2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Semi-Dense Visual-Inertial Odometry and Mapping for Quadrotors with SWAP Constraints

Wenxin Liu, Giuseppe Loianno, Kartik Mohta, Kostas Daniilidis, and Vijay Kumar

Abstract— **Micro Aerial Vehicles have the potential to assist humans in real life tasks involving applications such as smart homes, search and rescue, and architecture construction. To enhance autonomous navigation capabilities these vehicles need to be able to create dense 3D maps of the environment, while concurrently estimating their own motion. In this paper, we are particularly interested in small vehicles that can navigate cluttered indoor environments. We address the problem of visual inertial state estimation, control and 3D mapping on platforms with Size, Weight, And Power (SWAP) constraints. The proposed approach is validated through experimental results on a 250 g, 22 cm diameter quadrotor equipped only with a stereo camera and an IMU with a computationally-limited CPU showing the ability to autonomously navigate, while concurrently creating a 3D map of the environment.**

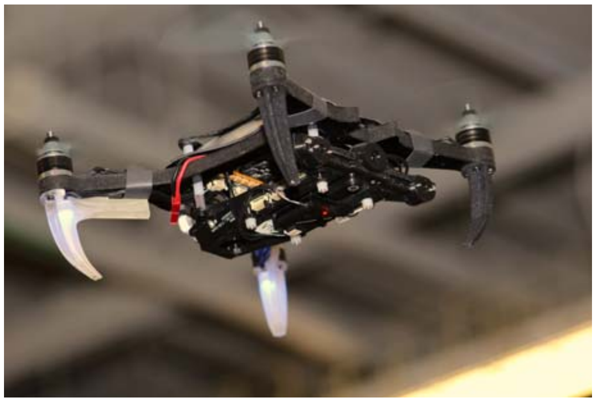
Fig. 1. A 22 cm, 250 g quadrotor with a forward-facing stereo camera pair.

I. Introduction

Micro Aerial Vehicles (MAVs) equipped with onboard sensors are ideal platforms to navigate in complex and confined environments for solving tasks such as exploration [1], inspection [2], mapping [3], interaction with the environment [4], [5], and search and rescue [6]. To guarantee fully autonomous navigation, these vehicles need to be able to navigate in 3D environment using onboard sensors such as cameras and IMUs (Inertial Measurement Units consisting of gyroscope and accelerometer) concurrently creating maps of the environment.

Recent research on MAVs has yielded a number of significant results in Simultaneous Localization and Mapping (SLAM) enabling navigation for MAVs. Results have been obtained using monocular cameras and IMUs [7]–[9], and stereo camera configurations [1], [10]. The combination of camera and IMU for localization purposes gained research interest due to their low cost and small form factor. Moreover, they represent two complementary sensors. Cameras are generally slow compared to the IMU, but image processing techniques are able to provide more accurate information. The IMU is fast (100 to 500 Hz), but due to noise and unbounded accumulation of integration errors, the vehicle's pose can be only estimated satisfactorily for no more than a few seconds. However the combination of camera and IMU can give odometry that is both accurate and fast enough for control.

This work was supported by Qualcomm Research, the ARL grant W911NF-08-2-0004, ONR grants N00014-07-1-0829, N00014-14-1-0510, ARO grant W911NF-13-1-0350, NSF grants IIS-1426840, IIS-1138847, DARPA grants HR001151626, and HR0011516850.

The authors are with the GRASP Lab, University of Pennsylvania, Philadelphia, USA. email: {wenxinl, loiannog, kmohta, kostas, kumar}@seas.upenn.edu.

Approaches for visual odometry are usually categorized into (a) feature-based or indirect methods [11], [12] and (b) photometric or direct methods [13]–[16]. Feature based methods extract corners and track them over multiple frames whereas, photometric methods directly use the intensity values of the image pixels. There are also methods that combine direct and indirect methods [17], [18], which make use of both the reprojection and intensity errors. Feature-based approaches for visual odometry have proved to work robustly in well textured areas, but direct methods are more robust in sparsely textured conditions [14].

To achieve fully autonomous navigation, we need visual odometry for state estimation, control for autonomous flight, mapping for operation in unknown environments and finally motion planning. When dense maps are required, we believe that direct methods are a better choice. In feature-based methods, there is a density limit of the selected feature points beyond which finding correspondences becomes computationally inefficient and algorithmically redundant. Feature extraction and the matching process restricts the feature points to be sparse, so mapping needs to be done separately, which results in an inefficient use of computational resources. In addition, the quality of the map depends only on the quality of dense stereo matching. There are existing works comparing different types of matching algorithms. Block matching algorithms are fast but the generated disparity map is noisy. Semi-global matching balances computational power and accuracy, but a GPU is required for real-time applications [19], and is currently infeasible to run on light weight processors such as ARM CPUs. Direct methods on the other hand, potentially allow the use of all pixels, providing a chance to create a denser map alongside tracking

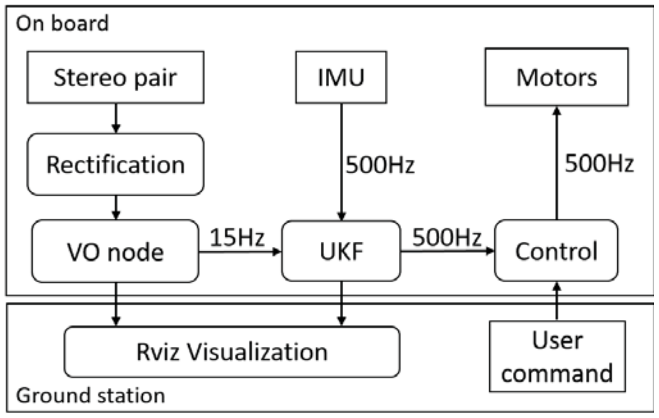

Fig. 2. Block diagram of our system.

by encoding dense or semi-dense feature points into the optimization framework [14].

Very few works have addressed the problem of flying aerial vehicles based on direct methods and concurrently creating a semi-dense map of the surrounding environment. In [20], the estimation is done online but mapping is carried out in a post processing step on collected data, whereas in [21], this problem has been addressed for large scale platforms using powerful computational units, and eventually GPUs [22] for mapping.

In this work, we are interested in autonomous flight for a small size quadrotor with SWAP constraints. These constraints only allow the use of a computationally-limited CPU and light weight sensors such as cameras and IMU. This work shows for the first time that direct visual odometry and semi-dense mapping can run concurrently for closed loop control on a limited computation board, allowing autonomous flight for a small size quadrotor. Our main contributions are:

- We show the use of direct visual odometry to estimate the motion of the vehicle leveraging a small baseline stereo camera.
- We show how to fuse this estimate with IMU data in a nonlinear filter to provide smooth state estimates for control.
- We analyze the various parameters affecting the system performance and discuss the trade-off between accuracy and CPU usage.

II. System Overview

We use a forward-facing stereo camera pair and IMU as the sensor suite which provides sufficient information for autonomous navigation. Stereo cameras have the advantage of avoiding scale drift [23] compared to monocular cameras, and forward-facing cameras give a good viewpoint for mapping. The stereo camera is used to compute the pose of the robot which is fused with the IMU to obtain high rate odometry for control. We leverage a small 8 cm baseline stereo camera configuration available on a low power and lightweight quadrotor with limited computational capabilities. Our platform of choice, shown in Fig. 1, is a 22 cm diameter, 250 g quadrotor using Qualcomm® Snapdragon™ Flight™.

Fig. 2 shows a block diagram of our system. The images captured by the forward-facing stereo camera are first rectified, then sent to the semi-dense visual odometry node where pose estimates and 3D point clouds are computed. Then a UKF node fuses the 15 Hz pose estimates and 500 Hz IMU data to obtain odometry used for control. All the computations are done on board in ROS environment, and 3D point clouds and odometry are published as ROS topics, which can be obtained and stored on the ground station for visualization purposes.

III. Direct Visual Odometry and Mapping

A. Direct Semi-Dense Visual Odometry

We model the stereo camera pair using a pinhole camera model [24]. The pair is calibrated to obtain camera intrinsic parameters, distortion coefficients and relative pose. The images are undistorted and rectified, and an image pyramid is created by blurring and sub-sampling to help achieve better convergence.

We use a keyframe based approach [11] to minimize drift while reducing the computational cost for the visual odometry. When a frame is selected to be a keyframe, new points to track are initialized by selecting high-gradient pixels and a disparity map is generated. New keyframes are created if any of the following conditions are met: 1) the percentage of points visible during tracking falls below 70%; 2) the number of frames processed without creating a new keyframe exceeds 50; 3) the angle between the current frame and the last keyframe exceeds 0.2 rad; 4) the translation from last keyframe exceeds 10% of the average scene depth; 5) if tracking is lost. We find the optimal relative pose between the current frame and the latest keyframe by minimizing the photometric error of the re-projected high-gradient points. We extract the high-gradient points for tracking only from the keyframe and calculate the photometric error using the points visible in the current frame.

In order to compute the photometric error from the re-projected points, we find the 3D positions by computing their disparities from the stereo camera. Disparity maps are generated by a block matcher for each image pyramid level and the 3D point positions can then be obtained using the known camera intrinsics. The disparity map requires the use of both camera images, but we only use the left camera images during the tracking phase.

1) Optimization problem formulation: For each new frame our objective is to find the optimum transformation from the latest keyframe to the current frame which we represent as $T(\boldsymbol{\xi}) \in SE(3)$. Here $\boldsymbol{\xi}$ denotes the 6-dimensional vector form of $\mathfrak{se}(3)$, and $T(\boldsymbol{\xi})$ can be obtained by the exponential map from $\mathfrak{se}(3)$ to $SE(3)$ [24]. We use $\boldsymbol{\xi}$ as the minimal parametrization for solving the non-linear optimization problem.

For each 3D feature point $\boldsymbol{p}_i$ visible in the current frame, we define its error term as a function of the relative pose $\boldsymbol{\xi}$ between the current frame and the last keyframe:

$$e_i(\boldsymbol{\xi}) = I^*(\pi(\boldsymbol{p}_i)) - I(\pi(T(\boldsymbol{\xi})\boldsymbol{p}_i)),$$

where I^* and I are pixel intensities in the keyframe and the current frame respectively, and π is the projection function from 3D points in camera frame coordinates to image pixel coordinates.

The cost function for the optimization is defined as the weighted inner product of the error vector $\boldsymbol{e}$, containing the error terms of all the tracked pixels, with the weights give by the information matrix Ω:

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} \boldsymbol{e}^\mathsf{T}\Omega\boldsymbol{e}.$$

2) Optimization scheme: To solve this optimization problem we use an iterative method called the Inverse Compositional Approach (ICA) [25]. The i^{th} row of the $n \times 6$ Jacobian is calculated by linearizing the i^{th} pixel error with respect to the pose $\boldsymbol{\xi}$ as follows:

$$e_i(\boldsymbol{\xi} \oplus \boldsymbol{\delta}) = I^*(\pi(T(\boldsymbol{\delta})\boldsymbol{p}_i)) - I(\pi(T(\boldsymbol{\xi})\boldsymbol{p}_i)).$$

The operation $\oplus$ is defined as $T^{-1}(T(\boldsymbol{\xi}) \cdot T(\boldsymbol{\delta}))$. Note that instead of putting the increment $\boldsymbol{\delta}$ on the current frame term, it is put on the keyframe so that the jacobian only depends on the gradients of the keyframe image. This jacobian is constant for all frames following that keyframe, enabling us to compute it only once per keyframe [25]. In each iteration, the error vector is recomputed with the new pose, and we find $\boldsymbol{\delta}$ by computing the minimum of the linearized cost function:

$$H = J^T\Omega J, \quad \boldsymbol{b} = J^T\Omega\boldsymbol{e}, \quad \boldsymbol{\delta} = -H^{-1}\boldsymbol{b},$$

where J is the $n \times 6$ jacobian matrix. At each iteration, the pose is updated as follows:

$$T(\boldsymbol{\xi}) \leftarrow T(\boldsymbol{\xi}) \cdot T(\boldsymbol{\delta})^{-1}.$$

Instead of using Gauss-Newton approach described above, we use Levenberg-Marquardt Algorithm [26] to improve convergence. We run this optimization in a coarse-to-fine manner on the image pyramid and use the optimized pose from coarser level as an initial guess for the next level to obtain faster and more robust convergence.

B. Semi-dense Mapping

The map is generated by accumulating the triangulated high gradient pixels from each keyframe once its pose is optimized. The pixels we use for mapping are the same pixels we use for tracking, creating a semi-dense map along with the tracking process. We use a simple block matcher to obtain disparity maps with low CPU usage. Due to the simplicity of the block-matching algorithm, the resulting disparity map is noisy. By selecting only the high-gradient points in the image, we use the points that are more accurate in the disparity map and avoid most of the noisy pixels.

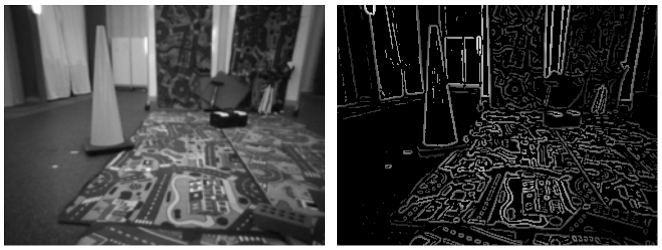
Fig. 3. The left image shows the grayscale image, and the right image shows only the high gradient pixels selected by the system.

IV. Visual Inertial data Fusion

To enable onboard control it is necessary to fuse the camera pose estimate with the IMU to obtain a reliable high rate state estimate of the vehicle. We use an UKF to estimate the full state of the vehicle at 500 Hz as described in [9]. Very briefly, the state is represented by

$$\boldsymbol{x}_f = \begin{bmatrix} \boldsymbol{x}^\mathsf{T} & \boldsymbol{v}^\mathsf{T} & \Phi^\mathsf{T} & \boldsymbol{b}_a^\mathsf{T} \end{bmatrix}^\mathsf{T},$$

where $\boldsymbol{x}$, $\boldsymbol{v}$ are the position and linear velocity of the robot in the world frame respectively, Φ is the quaternion representing the orientation of the robot in the world frame and $\boldsymbol{b}_a$ are the accelerometer biases. The prediction step uses the linear acceleration and angular velocity measurements given by the IMU as input. The visual odometry pose estimates are then used to update the state using a linear measurement model. The measurement model is linear since the visual odometry output consists of the absolute position and orientation of the vehicle which is part of the filter's state.

To compensate for the measurement delay due to image processing, the filter also keeps a history buffer of past IMU and state values. When it receives a new measurement from visual odometry, the filter uses the measurement's timestamp to find the corresponding data in the IMU and state buffers, applies the measurement update and integrates the IMU data from that time onwards [27].

V. Platform Control

The position controller uses the estimated state from the UKF as feedback to follow trajectories given as an output of a high-level trajectory planner. In many previous works, a backstepping approach is used for UAV control because the attitude dynamics can be assumed to be faster than the dynamics governing the position and so linearized controllers are used for both loops [28]–[30]. However, we need the system to be capable of large deviations from the hover configuration to take advantage of the agility that is inherent with small platforms. Therefore, we use a nonlinear controller with a larger basin of attraction. The moment $\boldsymbol{M} \in \mathbb{R}^3$ along the three axes of the body-fixed frame and the thrust $\tau \in \mathbb{R}$ are control inputs of the robot. For control, we build on the work in [31] and [32] with 500 Hz feedback control inputs chosen as

$$\begin{aligned}\tau &= (-k_x\boldsymbol{e}_x - k_v\boldsymbol{e}_v + mg\boldsymbol{e}_3 + m\ddot{\boldsymbol{x}}_d) \cdot R\boldsymbol{e}_3 = \boldsymbol{f} \cdot R\boldsymbol{e}_3,\\ \boldsymbol{M} &= -k_R\boldsymbol{e}_R - k_\omega\boldsymbol{e}_\omega + \boldsymbol{\omega} \times J\boldsymbol{\omega} - J\left(\boldsymbol{\omega} \times R^\mathsf{T}R_c\boldsymbol{\omega}_c\right),\end{aligned}$$

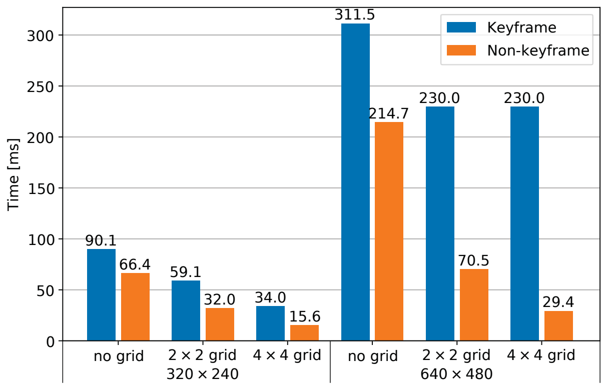

Fig. 4. Average running time of keyframe and non-keyframe with different resolutions and grid sizes. Note that the keyframe processing time for the 2×2 and 4×4 grid sizes with 640×480 resolution is the same since it is dominated by the disparity computation time which is the same for both.

with $\ddot{\boldsymbol{x}}_d$ is the desired acceleration and k_x, k_v, k_R, and k_Ω are positive definite terms. The quantities $\boldsymbol{e}_x, \boldsymbol{e}_v, \boldsymbol{e}_R, \boldsymbol{e}_\Omega$ are the position, velocity, orientation and angular rate errors respectively, as defined in [31], [32]. The subscript C denotes a commanded value, with $R_C = \begin{bmatrix}\boldsymbol{b}_{1,C}, & \boldsymbol{b}_{2,C}, & \boldsymbol{b}_{3,C}\end{bmatrix}$ representing the commanded orientation, and Ω_C the commanded angular velocity. These are calculated as

$$\boldsymbol{b}_{2,des} = \begin{bmatrix}-\sin\psi_{des}, & \cos\psi_{des}, & 0\end{bmatrix}^\top, \ \boldsymbol{b}_{3,C} = \frac{\boldsymbol{f}}{||\boldsymbol{f}||},$$
$$\boldsymbol{b}_{1,C} = \frac{\boldsymbol{b}_{2,des} \times \boldsymbol{b}_{3,C}}{||\boldsymbol{b}_{2,des} \times \boldsymbol{b}_{3,C}||}, \ \boldsymbol{b}_{2,C} = \boldsymbol{b}_{3,C} \times \boldsymbol{b}_{1,C},$$
$$[\boldsymbol{\omega}_C]_\times = R_C^\top \dot{R}_C,$$

where ψ_{des} is the desired yaw from the planner. Note that here we define $\boldsymbol{b}_{2,des}$ based on the yaw instead of $\boldsymbol{b}_{1,des}$ as done in [32] due to a different Euler angle convention (we use the ZYX convention instead of ZXY). The thrust and moments are then converted to motor speeds according to the characteristics of the vehicle.

VI. Experimental Results

The experimental platform shown in Fig. 1 is equipped with a Qualcomm® Snapdragon™ Flight™ board. This board is based on the Snapdragon™ 801 processor and features Wi-Fi connectivity, downward facing VGA camera with 160° field of view, a VGA stereo camera, and a 4K camera, all packed in a compact $58\,\text{mm} \times 40\,\text{mm}$ size.

A video of our experimental results can be found at `https://youtu.be/gU11R-CXwSM`.

A. Running time analysis

Because of the constraints on the CPU, there is a clear trade off between accuracy and computational speed. There are various parameters that can be tuned in our system in order to balance the trade-off for the best overall system performance:

1) **Frame rate**: There are three frame rate modes available on the cameras onboard the robot, namely 10 Hz, 15 Hz and 30 Hz. Higher frame rate makes tracking easier because changes are smaller between image frames so it is more likely that the tracked pixels fall into the gradient region of convergence, which helps direct methods. Higher frame rate also has a positive effect on the UKF estimation fusion process. Our small platform uses an inexpensive IMU that gives high frequency but noisy observations. In the prediction step the UKF keeps accumulating IMU measurements with high uncertainty and it drifts away from actual states quickly. If the VO measurement updates come in too slowly, there can be jumps in state estimate during the update step. Thus, shorter time between visual odometry measurements will give smoother state estimates. However setting frame rates to 30 Hz overloads the CPU and slows down the whole system causing the control to be unstable. We find that a frame rate of 15 Hz is a reasonable compromise.
2) **Image resolution**: Fig 4 shows that the runtime for keyframe generation makes it hard for full resolution (640×480) images to be used for visual odometry in real time (at least 10 Hz), therefore we resort to using the down-sampled resolution of 320×240 as the largest layer in the image pyramid. We also found that using two pyramid levels works best in terms of the compromise between accuracy and CPU usage for our system.
3) **Number of tracked pixels**: Tracking a larger number of pixels in direct method increases tracking accuracy [33], but it will also result in proportionally longer runtime as well as more CPU usage. We first obtain a gradient image from the input gray-scale image and apply an adaptive threshold to select an initial set of high gradient points. Then we divide image into grids and only select the highest gradient point in each grid cell. The grid is used so that we can use high resolution images while still keeping the number of tracked pixels relatively small. This reduces the total number of pixels in the optimization and at the same time spreads out the pixels evenly. Fig 4 shows that using grid of size 2×2 can reduce processing time so that the VO pipeline can be run at 15 Hz and experiments show that the system performs best under this configuration.

Based on the above considerations, we decided to fly with two image pyramid levels with the base layer set to the sub-sampled 320×240 resolution, a grid size of 2×2, and running the image stream and VO pipeline at 15 Hz. Table I shows detailed running time statistics for this configuration. We see that the average processing time for keyframes is about twice the time for non-keyframes. Unlike non-keyframes which only involves tracking, handling keyframes also involves image processing to generate disparity images and high gradient pixel masks.

In the table, for keyframes the running time for individual components do not add up to the total time, because these components are running in parallel in different threads, and

TABLE I

TIME STATISTICS FOR KEYFRAME AND NON-KEYFRAME PROCESSING WITH BASE LAYER RESOLUTION OF 320×240 AND GRID SIZE 2×2.

Time [ms]	Mean	Std Dev	Min	Max
Keyframe (total)	59.1	8.2	43.7	80.2
– extract features	45.5	6.8	35.1	66.4
– feature tracking	41.7	13.1	6.3	71.2
– disparity	18.7	4.9	14.7	35
Non Keyframe (only tracking)	32	11.3	5	57.9

we record the time for each individual thread to finish. Details of parallelization for keyframes are as follows: For image preprocessing, two threads are used for left and right images to do pyramid construction. The feature extraction and disparity map generation for each pyramid resolution are put into different threads. The tracking is done with respect to the last keyframe and can run in parallel with the feature extraction and disparity computation. This parallelization scheme greatly reduces the time for keyframe processing, and is one of the main factors allowing us to run the pipeline at 15 Hz.

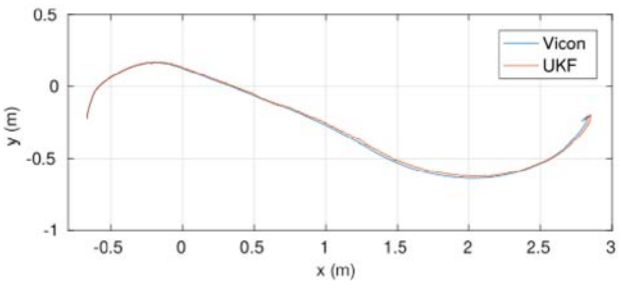

Fig. 5. The plot shows the comparison of position between UKF estimates and Vicon data as the robot follows a slalom trajectory.

B. Accuracy

Fig. 5 and 6 shows the comparison of visual inertial pose estimation to ground truth when our vehicle follows a slalom and a repeated elliptical trajectory, respectively. The UKF uses visual odometry as measurement update and its fused estimation result closely follows visual odometry, hence we are only showing the UKF estimates in the plots. The drift for the ellipse trajectory over a distance of 22.62 m is 0.26 m which is a drift of only 1.15% of the distance traveled.

C. Mapping

Fig. 8 shows the map created while the robot follows the slalom trajectory. The readers can get a sense of the environment we used when creating the map in Fig. 7. The moving axis represents the pose estimates from our visual odometry. The map clearly shows textured ground and a textured board towards the end of the trajectory, a barrel to the right and a traffic cone on the left.

VII. CONCLUSION

In this paper, we demonstrated for the first time that direct visual inertial odometry, closed-loop control and semi-dense mapping can be achieved concurrently on a smart phone grade processor, allowing autonomous flight of a small-sized quadrotor under SWAP constraints. We strongly believe that autonomy at this scale is an important enabler for search and rescue and first response missions. Future work will focus on further improving the quality of the map with outlier rejection based on the residuals during optimization and using it for autonomous navigation for SWAP constrained

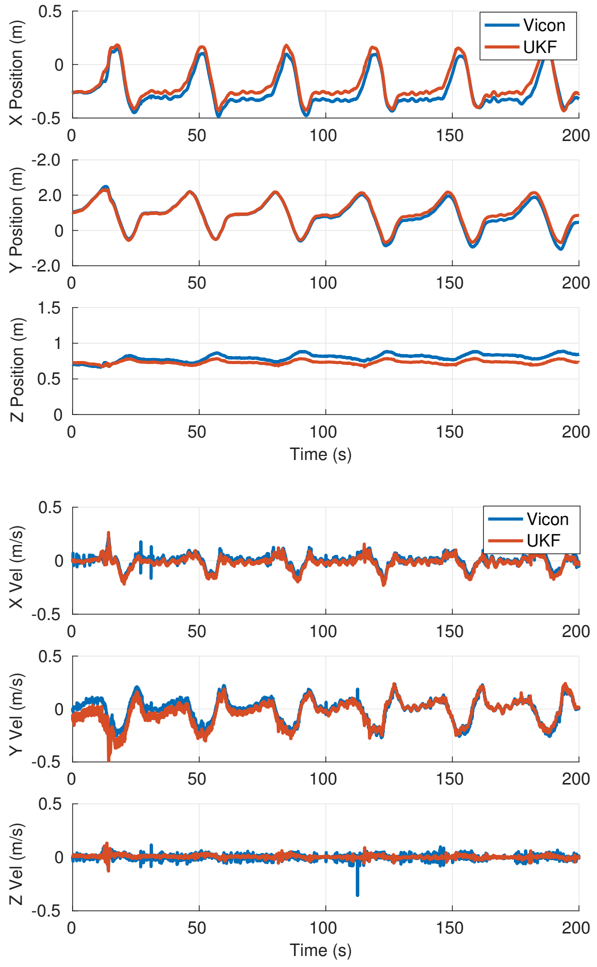

Fig. 6. Plots of the position and linear velocity from the UKF and Vicon for the ellipse trajectory.

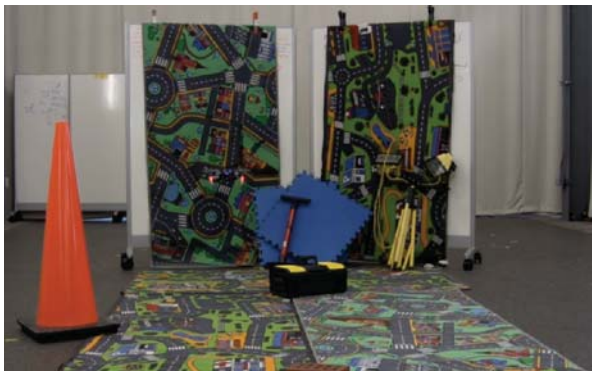

Fig. 7. A picture of the testing environment.

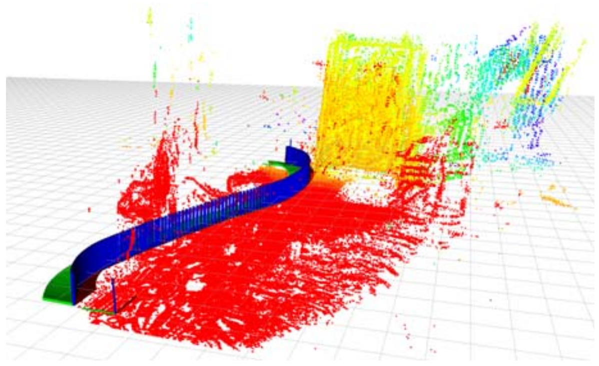

Fig. 8. The semi-dense map created by while following a slalom trajectory.

aerial platforms.

References

[1] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 46–56, Sept 2012.

[2] T. Ozaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, "Autonomous navigation and mapping for inspection of penstocks and tunnels with mavs," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1740–1747, July 2017.

[3] G. Loianno, J. Thomas, and V. Kumar, "Cooperative localization and mapping of mavs using rgb-d sensors," in *IEEE International Conference on Robotics and Automation*, May 2015, pp. 4021–4028.

[4] F. Forte, R. Naldi, and L. Marconi, "Impedance Control of an Aerial Manipulator," in *American Control Conference (ACC)*, Montreal, Canada, 2012, pp. 3839–3844.

[5] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual servoing of quadrotors for perching by hanging from cylindrical objects," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 57–64, Jan 2016.

[6] N. Michael, S. Shen, K. Mohta, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "Collaborative Mapping of an Earthquake-Damaged Building via Ground and Aerial Robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, 2012.

[7] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[8] S. Shen, N. Michael, and V. Kumar, "Tightly-Coupled Monocular Visual-Inertial Fusion for Autonomous Flight of Rotorcraft MAVs," in *IEEE International Conference on Robotics and Automation*, Seattle, USA, 2015.

[9] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.

[10] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor," in *Robotics: Science and Systems (RSS)*, Berlin, Germany, 2013.

[11] G. Klein and D. Murray, "Parallel Tracking and Mapping on a Camera Phone," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, Orlando, USA, 2009, pp. 83–86.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: A versatile and accurate monocular slam system." *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[13] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *eccv*, September 2014.

[14] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," in *arXiv:1607.02565*, July 2016.

[15] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time." in *ICCV*, D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, Eds. IEEE Computer Society, 2011, pp. 2320–2327.

[16] T. Schps, J. Engel, and D. Cremers, "Semi-dense visual odometry for ar on a smartphone," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Sept 2014, pp. 145–150.

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

[18] N. Krombach, D. Droeschel, and S. Behnke, "Combining feature-based and direct methods for semi-dense real-time stereo visual odometry." in *IAS*, ser. Advances in Intelligent Systems and Computing, W. Chen, K. Hosoda, E. Menegatti, M. Shimizu, and H. Wang, Eds., vol. 531, 2016, pp. 855–868.

[19] H. Hirschmüller, "Semi-global matching-motivation, developments and applications," *Photogrammetric Week 11*, pp. 173–184, 2011.

[20] S. Omari, M. Bloesch, P. Gohl, and R. Siegwart, "Dense visual-inertial navigation system for mobile robots," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2634–2640.

[21] Y. Ling, T. Liu, and S. Shen, "Aggressive quadrotor flight using dense visual-inertial fusion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1499–1506.

[22] Z. Yang, F. Gao, and S. Shen, "Real-time monocular dense mapping on aerial robots using visual-inertial fusion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4552–4559.

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

[24] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer Science & Business Media, 2012, vol. 26.

[25] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International journal of computer vision*, vol. 56, no. 3, pp. 221–255, 2004.

[26] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[27] R. Van Der Merwe, E. Wan, and S. Julier, "Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 5120.

[28] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The Grasp Multiple Micro–UAV Test Bed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[29] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[30] B. Hérissé, T. Hamel, R. Mahony, and F. X. Russotto, "Landing a VTOL Unmanned Aerial Vehicle on a moving platform using optical flow," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 77–89, 2012.

[31] T. Lee, M. Leok, and N. H. McClamroch, "Nonlinear Robust Tracking Control of a Quadrotor UAV on SE(3)," *Asian Journal of Control*, vol. 15, no. 2, pp. 391–408, 2013.

[32] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors," in *IEEE International Conference on Robotics and Automation*, Shangai, China, 2011, pp. 2520–2525.

[33] 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.