2017 IEEE International Symposium on
Safety, Security and Rescue Robotics (SSRR)
Shanghai, China, October 11-13, 2017

Vision-based Autonomous Quadrotor Landing on a Moving Platform

Davide Falanga, Alessio Zanchettin, Alessandro Simovic, Jeffrey Delmerico, and Davide Scaramuzza

***Abstract*— We present a quadrotor system capable of autonomously landing on a moving platform using only onboard sensing and computing. We rely on state-of-the-art computer vision algorithms, multi-sensor fusion for localization of the robot, detection and motion estimation of the moving platform, and path planning for fully autonomous navigation. Our system does not require any external infrastructure, such as motion-capture systems. No prior information about the location of the moving landing target is needed. We validate our system in both synthetic and real-world experiments using low-cost and lightweight consumer hardware. To the best of our knowledge, this is the first demonstration of a fully autonomous quadrotor system capable of landing on a moving target, using only onboard sensing and computing, without relying on any external infrastructure.**

SUPPLEMENTARY MATERIAL

Video of the experiments: `https://youtu.be/Tz5ubwoAfNE`

I. INTRODUCTION

Quadrotors are highly agile and versatile flying robots. Recent work has demonstrated their capabilities in many different applications including but not limited to: search-and-rescue, object transportation, inspection, surveillance and mapping [1], [2], [3]. The drawback of multirotors in general is a lower efficiency of the propulsion system when compared to other aerial vehicles, such as fixed-wing aircrafts. This limits the autonomy and utility of quadrotors as the time during which the vehicle can remain airborne is relatively short. One possible solution is to have a quadrotor autonomously land on a ground-station where its battery is charged or replaced.

Search and rescue robotics is a domain that could greatly benefit from aerial robots capable of landing autonomously on moving platforms. One day, flying robots will assist rescuers during their missions by providing an optimal platform for aerial inspection and mapping of the surroundings. Allowing these vehicles to autonomously land on predefined targets for battery charging/swapping or delivery of supplies would drastically enhance their usefulness while requiring limited or no human intervention. This would represent a major step forward in the use of autonomous robots in search-and-rescue missions, whose duration is usually significantly longer then the typical flight time of a drone.

This work focuses on the case where the ground-station moves inside a large mission area of known size (cf. Fig. 1).

The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—`http://rpg.ifi.uzh.ch`. This work was supported by the Swiss National Science Foundation through the National Centre of Competence in Research (NCCR) Robotics and the Mohamed Bin Zayed International Robotics Challenge (MBZIRC).

Fig. 1: Our quadrotor during the landing on a moving platform.

Our system relies on state-of-the-art algorithms for state estimation, trajectory planning, quadrotor control and detection of the moving target, all using only onboard sensing and computing. To the best of our knowledge, this is the first demonstration of a fully autonomous quadrotor system capable of landing on a moving target, using only onboard sensing and computing, without relying on any external infrastructure.

A. Related Work

Unmanned Aerial Vehicle (UAV) landing on a desired target has been an active research field during the last decades. A large body of the literature focuses on landing a UAV on a static target, such as a predefined tag or a runway. The state of the flying vehicle is estimated using motion-capture systems [4], GPS [5], [6] or computer vision [7]. Computer vision is the most common approach when it comes to detecting the landing target [6], [5], [7], [8]. Nevertheless, solutions for detecting the target based on motion-capture systems [4], or other sensors (e.g., GPS [9]) are available in the literature. Although interesting results have been achieved, they are not necessarily applicable to dynamically moving targets in an open outdoor environment. In regard to moving targets, a number of works focused on collaboration between a flying and a ground-based vehicle to coordinate the landing maneuver [10], [11], [12]. In this work, we do not assume that the two platforms are able to communicate or coordinate a landing.

In order to detect the landing platform, most state-of-the-art works exploit computer vision from onboard cameras. Visual servoing is a valid option to some extent [13], [14]; nevertheless, it requires the landing platform to be visible throughout the entire duration of the task, the reason being that the UAV is pulled towards the goal using solely visual

978-1-5386-3923-8/17/$31.00 ©2017 IEEE

information from its camera. To deal with missing visual information, model-based approaches have been proposed to predict the motion of the landing target [15], [16]. Alternative solutions are realized with the use of additional sensors attached to the moving target. Among many, these sensors include Inertial Measurement Units (IMU), GPS receivers [11], [17] or infrared markers [18].

For the UAV to be truly autonomous, all of the computation necessary to achieve the goal must be performed onboard. This is by no means standard in the literature, since all the approaches mentioned before rely on external computation for state estimation, trajectory planning or control [13], [15], [16]. Additionally, GPS [9], [19], [11], [16] or motion-capture systems [13], [12] are often used for state estimation, either only while patrolling or throughout the entire task. Conversely, we rely only on onboard visual-inertial odometry for state estimation.

B. Contribution

In this paper, we present a quadrotor system capable of autonomously landing on a moving target using only onboard sensing and computing. No prior knowledge about the location of the moving landing target is needed. We exploit state-of-the-art visual-inertial odometry to estimate the state of the quadrotor itself, complemented by nonlinear control algorithms to drive the vehicle. Our system detects the landing target using an onboard camera and deals with temporarily missing visual information by exploiting the the target's dynamical model. Therefore, no external infrastructure such as a motion-capture system is needed. We compute trajectories that take into account the dynamical model of the quadrotor and are optimal with respect to a cost function based on the energy necessary to execute it. We validate our approach in simulation as well as in real-world experiments, using low-cost, lightweight consumer hardware.

The remainder of this paper is structured as follows: Sec. II provides an overview on the proposed framework and details the algorithms used to estimate the state of the quadrotor, detect and track the moving platform, plan trajectories for the aerial vehicle, and control it along these trajectories. Sec. III describes the experimental platform and the simulation tools used to validate our approach, and provides the experimental results. In Sec. IV, we discuss the proposed method and provide insights on the experiments. Finally, we draw conclusions in Sec. V.

II. System Overview

Our system makes use of the following modules:

- quadrotor state estimation (Sec. II-A);
- moving target detection (Sec. II-B);
- moving target state estimation (Sec. II-C);
- trajectory planning (Sec. II-D);
- quadrotor control (Sec. II-E);
- state machine (Sec. II-F).

Fig. 2 provides a visual overview of these components. The modular structure of our framework allows us to easily modify or replace the algorithms inside each module without requiring changes to the others. Therefore, the one proposed in this work is a general purpose approach for landing a UAV on a moving target. It requires relatively few changes to be adapted to different platforms (e.g., fixed wings), algorithms, or scenarios.

A. Quadrotor State Estimation

We use monocular visual-inertial odometry to estimate the state of the quadrotor. More specifically, we rely on our previous work [20] for pose estimation. Pose estimates are computed at $40\,\mathrm{Hz}$ and fused with measurements coming from an Inertial Measurement Unit (IMU) using an Extended Kalman Filter [21] at $200\,\mathrm{Hz}$. Our state estimation pipeline provides an accurate estimate of the vehicle position, linear velocity and orientation with respect to the world frame $\{W\}$. The complete pipeline runs entirely on the onboard computer.

B. Vision-based Platform Detection

We employ onboard vision to estimate the position of the moving platform in a world frame $\{W\}$. To simplify the detection task, our moving platform is equipped with a visually distinctive tag. In this work, we leverage a tag like the one depicted in Fig. 3. The tag consists of a black cross surrounded by a black circle with a white backdrop. Nevertheless, our framework can easily generalize to a variety of tags, as for example April Tags [22], and to different detection algorithms. Our algorithm attempts to detect the landing platform in each camera image and estimate its position in the quadrotor body frame $\{B\}$. We first convert the image from the onboard camera into a binary black-and-white image by thresholding. Next we search for the white quadrangle with the largest area. In the case where no white quadrangle is visible, the landing platform cannot be found and the detection algorithm is concluded. Conversely, if a white quadrangle is found, we search for the pattern inside the quadrangle that composes our tag and extract its corners. More specifically, we first search for the circle and approximate it with a polygon, whose corners are used to estimate the position of the platform. If the circle is not entirely visible, we search for the four inner corners of the cross. If neither cross nor circle are visible, we use the four corners of the white quadrangle. To render our algorithm robust to outliers, we use RANSAC for geometric verification. Assuming the metric size of the tag to be known allows us to use the detected corners to solve a Perspective-n-Points (PnP) problem. In doing so, we obtain an estimate of the landing platform's position with respect to the quadrotor. Finally, we exploit the knowledge of the quadrotor's pose in world frame $\{W\}$ to transform the position of the ground platform from frame $\{B\}$ to $\{W\}$. The algorithm used to detect the platform is summarized in Alg. 1, and runs at $20\,\mathrm{Hz}$ on the onboard computer.

C. Platform State Estimation

The algorithm presented in Sec. II-B provides an estimate of the position of the ground vehicle in the world frame W.

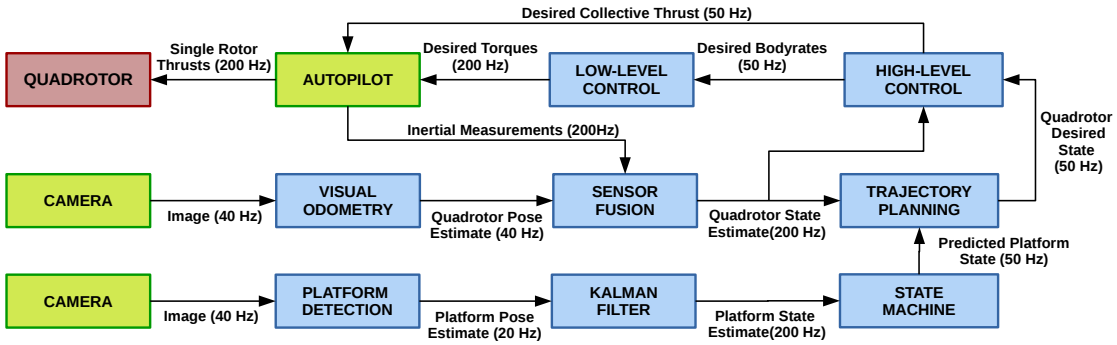

Fig. 2: A schematic representing our framework. Blue boxes represent software modules, green boxes are hardware components. The quadrotor platform is represented in red. Communication between modules happens through ROS.

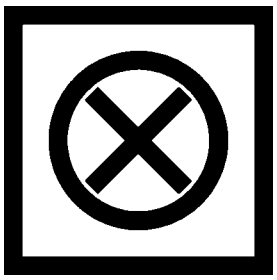

Fig. 3: The tag we used to detect the landing platform. Our framework does not strictly depend on specifics of the tag, and thanks to its modularity can easily generalize to other patterns.

However, the landing platform is not guaranteed to be visible at all times. To deal with missing visual detections, as well as to estimate the full state of the platform (namely the position, velocity and orientation), we use an Extended Kalman Filter [23]. We exploit a dynamic model of a ground vehicle based on *non-holonomic* movement constraints for the prediction phase [24], and consider tag detections from the onboard camera as measurements for the correction phase. For brevity reasons, we report only the main equations of the filter and refer the reader to [23] and [25] for further details.

1) Time Update: In the prediction step, the filter provides a prediction of the state of the moving platform based on the following non-linear equation:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{w}(t), \tag{1}$$

where $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$ and $\boldsymbol{w}(t)$ are the state of the system, the input and the process noise, respectively. We model the process noise as white Gaussian noise, namely $\boldsymbol{w}(t) \sim \mathcal{N}(0, \sigma_w^2)$. The function $f(\boldsymbol{x}, \boldsymbol{u})$ represents the dynamical

Algorithm 1 Moving landing platform detection

Input: Onboard camera image
Outputs: Landing platform position in {W}

```
binary_image ← black_and_white(camera_image)
polygons ← detect_polygons(binary_image)
landing_tag ← largest_quadrangle(polygons)
if landing_tag found then
    if circle ← detect_circle_in(landing_tag) then
        return circle.position()
    else
        if cross ← detect_cross_in(landing_tag) then
            return cross.position()
        else
            return landing_tag.position()
else
    return 0
```

model of the moving platform:

$$\dot{p_x} = v_t \cos(\theta) \tag{2a}$$
$$\dot{p_y} = v_t \sin(\theta) \tag{2b}$$
$$\dot{p_z} = 0 \tag{2c}$$
$$\dot{\theta} = u_1 \tag{2d}$$
$$\dot{v_t} = u_2 \tag{2e}$$

In (2), p_x, p_y, p_z are the 3D coordinates of the position of the platform in the world frame {W}, θ is the angle between the x-axis of the vehicle's body frame (i.e., its forward direction) and the world x-axis, v_t is the tangential velocity of the vehicle (see Fig.4), and u_1 and u_2 represent the control input to the system. In our case, we assume the velocity of the platform to be constant and therefore, that the inputs u_1 and u_2 are zero all the time. If any prior information about the motion of the vehicle is available (e.g., the path along which it moves), this can be easily incorporated into the dynamical model.

2) Measurement Update: The correction phase is performed each time a measurement $\boldsymbol{z}_k$ (the 3D position of

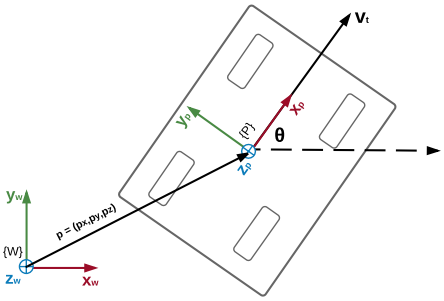

Fig. 4: A schematics representing the dynamical model of the moving platform. The world frame is indicated as $\{\mathrm{W}\}$, while the platform body frame as $\{\mathrm{P}\}$.

the moving platform) is provided by the detection algorithm, according to the following equations:

$$\hat{\boldsymbol{x}}(t) = f(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) + \boldsymbol{K}(t)(\boldsymbol{z}(t) - h(\hat{\boldsymbol{x}}(t)), \qquad (3)$$

where the matrix $\boldsymbol{K}(t)$ represents the Kalman gain.

D. Trajectory Planning

We use the approach proposed in [26] to plan optimal, feasible trajectories that prevent the vehicle from colliding with obstacles. The authors of that work propose a fast polynomial trajectory generation method that minimizes the third derivative of the position (namely, the *jerk*). Such an approach solves the minimization problem in closed form, therefore it is able to provide an optimal trajectory within a few micro-seconds running entirely onboard. Furthermore, the same method provides tools to verify whether the planned trajectory is feasible or not. More specifically, it allows the system to quickly check that each candidate: (i) does not exceed the physical actuation constraints of the platform, and (ii) does not collide with known obstacles (e.g., with the ground).

Additionally, during the platform following stage, we exploit the speed of the trajectory planning method [26] to provide the quadrotor with a set of feasible candidate trajectories, and we select the one with the lowest cost. Such a cost is the integral of the jerk along the trajectory, which the authors of [26] show to be an upper bound on the product of the inputs to the vehicle, namely the collective thrust and the angular velocities around the three body axes. Also, this allows us to quickly replan the desired trajectory during the platform following phase (see Sec. II-F). At each control cycle, we select n prediction times t_k by uniformly sampling a fixed-duration prediction horizon. For each time t_k, we predict the future state $\hat{\boldsymbol{x}}(t_k)$ that the landing platform will reach, starting from its last estimate available $\hat{\boldsymbol{x}}(t_c)$ at the current time t_c. The prediction is based on the dynamical model proposed in Sec. II-C. The future predicted state is used as the final state for each candidate trajectory. Out of all candidate trajectories, the one requiring a minimum amount of energy for execution is selected. We indicate the duration of the selected candidate as t_s.

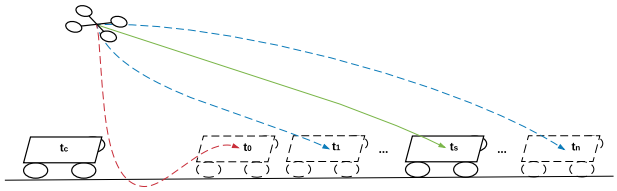

Fig. 5: An example of our planning strategy. The quadrotor plans n trajectories to reach the moving platform. Each one starts from its current position and has the ground vehicle's predicted position and velocity as final state. The future state of the moving target is predicted using its dynamical model, starting from the last estimate available from the Kalman Filter. Trajectories requiring inputs outside the allowed bounds or colliding with obstacles (e.g., with the ground), are rejected (dashed red lines in this image). We select the minimum-energy trajectory (green solid line, duration t_s) out of the set of all the feasible candidate trajectories (blue dashed lines).

E. Quadrotor Control

We use state-of-the-art, nonlinear control to drive our quadrotor along the desired trajectory. Broadly speaking, our controller is composed of a high-level controller for position and attitude corrections, and a low-level controller for reaching the required body rates. The high-level controller takes the difference between desired and estimated position, velocity, acceleration and jerk as input and returns the desired collective thrust and body rates. These body rates are passed as input to the low-level controller, which computes the necessary torques to be applied to the rigid body. The desired torques and the collective thrust are then converted to single motor thrusts. We refer the reader to our previous works [27] and [28] for further details on the dynamical model and the control algorithm used in this work.

F. State Machine

The state machine module governs the behavior of the quadrotor during the entire mission. It has four states, namely: *takeoff*, *exploration*, *platform tracking*, *landing*. Fig. 6 depicts the state machine with its states and the respective transitions triggered by events. In the following few sections we describe each of states in more detail.

1) Takeoff: Our quadrotor launches from the ground and is commanded to reach a hover point within a given amount of time. During the takeoff maneuver, we rely solely on the onboard IMU and a distance sensor. Once the vehicle is hovering, we initialize our visual odometry pipeline (Sec. II-A) to acquire and maintain a full state estimate. At this point, we switch the state machine to the *exploration* mode.

2) Exploration: The quadrotor explores an bounded area with known dimensions, flying at a given height. The vehicle autonomously computes waypoints to inspect the area and generates trajectories according to the strategy in Sec. II-D. This mode ends when the quadrotor detects the landing platform for the first time.

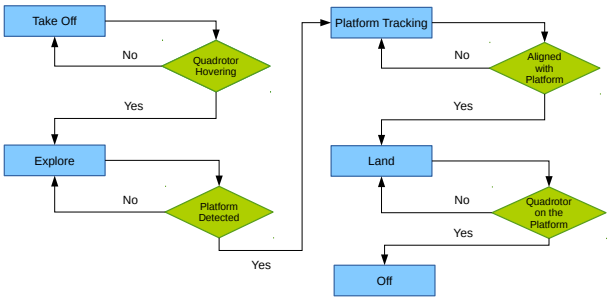

Fig. 6: The flowchart of our state machine.

3) Platform Tracking: In this phase, the quadrotor follows the moving platform and attempts to reach it and fly above it. We initialize the Kalman Filter (Sec. II-C) after the first detection and use its output to provide the trajectory planner with waypoints. At each control cycle, the quadrotor plans a set of candidate trajectories as described in Sec. II-D. Once the best candidate is selected, it is compared with the previous candidate and is executed only if the final position of the two trajectories differ significantly. We consider the tracking phase concluded when the quadrotor is above the ground platform and is moving at the same velocity.

4) Landing: When the vehicle is close enough to the landing platform and has matched its velocity, the state machine switches to the landing mode. In this phase, we command the vehicle to start a descent at a given vertical speed, while continuing to match the speed of the landing platform along the x and y axes. We use an onboard distance sensor to estimate the relative vertical distance between the quadrotor and the ground platform. The vehicle stops the motors when the distance to the platform is below a given threshold, concluding the landing maneuver.

III. Experiments

A. Simulation Environment

We used RotorS [29] and Gazebo to validate our framework in simulation. We replaced the default controller provided in the simulator with our own described in Sec. II-E. State estimation is provided by a simulated odometry sensor and, to bring the simulation closer to real experiments, we added white Gaussian noise to the estimated state of the vehicle. Furthermore, we used an onboard simulated camera to detect the landing platform. We used a Clearpath Husky UGV simulated model as ground vehicle, on top of which we mounted the tag to be detected.

B. Simulation Results

We tested our framework using this simulation environment in a number of different scenarios. More specifically, we run simulation experiments with the landing platform moving along paths with different properties (i.e., straight line, circle, figure-8). The landing platform's speed is varied between $1\,\mathrm{m/s}$ and $4.2\,\mathrm{m/s}$. In our experiments, the quadrotor takes off from the ground and explores a pre-defined area. When the landing platform is detected, the quadrotor starts following. Once it is close enough, the quadrotor initiates the landing maneuver. The results of one of our simulated experiments are visualized in Fig. 7.

C. Experimental Platform

For validating our framework in the real world, we used a custom-made quadrotor platform. The vehicle (cf. Fig. 8) is constructed from both, off-the-shelf and custom 3d-printed components. We used a DJI F450 frame, equipped with RCTimer MT2830 and soft 8-inch propellers from Parrot for safety reasons. The motors are driven by *Afro Slim Electronic Speed Controllers (ESC)*. The ESCs are commanded by the *PX4 autopilot*, which also sports an Inertial Measurement Unit. Our quadrotor is equipped with two *MatrixVision mvBlueFOX-MLC200w* cameras providing an image resolution of 752×480-pixel. One camera is looking forward and is tilted down by $45°$, while the second is facing towards the ground. We motivate this camera setup in Sec. IV-B. Furthermore, we mounted a *TeraRanger One* distance sensor to estimate the scale of the vision-based pose estimation, as well as to help the quadrotor during the takeoff and landing maneuvers. The software modules of our framework (i.e., trajectory planning, quadrotor control, visual odometry and visual-inertial fusion, platform detection and tracking) run in real time in ROS on one of the two onboard *Odroid XU4* computers. The two computers are interconnected through their Ethernet ports, providing a low latency connection. The overall weight of the platform is $1\,\mathrm{kg}$, with a thrust-to-weight ratio of 1.85.

D. Landing Platform

In our real-world experiments we use a *Clearpath Jackal* [1] as ground vehicle carrying the landing platform and control it manually. In nominal conditions the platform can reach a maximum speed of $2\,\mathrm{m/s}$. We installed a $150 \times 150\,\mathrm{cm}$ wooden landing pad equipped with the tag on the top of the vehicle, reducing its maximum speed to approximately $1.5\,\mathrm{m/s}$ due to the additional weight.

E. Real Experiments Results

We demonstrated our framework in a number of real experiments using the previously describe quadrotor platform. Similarly to our simulations, we tested the effectiveness of the proposed approach in different scenarios. More specifically, we had the landing platform moving along different paths, at different speeds. Fig. 9 reports the results for one of the experiments we conducted, with the landing platform moving on a straight line at $1.2\,\mathrm{m/s}$. The choice of such a speed is not due to limitations of our quadrotor system, but rather to the maneuverability of the ground robot used as moving target. The quadrotor starts the exploration at $t = 0$. The first platform detection happens at $t = t_d$, when the quadrotor starts the following phase. At $t = t_l$, the state machine detects that the vehicle is above the platform and

[1] https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/

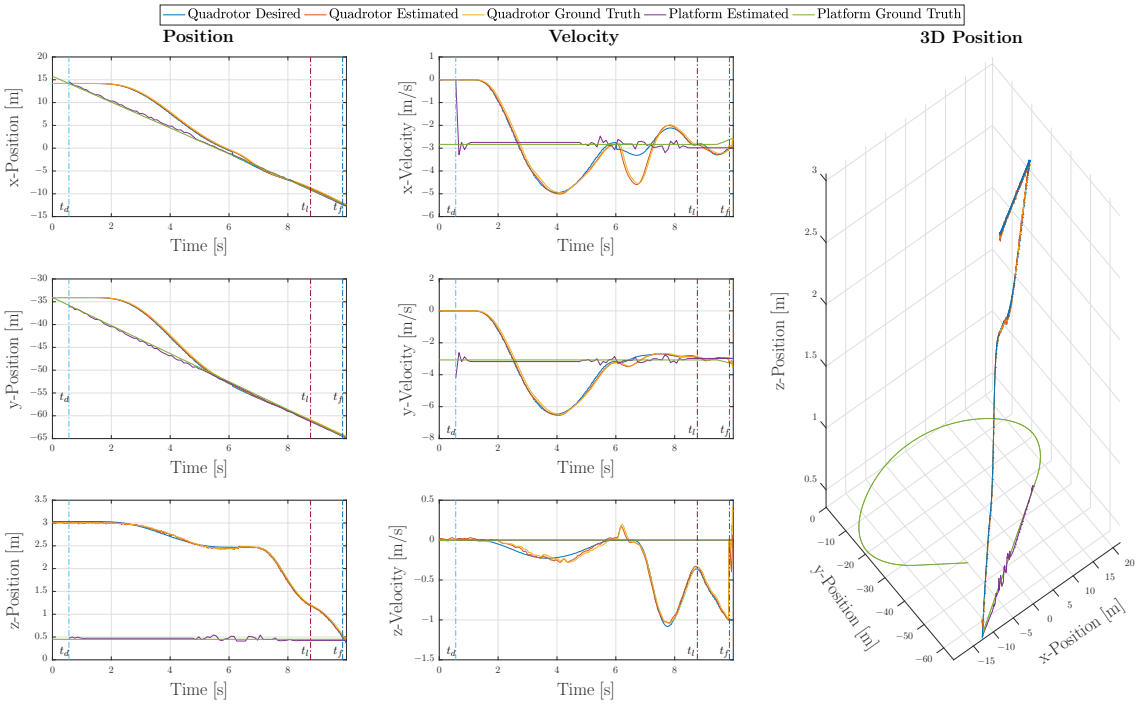

Fig. 7: The results of one of our simulations. We report data for position (left and right columns) and velocity (center column). The quadrotor starts the exploration at $t = 0$ and detects the moving platform for the first time at $t = t_d$. At this point, the tracking starts and the vehicle starts the landing phase at $t = t_l$. The maneuver is completed at $t = t_f$. The platform moves at a constant speed of $4.2\,\mathrm{m/s}$ along a figure-8 path.

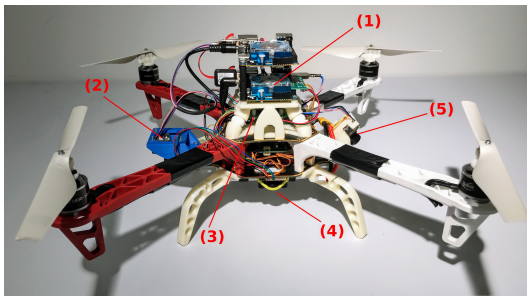

Fig. 8: The quadrotor used in our experiments. (1) The on-board computer running our algorithms. (2) The downward-looking camera used to detect the platform. (3) The PX4 autopilot. (4) The TeraRanger distance sensor. (5) The 45° angled-down camera used for visual odometry.

moves at approximately its speed, entering the landing stage. Finally, the quadrotor reaches the platform at $t = t_f$ and the maneuver is completed. For brevity reasons, we do not report any comparison between the estimated state of the quadrotor and ground-truth. We refer the reader to [20] for an extensive evaluation of the performance of our visual odometry pipeline.

IV. Discussions

A. Generality of the Framework

With the modular architecture of our framework as presented in Sec. II it is straight-forward to adapt it for different scenarios: Depending on the severity, changes in the hardware setup might require adjustments of the state estimation (Sec. II-A and quadrotor control (Sec. II-E) modules. In most cases, however, a re-tuning of the low- and high-level controller's parameters should suffice. Should it be required to equip the landing platform with a different kind of tag or markers or even active beacons, all necessary changes are confined to the target detection module (Sec. II-B). Likewise, the module for estimating the moving target's state (Sec. II-C) can be modified in cases where the landing platform exhibits drastically different dynamics than our model described in Eq (2). By modifying the state machine (Sec. II-F), the nature of the task can be altered. An example of such are autonomous reconnaissance missions where the quadrotor takes off and lands on a larger mobile robot. Another use case might be to have the quadrotor, or any kind of UAV for that matter, track a ground vehicle and provide a bird's view of it.

B. Motivation of the Vision Hardware Setup

Our experimental platform is equipped with two cameras, one forward-facing and is tilted down by 45° for visual odometry, one downward-looking to detect the platform. We chose this setup in order to have robust state estimation and to better detect the platform. Indeed, when the quadrotor is close to the ground vehicle, the image from the camera looking downwards contains mainly, if not only, the moving platform. Thus our visual odometry pipeline would estimate

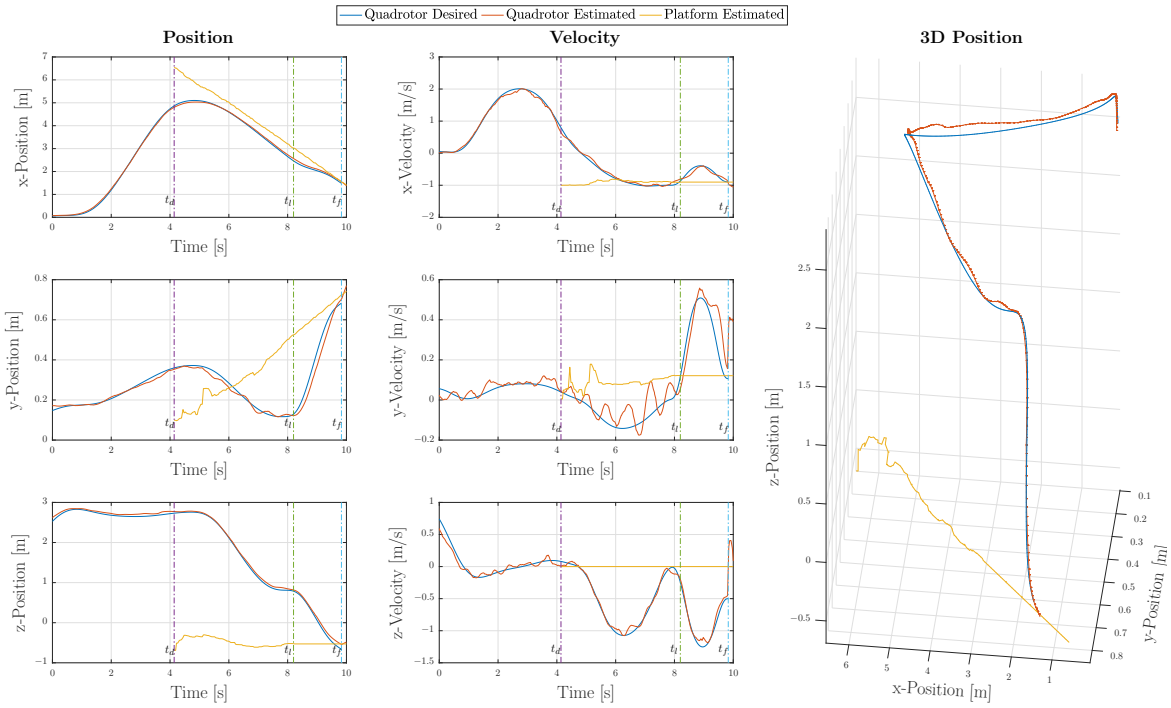

Fig. 9: The results of one of real experiments. We report data for position (left and right columns) and velocity (center column). The quadrotor starts the exploration at $t = 0$ and detects the moving platform for the first time at $t = t_d$, starting the tracking stage begins. During the landing stage, starting at $t = t_l$, the platform exits the field of view of the camera and the prediction of its motion is based solely on the dynamical model. The maneuver is completed at $t = t_f$. The platform moves at a constant speed of $1.2\,\mathrm{m/s}$ along a straight line.

only the relative motion with respect to the platform instead of a static world frame. A forward-looking camera solves this problem.

C. Computational Load

As mentioned in Sec. III-C, our quadrotor is equipped with two onboard computers even though all algorithms composing our framework can be run off a single computer. The second computer is used solely for data recording and rapid prototyping. Our control and visual odometry pipelines have been demonstrated to run onboard the quadrotor in our previous work [1] and we refer the reader to that for further details. The trajectory planning algorithm we use in this work typically needs approximately $0.02\,\mathrm{ms}$ per trajectory. Since we replan our desired trajectory at $50\,\mathrm{Hz}$, we can potentially compute up to 1000 candidate trajectories per replanning-cycle. Nevertheless, we fix the number of candidate trajectories to be computed at 20, which is usually sufficient to find a feasible trajectory during the platform following phase.

The statistics of the time required by our vision-based platform detection algorithm are reported in Table I. On average, it takes approximately $12\,\mathrm{ms}$ to detect the landing platform in each image, leading to a potential maximum rate of approximately $80\,\mathrm{Hz}$. However, we found that a rate of $20\,\mathrm{Hz}$ is sufficient to obtain reliable and accurate results in tracking the landing platform.

TABLE I: Computation time statistics for our onboard, vision-based platform detection algorithm.

	Mean	Standard Deviation	
Image Thresholding	0.87	0.51	[ms]
Quadrangle Detection	4.35	1.89	[ms]
Circle Detection	0.06	0.03	[ms]
Cross Extraction	1.81	1.01	[ms]
Perspective-n-Points	4.95	2.31	[ms]
Total	12.04	5.75	[ms]

D. Trajectory Planning

In this work, we use trajectories that minimize the jerk to provide our controller with reference states that drive the vehicle towards the accomplishment of the mission (cf. Sec. II-D). Previous work has shown that trajectories that minimize the snap, namely the fourth derivative of the position, lead to a smoother behavior for a quadrotor [30]. However, computing minimum snap trajectories typically requires longer then the closed form solution for minimum jerk trajectories we exploit. Also, to the best of our knowledge, no efficient feasibility verification method is available for minimum snap trajectories. In our experiments, we observed a better overall behavior of the entire pipeline when using minimum jerk trajectories. The reasons behind this are twofold: (i) the very efficient computation of minimum jerk trajectories make it

possible to re-plan the desired trajectory at high frequency to deal with changes in the motion of the moving target; (ii) the feasibility verification method lets us plan trajectories which satisfy the physical limits of the platform, i.e. avoid motors saturation.

E. Dealing with Missing Platform Detection

We deal with temporarily missing detections of the moving platform during the following and landing phases by using the Extended Kalman Filter described in Sec. II-C. Despite the lack of prior information about the motion of the platform and the constant velocity assumption, the dynamical model used for the prediction phase provides reliable results in both simulation and real world experiments. Therefore, our framework is capable of landing a quadrotor on a moving target even in the case when the platform is not temporarily visible.

V. Conclusions

In this work, we presented a quadrotor system capable of autonomously landing on a moving platform using only onboard sensing and computing. We relied on state-of-the-art computer vision algorithms, multi-sensor fusion for localization of the UAV, detection and motion estimation of the moving platform, and path planning for fully autonomous navigation. No external infrastructure, such as motion-capture systems or GPS, is needed. No prior information about the location of the moving landing target is required to execute the mission. We validated our framework in simulation as well as with real-world experiments using low-cost and lightweight consumer hardware. To the best of our knowledge, this is the first demonstration of a fully autonomous quadrotor system capable of landing on a moving target, using only onboard sensing and computing, without relying on any external infrastructure.

References

[1] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor MAV," *J. Field Robot.*, vol. 33, no. 4, pp. 431–450, 2016.

[2] D. Mellinger, M. Shomin, and V. Michael, N. Kumar, *Cooperative Grasping and Transport Using Multiple Quadrotors*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 545–558.

[3] L. Doitsidis, S. Weiss, A. Renzaglia, M. W. Achtelik, E. Kosmatopoulos, R. Siegwart, and D. Scaramuzza, "Optimal surveillance coverage for teams of micro aerial vehicles in gps-denied environments using onboard vision," *Auton. Robots*, vol. 33, no. 1, pp. 173–188, 2012.

[4] D. Mellinger, M. Shomin, and V. Kumar, "Control of quadrotors for robust perching and landing," pp. 205–225, 2010.

[5] C. S. Sharp, O. Shakernia, and S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, 2001, pp. 1720–1727.

[6] S. Lange, N. Sünderhauf, and P. Protzel, "Autonomous landing for a multirotor uav using vision," in *Int. Conf. on Simulation, Modeling, and Programming for Auton. Robots (SIMPAR)*, 2008, pp. 482–491.

[7] B. Herisse, F. Russotto, T. Hamel, and R. Mahony, "Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2008, pp. 801–806.

[8] D. Tang, F. Li, N. Shen, and S. Guo, "Uav attitude and position estimation for vision-based landing," in *Electronic and Mechanical Engineering and Information Technology (EMEIT), International Conference on*, vol. 9, 2011, pp. 4446–4450.

[9] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 3, 2002, pp. 2799–2804.

[10] J. M. Daly, Y. Ma, and S. L. Waslander, "Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays," *Auton. Robots*, vol. 38, no. 2, pp. 179–191, 2015.

[11] T. Muskardin, G. Balmer, S. Wlach, K. Kondak, M. Laiacker, and A. Ollero, "Landing of a fixed-wing uav on a mobile ground vehicle," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1237–1242.

[12] K. A. Ghamry, Y. Dong, M. A. Kamel, and Y. Zhang, "Real-time autonomous take-off, tracking and landing of uav on a moving ugv platform," in *Control and Automation (MED), 2016 24th Mediterranean Conference on*, 2016, pp. 1236–1241.

[13] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a vtol uav on a moving platform using image-based visual servoing," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012, pp. 971–976.

[14] P. Serra, R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre, "Landing of a quadrotor on a moving target using dynamic image-based visual servo control," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1524–1535, Dec 2016.

[15] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Quadrotor landing on an inclined platform of a moving ground vehicle," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 2202–2207.

[16] J. Kim, Y. Jung, D. Lee, and D. H. Shim, "Landing control on a mobile platform for multi-copters using an omnidirectional image sensor," *J. Intell. Robot. Syst.*, pp. 1–13, 2016.

[17] A. Borowczyk, D. Nguyen, A. Phu-Van Nguyen, D. Q. Nguyen, D. Saussié, and J. Le Ny, "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *ArXiv*, vol. abs/1611.07329, 2016. [Online]. Available: http://arxiv.org/abs/1611.07329

[18] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle," *J. Intell. Robot. Syst.*, vol. 61, no. 1-4, pp. 221–238, 2011.

[19] T. S. Richardson, C. G. Jones, A. Likhoded, E. Sparks, A. Jordan, I. Cowling, and S. Willcox, "Automated vision-based recovery of a rotary wing unmanned aerial vehicle onto a moving platform," *J. Field Robot.*, vol. 30, no. 5, pp. 667–684, 2013.

[20] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 15–22.

[21] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2013, pp. 3923–3929.

[22] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011.

[23] H. Sorenson, *Kalman Filtering: Theory and Application*, ser. IEEE Press selected reprint series. IEEE Press, 1985.

[24] B. Siciliano, L. Sciavicco, and L. Villani, *Robotics: modelling, planning and control*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer, 2009.

[25] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[26] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1294–1310, 2015.

[27] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 1722–1729.

[28] M. Faessler, D. Falanga, and D. Scaramuzza, "Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 476–482, Apr. 2017.

[29] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[30] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011, pp. 2520–2525.