European Conference on Mobile Robotics (ECMR), Paris, France, September 2017.

Fast Autonomous Landing on a Moving Target at MBZIRC

Marius Beul, Sebastian Houben, Matthias Nieuwenhuisen, and Sven Behnke

***Abstract*— The ability to identify, follow, approach, and intercept a non-stationary target is a desirable capability of autonomous micro aerial vehicles (MAV) and puts high demands on reliable target perception, fast trajectory planning, and stable control. We present a fully autonomous MAV that lands on a planar platform mounted on a ground vehicle, relying only on onboard sensing and computing.**

We evaluate our system in simulation as well as with real robot experiments. Its resilience was demonstrated at the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) where it worked under competition conditions. Our team NimbRo ranked third in the MBZIRC Challenge 1 and – in combination with two other tasks – won the MBZIRC Grand Challenge.

I. Introduction

Fast autonomous landing of micro aerial vehicles (MAV) on moving platforms is challenging. One of the three tasks during the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2017 consisted of landing a flying robot on an artificial pattern on top of a golf cart, moving with $15\,\mathrm{km\,h^{-1}}$ on a figure eight course (cf. Fig. 1). The arena for this task had a size of $90\,\mathrm{m} \times 60\,\mathrm{m}$. The entire competition was executed under challenging outdoor conditions with temperatures of up to $38\,^{\circ}\mathrm{C}$ and strong gusts of wind. Although the vehicle would slow down after a given amount of time, a team had to land the robot in the first minutes, autonomously, and without any damage in order to receive a high score for its run. In fact, the teams ranked highest were able to complete the task in less than $30\,\mathrm{s}$ after takeoff, including the time needed to search for the moving target.

The MBZIRC task may seem like a toy example with no immediate application, but landing on a moving platform is an important step towards operating MAVs in more complex dynamical environments in the future. Not only does the task require precise state estimation but also low-latency, accurate detection and prediction of the target and MAV movements.

In this paper, we present our integrated MAV system designed for landing on a moving platform, including

- tailored hardware design,
- robust perception and fast tracking of a landing pattern,
- state estimation for the moving target, and
- fast, analytical trajectory generation for interception.

We evaluate our approach in both simulated and robot experiments, and report results from the MBZIRC competition.

This work has been supported by a grant of the Mohamed Bin Zayed International Robotics Challenge (MBZIRC), and grants BE 2556/7-2 and BE 2556/8-2 of the German Research Foundation (DFG).

The authors are with the Autonomous Intelligent Systems Group, Computer Science VI, University of Bonn, Germany mbeul@ais.uni-bonn.de

Fig. 1. Final approach only seconds before a successful landing on a moving vehicle at the MBZIRC Grand Challenge.

II. Related Work

For reasons of brevity, in this overview, we cover lines of research performing the landing without cooperation between the robot and the ground vehicle. Lee et al. [1] demonstrated the viability of the task by using visual servoing in order to maneuver above the moving pattern, and relying on a motion-capture system for external state estimation. A similar system was demonstrated by Serra et al. [2] who also use visual servoing but do not rely on a vision-based distance estimation to the target. In comparison to our system, both approaches are evaluated with a slow or even static target. Borowczyk et al. [3] use a system of two cameras and filter the detections together with an IMU and GPS receiver mounted at the target. They report landing velocities of up to $50\,\mathrm{km\,h^{-1}}$. Landing indoors on an inclined plane was achieved by Vlantis et al. [4] who designed an adapted model-predictive controller to optimize the local trajectory in real time. However, due to the computational demand, optimization was done on an external base station and, thus, required a stable network connection.

Fast real-time trajectory planning and control is an active area of research. Ezair et al. [5] compare polynomial trajectory generation algorithms regarding the order, state constraints, and constraints on initial and final conditions. In their works [6] and [7], Mueller et al. present a trajectory generator similar to our work. It is also capable of approaching the full target state (position, velocity, and acceleration) and is real-time capable. Analogue to our approach, they use jerk (respectively the rotational velocity ω) as system input, but the convex optimization problem is solved numerically. Generated trajectories are not time-optimal.

From other MBZIRC participants, we want to cite the early work by the team of the Korea Advanced Institute

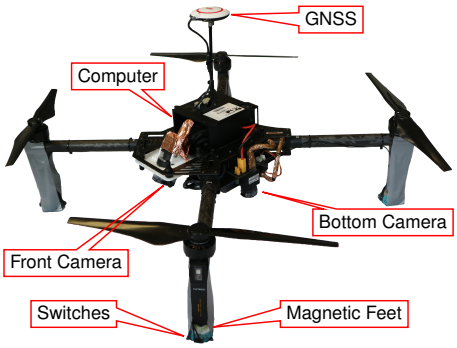

Fig. 2. Design of our MAV equipped with two cameras, magnetic feet, and a lightweight but fast onboard PC (shielding removed for better view).

of Science and Technology [8] where landing on a larger platform at a velocity of $0.75\,\mathrm{m\,s^{-1}}$ was demonstrated, but the visual detection was still simplified by a marker reflecting infrared light.

In contrast to prior works, the setup of the MBZIRC challenge required most of the processing (i.e., state estimation, control, sensor processing, mission- and trajectory planning) to take place onboard the system. With the exception of a ground station for GPS where a reference positioning signal is provided over WiFi if and when a connection is available, no external resources are used in our system.

III. System Setup

Our MAV, shown in Fig. 2, is based on the DJI Matrice 100 platform. It is equipped with a small but fast Gigabyte GB-BSi7T-6500 onboard PC with an Intel® Core™ i7-6500U CPU running at $2.5/3.1\,\mathrm{GHz}$ and $16\,\mathrm{GB}$ of RAM. The landing pattern is perceived by two Point Grey BFLY-U3-23S6M-C greyscale cameras with 2.3 MP. The first camera—equipped with a wide-angle lens with an apex angle of $195° \times 195°$—points downwards. To facilitate the detection of a far-away pattern and to keep it in the field of view (FoV) during descent on a glide path, the second camera—with an apex angle of $69° \times 85°$— points $30°$ into forward direction. Both cameras capture 40 frames per second, resulting in 80 frames per second in total.

We replaced the landing feet of the MAV with strong magnets with a total rated force of $860\,\mathrm{N}$ to keep it in place on the moving target with ferromagnetic surface after landing. A successful landing is detected by eight micro switches attached to the landing feet. The switches are individually connected to an Arduino Nano v3.0 that serves as bridge to our onboard computer.

For allocentric localization and state estimation, we employ the filter onboard the DJI flight control that incorporates GNSS (global navigation satellite system) and IMU data. To avoid electromagnetic interference between components—in particular USB 3.0 and GPS—the core of our MAV is wrapped in electromagnetic shielding material. This increases the system stability significantly. Fig. 3 gives an overview of the information flow in our system. We use the robot operating system (ROS) as middleware on the MAV and the ground control station. We communicate over WiFi

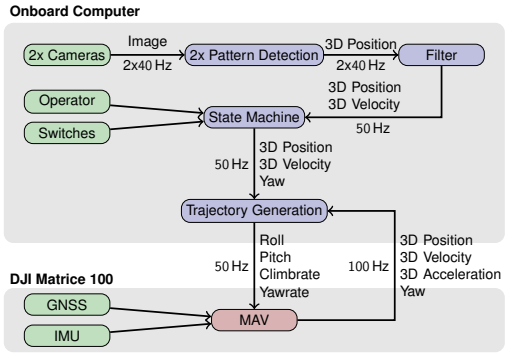

Fig. 3. Structure of our method. Green boxes represent external inputs like sensors, blue boxes represent software modules, and the red box indicates the MAV hardware. All software components use ROS as middleware. Position, Velocity, Acceleration and Yaw are allocentric.

with a robust UDP protocol, developed for connections with low-bandwidth and high-latency [9].

IV. Landing Pattern Perception

When detecting the pattern with a camera, one must consider two main objectives:

- the detection process itself should be low-latency and yield accurate results and
- the detection range should be as large as possible.

A. Landing Pattern Detection

We developed a multi-stage detection pipeline (cf. Fig. 4): The camera image is transformed to a bird's eye representation (cf. Fig. 4 (c))[1], a segmentation step detects line-like structures within the image (cf. Fig. 4 (d)) that are processed via a circular Hough transform in order to generate a number of hypotheses and their respective confidence.

Let r be the radius of the landing pattern in meters. In order to maximize the detection range the MAV's flying

[1]Please note that a) the camera setup is not aligned to the ground plane and b) the camera may have an arbitrary orientation during rapid manoeuvres, thus, a prior image transform is necessary.

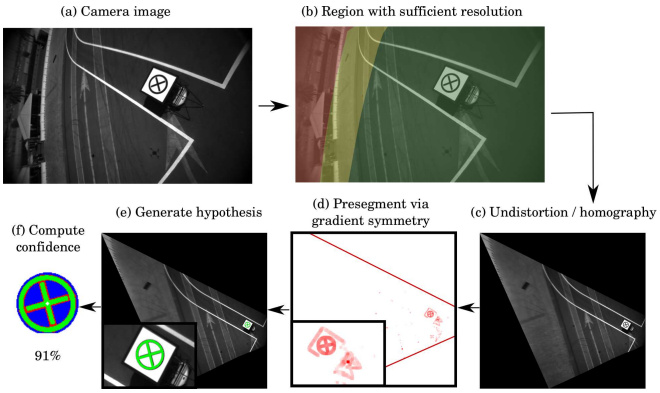

Fig. 4. Landing pattern detection from the front camera during the competition: (a) original camera image, (b) image regions with sufficient resolution for pattern detection (green), insufficient resolution (yellow), and regions above the ground plane (red), (c) bird's eye representation from a region with (mostly) sufficient resolution, (d) results from symmetry segmentation shown dilated for better visibility, (e) initial hypothesis in green, (f) confidence computation via pattern-detection overlay: green and blue denote correct pixels, red incorrect ones.

height h, obtained by relative barometric measurements, and its attitude, represented by the IMU gravitational vector $\hat{r}_z$ within camera coordinates, is taken into account. Then the rotation matrix

$$\begin{aligned} \hat{R} &:= (\hat{r}_x, \hat{r}_y, \hat{r}_z) \\ \text{with } \hat{r}_x &:= \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T \times \hat{r}_z, \\ \hat{r}_y &:= \hat{r}_z \times \hat{r}_x \end{aligned}$$

describes the rotation from the camera frame into a camera frame where the image plane is aligned with the ground plane, i.e., the matrix

$$M := K_g \hat{R} K_c^{-1}$$

with accordingly chosen camera matrices K_g, K_c describes a pixel coordinate transform into a bird's eye representation via homogeneous coordinates. Finally, taking the lens distortion into account, we arrive at

$$(u, v) \mapsto M \begin{pmatrix} f(u, v) \\ 1 \end{pmatrix} \tag{1}$$

with an invertible radial-tangential lens undistortion function f operating on the image coordinates (u, v). K_c is given by the camera intrinsics, K_g is set to

$$K_g := diag\left(\tfrac{2r \cdot \rho}{h}, \tfrac{2r \cdot \rho}{h}, 1\right)$$

where ρ is the desired resolution of the pattern (for ρ we choose 60 pixels or lower, depending on the maximum possible resolution of the pattern in the original camera image). It remains to compute those image regions that yield at least the resolution ρ_{min} required for detection (chosen as 20 pixels) when transformed by M. To this end, a point grid with a fixed stride of k pixels in the original image is mapped via M and the distance between neighboring points is computed. If this distance is below $\frac{2rk}{\rho_{min}}$, the resolution of the the corresponding image patch is too low for detection. The maximum rectangular region in the camera image containing all grid points with sufficient resolution after mapping (i.e., the maximum rectangle enclosing only green points in Fig. 4 (b)) is computed via a heuristic approach and the resulting region of the camera image is subsequently transformed. In order to efficiently execute this transform, please note that f is static such that a pixel-wise lookup table can be precomputed. The second part of the mapping in Equation (1) is linear-projective and can be computed very efficiently, in particular on rectangular regions when not the entire image has to be traversed.

For segmentation, fast symmetry detection is performed. Local pairs of pixels with approximately converse gradients are identified and their center is segmented if their distance matches the expected line width of the pattern ($10\,\text{cm}$). The procedure is detailed in [10]. A circular Hough transform—the approximate diameter of the circle in image dimensions is known—provides a fixed number of hypothesis that are subsequently confirmed if two lines with a central perpendicular intersection are detected within. Finally, a confidence measure is computed by thresholding the potential region of the intensity image with the expected quantile of dark versus white pixels and computing the ratio of the correct pixels with an artificial overlay (cf. Fig. 4 (f)).

In order to meet the required latency, after a sufficiently confident detection only a rectangular image region around the previous position is considered in the following iterations, reducing the algorithm to steps (c) – (f) from Fig. 4. In order to follow the pattern as long as possible, the requirement on minimum image resolution of the pattern is ignored during this tracking phase.

B. Landing Pattern Tracking

To predict the movement of the landing target in a world frame, we use a simpler version of our onboard state estimation filter presented in previous work [11]. We estimate position and velocity of the target in an allocentric frame with a constant velocity assumption in the prediction step. We consider detections from both cameras as independent observations and the filter merges them to a coherent world view. The pose of the MAV and the projection of the landing target perceptions into the allocentric frame are subject to the same localization error. Thus, the allocentric estimate of the target is consistent with the egocentric control of the MAV. Since we do not make any assumptions about the path of the landing target, e.g., moving in an eight pattern, our method is applicable to arbitrary pattern motions and independent from exact absolute MAV localization.

V. Landing Control

Since the total time to land during the challenge is crucial, we employ our time-optimal trajectory generation method described in [12] with the extensions from [13].

A. MAV Model

We assume the MAV to follow rigid body dynamics and simplify it as a point mass with jerk j as system input. Following Newton's second law, the system is a triple integrator in each dimension (x, y, z) with position p, velocity v, acceleration a, and jerk j:

$$\dot{p} = v, \qquad \dot{v} = a, \qquad \dot{a} = j. \tag{2}$$

Thus, the three-dimensional allocentric state of the MAV $\boldsymbol{x}$ can be expressed by

$$\boldsymbol{x} = \begin{pmatrix} p_x & p_y & p_z \\ v_x & v_y & v_z \\ a_x & a_y & a_z \end{pmatrix}. \tag{3}$$

We assume jerk j to be the direct control input to the linear system. Without loss of generality, we define the z-axis to be collinear to the gravity vector. Furthermore, we define the origin to be the middle of the arena and the xy-plane equal with the ground plane. We do not model

- moment of inertia,
- drag,
- yaw dynamics, and
- coupling of the axes that occurs in non-hover conditions,

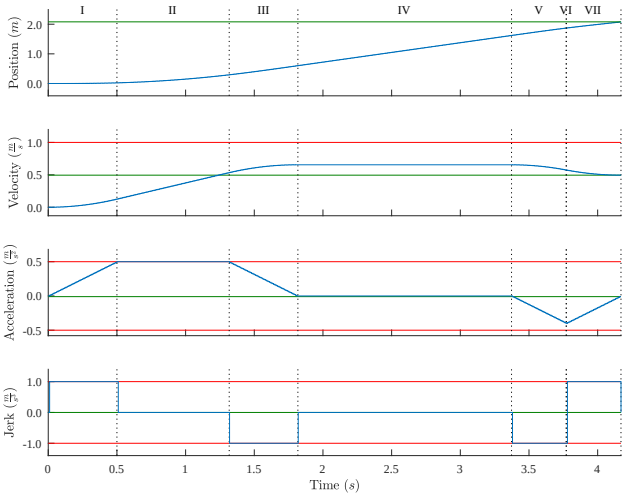

Fig. 5. This time-optimal trajectory was generated with our method. Starting from state ($p_x = 0\,\mathrm{m}, v_x = 0\,\mathrm{m\,s^{-1}}, a_x = 0\,\mathrm{m\,s^{-2}}$), it brings the MAV to the target state ($p_x = 2.08\,\mathrm{m}, v_x = 0.5\,\mathrm{m\,s^{-1}}, a_x = 0\,\mathrm{m\,s^{-2}}$). The trajectory satisfies constraints $v_{max} = 1\,\mathrm{m\,s^{-1}}$, $a_{max} = 0.5\,\mathrm{m\,s^{-2}}$, and $j_{max} = 1\,\mathrm{m\,s^{-3}}$. The calculated switching times are $t_1 = 0.5\,\mathrm{s}$, $t_2 = 0.82\,\mathrm{s}$, $t_3 = 0.5\,\mathrm{s}$, $t_4 = 1.55\,\mathrm{s}$, $t_5 = 0.4\,\mathrm{s}$, $t_6 = 0.0\,\mathrm{s}$, and $t_7 = 0.4\,\mathrm{s}$. The trajectory corresponds to the x-axis in Fig. 7. It is suboptimal (maximum velocity not reached), since this axis is slowed down to synchronize with the slower y-axis.

but rely on fast replanning to account for model uncertainties and unmodeled effects instead. Since our model is parameterless, our approach generalizes to all multicopters and no cumbersome parameter tuning is required.

B. Time-optimal Control

Based on the simple triple integrator model, our method analytically generates third-order time-optimal trajectories that satisfy input ($j_{min} \leq j \leq j_{max}$) and state constraints ($a_{min} \leq a \leq a_{max}$, $v_{min} \leq v \leq v_{max}$). The planned trajectory consists of up to seven phases of constant jerk ($j = j_{max} \vee j = 0 \vee j = j_{min}$), resulting in a third-order bang-zero-bang trajectory.

Fig. 5 shows a 1-dimensional trajectory. We synchronize all axes to arrive at the target at the same time. By doing so, the MAV flies on a straight glide path towards the landing position.

Furthermore, we use the ability of our trajectory generation method to calculate an optimal interception point, based on the current velocity of the target. We predict the target motion and do not fly to the current target location, but to the position, the MAV can intercept the target assuming a constant velocity motion and respecting the MAVs constraints. Although the assumption of constant velocity may not be justified in the curved parts of the figure eight since the acceleration is relatively large ($|a_{target}| \approx 1\,\mathrm{m\,s^{-2}}$), we found the error to be compensable by fast replanning.

C. MPC Application

We use the above mentioned trajectory generation method as an MPC (Model Predictive Controller), running in a closed loop with 50 Hz. Our hardware does not support directly sending jerk commands. We therefore assume pitch and roll to directly relate to $\theta = \mathrm{atan2}(a_x, g)$ and $\phi = \mathrm{atan2}(a_y, g)$.

TABLE I
PARAMETERS USED AT MBZIRC

Param	Axis	Value	Param	Axis	Value
v_{max}	x,y	$8.33\,\mathrm{m\,s^{-1}}$	v_{max}	z	$1.00\,\mathrm{m\,s^{-1}}$
a_{max}	x,y	$4.73\,\mathrm{m\,s^{-2}}$	a_{max}	z	$10.0\,\mathrm{m\,s^{-2}}$
j_{max}	x,y	$5.00\,\mathrm{m\,s^{-3}}$	j_{max}	z	$50.0\,\mathrm{m\,s^{-3}}$
Δt_{setp}	x,y	0.15 s	Δt_{setp}	z	0.50 s

So, instead, we send smooth pitch θ and roll ϕ commands for horizontal movement and smooth climb rates v_z in z direction. Due to the linearization, the acceleration constraint relates to an attitude constraint with $\theta_{max} = \mathrm{atan2}(a_{max}, g)$.

Our method plans the whole trajectory to the target instead of relying on a small constant lookahead commonly found in MPCs. Since the whole future trajectory is known at every replanning time step, one can choose from which future time Δt_{setp} to send the commands to the system. If the value is small (e.g., $\Delta t_{setp} = 0\,\mathrm{s}$), the system will react slowly. This is because small lookahead values will render the setpoint changes from the current state to be small and thus the underlying control loops slow. If the lookahead value is too large, the system can become unstable or perform suboptimal. Also communication delay has negative impact on the system and is compensated by choosing an appropriate lookahead. We experimentally determined that the values found in Tab. I offer good performance.

In Sec. V-A we report that we model the MAV in three orthogonal axes with the z-axis collinear to the gravity vector. The rotation about the z-axis α however is not defined. We define the rotation to be the allocentric angle of the current position to the target position $\alpha = \mathrm{atan2}(p_{y_{wayp}} - p_y, p_{x_{wayp}} - p_x)$. By doing so, we project the per-axis velocity constraint to lie in the axis of the dominant motion. Otherwise, the global horizontal velocity constraint would result in being $v_{max} = \sqrt{v_{max_x}^2 + v_{max_y}^2}$ and thus violating the maximum allowed velocity at the MBZIRC of $30\,\mathrm{km\,h^{-1}}$.

D. Yaw Control

Although an arbitrary number of axes can be synchronized, we do not consider the yaw-axis to be synchronized with the x, y and z-axis. For simplicity, we use proportional control for the yaw-axis Ψ. The yaw rate setpoint $\dot{\Psi}_{setp} = K_p \cdot (\Psi_{setp} - \Psi)$ is sent to the MAV. A couple of different yaw behaviors can be selected by the state machine described in Sec. VI, depending on the current situational requirement. The MAV can point towards

- a defined allocentric yaw angle,
- the current target,
- the optimal interception point,
- forward direction (current MAV horizontal velocity vector), and
- direction of target motion (current target horizontal velocity vector).

VI. Mission Control State Machine

The behavior of the MAV is controlled by a state machine that serves as a generator for position, velocity, and yaw

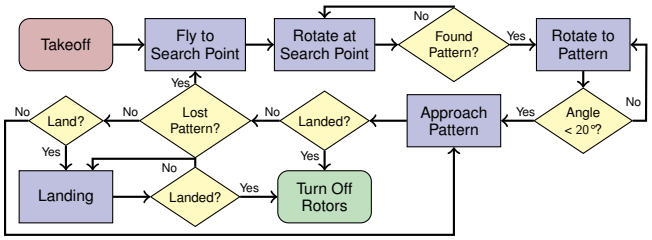

Fig. 6. The flowchart of our state machine. Besides the basic behavioral control, it features strategies to recover from failed landing approaches.

setpoints for the lower layers. Fig. 6 shows a flowchart of our state machine. After takeoff, the MAV flies with maximum velocity to a search point 8 m above the field, already exploring the arena through rotations. When the landing pattern is first detected, we rotate the front camera approximately into pattern direction before starting the descent.

We restrict the descent rate based on the distance to the target to ensure good perceivability of the pattern in the cameras. The state machine transfers the yaw authority to the low-level trajectory generation during pattern following. Depending on the distance to the target, the MAV yaws towards the target or into flight direction, to ensure that the pattern is visible in either the bottom camera or—in stable forward flight without fast rotations—the front camera.

The final landing decision is based on relative orientation, distance and relative height to the pattern, and visibility in the cameras. If the landing decision has been taken, the MAV descents until ground contact is detected by the switches at its feet. This is necessary as the pattern cannot be reliably tracked during the landing due to its proximity to the MAV.

To prevent unstable behavior while manoeuvring in the vicinity of the fast moving landing pattern or in corner cases for the perception, the descent is completely aborted and the landing procedure restarts from the initial search point above the field when the pattern is lost during following. For safety, we also detect premature landings in the pattern following state and turn off the rotors. Since the state machine is the only subsystem which the operator interfaces with during flight, we built a distinct GUI for situational awareness of the operator.

VII. Evaluation

We evaluate our system in simulation as well as with a real MAV. Videos of our evaluation can be found on our website[2].

A. Evaluation in Simulation

Landing on a target moving at relatively high speed imposes a risk for the MAV and persons who move the target. Thus, we tested all individual software components and the integrated software system in simulation.

Fig. 7 shows our Matlab simulation for the optimal interception of a moving landing pattern. We model the MAV as pictured in Sec. V-A. It can be seen that the MAV lunges to eliminate any velocity difference to the target when arriving at the interception point.

[2]http://www.ais.uni-bonn.de/videos/ECMR_2017_Beul

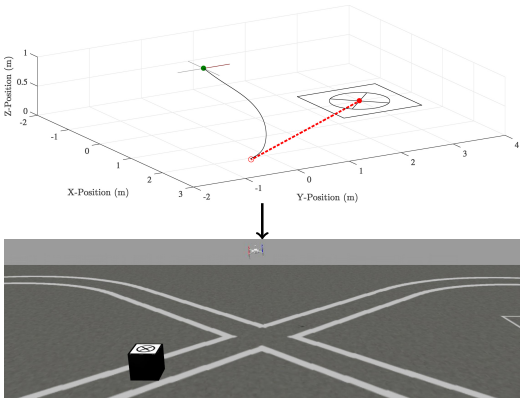

Fig. 7. Landing simulation in Matlab and Gazebo. We first simulate the interception of the target with a simplified linear model. The MAV is marked with a green dot. The target is marked with an solid red dot. The predicted target trajectory is marked with a dashed line, ending in the interception point (red ring). Subsequently, we modeled the MBZIRC arena including the moving target in Gazebo. The MAV can be simulated with HIL, employing a complex motion model and challenging environmental conditions.

To achieve a high level of realism, we also modeled the MBZIRC arena and the moving target in the RotorS simulator [14]. In addition to the physics-based MAV simulation of RotorS, we implemented a hardware in the loop (HIL) bridge, employing the DJI simulator. Here, the flight control is connected to the DJI Assistant 2 software via USB. Instead of controlling the motors of the real MAV, the flight control firmware sends control commands to the simulator, where MAV dynamics and sensors like IMU and GPS are simulated and sent back to the flight control. Thus, for our ROS middleware it is undeterminable if the real or the simulated MAV is used.

B. Real-robot Evaluation at MBZIRC

Our system was used to compete in the MBZIRC. We were able to place third in Challenge 1 and – in combination with two other tasks – first in the Grand Challenge of the total 24 resp. 14 competitors. During the first run in Challenge 1, we first experienced a hardware problem with the USB 3.0 connection of the front camera and were forced to restart. After fixing this issue, we were able to successfully land in 34 s—measured from spinning up the rotors to landing on the pattern. In total, the time from the start of the challenge to landing—including fixing the MAV—was 112 s, resulting in the third place in the final ranking. In order to fix the connection, we attached more shielding for the second trial. Unfortunately, this shielding negatively affected the compass of the MAV so that it went into failsafe mode directly after the start. We canceled the second trial since we could not fix this issue fast enough to improve our time from the first challenge run.

In the first trial of the Grand Challenge, we were able to land in 42 s. Fig. 8 shows the trajectory and detections of this trial. After reaching the center of the field, the MAV searched for another 11.9 s for the target because the cart was in a disadvantageous position. In the second trial, we could not improve our landing time and canceled this trial after 42 s.

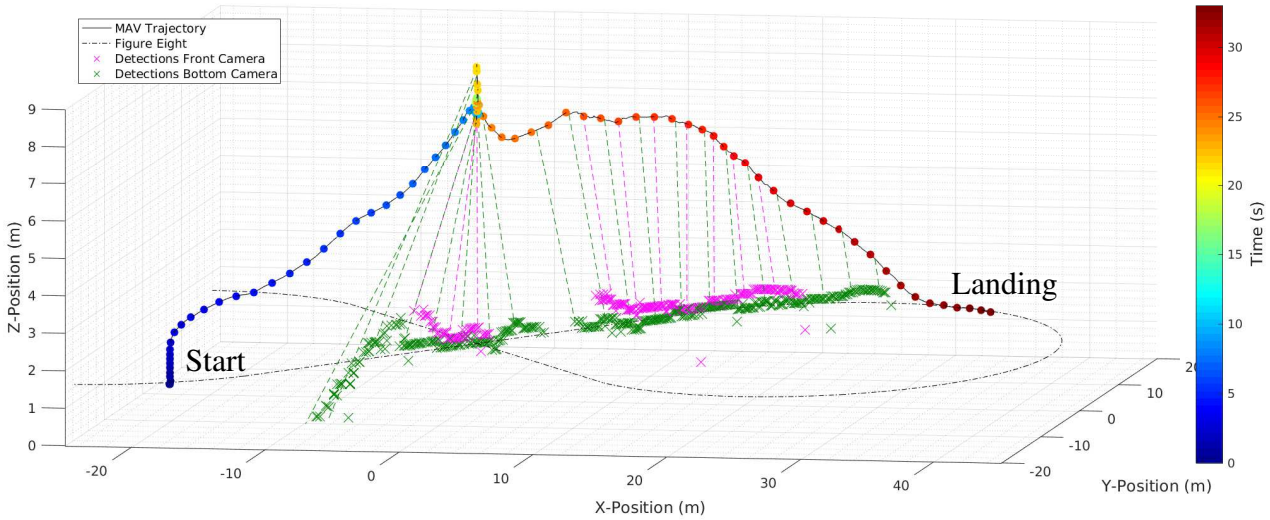

Fig. 8. Landing in MBZIRC Grand Challenge. First, the MAV starts in the middle of the left circle and flies straight up to a height of 1.5 m to not collide with near objects. Next, it flies to the center of the field with a height of 8 m to search for the landing target. The total ascent takes 8.5 s. After 20.4 s, the landing target is first detected in the bottom camera. Immediately, the MAV begins to descent while tracking the target in both cameras. The descent only takes 11.6 s, resulting in a total completion time of 32 s. Due to the fast motion of the target, the MAV cannot descent fast enough to reach the target on the straight segment and has to land in the curved segment of the figure eight. The challenge completion time from start signal to landing is 42 s. Colored markers are placed every 250 ms on the trajectory. Every 20th of all 585 detections is indicated with the corresponding viewpoint on the trajectory.

VIII. Conclusion

We have provided detailed insight into our robust MAV setup for quickly landing on a fast moving target. The viability of this approach has been demonstrated in an outside scenario with minimum preparation time during the MBZIRC where the MAV consistently performed the landing as one of the fastest among all competitors.

In particular the adaptive and fast trajectory replanning combined with a high-frequency pattern detection turned out to reliably match direction and velocity with the moving target. Furthermore, the use of two cameras in combination with an adaptive yawing strategy enabled us to track the target pattern under fast manoeuvres and in close proximity.

We believe that our contribution, but in general all experience from the MBZIRC landing challenge, will facilitate new ideas of how to operate flying robots in dynamic—and hence real-world—environments.

References

[1] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a vtol uav on a moving platform using image-based visual servoing," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[2] 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. on Robotics*, vol. 32, no. 6, pp. 1524 – 1535, 12 2016.

[3] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussié, and J. L. Ny, "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *ArXiv e-prints*, vol. 1611.07329, 11 2016.

[4] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Quadrotor landing on an inclined platform of a moving ground vehicle," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[5] B. Ezair, T. Tassa, and Z. Shiller, "Planning high order trajectories with general initial and final conditions and asymmetric bounds," *The Int. J. of Robotics Research*, vol. 33, no. 6, pp. 898–916, 2014.

[6] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *Proc. of the European Control Conference (ECC)*, 2011.

[7] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasability verification," in *Proc of IEEE/RSJ Int Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[8] H. Lee, S. Jung, and D. H. Shim, "Vision-based uav landing on the moving vehicle," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2016.

[9] M. Schwarz, T. Rodehutskors, D. Droeschel, M. Beul, M. Schreiber, N. Araslanov, I. Ivanov, C. Lenz, J. Razlaw, S. Schüller, D. Schwarz, A. Topalidou-Kyniazopoulou, and S. Behnke, "NimbRo Rescue: Solving disaster-response tasks through mobile manipulation robot Momaro," *Journal of Field Robotics*, vol. 34, no. 2, pp. 400–425, 2017.

[10] S. Houben, M. Neuhausen, M. Michael, R. Kesten, F. Mickler, and F. Schuller, "Park marking-based vehicle self-localization with a fisheye topview system," *J. of Real-Time Image Processing*, pp. 1–16, 2015.

[11] M. Beul, N. Krombach, Y. Zhong, D. Droeschel, M. Nieuwenhuisen, and S. Behnke, "A high-performance mav for autonomous navigation in complex 3d environments," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 6 2015.

[12] M. Beul and S. Behnke, "Analytical time-optimal trajectory generation and control for multirotors," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2016.

[13] ——, "Fast full state trajectory generation for multirotors," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2017.

[14] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS–a modular gazebo MAV simulator framework," in *Robot Operating System (ROS): The complete reference*, A. Koubaa, Ed., 2016, vol. 1, ch. 23, pp. 595–625.