# Collision Avoidance for Quadrotors with a Monocular Camera

H. Alvarez[1], L.M. Paz[2,3], J. Sturm[1], and D. Cremers[1] ⋆

[1] Department of Computer Science,Technische Universitt Mnchen,
Informatik 9, Boltzmannstrasse 3, 85748 Garching, Germany
humalvarez@mytum.de, {sturmju,cremers}@in.tum.de
[2] Department of Engineering Science, University of Oxford, Parks Road, OX1 3PJ Oxford,UK
linapaz@robots.ox.ac.uk
[3] I3A Instituto de Invertigación en Ingeniería de Aragón, University of Zaragoza, C/Mariano
Esquillor s/n, 50800 Zaragoza, Spain
linapaz@unizar.es

**Abstract.** Automatic obstacle detection and avoidance is a key component for the success of micro-aerial vehicles (MAVs) in the future. As the payload of MAVs is highly constrained, cameras are attractive sensors because they are both lightweight and provide rich information about the environment. In this paper, we present an approach that allows a quadrotor with a single monocular camera to locally generate collision-free waypoints. We acquire a small set of images while the quadrotor is hovering from which we compute a dense depth map. Based on this depth map, we render a 2D scan and generate a suitable waypoint for navigation. In our experiments, we found that the pose variation during hovering is already sufficient to obtain suitable depth maps. The computation takes less than one second which renders our approach applicable for obstacle avoidance in real-time. We demonstrate the validity of our approach in challenging environments where we navigate a Parrot Ardrone quadrotor successfully through narrow passages including doors, boxes, and people.

**Keywords:** collision avoidance, monocular depth-maps, micro-aerial vehicles.

## 1 Introduction

Micro-aerial vehicles (MAVs) have enormously gained in popularity over the past years. They are envisioned as versatile helpers for many different applications such as aerial surveillance, visual inspection, remote farming and filming.

All of these applications require that the quadrotors do not collide with the environment. While this can be achieved by a human pilot, it requires much training and puts a significant cognitive load onto the human operator. Therefore, it is desirable to enable the quadrotor to detect and avoid obstacles automatically. This can be achieved

⋆ This work was partially supported by the German Academic Exchange Service (DAAD), the DFG under contract number FO 180/17-1 in the Mapping on Demand (MOD) project and the Ministerio de Economía y Competitividad under project DPI2012-36070: Semantic and Active SLAM for heterogeneous systems.

by adding distance sensors to the quadrotor, such as ultrasound sensors, laser-scanners [1,2], stereo cameras [3,4], Kinect [5], or combinations of multiple sensors [6,7]. However, adding such sensors comes at the price of an increased weight and power consumption, so that they are not applicable to small-scale or nano-quadrotors. For example, platforms such as the Parrot Ardrone or the Bitcraze Crazyflie can only support a single monocular camera in terms of payload. Therefore, we are interested in methods for range mapping and obstacle avoidance for mobile robots that only have access to a single monocular camera.

Most structure-from-motion approaches (SfM) based on visual features such as PTAM [8] produce sparse maps that are not suited for collision-free navigation [9]: Typically the resulting point cloud of 3D features is highly noisy and, more importantly, the absence of visual features in regions with low texture does not necessarily imply free space. For example, although a cabinet might generate visual features around the edges or at the corners, its unicolored surface might not. In contrast, dense stereo methods [10,11] are able to propagate depth information from the corners to texture-less regions through regularization, but are computationally intensive and difficult to apply on noisy quadrotor data. Other approaches implement reactive obstacle avoidance for quadrotors using optical flow [12], relative size change [13] or reinforcement learning [14].

The goal of this paper is to combine the advantages of dense mapping [11] with the robustness of feature-based SfM methods [8]. This enables collision-free navigation of a quadrotor during forward flight. Our approach consists of two steps: First, we compute a dense depth map from a small set of images (typically 30). Subsequently, we use this depth map to generate the next obstacle-free waypoint in forward direction (see Fig. 1). In contrast to previous work [9], this approach yields a dense depth map at approximately 1 Hz, so that we obtain absolute distance estimates for every pixel in the image in near real-time. In our experiments, we demonstrate that by using our approach, a quadrotor can safely navigate through and around obstacles such as boxes, doors, and persons. Furthermore, and not expected before experimentation, we found that no additional quadrotor motion is required during image acquisition: Small movements of the quadrotor during hovering are already sufficient to generate satisfactory depth maps. As a consequence, we expect that our approach applies to a large range of micro aerial vehicles.

In a series of experiments with a real quadrotor, we demonstrate the validity of our approach. The accompanying video `collision_avoidance.m4v` provide additional examples and more details. For all of our experiments, we used a Parrot Ardrone 2 quadrotor, but the proposed approach equally transfers to any mobile platform equipped with a monocular camera. Due to the limited computational resources on the quadrotor, we currently perform all computations on an external base station that communicates over wireless network with the quadrotor.

## 2 Related Work

Existing approaches to obstacle avoidance range from purely reactive approaches to full planning-based approaches [15,16]. Reactive approaches aim at directly generating motion commands from sensor readings. As no intermediate representation has to

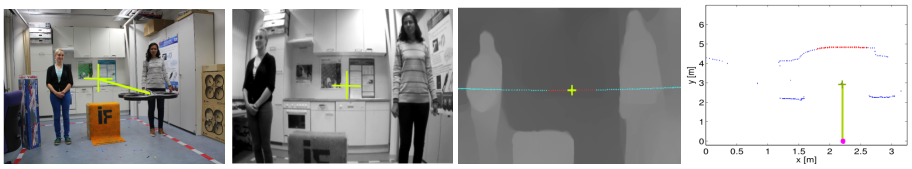



Fig. 1: Our goal is to enable a quadrotor with a monocular camera to avoid frontal obstacles and to navigate through narrow spaces. Left: Quadrotor hovering in front of several obstacles, including persons. Left middle: Camera view from the quadrotor. Right middle: Estimated dense depth image. Right: Top view of extracted 2D range scan and visualization of generated path.

be formed, reactive approaches are typically computationally lightweight and have low latency with well defined performance guarantees. Therefore, reactive approaches are well suited as safety mechanisms. Next to distance sensors, reactive approaches can also be applied to visual cues. For example, optical flow [12] can be used to determine the relative distance of an object to the quadrotor if the absolute speed of the quadrotor is known. However, optical flow can only be used with side-view cameras, as the optical flow of frontal obstacles is zero or close to zero. As an alternative, Mori and Scherer [13] recently proposed to detect relative size change in the image, which indicates approaching objects. Ross et al. [14] proposed an approach based on imitation learning, where multiple cues including optical flow and visual features are used for motion prediction.

When two frontal cameras are available, stereo methods can be used to generate a depth map [17]. If only a single camera is available (or, as in our case, affordable in terms of payload), stereo can also be computed between two or more consecutive images when the camera is moving. This is also called motion parallax. As images are inexpensive to acquire, Newcombe et al. [11] proposed in their DTAM approach to first accumulate a cost volume of 10's to 100's of images before generating a depth map. In contrast to classical two-view stereo methods, much more data from the environment is available in the cost volume which leads to more accurate depth maps. On the cost volume, regularization can be applied to infer missing values in the depth map and to reduce noise. Graber et al. [18] similarly compute a depth map from multiple images using the computationally and memory-wise less expensive plane sweep algorithm, but perform a more complex regularization during subsequent 3D reconstruction.

Both methods need accurate camera poses to construct the cost volume. While DTAM employs direct tracking on the currently estimated depth map, it implicitly assumes that the camera moves only slowly and with sufficient translational motion. On a quadrotor as the Ardrone, this is difficult to guarantee, as all flying maneuvers induce a change in attitude which presumably would often lead to a loss of tracking. Therefore, we decided to use the feature-based structure-from-motion approach of Klein and Murray [8], which contains an automatic recovery procedure and provides in our experience highly accurate camera poses. To integrate the visual information with the IMU data and to generate the control commands for the quadrotor, we make use of the tum_ardrone package of Engel et. al [9].

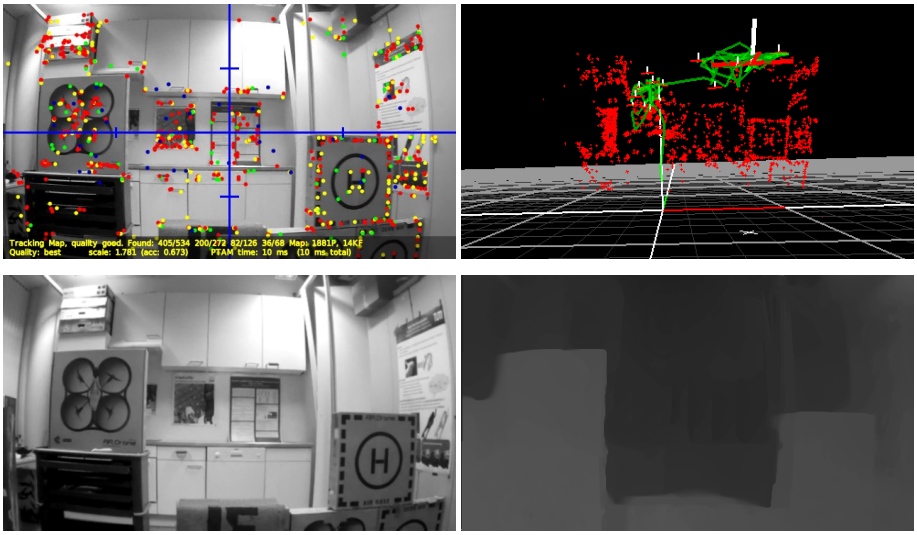


Fig. 2: Top: Sparse feature maps, as for example generated by PTAM, cannot represent regions without texture (e.g., the upper kitchen cabinets). Therefore, they are generally not suitable for path planning or obstacle avoidance. Bottom: A dense depth map using regularization provides distance estimates for every pixel.

In this paper, we demonstrate that a single forward-facing camera on a low-cost quadrotor is enough to generate dense depth maps that are suitable for obstacle avoidance. We incrementally generate the next way point based on the current depth map. It should be noted that in this work, we do explicitly not deal with the problem of global path planning [15], as this requires a global (obstacle) map which is typically not available beforehand. Yet, it would be interesting in future work to integrate the individual depth maps into a global 3D map that could then be used for global path planning.

## 3 Dense Depth Map Estimation

The main challenge for our application is that although a sparse, feature-based representation provided by popular monocular SLAM/SfM approaches is sufficient for localization, it is generally not sufficient for autonomous path-planning. The reasons for this are that (1) large sparely-textured obstacles typically have no visual features on them and (2) outliers introduce unwanted obstacles in free space.

In Fig. 2, we illustrate this difficulty using the example of the PTAM system. The images in the top show the sparse keypoints (visualized as points) and the key frames (visualized with small axes in the top right image). Although PTAM detects many features in various parts of the scene, it does not contain any visual features for the upper cabinets, and therefore, a path planning system might generate a path that leads straight through the cabinet. In contrast, the dense depth map visualized in the bottom right

image provides distance information for every pixel in the scene, in particular on the cabinet in the background as well as on the boxes in front.

In the following, we introduce our algorithm to estimate a regularized depth map from multiple input images. We assume that we obtain from a quadrotor a sequence of $n$ gray-scale images $I_1,\dots,I_n : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$ with corresponding camera poses $T_1,\dots,T_n \in SE(3)$ that were gathered during hovering. In practice, we obtain the camera poses from PTAM as used by [9]. The geometry of our setup is visualized schematically in Fig. 3, where the quadrotor (left column) is looking at a simple scene (right column).

Analogous to DTAM [11], we follow a multi-view stereo approach based on energy minimization that consists of two stages: The first stage involves the calculation of a cost volume that accumulates the photo-consistency errors of the overlapping images for different inverse depths. In the second stage, we minimize an energy functional comprising the cost volume as data term and a regularization term that penalizes deviation from a spatially smooth inverse depth map.

In the following, we explain in more detail how we apply this approach to generate a dense depth map from live images acquired during quadrotor flight.

### 3.1 Cost volume creation and update

Our algorithm starts by creating a cost volume $\mathbf{C} : \mathbb{R}^3 \mapsto \mathbb{R}$ that reflects our belief about the depth of every pixel. The cost volume is always located in front of the first camera pose $T_1$, which we also call the *reference* pose. Every cell $\mathbf{C}(u,v,\xi)$ in this volume indicates the cost (or negative loglikelihood) that the pixel $\mathbf{u} = (u,v)$ in the reference image $I_1$ has an inverse depth of $\xi$. We use a uniform discretization in the inverse depth range $[\xi_{min}, \xi_{max}]$ to assure an uniform sampling of the projected epipolar lines in the reference image space.

For every image $k = 2,\dots,n$ other than the reference image, we now update the cost volume depending on how well the $k$ image $I_k$ matches the reference image $I_1$ at a certain inverse depth. This is done as follows: For every pixel $\mathbf{u}$ in the reference image and each hypothesized inverse depth $\xi_i$ we update the accumulated average cost at cube voxel $\mathbf{C}(\mathbf{u},\xi_i)$ with the photometric error $\rho(I_k,\mathbf{u},\xi_i)$ defined by

$$\rho(I_k,\mathbf{u},\xi_i) = I_1(\mathbf{u}) - I_k(\pi(T_k\pi^{-1}(\mathbf{u},\xi_i)) \tag{1}$$

where $\pi(\mathbf{p})$ describes a perspective projection of a 3D point $\mathbf{p}$ and $\pi^{-1}(\mathbf{u},\xi_i)$ refers to the back-projection of a pixel $\mathbf{u}$ with inverse depth $\xi_i$.

We adopt the camera model of [19], where the intrinsic calibration of the camera is given by the intrinsic parameters $(f_u, f_v, c_u, c_v)$ and a single radial distortion parameter $w$. Since the images captured by the camera are of low resolution (in our case $640\times360$) and are transmitted to the ground-based desktop with lossy compression, we decided to not pre-rectify the images in order to not loose any more valuable information. Instead, we explicitly take the distortion model into account during projection and back-projection. The perspective projection $\mathbf{u} = \pi(\mathbf{p})$ is given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c_u \\ c_v \end{bmatrix} + \begin{bmatrix} f_u & 0 \\ 0 & c_v \end{bmatrix} \frac{r_d}{r} \begin{bmatrix} p_x/p_z \\ p_y/p_z \end{bmatrix} \tag{2}$$

where $r = \sqrt{\frac{p_x^2+p_y^2}{p_z^2}}$ and $r_d = \frac{1}{w}\arctan(2r\tan\frac{w}{2})$. On the contrary, to back-project $\pi^{-1}(\mathbf{u},\xi_i)$ a pixel $\mathbf{u}$ with inverse depth $\xi_i$ we use:

$$\mathbf{p} = \frac{1}{\xi_i}\begin{bmatrix}\frac{r}{r_d}\mathbf{u}_n\\ 1\end{bmatrix} \tag{3}$$

$$\mathbf{u}_n = \begin{bmatrix}\frac{u-c_u}{f_u}\\ \frac{v-c_v}{f_v}\end{bmatrix} \tag{4}$$

where $\mathbf{u}_n$ is a normalized pixel, $r_d = \sqrt{u_n^2+v_n^2}$ and $r = \frac{\tan(wr_d)}{2\tan(w/2)}$.

After all images have been integrated, the average photometric error $\mathbf{C}(\mathbf{u},\xi_i)$ stored in the cost volume for pixel $\mathbf{u}$ at inverse depth $\xi_i$ is

$$\mathbf{C}(\mathbf{u},\xi_i) = \frac{1}{n}\sum_{k=2}^{n}||\rho(I_k,\mathbf{u},\xi_i)||_1. \tag{5}$$

Once the cost volume has been computed, an initial inverse depth map $\xi_r$ can be extracted by searching for the inverse depth associated with the minimum cost for each pixel:

$$\xi_k(\mathbf{u}) = \arg\min_{\xi_i}\mathbf{C}_k(\mathbf{u},\xi_i) \tag{6}$$

This solution is also called the *winner-takes-all* method. It should be noted that this initial inverse depth map will be very noisy, as so far no regularization has been applied. Moreover, it might happen that multiple disparities have equal cost, for example due to surfaces with little texture.

### 3.2 Primal-Dual optimization of the cost volume

In order to find a better inverse depth map, the simple approach of (6) is used as a starting solution of a spatially regularized formulation based on total variation (TV). TV-regularization is known for its ability to preserve edges while smoothing homogeneous regions. The solution $\xi(\mathbf{u})$ is given by the minimization of the energy functional

$$E_\xi = \int_\Omega w(\mathbf{u})||\nabla\xi(\mathbf{u})||_\varepsilon + \lambda\mathbf{C}(\mathbf{u},\xi(\mathbf{u}))d\mathbf{u}, \tag{7}$$

where $\Omega$ is the signal domain (in this case the image dimensions), $w(\mathbf{u})$ is a per pixel weight based on the image gradient that reduces the regularization across image edges, $||\ ||_\varepsilon$ is the Huber norm and $\lambda$ is a parameter used to define the tradeoff between the convex regularizer $w(\mathbf{u})||\nabla\xi(\mathbf{u})||_\varepsilon$ and the non-convex data term $\mathbf{C}_r(\mathbf{u},\xi(\mathbf{u}))$. Since the data term is non-convex the energy functional is approximated by decoupling the data and regularization terms through an intermediate function $\alpha(\mathbf{u})$:

$$\begin{aligned}E_{\xi,\alpha} = \int_\Omega & w(\mathbf{u})||\nabla\xi(\mathbf{u})||_\varepsilon + \frac{1}{2\theta}(\xi(\mathbf{u})-\alpha(\mathbf{u}))^2\\ & +\lambda\mathbf{C}(\mathbf{u},\alpha(\mathbf{u}))d\mathbf{u},\end{aligned} \tag{8}$$

such that $E_{\xi,\alpha} \to E_{\xi}$ as $\theta \to 0$.

The new energy functional allows us to split the minimization into two different problems that are alternatively solved untill convergence:

- First, for a fixed $\alpha(\mathbf{u})$ solve:

$$\min_{\xi} \int_{\Omega} w(\mathbf{u})||\nabla\xi(\mathbf{u})||_{\varepsilon} + \frac{1}{2\theta}(\xi(\mathbf{u}) - \alpha(\mathbf{u}))^2 d\mathbf{u} \tag{9}$$

  which corresponds to the well known Huber-ROF denoising problem that can be solved using a primal-dual algorithm [20]. In this case $\alpha(\mathbf{u})$ represents the noisy image whereas $\xi(\mathbf{u})$ is the searched denoised result.
- Second, for a fixed $\xi(\mathbf{u})$ solve:

$$\min_{\alpha} \int_{\Omega} \frac{1}{2\theta}(\xi(\mathbf{u}) - \alpha(\mathbf{u}))^2 + \lambda \mathbf{C}(\mathbf{u}, \alpha(\mathbf{u}))d\mathbf{u} \tag{10}$$

  .

  This optimization is performed by a point-wise exhaustive search for each $\alpha$ in $\mathbf{C}$.

In practice, the update steps for $\xi(\mathbf{u})$ and $\alpha(\mathbf{u})$ can be performed efficiently and in parallel on modern GPU hardware for each independent pixel $\mathbf{u}$. In addition for the second step we implemented the accelerated and the improved sub-sample accuracy methods recommended in [11]. Finally, we convert the inverse depth map into a depth map $D$ using

$$D(\mathbf{u}) := (\xi(\mathbf{u}))^{-1}. \tag{11}$$

In sum, this algorithm takes a sequence of $n$ input images and camera poses, generates a cost volume, and then finds a regularized inverse depth map $\xi(\mathbf{u})$ with minimum total variation.

## 4 Obstacle Avoidance and Waypoint Selection

When a new depth map $D$ becomes available, we use it to generate the next waypoint in forward direction. Our goal is to select the point where the quadrotor can fly the furthest at the current flying height. To this end, we calculate the most distant point in 3D space reachable by the quadrotor without collisions.

This selection process is illustrated in Fig. 3: The quadrotor at first considers $m$ distant candidate waypoints $\mathbf{p}_1, \ldots, \mathbf{p}_m \in \mathbb{R}^3$ along a horizontal line defined in world coordinates in front of its current position. We convert these points into the frame of the current reference camera $T_1$ and project them into the depth map. Fig. 3 right, shows two different depth maps with an overlay of the projected line. Note that in both cases, the quadrotor was tilted manually while it was recording the reference image to demonstrate that the projected line is in fact horizontal in world coordinates but not necessarily horizontal in the image.

The corresponding pixel coordinates $\mathbf{u}_i$ and depths $d_i$ of these points are

$$\mathbf{u}_i := \pi(\mathbf{p}_i) \tag{12}$$

$$d_i := D(\mathbf{u}_i). \tag{13}$$

Using these virtual depth measurements, we generate a new set of points

$$\mathbf{q}_i := \pi^{-1}(\mathbf{u}_i, d_i^{-1}), \tag{14}$$

which are now located on the estimated surface. The resulting points are visualized from a top-down view in the right column of Fig. 3, and resemble a 2D laser scan.

In order to find the farthest admissible waypoint, we first maximize the minimum convolution given by

$$i^* = \arg \max_{i=1,\ldots,m} \min_{j \in F_i} D(\mathbf{u}_j), \tag{15}$$

where $F_i \subset \{1,\ldots,m\}$ refers to the footprint of the quadrotor on its way to point $\mathbf{p}_i$. We define the footprint of the quadrotor as all 3D points that the quadrotor would touch on its way from its current position to the respective point, under consideration of the quadrotor's shape. Subsequently, we determine the maximum distance the quadrotor can safely move forward as

$$d^* = \min_{j \in F_{i^*}} D(\mathbf{u}_j). \tag{16}$$

To generate the actual waypoint, we subtract a small safety boundary $b$ (e.g., 1 m or 1.5 m) from this depth to prevent the quadrotor from crashing into the next obstacle. The selected waypoint thus becomes

$$\mathbf{p}^* := \pi^{-1}(\mathbf{u}_{i^*}, (d^* - b)^{-1}). \tag{17}$$

While it is in principle possible to evaluate the footprint for arbitrary trajectories, we found that the actual flying behavior of the quadrotor along two axes was somewhat undeterministic. Therefore, we decided to split the motion into two orthogonal segments (which we correctly consider during footprint generation): In the first segment, the quadrotor positions itself at the desired $X$ position (left/right, see Fig. 5). In the second part, it then approaches the waypoint by a forward motion along the $Y$ axis. Note that we keep the flying height fixed ($Z$ axis) in all of our experiments, although it would be straight-forward to use it as a third degree-of-freedom during obstacle avoidance. Two examples of the selected waypoints as well as the corresponding footprint are visualized with a yellow cross and red dots, respectively, in the right column of Fig. 3.

To summarize this section, our waypoint generation method takes a dense depth map $D$ as input and outputs a waypoint $\mathbf{p}^*$ that is expected to lead to the largest forward increment without collisions.

## 5 Experiments and Results

For all our experiments, we used a low-cost Parrot AR.Drone 2 quadrotor. It is equipped with an inertial measurement unit (IMU) consisting of a 3-axis gyroscopes and accelerometers operating at 200 Hz, an ultrasound altimeter with an update rate of 25 Hz and a frontal monocular camera covering a field of view of $73.5^\circ \times 58.5^\circ$, providing an image resolution of $640 \times 360$. The quadrotor communicates with a ground-based

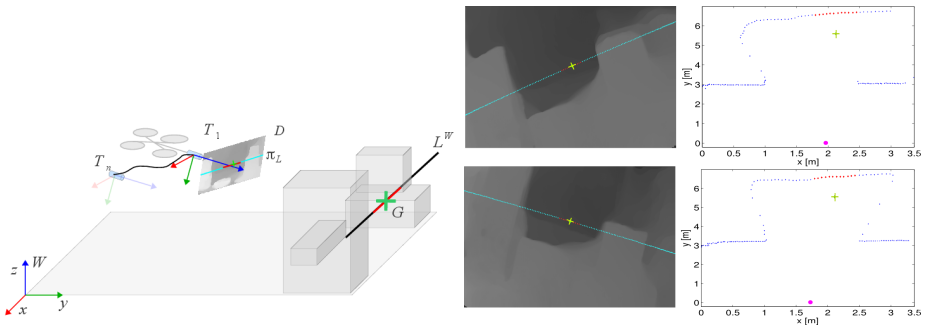


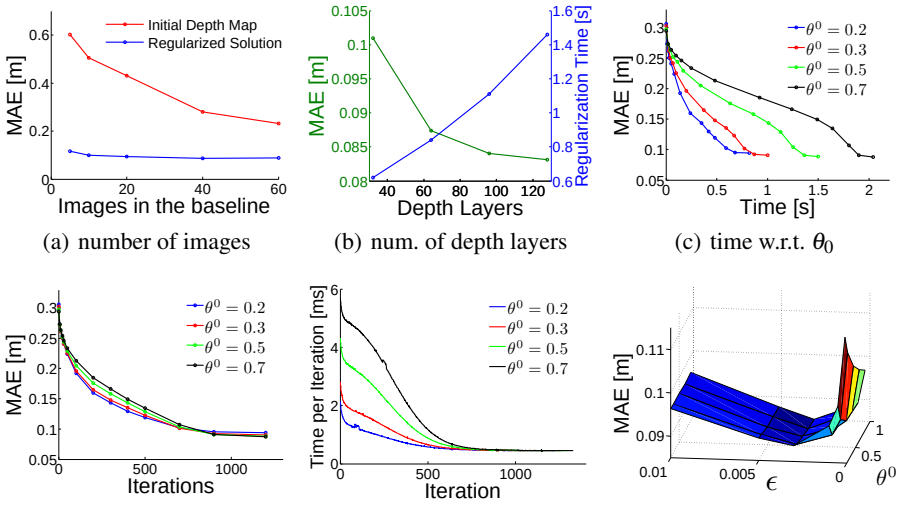

Fig. 3: Illustration of our approach and the involved coordinate systems: The quadrotor hovers in front of several obstacles, while it takes a series of images at camera poses $T_1, \ldots, T_n$. From these images, it computes a depth map $D$ and selects the next waypoint $G$. The quadrotor at first considers $m$ distant candidate waypoints along a horizontal line $L^W$ defined in world coordinates in front of its current position. We convert these points into the frame of the current reference camera, $\mathbf{u}_i := \pi(\mathbf{p}_i)$ and project them into the depth map to find the furthest 3D point. The left column shows two different depth maps with the projected horizontal lines (cyan dots), footprint points (red), and the selected waypoint as a yelow cross. We also show a top view of the same scene with the resemble 2D scan.

desktop over wireless LAN to perform the calculations of the all time demanding tasks. The video of the frontal camera is streamed in real-time to the desktop at 30 fps using a lossy compression to reduce bandwidth. The open-source implementation of [9] (tum_ardrone [4]) for quadrotor control and our path planning module run on the Intel Core i3-2120 CPU $\times 4$ at 3.36 GHz, while the dense depth map estimation is executed on a NVIDIA GeForce GTX 560 TI graphic card with 384 CUDA cores and 1 GB of device memory. The tum_ardrone package continuously estimates the absolute scale of the visual map. As a consequence, the computed depth maps provide absolute distances, which is very useful for our purpose.

### 5.1 Dense Depth Map evaluation

We first evaluated the performance of our depth map estimation approach using a synthetic dataset [21] which provides perfect camera poses and ground truth depth maps. This allowed us to analyze the impact on the accuracy with respect to the number of images integrated in the cost volume and to determine the required number of discrete depth layers. We calculated the Mean Absolute Error (MAE) on both the initial depth map and the final solution with respect to the ground truth. To the best of our knowledge, we are the first to provide such a quantitative evaluation of dense mapping approaches. Based on our findings, we then selected the parameters for the problem defined in Eq. 8 in terms of accuracy, computation time and convergence. The results are given in Fig. 4.

[4] http//wiki.ros.org/tum_ardrone
We would like to thank Jakob Engel for his support and advice on using the tum_ardrone package.




(a) number of images (b) num. of depth layers (c) time w.r.t. $\theta_0$

(d) iterations w.r.t. $\theta_0$ (e) time per iteration (f) $\varepsilon$ vs. $\theta_0$

Fig. 4: Performance evaluation for the dense depth map estimation approach. (a) Error w.r.t. the number of images. Increasing the number of images within the same baseline significantly improves the accuracy of the initial depth map (red curve). In contrast, the quality of the regularized solution barely changes (blue curve). We selected 20 images to achieve the lower error. (b) Error (green) and computation time (blue) w.r.t. the number of layers of the cost volume. Increasing the number of depth layers improves the quality, but at the expense of increasing running time. A good compromise is achieved for 64 depth layers. (c) Error w.r.t. elapsed regularization time for different values of $\theta^0$. Notice that for large values of $\theta^0$, the regularization time increases substantially. A value of $\theta^0 = 0.2$ yields the least time for the same number of iterations without impairing precision. (d) Error w.r.t. the number of iterations. In all cases approx. 900 iterations are required to achieve convergence. (e) Time per iteration. Notice that the time per iteration decreases until convergence as the search interval–induced by the Quadratic Penalty term, decreases. (f) Error w.r.t. $\varepsilon$ and $\theta^0$. A valley is yielded for $\varepsilon = 0.003$, and it is independent of $\theta^0$.

In the final experiments with the quadrotor we consider a depth range from 0.5 m to 7 m, which we sampled into a cost volume of 64 inverse depth layers requiring approximately 84 MB ($640 \times 360 \times 64 \times 6$ bytes). In all experiments, we set $\theta = 0.2$, $\lambda = 0.9$ and 900 primal-dual iterations. Note that this process is carried out on the GPU and hence does not affect visual tracking and position control of the quadrotor. As expected, the energy optimization is the most expensive calculation, requiring approximately 500 ms in total to obtain the final depth map. However, integrating a single image into the cost volume requires only around 5 ms per frame. This means depth maps can be generated at 1.5 Hz.

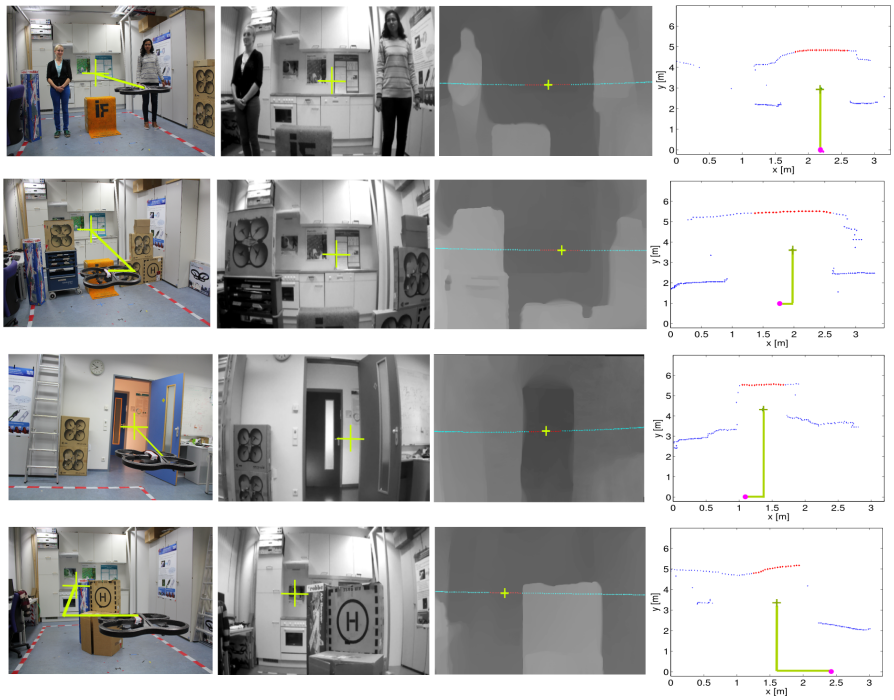



Fig. 5: Evaluation of the collision avoidance approach on four different environment configurations. From top to bottom: people, boxes on the outside, door, boxes on the inside. From left to right: External view, quadrotor view, depth map, top view of range scan and generated trajectory (yellow lines, added manually for visualization purposes).

### 5.2 Collision Avoidance Evaluation

To evaluate the reliability of our approach, we set up increasingly challenging environments through which the quadrotor had to navigate. The goal of our experiments was to measure the number of system successes and failures to pass the obstacle. Fig. 5 shows the four different environments in which we tested our approach. In the "person" environment, two persons were standing approximately 1.5 m away from each other. In the "boxes on the outside" environment, we set up various boxes and a tool carriage in such a way that the quadrotor had to pass through the middle. In the "door" environment, the quadrotor had to navigate through an open door with a clearance of 90 cm. For the "boxes on the inside" environment, we put a stack of boxes in the middle of the room that the quadrotor had to circumvent, that is, either to the left or to the right.

In each environment, we carried out 10 trials with our collision avoidance approach. In each trial, the quadrotor was hovering for a couple of seconds in front of an obstacle, acquiring 30 images, computing the depth map, generating the waypoint and executing it. The overall time between starting the image capture and the generation of the next way point was less than 1.5 seconds. We deemed a trial successful when the quadrotor reached its desired waypoint without collision. Any other case, i.e., a failure to compute

| Setting | Successes | Failures |
|---|---|---|
| People | 10 | 0 |
| Boxes on the outside | 9 | 1 |
| Boxes in the middle | 9 | 1 |
| Door | 5 | 5 |
| Sum | 33 | 7 |
| Overall performance | 82.5% | 17.5% |

Table 1: Performance evaluation over 10 runs in each of the four experiments.

the depth map, to find a suitable waypoint, or a collision with the environment was counted as a failure. After every trial, the quadrotor was returned manually to its initial position, before it selected another random starting position for the next trial.

The results are summarized in Table 1. Over 40 trials, we achieved an overall success rate of 82.5%, where the quadrotor reached the desired target position without collision. In 17.5% of all trials, it failed. Most failures occurred in the "door" environment, where the quadrotor had to navigate through a very narrow space.

We consider our evaluation of the parameters a valuable contribution for future work on dense mapping approaches. We also demonstrated that obstacle avoidance and collision-free path planning is feasible from monocular images, which is in particular useful for upcoming nano-quadrotors with minimal payload. Interestingly, and opposed to our earlier assumption, we found that no additional quadrotor motion is required during image acquisition: Small movements of the quadrotor during hovering generate sufficient baselines leading to satisfactory depth maps.

## 6 Conclusion

In this paper, we presented a novel approach to obstacle avoidance for quadrotors that have only a single monocular camera. From a small set of consecutive images, we compute a regularized depth map that we subsequently use for waypoint generation. We demonstrated the applicability of our approach in four challenging environments using a Parrot Ardrone quadrotor. As our approach only requires an IMU and a monocular camera, it can be applied easily to many other lightweight aerial vehicles.

It would be interesting to augment our approach to full autonomous exploration, and to adapt it to CPU so that it becomes applicable to on-board computing. Furthermore, we believe that a collision avoidance system would clearly benefit from semantic scene understanding approaches (e.g., to recognize a door as a door), which we plan to look into in the near future.

## 7 Acknowledgment

The authors would like to thank Pedro Piniés for the so many fruitful discussions about convex optimisation and variational methods. His insights about saddle point methods

acquired during his stay in Graz University of Technology was crucial to understand and implement the core of the energy minimisation with TV regularisation.

## References

1. Bachrach, A., de Winter, A., He, R., Hemann, G., Prentice, S., Roy, N.: RANGE - robust autonomous navigation in GPS-denied environments. In: ICRA. (2010)
2. Grzonka, S., Grisetti, G., Burgard, W.: A fully autonomous indoor quadrotor. IEEE Transactions on Robotics (T-RO) **8**(1) (2012) 90–100
3. Fraundorfer, F., Heng, L., Honegger, D., Lee, G., Meier, L., Tanskanen, P., Pollefeys, M.: Vision-based autonomous mapping and exploration using a quadrotor MAV. In: IROS. (2012)
4. Wagter, C.D., Tijmons, S., Remes, B., de Croon, G.: Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In: ICRA, Hong Kong, China (Accepted 2014)
5. Huang, A.S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., Roy, N.: Visual odometry and mapping for autonomous flight using an RGB-D camera. In: Int. Symposium on Robotics Research (ISRR). (2011)
6. Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I.L., Ruess, F., Suppa, M., Burschka, D.: Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. Robotics & Automation Magazine, IEEE **19**(3) (2012) 46–56
7. Nieuwenhuisen, M., Droeschel, D., Holz, D., , Laebe, T., Behnke, S.: Multimodal obstacle detection and collision avoidance for micro aerial vehicles. In: ECMR. (2013)
8. Klein, G., Murray, D.: Parallel Tracking and Mapping for Small AR Workspaces. In: Proc. of the IEEE Intl. Symposium on Mixed and Augmented Reality (ISMAR). (2007)
9. Engel, J., Sturm, J., Cremers, D.: Camera-based navigation of a low-cost quadrocopter. In: IROS. (2012)
10. Hirschmüller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In: CVPR. (2005)
11. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: Dense tracking and mapping in real-time. In: Proc. of the Intl. Conf. on Computer Vision ICCV. (2011)
12. Green, W., Oh, P.: Optic-flow-based collision avoidance. Robotics Automation Magazine, IEEE **15**(1) (2008) 96–103
13. Mori, T., Scherer, S.: First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In: ICRA. (2013)
14. Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A.D., Hebert, M.: Learning monocular reactive uav control in cluttered natural environments. In: ICRA. (2013)
15. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. Journal of Intelligent and Robotic Systems **57**(1-4) (2010) 65–100
16. Kendoul, F.: Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. Journal of Field Robotics **29**(2) (2012)
17. Hrabar, S., Sukhatme, G., Corke, P., Usher, K., Roberts, J.: Combined optic-flow and stereo-based navigation of urban canyons for a uav. In: IROS. (2005)
18. Graber, G., Pock, T., Bischof, H.: Online 3d reconstruction using convex optimization. In: ICCV LDRMC. (2011)

19. Devernay, F., Faugeras, O.: Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments. Mach. Vision Appl. **13**(1) (2001) 14–24
20. Chambolle, A., Pock, T.: A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. Mathematical Imaging and Vision **40**(1) (2011) 120–145
21. Handa, A., Newcombe, R.A., Angeli, A., Davison, A.J.: Real-time camera tracking: when is high frame-rate best? In: ECCV. (2012)