LUND UNIVERSITY

# Dense Tracking and Mapping with a Quadrocopter

Sturm, Jürgen; Bylow, Erik; Kerl, Christian; Kahl, Fredrik; Cremers, Daniel

2013

Link to publication

*Citation for published version (APA):*
Sturm, J., Bylow, E., Kerl, C., Kahl, F., & Cremers, D. (2013). *Dense Tracking and Mapping with a Quadrocopter*. Paper presented at UAV-g 2013, Rostock, Germany.

*Total number of authors:*
5

**General rights**
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

**Take down policy**
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Download date: 17. Dec. 2025

# DENSE TRACKING AND MAPPING WITH A QUADROCOPTER

J. Sturm[a,*], E. Bylow[b], C. Kerl[a], F. Kahl[b], and D. Cremers[a]

[a] Computer Vision Group, Department of Computer Science, Technical University of Munich, Germany
{juergen.sturm,christian.kerl,cremers}@in.tum.de
[b] Mathematical Imaging Group, Centre for Mathematical Sciences Faculty of Engineering, Lund University, Sweden
bylowerik@gmail.com, fredrik@maths.lth.se

**KEY WORDS:** quadrocopter, localization, 3D reconstruction, RGB-D sensors, real-time

## ABSTRACT

*In this paper, we present an approach for acquiring textured 3D models of room-sized indoor spaces using a quadrocopter. Such room models are for example useful for architects and interior designers as well as for factory planners and construction managers. The model is internally represented by a signed distance function (SDF) and the SDF is used to directly track the camera with respect to the model. Our solution enables accurate position control of the quadrocopter, so that it can automatically follow a pre-defined flight pattern. Our system provides live feedback of the acquired 3D model to the user. The final model consisting of a textured 3D triangle mesh can be saved in several standard CAD file formats.*

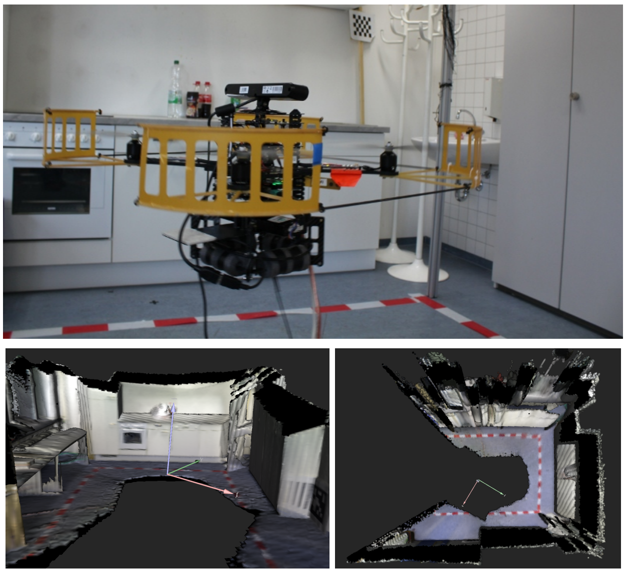

Figure 1: We present an efficient method for 3D reconstruction of indoor scenes using a quadrocopter equipped with an RGB-D camera. Top: AscTec Pelican platform used in our experiments. Bottom: Two different views on the reconstructed 3D model of our lab space.

## 1 INTRODUCTION

The ability to quickly scan a 3D model of a room or factory floor has many potential applications: For example, craftsmen can read of from such room models the size of a window or its height directly from the model. Furthermore, interior designers can illustrate the effects of decoration using such models, and potential buyers of an apartment can get a better impression of the real estate. While it was recently demonstrated that scanning a room with a hand-held sensor is feasible (Newcombe et al., 2011; Whelan et al., 2012; Bylow et al., 2013), quadrocopter-based scanning (Huang et al., 2011) has several advantages: First, scanning can be performed fully automatically and no human intervention is required. Second, it is possible to scan larger rooms (for example, industrial production halls) from above which might not be possible from the ground. Third, it is possible to use quadrocopters to scan inaccessible or dangerous buildings, e.g., after an earthquake.

In this paper, which is an extension of our recent work (Bylow et al., 2013), we first present our approach on 3D scanning using an RGB-D camera that (1) provides accurate pose information in real-time and (2) estimates a dense, textured 3D model of the scene. We demonstrate that our approach is fast, accurate, and robust enough to control the position of a quadrocopter and thus to generate a dense 3D model of a room fully automatically. Our approach relies on an RGB-D camera that yields dense depth images at video frame rate. While our current implementation relies on GPU support provided by an external ground station, we are working towards a scalable CPU implementation that will allow us to perform all computations onboard the quadrocopter.

While feature-based approaches to the structure-from-motion problem can be successfully applied to quadrocopter imagery and control, the resulting feature maps are typically sparse and thus not well suited for a visually pleasant 3D reconstruction (Agarwal et al., 2009; Weiss et al., 2012; Engel et al., 2012). Therefore, we advocate to represent the scene geometry in a dense grid using a truncated signed distance function (Curless and Levoy, 1996) similar as in the famous KinectFusion approach (Newcombe et al., 2011). While KinectFusion uses ray tracing to generate a point cloud in each frame and the iterated closest point (ICP) algorithm for subsequent alignment, we present in this paper a novel method that allows us to directly compute the camera pose based on the signed distance function.The key idea behind our approach is that the correct camera pose should minimize the value of the signed distance function evaluated at the back-projected points of the depth image. Furthermore, we fuse color information in an additional 3D voxel grid to generate a texture for the model. Figure 1(bottom) shows two views on the model acquired in this way from our lab space.

We employ an AscTec Pelican quadrocopter that we equipped with an Asus Xtion Pro Live sensor (see image at the top of Figure 1). An external ground station equipped with an Nvidia GeForce GTX 560 performs camera tracking and 3D reconstruction in real-time, and sends the estimated position back to the quadrocopter at 30 fps with a delay of approximately 50 ms. We integrate the position estimate using an extended Kalman filter and perform position control using LQR control (Weiss et al.,

---

This work has partially been supported by the DFG under contract number FO 180/17-1 in the Mapping on Demand (MOD) project.

2012). To evaluate our algorithm, we performed various flight patterns and measured the deviation from the desired route. Furthermore, we acquired 3D models of several rooms to demonstrate its applicability. Additionally, we evaluated our algorithm on a publicly available benchmark (Sturm et al., 2012). We found that our algorithm outperforms the KinectFusion implementation of the point cloud library (PCL) in terms of accuracy and robustness. Furthermore, we found that our approach yields a comparable performance in comparison to the RGB-D SLAM system (Endres et al., 2012). In contrast to these methods, our solution outputs a color-textured 3D model of the scene and is significantly faster.

This paper is an extension of our recent work (Bylow et al., 2013) in which we provide a more detailed evaluation of our quadrocopter experiments. In particular, we measured the position accuracy of a quadrocopter while hovering and path following using our approach. Furthermore, we provide additional scans that we acquired using the quadrocopter.

## 2 RELATED WORK

Simultaneous localization and mapping refers to both the estimation of the camera pose and mapping of the environment. This requires a suitable representation of the scene geometry, and the choice of this representation strongly influences the efficiency of pose estimation and map optimization.

Laser-based localization and mapping approaches often use scan matching or the iterated closest point algorithm (ICP) (Besl and McKay, 1992) to estimate the motion between frames. Graph SLAM methods use these motion estimates as input to construct and optimize a pose graph (Kümmerle et al., 2011). Typically, these methods render a joint map only after pose graph optimization, and this map is generally not used for further pose optimization. The resulting maps are often represented as occupancy grid maps or octrees (Wurm et al., 2010) and are therefore well suited for robot localization or path planning. Henry et al. (2010) were the first to apply the Graph SLAM approach to RGB-D data using a combination of visual features and ICP. A similar system was recently presented by Endres et al. (2012) and extensively evaluated on a public benchmark (Sturm et al., 2012). In this paper, we compare the performance of our approach to the RGB-D SLAM system and demonstrate that we achieve more detailed reconstructions and higher frame rates at a comparable pose accuracy.

Newcombe et al. (2011) recently demonstrated with their famous KinectFusion approach that dense reconstruction is possible in real-time by using a Microsoft Kinect sensor. To represent the geometry, Newcombe et al. employ a signed distance function (SDF) (Curless and Levoy, 1996) and use ICP in a coarse-to-fine manner to estimate the camera motion. For each image, the algorithm first renders a point cloud from the SDF at the previous pose using ray tracing and subsequently aligns this with the next depth image. Point correspondences are found using projective data association (Blais and Levine, 1993) and the point-to-plane distance (Chen and Medioni, 1992). As the original implementation is not available and no benchmark evaluation is provided, we compare our approach to the KinFu open-source implementation as available in the point cloud library (Kin). We show in this paper that our approach outperforms KinFu in terms of speed and accuracy.

While ICP only minimizes the error on point clouds, several approaches have recently appeared that minimize the photometric error (Steinbrücker et al., 2011; Kerl et al., 2013a) or combinations of both (Tykkälä et al., 2011; Kerl et al., 2013b), however without subsequent 3D reconstruction. Whelan et al. (2012) recently integrated these methods with the KinectFusion approach and demonstrated that superior tracking performance can be achieved, however without evaluating the global consistency of the resulting model.

While our approach on dense tracking and 3D reconstruction was first introduced in (Bylow et al., 2013), we provide in this paper a more in-depth evaluation of the resulting accuracy and stability when used with an autonomous quadrocopter. In particular, we evaluate the accuracy of keeping a particular position and following a pre-defined scanning trajectory. Furthermore, we provide scans from additional scenes to demonstrate its robustness and applicability in practice.

## 3 QUADROCOPTER-BASED 3D MODEL ACQUISITION

In this section, we explain how we acquire the 3D model of a room using an RGB-D camera mounted on a quadrocopter. As our approach currently requires a GPU to achieve real-time processing, we connected the RGB-D camera directly to a work station using a USB cable and perform all computations off-board. The estimated camera pose is then sent back to the quadrocopter and used for data fusion and position control which runs onboard the quadrocopter.

### 3.1 Live Dense 3D Reconstruction

In the following, we briefly explain how we track the camera pose and generate the dense 3D model. We kept this section intentionally short and refer the interested reader to (Newcombe et al., 2011; Bylow et al., 2013) for more details on signed distance functions, the KinectFusion algorithm, and our recent extensions.

**Preliminaries** In each time step, we obtain a color image and a depth image from the Kinect sensor, i.e.,

$$I_{RGB} : \mathbb{R}^2 \to \mathbb{R}^3 \text{ and } I_Z : \mathbb{R}^2 \to \mathbb{R}. \tag{1}$$

We assume that the depth image is already registered on to the color image, so that pixels between both images correspond. Furthermore, we require a signed distance function (SDF), a weight function, and a color function that are defined for each 3D point $\mathbf{p} \in \mathbb{R}^3$ within the reconstruction volume:

$$D : \mathbb{R}^3 \to \mathbb{R}, W : \mathbb{R}^3 \to \mathbb{R}, \text{ and } C : \mathbb{R}^3 \to \mathbb{R}^3. \tag{2}$$

The SDF represents the distance of each point to the closest surface, i.e., $D(\mathbf{p}) = 0$ holds for all points $\mathbf{p}$ lying on surface, $D(\mathbf{p}) < 0$ for free space, and $D(\mathbf{p}) > 0$ for occupied space. In the following, we treat $I_{RGB}$, $I_Z$, $D$, $W$, and $C$ as continuous functions, but we represent them internally as discrete pixel/voxel grids (of size 640×480 and 256×256×256, respectively). When access to a non-integer value is needed, we apply bi-/tri-linear interpolation between the neighboring values. We assume the pinhole camera model, which defines the relationship between a 3D point $\mathbf{p} = (x, y, z)^\top \in \mathbb{R}^3$ and a 2D pixel $\mathbf{x} = (i, j)^\top \in \mathbb{R}^2$ as follows,

$$(i, j)^\top = \pi(x, y, z) = \left( \frac{f_x x}{z} + c_x, \frac{f_y y}{z} + c_y \right)^\top. \tag{3}$$

Here, $f_x, f_y, c_x, c_y$ refer to the focal length and the optical center of the camera, respectively. In reverse, given the depth $z =$

---

http://svn.pointclouds.org/pcl/trunk/

Table 1: The root-mean square absolute trajectory error for KinFu and our method for different resolutions, metrics and datasets. Also the result for RGB-D SLAM are presented. More details on this evaluation can be found in (Bylow et al., 2013).

| Method | Res. | Teddy | fr1/desk | fr1/desk2 | fr3/house | fr1/floor | fr1/360 | fr1/room | fr1/plant | fr1/RPY | fr1/XYZ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| KinFu | 256 | 0.154 | 0.057 | 0.420 | 0.064 | Failed | 0.913 | 0.313 | 0.598 | 0.133 | 0.026 |
| KinFu | 512 | Failed | 0.068 | 0.635 | 0.061 | 1.479 | 0.591 | Failed | 0.281 | 0.081 | 0.025 |
| Point-To-Plane | 256 | **0.073** | 0.099 | 0.089 | 0.053 | **0.359** | 0.669 | 0.346 | 0.045 | 0.047 | 0.031 |
| Point-To-Plane | 512 | 0.122 | 0.091 | 0.515 | 0.054 | 0.732 | 0.562 | 0.123 | **0.041** | **0.043** | 0.026 |
| Point-To-Point | 256 | 0.089 | 0.038 | 0.072 | **0.039** | 0.674 | **0.357** | 0.187 | 0.050 | 0.045 | 0.028 |
| Point-To-Point | 512 | 0.122 | **0.037** | **0.069** | 0.040 | 0.544 | 0.375 | **0.078** | 0.047 | 0.043 | **0.023** |
| RGB-D SLAM | | 0.111 | 0.026 | 0.043 | 0.059 | 0.035 | 0.071 | 0.101 | 0.061 | 0.029 | 0.013 |

$I_Z(i,j)$ of a pixel $(i,j)$, we can reconstruct the corresponding 3D point using

$$\rho(i,j,z) = \left(\frac{(i-c_x)z}{f_x}, \frac{(j-c_y)z}{f_y}, z\right)^\top. \tag{4}$$

In each time step, we first estimate the current camera pose $\boldsymbol{\xi}$ given the current depth image $I_Z$ and SDF $D$, and subsequently integrate the new data into the voxel grids. We represent the current camera pose using twist coordinates, i.e.,

$$\boldsymbol{\xi} = (\omega_1, \omega_2, \omega_3, v_1, v_2, v_3) \in \mathbb{R}^6. \tag{5}$$

These Lie algebra coordinates form a minimal representation and are well suited for numerical optimization. Twist coordinates can be easily converted to a rotation matrix $R \in \mathbb{R}^{3\times 3}$ and translation vector $\mathbf{t} \in \mathbb{R}^3$ (and vice versa) when needed (Ma et al., 2003).

Finally, we assume that the noise of the Kinect sensor can be modeled with a Gaussian error function, i.e.,

$$p(z_{\text{obs}} \mid z_{\text{true}}) \propto \exp\left(-(z_{\text{true}} - z_{\text{obs}})^2/\sigma^2\right). \tag{6}$$

In principle, the noise of the Kinect (and any disparity-based distance sensor) is quadratically proportional to the distance, i.e., $\sigma \propto z_{\text{true}}^2$. However, in our current implementation, we assume a fixed $\sigma$ over all pixels.

**Camera pose estimation** Given a new depth image $I_Z$ and our current estimate of the SDF $D$, our goal is to find the camera pose $\boldsymbol{\xi}$ that best aligns the depth image with the SDF, i.e., each pixel of the depth image should (ideally) map onto the zero crossing in the signed distance function. Due to noise and other inaccuracies, the depth image will of course never perfectly match the SDF (nor will our estimate of the SDF be perfect). Therefore, we seek the camera pose that maximizes the observation likelihood of all pixels in the depth image, i.e.,

$$p(I_Z \mid \boldsymbol{\xi}, D) \propto \prod_{i,j} \exp(-D(R\mathbf{x}_{ij} + \mathbf{t})^2/\sigma^2), \tag{7}$$

where $R = R(\boldsymbol{\xi})$ is a short hand for the current camera rotation, $\mathbf{t} = \mathbf{t}(\boldsymbol{\xi})$ for the camera translation, and $\mathbf{x}_{ij} = \rho(i,j,I_Z(i,j))$ for the reconstructed 3D point to keep our notation uncluttered. Note that a circular motion constraint is not imposed in the estimation process. By taking the negative logarithm, we obtain

$$L(\boldsymbol{\xi}) \propto \sum_{i,j} D(R\mathbf{x}_{ij} + \mathbf{t})^2. \tag{8}$$

To find its minimum, we set the derivative to zero and apply the Gauss-Newton algorithm, i.e., we iteratively linearize $D(R\mathbf{x}_{ij} + \mathbf{t})$ with respect to the camera pose $\boldsymbol{\xi}$ at our current pose estimate and solve the linearized system.

Note that KinectFusion pursues a different (and less effective) approach to camera tracking, as it first extracts a second depth image from the SDF that it then aligns to the current depth image using the iteratively closest point algorithm (ICP). As this requires an intermediate data association step between both point clouds, this is computationally more involved. Furthermore, the projection of the SDF onto a depth image looses important information that cannot be used in the iterations of ICP. To evaluate the performance of both approaches, we recently compared (Bylow et al., 2013) our approach with the free KinFu implementation in PCL on publicly available datasets (Sturm et al., 2012). The results are presented in Tab. 1 and clearly show that our approach is significantly more accurate.

**Updating the SDF** After the current camera pose has been estimated, we update the SDF $D$, the weights $W$, and the texture $C$ similar to (Curless and Levoy, 1996; Bylow et al., 2013). We transform the global 3D coordinates $\mathbf{p} = (x,y,z)^\top$ of the voxel cell into the local frame of the current camera $\mathbf{p}' = (x', y', z')^\top = R^\top(\mathbf{p} - \mathbf{t})$. Then we compare the depth of this voxel cell $z'$ with the observed depth $I_Z(\pi(x', y', z'))$,

$$d_{\text{obs}} = z' - I_Z(\pi(x', y', z')). \tag{9}$$

As $d_{\text{obs}}$ is not the true distance but an approximation, $d_{\text{obs}}$ gets increasingly inaccurate the further we are away from the surface. Furthermore, the projective distance is inaccurate when the viewing angle is far from 90° onto the surface as well as in the vicinity of object boundaries and discontinuities. Therefore, we follow the approach of Curless and Levoy (1996) by truncating the distance function at a value of $\delta$ and defining a weighting function to express our confidence in this approximation:

$$d(d_{\text{obs}}) = \begin{cases} -\delta & \text{if} \quad d_{\text{obs}} < -\delta \\ d_{\text{obs}} & \text{if} \quad |d_{\text{obs}}| \leq \delta \\ \delta & \text{if} \quad d_{\text{obs}} > \delta \end{cases}, \tag{10}$$

$$w(d_{\text{obs}}) = \begin{cases} 1 & \text{if } d_{\text{obs}} \leq 0 \\ exp(-(d_{\text{obs}}/\sigma)^2) & \text{if } d_{\text{obs}} > 0 \end{cases}. \tag{11}$$

A visualization of these functions is given in Fig. 2.

Experimentally, we determined $\delta = 0.3m$ to work well for our application. To interpret this number, notice that $\delta$ expresses our prior about the average thickness of objects in the scene. As our primary objects of interest are things such as walls, cabinets, chairs, and tables, such a prior seems reasonable. In our experiments, we found that larger $\delta$ lead to more stable tracking but less accurate reconstructions, while smaller $\delta$ generally lead to more accurate reconstructions with more details, but reduced stability, i.e., a diverging map after jumps in the flight behavior. We believe that this instability can be explained as follows: Imagine the quadrocopter has built up the SDF up to a distance of $\delta$ from the observed surface. When the quadrocopter now jumps forward by a distance of $\delta$, the SDF does not contain any useful information

---

http://svn.pointclouds.org/pcl/trunk/

anymore to recover the camera pose. In our flight experiments, we observed (occasionally) translational jumps of up to 0.08m, so that $\delta = 0.3m$ is a reasonable choice.

We update each voxel cell with (global) 3D coordinates $(x, y, z)^\top$ according to

$$D \leftarrow (WD + wd)/(W + w), \tag{12}$$
$$C \leftarrow (WC + wc)/(W + w), \tag{13}$$
$$W \leftarrow W + w, \tag{14}$$

where $c = I_{RGB}(\pi(x', y', z'))$ is the color from the RGB image.

Both steps (the estimation of the camera pose and updating the voxel grids) can be easily parallelized using CUDA. With our current implementation, the computation time per frame is approximately 27ms on a Nvidia GeForce GTX 560 with 384 cores, and runs thus easily in real-time with 30fps.

**Visualization** With the algorithm described above, we obtain an estimate of the signed distance and color for every cell of the voxel grid. To display this model to the user, we copy the data every two seconds from the GPU to the CPU (which consumes 60ms) and run a threaded CPU implementation of the marching cubes algorithm (Lorensen and Cline, 1987). The mesh generation takes around between 1000 and 1500ms on a single CPU core. The resulting triangle mesh typically consists of approximately 200.000–500.000 vertices (and faces), that we display together with the estimated camera trajectory to the user using OpenGL.

**Quadrocopter Control** Figure 3 shows the flow diagram of the control architecture used in our approach (Weiss et al., 2012). The low level processor (LLP) provides the attitude control for the platform. The high level processor (HLP) runs the extended Kalman filter and position/velocity control at 1KHz, and accepts velocity commands and waypoints. The error controller is a LQR controller for each axis. The feed forward model allows the quadrocopter to quickly reach the waypoint and reduces overshoot. It is possible to specify the approach speed and accuracy with which the quadrocopter should reach the goal location. The onboard PC running ROS supplies the HLP with the visual pose estimates and waypoints at 30Hz. The dense tracking and 3D reconstruction module runs on an external workstation with a GPU that is directly connected to the RGB-D camera on the quadrocopter and the onboard PC.

## 4 QUADROCOPTER EXPERIMENTS

In this section, we present the results of our experiments evaluation. First, we provide an evaluation of the flight stability and accuracy when the quadrocopter is controlled using the poses from our approach. Second, we present and discuss the 3D models that we obtained in different indoor environments using our approach.

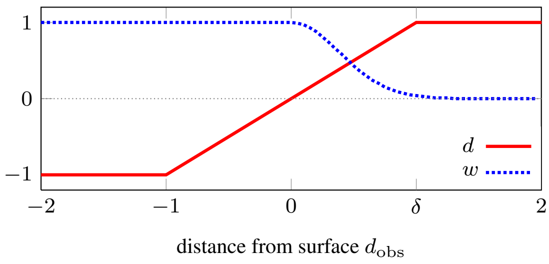

Figure 2: We use a truncated distance function and a weighting function that decreases exponentially behind the observed surface (i.e., for $d_{\text{obs}} > 0$).

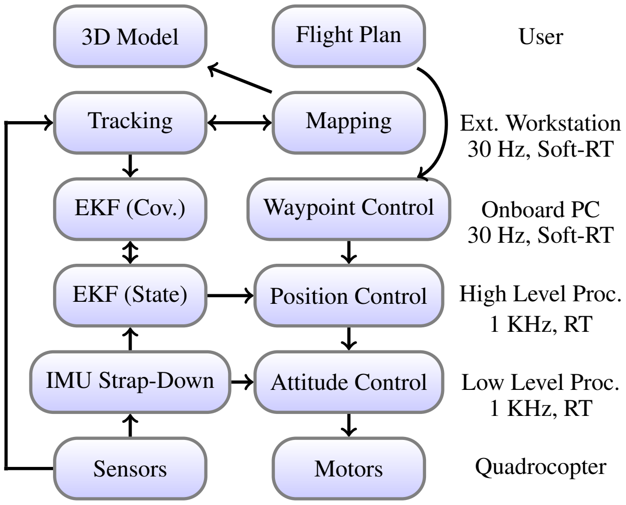

Figure 3: Control architecture used by our approach. The quadrocopter has three processors that operate at different cycle times.

### 4.1 Flight Accuracy

In all of our experiments, we initialized the map with the quadrocopter standing on the ground. From then onward, the visual pose estimate from our approach is fed into the EKF and used for position control. To initiate take-off, we send a positive velocity in z-direction to the position controller. After the quadrocopter reaches a certain height, we switch to waypoint control.

In our first experiment, we started the quadrocopter as described and commanded a single way point as its goal location. Figure 4 shows the result. As can be seen from this plot, the quadrocopter is able to accurately maintain the desired goal location. In particular, we measured an average standard deviation of 2.1cm during hovering.

In our second experiment, we provided a rectangle as the flight plan to the quadrocopter. As can be seen from Fig. 5, the quadrocopter closely follows the generated waypoints. Furthermore, we could repeat this procedure for several minutes without noticing any degradation of the 3D model or the flight stability.

### 4.2 3D Scans

We used our approach to scan three different rooms to verify the stability, robustness, and applicability of our approach.

The first room, shown in Fig. 1, is our quadrocopter lab. It has a kitchen unit in front, several cabinets with little structure and texture on the right, and tables on the left. The quadrocopter performed an autonomous take-off and followed a pre-defined motion along a half-circle. As can be seen from the two images in the bottom row, the reconstructed 3D model provides a good impression of the scene. An architect or interior designer could clearly use this model to measures the size of the room or other distances. Furthermore, as can be seen from the top-down view, there is only little drift: The opposing walls (which have never been observed simultaneously) are mostly parallel.

Furthermore, we scanned a normal office in our lab with four workspaces, see Fig. 6. Here, we sent a different flight pattern to

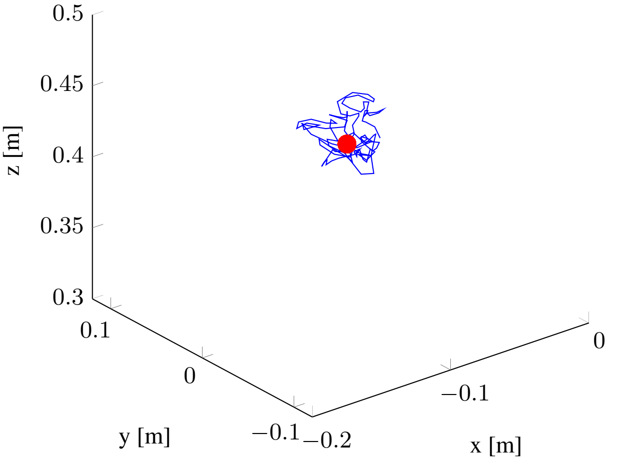

Figure 4: Hovering experiment using the position information from the proposed approach. The average deviation from the set point was 2.1cm. Blue: Estimated position of the quadrocopter. Red: Goal location.

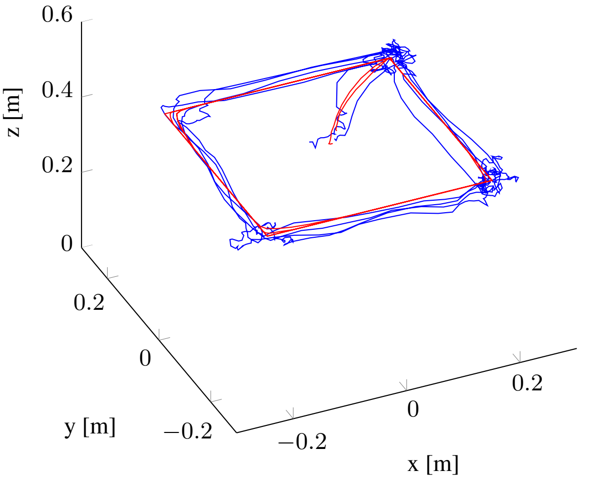

Figure 5: Path following experiment (rectangle). Blue: Estimated position of the quadrocopter. Red: Waypoints. As can be seen from this plot, the quadrocopter follows accurately and robustly the commanded trajectory for several rounds.

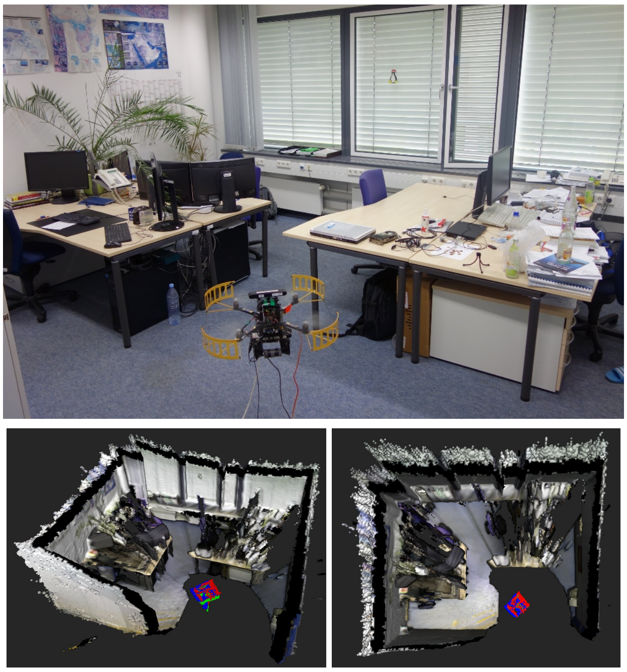

Figure 6: Office space with four desks scanned with our quadrocopter. The quadrocopter followed a pre-defined flight plan (rectangle and rotations) to acquire the 3D model. Top: Image from external camera. Bottom: Reconstructed 3D models from side view and top view. The coordinate axes correspond to the estimated trajectory.

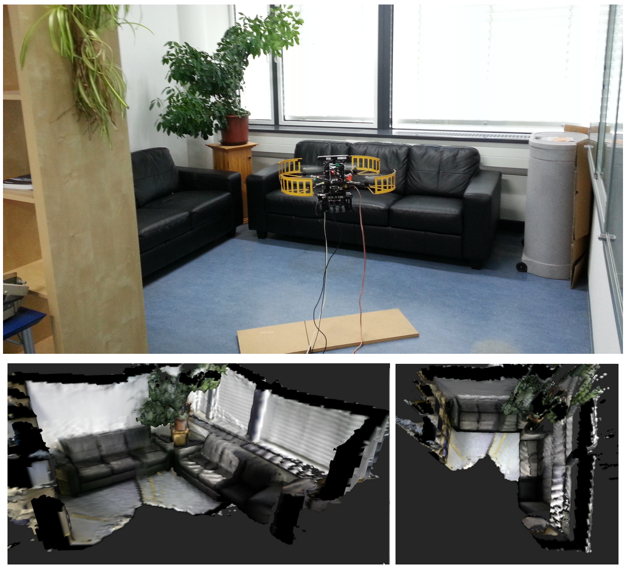

Figure 7: Couch corner in our lab. In this experiment, a human pilot specified the waypoints during scanning. The quadrocopter approached these waypoints using our approach. Top: Image from external camera. Bottom: Reconstructed 3D models from side view and top view.

the robot, consisting of a rectangular motion in combination with a 270° turn. The estimated trajectory is visualized on top of the reconstructed models in the bottom row.

Finally, we scanned the couch space in our lab as shown in Fig. 7. In this experiment, we used the assisted flight mode, i.e., a pilot specified the waypoint manually with the remote control, while the quadrocopter used the estimated pose to approach this pose. This mode has the advantage that this solution is more flexible, as the pilot can specify additional waypoints while the room is being scanned. In contrast to manual flight, the cognitive load of the pilot is greatly reduced as only a waypoint has to be specified and no control commands have to be issued.

## 5 CONCLUSION

In this paper, we have presented a novel technique to acquire high-quality 3D models of rooms using an autonomous quadrocopter. We incrementally construct a signed distance function of the scene and estimate the current camera pose with respect to the SDF. In our experiments, we demonstrated that camera pose estimation is fast, accurate, and robust enough to be used for position control of an autonomous quadrocopter. We acquired various 3D models of office rooms in our lab to demonstrate the validity of our approach. The resulting 3D models are valuable for various tasks, including interior design, architecture, or refurnishing work.

Despite these promising results, there are several aspects remaining for future work. First, we would like to investigate whether similar results can be obtained with stereo cameras. This would allow us to scan outdoor scenes, for example, on construction sites. Second, we are currently working on a CPU implementation that can run in real-time (but at a reduced resolution) on board the quadrocopter. Third, we solely use the reconstructed 3D model at the moment to display it to the user. However, it could also be used during assisted flight, e.g., to avoid collisions. Moreover, it would be interesting to generate the next waypoint based on autonomous exploration using the partial map.

## References

KinectFusion Implementation in the Point Cloud Library (PCL). `http://svn.pointclouds.org/pcl/trunk/`.

S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, and R.Szeliski. Building rome in a day. In *ICCV*, 2009.

P.J. Besl and N.D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

G. Blais and M.D. Levine. Registering multiview range data to create 3D computer objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17:820–824, 1993.

E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *RSS*, 2013.

Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992.

B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *ICRA*, May 2012.

J. Engel, J. Sturm, and D. Cremers. Camera-Based Navigation of a Low-Cost Quadrocopter. In *IEEE/RSJ Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proc. of the International Symposium on Experimental Robotics (ISER)*, 2010.

A.S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Intl. Symp. of Robotics Research (ISRR)*, 2011.

C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2013a.

C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2013b.

R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision: From Images to Geometric Models*. Springer Verlag, 2003.

R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, and A.W. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011.

F. Steinbrücker, J. Sturm, and D. Cremers. Real-time visual odometry from dense rgb-d images. In *Workshop on Live Dense Reconstruction with Moving Cameras at ICCV*, 2011.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

T.M. Tykkälä, C. Audras, and A.I Comport. Direct iterative closest point for real-time visual odometry. In *Workshop on Computer Vision in Vehicle Technology at ICCV*, 2011.

S Weiss, M Achtelik, M Chli, and R Siegwart. Versatile Distributed Pose Estimation and Sensor Self-Calibration for an Autonomous MAV. In *IEEE Intl. Conference on Robotics and Automation (ICRA)*, 2012.

T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J.B. McDonald. Robust tracking for real-time dense RGB-D mapping with Kintinuous. Technical report, MIT, 2012.

K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation at ICRA*, 2010.