# Autonomous Approach and Landing for a Low-Cost Quadrotor Using Monocular Cameras

Sergiu Dotenco[1], Florian Gallwitz[1], and Elli Angelopoulou[2]

[1] Nuremberg Institute of Technology, Germany
Department of Computer Science
[2] Friedrich-Alexander-University Erlangen-Nürnberg, Germany
Pattern Recognition Lab

**Abstract** In this paper, we propose a monocular vision system for approach and landing using a low-cost micro aerial vehicle (MAV). The system enables an off-the-shelf Parrot AR.Drone 2.0 quadrotor MAV to autonomously detect a landpad, approach it, and land on it. Particularly, we exploit geometric properties of a circular landpad marker in order to estimate the exact flight distance between the quadrotor and the landing spot. We then employ monocular simultaneous localization and mapping (SLAM) to fly towards the landpad while accurately following a trajectory. Notably, our system does not require the landpad to be located directly underneath the MAV.

**Keywords:** Approach and landing, ellipse detection, conic sections, pose estimation, MAV, PTAM, SLAM.

## 1 Introduction

Approach and landing is one of the most fundamental maneuvers performed by aerial vehicles. The maneuver requires a considerable amount of precision in order to avoid any damage to the vehicle and its surroundings while the maneuver is executed. This is especially important in case an unmanned aerial vehicle (UAV) has to autonomously navigate in unknown, possibly GPS-denied environments. In this paper, we focus on this specific scenario.

In our method, we detect the landpad using a forward-facing camera mounted on the MAV. We use a heliport-like landpad marker of known size (see fig. 1) that is typically used to label helicopter landing sites. We estimate the distance between the quadrotor and the landpad, and employ a monocular SLAM framework to let the quadrotor follow a trajectory in an accurate manner. Since the monocular SLAM framework is subject to drift, we have to recover from possible trajectory deviations by centering the quadrotor over the landpad once the MAV has reached the landing spot. In this step we use the quadrotor's downward-facing camera. Once the quadrotor has stabilized its position, it performs the landing. We provide a video showing our system in action at http://youtu.be/Og3VZX2jE0A.

In contrast to existing work, we do not assume the landpad to be located directly underneath the quadrotor. Instead, the quadrotor has to locate it using

**Figure 1.** In our method, we detect the circular landpad marker using the forward-facing camera, estimate the distance, and fly towards landing spot by employing monocular SLAM. Once the quadrotor has reached the target position, it tries to detect the landpad using the downward-facing camera, stabilize its position, and finally land.

its forward-facing camera, fly towards it, and finally land on the marker. At the same time, we use a quadrotor platform with a fairly constrained camera setup rather than with cameras that are strategically positioned.

Our goals are, first of all, to demonstrate that autonomy can be achieved using constrained, inexpensive, off-the-shelf hardware. Also, we want to show that a (non-stereo) two-camera-setup can have an advantage over a setup where a single, downward-facing camera is used, since the two-camera-setup effectively extends the visibility range of the MAV. This allows the quadrotor to better perceive and eventually explore the environment [1].

As an MAV platform, we use Parrot AR.Drone 2.0, a low-cost quadrotor with a simple IMU and two monocular cameras. The quadrotor features 1 GHz ARM Cortex-A8 processor, 32-bit 800 MHz DSP, and 1 GB of DDR2 RAM 200 MHz. Computations, however, cannot be performed directly on-board. Instead, the quadrotor has to communicate with a ground station that receives sensory data, performs the computations, and generates steering commands. The steering commands are eventually sent back to the quadrotor through wireless LAN.

The MAV platform we use poses several challenges. Due to the very narrow vertical field of view of the forward-facing camera, the landpad has to be detected from an extremely oblique perspective. A blind spot between both cameras as shown in fig. 2 makes it impossible to track the position of the landpad while the quadrotor is flying towards it. There are also significant delays in the communication between the quadrotor and the ground station. Finally, the downward-facing camera of the quadrotor provides only low resolution images, and also features a very narrow field of view.

The remainder of the paper is organized as follows. In section 2 we present the related work. Section 3 provides an overview of our hardware platform. Section 4

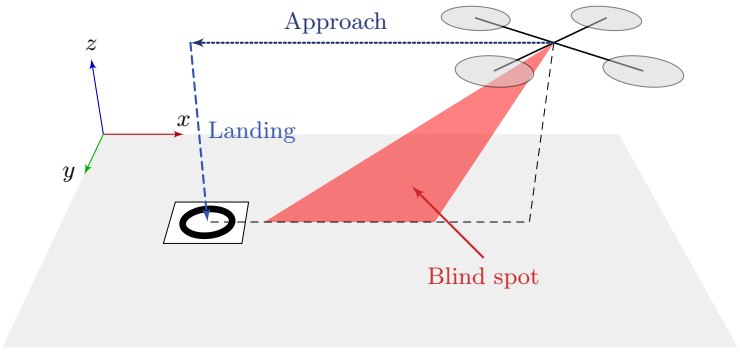



**Figure 2.** Our experimental setup. The quadrotor has to locate the circular landpad using the forward-facing camera, estimate the distance, approach the landpad, and after recovering from possible drift using the downward-facing camera finally land on the marker.

describes our approach, and in section 5 we discuss the experimental results. Finally, the conclusions are given in section 6.

## 2 Related Work

Most of the related work on monocular vision-based landing focuses on setups that use only a downward-facing camera for landing purposes. Lange et al. [2], for instance, presented a vision-based approach for autonomous landing and position control. In their work, they estimate the 3-dimensional position of the MAV relative to the landing pad. The velocity and position is stabilized using a dedicated optical flow sensor. As landing marker, they use a custom designed pattern consisting of four concentric white rings on black background. Each of the white rings has a unique ratio of its inner to outer border radii allowing fast detection. Lange et al. use the estimated center of the landpad to stabilize the quadrotor. Compared to our approach, they do not account for perspective projection, which shifts the center of the projected rings unevenly, and can possibly reduce the stabilization accuracy.

Eberli et al. [3] presented a similar approach. They used a downward-facing monocular camera mounted on the quadrotor to detect a marker consisting of two concentric circles. The detected marker is subsequently used for five degrees of freedom pose estimation and MAV set-point control. By using concentric circles it is possible to disambiguate between the inherently multiple camera poses determined from the projection of the circles. To stabilize the quadrotor, Eberli et al. use a linear-quadratic-Gaussian control with loop transfer recovery (LQG/LTR).

A fuzzy visual servoing approach for controlling an MAV was presented by Wendel et al. [4], which allows to perform takeoff, hovering, and landing without a marker. In their method, the forward-facing camera is used to obtain visual pose

estimates using Parallel Tracking and Mapping (PTAM) based SLAM. In order to obtain the correct map scale, Wendel et al. use an ARToolKitPlus marker, which is placed into the scene during initialization.

Yang et al. [5] introduced an on-board monocular vision system for autonomous takeoff, hovering, and landing of an MAV. They estimate a five degrees of freedom pose from an elliptical projection of the circular pattern which encloses an "H". The remaining geometric ambiguity is resolved using the gravity vector estimated by an inertial measurement unit (IMU). To detect the landpad, Yang et al. find and classify connected components in binarized images using artificial neural networks. To train the network, they use a labeled data set containing 7000 samples of background clutter, and 3000 samples each of both parts of the landing pad corresponding to the circle and the letter "H", respectively. In contrast to our method, Yang et al. do not approach the landpad autonomously, but rather assume that it already can be seen using the downward pointing camera mounted on the quadrotor.

In a more recent work, Yang et al. [6] extended their system to actively search for, and land on an arbitrary landing site specified by a single reference image of unknown size. For autonomous navigation, they employ PTAM based monocular SLAM. A multi-scale ORB feature based detector is integrated into the SLAM framework for landing site detection. As opposed to our method, the navigation trajectory that determines the search path is predefined. To automatically initialize PTAM, they estimate the camera pose using a circular landing pad during the takeoff phase.

## 3 Hardware Platform

In our experimental setup, we use a stock Parrot AR.Drone 2.0 quadrotor MAV. Equipped with the indoor hull, the quadrotor weighs 420 g, and 380 g with the outdoor hull. With the indoor hull, the quadrotor measures 517 mm × 517 mm, and with the outdoor hull 451 mm × 451 mm.

### 3.1 Sensors

The quadrotor has two cameras, a 3-axis gyroscope, a 3-axis accelerometer, a 3-axis magnetometer, a sonar and a pressure altimeter. The MAV sends gyroscope measurements to the ground station at a rate of 200 Hz, and sonar measurements at 25 Hz.

The two cameras on the quadrotor are mounted perpendicularly to each other. One camera is pointing in the flight direction, and the other downwards. The forward-facing camera captures 30 frames per second with a resolution of 1280 × 720 pixels. The diagonal field of view of the forward-facing camera covers 92°. The downward-facing camera has a diagonal field of view of 64°, and captures video frames with a resolution of 320 × 240 pixels at a rate of 60 frames per second. The MAV on-board software uses the downward-facing camera mainly for velocity estimation, which however works reliably only if the quadrotor is

flying above highly textured ground. The video streams of both cameras are compressed using H.264 baseline profile before they are sent to the ground station over wireless LAN. Due to a hardware limitation, the streams cannot be accessed simultaneously.

### 3.2 Control

The Parrot AR.Drone 2.0 on-board software uses the sensors to control the roll $\Phi$, pitch $\Theta$, the yaw rotational velocity $\dot{\Psi}$, and the vertical velocity $\dot{z}$ of the quadrotor according to a reference value [7]. The reference is set by sending a new control command $\boldsymbol{u} = (\tilde{\Phi}, \tilde{\Theta}, \tilde{\dot{\Psi}}, \tilde{\dot{z}}) \in [-1, 1]^4$ at least every 30 ms.

## 4 Our Method

In the proposed method, we first detect the landpad using the quadrotor's forward-facing camera. This is a particularly challenging task, since the landpad is seen from an extremely oblique view angle. Then, we estimate the distance between the mounting position of the quadrotor's downward-facing camera and the center of the landpad. To accurately approach the landpad, we use a monocular simultaneous localization and mapping framework based on Parallel Tracking and Mapping (PTAM) [7]. Since the monocular SLAM framework can introduce drift over time, we have to recover from possible trajectory deviations once the quadrotor reached the landing spot. For this purpose, we switch to the downward-facing camera, and assume the landpad to be now approximately underneath the quadrotor. We then detect the landpad again, and center the quadrotor over the landpad by estimating the landpad pose. A PID controller is used for each of the axes to stabilize the quadrotor. The estimated landpad pose is used as reference.

### 4.1 Landpad Detection

In our setup, we use a heliport-like landpad marker. The landpad consists of the letter "H" enclosed by a circle. Under perspective projection, a circle is typically observed as an ellipse on the image plane. To detect the landpad, we first rectify the image using the camera intrinsic parameters in order to eliminate lens distortions, and correct the image aspect ratio such that the intrinsic parameters $\alpha_x, \alpha_y$ that represent the focal length $f$ in pixels along the $x$ and $y$ axes match. Then, we perform ellipse detection.

**Ellipse Detection** To detect an ellipse, we modified the edge-grouping-based method of Nguyen et al. [8]. Their approach estimates ellipse parameters at curve level instead of inspecting individual edge points, which reduces the overall execution time. The algorithm consists of the following steps: (1) edge detection, (2) contour extraction, (3) line segmentation, (4) curve segmentation, (5) and curve grouping.

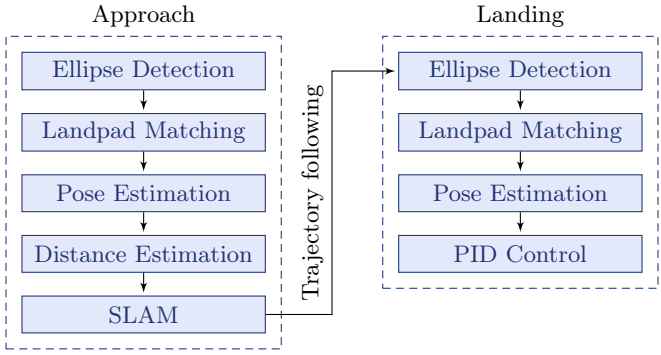



**Figure 3.** The pipeline of our approach

In the first step, we detect the edges in the image, and link the edge points into sequential lists. Edge contours with a length $\ell$ below 10 pixels are discarded. In order to reduce the number of edge points, the contours are segmented to lines. Instead of applying the line segmentation algorithm suggested by Nguyen et al., we use the Douglas-Peucker algorithm [9] with an approximation accuracy of $\epsilon = 1.5$, which produces more accurate curves at the expense of slightly higher computation time.

In a fairly cluttered scene, the extracted edge contours do not necessarily belong to a single ellipse. Rather, they could have resulted from several shapes merged together. Thus, the contours have to be partitioned into different curve segments, allowing curves that belong to a particular ellipse to be subsequently grouped. Under general observations, a curve belonging to an ellipse has to meet the following conditions. If one of the conditions is not met, the curve is segmented.

**Curvature** A sequence of connected line segments traces an arc segment. The curvatures of these arcs should of the same sign throughout the sequence.

**Length** The difference in the lengths of two neighboring line segments is small.

**Angle** The difference between the gradients of tangents belonging to neighboring line segments is small.

Curve segmentation results in several arc segments. Arc segments that are not directly connected, but may belong to the same ellipse are now grouped. Two curves are grouped only if the end points are close to each other, and the difference between the gradient of the end point tangents is small.

Finally, we estimate ellipse parameters from the resulting set of curve points using a non-iterative least squares based geometric ellipse fitting [10] instead of the direct least squares based on an algebraic cost function as proposed by Nguyen et al. Only ellipses with a residual $\rho$ below 0.075 are kept.

**Landpad Matching** Once an ellipse is detected, we check whether it contains the letter “H”. This is accomplished by first rectifying the ellipse. To achieve this,

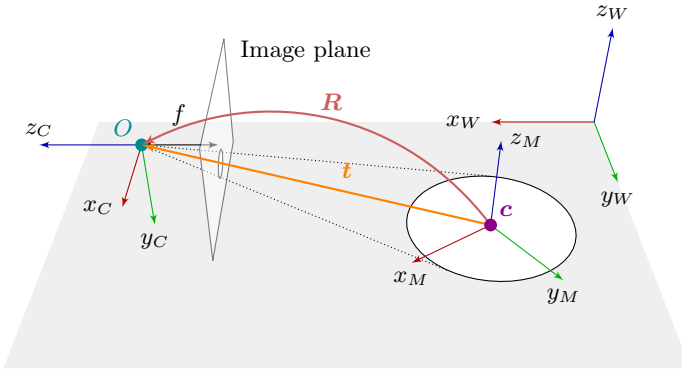



**Figure 4.** The geometry of the scene. Three coordinate systems involved: a right-handed camera coordinate system (CCS), marker coordinate system (MCS), and world coordinate system (WCS). $O$ denotes the center of projection, and $z = -f$ identifies the location of the corresponding image plane, $f$ being the focal length. $\boldsymbol{R}$ denotes the rotation matrix that produces an oblique viewpoint, and $\boldsymbol{t}$ is the translation vector from the center of the circle $\boldsymbol{c}$ to the center of projection. The dotted lines contour the oblique elliptical cone.

we first estimate the planar homography up to scale between each ellipse and the landpad reference by sampling four points on the ellipse and the circle in the reference image. Then, we rotate the rectified ellipse with respect to the angle of most dominant lines that were detected in the marker. This allows one to normalize the rotation of the letter "H". We use a voting scheme to determine the angle of most dominant lines, which should ideally correspond to the lines belonging to the vertical bars of the "H". For this, let $(b_i)_{i=1,\ldots,n}$, be the bins of a histogram, where $b_i \subset \{\beta \in \mathbb{R} \mid 0 \leq \beta < \pi\}$, and $n = 18$, i.e., we use a bin width of 10°. Let $0 \leq \alpha_j < \pi$ be the angles of the lines detected in the now circular image region. Each line angle $\alpha_j$ falls into a bin $b_i$. We select the bin with the most values $b_k$ as the target angle candidate bin. The final rotation angle $\alpha$ is calculated as arithmetic mean $\alpha = \frac{1}{|b_k|} \sum_{a_j \in b_k} a_j$ over all the line angles in the candidate bin.

Finally, we apply Otsu's global thresholding [11] method to the circular image region, and use normalized cross-correlation as a match score between the landpad candidate and the reference image. The angle $\alpha$ is used as the yaw angle for pose estimation.

**Pose Estimation** In order to estimate the landpad pose, we closely follow the method by Chen et al. [12]. In general, an ellipse is a type of conic section, i.e., a curve generated from an intersection of a cone with a plane. Considering a right-handed camera coordinate system with center of projection as origin and the optical axis as the $z$-axis, a conic section in the image domain can be described

by the implicit second-order polynomial

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0, \tag{1}$$

which can be defined in matrix form as

$$\boldsymbol{x}^\top \boldsymbol{C} \boldsymbol{x} = 0, \tag{2}$$

where $\boldsymbol{x} = \begin{bmatrix} x & y & 1 \end{bmatrix}^\top$. $\boldsymbol{C}$ is a nonzero real symmetric matrix given by

$$\boldsymbol{C} = \begin{bmatrix} A & B & D \\ B & C & E \\ D & E & F \end{bmatrix} \tag{3}$$

A bundle of straight lines passing through the center of projection and the ellipse defines an oblique elliptical cone. Assuming the focal length of the camera to be $f$, the corresponding image plane will be located at $z = -f$. The oblique elliptical cone can then be described by the points

$$\boldsymbol{p} = k \begin{bmatrix} x & y & -f \end{bmatrix}^\top, \tag{4}$$

where $k$ is a scale factor that defines the distance from the center of projection to $\boldsymbol{p}$. From eqs. (3) and (4) follows

$$\boldsymbol{p}^\top \boldsymbol{Q} \boldsymbol{p} = 0, \tag{5}$$

where

$$\boldsymbol{Q} = \begin{bmatrix} A & B & -\frac{D}{f} \\ B & C & -\frac{E}{f} \\ -\frac{D}{f} & -\frac{E}{f} & \frac{F}{f^2} \end{bmatrix} \tag{6}$$

Here, similar to Kanatani et al. [13], we adapt the scale $\det \boldsymbol{Q} = -1$, where $\det \boldsymbol{Q} \neq 0$, in order to eliminate the scale indeterminacy of the cone.

*Determining the 5 Degrees of Freedom* Due to the cone's rotation invariance around the supporting plane normal, a circle pose can be determined only up to five degrees of freedom.

To determine the circle pose, let $\lambda_1, \lambda_2, \lambda_3$ be the ordered eigenvalues of $\boldsymbol{Q}$, and $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ its normalized eigenvectors. Then, $\boldsymbol{Q}$ can be expressed as

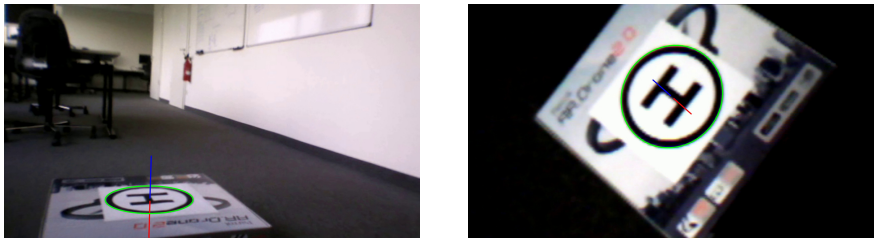
$$\boldsymbol{Q} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\top, \tag{7}$$

where

$$\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \lambda_3), \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{bmatrix} \tag{8}$$

Without loss of generality, we assume that

$$\lambda_1 \lambda_2 > 0, \quad \lambda_1 \lambda_3 < 0, \quad |\lambda_1| \geq |\lambda_2|, \tag{9}$$



(a) (b)

**Figure 5.** Landpad detection results using the forward-facing camera (a) and the downward-facing camera (b) of the quadrotor with superimposed normals to the supporting plane of the landpad.

i.e., the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ are given in descending order, and $\boldsymbol{Q}$ has a signature of $(2, 1)$.

Now let $r$ be the radius of the circle. The unit normal vector $\boldsymbol{n}$ of the circle plane described in camera coordinate system can be computed as

$$\boldsymbol{n} = \boldsymbol{V} \begin{bmatrix} S_2 \sqrt{\frac{\lambda_1 - \lambda_2}{\lambda_1 - \lambda_3}} \\ 0 \\ -S_1 \sqrt{\frac{\lambda_2 - \lambda_3}{\lambda_1 - \lambda_3}} \end{bmatrix}. \tag{10}$$

The center of the circle in the camera coordinate system is given by

$$\boldsymbol{c} = z_0 \boldsymbol{V} \begin{bmatrix} S_2 \frac{\lambda_3}{\lambda_2} \sqrt{\frac{\lambda_1 - \lambda_2}{\lambda_1 - \lambda_3}} \\ 0 \\ -S_1 \frac{\lambda_1}{\lambda_2} \sqrt{\frac{\lambda_2 - \lambda_3}{\lambda_1 - \lambda_3}} \end{bmatrix}, \tag{11}$$

where $S_1, S_2, S_3 \in \{-1, 1\}$ are undetermined signs, and

$$z_0 = S_3 \frac{\lambda_3 r}{\sqrt{-\lambda_1 \lambda_3}} \tag{12}$$

is the distance between the origin and the circle plane. This results in eight possible solutions for $\boldsymbol{n}$ and $\boldsymbol{c}$. Since the normal to the supporting plane has two sides, we let it face the camera

$$\boldsymbol{n} \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top > 0, \tag{13}$$

and require the center of the circle to be in front of the camera

$$\boldsymbol{c} \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top < 0 \tag{14}$$

The visibility constraints from eqs. (13) and (14) allow us to determine two out of the three unknown signs.

Following eqs. (13) and (14), we end up with two sets of solutions: $(\boldsymbol{n}_i, \boldsymbol{c}_i)_{i=1,2}$. The preliminary pose estimates $(\boldsymbol{R}_i, \boldsymbol{t}_i)_{i=1,2}$ can then be computed as

$$\boldsymbol{R}_i(\alpha) = \begin{bmatrix} g\cos\alpha & S_1 g \sin\alpha & S_2 h \\ \sin\alpha & -S_1\cos\alpha & 0 \\ S_1 S_2 h\cos\alpha & S_2 h \sin\alpha & -S_1 g \end{bmatrix}, \tag{15}$$

and

$$\boldsymbol{t}_i(\alpha) = \begin{bmatrix} -S_2 S_3 \sqrt{\dfrac{(\lambda_1-\lambda_2)(\lambda_2-\lambda_3)}{-\lambda_1\lambda_3}} r\cos\alpha \\ -S_1 S_2 S_3 \sqrt{\dfrac{(\lambda_1-\lambda_2)(\lambda_2-\lambda_3)}{-\lambda_1\lambda_3}} r\sin\alpha \\ z_0 \end{bmatrix}, \tag{16}$$

where $\alpha$ is the yaw angle, $z_0$ as in eq. (12), and $g, h$ are given by

$$g = \sqrt{\frac{\lambda_2-\lambda_3}{\lambda_1-\lambda_3}}, \quad h = \sqrt{\frac{\lambda_1-\lambda_2}{\lambda_1-\lambda_3}}. \tag{17}$$

$\boldsymbol{t}_i$ is the translation from the center of the circle to the optical center, while $\boldsymbol{R}_i$ is a rotation matrix that produces an oblique viewpoint. For exact details on these derivations refer to [12,13].

*Resolving the Geometric Ambiguity* At this point we are left with two possible solutions for the pose estimates $(\boldsymbol{R}_i, \boldsymbol{t}_i)_{i=1,2}$, and thus have to determine which one is correct. Here, we assume the pinhole camera model and the camera calibration matrix $\boldsymbol{K}$ to be known. We now consider two separate cases in order to disambiguate the landpad pose.

In the first case, the landpad is observed using the forward-facing camera. The camera's optical axis is located perpendicularly to the landpad's normal, which is assumed to be lying on the ground and in front of the camera. Therefore, the corresponding pitch angle $\Theta$, which can be computed by decomposing the rotation matrix $\boldsymbol{R}_i$, cannot exceed 90°. The solution is thus *unique*.

In the second case, the landpad is observed using the downward-facing camera. Here, we follow the approach by Yang et al. [6] and compare the normals $\boldsymbol{n}_i$ with the gravity vector $\boldsymbol{g}$, which is estimated using the IMU. The normal with the smallest angle with respect to $\boldsymbol{g}$ is used as the final pose estimate.

### 4.2 Distance Estimation

After detecting the landpad, we estimate the horizontal distance $d_t$ between the mounting position of the downward-facing camera and the projection of the center of the landpad. Given $\theta$ is the pitch angle of the landpad's supporting plane (which can be determined from the rotation matrix $\boldsymbol{R}_i$) with respect to the forward-facing camera, an estimate for the horizontal distance $d_t$, i.e., the distance along the optical axis as shown in fig. 6 can be computed as

$$d_t = \|\boldsymbol{t}\|_2 \cos\theta + s, \tag{18}$$

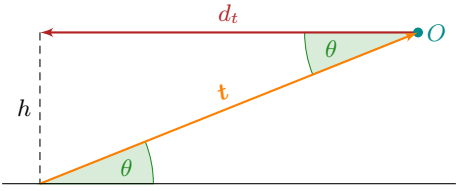



**Figure 6.** Under the assumption that the quadrotor is flying parallel to the ground plane, the travel distance $d_t$ can be computed using landpad's pitch angle $\theta$ and the translation $\boldsymbol{t}$.

where $s$ is the displacement between the mounting position of the downward-facing and the forward-facing camera. We used a displacement value of $s = 25.7\,\text{cm}$, which was determined by measuring the distance between the tip of the forward-facing camera and the center of the downward-facing camera. Accounting for the displacement $s$ allows to position the quadrotor (once it has reached the landpad) in such a way that the landpad is in the field of view of the downward-facing camera.

Estimating the distance according to eq. (18) requires the quadrotor to be flying parallel to the ground. To ensure this, we account for the quadrotor's pitch angle $\Theta$ using gyroscope measurements by subtracting quadrotor's pitch angle $\Theta$ from $\theta$.

### 4.3 Moving Towards the Landpad

Once we detected the landpad and estimated the flight distance $d_t$ between the marker and the quadrotor, we employ the monocular SLAM framework introduced by Engel et al. [7] to accurately navigate towards the landpad.

The monocular SLAM framework consists of three main components: (1) a monocular SLAM system, (2) an extended Kalman filter (EKF) for data fusion and state estimation, (3) and PID control used to generate steering commands.

The monocular SLAM system is based on Parallel Tracking and Mapping (PTAM) combined with a closed-form solution for estimating the map scale. The EKF is used to fuse all the available data, and compensate for different time delays in the system that arise from the communication through wireless LAN and the computationally expensive visual tracking. The PID control is used in combination with the position and velocity estimates from the EKF to steer the quadrotor towards the desired location $\boldsymbol{q} = \left[\hat{x}\ \hat{y}\ \hat{z}\ \hat{\Psi}\right]^\top \in \mathbb{R}^4$ defined in a world coordinate system. For each of the four degrees of freedom, a separate PID controller is used with corresponding gains determined experimentally.

The scale of the visual map is estimated using both the visual SLAM and the metric sensors, such as the sonar altimeter, by minimizing the negative log-likelihood over pairs of samples that correspond to the traveled distance.

Steering the quadrotor towards the landing spot using the monocular SLAM framework can be summarized to two steps (1) adjusting the yaw angle, such

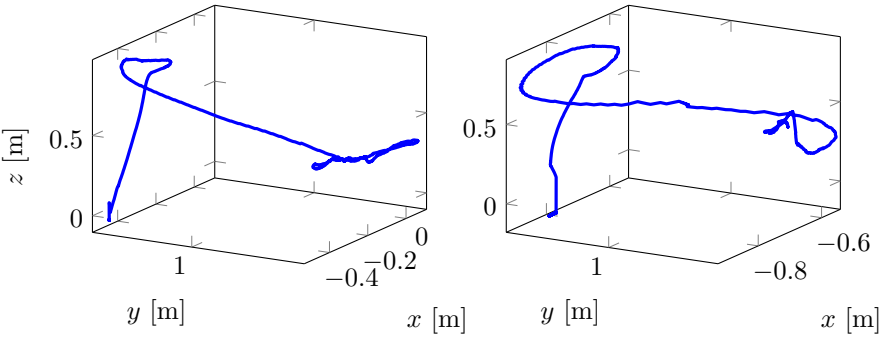



**Figure 7.** Estimated trajectories of our approach and landing. During the landing phase no visual observations by SLAM are made as only the downward-facing camera is active.

that the quadrotor faces the center of the landpad, (2) and then estimating the target location $\boldsymbol{p}$ using quadrotor's current location and the travel distance $d_t$. Additionally, we increase the altitude of the quadrotor to 1 m while approaching the landpad to enlarge the field of view of the downward-facing camera.

### 4.4 Visual Position Control

Once the quadrotor has reached the landpad, it has to recover from possible trajectory deviations introduced by SLAM. At this stage we switch from the forward-facing camera to the downward-facing one. We perform the stabilization by first detecting the landpad again, this time using the downward-facing camera. Then, we estimate the pose of the landpad with respect to the pose of the quadrotor effectively aligning the corresponding coordinate systems in the horizontal direction. To generate the steering commands, we use PID control with gains determined experimentally.

## 5 Experimental Results

We performed several runs of an experiment in indoor setting to evaluate the proposed method. In our experimental setup, we used a landpad with a radius of $r = 13.5\,\text{cm}$. We initialized the map by manually steering the quadrotor in the test environment in order to get an accurate map scale estimate. Due to the very narrow vertical field of view of the forward-facing camera, we choose the origin of the quadrotor after the takeoff in a distance of approximately 1.20 m from the landing spot, at an altitude of about 50 cm. Positioning the quadrotor nearer or further away makes it impossible to detect the landpad as it either cannot be seen using the forward-facing camera anymore, or the perspective becomes extremely oblique. The landpad marker was placed on a box with a height of 14 cm.

| Step | Time [ms] |
|---|---|
| Rectification/aspect ratio correction | 11.20 |
| Ellipse detection | 24.70 |
| Landpad matching | 11.33 |
| Pose estimation | 0.01 |
| Total | 47.24 |

**Table 1.** Average runtime of individual landpad detection steps. Ellipse detection is computationally the most expensive part.

All the computations were performed on a consumer laptop with an Intel Core i7 3610QM CPU and 8 GB RAM. The actual resolution of the frames that were captured by both cameras was $640 \times 360$ pixels. Depending on the complexity of the scene, i.e. the number of edges in the image, the landpad detection was running at a rate of 10 to 25 Hz, which can cause the quadrotor to drift while hovering over the landpad since the steering commands can not be sent fast enough. As seen in table 1, the computationally most expensive part of the landpad detection pipeline is the ellipse detection followed by landpad matching and image rectification. The computational cost for pose estimation is, however, negligible.

After initializing the map, we ran the experiment 12 times in a row. Once the quadrotor has landed, we manually measured the distance between the downward-facing camera and the landpad center. The average distance from the center after the landing was 15.7 cm with a standard deviation of 10.6 cm.

As long as the map was properly initialized, the landing spot was reached by the quadrotor with just minor corrections performed by visual position control. In cases where the landpad has been missed by over 25 cm (measured from the landpad center) after aproaching it, e.g. due to inaccurate map scale, or perturbances, no landing could be initiated as the landpad was not visible anymore. Further increasing the quadrotor's altitude while hovering over the landpad could be used as a strategy to overcome the limited range of the downward-facing camera.

## 6 Conclusions and Future Work

We presented a monocular vision system for approach and landing using an off-the-shelf Parrot AR.Drone 2.0 quadrotor MAV. In our method, we detect a circular landpad marker using the forward-facing camera by employing a contour-based ellipse detection method. We then estimate the distance between the quadrotor and the landpad, and use a PTAM based SLAM framework to fly towards the landing spot. Once the target position has been reached, we detect the landpad again, now using the downward-facing camera. To correct possible drift while approaching the landpad, the pose of the quadrotor with respect to the landpad is estimated and used to correct quadrotor's position.

We tested our method in an indoor setting where it has shown to work accurately by reaching the landing spot with an average deviation of 15.7 cm between the landpad center and the mounting position of the downward-facing camera. However, the accuracy of the method depends both on the quality of the map built by the PTAM based SLAM framework, and on the precision of the estimated map scale. In this work, we did not use a synchronization mechanism for landpad detection and pose estimation that compensates measurement delays between on-board and ground computers. Every video frame was processed instantly without storing and processing video frames that arrive in between. A synchronization mechanism, however, could improve the robustness of the system. In future work, we will integrate such a mechanism into our system. Our ultimate goal is to eliminate the use of a synthetic marker altogether, and use natural, on-line learned landmarks instead.

## References

1. Fraundorfer, F., Heng, L., Honegger, D., Lee, G.H., Meier, L., Tanskanen, P., Pollefeys, M.: Vision-based autonomous mapping and exploration using a quadrotor MAV. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE (October 2012) 4557–4564
2. Lange, S., Sunderhauf, N., Protzel, P.: A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments. In: Proceedings of the International Conference on Advanced Robotics, IEEE (2009) 1–6
3. Eberli, D., Scaramuzza, D., Weiss, S., Siegwart, R.: Vision based position control for MAVs using one single circular landmark. Journal of Intelligent & Robotic Systems **61**(1-4) (January 2011) 495–512
4. Wendel, A., Maurer, M., Katusic, M., Bischof, H.: Fuzzy visual servoing for micro aerial vehicles. In: Proceedings of the Austrian Robotics Workshop. (2012)
5. Yang, S., Scherer, S.A., Zell, A.: An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. Journal of Intelligent & Robotic Systems **69**(1–4) (January 2013) 499–515
6. Yang, S., Scherer, S.A., Schauwecker, K., Zell, A.: Onboard monocular vision for landing of an MAV on a landing site specified by a single reference image. In: Proceedings of the International Conference on Unmanned Aircraft Systems. (2013) 317–324
7. Engel, J., Sturm, J., Cremers, D.: Camera-based navigation of a low-cost quadrocopter. In: Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems. (October 2012) 2815–2821
8. Nguyen, T.M., Ahuja, S., Wu, Q.M.J.: A real-time ellipse detection based on edge grouping. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. (2009) 3280–3286
9. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization **10**(2) (October 1973) 112–122
10. Prasad, D.K., Leung, M.K.H., Quek, C.: ElliFit: An unconstrained, non-iterative, least squares based geometric ellipse fitting method. Pattern Recognition **46**(5) (2013) 1449–1465

11. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics **9**(1) (January 1979) 62–66
12. Chen, Q., Wu, H., Wada, T.: Camera calibration with two arbitrary coplanar circles. In Pajdla, T., Matas, J., eds.: Computer Vision – ECCV 2004. Volume 3023 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2004) 521–532
13. Kanatani, K., Liu, W.: 3D interpretation of conics and orthogonality. CVGIP: Image Understanding **58**(3) (1993) 286–301