Vision-based algorithm for autonomous aerial landing

A. E.S. Morando, M. Ferreira Santos, Pedro Castillo Garcia, Alessandro Corrêa Victorino

Vision-based algorithm for autonomous aerial landing

A. E. S. Morando[1], M. Ferreira Santos[1], P. Castillo[1] and A. Correa-Victorino[1].

***Abstract*— The landing phase is a critical stage in autonomous aerial landing, especially when the aerial vehicle lands in a moving platform, as ground vehicles. In this paper, a solution combining the information from the onboard camera of the drone with an observer is used to estimate and predict the future position of the landing platform. This landing estimation is used in the control algorithm, based on quaternions, for generating and tracking a landing trajectory. The proposed solution is then validated in real-time experiments (two scenarios) to demonstrate the well performance and efficiency of the closed-loop system. Main graphs from these experiments are reported in this paper. Moreover, as this work aims to set the base for future developments, existing limitations from this work are discussed in the last section.**

I. Introduction

Aerial vehicles are interesting platforms that have been developed for several applications. Nevertheless, an open issue is their autonomous landing (since it is a high-incidence stage whose accuracy and success can compromise the entire mission). This challenge is even harder if the aerial vehicle lands on a mobile platform evolving in unstructured environments. An emerging solution for this problem is the use of cameras with vision algorithms (for a detailed review see [1]). A crucial step in aerial vehicle vision-based landing is target detection in the camera frame.

A common solution is to denote the landing platform with artificial markers accurately designed to be recognized more easily, called fiducial markers. Indeed, these last are a cost-effective solution since the pose of the camera can be accurately computed based on the marker's shape alone with low CPU usage. For this reason fiducial markers are in vogue: promising works are [2], [3], [4], and [5]. Originally developed in 2014, squared ArUco markers [6] are among the most used planar markers. With an open-source library part of OpenCV available, it is possible to detect the marker and compute the pose of its four corners knowing its size and the camera calibration parameters. Thanks to their accuracy, different works have achieved promising results in detecting still [7] [8] and moving platforms [9] [10] [11]. However, one of the main difficulties when resorting to fiducial markers is when detection is missed due to occlusion, shadows, or because the marker is not entirely in the field of view of the camera. Hence, the challenge is to predict the future position of the landing platform to both preventively keep the target in the camera field and have a prediction in case of no recognition. Different studies have been carried out over the years.

A possible solution is the well known Kalman filter (KF) and its variations. In [8] authors resorted to a KF for dealing with missed recognition of the still target. In [9], an Extended Kalman Filter is used to predict the future position of the landing platform in motion. More recently an Alternating Predictive Observer [12] has been proposed always regarding the vision-based landing on a ground robot moving fast. It is worth pointing out that alternative solutions for pose estimation and trajectory prediction are proposed in the literature, notably deep learning methods have emerged in recent years. On the one hand, PoseCNN [13], Object-posenet [14], and ROFT [15] are just some examples of Neural Networks aiming to detect and estimate the pose of moving objects. On the other hand, Long Short-Term Memory networks are widely used for predicting trajectories [16] [17], as these last can be used without making any assumptions about the model of the moving agent and the measurement noise. Actually, also the vision-based landing mission can be solved applying this paradigm, notably reinforcement learning [18] [19]. Still, the main drawbacks of these methods are the computational power (which for embedded systems is limited) and the training phase (indeed just generating a custom dataset is time-consuming especially when using 3D bounding boxes).

In this paper, an autonomous vision-based control algorithm for landing on a platform whose motion is proposed. The cooperative target is in this case identified by an ArUco-based marker. To deal with missed recognitions, it has been chosen to resort to a KF as in [8] to predict the *(x,y)*-position of the landing platform, which is used as desired value for the aerial vehicle. Thanks to these two cost-effective solution, the proposed scheme provides accurate estimates and predictions ensuring a safe landing with a limited onboard equipment, few parameters to be tuned, and a limited computational load which makes it suitable for real-time applications. This paper is organized in five sections. In section II the key points of the implemented solution are described. In section III the vision algorithm for estimating the pose of the moving platform is introduced. The estimated pose is employed in the landing trajectory tracking using a control action presented in Section IV. In Section V the setup and the main results obtained in experiments are illustrated. Finally, in Section VI, conclusions are drawn with a focus on the limitations of the presented work and possible future developments.

[1]Université de technologie de Compiègne, CNRS, Heudiasyc UMR 7253, CS 60 319 - 60 203 Compiègne, France. alessandra.elisa.sindi.morando@edu.unige.it, (marcone.ferreira-santos, castillo, acorreav)@hds.utc.fr

II. Problem Statement

As it has been already stressed, the main objective of this work was to guide an Unmanned Aerial Vehicle (UAV) towards a Unmanned Ground Vehicle (UGV) playing the role of landing platform using the information got from the onboard vertical camera.

The following assumptions are stated: 1) the UAV is equipped with a RGB camera pointing down, 2) the aerial robot is flying with a defined altitude, 3) the ground vehicle either is steady or it moves slowly with a constant speed, 4) due to the drone microcontroller limitation, the image processing pose is estimated for experimental validation on a ground station, 5) at the end of the landing mission, the drone keeps a constant height over and close the ground robot (therefore, ground effect of propellers can be ignored).

The two main sub-issues identified here are to estimate and predict the future position of the target on the ground vehicle, and to compute the control action for trajectory tracking.

III. Target detection and pose estimation

The first challenge to tackle is to compute the pose of the target, $\breve{\vec{x}}_t$, in the global reference frame starting from the images of the vertical RGB camera. As shown in Figure 1, the required computations are executed on the ground station which communicates with the UAV through a TCP client-server structure and through VRPN with the motion capture system (MoCap) (that will be also used as ground truth). The camera frame is processed following several steps and for each of them a dedicated ROS node has been implemented.

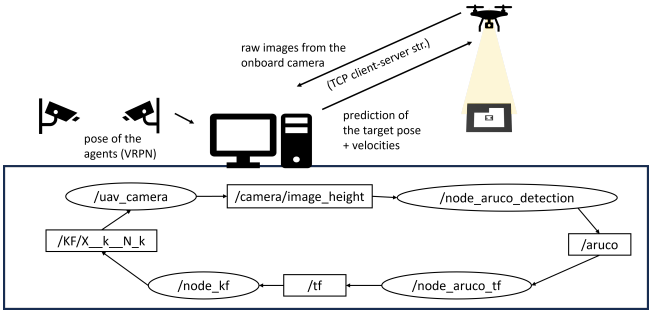

Fig. 1: Implemented solution with a focus on the communication protocols and the image processing steps.

The first ROS node *uav_camera* receives the vision information and the global position of the UAV measured by the OptiTrack motion capture system via VRPN streaming. In short, the node combines this information in a custom message which is then published on the */camera/image_height* topic.

The listening node *node_aruco_detection* catches this last message and computes the relative pose of the target in the camera frame, which is expressed by the translational vector $\vec{T}$ and the rotational vector $\vec{R}$, using ArUco markers. Note that in this work, instead of using a classical ArUco marker, a custom marker has been defined. Indeed, during some experiments intended to test the detection and accuracy of the pose estimation, it has been seen that the marker was not found when the altitude of the UAV was less than 60 cm, which is still too far to ensure a safe landing. This problem has been settled by using different markers of different sizes. The intuition is that bigger markers are detectable at higher heights, while the smaller ones come to help as the drone comes closer to the landing platform. Different markers can be put either side by side or embedded them, from which the name eArUco [20]. For this work, the second arrangement has been chosen. Moreover, it has not been used the marker proposed in [20] since at high altitudes the image was too blurred and it was not possible to detect the marker. Therefore, in the end a custom marker has been defined embedding ArUco markers from a 4×4 dictionary instead of a 7×7 one. The hint is that if on one hand by using more bits it is possible to encode more markers, on the other hand a dictionary with fewer squares is easier to detect. For the choice of the embedded markers composing the e-Aruco, it must be taken into consideration that: 1) the outer marker should be white in the center since ArUco markers require a white border to be segmented, 2) the inner marker should have a percentage of white higher than black squares since at higher distances it should result in a white dot. The proposed marker is the one in Figure 2. The outer marker has id 190 and length 28.5 cm, while the inner one has id 979 and size 5 cm.

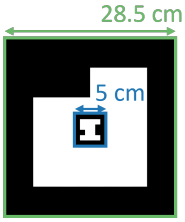

Fig. 2: Proposed eArUco marker.

Figure 3 shows a comparison of the detection of the inner and the outer markers, in blue and red, respectively, at the different heights of the UAV. In this case, the marker was not detected at heights lower than 80cm since it was not entirely contained in the frame which was not the case when using the classical ArUco marker.

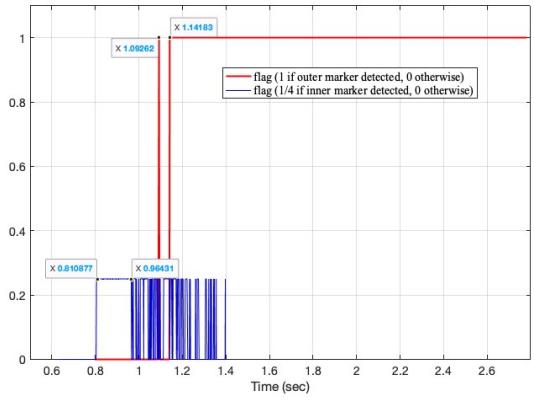

Fig. 3: Detection of the inner and outer marker composing the eArUco.

The relative pose of the target is estimated using either the inner or the outer markers. In the case of both markers are detected, if the altitude of the drone is greater than 1.15m the outer marker is chosen for the pose estimation. In the other case, the inner marker is preferred. The idea is that the larger is the marker in the camera image, the more the estimated relative position is reliable. Still, at smaller altitudes the outer marker is not entirely contained in the field of view of the camera and the inner marker becomes bigger in the image. The vectors $\vec{T}$ and $\vec{R}$ obtained using the OpenCV functions library are then published on the *aruco* topic, see Figure 1. Subsequently, a dedicated node called *node_aruco_tf* broadcasts the transformation from the reference frame corresponding to the target, *estimated_target*, and the camera frame which is defined by those two vectors.

To handle all these frames with the Fl-Air framework, the *tf* tool has been used. On one hand, Fl-Air is an open-source framework developed by the Heudiasyc laboratory in C++ with the purpose of developing applications for UAVs. On the other hand, the *tf* tool [21] organizes the frames in a tree structure and is possible to listen to a transformation from one frame to another if these two are connected in the tree. In case of detection, there exists a path between the frame *estimated_target* and *flair*.

Notice that, the target could be not always detected and the measurement could be also noisy, therefore, a KF was introduced for improving estimation. Indeed, the estimated pose with the ArUco marker in most of the cases presents some variations that can be modeled as additive white noises with zero mean. The use of the KF helps also to smooth estimated references to nearly constant values preventing undesired behaviors. To use the KF algorithm, we propose an evolution model relying on the simplifying assumption that the target is moving slowly with a constant speed. Considering this assumption, the target motion can be then described by the following linear model (see Figure 4)

$$\begin{aligned} x_{k+1} &= x_k + v_{x,k}\,\Delta t \quad & v_{x,k+1} &= v_{x,k} \\ y_{k+1} &= y_k + v_{y,k}\,\Delta t \quad & v_{y,k+1} &= v_{y,k} \\ z_{k+1} &= z_k + v_{z,k}\,\Delta t \quad & v_{z,k+1} &= v_{z,k} \\ \psi_{k+1} &= \psi_k + \omega_k\,\Delta t \quad & \omega_{k+1} &= \omega_k \end{aligned}$$

where x_k, y_k, z_k are the estimated position coordinates of the target expressed in discret time, v_x, v_y, v_z define their linear velocities, ψ and ω denote the angular position and rate of the heading, and Δt is the sampling time.

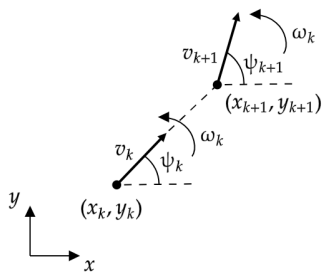

Fig. 4: Evolution model describing the target motion.

From Figure 1 the ROS node *node_kf* tries to look up for the transformation between the frames *flair* and *estimated_target*. In case no exception is thrown, the measure vector is defined as $\vec{y} = \begin{bmatrix} T_x & T_y & T_z & \psi \end{bmatrix}^T$ where T_i, for $i = x, y, z$, is the translational vector in each axis, and ψ is computed using the quaternion q representation.

The state at the current instant k, $\hat{\vec{x}}_{k|k}$, is updated using the measurement $\vec{y}$ and afterwards the state at the next instant $k+1$ is predicted, $\hat{\vec{x}}_{k|k+1}$. Then, in order to predict the future position of the target at $N > 1$ future steps, the prediction is integrated $N-1$ times using the evolution model, i.e., multiplying it by A^{N-1}, obtaining $\hat{\vec{x}}_{k|k+N}$. $\hat{\vec{x}}_{k|k+N}$ is published using an odometry message to arrive on the topic *KF/X_k_N_k* at which the node *uav_camera* subscribed, see Figure 1.

IV. Control action

Up to now, all the ROS nodes running on the ground station to estimate the target pose have been explained. At the end of the chain of ROS nodes, the prediction of the future pose and velocities of the target at N next steps are sent to the UAV via TCP. Once the drone has received the data, this information is used to compute the control action. In particular, the dynamic model and controller proposed in [22] have been used. The aerial configuration used is a quadcopter vehicle with state $\vec{x}_{av} = \begin{bmatrix} x_{av} & y_{av} & z_{av} & \phi_{av} & \theta_{av} & \psi_{av} \end{bmatrix}^T$ defining its position x_{av}, y_{av}, z_{av} in the inertial frame and orientation $\phi_{av}, \theta_{av}, \psi_{av}$. Notice that in [22] the system is proposed to be fully controlled virtually with 6 control inputs defined as $\vec{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$ and $\vec{\tau} = \begin{bmatrix} \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T$. The controller is based on quaternion formulation which is an extension of complex numbers where the imaginary part, $\overline{q}$, is composed of three components that can be collected in a three-dimensional vector. The quaternion is therefore defined as $\mathbf{q} = q_0 + \overline{q} \in \mathbb{H}$, $\overline{q} \in \mathbb{R}^3$, $q_0 \in \mathbb{R}$. The controller is defined in two parts. The first one is an inner attitude controller, $\vec{\tau}$, defined to track the desired trajectory $\vec{x}_d = \begin{bmatrix} x_d & y_d & z_d \end{bmatrix}^T$ The second part is the high translational controller for the trajectory tracking given by $\vec{u} := -K_{\vec{x}}\vec{e}_{\vec{x}} - K_{\dot{\vec{x}}}\dot{\vec{e}}_{\vec{x}}$ where

$$\vec{e} := \begin{bmatrix} x_{av} - x_d & y_{av} - y_d & z_{av} - z_d \end{bmatrix}^T$$

is the error position, $K_{\vec{x}}$, $K_{\dot{\vec{x}}}$ are gains matrices. The attitude controller is computed using the desired quaternion $\boldsymbol{q}_d$ that is calculated with the position controller $\vec{u}$ with the form

$$\boldsymbol{q}'_d = \left(\vec{b} \cdot \vec{u} + \|\vec{u}\|\right) + \vec{b} \times \vec{u} \qquad \boldsymbol{q}_d = \frac{\boldsymbol{q}'_d}{\|\boldsymbol{q}'_d\|}$$

where $\vec{b}$ is a unitary vector denoting the axis in which the thrust acts in the body frame. The attitude controller is defined as

$$\vec{\tau} := -2K_\eta \ln \boldsymbol{q}_e - k_{\dot{\eta}}\,\vec{\Omega}$$

where $\vec{\Omega}$ is the angular velocity vector, K_η and $K_{\dot{\eta}}$ are gains matrices, and $\boldsymbol{q}_e \in \mathbb{H}$ is the quaternion attitude error defined

as

$$\boldsymbol{q}_e := \boldsymbol{q}_d^* \otimes \boldsymbol{q}$$

In [22] the stability analysis is demonstrated ensuring the convergence to the desired landing trajectory.

Before to land, the aerial vehicle must be aligned with the mobile platform. For this case, $x_d = \hat{x}_t$ and $y_d = \hat{y}_t$, where $\hat{x}_t$ and $\hat{y}_t$ are just the first two components of $\hat{\vec{x}}_{k|k+N}$. Concerning z_d, the reference altitude decreases slowly over time to prevent blurs and have better detection conditions.

V. Results and Discussions

To validate and show the efficiency of the proposed solution, some tests have been carried out in the indoor arena considering two challenging scenarios. The experimental set up includes the ground station OptiTrack computer, the OptiTrack system (MoCap), and the two robots (aerial and ground). The aerial vehicle is a Parrot AR 2.0 drone equipped with a vertical camera with a resolution of 320×240 pixels, focal length of 27.17 mm, and rate of 60 Hz. The UGV is a JetRacer with a Jetson Nano computer onboard. In order to put the marker on the robot and let the quadcopter lands on it, a metallic structure with a grid has been installed on the agent. The ground vehicle has a dedicated controller for moving slowly in line. Both the experiments are considered in three stages: a) hover flight at a fixed altitude, b) alignment with respect to the ground vehicle, c) landing. As the main goal is to prove the vision algorithm for landing, in the first stage, the drone takes off, in manual mode, close to the target until a desired altitude, with the goal to cover the marker in the camera frame. After that, the second stage begins. Once the aerial vehicle is aligned with respect to the marker frame (ground vehicle), the last stage starts. In real-time experiments, the "landing" task ends when the quadcopter reaches a constant height small enough over the marker (ground robot). At this time, the user can declare the end of the mission and then stop the motors. It is worth pointing out that in both case studies the KF parameters have been tuned. Indeed some experiments were done using only in the closed-loop system the MoCap measurements. In those experiments images were recorded from the camera as well as the OptiTrack measurements in a *rosbag* file. Then offline, the *rosbag* has been run together with all the implemented ROS nodes and by trial and error the parameters have been tuned to get the best as possible estimates.

A. Static landing platform

These KF parameters for the landing in a static platform are the following

$$\begin{aligned} x_0 &= \begin{bmatrix} 0.5 & -0.5 & -0.2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\ P_0 &= diag(2.25,\ 2.25,\ 1\text{e}{-27},\ 2.25, \\ &\quad 1\text{e}{-27},\ 1\text{e}{-27},\ 1\text{e}{-27},\ 1\text{e}{-27}) \end{aligned}$$

while the covariance matrices of the model and measurement noises are

$$Q = 1\text{e}{-14}\, I_{8\times 8} \quad R = diag\,(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\psi^2)$$

where $I_{8\times 8}$ denotes the identity matrix, $\sigma_x = \sigma_y = \sigma_z = 0.7$, and $\sigma_\psi = 0.01449$. Figure 5 shows that the custom marker is detected for almost the entire duration of the maneuver. Concerning the estimation and prediction of the horizontal position, Figures 6 and 7 illustrate these performances. From figures, the estimated values using just ArUco markers is in orange line, the prediction values after $N = 5$ steps is in blue line and the position of the target using the MoCap system is in black line.

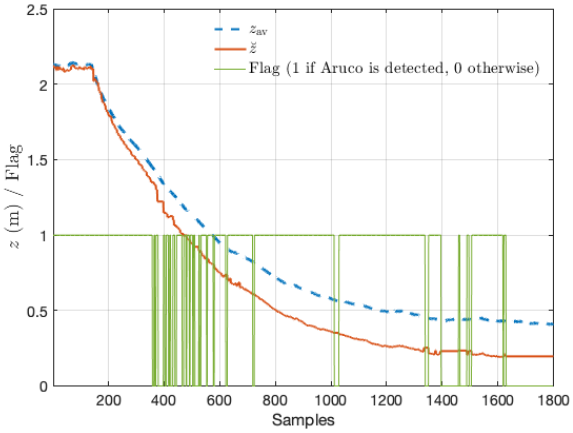

Fig. 5: Altitude performance of the drone when using the vision algorithm. Notice from figure that in green line the 'flag' used for notifying if the marker is detected or not. Notice also that the vision estimation $\check{z}$ of the target is pretty well even if the target is not detected.

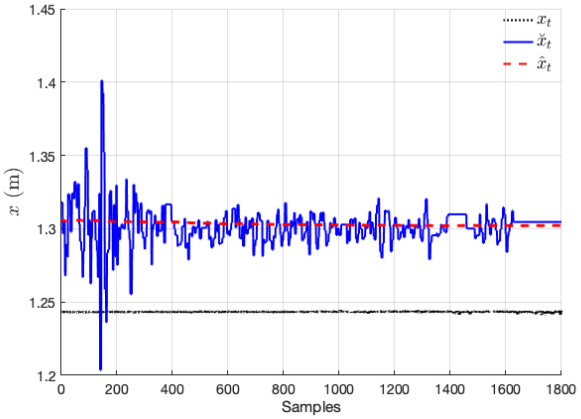

Fig. 6: Horizontal performance of the target x_t, its estimated value using the vision algorithm, $\check{x}_t$ and its estimated prediction $\hat{x}_t$.

Finally, Figure 8 shows the predicted horizontal velocities of the target $\hat{v}_{x_t}$ and $\hat{v}_{y_t}$. Notice that there is obviously no changes in these estimations because the platform is still. A video of this experiment can be seen at : https://youtu.be/ftxDdAU0fHA.

B. Moving landing platform

The next experiment considered the scenario where the target is in motion. The reference trajectory for the UGV

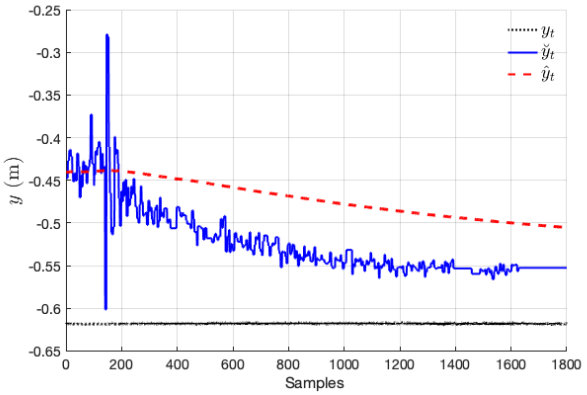

Fig. 7: Position behavior of the target y_t, its estimated value using the vision algorithm, $\breve{y}_t$ and its estimated prediction $\hat{y}_t$.

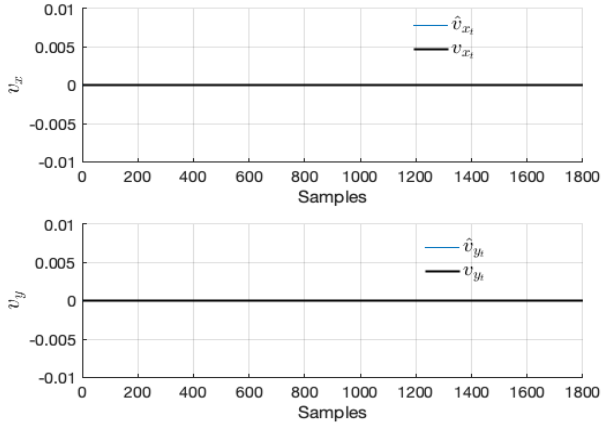

Fig. 8: Predicted translational velocities of the still target in the horizontal plane.

has been chosen as a straight line along the x-axis with a constant reference velocity, to keep everything linear.

The test is organized as follows: in a first phase, the drone tracks (during a small time) the ground robot with a constant altitude and using the MoCap measurements. This is done for converging the KF estimations and avoiding undesired behaviors. In a second stage, the aerial drone follows and land, autonomously, on the UGV using the vision information.

The new values for the initial guesses and the model noise covariance are

$$\begin{aligned} x_0 &= \begin{bmatrix} -0.5 & -0.5 & -0.2 & 0.15 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\ P_0 &= diag(4,\ 4,\ 1e{-}27,\ 2.25, \\ &\quad 0.01,\ 0.01,\ 1e{-}27,\ 1e{-}27) \\ Q &= diag(1e{-}14,\ 1e{-}14,\ 1e{-}14,\ 1e{-}14, \\ &\quad 1e{-}7,\ 1e{-}7,\ 1e{-}27,\ 1e{-}7) \end{aligned}$$

while R is unchanged from the previous experiment.

From the experimental results it can be observed that the altitude decreases more quickly (see Figure 9) and there are time intervals in which the target is not detected due to blurs in the image, which were not the case in the first test.

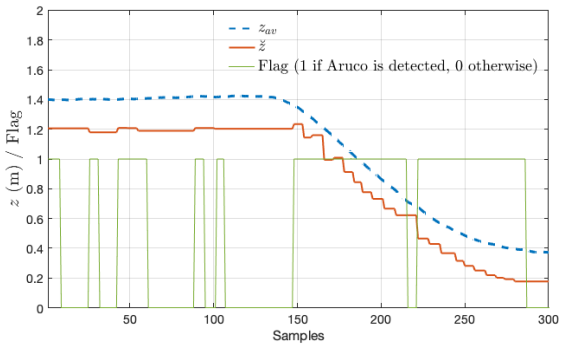

Fig. 9: Altitude of the drone and detection of the custom marker, here in some time intervals the marker was not found.

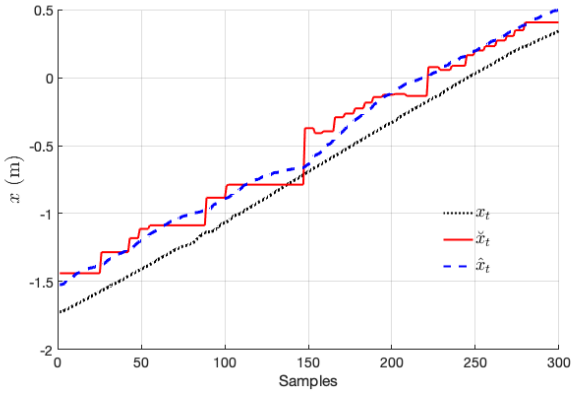

Fig. 10: Horizontal performance of the target x_t, its estimated value using the vision algorithm, $\breve{x}_t$ and its estimated prediction $\hat{x}_t$, when the platform is moving.

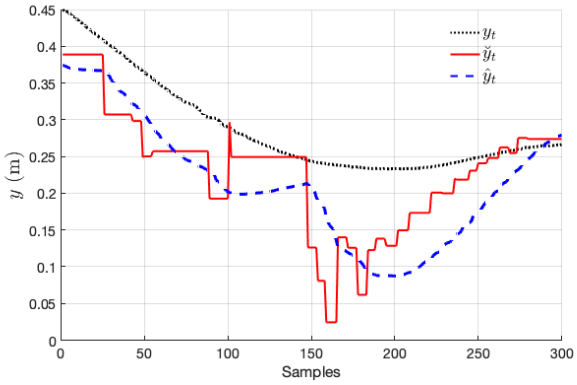

Fig. 11: Lateral performance of the target y_t, its estimated value using the vision algorithm, $\breve{y}_t$ and its estimated prediction $\hat{y}_t$, when the platform is moving.

Concerning the accuracy of the estimated position, the mean error between the true value x_t and the predicted value $\hat{x}_t$ is 0.18 m while between y_t and $\hat{y}_t$ is -0.06 m, see Figures 10 and 11. Even if some errors (no detection) are presented in the experiment, this does not prevent the success of the mission, nevertheless, we consider that a better tuning of the parameters could lead to better results. Observe also that experiment works well because following the hypothesis 2) in section II, the measure obtained in the

last detection is used as current measure $\vec{y}$ to give as input to the KF. This is reasonable since, assuming the marker is moving slowly, the e-ArUco will be in a neighborhood of the previous position. In addition it is worth pointing out that the horizontal reference is not the prediction from the KF $\hat{x}_{k|k+1}$ but $\hat{x}_{k|k+N} = A^{N-1}\hat{x}_{k+1|k}$, so that the UAV can anticipate the UGV and keep it in the field of view. A video of this experiment can be seen at : https://youtu.be/9gGIiv-lWzY.

VI. Conclusions

In this paper, a solution for autonomous landing of an aerial vehicle devoid of GPS measurements, using vision algorithms was presented. The challenge was complicated when introducing the dare to land on a moving platform. The solution used fiducial markers for estimating the pose of the target to land. Two different experiments were proposed for validating the landing solution. In both scenarios, tests shown that accurate estimates and predictions concerning the UGV state can be obtained with the proposed methodology, ensuring the success of the landing mission. Nevertheless, some further improvements can be made.

Future work will consider the relative pose between the agents, expressed either in the camera or in the Fl-Air reference, to control the drone. Observe that this would introduce some non-linearities in the evolution and measurement models. In case those non-linearities are strong, the KF will be no more sufficient and it will be necessary to resort to more sophisticated tools. In the perspective of doing some outdoor experiments, the Parrot AR 2.0 cannot be used as it does not come with a built-in GPS receiver. Nonetheless, other kind of aerial vehicles need to be considered as, Parrot Bebop 2.0 and the Intel Aero Drone that are equipped with GPS-capabilities. In addition it could be interesting to introduce an ascending phase in case of no detection target. By increasing its height, the drone will have a wider field of view of the scene and potentially it will find again the target.

Acknowledgment

This paper was supported by the Heudiasyc UMR CNRS 7253 laboratory and the National Network of Robotics Platforms (ROBOTEX) in France. Special thanks to G. Sanahuja and T. Monglon for their help during the experimental tests.

References

[1] L. Xin, Z. Tang, W. Gai, and H. Liu, "Vision-Based Autonomous Landing for the UAV: A Review," *Aerospace*, vol. 9, no. 11, p. 634, Oct. 2022. [Online]. Available: https://www.mdpi.com/2226-4310/9/11/634

[2] Z. Zhao, P. Han, Y. Xu, W. Xie, W. Zhang, K. Liang, and Q. Zeng, "Vision-based Autonomous Landing Control of a Multi-rotor Aerial Vehicle on a Moving Platform with Experimental Validations," *IFAC-PapersOnLine*, vol. 55, no. 3, pp. 1–6, 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896322002658

[3] Y. Zhu, H. Pei, L. Wang, Y. Huang, and K. Ou, "A Vision-Based Autonomous Landing Method on Mobile Platform for UAV," in *2023 42nd Chinese Control Conference (CCC)*. Tianjin, China: IEEE, Jul. 2023, pp. 4089–4094. [Online]. Available: https://ieeexplore.ieee.org/document/10240611/

[4] P. Ladosz, M. Mammadov, H. Shin, W. Shin, and H. Oh, "Autonomous Landing on a Moving Platform Using Vision-Based Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4575–4582, May 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10476692/

[5] D. Pieczyński, B. Ptak, M. Kraft, M. Piechocki, and P. Aszkowski, "A fast, lightweight deep learning vision pipeline for autonomous UAV landing support with added robustness," *Engineering Applications of Artificial Intelligence*, vol. 131, p. 107864, May 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0952197624000228

[6] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, p. 2280–2292, 06 2014.

[7] M. Theunissen, H. Pousseur, P. Castillo, and A. Victorino, "Co-operative architecture using air and ground vehicles for the search and recognition of targets," in *IEEE International Conference on Intelligent Transportation Systems ITSC*, 2023.

[8] Y. Park, C. Park, W. Song, C. Lee, J. Kwon, J. Park, G. Noh, and D. Lee, "Fiducial Marker-Based Autonomous Landing Using Image Filter and Kalman Filter," *International Journal of Aeronautical and Space Sciences*, vol. 25, no. 1, pp. 190–199, Jan. 2024. [Online]. Available: https://link.springer.com/10.1007/s42405-023-00635-y

[9] R. Acuña, D. Zhang, and V. Willert, "Vision-based uav landing on a moving platform in gps denied environments using motion prediction," 11 2018, pp. 515–521.

[10] O. Elmakis, T. Shaked, and A. Degani, "Vision-based uav-ugv collaboration for autonomous construction site preparation," *IEEE Access*, vol. 10, pp. 1–1, 01 2022.

[11] J. Morales, I. Castelo, R. Serra, P. Lima, and M. Basiri, "Vision-based autonomous following of a moving platform and landing for an unmanned aerial vehicle," *Sensors*, vol. 23, p. 829, 01 2023.

[12] L. Shao, Z. Guo, J. Yang, and S. Li, "Vision-Based Quadrotor Rapid Landing Control with An Uncooperative Platform: An Alternating Predictive Observer Approach," *IEEE Transactions on Intelligent Vehicles*, pp. 1–13, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10480572/

[13] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 06 2018.

[14] M. Tian, L. Pan, M. H. Ang Jr, and G. H. Lee, "Robust 6d object pose estimation by learning rgb-d features," in *International Conference on Robotics and Automation (ICRA)*, 2020.

[15] N. Piga, Y. Onyshchuk, G. Pasquale, U. Pattacini, and L. Natale, "Roft: Real-time optical flow-aided 6d object pose and velocity tracking," *IEEE Robotics and Automation Letters*, vol. 7, pp. 1–1, 01 2021.

[16] X. Du, A. Li, X. Li, and Q. Liu, "Summary of trajectory prediction method for unmanned ground vehicle," 10 2022, pp. 340–343.

[17] W. Luo, H. Ebel, and P. Eberhard, "An lstm-based approach to precise landing of a uav on a moving platform," *International Journal of Mechanical System Dynamics*, vol. 2, 04 2022.

[18] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, I. G. Moreno, and P. Campoy, "A Deep Reinforcement Learning Technique for Vision-Based Autonomous Multirotor Landing on a Moving Platform," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, Oct. 2018, pp. 1010–1017. [Online]. Available: https://ieeexplore.ieee.org/document/8594472/

[19] V. Saj, B. Lee, D. Kalathil, and M. Benedict, "Robust Reinforcement Learning Algorithm for Vision-based Ship Landing of UAVs," Sep. 2022, arXiv:2209.08381 [cs]. [Online]. Available: http://arxiv.org/abs/2209.08381

[20] A. Khazetdinov, A. Zakiev, T. Tsoy, M. Svinin, and E. Magid, "Embedded aruco: a novel approach for high precision uav landing," 05 2021, pp. 1–6.

[21] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, ser. Open-Source Software workshop, 4 2013, pp. 1–6.

[22] J. Cariño, H. Abaunza, and P. Castillo, "Quadrotor quaternion control," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015.