ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/330155244

Autonomous landing on a moving vehicle with an unmanned aerial vehicle

Article *in* Journal of Field Robotics · January 2019

DOI: 10.1002/rob.21858

CITATIONS
54

READS
1,748

9 authors, including:

Tomáš Báča
Czech Technical University in Prague
65 PUBLICATIONS **1,251** CITATIONS
SEE PROFILE

Petr Štěpán
Czech Technical University in Prague
27 PUBLICATIONS **279** CITATIONS
SEE PROFILE

Vojtěch Spurný
Czech Technical University in Prague
33 PUBLICATIONS **612** CITATIONS
SEE PROFILE

Daniel Hert
Czech Technical University in Prague
16 PUBLICATIONS **428** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Sampling-based planning of actions and motions using approximate solutions View project

Autonomous robotic pouring assistant View project

All content following this page was uploaded by Tomáš Báča on 05 January 2019.

The user has requested enhancement of the downloaded file.

Autonomous Landing on a Moving Vehicle with an Unmanned Aerial Vehicle

Tomas Baca*
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
tomas.baca@fel.cvut.cz

Petr Stepan
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
petr.stepan@fel.cvut.cz

Vojtech Spurny
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
vojtech.spurny@fel.cvut.cz

Daniel Hert
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
vojtech.spurny@fel.cvut.cz

Robert Penicka
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
robert.penicka@fel.cvut.cz

Martin Saska
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
martin.saska@fel.cvut.cz

Justin Thomas†
GRASP Laboratory
University of Pennsylvania
Philadelphia, US
jut@seas.upenn.edu

Giuseppe Loianno
Department of ECE and MAE
Tandon Schoolof Engineering
New York University
NewYork City, New York
loiannog@seas.upenn.edu

Vijay Kumar
GRASP Laboratory
University of Pennsylvania
Philadelphia, US
kumar@seas.upenn.edu

Abstract

This paper addresses the perception, control and trajectory planning for an aerial platform to identify and land on a moving car at 15 km/h. The hexacopter Unmanned Aerial Vehicle (UAV), equipped with onboard sensors and a computer, detects the car using a monocular camera and predicts the car future movement using a nonlinear motion model. While following the car, the UAV lands on its roof, and it attaches itself using magnetic legs. The proposed system is fully autonomous from takeoff to landing. Numerous field tests were conducted throughout the year-long development and preparations for the MBZIRC 2017 competition, for which the system was designed. We propose a novel control system in which a Model Predictive Controller is used in real time to generate a reference trajectory for the UAV, which are then tracked by the nonlinear feedback controller. This combination allows to track predictions of the car motion with minimal position error. The evaluation presents three successful autonomous landings during the MBZIRC 2017, where our system achieved the fastest landing among all competing teams.

*http://mrs.felk.cvut.cz
†http://www.grasp.upenn.edu

1 Introduction

Autonomous take-off and landing are the key components and also the most challenging components of all fully autonomous UAV systems. Precise landing ability is important for autonomous docking of UAV platforms (mainly Micro Aerial Vehicles - UAVs) into a recharging station in missions requiring repeated flight operations, and also in information gathering and delivery applications, where it is required to reach a precise, desired position and then return to a base. Even more challenging abilities are required for landing on a moving platform, especially if the platform may not be equipped with a precise localization system. Although the use of a moving helipad introduces uncertainty and a source of possible failures into the UAV system, it extends the application domain of UAVs and especial of multi-rotor helicopters. These platforms benefit from high robustness and maneuverability. However, they suffer from a short operational time, and a cooperation with another vehicle is often required. A UAV system capable of vertical take-off and landing on a moving vehicle may be deployed from boats, trains or cars in areas close to the target locations of the UAV mission. Short-term flights of this kind efficiently exploit the abilities of UAVs, and combining them with a moving platform extends their operational range.

Hundreds of works dealing with autonomous landing on static and dynamic helipads have been published in this decade in the robotics literature describing advanced control and landing pattern detection algorithms and showing promising simulations and laboratory experiments. However, only a few of these works have demonstrated deployment in real-world conditions, and none of them have presented a reliable performance that enables repeated landing on a fast-moving helipad in a demanding outdoor environment. This huge reality gap was identified by the scientific board of the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2017 competition, organized by the Khalifa University of Science in Abu Dhabi. The aim of this board of top scientists in robotics was to select tasks on the edge of the current state-of-the-art to provide a significant impact on the robotic community. Automatic landing on a fast-moving vehicle was the first challenge on their list.

The existence of the reality gap was confirmed in the MBZIRC competition, where only five teams (out of 142 registered teams from almost all the best robotic groups worldwide) successfully landed during the competition on a car moving at a speed of 15 km/h and only two teams (CTU-UPenn-UoL and the team of the University of Bonn) landed precisely in both trials of Challenge 1 of the competition. The CTU-UPenn-UoL system is presented here. Most importantly, the MBZIRC competition can be considered as a relevant and objective benchmark of this task, which is currently being investigated by the robotic community, since several competitive solutions were compared in the same experimental setup. The same car, the same landing pattern and the same trajectory and velocity profile of the car were used for all competitors in the same environment. The criterion for success was the shortest time of landing. Moreover, the successful teams had to achieve the goal after only a few minutes of preparation, without the option of postponing the beginning of their time slot. By standard practice in most laboratory experiments, no repeated tests were allowed, and moreover, the system robustness was exhibited in the current environmental conditions (light and windy), since the teams could not influence the start of their trial.

The solution described in this paper presented the best reliability among all teams and it achieved the fastest performance in the entire competition. Only our system was able to land three times in the competition, without a failure in autonomous mode - see Table 1. The fastest time of landing was achieved by the proposed system during the grand challenge, where all MBZIRC challenges were solved simultaneously. This even increased the demands on system robustness and immediate deployment without any preparation. The key components of the system (HW and SW) that provided this high reliability and performance in comparison with state-of-the-art works are described in the following paper. A novel UAV state estimation approach is presented together with a predictive trajectory tracking technique that enables us to track and predict an estimated position of the landing pattern, with the necessary precision and maneuverability to be able to follow the car even in turns of its path. For precise ground vehicle state estimation, which is crucial information for the landing UAV, fast and robust visual localization of the landing pattern is proposed. The detected positions of the car are filtered using an Unscented Kalman Filter (UKF) based technique with an

assumed car-like model of the vehicle, while the prediction of the car position in future takes into account a known profile of the track that is followed by the moving helipad. The model predictive control based approach applied for car tracking using an estimate of its movement in future is the most important element of the proposed system that enables the UAV to land precisely on a platform following a speed profile that is close to the speed limit of the UAVs and on a non-straight path.

1.1 State-of-the-art

The academic community has identified great interest in the task of autonomous landing of Unmanned Aerial Vehicles on ground or marine vehicles. The survey in (Jin et al., 2016) provides an overview of techniques used for vision-based autonomous landing. A list of various visual markers with the corresponding detection techniques is referenced, as well as hardware design, control and estimation methods for both indoor and outdoor tasks. Similarly, methods for more general autonomous landing of an unmanned aerial system are described in (Kong et al., 2014).

Vision-based estimation of a ground vehicle states using an only UAV onboard sensors is proposed in (Benini et al., 2016). Robust marker detection using onboard GPU (Graphics Processing Unit) provides precise pose information even in occluded and cluttered environments. The authors of (Lin et al., 2017) also propose a method for automatic detection and estimation of a landing marker onboard a ship deck. They also aim to provide a robust pose estimate when the marker is partially occluded, or when the scene contains marker reflections. However, both solutions lack an experimental evaluation that would test the system during fully autonomous landing.

The authors of (Jung et al., 2015), (Jung et al., 2016), (Fu et al., 2016) and (Ghommam and Saad, 2017) deal with simulations of landing marker detection. They also propose guidance laws for autonomous landing, but also in simulation. Simulated autonomous landing on a ship is presented in (Tan et al., 2016).

Many of indoor experiments on autonomous landing on a slow-moving target are presented in works by (Ghamry et al., 2016), (Araar et al., 2017), (Lee et al., 2012) and (Bi and Duan, 2013). An indoor solution with a motion capture system is presented in (Ghamry et al., 2016). A ground robot and a UAV are both controlled by a centralized system to fulfill missions which include autonomous takeoff and landing of the UAV, atop the ground vehicle. The system presented in (Lee et al., 2012) uses sensors onboard the UAV and relies on the Vicon motion capture system and external computational unit. Thanks to the motion capture system, the UAV is able to conduct a patrol search for the ground vehicle. When the ground vehicle is located, it switches to relative localization based on visual marker detection. Similarly, a system for indoor autonomous tracking and landing is presented in (Hui et al., 2013). Camera images are also processed off board on an external computer.

Multiple works describe systems capable of autonomous outdoor flight while tracking a static or moving marker. In (Yang et al., 2013), (Masselli et al., 2014) and (Yang et al., 2015), UAV systems capable of hovering and landing on a static target are proposed. Autonomous landing on a target moving at slow speed up to 1 m/s is presented in (Kim et al., 2014) and in (Lee et al., 2016). The authors of (Xu and Luo, 2016) present a solution capable of landing on a moving car at speeds of 7 m/s. The presented system was tested in scenarios with the ground vehicle moving along a straight line.

The most similar approach to our work is presented by (Borowczyk et al., 2017) and (Hoang et al., 2017). The authors of (Borowczyk et al., 2017) propose a system that utilizes a vision-based approach combined with inertial and GPS measurements from a cell phone placed on the ground vehicle. Experiments show landings at speeds up to 50 km/h. However, it is unclear whether the system is capable of landing during non-linear motion of the car. Moreover, precise knowledge of the global position of the car is an assumption that is problematic in most applications. Successful landing on a moving vehicle in an outdoor environment is also described in (Hoang et al., 2017). Only onboard sensory data and computation power are used. The proposed solution is able to track and land on a car moving at a speed of up to speed 2 m/s.

The competitive solutions in the MBZIRC competition were presented by the team of the Beijing Institute of Technology and by the team of the University of Bonn (Beul et al., 2017). They also landed multiple times, but their systems have not yet been published. It is therefore not possible to compare the two systems and to highlight differences. Nevertheless, all three solutions can be considered as a valuable contribution to the field of robotics since according to our knowledge no other system exists that can offer a complete solution to this very demanding and complex challenge in these outdoor conditions (which was also the reason why this task was selected by respected leaders in the field of robotics for the competition).

1.2 Contributions

This manuscript presents a complete system for automatic detection, estimation, tracking and landing on a moving car with an unmanned aerial vehicle. The proposed method enables the UAV to detect a landing pattern in images from a single onboard camera and to calculate the position of the car relative to the UAV. The computer vision algorithm provides fast and robust detection of the landing marker using a SuperFisheye camera lens. Position and velocity of the car are estimated and predicted in a global frame of reference using a non-linear motion model by an Unscented Kalman Filter. The proposed control architecture is tailored specifically to the challenge of following fast dynamic objects with a UAV by leveraging the non-linear state controller in conjunction with a novel model predictive control tracker. Future predictions of the car movement are reshaped by the MPC tracker into a feasible state reference, which is reflected on the states of the real UAV by the non-linear state controller. The novelty of this approach is in the combination of the state-of-the-art methods to accomplish a robust execution of the demanding task in real-world experimental conditions. Moreover, the novel MPC tracker is a contribution which allows to track arbitrarily long reference trajectories without solving an MPC task for the whole reference. The experimental results show that the UAV can follow a car moving at 15 km/h autonomously and land on its roof, while attaching itself using magnetic legs. The system is robust to very challenging outdoor conditions with a wind speed of up to 10 m/s, varying light conditions and blowing sand in the air. The approach presented in the manuscript provides precision and repeatability in the landing task, which is a crucial element for fully autonomous missions (such as periodical surveillance, reconnaissance, object carrying, and monitoring), for which UAVs are an especially appealing option.

1.3 Problem definition

The task, as it is described by the rules of the competition, consists of an autonomous search and landing on a moving ground vehicle by an unmanned aerial vehicle, a multirotor helicopter. The competition takes place in an arena with dimensions of 90×60 meters, as shown in Figure 1a. A track in the shape of figure 8 is marked on the ground where the car is supposed to drive starting at a random place and heading in a random direction. The landing area is a square with dimensions of 1.5×1.5 m with a marker X, as shown in Figure 1b, placed 1.5 m above the ground on the roof of the vehicle. A magnetic or suction gripper can be used to attach the UAV to the surface, which is made of a ferrous material. The moving vehicle starts at a constant speed of 15 km/h. It reduces the speed to 10 km/h after 6 minutes and to 5 km/h after 12 minutes from the start. However, our system was designed to land independently of the speed level. No human intervention is allowed in the fully autonomous mode.

2 Experimental hardware platform

The experimental platform was designed from off-the-shelf parts, with the aim to simplify reproducibility and potential maintenance. The same platform was also successfully used for the treasure hunt challenge – MBZIRC challenge No. 3 (our team won this challenge, as described in (Spurny et al., 2018)), where three UAVs cooperatively collected small objects. More importantly, we intended to reuse the platform for future research activities, which introduced a need for simple potential modifications to the system.

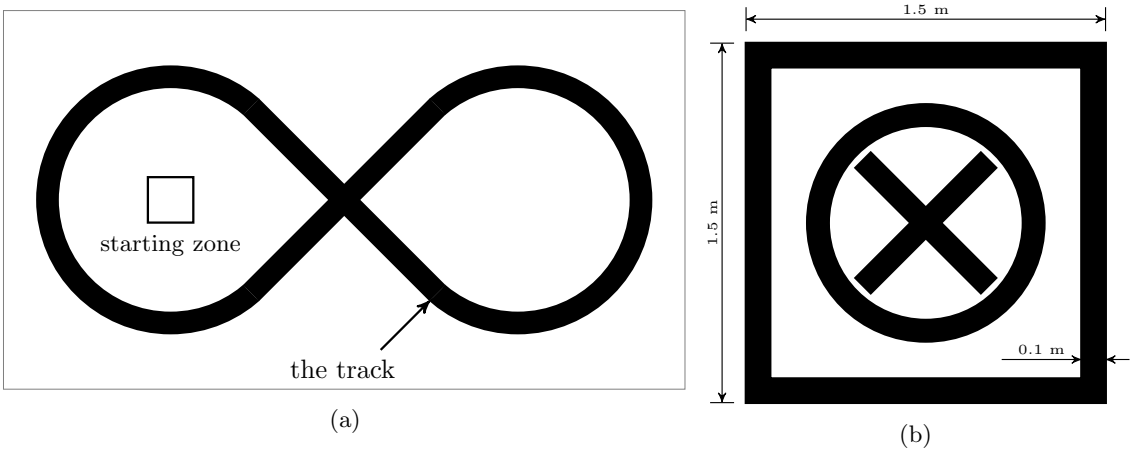


Figure 1: (a) a schematic image of the arena, showing the track for the ground vehicle, (b) the visual marker attached to the helipad of the ground vehicle, as described in the rules of the competition.

The proposed platform is a multirotor vehicle based on a DJI F550 hexacopter frame equipped with the DJI E310 propulsion system. Most components were chosen as individual and commercially available parts, to maximize the simplicity of the system, minimize the cost and to allow custom modifications if needed for any particular task. Key components are shown in Figure 2. See (Spurny et al., 2018) for a different configuration of the system, proposed for the MBZIRC treasure hunt challenge. A flight controller board is required to allow basic flight capability. The PixHawk flight controller (Meier et al., 2012) was chosen for its open source firmware and for its well-documented interface, which allows us to connect it to a high-level onboard computer. PixHawk contains sensors such as gyroscopes, accelerometers, an atmospheric pressure sensor, a magnetometer, and GPS, and it produces a single position, velocity and orientation estimate of the UAV in global world frame by their measurements.

Onboard computations are performed on an Intel NUC-i7 computer with an Intel i7 processor and 8 GB of RAM. The computer is installed with GNU Linux Ubuntu 16.04 and the Robot Operating System (ROS) in the Kinetic version. The Robot Operating System is a middleware library for C++ and Python programming languages. It provides a convenient way of building a complex system of applications with the asynchronous exchange of messages. An ecosystem of existing programs exists covering functionalities such as visualization, logging and data sharing, geometric transformations, etc. Sensor drivers are often found with the ROS interface already integrated, which makes them simpler to integrate.

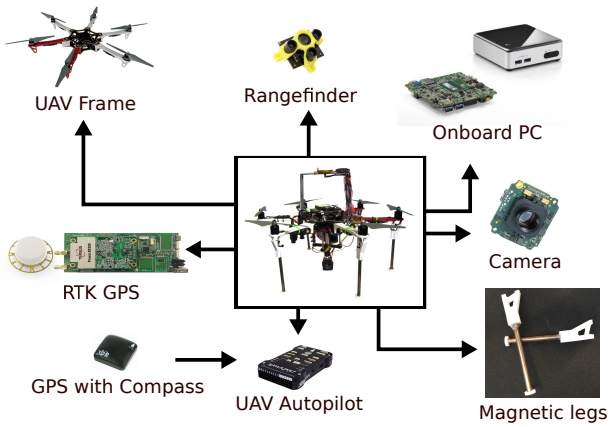


Figure 2: Schematic of individual hardware modules on the UAV

To improve the localization accuracy of the UAV in space, we integrated the PRECIS-BX305 GNSS RTK BOARD differential GPS receiver (Tersus-GNSS, 2017). Differential RTK (Real Time Kinematics) GPS uses a ground base station to transmit corrections to the UAV, which practically eliminates GPS drift. The TeraRanger time-of-flight laser rangefinder (Ruffo et al., 2014) serves two purposes. During a flight, it measures the distance to the ground, which is used to improve the estimation of the UAV height. In the landing task, during touchdown on the ground vehicle, it serves as a trigger for switching off the propellers. To detect the car, a single Matrix-vision mvBlueFOX-MLC200w camera is mounted on a fixed, down-facing mount beneath the UAV. A SuperFisheye lens was chosen to maximize the chance of detection in the final stages of landing when the landing pattern is close to the camera. Its global shutter provides images free of the rolling shutter effect.

3 System structure

The guidance law presented in this paper is a modular pipeline consisting of components which are depicted in Figure 3. The following paragraphs give a list of the components, which are subsequently described in sections 4 to 9 of this paper.

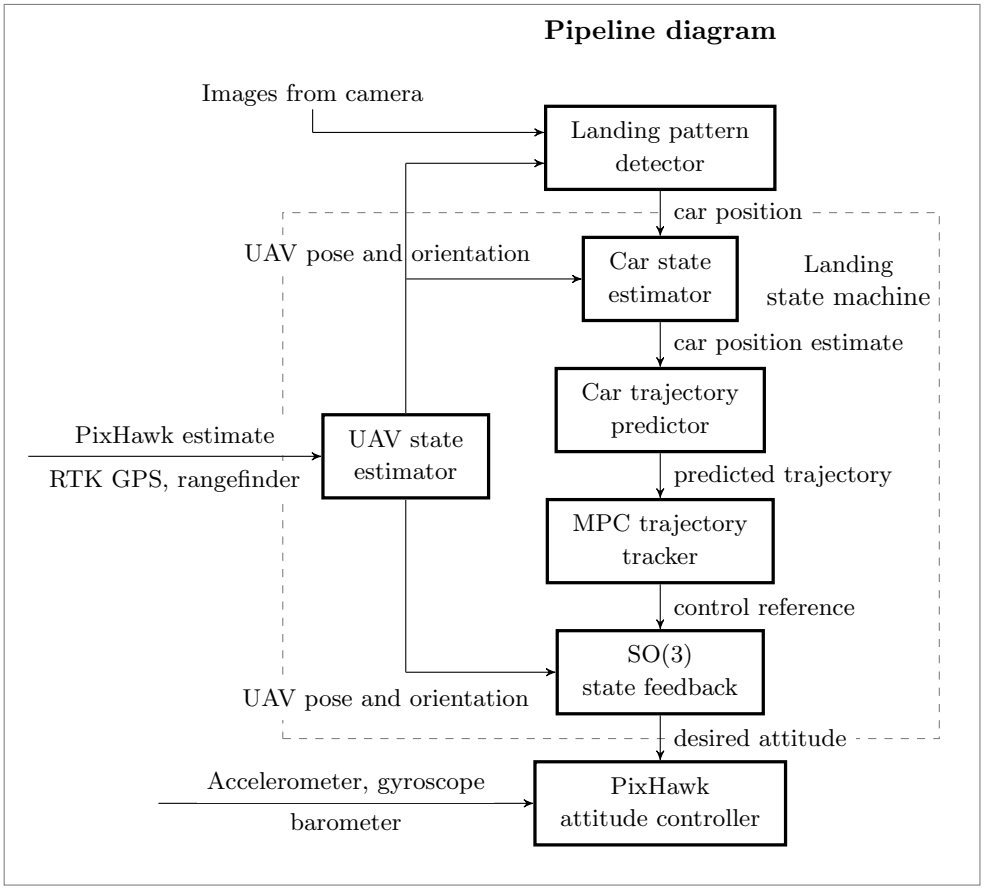


Figure 3: Scheme of the software pipeline for landing on a moving vehicle. See Section 4 for *Landing pattern detector*, Section 6 for *Car state estimator* and *Car state predictor*, Section 7 for *UAV state estimator*, Section 8 for *MPC trajectory tracker* and Section 9 for *SO(3) state feedback*. The dashed line surrounds the parts which are controlled by a landing state machine, later discussed in Section 5.

The first component is the **landing pattern detector** (presented in Section 4), which provides measure-

ments of car position in the world frame coordinate system. Position measurements are processed by the **car state estimator** (Section 6), using an Unscented Kalman Filter. Unmeasured states such as acceleration and heading are required to fully predict the future trajectory of the car. The **car state predictor** calculates the future trajectory of the car, starting from the latest state estimate and using the same model and the same non-holonomic model as is used for the estimation. The predicted future trajectory serves as a reference for the **MPC tracker** (Section 8), which minimizes a quadratic error of UAV future states over a prediction horizon to fly precisely above the car given the dynamical constraints of the aircraft. The MPC tracker then outputs desired states (position, velocity, and acceleration) to the **state feedback controller** (Section 9). The state feedback controller, being the last part of the pipeline implemented in the high-level computer, produces attitude and thrust commands for the PixHawk flight controller.

The **car state estimator** serves two purposes within our pipeline. First, it filters the incoming signal from the **landing pattern detector**, the measurement variance of which is adjusted with respect to the UAV height. Secondly, it estimates unmeasured states (velocity, acceleration, heading, turn curvature), which are required to predict the car future movement. The resulting estimate is outputted at 100 Hz. Using the information from the car state estimator, we predicted the future movement of the car, using the same dynamic model as during the estimation. The curvature of the predicted trajectory is biased using a known map of the arena and the track on which the car was driven. The predicted trajectory is updated at 30 Hz.

In our pipeline, a trajectory tracker is responsible for generating a set of desired states of the UAV (position, velocity, and acceleration) to follow the trajectory generated by the car state estimator. It uses decoupled, third order translational dynamics to simulate a virtual UAV at 100 Hz. The virtual UAV is then controlled by Model Predictive Control with a 8 s prediction horizon, also at 100 Hz. States of the virtual UAV are sampled and are handed out to the state feedback controlled as a reference. Thanks to the MPC, the tracker provides the necessary feed-forward action to follow the known future path. The particular MPC control approach is based on previous work presented in (Baca et al., 2016), further extended to support the state constraints in velocity and acceleration.

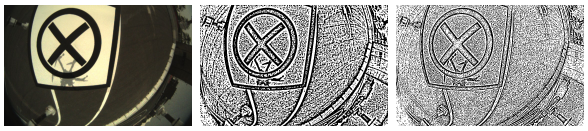
4 Visual localization of the landing platform

Robust, precise and fast detection of the landing pattern is a crucial ability to achieve reliable landing on moving vehicles. In the system designed for the MBZIRC 2017 competition, we relied on a color mvBlueFOX-MLC200w camera with a global shutter, which is important for recognizing moving objects from a camera on the fast-moving UAV. Another advantage of this light camera is the fast frame rate, 93 images per second, with resolution 752×480. Although a color camera was used, the image analysis was conducted after converting the obtained images to greyscale, thanks to the landing pattern being black and white. Using a Sunex DSL215 miniature SuperFisheye lens, the camera provides a horizontal field of view of 185°. It observes the car under the UAV even in the event of UAV tilting, so it is not necessary to use a gimbal camera stabilizer. This reduces the complexity and weight of the system. This scheme provides a very simple, cheap, and robust solution that can be applied in various landing scenarios beyond the MBZIRC competition.

Let us now briefly describe the image processing algorithm that was used for landing on the moving helipad in the MBZIRC competition. In this paper, we focus on general approaches that could be re-used for detecting landing patterns similar to the pattern used in MBZIRC 2017 to provide a complete system for autonomous vision-based landing. For special details on the technique adapted for localizing the MBZIRC pattern, see (Stepan et al., 2018), where all vision approaches employed by our team in the MBZIRC competition are summarized.

As was mentioned above, to ensure outdoor deployment in real scenarios, the detection procedure has to be robust to various weather conditions, changes in light intensity, and direct sunshine with shadows cast by the aircraft and other objects in the environment, such as the support structure of the MBZIRC arena.

Other requirements are low computational complexity to be able to use small and simple platforms, fast response, and the use of standard computer vision libraries, e.g., OpenCV, to provide simple implementation and reproducibility. Mainly the very fast response (50 FPS and more) and the availability of low computational power are contradictory requirements that are hardly achievable by state-of-the-art computer vision approaches and require the design of new methods suited for this special application.



(a) (b) (c)

Figure 4: (a) Image taken by the camera, (b) result of the adaptive threshold with box size 11 pixels, (c) adaptive threshold with box size 5 pixels. The box size is variable and depends on the thickness of the expected segments in the landing pattern and decreases with the height of the UAV.

In the proposed pattern detection approach, the first step is adaptive thresholding with a variable box, the size of which depends on the UAV height and the known parameters of the landing pattern (for examples with box size 5 pixels and 11 pixels, see Figure 4b and 4c, respectively). During the experiments, the size of the box spanned on the interval $[5, 21]$, where it was was defined as $2(w/12) + 1$, where w stands for the width of the detected white square of the landing pattern in pixels. The advantage of the adaptive threshold is its robustness to light intensity. The contours of the painted pattern (the circle and the lines in our case) are then simply detected in these segmented images.

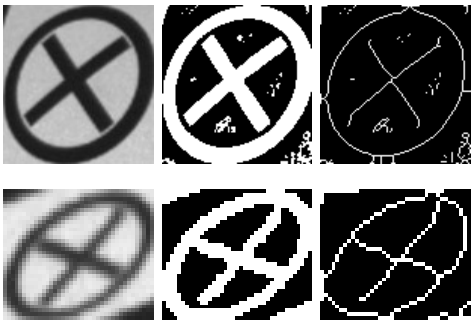
An important part of the algorithm is the undistort procedure. That needs to be applied to compensate the distortion caused by the SuperFisheye lens. The lens parameters can be identified using OpenCV and its fish-eye model. However, the undistort function provided in the library is too slow, and a new method needs to be designed. The employed approach, which is described in details in (Stepan et al., 2018), relies on the fact that the distortion coefficients are known in advance, and the scales required for computing undistorted coordinates can be precomputed.

(a) Original images of the landing pattern. (b) The landing pattern after morphological closing.

Figure 5: Operation of morphological closing was applied as a part of the recognition pipeline. The original camera images were cropped, which results in the images with resolution of 24×24 px.

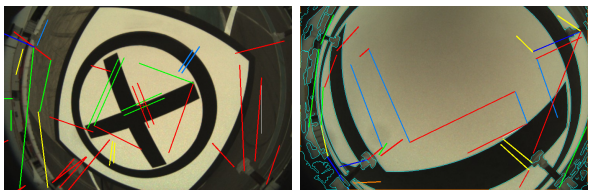
Robust detection of the MBZIRC 2017 landing pattern is based on detecting the outer circle and then the inner cross, to exclude false positive detections. Based on our experience, this design enables the pattern to be detected robustly in all phases of the landing approach. We can recommend it for other projects, where the autonomous UAV landing is required. A combination of the circle and the inner cross should also be used, if possible, in designs of landing patterns. In the initial phase of the approach, where the length of one of the axes of the ellipse (the detected circle) is shorter than 30 pixels, due to the long distance between the helipad and the UAV, the lines of the cross cannot be detected reliably. Then the cross is detected using the morphology operation *closing* and searching for areas similar in size. Circle detection is positively confirmed if four closed areas similar in size are found within the circle, see Figure 5 for example.

Later, if the UAV approaches closer to the helipad and the circle size is 30-150 pixels, the cross can be detected by Guo Hall thinning (Guo and Hall, 1989), which enables the lines inside the circle to be detected robustly. Positive detection of the landing pattern is confirmed if the crossing point of the two biggest lines



(a) (b) (c)

Figure 6: (a) Image taken by the camera, (b) application of the adaptive threshold with box size 11 pixels, (c) results of Guo Hall thinning.



(a) (b)

Figure 7: Line detection (a) if the cross is inside the circle and (b) if only a part of the landing pattern is detected.

of the cross inside the circle is detected near the center of the ellipse. Results of the landing pattern detection process using the Guo Hall thinning algorithm are shown in Figure 6.

Finally, if the circle is larger than 150 pixels, the cross is detected by recognizing its border, i.e., by detecting two pairs of parallel lines (the red and green lines in Figure 7a). This approach provides robust detection of the landing pattern if the entire circle is not visible in the image. Figure 7b depicts the pattern reconstructed only from two visible lines of the cross.

An estimate of the height of the UAV is required to select the proper approach for pattern detection. In the proposed system, this information is obtained using the laser rangefinder and its fusion with the onboard IMU (Inertial Measurement Unit) to compensate the deviation of the measurement caused by a vehicle with a variable height profile, when it appears under the approaching UAV. If the pattern is detected, its known parameters (the known dimension of the circle and the cross) can be used to make a precise measurement of the relative distance between the helipad and the UAV. This information is used for estimating the position of the landing pattern in the global world coordinate system, which is used as the desired state of the position controller.

5 Landing state machine

The autonomous task of the UAV is driven by a single state machine from autonomous takeoff to landing. The state machine (Figure 8) takes control of the UAV after a signal has been given by an operator or after the start time is reached. After taking off, it moves the UAV above the center of the map by switching to the *Fly to waiting point* state. The UAV waits for the car to appear in the field of view, while it hovers at the height of 8 m above the crossing of the two roads. Several factors have influenced the choice this strategy, e.g., the real speed and acceleration constraints of the UAV and the known trajectory and velocity profile of

the car. This approach minimizes the complexity of the system and also provides the shortest mean time for locating the target, given the mentioned constraints. By using this strategy, we also maximize the possible quality of the images being captured onboard the UAV since any movement of the UAV introduces a motion blur, which negatively influences the initial spotting of the target.

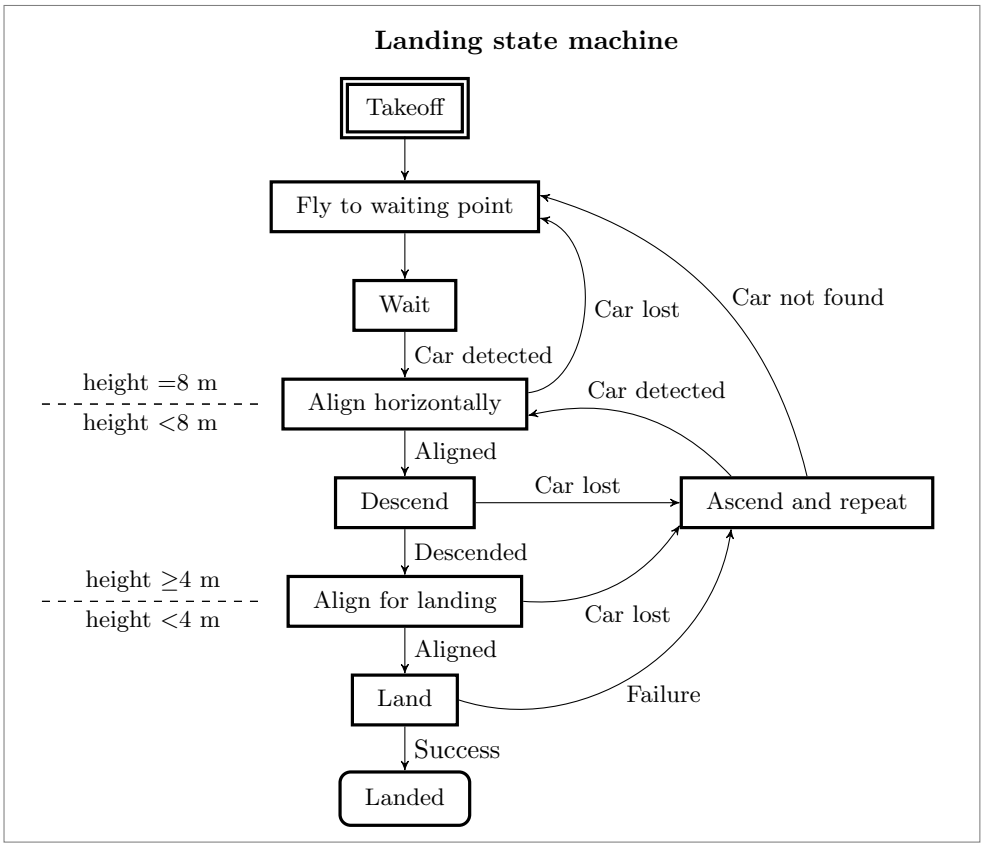


Figure 8: The autonomous flight, from takeoff to landing, is controlled by the *landing state machine*. The linear passage from the *Takeoff* state to the *Landed* state can be divided to three groups of states, based on the current height above ground – 8 m or higher, between 4 and 8 m and below 4 m. The state *Ascend and repeat* handles situations when the car was lost from the field of view of the camera.

When the target is first spotted, and the covariance of its state estimate exceeds a defined threshold, the UAV starts to align its horizontal position with the car while maintaining 8 m height (*Align horizontally* state). The aligning uses an approach strategy, which exploits the fact that the car velocity vector points towards the center of the map and thus towards the UAV. The approaching trajectory is created around a mutual meeting point $\mathbf{M}_{[t]}$, which is the closest point of the current position of the UAV $\mathbf{U}_{[t]}$ to the predicted trajectory of the car. The trajectory meets the following properties, where $\mathbf{C}_{[t]}$ is the current position of the car and $\mathbf{E}_{[t]}$ is the last point of the car prediction:

- the portion of the UAV trajectory in between $\mathbf{M}_{[t]}$ and $\mathbf{E}_{[t]}$ is found by sampling the predicted trajectory of the car starting at $\mathbf{M}_{[t]}$,
- the other portion of the UAV trajectory in between $\mathbf{U}_{[t]}$ and $\mathbf{M}_{[t]}$ requires creating a trajectory of the same time duration as the portion of the predicted trajectory of the car from $\mathbf{C}_{[t]}$ to $\mathbf{M}_{[t]}$,
- the resulting trajectory does not require motion (and thus a control action) in the direction parallel the current car motion, which means that the trajectory will not lose it from the field of view because the onboard camera is tilted away from the car.

In the special case, where $\mathbf{M}_{[t]}$ does not reside on the predicted trajectory of the car, the whole prediction is used as a reference for the MPC tracker. This situation may occur if the car is first spotted while driving away from the UAV.

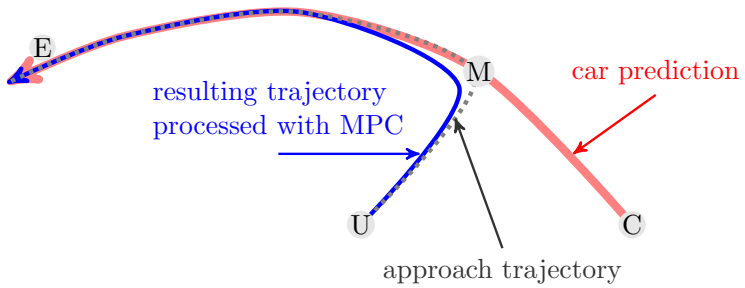


Figure 9: Illustration of the approach strategy. Point **M** represents the common meeting point of the UAV and the car, **U** is the current position of the UAV, **C** is the current position of the car and **E** marks the final point in the car prediction. The car predicted trajectory is shown in red, the approach trajectory is marked as dotted and the resulting feasible trajectory, optimized by MPC, is shown in blue.

After the UAV is aligned with the car horizontally within 1.5 m, the state machine switches to the *Descend* state. While in the *Descend* state, the height decreases to 4 m, the lowest height at which it is still possible to follow the car continuously given the particular UAV and camera configuration. Once a height of 4 m is reached, the states machine transitions to the *Align for landing* state, where it waits for two conditions to be met to initiate the final landing on the moving car. First, the UAV has to be aligned horizontally within 0.3 m of the center of the landing pattern. Second, transition to the *Land* state is allowed only above the straight parts of the track. Finally, the landing maneuver is executed, in which a fast descent is made to the roof of the car. During the landing, the motors are cut off by a signal from the down-facing laser range finder, or the whole landing is aborted due to a low height threshold being met (1.5 m above the ground). If the car is lost from sight during any of the previously mentioned phases of the landing, the state machine transitions to the *Ascend and repeat* state. In the *Ascend and repeat* state, the UAV ascends while it follows the car prediction based solely on the estimate. If the car is not detected again, the state machines transitions back to the *Fly to waiting point* state. If the car is detected while in the state *Ascend and repeat*, the horizontal alignment process is repeated via the state *Align horizontally.*

6 Ground vehicle state estimation and prediction

Several sub-problems have to be solved to follow a moving object with an autonomous helicopter. The first part of the pipeline, which provides visual detection of the landing surface, was presented in section 4. Motion estimation is necessary to compensate for inherent flaws in the data that are extracted from camera images. Information provided by the *landing pattern detector* is naturally skewed by phenomena such as *signal noise*, *false positive detections*, *irregular detection rate* and *time delay.* These issues are common for most real-world sensors and are usually addressed by filtration and fusion with other available data. Moreover, since the dynamical system of the vehicles is known and can be described by a mathematical model, we can use the knowledge to maximize the information we gain from camera observations of the car. In particular, we can estimate unknown states that are difficult or even impossible to measure directly, namely velocity, heading and curvature of the turn. Estimation of hidden states further allows us to predict the future movement of the vehicle.

6.1 LKF with a liner model

The simplest model that can be used to estimate and predict the motion of the car is a linear model of 2nd order translational dynamics. This model does not impose any constraints on the holonomy of the system,

and therefore lacks an estimate of the turning radius ($1/K$, where K is the turning curvature). States can easily be estimated using the Linear Kalman Filter as

$$\begin{aligned} \mathbf{x}_{[n+1]} &= \mathbf{A}\mathbf{x}_{[n]} + \mathbf{B}\mathbf{u}_{[n]}, \\ \mathbf{y}_{[n+1]} &= \mathbf{C}\mathbf{x}_{[n+1]} + \mathbf{D}\mathbf{u}_{[n]}, \end{aligned} \tag{1}$$

where $\mathbf{x}_{[n]} \in \mathbb{R}^n$ is the state vector and $\mathbf{u}_{[n]} \in \mathbb{R}^k$ is the input vector in sample n. We assume that $\mathbf{C} = \mathbf{I}$, $\mathbf{D} = \mathbf{0}$. The lateral motion of the car is captured by the matrices

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{2}$$

where the state vector is defined as $\mathbf{x}_{[n]} = (x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y})^T$. Testing with the linear model demonstrated satisfactory performance during linear motion of the car, but showed a significant tracking error when the car was turning. The scenario is showcased in the video `http://mrs.felk.cvut.cz/jfr2018landing-video1`.

6.2 UKF with a car-like model

To improve the car state estimation for non-linear motion, a different model is required, e.g., the nonholonomic car-like model

$$\begin{aligned} \mathbf{x}^o_{[n+1]} &= \mathbf{x}^o_{[n]} + \dot{\mathbf{x}}^o_{[n]}\Delta t, \\ \dot{\mathbf{x}}^o_{[n+1]} &= \begin{pmatrix} \cos\phi_{[n+1]} \\ \sin\phi_{[n+1]} \end{pmatrix} v_{[n+1]}, \\ \phi_{[n+1]} &= \phi_{[n]} + \dot{\phi}_{[n]}\Delta t, \\ \dot{\phi}_{[n+1]} &= K_{[n]} v, \\ v_{[n+1]} &= v_{[n]} + a_{[n]}\Delta t, \\ K_{[n+1]} &= K_{[n]} + \dot{K}_{[n]}\Delta t, \end{aligned} \tag{3}$$

where $\mathbf{x}^o_{[n]} = (x, y)^T_{[n]}$ is the position of the car in the global coordinate system, $\phi_{[n]}$ is its heading, $K_{[n]}$ is the curvature of its turn, $v_{[n]}$ is its scalar velocity, $a_{[n]}$ is its scalar acceleration, and Δt is the time difference. Car-like model better reflects the physics of the car motion thanks to adding non-holonomic constraints and effectively by coupling the heading with the curvature of its turn. Estimates of the heading of the vehicle allow its motion to be tracked while the onboard camera is oriented properly to maximize successful detection even, while the car is in turn. The Unscented Kalman Filter (UKF) (Wann and van der Merwe, 2000) was used as a filtration method and as a predictor. In contrast with LKF, UKF utilizes a general function as a form of model iteration. Covariance of the estimated hypothesis is transformed as a set of sampled points of the ellipsoid, which is later reconstructed using the known prior distribution of the points. Therefore, it is not required to differentiate the model function, as it would be using nowadays obsolete Extended Kalman Filter (EKF), which uses a linear approximation of the model. Due to the high speed of the car, UKF is needed for its robustness and better performance, comparing to EKF and LKF.

6.3 Ground vehicle state prediction

As we will discuss below in section 8, knowing the future trajectory of the ground vehicle is a key element in tracking its motion with the unmanned aircraft. To predict the future trajectory, the same model as for its state estimation is applied, creating a discrete, time-parameterized trajectory in 2D space. The output of the UKF estimator is directly used as an initial condition for the prediction. Additionally, thanks to the

knowledge of all states of the dynamical model in Equation (3), the predicted trajectory can be offset to compensate for the delay in the vision system. The prediction for n future steps takes form of

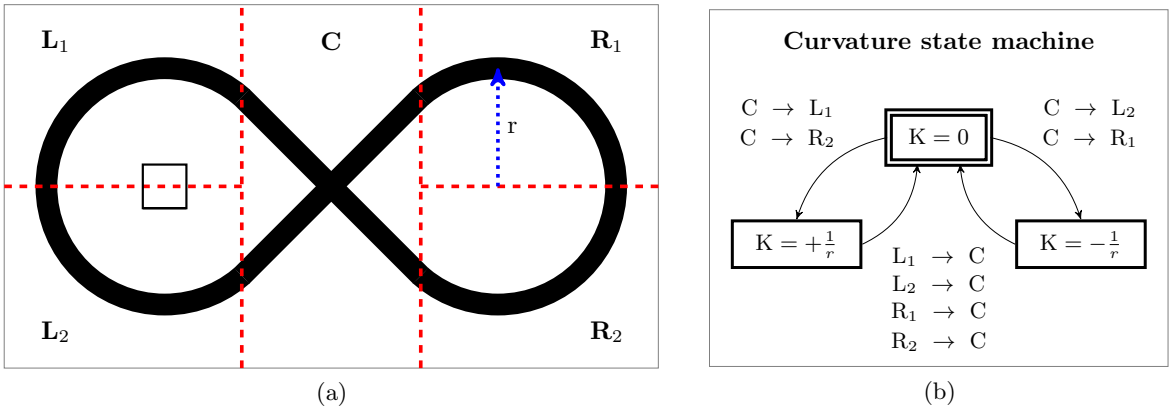
$$\mathbf{q}_{[n+1]} = f(\mathbf{q}_{[n]}), \forall n = 0, \ldots, n-1, \tag{4}$$

where $f()$ is the model function according to Equation (3), $\mathbf{q}_{[n]}$ is a complete state vector defined as

$$\mathbf{q}_{[n]} = \left[{\mathbf{x}^o_{[n+0]}}^T, \phi_{[n+1]}, K_{[n+1]}, v_{[n+1]}, a_{[n+1]} \right]^T \tag{5}$$

and $\mathbf{q}_{[0]}$ is the initial condition provided in real-time by the UKF estimator.

In general, the car can ride with changes in the curvature and acceleration depending on the driver. However, the competition rules specify the shape of the track in the arena (see Figure 1), and also the speed profile, which allows us to bias the estimate or the prediction to achieve more accurate trajectory tracking.



(a) (b)

Figure 10: (a) The competition arena is divided into five areas. The center area (C) contains track with zero curvature. The four corner areas (L_1, L_2, R_1, R_2) contain track with curvature $|K| = 1/r$. (b) State machine producing the bias for the curvature of the car. Inputs to the state machine are transitions of the car between the different parts of the arena.

The first level of biasing the curvature relies solely on the known curvature in different parts of the track. Knowledge of coordinates of only a few pre-defined points is required to identify the curvature for any given coordinates on the map. To bias the curvature, the arena is divided into five separate parts. Figure 10a shows the partitioning into the corner parts $\mathbf{L}_1$, $\mathbf{L}_2$, $\mathbf{R}_1$, $\mathbf{R}_2$, where the curvature of the turn is $|K| = 1/r$, and the center area $\mathbf{C}$, where $K = 0$. However, the sign of the curvature depends on the direction in which the car is driving in the particular part. To solve the estimation of the direction, we designed a simple state machine (Figure 10b), which describes all possible transitions between the different parts of the map that correspond to a change in the curvature. Later, when creating the prediction of the car using the model (Equation 3), the initial curvature $K_{[0]}$ is set based on the current state of the curvature state machine. Further states of the prediction undergo the same process with an identical temporary state machine, to ensure that the curvature is correctly biased throughout the whole future.

6.4 Biasing the predicted trajectory to the measured track

To further improve the performance of the tracking in the particular scenario of the competition, the predicted trajectory of the car was biased towards the known global GPS coordinates of the track. The predicted trajectory of the car is snapped towards the analytically described track. The snapping is proportional to the covariance of the predicted points. This leaves space for the car to drive off center of the track since the initial part of the prediction is close to the estimate. Figure 11 also illustrates a typical example of a predicted and snapped trajectory.

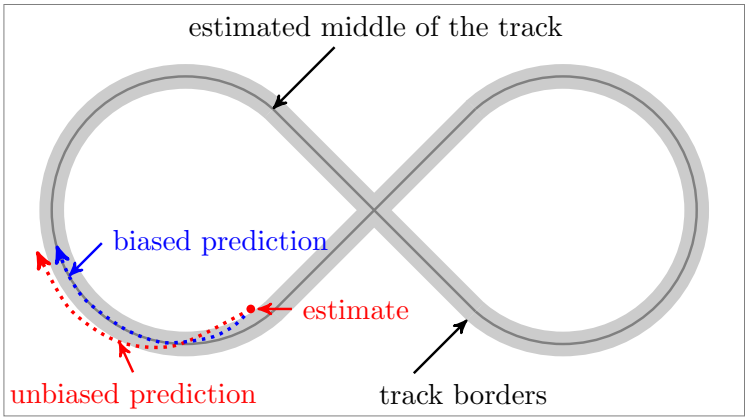


Figure 11: Scheme of the real track, on which the car was supposed to drive, with the path denoted in the middle of the track, to which the car prediction was biased. An example of a predicted (red) and a biased trajectory (blue) is shown.

7 UAV state estimation

Autonomous UAV control relies on an estimate of the states of the UAV dynamical system. Namely, knowledge of position and velocity (both vertical and horizontal) is required to control the movement for precise landing on the moving vehicle. Our platform is equipped with several independent sources of information, which are fused to obtain a single, reliable and smooth estimate of the UAV pose. It is essential to ensure smoothness of the resulting signal since SO(3) (Section 9) state feedback is sensitive to noise.

The main source of data for both the vertical axis and the horizontal axis in the proposed system is the PixHawk flight controller. Its Extended Kalman Filter fuses traditional inertial sensors – a three-axis accelerometer and a gyroscope with an height pressure sensor and a GPS receiver. Although the aircraft is already capable of autonomous flight with this off-the-shelf setup, we make use of other sensors, a time-of-flight laser rangefinder (Ruffo et al., 2014) and a differential GPS receiver, to provide more precise localization,

7.1 Horizontal position estimation

Position estimation in the lateral axes is based on the estimate provided by PixHawk, namely position $\mathbf{x}^p$, and velocity $\dot{\mathbf{x}}^p$. Although its precision may be satisfactory locally for short periods of time, it is prone to heavy drift in time spans of minutes. To correct this drift and thus to ensure repeatability of the experiments and, e.g., locating the dropping zone, the horizontal position from PixHawk is corrected by differential RTK GPS. Position measurements from the RTK GPS receiver are fused using the Linear Kalman Filter with the model

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \Delta t \\ \Delta t \end{pmatrix}, \tag{6}$$

where $\mathbf{x}^e_{[n+1]} = \mathbf{A}\mathbf{x}^e_{[n]} + \mathbf{B}\mathbf{u}_{[n]}$ is the linear system equation, $\mathbf{x}^e_{[n]} = (x, y)^T_{[n]}$ is the state vector finally used for control, and $\mathbf{u}_{[n]}$ is the system input. According to our experience,

$$\mathbf{x}^p_{[0]} + \sum_{n=0}^{k} \dot{\mathbf{x}}^p_{[n]} \Delta t_{[n]} = \mathbf{x}^p_{[k]}, \forall k \in \mathbb{N} \tag{7}$$

does not hold for the position and velocity estimate provided by PixHawk. This is a very useful observation for somebody building a fully autonomous UAV system using an off-the-shelf controller. The input vector $\mathbf{u}$ consists of velocities obtained by integrating differentiated positions $\mathbf{x}^p$, which ensures that our filter does not introduce any more drift into the resulting estimate when no RTK GPS corrections are involved. In

situations when the position is not being corrected, the resulting estimate follows the same relative state trajectory as $\mathbf{x}^p$, just shifted according to the latest correction.

In other words, the position estimate fused by the PixHawk is used as a main source of information, regardless of whether the RTK GPS is currently available. The difference is, the PixHawk position estimated is updated by input

$$\mathbf{u}_{[k]} = \left(\mathbf{x}^p_{[k]} - \mathbf{x}^p_{[k-1]}\right) / \Delta t_{[k]} \tag{8}$$

which results in position update

$$\mathbf{x}^e_{[k+1]} = \mathbf{x}^e_{[k]} + \left(\mathbf{x}^p_{[k+1]} - \mathbf{x}^p_{[k]}\right), \tag{9}$$

that follows the PixHawk estimate when left uncorrected. However, when the RTK GPS is available, the estimate $\mathbf{x}^e_{[k+1]}$ can be freely corrected by the LKF, effectively adding an offset using the more precise source of information. By using such approach, the correction will be still applied even during long outages of the RTK GPS system.

7.2 Vertical position estimation

UAV state estimation relies less on the PixHawk in the vertical axis than in the horizontal axis. Height corrections come not only from differential RTK GPS but also from the down-facing TeraRanger rangefinder and from the landing pattern detector, which can provide height data when flying above the car. The estimator provides an option to switch between these sources of data, depending on the current state of the landing state machine. The PixHawk height is fused by the same technique as in the horizontal system, as denoted by Equation 8.

It is feasible to correct the height using the TeraRanger rangefinder when flying above uneven ground, but it cannot be used reliably when the down-facing sensor is obstructed by the car. However, RTK GPS can provide precise relative height measurements, but only when RTK FIX has been established. RTK FIX is one of several precision states of RTK GPS, which provides the best accuracy and guarantees a correct position signal. The standard states include RTK FLOAT and DGPS, but only RTK FIX guarantees the precision need for actually correcting the built-in PixHawk GPS. Finally, correcting the height using data from the *landing pattern detector* might bring unexpected steps in the signal due to false positive detections or signal dropouts. Since none of the additional sources is completely reliable, we employed a safety mechanism for detecting anomalies, which can toggle off any of the above-mentioned sensors.

8 Predictive trajectory tracking

While the state feedback described in section 9 provides precise position and velocity control, it requires a smooth and feasible reference. The reference consists of all states of the translational dynamics – position, velocity, and acceleration it is provided at 100 Hz, the same rate as the resulting control signal. There are various ways of creating the reference. Typically, thanks to the differential flatness of the UAV dynamical system, QP optimization can be performed to find a polynomial, given the initial and final state conditions (Mellinger and Kumar, 2011), which can then be derived and sampled to create the reference. In our case, we chose to generate the reference using a Model Predictive Control approach. The following text describes a simpler variant of the original approach designed for multiple vehicles, which was proposed in (Baca et al., 2018).

The Model Predictive Control Tracker (MPC Tracker) uses a QP formulation of the minimal sum-of-squares problem, where the optimal control action $\mathbf{u}$ is found for a future prediction horizon of states

$\mathbf{x}_{[n]} = (x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}, z, \dot{z}, \ddot{z})^T_{[n]}$ by minimizing the function

$$\begin{aligned} \mathrm{V}\left(\mathbf{x}_{[0,\dots,m-1]}, \mathbf{u}_{[0,\dots,m-1]}\right) = \quad & \frac{1}{2}\sum_{i=1}^{m-1}\left(\mathbf{e}^T_{[i]}\mathbf{Q}\mathbf{e}_{[i]} + \mathbf{u}^T_{[i]}\mathbf{P}\mathbf{u}_{[i]}\right), \\ \text{s.t.} \quad & \mathbf{x}_{[0,\dots,m-1]} \geq \mathbf{x}_L, \\ & \mathbf{x}_{[0,\dots,m-1]} \leq \mathbf{x}_U, \end{aligned} \tag{10}$$

where $\mathbf{e}_{[n]} = \mathbf{x}_{[n]} - \tilde{\mathbf{x}}_{[n]}$ is the control error, $\tilde{\mathbf{x}}_{[n]}$ is the setpoint for the MPC, m is the length of the prediction horizon, and $\mathbf{x}_L$ and $\mathbf{x}_U$ represent box constraints on states. The control error $\mathbf{e}_{[n]}$ requires the formation of a general prediction of $\mathbf{x}_{[n]}$, which has been described in (Baca et al., 2016). In our case, the optimized control action is not directly used to control the real UAV. Instead, it controls a model of the UAV translational dynamics in real-time simulation. States of the simulated model are then sampled at 100 Hz to create the reference for the state feedback.

An important notion is a difference between the trajectory setpoint $\tilde{\mathbf{x}}$ and the reference, which is generated by the MPC tracker. The trajectory setpoint $\tilde{\mathbf{x}}$ is provided by high-level planning or, in this case, by the car predictor. No requirements are imposed on $\tilde{\mathbf{x}}$ in general. By contrast, the reference produced by the MPC tracker is feasible, satisfies UAV dynamics and state constraints, and serves as a control reference for the SO(3) state feedback (see section 9). The inherent predictive nature of MPC provides trajectory tracking optimizing actions over the future, and this makes it ideal for tracking moving targets.

The simulated model is an LTI system covering the 3rd order translational dynamics of the UAV with the system matrices

$$\mathbf{A} = \begin{pmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \Delta t & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta t \end{pmatrix}, \tag{11}$$

where $\Delta t = 0.01$ s. The same matrices are used to formulate the MPC prediction. In our MPC formulation, Δt is different for the first iteration ($\Delta t = 0.01$ s) and for all the other iterations ($\Delta t = 0.2$ s). This allows the simulation to be controlled smoothly if the MPC is executed at 100 Hz, while there is a relatively sparse distribution of further states. Sparse distribution provides a much longer prediction horizon than these would normally be with Δt being constant. As in traditional MPC, only the control action in the first step is used to control the model in the simulation. Before action from the second step would be required, a new instance of the optimization task is formulated and solved, starting from new initial conditions. This results in a fresh control action for the next simulation step. This method is valid only if the MPC can be solved repeatedly within the 0.01 s simulation step.

Penalization parameters $\mathbf{Q}$ and $\mathbf{P}$ in (10) have been found empirically as

$$\mathbf{Q} = \begin{pmatrix} 5000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 800 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 800 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 800 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 800 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 800 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 800 & 0 \end{pmatrix}, \mathbf{P} = \begin{pmatrix} 500 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 500 \end{pmatrix}. \tag{12}$$

As in our previous work (Baca et al., 2016), we used the *move blocking* technique to effectively prolong the prediction horizon while maintaining the computational complexity. The particular control action distribution for the MBZIRC competition was

$$\mathbf{U} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 5 & 5 & 5 & 5 & 5 & 10 \end{pmatrix}, \tag{13}$$

which results in in 8 s prediction horizon with only 33 variables in the optimization task. Without *move blocking* 120 variables would be required to solve the control problem.

As defined in Equation (10), MPC handles state constraints as linear inequalities. We impose maximum acceleration and velocity box constraints on the UAV to ensure safe and feasible resulting trajectories. The optimization being solved lies in the family of Linearly Constrained Quadratic Programming, which acquires a global optimum in a convex polytope. A custom solver based on sequential closed-form solution (Algorithm 1) is implemented to ensure guaranteed real-time performance, while sacrificing optimality. According to (Rossiter, 2003), in control design we should, if possible, rely on input pre-shaping rather than more complicated control-oriented solutions. Thus in every iteration, a reference trajectory is first pre-shaped to satisfy the velocity constraints by a low-pass filter as well as to initiate it in the current state of the UAV. Then the unconstrained problem is solved analytically as in (Baca et al., 2016). Although the resulting trajectory approximately satisfies the velocity constraints, the acceleration constraints are in general violated. In the second step, the acceleration part of the optimized trajectory is again pre-shaped to satisfy the acceleration constraints. Then we solved the unconstrained MPC again, however now with acceleration double-integrated to serve as the new position reference. The result of the second MPC step is a solution which, according to our empirical results, satisfies both acceleration and position constraints within 10% margin of error, which is a tolerable trade-off for complete control over the deterministic execution of the algorithm.

Algorithm 1 Sequential closed-form MPC

```
1: procedure ITERATEMPC
2: input:
3:     reference ← desired reference
4:     current_state ← current state of the UAV
5:     max_v ← maximum velocity constraint
6:     max_a ← maximum acceleration acceleration
7: execution:
8:     # the first iteration of the MPC generates trajectory not violating velocity constraints
9:     reference ← preshapeVelocity(current_state, reference, max_v)
10:    trajectory, control_input ← analyticMPC(current_state, reference)
11:
12:    # the second iteration of the MPC generates trajectory not violating acceleration constraints
13:    reference ← preshapeAcceleration(current_state, trajectory, max_a)
14:    trajectory, control_input ← analyticMPC(current_state, reference)
15:
16:    return trajectory
```

MPC-based trajectory tracking operates in two modes. The first simple positioning mode, used mainly for short-distance position changes, accepts either relative or absolute position commands and tries to reach a given position in the fastest way with respect to the MPC scheme. The second trajectory-following mode utilized by *high-level trajectory planning* uses a precomputed path plan. It tries to track the trajectory precisely while respecting the plan waypoints schedule, which is crucial for precise landing on the moving vehicle.

9 Feedback control

The position controller uses the estimated state as feedback to follow the trajectories given as an output of the high-level trajectory planner. In many previous works, a backstepping approach is used for UAV control, because the attitude dynamics can be assumed to be faster than the dynamics governing the position, so linearized controllers are used for both loops (Mellinger et al., 2013; Weiss et al., 2011; Heriss et al., 2012). However, we need the system to be capable of large deviations from the hover configuration during operations like fast mapping of objects, or for heavy wind compensation. We therefore use a nonlinear controller. Let us consider an inertial reference frame denoted by $\begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{bmatrix}$ and a body reference frame centered in the center of mass of the vehicle with an orientation denoted by $\mathbf{R} = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3\end{bmatrix}$, where $\mathbf{R} \in SO(3)$. The dynamic model of the vehicle can be expressed as

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v}, \\ m\dot{\mathbf{v}} &= f\mathbf{R}\mathbf{e}_3 + mg\mathbf{e}_3, \\ \dot{\mathbf{R}} &= \mathbf{R}\hat{\mathbf{\Omega}}, \\ \mathbf{J}\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} &= \mathbf{M}, \end{aligned} \tag{14}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the Cartesian position of the vehicle expressed in the inertial frame, $\mathbf{v} \in \mathbb{R}^3$ is the velocity of the vehicle in the inertial frame, $m \in \mathbb{R}$ is the mass, $f \in \mathbb{R}$ is the net thrust, $\mathbf{\Omega} \in \mathbb{R}^3$ is the angular velocity in the body-fixed frame, and $\mathbf{J} \in \mathbf{R}^{3\times3}$ is the inertia matrix with respect to the body frame. The hat symbol $\hat{}$ denotes the skew-symmetry operator according to $\hat{\mathbf{x}}\mathbf{y} = \mathbf{x}\times\mathbf{y}$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$, g is the standard gravitational acceleration, and $\mathbf{e}_3 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^\top$. The total moment $\mathbf{M} \in \mathbb{R}^3$, with $\mathbf{M} = \begin{bmatrix}M_1 & M_2 & M_3\end{bmatrix}^\top$, along all axes of the body-fixed frame and the thrust $\tau \in \mathbb{R}$ are control inputs of the plant. The dynamics of the rotors and propellers are neglected, and it is assumed that the force of each propeller is directly controlled. The total thrust, $f = \sum_{j=1}^{6} f_j$, acts in the direction of the z axis of the body-fixed frame, which is orthogonal to the plane defined by the centers of the four propellers. The relationship between a single motor thrust f_j, the net thrust f, and the moments $\mathbf{M}$ can be written as

$$\begin{bmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ sd & 1 & sd & -sd & -1 & -sd \\ -cd & 0 & cd & cd & 0 & -cd \\ -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} \tag{15}$$

where $c = \cos(30°)$, $s = \sin(30°)$ and d is the distance from the center of mass to the center of each rotor in the $\mathbf{b}_1$, $\mathbf{b}_2$ plane. For non-zero values of d, eq. (15) can be inverted using the right pseudo-inverse.

For control, we build on the work in (Lee et al., 2013) and in (Mellinger and Kumar, 2011) with control inputs $f \in \mathbb{R}$ and $\mathbf{M} \in \mathbb{R}^3$ chosen as

$$\mathbf{M} = -k_R\mathbf{e}_R - k_\Omega\mathbf{e}_\Omega + \mathbf{\Omega} \times J\mathbf{\Omega} - J\left(\hat{\mathbf{\Omega}}R^TR_c\mathbf{\Omega}_c - R^TR_c\dot{\mathbf{\Omega}}_c\right), \tag{16}$$

$$f = -\left(-k_x\mathbf{e}_x - k_{ib}R\int_0^t R(\tau)^T\mathbf{e}_x d\tau - k_{iw}\int_0^t \mathbf{e}_x d\tau - k_v\mathbf{e}_v - mg\mathbf{e}_3 + m\ddot{\mathbf{x}}_d\right) \cdot R\mathbf{e}_3, \tag{17}$$

with $\ddot{\mathbf{x}}_d$ the desired acceleration, and k_x, k_v, k_R, k_Ω positive definite terms. We extend the referenced controllers by including two integral terms which accumulate error in the body frame and in the world frame, respectively. We include both terms to provide the opportunity to capture external disturbances (*e.g.,* wind) separately from internal disturbances (*e.g.,* an inefficient prop or a payload imbalance), particularly when the vehicle is permitted to yaw or rotate about the vertical axis. The thrust and the moments are then

converted to motor rates according to the characteristic of the proposed vehicle. Subscript C denotes a commanded value, and $R_C = \begin{bmatrix}\mathbf{b}_1, & \mathbf{b}_2, & \mathbf{b}_3\end{bmatrix}$ is calculated as

$$\mathbf{b}_{2,des} = \begin{bmatrix}-\sin\psi_{des}, & \cos\psi_{des}, & 0\end{bmatrix}^\top, \; \mathbf{b}_{3,C} = \frac{\mathbf{f}}{||\mathbf{f}||}, \quad \mathbf{b}_{1,C} = \frac{\mathbf{b}_{2,des} \times \mathbf{b}_3}{||\mathbf{b}_{2,des} \times \mathbf{b}_3||}, \quad \mathbf{b}_{2,C} = \mathbf{b}_3 \times \mathbf{b}_1,$$

$$\dot{\mathbf{b}}_{2,des} = \begin{bmatrix}-\cos\psi_{des}\dot{\psi}_{des}, & -\sin\psi_{des}\dot{\psi}_{des}, & 0\end{bmatrix}^\top, \quad \dot{\mathbf{b}}_{3,C} = \mathbf{b}_{3,C} \times \frac{\dot{\mathbf{f}}}{||\mathbf{f}||} \times \mathbf{b}_{3,C},$$

$$\dot{\mathbf{b}}_{1,C} = \mathbf{b}_{1,C} \times \frac{\dot{\mathbf{b}}_{2,des} \times \mathbf{b}_{3,C} + \mathbf{b}_{2,des} \times \dot{\mathbf{b}}_{3,C}}{||\mathbf{b}_{2,des} \times \mathbf{b}_{3,C}||} \times \mathbf{b}_1, \quad \dot{\mathbf{b}}_{2,des} = \dot{\mathbf{b}}_{3,C} \times \mathbf{b}_{1,C} + \mathbf{b}_{3,C} \times \dot{\mathbf{b}}_{1,C},$$

$$\hat{\mathbf{\Omega}}_C = R_C^\top \dot{R}_C. \tag{18}$$

Note that here we have to define $\mathbf{b}_{2,des}$ based on the yaw, instead of defining $\mathbf{b}_{1,des}$ as it was defined in (Mellinger and Kumar, 2011), due to a different Euler angle convention (we use the ZYX convention instead of ZXY). The definition of the tracking errors can be found in (Spurny et al., 2018).

10 Experimental evaluation

The platform was thoroughly tested during all stages of development. All features were developed and verified in simulation before experiments on the real hardware. The Gazebo simulator and the Robot Operating System allowed the same instances of software to be implemented and tested in simulation and also in the field. To ensure safety, all experiments were performed in unoccupied rural areas, and the UAV was supervised by a human operator at all times.

Supplementary multimedia material

A video attachment to this work is available at website `http://mrs.felk.cvut.cz/jfr2018landing`.

10.1 Experiments prior to the competition

The initial experiments involved testing the landing pattern detection and landing on a full-sized static target, as depicted in Figure 12a. State estimation based on the linear model of the moving target showed that following a non-linear motion requires a more descriptive car-like model. Figures 12a and 12b show the UAV following a car with the visual marker attached to its roof. Figure 13 depicts a test using a Linear Kalman Filter and a linear motion prediction of the future trajectory of the car. During numerous test runs, the UAV was able to follow the ground vehicles. However a significant position error was observed in turns of the path. The car-like model, which was tested later, exhibited significantly better performance than the linear model for following general trajectories, which include turning (see Figure 14).

The vision and guidance system were verified without taking of weather conditions into account. Although it was known that the competition would be held on concrete surface in summer weather, we tuned the landing pattern detection algorithm to various surfaces, including grass, snow, concrete and asphalt throughout the seasons to achieve maximal possible reliability and to account for unforeseen conditions during the competition. Figure 15 shows snapshots from experiments in winter conditions.

Experiments on autonomous landing, including the *Landing State Machine* (Section 5), are portrayed in Figure 14. Initially, the rate of a successful landing was 54% out in 22 trials. These experiments showed the need for a camera equipped with a SuperFisheye lens to improve pattern detection during the final stage of landing.

Figure 12: Photos from the first experiments in the field, a) shows a UAV hovering above a static target, b) shows a UAV tracking a vehicle moving along a linear trajectory, and c) depicts first attempts with tracking a ground vehicle moving in circles. The Linear Kalman Filter was used to estimate the states of the ground vehicle during this stage of development. Additional video material documenting these experiments can be found at `http://mrs.felk.cvut.cz/jfr2018landing-video1`.

Figure 13: Figures showing a UAV following the ground vehicle at a speed of 10 km/h. The motion of the car was estimated by the Linear Kalman Filter. The system was thoroughly tested on general trajectories of the ground vehicle. Video material for this experiment can be found at `http://mrs.felk.cvut.cz/jfr2018landing-video2`.

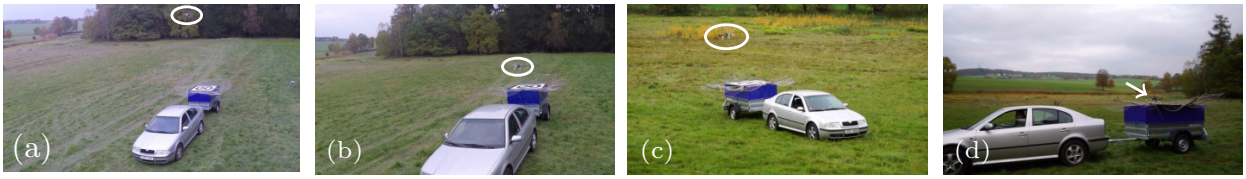


Figure 14: Experiments aimed at the autonomous following and landing on the ground vehicle using car-like motion model. Various stages of the Landing State Machine (see Section 5) are shown, a) the UAV tracking the car as it aligns horizontally, b) the descending phase, c) the UAV aligns for the second time before turning off the propellers and d) the UAV after the successful landing. Video summary of the experiments can be found at `http://mrs.felk.cvut.cz/jfr2018landing-video3`.

Figure 15: The performance of the system was tested in all weather conditions. Experiments in snowy and desert environments helped to fine-tune the computer vision for various lighting conditions and ground textures. A video showing a landing in a desert car be seen at `http://mrs.felk.cvut.cz/jfr2018landing-video4`.

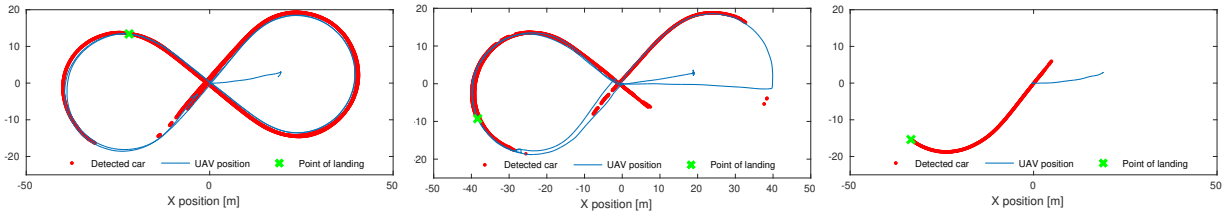
10.2 The competition trials

Each team that participated in the competition in Abu Dhabi had an option to take part in two rehearsal trials (2×30 minutes), two competition trials (2×15 minutes) and two rounds of the Ground challenge

Figure 16: A sequence of images from the second trial during the competition. The whole videos, as well as additional material including onboard footage, can be found at `http://mrs.felk.cvut.cz/jfr2018landing`.

(2×25 minutes). Our team competed in autonomous mode in both trials of the first challenge and the first round of the Grand Challenge. The second round of the Grand challenge was performed in manual mode, to allow manual operation on the ground robot at the same time. A combined manual and autonomous mode for different robots in the same trial was not allowed. In both trials of the landing challenge, we experienced a successful landing. Touchdown during trial 1 took place 143.2 s after the start. In trial 2 it took the UAV 84.6 s to land, which brought us the second place among all teams in the autonomous landing challenge, just behind the UAV of Beijing Institute of Technology, with a time of 63.4 s. Figure 16 depicts the autonomous landing using the proposed system during the first trial of the competition. Table 1 compares our results with these of other teams.

Independently of the three separate robotic challenges, in the Grand Challenge, the teams competed in all three challenges simultaneously. During the Grand Challenge, our system scored the fastest landing ever performed among all teams in the entire MBZIRC competition, with 25.1 s time from the start. Figure 19 shows the relative position and control error plots from the fastest trial. Figure 17 shows top-down plots of two trials from the landing challenge, and also the first trial from the Grand Challenge. The same trials are presented in Figures 18a, 18b and 18c, with the states of the landing state machine (Section 5) color-coded in the trajectory of the UAV.



(a) (b) (c)

Figure 17: Top-down view of all three successful trials: (a) 1:44 min flight, (b) 1:28 min flight, (c) 0:25 min flight.

Team	TRIAL 1	TRIAL 2	GRAND 1	GRAND 2
Beijing Institute of Technology	63,4	63,4	NQ	NQ
CTU in Prague, UPENN and UoL	143,2	84,6	**25,1**	M
University of Bonn	110,5	*not landed*	58,6	42,3
University of Catania	134,5	*falling off*	NQ	NQ

Table 1: Time in seconds of all successful autonomous landing attempts in the MBZIRC 2017 competition, when the car was moving at its maximum speed of 15 km/h. NQ - not qualified for the final challenge, M - manual mode applied due to other robots in the arena, *not landed* - UAV not landed due to a crash or time limit of 15 minutes exceeding, *falling off* - UAV touched the mobile landing platform but was not able to fix there, TRIAL 1 & 2 - trials of the MBZIRC Challenge 1, GRAND 1 & 2 - trials of the MBZIRC Grand Challenge.

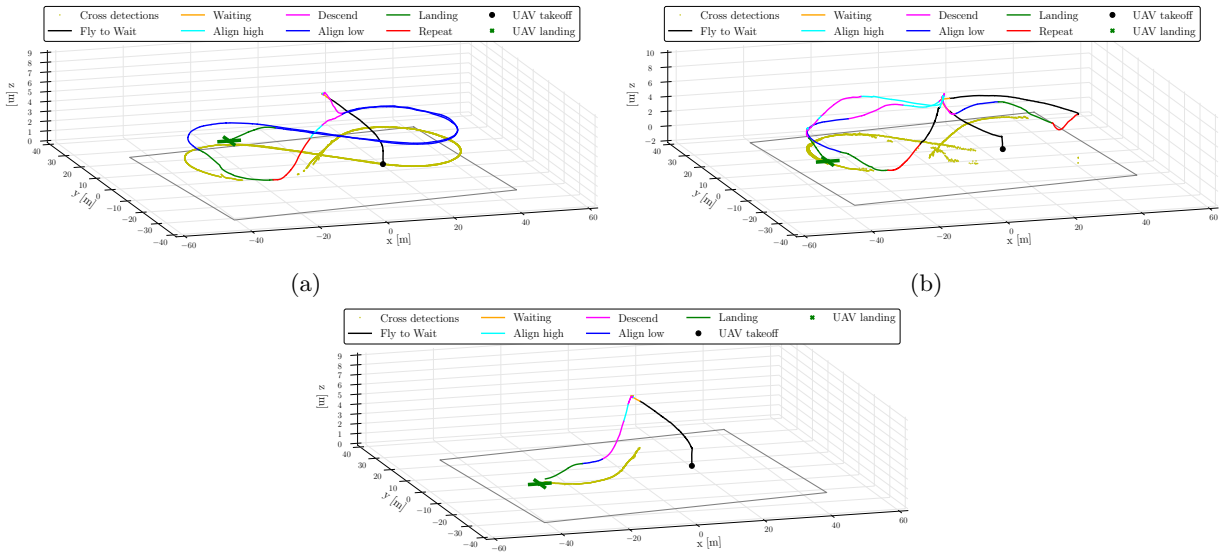
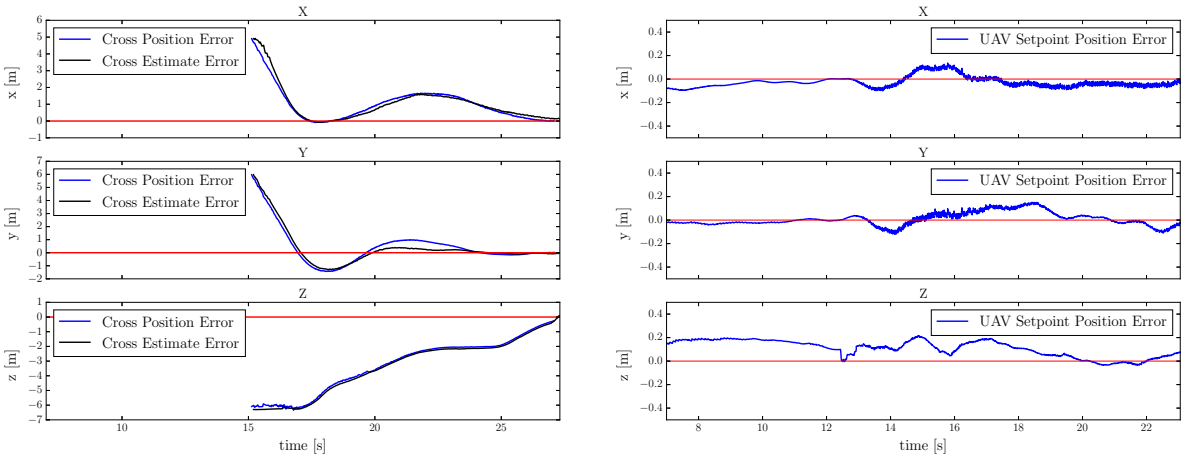


Figure 18: Three-dimensional plot of positions of the ground vehicle, as detected by the UAV, and the UAV during the (a) first competition trial, (b) second competition trial and (c) first Grand challenge trial. The trajectory of the UAV is color-coded according to the states of the *Landing State Machine*.



(a) Relative position of the car to the UAV.

(b) Position control error.

Figure 19: Plots of (a) the relative position of the detected and estimated ground vehicle, and the position of the UAV during the third (the fastest) landing and (b) the position control error during the flight.

11 Lessons learned

Although the competition results can be considered a major success, it was not without hurdles, mainly during implementation and testing of the proposed system. The proposed control pipeline consisting of estimator, predictor, tracker, and controller showed to be depended mainly on the performance of the car estimator. Tuning of the estimator parameters on real data was an essential factor which influences the overall performance of the remaining components in the pipeline. Hence we stress the significance of the real-world outdoor experiments above simulation, to obtain real sensor data. Finally, we cannot stress enough how important is the team and the dedication of the individual members of the team. We value the skill of

getting things done and a capability of delivering performance on time when it is needed together with the mindset of a scientist. The ability to transfer scientific concepts in robotics to a working prototype is vital for experimental evaluation such as the one in this competition.

11.1 Toward a more general solution

Despite our best effort to develop a general solution capable of autonomous landing on a moving car, couple sub-systems have been tailored specifically to the competition scenario. The computer vision system was designed to locate and track the landing pattern specified by the rules of the competition. We can speculate that similar pattern could be used in practice to mark a landing spot on a vehicle, which is designed to receive the UAV. In such case, the proposed vision system would be a viable option. However, in the case of an unmarked and possibly arbitrary vehicle, a different approach to localization and tracking would be required, e.g., based on nowadays popular artificial neural networks. Estimation and prediction of the car movement using a non-linear car-like model provide a framework suitable for tracking and landing on most common vehicles. A more precise model could be used to better estimate state of a specific vehicle. Our approach to bias the prediction of the future car movement based on the known parameters of the arena is optional and can be omitted in the case of a general area and car trajectory. Moreover, all field experiments, before the competition trials, were performed without particular bias towards the known trajectory of the car. See Figure 13 for an example of such experiment. The presented state machine is also designed around the competition scenario, however, the need to customize it is apparent. Besides those two cases, the presented approach can be applied to a general scenario of locating, tracking and landing on a vehicle in an outdoor environment.

12 Conclusion

We proposed, developed and experimentally evaluated an unmanned aerial system for autonomous landing on a fast-moving vehicle. The solution described in this paper is a multirotor helicopter platform equipped with sensors and a computer, capable of onboard image processing and state estimation of the ground vehicle and also predictive planning and automatic control. Images from an onboard camera are processed online to extract the position of the landing marker on top of the vehicle. States of the car-like dynamical model are estimated using the Unscented Kalman Filter, based on image processing and the known parameters of the car trajectory. The same model is then used to predict the future trajectory of the vehicle. The Model Predictive Control tracker creates an optimal feed-forward reference in third order dynamics based on the predicted trajectory. Non-linear SO(3) state feedback controls the UAV along the reference. A state machine controls the UAV from takeoff, through finding the target and tracking it, to finally turning off its propellers during the touchdown. The proposed system utilizes state-of-the-art techniques from control engineering in a unique combination to solve the difficult challenge, which only a handful of teams from all over the world were able to tackle. The proposed control pipeline relies on the novel MPC Tracker, which was proposed specifically for this challenge (Baca et al., 2018), to achieve the high accuracy of the autonomous landing.

The system was extensively tested in the course of more than one year of development. The experiments showed that the UAV is capable of autonomous tracking and landing on a car moving at a speed of 15 km/h. In the MBZIRC 2017 challenge, the system proved to be robust by successfully landing in both trials of the challenge 1 of the competition. During the Grand Challenge of the competition, it landed in 25 s, which is the shortest time among all teams during the entire competition, which may be considered as a relevant and objective benchmark of this task.

Acknowledgments

The outstanding results of this project could not have been achieved without the full cooperation of each member of our team, comprising people from the Czech Technical University in Prague, the University of

Pennsylvania and the University of Lincoln, UK (Figure 20). The work has been supported by CTU grant no. SGS17/187/OHK3/3T/13, the Grant Agency of the Czech Republic under grant no. 17-16900Y, the Office of Naval Research Global. Grant Numbers: N000140710829, N000141410510, by Army Research Laboratory. Grant Number: W911NF1720181, and by Khalifa University of Science, Technology and Research. Grant Number: MBZIRC 2017.

Figure 20: Team members after the award ceremony of the MBZIRC competition in Abu Dhabi, United Arab Emirates.

References

Araar, O., Aouf, N., and Vitanov, I. (2017). Vision based autonomous landing of multirotor uav on moving platform. *Journal of Intelligent & Robotic Systems*, 85(2):369–384.

Baca, T., Hert, D., Loianno, G., Saska, M., and Kumar, V. (2018). Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Baca, T., Loianno, G., and Saska, M. (2016). Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles. *International Conference on Methods and Models in Automation and Robotics (MMAR)*.

Benini, A., Rutherford, M. J., and Valavanis, K. P. (2016). Real-time, gpu-based pose estimation of a uav for autonomous takeoff and landing. In *International Conference on Robotics and Automation (ICRA)*, pages 3463–3470. IEEE.

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

Bi, Y. and Duan, H. (2013). Implementation of autonomous visual tracking and landing for a low-cost quadrotor. *International Journal for Light and Electron Optics*, 124(18):3296–3300.

Borowczyk, A., Nguyen, D.-T., Phu-Van Nguyen, A., Nguyen, D. Q., Saussié, D., and Le Ny, J. (2017). Autonomous landing of a quadcopter on a high-speed ground vehicle. *Journal of Guidance, Control, and Dynamics*, 40:2378–2385.

Fu, M., Zhang, K., Yi, Y., and Shi, C. (2016). Autonomous landing of a quadrotor on an ugv. In *International Conference on Mechatronics and Automation (ICMA)*, pages 988–993. IEEE.

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

Ghommam, J. and Saad, M. (2017). Autonomous landing of a quadrotor on a moving platform. *Transactions on Aerospace and Electronic Systems*, 53:1504–1519.

Guo, Z. and Hall, R. W. (1989). Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, 32(3):359–373.

Heriss, B., Hamel, T., Mahony, R., and Russotto, F. X. (2012). Landing a vtol unmanned aerial vehicle on a moving platform using optical flow. *Transactions on Robotics*, 28(1):77–89.

Hoang, T., Bayasgalan, E., Wang, Z., Tsechpenakis, G., and Panagou, D. (2017). Vision-based target tracking and autonomous landing of a quadrotor on a ground vehicle. In *American Control Conference (ACC)*, pages 5580–5585. IEEE.

Hui, C., Yousheng, C., Xiaokun, L., and Shing, W. W. (2013). Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision. In *Chinese Control Conference (CCC)*, pages 5895–5901. IEEE.

Jin, S., Zhang, J., Shen, L., and Li, T. (2016). On-board vision autonomous landing techniques for quadrotor: A survey. In *35th Chinese Control Conference (CCC)*, pages 10284–10289. IEEE.

Jung, W., Kim, Y., and Bang, H. (2016). Target state estimation for vision-based landing on a moving ground target. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 657–663. IEEE.

Jung, Y., Lee, D., and Bang, H. (2015). Close-range vision navigation and guidance for rotary uav autonomous landing. In *International Conference on Automation Science and Engineering (CASE)*, pages 342–347. IEEE.

Kim, J., Jung, Y., Lee, D., and Shim, D. H. (2014). Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1243–1252. IEEE.

Kong, W., Zhou, D., Zhang, D., and Zhang, J. (2014). Vision-based autonomous landing system for unmanned aerial vehicle: A survey. In *International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*, pages 1–8. IEEE.

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 *International Conference on Robotics and Automation (ICRA)*, pages 971–976. IEEE.

Lee, H., Jung, S., and Shim, D. H. (2016). Vision-based uav landing on the moving vehicle. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1–7. IEEE.

Lee, T., Leok, M., and McClamroch, N. H. (2013). Nonlinear robust tracking control of a quadrotor uav on $se(3)$. *Asian Journal of Control*, (2):391–408.

Lin, S., Garratt, M. A., and Lambert, A. J. (2017). Monocular vision-based real-time target recognition and tracking for autonomously landing an uav in a cluttered shipboard environment. *Autonomous Robots*, 41(4):881–901.

Masselli, A., Yang, S., Wenzel, K. E., and Zell, A. (2014). A cross-platform comparison of visual marker based approaches for autonomous flight of quadrocopters. *Journal of Intelligent & Robotic Systems*, 73(1-4):349–359.

Meier, L., Tanskanen, P., Heng, L., Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2012). Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1):21–39.

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

Mellinger, D., Shomin, M., Michael, N., and Kumar, V. (2013). Cooperative grasping and transport using multiple quadrotors. In *The 10th International Symposium Distributed Autonomous Robotic Systems*, pages 545–558. Springer Berlin Heidelberg.

Rossiter, J. A. (2003). *Model-based predictive control: a practical approach*. CRC press.

Ruffo, M., Di Castro, M., Molinari, L., Losito, R., Masi, A., Kovermann, J., and Rodrigues, L. (2014). New infrared time-of-flight measurement sensor for robotic platforms. In *18th International Workshop on ADC Modelling and Testing*. IMEKO.

Spurny, V., Baca, T., Saska, M., Penicka, R., Krajnik, T., Thomas, J., Dinesh, T., Loianno, G., and Kumar, V. (2018). Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles. *Journal of Field Robotics*.

Stepan, P., Krajnik, T., Petrlik, M., and Saska, M. (2018). Vision techniques for on-board detection, following, and mapping of moving targets. *Journal of Field Robotics*, pages 1–18.

Tan, C. K., Wang, J., Paw, Y. C., and Liao, F. (2016). Autonomous ship deck landing of a quadrotor using invariant ellipsoid method. *Transactions on Aerospace and Electronic Systems*, 52(2):891–903.

Tersus-GNSS (2017). PRECIS-BX305 GNSS RTK Board. Available: https://www.tersus-gnss.com (citied on 2017-07-17).

Wann, E. A. and van der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158. IEEE.

Weiss, S., Scaramuzza, D., and Siegwart, R. (2011). Monocular-slam-based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874.

Xu, L. and Luo, H. (2016). Towards autonomous tracking and landing on moving target. In *International Conference on Real-time Computing and Robotics (RCAR)*, pages 620–628. IEEE.

Yang, S., Scherer, S. A., and Zell, A. (2013). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, pages 1–17.

Yang, S., Ying, J., Lu, Y., and Li, Z. (2015). Precise quadrotor autonomous landing with srukf vision perception. In *International Conference on Robotics and Automation (ICRA)*, pages 2196–2201. IEEE.

View publication stats