Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor Micro Aerial Vehicle

Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza*
Robotics and Perception Group, University of Zurich, 8050 Zurich, Switzerland

Received 14 June 2014; accepted 22 January 2015

The use of mobile robots in search-and-rescue and disaster-response missions has increased significantly in recent years. However, they are still remotely controlled by expert professionals on an actuator set-point level, and they would benefit, therefore, from any bit of autonomy added. This would allow them to execute high-level commands, such as "*execute this trajectory*" or "*map this area*." In this paper, we describe a vision-based quadrotor micro aerial vehicle that can autonomously execute a given trajectory and provide a live, dense three-dimensional (3D) map of an area. This map is presented to the operator while the quadrotor is mapping, so that there are no unnecessary delays in the mission. Our system does not rely on any external positioning system (e.g., GPS or motion capture systems) as sensing, computation, and control are performed fully onboard a smartphone processor. Since we use standard, off-the-shelf components from the hobbyist and smartphone markets, the total cost of our system is very low. Due to its low weight (below 450 g), it is also passively safe and can be deployed close to humans. We describe both the hardware and the software architecture of our system. We detail our visual odometry pipeline, the state estimation and control, and our live dense 3D mapping, with an overview of how all the modules work and how they have been integrated into the final system. We report the results of our experiments both indoors and outdoors. Our quadrotor was demonstrated over 100 times at multiple trade fairs, at public events, and to rescue professionals. We discuss the practical challenges and lessons learned. Code, datasets, and videos are publicly available to the robotics community.

SUPPLEMENTARY MATERIAL

This paper is accompanied by videos demonstrating the capabilities of our platform in outdoor and indoor scenarios:

- Indoor evaluation (disturbance and autonomous, vision-based live 3D mapping): http://youtu.be/sdu4w8r_fWc
- Outdoor autonomous, vision-based flight over disaster zone: http://youtu.be/3mNY9-DSUDk
- Outdoor autonomous, vision-based flight with live 3D mapping: http://youtu.be/JbACxNfBI30

More videos can be found on our Youtube channel: https://www.youtube.com/user/ailabRPG/videos

Our visual odometry code (called SVO) for vision-based navigation has been released open source and is freely available on the authors' homepage.

*The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—http://rpg.ifi.uzh.ch.
Direct correspondence to: Matthias Faessler, e-mail; faessler@ifi.uzh.ch

1. INTRODUCTION

1.1. Motivation

For search-and-rescue, disaster response, and surveillance missions, it is crucial for human rescuers to get an overview of the situation in order to take appropriate measures. In this paper, we present a vision-based quadrotor that can autonomously execute a trajectory, build a dense 3D map of an unknown area in real-time, and present it live to a user during the mission. Live feedback is, indeed, crucial to avoid any unnecessary delays during a rescue operation.

When robots are deployed in disaster situations, an average of three expert professionals are required for each robot to control them (Murphy, 2014). Additionally, they are teleoperated on an actuator set-point level, which makes the execution of tasks slow and cumbersome.

At the current state, all micro aerial vehicles (MAVs) used in search-and-rescue and remote-inspection scenarios are controlled under direct line of sight with the operator (Murphy, 2014). If wireless communication can be maintained, there is the possibility to teleoperate the MAV by transmitting video streams from the onboard cameras to the operator. However, teleoperation from video streams is extremely challenging in indoor environments.

Since such systems exhibit very limited or no autonomy at all, the stress level on the operator is very high,

View this article online at wileyonlinelibrary.com • DOI: 10.1002/rob.21581

which limits the mission time drastically. Operator errors could harm the robot or, even worse, cause further damage. For these reasons, there is a large need for flying robots that can navigate autonomously, without any user intervention, or execute high-level commands, such as "*execute this trajectory*" or "*map this area*." This would bring several advantages over today's disaster-response robots. First, the robot could easily be operated by a single person, who could focus on the actual mission. Secondly, a single person could operate multiple robots at the same time to speed up the mission. Finally, rescuers could operate such systems with very little training. These advantages, in combination with the low cost of the platform, will soon make MAVs become standard tools in disaster response operations.

1.2. System Overview

Our system consists of a quadrotor and a laptop base station with a graphical user interface for the operator (see Figure 1). The quadrotor is equipped with a single, down-looking camera, an inertial measurement unit (IMU), and a single-board computer. All required sensing, computation, and control is performed onboard the quadrotor. This design allows us to operate safely even when we temporarily lose wireless connection to the base station. It also allows us to operate the quadrotor beyond the range of the wireless link as, for instance, inside buildings. We do not require any external infrastructure such as GPS, which can be unreliable in urban areas or completely unavailable indoors.

We chose quadrotors because of their high maneuverability, their ability to hover on a spot, and their simple mechanical design. To make the system self-contained, we rely only on onboard sensors (i.e., a camera and an IMU).

For operating in areas close to humans, safety is a major concern. We aim at achieving this passively by making the quadrotor as lightweight as possible (below 450 g). Therefore, we chose passive sensors, which are typically lighter and consume less power. However, when using cameras, high computational power is required to process the huge amount of data. Due to the boost of computational power in mobile devices (e.g., smartphones, tablets), high-performance processors that are optimized for power consumption, size, and cost are available today. An additional factor for real-world applications is the cost of the overall system. The simple mechanical design and the use of sensors and processors produced millionfold for smartphones makes the overall platform low cost (1,000 USD).

1.3. Contributions and Differences with Other Systems

The main contribution of this paper is a self-contained, robust, low-cost, power-on-and-go quadrotor MAV that can autonomously execute a given trajectory and provide a live dense 3D mapping without any user intervention and using only a single camera and an IMU as the main sensory modality.

The most similar to our system is the one which resulted from the European project SFLY Weiss et al. (2013); Scaramuzza et al. (2014). However, in SFLY, dense 3D maps were computed offline and were available only after several minutes after landing.

Another difference with the SFLY project lies in the vision-based motion-estimation pipeline. Most monocular, visual-odometry algorithms used for MAV navigation (see Section 3.2) rely on PTAM Klein & Murray (2007), which is a feature-based visual SLAM algorithm, running at 30 Hz, designed for augmented reality applications in small desktop scenes. In contrast, our quadrotor relies on a novel visual odometry algorithm [called SVO, which we proposed in Forster, Pizzoli, & Scaramuzza (2014b)] designed specifically for MAV applications. SVO eliminates the need of costly feature extraction and matching as it operates directly on pixel intensities. This results in high precision, robustness, and higher frame-rates (at least twice that of PTAM) than current state-of-the-art methods. Additionally, it uses a probabilistic mapping method to model outliers and feature-depth uncertainties, which provide robustness in scenes with repetitive, and high-frequency textures.

In SFLY, a commercially available platform was used, with limited access to the low-level controller. Conversely, we have full access to all the control loops, which allows us to tune the controllers down to the lowest level. Furthermore, we propose a one-time-per-mission estimation of sensor biases, which allows us to reduce the number of states in the state estimator. In addition, the estimated biases are also considered in the low-level controller (body rate controller), while in SFLY, they were only used in the high-level controller (position controller). Furthermore, we propose a calibration of the actually-produced thrust, which ensures the same control performance regardless of environmental conditions such as temperature and air pressure.

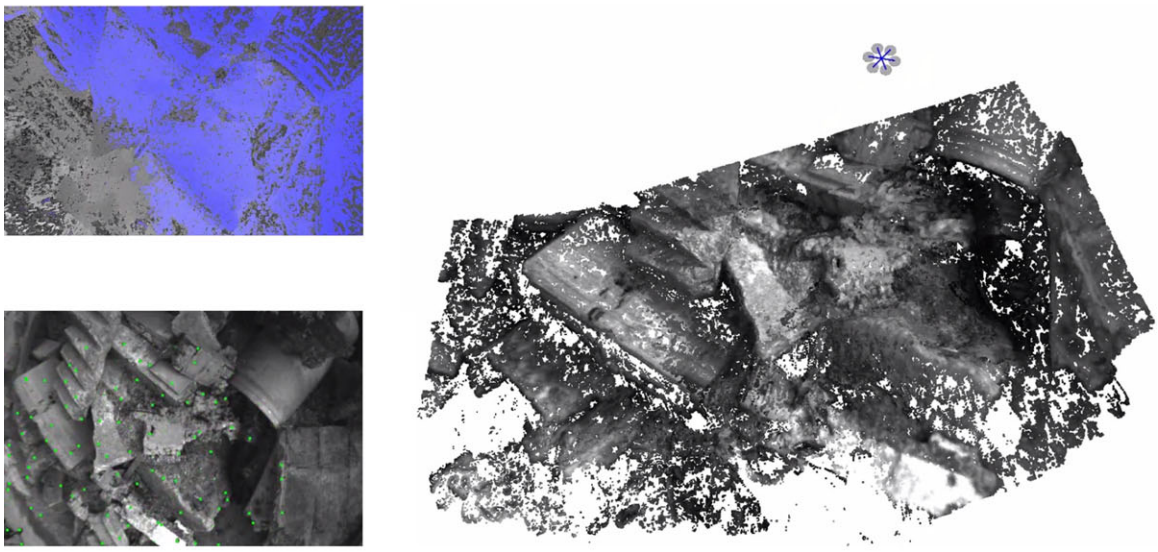
1.4. Outline

The paper is organized as follows. Section 3 reviews the related work on autonomous MAV navigation and real-time 3D dense reconstruction. Section 4 presents the hardware and software architecture of our platform. Section 5 describes our visual odometry pipeline, while Section 6 details the state estimation and control of the quadrotor. Section 7 describes our live, dense 3D reconstruction pipeline. Finally, Section 8 presents and discusses the experimental results, and Section 9 comments on the lessons learned.

2. RELATED WORK

To date, most autonomous MAVs rely on GPS to navigate outdoors. However, GPS is not reliable in urban settings and

(a)



(b)

Figure 1. Our system (a) can autonomously build a dense 3D map of an unknown area. The map is presented to the operator through a graphical user interface on the base station laptop while the quadrotor is mapping (b, right inset) together with the onboard image (b, bottom left inset) and a confidence map (b, top left inset).

is completely unavailable indoors. Because of this, most works on autonomous indoor navigation of flying robots have used external motion-capture systems (e.g., Vicon or OptiTrack). These systems are very appropriate for testing and evaluation purposes (Lupashin et al., 2014; Michael, Mellinger, Lindsey, & Kumar, 2010), such as prototyping control strategies or executing fast maneuvers. However, they need pre-installation of the cameras, and thus they cannot be used in unknown yet unexplored environments. Therefore, for truly autonomous navigation in indoor environments, the only viable solution is to use onboard sensors. The literature on autonomous navigation of MAVs using onboard sensors includes range (e.g., laser rangefinders or RGB-D sensors) and vision sensors.

2.1. Navigation Based on Range Sensors

Laser rangefinders have been largely explored for simultaneous localization and mapping (SLAM) with ground mobile robots (Thrun et al., 2007). Because of the heavy weight of 3D scanners [see, for instance, the Velodyne sensor (more than 1 kg)], laser rangefinders currently used on MAVs are only 2D. Since 2D scanners can only detect objects that intersect their sensing plane, they have been used for MAVs in environments characterized by vertical structures (Achtelik, Bachrach, He, Prentice, & Roy, 2009; Bachrach, He, & Roy, 2009; Shen, Michael, & Kumar, 2011, 2012b) and less in more complex scenes.

RGB-D sensors are based upon structured-light techniques, and thus they share many properties with stereo cameras. However, the primary differences lie in the range and spatial density of depth data. Since RGB-D sensors illuminate a scene with a structured-light pattern, contrary to stereo cameras, they can estimate depth in areas with poor visual texture but are range-limited by their projectors. RGB-D sensors for state estimation and mapping with MAVs have been used in Bachrach et al. (2012) as well as in Shen, Michael, & Kumar (2012a, 2012b), where a multi-floor autonomous exploration and mapping strategy was presented.

2.2. Navigation Based on Vision Sensors

Although laser rangefinders and RGB-D sensors are very accurate and robust to illumination changes, they are too heavy and consume too much power for lightweight MAVs. In this case, the alternative solution is to use vision sensors. Early works on vision-based navigation of MAVs focused on biologically inspired algorithms (such as optical flow) to perform basic maneuvers, such as takeoff and landing, reactive obstacle avoidance, and corridor and terrain following (Hrabar, Sukhatme, Corke, Usher, & Roberts, 2005; Lippiello, Loianno, & Siciliano, 2011; Ruffier & Franceschini, 2004; Zingg, Scaramuzza, Weiss, & Siegwart, 2010; Zufferey & Floreano, 2006). Since optical flow can only measure the relative velocity of image features, the position estimate of the MAV will inevitably drift over time. This can be avoided using visual odometry or visual simultaneous localization and mapping (SLAM) methods, in monocular (Forster et al., 2014b; Scaramuzza et al., 2014; Weiss, Scaramuzza, & Siegwart, 2011; Weiss et al., 2013) or stereo configurations (Achtelik et al., 2009; Schmid, Lutz, Tomic, Mair, & Hirschmuller, 2014; Shen, Mulgaonkar, Michael, & Kumar, 2013). Preliminary experiments for MAV localization using a visual extended Kalman filter (EKF) -based SLAM technique were described in Ahrens, Levine, Andrews, & How (2009). However, the first use of visual SLAM to enable autonomous basic maneuvers was done within the framework of the SFLY European project, where a single camera and an IMU were used for state estimation and point-to-point navigation over several hundred meters in an outdoor, GPS-denied environment (Scaramuzza et al., 2014; Weiss et al., 2013).

Most monocular visual odometry (VO) algorithms for MAVs (Bloesch, Weiss, Scaramuzza, & Siegwart, 2010; Engel, Sturm, & Cremers, 2012; Scaramuzza et al., 2014; Weiss et al., 2013) rely on PTAM (Klein & Murray, 2007). PTAM is a feature-based SLAM algorithm that achieves robustness through tracking and mapping several hundreds of features. Additionally, it runs in real time (at around 30 Hz) by parallelizing the motion estimation and mapping tasks and by relying on efficient keyframe-based bundle adjustment (BA) (Strasdat, Montiel, & Davison, 2010). However, PTAM was designed for augmented reality applications in small desktop scenes, and multiple modifications (e.g., limiting the number of keyframes) were necessary to allow operation in large-scale outdoor environments (Weiss et al., 2013).

2.3. Real-time Monocular Dense 3D Mapping

In robotics, a dense reconstruction is needed to interact with the environment—as in obstacle avoidance, path planning, and manipulation. Moreover, the robot must be aware of the uncertainty affecting the measurements in order to intervene by changing the vantage point or deploying different sensing modalities (Forster, Pizzoli, & Scaramuzza, 2014a).

A single moving camera represents the most general setting for stereo vision. Indeed, in stereo settings a fixed baseline constrains the operating range of the system, while a single moving camera can be seen as a stereo camera with an adjustable baseline that can be dynamically reconfigured according to the requirements of the task.

Few relevant works have addressed real-time, dense reconstruction from a single moving camera, and they shed light on some important aspects. If, on the one hand, estimating the depth independently for every pixel leads to efficient, parallel implementations, on the other hand Gallup, Frahm, Mordohai, Yang, & Pollefeys (2007), Stühmer et al.

(2010), Newcombe et al. (2011), and Wendel, Maurer, Graber, Pock, & Bischof (2012) argued that, similar to other computer vision problems, such as image denoising (Rudin, Osher, & Fatemi, 1992) and optical flow estimation (Werlberger, Pock, & Bischof, 2010), a smoothing step is required in order to deal with noise and spurious measurements. In Stühmer et al. (2010), smoothness priors were enforced over the reconstructed scene by minimizing a regularized energy functional based on aggregating a photometric cost over different depth hypothesis and penalizing nonsmooth surfaces. The authors showed that the integration of multiple images leads to significantly higher robustness to noise. A similar argument is put forth in Newcombe et al. (2011), where the advantage of photometric cost aggregation (Szeliski & Scharstein, 2004) over a large number of images taken from nearby viewpoints is demonstrated.

However, despite the groundbreaking results, these approaches present some limitations when addressing tasks in robot perception. Equally weighting measurements from small and large baselines, in close and far scenes, causes the aggregated cost to frequently present multiple or no minima. Depending on the depth range and sampling, these failures are not always recoverable by the subsequent optimization step. Furthermore, an inadequate number of images can lead to a poorly constrained initialization for the optimization and erroneous measurements that are hard to detect. It is not clear how many images should be collected, depending on the motion of the camera and the scene structure. Finally, the number of depth hypotheses controls the computational complexity, and the applicability is thus limited to scenes bounded in depth.

Therefore, in Pizzoli, Forster, & Scaramuzza (2014) we presented the REMODE framework that overcomes these limitations by using a probabilistic approach handling measurement uncertainty. We build on the Bayesian depth estimation proposed in Vogiatzis & Hernández (2011) for per-pixel depth estimation and introduce an optimization step to enforce spatial regularity over the recovered depth map. We propose a regularization term based on the weighted Huber norm, but, differently from Newcombe et al. (2011), we use the depth uncertainty to drive the smoothing and exploit a convex formulation for which a highly parallelizable solution scheme has been recently introduced (Chambolle & Pock, 2011).

3. SYSTEM OVERVIEW

We propose a system consisting of a quadrotor equipped with a monocular camera and a laptop serving as the ground station. The quadrotor is able to navigate fully autonomously without requiring any communication with the ground station. On the ground station, we can compute a dense 3D reconstruction from the images taken by the quadrotor in real time. In the following, we describe the aerial platform and the software modules.

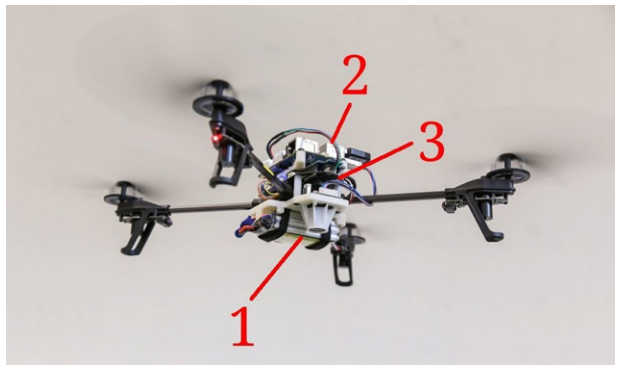


Figure 2. A closeup of our quadrotor: 1) down-looking camera, 2) Odroid U3 quad-core computer, 3) PIXHAWK autopilot.

3.1. Aerial Platform

We built our quadrotor from selected off-the-shelf components and custom 3D printed parts (see Figure 2). The components were chosen according to their performance and their ability to be easily customized.

Our quadrotor relies on the frame of the Parrot AR.Drone 2.0[1] including their motors, motor controllers, gears, and propellers. To reduce play and vibrations on the platform, we replaced the bushings of the propeller axes by ball bearings. The platform is powered by one 1,350 mA h LiPo battery, which allows a flight time of 10 min.

We completely replaced the electronic parts of the AR.Drone by a PX4FMU autopilot and a PX4IOAR adapter board developed in the PIXHAWK Project (Meier et al., 2012). The PX4FMU consists, among other things, of an IMU and a micro controller to read the sensors, run a body rate controller, and command the motors. In addition to the PX4 autopilot, our quadrotors are equipped with an Odroid-U3 single-board computer.[2] It contains a 1.7 GHz quad-core processor running XUbuntu 13.10[3] and ROS.[4] The PX4 micro controller communicates with the Odroid board over UART, whereas the Odroid board communicates with the ground station over 5 GHz WiFi.

To stabilize the quadrotor, we make use of the gyros and accelerometers of the IMU on the PX4FMU as well as a downward-looking MatrixVision mvBlueFOX-MLC200w (752 × 480)-pixel monochrome camera with a 130-degree field-of-view lens.[5]

Our platform is easily reparable due to off-the-shelf components, inexpensive (1,000 USD; cf. Table I),

[1]http://ardrone2.parrot.com/
[2]http://www.hardkernel.com/main/products/prdt_info.php?g_code=G138745696275
[3]http://www.xubuntu.org/
[4]http://www.ros.org/
[5]http://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html

Table I. Weight and price of the individual components of our quadrotors.

Component	Weight (g)	Price (USD)
Frame, Gears, Propellers	119	63
Motors, Motor Controllers	70	214
PX4FMU, PX4IOAR	35	181
Hardkernel Odroid-U3	49	65
Camera, Lens	16	326
1350mA h Battery	99	44
Other Parts	54	110
Total	**442**	**1,003**

lightweight (below 450 g), and, due to its flexible propellers, safe to use.

3.2. Software Modules

The software used in our system runs on three different processing units (see Figure 3), namely the PX4 micro controller, the Odroid computer, and a laptop, which serves as the ground station. All the computations required to stabilize the quadrotor are performed onboard. On the ground station (a W530 Lenovo laptop), only the dense 3D reconstruction is computed using its graphics processing unit (GPU).

The PX4 micro controller reads the IMU and controls the desired body rates and collective thrust that it receives from the high-level controller running on the Odroid.

The Odroid processes the camera images by means of our semidirect visual odometry [SVO (Forster et al., 2014b)] pipeline (see Section 5). The visual odometry pipeline outputs an unscaled pose, which is then fused with the IMU readings in an extended Kalman filter framework [multisensor fusion (MSF) (Lynen, Achtelik, Weiss, Chli, & Siegwart, 2013)] to compute a metric state estimate. From this state estimate and a reference trajectory, we compute the desired body rates and collective thrust, which are then sent to the PX4. Alongside this pipeline, we send the images from the camera together with the corresponding pose estimate to the ground station.

On the ground station, we run a dense 3D reconstruction (Pizzoli et al., 2014) in real time using the camera images with their corresponding pose estimate (see Section 7).

4. SEMIDIRECT VISUAL ODOMETRY (SVO)

Using visual odometry with a single downward-looking camera, we can simultaneously estimate the motion of the vehicle (up to an unknown scale and rigid-body transformation) and the local structure of the scene in the form of a sparse point-cloud. In Forster et al. (2014b), we proposed a novel VO algorithm called SVO that is two times faster in terms of processing time compared to previous methods. The motivation to increase the frame-rate is twofold: first, it allows the MAV to fly faster and more agilely; second, as we will show in the experimental results, it allows the MAV to localize in environments of highly repetitive and high-frequency texture (see Figure 11).

Figure 4 provides an overview of SVO. The algorithm uses two parallel threads, one for estimating the camera motion with respect to the local map, and a second one for extending the map as the environment is being explored. This separation allows fast and constant-time tracking in

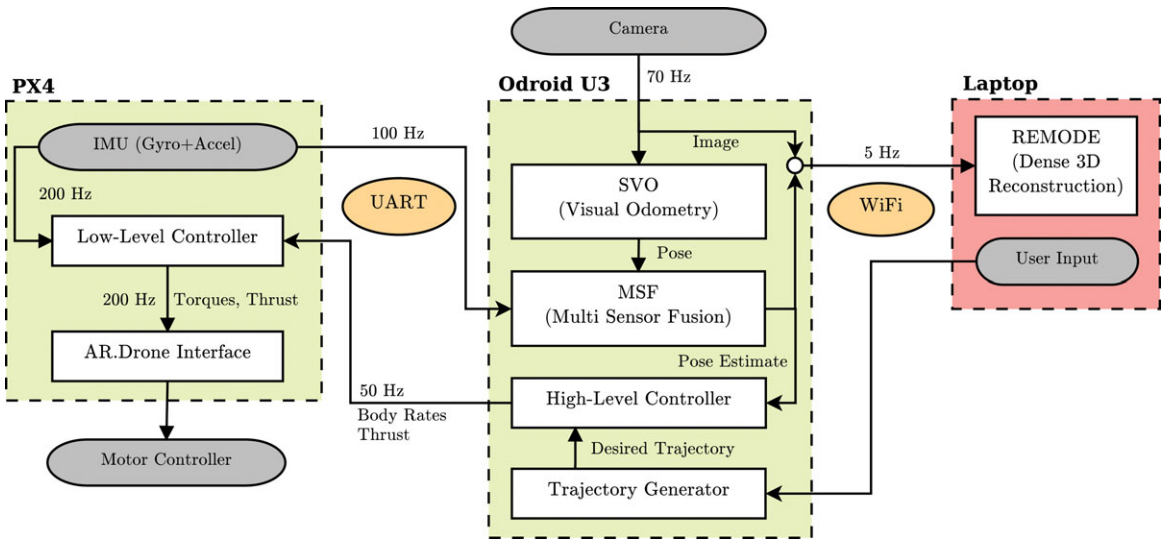


Figure 3. System overview: the PX4 and Odroid U3 communicate with each other over a UART interface. Communication to the Laptop is over WiFi. Gray boxes are sensors and actuators; software modules are depicted as white boxes.

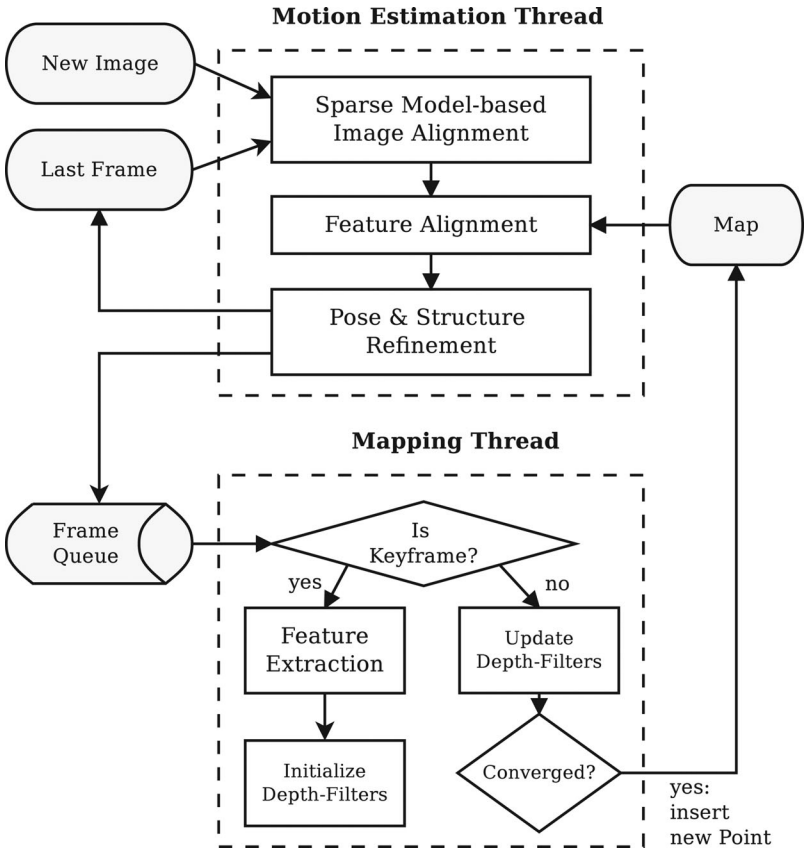


Figure 4. SVO system overview. Two concurrent threads are used, one for estimating the motion of the camera with respect to the map and the second one for extending the map.

one thread, while the second thread extends the map, decoupled from hard real-time constraints. In the following, we provide an overview of both motion estimation and mapping. We refer the reader to Forster et al. (2014b) for more details.

4.1. Motion Estimation

Methods that estimate the camera pose with respect to a map (i.e., a set of key-frames and 3D points) can be divided into two classes:

(A) Feature-based methods extract a sparse set of salient image features in every image, match them in successive frames using invariant feature descriptors, and finally recover both camera motion and structure using epipolar geometry (Scaramuzza & Fraundorfer, 2011). The bottleneck of this approach is that approximately 50% of the processing time is spent on feature extraction and matching, which is the reason why most of the VO algorithms still run at 20–30 fps despite the availability and advantages of high-frame-rate cameras.

(B) Direct methods (Irani & Anandan, 1999), on the other hand, estimate the motion directly from intensity values in the image. The rigid-body transformation is found through minimizing the photometric difference between corresponding pixels, where the local intensity gradient magnitude and direction are used in the optimization. Since this approach starts directly with an optimization, increasing the frame-rate means that the optimization is initialized closer to the optimum, and thus it converges much faster. Hence, direct methods in general benefit from increasing the frame-rate (Handa, Newcombe, Angeli, & Davison, 2012) as the processing time per frame decreases.

In Forster et al. (2014b), we proposed a semidirect visual odometry (SVO) that combines the benefits of feature-based and direct methods. SVO establishes feature correspondence by means of direct methods rather than feature extraction and matching. The advantage is increased speed (up to 70 fps onboard the MAV and 400 Hz on a consumer laptop) due to the lack of feature extraction at every frame and increased accuracy through subpixel feature correspondence. Once feature correspondences are established, the algorithm continues using only point-features,

hence the term "*semidirect*." This switch allows us to rely on fast and established frameworks for bundle adjustment, i.e., joint optimization of both 3D points and camera poses (Triggs, McLauchlan, Hartley, & Fitzgibbon, 2000).

4.2. Mapping

The mapping thread estimates the depth at new 2D feature positions [we used FAST corners (Rosten, Porter, & Drummond, 2010)] by means of a *depth filter*. Once the depth filter has converged, a new 3D point is inserted in the map at the found depth and immediately used for motion estimation. The same depth filter formulation is used for dense reconstruction, and its formulation is explained in more detail in Section 7.1.

New depth-filters are initialized whenever a new keyframe is selected in regions of the image where few 3D-to-2D correspondences are found. A keyframe is selected when the distance between the current frame and the previous keyframe exceeds 12% of the average scene depth. The filters are initialized with a large uncertainty in depth and with a mean at the current average scene depth.

At every subsequent frame, the epipolar line segment in the new image corresponding to the current depth confidence interval is searched for an image patch that has the highest correlation with the reference feature patch (see Figure 6). If a good match is found, the depth filter is updated in a recursive Bayesian fashion [see Eq. (31)]. Hence, we use many measurements to verify the position of a 3D point, which results in considerably fewer outliers compared to triangulation from two views.

4.3. Implementation Details

The source-code of SVO is available open-source at https://github.com/uzh-rpg/rpg_svogithub.com/uzh-rpg/rpg_svo. No special modifications are required to run SVO onboard the MAV. For all experiments, we use the *fast* parameter setting that is available online. The *fast* setting limits the number of features per frame to 120, maintains always at a maximum 10 keyframes in the map (i.e., older keyframes are removed from the map), and for processing-time reasons no bundle adjustment is performed.

5. STATE ESTIMATION AND CONTROL

In this section, we describe the state estimation and control used to stabilize our quadrotor. Furthermore, we explain how we estimate sensor biases and the actually produced thrust. The control and calibration sections are inspired by Lupashin et al. (2014).

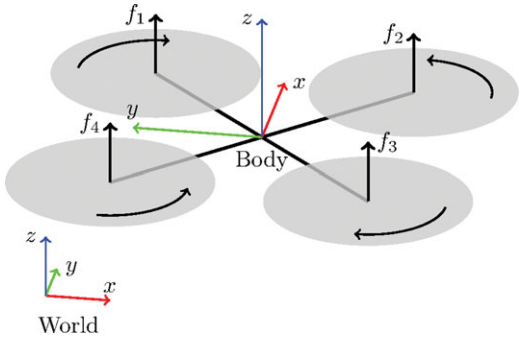


Figure 5. Quadrotor with coordinate system and rotor forces.

5.1. Dynamical Model

For state estimation and control, we make use of the following dynamical model of our quadrotor:

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{1}$$

$$\dot{\mathbf{v}} = \mathbf{g} + \mathbf{R} \cdot \mathbf{c}, \tag{2}$$

$$\dot{\mathbf{R}} = \mathbf{R} \cdot \hat{\boldsymbol{\omega}}, \tag{3}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1} \cdot (\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}), \tag{4}$$

where $\mathbf{r} = [x\ y\ z]^T$ and $\mathbf{v} = [v_x\ v_y\ v_z]^T$ are the position and velocity in world coordinates, $\mathbf{R}$ is the orientation of the quadrotor's body coordinates with respect to the world coordinates, and $\boldsymbol{\omega} = [p\ q\ r]^T$ denotes the body rates expressed in body coordinates. The skew symmetric matrix $\hat{\boldsymbol{\omega}}$ is defined as

$$\hat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}. \tag{5}$$

We define the gravity vector as $\mathbf{g} = [0\ 0\ -g]^T$, and $\mathbf{J} = \text{diag}\left(J_{xx}, J_{yy}, J_{zz}\right)$ is the second-order moment-of-inertia matrix of the quadrotor. The mass-normalized thrust vector is $\mathbf{c} = [0\ 0\ c]^T$, with

$$mc = f_1 + f_2 + f_3 + f_4, \tag{6}$$

where f_i are the four motor thrusts as illustrated in Figure 5. The torque inputs $\boldsymbol{\tau}$ are composed of the single-rotor thrusts as

$$\boldsymbol{\tau} = \begin{bmatrix} \frac{\sqrt{2}}{2} l(f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2} l(-f_1 - f_2 + f_3 + f_4) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \tag{7}$$

where l is the quadrotor arm length and κ is the rotor-torque coefficient.

Table II. Parameters and control gains used for the experiments.

Parameter	Description	Value	Unit
J_{xx}	x-axis Moment of Inertia	0.001	Kg m^2
J_{yy}	y-axis Moment of Inertia	0.001	Kg m^2
J_{zz}	z-axis Moment of Inertia	0.002	Kg m^2
m	Quadrotor Mass	0.45	kg
l	Quadrotor Arm Length	0.178	m
κ	Rotor Torque Coefficient	0.0195	m
p_{xy}	Horizontal Position Control Gain	5.0	s^{-2}
p_z	Vertical Position Control Gain	15.0	s^{-2}
d_{xy}	Horizontal Velocity Control Gain	4.0	s^{-1}
d_z	Vertical Velocity Control Gain	6.0	s^{-1}
p_{rp}	Roll/Pitch Attitude Control Gain	16.6	s^{-1}
p_{yaw}	Yaw Attitude Control Gain	5	s^{-1}
p_{pq}	Roll/Pitch Rate Control Gain	33.3	s^{-1}
p_r	Yaw Rate Control Gain	6.7	s^{-1}

The used coordinate systems and rotor numbering are illustrated in Figure 5, and the used parameter values are listed in Table II.

5.2. State Estimation

To stabilize our quadrotor, we need an estimate of the metric pose as well as the linear and angular velocities. We compute this state estimate by fusing the sensor data from the IMU and the output of the visual odometry in an extended Kalman filter. To do so, we make use of an open-source multisensor fusion package (Lynen et al., 2013). Since we did not modify this package, we are not describing the sensor fusion in more detail here.

5.3. Controller

To follow reference trajectories and stabilize the quadrotor, we use cascaded controllers. The high-level controller running on the Odroid includes a position controller and an attitude controller. The low-level controller on the PX4 contains a body rate controller. The used control gains are listed in Table II.

5.3.1. High-level Control

The high-level controller takes a reference trajectory as input and computes desired body rates that are sent to the low-level controller. A reference trajectory consists of a reference position $\mathbf{r}_{\text{ref}}$, a reference velocity $\mathbf{v}_{\text{ref}}$, a reference acceleration $\mathbf{a}_{\text{ref}}$, and a reference yaw angle ψ_{ref}. First, the position controller is described followed by the attitude controller. The two high-level control loops are synchronized and run at 50 Hz.

Position Controller. To track a reference trajectory, we implemented a PD controller with feedforward terms on velocity and acceleration:

$$\mathbf{a}_{\text{des}} = \mathbf{P}_{\text{pos}} \cdot (\mathbf{r}_{\text{ref}} - \hat{\mathbf{r}}) + \mathbf{D}_{\text{pos}} \cdot (\mathbf{v}_{\text{ref}} - \hat{\mathbf{v}}) + \mathbf{a}_{\text{ref}}, \tag{8}$$

with gain matrices $\mathbf{P}_{\text{pos}} = \text{diag}\left(p_{xy}, p_{xy}, p_z\right)$ and $\mathbf{D}_{\text{pos}} = \text{diag}\left(d_{xy}, d_{xy}, d_z\right)$. Since a quadrotor can only accelerate in its body z direction, $\mathbf{a}_{\text{des}}$ enforces two degrees of the desired attitude. Now we want to compute the desired normalized thrust such that the z component of $\mathbf{a}_{\text{des}}$ is reached with the current orientation. To do so, we make use of the last row of Eq. (2) to compute the required normalized thrust c_{des} as

$$c_{\text{des}} = \frac{\mathbf{a}_{\text{des},z} + g}{\mathbf{R}_{3,3}}. \tag{9}$$

The output of the position controller is composed of the desired accelerations $\mathbf{a}_{\text{des}}$, which, together with the reference yaw angle ψ_{ref}, encodes the desired orientation as well as a mass normalized thrust c_{des}.

Attitude Controller. In the previous paragraph, we computed the desired thrust such that the desired acceleration in the vertical world direction is met. Since a quadrotor can only produce thrust in the body z direction, the attitude controller has to rotate the quadrotor in order to achieve the desired accelerations in the world x-y plane with the given thrust. The translational behavior of the quadrotor is independent of a rotation around the body z axis. Therefore, we first discuss the attitude controller for roll and pitch and, second we present the yaw controller.

For the x-y plane movement, we restate the first two rows of (2) and insert the normalized thrust from Eq. (9),

$$\begin{bmatrix} \mathbf{a}_{\text{des},x} \\ \mathbf{a}_{\text{des},y} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{1,3} \\ \mathbf{R}_{2,3} \end{bmatrix} c_{\text{des}}, \tag{10}$$

where $\mathbf{R}_{i,j}$ denotes the (i, j) element of the orientation matrix $\mathbf{R}$. Solving for the two entries of $\mathbf{R}$ which define the x-y plane movement, we find

$$\begin{bmatrix} \mathbf{R}_{\text{des},1,3} \\ \mathbf{R}_{\text{des},2,3} \end{bmatrix} = \frac{1}{c_{\text{des}}} \begin{bmatrix} \mathbf{a}_{\text{des},x} \\ \mathbf{a}_{\text{des},y} \end{bmatrix}. \tag{11}$$

To track these desired components of $\mathbf{R}$, we use a proportional controller on the attitude error,

$$\begin{bmatrix} \dot{\mathbf{R}}_{\text{des},1,3} \\ \dot{\mathbf{R}}_{\text{des},2,3} \end{bmatrix} = p_{rp} \begin{bmatrix} \mathbf{R}_{1,3} - \mathbf{R}_{\text{des},1,3} \\ \mathbf{R}_{2,3} - \mathbf{R}_{\text{des},2,3} \end{bmatrix}, \tag{12}$$

where p_{rp} is the controller gain and $\dot{\mathbf{R}}$ is the change of the orientation matrix per time step.

We can write the first two rows of Eq. (3) as

$$\begin{bmatrix} \dot{\mathbf{R}}_{\text{des},1,3} \\ \dot{\mathbf{R}}_{\text{des},2,3} \end{bmatrix} = \begin{bmatrix} -\mathbf{R}_{1,2} & \mathbf{R}_{1,1} \\ -\mathbf{R}_{2,2} & \mathbf{R}_{2,1} \end{bmatrix} \cdot \begin{bmatrix} p_{\text{des}} \\ q_{\text{des}} \end{bmatrix}. \tag{13}$$

Finally, the desired roll and pitch rate can be computed by plugging Eq. (12) into Eq. (13) and solving for p_{des} and q_{des}.

The yaw-angle control does not influence the translational dynamics of the quadrotor and thus it can be controlled independently. First, we compute the current yaw angle ψ from the quadrotor orientation $\mathbf{R}$. Second, we compute the desired angular rate in the world z direction with a proportional controller on the yaw angle error,

$$r_{\text{world}} = p_{\text{yaw}} (\psi_{\text{ref}} - \psi). \tag{14}$$

The resulting desired rate can then be converted into the quadrotor body frame using its current orientation

$$r_{\text{des}} = \mathbf{R}_{3,3} \cdot r_{\text{world}}. \tag{15}$$

5.3.2. Low-level Control

The commands sent to the low-level control on the PX4 are the desired body rates $\boldsymbol{\omega}_{\text{des}}$ and the desired mass-normalized thrust c_{des}. From these, the desired rotor thrusts are then computed using a feedback linearizing control scheme with the closed-loop dynamics of a first-order system. First, the desired torques $\boldsymbol{\tau}_{\text{des}}$ are computed as

$$\boldsymbol{\tau}_{\text{des}} = \mathbf{J} \begin{bmatrix} p_{pq}(p_{\text{des}} - p) \\ p_{pq}(q_{\text{des}} - q) \\ p_r(r_{\text{des}} - r) \end{bmatrix} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}. \tag{16}$$

Then, we can plug $\boldsymbol{\tau}_{\text{des}}$ and c_{des} into Eqs. (7) and (6) and solve them for the desired rotor thrusts which has to be applied.

5.4. Calibration

5.4.1. Sensor Bias Calibration

For the state estimation and the control of our quadrotors, we make use of their gyros and accelerometers. Both of these sensor units are prone to having an output bias that varies over time and that we, therefore, have to estimate. We noticed that the changes of these biases during one flight are negligible. This allows us to estimate them once at the beginning of a mission and then keep them constant. Thus, we do not have to estimate them online and can therefore reduce the size of the state in the state estimation. In the following, we will present a procedure that allows us to estimate the biases during autonomous hover. Note that the quadrotor can hover autonomously even with sensor biases. However, removing the biases increases the state estimation and tracking performance. For the sensor bias calibration, we look at the gyros and the accelerometers separately.

The gyro measurement equation reads

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{n}_\omega, \tag{17}$$

where $\tilde{\boldsymbol{\omega}}$ denotes the measured angular velocities, $\boldsymbol{\omega}$ is the real angular velocities, $\mathbf{b}_\omega$ is the bias, and $\mathbf{n}_\omega$ is the noise of the gyros. This, in hover conditions, becomes

$$\tilde{\boldsymbol{\omega}} = \mathbf{b}_\omega + \mathbf{n}_\omega. \tag{18}$$

We assume the noise $\mathbf{n}_\omega$ to have zero mean and can therefore average the gyro measurements over N samples to estimate the gyro bias $\mathbf{b}_\omega$ as

$$\hat{\mathbf{b}}_\omega = \frac{1}{N} \sum_{k=1}^{N} \tilde{\boldsymbol{\omega}}_k. \tag{19}$$

The accelerometer measurement equation reads

$$\tilde{\mathbf{a}} = \mathbf{c} + \mathbf{a}_{\text{dist}} + \mathbf{b}_{\text{acc}} + \mathbf{n}_{\text{acc}}, \tag{20}$$

where $\tilde{\mathbf{a}}$ denotes the measured accelerations, $\mathbf{c}$ is the mass normalized thrust, $\mathbf{a}_{\text{dist}}$ are the accelerations due to external disturbances, $\mathbf{b}_{\text{acc}}$ is the bias, and $\mathbf{n}_{\text{acc}}$ is the noise of the accelerometer. In hover conditions, $\mathbf{c} = -\mathbf{g}$ and we assume to have only small and zero mean disturbances, so the equation simplifies to

$$\tilde{\mathbf{a}} = -\mathbf{g} + \mathbf{b}_{\text{acc}} + \mathbf{n}_{\text{acc}}. \tag{21}$$

As for the gyro bias, we assume the noise $\mathbf{n}_{\text{acc}}$ to have zero mean and can therefore average the accelerometer measurements over N samples to estimate the accelerometer bias $\mathbf{b}_{\text{acc}}$ as

$$\hat{\mathbf{b}}_{\text{acc}} = \frac{1}{N} \sum_{k=1}^{N} \tilde{\mathbf{a}}_k + \mathbf{g}. \tag{22}$$

When performing the sensor bias calibration, we sample the IMU readings over a 5 s period, which is enough to provide an accurate estimate of their biases.

5.4.2. Thrust Calibration

When flying our quadrotors under very different conditions indoors and outdoors, we noticed that the produced rotor thrust can vary substantially. This can significantly reduce the control authority and hence the flight performance in situations in which the produced thrust is low. To overcome this limitation, we estimate the actually produced thrust in flight.

The thrust f of a single rotor can be computed as

$$f = \frac{1}{2} \cdot \rho \cdot \Omega^2 \cdot C \cdot A, \tag{23}$$

where ρ is the air density, Ω is the rotor speed, C is the lift coefficient, and A is the rotor area. The rotor speed Ω is the input parameter, through which we control the thrust. We assume that the rotor speed is controlled by the motor controllers such that we can neglect differences of the battery voltage. However, the density of the air ρ is not constant as it depends on the air pressure and temperature. Additionally, wear and possible damages to the rotors might cause unexpected changes in C and A. These three values are difficult to measure, but we can estimate them together in hover flight. To do so, we first combine the three parameters and write Eq. (23) as

$$f = G\Omega^2. \tag{24}$$

We refer to this equation as thrust mapping, i.e., the mapping of the rotor speed to the resulting rotor thrust. Under nominal conditions, this thrust mapping can be estimated (e.g., with a load cell) to obtain the nominal coefficient $\check{G}$ leading to a nominal thrust mapping $\check{f} = \check{G}\Omega^2$. Due to the multiplicative nature of Eq. (23) and correspondingly Eq. (24), we can express the real thrust mapping coefficient as

$$G = \lambda \check{G}, \tag{25}$$

and hence the real produced thrust as

$$f = \lambda \check{f}. \tag{26}$$

This formulation allows us to estimate λ in hover and therefore calibrate the thrust mapping. For the quadrotor to hover, we know that $\boldsymbol{\tau} \stackrel{!}{=} 0$ and $c \stackrel{!}{=} g$. Thus, from Eqs. (7) and (6) we obtain the following matrix equation:

$$\begin{bmatrix} d & -d & -d & d \\ -d & -d & d & d \\ \kappa & -\kappa & \kappa & -\kappa \\ 1/m & 1/m & 1/m & 1/m \end{bmatrix} \begin{bmatrix} \check{f}_1\lambda_1 \\ \check{f}_2\lambda_2 \\ \check{f}_3\lambda_3 \\ \check{f}_4\lambda_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g \end{bmatrix}, \tag{27}$$

where $d = \frac{\sqrt{2}}{2}l$. This system of equations can be solved for $\lambda_{1,\dots,4}$. The nominal thrusts $\check{f}_i$ are obtained by averaging the applied nominal thrusts over N samples,

$$\check{f}_i = \frac{1}{N}\sum_{k=1}^{N} \check{f}_{i,k}. \tag{28}$$

To perform the thrust calibration, we sample the applied thrust commands over a 5 s period, which is sufficient to get a robust thrust estimation. Note that the nominal thrust mapping is stored on the vehicle and the nominally applied rotor thrusts $\check{f}_{i,k}$ are computed on the vehicle using the actually commanded rotor speeds and the nominal thrust mapping.

When controlling the vehicle, we first compute the actual desired rotor thrusts, as described in Section 6.3.2, which we then have to convert into the corresponding nominal rotor thrusts,

$$\check{f}_{i,\text{des}} = \frac{f_{i,\text{des}}}{\lambda_i}. \tag{29}$$

These nominal rotor thrusts are then converted into motor commands using the nominal thrust mapping onboard the vehicle.

6. REAL-TIME DENSE 3D RECONSTRUCTION

The MAV streams, through WiFi, the images $\mathbf{I}_k$ and corresponding poses $\mathbf{T}_{k,w}$ computed by SVO to the ground station at a rate of 5 Hz.[6] The observations are used to compute a dense reconstruction in real-time on the GPU. Therefore, we use the REMODE ("Regularized Monocular Depth") algorithm that we proposed in Pizzoli et al. (2014) and that we summarize in the following. REMODE computes dense depth maps for automatically selected reference frames. New reference frames are selected when the Euclidean distance to the previous reference frame, normalized by the average depth in the scene, exceeds a threshold. A depth map is computed by initializing a depth-filter for every pixel in a reference view r. We use the same depth-filter formulation as in the SVO algorithm, with the difference that now every pixel of the image has a depth-filter (rather than only salient features) and that the computation is performed highly parallelized on the GPU.

Finally, smoothness on the resulting depth map is enforced by minimizing a regularized energy functional. In the following, we give an overview of the depth-filter formulation and the smoothing step.

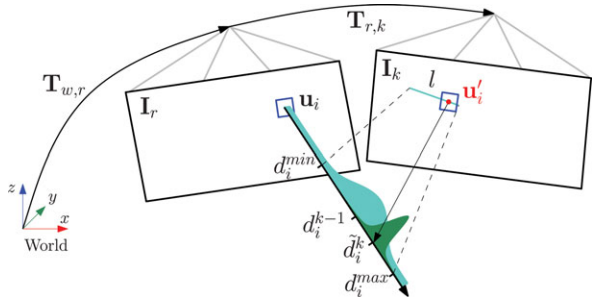


Figure 6. Probabilistic depth estimate $\hat{d}_i$ for feature i in the reference frame r. The point at the true depth projects to similar image regions in both images (blue squares). Thus, the depth estimate is updated with the triangulated depth $\tilde{d}_i^k$ computed from the point $\mathbf{u}'_i$ of highest correlation with the reference patch. The point of highest correlation lies always on the epipolar line in the new image.

6.1. Depth Filter

The depth computation for a single pixel is formalized as a Bayesian estimation problem. Let the rigid body transformation $\mathbf{T}_{w,r} \in SE(3)$ describe the pose of a reference frame relative to the world frame. We initialize a depth-filter at every pixel in the reference view with high uncertainty in depth and a mean set to the average scene depth of the previous reference frame. The depth filter is described by a parametric model (30) that is updated on the basis of every subsequent frame k.

Given a new observation $\{\mathbf{I}_k, \mathbf{T}_{k,w}\}$, we project the 95% depth-confidence interval $[d_i^{\min}, d_i^{\max}]$ of the depth filter corresponding to pixel i into the image k, and we find a segment of the epipolar line l (see Figure 6). Using the

[6]Although in our lab, we can transmit uncompressed, full-resolution images at rates of up to 50 Hz, we observed, during public exhibitions and outdoor experiments with more than 20-m height, that a lower bound of 5 Hz can usually be assumed.

cross-correlation score on a 5×5 patch, we search the pixel on the epipolar line segment $\mathbf{u}'_i$ that has the highest correlation with the reference pixel $\mathbf{u}_i$. A depth hypothesis $\tilde{d}_i^k$ is generated from the observation by triangulating $\mathbf{u}_i$ and $\mathbf{u}'_i$ from the views r and k, respectively.

Let the sequence of $\tilde{d}_i^k$ for $k = r, \ldots, r+n$ denote a set of noisy depth measurements. As proposed in Vogiatzis & Hernández (2011), we model the depth filter as a distribution that mixes a good measurement (normally distributed around the true depth d_i) and an outlier measurement (uniformly distributed in an interval $[d_i^{\min}, d_i^{\max}]$, which is known to contain the depth for the structure of interest):

$$p\left(\tilde{d}_i^k \mid d_i, \rho_i\right) = \rho_i \mathcal{N}\left(\tilde{d}_i^k | d_i, {\tau_i^k}^2\right) + (1-\rho_i)\mathcal{U}\left(\tilde{d}_i^k | d_i^{\min}, d_i^{\max}\right), \quad (30)$$

where ρ_i and ${\tau_i^k}^2$ are the probability (i.e., inlier ratio) and the variance of a good measurement, respectively. Assuming independent observations, the Bayesian estimation for d_i on the basis of the measurements $\tilde{d}_i^{r+1}, \ldots, \tilde{d}_i^k$ is given by the posterior

$$p(d_i, \rho_i | \tilde{d}_i^{r+1}, \ldots, \tilde{d}_i^k) \propto p(d_i, \rho_i) \prod_k p(\tilde{d}_i^k | d_i, \rho_i), \quad (31)$$

with $p(d_i, \rho_i)$ being a prior on the true depth and the ratio of good measurements supporting it. A sequential update is implemented by using the estimation at time step $k-1$ as a prior to combine with the observation at time step k. To this purpose, the authors of Vogiatzis & Hernández (2011) show that the posterior in Eq. (31) can be approximated by the product of a Gaussian distribution for the depth and a Beta distribution for the inlier ratio:

$$q(d_i, \rho_i \mid a_i^k, b_i^k, \mu_i^k, {\sigma_i^k}^2) = \text{Beta}(\rho_i | a_i^k, b_i^k)\mathcal{N}(d_i | \mu_i^k, {\sigma_i^k}^2), \quad (32)$$

where a_i^k and b_i^k are the parameters controlling the Beta distribution. The choice is motivated by the fact that $Beta \times Gaussian$ is the approximating distribution minimizing the Kullback-Leibler divergence from the true posterior (31). We refer to Vogiatzis & Hernández (2011) for an in-depth discussion and formalization of this Bayesian update step.

6.2. Depth Smoothing

In Pizzoli et al. (2014), we introduced a fast smoothing step that takes into account the measurement uncertainty to enforce spatial regularity and mitigates the effect of noisy camera localization.

We now detail our solution to the problem of smoothing the depth map $D(\mathbf{u})$. For every pixel $\mathbf{u}_i$ in the reference image $\mathbf{I}_r : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$, the depth estimation and its confidence upon the kth observation are given, respectively, by μ_i^k and σ_i^k in Eq. (32). We formulate the problem of computing a denoised depth map $F(\mathbf{u})$ as the following minimization:

$$\min_F \int_\Omega \left\{ G(\mathbf{u}) \left\| \nabla F(\mathbf{u}) \right\|_\epsilon + \lambda \left\| F(\mathbf{u}) - D(\mathbf{u}) \right\|_1 \right\} d\mathbf{u}, \quad (33)$$

where λ is a free parameter controlling the tradeoff between the data term and the regularizer, and $G(\mathbf{u})$ is a weighting function related to the "G-weighted total variation" introduced in Bresson, Esedoglu, Vandergheynst, Thiran, & Osher (2007) in the context of image segmentation. We penalize nonsmooth surfaces by making use of a regularization term based on the Huber norm of the gradient, defined as

$$\left\| \nabla F(\mathbf{u}) \right\|_\epsilon = \begin{cases} \dfrac{||\nabla F(\mathbf{u})||_2^2}{2\epsilon} & \text{if } ||\nabla F(\mathbf{u})||_2 \leq \epsilon, \\ ||\nabla F(\mathbf{u})||_1 - \dfrac{\epsilon}{2} & \text{otherwise.} \end{cases} \quad (34)$$

We chose the Huber norm because it allows smooth reconstruction while preserving discontinuities at strong depth gradient locations (Newcombe et al., 2011). The weighting function $G(\mathbf{u})$ influences the strength of the regularization and we propose to compute it on the basis of the measure confidence for $\mathbf{u}$:

$$G(\mathbf{u}) = \mathbb{E}_\rho[q](\mathbf{u}) \frac{\sigma^2(\mathbf{u})}{\sigma_{\max}^2} + \left\{1 - \mathbb{E}_\rho[q](\mathbf{u})\right\}, \quad (35)$$

where we have extended the notation for the expected value of the inlier ratio $\mathbb{E}_\rho[q]$ and the variance σ^2 in Eq. (32) to account for the specific pixel $\mathbf{u}$. The weighting function (35) affects the strength of the regularization term: for pixels with a high expected value for the inlier ratio ρ, the weight is controlled by the measurement variance σ^2; measurements characterized by a small variance (i.e., reliable measurements) will be less affected by the regularization; in contrast, the contribution of the regularization term will be heavier for measurements characterized by a small expected value for the inlier ratio or higher measurement variance.

The solution to the minimization problem (33) is computed iteratively based on the work in Chambolle & Pock (2011). The algorithm exploits the primal dual formulation of Eq. (33) and is detailed in Pizzoli et al. (2014).

7. RESULTS

7.1. Indoor flight

Our flying arena is equipped with an OptiTrack motion-capture system by NaturalPoint,[7] which we only used for ground-truth comparison. Its floor is covered with a texture-rich carpet and boxes for 3D structure as shown in Figure 7. The quadrotor was requested to autonomously execute a closed-loop trajectory specified by waypoints. The trajectory was 20 m long and the MAV flew on average at 1.7 m above ground. Figures 8, 9, and 10, show the accurate

[7] http://www.naturalpoint.com/optitrack/

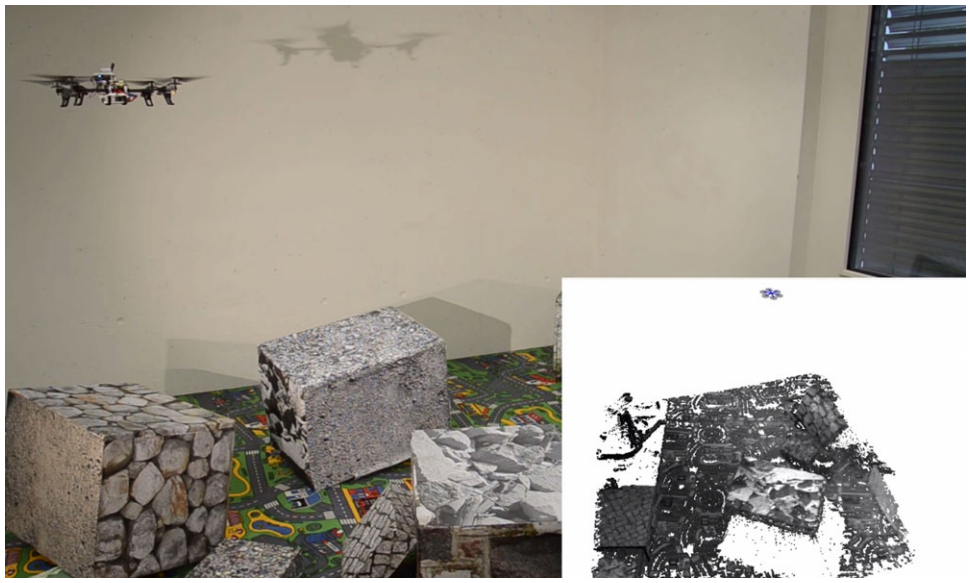


Figure 7. Our flying arena equipped with an OptiTrack motion-capture system (for ground-truth recording), carpets, and boxes for 3D structure.

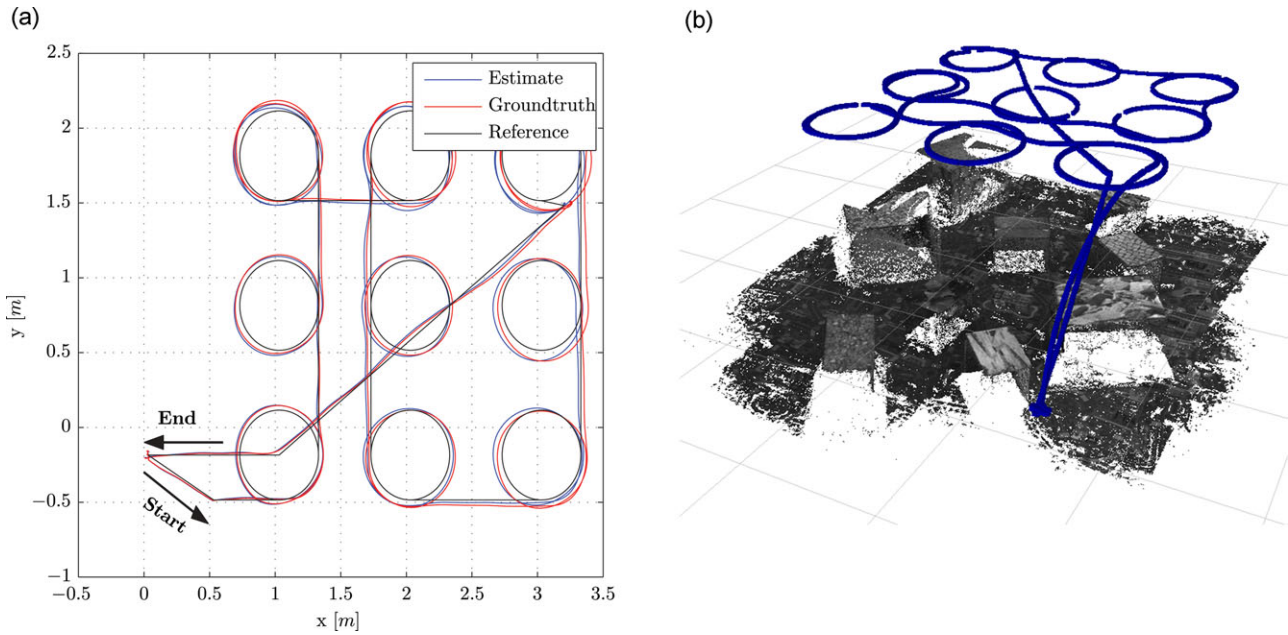


Figure 8. Comparison of estimated and ground-truth position to a reference trajectory (a) flown indoors over a scene as reconstructed in (b).

trajectory following as well as the position and orientation estimation errors, respectively. For ground-truth comparison, we aligned the first 2 m of the estimated trajectory to the ground truth (Umeyama, 1991). The maximum recorded position drift was 0.5% of the traveled distance. As observed, the quadrotor was able to return very close to the start point (with a final absolute error smaller than 5 cm). This result was confirmed in more than 100 experiments run at public events, exhibitions, and fairs. These results outperform those achieved with MAVs based on PTAM. This is due to the higher precision of SVO. Comparisons between SVO and PTAM are reported in Forster et al. (2014b).

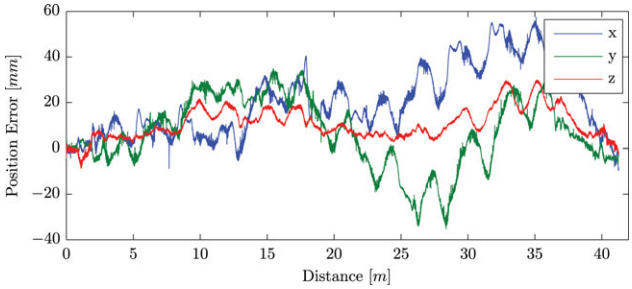


Figure 9. Error between estimated and ground-truth position for the trajectory illustrated in Figure 8.

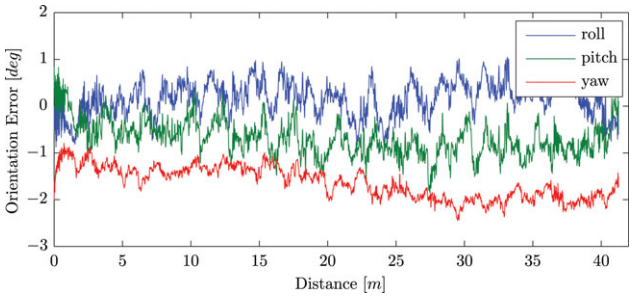


Figure 10. Error between estimated and ground-truth orientation for the trajectory illustrated in Figure 8.

Apart from its higher precision and frame rate, another main advantage of our SVO compared to PTAM is its robustness in scenes with repetitive and high-frequency textures (e.g., asphalt, grass); cf. Figure 11. Figure 12 shows a comparison of the map generated with PTAM and SVO in the same scene. While PTAM generates outlier 3D points,

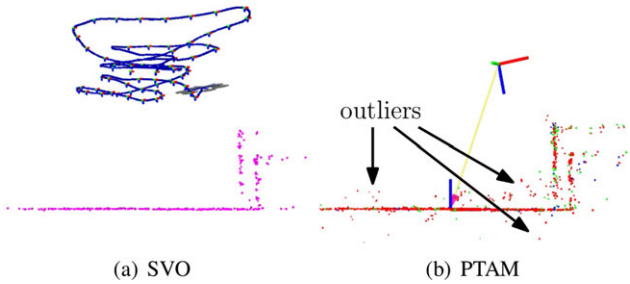


Figure 12. Side-view of a piecewise-planar map created by SVO and PTAM. The proposed method has fewer outliers due to the depth-filter.

by contrast SVO has almost no outliers thanks to the use of the depth-filter.

7.2. Outdoor Flight

An outdoor demonstration was performed at the Zurich firefighter training area in front of real firemen. This area features a large mock-up disaster site, consisting of two main towers and several stone blocks gathered on the ground. Due to the unavailability of a reliable GPS signal in this area, results are not compared to GPS. The quadrotor was requested to reach a height of 20 m and follow a piecewise straight trajectory (cf. Figure 13). The overall trajectory length was 140 m. The first 100 m of this trajectory were executed fully autonomously and are indicated in blue. After the autonomous flight, we handed a joypad game controller to a fireman with no pilot experience. The joypad was connected to the ground station and allowed sending the quadrotor simple up-down, left-right, forward-backward

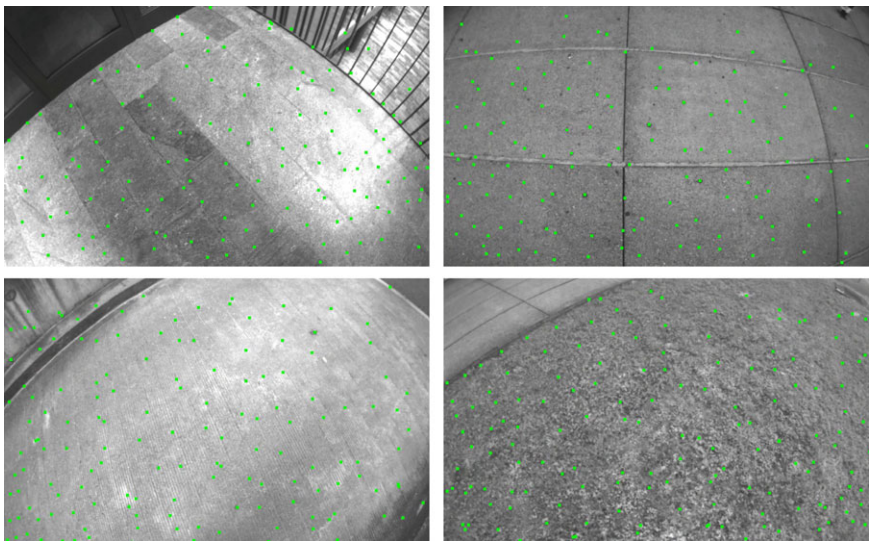
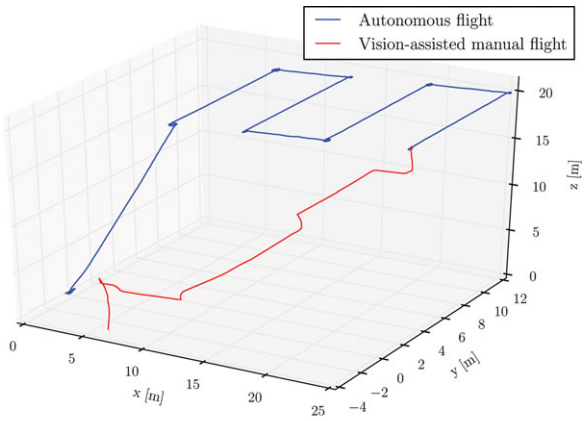
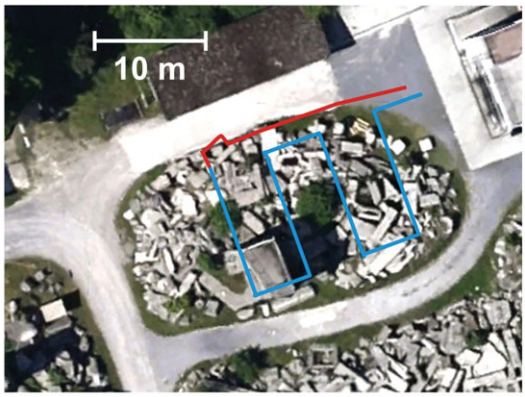


Figure 11. Successful tracking in scenes of high-frequency texture.



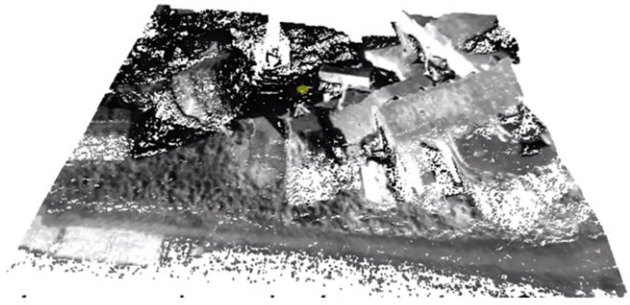
(a)



(b)

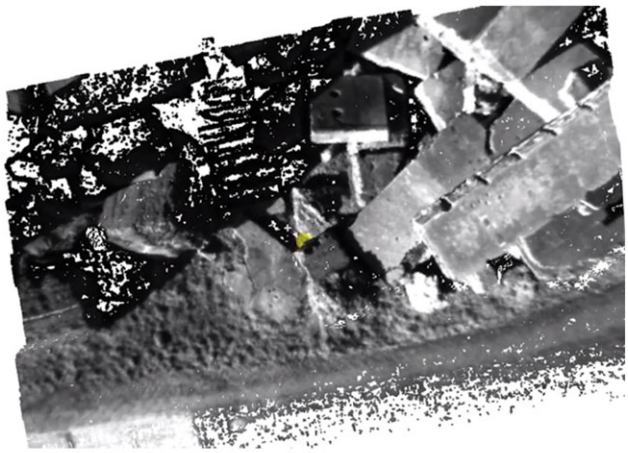
Figure 13. Outdoor trajectory of 140 m. Blue denotes the trajectory executed autonomously, red the one executed manually by a firefighter with no pilot experience assisted by the onboard vision-based controller.

velocity commands expressed in the world reference frame. Thanks to the assistance of the vision-based control running onboard the quadrotor, the firefighter was able to successfully and easily control the vehicle back to his position.



(a)

(b)



(c)

Figure 14. Outdoor, dense 3D reconstruction.

7.3. Reconstruction

The laptop that served as a ground station to run the live, dense 3D reconstruction is a Lenovo W520 with an Intel i7-3720QM processor, equipped with 16 GB of RAM, and an NVIDIA Quadro K2000M GPU with 384 CUDA cores.

Figures 14 and 15 show the dense reconstruction results from an indoor and outdoor scene, respectively.

To quantitatively evaluate our approach, we tested REMODE in Pizzoli et al. (2014) on a synthetic dataset provided in Handa et al. (2012). Figure 16 reports the main results of the reconstruction performance. The dataset consisted of views generated through ray-tracing from a three-dimensional synthetic model. The evaluation was based on a comparison with the ground truth depth map corresponding to the view taken as reference in the reconstruction process. As an evaluation metrics, we used the percentage of ground truth depths that have been estimated by the proposed method within a certain error (see Figure 17). To show the effectiveness of our approach, we compared our result with the depth map computed according to the state-of-the-art method introduced in Vogiatzis & Hernández (2011).

Our approach was capable of recovering a number of erroneous depth estimations, thus yielding a sensible improvement in terms of completeness. To verify the

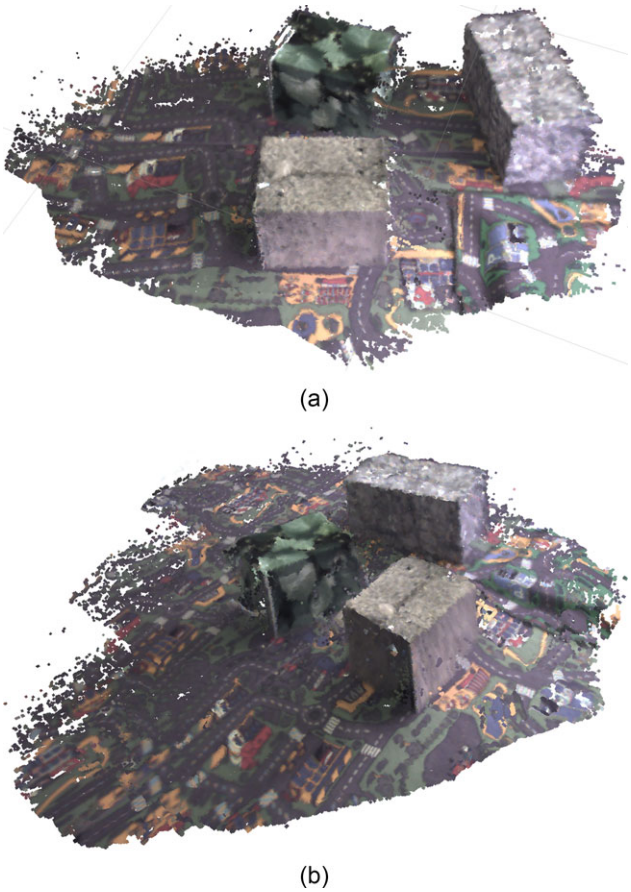


Figure 15. Indoor, dense 3D reconstruction.

robustness against noisy camera-pose estimation, we corrupted the camera position with Gaussian noise, with zero mean and 1-cm standard deviation on each coordinate. The results show that the completeness drops. This is inevitable due to the smaller number of converged estimations. However, the computation of the depth map takes advantage of the denoising step.

8. DISCUSSIONS AND CONCLUSION

8.1. Lessons Learned

8.1.1. System Design

The use of open-source software and hardware components was crucial to adapt them exactly to our needs. This allowed us to tune all the components such that they could work together properly and, hence, achieve a very good overall system performance. Furthermore, since we mostly used off-the-shelf hardware components, our quadrotor is relatively cheap, easy to upgrade, and fast to repair.

8.1.2. Indoor Experiments

In the proposed setup, the MAV relies on our visual odometry pipeline (i.e., SVO). However, several factors may disturb visual-odometry pipelines. Examples include flashlights of photographers, sudden illumination changes (e.g., when moving from a shadow area to an illuminated area), too rapid motion of the vehicle, or poor texture on the ground. If the visual odometry cannot recover within a certain amount of time, it is not possible to continue the mission and the quadrotor has to land. Since this can happen often in an experimental phase, it is crucial to have a safe landing procedure such that the quadrotor does not get damaged. To do so, we implemented an open-loop emergency-landing procedure, where the quadrotor stabilizes its attitude with the IMU and applies a thrust to slowly descend to the ground. An open-loop procedure is necessary since, especially in indoor environments, fall-backs, such as GPS, are not available.

8.1.3. Outdoor Experiments

In outdoor experiments, we noticed that the produced thrust can vary significantly due to different air temperature and pressure. This can have noticeable effects on the flight performance. Especially for an open-loop landing maneuver, after losing visual tracking it is crucial to know what the actually produced thrust is. For this reason, we implemented a routine to estimate the produced thrust (see Section 6.4.2) in the beginning of a mission during a short period of hover flight. With this routine we can account for different air densities and possible damage of the rotors.

In addition, outdoor experiments on partially cloudy days showed that it is crucial to have a camera with a high dynamic range. We set the camera settings before every mission and kept them fixed during the mission due to the weak performance of the camera-driver's auto-adjustment, which either flickers, overexposes, or underexposes the scene. This can cause the visual odometry to lose tracking when the quadrotor is transitioning between dark and bright scenes.

8.1.4. Dense Reconstruction

In the current reconstruction pipeline, the depth maps computed by REMODE are visualized as a point-cloud at the pose computed by SVO. However, as SVO drifts over time, the depth maps are not perfectly aligned. Therefore, in future work we will integrate REMODE in our collaborative mapping optimization back-end (Forster, Lynen, Kneip, & Scaramuzza, 2013a) in order to compute globally optimized maps. Additionally, in order to minimize the noise, we will apply a depth-map-fusion stage such as in Forster, Pizzoli, & Scaramuzza (2013b). Finally, depending on the height, the structure, and the appearance of the scene, different

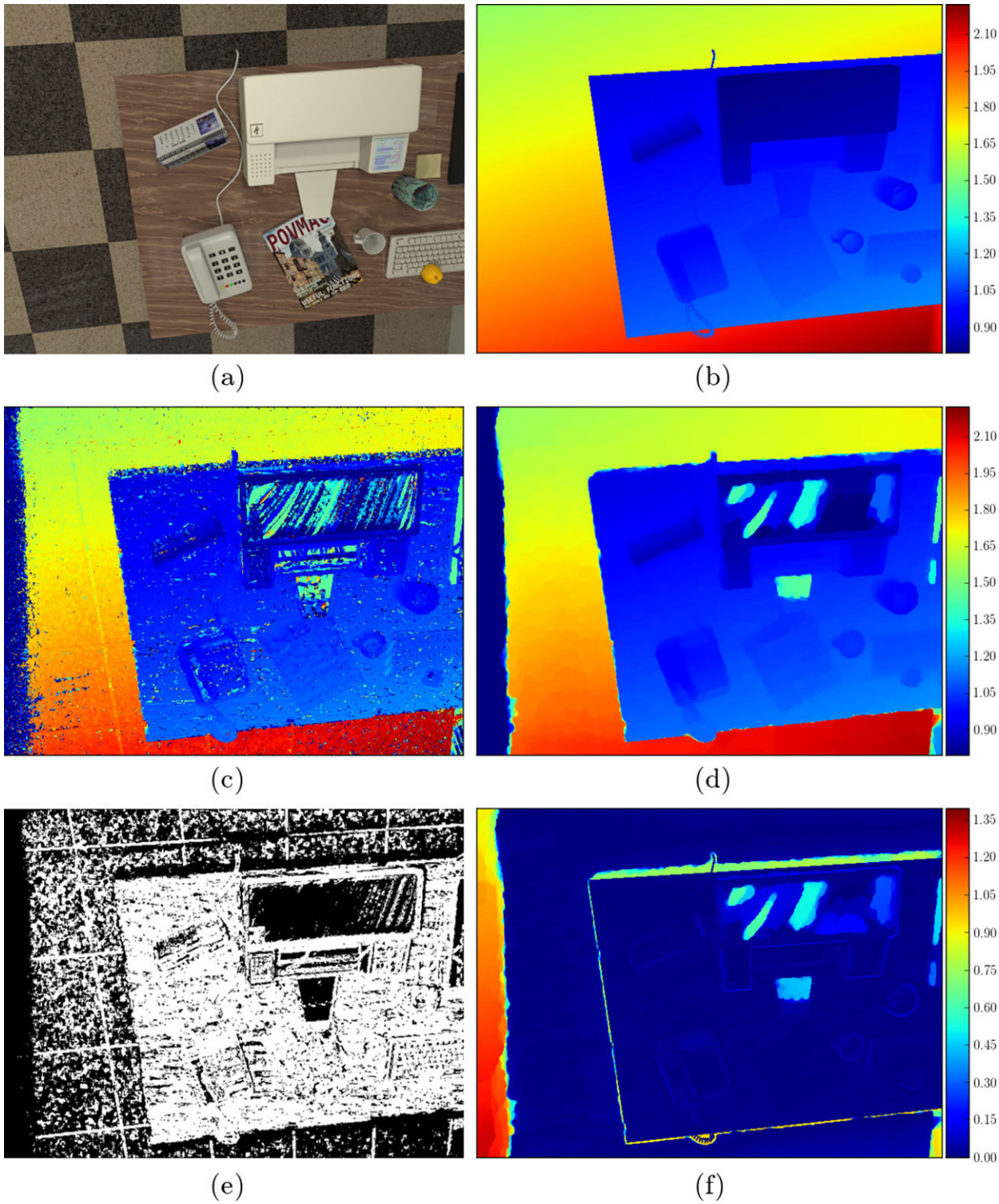


Figure 16. Evaluation of dense reconstruction on a synthetic dataset. (a) The reference view. (b) Ground truth depth map. (c) Depth map based on Vogiatzis & Hernández (2011). (d) Depth map computed by REMODE. (e) Map of reliable measurements (white pixels are classified as reliable). (f) Error of REMODE.

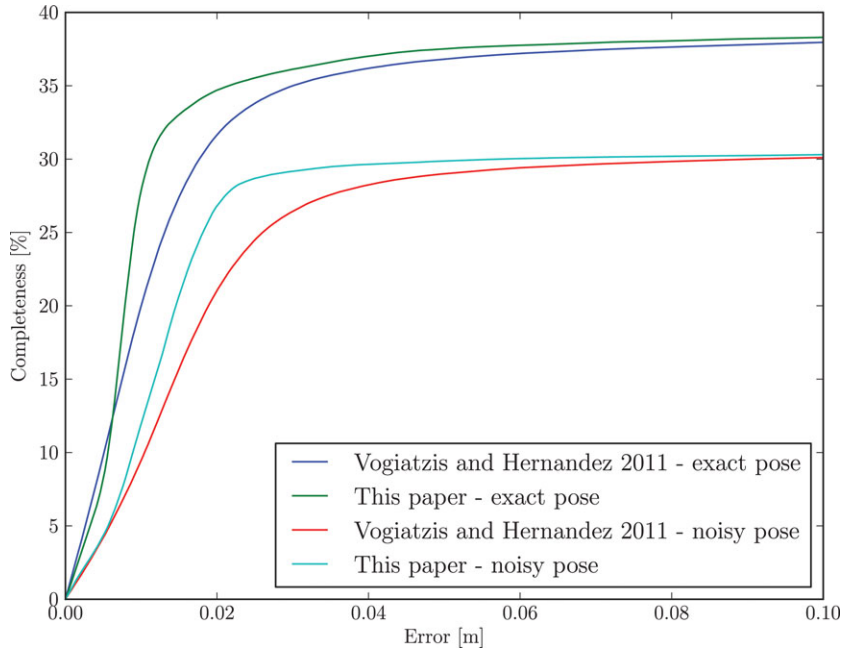


Figure 17. The completeness of the synthetic experiment, i.e., the percentage of ground truth measurements that are within a certain error from the converged estimations. For color blind readers: top line, this paper – exact pose; second line from the top, Vogiatzis and Hernandez 2011 – exact pose; third line from the top, this paper – noisy pose; bottom line, Vogiatzis and Hernandez 2011 – noisy pose.

motions are required to perform the depth-map reconstruction as fast as possible. Therefore, in Forster et al. (2014a), we proposed an active mapping formalization that we will integrate tightly into the current system.

8.2. Conclusion

We presented a system for mapping an unknown indoor or outdoor environment from the air in real time. Our system consists of an autonomous vision-based quadrotor and a ground station laptop for dense 3D reconstruction. The quadrotor can fly fully autonomously without relying on any external positioning system, which allows it to fly in unknown indoor and outdoor environments. This is crucial for search-and-rescue, where one cannot rely on any functional infrastructure. Furthermore, the autonomy of our quadrotor allows nonexpert operators to use our system with no training, as we demonstrated in the outdoor firefighter training area. From the images streamed by the quadrotor, we compute a live, dense 3D map on the ground station. Since the 3D map computation is performed in real time, the operator gets an immediate feedback without any unnecessary delays in the mission.

In numerous (more than 100) demonstrations, as well as indoor and outdoor experiments, we showed that our system works robustly, is easy to set up, and can easily be controlled by only one operator.

ACKNOWLEDGMENTS

This research was supported by the Swiss National Science Foundation through Project No. 200021-143607 ("Swarm of Flying Cameras"), the National Centre of Competence in Research Robotics, and the CTI Project No. 15424.2.

REFERENCES

Achtelik, M., Bachrach, A., He, R., Prentice, S., & Roy, N. (2009). Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. In SPIE Conference on Unmanned Systems Technology.

Ahrens, S., Levine, D., Andrews, G., & How, J. (2009). Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments. In IEEE International Conference on Robotics and Automation (ICRA).

Bachrach, A., He, R., & Roy, N. (2009). Autonomous flight in unstructured and unknown indoor environments. In European Micro Aerial Vehicle Conference.

Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A., Krainin, M., Maturana, D., Fox, D., & Roy, N. (2012). Estimation, planning and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. Journal of Field Robotics, 31(11), 1320–1343.

Bloesch, M., Weiss, S., Scaramuzza, D., & Siegwart, R. (2010). Vision based MAV navigation in unknown and

unstructured environments. In IEEE International Conference on Robotics and Automation (ICRA).

Bresson, X., Esedoglu, S., Vandergheynst, P., Thiran, J.-P., & Osher, S. (2007). Fast global minimization of the active contour/snake model. Journal of Mathematical Imaging and Vision, 28(2), 151–167.

Chambolle, A., & Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision, 40(1), 120–145.

Engel, J., Sturm, J., & Cremers, D. (2012). Camera-based navigation of a low-cost quadrocopter. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Forster, C., Lynen, S., Kneip, L., & Scaramuzza, D. (2013a). Collaborative monocular slam with multiple micro aerial vehicles. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Forster, C., Pizzoli, M., & Scaramuzza, D. (2013b). Air-ground localization and map augmentation using monocular dense reconstruction. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014a). Appearance-based active, monocular, dense depth estimation for micro aerial vehicles. In Robotics: Science and Systems (RSS), 1–8.

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014b). SVO: Fast semi-direct monocular visual odometry. In IEEE International Conference on Robotics and Automation (ICRA).

Gallup, D., Frahm, J.-M., Mordohai, P., Yang, Q., & Pollefeys, M. (2007). Real-time plane-sweeping stereo with multiple sweeping directions. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition.

Handa, A., Newcombe, R. A., Angeli, A., & Davison, A. J. (2012). Real-time camera tracking: When is high frame-rate best?

Hrabar, S., Sukhatme, G. S., Corke, P., Usher, K., & Roberts, J. (2005). Combined optic-ow and stereo-based navigation of urban canyons for a UAV. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Irani, M., & Anandan, P. (1999). All about direct methods. In Proceedings of the Workshop Vision Algorithms: Theory Practice (pp. 267–277).

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR) (pp. 225–234), Nara, Japan.

Lippiello, V., Loianno, G., & Siciliano, B. (2011). Mav indoor navigation based on a closed-form solution for absolute scale velocity estimation using optical flow and inertial data. In IEEE International Conference on Decision and Control.

Lupashin, S., Hehn, M., Mueller, M. W., Schoellig, A. P., Sherback, M., & D'Andrea, R. (2014). A platform for aerial robotics research and demonstration: The Flying Machine Arena. Journal of Mechatronics, 24(1), 41–54.

Lynen, S., Achtelik, M., Weiss, S., Chli, M., & Siegwart, R. (2013). A robust and modular multi-sensor fusion approach applied to MAV navigation. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Meier, L., Tanskanen, P., Heng, L., Lee, G. H., Fraundorfer, F., & Pollefeys, M. (2012). PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. Autonomous Robots, 33(1–2), 21–39.

Michael, N., Mellinger, D., Lindsey, Q., & Kumar, V. (2010). The GRASP multiple micro UAV testbed. IEEE Robotics Automation Magazine, 17(3), 56–65.

Murphy, R. (2014). Disaster Robotics. Cambridge, MA: MIT Press.

Newcombe, R., Lovegrove, S., & Davison, A. (2011). DTAM: Dense tracking and mapping in real-time. In International Conference on Computer Vision (ICCV) (pp. 2320–2327), Barcelona, Spain.

Pizzoli, M., Forster, C., & Scaramuzza, D. (2014). REMODE: Probabilistic, monocular dense reconstruction in real time. In IEEE International Conference on Robotics and Automation (ICRA).

Rosten, E., Porter, R., & Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. IEEE Transactions in Pattern Analysis and Machine Intelligence, 32(1), 105–119.

Rudin, L. I., Osher, S., & Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena, 60(1-4).

Ruffier, F., & Franceschini, N. (2004). Visually guided micro-aerial vehicle: Automatic take off, terrain following, landing and wind reaction. In IEEE International Conference on Robotics and Automation (ICRA).

Scaramuzza, D., Achtelik, M., Doitsidis, L., Fraundorfer, F., Kosmatopoulos, E. B., Martinelli, A., Achtelik, M. W., Chli, M., Chatzichristofis, S., Kneip, L., Gurdan, D., Heng, L., Lee, G., Lynen, S., Meier, L., Pollefeys, M., Renzaglia, A., Siegwart, R., Stumpf, J. C., Tanskanen, P., Troiani, C., & Weiss, S. (2014). Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments. IEEE Robotics Automation Magazine.

Scaramuzza, D., & Fraundorfer, F. (2011). Visual odometry. Part I: The first 30 years and fundamentals. IEEE Robotics Automation Magazine, 18(4), 80–92.

Schmid, K., Lutz, P., Tomic, T., Mair, E., & Hirschmuller, H. (2014). Autonomous vision-based micro air vehicle for indoor and outdoor navigation. Journal of Field Robotics, 31, 537–570.

Shen, S., Michael, N., & Kumar, V. (2011). Autonomous multi-floor indoor navigation with a computationally constrained MAV. In IEEE International Conference on Robotics and Automation (ICRA).

Shen, S., Michael, N., & Kumar, V. (2012a). Autonomous indoor 3D exploration with a micro-aerial vehicle. In IEEE International Conference on Robotics and Automation (ICRA).

Shen, S., Michael, N., & Kumar, V. (2012b). Stochastic differential equation-based exploration algorithm for autonomous indoor 3D exploration with a micro aerial vehicle. Journal of Field Robotics, 31(12), 1431–1444.

Shen, S., Mulgaonkar, Y., Michael, N., & Kumar, V. (2013). Vision-based state estimation and trajectory control towards aggressive flight with a quadrotor. In Robotics: Science and Systems (RSS), 1–8.

Strasdat, H., Montiel, J., & Davison, A. (2010). Real-time monocular SLAM: Why filter? In IEEE International Conference on Robotics and Automation (ICRA).

Stühmer, J., Gumhold, S., & Cremers, D. (2010). Real-time dense geometry from a handheld camera. In Pattern Recognition, vol. 6376 of Lecture Notes in Computer Science (pp. 11–20). Berlin: Springer.

Szeliski, R., & Scharstein, D. (2004). Sampling the disparity space image. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(3), 419–425.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., & Mahoney, P. (2007). Stanley: The robot that won the DARPA grand challenge. In Buehler, M., Iagnemma, K., & Singh, S. (eds.), The 2005 DARPA Grand Challenge, vol. 36 of Springer Tracts in Advanced Robotics (pp. 1–43). Berlin: Springer.

Triggs, B., McLauchlan, P., Hartley, R., & Fitzgibbon, A. (2000). Bundle adjustment—A modern synthesis. In Triggs, W., Zisserman, A., & Szeliski, R. (eds.), Vision Algorithms: Theory and Practice, vol. 1883 of LNCS (pp. 298–372). Springer-Verlag.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(4), 376–380.

Vogiatzis, G., & Hernández, C. (2011). Video-based, real-time multi view stereo. Image and Vision Computing, 29(7), 434–441.

Weiss, S., Achtelik, M. W., Lynen, S., Achtelik, M. C., Kneip, L., Chli, M., & Siegwart, R. (2013). Monocular vision for longterm micro aerial vehicle state estimation: A compendium. Journal of Field Robotics, 30(5), 803–831.

Weiss, S., Scaramuzza, D., & Siegwart, R. (2011). Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments. Journal of Field Robotics, 28(6), 854–874.

Wendel, A., Maurer, M., Graber, G., Pock, T., & Bischof, H. (2012). Dense reconstruction on-the-fly. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition.

Werlberger, M., Pock, T., & Bischof, H. (2010). Motion estimation with non-local total variation regularization. In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition.

Zingg, S., Scaramuzza, D., Weiss, S., & Siegwart, R. (2010). MAV navigation through indoor corridors using optical flow. In IEEE International Conference on Robotics and Automation (ICRA).

Zufferey, J.-C., & Floreano, D. (2006). Fly-inspired visual steering of an ultralight indoor aircraft. IEEE Transactions on Robotics 22(1), 137–146.