2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

DenseFusion: Large-Scale Online Dense Pointcloud and DSM Mapping for UAVs

Lin Chen, Yong Zhao, Shibiao Xu*, Shuhui Bu*, Pengcheng Han and Gang Wan

Abstract— **With the rapidly developing unmanned aerial vehicles, the requirements of generating maps efficiently and quickly are increasing. To realize online mapping, we develop a real-time dense mapping framework named DenseFusion which can incrementally generates dense geo-referenced 3D point cloud, digital orthophoto map (DOM) and digital surface model (DSM) from sequential aerial images with optional GPS information. The proposed method works in real-time on standard CPUs even for processing high resolution images. Based on the advanced monocular SLAM, our system first estimates appropriate camera poses and extracts effective keyframes, and next constructs virtual stereo-pair from consecutive frame to generate pruned dense 3D point clouds; then a novel real-time DSM fusion method is proposed which can incrementally process dense point cloud. Finally, a high efficiency visualization system is developed to adopt dynamic levels of detail (LoD) method, which makes it render dense point cloud and DSM smoothly. The performance of the proposed method is evaluated through qualitative and quantitative experiments. The results indicate that compared to traditional structure from motion based approaches, the presented framework is able to output both large-scale high-quality DOM and DSM in real-time with low computational cost.**

I. Introduction

A rapid overview of designated areas has become a growing demand in some industries, such as agriculture, disaster monitoring, urban planning, etc. DOM and DSM are efficient tools for representing the environment. An orthomosaic gives a two-dimensional overview of the surroundings and it can be made by stitching multiple undistorted aerial photos together [1]. However, a true orthophoto requires a three-dimensional model of the scene. DSM represents the earth's surface and includes all objects on it, which is commonly used for three-dimensional model representation. Through DSM, it's easy to map pixels observed by the perspective camera to their location with respect to the orthographic camera.

The fundamental methods for DSMs and orthomosaics are mainly based on structure from motion (SfM) at present, which requires high computing and memory resources. Normally, it takes hours to get the final results and when it comes to larger scale scenarios, the computation time increases unacceptably. In order to resolve the problem, Wang *et al.* propose TerrainFusion [2] which incrementally generates DSM with high efficiency and real-time speed. However, the method only deals with sparse point cloud, and the resolution of the result is relatively low.

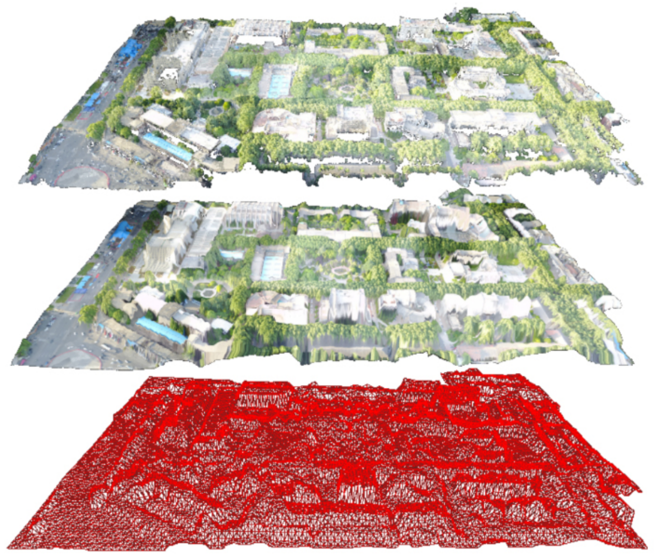


Fig. 1. Our method efficiently generates dense maps with high speed. The top subfigure is dense point cloud; The middle is digital surface model; and the bottom is reconstructed mesh.

To solve above problems, we propose a novel framework called DenseFusion. With well designed pipeline and optimized algorithm, our framework is capable of processing high definition images in real-time. A demo reconstruction example can be found in Fig. 1, and the overall processing flow is shown in Fig. 2. To achieve real-time speed, SLAM tech is adopted to estimate camera pose. The speed of multi view geometry (MVG) based dense point cloud reconstruction methods didn't meet the real-time requirements, we therefore constructed virtual stereo-pairs to obtain the dense depth information, nevertheless, the accuracy of the original dense point cloud was not enough. Therefore, a prune algorithm was applied to remove low accuracy points. Inspired by Terrainfusion [2], we further improved the fusion algorithm to run at least four times faster, enabling it to generate DSM and DOM from dense point cloud in real-time. In summary, contributions are summarized as follows:

- We present a highly modularized framework that incrementally generates dense point cloud, DSM and DOM in real-time based on monocular SLAM, which is able to process the sequential aerial images.
- A novel DSM fusion method is proposed which can process dense point cloud incrementally. In the method,

Shibiao Xu and Shuhui Bu are the corresponding authors (shibiao.xu@nlpr.ia.ac.cn,bushuhui@nwpu.edu.cn).

Lin Chen, Yong Zhao, Shuhui Bu and Pengcheng Han are with Northwestern Polytechnical University, 710072 Xi'an, China

Shibiao Xu is with Institute of Automation, Chinese Academy of Sciences, 100190 Beijing, China

Gang Wan is with the School of Aerospace Information, Aerospace Engineering University, Beijing 101416, China.

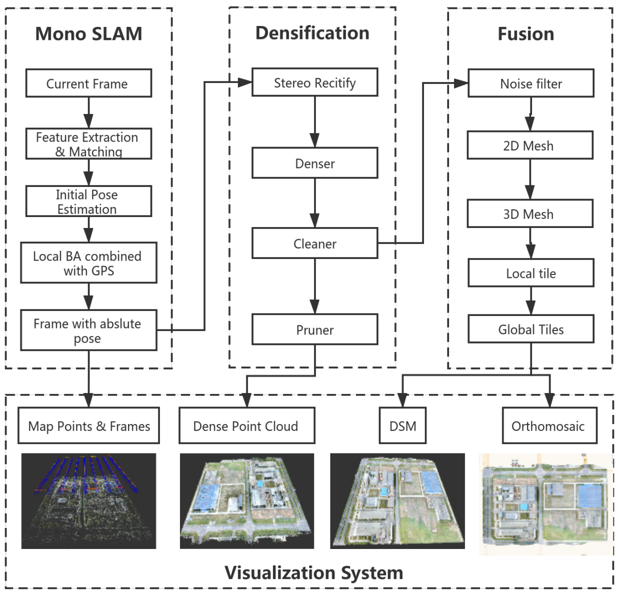


Fig. 2. DenseFusion: Real-time large-scale dense mapping framework with orthomosaic and digital surface model for UAVs.

efficient filtering and interpolation algorithm ensure high speed, while the multiband algorithm guarantees sufficient visual quality.

- Data management system and visualization system are developed to handle large-scale scenario. Levels of detail (LoD) ensures smooth rendering of dense point cloud and DSM, meanwhile the least recently used (LRU) algorithm makes the scenario scale limit only related to the hard disk size.

II. Related Works

In this section, we introduce related works with respect to three fields: visual SLAM, stereo reconstruction and digital surface model generation.

A. *Visual SLAM*

SLAM technology develops rapidly and a variety of SLAM systems have been proposed in the past years. From working principle, they are classified as feature-based [3], direct [4]–[6], and semi-direct [7], [8] methods. In recent years, with the rising of deep learning, there are some SLAM frameworks based on it: supervised [9]–[11], and unsupervised [12], [13] methods.

Some related methods are briefly described. ORB-SLAM2 [3] relies on the well-designed pipeline and efficient ORB features to achieve accurate localization, in addition it has a dedicated thread to perform global optimization. As for direct monocular SLAM, DTAM [14] achieves real-time dense reconstruction on GPU by using short baseline stereo matching and a regularization method, which make the depth estimation of low texture area smoother. LSD-SLAM [4] tracks depth values only on gradient areas, which makes it efficient to run direct SLAM in real-time on a CPU.

Considering the camera carried by the UAVs is monocular, we focus on monocular SLAM approaches in this paper. As for feature-based monocular SLAM, ORB-SLAM2 [3] has overall better accuracy and robustness. Therefore, our method adopts the feature-based method and local optimization to realize the pipeline. Furthermore, inspired by ORB-SLAM2, some ideas and tricks are applied to our method. However, in order to obtain accurate scale and generate geo-referenced data at the same time, GPS information guided joint optimization is also studied.

B. *Stereo Reconstruction*

Stereo matching for disparity estimation is the function of finding the pixels in the multi-view that correspond to the same 3D points in the environment. Because it can provide fundamental information, it is an active research topic in computer vision field. Stereo matching algorithms can be divided into local matching techniques [15]–[17] and global energy minimization methods [18]–[20].

Usually, local methods deal efficiently with large scale images, because they restrict the search domain to a small interval. Bleyer *et al.* [15] proposed slanted support windows and computed 3D planes at the level of pixels and projects the support region onto the local 3D plane. Hirschmüller *et al.* [16] proposed semi-global stereo matching (SGM). Only a small subset of possible interaction paths is considered. ELAS [17] is a near constant time local stereo matching algorithm. It searches robust matches called supporting points from rectified stereo images and generates 2D meshes from them by Delaunay triangulation. Then the algorithm uses Bayesian inference with the assumption that the disparity of a pixel is independent of all other pixels on the reference image given the disparities of the triangle corners it belongs to. For the pixels inside the triangles the matching is done in constant time.

C. *Digital Surface Model Generation*

In recent years, some researches [2], [21], [22] have attempted to adopt SLAM to photogrammetric survey for obtaining orthoimage and DSMs. Zienkiewicz *et al.* [21] reconstructed surface by performing depth-map and color fusion directly into a multi-resolution triangular mesh. This method is implemented on the GPU for achieving real-time speed, and only reconstructs height field without texture. Both Hinzmann *et al.* [22] and Wang *et al.* [2] implemented a mapping framework that incrementally generates a geo-referenced point cloud, a digital surface model and an orthomosaic. However, Hinzmann's method directly projected the dense point cloud onto the grid to generate the elevation map, which may result in a slight color deviation around stitching line. Wang's method used sparse point cloud as input data, which leads to poor performance on complex terrain. Different from above discussed methods, in this research dense point clouds are adopt to generate DSM. To realize real-time map generating, feature-based SLAM is adopted to estimate camera pose and attitude, and then virtual cameras among consecutive images and stereo matching are adopt to estimate dense depth. Finally high performance dense fusion is applied for fusing high quality DSMs.

III. Methodology

The overall processing flow is shown in Fig. 2. The system consists four modules: monocular SLAM, densification, fusion, and visualization. They are organized in separated modules and data transfer through message passing.

A. SLAM Framework

A visual SLAM system is used in our framework for estimating a robust pose for each frame. For the convenience and extendibility, we implemented a SLAM plugin based on GSLAM framework [23].

1) Localization: For each frame, a SiftGPU [24] feature extractor is used to detect key points and calculate descriptors. Once some effective descriptor matching are adopted, the relative pose is obtained by singular value decomposition (SVD) of essential matrix. And then, with the help of GPS information, the map is transformed to the Earth-Centered, Earth-Fixed (ECEF) rectangular coordinate. At the meantime, we can get the 3D map points from the triangulation of matching pairs. After the map is initialized, the upcoming frame's pose is calculated by solving the perspective n-point (PnP) problem from pairs of 3D map points and 2D key points.

2) Optimization: GPS information is generally available in aerial images, which improves the accuracy of poses and map points while GPS information can be joint optimized. In addition, GPS information makes the map have real scale which make the big problem exist in traditional monocular SLAM become easily resolved.

In this research, the local optimization focused on two adjacent frames. As illustrated in Fig. 3, the loss function contains two parts: reprojection error $\boldsymbol{e}_r$ and GPS error $\boldsymbol{e}_g$. $\boldsymbol{e}_r$ represents the sum of the reprojection errors between the matching point pairs ($\boldsymbol{x_i}$&$\boldsymbol{x_i^{'}}$).

$$\boldsymbol{e}_r = \arg\min_{\boldsymbol{\xi}} \frac{1}{2}\sum_{i=1}^{n} \left\| \boldsymbol{x}_i^{'} - \hat{\boldsymbol{x}}_{\boldsymbol{i}} \right\|_2^2 \tag{1}$$

$\boldsymbol{\pi_i}$ is the image plane, $\boldsymbol{x_i^{'}}$ stands for the corresponding key point in current frame. $\hat{\boldsymbol{x}}_{\boldsymbol{i}} = \frac{1}{s_i}\boldsymbol{K}\exp\left(\boldsymbol{\xi}^{\wedge}\right)\boldsymbol{P}_i$, $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ means camera internal parameter matrix, $\boldsymbol{\xi} \in \mathfrak{se}(3)$ represents the camera pose, which is given as a member of Lie algebra [25], and $\boldsymbol{P}_i \in \mathbb{R}^3$ means map points, which are either triangulated by $\boldsymbol{x_i}$ and $\boldsymbol{x_i^{'}}$, or calculated previously.

The GPS error $\boldsymbol{e}_g$ is defined as:

$$\boldsymbol{e}_g = \left\| \boldsymbol{t}_{GPS} - \boldsymbol{t}_{SLAM} \right\|_2 . \tag{2}$$

The form of $\boldsymbol{e}_g$ has a simple form, which is a norm of difference between the displacement provided by SLAM and observed GPS information.

The overall error function is defined as:

$$\boldsymbol{e} = \boldsymbol{e}_r + \boldsymbol{\alpha}\boldsymbol{e}_g . \tag{3}$$

In order to avoid the large difference between $\boldsymbol{e}_r$ and $\boldsymbol{e}_g$, a weight coefficient $\boldsymbol{\alpha}$ is applied to GPS error item. For aerial images, through our evaluation experiments $\boldsymbol{\alpha}$ is empirically set to 100.

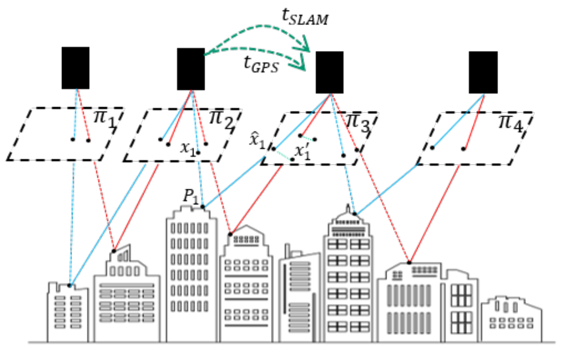


Fig. 3. Reprojection error and GPS error. For two consecutive frames, the earlier is the reference frame, and the other is the current frame.

In this research, parameters are estimated by using the nonlinear optimization algorithm the Ceres library. After localization and optimization, we can publish the frame with accurate pose information and sparse depth information to the densification module for dense depth estimation.

B. Densification

Densification step aims to process frames to generate accurate depth map. It contains three parts: denser, cleaner and pruner. Denser performs stereo rectification and matching. When the depth information of each pixel is obtained, cleaner filters the point cloud to reduce inconsistent points and pruner eliminates redundant points.

1) Denser: normally, the real-time dense point cloud generation is based on stereo camera. However, the flight height of aerial surveying and mapping is generally hundreds or up to thousands meters, so the traditional stereo camera is not suitable for estimating accurate depth because of the limited baseline length. Therefore, we construct a virtual stereo-pair from two consecutive frames and obtain the new images by stereo rectification. In the proposed framework, Bouguet's rectification method [26] is used to maximize the common area of the left and right views. After rectification, the poles are at infinity and the optical axes of the two cameras are parallel. The heights of correspondence points pair on the left and right images are the same, which greatly improve the matching efficiency.

To obtain dense disparity maps, we use ELAS [17], which is a classical approach for fast stereo matching of high resolution imagery. It builds a prior on the disparities by forming a triangulation on a set of supporting points which can be robustly matched. These triangle matching pairs greatly reduce the disparity search space, yielding accurate dense reconstructions without global optimization.

After the disparity map is converted to the depth map, the frame is sent to cleaner module.

2) Cleaner: due to the disparity map is not accurate enough, there are errors in depth map. We use the consistency constraints of multiple consecutive images to refine the depth. In this research, consistency reflects the correlation

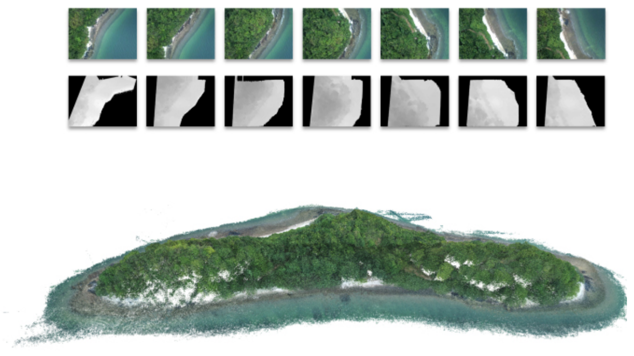


Fig. 4. Dense point cloud from images. At the top are original images, the middle are depth maps and bottom is a fused dense point cloud.

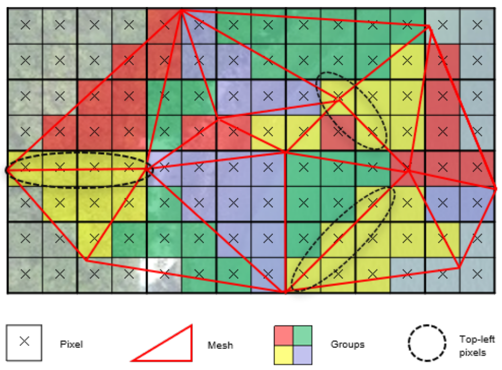


Fig. 5. DEM pixel allocation strategy. Different colors of each pixel represent different triangular patches. Top-left rule ensures that every pixel can only be assigned once, thus greatly speeding up the interpolation speed.

between pixels, when correlation is high, pixel depth has higher accuracy.

After receiving the depth map, it is inserted to a queue. When the number of maps in the queue exceeds N (usually odd), the consistency check is started. We take the middle frame in the queue as the current frame, convert its depth map into 3D points and reproject them to each of the $N-1$ other views, resulting in a 2D coordinate $\boldsymbol{p}_i$ and a depth value $\boldsymbol{d}_i^{'}$. A match is considered consistent if $(\boldsymbol{d}_i - \boldsymbol{d}_i^{'})/\boldsymbol{d}_i \leq f_{eq}$, where $\boldsymbol{d}_i^{'}$ represents the depth value in $\boldsymbol{p}_i$ and $\boldsymbol{d}_i$ is declared as the depth value of 3D points in other perspectives. If the depth in at least f_{con} other views is consistent with the current view, the corresponding pixel is accepted. Otherwise, it is removed. For all accepted points, the depth is set to the mean value of the consistent depth in other views to suppress noise. In this paper, N is 7, the threshold f_{eq} is 0.01 and f_{con} is 4.

After the consistency check, the last map in queue is popped out and sent to fusion module to generate DSM and pruner module to eliminate the redundancy concurrently.

3) Pruner: the purpose of prune is to remove redundant points in adjacent frames for reducing computation burden of subsequent map fusion. It contains a queue, like a cleaner, that starts to prune when the number of frames in the queue exceeds N. Taking the $N/2$ frame as the current frame, every point with a depth greater than 0 is back projected into the 3D space and then reprojected into the depth map of the other $N-1$ frames, and the corresponding point depth is set to zero. At the end, we pop the last frame out of the queue, and publish its point cloud into visualization system. Figure 4 shows a point cloud result reconstructed from dataset *mavic-island*.

C. Local Tiles

In order to improve the processing performance, orthomosaic and DSM are organized by tiles in this paper. The position of a geo-referenced tile is defined as follows: the global map is divided into N levels, and N_{th} level represents that the world is divided into $2^N \times 2^N$ tiles. Therefore, we can use $(x, y, z) \in Z^+$ to uniquely represent a tile. This three-dimensional vector represents the tile in column x and row y under the level of z. Our tile numbering scheme is that (0 to 2^z-1, 0 to 2^z-1) for the range $(-180, +85.0511)$ - $(+180, -85.0511)$ [27]. Each tile has three layers: depth layer for digital elevation model (DEM), weight layer for stitching and color layer for orthomosaic. Each tile has 256×256 pixels. A tile based coordinate system is a coordinate system that uses tile pixels to represents the world. In this section, we filter the dense point, split them into tiles, provide elevations and weights for these tiles and map texture onto them under level $N = 21$ ($0.138 m/pix$). In short, we generate local DSM and local orthomosaic here.

1) Points Filter: Because we only care about the elevation information of each location, and in order to speed up the calculation, we first filter the dense point cloud to filter out those points with the same location but have higher depth.

2) Mesh Generation: To rapidly generate geo-referenced 3D mesh for each key frame, we start with applying Delaunay triangulation to the 2D points for obtaining the 2D triangular mesh M_{img} on the image plane. Correspondingly, since the 3D points correspond to the 2D points one by one, we also get the mesh M_{depth} in the 3D space. To make the fusion more efficient, we convert the irregular 3D triangular mesh M_{depth} from ECEF coordinate system into 3D tile coordinate system $M_{depth}^{'}$. For generating orthomosaic later, the elevation information of $M_{depth}^{'}$ is discarded to get a 2D triangular mesh M_{ortho} in 2D tile coordinate system.

3) Local DEM Generation: The depth of the DEM is calculated by interpolating the meshes in $M_{depth}^{'}$. Because our point cloud is dense enough, using linear interpolation can still ensure enough accuracy.

During interpolation process, the central position of a pixel determines which triangle it is assigned to. Pixels at the edge of a triangle are assumed to be inside if it passes the top-left rule. The top-left rule is that a pixel center is defined to lie inside of a triangle if it lies on the top edge or the left edge of a triangle. In the method, the definition of edge types are: 1) A top edge, is an edge that is exactly horizontal and is above the other edges. 2) A left edge, is an edge that is not exactly horizontal and is on the left side of the triangle. A triangle can have one or two left edges.

The top-left rule ensures that adjacent triangles are drawn once. Figure 5 shows examples of pixels that are drawn because they either lie inside a triangle or follow the top-left rule.

4) Orthomosaic Generation: Because the depth and position in image of each point are available, the orthomosaic generation becomes simple. Basically, it's a transformation from perspective to ortho. We can see from Section III-C.2) that M_{img} on the image plane and M_{ortho} are corresponding one by one, so we just project the texture in M_{img} to M_{ortho} to obtain a orthomosaic.

5) Weight Determine: We assume that depth value estimated near the center of the image is more accurate, a weight mask for following stitching is generate according the founding. The size of weight map has the same of input image. It is calculated as follows: the central of weight map is set to 255, while the four corners are set to 0. As the pixel distance d is between 0 and d_{max}, the weight value is set corresponding to the distance $w_i = 255 \times (1 - d_i/d_{max})$.

D. Weighted Multiband Rendering

After the above process, orthomosaic, DEM and weight map are all settled and divided into rectangular tiles. Consequently, a tile contains several parts including a DEM patch, an orthomosaic patch, and a weight map patch. Our framework fuse local tiles into global tiles in an incremental manner, therefore the global tiles need to be updated after each local tile. To achieve a high-quality stitching and blending, a weighted multiband rendering algorithm is purposed for smooth transition in color and depth.

For each local tile, we can find the corresponding global tile according to the position. If the global tile is unsettled, copy the local tile into it. If it's not, two laplacian pyramids are expanded from the DEM, orthomosaic and a Gaussian pyramid is expanded from weight map. Our Laplacian pyramid contains 5 levels and each level L_n is computed through the following steps: 1) compute a 5 levels Gaussian pyramid $G(img)$. 2) L_n is the difference between G_n and the upsampling of G_{n+1}.

$$\begin{aligned} G_{n+1} &= DOWN\left(G_n \otimes \mathcal{G}_{5\times5}\right) \\ L_n &= G_n - UP\left(G_{n+1}\right) \otimes \mathcal{G}_{5\times5} \end{aligned} \tag{4}$$

Here $\otimes$ is a convolution symbol, $\mathcal{G}_{5\times5}$ is a Gaussian Kernel. The highest level L_5 equals to G_5 since there is no higher level G_6 computed. As a result, we have four Laplacian pyramids: $L(DEM_l), L(ortho_l), L(DEM_g), L(ortho_g)$ and two Gaussian pyramids: $G(weight_l), G(weight_g)$.

Then, a mask pyramid is computed by comparing $G(weight_l)$ with $G(weight_g)$ at each level. Instead of directly stitching local tiles into global, we stitch Laplacian pyramids together according to the mask pyramid. In the end, the global DEM and orthomosaic are recovered by collapse the final stitched Laplacian pyramids from low to high:

$$G_n = L_n + UP\left(G_{n+1}\right) \otimes \mathcal{G}_{5\times5} \tag{5}$$

After the weighted multiband rendering algorithm, the exposure differences height deviation on DEMs are minimized, and misalignments are reduced.

E. Visualization System

The visualization system is implemented based on GSLAM visualization module [23]. It can dynamically display frames, dense point cloud, orthomosaic and DSM.

Our approach follows the philosophy to generate the best possible individual depth maps, and then merge them into the global point cloud directly. Nevertheless, each frame's depth data are stored separately for better data management performance. In the visualization system, we use the form of linked list to upload the point cloud of each frame to the video random access memory, which manage individual frame's data for avoiding repeatedly uploading the whole point cloud.

For rendering massive terrain data in real-time with high efficiency, we create 5-layer pyramids for each tile to dynamically display different resolution DSM according to the distance which greatly decreased the workload of graphics card.

Though above visualization improvements, our system provides smooth and fluent viewing and operational experience.

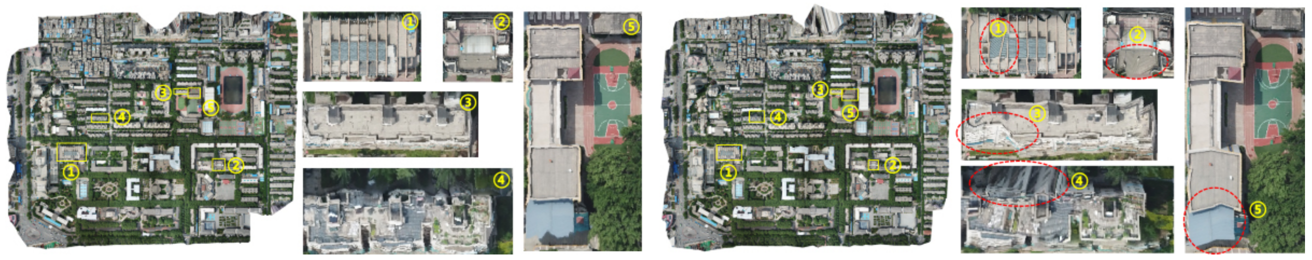
F. Data Management System

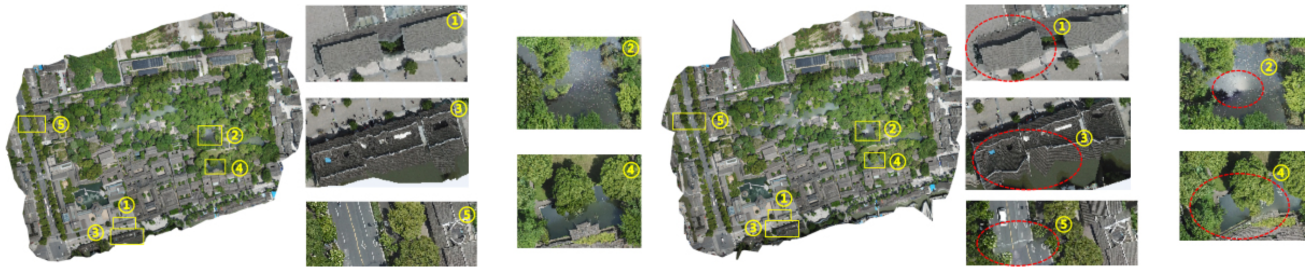
In order to handle large-scale scene, a LRU based tile management system is implemented in our work. For boosting the tile access speed and memory efficiency, two-level storage scheme is adopted which contains a fixed length queue and a hash map. The queue holds the most recently used tiles, and the hash map keeps the fast access index, where tile position as key and address of the corresponding queue node as value. If a given tile is required and it is existed in the queue, the system will directly return the node pointer of the tile, otherwise, the system will remove the least recently seen tile from the queue and hash map, then load it from disk to queue and hash map. That means the most recently accessed tile will be at the top of the queue whereas the least recently used tile will be at the end of the queue. Because the system only keep given amount of tiles in memory, it can handle very large scene which only limited by the size of hard disk.

IV. Experiments

We implement a real-time framework for generating dense point cloud, DSM, and orthomosaic in C++. In order to evaluate the performance of the proposed method, several experiments are performed. We first show several results of our framework, and then demonstrate the speed, performance and calculation resource consumption on different environments (city, hill, island, etc.) compared with the state-of-the-art commercial softwares Pix4DMapper and TerrainFusion. Since currently available dataset does not contain ground-truth depth, we use the error between different methods to analyze the accuracy of proposed method. The results show that our system achieves high robustness, comparative quality, and more efficient DSM reconstruction.

All experiments are tested on a desktop PC with an Intel i7-6700 CPU, a 16 GB RAM, and a GTX 1060 GPU. We


(a) *mavic-campus*


(b) *mavic-garden*

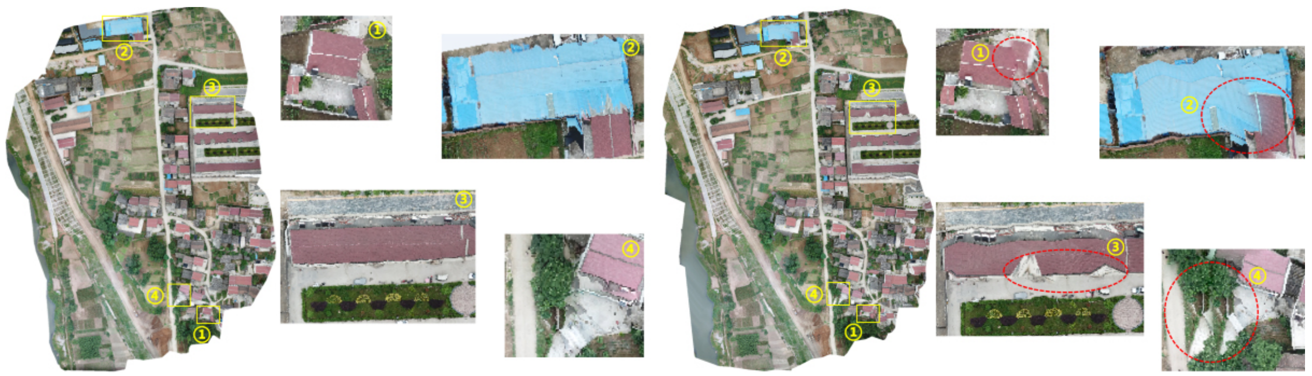
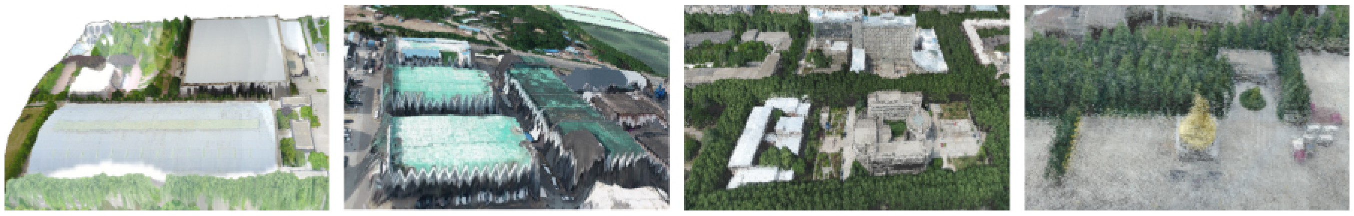

(c) *mavic-village*

Fig. 6. Orthomosaic results comparison between DenseFusion (left) and TerrainFusion (right). TerrainFusion generates acceptable result in low elevation offset areas but outputs twisted details for buildings. Benefit from the dense reconstruction, details of DenseFusion looks much better.


(a) DenseFusion

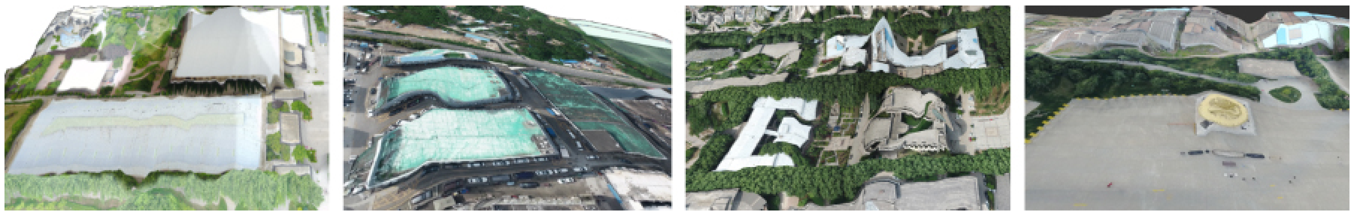

(b) TerrainFusion

Fig. 7. Comparison on DSM. The first two columns show the DSM performance between our method (top) and TerrainFusion (down), and the last two columns show the reason for this difference: our method generates DEM from dense point cloud, which is more detailed than sparse point cloud.

run our approach and TerrainFusion in a 64-bit Linux system, while Pix4DMapper executes on 64-bit Windows 10.

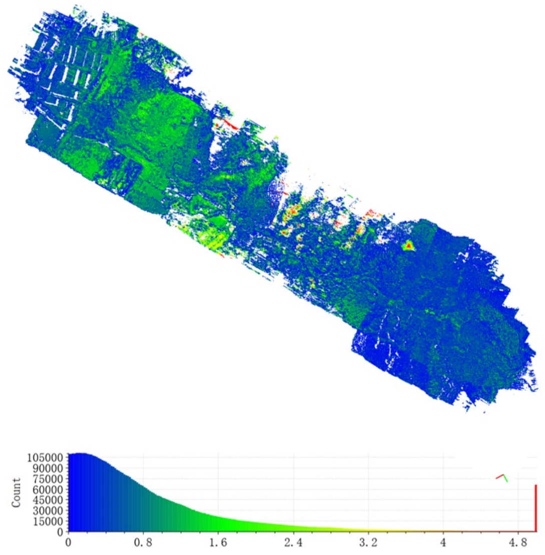
A. *Visual Comparison*

The visual effect comparison is shown in Fig. 6 and Fig. 7. we can clearly see advantages of our approach

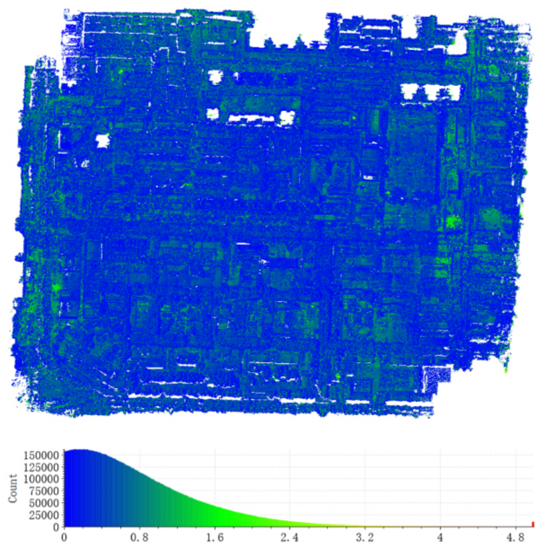
TABLE I

TIME STATISTICS IN SECONDS FOR DATASETS NPU-DRONEMAP.

Sequence	Images	Resolution	Height (m)	Area (km^2)	TerrainFusion	Pix4D	Ours
mavic-campus	293	4000*3000	213	0.974	368.182	4707.0	**92.13**
mavic-fengniao	109	4000*3000	60	0.073	377.302	3303.0	**44.31**
mavic-warriors	95	4000*3000	143	0.157	218.324	1623.0	**27.11**
mavic-yangxian	165	4000*3000	100	0.113	171.719	2935.0	**40.89**
mavic-factory	359	4000*3000	153	0.545	1003.89	4811.0	**106.23**
mavic-huangqi	229	4000*3000	188	0.478	809.678	4351.0	**74.94**
p4r-village	136	4864*3648	250	1.076	288.272	2762.0	**61.50**
phantom4-mountain	81	4864*3648	300	0.929	173.598	1752.0	**39.90**
phantom3-olathe	160	4000*3000	122	0.257	161.295	2514.0	**67.26**



(a) *phantom4-mountain*



(b) *mavic-campus*

Fig. 8. Error distribution (left) and statistics (right) between our method and Pix4D. In dataset (a), 90% of the dense point cloud of our method is no more than 1.91 m away from Pix4d's, and 50% is no more than 0.63 m. In dataset (b), 90% of the dense point cloud of our method is no more than 1.72 m away from Pix4d's, and 50% is no more than 0.65 m.

over TerrainFusion in generating DSMs. In farmland, hills and other areas with low inclination, TerrainFusion can produce good results. But when it applies to villages or cities, TerrainFusion may not be satisfactory: the walls is tilted, and small details is smoothed, which is unacceptable for most applications. Compared with TerrainFusion, our method could achieve more reasonable results on buildings, trees. The reason is that our framework keeps the depth information as much as possible, while TerrainFusion only uses the depth information at key points, and the weighted multiband rendering algorithm makes the stitched part of the result smoother and more natural.

B. Computational Performance

The computation performance of the proposed method and related methods are compared. Both DenseFusion and TerrainFusion operate at tile level 21 (the lower computation time claimed in [2] is due to the lower tile level). The result demonstrate that our method is much faster than state-of-the-art commercial software and is able to process about 2-3 high resolution images in one second. The top-left rule ensures that every pixel can only be assigned once, which reduces redundant computing, and well-designed pipeline deals with a large number of data in parallel. This makes the proposed method several times faster than TerrianFusion, while maintaining better performance.

C. Precision Comparison

In precision comparison, some experiments are conducted with a variety of scenarios. We use point cloud distance between different methods to evaluate the accuracy of our methods. Theoretically, the accuracy of Pix4D, which is based on offline SfM method, is higher than our online method. Therefore, the comparison with Pix4D can reflect the accuracy of our method to a certain extent. We use the method based on least square plane to get the distance between point clouds. The target point cloud is generated by our method, and the reference point cloud which generated by Pix4D.

Figure 8 shows the error distributions and statistics between our method and Pix4D. Scenario (a) mainly contains villages, mountains and trees, while scene (b) mainly includes urban buildings and streets. From statistic charts we find that in dataset (a), 94.47% points' errors are less than 2.0 m, and 69.93% are less than 1.0 m; in dataset (b), 94.26% points' errors are less than 2.0 m, and 68.77% are less than

1.0 m.

The accuracy is affected by the stereo matching result, and the selection of a series of reliable supporting points ensures that the results will not be too bad; what's more, an effective filter can filter out outliers and guarantee point cloud quality.The comparison results demonstrate that the propose method can output acceptable accuracy for most applications which require real-time processing.

V. Conclusion

In this paper, we present a framework that can incrementally generate DSM, orthomosaic and dense point cloud at the same time with real-time speed. The framework has the following advantages: 1) Fast: It can produce real-time maps at 2-3 frames per second with high resolution, which is four times faster than TerrainFusion, forty times faster than Pix4D; 2) Accurate: The accuracy of our method is close to Pix4D. The point cloud shows the outline of the building well, and even the street lights. Compared with TerrainFusion, it has more details and is closer to the real scene; 3) True orthomosaic: Homography-based orthomosaic generation is only suited for planar scenery or high flight altitudes. Our point cloud based orthomosaic renders a true orthomosaic taking the surface model and optimal viewing angle into considered and still capable of real-time performance.

Although the proposed method achieves better performance overall, it still has some limitations. Because the generated DSM is 2.5D, the reconstruction results of occluded areas, vertical areas are not satisfactory. In the following research, several improvements will be considered: 1) Stereo matching is a relatively time-consuming process and is easy to parallelize, so we may consider using GPU to speed up the calculation. 2) Implementing a real-time 3D reconstruction based on TSDF.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (Nos. 61573284, 91646207, 61620106003, 61971418, 61771026 and 61671451), the National Key R&D Program of China (Grant 2018YFB2100602), and the Research Funds for Interdisciplinary Subject, NWPU.

References

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

[2] W. Wang, Y. Zhao, P. Han, P. Zhao, and S. Bu, "Terrainfusion: Real-time digital surface model reconstruction based on monocular slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7895–7902.

[3] 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.

[4] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[5] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1449–1456.

[6] 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, 2017.

[7] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[8] 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, 2016.

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

[10] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-aware learning of maps for camera localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2616–2625.

[11] G. L. Oliveira, N. Radwan, W. Burgard, and T. Brox, "Topometric localization with deep learning," in *Robotics Research*. Springer, 2020, pp. 505–520.

[12] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[13] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1983–1992.

[14] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2320–2327.

[15] M. Bleyer, C. Rhemann, and C. Rother, "Patchmatch stereo-stereo matching with slanted support windows." in *Bmvc*, vol. 11, 2011, pp. 1–11.

[16] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2007.

[17] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian conference on computer vision*. Springer, 2010, pp. 25–38.

[18] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1. IEEE, 2003, pp. I–I.

[19] ——, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[20] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

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

[22] 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.

[23] Y. Zhao, S. Xu, S. Bu, H. Jiang, and P. Han, "Gslam: A general slam framework and benchmark," *arXiv preprint arXiv:1902.07995*, 2019.

[24] C. Wu, "Siftgpu: A gpu implementation of scale invariant feature transform (sift)(2007)," *URL http://cs. unc. edu/~ ccwu/siftgpu*, 2011.

[25] J. Byrnes and G. Ostheimer, *Computational Noncommutative Algebra and Applications: Proceedings of the NATO Advanced Study Institute, on Computatoinal Noncommutative Algebra and Applications, Il Ciocco, Italy, 6-19 July 2003*. Springer Science & Business Media, 2006, vol. 136.

[26] J.-Y. Bouguet, "Camera calibration toolbox for matlab (2008)," *URL http://www. vision. caltech. edu/bouguetj/calib_doc*, vol. 1080, 2008.

[27] J. Schwartz. (2018, Feb.) Bing maps tile system. [Online]. Available: https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system