Autonomous Landing of an Unmanned Aerial Vehicle on a Mobile Platform

2023 7th International Conference on Automation, Control and Robots (ICACR) | 979-8-3503-0288-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICACR59381.2023.10314602

Felix Sihitshuwam Dalang
Robotics and Computer Vision
Innopolis University
Innopolis, Russia
f.dalang@innopolis.university

Geesara Kulathunga
Center of Autonomous Technologies
Innopolis University
Innopolis, Russia
ggeesara@gmail.com

Ramil Khusainov
Robotics
Innopolis University
Innopolis, Russia
r.khusainov@innopolis.ru

***Abstract*—Landing an Unmanned Aerial Vehicle (UAV) on a mobile vehicle, has been an active research area due to its much needed applicability in numerous directions such as landing an aerial vehicle on the international space station or a ship at sea. Such research is conducted mainly in two directions: control-based and vision-based. Vision-based approaches require sensing from cameras that detect features, such as visual markers, on the mobile vehicle to estimate and track the mobile vehicle pose till the landing pose. The visual markers are usually utilized for feature detection, which most mobile vehicles might not have. This paper proposes a new approach for achieving autonomous landing of a UAV by generating constrained uniform B-spline trajectories assuming known pose of both the UAV and the UGV. Then applying PID controller to regulate velocity set point to land the UAV on the mobile vehicle. The overall effectiveness of the proposed approach is demonstrated using ROS/Gazebo simulations, with a UAV (Px4 enabled IRIS) and a mobile vehicle (Jackal from ClearPath Robotics). Landing tasks for static and dynamic cases are carried out and the findings and limitations are stated thereafter.**

***Keywords*—UAV Landing, UGV, PID, B-Spline Interpolation, Autonomous Landing**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have been used intensively in recent years [1]. UAV industrial applications range from exploration [2]–[4], construction [5], map building [6], [7], equipment inspection [8]–[10], military [11]–[13] and civilian [14] surveillance.

However, since UAVs have limited battery capacity [15], they require frequent charging and mobile charging platforms offer a solution to this problem. Moreover, UAVs might require landing on moving larger aerial, space [2], aquatic vehicles or quadruped robots [16].

Such landing involves locating a point on the mobile platform, tracking this point and treading a trajectory to reach this point. At each time step, the trajectory is dynamic since initially, the mobile platform is moving.

This *research* proposes an odometry-based system wherein, the UAV autonomously follows the *mobile platform* (MP), using pose and odometry data from the MP, and finally landing on the MP when it is close enough. Following this high-level framework, this research aims to achieve the following:

This research was supported by the Russian Science Foundation (RSF) and Ministries of the Republic of Tatarstan (grant agreement 22-21-20108).

- Extract pose data from the localization data of the Mobile Platform, and use these data to generate the pose of the MP in a global frame. Then transform these data to the *UAV*.
- Use B-Spline Interpolation techniques to generate trajectories from the *UAV* to the *MP* continuously.
- Control the *UAV* by sending velocity commands.
- Land the *UAV* when it is sufficiently close to the mobile platform.
- Test the implementation using realistic multi-robot simulations.

This paper is organized as follows:

- Section II describes the research related to this research topic.
- Section III describes the ground and aerial vehicles, simulation tools and the required equipment used in this research.
- Section IV presents the implementation of the autonomous tracking, following, and landing.
- Section V discusses the simulation results
- VI concludes the paper.

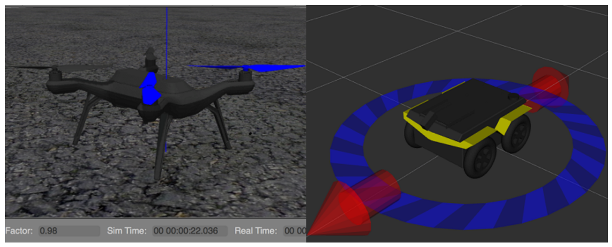

Fig. 1. Px4 Iris - A Quadrotor and a Jackal.

II. Related Works

The research in this field has mainly been based on using *marker* techniques and different control approaches to track, follow, and land on a mobile platform. Common fiducial markers include ARTag, AprilTag, ArUcO and STag [17]. However, [18] and [19] used custom markers in autonomous landing. The approach in [1], involved ArUcO markers to obtain an estimate of the relative pose between the UAV and the MP. The images obtained from a monocular camera on the UAV is processed to obtain pose estimates which were then fed to a trajectory planner and controllers generating speed set-points for following and landing. Testing their pose estimations, yielded accurate results, aligning to the ground truth data with ± 0.01mm accuracy. A limitation of this system was that markers were lost when moving out of the camera view, leading to no, or poor relative pose estimation. Thus the MP-mounted marker, which must be always in the view of the UAV camera. This limitation was addressed by ensuring that the UAV always moved faster than the MP. The other solution was to use a wide angle camera. Simulation and Real-world tests were properly performed. However, real-world proved to be less accurate due to external disturbances.

[20], used Model Predictive Control (MPC) to autonomously land a UAV. A linear MPC technique was used to track an MP indoors with high accuracy. The high level task was segmented; target detection, target tracking and autonomous landing. Initially, an MPC controller was formulated, then a guidance law was developed. The guidance law had the following four states, implemented in a state machine paradigm: Take-Off, Patrol, Tracking and, landing. Control returns to Patrol if a target is lost at any state.

When simulated, and compared against Backstepping, LQR and PID, used by [21] for path tracking by quadcopter, Backstepping and MPC had the best response times. However, the MPC controller had less overshoot and hence was the better alternative for target tracking. The landing was also successful and worked well according to the control.

[22] explored the autonomous landing of a UAV on an MP from a mathematical perspective. They hypothesized that a succcesful landing in this context requires accurate localizing of the landing zone, fast trajectory planning, and a robust control configuration. To achieve this goal, the following data are crucial: the intersection point between the UAV and the mobile platform, the pose of the UAV/mobile platform and the inclination angle that is needed to land. Therefore these data must be intensely observed. The approach limited the configuration space of the UAV and the platform to the $X-Y$ plane. Thus the researchers derived the intersection point, position profile from takeoff to landing and the inclination angle needed to land the UAV on a mobile platform using mathematical equations. The method required knowing air density and mass of the UAV. The experimental results were presented succesfully, with one advantage of the system being that the distance between the UAV and the MP can be predicted in real time.

[23] tackled the UAV landing problem in dynamic environments using the PID controller combined with deep reinforcement learning and corrective feedback based on heuristics. PID parameter tuning is notoriously diificult and usually done by human designers. It is time-consuming and challenging for UAV landing. The parameter tuning is hence achieved via heuristics accelerating the reinforcement learning. Particularly, [23] uses the Deep Deterministic Policy Gradient (DDPG) reinforcement learning algorithm to tune the parameters of the PID controller. This controller at the low control level provides fast reactive signals that dictate the speed of the rotors and generally the velocity of the UAV. The dynamics of the UAV are derived, then inertial and body frames for the UAV and for the mobile platform are specified. Afterwards, the PID controller is derived and a framework is generated to integrate the PID controller with the reinforcement learning agent which has corrective feedback as previous explained. Experiments were carried out in Gazebo, and the UAV used had a downside camera used for detection and localization, aided by markers with circular patterns. The approach yielded a near 100% success rate.

The work done by [24], proposed an optimization-based reference trajectory tracking for UAVs executing slow-speed manoeuvres. NMPC combines trajectory generation and tracking as one problem and solves it. The approach in this study was restricted to four degrees of freedom for a quad-copter; three degrees for position plus one degree for the yaw angle. Reducing the degrees of freedom was to reduce computational complexity. The objective was to minimize the error between the quad-copter's current pose and the reference trajectory at each time step. This reference trajectory was generated using the B-Spline interpolation technique and ensured dynamic feasibility.

III. Methodology

A. Simulation Hardware and Software

The system used consisted of simulations involving a drone and a UGV. All tasks were performed in Ubuntu 18.04 and ROS Melodic. The ROS packages used included *multi_jackal* and *tf/tf2* for transformation.

The drone was based on the *Px4 Autopilot* using a classic quadrotor as shown in the image on the left side of Fig. 1. Control interface for this drone was provided via PX4 which supports both Software In the Loop (SITL) and Hardware In the Loop (HITL) simulations. Communication was achieved using MAVROS which is a MAVLink to ROS proxy.

For the MP, the simulations of a Jackal mobile robot from Clearpath robotics was used. This is included into the ROS/Gazebo simulations using the *multi_jackal* ROS package. An image of this vehicle is shown on the right side in Fig. 2 modified to include a landing pad.

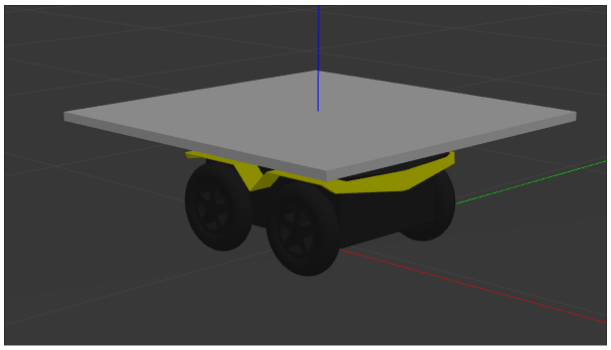

Fig. 2. Modified Jackal having a landing pad

As described in Section II, at a high level, the task of the autonomous landing of a UAV on a mobile platform can be segmented into smaller chunks. These chunks are listed and described below as used in this work.

B. Pose Detection, Estimation, and Transformation.

Pose detection and estimation was achieved by getting pose data from the mobile platform and the aerial vehicle. Pose data for the mobile platform was x_t, y_t. This was sufficient, while for the drone it was x_t, y_t, z_t, θ_t, with θ being the yaw angle. The reason for the choice of these particular parameters would be explained subsequently.

Several techniques exist to localize a robot as compared in [25], [26]. Mobile robots generally include at least one form of localization to estimate their pose relative to a global frame. Thus, the proposed solution assumed that after proper localization, accurate pose data from the aerial vehicle and the mobile platform was made available. Internally, the vehicles in the simulation utilize Extended Kalman Filter (EKF) for localization to obtain their individual pose estimates from their local frames to the global frame.

Thus, after obtaining these pose data, the proposed solution began by generating transformations to relate the pose of the mobile platform and the pose of the aerial vehicle. Pose data from the following three different frames were required:

- the pose of the drone in its local frame, p_l^d;
- the pose of the drone in the global/world frame, p_g^d;
- the pose of the mobile vehicle in the global frame, p_g^m;

p_l^d, is required because control of the UAV is always in its local frame. The important pose data from the drone used by the proposed solution dealt with position in the x, y, z, orientation data was not required as much.

Given these data, the MP pose in the drone frame, denoted as p_d^m was then derived.

C. Trajectory Generation

Trajectory was generated from the drone to a reasonable altitude above the mobile platform. Two set of pose data were required: the pose of the drone in its local frame p_l^d and the pose of the mobile platform in the drone frame, p_d^m. The destination z was set relative to the height of the mobile platform, a little distance above the height of the mobile platform.

As explained in Section II, the B-Spline interpolation technique is used to generate a trajectory [24]. This technique had the advantage of being able to generate derivatives useful for control purposes. The B-Spline technique generates a spline in the B-spline basis. The Spline produced the trajectory, while its 1^{st} order derivative gave the velocity with which a vehicle should follow this trajectory.

At least four control points were needed to generate a B-Spline having derivatives up to the 3^{rd} order.

D. Trajectory Tracking

After trajectory generation, the trajectory was tracked using a regulator which used the position and velocity information from the trajectory generator. A quad-rotor can be controlled via position control (x, y, z), velocity control $(\dot{x}, \dot{y}, \dot{z}$ and yaw rate) or via attitude (orientation relative to a world frame). Velocity control was chosen, because it has an added advantage of solving the problem of high *jerk* which causes vibrations in the system.

To reduce the computational intensity, a 4-DoF motion model was utilized. The simplified motion model was expressed by (1)

$$\dot{x}_t = \begin{bmatrix} \dot{p}_t^x & \dot{p}_t^y & \dot{p}_t^z & \dot{\alpha}_t^z \end{bmatrix}^T = \begin{bmatrix} v_t^x & v_t^y & v_t^z & \omega_t^z \end{bmatrix}^T \tag{1}$$

A PD controller is then used to regulate the UAV to the desired pose.

E. UAV Landing

UAV landing was done via a control loop. Initially the drone moved to the target location, i.e. the centre of the mobile platform. The control loop checked to see if the MP had deviated significantly. If it deviated significantly, a new trajectory was generated which had position and velocity commands to move the drone to the new target pose. If the MP did not deviate deviate significantly when the drone arrived to the reference pose, the drone adjusted its pose relative to the MP center and then landed on it.

As stated above, the target altitude was designed as a factor of the height of the mobile platform. This was such that once the drone arrived the reference pose, a command was issued to the drone to land. which is just a small distance from the height of the MP and then a signal is issued to de-activate (disarm) itself.

IV. Implementation

A. Transformation

Poses were obtained from the UAV, and MP and transformed to the world frame. Transformation was achieved via static transforms that gave the initial transformation between the UAV to the world frame and between the MP to the world frame. Then a transform between the mobile platform and the UAV was calculated. p_g^m was obtained via **tf** transformations while p_g^d was obtained by summing the local pose with a known global static transform $p_{transform}^{static}$.

$$p_g^d = p_l^d + p_{transform}^{static} \tag{2}$$

$$p_d^m = p_g^m - p_{transform}^{static} \tag{3}$$

$$p_{tr}^m = p_g^m - p_g^d \tag{4}$$

The tracking position of the mobile platform relative to the drone, p_{tr}^m, was calculated as shown in (4). It gave the relative distance between the UAV and the mobile platform at each time step.

The poses, p_{tr}^m and p_l^d were used to generate trajectory. p_{tr}^m was used instead of p_d^m because it provided a way of knowing if the mobile platform had deviated sufficiently to re-generate a trajectory or not.

B. Trajectory Generation

Trajectory was generated using the B-Spline interpolation technique which gives a smooth continuous curve. Four control points were required, with two of them being the UAV pose p_l^d and the pose of the mobile platform in the drone frame p_l^m. The remaining two were intermediate points, between p_l^d and p_d^m which are calculated as shown in the psedocode in Fig 3.

Algorithm 1 Control Points

```
$x,\ y,\ z \leftarrow p_l^d$
$x_1,\ y_1,\ z_1 \leftarrow p_l^d + p_{tr}^m$
$p_0 \leftarrow [x,\ y,\ z]$
$p_3 \leftarrow [x_1,\ y_1,\ z_1]$
if $x > x_1$ then
    $dx \leftarrow x - x_1$
else
    $dx \leftarrow x1 - x$
end if
if $y > y_1$ then
    $dy \leftarrow y - y_1$
else
    $dy \leftarrow y1 - y$
end if
$z_c \leftarrow 3$ ▷ Clearance z, any value some distance above mobile platform height
$p_1 \leftarrow [x + (0.25dx), y + (0.25dy), z_c]$
$p_2 \leftarrow [x + (0.75dx), y + (0.75dy), 0.5z_c]$
if $x > x_1$ then
    $p_1[0] \leftarrow x - 0.25dx$
    $p_2[0] \leftarrow x - 0.75dx$
end if
if $y > y_1$ then
    $p_1[1] \leftarrow y - 0.25dy$
    $p_2[1] \leftarrow y - 0.75dy$
end if
return $p_0, p_1, p_2, p_3$
```

Fig. 3. Transformations between frames.

The B-Spline trajectory generator is given by (5), with $S(t) \in \Re^3$

$$S(t) = \sum_{i=0}^{n} N_{i,j}(t) P_i \tag{5}$$

where $N_{i,j}$ is the basis function and P_i is the i^{th} control point. j refers to the polynomial degree and there are $n + 1$ control points, $n = 3$ in this case.

The basis function was calculated using the **Cox-de Boor** recursive formula.

$$N_{i,0}(t) = \begin{cases} 1 \text{ if } t_i \leq t \leq t_{i+1} \\ 0 \text{ otherwise} \end{cases} \tag{6}$$

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t) \tag{7}$$

The derivative of the spline gives the velocity, and is given by (8) and the derivative of the basis function $\dot{N}$ is given by (9)

$$\dot{S}(t) = \sum_{i=0}^{n} \dot{N}_{i,k}(t) P_i \tag{8}$$

$$\dot{N}_{i,j}(t) = (j-1)\left[\frac{N_{i,j-1}(t)}{t_{i+j} - t_i} - \frac{N_{i+1,j-1}(t)}{t_{i+j+1} - t_{i+1}}\right] \tag{9}$$

C. Tracking Control

Once the trajectory generator was created, it was indexed from start to finish, with each time index producing the reference position and velocity, from the B-Spline generator described above. A value of $0.08s$ was used for the sampling time (δ). This was chosen experimentally from observation. Generally, for control applications a sampling time of $0.04 \leq \delta \leq 0.08$ sufficed for most applications. The reference position ($p_r \in \Re^3$) and velocity ($v_r \in \Re^3$) for each time index were sent to the PD regulator.

$$p_e = p_r - p_c \tag{10}$$

$$v = p_e \cdot k_p - v_c \cdot k_d \qquad v \in \Re^3 \tag{11}$$

$$\tag{12}$$

where $p_c,\ v_c$ is the current UAV position and velocity, respectively, while $k_p = 0.8$ and $k_d = 0.27$ are position proportional and differential constants for the PD regulator, derived experimentally, and v is the position velocity sent to the UAV's low-level set point controller.

Similarly, the yaw rate (ω) is calculated as

$$\alpha_r = atan2(v_{ry},\ v_{rx}) \tag{13}$$

$$\alpha_e = \alpha_r - \alpha_c \tag{14}$$

$$\omega = \alpha_e \cdot p_a - \omega_c \cdot d_a \tag{15}$$

where v_{ry} and v_{rx} are the reference velocities in the x and y directions respectively, α_c is the current yaw angle of the UAV, $p_a = 2.5$ is the angular proportional constant, $d_a = 0.5$ is the angular differential constant, derived experimentally and ω is the yaw rate sent to the UAV's low-level controller.

When tracking, p_{tr}^m was always checked to see if it had substantially deviated from its initial value when the trajectory was generated, if it did deviated, then a new set of control points were calculated and fed to the trajectory generator which supplies reference position and velocity information for tracking.

D. Landing

The final landing took place when p_l^d is at (or very close to) p_d^m, mathematically,

$$p_{tr}^m \leq threshold \text{ (or)} \tag{16}$$

$$euclidean\_dist(p_d^m, p_l^d) \leq threshold \tag{17}$$

The Euclidean distance was calculated only for x, y of the two poses, since the reference trajectory was generated to get the z close to the desired point above the MP and including it in the calculation of euclidean distance will not yield a desired effect.

The landing was performed by calling functions on the drone which simply dropped it on the MP and disarmed it.

The autonomous landing is an iterative process that starts from pose detection to final landing as has been explained. Based on the iterative process in [20], trajectory was regenerated each time the robot moved substantially from its initial pose when the trajectory was generated. A pseudo-code for this architecture is shown in Fig. 4.

Algorithm 2 Autonomous Landing Loop

```
$startTime = null$
$endTime = null$
$t_s = 0.08$
$finishedTrajectory \leftarrow false$
while $true$ do
    $p_l^d \leftarrow getDroneLocalPose()$
    $p_{tr}^m \leftarrow getMobilePlatformTrackingPose()$
    if $platformHasMoved()$ then                    ▷ platform has changed position
        $c_p \leftarrow getControlPoints(p_l^d,\ p_{tr}^m)$
        $traj \leftarrow generateTrajectory(c_p)$
        $startTime = traj.getStartTime()$
        $endTime = traj.getEndTime()$
    end if
    $dist \leftarrow euclidean(p_l^d,\ p_{tr}^m)$
    if $norm(p_{tr}^m) \leq threshold$ and $finishedTrajectory$ then
        landDrone()
        break
    end if
    if $finishedTrajectory$ then
                            ▷ jackal hasn't moved enough to regenerate trajectory
        $p \leftarrow p_l^d$
        $v \leftarrow 0$                                ▷ compensated for by regulator
    else
        $p \leftarrow traj.getRefPose(startTime)$
        $v \leftarrow traj.getRefVelocity(startTime)$
    end if
    $moveDrone(p,\ v)$
    $startTime = startTime + t_s$
    if $startTime > endTime$ then
        $finishedTrajectory \leftarrow true$
    end if
end while
```

Fig. 4. Transformations between frames.

V. Results and Discussion

The proposed solution was tested via several simulation experiments designed to ensure correct transformations between reference frames, accurate trajectory generation, accurate tracking, eventual landing when the mobile platform is stationary; and, trajectory generation and regeneration when the mobile platform is moving and eventual landing.

Transformations were easily verified by analytic means, given p_g^m and p_l^d, outputted p_{tr}^m was generated by the written program which aligns with the expected analytical result.

Trajectory generation and following also worked accurately as shown in Fig. 5.

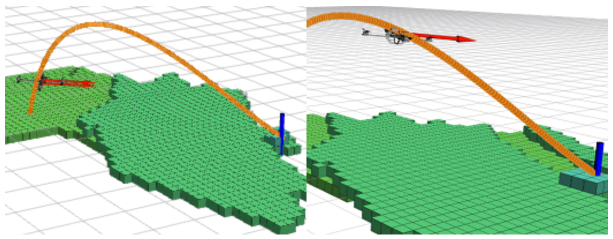

Fig. 5. Trajectory generation and following.

The reference path tracking worked well as shown in Fig. 6 which shows the generated trajectory and the trajectory traversed by the UAV with the path followed by the mobile platform. The space between the UAV trajectory and the mobile platform is the height of the mobile platform. The UAVs trajectory showed that it changed as the mobile platform moved, until it was able to land on the MP.

Landing occured at the end of the trajectory, or when the UAV and the UGV were sufficiently close. A threshold of $0.1m$ was used to define this *closeness*, setting it at a lower

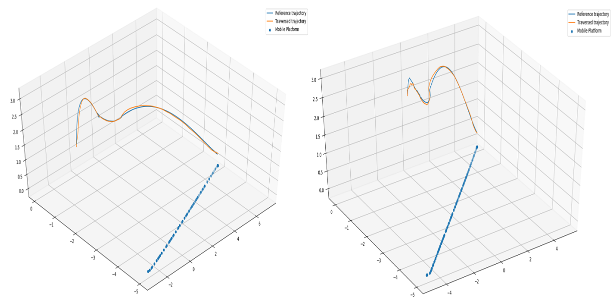

Fig. 6. Generated, actual UAV trajectory & mobile platform trajectory.

value complicated landing when the platform was moving. Gazebo simulations in Fig. 7 shows landing.

Table I shows the success rate given a deviation threshold. A new trajectory was generated if the mobile platform moved beyond this threshold. According to Table I, a landing status of *MPS* indicates that the UAV is only able to land when the vehicle has stopped moving (i.e, reaches its destination), while *Success* indicates that the UAV is able to land on the UGV while it is in motion. Obviously, a higher deviation threshold made landing while moving successful. This was because when a trajectory is generated, the drone always followed this trajectory but the MP might have moved by the time the UAV reached the end of the trajectory. Setting a higher threshold means that the UAV tries to get to the drone rather than generating and following a trajectory.

TABLE I
Deviation Thresholds and Landing Status

Deviation Threshold (m)	Landing Status
0.2	MPS
0.8	MPS
1.0	Success

Table II shows the relationship between UAV/UGV maximum velocities and the landing status. When the UGV had a high velocity relative to the UAV, landing only occurred when the UGV stops. This is because, when the UAV reached the end of its trajectory, it discovered that the UAV was no longer there, thus, trajectory regeneration continued. At a low UGV velocity, the UAV could land on the platform even while it was in motion. This supports findings by other researchers which indicate that, generally, the UAV must have a higher velocity than the mobile vehicle for landing to become possible.

TABLE II
Velocities and Landing Status

UAV (m/s)	UGV (m/s)	Landing Status
1.0	1.0	MPS
1.0	0.2	MPS
1.0	0.1	Success

The results show that a fundamental assumption of the success of the proposed solution is that the UAV and the MP localize accurately and produce accurate pose data.

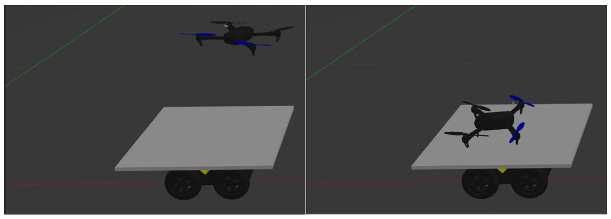

Fig. 7. UAV Tracking & Landing.

VI. Conclusion

This research sets out to perform the autonomous landing of a UAV on a mobile platform. The solution is carried out using control strategies and odometry data without additional sensing input from LiDAR or cameras. Implementation of the proposed solution was done in ROS.

The framework was tested using realistic simulations from Gazebo and RViz. Testing was done for stationary and dynamic cases. In all cases, the aim of the study was found to be achieved successfully under specified conditions.

The suggested future work is to include obstacle avoidance on the trajectory of the UAV, and to utilize Behavior Trees for a more robust control architecture.

Acknowledgment

Special appreciation goes to Dr Georgii Gelvanovsky for proofreading this paper, and to the Education and Student Affairs Departments at Innopolis University.

References

[1] J. Morales, I. Castelo, R. Serra, P. U. Lima, and M. Basiri, "Vision-based autonomous following of a moving platform and landing for an unmanned aerial vehicle," *Sensors*, vol. 23, no. 2, p. 829, 2023.

[2] M. Hassanalian, D. Rice, and A. Abdelkefi, "Evolution of space drones for planetary exploration: A review," *Progress in Aerospace Sciences*, vol. 97, pp. 61–105, 2018.

[3] B. Heincke, R. Jackisch, A. Saartenoja, H. Salmirinne, S. Rapp, R. Zimmermann, M. Pirttijärvi, E. V. Sörensen, R. Gloaguen, L. Ek *et al.*, "Developing multi-sensor drones for geological mapping and mineral exploration: Setup and first results from the mulsedro project," *GEUS Bulletin*, vol. 43, 2019.

[4] R. Williams, B. Konev, and F. Coenen, "Multi-agent environment exploration with ar. drones," in *Advances in Autonomous Robotics Systems: 15th Annual Conference, TAROS 2014, Birmingham, UK, September 1-3, 2014. Proceedings 15*. Springer, 2014, pp. 60–71.

[5] F. Elghaish, S. Matarneh, S. Talebi, M. Kagioglou, M. R. Hosseini, and S. Abrishami, "Toward digitalization in the construction industry with immersive and drones technologies: a critical literature review," *Smart and Sustainable Built Environment*, vol. 10, no. 3, pp. 345–363, 2021.

[6] T. F. Sonnemann, J. Ulloa Hung, and C. L. Hofman, "Mapping indigenous settlement topography in the caribbean using drones," *Remote Sensing*, vol. 8, no. 10, p. 791, 2016.

[7] O. Risbøl and L. Gustavsen, "Lidar from drones employed for mapping archaeology–potential, benefits and challenges," *Archaeological Prospection*, vol. 25, no. 4, pp. 329–338, 2018.

[8] M. Shafiee, Z. Zhou, L. Mei, F. Dinmohammadi, J. Karama, and D. Flynn, "Unmanned aerial drones for inspection of offshore wind turbines: A mission-critical failure analysis," *Robotics*, vol. 10, no. 1, p. 26, 2021.

[9] G. Vom Bögel, L. Cousin, N. Iversen, E. S. M. Ebeid, and A. Hennig, "Drones for inspection of overhead power lines with recharge function," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2020, pp. 497–502.

[10] D. Moreno-Jacobo, G. Toledo-Nin, A. Ochoa-Zezzatti, V. Torres, and F. Estrada-Otero, "Evaluation of drones for inspection and control in industry 4.0," *Technological and Industrial Applications Associated with Intelligent Logistics*, pp. 579–595, 2021.

[11] A. Vacca and H. Onishi, "Drones: military weapons, surveillance or mapping tools for environmental monitoring? the need for legal framework is required," *Transportation research procedia*, vol. 25, pp. 51–62, 2017.

[12] I. Verdiesen, A. Aler Tubella, and V. Dignum, "Integrating comprehensive human oversight in drone deployment: a conceptual framework applied to the case of military surveillance drones," *Information*, vol. 12, no. 9, p. 385, 2021.

[13] C. Paucar, L. Morales, K. Pinto, M. Sánchez, R. Rodríguez, M. Gutierrez, and L. Palacios, "Use of drones for surveillance and reconnaissance of military areas," in *Developments and Advances in Defense and Security: Proceedings of the Multidisciplinary International Conference of Research Applied to Defense and Security (MICRADS 2018)*. Springer, 2018, pp. 119–132.

[14] J. P. West and J. S. Bowman, "The domestic use of drones: An ethical analysis of surveillance issues," *Public Administration Review*, vol. 76, no. 4, pp. 649–659, 2016.

[15] D. Kumar, J. Raj, K. Raghuwaiya, and J. Vanualailai, "Autonomous uav landing on mobile platforms," in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 2021, pp. 1–6.

[16] B. Lindqvist, S. Karlsson, A. Koval, I. Tevetzidis, J. Haluška, C. Kanellakis, A. akbar Agha-mohammadi, and G. Nikolakopoulos, "Multimodality robotic systems: Integrated combined legged-aerial mobility for subterranean search-and-rescue," *Robotics and Autonomous Systems*, vol. 154, p. 104134, 2022.

[17] M. Kalaitzakis, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Experimental comparison of fiducial markers for pose estimation," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 781–789.

[18] C. Hui, C. Yousheng, L. Xiaokun, and W. W. Shing, "Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision," in *Proceedings of the 32nd Chinese control conference*. IEEE, 2013, pp. 5895–5901.

[19] A. A. Cabrera-Ponce and J. Martinez-Carranza, "Onboard cnn-based processing for target detection and autonomous landing for mavs," in *Pattern Recognition: 12th Mexican Conference, MCPR 2020, Morelia, Mexico, June 24–27, 2020, Proceedings 12*. Springer, 2020, pp. 195–208.

[20] Y. Feng, C. Zhang, S. Baek, S. Rawashdeh, and A. Mohammadi, "Autonomous landing of a uav on a moving platform using model predictive control," *Drones*, vol. 2, no. 4, p. 34, 2018.

[21] M. Lecointe, C. Ponzoni Carvalho Chanel, and F. Defay, "Backstepping control law application to path tracking with an indoor quadrotor," 2015.

[22] M. Alijani and A. Osman, "Autonomous landing of uav on moving platform: A mathematical approach," in *2020 International Conference on Control, Automation and Diagnosis (ICCAD)*. IEEE, 2020, pp. 1–6.

[23] L. Wu, C. Wang, P. Zhang, and C. Wei, "Deep reinforcement learning with corrective feedback for autonomous uav landing on a mobile platform," *Drones*, vol. 6, no. 9, p. 238, 2022.

[24] G. Kulathunga, D. Devitt, and A. Klimchik, "Trajectory tracking for quadrotors: An optimization-based planning followed by controlling approach," *Journal of Field Robotics*, vol. 39, no. 7, pp. 1001–1011, 2022.

[25] R. Vincent, B. Limketkai, and M. Eriksen, "Comparison of indoor robot localization techniques in the absence of gps," in *Detection and sensing of mines, explosive objects, and obscured targets XV*, vol. 7664. SPIE, 2010, pp. 606–610.

[26] F. N. Sibai, H. Trigui, P. C. Zanini, and A. R. Al-Odail, "Evaluation of indoor mobile robot localization techniques," in *2012 International Conference on Computer Systems and Industrial Informatics*. IEEE, 2012, pp. 1–6.