TerrainFusion: Real-time Digital Surface Model Reconstruction based on Monocular SLAM

Wei Wang , Yong Zhao , Pengcheng Han , Pengcheng Zhao , and Shuhui Bu

***Abstract*— This paper aims to generate live digital surface model (DSM) during the flight based on simultaneous localization and mapping (SLAM) algorithm. We process the keyframe which is output by a monocular SLAM system to generate a local DSM, and fuse the local DSM to the global tiled DSM incrementally. During the local DSM generation, a local digital elevation model (DEM) is estimated by projecting the filtered 2D Delaunay mesh to a 3D mesh, and a local orthomosaic is obtained by projecting triangle image patches onto a 2D mesh. During the DSM fusion, both the local DEM and orthomosaic are split into tiles and fused to the global tiled DEM and orthomosaic respectively with multiband algorithm. Both the efficient DSM generation and fusion algorithms contribute to achieving a real-time reconstruction. Qualitative and quantitative experiments on a public aerial image dataset with different scenarios are performed to validate the effectiveness of the proposed method. Compared with traditional structure from motion (SfM) based approaches, the presented system is able to output both large-scale high-quality DEM and orthomosaic in real-time with low computational cost.**

I. INTRODUCTION

A digital surface model (DSM) [1], [2], [3] is a raster-based description of the terrain including topography and all natural or human-made features located on the surface of the earth, which plays fundamental roles in 3D modeling for aviation, urban planning, and telecommunications. In recent years, DSM is more widely used in multiple fields [4], [5], and the demand for the real-time DSM reconstruction technique is increasing rapidly, such as fire monitoring [6] and urban surveillance [7], [8]. Besides, with the rapid development of unmanned aerial vehicles (UAVs), aerial images are able to be collected at a low cost and high mobility. Considering the above two aspects, we conceive an idea to reconstruct live DSM from aerial images collected by UAVs when they are flying.

Frequently used implementations for DSM reconstruction are based on structure from motion (SfM) [9], [10], [11], [12], such as projects OpenMVG [13], openDroneMap, and commercial softwares Pix4DMapper [14] or Photoscan [15]. They share similar pipeline [16], [17], [18], [19]: feature detection, matching, image alignment, sparse pointcloud generation with bundle algorithm, dense pointcloud generation, mesh and texture generation. Nevertheless traditional SfM methods always need all images prepared before computation and then take hours to generate final results, thus these reconstruction methods based on SfM [13], [15], [14], [20], [18] are not suitable for real-time and incremental usage.

Shuhui Bu, Wei Wang, Yong Zhao, PengCheng Zhao, and Pengcheng Han are with Northwestern Polytechnical University, 710072 Xi'an, China bushuhui@nwpu.edu.cn, shaxikai@outlook.com

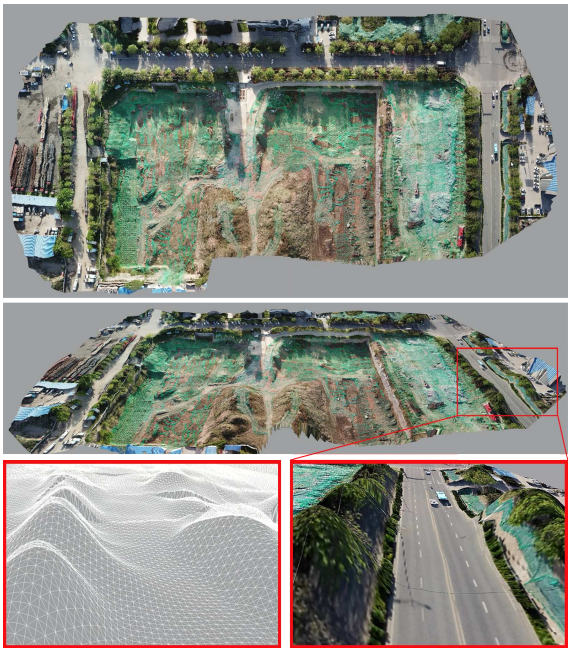


Fig. 1. Our method efficiently reconstructs DSM of a scene, and is capable of creating high-quality details. The two subfigures at the bottom are the details of the street in the scene, Left: DEM, Right: DSM.

Another favorable choice for real-time reconstruction is simultaneous localization and mapping (SLAM) [21], [22], [23], [24], which is able to output environment map and camera position in real-time due to its well designed processing pipeline. Therefore, some researchers utilize SLAM for real-time UAV image mosaicing [25] and height field reconstruction [26]. However, method presented in [25] only generates orthomosaic without height information and [26] is limited to use height map and GPU for achieving real-time speed.

In this paper, we explore to incrementally obtain DSM consisting of generating digital elevation model (DEM) [27] and orthomosaic [28], which are simultaneously performed and then real-time fused. To achieve this, we use the visual SLAM to obtain the camera poses and key-points with depth information in real-time, and calculate the sparse pointcloud for each keyframe. Then we combine the surface property of the captured image and 3D structure of the sparse pointcloud to generate the local DEM and orthomosaic simultaneously. Finally we fuse them to the global tiled DEM and orthomosaic, respectively. Because of the advanced processing pipeline, our method called as TerrainFusion is able to reconstruct high-quality DSM of a scene in real-time, as

demonstrated in Fig. 1. Compared with those prior works, our approach makes the following contributions:

- A novel real-time DSM reconstruction framework based on monocular SLAM, which is able to process the sequential aerial images to reconstruct DSM incrementally.
- A noise filtering algorithm, which eliminates both outliers and pits in the pointcloud, and effectively makes the generated surface to be a 2D manifold.
- A robust DSM generating algorithm, which rapidly and simultaneously generates DEM and orthomosaic from each keyframe output by SLAM.
- A tile format for managing DSM, which allows our system to handle large-scale scenes.
- A DSM fusing algorithm, which not only efficiently updates the global DSM in real-time, but also ensures high-quality reconstruction.

II. Related Work

Reconstructing 3D surface from collected images is a hot research topic in recent decades, traditional methods' description and comparisons could be found at [29], [27]. In this paper, we focus on SfM and SLAM based methods, which are summarized as follows.

A. SfM based methods

SfM as the central problem in computer vision community for estimating 3D structure from image collection [30], [31], [32] is commonly used as the base for DSM reconstruction [13], [15], [14], [20], [18]. So far, there are a variety of SfM strategies have been proposed including incremental [9], [10], global [11], [12], and hierarchical approaches [33].

Incremental reconstruction of manifold surface [20] utilizes 3D Delaunay triangulation to reconstruct surface from sparse 3D pointcloud calculated by incremental SfM. However, [20] aims to reconstruct a closed surface, which makes it finite for DSM reconstruction. Poisson surface reconstruction [18] is one of the most famous 3D reconstruction approaches which expresses surface reconstruction as the solution to a Poisson equation. Although Poisson reconstruction approach shows a mastery of technical skill and detail, it could not reconstruct surface in real-time.

B. SLAM based methods

Unlike traditional SfM, which does not perform real-time calculation, SLAM emphasizes calculating the camera pose and pointcloud for each keyframe in real-time. SLAM technology develops rapidly and a variety of SLAM systems have been proposed, including monocular SLAM system (key-point based [34], [35], [21], direct [36], [22], [37], and semi-direct methods [23], [38]), multi-sensor SLAM system (RGBD [39], [40], [41], Stereo [42], [38], [43] and inertial aided methods [44], [45], [24]), and learning based SLAM system (supervised [46], [47], [48] and unsupervised [49], [50]).

In recent years, some researches [25], [26], [51] attempt to adopt SLAM to photogrammetric survey for obtaining orthoimage or height field. Map2DFusion [25] mosaics aerial images incrementally to generate 2D map. Despite Map2DFusion using multiband algorithm outputs high quality mosaics in most circumstances, the plane fit makes it perform poorly for non-planar environment, and it only generates digital orthophoto map (DOM) without height information. In the case of orthomosaic generation, our method handles the non-planar environment effectively by more considering 3D structure. In particular our method is able to reconstruct both orthomosaic and height field in real-time. Zienkiewicz *et al.* [26] reconstruct surface by performing depth-map and color fusion directly into a multi-resolution triangular mesh. Nevertheless, this method is implemented on the GPU for achieving real-time speed, and only reconstructs height field without texture. In addition, this approach requires that the camera is as close as possible to the surface, hence it is not appropriate for DSM reconstruction of large-scale scene when the camera far above the ground. Hinzmann *et al.* [51] implement a mapping framework that incrementally generates a dense georeferenced pointcloud, a digital surface model, and an orthomosaic. They aim to generate DSM from dense pointcloud which may fail to handle complex scenes, while our method focuses on using the sparse pointcloud to real-time reconstruct large-scale DSM.

III. TerrainFusion

The overview of our pipeline is depicted in Fig. 2, which consists of three distinct stages: visual SLAM, local DSM generation, and DSM fusion. Starting from an aerial image, we first employ a visual SLAM to estimate camera pose and sparse structure (Section III-A). We then execute 2D Delaunay triangulation on key-points to generate a 2D mesh, and project the 2D mesh to a 3D mesh based on the projective relationship between key-points and pointcloud. Because many conflicts exist in original pointcloud which disturb the DEM generation, iterative filter is adopted to remove the noises. After filtering noises, we calculate a local DEM from the filtered 3D mesh and generate a local orthomosaic from the meshes and image (Section III-B). In the last stage, the local DEM and orthomosaic are stitched to the corresponding global maps respectively, and multiband algorithm is applied on both stitched DEM and orthomosaic for smooth transition (Section III-C).

A. Monocular SLAM

A key-point based monocular SLAM is used in our system for estimating the live pose and sparse depth. The implementation of this framework is similar to the visual SLAM and georeferencing performed in [25], but some improvements are adopted to make it robust to process both high frequency video and low frequency photos. As it is not the main focus of this paper, we only give a brief introduction.

For every captured image, a SIFT extractor accelerated by GPU is used to detect key-points and extract features in real-time. The map is initialized by decomposing the relative pose from a fundamental or homography matrix. When GPS

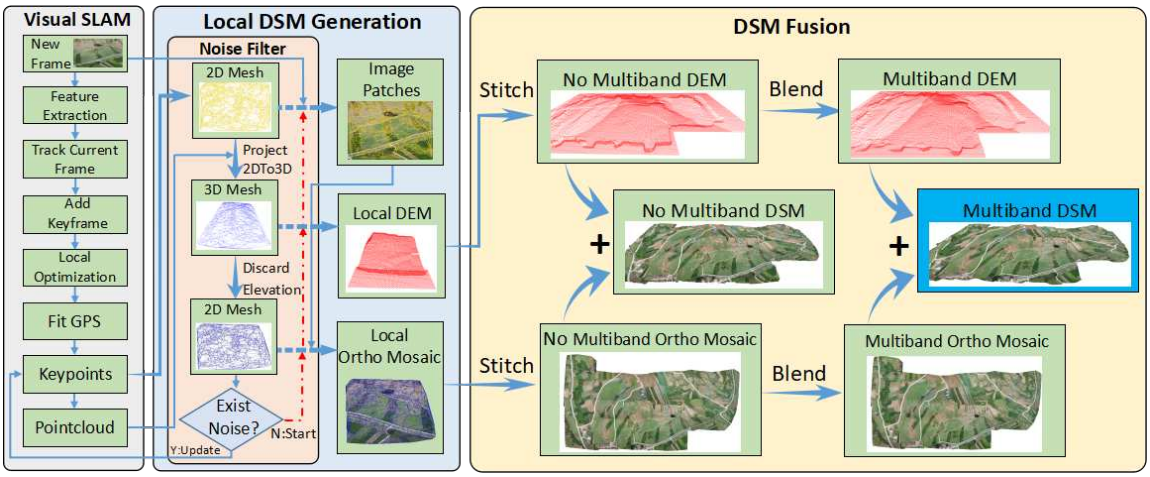


Fig. 2. TerrainFusion: Overall pipeline for real-time DSM reconstruction of a scene from aerial images. We use keyframe data to generate local DEM and orthomosaic, and fuse them to the global DEM and orthomosaic respectively. As a result, the global DSM of the scene is incrementally reconstructed by integrating the continuously updated global DEM and orthomosaic.

is available, the map is transformed to the earth-centered, earth-fixed (ECEF) coordinate, otherwise an initial plane is fitted as the ground plane. After the map is initialized, the tracking thread processes every frame and tries to estimate current pose by tracking the last frame. The pose is calculated by solving the perspective n-point (PnP) problem from pairs of 3D map-points and 2D key-points. The local sub-map is then projected to the current frame for obtaining more PnP observations and a better pose estimation can be achieved. Keyframes are inserted to the mapping thread when the relative distance between the current frame and the last keyframe exceeds a certain threshold. New mappoints are triangulated with epipolar search and some data association operators are done, followed with a local bundle executed to optimize the local map-points and keyframe poses with GPS constraints. The last keyframe with sparse depth information is published for the later reconstruction.

B. Local DSM

Unlike most SfM based methods that generate the whole DSM at once, our goal is to generate local DSM for each keyframe and then fuse it to the global DSM in an incremental manner. The visual SLAM system has already real-time output pose and key-points with depth information for each keyframe, from which pointcloud of the keyframe is computed. Unexpectedly, outliers and pits (collectively called noises) which break the DSM attribute exist in key-points and pointcloud. In order to ensure that the generated surface is a 2D manifold, a filtering algorithm is designed to filter out the noises. This section firstly describes the filter algorithm, and then introduces the implementation details of DEM and orthomosaic generation.

a) Mesh Generation: To rapidly generate 3D mesh for each keyframe, we propose a project 2D-to-3D method which is the core part of the local DSM generation. According to the pixel coordinates of the key-points P_i in the current keyframe, the 2D Delaunay triangulation is performed to obtain the 2D triangular mesh M_i on the image plane. In terms of the projective relationship between the 3D pointcloud and the key-points, projecting the topology of M_i onto 3D pointcloud results in a 3D triangular mesh M_w. In the world coordinate system, the 3D triangular mesh M_w which uses the pointcloud as the mesh points contains geographic information. Since the M_w exposes the outliers and pits, filtering operations are performed to eliminate them. For generating orthomosaic later, the elevation information of M_w is discarded to get a 2D triangular mesh M_h on the horizontal plane in the world coordinate system.

b) Edge Detection: Our filtering algorithm is based on geometry constraints, and the edge points and non-edge points of M_h have different geometries, hence different filtering strategies are adopted on them to filter out noises respectively. The judgment $\mathscr{E}_e$ aims to distinguish the set of edge points P_e and the set of non-edge points P_n from the mesh points of M_h: For each mesh point p in M_h, only if the sides opposite to p in the neighboring triangles form a simple closed polygon, is the point a non-edge point. The neighboring triangles refer to the triangles which contain p in M_h. The red lines in Fig. 3 are the edge of M_h which is connected by the edge points P_e.

c) Noise Filter: Because both outliers and pits exist in P_e, two judgments are performed for each edge point to estimate whether it is a noise. The first judgment $\mathscr{E}_{e1}$ mainly estimate whether the edge point is a pit and the second judgment $\mathscr{E}_{e2}$ mainly estimate whether it is an outlier. During the first judgment, the opposite sides of an edge point p in the neighboring triangles form a polyline (the polyline is a line segment when p has only one neighboring triangle). Then two lines are obtained by jointing p with the two end points of the polyline respectively. If the two lines intersect with any triangle which does not contain the neighboring points of p as the vertexes, we treat p as an edge noise. The neighboring points of p refer to all the other two vertexes of each triangle which contains p in M_h. In Fig. 3, P_1 is an

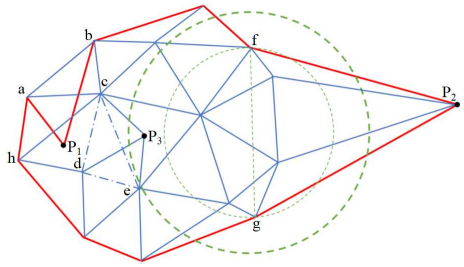
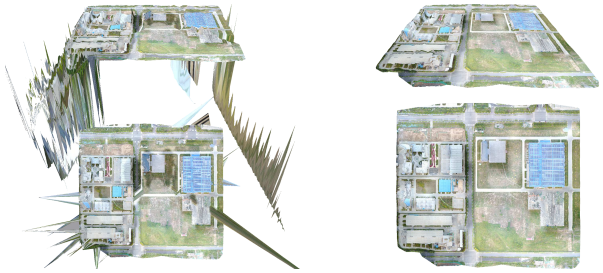


Fig. 3. This figure demonstrates different strategies for filtering noises. The 2D mesh M_h (blue and red) is generated by discarding the elevation in the 3D mesh M_w. The red lines are the edge of M_h. P_1 is an edge noise since the edge line segments P_1a, P_1b which use P_1 as endpoint intersect with the $\triangle cdh$. P_2 is an edge noise since P_2 is located outside the concentric circle (thick green) whose radius is k times to the circle (thin green) which uses the line segment fg as the diameter. P_3 is a non-edge noise since it is outside the closed polygon cde formed by the opposite sides of P_3 in the neighboring triangles.



(a) Results without noise filter (b) Results with noise filter

Fig. 4. Comparison of DSM (top) and orthomosaic (bottom) with and without noise filter using data of *mavic-factory*.

edge noise. During the second judgment, a line segment is made by jointing the two endpoints of the polyline which is given in the first judgment. Then we obtain a concentric circle whose radius is k times to the circle which uses the line segment as the diameter. If the edge point is located outside the k-radius concentric circle, we judge the edge point is uncorrelated with mesh and treat it as a noise. k is a parameter and we empirically set it to 1.4. As shown in Fig. 3, P_2 is a noise. When it comes to the judgment $\mathscr{E}_n$ for noises in P_n. For each point p in P_n, if p is located outside the closed polygon formed by the opposite sides of p in the neighboring triangles, it is a noise. As shown in Fig. 3, P_3 is a noise.

The iterative method illustrated in Algorithm 2, is used to filter out the edge noises and non-edge noises. After generating mesh and filtering out noises once, the updated set of key-points P_i and pointcloud P_c are used to generate mesh and filter the noises repeatedly until no noises can be detected. In addition, edge noise filter is embedded in the iterative of non-edge noise filter to avoid the edge (non-edge) noises is not filtered out after once non-edge (edge) noise filter which has already changed the mesh structure. Fig. 4 shows that our filter algorithm is able to effectively remove noises.

Algorithm 1 Edge Detection

function EDGEDETECT(P_i, P_c)
  $P_e = \emptyset, P_n = \emptyset$
  $M_i =$ 2D-Delaunay(P_i)
  $M_w =$ Project2DTo3D(M_i)
  $M_h =$ DiscardHeight(M_w)
  for each $p \in M_h$ **do**
    if $\mathscr{E}_e(p)$ **then**
      $P_n \leftarrow P_n \cup \{p\}$
    else
      $P_e \leftarrow P_e \cup \{p\}$
    end if
  end for
  return (P_e, P_n)
end function

Algorithm 2 Noise Filter

function NOISEFILTER(P_i, P_c)
  do
    do
      $(P_e, P_n) =$ EDGEDETECT(P_i, P_c)
      $P_o = \emptyset$
      for each $p \in P_e$ **do**
        if $\mathscr{E}_{e1}(p)$ and $\mathscr{E}_{e2}(p)$ **then**
          $P_o \leftarrow P_o \cup \{p\}$
        end if
      end for
      $P_i = P_i \backslash P_o, P_c = P_c \backslash P_o$
    while $P_o \neq \emptyset$
    $(P_e, P_n) =$ EDGEDETECT(P_i, P_c)
    $P_o = \emptyset$
    for each $p \in P_e$ **do**
      if $\mathscr{E}_n(p)$ **then**
        $P_o \leftarrow P_o \cup \{p\}$
      end if
    end for
    $P_i = P_i \backslash P_o, P_c = P_c \backslash P_o$
  while $P_o \neq \emptyset$
end function

d) Orthomosaic Generation: Different from the methods wrapping images, we project triangle image patches onto W_h to generate orthomosaic. Since M_i on the image plane has already divided the image of current keyframe into a set of triangle patches, a local orthomosaic could be obtained by projecting the triangular patches onto M_h.

e) DEM Generation: To make the later fusion more efficient, we convert the irregular 3D triangular mesh M_w to a local tiled DEM. The height of the local DEM is calculated by interpolating the mesh points' heights of M_w. As Fig. 5 shows, in our implementation, the linear interpolation is selected due to its rapidity and stability.

C. DSM Fusion

When the local DSM is generated, the global DSM needs to be updated. To achieve a high-quality stitching,

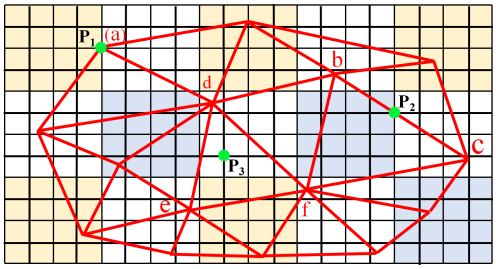


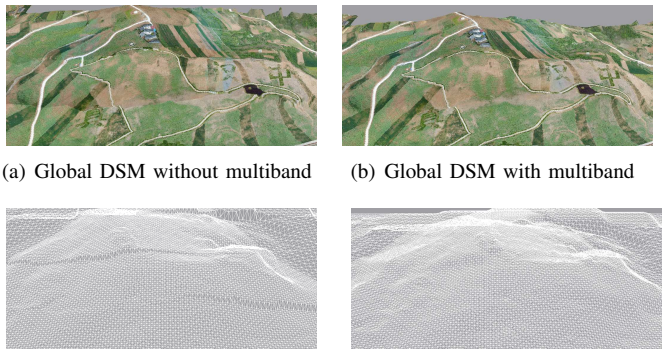
Fig. 5. This figure shows the top view of the transformation process from the 3D triangle mesh M_w (red line) which uses pointcloud as mesh points to the local tiled DEM (black line). The different background color represents that the local DEM are split up and stored in different tiles (it is a schematic diagram, only 4×4 drawn in a tile for clarity). a,b,c,d,e,f are vertexes of M_w and P_1,P_1,P_1 are three vertexes of the local DEM. The height value of the P_1 is equal to the height value of a. The height value of the P_2 is obtained by interpolating the height values of two endpoints of the line segment bc. The height value of the P_3 is obtained by interpolating the height values of three vertexes of $\triangle def$.

a weight image is adaptively computed considering the sparse pointcloud distribution in DEM and view angle of the current keyframe. The local DSM with larger weight value is stitched to the global DSM in units of tile. Finally, multiband algorithm is performed in the stitched DSM tiles for smooth transition.

a) Fusion Weight: In our implementation, the weight used for DSM stitching is also stored as a map, which is a single-channel unsigned char image. Each pixel value of the weight image is calculated as follows: The arithmetic mean of all mesh points' coordinates in M_h of the current keyframe is calculated to obtain center coordinate o. The weight of o is set to 255, while the weight of o' whose position is farthest from o is set to 0. As we calculate the pixel distance ζ between o and o', the gradient ρ between the value and the pixel distance could be calculated as $\rho = 255.0/\zeta$. Finally, the values $\{w_i\}$ of all pixels $\{p_i\}$ in the weight image could be calculated according to the gradient ρ and the pixel distance $\{dis_i\}$ between the $\{p_i\}$ and o using the formula $w_i = \rho \times (\zeta - dis_i)$.

b) Stitching: Most of the online maps are provided in form of tiles by service providers and the tiled DSM is beneficial for the blending later, so that we manage DSM in tile format. In our system, the local DEM is split into rectangular patches and each patch is stored in an individual tile. The local orthomosaic and weight image are also split up in the same manner, and the patches are stored in the corresponding tiles, too. Consequently, a DSM tile contains several layers including a DEM patch, an orthomosaic patch, and a weight image patch. We calculate a local mask by comparing the local tiled weight image with the global tiled weight image. According to this mask, we stitch the local tiled DEM and orthomosaic to the global ones respectively.

c) Blending: Although the above steps are able to achieve a real-time DSM reconstruction, visible seams exist in the stitched DSM, as shown in Fig. 6 (a) and (c). In fact, this is caused by two reasons: 1) Stitching the orthomosaics which are generated from the images with different exposures



(a) Global DSM without multiband (b) Global DSM with multiband

(c) Global DEM without multiband (d) Global DEM with multiband

Fig. 6. Comparison of DEM and DSM with and without multiband algorithm.

results in a slight color deviation around stitching lines; 2) The stitched DEM has small offsets on stitching lines as well. To smooth the color deviation and the offsets, we apply multiband algorithm on both stitched DEM and orthomosaic.

For implementing multiband algorithm, the following steps are added to our system. Two Laplacian pyramids are expanded from the local DEM and orthomosaic by expanded operation [52], we call them local DEM pyramid and local orthomosaic pyramid. Correspondingly, the weight pyramid is expanded from weight image by down-sampling. The three pyramids are also split into rectangular sub-pyramids, and the sub-pyramids are stored in the corresponding tiles. A local mask pyramid is computed by comparing the local tiled weight pyramid with the global one. Instead of stitching local tiled DEM, we stitch its pyramid to the global one according to the mask pyramid, and so does the tiled orthomosaic pyramid stitching. Finally, the original DEM and orthomosaic are recovered by summing the levels of the their global stitched Laplacian pyramids from low to high.

IV. Experiments

We implement the proposed real-time DSM reconstruction system in C++ and evaluate its performance in terms of computational efficiency and accuracy. We first test our system on a public dataset and show three DSMs which are reconstructed from different environments. More results are shown on the website *https://shaxikai.github.io/TerrainFusion*. Then we compare the orthomosaics on the mountainous region which are generated by TerrainFusion and Map2DFusion. For the efficient comparison, as far as we know this is the first open source online DSM reconstruction system, which leaves no choice but to compare our system with two offline existed softwares, Photoscan and Pix4DMapper. Since there is no dataset contains both ground-truth height and aerial images of the environments, we finally make a cross-comparison among the three surfaces which are generatedrespectively by our method, Photoscan, and Pix4DMapper, to analyze the accuracy of our method. The results show that our system achieves high robustness, comparative quality, and more efficient real-time DSM reconstruction.

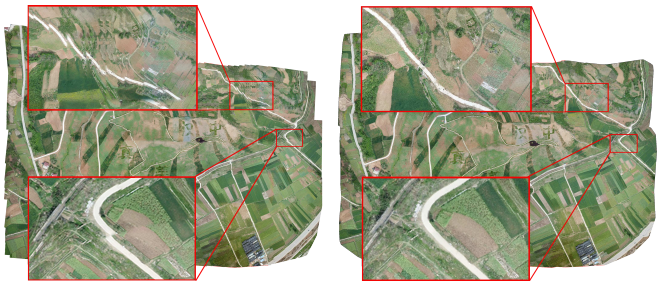


Fig. 7. Orthomosaic comparison on dataset *mavic-mountainlong*. Left: Map2DFusion, Right: TerrainFusion.

All experiments are performed on a desktop PC with an Intel i7-7700 CPU, a 16 GB RAM, and a GTX 1060 GPU. We run our approach and Photoscan in a 64-bit Linux system, while Pix4DMapper executes on 64-bit Windows 10.

A. Results On TerrainFusion Dataset

To evaluate our system, we create a TerrainFusion dataset which contains several scenarios, including city, mountains, desert, and plains. The resolution of the aerial images is 4000×3000. They are publicly downloaded from the website, and the demonstration video is also available at the site.

In the first experiment, we compare the quality of oromosaics which are generated by TerrainFusion and Map2DFusion [25]. For planar environment, two methods output nearly identical orthomosaic, and further comparisons with Photoscan or Pix4DMapper could be found at [25]. However, when it comes to the non-planar environment, especially the mountainous regions where the slope gradient is large, the plane fit performed in Map2DFusion may generates ghosting, as shown in Fig.7. While our method considers much more 3D construct from pointcloud to generate orthomosaic, which could supposedly avoids ghosting.

In order to illustrate the practicability of our approach, some experiments are conducted with a variety of scenarios. For the sake of contrast, we present the results reconstructed by both our method and Photoscan in Fig. 8. The comparison results show clearly that our approach is able to achieve the similar acceptable DSM reconstruction as Photoscan does. More reconstruction results are accessible on the website and they all demonstrate that our method not only performs excellent on natural features reconstruction, but also achieves acceptable reconstruction for urban and rural areas.

B. Efficiency Comparison

We compare the execution efficiency of different methods (our method, Photoscan and Pix4DMapper) using TerrainFusion dataset. The time usage statistics is listed in Table I. To make this a fair comparison, we detect almost the same number of key-points (around 1000) in each keyframe for the three methods and use defaults for the other parameters. Different from Photoscan and Pix4DMapper which perform offline reconstruction, our method process reconstruction in an incremental manner. So the time illustrated in Table I of our method is the sum of the time spent on generating and fusing DSM for all keyframes. After all, what we achieve is an online reconstruction, but either Photoscan or Pix4DMapper performs offline and batch-optimization reconstruction. Therefore, our approach takes much less time than Photoscan or Pix4DMapper.

TABLE I

TIME USAGE STATISTICS FOR DSM RECONSTRUCTION BY DIFFERENT METHODS (THE TIME UNIT IS SECOND).

Sequences	KFs	Photoscan	Pix4D	Ours
olathe	160	378.0	672.2	**9.1**
mound60m	106	277.6	1504.8	**12.4**
mountainlong	285	368.0	1480.5	**23.2**
fengniao	157	136.9	686.2	**13.5**
village	235	166.1	840.0	**22.1**
factory	358	910.1	1380.5	**36.8**

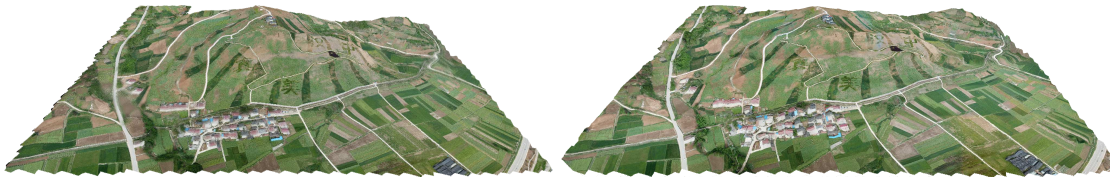
C. Quantitative Comparison

In quantitative comparison, the surface error among different methods (TerrainFusion, Photoscan and Pix4DMapper) are calculated to analyze the accuracy of our method. We sample points $\{p_i\}$ on the model and obtain the height values $\{h'_i\}$ of $\{p_i\}$ on the surface generated by one method. Similarly, we obtain the height value $\{h_i\}$ of $\{p_i\}$ on the surface generated by another method. Then we use $|h_i - h'_i|$ to calculate the error on $\{p_i\}$ between two surfaces generated by different methods. The error distributions and statistics between TerrainFusion and the two softwares on the data of *mavic-fengniao* are shown separately in Fig. 9 (a) and (b). We find that 90.13% sample points' errors are less than 2.0 m, and 62.92% are less than 1.0 m between our method and Photoscan. While comparing our method with Pix4DMapper, 88.96% sample points' errors are less than 2.0 m, and 62.32% are less than 1.0 m. It is easy to see that the error distributions are similar between Fig. 9 (a) and (b).

In order to evaluate the accuracy, we calculate the error of the surfaces generated by Photoscan and Pix4DMapper as well, as shown in Fig. 9 (c). There are 92.35% sample points' error are less than 2.0 m and 81.56% are less than 1.0 m. It is obvious that the error between the two softwares is less than that between our method and either of them. The red bars in the right of Fig. 9 represent the sample points whose errors are greater than or equal to 8.0 m. When extracting image key-points, one method may extract the key-points at the edge of the image, so it could reconstruct the surface at the edge. While in another method, the key-points at the edge of the image are not extracted, so it could not reconstruct the surface. As a result, the surface reconstructed by different methods has a large error at the edge. As the error distribution in Fig. 9 shows, almost all of those large errors are occurred on the edge.

All in all, what we implement is an online DSM reconstruction from sparse pointcloud. Therefore, it is acceptable to reconstruct DSM in real-time by losing minor accuracy in exchange for speed.

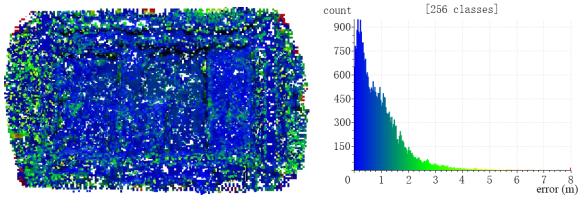
(a) *phantom3-olathe*



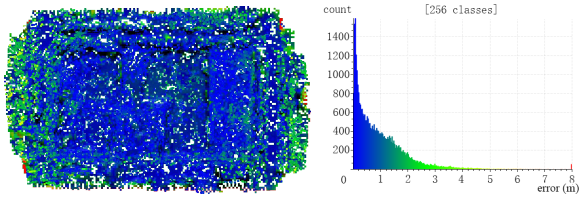
(b) *mavic-mountainlong*

(c) *mavic-fengniao*

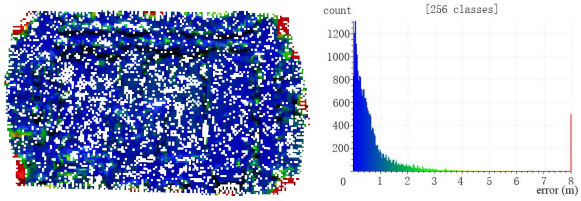
Fig. 8. Reconstruction results compared with SfM based method (Left: Photoscan, Right: our method). Three sequences show the reconstruction of plains, mountain, and city.



(a) Our method and Photoscan



(b) Our method and Pix4DMapper



(c) Photoscan and Pix4DMapper

Fig. 9. Error distribution (left) and statistics (right) of different methods on the data of *mavic-fengniao*. (a) Surface error between our method and Photoscan. 90% sample points' errors are less than 1.96 m. (b) Surface error between our method and Pix4DMapper. 90% sample points' errors are less than 2.09 m. (c) Surface error between Photoscan and Pix4DMapper. 90% sample points' errors are less than 1.72 m.

V. Conclusions

We have presented a novel method for live DSM reconstruction from sparse pointcloud generated by monocular SLAM. Although the accuracy of the proposed method is not superior than commercial softwares, the speed is hight enough for an online application to reconstruct DSM in real-time. In the future, we will devote to densifying the sparse pointcloud and reconstructing high fidelity DSM from the dense pointcloud in real-time.

References

[1] L. Zhang, *Automatic digital surface model (DSM) generation from linear array images*. ETH Zurich, 2005.

[2] A. El Garouani, A. Alobeid, and S. El Garouani, "Digital surface model based on aerial image stereo pairs for 3d building," *International Journal of Sustainable Built Environment*, vol. 3, no. 1, pp. 119–126, 2014.

[3] P. Gamba and B. Houshmand, "Digital surface models and building extraction: A comparison of ifsar and lidar data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 4, pp. 1959–1968, 2000.

[4] J. Järnstedt, A. Pekkarinen, S. Tuominen, C. Ginzler, M. Holopainen, and R. Viitala, "Forest variable estimation using a high-resolution digital surface model," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 74, pp. 78–84, 2012.

[5] L. Waser, E. Baltsavias, K. Ecker, H. Eisenbeiss, E. Feldmeyer-Christe, C. Ginzler, M. Küchler, and L. Zhang, "Assessing changes of forest area and shrub encroachment in a mire ecosystem using digital surface models and cir aerial images," *Remote Sensing of Environment*, vol. 112, no. 5, pp. 1956–1968, 2008.

[6] I. Aicardi, M. Garbarino, A. Lingua, E. Lingua, R. Marzano, and M. Piras, "Monitoring post-fire forest recovery using multitemporal digital surface models generated from different platforms," *Earsel Eproceedings*, vol. 15, no. 1, pp. 1–8, 2016.

[7] D. Grigillo and U. Kanjir, "Urban object extraction from digital surface model and digital aerial images," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3, pp. 215–220, 2012.

[8] G. Priestnall, J. Jaafar, and A. Duncan, "Extracting urban features from lidar digital surface models," *Computers, Environment and Urban Systems*, vol. 24, no. 2, pp. 65–78, 2000.

[9] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] N. Snavely, "Bundler: Structure from motion (sfm) for unordered image collections," *http://phototour. cs. washington. edu/bundler/*, 2008.

[11] P. Moulon, P. Monasse, and R. Marlet, "Global fusion of relative motions for robust, accurate and scalable structure from motion," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3248–3255.

[12] C. Sweeney, "Theia multiview geometry library: Tutorial & reference," http://theia-sfm.org.

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

[14] 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.

[15] 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.

[16] E. Zheng and C. Wu, "Structure from motion using structure-less resection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2075–2083.

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

[18] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Eurographics Symposium on Geometry Processing*. The Eurographics Association, 2006, pp. 61–70.

[19] S. Fuhrmann, F. Langguth, and M. Goesele, "Mve-a multi-view reconstruction environment." in *GCH*, 2014, pp. 11–18.

[20] S. Yu and M. Lhuillier, "Incremental reconstruction of manifold surface from sparse visual mapping," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE, 2012, pp. 293–300.

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

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

[23] 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.

[24] L. von Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," *arXiv preprint arXiv:1804.05625*, 2018.

[25] S. Bu, Y. Zhao, G. Wan, and Z. Liu, "Map2dfusion: real-time incremental uav image mosaicing based on monocular slam," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4564–4571.

[26] J. Zienkiewicz, A. Tsiotsios, A. Davison, and S. Leutenegger, "Monocular, real-time surface reconstruction using dynamic level of detail," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 37–46.

[27] H. Eisenbeiß, "Uav photogrammetry," Ph.D. dissertation, ETH Zurich, 2009.

[28] A. Lucieer, S. M. d. Jong, and D. Turner, "Mapping landslide displacements using structure from motion (sfm) and image correlation of multi-temporal uav photography," *Progress in Physical Geography*, vol. 38, no. 1, pp. 97–116, 2014.

[29] F. Remondino and S. El-Hakim, "Image-based 3d modelling: a review," *The photogrammetric record*, vol. 21, no. 115, pp. 269–291, 2006.

[30] P. Beardsley, P. Torr, and A. Zisserman, "3d model acquisition from extended image sequences," in *European conference on computer vision*. Springer, 1996, pp. 683–695.

[31] A. W. Fitzgibbon and A. Zisserman, "Automatic camera recovery for closed or open image sequences," in *European conference on computer vision*. Springer, 1998, pp. 311–326.

[32] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[33] R. Gherardi, M. Farenzena, and A. Fusiello, "Improving the efficiency of hierarchical structure-and-motion," 2010.

[34] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[35] 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.

[36] 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.

[37] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[38] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[39] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 2100–2106.

[40] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[41] S. Bu, Y. Zhao, G. Wan, and K. Li, "Semi-direct tracking and mapping with rgb-d camera for mav," *Multimedia Tools and Applications*, vol. 75, pp. 1–25, 2016.

[42] 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.

[43] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[44] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[45] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *arXiv preprint arXiv:1708.03852*, 2017.

[46] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2043–2050.

[47] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Mapnet: Geometry-aware learning of maps for camera localization," *arXiv preprint arXiv:1712.03342*, 2017.

[48] G. L. Oliveira, N. Radwan, W. Burgard, and T. Brox, "Topometric localization with deep learning," *arXiv preprint arXiv:1706.08775*, 2017.

[49] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, vol. 2, no. 6, 2017, p. 7.

[50] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," *arXiv preprint arXiv:1803.02276*, 2018.

[51] T. Hinzmann, J. L. Schönberger, M. Pollefeys, and R. Siegwart, "Mapping on the fly: Real-time 3d dense reconstruction, digital surface map and incremental orthomosaic generation for unmanned aerial vehicles," in *Field and Service Robotics*. Springer, 2018, pp. 383–396.

[52] 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.