Autonomous Landing On A Moving Car With Unmanned Aerial Vehicle

Tomas Baca[1], Petr Stepan[1] and Martin Saska[1]

***Abstract*— This paper presents an implementation of a system that is autonomously able to find, follow and land on a car moving at $15\ \mathrm{km/h}$. Our solution consists of two parts, the image processing for fast onboard detection of landing platform and the Model Predictive Control tracker for trajectory planning and control. This approach is fully autonomous using only the onboard computer and onboard sensors with differential GPS. Besides the description of the solution, we also present experimental results obtained at MBZIRC 2017 international competition.**

MULTIMEDIA MATERIAL

A video attachment to this work is available at [11].

I. INTRODUCTION

Multirotor helicopters, in robotic literature called Unmanned Aerial Vehicles (UAVs) or Micro Aerial Vehicles (MAVs), due to their possible very small size, have become widely popular within the scientific community for their well studied dynamic properties and high applicability. Ability to hover in one place has advocated their use as a sensor carrying platform and for testing various approaches of fully autonomous flying, even without any human intervention during the mission in future. Three tasks need to be solved in such a deployment of autonomous MAV systems: take off, trajectory following possibly with an environment interaction, and precise landing. Robust and safe autonomous landing seems to be the most challenging part of the overall MAV mission mainly if a dynamic target of unknown position and windy outdoor conditions are considered. In this paper, a system designed to solve this task is introduced together with breathtaking results of repeated fully autonomous landing on a $15\ \mathrm{km/h}$ moving platform in an outdoor arena with the wind reaching $10 - 20\ \mathrm{km/h}$ and achieving a precision in tens of centimeters.

A. MBZIRC 2017 competition

The research achievements presented in this paper were motivated and results obtained within the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2017 competition organized by the Khalifa University of Science in Abu Dhabi. The MBZIRC competition brought a significant impact to the robotic community, and mainly to the field of MAVs, due to the ambitiously selected robotic challenges on the edge of current state-of-the-art chosen by a board of top scientists in the field. One of these challenges motivated by current needs of industry was autonomous landing on a moving vehicle discussed in this paper.

[1]Authors are with the Faculty of Electrical Engineering, Czech Technical University in Prague, Technicka 2, Prague 6, Email: tomas.baca@fel.cvut.cz.

The MBZIRC challenges were designed to provide a spectacular performance and to attract a broad general audience. The most importantly, the MBZIRC competition could be considered as a relevant and objective benchmark of the tasks, which are currently solved by the robotic community since the successful teams had to achieve the given goal after only a few minutes of preparation in one trial. No repeated tests, which is the standard practice in most of the laboratory experiments, were allowed and moreover the system robustness was exhibited in current environment (light and windy) conditions (teams could not influence the start of their run). Out of 140 registered teams from almost all best robotic groups worldwide, 25 top teams were selected after several preliminary rounds to compete in the final competition in Abu Dhabi in March 2017[1]. The solution, described in this paper, presents the best reliability and robustness (only two teams landed precisely in both final trials) and the fastest performance from all competition trials (the fastest time of landing in the entire competition was achieved by the proposed system during the grand challenge, where together three MAVs were deployed simultaneously). The only competitive solution was presented by team Fly Eagle from Beijing Institute of Technology. They also landed in both trials in a similar time, but their system is not published yet to be able to compare both systems and to highlight differences. Nevertheless, both solutions can be considered as a valuable contribution to the robotic field since according to our knowledge no other system does exist to be able to solve this very demanding and complex challenge in these outdoor conditions (which was also the reason, why this task was selected by respected robotic leaders for the competition).

Before the competition, few solutions of autonomous landing with visual feedback were described in literature [5], [6], but most of these systems are capable of landing only on a static or slowly moving pattern [8], [9], [10] and only in laboratory conditions, without the presence of wind and with stable light conditions.

To sum up the contribution of this paper, the proposed method enables to detect a landing pattern, to estimate its relative position and velocity, to predict motion of both systems (the MAV and the landing platform) and based on these states estimation it designs sequence of optimal control inputs taking into account external disturbances such as wind and imprecisely identified MAV model in MPC fashion using only onboard computational resources. This

[1]Results of our team from this qualification process can be found at http://mrs.felk.cvut.cz/projects/mbzirc

approach provides high robustness and precision in the landing task, which is a crucial element for fully autonomous missions (such as periodical surveillance, reconnaissance, object carrying, and monitoring) in which MAVs are especially appealing.

B. Contributions

We present a solution to the challenge 1 of MBZIRC 2017 competition. The MAV can follow a car moving at $15\ \mathrm{km/h}$ autonomously. While following, it lands on the car roof and attaches itself using magnetic legs. The landing on a moving car is robust on very challenging outdoor condition with wind speed up to $10\ \mathrm{m/s}$.

II. The problem definition

The task consists of an autonomous landing of an MAV on a moving car. The car is equipped with a graphical black pattern on a white metal board. The landing pattern consists of a single circle with a cross drawn in its center. The autonomous landing is conducted with a multirotor aircraft equipped with cameras, differential GPS receiver, and other sensors. Although the GPS is not a necessity, long-term and robust autonomous flight proves to be simpler than using, e.g., only visual odometry for localization. The speed profile of the car is known, starting at $15\ \mathrm{km/h}$ and slowing down, as well as its trajectory, except its initial conditions – position and direction. The track, in which the car will drive, can be measured before experiments to improve the search or tracking.

We assume our MAV is equipped with a down-facing camera, differential GPS and laser rangefinder for measuring its altitude above the ground.

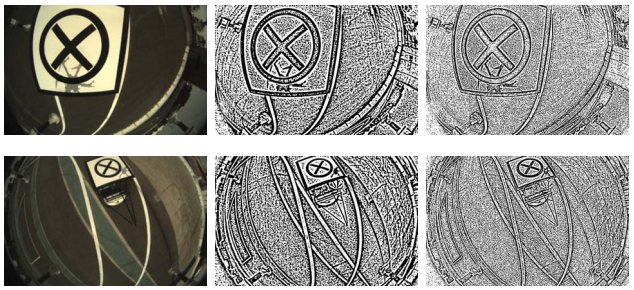
III. Computer vision

The goal of pattern detection is to detect the landing pattern robustly. The image processing was computed on Intel NUC embedded PC with Intel Core i7 5557U processor. A single mvBlueFOX-MLC200w color camera was used. This camera has a global shutter and therefore is suitable for aerial robotic purposes. The camera can provide 93 images per second with resolution 752×480. A miniature SuperFisheye lens Sunex DSL215, together with with 1/3" camera sensor, created an image with horizontal FOV of $185°$. Our detection algorithm can detect the landing pattern in one image using a single thread in $15\ \mathrm{ms}$. The final detection rate is $50\ \mathrm{Hz}$.

A. Algorithm overview

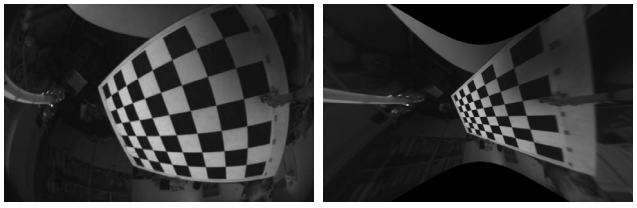
Landing pattern detection is based on standard computer vision approaches using OpenCV tool. The landing pattern (see Fig. 1a) contains circle and cross. The detection algorithm is based on circle detection combined with cross detection inside the circle. The detection has to be robust to various weather conditions, changes of light intensity, and direct sunshine with shadows cast by the aircraft.

The first step of pattern detection is a method of adaptive thresholding. The result of the thresholding with box size 5 pixels and 11 pixels can be seen in Fig. 1b resp. 1c. The adaptive threshold is robust to light intensity and is used to segment the border of the circle in the image. The size of the block for adaptive threshold depends on expected pattern size, which depends on drone altitude.



(a) (b) (c)

Fig. 1: a) original image from the camera, b) adaptive threshold with box size 11 pixels, c) adaptive threshold with box size 5 pixels.



(a) (b)

Fig. 2: a) original image from the camera, b) undistorted image.

A variant of the detection algorithm is selected based on the current altitude of the MAV:

- altitude more than $5\ \mathrm{m}$, the pattern is small, the size of circle in the image ranges from 10 to 20 pixels
- altitude between $1.5\ \mathrm{m}$ and $6\ \mathrm{m}$, the whole circle can be detected
- altitude less than $1.5\ \mathrm{m}$, only a part of circle is detected, detection is based only on cross detection

The circles and lines are detected in the segmented image from the adaptive threshold. The object detection is done in image coordinates, in contrast with the classification, which is applied on undistorted images. Undistorting images before further computations is crucial for further processing.

B. Fish-eye lens

The identification of the lens was performed using OpenCV 3.2 and its fish-eye model. The original image from the drone can be seen in Fig. 2a, the corrected image in Fig. 2b.

Due to the slow implementation of the fish-eye model in OpenCV, a custom version of the undistort method was implemented. This new undistort function is even faster than the conventional camera model. The original fish-eye undistort function computes transformation from image coordinates (x, y) to undistorted coordinates by following steps:

1) compute world point

$$(w_x, w_y) = \big((x - c_x)/f_x, (y - c_y)/f_y\big), \quad (1)$$

where (c_x, c_y) is the image center and (f_x, f_y) focus in axis x and y.

2) compute distance from center as

$$\theta_0 = \sqrt{w_x^2 + w_y^2}, \quad (2)$$

3) iteratively compute undistortion coefficient (10 times)

$$\theta_{i+1} = \frac{\theta_0}{1 + k_0 \cdot \theta_i^2 + k_1 \cdot \theta_i^4 + k_2 \cdot \theta_i^6 + k_3 \cdot \theta_i^8}, \quad (3)$$

where k_0, k_1, k_2, k_3 are distortion coefficients detected by fish-eye calibration.

4) using θ_{10}, compute $scale = \frac{\tan(\theta_{10})}{\theta_0}$ and undistorted coordinates $(ud_x, ud_y) = (w_x * scale, w_y * scale)$

This approach is very slow, because iterative computation of θ_{10} needs 80 float multiplication and 17 float divisions, so together it needs 97 float operations.

Because the distortion coefficients are known in advance, we can prepare results of scale for all values of θ_0 with sufficient precision. For our image resolution 752×480, we are using array $scale_k$ with 1000 of values in $\langle \theta_0, \theta_{max} \rangle$, where $\theta_0 = 0$, $\theta_{max} = \frac{\Pi}{2}$. Additionally, we can omit square root function, because we can prepare scale values for the square distance from image origin. Finally, the computation of undistorted coordinates for image coordinates (x, y) takes following steps:

1) compute world point

$$(w_x, w_y) = ((x - c_x)/f_x, (y - c_y)/f_y), \quad (4)$$

where (c_x, c_y) is the image center and (f_x, f_y) focus in axis x and y,

2) compute square distance from center as

$$\theta_0 = w_x^2 + w_y^2. \quad (5)$$

3) using precomputed array $scale_k$,

$$scale = scale_k \left(\frac{\theta_0}{\theta_{max}} \cdot 1000 \right) \quad (6)$$

and compute undistorted coordinates

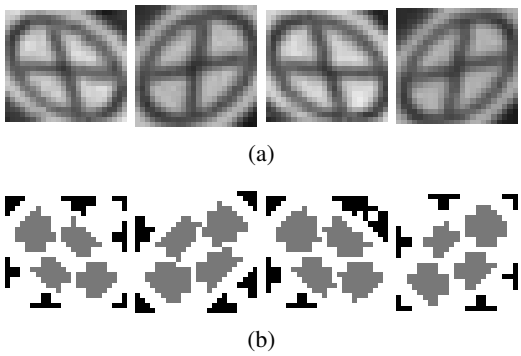
$$(ud_x, ud_y) = (w_x * scale, w_y * scale) \quad (7)$$

This approach is using only 7 float multiplications and is approx. 15 times faster than the conventional one.

C. Robust detection

The robust detection of the landing pattern is based on circle detection on images from adaptive threshold procedure. To eliminate false positive detections, the circle must contain a cross. The detection of the cross relies on three algorithms, depending on circle size.

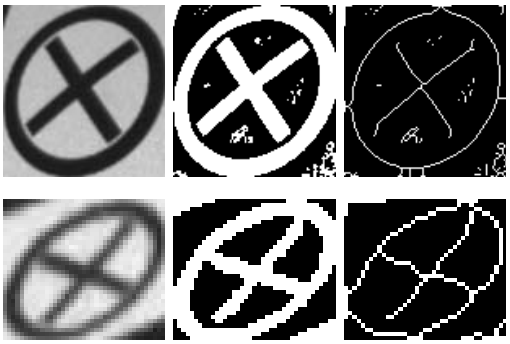
If one of the axis of the ellipse (circle) is shorter than 30 pixels, then a line, making a part of the cross, cannot be detected reliably. In such case, the cross is detected by searching for four areas with similar size. Mathematical morphology operation *closing* is applied, and the number of closed areas is computed. The cross is detected if there are four closed areas, having similar size. Figure 3a shows original images, wheres Fig. 3b shows detected areas depicted by gray color.



(a)

(b)

Fig. 3: a) Original images from camera, b) result of operation morphological closing.



(a) (b) (c)

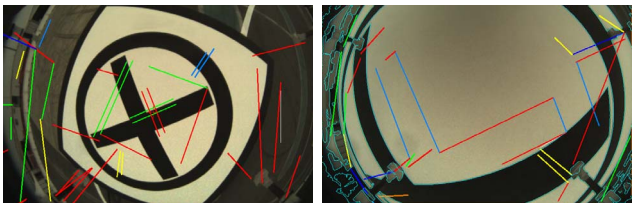
Fig. 4: a) original image from the camera, b) adaptive threshold with box size 11 pixels, c) result of Guo Hall thinning.

Cross of a size between 30 and 150 pixels can be detected by Guo Hall thinning [2]. This algorithm cannot be used for larger circles because of its computational demands. The Guo Hall thinning enables very robustly detect correct lines inside the circle. The cross inside the circle is detected if a crossing point of two the biggest lines is near the center of the ellipse. Figure 4 shows the original images from the camera in comparison to images after applying adaptive thresholding and the results of Guo Hall thinning algorithm.

For circles larger than 150 pixels, the cross is detected by finding two pairs of parallel lines (see fig. 5a red and green lines) forming a border of the cross. At least two pairs of parallel lines with correct size and thickness are required, to positively detect the cross and its center. This method provides detection of the landing pattern if the whole circle cannot be seen. Figure 5b depicts the pattern reconstructed only from two visible lines of the cross.

D. Global pattern position

The last step is computing the position of the landing pattern in global world coordinate system. Suppose that we



(a) (b)

Fig. 5: Line detection for a) cross inside of the circle and b) when only a part of the landing pattern is detected. Only boundaries are shown, as would be in undistorted images.

know the camera position $cam = (cam_x, cam_y, cam_z)$ and its orientation. Using position of the center of the landing pattern $(ud_x, ud_y, 1)$ in undistorted camera coordinate system we can compute a vector $d = (d_x, d_y, d_z)$ from camera to center to the landing pattern in world coordinate system. Knowing the altitude alt of the camera from ground and height of the landing vehicle we can compute the position of the landing pattern lan as:

$$lan = cam + d * \frac{alt - high}{d_z}. \tag{8}$$

A precise estimate of the MAV altitude is required. Experiments showed that the altitude error could be up to $1\ \mathrm{m}$ when relying on a laser rangefinder or other available sensors. Therefore, for purposes of the pattern detection, we compute the distance to the pattern from the apparent size of the ellipse.

IV. Camera calibration

The successful landing depends on precise localization of the landing pattern in world coordinates. Synchronization of MAV position to the time of image acquisition is important to correctly compute the global coordinates of the car. Calibration of the camera coordinate system with MAV coordinate system has the same importance.

Following steps describe the parameters taken into account when calculating the global position of the car:

- position translation of the camera origin relative to the MAV body
- angular shift of a camera mount to the MAV body
- time shift of camera data to the time of known position and orientation of the MAV

The position shift of the camera coordinate system to the MAV coordinate system was measured on a bench during the construction of the MAV. The experiments show that the best way to estimate the angular shift is to use real-world data. The time shift of camera data has to be estimated by experiment too because it depends on many hidden parameters of ROS and OS system. It varies with camera type, camera interface and real-time features of the underlying operating system.

We found it difficult to set all parameters at once, due to the time shift and angular shift being strongly connected. We designed two flight scenarios to automate the estimation.

Both scenarios rely on differential GPS up-to $5\ \mathrm{cm}$ localization precision. The MAV takes off from the center of the landing pattern, and therefore the target's position in world coordinate system can be automatically detected. The first scenario was designed to detect the angular shift of the camera, and therefore the MAV flies as stable along a predefined trajectory around the landing pattern. The second scenario contains aggressive maneuvers with a high angular rate of the MAV.

Finding the angular shift is split into two parts. First, the yaw angle α is estimated as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}. \tag{9}$$

Then roll and pith angles are found separately using x and y global coordinates respectively, all obtained from designed flight. For example, to estimate the angular correction using angle $\gamma = \tan\frac{d_x}{f_x}$ we apply

$$x' = \frac{(x - d_x) \cdot f_x^2}{f_x^2 + d_x \cdot x}. \tag{10}$$

The search for the angular shift can be defined as an optimization task with variables α, d_x, d_y. The task was solved by finding a minimal value for all combinations of values α from interval $\langle -5°, 5° \rangle$ with step $0.5°$, values d_x, d_y from $\langle -50, 50 \rangle$ with step 2 using focal length $f_x = f_y = 261\ \mathrm{pixels}$. The task was solved offline using recorded flight data.

V. Guidance law

The guidance law we present is a modular pipeline consisting of five components. Following paragraph gives a shortlist of those components, which are subsequently describe in following sessions V-.1 to V-.4.

The first component is the **Cross detector** (presented in Chapter III), which provides measurements of car position in the world frame coordinate system. The measurements are processed by **Car state estimator**, which outputs an estimate of the car states in the means of nonholonomic, car-like model. The estimate is then conveyed to the **Car state predictor**, whose output is a future trajectory of the car. The trajectory is then taken as reference by the **MPC tracker**, which minimizes a quadratic error of MAV future states over the predicted future, to fly above the car. As a result, the MPC tracker outputs desired states (position, velocity, and acceleration) to the **State feedback controller**. The state feedback controller, being the last part of the pipeline, produces commands for the Pixhawk flight controller.

1) Car state estimator: The car state estimator was designed using the Unscented Kalman Filter (UKF) [4]. UKF allows estimation of system states using a nonlinear transfer function. A discrete, nonholonomic, car-like model with state box constraints was used to further allow precise prediction

of vehicle future movement:

$$\begin{aligned}
\mathbf{x}_{[n+1]} &= \mathbf{x}_{[n]} + \dot{\mathbf{x}}_{[n]}\Delta t, \\
\dot{\mathbf{x}}_{[n+1]} &= \begin{pmatrix} \cos\phi_{[n+1]} \\ \sin\phi_{[n+1]} \end{pmatrix} v_{[n+1]} \\
\phi_{[n+1]} &= \phi_{[n]} + \dot{\phi}_{[n]}\Delta t, \\
\dot{\phi}_{[n+1]} &= K_{[n]} v, \\
v_{[n+1]} &= v_{[n]} + a_{[n]}\Delta t, \\
K_{[n+1]} &= K_{[n]} + \dot{K_{[n]}}\Delta t,
\end{aligned} \tag{11}$$

where $\mathbf{x} = (x_x, x_y)^T$ is position of the car, ϕ is its heading, K is curvature of its turn, v is car scalar velocity, a is car scalar acceleration and Δt is sampling time difference.

State estimation served two purposes within our pipeline. First, it acted as a filter for incoming measurements from the *cross detector*, with measurements variance being adjusted with respect to the altitude. Second, it allowed estimating unmeasured states, later important for predicting car future movement. The resulting estimate is outputted at 100 Hz.

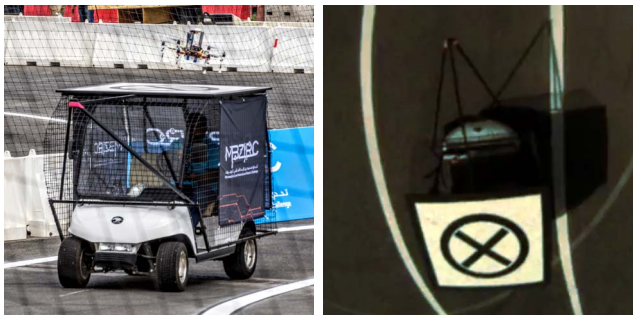
2) Car state predictor: Using the information from the *car state estimator*, we predicted its future trajectory using the same dynamic model. In the case of a general car trajectory, tracking with such prediction worked reliably up to 20 km/h. For purposes of the competition, we decided to use a prior knowledge about the competition arena. The curvature of the predicted trajectory was biased using a known map of the arena and the track on which the car was driving. Based on the observations during rehearsals, the day before the competition, this technique was further extended to bias predicted points by projecting them directly in the driving track. The predicted trajectory was outputted at 30 Hz.

3) MPC tracker: In our pipeline, a trajectory tracker is responsible for generating a set of desired states of the MAV (position, velocity, and acceleration) to follow a set trajectory (generated by the *car state estimator*). It uses a decoupled, 3-rd order translational dynamics to simulate a virtual MAV at 100 Hz. The virtual MAV is then controlled by Model Predictive Control with 8 s prediction horizon, also at 100 Hz. States of the virtual MAV are sampled and handed out to the *state feedback controlled* as a reference. Thanks to the MPC, the tracker provides necessary feed-forward action to follow the known future path. The particular MPC control approach is based on previous work presented in [3], further extended to support state constraints in velocity and acceleration.

4) State feedback controller: The final part of the pipeline is SO3 state feedback controller presented in [7]. SO3 controller provides 100 Hz feedback on the desired position, velocity, and acceleration of the MAV. Controller output is desired tilt angles, yaw rate and total thrust which are sent to Pixhawk embedded controller as a reference.

A. Approach and landing strategy

The whole mechanism of approaching and landing was realized as deterministic finite state automata. After the sequence was initiated, the MAV took off and moved to the center of the workspace, where it waited for the car to appear in the field of view of the camera. Such strategy minimizes the mean time of the search given the constraints of our MAV and the unknown (random) initial condition of car position and direction. After the car was spotted and the covariance of its state estimate surpassed a given threshold, the MAV started to follow the car while it maintained current altitude. In this case, the approach trajectory, produced by *Car state predictor*, was designed as minimum-snap with orthogonality constraint – the MAV first moved orthogonally to a point, where it would meet the car. Such approach mitigated unwanted direct movement towards the car, which would tilt the MAV in car direction and possibly lost it from the line of sight. After the MAV had been aligned with the car, a first descending stage was initiated. When descended to 4 m, the MAV was waiting for a precise alignment, after which a rapid landing stage started. During the rapid landing stage, a laser rangefinder provided a trigger for motor shut-off. In the case of lost alignment or lost tracking, the respective states were stepped back, or the whole automata was reset.



(a) (b)

Fig. 6: a) Our MAV landing on the car. b) The landing platform from MAV camera point of view.

VI. Experimental results

The challenge took place on a flat, rectangular area with dimensions 90×60 m. The area contained a marked track in a shape of figure 8 (see Fig. 8). The car was driving within the marked track, starting on a random place and heading in a random direction. A decreasing speed profile was defined, starting at 15 km/h and later slowing down to 5 km/h. The car was equipped with a 1.5×1.5 m landing platform, bearing a landing pattern (see Fig. 6).

Before the round, the competing team would place the MAV at a starting location and wait for a mark to initiate the flight. The team whose MAV would land the fastest wins. Penalization was issued if parts of the drone would fell off during the landing or for any human intervention after it took off.

A. Experimental platform

The MAV was assembled mostly from off-the-shelf components, except a few 3D-printed parts. Common DJI F550 hexacopter frame (see Fig. 7) was equipped with Pixhawk

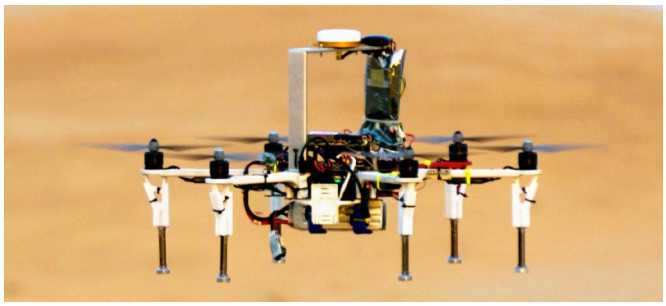


Fig. 7: MAV used in the experiments was based on DJI F550 hexacopter frame, Pixhawk stabilization board, and Intel NUC computer.

autopilot system, flashed with stock PX4 software. Our computer vision and guidance software were executed on Intel NUC with Core i7 processor. The MAV utilized a single down-facing MatrixVision Bluefox camera with fisheye lens. Drone's landing gear was equipped with neodymium permanent magnets to fix it after the landing. All custom software was build using ROS (Robot Operating System), running upon Ubuntu operating system.

B. Trial results

Our platform was tested in 4 trials over two days of the competition. First two were part of a separate challenge solely focused on landing on a moving vehicle, wheres the other two were part of a *Grand challenge*, where other robotic tasks were performed simultaneously in the same arena (Cooperative collecting of objects by multiple MAVs and a ground robot task). First two trials were completed successfully with the flight time of 1 min, 44 sec, and 0 min, 1 min, 28 sec. The third trial was also successful while having the best score of all teams – 0 min, 25 sec of flight time. The fourth trial was the only unsuccessful one due to the MAV misalignment in the final stage of the landing, which was falsely classified. Fig. 8 plots all three successful trials in a top-down view, as recorded by the MAV during the flight. Videos, capturing the trials, can be found at [11].

VII. Conclusion

We showed that platform presented in this paper is capable of landing on a car driving at speed 15 km/h, using only onboard computational resources. The platform was tested in MBZIRC 2017 competition, where it performed with the fastest landing time among all competing teams.

VIII. Acknowledgements

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 and by Khalifa University.

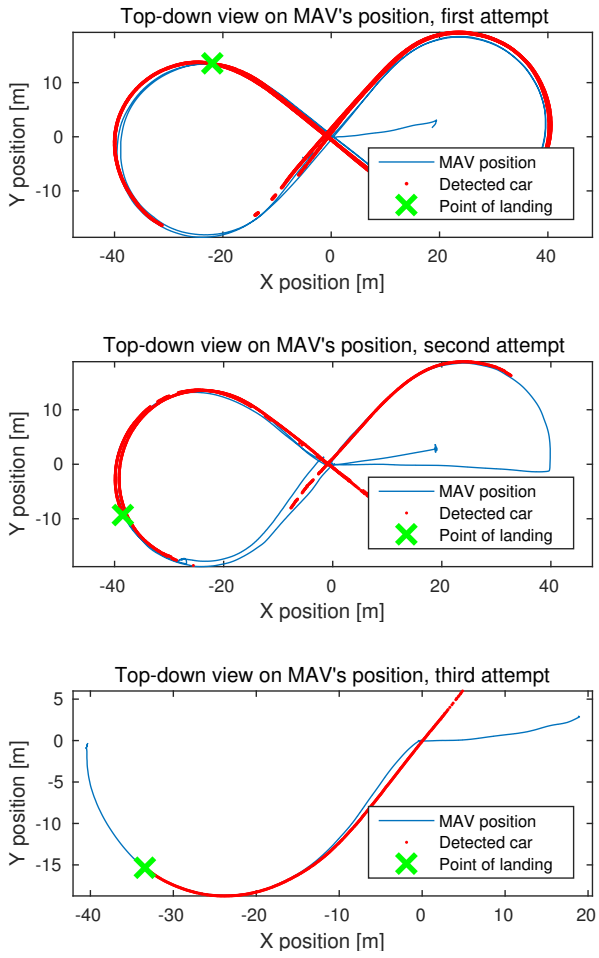


Fig. 8: Top-down view of all three successful trials: 1:44 min flight (top), 1:28 min flight (middle) and 0:25 min flight (bottom).

References

[1] J. Kannala and S. S. Brandt, A generic camera model and calibration method for conventional, wide-angle and fish-eye lenses, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 8, August 2006.

[2] Z. Guo and R. W. Hall, Parallel Thinning with Two-Subiteration Algorithms, Communications of ACM, vol. 32(3), March 1989, pp. 359-373.

[3] T. Baca, G. Loianno and M. Saska, Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles, in IEEE MMAR 2016.

[4] E. Wan and R. Van Der Merwe, The unscented Kalman filter for nonlinear estimation, IEEE AS-SPCC 2000.

[5] M. Saska, T. Krajnik and L. Preucil, Cooperative UAV-UGV autonomous indoor surveillance, IEEE SSD 2012.

[6] S. Lange, N. Sunderhauf, and P. Protzel, A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments, IEEE ICAR 2009.

[7] T. Lee, M. Leok, and N. McClamroch, Nonlinear robust tracking control of a quadrotor UAV on SE(3), Asian Journal of Control 15.2 (2013): 391-408.

[8] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, Visually guided landing of an unmanned aerial vehicle, IEEE Transactions on Robotics and Automation, vol. 19, no. 3, pp. 371380, Jun. 2003.

[9] M. Fu, K. Zhang, Y. Yi, Ch. Shi, Autonomous landing of a quadrotor on an UGV, IEEE ICMA 2016.

[10] J. Kim, Y. Jung, D. Lee, D. H. Shim, Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera, IEEE ICUAS 2014.

[11] Supplementary multimedia `http://mrs.felk.cvut.cz/ecmr2017landing`