

Many works have been conducted to generate orthoimage from aerial images in recent years [5], [11], [13]. Most methods can not output acceptable results at real-time speed, while the proposed approach is able to generate high-quality orthoimage incrementally from aerial images. The two related key techniques of our work are monocular SLAM and image stitching, which are summarized in below.

SLAM is regarded as the core to realize navigation and environment sensing for fully autonomous robotics. Parallel Tracking and Mapping (PTAM) [16] is one of the most famous monocular SLAM implementations which shows high efficiency but is limited to small workspaces. As most methods detect keypoint features for matching, Newcombe et al. propose a direct approach named DTAM [17] where camera motion is computed by minimizing a global spatially regularized energy function on GPU. To reduce the intensively computational demand of DTAM, Forster et al. propose a semi-direct monocular visual odometry (SVO) [18] which only processes strong gradient pixels and brings high frame-rates. While all these methods only locally track camera motion and do not have loop-closures, Engel et al. show a well-known large-scale direct monocular SLAM (LSD-SLAM) [19]. The approach utilizes FabMap [20] for loop detection and a similarity transform pose graph for optimization, which allows to build consistent, large-scale maps of the environment. It is extended for omni-directional and stereo cameras [21], [22]. The direct methods rely on small baselines and seem lack robustness when frame-rate is not high enough. Mur-Artal et al. propose a novel ORB feature based monocular SLAM algorithm (ORB-SLAM) [23], which is able to handle large baselines robustly with high precision. It is also further explored for semi-dense reconstruction and gets even better results than direct methods [24].

Another critical part of our work is image stitching, which has reached a stage of maturity where there are now an abundance of tools especially for panoramic image stitching, including OpenCV functions, the well-known software Photoshop, web-based photo organization tools like Microsoft Photosynth, smartphone apps like Autostitch [25], as well as the built-in image stitching functionality on off-the-shelf digital cameras. Many efforts are conducted to obtain better alignment and blending. Zaragoza et al. [26] improve the image alignment stage with Moving Direct Linear Transformation (DLT) which produces better alignment when the scene is not planar and camera has both rotation and translation. However, perfect alignment is still difficult to be obtained for real world situations, so that some seam cutting methods [27], [28] and blending algorithms [29], [30] are used to minimize the visible seams.

In this paper, we present a novel approach which generates high quality orthoimage incrementally with real-time speed. Different from traditional image stitching methods which just compute homograph transformation, in the proposed method, the camera position and attitude are estimated through well designed SLAM system, therefore, a more reasonable plane and precise camera poses are obtained. In order to handle large-scale scenes, GPS information is fused in our SLAM system to obtain WGS84 coordinates and reduce tracking drift. The loop closure technique is also used to further minimize the drift especially when GPS information is absented. Most image stitching methods perform blending after all images are prepared, which reduces the overall efficiency. To achieve real-time speed, an incremental adaptive weighted multi-band is proposed to blend SLAM keyframes efficiently. In the blending, Laplacian pyramids are stored in grids which brings high performance and reduces the memory consumption. Thereby, resulting orthoimage can be stitched incrementally and visualized immediately.

III. Real-time Incremental UAV Image Mosaicing

The overview of our system is depicted in Fig. 2 which contains five main parts running in separated threads: video preparation, tracking, mapping, map optimization, and map fusion. The outline of each part is briefly introduced as follows:

1) The images are undistorted to an ideal pinhole model, and keypoints with features are extracted for tracking usage.
2) The initial motion of current frame is tracked against the last frame and then aligned with the local sub-map precisely.
3) New keyframe will be added to the global map when the distance between current frame and the last keyframe exceeds a certain threshold, and then a local bundle is performed to optimize the local map-points and keyframe poses.

[2]http://zhaoyong.adv-ci.com/map2dfusion

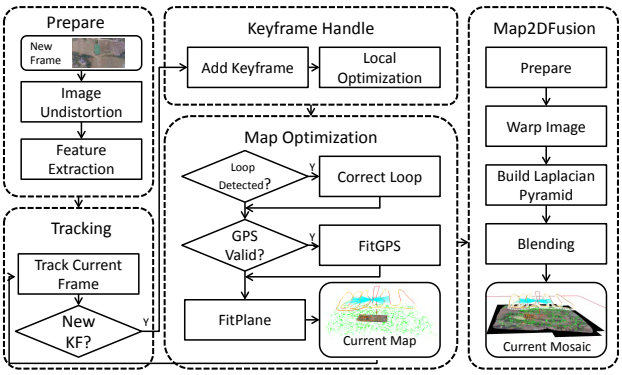


Fig. 2. The framework overview of our system.

4) The global optimization is taken to detect and close loops so that a consistent map can be obtained. An online calibration of time difference between video and GPS timestamp is performed from coarse to fine with a $SIM(3)$ pose graph, and the final similar transformation is applied to the estimated trajectory and map-points. After that, GPS constraints are also considered in the local bundle optimization of mapping thread and the mosaic plane is fitted with sampled map-points.
5) The map fusion unit blends keyframes into the mosaic plane incrementally.

In this section, a brief notation of transformation and projection is firstly given, and then we will introduce the implementation details of our system.

A. Notations

In order to make reader easily understand the representation of transformation and projection, the notations are briefly introduced here. The camera pose is represented with a twist coordinate representation ξ, which is given as a member of the Lie group $SE(3)$ [31]:

$$\mu = (v_1, v_2, v_3, q_1, q_2, q_3)^T \in \mathbb{R}^6, \tag{1}$$

where v_1, v_2, v_3 represent translation and q_1, q_2, q_3 are the angular positions corresponding to rotation matrix $\mathbf{R}$, $SE(3)$ denotes the 3D rigid transformations. The quarternion representation rx, ry, rz, w or matrix representation $\mathbf{R}$ of rotation are alculated from q_1, q_2, q_3 using the exponential function according to Lie algebra [31]. A left-multiplication is defined to transform a point $\mathbf{P} = (X, Y, Z)^T$ in the world coordinate to the camera coordinates:

$$\mathbf{P}' = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t} = exp(\mu)\mathbf{P}. \tag{2}$$

A standard pinhole camera model is used to represent the projection:

$$\mathbf{p} = Proj(\mathbf{P}_C) = \left(\frac{X_C f_x}{Z_C} + x_0, \frac{Y_C f_y}{Z_C} + y_0 \right)^T. \tag{3}$$

Here f_x, f_y are the focal lengths and x_0, y_0 are the coordinates of the camera center in the standard pinhole camera model.

On the contrary, the inverse project function $Proj^{-1}$ is used to compute corresponding position of $\mathbf{P}_C$ in the $Z_C = 1$ plane:

$$\mathbf{P}_C = Proj^{-1}(\mathbf{p}) = \left(\frac{x - x_0}{f_x}, \frac{y - y_0}{f_y}, 1 \right)^T. \tag{4}$$

B. VideoFrame Preparation and Tracking

To ease the heavy burdens of tracking thread, preparation works like image undistortion and feature extraction are performed in a separated thread. The oriented FAST and rotated BRIEF (ORB) [32] detector and descriptor are selected due to the satisfactory performance.

A map constructed with several keyframes $\mathbf{K}$ and map-points $\mathbf{P}$ is initialized automatically with the first two keyframes K_1, K_2. After the map is initialized, the pose μ of the current image is obtained from several pairs of 3D map-points $\mathbf{P}$ and 2D keypoints $\mathbf{p}$, which is called the perspective-n-point (PnP) problem. The corrospondences between map-points and keypoints are found with a similar strategy of the ORB-SLAM [23]. A window searching between current frame and the last keyframe K_{i-1} is applied to collect tens matches followed with a coarse pose determination. All keyframes observing these map-points are collected to construct the local sub-map, and more matches are found by projecting all local map-points with the coarse pose.

Given a set of matches between 3D map-points $\mathbf{P}$ and current keypoints $\mathbf{p}$, the camera pose μ can be estimated by minimizing the weighted squared residual function:

$$\mu^* = \underset{\mu}{argmin} \sum_{p_i \in \mathbf{p}} w_i \mathbf{e}_i^T \Sigma_i^{-1} \mathbf{e}_i. \tag{5}$$

The error $\mathbf{e}_i$ is defined as the difference between measurements $\mathbf{p}_i = (x_i, y_i)^T$ and predicted $\mathbf{p'}_i$ of a map-point $\mathbf{P}_i$:

$$\mathbf{e}_i = \mathbf{p}_i - \mathbf{p'}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - Proj(exp(\mu)\mathbf{P_i}). \tag{6}$$

The weight w_i is given to reduce influence of large residuals with a robust weighted function and information matrix Σ_i^{-1} is associated with the pyramid level of $\mathbf{p}_i$.

The relative distance between two frames is curved by a weighted combination of translation and rotation described in LSD-SLAM [19]:

$$dist(i, j) = \mu_{ji}^T \mathbf{W} \mu_{ji}, \tag{7}$$

here $\mathbf{W}$ is a diagonal matrix with different weights for each parameter in μ_{ji}, and the translation weights are scaled according to the mean inverse depth. Once the distance between current frame and the last keyframe K_{i-1} exceeds the fixed threshold, a new keyframe K_i will be created to ensure enough overlaps between keyframes.

C. Keyframe Handle and Local Optimization

When a keyframe K_i is inserted, new map-points need to be added to the map as soon as possible so that enough correspondences are able to be matched for tracking. For all keypoints are not observed in tracking, we try to

find their correspondences in neighbor keyframes along the epipolar line. A bag-of-words (BoW) vector is computed and registered to fasten the search, and new map-points are triangulated from matches. After new map-points are created, a data association step is performed to add more observations, combine redundancy map-points, and remove bad map-points.

A local bundle is executed to refine the local sub-map consisted with local keyframes K_L and map-points P_L seen by K_L. The keyframes' poses μ and map-points' positions $\mathbf{P}$ are optimized by minimizing the weighted residuals from all local observations:

$$\{\mu^*, \mathbf{P}^*\} = \underset{\mu,\mathbf{P}}{argmin} \sum_{P_i \in P_L} \sum_{K_j \in K_L} w_{ij} \mathbf{e}_{ij}^T \Sigma_{ij}^{-1} \mathbf{e}_{ij} + \sum_{K_j \in K_L} \gamma(j)^T \Lambda_j^{-1} \gamma(j), \quad (8)$$

where the second item of above equation denotes the GPS constraint which will be described in section III-E. In our implementation, the optimization is performed by G2O framework [33].

D. Loop Detection and Fusion

For generating high-quality map, it is necessary to obtain high quality trajectory, but tracking drift is inevitable in visual SLAM which may lead to intolerable misalignment error over long journey accumulation. Although the accumulated drift can be jointly optimized by fusing GPS information which will be described in next sub-section, it is preferable to perform loop closure and global optimization to achieve better consistency in large-scale scenes.

The first step is detecting appropriate keyframe which is closest to the current position. Because the estimated position has large drift, therefore, it is not easy to estimate the cross point of the loop directly. Recently, appearance based SLAM method such as FabMap [20] gives a vision-based solution. In this work, DBoW2 [34] package is adopted for loop detection. All keyframes sharing visual words with current keyframe are collected and the similarity scores are computed. Then the candidates are sorted according to their visual similarities. In order to gain robustness, they are chosen to be loop candidates only if its neighbors are also candidates. For all loop candidates, we find their correspondences with current keyframe and compute a coarse similarity transformation through the RANSAC method. The candidate which performs the best fitting with enough inliers, is accepted as the loop keyframe, and then the relative pose is optimized by minimizing sum of projection error of map-points.

After the loop is detected, a data association step is firstly performed to fuse duplicated map-points. Since direct bundle adjust approach takes too much time to do optimization, a pose graph optimization described in [35] is utilized to decrease the drifts of key-frames and refine corresponding map-points. This approach retains preciseness which benefits from local optimization with little computation.

E. GPS and Plane Fitting

In our experiments, UAVs are equipped with GPS and the trajectory with timestamp is available for most sequences. However, the video is not synchronized with GPS and a constant time difference needs to be calibrated for each sequence. In our system, the time difference t_{vg} is used to build correspondences between global positions and camera positions, and a similarity transformation S_{vg} is adopted to transform the whole map to WGS84 coordinates. A robust and efficient algorithm is proposed to jointly optimize the time difference t_{vg} and the similarity transformation S_{vg} from coarse to fine. For keyframe K_j whose timestamp in the video is t_j and translation in the map is $\mathbf{P}_{t_j}^{map}$, its corresponding timestamp for GPS should be $t_g = t_j + t_{vg}$ with WGS84 coordinates $\mathbf{S}_{vg}\mathbf{P}_{t_j}^{map}$ after transformation. The goal is to find the best t_{vg} and $\mathbf{S}_{vg}$ by minimizing the following squared sum of residuals for all keyframes $\mathbf{K}$:

$$\gamma(j) = \mathbf{P}_{t_g}^{gps} - \mathbf{S}_{vg}\mathbf{P}_{t_j}^{map} = \mathbf{P}_{t_j+t_{vg}}^{gps} - \mathbf{S}_{vg}\mathbf{P}_{t_j}^{map} \quad (9)$$

$$\{t_{vg}^*, \mathbf{S}_{vg}^*\} = \underset{t_{vg},\mathbf{S}_{vg}}{argmin} \sum_{K_j \in \mathbf{K}} \gamma(j)^T \Lambda_j^{-1} \gamma(j), \quad (10)$$

where Λ_j^{-1} controls the weight for each component in global coordinates. To estimate t_{vg} and $\mathbf{S}_{vg}$ more robustly, a coarse calibration is firstly performed to narrow the time shift range and obtain an initial $\mathbf{S}_{vg}$. Generally the time shift won't be too large and is assumed to be less than 60 seconds. Consequently, the time shift range is divided into several slices, for each $t_{vg} \in [-60, 60]$ is used to compute $\mathbf{S}_{vg}$ with a fast $SIM(3)$ solver [36]. The best t_{vg}^* and $\mathbf{S}_{vg}^*$ are chosen to be the initial values and precise results are optimized with the G2O framework [33]. After the t_{vg} is calibrated, $\mathbf{S}_{vg}$ is applied to the map-points and keyframes. GPS constraint used in formula (8) is considered during the local optimization to eliminate tracking drift.

Since a ground plane is necessary for 2D map mosaicing and visualization, a plane is fitted from sampled map-points with a RANSAC approach.

F. Map Fusion

In the previous sections, images are correctly aligned and a mosaic plane is fitted, in this section, our goal is to stitch the keyframes together incrementally. Unlike traditional panoramic stitching circumstances, challenges in the instantaneous aerial image mosaicing exist as follows:

1) It must be able to achieve real-time speed.
2) The stitching needs to be incremental for both blending and visualization.
3) Most scenes are not totally planar and the camera contains not only rotation but also translation, which leads to inevitable misalignments.
4) The mosaicing result should be as ortho as possible and able to be updated properly.

A seam cutting step is generally performed before blending to obtain better mosaic, however, most seam cutting methods [27], [28] are computational expensive for instantaneous applications and they also discard the weights and

need masks for all images prepared. In our system, the orthomosaic is split up into rectangular patches and a seam is generated naturally during the blending with a proposed weighted multi-band algorithm. For each patch, a Laplacian pyramid and a weight pyramid are stored. With a Laplacian pyramid, the exposure differences and misalignments are minimized, and brightness are mixed to be continuous, while all details are reserved in low levels of the pyramid. The Laplacian pyramid L_l for level l can be computed with an expanded operation [29]. To fasten the operation, a k level Gaussian pyramid is first computed and each level G_l is subtracted from the lower level G_{l-1} of the pyramid:

$$G_l(x,y) = \sum_{d_x=-n}^{n}\sum_{d_y=-n}^{n} w_{d_x,d_y} G_{l-1}(x+d_x, y+d_y), \quad (11)$$

$$L_l = G_l - G_{l+1} \qquad l < k. \quad (12)$$

Here $2n-1$ is the Laplacian kernel size and the sum of weights w_{d_x,d_y} should be one. The highest level L_k just equals G_k since there is no higher level G_{k+1} computed. The default band number k is 5 and the patch resolution is multiple of 2^k, where a 256×256 resolution is used in our system. The patches not only make the blending incremental, but also ensure the warped images in a proper size.

The pipeline to stitch a frame is concluded as follows:

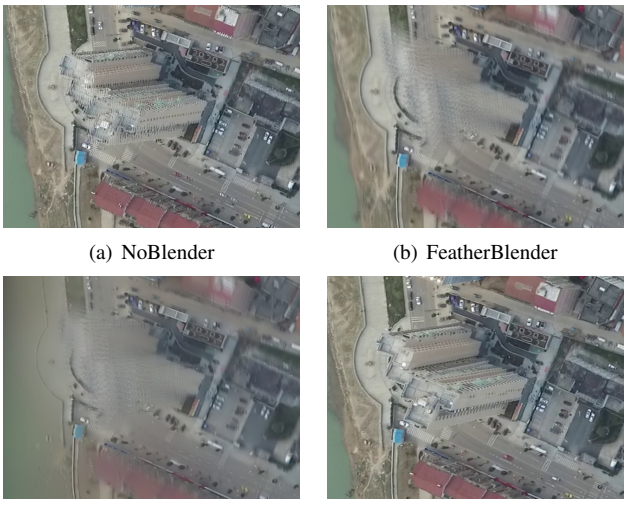
1) The rectangular margins are obtained according to the relative pose and the mosaic area grows when the margins exceed the existed area.
2) To make the mosaic as ortho and precise as possible, a weighted image is adaptively computed considering the height, view angle and pixel localization. The homography matrix is computed and applied to warp the color image and weighted image.
3) A Laplacian pyramid is expanded from the warped image and the corresponding weight pyramid is computed. Instead of the weighted summation, the results are fused to the global mosaic patches with the best weighted value.

The original image is recovered with a sum of the Laplacian pyramid $G_0 = \sum_{l=0}^{k} L_l$. It should be noted that the neighbor patches need to be considered to achieve better consistency. A brief comparison with the feather and multi-band blenders implemented by OpenCV is conducted and the results are illustrated in Fig. 3. Both feather and multi-band fail to handle the misalignments caused by the high buildings which are even unrecognizable since the blurs. On the other hand, the buildings are not smooth when no blender used and all pixels are determined with the highest weight. Our algorithm is able to obtain satisfactory results although no seam cutting methods used.

IV. Experiments

To evaluate our system, the NPU DroneMap Dataset with several sequences is created and all data are available on the website[3]. We demonstrate the results of the proposed approach and compare the orthoimages with two state-of-the-art commercial softwares Photoscan [9] and Pix4DMapper [7], [8]. The results show that our system achieves higher robustness and comparative quality with less computation.

[3]http://zhaoyong.adv-ci.com/npu-dronemap-dataset



(a) NoBlender (b) FeatherBlender

(c) MultibandBlender (d) Map2DFusion

Fig. 3. Compositing comparison using different blender. (a) Pixels are overwritten with the maximum weight. (b) The blending result of feather blender. (c) The blending result of multi-band blender. (d) The result using our approach. Both feather and multi-band use the implementation of OpenCV.

A. NPU DroneMap Dataset

A self-made hexacopter and a DJI Phantom3 (advanced version) are used to record the sequences. The hexacopter is equipped with a GoPro Hero3+ camera and a DJI Lightbridge is used for real-time video transmission. Table I summarizes some basic statistics over all the sequences. The dataset contains sequences captured in different terrains and heights, furthermore, new captured sequences will be added to the dataset and published on the website.

For each sequence, the following contents are supplied:

1) Original data consist of video, flight log, GCPs locations, and camera calibration data. The cameras are calibrated with three different models including ATAN (used in PTAM [16]), OpenCV (used by ROS), and OCamCalib [37]. The flight logs have different formats since two types UAVs are utilized.
2) We convert the original data to an unified and readable format which is used by our system in the following evaluations. For each sequence, the video is divided into undistorted images and flight log is converted into text file.
3) The keyframe images and trajectories used in SfM methods and our Map2DFusion are offered.

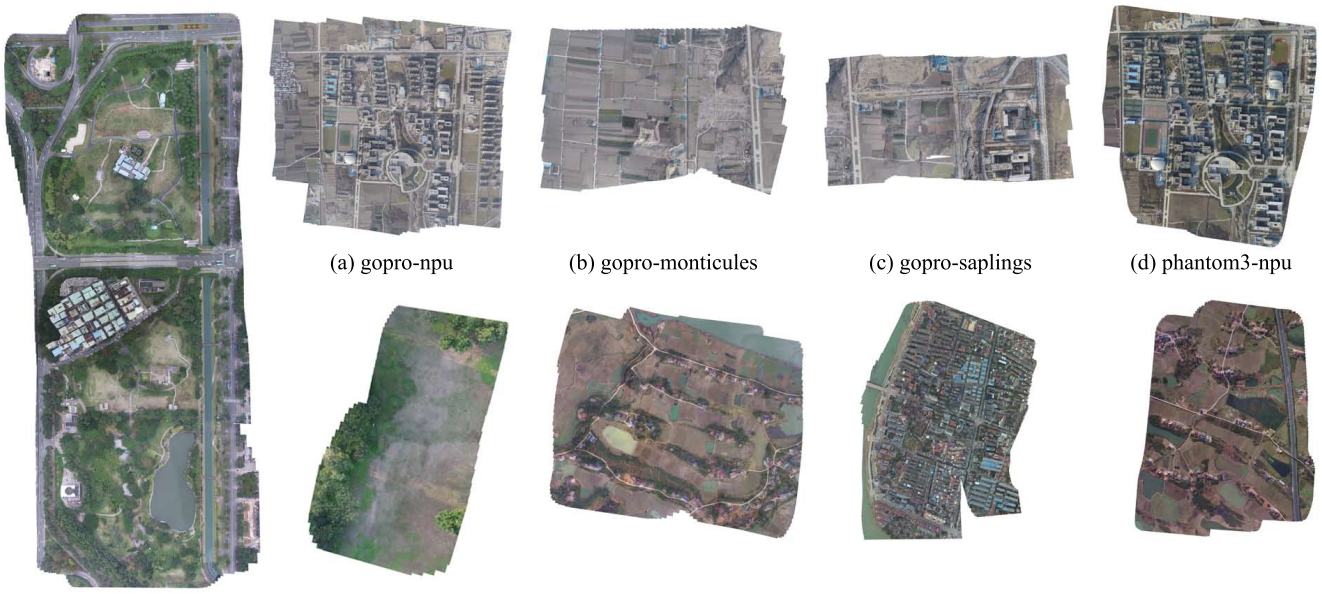
B. Fusion Results

We evaluate our system on the NPU DroneMap dataset, where thumbnails of the results are shown in Fig. 4 and the full resolution images are available on the website. The result shows that our approach not only achieves satisfactory quality on plain sequences, but also obtains acceptable

TABLE I

SUMMARIZES OF NPU DRONEMAP DATASET. 'H-MAX' DENOTES THE MAXIMUM FLIGHT HEIGHT (ABOVE GROUND) AND 'V-MAX' REPRESENTS MAXIMUM FLIGHT SPEED. 'TRAJ-LENGTH' REPRESENTS THE LENGTH OF RECORDED FLIGHT TRAJECTORY AND 'TRAJ-GPS' INDICATES WHETHER GLOBAL POSITIONS ARE AVAILABLE.

Sequence Name	Localization	UAV	H-Max (m)	V-Max (m/s)	Area	Traj-Length	Traj-GPS	GCPs
gopro-npu	Xi'an, Shaanxi	Hexacopter	376.8	10.39	2.739 km^2	5.547 km	yes	6
gopro-monticules	Xi'an, Shaanxi	Hexacopter	147.2	10.44	0.571 km^2	4.680 km	yes	0
gopro-saplings	Xi'an, Shaanxi	Hexacopter	129.2	12.40	0.455 km^2	4.566 km	yes	0
phantom3-npu	Xi'an, Shaanxi	Phantom3	254.5	16.96	1.598 km^2	6.962 km	yes	6
phantom3-centralPark	Shenzhen, Guangdong	Phantom3	161.8	16.61	0.606 km^2	4.536 km	yes	0
phantom3-grass	Shenzhen, Guangdong	Phantom3	78.6	10.81	3559.2 m^2	502.85 m	no	0
phantom3-village	Hengdong, Hunan	Phantom3	196.6	17.47	0.932 km^2	8.323 km	yes	0
phantom3-hengdong	Hengdong, Hunan	Phantom3	358.0	16.37	-	-	no	0
phantom3-huangqi	Hengdong, Hunan	Phantom3	222.3	16.57	1.313 km^2	6.945 km	yes	0



(a) gopro-npu (b) gopro-monticules (c) gopro-saplings (d) phantom3-npu

(e) phantom3-centralPark (f) phantom3-grass (g) phantom3-village (h) phantom3-hengdong (i) phantom3-huangqi

Fig. 4. The mosaicing results of NPU DroneMap Dataset using Map2DFusion. Our algorithm is able to handle all the sequences successfully and the result shows high robustness and quality in different environments.

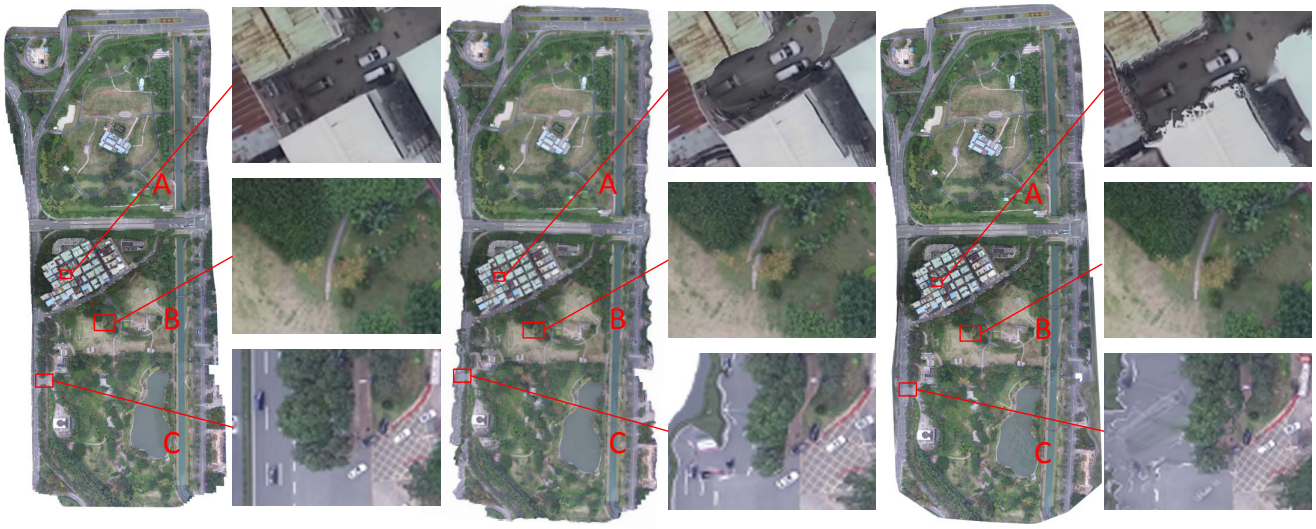
performances over trees, buildings, and water. Moreover, the proposed method remains high robustness on sequence *phantom3-grass*, which is very challenging because of the fast motion and repetitive views.

C. Comparisons to State-of-the-art Methods

We compare orthoimages generated by our method with two well-known softwares Photoscan [9] and Pix4DMapper [7]. All comparison data including the original images and orthomosaics are publicly available on the project website. For software Photoscan and Pix4DMapper, the parameter for aligning images is set to 'precise level'. The comparisons of some details in sequences *phantom3-centralPark* and *phantom3-village* are illustrated in Fig. 5. We notice that our approach obtains better results in some circumstances:

1) Firstly, our approach is able to obtain more nature mosaicing over vertical structures like trees and buildings. As we can see in part A of Fig. 5(a) and part B of Fig. 5(b), the borders of houses are twisty in Photoscan and rough in Pix4DMapper.
2) Secondly, the results generated by Photoscan contain some misalignments founded not only in part B of Fig. 5, but also some other sequences like *phantom3-npu*. Especially for challenging sequence *phantom3-grass*, our method shows high robustness while both Photoscan and Pix4DMapper are failed to obtain an acceptable result. The Pix4DMapper obtains better alignment than Photoscan in Fig. 5 but its blending is not smooth enough, while some obvious light spots exist over water area such as part A of Fig. 5(b).
3) Another important merit is that our algorithm remains high quality over areas with few overlap. This brings better result at margins such as part C of Fig. 5(a).

It should be noted that although hours are taken to align the keyframes, both Pix4DMapper and Photoscan failed to output correct results for the sequence *phantom3-grass* because of fast UAV speed and low altitude.



(a) *phantom3-centralPark*

(b) *phantom3-village*

Fig. 5. Comparison on sequences *phantom3-centralPark* (a) and *phantom3-village* (b). Left: Map2DFusion (our method), Middle: Photoscan, Right: Pix4DMapper.

TABLE II

TIME USAGE STATISTICS FOR ORTHOIMAGE GENERATING. THE TIME UNIT IS MINUTE.

Sequence	Frames	KFs	Ours	Pix4D	Photoscan
gopro-npu	28,493	337	15.84	107.05	153.62
gopro-monticules	18,869	395	10.49	52.62	334.73
gopro-saplings	19,371	482	16.44	83.75	683.98
phantom3-npu	13,983	457	9.32	140.08	532.38
phantom3-centralPark	12,744	471	8.49	127.73	563.57
phantom3-grass	4,585	648	2.39	154.77	999.67
phantom3-village	16,969	406	11.31	132.07	360.70
phantom3-hengdong	16,292	221	10.86	72.13	145.52
phantom3-huangqi	14,776	393	10.36	102.83	462.32

D. Computational Performance

All evaluations are performed on a computer equipped Intel i7-4710 CPU, 16 GB RAM, and GTX960 GPU. We run our approach and Photoscan in a 64-bit Linux system, while Pix4DMapper executes on 64-bit Windows 10. The time usage statistics with time unit of minute are illustrated in Table II.

Our method tracks all frames, while both Pix4DMapper and Photoscan only process the keyframes to generate the final images. As the keyframe number increases, the processing time of Pix4DMapper and Photoscan increase dramatically. However, the computational complexity of our system is approximately linear, consequently, large-scale scenes can be handled fluently. In the proposed system, the tracking runs at about 30 Hz and the stitching runs at about 10 Hz for full 1080p video. In fact, it is not necessary to process every frame due to the large baseline tracking ability, thus the speed can be further boosted by processing selected frames. Based on the above merits, it is sufficiently to use this system on low-cost embedded devices.

V. CONCLUSIONS

This paper presents a novel approach to mosaic aerial images incrementally in real-time with high robustness and quality. Most traditional SfM methods need all data prepared and hours for computation, while the proposed method outputs comparative or even better results with real-time speed. Moreover, the results are able to be visualized in the map widget immediately. We publish the detailed results, introduction video, and dataset on the project website, in addition, the code of Map2DFusion is also provided.

Although our system outputs high quality mosaics in most circumstances, there are still some improvements can be made especially for non-planar environments. PhotoScan and Pix4DMapper output dense point cloud and mosaic with

better orthogonality. However, our method just computes sparse point cloud and the orthoimages are not totally ortho although great efforts are conducted to make the result as ortho as possible. Some other image warping functions considering 3D structure should be investigated to minimize the misalignments. In addition, quantitative evaluations will be conducted in the future researches.

ACKNOWLEDGEMENTS

This work is partly supported by grants from National Natural Science Foundation of China (61202185, 61473231, 61573284), the Fundamental Research Funds for the Central Universities (310201401-(JCQ01009,JCQ01012)), Open Projects Program of National Laboratory of Pattern Recognition (NLPR).

REFERENCES

[1] T. Samad, J. S. Bay, and D. Godbole, "Network-centric systems for military operations in urban terrain: the role of uavs," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 92–107, 2007.

[2] W. Tao, "Multi-view dense match for forest area," *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, pp. 397–400, 2014.

[3] B. H. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmad, A. Sirota, and B. Cooper, "Athlete: A cargo handling and manipulation robot for the moon," *Journal of Field Robotics*, vol. 24, no. 5, pp. 421–434, 2007.

[4] J. Shi, J. Wang, and Y. Xu, "Object-based change detection using georeferenced uav images," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, vol. 38, pp. 177–182, 2011.

[5] U. Niethammer, M. James, S. Rothmund, J. Travelletti, and M. Joswig, "Uav-based remote sensing of the super-sauze landslide: Evaluation and results," *Engineering Geology*, vol. 128, pp. 2–11, 2012.

[6] P. Moulon, P. Monasse, R. Marlet, and Others, "Openmvg. an open multiple view geometry library." https://github.com/openMVG/openMVG.

[7] J. Vallet, F. Panissod, C. Strecha, and M. Tracol, "Photogrammetric performance of an ultra light weight swinglet uav," in *UAV-g*, no. EPFL-CONF-169252, 2011.

[8] O. Küng, C. Strecha, A. Beyeler, J.-C. Zufferey, D. Floreano, P. Fua, and F. Gervaix, "The accuracy of automatic photogrammetric techniques on ultra-light uav imagery," in *UAV-g 2011-Unmanned Aerial Vehicle in Geomatics*, no. EPFL-CONF-168806, 2011.

[9] G. Verhoeven, "Taking computer vision aloft–archaeological three-dimensional reconstructions from aerial photographs with photoscan," *Archaeological Prospection*, vol. 18, no. 1, pp. 67–73, 2011.

[10] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustmentła modern synthesis," in *Vision algorithms: theory and practice*. Springer, 1999, pp. 298–372.

[11] R. Hirokawa, D. Kubo, S. Suzuki, J.-i. Meguro, and T. Suzuki, "A small uav for immediate hazard map generation," in *AIAA Infotech@ Aerospace 2007 Conference and Exhibit*, 2007, p. 2725.

[12] T. Botterill, S. Mills, and R. Green, "Real-time aerial image mosaicing," in *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*. IEEE, 2010, pp. 1–8.

[13] P. Xiong, X. Liu, C. Gao, Z. Zhou, C. Gao, and Q. Liu, "A real-time stitching algorithm for uav aerial images," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*. Atlantis Press, 2013.

[14] S. Lovegrove and A. J. Davison, "Real-time spherical mosaicing using whole image alignment," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 73–86.

[15] T. Maki, H. Kondo, T. Ura, and T. Sakamaki, "Photo mosaicing of tagiri shallow vent area by the auv" tri-dog 1" using a slam based navigation scheme," in *OCEANS 2006*. IEEE, 2006, pp. 1–6.

[16] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[17] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2320–2327.

[18] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[19] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 834–849.

[20] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, "Openfabmap: An open source toolbox for appearance-based loop closure detection," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 4730–4735.

[21] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 141–148.

[22] J. Engel, J. Stuckler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1935–1942.

[23] R. Mur-Artal, J. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *arXiv preprint arXiv:1502.00956*, 2015.

[24] R. Mur-Artal and J. D. Tardós, "Probabilistic semi-dense mapping from highly accurate feature-based monocular slam," *Proceedings of Robotics: Science and Systems, Rome, Italy*, 2015.

[25] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007.

[26] J. Zaragoza, T.-J. Chin, Q.-H. Tran, M. S. Brown, and D. Suter, "As-projective-as-possible image stitching with moving dlt," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 7, pp. 1285–1298, 2014.

[27] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," in *ACM Transactions on Graphics (ToG)*, vol. 22, no. 3. ACM, 2003, pp. 277–286.

[28] Y. Zhao and D. Xu, "Fast image blending using seeded region growing," in *Advances in Image and Graphics Technologies*. Springer, 2015, pp. 408–415.

[29] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics (TOG)*, vol. 2, no. 4, pp. 217–236, 1983.

[30] M. G. P. Perez and A. Blake, "Poisson image editing," in *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, 2003.

[31] J. Selig, "Lie groups and lie algebras in robotics," in *Computational Noncommutative Algebra and Applications*. Springer, 2004, pp. 101–125.

[32] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[33] G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation*, 2011.

[34] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based slam," 2014.

[35] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolig, "Double window optimisation for constant time visual slam," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2352–2359.

[36] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *JOSA A*, vol. 4, no. 4, pp. 629–642, 1987.

[37] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*. IEEE, 2006, pp. 45–45.