Real-time Monocular Dense Mapping on Aerial Robots Using Visual-Inertial Fusion

Zhenfei Yang, Fei Gao, and Shaojie Shen

***Abstract*— In this work, we present a solution to real-time monocular dense mapping. A tightly-coupled visual-inertial localization module is designed to provide metric and high-accuracy odometry. A motion stereo algorithm is proposed to take the video input from one camera to produce local depth measurements with semi-global regularization. The local measurements are then integrated into a global map for noise filtering and map refinement. The global map obtained is able to support navigation and obstacle avoidance for aerial robots through our indoor and outdoor experimental verification. Our system runs at 10Hz on an Nvidia Jetson TX1 by properly distributing computation to CPU and GPU. Through onboard experiments, we demonstrate its ability to close the perception-action loop for autonomous aerial robots. We release our implementation as open-source software[1].**

I. Introduction

Micro aerial vehicles (MAVs) are playing an important role in various tasks like exploration, surveillance and search and rescue because of their extreme agility. To enable MAVs to operate autonomously in cluttered indoor and outdoor environments, the robots must be equipped with sufficient perceptual ability to simultaneously localize themselves and sense the surrounding environments. This problem is referred to simultaneous localization and mapping (SLAM) and is especially challenging for aerial robots where the payload and power consumption are strictly limited. In this case, a monocular visual-inertial sensor suite consisting of one camera and an inertial measurement unit (IMU) is an appealing sensor choice because of its superior size, weight, and power (SWaP) characteristics. Considering that laser finders and stereo cameras are not applicable due to their weight and size, respectively, and RGB-D cameras perform poorly in bright environments, monocular visual-inertial sensor suites could be the best choice for MAVs.

Historically monocular visual-inertial localization in GPS-denied environments has been well studied, and the performance of state-of-the-art approaches is satisfactory both in terms of accuracy and efficiency. However, for the other part of the problem beyond localization – mapping, rare research has demonstrated real-time dense mapping that is sufficient to assist autonomous missions such as obstacle avoidance, motion planning and so on. Successful attempts either use more confident sensing modules such as RGB-D or stereo cameras or lower the computational demands by working on a sparse map or moving the computation to a base station.

All authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. {zyangag,fgaoaa}@connect.ust.hk, eeshaojie@ust.hk

[1] https://github.com/dvorak0/VI-MEAN

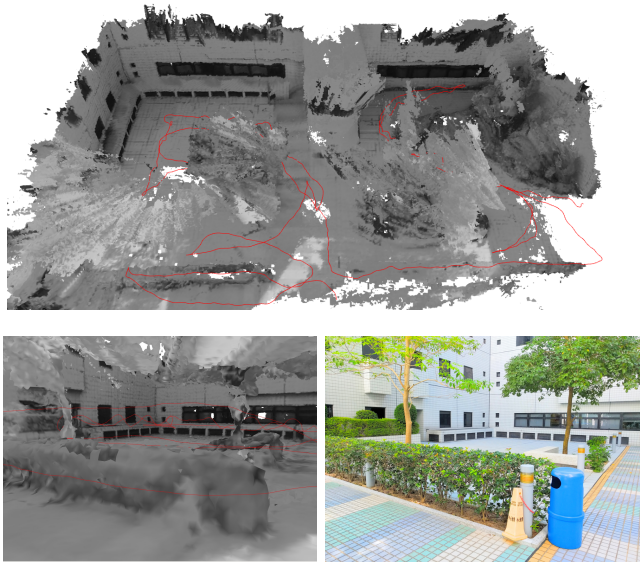


Fig. 1: Dense 3D model reconstructed by our system. A closer look of the dense reconstruction, as well as a snapshot of the scene, are shown in the second row. The global map, which is represented using sparse truncated signed distance field, is generated real-time on an Nvidia Jetson TX1 to support autonomous flight. The meshing is *offline* only for visualization. A video demonstration of more examples is available in supplementary material [2].

In this work, we design a system that uses only one camera and an IMU to reconstruct the surrounding environment with very high density. The system is implemented on an embedded platform with real-time performance by properly distributing the computation to CPU and GPU. Our system comprises three major modules: localization, which provides high-accuracy odometry; motion stereo, which generates depth maps locally; and truncated signed distance field (TSDF) fusion, which maintains the global map. We design these three modules cooperatively by analyzing their intrinsic properties.

Onboard experiments prove our system's ability to support autonomous flight for MAVs. In the supplementary material[2] we demonstrate how a quadrotor follows a trajectory generated online to reach a target point while avoiding obstacles with the help of our system. We believe this is a significant step towards equipping MAVs with full autonomy.

Our contributions are summarized as follows:

- We design and implement a dense mapping system including localization, local depth measurements, global

[2] http://www.ece.ust.hk/~eeshaojie/icra2017zhenfei.mp4

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

map fusion that runs real-time on an Nvidia Jetson TX1. We release our system as open-source software [1].

- We exploit the intrinsic structure of the tightly-coupled visual-inertial localization module to make it better serve motion stereo, unlike the majority of existing mapping algorithms, which treat the localization system as a black box.
- We propose a motion stereo algorithm that integrates information from highrate video input and follow it with a semi-global regularization to produce depth maps in high quality, but without losing real-time efficiency.
- The output of our system – a scalable global map – can directly support autonomous flight, which is verified through outdoor experiments on an aerial robot.

II. Related Work

There is a rich body of scholar work on real-time dense mapping with both RGB and RGB-D cameras.

An impressive result is given by KinectFusion [1], which takes the depth map measured by a Kinect as input, simultaneously localizes the camera itself using iterative closest point (ICP), and fuses all of the measurements into a global map using TSDF fusion.

While RGB-D cameras provide almost outlier-free depth measurements, they are not able to work in outdoor environments. This limitation encourages researchers to explore RGB camera-based dense reconstruction. RGB cameras are small, cheap, and widely equipped, making them a perfect choice for mobile dense reconstruction.

MonoFusion [2] and MobileFusion [3] present real-time dense reconstruction with a single web camera, but in a very small workspace. MobileFusion takes more consideration of real-time performance by making use of mobile GPU, which we believe is a growing trend because mobile GPUs are developing rapidly and many computer vision tasks could benefit from parallel computing.

Research into semi-dense mapping, where only pixels with high gradients are taken into consideration, has been active recently [4, 5]. Although the results are impressive and the algorithms run real-time on a CPU, we concern whether such a density is enough for safe navigation.

Another line of research is to start from a sparse or semi-dense map and grow toward dense reconstruction [6, 7]. However, it is not possible to achieve equal density with the limitation of real-time performance since the computational complexity also grows rapidly as the density increases.

A major disadvantage of the aforementioned RGB camera-based approaches is the lack of scale information. A solution to this is to integrate the inertial measurements from an IMU.

3D Modeling on the Go [8] might be the closest approach to ours. The localization part is an implementation of MSCKF [9], and plane sweeping is used to generate depth maps. After a number of filtering steps aimed at removing noise, the local maps are fused into a global map using volumetric fusion. A voxel-hashing-based storage strategy [10] guarantees its memory efficiency and hence scalability.

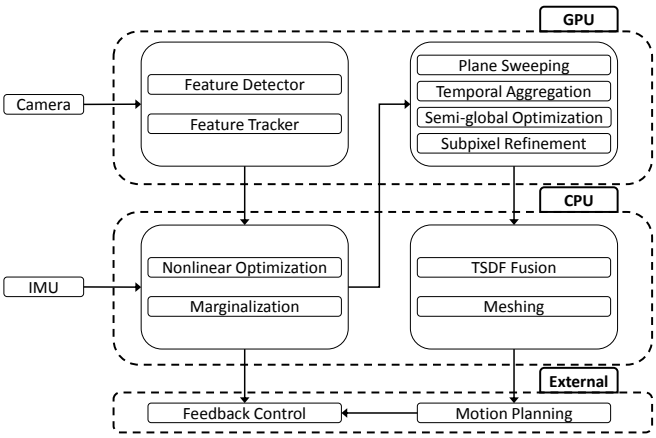


Fig. 2: System diagram: modules are distributed to the CPU and GPU according to their computational types. The GPU takes charge of tasks that are parallel in nature. And the CPU is suitable for sequential operations. The last optional block is one of the possible applications of our system. With feedback provided by the localization module, a closed-loop controller receives commands from a motion planning module and navigates a MAV.

Different to [8], we choose a graph-optimization-based localization algorithm instead of the filter-based MSCKF, and although we also use plane sweeping for cost volume calculation, we do not design filters to remove noise. Instead a smoothness prior is introduced by doing semi-global optimization over the cost volume, producing high-quality depth estimation. A similar prior is also enforced by DTAM [11] and REMODE [12] through total variation minimization. However, the convex optimization used in DTAM and REMODE requires a desktop GPU. The algorithm choices for all modules affect each other and can not be determined individually. We will detail later the principles of designing the algorithms for each module and how they cooperate with each other.

III. System Overview

Our system comprises three modules, as illustrated in Fig. 2. In the localization module, the images and inertial measurements from one camera and an IMU are fused under a probabilistic framework (Sect. IV). In the motion stereo module, image-pose pairs are received from the localization module and are used to generate local depth measurements (Sect. V). The final map, which will assist autonomous missions for aerial robots, is obtained by fusing the frame-rate local depth measurements in a memory-efficient way (Sect. VI). Optionally, given the odometry provided by the localization module and the trajectory generated by the motion planning algorithm to follow, a feedback controller can be used to close the perception-action loop to navigate the robots. In our experiments, we implement all system modules to show the usability of our approach.

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

IV. Tightly-coupled Visual-Inertial Localization

Compared with loosely-coupled visual-inertial localization [13, 14], tightly-coupled approaches [9, 15]–[19] show impressive accuracy at the cost of slightly higher computational demands. We can categorize tightly-coupled approaches as filtering-based [9], or graph optimization-based [15]–[19]. MSCKF [9] and GtSAM[3] [15] are recognized as the state-of-the-art methods based on filtering and optimization, respectively. MSCKF maintains a number of states using stochastic cloning [20] and uses the visual measurements to update the states under an EKF framework. GtSAM builds and optimizes the graph incrementally by exploring its sparsity and performing lazy linearization of the nonlinear measurements. We adopt a modified version of our previous work [18], where we only optimize the states within a sliding window and maintain constant computational complexity using two-way marginalization [19].

In this section, we describe our approach and emphasize that it is more suitable to the mapping task, while all of these three approaches perform well in localization task.

The definition of the full state for a sliding window with N IMU frames and M features is (the transpose is ignored)

$$\begin{aligned}\mathcal{X} &= \left[\mathbf{x}_n,\, \mathbf{x}_{n+1},\, \cdots \mathbf{x}_{n+N-1},\, \mathbf{x}_c^b,\, \lambda_m\, \lambda_{m+1}\, \cdots\, \lambda_{m+M-1}\right] \\ \mathbf{x}_k &= \left[\mathbf{p}_{b_k}^w,\, \mathbf{v}_{b_k}^w,\, \mathbf{q}_{b_k}^w, \mathbf{a}_{b_k}, \boldsymbol{\omega}_{b_k}\right] \\ \mathbf{x}_c^b &= \left[\mathbf{p}_c^b,\, \mathbf{q}_c^b\right],\end{aligned} \tag{1}$$

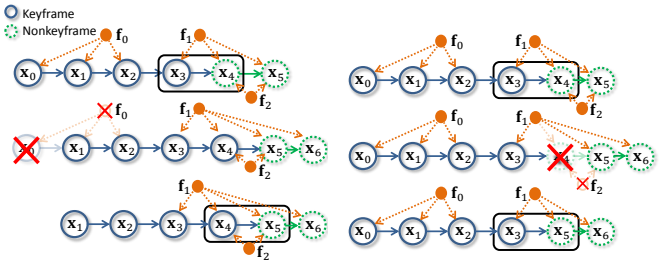
where the k^{th} IMU state consists of the position $\mathbf{p}_{b_k}^w$, velocity $\mathbf{v}_{b_k}^w$, orientation $\mathbf{q}_{b_k}^w$ of the body frame b_k with respect to the world frame w, and IMU bias $\mathbf{a}_{b_k}$, $\boldsymbol{\omega}_{b_k}$ expressed in the body frame b_k; 3D landmarks are parameterized using their inverse depth λ when first observed in the camera frame c; and $\mathbf{x}_c^b$ is the 6-DOF extrinsic transformation from the camera frame c to the body frame b.

The estimation is formulated as a nonlinear least-square problem:

$$\min_{\mathcal{X}}\{\sum_{k\in\mathcal{B}}\left\|r_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k},\,\mathcal{X})\right\|_{\mathbf{P}_{b_{k+1}}^{b_k}}^2 + \sum_{(l,j)\in\mathcal{C}}\rho\left(\left\|r_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j},\mathcal{X})\right\|_{\mathbf{P}_l^{c_j}}^2\right)\}, \tag{2}$$

where $r_{\mathcal{B}}(\cdot)$ and $r_{\mathcal{C}}(\cdot)$ are the nonlinear residual functions for the inertial and visual measurements, $\|\cdot\|_{\mathbf{P}}^2$ is the Mahalanobis distance weighted by $\mathbf{P}$, and $\rho(\cdot)$ is the robust Huber norm used to resist incorrect feature correspondences. To be specific, $r_{\mathcal{B}}(\cdot)$ constrains each pair of successive IMU frames b_k and b_{k+1} using the integration of inertial measurements $\hat{\mathbf{z}}_{b_{k+1}}^{b_k}$ (with covariance $\mathbf{P}_{b_{k+1}}^{b_k}$) from the IMU. $r_{\mathcal{C}}(\cdot)$ builds the connection between landmarks and IMU states through projecting the 3D landmark λ_l onto the camera frame c_j at the 2D observation $\hat{\mathbf{z}}_l^{c_j}$ (with covariance $\mathbf{P}_l^{c_j}$). We refer readers to [21] for the detailed derivation of nonlinear uncertainty propagation and efficient optimization on manifold.

[3]GtSAM is a generalized optimization framework, but here we use it to refer to the approach by Forster et al. [15] since the source code is now part of GtSAM .



(a) Back Marginalization (b) Front Marginalization

Fig. 3: An example with seven IMU states $\mathbf{x}_k$ and three features λ_l. Fig. 3a shows the procedure where the oldest keyframe $\mathbf{x}_0$ is back-marginalized because the visual parallax between the latest keyframe $\mathbf{x}_3$ and the latest non-keyframe $\mathbf{x}_5$ is large enough. After that, the *second-latest* non-keyframe $\mathbf{x}_4$ switches to a keyframe and $\mathbf{x}_5$ remains a non-keyframe. Otherwise, as shown in Fig. 3b, the *second-latest* non-keyframe $\mathbf{x}_4$ is front-marginalized. The black box illustrates the reference and measurement image pair to be sent.

Two-way marginalization [19] happens *after* optimization. It tries to keep enough visual parallax between keyframes, preventing the monocular algorithm from producing unreliable estimates under special motion patterns (hovering or constant velocity motion). The strategy back-marginalizes the oldest keyframe when the visual parallax between the latest keyframe and the latest non-keyframe is large enough, and then the *second-latest* non-keyframe switches to a keyframe. Otherwise, we front-marginalize the *second-latest* non-keyframe. The latest keyframe is chosen as the reference frame in motion stereo. Specially, we choose the *second-latest* non-keyframe with bidirectional constraints as the measurement frame, aiming at a more stable estimation result. A pictorial example of how the procedure maintains the structure of the estimator and how frames are selected for motion stereo is given in Fig. 3.

A fundamental assumption of motion stereo is that the relative odometry between reference and measurement frames is accurate enough. Even small error on poses will result in inaccurate epipolar lines and meaningless searches along the lines. While almost all mapping systems use the localization as a black box, we highlight the necessity of mining the structure of the estimator and selecting the proper frame pairs. It has been clearly pointed out that the global yaw angle and position are not observable in visual-inertial localization [22, 23]. Although GtSAM, MSCKF, and our approach all perform well in reducing drift over time, using them as a black box ignores that adjustments happen inside the estimator. To be specific, the adjustments could be caused by Gauss-Newton iteration in GtSAM and our method, or Kalman filter update in MSCKF. The issue is solved in our design by reporting a pair of odometries each time from the sliding window to the mapping module. The relative odometry within the sliding window is observable, and directly optimizing the reprojection error iteratively guarantees the accuracy. However, neither the lazy linearization in GtSAM nor the one-time linearization in MSCKF can achieve this.

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

V. Motion Stereo with Video Input

Spatial stereo with two cameras is popular in 3D dense mapping. The camera pair is pre-calibrated and the extrinsic transformation is assumed to be known. Two problems limit application of the algorithms designed for spatial stereo to motion stereo:

1) Commonly, spatial stereo algorithms force horizontal epipolar lines through rectification. The rectification brings efficient cost volume computation, but assumes the scene scale to be identical for two cameras. The assumption doesn't hold in motion stereo.
2) Spatial stereo processes image pairs independently, losing the temporal constraints introduced by video input. This inspires us to explore how to integrate the temporal information.

For each pair of reference image $\mathbf{I}_r$ and measurement image $\mathbf{I}_m$ we obtain their positions and orientations: ($\mathbf{R}_r = \mathbf{q}^w_{b_{n+N-3}}$, $\mathbf{T}_r = \mathbf{p}^w_{b_{n+N-3}}$) and ($\mathbf{R}_m = \mathbf{q}^w_{b_{n+N-2}}$, $\mathbf{T}_m = \mathbf{p}^w_{b_{n+N-2}}$) from the localization module. L front-parallel planes are enumerated for each pixel in the reference image $\mathbf{I}_r$. One entry, $\mathbf{Cost}[p_u][p_v][k]$, in the cost volume records the similarity between the pixel $\mathbf{p} = \begin{bmatrix} p_u & p_v \end{bmatrix}^\intercal$ in the reference image $\mathbf{I}_r$ and the corresponding pixel $\mathbf{p}'$ in the measurement image $\mathbf{I}_m$ under the assumption that it is on the k^{th} plane. The enumerated depth value is uniformly sampled in the inverse depth field. If the inverse depth of the nearest plane is D, the k^{th} inverse depth value we enumerate is kD. The similarity is calculated by first back-projecting the pixel $\mathbf{p}$ to 3D point $\mathbf{P}$ on the k^{th} plane with depth $d_k = \frac{1}{kD}$ using

$$\mathbf{P} = \mathbf{R}_r \left(d_k \cdot \mathbf{K}^{-1} \bar{\mathbf{p}} \right) + \mathbf{T}_r, \tag{3}$$

where $\bar{\mathbf{p}} = \begin{bmatrix} p_u & p_v & 1 \end{bmatrix}^\intercal$ is $\mathbf{p}$ in homogeneous coordinates, $\mathbf{K}$ is the intrinsic parameter, and then projecting the 3D point $\mathbf{P}$ to the measurement frame:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{K}\mathbf{R}_m^\intercal \left(\mathbf{P} - \mathbf{T}_m \right), \ \mathbf{p}' = \begin{bmatrix} x/z \\ y/z \end{bmatrix}. \tag{4}$$

To reduce the number of arithmetic operations on the GPU, we combine (3) and (4) to have:

$$\begin{aligned} \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= \mathbf{K}\mathbf{R}_m^\intercal \left(\mathbf{R}_r \left(d_k \cdot \mathbf{K}^{-1} \bar{\mathbf{p}} \right) + \mathbf{T}_r - \mathbf{T}_m \right) \\ &= \frac{1}{kD} \mathbf{K}\mathbf{R}_m^\intercal \mathbf{R}_r \mathbf{K}^{-1} \bar{\mathbf{p}} + \mathbf{K}\mathbf{R}_m^\intercal \left(\mathbf{T}_r - \mathbf{T}_m \right) \\ &= \frac{1}{kD} \mathbf{H}_1 \bar{\mathbf{p}} + \mathbf{H}_2 = \frac{1}{kD} \begin{bmatrix} -\mathbf{h}_{11}- \\ -\mathbf{h}_{12}- \\ -\mathbf{h}_{13}- \end{bmatrix} \bar{\mathbf{p}} + \begin{bmatrix} h_{21} \\ h_{22} \\ h_{23} \end{bmatrix} \\ \mathbf{p}' &= \begin{bmatrix} (\frac{1}{kD}\mathbf{h}_{11}\bar{\mathbf{p}} + h_{21}) / (\frac{1}{kD}\mathbf{h}_{13}\bar{\mathbf{p}} + h_{23}) \\ (\frac{1}{kD}\mathbf{h}_{12}\bar{\mathbf{p}} + h_{22}) / (\frac{1}{kD}\mathbf{h}_{13}\bar{\mathbf{p}} + h_{23}) \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{h}_{11}\bar{\mathbf{p}} + kDh_{21}) / (\mathbf{h}_{13}\bar{\mathbf{p}} + kDh_{23}) \\ (\mathbf{h}_{12}\bar{\mathbf{p}} + kDh_{22}) / (\mathbf{h}_{13}\bar{\mathbf{p}} + kDh_{23}) \end{bmatrix}. \end{aligned} \tag{5}$$

We repeat the procedure for both $\mathbf{p}$ and its neighborhood. Among various similarity metrics, we choose sum of absolute difference (SAD) for its efficiency. The intensity interpolation happens during the calculation of the similarity benefits from hardware bilinear filtering in texture memory to achieve subpixel accuracy without losing efficiency. The area covered by patches in the reference image and in the measurement image is not limited to be identical, which explains how plane sweeping naturally handles different scales.

The resulting cost volume is usually noisy, and so is the depth map generated by a simple winner-takes-all strategy. Using a larger patch can help get finer results by reducing the ambiguity, but at the expense of getting incorrect matches around edges. Instead, we use patches of a fixed size of 3×3 and propose a two-step method to improve the quality of depth maps: temporal cost aggregation and semi-global optimization on the temporal cost volume.

Cost aggregation aims to decrease the sensitivity to image noise by expanding the matching patch. The expansion in most existing approaches happens spatially by aggregating the matching cost in regions where the intensity values are similar. We propose to do expansion temporally by accumulating the matching cost from multiple measurement frames in different time instances. We switch reference frame and reset the cost volume only when the back-marginalization happens in the localization module. In the experiments, we will show how the temporal cost aggregation helps to increase the accuracy (Sect. VII-B).

However, even with a noiseless cost volume, the winner-takes-all strategy won't produce reliable estimation in the texture-less region. Following SGM [24], we introduce smoothness constraints on the depth map $\mathbf{D}$ by defining the global energy as

$$\begin{aligned} E(\mathbf{D}) = \sum_{\mathbf{p}} (& \mathbf{Cost}[p_u][p_v][\mathbf{D_p}] \\ & + P_1 \cdot \sum_{\mathbf{q} \in \mathcal{N}(\mathbf{p})} T\left[|\mathbf{D_p} - \mathbf{D_q}| = 1 \right] \\ & + P_2 \cdot \sum_{\mathbf{q} \in \mathcal{N}(\mathbf{p})} T\left[|\mathbf{D_p} - \mathbf{D_q}| > 1 \right]), \end{aligned} \tag{6}$$

where the cost for each pixel $\mathbf{p} = \begin{bmatrix} p_u & p_v \end{bmatrix}^\intercal$ at depth $\mathbf{D_p}$ consists of three kinds of error terms: the cost directly from the cost volume $\mathbf{Cost}$, the penalty caused by its neighbor pixels $\mathbf{q} \in \mathcal{N}(\mathbf{p})$ with which the depth difference is 1, and the penalty caused by its neighbor pixels $\mathbf{q} \in \mathcal{N}(\mathbf{p})$ with which the depth difference is larger than 1.

SGM [24] proposes to solve an approximated version of (6):

$$\begin{aligned} S(\mathbf{D}) = & \sum_{\mathbf{p}} \sum_{\mathbf{r}} L_{\mathbf{r}}(\mathbf{p}, \mathbf{D_p}) \\ L_{\mathbf{r}}(\mathbf{p}, k) = & \mathbf{Cost}[p_u][p_v][k] + \min(L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k), \\ & L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k-1) + P_1, \\ & L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k+1) + P_1, \\ & \min_i L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2), \end{aligned} \tag{7}$$

where the global cost function is simplified as a 1D problem $L_{\mathbf{r}}(\mathbf{p}, k)$ along direction $\mathbf{r}$ and the final cost $S(\mathbf{D})$ is the

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

averaged cost of multiple 1D problems. The approximated global cost function can be efficiently solved by running dynamic programming multiple times. With the final cost S, the depth value for each pixel is decided by choosing the entry corresponding to the lowest cost. To prevent getting discrete depth values, we further do subpixel interpolation by fitting a parabola using the matching cost of the two neighboring planes and taking the extreme point.

The semi-global optimization makes use of the measurements from both the texture-rich and texture-less areas to produce high-quality depth maps. The number of outliers in the depth map after SGM is low enough in practice that further post-processing is not necessary, saving the efforts of designing filters and tuning parameters.

A recently proposed two-step pipeline divides the problem into two sub-problems: per-pixel depth estimation using a 1D Kalman filter and outlier removal on the noisy depth maps [8, 12, 25]. The division may lead to suboptimal estimates because the information-rich cost volume is not accessible in the second step. In our approach, the optimization directly operates on the cost volume to produce optimal estimates.

VI. Uncertainty-aware TSDF Fusion

Instead of simply stitching all local depth maps, we maintain an occupancy map and continuously update it using uncertainty-aware TSDF fusion. Fusing the depth maps is not only to maintain a global map that is meaningful for autonomous flight, but also to filter the noisy measurements in a probabilistic way. Our implementation is based on the open-source software Chisel [10], with subtle but important modifications detailed below.

The surrounding environment is represented using an occupancy map, where the whole world is split into grids and each grid records the value of the signed distance and weight. The signed distance is the distance from the origin of the voxel to its nearest surface, and its sign is determined by whether the voxel is inside or outside the surface. With the help of the SDF, the surfaces are modeled as the zero-level set of the field. The SDF is usually truncated to filter out outliers that are inconsistent with historical measurements.

Observations with higher depths are usually more uncertain for both RGB-D and RGB mapping. We encode the uncertainty using the dynamic truncation distance. Suppose the standard deviation of the depth measurement in the inverse depth id is σ_{id}, then the standard deviation of the depth measurement in depth d is

$$\sigma_d = \sqrt{(\frac{\partial d}{\partial id})^2 \sigma_{id}^2}. \tag{8}$$

We set the constant parameter σ_{id} to be D, which is the minimal distance in inverse depth between enumerated planes. Then σ_d increases as the depth becomes larger and is used during the fusion procedure.

The fact that most space in the 3D world is, in reality, empty and meaningful grids only live around the surfaces inspires us to make use of the sparsity [26]. We only store the voxels around surfaces using a hash table. The hash table stores the mapping from the position of the voxel to its address in memory. This technique enables large-scale dense reconstruction.

VII. Experiments

A. Implementation Details

We implement the whole system on an Nvidia Jetson TX1[4], which has 4 ARM cores and 2 Maxwell SMs with a TDP of 10W. The cameras and IMU come from VI-Sensor [27], which has hardware camera-IMU synchronization and a pair of cameras with 11cm baseline. Only the left camera is used by our system, and the other one helps to generate ground truth using spatial stereo for comparison purpose (Sect. VII-B). The cameras have an approximately 100-degree field-of-view (FOV) and run at 20Hz. We downsample the images to 376×240 for mapping and use the original resolution (752 × 480) for localization. Approximated timing statistics for each component can be found in Table I. The flight platform is a DJI M100[5]. The Intel NUC equipped on the quadrotor is responsible for other tasks such as position controlling and trajectory generation. The communication between the Intel NUC and the Nvidia Jetson TX1 is handled by ROS[6] through Ethernet. Parameter setting and efficient CPU-GPU implementation are accessible from the released source code.

B. Comparison with Spatial Stereo

In this part, we compare our motion stereo with spatial stereo to evaluate our algorithm quantitatively.

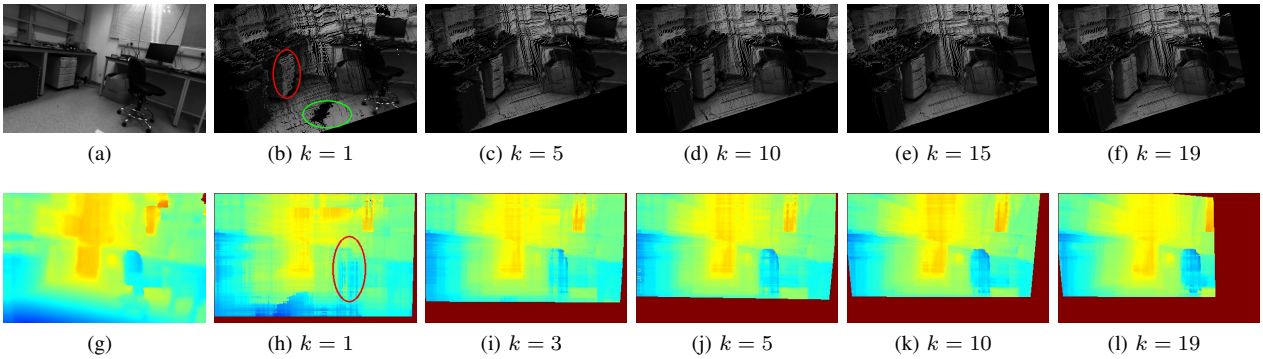
We choose the depth map generated by the state-of-the-art stereo matching algorithm MC-CNN [28], which shows impressive performance in public benchmarks (KITTI and Middlebury), as the ground truth. We run the pre-trained CNN in the *original* resolution for higher accuracy. The cross-check used in MC-CNN helps a lot to remove outliers caused by either image noises or occlusions. However, it is not suitable for our real-time implementation because it doubles calculation.

The data was collected in a typical lab environment, and contains challenging cases where the texture is repeated or

[4] http://www.nvidia.com/object/jetson-tx1-module.html
[5] http://www.dji.com
[6] http://www.ros.org

TABLE I: Approximated Timing Statistics

Unit	Modules	Time (ms)	Rate (Hz)
GPU	Feature Detector	30	10Hz
	Feature Tracker	20	20Hz
CPU	Nonlinear Optimization	50	10Hz
	Marginalization	15	10Hz
GPU	Cost Aggregation	20	10Hz
	Semi-global Optimization	50	10Hz
	Subpixel Refinement	10	10Hz
CPU	TSDF Fusion	50	10Hz
	Meshing (Optional)	> 100	< 1Hz



(a) (b) $k = 1$ (c) $k = 5$ (d) $k = 10$ (e) $k = 15$ (f) $k = 19$

(g) (h) $k = 1$ (i) $k = 3$ (j) $k = 5$ (k) $k = 10$ (l) $k = 19$

Fig. 4: Fig. 4a and Fig. 4g are the grayscale image and depth map generated by MC-CNN in the original resolution. 4b to 4f shows how the point cloud becomes finer as the number of measurements t increases. The accuracy of the point cloud can be visually verified by observing the geometric structure from a side view. The cabinet (circled in red) is distorted at the beginning and is recovered as temporal information is integrated. The same applies to the texture-less ground (circled in green). Also more details appear progressively in the depth map, (Fig. 4h to 4l), which can be inferred from the observation that the contours of the chair (circled in red) are better persevered.

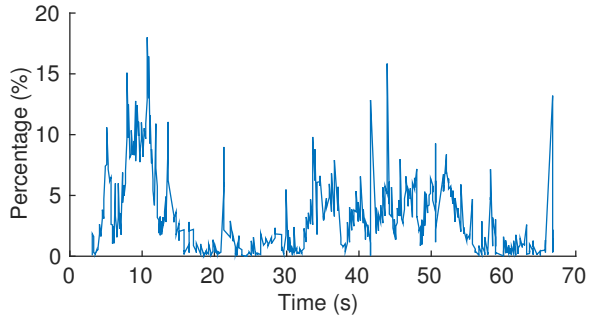


Fig. 5: Outlier ratio of motion stereo against MC-CNN. Before comparison, we calibrate the disparity-depth mapping. Pixels whose error is larger than 3 pixels are regarded as outliers, following the definition from KITTI. The overall outlier ration is 2.84%. The reason for the peak at $10s$ is the incorrect handling of reflection in both approaches.

less. We evaluate our motion stereo quantitatively for three metrics: outlier ratio, average disparity error, and density.

We follow KITTI[7] to define pixels whose disparity error against the ground truth is larger than 3 pixels as outliers. The disparity-depth mapping is calibrated to be consistent between motion stereo and spatial stereo. Fig. 5 presents the ratio curve during this experiment. The overall outlier ratio in this sequence is 2.84%, showing very comparable precision with the time-consuming CNN-based approach ($\sim 5s$ per frame with Intel i7-6700K and Nvidia GTX-1080), for which the percentage of erroneous pixels in non-occluded areas is 2.61% according to KITTI.

We compare the average absolute value of the disparity error when temporal cost aggregation is not involved ($k = 1$ measurement frame is used) and is involved ($k > 1$ measurement frames are used). As illustrated in Fig. 6a, the proposed temporal cost aggregation consistently decreases the error, which can also be confirmed visually in Fig. 4. The aggregation denoises the cost volume through averaging and reducing ambiguities. The ratio between valid depth measurements and all pixels in the downsampled image is given in Fig. 6b to show the density of our depth map. The temporal cost aggregation lowers density because we only consider pixels whose epipolar segment (determined by the enumerated depth values and relative odometry) does not exceed the image. Although the density is lowered, it is still double or even triple of that in semi-dense mapping, whose density is rarely higher than 25%. Sampled depth maps shown in Fig. 7 demonstrate how well the texture-less regions are estimated, which benefits from the smoothness enforcement. Finally, we visualize our reconstruction in Fig. 8.

[7] http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo

C. Autonomous Flight in Outdoor Environments

In this section, we present an outdoor experiment to demonstrate the capability of our system in supporting autonomous flight. One more module, which is responsible for motion planning, is integrated into the system, as shown in Fig. 2. The module receives the occupancy map and a target position as input and plans a trajectory that guarantees safety and is optimal in terms of minimal jerk [29]. The zero-level set in TSDF models the surfaces and hence tells whether a grid is occupied or not. The hashing-based sparse representation allows us to output the map in an incremental way, where only information about added or deleted grids is sent out. The motion planning module first samples points in the whole map to search for a feasible path, and then around the path a trajectory constrained by the safe space and dynamics of the flying robot is generated using convex optimization. Each time the module is notified about a change of the map, the feasibility of the generated trajectory is checked to determine the need for re-planning. Re-planning is suitable to our mapping system because the map is estimated online and the completeness and accuracy of the map grow as the

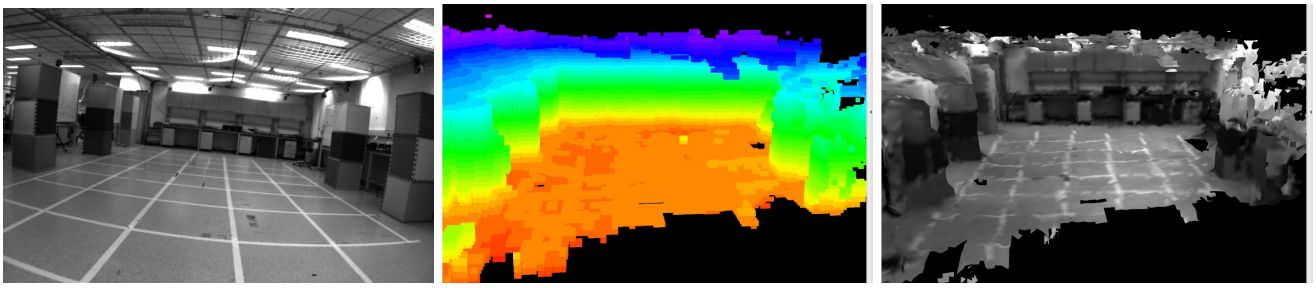
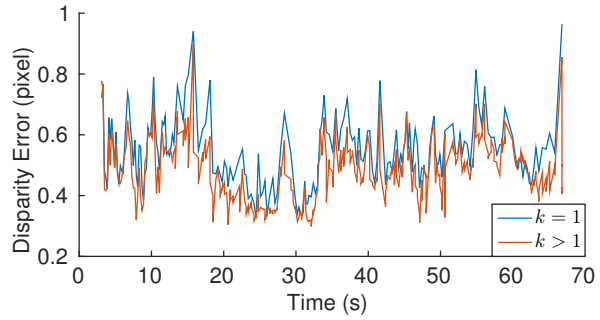
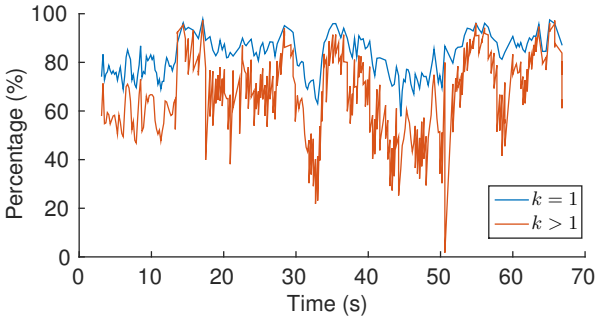


Fig. 8: Final global map: Fig. 8a is a snapshot of the environment; Fig. 8b is the occupancy map and the height is visualized using a color map; and Fig. 8c is the mesh map generated by running marching cubes on the TSDF. Note that the mesh is again be processed *offline*, because it is the occupancy map playing an essential role in assisting autonomous flight.



(a) Average Disparity Error



(b) Depth Map Density

Fig. 6: Effect of temporal cost aggregation is verified by comparing the result with $k = 1$ and $k > 1$ measurement frames. The average disparity error drops as the aggregation decreases the sensitivity to image noise through averaging. The density is lowered due to the shrinking of the shared FOV across measurement frames, but still remains to be much higher than that in sparse or semi-dense mapping.

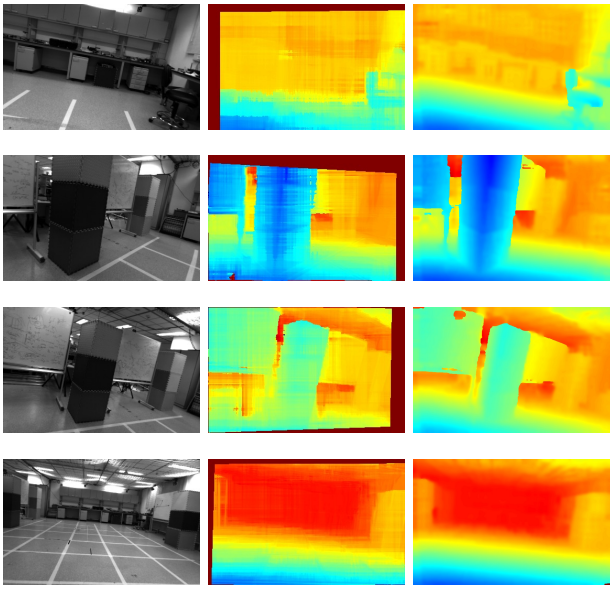


Fig. 7: Sampled results in a typical lab environment. The first, second, and third columns correspond to the grayscale image, the depth map generated by motion stereo, and the depth map generated by MC-CNN.

quadrotor observes more.

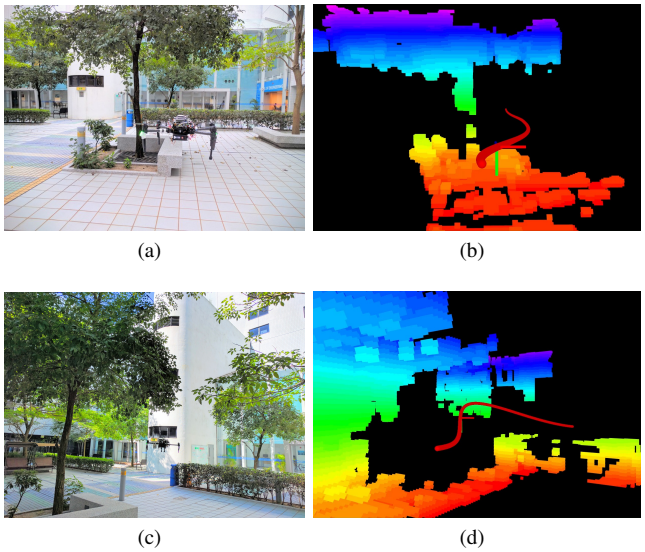
The experiment was designed to be carried out in a garden-like area to verify its performance in reconstructing both artificial obstacles (walls, doors) and natural obstacles (trees, bushes). The motion planning module always gives a trajectory with the latest map. We sample several pictures to show the procedure in Fig. 9. In Fig. 9b, the quadrotor takes off and obtains a rough estimation of the obstacles in front of it, and the motion planning module plans a trajectory to go around them. As the quadrotor flies forward, more obstacles are observed and the module decides to adjust the trajectory to avoid collision with the newly observed obstacles, as shown in Fig. 9d.

VIII. Conclusion and Future Work

In this work, we propose and implement a dense mapping system that is capable of assisting aerial robots equipped with a minimal sensor suite to operate autonomously in cluttered environments. Three major modules are designed cooperatively to take charge of localization, local depth measurement and global map maintenance. An application of our system – obstacle avoidance – is verified with onboard experiments in an outdoor environment and the details can be found in the supplementary video material.

A major limitation of our system is the strict assumption of a static scene. A possible solution to this is to modify the cost function defined in the optimization over the cost

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.



(a) (b) (c) (d)

Fig. 9: The motion planning module takes the occupancy map maintained by our system as input. Based on the latest map, the module generates a collision-free and dynamics-feasible trajectory (the red curve in Fig. 9b). As the quadrotor flies forward, more measurements complete the map and new obstacles are able to be observed. Re-planning will be triggered if the newly observed obstacles break the collision-free property. As a result, the motion planning module updates the trajectory (red curve in Fig. 9d). The detect-plan loop keeps running to guide the autonomous flight.

volume to model the possible motion of objects in the scene. Also, the current monocular system can be extended to a multi-camera version to improve the situational awareness by having a large FOV. In future work, we will focus on how to efficiently solve the mixed inter-camera and intra-camera motion stereo.

REFERENCES

[1] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. of the IEEE and ACM Int. Sym. on Mixed and Augmented Reality*, Oct. 2011, pp. 127–136.

[2] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche, "MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera," in *Proc. of the IEEE and ACM Int. Sym. on Mixed and Augmented Reality*, Oct. 2013, pp. 83–88.

[3] P. Ondruska, P. Kohli, and S. Izadi, "MobileFusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 11, pp. 1251–1258, Nov. 2015.

[4] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. of the European Conf. on Comput. Vis.*, Sep. 2014.

[5] R. Mur-Artal and J. D. Tardós., "Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM," in *Proc. of Robot.: Sci. and Syst.*, 2015.

[6] S. Pillai, S. Ramalingam, and J. J. Leonard, "High-performance and tunable stereo reconstruction," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.* IEEE, 2016.

[7] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian Conference on Computer Vision (ACCV)*, 2010.

[8] T. Schöps, T. Sattler, C. Häne, and M. Pollefeys, "3D Modeling on the Go: Interactive 3D reconstruction of large-scale scenes on mobile devices," in *Proc. of the Int. Conf. on 3D Vis.*, Oct. 2015, pp. 291–299.

[9] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Roma, Italy, Apr. 2007, pp. 3565–3572.

[10] M. Klingensmith, I. Dryanovski, S. S. Srinivasa, and J. Xiao, "CHISEL : Real time large scale 3D reconstruction onboard a mobile device using spatially-hashed signed distance fields," in *Proc. of Robot.: Sci. and Syst.*, Jul. 2015.

[11] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. of the IEEE Int. Conf. on Comput. Vis.*, Nov. 2011, pp. 2320–2327.

[12] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, May 2014, pp. 2609–2616.

[13] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Saint Paul, MN, May 2012, pp. 957–964.

[14] D. Scaramuzza et al., "Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robot. Autom. Mag.*, vol. 21, no. 3, 2014.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proc. of Robot.: Sci. and Syst.*, Rome, Italy, Jul. 2015.

[16] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial SLAM using nonlinear optimization," in *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, Jun. 2013.

[17] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robot. and Auton. Syst.*, vol. 61, no. 8, pp. 721–738, 2013.

[18] Z. Yang and S. Shen, "Monocular visual-inertial state estimation with online initialization and camera-IMU extrinsic calibration," *IEEE Trans. Autom. Sci. and Engineering*, pp. 1–13, 2016.

[19] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Seattle, WA, May 2015.

[20] S. I. Roumeliotis and J. W. Burdick, "Stochastic Cloning: A generalized framework for processing relative state measurements," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, vol. 2. IEEE, 2002, pp. 1788–1795.

[21] Z. Yang and S. Shen, "Tightly-coupled visual-inertial sensor fusion based on IMU pre-integration," Hong Kong University of Science and Technology, Tech. Rep., 2016, URL: http://www.ece.ust.hk/~eeshaojie/vins2016zhenfei.pdf.

[22] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *Int. J. Robot. Research*, vol. 30, no. 4, pp. 407–430, Apr. 2011.

[23] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *Int. J. Robot. Research*, vol. 30, no. 1, pp. 56–79, Jan. 2011.

[24] H. Heiko, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[25] W. N. Greene, K. Ok, P. Lommel, and N. Roy, "Multi-level mapping: Real-time dense monocular SLAM," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.* IEEE, 2016.

[26] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graphics*, 2013.

[27] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.* IEEE, 2014, pp. 431–437.

[28] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. Mach. Learning Research*, vol. 17, pp. 1–32, 2016.

[29] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point cloud," in *Proc. of the IEEE Int. Sym. on Safty, Security, and Rescure Robot.*, Oct. 2016, URL: http://www.ece.ust.hk/~eeshaojie/ssrr2016fei.pdf.