Fully autonomous MAV flight and landing on a moving target using visual-inertial estimation and model-predictive control

Dimos Tzoumanikas[1,*], Wenbin Li[1], Marius Grimm[2,1], Ketao Zhang[3]
Mirko Kovac[3] and Stefan Leutenegger[1]

[*]dt214@ic.ac.uk

[1]Department of Computing, Imperial College London, UK
[2]Department of Mechanical Engineering, ETH Zurich, Switzerland
[3]Department of Aeronautics, Imperial College London, UK

Abstract

The Mohamed Bin Zayed International Robotics Challenge (MBZIRC) held in spring 2017 was a very successful competition well attended by teams from all over the world. One of the challenges (Challenge One) required an aerial robot to detect, follow and land on a moving target in a fully autonomous fashion. In this paper, we present the hardware components of the Micro Air Vehicle (MAV) we built with off the self components alongside the designed algorithms that were developed for the purposes of the competition. We tackle the challenge of landing on a moving target by adopting a generic approach, rather than following one that is tailored to the MBZIRC Challenge One setup, enabling easy adaptation to a wider range of applications and targets, even indoors, since we do not rely on availability of GPS. We evaluate our system in an uncontrolled outdoor environment where our MAV successfully and consistently lands on a target moving at a speed of up to 5.0 m/s.

1 Introduction

Robotics teams from all over the world attended the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) which was held in spring 2017 for the first time. The venue included an arena designed to deploy aerial vehicles capable of executing autonomous flights. Challenge one required a drone to autonomously detect and land on a flat ferrous target (1.5 m $\times$ 1.5 m) placed on top of a moving ground vehicle. The duration of the challenge was constrained to 15 minutes. The moving ground vehicle had a speed of 15 km/h for the first 8 minutes and 5 km/h for the next 7 minutes. A landing was considered successful only if the drone neither had visible damage nor fell off from the target. Autonomous flight in such an uncontrolled environment is challenging, as it requires fast response from sensing and precise state estimation, as well as good synchronisation to other subsystems i.e. controller and target tracking.

Rather than custom-tailoring an approach to the specific challenge, we developed and integrated a suite of algorithms that allow us fully autonomous flight in a wider range of scenarios and conditions, including the ones of MBZIRC. In the design of both the platform and the software stack, we decided for a low-cost, yet robust solution. Specifically, we adopt an extended visual-inertial odometry framework, OKVIS

(Leutenegger et al., 2014), providing the basis for control, even if GPS is either not available or not reliable enough. A downward-looking fisheye camera is used for detection and tracking of a moving target, formulated as a tracking problem in 3D space. We close the position control loop with a Model-Predictive Controller (MPC), which has proven extremely robust even in windy conditions. The overall system performance is evaluated in outdoor test flights simulating the MBZIRC challenge 1. We show that the proposed setup is capable of landing on a target moving at up to 18 km/h in presence of wind in the order of 15 km/h. In short, we make the following contributions that we believe will be of interest to the field robotics community:

- We describe in-depth a low-cost but robust hexacopter MAV with components dedicated for general autonomous flight as well as landing on a moving target.
- We present a suite of estimation algorithms for motion tracking in GPS-denied environments using an RGBD+IMU camera, as well as for detecting and tracking of the landing pattern.
- We introduce an MPC based on an identified linear MAV model that includes hard input constraints as well as soft state constraints, minimizing a cost function that considers both reference position and velocity for maximally accurate trajectory tracking.

The remainder of this work is organised as follows: we further detail related work in Section 2, followed by an overview of the hardware and software system in Section 3. Common notations and definitions used throughout this work are introduced in Section 4. The details of the visual-inertial estimation and control algorithms are presented in Section 5 and 6 respectively. Last, we show our flight results in Section 7 and discuss them in Section 8.

2 Related work

Micro Air Vehicles (MAV) have been deployed in a wide range of applications, described e.g. in (Kumar and Michael, 2012). They are often used in aerial surveillance and tasks within complex and hazardous environments where manual control may become difficult or impractical. Providing MAVs with the appropriate level of control autonomy, has thus been a major challenge tackled by the community. In this context, such systems should be designed to be robust under significant uncertainty for extended periods of time, as well as able to compensate for failures without human intervention.

Amongst the first quadrotor control papers, we find (Castillo et al., 2003) where a control algorithm able to stabilize the position of a quadcopter was presented. Later (Bouabdallah et al., 2004) presented a controller able to stabilize the orientation of a quadcopter which was fixed on a testbed. This work was followed by (Bouabdallah, 2007; Bouabdallah and Siegwart, 2007) where they presented experimental results of autonomous operation including collision avoidance. They used a custom-made quadrotor with on-board data processing equipped with all the necessary sensors for autonomous operation. Soon, the community started investigating deployment of MAVs in GPS-denied environments, e.g. in a disaster scenario, such as (Weiss et al., 2012), relying on an early adoption of visual-inertial estimation. (Bachrach et al., 2011) further draws the attention to autonomously flying MAV within GPS-denied environment using a SLAM system using a stereo camera and additionally relying on a laser scanner. Their system shows the potential of employing a multilevel sensing and control hierarchy that is capable to close the loop of the fast dynamics relying only on on-board sensors. Similar system setups with different cameras and sensors have been widely used to explore complex indoor and outdoor scenes (Mahony et al., 2012; Shen et al., 2011) using either monocular or stereo cameras. Compared to the stereo cameras, an RGB-D camera offers color images along with pixel-aligned dense depth information. (Huang et al., 2017) presented a visual odometry and mapping approach using an RGB-D camera, as well as an application to autonomous flight, while (Li et al., 2013a) and (Li et al., 2015) presented a combined RGBD-inertial based state estimation scheme which along with an RRT path planner enables autonomous navigation of an MAV in indoor environments (Li et al., 2013b).

Performing a fully autonomous landing, tracking and following a fast moving target is challenging. This requires an MAV to employ accurate state estimation as well as requires the MAV to venture into the domain of highly coupled nonlinear dynamics at high speeds. In the literature, a combination of vision-based SLAM and model-based control has crystallised to be very suitable for such scenarios.

Vision-based SLAM has been widely used for state estimation of MAVs within both indoor and outdoor scenarios. Its tremendous progress over the past years was driven by advances in computational power (e.g. GPUs), availability of novel and cheaper sensors (e.g. depth cameras, MEMS IMUs). We distinguish sparse systems, reconstructing the environment as a number of discrete 3D landmarks along with the motion of the camera, detected and associated across image; and dense ones, representing a continuous map, usually formulated in a direct manner, i.e. directly optimising for appearance consistency across successive image frames. Sparse systems are typically less computationally intensive and thus more suitable for integration into drones. For highest robustness and accuracy, the combination of visual-inertial SLAM has become very popular, especially for use on-board drones. Amongst earliest adopters, (Weiss et al., 2012) used a loosely-coupled combination of a visual SLAM system. More robust and accurate systems to detain the sparse world, however, combine visual and inertial sensing modalities in a probabilistic and *tightly* coupled manner, using filtering, such as (Mourikis and Roumeliotis, 2007) or a derivative of it, and optimisation techniques such as (Forster et al., 2015; Kaess et al., 2012) or (Leutenegger et al., 2014), available open source as "OKVIS: Open Keyframe-based Visual-Inertial SLAM", which we adopted and modified in this work.

Autonomous flight also relies on precise control. Significant progress in this field has been made over the last years. We would like to highlight the work of (Mellinger and Kumar, 2011) that was among the first ones showing the aggressive flying capabilities of MAV. They presented a trajectory generation technique which exploits the differential flatness property of an MAV; accompanied with a geometric tracking controller, this enables precise tracking of aggressive but feasible maneuvers. Similarly, (Mueller and D'Andrea, 2014) presented a non linear controller which can stabilize an MAV despite the loss of one, two or three propellers and demonstrate the significance of using the MAV model in the control synthesis. Regarding linear model predictive control we consider the work of (Papachristos et al., 2016), (Darivianakis et al., 2014) and (Oettershagen et al., 2014) to be the most related one with our control synthesis. They show that a predictive controller can successfully be used for a variety of complex and practical applications (e.g. physical interaction between MAV and the environment) and can easily be implemented in several platforms from quadcopters, tiltrotors to fixed wing aircraft. We would also like to highlight the excellent work from (Kamel et al., 2015) and (Kamel et al.,) where the idea of using the fully non-linear model in a predictive controller is implemented for controlling the attitude and the position of an MAV respectively.

Given impressive developments on both vision-based SLAM and MAV control, (Lee et al., 2012) introduced an MAV system able of tracking a moving target using an image-based visual servo system. Their vision system is reported to be less sensitive and computationally cheaper in state estimation, which yields fast response on MAV dynamics. (Serra et al., 2016) presented a system based on loosely-coupled visual-inertial fusion, capable of landing on a moving target. (Borowczyk et al., 2016) designed an MAV system with multiple cameras, IMUs and GPS. They implemented a Kalman filter based odometry algorithm which is reported to accurately estimate the relative position and velocity between the vehicle and the moving target. Their system successfully lands on a target moving with speed up to 50km/h. In some special scenario, the moving target may not be flat. (Vlantis et al., 2015) proposed a self-adaptive MAV system that detects and tracks the 3D pose of the moving target then adjust the MAV landing attitude. Their system shows a successful landing on an inclined target of a moving ground robot. More recently, (Thomas et al., 2017) presented a small MAV system(15cm diameter and 250g payload) that detects, localizes, and tracks moving spherical objects relying only on onboard sensors and computation. Their proposed odometry and control systems consider the underactuated dynamics, the actuator limitations, and the field of view constraints, in order to give extra robustness to abrupt variations in target motion. We would also like to highlight the work of other MBZIRC participants such as (Falanga et al., 2017), (Beul et al., 2017) and (Bähnemann et al., 2017), where the last two of them achieved successful landing in the competition and their work will be discussed later.

3 Overview of the system

3.1 Hardware

Our MAV is a custom built hexacopter, using off the shelf components. A hexacopter was preferred due to its payload capacity compared to a quadcopter of a similar size.

We use the DJI F550[1] frame due to its compact design and low cost. The propulsion system (motors-ESC-propellers) is the DJI E310[2]. We prioritized the MAV responsiveness over the maximum flight time and preferred the 960kV motor version.

The MAV is equipped with a Pixhawk[3] flight controller, which is commonly used in research, and which we flashed with a modified version of the PX4 firmware[4]. The firmware enables easy interaction, throught mavlink messages, between the flight controller and the Intel NUC i7-7567U[5] onboard computer.

An Intel RealSense ZR300 RGBD+IMU[6] sensor is used for the state estimation scheme while a FLIR Chameleon 3[7] is mounted on the lower plate of the MAV and is used for the target tracking. The whole system is powered by a 4S Zippy Compact 5800 mAh battery which results a 15 minute flight time when the system is fully loaded (Intel RealSense + onboard computer + downward looking camera). Special attention was payed on the mechanical assembly of the Intel RealSense RGBD+IMU sensor and the Pixhawk flight controller. They were soft mounted on the MAV frame to prevent the mechanical vibrations of the motors from significantly affecting the natively noisy IMU measurements. The custom built MAV with all the extra hardware components is illustrated in Figure 1 (left).

In order to absorb the impact energy when the hexacopter MAV performs high speed landing maneuvers, an origami-folding inspired landing pad made from soft materials was designed using multi-material additive manufacturing techniques. The landing pad shown in Figure 1 (right), is attached to each of the MAV arms, consists of compliant hinges with geometry in single curvature shell and morphing shells which can be passively bent by the impact forces when the six feet make contact with the landing site. The performance of the soft landing pad depends on the thickness of the shells as well as the material used. In this work, micro-carbon fiber reinforced thermoplastic, Markforged Onyx, was used and the modules were fabricated with 0.1 mm thickness of each slice and triangular fill pattern at 100% fill density.

3.2 Software components

On the software side there are three main components. The visual-inertial odometry is responsible for the estimation of the MAV position, orientation and the respective velocities. A target tracking EKF is responsible for the estimation of landing target position and velocity. This information is later used by a model predictive controller in order to stabilize the MAV and navigate it accordingly to a desired position. We use ROS[8] as a middleware for the exchange of data between the individual software components. An overview of the hardware and software components is illustrated in Figure 2.

[1] See `https://www.dji.com/flame-wheel-arf`.
[2] See `http://www.dji.com/e310`.
[3] See `https://pixhawk.org/`.
[4] See `https://github.com/PX4/Firmware`.
[5] See `https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html`.
[6] See `https://click.intel.com/realsense.html`.
[7] See `https://www.ptgrey.com/chameleon3-usb3-vision-cameras`.
[8] See `http://www.ros.org/`.

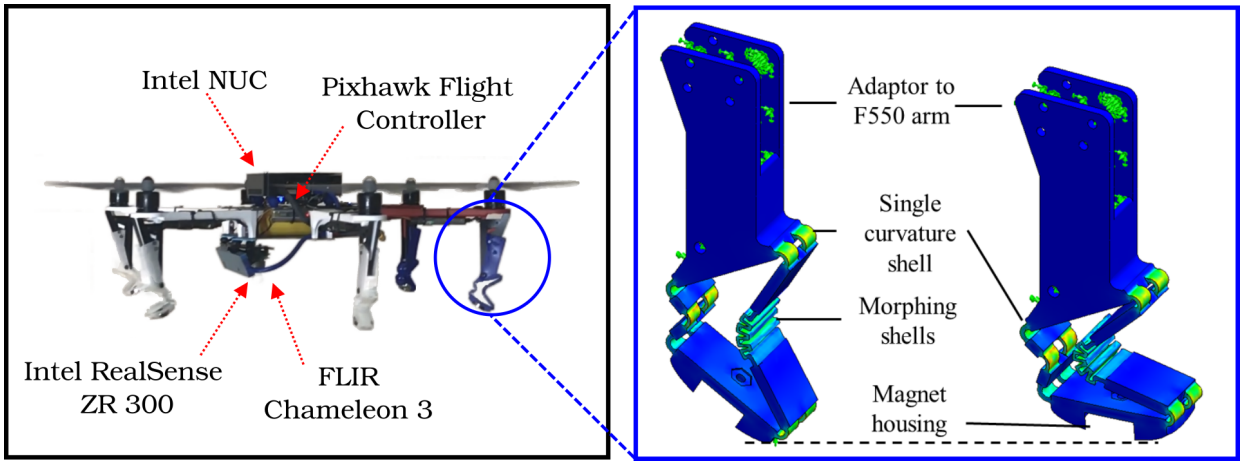

Figure 1: Left: MAV hardware explained with most important components. Right: The soft landing pad in its original configuration and the partially folded configuration where the magnet attaches to the horizontal surface.

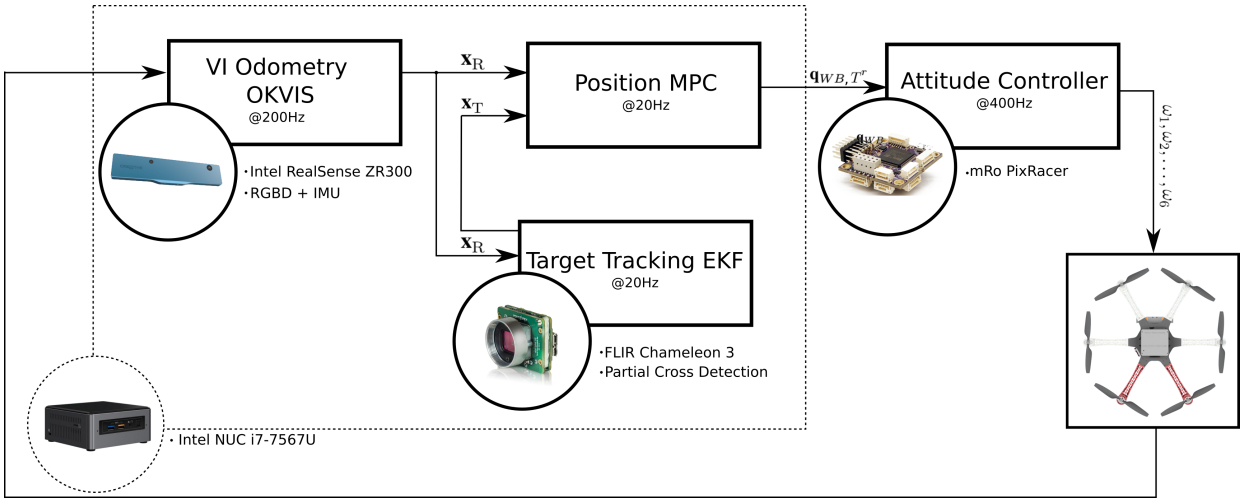

Figure 2: OKVIS (Leutenegger et al., 2014) is used in combination with the Intel RealSense ZR300 for the visual-inertial estimation of the MAV state $\mathbf{x}_R$. A monochrome FLIR Chameleon 3 is used for the detection of the moving target. The target state $\mathbf{x}_T$ and the MAV state $\mathbf{x}_R$ are further used in the MPC which generates a reference quaternion $\mathbf{q}_{WB}$ and reference collective thrust for the attitude controller. The attitude controller is implemented on an mRo PixRacer flight controller which outputs the corresponding $\omega_1, \cdots, \omega_6$ motor commands.

4 Notation and definitions

We denote vectors as bold lowercase symbols, e.g. $\mathbf{v}$. When required, we indicate the coordinate representation in a frame of reference $\underrightarrow{\mathcal{F}}_A$ with left-hand subscripts, ${}_A\mathbf{v}$. A rotation matrix $\mathbf{C}_{AB}$ changes the coordinate representation of a vector from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$ as ${}_A\mathbf{v} = \mathbf{C}_{AB}\,{}_B\mathbf{v}$. We also use quaternions to represent this orientation, i.e. $\mathbf{q}_{AB}$. Operator $\otimes$ denotes the quaternion multiplication.

In terms of positions, we denote the position of a point P relative to the origin of $\underrightarrow{\mathcal{F}}_A$ as ${}_P\mathbf{r}_A$. We use homogeneous transformations $\boldsymbol{T}_{AB}$ to change coordinate representations of position vectors in homogeneous coordinates as ${}_P\boldsymbol{r}_A = \boldsymbol{T}_{AB}\,{}_P\boldsymbol{r}_B$.

All motion is referenced relative to a World-frame $\underrightarrow{\mathcal{F}}_W$ (Earth-fixed and tangential to the surface with z-axis upward) that we approximate to be also an inertial frame. Moreover, we consider a moving landing target frame of reference $\underrightarrow{\mathcal{F}}_T$.

We formulate the MAV dynamics and controller relative to its centre of mass, with a respective coordinate frame $\underrightarrow{\mathcal{F}}_B$ (x: forward, y: left, z: upward). In the specific sensor setup at hand, we consider the RGB-D camera with its coordinate frame $\underrightarrow{\mathcal{F}}_C$, and the downward-looking camera frame $\underrightarrow{\mathcal{F}}_D$. We furthermore use the IMU coordinate frame $\underrightarrow{\mathcal{F}}_S$. The different coordinate frames used are illustrated in Figure 3.

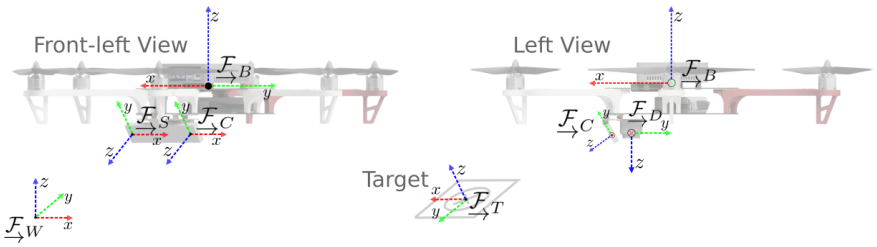

Figure 3: The different coordinate frames used. Namely, $\underrightarrow{\mathcal{F}}_W$: the Earth fixed frame, $\underrightarrow{\mathcal{F}}_B$: the MAV body fixed frame, $\underrightarrow{\mathcal{F}}_C$: the RGB-D camera frame, $\underrightarrow{\mathcal{F}}_S$: the IMU sensor frame, $\underrightarrow{\mathcal{F}}_D$: the downward looking camera frame and $\underrightarrow{\mathcal{F}}_T$: the moving target frame.

5 Estimation of the MAV and target state

We use OKVIS: Open Keyframe-based Visual Inertial SLAM[9] described in (Leutenegger et al., 2014) as a basis for our MAV state estimation. We made, however, several extensions in order for it to be usable with a low-cost RGB-D-inertial camera, as well as to increase accuracy. We furthermore present a suite of detection and tracking algorithms implemented in order to reliably estimate the landing pattern pose and velocity using the downward-looking fisheye camera.

5.1 Visual-inertial MAV state estimation

The variables to be estimated consist of the robot states at the image times (index k) $\mathbf{x}_{\mathrm{R}}^k$ and landmarks $\mathbf{x}_{\mathrm{L}}$. $\mathbf{x}_{\mathrm{R}}$ holds the robot position in the inertial frame ${}_W\mathbf{r}_S$, the body orientation quaternion $\mathbf{q}_{WS}$, the velocity expressed in the sensor frame ${}_S\mathbf{v}$, as well as the biases of the gyroscopes $\mathbf{b}_{\mathrm{g}}$ and the biases of the accelerometers $\mathbf{b}_{\mathrm{a}}$. Thus, $\mathbf{x}_{\mathrm{R}}$ is written as:

$$\mathbf{x}_{\mathrm{R}} := \left[{}_W\mathbf{r}_S{}^T, \mathbf{q}_{WS}{}^T, {}_W\mathbf{v}_{WS}{}^T, \mathbf{b}_{\mathrm{g}}^T, \mathbf{b}_{\mathrm{a}}^T\right]^T \in \mathbb{R}^3 \times S^3 \times \mathbb{R}^9. \tag{1}$$

[9] Available open-source at `http://ethz-asl.github.io/okvis/`.

We adopt the approach of ORB-SLAM 2 (Mur-Artal and Tardós, 2017) to create a *virtual* stereo camera to incorporate depth information per keypoint measurement in the RGB image (if available). We obtain virtual keypoint measurements by projecting 3D points into the virtual second camera frame at pose $\underrightarrow{\mathcal{F}}_{C_\mathrm{v}}$.

Let $\mathbf{u} = \mathbf{u}({}_P\boldsymbol{r}_C)$ be the RGB camera projection model (which may include distortion), and ${}_P\boldsymbol{r}_C = \boldsymbol{h}^{-1}(\mathbf{u}, \mathbf{D})$ the back-projection from image coordinates to a homogeneous point represented in $\underrightarrow{\mathcal{F}}_C$, given the depth image $\mathbf{D}$. We can now create *virtual* keypoint measurements $\mathbf{z}_\mathrm{v}^j$ from actual measurements $\mathbf{z}^j$, iff depth is given for said pixel, as

$$\mathbf{z}_\mathrm{v}^j = \mathbf{u}_\mathrm{v}(\boldsymbol{T}_{C_\mathrm{v}C}\,\mathbf{u}^{-1}(\mathbf{z}^j, \mathbf{D})), \tag{2}$$

where $\boldsymbol{T}_{C_\mathrm{v}C}$ denotes the pose of the virtual camera relative to the RGB camera and $\mathbf{u}_\mathrm{v}(\cdot)$ stands for the projection model that we define as $\mathbf{u}_\mathrm{v}(\cdot) := \mathbf{u}(\cdot)$ for convenience. Note that this scheme allows for easy compatibility with an N-camera system assumed in (Leutenegger et al., 2014). Furthermore, since the depth map of the camera we use was indeed obtained through stereo triangulation, formulating reprojection errors in a stereo setup (even if virtual), correctly accounts for noise in image space. We set the baseline in $\boldsymbol{T}_{C_\mathrm{v}C}$ to the actual distance between the infrared stereo camera, 10 cm in our case.

Now we can formulate the overall cost function consisting of reprojection errors $\mathbf{e}_\mathrm{r}^{j,k}$, virtual camera reprojection errors $\mathbf{e}_\mathrm{r,v}^{j,k}$ and IMU errors as $\mathbf{e}_\mathrm{s}^k$

$$J(\mathbf{x}) := \sum_{k=1}^{K} \sum_{j \in \mathcal{J}(k)} \mathbf{e}_\mathrm{r}^{j,k\,T} \mathbf{W}_\mathrm{r}^{j,k} \mathbf{e}_\mathrm{r}^{j,k} + \sum_{k=1}^{K} \sum_{j \in \mathcal{J}_\mathrm{v}(k)} \mathbf{e}_\mathrm{r,v}^{j,k\,T} \mathbf{W}_\mathrm{r,v}^{j,k} \mathbf{e}_\mathrm{r,v}^{j,k} + \sum_{k=1}^{K-1} \mathbf{e}_\mathrm{s}^{k\,T} \mathbf{W}_\mathrm{s}^{k} \mathbf{e}_\mathrm{s}^{k}, \tag{3}$$

where k stands for the camera frame index, and j denotes the landmark index. The indices of landmarks visible in the k^{th} RGB frame are written as the set $\mathcal{J}(k)$ (and analogously as $\mathcal{J}_\mathrm{v}(k)$ for the virtual camera). Moreover, $\mathbf{W}_\mathrm{r}^{j,k}$ represents the information matrix of the respective landmark measurement, and $\mathbf{W}_\mathrm{s}^k$ the information of the k^{th} IMU error.

Now, the reprojection error for the j^{th} landmark ${}_W\boldsymbol{l}^j$ viewed in the k^{th} RGB camera image is defined as

$$\mathbf{e}_\mathrm{r}^{j,k} = \mathbf{z}^{j,k} - \mathbf{u}\left(\boldsymbol{T}_{CS}^k\,\boldsymbol{T}_{SW}^k\,{}_W\boldsymbol{l}^j\right), \tag{4}$$

and analogously in the virtual camera. See (Leutenegger et al., 2014) for details about data-association, linearisation, the IMU error term formulation, the optimisation and the marginalisation scheme applied.

5.2 Landing pattern detection and tracking

Our vision pipeline of establishing the target pose at each frame the downward looking fisheye camera takes, consists of a *detection* step, requiring the entire landing pattern (as specified by the MBZIRC organisers) to be visible and a *tracking* step, where we use a motion model and observations of parts of the landing pattern that are sufficient to reliably and accurately maintain the pose estimates.

5.2.1 Landing pattern detection and initial pose estimation

Our landing pattern detection pipeline first applies basic image processing steps, namely undistortion, smoothing, locally adaptive-threshold binarisation, edge detection and contour extraction (with polygon simplification). We then run P3P pose estimation on detected quads. All of these steps are implemented using the OpenCV library[10].

Thus far, the procedure is similar to state-of-the-art marker detectors, such as AprilTags C++ (Kaess, 2013) – note, however, that the latter would not run in real-time on our platform.

[10] See `http://opencv.org/`.

The final step of verification involves a perspective warp and comparison to the template landing pattern via one BRISK (Leutenegger et al., 2011) descriptor match (which is extremely fast). As a template, we use a 255 by 255 image of the landing pattern.

Figure 4 shows an example of these steps based on an image taken with our downward looking fisheye camera.

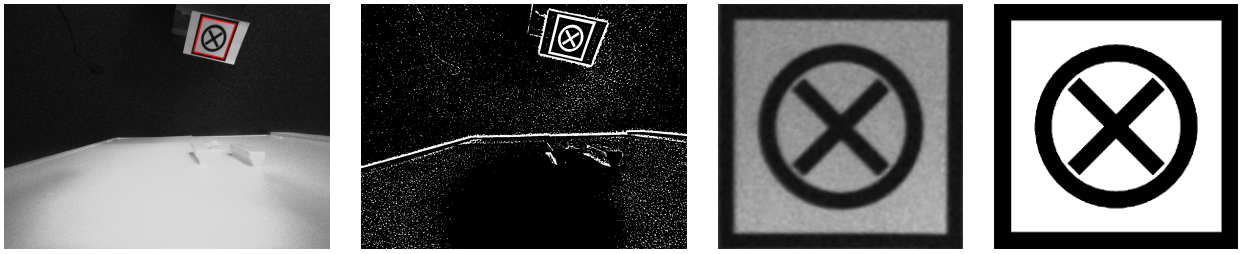

Figure 4: From left to right: undistorted image with detected quad (red), adaptively thresholded image for contour extraction, perspectively warped detection for appearance verification, and target template pattern.

5.2.2 Landing target tracking

We track the landing target by means of an Extended Kalman Filter (EKF), where we assume the MAV pose is given by the estimator described above, and poses are accurate enough.

The dynamics employed for the prediction step consists of a constant velocity model for linear translation, and a constant orientation model, since rotation speeds of the target remain small. Inclusion of linear velocity of the target into its estimated state, however, is crucial, since we need it for accurate Model Predictive Control (MPC).

In maths, we estimate the following target state:

$$\mathbf{x}_{\mathrm{T}} := \left[{}_W\mathbf{r}_T{}^T, \mathbf{q}_{WT}{}^T, {}_W\mathbf{v}_{WT}{}^T \right]^T \in \mathbb{R}^3 \times S^3 \times \mathbb{R}^3, \tag{5}$$

and we consider the following prediction model:

$$ {}_W\dot{\mathbf{r}}_T = {}_W\mathbf{v}_{WT}\,, \tag{6}$$

$$ \dot{\mathbf{q}}_{WT} = \frac{1}{2}\begin{bmatrix}\mathbf{w}_{\mathrm{rot}} \\ 0\end{bmatrix} \otimes \mathbf{q}_{WT}\,, \tag{7}$$

$$ {}_W\dot{\mathbf{v}}_{WT} = \mathbf{w}_{\mathrm{vel}}, \tag{8}$$

where $\mathbf{w}_{\mathrm{rot}}$ and $\mathbf{w}_{\mathrm{vel}}$ denote 3-dimensional uncorrelated Gaussian white noise processes affecting orientation, and velocity, respectively. In our experiments, we set the noise parameters of said processes to $\sigma_{\mathrm{rot}} = [0.02, 0.02, 0.2]^T \mathrm{rad}/\sqrt{\mathrm{hz}}$ and $\sigma_{\mathrm{vel}} = [0.1, 0.1, 0.1]^T \mathrm{m}/(\mathrm{s}\sqrt{\mathrm{hz}})$, respectively.

For the update step, we employ observations of pre-defined landing pattern keypoints. Please refer to Figure 5 for an illustration of the specific keypoint locations on the target pattern at hand.

As the measurement function, we therefore use the predicted keypoint location of the t^{th} keypoint into the (undistorted) downward-looking fisheye camera:

$$\mathbf{h}_t(\mathbf{x}_{\mathrm{T}}) = \mathbf{u}_{\mathrm{D}}(\boldsymbol{T}_{WD}^{-1}\, \boldsymbol{T}_{WT}\, {}_T\boldsymbol{r}_t), \tag{9}$$

where $\mathbf{u}_{\mathrm{D}}$ denotes the (undistorted) projection model of the downward looking camera, the transformation $\boldsymbol{T}_{WD}$ is obtained through visual-inertial MAV state estimation, and the location of the keyoint on the target, ${}_T\boldsymbol{r}_t$, is a known constant.

Note that in our implementation, we use a tangent space representation of the orientation, $\delta\boldsymbol{\alpha}_{WT}$, around the current estimate $\overline{\mathbf{q}}_{WT}$ analogous to (Leutenegger et al., 2014), as

$$\mathbf{q}_{WT} = \exp(\delta\boldsymbol{\alpha}_{WT}) \otimes \overline{\mathbf{q}}_{WT} = \begin{bmatrix} \operatorname{sinc}\left\| \frac{\delta\boldsymbol{\alpha}_{WT}}{2} \right\| \frac{\delta\boldsymbol{\alpha}_{WT}}{2} \\ \cos\left\| \frac{\delta\boldsymbol{\alpha}_{WT}}{2} \right\| \end{bmatrix} \otimes \overline{\mathbf{q}}_{WT} \,, \tag{10}$$

With the exponential map $\exp(\cdot)$. We refer the reader to Appendix A, where we provide the full set of EKF prediction and update equations, including the Jacobians of Equation (9).

In order to obtain the keypoint measurements to formulate the EKF update residual, we employ tracking in image space: we use 20 by 20 pixel patches around the keypoints warped from the pattern template image using the predicted pose relative to the camera observing them. Then, using keypoint location predictions from Equation (9), we brute-force search their neighbourhood in the binarised image (60 by 60 pixel) by means of finding the square difference minimum w.r.t. the warped template patches. In our implementation, we used a keypoint measurement standard deviation of 5 pixels – partly accounting for the lack of proper hardware synchronisation between the RealSense (used for OKVIS) and the downward looking fisheye camera. For outlier removal, we discard keypoint measurements where this difference is two high, and also those whose residuals don't pass a Chi-square test (threshold: 9).

We illustrate these steps in Figure 5.

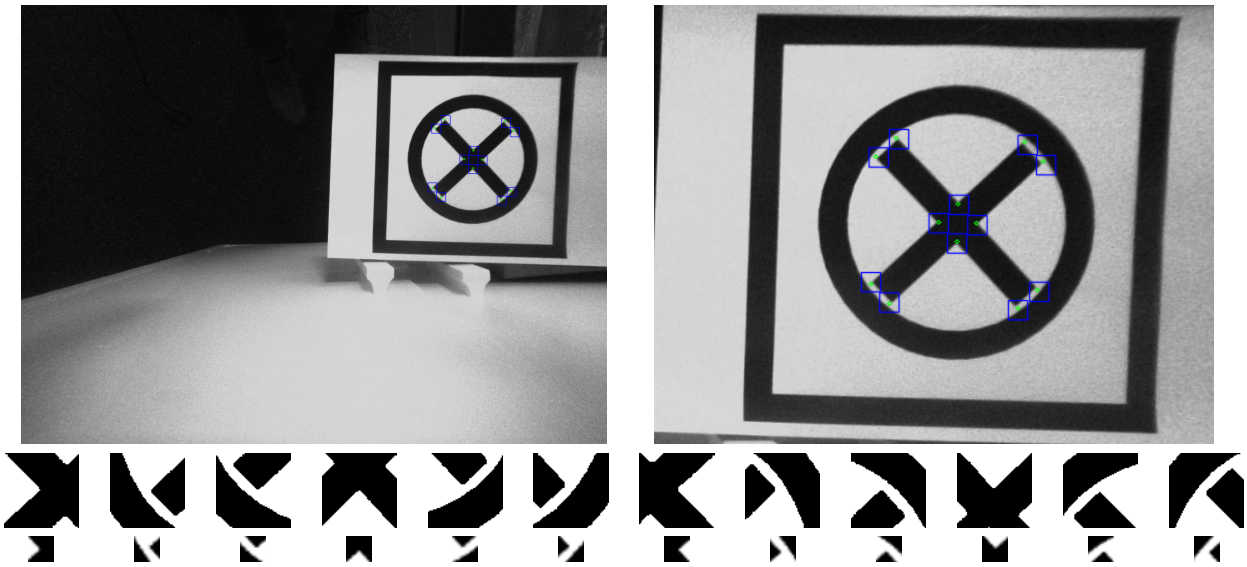

Figure 5: Top row: projected predicted keypoint neighbourhoods (search area) as blue boxes, and actual found detections as green circles (right: zoomed in). Middle row: binarised search neighbourhoods. Bottom row: respective warped templates to match.

6 Control architecture

Position tracking is achieved by means of a cascaded connection of a Model Predictive Controller (MPC) for the MAV position and a PID controller for its attitude. We implemented our MPC on the onboard computer whereas the PID controller is implemented on the Pixhawk flight controller as an off-the-shelf component. The above approach was motivated by the fact that the majority of the commercially available flight controllers and MAV come with a pre-implemented attitude controller which requires little or even no tuning enabling easy adaptation to a wide range of platforms. This approach also allows a simpler model to

be used in the position MPC. The MAV non-linear dynamics combined with a well tuned attitude controller, can be sufficiently approximated, as it will be shown later, with a much simpler linear model which describes the evolution of the combined attitude controlled-MAV dynamics.

Our work on the control synthesis is closely related to previous work of (Papachristos et al., 2016; Darivianakis et al., 2014) that has been successfully implemented on similar multirotor platforms and (Oettershagen et al., 2014; Oettershagen et al., 2016) that has been deployed on fixed wing platforms.

However, our work differs from them in that we employ soft state constraints which can be occasionally violated in order to guarantee the feasibility of the underlying optimization problem. Moreover, we do not make use of a pre-computed solution of the optimization problem, in the form of look up tables. This is formulated as a Quadratic Program (QP) and is being solved in realtime. The current approach enables seamless tuning of the control parameters and fast adaptation to different control models and constraints as the computation of an explicit solution in not needed.

6.1 Linear model and system identification

The model used for control uses Euler angles (ZYX convention with yaw $\psi \in [-\pi, \pi)$, pitch $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2})$, roll $\phi \in [-\pi, \pi)$) as a parameterization for the MAV orientation. Quaternions are used for state estimation as it was designed to describe any possible motion (not bound to the the application at hand) avoiding the numerical issues related to gimbal lock. Euler angles, however, are preferred for the control formulation as it is easier to derive the corresponding linear model. We ensure through optimization constraints that the MAV operates at pitch angles far from the gimbal lock.

Since the heading angle ψ does not contribute to the translational motion of the MAV and in order to eliminate it from the model used for control, we express the MAV position and linear velocities in the navigation frame $\underrightarrow{\mathcal{F}}_N$. The frame $\underrightarrow{\mathcal{F}}_N$ is obtained when the frame $\underrightarrow{\mathcal{F}}_W$ is rotated by an angle ψ around its z axis and translated to the origin of the body frame $\underrightarrow{\mathcal{F}}_B$. The used linear model takes the following form:

$$ {}_N\dot{\mathbf{v}} = \begin{bmatrix} g\theta - c_x\dot{x} \\ -g\phi - c_y\dot{y} \\ T - c_z\dot{z} \end{bmatrix}, \tag{11} $$

with $g = 9.81$ m/s^2 the gravitational acceleration. The linear model was derived by assuming that the MAV is in a near-hover operation and that the translational dynamics are controlled by the projection of the MAV thrust force on the $\underrightarrow{\mathcal{F}}_N$ in order to generate translational acceleration. The terms $c_x\dot{x}$, $c_y\dot{y}$, $c_z\dot{z}$ model the system damping due the aerodynamic friction, approximated as linear damping.

Regarding the closed loop attitude dynamics, we follow a similar approach to (Darivianakis et al., 2014) and assume that these can be approximated by a second order system. Namely:

$$ \begin{aligned} \ddot{\theta} &= -b_{\ddot{\theta}\theta}\theta - b_{\ddot{\theta}\dot{\theta}}\dot{\theta} + b_{\theta^r}\theta^r, \\ \ddot{\phi} &= -b_{\ddot{\phi}\phi}\phi - b_{\ddot{\phi}\dot{\phi}}\dot{\phi} + b_{\phi^r}\phi^r, \end{aligned} \tag{12} $$

where θ^r and ϕ^r refer to the reference θ and ϕ angles, respectively (θ^r and ϕ^r are the angles commanded to the attitude controller while θ and ϕ the ones actually achieved by the MAV). As far as the motor dynamics are concerned, it is assumed that they are significantly faster than the closed loop attitude dynamics and can thus be ignored. This means that the applied thrust T coincides with the reference one T^r. It is noted that the reference thrust T^r also contains a feed forward term to compensate the acceleration due to gravity.

The unknown constant parameters c_i and b_i of the Equations 11 and 12 were identified offline using frequency domain grey-box identification with experimental data captured from manual flights. An accurate system identification would require the application of a chirp signal, which would expose the dominant frequencies

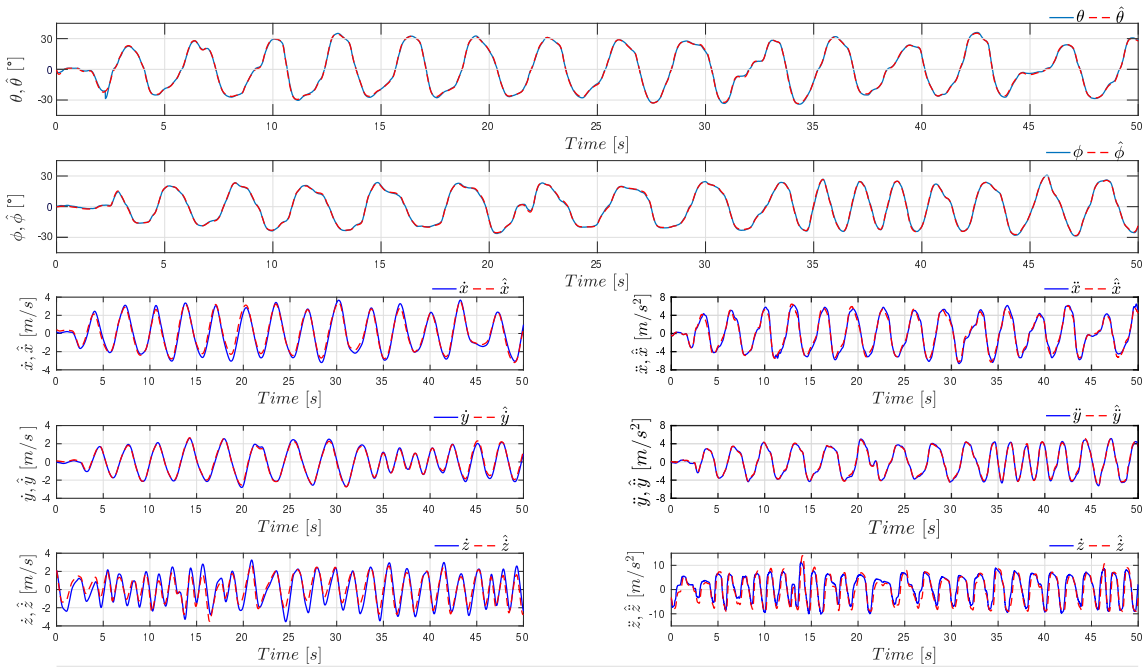

Figure 6: Real and simulated response for θ, ϕ, $\dot{x}$, $\dot{y}$, $\dot{z}$, $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$. Despite the use of a linear model, the simulated and the real attitude response (upper two plots) of the MAV almost always coincide. The simulated translational dynamics (lower 6 plots) – not surprisingly – best match the real ones in the low velocity/acceleration region.

of the system. This would yet lead to an unsafe experiment where the MAV rotates according to the chirp input but its position remains uncontrolled, and that is why the manual flight was preferred.

Two different types of datasets were recorded. In the first one, all the axes were excited independently and it was used for the parameter estimation while in the second one all the axes were excited simultaneously and was used for validation of the estimated parameters in order to avoid over-fitting. Figure 6.1 shows the real and the simulated response of the identified closed loop attitude dynamics when the inputs of the validation dataset are used and also the real and simulated responses for the translational dynamics.

Using the Equations 11 and 12 and by defining the state vector $\mathbf{x} := \left[{}_N\mathbf{r}_B{}^T, {}_N\mathbf{v}_B^T, \theta, \phi, \dot{\theta}, \dot{\phi} \right]^T \in \mathbb{R}^6 \times \mathbb{S} \times \mathbb{R}^2$ the input vector $\mathbf{u} := [\theta^r, \phi^r, T^r]^T \in \mathbb{S} \times \mathbb{R}$ where $\mathbb{S} := \{\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right), \phi \in [-\pi, \pi)\}$, the system dynamics can be written in the following time-invariant state space representation:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{A}_{\text{Lon}} & \mathbf{0}_{4\times4} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{4\times4} & \mathbf{A}_{\text{Lat}} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times4} & \mathbf{0}_{2\times4} & \mathbf{A}_{\text{Alt}} \end{bmatrix}}_{\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} \mathbf{B}_{\text{Lon}} & \mathbf{0}_{4\times1} & 0 \\ \mathbf{0}_{4\times1} & \mathbf{B}_{\text{Lat}} & 0 \\ \mathbf{0}_{2\times1} & \mathbf{0}_{2\times1} & \mathbf{B}_{\text{Alt}} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{13}$$

where $\mathbf{C} = \mathbf{I}_{10\times 10}$ and the submatrices $\mathbf{A}_{\text{Lon}}$, $\mathbf{A}_{\text{Lat}}$, $\mathbf{A}_{\text{Alt}}$, $\mathbf{B}_{\text{Lon}}$, $\mathbf{B}_{\text{Lat}}$, $\mathbf{B}_{\text{Alt}}$ are given by the following:

$$\mathbf{A}_{\text{Lon}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -c_x & g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -b_{\ddot{\theta}\theta} & -b_{\ddot{\theta}\dot{\theta}} \end{bmatrix}, \ \mathbf{B}_{\text{Lon}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ b_{\theta^r} \end{bmatrix}, \tag{14}$$

$$\mathbf{A}_{\text{Lat}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -c_y & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -b_{\ddot{\phi}\phi} & -b_{\ddot{\phi}\dot{\phi}} \end{bmatrix}, \ \mathbf{B}_{\text{Lat}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ b_{\phi^r} \end{bmatrix}, \tag{15}$$

$$\mathbf{A}_{\text{Alt}} = \begin{bmatrix} 0 & 1 \\ 0 & -c_z \end{bmatrix}, \ \mathbf{B}_{\text{Alt}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{16}$$

Notice that the state vector used for control $\mathbf{x} := [{}_N\mathbf{r}_B{}^T, {}_N\mathbf{v}_B^T, \theta, \phi, \dot{\theta}, \dot{\phi}]^T$ is a subset (with slight transformation) of the full state vector as estimated in Equation 1. Specifically ${}_N\mathbf{r}_B{}^T$ and ${}_N\mathbf{v}_B^T$ stand for the position and velocity of the MAV expressed in the $\underrightarrow{\mathcal{F}}_N$ while θ, ϕ and $\dot{\theta}$, $\dot{\phi}$ stand for the MAV pitch and roll angles and their corresponding time derivatives. θ, ϕ can easily be obtained by converting the estimated quaternion $\mathbf{q}_{WB}$ to Euler angles while their time derivatives can be obtained by the following:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \end{bmatrix} {}_B\boldsymbol{\omega}\,, \tag{17}$$

with ${}_B\boldsymbol{\omega}$ the rotational velocity of the MAV expressed in the $\underrightarrow{\mathcal{F}}_B$ frame.

Since the controller is implemented in discrete time, the above equations are discretized using zero order hold for the input $\mathbf{u}$. The discrete equivalent of the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ can be obtained by:

$$\begin{aligned} \mathbf{A}_d &= e^{\mathbf{A}dt}, \\ \mathbf{B}_d &= \int_0^{dt} e^{\mathbf{A}\tau} d\tau, \\ \mathbf{C}_d &= \mathbf{C}. \end{aligned} \tag{18}$$

The discretization step dt coincides with the control update rate and was set to 50 ms.

6.2 MPC with soft constraints formulated as a QP

The implemented MPC computes the optimal input sequence $\bar{\mathbf{u}}^* = \mathbf{u}_0^* \dots \mathbf{u}_{N-1}^*$ which is the solution of the following optimization problem:

$$\begin{aligned} \bar{\mathbf{u}}^* = & \underset{\mathbf{u}_0 \dots \mathbf{u}_{N-1}}{\text{argmin}}\ J, \\ \text{s.t. : } & \mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k, \\ & \mathbf{y}_k = \mathbf{C}_d\mathbf{x}_k, \\ & \mathbf{x}_0 = \hat{\mathbf{x}}_0, \\ & \bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}, \end{aligned} \tag{19}$$

where: $\mathbf{x}_k \in \mathbb{R}^n$ is the system state at time k, $\hat{\mathbf{x}}_0 \in \mathbb{R}^n$ the estimated state at time 0, $\mathbf{y}_k \in \mathbb{R}^p$ the system output at time k, $\mathbf{s}_k^y \in \mathbb{R}^p$ the reference output at time k, $\mathbf{s}_k^u \in \mathbb{R}^m$ is the reference input at time k, $N \in \mathbb{Z}^+$ the length of the prediction horizon and $\mathbf{A}_d \in \mathbb{R}^{n\times n}, \mathbf{B}_d \in \mathbb{R}^{n\times m}, \mathbf{C}_d \in \mathbb{R}^{p\times n}$are the discrete state, input and output transition matrices as defined in Equation18.

The cost function

$$J = \sum_{k=0}^{N-1} \left(\left\| \mathbf{Q}_{k+1}(\mathbf{y}_{k+1} - \mathbf{s}_{k+1}^y) \right\|_2^2 + \left\| \mathbf{R}_k(\mathbf{u}_k - \mathbf{s}_k^u) \right\|_2^2 \right) \tag{20}$$

is the quadratic penalty function on the states and inputs commonly used in optimal control, where the input and output gain matrices $\mathbf{R}_k \in \mathbb{R}^{m\times m}$ and $\mathbf{Q}_k \in \mathbb{R}^{n\times n}$ are tuning parameters. By concatenating the two squared 2-norms that appear in the cost function J, we can rewrite it as:

$$J = \left\| \begin{array}{c} \mathbf{Q}_1(\mathbf{y}_1 - \mathbf{s}_1^y) \\ \mathbf{Q}_2(\mathbf{y}_2 - \mathbf{s}_2^y) \\ \vdots \\ \mathbf{Q}_N(\mathbf{y}_N - \mathbf{s}_N^y) \\ \mathbf{R}_0(\mathbf{u}_0 - \mathbf{s}_0^u) \\ \mathbf{R}_1(\mathbf{u}_1 - \mathbf{s}_1^u) \\ \vdots \\ \mathbf{R}_{N-1}(\mathbf{u}_{N-1} - \mathbf{s}_{N-1}^u) \end{array} \right\|_2^2 \tag{21}$$

and by assigning $\bar{\mathbf{y}} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N]^T$, $\bar{\mathbf{u}} = [\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{N-1}]^T$, $\bar{\mathbf{s}}^y = [\mathbf{s}_1^y, \mathbf{s}_2^y, \cdots, \mathbf{s}_N^y]^T$, $\bar{\mathbf{s}}^u = [\mathbf{s}_0^u, \mathbf{s}_1^u, \cdots, \mathbf{s}_{N-1}^u]^T$, $\bar{\mathbf{Q}} = \mathrm{diag}(\mathbf{Q}_1, \mathbf{Q}_2, \cdots, \mathbf{Q}_N)$ and $\bar{\mathbf{R}} = \mathrm{diag}(\mathbf{R}_0, \mathbf{R}_1, \cdots, \mathbf{R}_{N-1})$ the original optimization problem, takes the following form:

$$\begin{aligned} \bar{\mathbf{u}}^* = & \underset{\mathbf{u}_0 \cdots \mathbf{u}_{N-1}}{\mathrm{argmin}} \left\| \begin{array}{c} \bar{\mathbf{Q}}(\bar{\mathbf{y}} - \bar{\mathbf{s}}^y) \\ \bar{\mathbf{R}}(\bar{\mathbf{u}} - \bar{\mathbf{s}}^u) \end{array} \right\|_2^2, \\ & \text{s.t.} : \mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k, \\ & \mathbf{y}_k = \mathbf{C}_d \mathbf{x}_k, \\ & \mathbf{x}_0 = \hat{\mathbf{x}}_0, \\ & \bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}. \end{aligned} \tag{22}$$

We can eliminate the model equality constraints from 22 by substituting $\bar{\mathbf{y}} = \bar{\mathbf{C}}\bar{\mathbf{x}} = \bar{\mathbf{C}}\boldsymbol{\Phi}\hat{\mathbf{x}} + \bar{\mathbf{C}}\boldsymbol{\Gamma}\bar{\mathbf{u}}$ into the cost function J, where $\bar{\mathbf{x}} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^T$ and the matrices $\boldsymbol{\Phi} \in \mathbb{R}^{Nn\times n}$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{Nn\times Nm}$ can be obtained by applying the equations $\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k$ and $\mathbf{y}_k = \mathbf{C}_d\mathbf{x}_k$ to every element of $\bar{\mathbf{x}}$. Specifically, these are given by:

$$\boldsymbol{\Phi} = \begin{pmatrix} \mathbf{A}_d \\ \mathbf{A}_d^2 \\ \vdots \\ \mathbf{A}_d^N \end{pmatrix}, \ \boldsymbol{\Gamma} = \begin{pmatrix} \mathbf{B}_d & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}_d\mathbf{B}_d & \mathbf{B}_d & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_d^{N-1}\mathbf{B}_d & \mathbf{A}_d^{N-2}\mathbf{B}_d & \cdots & \mathbf{B}_d \end{pmatrix}. \tag{23}$$

The optimization problem is transformed into the following equivalent QP with inequality constraints:

$$\begin{aligned} \bar{\mathbf{u}}^* = & \underset{\bar{\mathbf{u}}}{\mathrm{argmin}} \ \bar{\mathbf{u}}^T \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\boldsymbol{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix}^T \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\boldsymbol{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix} \bar{\mathbf{u}} - 2 \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{s}}^y - \bar{\mathbf{Q}}\bar{\mathbf{C}}\boldsymbol{\Phi}\hat{\mathbf{x}}_0 \\ \bar{\mathbf{R}}\bar{\mathbf{s}}^u \end{pmatrix}^T \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\boldsymbol{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix} \bar{\mathbf{u}} \\ & \text{s.t.} : \bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}. \end{aligned} \tag{24}$$

In order to ensure operation within a safe state envelope, it is common in MPC to impose additional state constraints. These can be modeled as hard constraints similar to the input constraints $\bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}$. In this case the optimization solver may face an infeasible problem since the set of admissible points as defined by the problem constraints is empty e.g. when a large disturbance has occurred or when the real and the estimated model used for control behave differently.

In order to avoid the case of infeasibility we model the state constraints $\mathbf{G}\bar{\mathbf{x}} \leq \mathbf{h}$, with $\mathbf{G} \in \mathbb{R}^{Nl\times Nn}$ as soft constraints which can be violated if necessary. We incorporate them into the cost function J following a similar approach discussed in (Maciejowski, 2002) and (Kerrigan and Maciejowski, 2000).

The modified cost function J_m is:

$$J_m = J + \lambda \mathbf{1}^{\mathbf{T}} (\mathbf{G}\bar{\mathbf{x}} - \mathbf{h})_+. \tag{25}$$

The subscript + implies that $(\mathbf{G\bar{x}} - \mathbf{h})_+ = \mathbf{G\bar{x}} - \mathbf{h}$ when $\mathbf{G\bar{x}} - \mathbf{h} \geq \mathbf{0}$ and $\mathbf{0}$ otherwise. Overall, the term $\mathbf{1^T}(\mathbf{G\bar{x}} - \mathbf{h})_+$ is the sum of constraints violation while the gain $\lambda \in \mathbb{R}$ is large enough in order to ensure that the modified optimization problem with soft constraints, when none of the constraints is active, is equivalent to the optimization problem where the state constraints are modeled as hard.

The optimization problem with the soft state constraints can be rewritten as the following equivalent QP by introducing the slack variables $\mathbf{s}$.

$$\begin{aligned} \bar{\mathbf{u}}^*, \mathbf{s}^* &= \underset{\bar{\mathbf{u}}, \mathbf{s}}{\operatorname{argmin}} \, J + \lambda \mathbf{1}^T \mathbf{s} \\ \text{s.t. :} \; & \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} \\ \mathbf{G\Gamma} & -\mathbf{I} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \bar{\mathbf{u}} \\ \mathbf{s} \end{pmatrix} \leq \begin{pmatrix} \bar{\mathbf{u}}_{max} \\ -\bar{\mathbf{u}}_{min} \\ \mathbf{h} - \mathbf{G\Phi\hat{x}}_0 \\ \mathbf{0} \end{pmatrix} \end{aligned} \tag{26}$$

By introducing $\mathbf{t} = [\bar{\mathbf{u}}, \mathbf{s}]^T \in \mathbb{R}^{N(l+m)}$ the above QP is written in its canonical form as:

$$\begin{aligned} \mathbf{t}^* = \underset{\mathbf{t}}{\operatorname{argmin}} \; & \mathbf{t}^\top \begin{bmatrix} \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix}^T \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Gamma} \\ \bar{\mathbf{R}} \end{pmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{t} + \begin{bmatrix} -2 \begin{pmatrix} \bar{\mathbf{Q}}\bar{\mathbf{s}}^y - \bar{\mathbf{Q}}\bar{\mathbf{C}}\mathbf{\Phi\hat{x}}_0 \\ \bar{\mathbf{R}}\bar{\mathbf{s}}^u \end{pmatrix}^T & \lambda \mathbf{1}^\top \end{bmatrix} \mathbf{t}, \\ \text{s.t. :} \; & \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} \\ \mathbf{G\Gamma} & -\mathbf{I} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix} \mathbf{t} \leq \begin{pmatrix} \bar{\mathbf{u}}_{max} \\ -\bar{\mathbf{u}}_{min} \\ \mathbf{h} - \mathbf{G\Phi\hat{x}}_0 \\ \mathbf{0} \end{pmatrix}. \end{aligned} \tag{27}$$

The above QP can be solved in realtime using any generic QP solver. We use CVXGEN (Mattingley and Boyd, 2012) which generates a tailored to the specific problem interior-point based C code and in practice was the fastest QP solver tested. According to the authors of the optimization toolbox, nearly all the computational effort in each iteration stems from the solution of two linear systems resulting in a worst case computation complexity of $\mathcal{O}(n^3)$ where $n = N(l+m)$ corresponds to the number of optimization variables in the final QP problem and N, l and m to the length of the prediction horizon, the number of soft constraints and the number of inputs, respectively, as introduced above.

The mean computation delay of the QP solver for all the experiments presented in the Section 7 was 709.53 μs with a standard deviation $\sigma = 69.85$ μs. The hardware used consists of an Intel NUC i7-7567U with 16 Gb RAM and Ubuntu Server 16.04. Figure 7 shows the computation delay data for all the experiments conducted. In all the experiments the worst case delay is lower than 1.0 ms which means that we could run the controller at a higher update rate than the current 20 Hz rate.

6.3 Position and velocity reference generation

Although our controller can natively handle a reference input that is time varying over the prediction horizon $\bar{\mathbf{s}}^u = [\mathbf{s}_0^u, \mathbf{s}_1^u, \cdots, \mathbf{s}_{N-1}^u]^T$ and output $\bar{\mathbf{s}}^y = [\mathbf{s}_1^y, \mathbf{s}_2^y, \cdots, \mathbf{s}_N^y]^T$, for simplification, we do not explicitly generate a time varying reference but a static one which remains constant over the prediction horizon.

The approach of position setpoint instead of trajectory control is sufficient for waypoint navigation but insufficient when tracking of a moving target is required. Generating a sequence of reference position setpoints based on the observed position of the moving target will result a constant offset, similar to the case of PD control, between the position of the MAV and the target. This is the result of the zero velocity requirement at each position setpoint which does not hold in the case of a moving target. We tackle this problem by not only penalizing deviation from a reference position but also from a reference velocity which in this case corresponds to the estimated velocity of the target. It is also worth mentioning that in our implementation, the prediction horizon is relatively short 0.1 s. Similarly, we can penalize deviation from a reference acceleration by incorporating it in $\bar{\mathbf{s}}^u$.

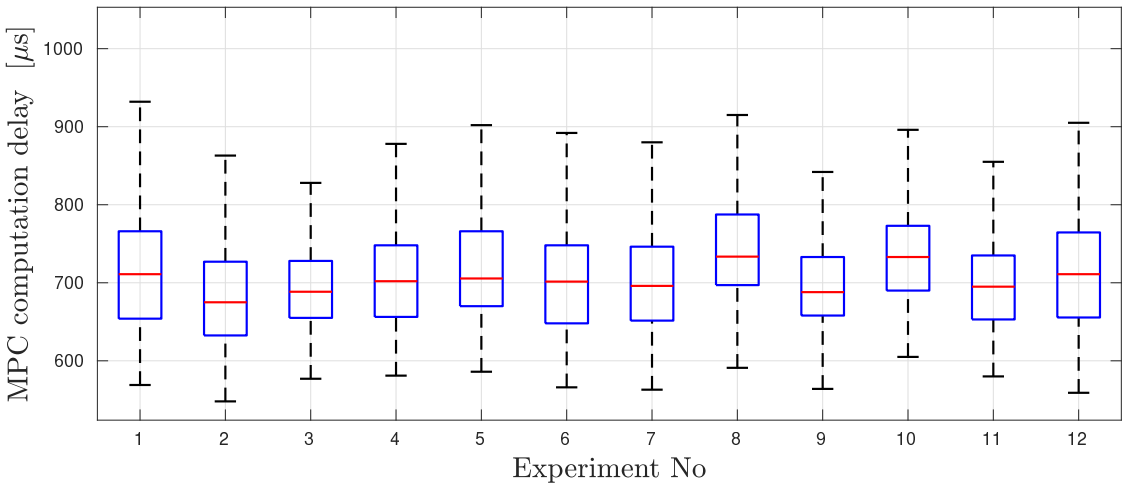

Figure 7: Box plot with the MPC computation delay data. In all the experiments the worst case delay is less than 1.0 ms showing that we could run the controller at a much higher rate than the current 20 Hz rate.

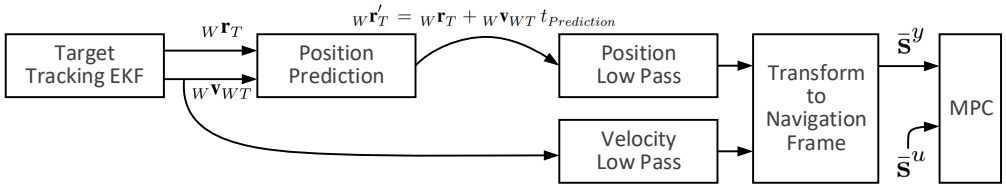

Figure 8: The reference generation scheme. The incorporation of the reference velocities and not only the position, improves the overall tracking performance of the system. The low pass filters for the position and velocity are used in order to prevent jittery MAV motion due to noisy target detections. The target position prediction by the $t_{\text{Prediction}}$ time offset, is performed to counteract significant unmodeled delays that exist in the system.

The scheme for the generation of the commanded setpoint(which includes both reference position and velocity) is illustrated in Figure 8. For simpler tasks such as waypoint navigation we follow the standard approach where position commands with zero reference velocities are sent to the controller.

We use low pass filters for the reference position and velocity with cutoff frequencies $f_{\text{CutOffPosition}}$, $f_{\text{CutOffVelocity}}$ in order to avoid passing jittery commands to the MAV due to noisy target position and velocity estimates. The target position prediction by the time offset $t_{\text{Prediction}}$, assuming constant velocity, is used in order to counteract the delay between a successful target detection and the actual maneuver of the MAV. The use of such a time offset was also motivated to compensate significant unmodeled delays that exist in the system such as the one related to the downward looking camera vision processing which arises from the lack of its hardware synchronization. Notice that the prediction of the target position also depends on its estimated velocity, while the time offset $t_{\text{Prediction}}$ was a constant parameter determined by tuning of the whole system (before conducting the series of reported experiments). Its numeric value is given – along with all the other parameters – in Table 2.

6.4 High level mission profile

The overall behavior of the MAV is controlled by a state machine with the following operation modes which in an ideal experiment, are triggered sequentially.

Idle Mode: This is the default mode when the mission is initially triggered. The motors are disarmed and a zero thrust and orientation command $T^r = 0$, $\mathbf{q}^r_{WB} = [1, 0, 0, 0]^T$ is sent to the MAV.

Taking Off Mode: This mode is triggered once the first valid state estimation message becomes available. An arming command is sent to the motors and the MAV takes off to the predefined altitude $z_{\text{TakeOffHeight}}$ with a predefined ascending velocity $\dot{z}_{\text{TakeOffVelocity}}$.

Waypoint Mode: Immediately after take off, the MAV flies towards a predefined waypoint which coincides with the cross point of the figure-eight trajectory that the ground vehicle was following in the MBZIRC challenge.

Follow Mode: This mode is triggered when the landing pattern is successfully detected for the first time. A position and velocity command based on the procedure described in 6.3 is sent to the controller. When the difference between the current time $t_{\text{CurrentTime}}$ and the time of the last target detection $t_{\text{LastDetection}}$ is greater than the timeout parameter $t_{\text{DetectionTimeOut}}$, we assume that the target is not visible anymore and return back to the waypoint mode. The $t_{\text{DetectionTimeOut}}$ has to be set high enough to allow the MAV to move close to where the target was detected but low enough to enable fast recovery to the waypoint in the case of an actual detection loss.

Landing Phase 1: Once the MAV has properly caught up the moving target, it starts descending towards it with a predefined velocity $\dot{z}_{\text{DescendingP1}}$. This mode is triggered when $\sqrt{(x_B - x_T)^2 + (y_B - y_T)^2} \leq d_{\text{PositionThreshold}}$ and $\sqrt{(\dot{x}_B - \dot{x}_T)^2 + (\dot{y}_B - \dot{y}_T)^2} \leq d_{\text{VelocityThreshold}}$ where x_B, x_T, y_B, y_T stands for the x, y position and velocity, expressed in the $\underrightarrow{\mathcal{F}}_W$, of the MAV and the target respectively. Similarly to the follow mode, if $t_{\text{CurrentTime}} - t_{\text{LastDetection}} \geq t_{\text{DetectionTimeOut}}$ we consider that the target is not visible anymore and return back to the waypoint mode.

Landing Phase 2: This is similar to the Landing Phase 1 mode apart from the facts that the descending velocity is now $\dot{z}_{\text{DescendingP2}}$ and that the target detection timeout check is ignored. This means that once the MAV enters this mode when $z_B - z_T \leq z_{\text{LandingP2Offset}}$ – where z_B, z_T are the z position expressed in the $\underrightarrow{\mathcal{F}}_W$ of the MAV and the target – a landing that cannot be canceled will be attempted. The above feature is necessary since reliable target tracking cannot be guaranteed when the MAV is really close to the target (due to partial visibility, blur, etc.).

Landed: This is the last operation mode which is triggered when $z_B - z_T \leq z_{\text{SuccessfulLandingOffset}}$. After successful landing a disarming command is sent to the motors and the mission has ended.

Figure 9 shows a flowchart with the described operating modes including the strategies in the case of target detection loss.

7 Field experiments

In order to validate the performance of our platform and the developed algorithms, we performed a series of outdoor experiments replicating a similar to the MBZIRC Challenge 1 scenario. Note that we performed these experiments *after* our participation in MBZIRC, where the preliminary version of the described system proved too unreliable, specifically in the extreme wind conditions, and triggered us to improve several aspects of hardware, algorithms, and software.

The experimental flow is as follows: The MAV takes off from the starting point location where it is initially placed on a 45cm height box. This is necessary in order to make sure that OKVIS can detect and track salient points on the RGB image. Once take off is completed, the MAV navigates to the predefined waypoint where it hovers till the target gets detected. After successful detection, the MAV tries to follow the target and land on it. The transition between the different operating modes is the one described in Section 6.4. The MAV executes the whole task autonomously and does not use any user provided information about the

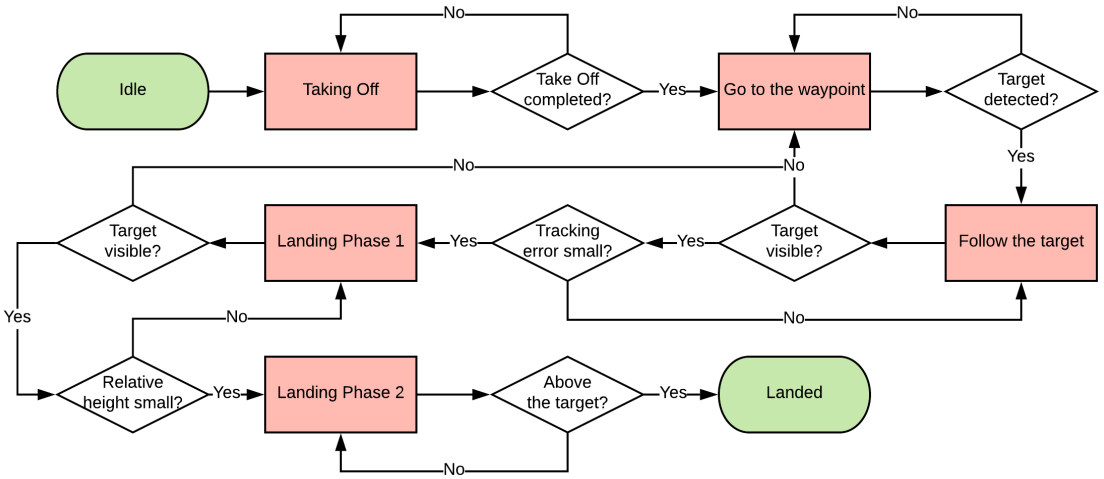

Figure 9: Flowchart indicating the transition between the various operating modes.

target position and velocity but it exclusively relies on its onboard sensors and algorithms. A human pilot triggers the start of the mission and can also manually intervene in order to prevent a catastrophic crash in case of an algorithmic failure. We annotate every experiment as *Successful* or *Failed* based on whether the MAV managed to land successfully on the target according to the MBZIRC Challenge 1 specifications [11]. Specifically, a successful landing is when the MAV comes to a rest on the landing target, with the MAV intact.

The landing target is identical to the one used in the MBZIRC Challenge 1 and was being pulled manually, in a random way, with a rope. We tested moving the target with velocities between 1.3 m/s and 5.7 m/s which exceeds the maximum target velocity of 4.1 m/s during the MBZIRC Challenge 1. Table 1 contains the results of the conducted experiments while Table 2 contains a list for all the parameters used in the developed algorithms. Alongside the outcome (*Successful*/*Failed*) for each experiment, Table 1 contains the norm of the position error $e_{\text{xy}} = \sqrt{(x_B - x_T)^2 + (y_B - y_T)^2}$ between the MAV and target position when the MAV has landed and the time $t_{\text{TrackingDuration}}$ which corresponds to the time that was needed for a successful landing from the time instant where the target was initially detected.

Since no motion capture system or RTK GPS was used, we cannot provide ground truth for the MAV and target position. All the data presented in this section corresponds to the estimates from OKVIS and the target tracking EKF.

Table 1: Successful/unsuccessful landings

Experiment No	1	2	3	4	5	6	7	8	9	10	11	12
Average Velocity $\bar{v}$ $[m/s]$	1.3	1.7	2.0	2.0	2.3	2.4	3.5	3.8	4.0	4.6	5.0	5 .7
Position error e_{xy} [cm]	2.5	11.8	5.2	30.6	16.5	6.6	30.2	18.6	15.6	29.8	45.8	7.6
Tracking Duration [s]	5.06	4.78	5.55	7.65	7.12	5.64	4.97	5.91	7.47	6.09	5.59	4.97
Experiment Outcome*	*S*	*S*	*S*	*S*	*S*	*S*	*S*	*S*	*S*	*F*	*S*	*F*

* *S* = *Successful*, *F* = *Failed*.

[11] See `https://www.mbzirc.com/faqs/2017`.

Table 2: Parameters

$f_{\text{CutOffPosition}}$	30 hz	$z_{\text{TakeOffHeight}}$	5.2 m	$\dot{z}_{\text{TakeOffVelocity}}$	0.5 m/s
$f_{\text{CutOffVelocity}}$	30 hz	$d_{\text{PositionThreshold}}$	0.8 m	$d_{\text{VelocityThreshold}}$	0.8 m/s
$t_{\text{Prediction}}$	0.1 s	$z_{\text{LandingP2Offset}}$	2.5 m	$\dot{z}_{\text{DescendingP1}}$	1.3 m/s
$t_{\text{DetectionTimeOut}}$	0.5 s	$z_{\text{SuccessfulLandingOffset}}$	0.35 m	$\dot{z}_{\text{DescendingP2}}$	1.8 m/s

7.1 Experimental results

In this section, we present the MAV, the target position and attitude data for three selected experiments. Specifically, two of them (Figures 10, 12 and Figures 11, 13) were successful (one with low target velocity and another with high speed), whereas the third one (Figures 14, 15) was unsuccessful. The figures for the remaining nine experiments are included in the Appendix. Also, a video showing the successful experiments can be found here[12]. In all the figures presented, solid lines refer to the estimated values given by OKVIS while the dashed ones refer to their corresponding reference. Reference position x^r, y^r, z^r was generated as described in Section 6.3 while θ^r, ϕ^r and T^r were obtained by solving the MPC optimization problem. ψ^r was always kept to zero. Also, notice that there is no solid line for T^r (plotted in Newtons) since – as explained in Section 6.1 – the dynamics of the motors were ignored and it was assumed that the applied thrust T coincides with the reference T^r.

8 Discussion

From the experimental results, we realize that our hardware and software framework achieves the original goal that it was designed for with high repeatability. The majority of the experiments (10/12) were successful whereas there were only 2 failed attempts, both in the high velocity region (4.6 m/s and 5.7 m/s). The worst case position error e_{xy} between the target and the MAV after landing was 45.8 cm which –not surprisingly– corresponds to the fastest successful attempt No 11. The reference tracking and landing accuracy achieved by the controller were more than sufficient especially considering the size of the MAV and the landing target and the fact that the experiments were conducted in an outdoor environment. The position error is generally larger in the high velocity experiments (No 7, 8, 9, 11) and lower in the slower ones (No 1, 2, 3, 4, 5, 6). We consider the noisy and uncontrolled outdoor environment (presence of wind and varying lighting conditions, which affect the target state estimation) to be the main reason.

Concerning MAV state estimation, with appropriate settings of gain and brightness on the Realsense RGB-D camera, we have not experienced any problems related to lack of tracked keypoints, therefore leaving the visual-inertial estimation system working flawlessly.

Regarding the attitude commands generated by the MPC, these are generally not smooth in the high velocity Experiments but smoother for the low velocity ones. This was the result of over-penalizing the position/velocity error compared to the penalization of the smoothness terms. In other words, the gains of the optimization problem are chosen in a way such that position/velocity tracking is prioritized over smooth attitude changes. Regarding the MAV response to the attitude commands, there is an amplitude difference and a phase delay between the commanded attitude angles and the real ones. Since the linear model also includes an estimated model for the closed loop attitude dynamics (Equation 12), this difference between a command and the actual response is taken into account by the MPC. We also observe that although the ψ^r was always kept to zero, ψ was minimally affected when aggressive θ^r, ϕ^r were required. This is the result of existence of coupling between the MAV degrees of freedom; something not captured by the linear model used for control.

It is important to highlight that our controller does not contain any integral error term and generally it

[12] See `https://youtu.be/PbYXuLkxEPM`.

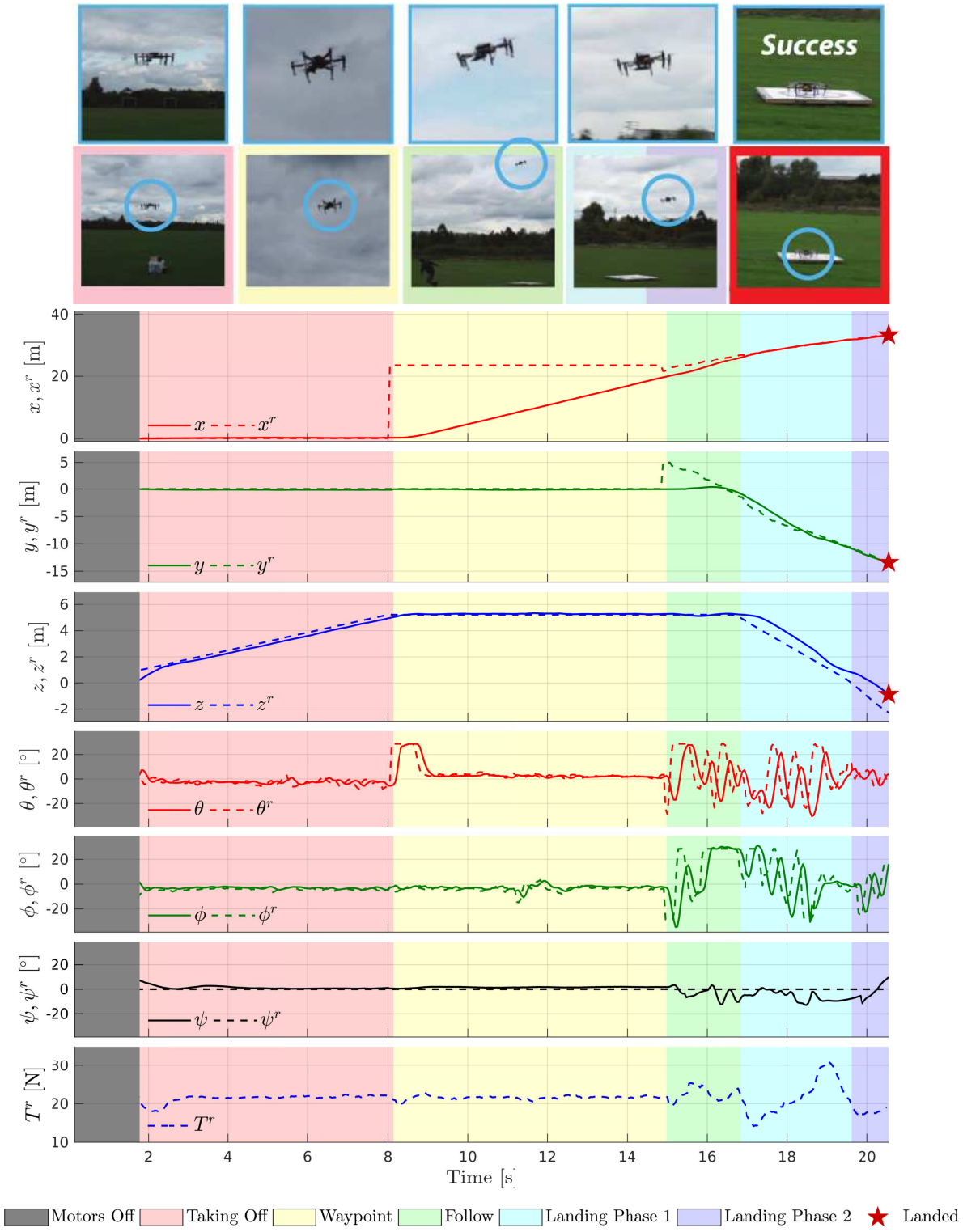

Figure 10: Position and attitude data for Experiment 11. A *successful* landing on a target moving with 5.0 m/s was achieved. The dashed lines on the upper 3 plots correspond to the reference position which is generated based on the estimated position and velocity of the target as described in Section 6.3. Dashed lines on the lower 3 plots correspond to the attitude angles and thrust generated by the MPC. The MAV lands 5.59 s after the first target detection and 45.8 cm away from the target center.

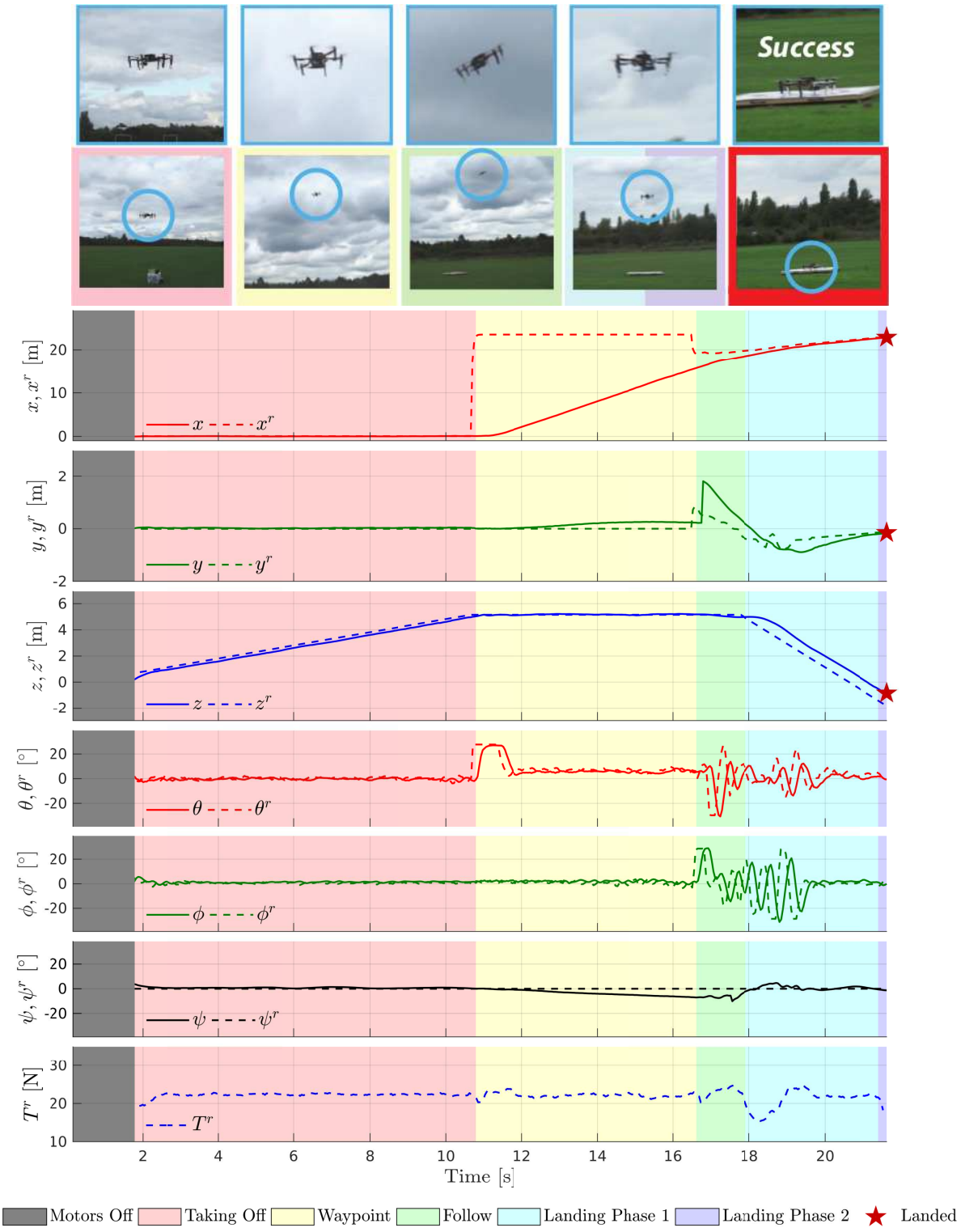

Figure 11: Position and attitude data for Experiment 1. A *successful* landing was achieved on a target moving with 1.3 m/s. The MAV lands 5.06 s after the first target detection and 2.5 cm away from the target center.

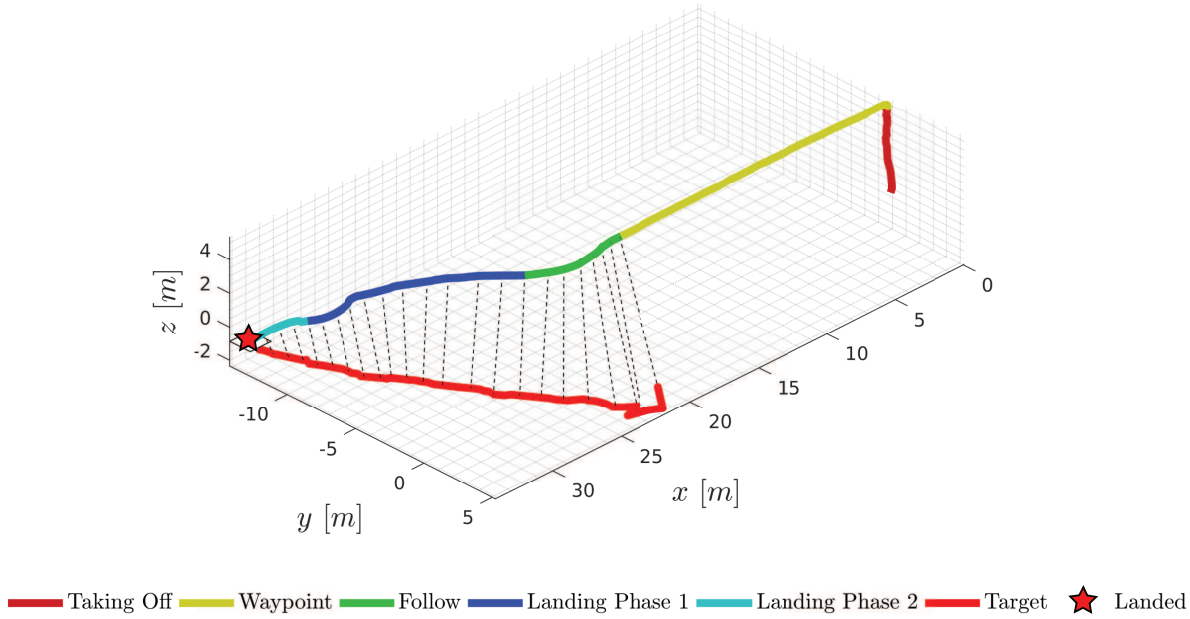

Figure 12: The MAV and target position (visualized as a 3D plot) for the Experiment 11. Dashed lines indicate the corresponding target detection/tracking.

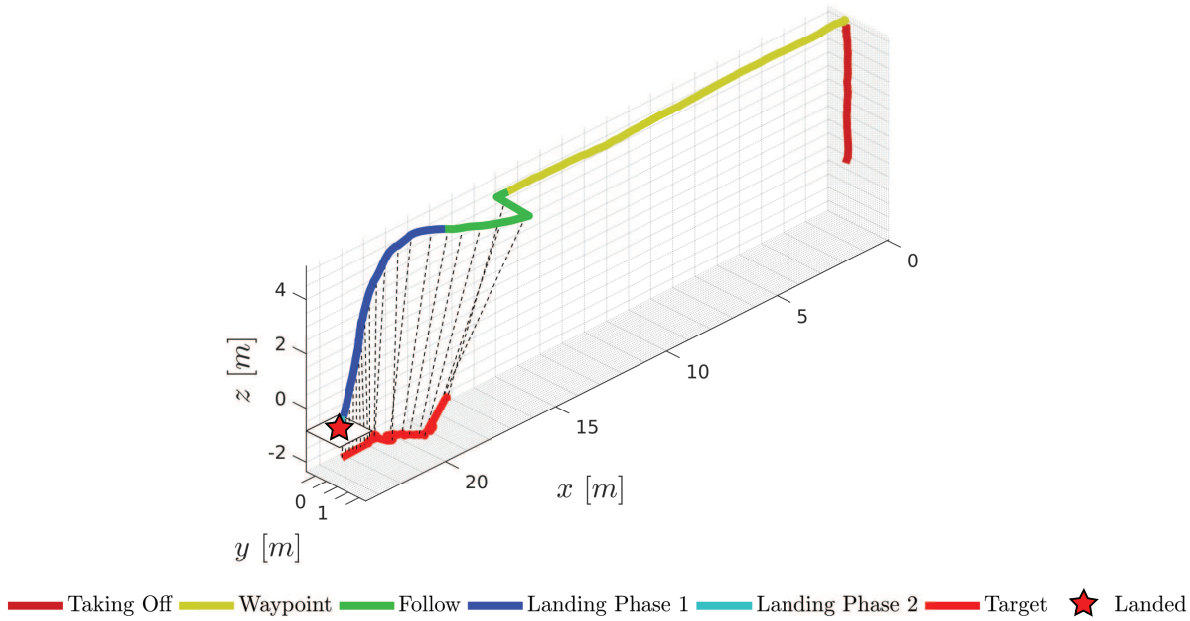

Figure 13: The MAV and target position in 3D for the Experiment 1.

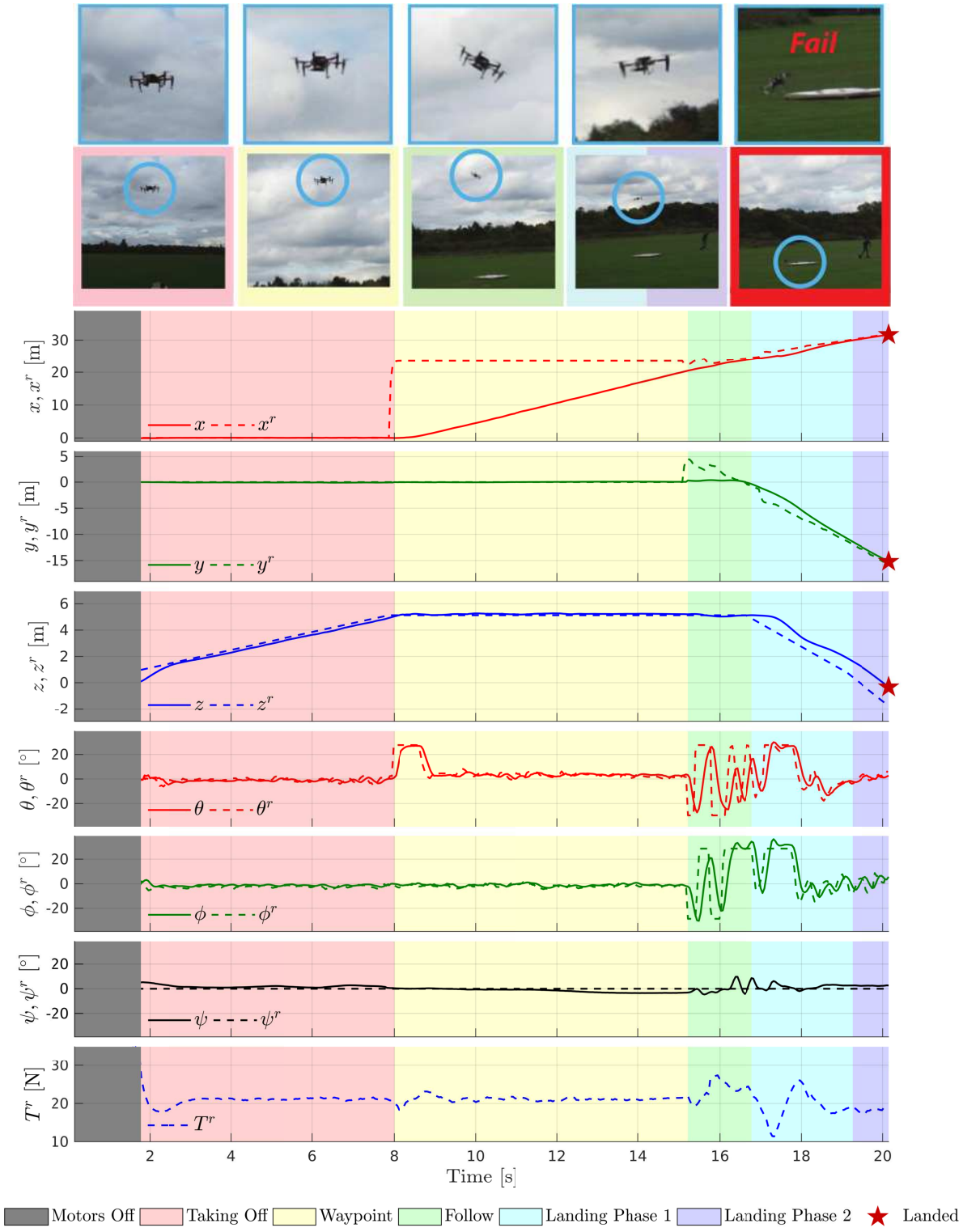

Figure 14: Position and attitude data for Experiment 12. A *failed* attempt of landing on a target moving with 5.7 m/s was performed. Although that the final position error e_{xy} between the target and the MAV is only 7.6 cm, the real MAV position differs from the target position (See also Figure 16). Unreliable estimation of the target position especially when flying close to it led to a landing on a predicted target position different from the real one.

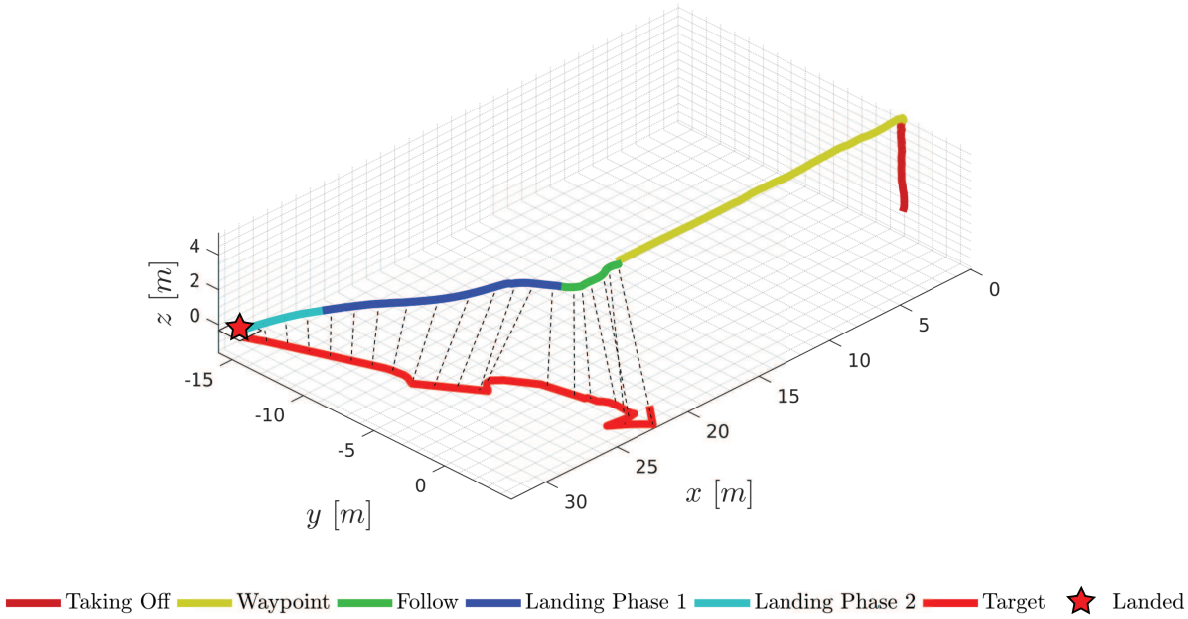

Figure 15: The MAV and target position in 3D for the Experiment 12.

is not aware of any not modeled disturbance acting on the system. Hence, it is unable to counteract any constant position/velocity error offset which is either the result of an external disturbance or imperfect system calibration. The controller can only compensate non-persistent external disturbances due to the feedback on the system. Rejection of constant disturbances would only possible, if these could be estimated and taken into account from the control model (linear/non linear), which we leave to future work. In order to make sure that the MAV will remain in a state envelope where the used linear model remains valid, we use appropriate constraints in the optimization problem. These also guarantee that the MAV operates far from the gimbal lock (since Euler angles are used for the control formulation).

In all the successful experiments, the MAV managed to land within less than 7.65 s after the first target detection. The final time in all the experiments did not exceed 30 s, and this is mainly dominated by the time required for take off and moving to the waypoint. We cannot make a direct comparison with the performance of the teams who participated in the real challenge, as the final time also depends on when the target becomes visible for the first time. We can, however, compare our approach with the ones by team ETH-MAV(Bähnemann et al., 2017) and NimbRo (Beul et al., 2017) which both successfully landed on the moving target during the MBZIRC competition within 56 s and 32 s, respectively.

- ETH-MAV team used two cascaded EKFs that combine data from an RTK GPS and a Visual Inertial sensor for the MAV state estimation. A downward looking camera is used for the target detection. Target tracking and landing is achieved through a non-linear MPC and a path planning algorithm which also relies on the prior knowledge of the MBZIRC track shape. They also use a Lidar sensor in order to check the distance between the MAV and the target and trigger the final stage of the landing procedure.
- NimbRo team used a GPS + IMU fusion algorithm for the MAV state estimation, along with a fast trajectory planning algorithm. In order to detect and track the target they use two different cameras. One is looking downwards, while the other forward-downward. The usage of such a camera configuration, not only robustifies the target detection but also enables faster initial detection of the moving target. Their MAV was programmed to wait at the waypoint while rotating around its vertical axis at the same time. Mechanical switches are placed on the legs of the MAV in order to detect contact and thus successful landing on the moving target.

Both teams followed a mission profile similar to ours with *Wait at waypoint – Follow – Final approach and landing* to be the main modes of operation, while the main difference was the active rotating/searching performed by the NimbRo team MAV which can greatly reduce the time needed for initial target detection. Our approach does not rely on RTK-GPS making it appropriate for similar tasks in GPS denied environments, and moreover removes the necessity for a ground station (which wirelessly sends the GPS corrections to the MAV). We consider our approach to be simple and general enough to be implemented in similar scenarios. Regarding the software developed, we consider the lack of dynamically feasible trajectories planning and the usage of a linear MPC to be the main limitation of our approach compared to the above. Regarding the hardware, we used – in our opinion – the absolute minimum number of sensors/components required for autonomous navigation. Since we employ only the necessary hardware components, we don't get the benefits of more reliable target detection (through use of multiple cameras) and prevention of unsuccessful landing attempts (through use of extra sensors such as Lidar).

As far as the failed attempts (Experiment 10 and 12) are concerned, both of them seem successful when we look at the position plots and the final position error after landing. As shown in Figure 16 in both Experiment 10 and 12 the MAV landed really close to the true target position. We argue that the reason behind the failed attempts is the quality of the estimated target state especially when the MAV flies really close to the target. Specifically, in Experiment 12 (Figure 14) the target was pulled sharply in a different direction while the MAV was in Landing Phase 2. Due to the proximity between the MAV and target, the latter was partially or completely invisible due to the horizontally mounted downward looking camera and MAV tilt. Given the average target velocity in these experiments (4.6 m/s and 5.7 m/s) and the fact that the frame rate of the downward looking camera was 20 hz, the target position between two consecutive camera frames was changing by 23 cm and 28.5 cm respectively. It becomes clear that in this frame rate and high target velocity, even one frame of lost tracking can result a failed landing attempt.

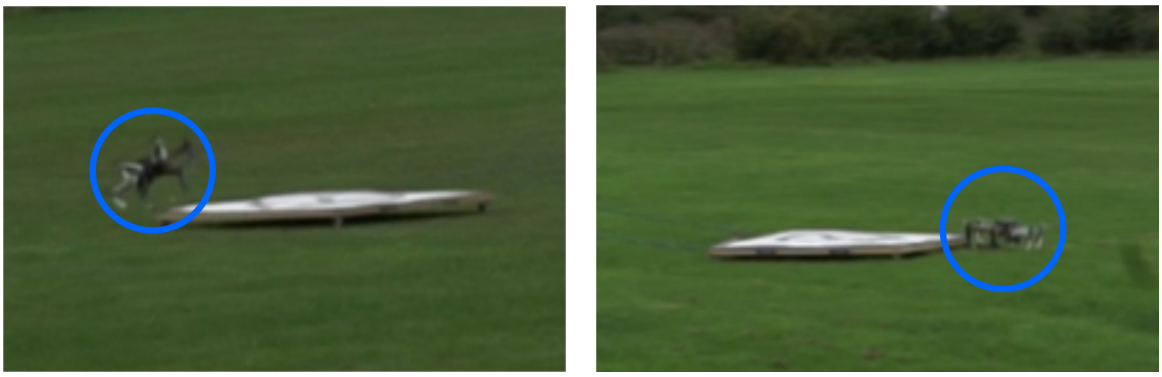

Figure 16: The two *failed* landing attempts. Experiment 12 on the left and Experiment 10 on the right.

Since all the experiments were completed outdoors, we cannot easily provide accurate ground truth for the target position, but instead we assess the distance from the target once the MAV has landed. We consider that our approach has two main disadvantages. The first one is unreliable target tracking when the MAV flies close above the target. This could by either fixed by improving the target tracking when it is partially visible or by mechanically rotating the downward looking camera such that the target always remains in the field of view. The second disadvantage is the criterion used for triggering the landing attempt when flying above the MAV. This depends only on the relative height between the MAV and the target. It would be better to use a criterion which takes into account the confidence/uncertainty of the estimated target states or use additional hardware (e.g. Lidar sensor) for more accurate relative height estimation. This would hopefully lead to higher probability for a successful landing. Nevertheless, we believe to have achieved a respectable success rate given the minimal hardware/sensor complexity.

9 Conclusion

In this paper, we presented the hardware and software components that were originally developed for our participation in the MBZIRC Challenge 1. Our system was able to land on a moving target within a range of speed from 1.3 m/s to 5.0 m/s – which we have thoroughly evaluated in a suite of outdoor experiments presented in detail to the reader. Employing a carefully engineered visual-inertial odometry system, a 3D target tracking EKF and MPC position control, we get reasonable tracking accuracy by appropriately penalizing deviation from a reference position and velocity. The tightly coupled visual-inertial state estimation proved to be extremely robust while the lack of a GPS sensor in the state estimation pipeline makes it adequate for indoor or generally GPS denied environments. The whole task was executed fully autonomously, while all the algorithms were executed on the onboard computer with input data exclusively provided by the onboard sensors.

In future we aim to improve the tracking performance of the MAV when aggressive maneuvers are commanded by generating feasible trajectories (based on MAV dynamics) and trajectory control. We aim to extend the current approach and utilize a non-linear model which captures more accurate dynamics of the MAV. We can achieve offset-free position control when persistent disturbances such as wind gusts act on the MAV by properly estimating them online and including their effects in the MAV model. Finally, we have to refine our target tracking algorithms but also the overall mission strategy in order to improve the robustness of the system and enable successful landing in more challenging conditions.

Acknowledgements

This research is supported by the EPSRC grant Aerial ABM EP/N018494/1 and Imperial College London.

A Target Tracker EKF

In its discrete-time implementation with time increment Δt, the target tracking process model Equation (6) becomes

$$ {}_W\mathbf{r}_T{}^k = {}_W\mathbf{v}_{WT}{}^{k-1}\Delta t, \tag{28} $$

$$ \mathbf{q}_{WT}{}^k = \exp(\mathbf{n}_{\text{rot}}) \otimes \mathbf{q}_{WT}{}^{k-1}, \tag{29} $$

$$ {}_W\mathbf{v}_{WT}{}^k = {}_W\mathbf{v}_{WT}{}^{k-1} + \mathbf{n}_{\text{vel}}, \tag{30} $$

with the zero-mean independent Gaussian random variables $\mathbf{n}_{\text{rot}}$ and $\mathbf{n}_{\text{vel}}$ as a discrete-time approximation of the corresponding Gaussian white noise processes with variance vector $\sigma_{\text{vel}}\Delta t$, and $\sigma_{\text{rot}}\Delta t$, respectively. The corresponding linearisation $\mathbf{x}_{\text{T}}^k = \mathbf{F}_d\mathbf{x}_{\text{T}}^{k-1}$ becomes

$$ \mathbf{F}_d = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{I}_3\Delta t \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 \end{bmatrix}, \tag{31} $$

where $\mathbf{I}_3$ denotes a 3 by 3 Identity matrix, and $\mathbf{0}_{3\times3}$ denotes a 3 by 3 block of zeros.

As far as the observation of the t^{th} target keypoint is concerned, Equation (9), we need to consider the undistorted projection model of the downward looking camera, $\mathbf{u}_{\text{D}}({}_D\mathbf{r}_t)$, with argument ${}_D\mathbf{r}_t := [r_1, r_2, r_3]$:

$$ \mathbf{u}_{\text{D}}({}_D\mathbf{r}_t) = \begin{bmatrix} f_u\frac{r_1}{r_3} + c_u \\ f_v\frac{r_2}{r_3} + c_v \end{bmatrix}, \tag{32} $$

where f_u, f_v denote the focal lengths in pixels and c_u, c_v the image centre. Now, linearisation of the observation model (9) in the sense $\mathbf{H}_t := \partial h_t / \partial \mathbf{x}_{\mathrm{T}}$ becomes:

$$\mathbf{H}_t = \begin{bmatrix} \mathbf{U}_t \boldsymbol{T}_{DW} \, \mathbf{J}_t & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \end{bmatrix}, \tag{33}$$

with $\boldsymbol{T}_{DW}$ denoting the pose of the downward looking camera obtained from the Visual-Inertial pose estimation, and $\mathbf{U}_t$ the Jacobian of the undistorted fisheye camera projection

$$\mathbf{U}_t({}_D\mathbf{r}_t) := \frac{\partial}{\partial {}_D\mathbf{r}_t} \mathbf{u}_{\mathrm{D}}({}_D\mathbf{r}_t) = \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \mathbf{D} \begin{bmatrix} \frac{1}{r_3} & 0 & -\frac{1}{r_3^2} \\ 0 & \frac{1}{r_3} & -\frac{1}{r_3^2} \end{bmatrix}, \tag{34}$$

with the Jacobian of Radial-Tangential distortion, $\mathbf{D}$, and, further, with

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_3 & [\mathbf{C}_{WT} \, {}_T\mathbf{r}_t]^{\times} \end{bmatrix}, \tag{35}$$

with $[\cdot]^{\times}$ denoting the 3 by 3 cross-product matrix corresponding to its vector argument.

We have now presented all the expressions necessary to apply the well-known discrete-time EKF prediction equations, as well as the update equations, which we apply for all the visible points sequentially. Note that the update of the quaternion follows the multiplicative scheme ${\mathbf{q}_{WT}}^{+} = \exp(\Delta\boldsymbol{\alpha}_{WT}) \otimes {\mathbf{q}_{WT}}^{-}$.

B Supplementary plots of flight experiments

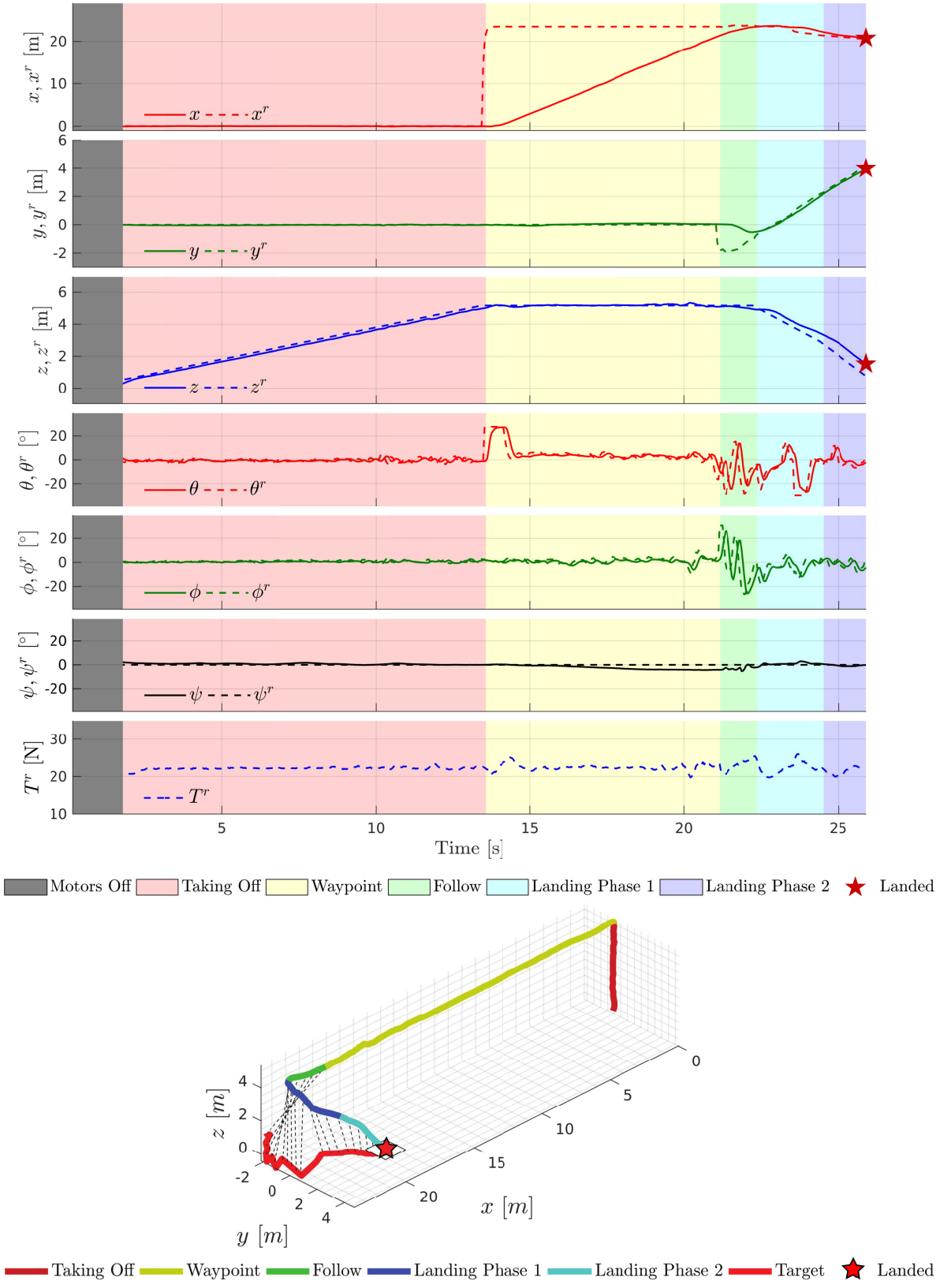

Figure 17: Position and attitude data for Experiment 2. A *successful* landing was achieved on a moving target with 1.7 m/s. The MAV landed 4.78 s after the first target detection and 11.8 cm from the target center.

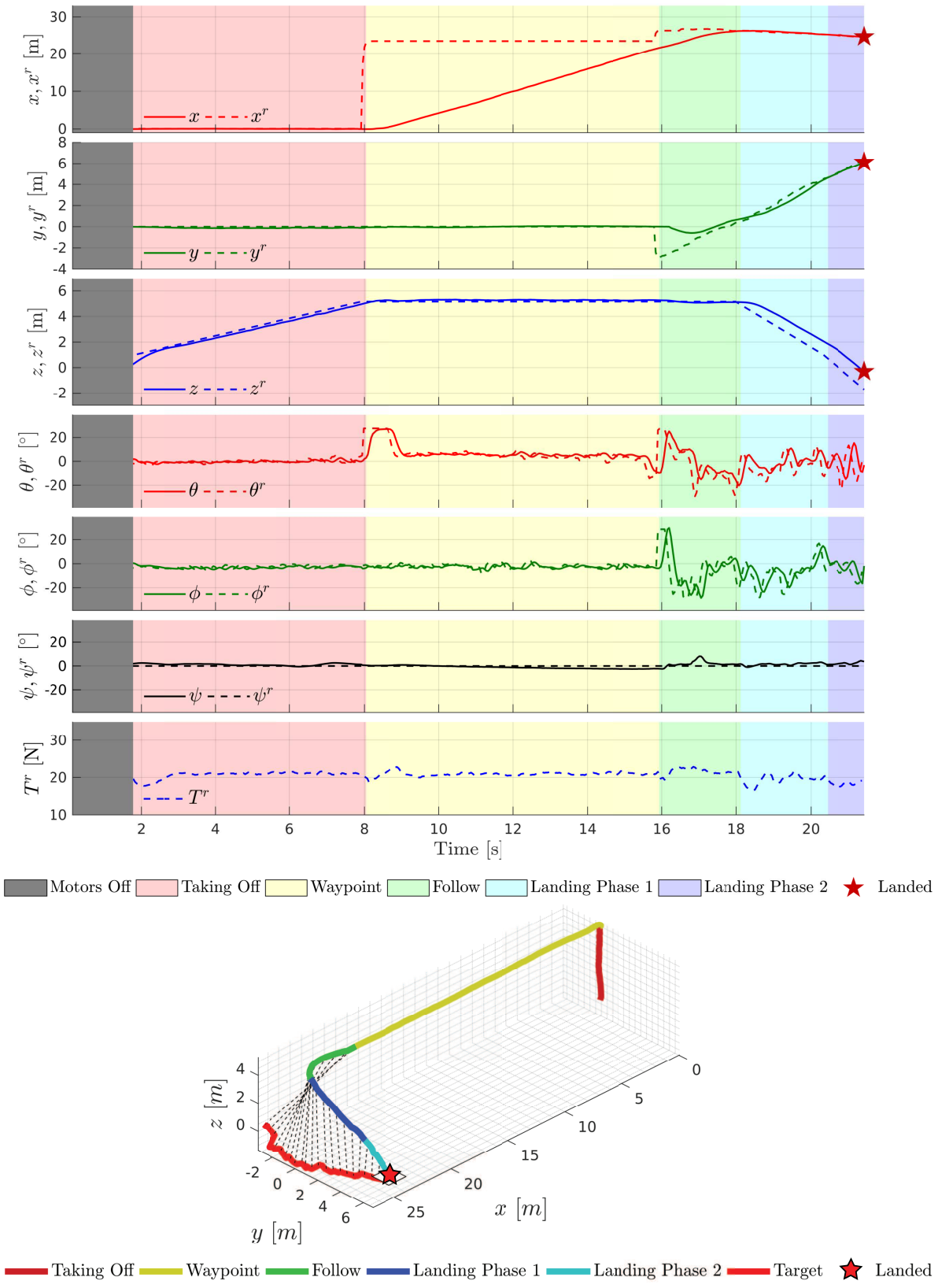

Figure 18: Position and attitude data for Experiment 3. A *successful* landing was achieved on a moving target with 2.0m/s. The MAV landed 5.55 s after the first target detection and 5.2 cm from the target center.

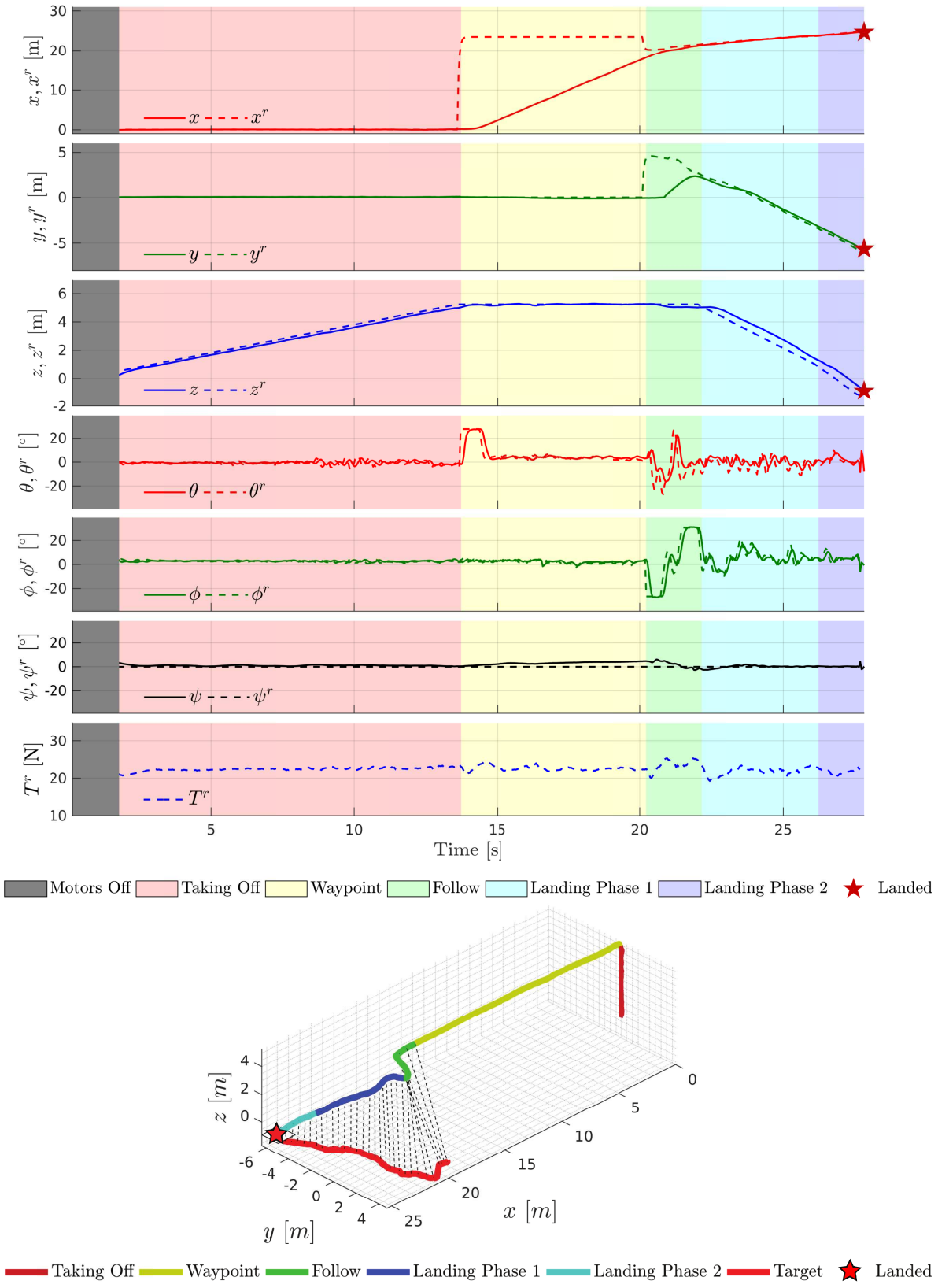

Figure 19: Position and attitude data for Experiment 4. A *successful* landing was achieved on a moving target with 2.0m/s. The MAV landed 7.65 s after the first target detection and 30.6 cm from the target center.

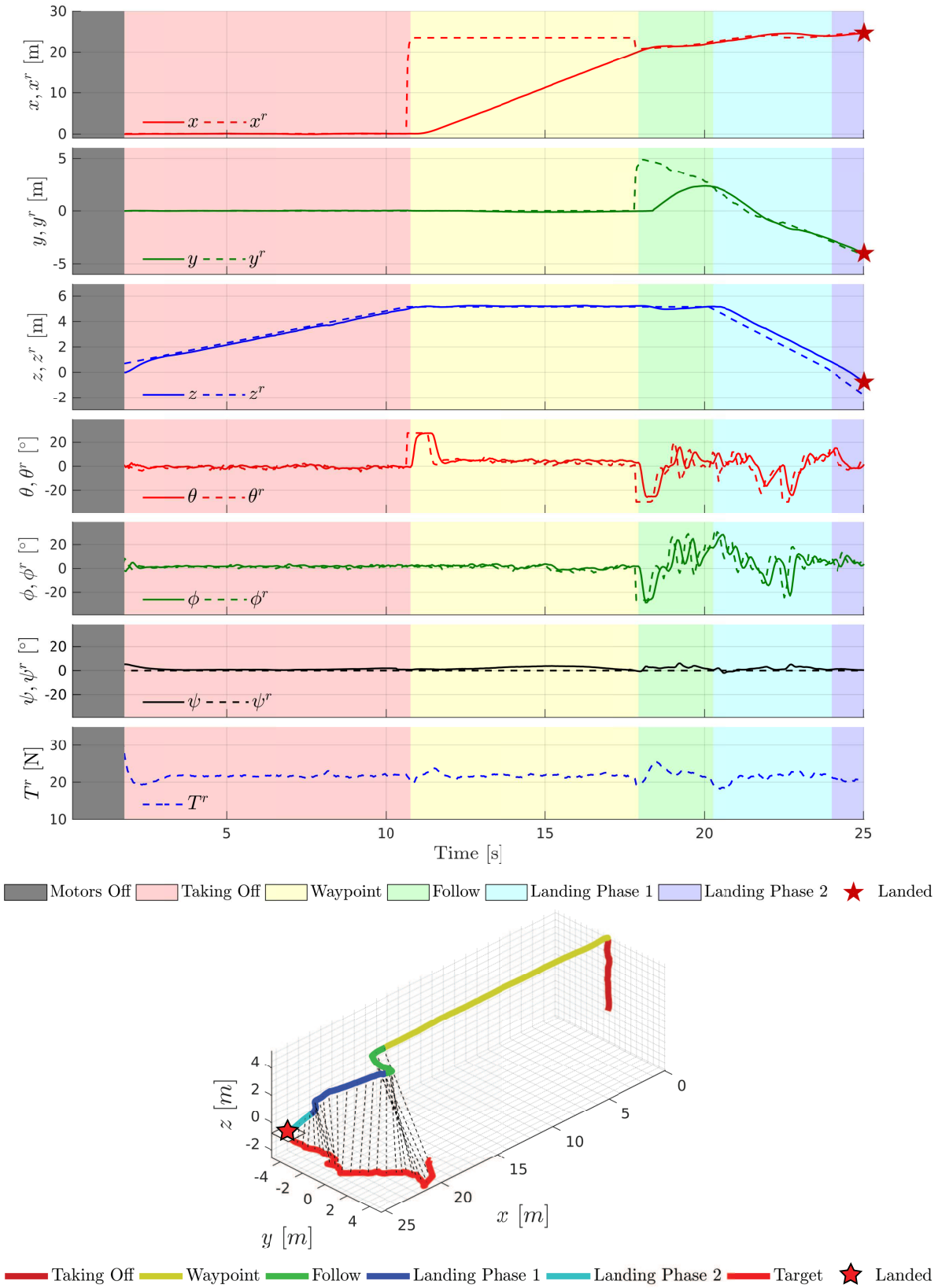

Figure 20: Position and attitude data for Experiment 5. A *successful* landing was achieved on a moving target with 2.3 m/s. The MAV landed 7.12 s after the first target detection and 16.5 cm from the target center.

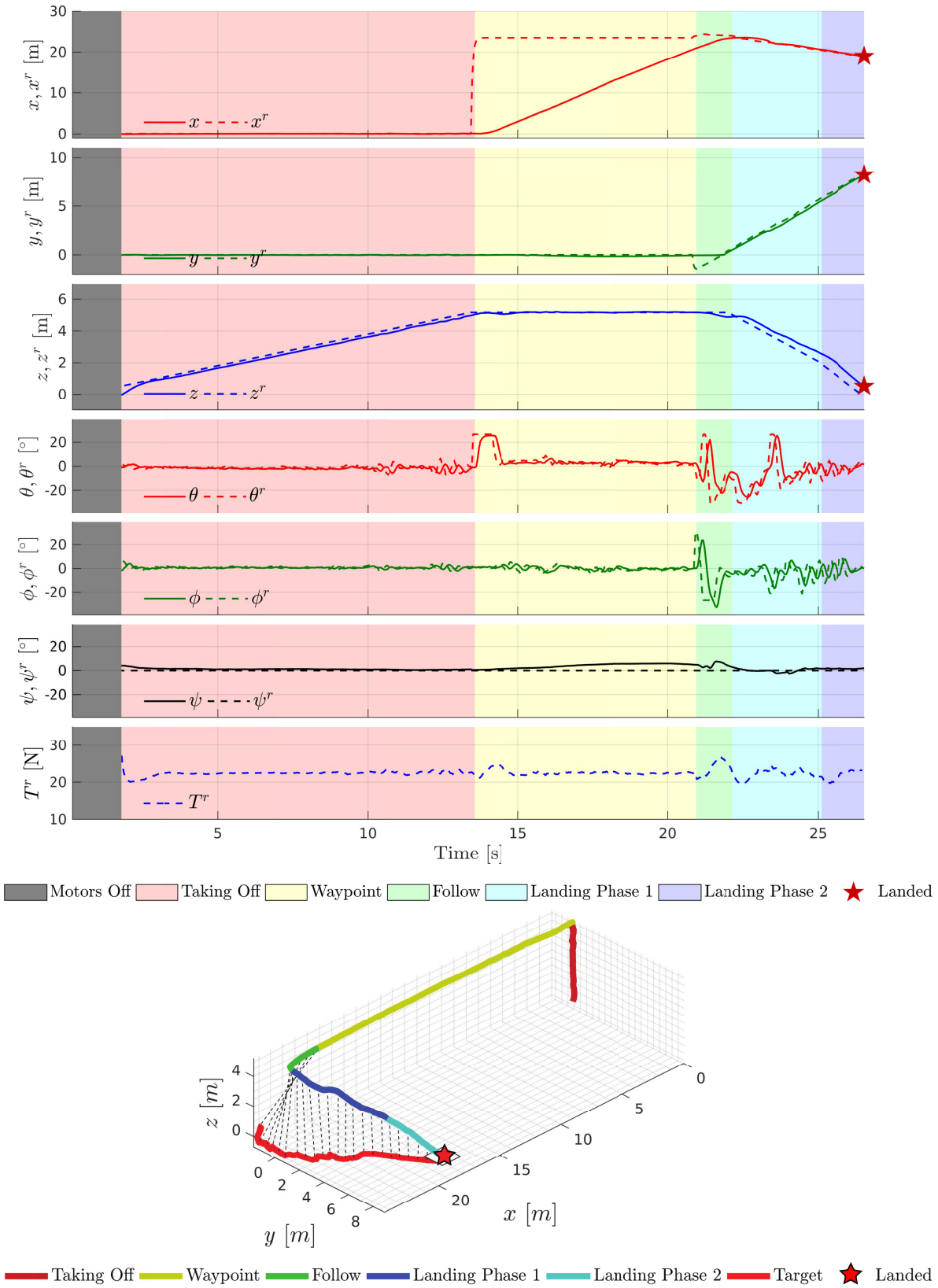

Figure 21: Position and attitude data for Experiment 6. A *successful* landing was achieved on a moving target with 2.4 m/s. The MAV landed 5.64 s after the first target detection and 6.6 cm from the target center.

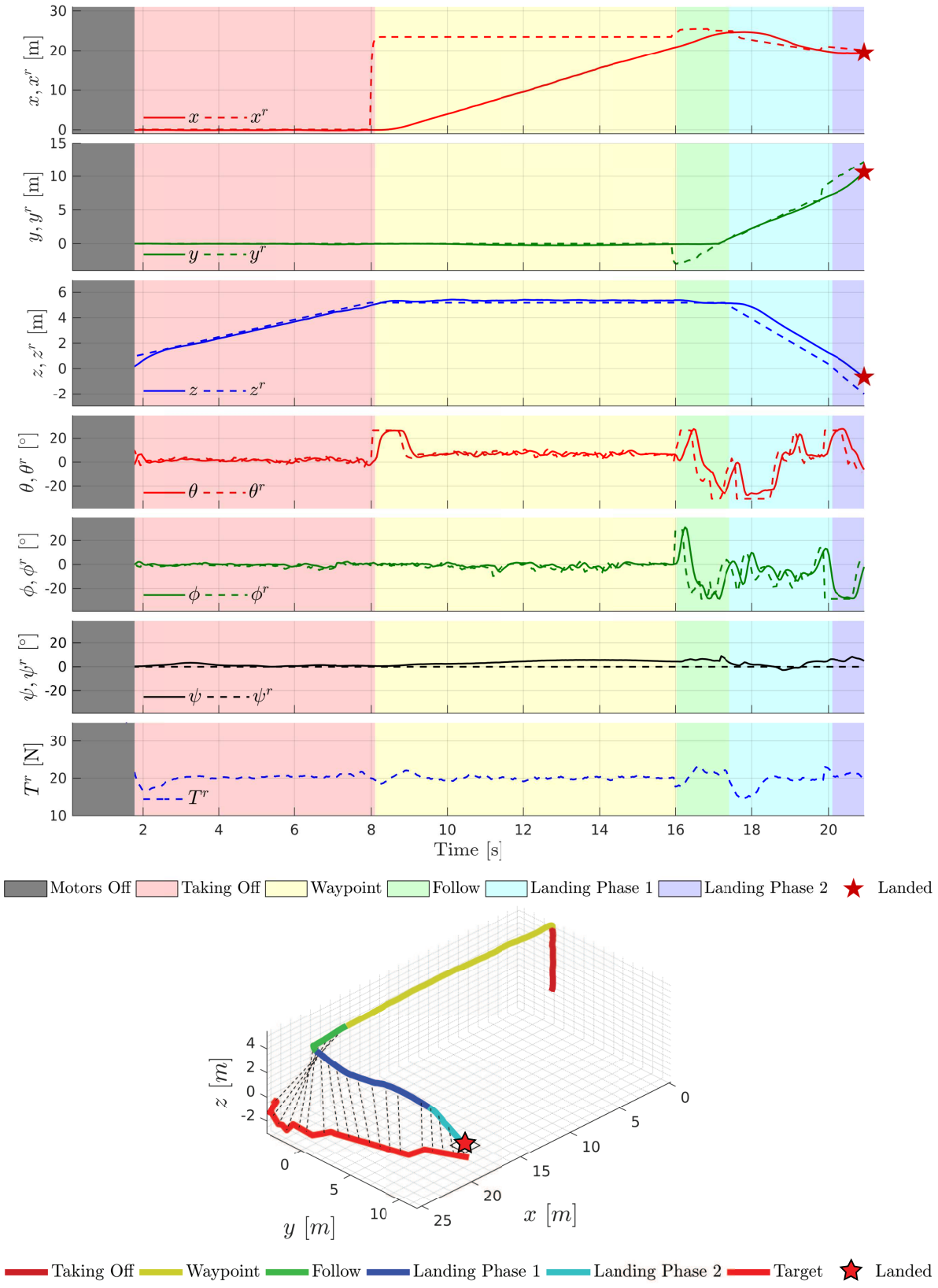

Figure 22: Position and attitude data for Experiment 7. A *successful* landing was achieved on a moving target with 3.5 m/s. The MAV landed 4.97 s after the first target detection and 30.2 cm from the target center.

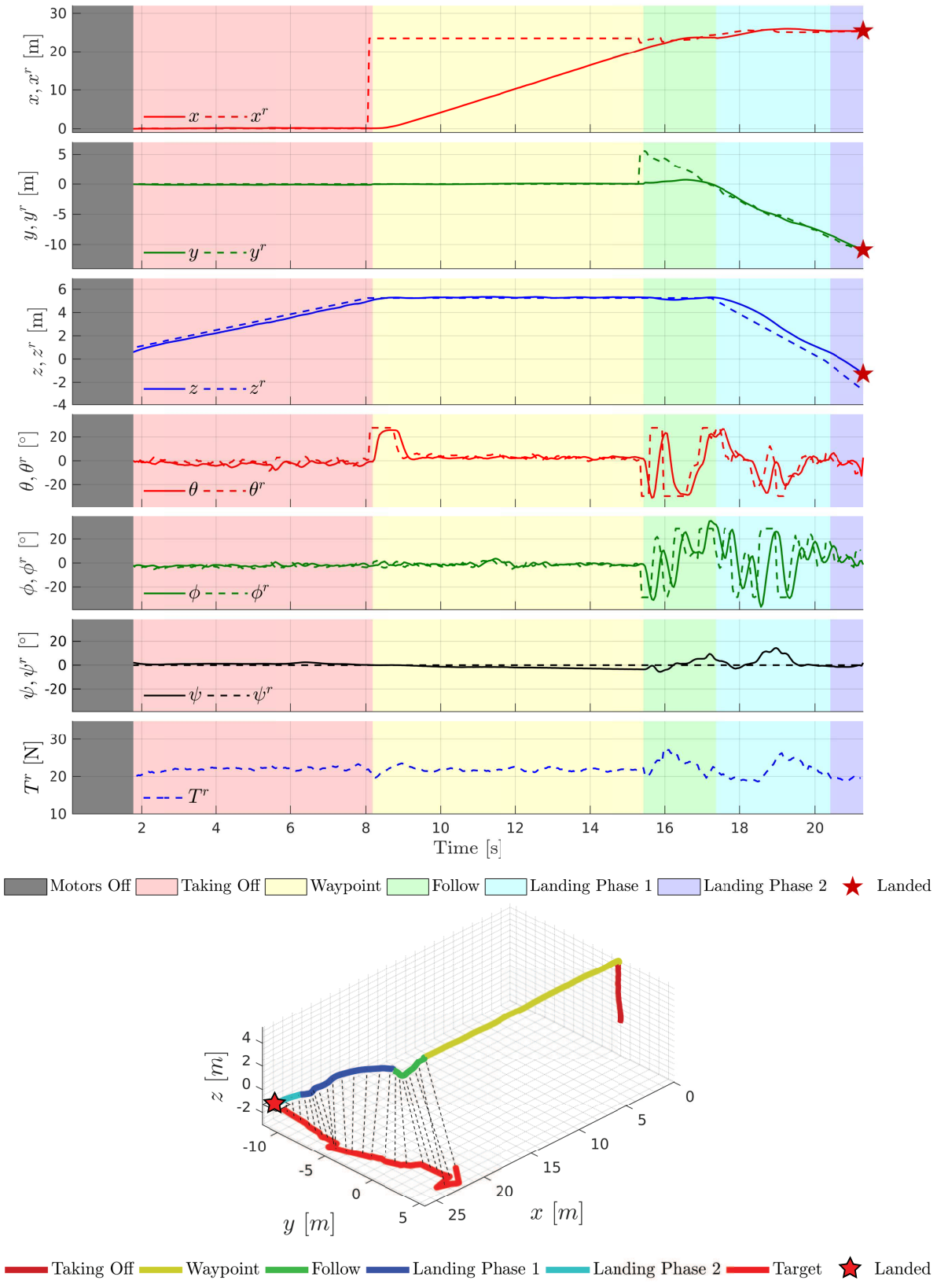

Figure 23: Position and attitude data for Experiment 8. A *successful* landing was achieved on a moving target with 3.8 m/s. The MAV landed 5.91 s after the first target detection and 18.6 cm from the target center.

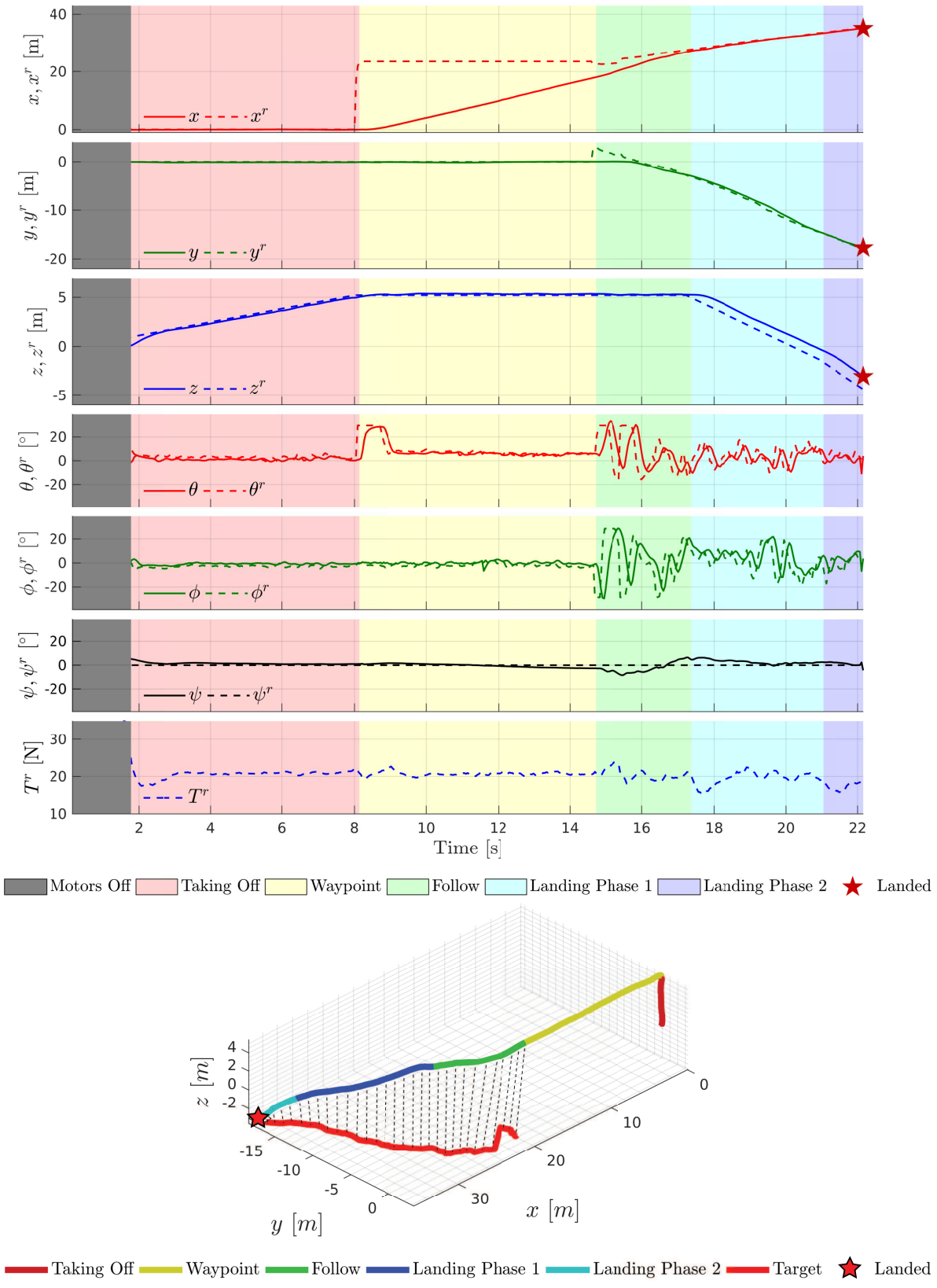

Figure 24: Position and attitude data for Experiment 9. A *successful* landing was achieved on a moving target with 4.0 m/s. The MAV landed 7.47s after the first target detection and 15.6cm from the target center.

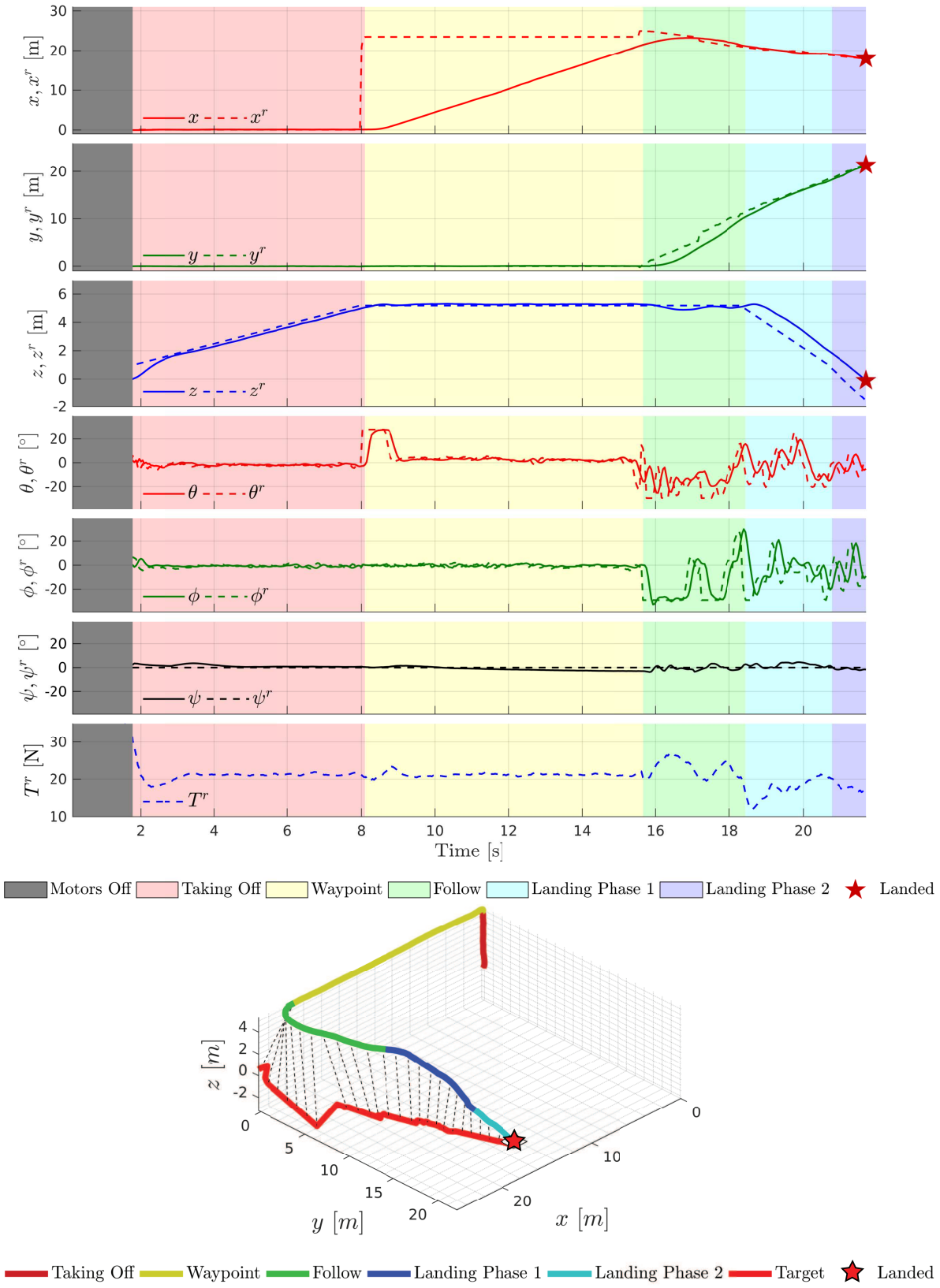

Figure 25: Position and attitude data for Experiment 10 with a *failed* landing attempt on a moving target with 4.6 m/s. Although that the estimated MAV final position is 29.8 cm away from the target center, the real one was significantly different than the target position.

References

Bachrach, A., Prentice, S., He, R., and Roy, N. (2011). Range–robust autonomous navigation in gps-denied environments. *Journal of Field Robotics*, 28(5):644–666.

Bähnemann, R., Pantic, M., Popovic, M., Schindler, D., Tranzatto, M., Kamel, M., Grimm, M., Widauer, J., Siegwart, R., and Nieto, J. I. (2017). THE ETH-MAV team in the MBZ international robotics challenge. *CoRR*, abs/1710.08275.

Beul, M., Houben, S., Nieuwenhuisen, M., and Behnke, S. (2017). Fast autonomous landing on a moving target at mbzirc. In *Proceedings of the European Conference on Mobile Robotics (ECMR)*.

Borowczyk, A., Nguyen, D.-T., Nguyen, A. P.-V., Nguyen, D. Q., Saussié, D., and Ny, J. L. (2016). Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle. *arXiv preprint arXiv:1611.07329*.

Bouabdallah, S. (2007). Design and control of quadrotors with application to autonomous flying.

Bouabdallah, S., Murrieri, P., and Siegwart, R. (2004). Design and control of an indoor micro quadrotor. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4393–4398. IEEE.

Bouabdallah, S. and Siegwart, R. (2007). Full control of a quadrotor. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 153–158. Ieee.

Castillo, P., Dzul, A., and Lozano, R. (2003). Real-time stabilization and tracking of a four rotor mini-rotorcraft. In *2003 European Control Conference (ECC)*, pages 3123–3128.

Darivianakis, G., Alexis, K., Burri, M., and Siegwart, R. (2014). Hybrid predictive control for aerial robotic physical interaction towards inspection operations. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 53–58.

Falanga, D., Zanchettin, A., Simovic, A., Delmerico, J., and Scaramuzza, D. (2017). Vision-based autonomous quadrotor landing on a moving platform. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 200–207.

Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2015). Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Proceedings of Robotics: Science and Systems (RSS)*.

Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2017). Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Robotics Research*, pages 235–252. Springer.

Kaess, M. (2013). Apriltags c++ library.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F. (2012). iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *International Journal of Robotics Research (IJRR)*. To appear.

Kamel, M., Alexis, K., Achtelik, M., and Siegwart, R. (2015). Fast nonlinear model predictive control for multicopter attitude tracking on so(3). In *2015 IEEE Conference on Control Applications (CCA)*, pages 1160–1166.

Kamel, M., Stastny, T., Alexis, K., and Siegwart, R. Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In Koubaa, A., editor, *Robot Operating System (ROS) The Complete Reference, Volume 2*. Springer.

Kerrigan, E. C. and Maciejowski, J. M. (2000). Soft constraints and exact penalty functions in model predictive control.

Kumar, V. and Michael, N. (2012). Opportunities and challenges with autonomous micro aerial vehicles. *The International Journal of Robotics Research*, 31(11):1279–1291.

Lee, D., Ryan, T., and Kim, H. J. (2012). Autonomous landing of a vtol uav on a moving platform using image-based visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 971–976. IEEE.

Leutenegger, S., Chli, M., and Siegwart, R. (2011). BRISK: Binary robust invariance scalable keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2014). Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, page 0278364914554813.

Li, D., Li, Q., Cheng, N., Wu, Q., Song, J., and Tang, L. (2013a). Combined rgbd-inertial based state estimation for mav in gps-denied indoor environments. In *2013 9th Asian Control Conference (ASCC)*, pages 1–8.

Li, D., Li, Q., Tang, L., Yang, S., Cheng, N., and Song, J. (2015). Invariant observer-based state estimation for micro-aerial vehicles in gps-denied indoor environments using an rgb-d camera and mems inertial sensors. *Micromachines*, 6(4):487–522.

Li, Q., Li, D.-C., fan Wu, Q., wen Tang, L., Huo, Y., xuan Zhang, Y., and Cheng, N. (2013b). Autonomous navigation and environment modeling for mavs in 3-d enclosed industrial environments. *Computers in Industry*, 64(9):1161 – 1177. Special Issue: 3D Imaging in Industry.

Maciejowski, J. (2002). *Predictive Control: With Constraints*. Pearson Education. Prentice Hall.

Mahony, R., Kumar, V., and Corke, P. (2012). Multirotor aerial vehicles. *IEEE Robotics and Automation magazine*, 20(32).

Mattingley, J. and Boyd, S. (2012). Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525.

Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3565–3572. IEEE.

Mueller, M. W. and D'Andrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 45–52.

Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*.

Oettershagen, P., Melzer, A., Leutenegger, S., Alexis, K., and Siegwart, R. (2014). Explicit model predictive control and l1-navigation strategies for fixed-wing uav path tracking. In *22nd Mediterranean Conference on Control and Automation*, pages 1159–1165.

Oettershagen, P., Melzer, A., Mantel, T., Rudin, K., Stastny, T., Wawrzacz, B., Hinzmann, T., Leutenegger, S., Alexis, K., and Siegwart, R. (2016). Design of small hand-launched solar-powered uavs: From concept study to a multi-day world endurance record flight. *Journal of Field Robotics*.

Papachristos, C., Alexis, K., and Tzes, A. (2016). Dual-authority thrust-vectoring of a tri-tiltrotor employing model predictive control. *Journal of Intelligent & Robotic Systems*, 81(3-4):471.

Serra, P., Cunha, R., Hamel, T., Cabecinhas, D., and Silvestre, C. (2016). Landing of a quadrotor on a moving target using dynamic image-based visual servo control. *IEEE Transactions on Robotics*, 32(6):1524–1535.

Shen, S., Michael, N., and Kumar, V. (2011). Autonomous multi-floor indoor navigation with a computationally constrained mav. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 20–25. IEEE.

Thomas, J., Welde, J., Loianno, G., Daniilidis, K., and Kumar, V. (2017). Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor. *IEEE Robotics and Automation Letters*, 2(3):1762–1769.

Vlantis, P., Marantos, P., Bechlioulis, C. P., and Kyriakopoulos, K. J. (2015). Quadrotor landing on an inclined platform of a moving ground vehicle. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2202–2207. IEEE.

Weiss, S., Achtelik, M. W., Lynen, S., Chli, M., and Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964. IEEE.