Cooperative Vision-based Automatic Landing System for a Micro Aerial Vehicle

Martim Braga
Instituto Superior Técnico, Universidade de Lisboa, Portugal
Email: martim.gard.braga@tecnico.ulisboa.pt

***Abstract*—Nowadays the use of Unmanned Aerial Vehicles is more popular than ever from hobby to military applications. The ability of stationary hover flight makes multi-rotor UAVs particularly interesting in inspection missions. The biggest drawback however, is the limited flight time that common drones present due to the size of their batteries. In that regard an autonomous landing system could mitigate the issue of constant need of human intervention in the landing and battery charging/replacing stages. The goal of this thesis was to control a quad-rotor UAV, from free motion to autonomous landing on a mobile charging station. The landing platform is placed on an UGV that acts as the mobile charging station and processing unit of the UAV. The controller design seeks to achieve full autonomy in the landing process based on off-board sensors. To best guide the drone to the landing pad, a multi-stage controller was developed. The proposed system presents an improved accuracy and robustness in the landing process, allowing the UAV to use the battery of the UGV as a mobile battery range extender for longer inspection missions. As the UGV usually has a higher payload limit, it can have large batteries, more capable sensors and higher computational power. The developed system was tested in a simulation environment and in 50 consecutive landing cycles achieved an average time of landing of 50 s, with a precision of 2 cm from the center of the landing pad and 100% success rate.**

***Index Terms*—UAV; UGV; Quadcopter; Automatic landing; ROS programming; ArduPilot; DroneKit-Python; Computer vision.**

I. Introduction

Unmanned Aerial Vehicles (UAV) are becoming ever more popular in applications where an aerial point of view is an advantage, as in military patrol, coastal and agriculture surveillance, infrastructures inspection, etc. With such a broad array of applications the degree of required complexity can drastically differ. On one end of the spectrum there are consumer level UAVs that can be simply composed by an Inertial Measurement Unit (IMU), Global Positioning System (GPS) sensors and a camera. On the other end, we have highly specialized hardware to suit each desired application.

An UAV can be powered in two different ways: tethered, where it is connected via cable in order to get power; or by on-board battery supplies. The main disadvantage of the battery powered UAV is the UAV's range. Although not so relevant in large UAVs, the smaller ones usually have a weight constraint that implies the use of small capacity and very light batteries, which then results in a very limited range. For each desired application the best battery size/weight can be calculated [1] [2], in order to get the optimal desired range and endurance.

Developing an autonomous flying and landing system can largely improve productivity and flexibility to perform the desired task, minimizing or completely removing human intervention. Concerning the limited flight time of a small UAV, a possible solution is to develop an automatic landing system to recharge the UAV's battery on a charging station. This would enable it to do the designated task without the need of the technical support to replace or re-charge the batteries.

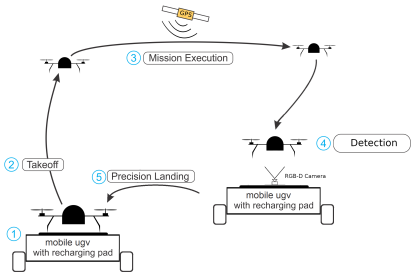


Fig. 1. Scheme of a typical inspection mission loop.

To further enhance the capabilities and versatility of an UAV, working cooperatively with Unmanned Ground Vehicles (UGV) can reduce the time to complete the task, or make it significantly more efficient. As the UGV does not have a payload limit, it can have large batteries, more capable sensors and higher computational power. For example, in solar panel field inspections, the cooperative work between the two vehicles further optimises the inspection process by using an UGV to work as a mobile charging station. This latter application of UAV and UGV cooperation allows the team, in a field inspection mission, to complete its mission faster [3]. It also increases the inspection range of the UAV given that it is no longer dependent on a fixed charging station.

A typical inspection mission can be divided in 5 main steps as seen in Figure 1. This work will focus on the detection and precision landing parts of the mission, that includes precisely detect, guide to the landing location and land the aircraft.

II. State of the Art

In order for an UAV to perform a succesfull landing on a small platform, a high positioning accuracy is required. Despite the GPS being sufficient for navigation in wide open spaces, its accuracy is, on average, of 5 m, which is insufficient to solve the UAV landing problem.

There are two main approaches to solve the localization problem in the UAV landing procedure. One possible solution is to have the main sensors to aid with the landing on the drone itself while the other one is to have them close to the landing

area. A common way to solve the localization problem of the drone relative to the landing platform is through computer vision. Compared with typical sensors, a camera is lighter and can provide rich information about the UAV self-motion and external environment. The following sub-sections focus on the key points related to the scope of this work.

A. On-board

A low cost approach was developed in [4], where by using a Nintendo Wii remote infrared camera, the drone can detect points on the platform. These points are infrared LED's, that are positioned in a pattern so that the flight controller can estimate the position and orientation of the UAV once the infrared LED pattern is identified.

In [5], two different computer vision algorithms were used in parallel in order to achieve a more robust landing system. Firstly a red blob tracking was implemented to get an estimate of the position of the UAV relative to the platform. In parallel to this, a corner tracking algorithm was implemented in order to get a robust motion tracking for when the target platform is not inside the Field of View (FOV) of the camera.

Another approach was used by [6], where a pattern was designed to overcome the limitations of other models where the target is only visible close to the landing pad. A set of concentric white rings was drawn on a black background. Although the pose estimation is very good using this technique, no successful landing was performed.

B. On-ground

Unlike the on-board sensors approach, this technique allows the use of high resolution cameras alongside with high computer processing power. In [7] a triple camera system was used. By extracting key features of the images, a robust 3D positioning and orientation were obtained.

A stereo camera system was implemented in [8] and [9]. This technique relies not only on an RGB camera but also on a 3D point cloud. Another stereo camera system was developed in [10] using a couple of Bumblebee cameras to detect the UAV and estimate its position. This was done with the usage of a CAMSHIFT algorithm to track the UAV in the image sequence. From a height of 6 m the pose estimation had a 2.5 times higher accuracy compared to that of the GPS.

An infra-red stereo camera was also used in [11]; in this application the UAV's could be successfully tracked and positioned. In [12] a stereo camera was used to estimate a 3D position of a quadrotor and achieve an autonomous hovering and automatic landing system. This system was capable of locating the MAV while flying at 6 m of altitude, being 3 times more accurate in longitudinal and lateral position estimation than the GPS.

C. Flight Controllers

Proportional-Integral-Derivative (PID) controllers are the most commonly used for a feedback linear control system. A Proportional-Derivative Controller design was presented by Erginer and Altug[13]. The developed controller was designed to control the UAV's altitude by controlling the pitch, roll and yaw angles.

To develop a flight and landing controller, a feedback linearisation controller was developed to autonomously land a quadrotor UAV onto a moving platform [14].

Back stepping nonlinear controls can also be used when designing a landing controller. This type of controller offers a recursive way of design that divides the system into subsystems. Such a system was used to land a rotary UAV using a tether in [15] and [16].

Neural networks can be incorporated into a controller design. A neural network has the ability of learning, so that given a set of observations and a class of functions, the network can learn what function will solve a given problem in an optimal way. An intelligent auto-landing controller was designed to enhance the landing safety in Variable wind conditions [17].

PID Controller: As previously stated, PID controllers are one of the most common forms of feedback controllers. With this type of controller that has a control variable, an actuator to allow the system to act on the control variable, and a form to read the state of the system, an accurate and responsive correction is made in order to stabilize the system in the desired set point. The output $u(t)$ of a PID controller can be expressed by the following equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt} \tag{1}$$

Where $u(t)$ is the output of the controller at the time t, $e(t)$ is the value of the error at that same instance, and K_p, K_i and K_d the gains for each part of the controller. In this application there are 3 variables to be controlled, X, Y and Z, so there is a PID controller for each control variable.

As seen in equation 1 the proportional part of the controller has a gain of K_p, and is responsible for the immediate corrections of the system. K_i is the gain for the integral component of the controller, which is responsible for the correction of steady state errors. This type of errors must be identified over time so instantaneous corrections cannot detect and correct. The derivative part has a gain of K_d, which acts like a damper slowing the state's approach to the set point.

D. Robot Development Tools

In order to develop the proposed system, it is necessary to choose a platform or framework capable of integrating the different types of hardware and software solutions. It is important that this platform aids in the development of the solution as well, by running already developed packages and libraries. ROS was created and developed by Standford University and later by Willow Garage in order to create a open-ended collaboration framework that would be easy to use and develop on. Nowadays ROS is currently the most used platform in robotics. By being open source, hardware and software support is ever growing due to the ease of adding support for new hardware, adapt existing drivers to the desired new hardware or add/modify packages for navigation, computer vision, mapping and control systems.

ROS is compatible with Gazebo physics simulator allowing to simulate several different systems like arm manipulators, UAV's, cameras, mobile robots, etc. ROS was chosen to be the best platform to develop the desired system on, since it has the best hardware/software support.

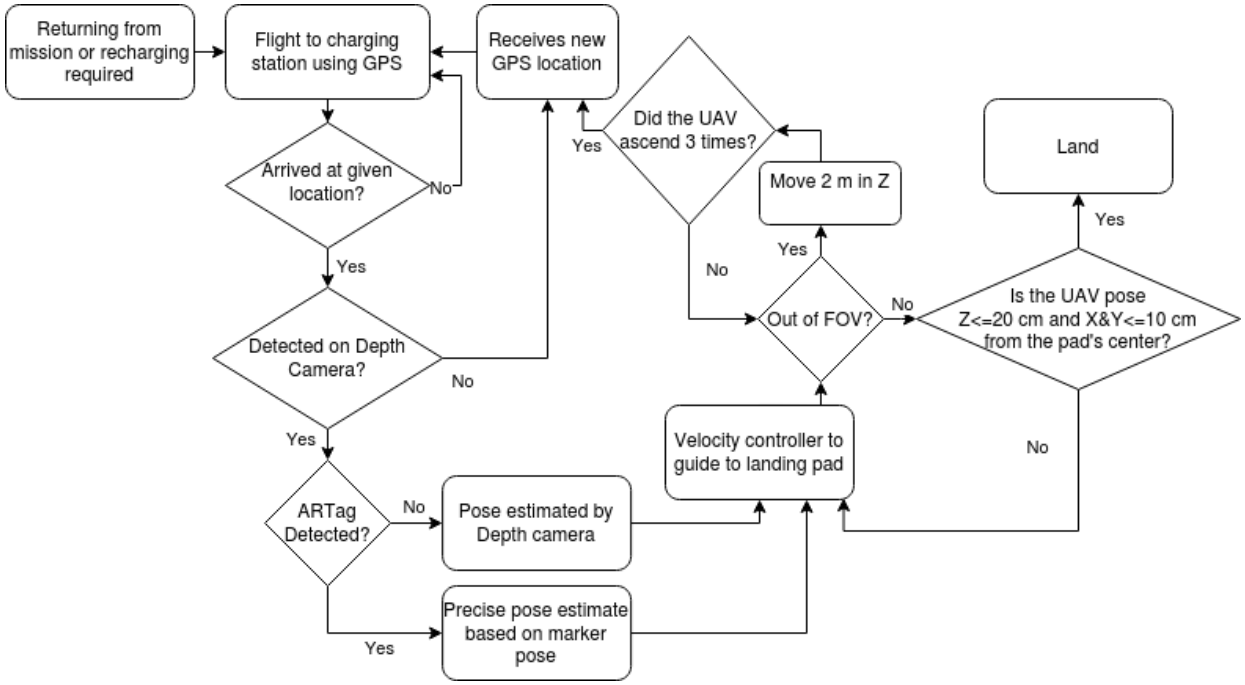


Fig. 2. Flowchart of the proposed system.

III. The Implementation

The major problem to implement an autonomous landing system with the most of the sensors located near the platform is to precisely locate the UAV. As previously stated, GPS accuracy is insufficient to guide the drone to the base, so in order to accomplish a robust landing system other methods had to be developed.

The system starts after the UAV completes its mission and needs to recharge as presented on the left side of the flowchart of Fig. 2. At this stage the drone will be flying using GPS, so in the first stage the ground station will receive a signal to start the retrieval procedure of the aircraft to recharge its battery. The GPS coordinates of the base are sent to the UAV and handled by its autopilot system. With the autopilot system the drone flies to the desired coordinates. If for some reason the drone gets out of the camera's FOV before it lands, a recovery action takes place. It raises 2 m in an attempt to return to where the depth camera's FOV is wider so it can be detected once again. Usually, in the experiments that were made, it is enough to get back to the FOV and be detected again but if it is not, then it raises another two times checking if it's out of the FOV. If raising three times 2 m is not enough to get detected, than the drone receives a new GPS point to go to and starts the landing cycle again.

In order to reduce pose uncertainty, a visual marker was added to the underside of the drone. When it arrives to the given location it switches to the vision based controller. The vision based controller will receive the relative position of the UAV and feed it to a PID controller to guide it closer to the platform. This relative position is the estimated position of the marker or, in case of the marker not being detected, the position calculated by the depth camera is used in its place.

IV. Methodologies

The landing platform will be placed on an UGV that will act as the ground station of the system. In order to reduce the payload on the UAV, and also allow for the use of a better sensor along with better computational power, a stereo RGB-D camera was placed at the centre of the landing platform to guide the drone to its target.

A. Detection

The detection system is composed of two sub-systems. In the first stage, the system uses a depth-based pose estimation. In the second stage, it uses an ARTag marker to estimate the drone's pose. The second and more precise pose estimation method is used whenever the marker is detected.

1) RGB-D: To reduce the drone's pose inaccuracy, once close to the landing location a RGB-D camera was used to detect the drone and get a more accurate pose estimation. To achive this, the sensor was positioned in the centre of the platform. By doing this, the best possible FOV is achieved and also the ideal position to center the landing drone on the landing platform. To convert the depth measurements of the camera to a 3D coordinate system, rs2_deproject_pixel_to_point was used. In figure 3 the point-cloud of the depth camera can be seen, where each white dot represents a depth measurement value for a given pixel. The system detects the closest point to the camera and converts the depth measure into (X, Y, Z), giving the relative position of the drone in relation to the camera. So, to run this detection method an empty open environment is assumed.

2) Fiducial Markers: Fiducial markers have been used in the recent past in computer vision for being computational cheap and very precise in relative pose estimation. These

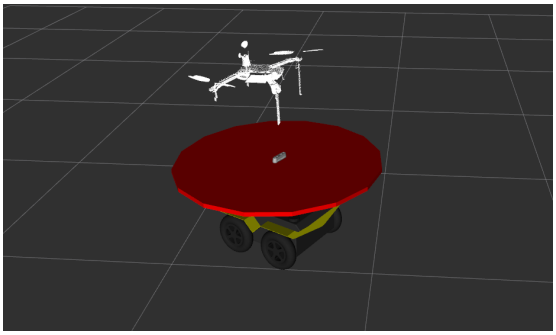


Fig. 3. Projection of depth camera's point cloud.

Fig. 4. Iris drone model with ARTag.

markers of known pattern and size, when attached to an object, can serve as a reference point of location, orientation and scale.

A typical fiducial marker is very easy to manufacture as it can be printed and glued to any surface that needs to be identified. This makes this marker a good solution to precisely locate the drone in the air once close to the platform, by sticking a marker to the underside of the UAV, as seen in figure 4. The chosen system was ARTag tracking library. ARTag is short for Augmented Reality Tag, and consists in a dual tone square symbol with a solid black background and a 6x6 grid of high contrast interior cells like the one shown in figure 5.

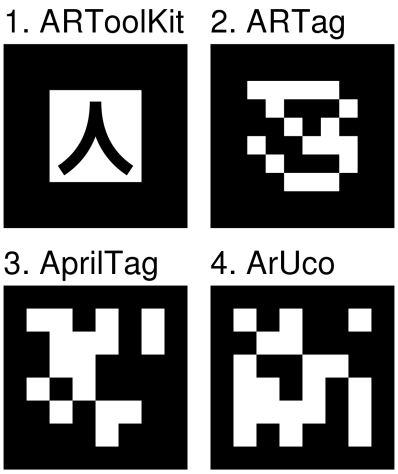


Fig. 5. Different types of AR markers. [18]

The library used to identify these markers was ar_track_alvar, which is an evolution of the standard ARTollkit [19]. One of the biggest challenges in detecting this type of markers is having the done approaching in variable light conditions. To overcome this difficulty, Alvar uses adaptive thresholding, which consists in calculating a threshold value for each pixel using the values of the surrounding pixels; and for a more stable pose estimations, it uses an optical flow based tracking system. In order to find a marker, the detection algorithm uses an edge based approach. The edge pixels are found and used to link them into segments, which once grouped form quadrangles. The corners of these quadrangles are used to calculate a homography that allows the marker to be transfered into the world frame. This approach makes it more robust in changing light conditions. In the simulated world, the marker was identified from around 2 m high.

B. Development Environment

1) Ardupilot: Ardupilot is an open source autopilot system that supports a wide range of different vehicles, from multi-copters, helicopters and fixed-wing aircrafts, to rovers, boats and submarines [20]. This software is very popular and has a large user base which provides quick bug fixes and testing. The manufacturer of the selected drone, 3DR's Iris, also supports Ardupilot, which was one of the reasons why this software was chosen for this work.

There are two main ways to control the UAV in Ardupilot; one of them is to use a conventional Radio Controller, and the other one is to use a ground control station. In order to communicate with the UAV, the ground control station uses MAVLink, which is a lightweight communication protocol. To enable compatibility with other software, MAVLink has other libraries such as MAVproxy, which is a command-line ground station software. This communication works in both ways, as the UAV sends information about its current state such as GPS, local position, IMU data, battery state and much more, and the ground station sends out control commands.

ArduPilot also offers the possibility to run over Software In The Loop, allowing it to run without any hardware. In this mode the sensor data comes from a flight simulator dynamic model. A big advantage of this system is that a flight simulation, with no hardware involved, is a fast and cheap way to test systems without the added risk of damaging physical components caused by a fault in the system, resulting in a potential crash and damage of expensive hardware.

There are various flying modes for the autopilot, each mode with a goal in mind; the modes that are used here are the guided mode and land mode. The GUIDED mode is meant to be used to dynamically guide a UAV to a desired location wirelessly, using a telemetry radio module and ground station application. The LAND mode is used to descend the drone until it lands, and shut down the motors once it has landed. In this project this mode is only used in the last stage of the landing process when it is less than 0.2 m high above the landing platform.

2) DroneKit: DroneKit is a cross-platform API that allows the development of Python applications for drones that

communicate over MAVLink, using a low-latency link, with an Ardupilot flight controller [21]. These applications run on the MAV's companion computer and further increase the capabilities of the autopilot software by performing computationally heavy tasks that need a low-latency link, as are computer vision or path planning algorithms. The API offers a methodical access to the connected vehicle's state and parameter information and also enables not only direct control over the vehicle's movement and operation, but also mission management. DroneKit also provides a comprehensive set of functions and classes that make MAVLink commands easier to understand.

In this project the main use of this development platform consisted in handling the data concerning the positions where the drone needs to be, and forwarding it on to the ArduPilot flight controller. This data is transfered using MAVLink, so to start the program a connection must be established between the autopilot and the DroneKit application. This allows the UGV to be run as a mobile Ground Control Station.

C. Gazebo Simulator

Gazebo is an open source simulator and physics engine [22]. It offers a very realistic 3D world with high-quality graphics that allow the users to simulate their systems in an environment that is as close to the reality as possible. The simulator includes interfaces for a wide range of simulated sensors out of the box, and with an ever growing support community, new hardware is added to the list of simulated sensors every day.

The simulated world and robot models can be designed to suit each desired application, which allows it to accurately simulate in complex environments. Plug-ins can also be developed to enable Gazebo to further improve its capabilities. There are other physics engines available but the Gazebo's already developed compatibility with Ardupilot's SITL and ROS, makes it a perfect fit for this project.

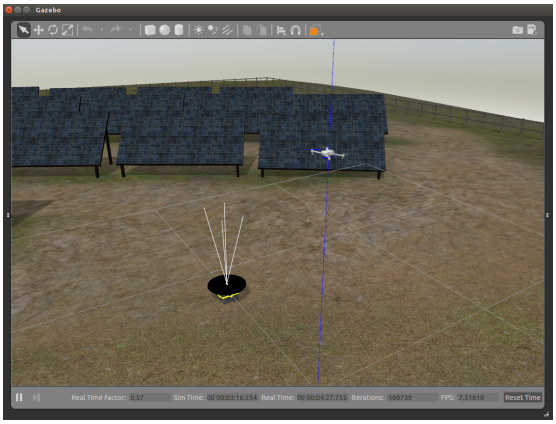
In order to run the developed system some modifications were made to the robot models in Gazebo. In the Jackal model, as shown in figure 6, a round landing platform with a stereo camera in the centre was added. In the Iris model, as seen earlier in figure 4, an ARTag was added to the underside of the drone.

V. Results

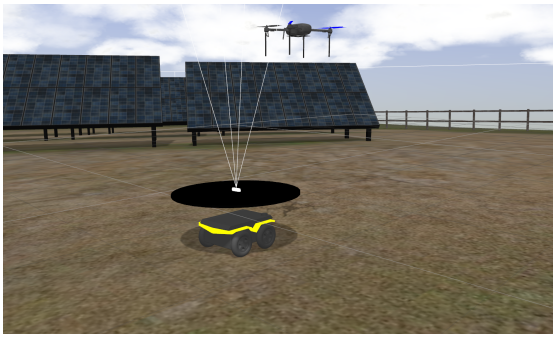
A. PID controller

The job of the velocity controller is to place the drone above the center of the platform and make it descend until it lands. It achieves this by the use of three PID controllers, one for each coordinate, as previously stated.

The initial gains of the PID controller caused the system to have a slow time of correction, and close to the setpoint it oscillated, so some tuning of the controller gains were required. Every gain has to be tuned in relation to the others, for the purpose of achieving a stable system, given that by increasing one gain to get a faster response can cause the system to overshoot and become unstable. With this in mind the gains were increased in small increments, checking the system response and adjusting the other gains to compensate and achieve a stable system.



(a)



(b)

Fig. 6. (a) Gazebo world. b) Modified Jackal Model.

	K_p	K_i	K_d
Initial	0.042	0.046	0.025
Tuned	0.27	0.0107	0.027

TABLE I
PID gain values for X and Y

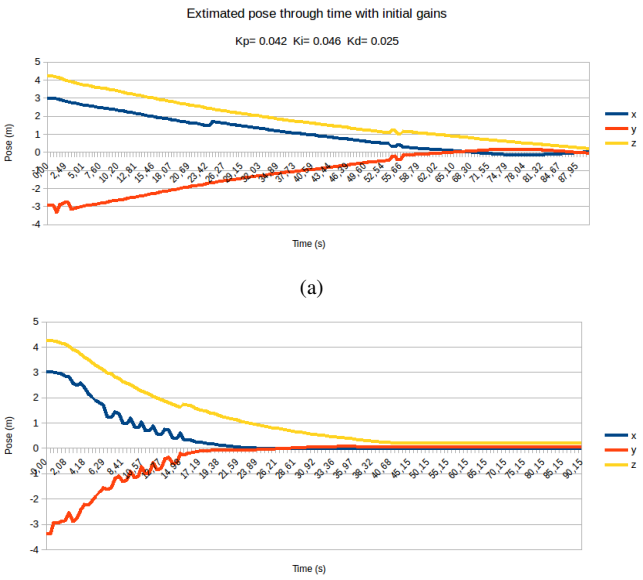
Table I presents the initial and the implemented gain values which were achieved after tunning the system. In Z only the proportional gain was changed to increase the rate of descent.

Figure 7 shows the difference in performance between the tuned controller and the initial controller performing a landing routine. To test this, the same initial conditions were used in order to get comparable results. The drone would start from 5 m height and from 3 m away in the x an y axis and approach the platform to land. At t=0 the approach starts and the end of the chart shows the moment when the drone lands.

A great improvement can be observed, as with the tuned gains the drone centers above the platform 3.4 times faster and lands 2.2 times faster. The jitter in the pose estimation can be justified by the switching between detection methods.

To validate that the obtained system is stable, a hovering routine was made where the drone started at the same point as the landing routine but with the setpoint of Z at 1 m height, forcing the drone to remain above the platform and trying to center at x=0 and y=0.

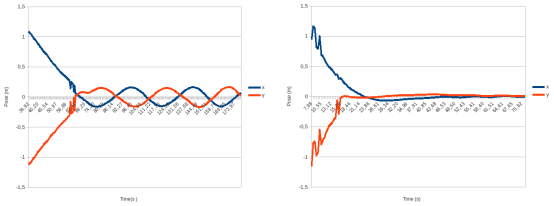
In Figure 8, it is clear that a much more stable system was achieved. Once the drone is above the platform with the tuned gains, it remains within 2 cm of the center, while the system with the initial gains oscillates around the setpoint with



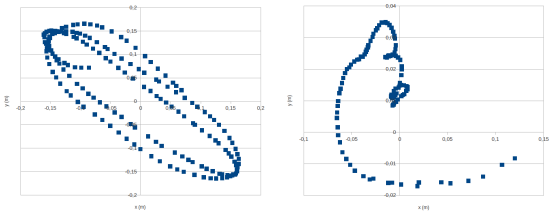
(a)

(b)

Fig. 7. (a) Plot of position over time of landing routine with initial gains (b) Plot of position over time of landing routine with tuned gains



(a) (b)



(c) (d)

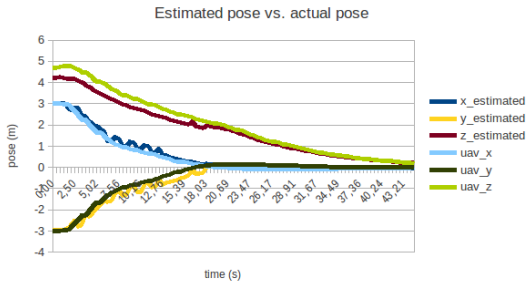
Fig. 8. (a) Plot of position over time with initial gains (b) Plot of position over time with tuned gains (c) Scatter plot of X and Y with initial gains (d) Scatter plot of X and Y with tuned gains

amplitudes in excess of 15 cm of the center.

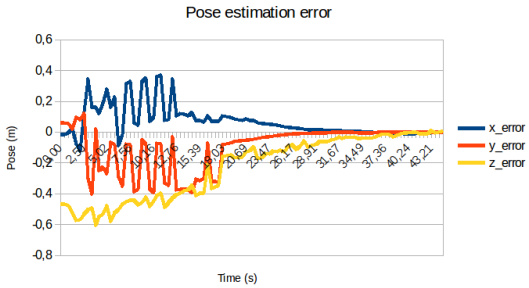
B. Pose estimation

To validate the chosen pose estimation method, the same landing routine was used. In Figure 9 the error between the detected position and the exact position of the drone is shown. In the beginning, where the error is larger, the depth camera positioning is used, but once the ARTag is detected, the position estimation becomes much more accurate as seen in table III.

To test if the system was able to land the drone successfully without the ARTag pose estimation some attempts to land the drone were made. In these test the drone landed successfully, but in some tests one of the legs landed outside of the landing



(a)



(b)

Fig. 9. (a) Plot of position over time of landing routine with initial gains (b) Plot of position over time of landing routine with tuned gains

	Error	X (m)	Y (m)	Z (m)
Depth camera pose	Average	0.145	0.219	0.45
	Maximum	0.370	0.398	0.605
ARTag pose	Average	0.027	0.015	0.068
	Maximum	0.107	0.826	0.172

TABLE II
COMPARISON BETWEEN ESTIMATION ERROR BETWEEN THE TWO METHODS

pad, but not enough to crash the drone. So we can see the even without the more precise detection method the drone was able to land successfully. This is helped by the fact that the landing platform is 50 cm in radius and Iris has a dimension of 55cm motor to motor, so what was observed was that this extra 5 cm plus the centering error was what landed outside the platform. If the platform would to be smaller the landing without the ARTag positioning possibly would not be successful.

C. Landing

To validate the landing system, a simulated launch and retrieve mission was simulated. The drone starts the flight from where it previously landed.

The drone takes off to a height of 5 m and then proceeds to go to a given GPS location away from the landing base. When it arrives to the given point it receives the GPS coordinates of the land pad. After this, the landing sequence is as previously described in figure 2.

Average	X (m)	Y (m)	Land Time(s)	Time σ	Success
Drone pose	0.022	0.024	49.79	6.524	100%
Estimated	0.018	0.018			

TABLE III
REPETITIVE LANDING STATISTICS

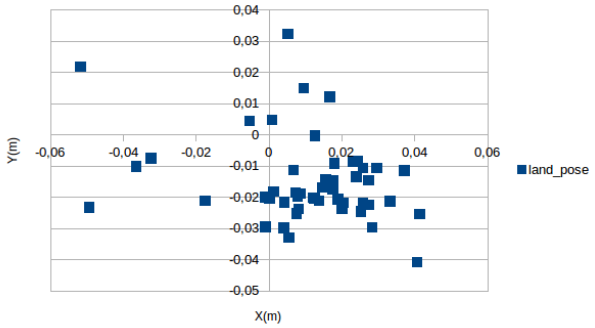


Fig. 10. Scatter plot of landing position over 50 landings

Average	X (m)	Y (m)	Land Time(s)	$Time\sigma$	Success
Drone pose	0,022	0,022	65,91	10,20	100%

TABLE IV
REPETITIVE LANDING ON MOBILE PLATFORM STATISTICS

50 landings were performed and the corresponding statistics are presented in table III.There it can be seen that the drone lands on average within 3 cm of the center of the base. It also can be seen that it takes on average about 50 seconds to land, counting from the moment when it's detected by the depth camera. All 50 landings were successful. In figure 10, the landing position variation relative to the center of the landing pad is shown.

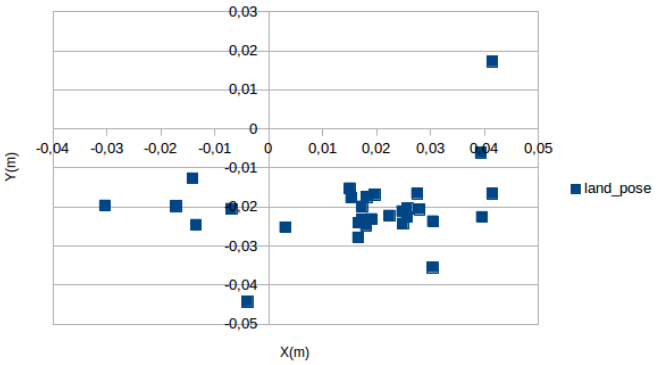
To simulate a mobile platform another mission scenario was simulated. The drone has the same routine as before where takes off to a height of 5 m and than proceeds to go to a given GPS location away from the landing base. Then receives the GPS coordinates of the landing pad but this time the landing pad moves randomly from place to place while the drone is executing the task. To test this 30 landings were done, and the respective statistics are presented in table IV.

In table IV a higher standard deviation and average landing time can be seen, comparing to the static landing platform this is due to, in some landing attempts, the base was still moving, so this resulted in a higher time. Also in some cases the drone was transported in the platform from one place to the other, this way simulating a mobile charging station. In figure 11 in image (a) it can be seen that the drone land very accurately within a radius of 4 cm of the center of the base. In the image (b) the landing pad position in the field can be seen for the 30 landings.

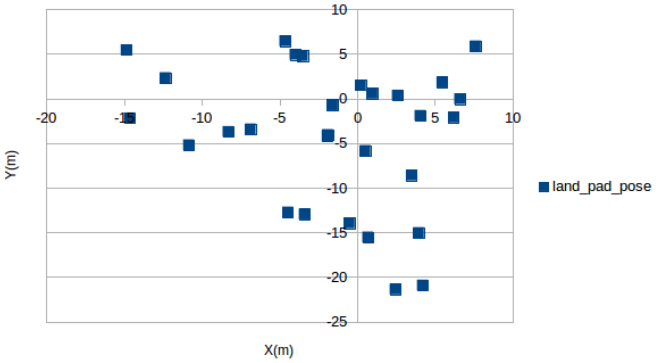
VI. Discussion and Conclusion

This work achieved the proposed goals. As seen in the previous section, the developed system is able to accurately detect, guide the drone to the base and make it land. As the designed system was developed and tested using Ardupilot, it can theoretically work with a wide range of MAVLink compatible devices and can easily be implemented in real hardware.

The results that were obtained after testing the systems in simulation were very good. Achieving a success rate of 100% in various tests and landing relatively quickly.



(a)



(b)

Fig. 11. (a) Scatter plot of the final landing pose in relation the the center of the landing pad. (b) Scatter plot of landing base position for each landing

A. Future Work

The most immediate future work would be to test and validate the proposed system on physical hardware. The depth camera detection algorithm is not a perfect solution as it assumes an empty open environment to estimate the pose of the drone. A possible solution would be to train a convolution neural network or use other image processing algorithms to be able to detect the drone in the air and get a depth measurement from the area identified as a drone. This way have a more robust detection system.

References

[1] M. Biczyski, R. Sehab, J. F. Whidborne, G. Krebs, and P. Luk, "Multirotor sizing methodology with flight time estimation," *Journal of Advanced Transportation*, vol. 2020, 2020.

[2] L. W. Traub, "Optimal battery weight fraction for maximum aircraft range and endurance," *Journal of Aircraft*, vol. 53, no. 4, pp. 1177–1179, 2016.

[3] K. Yu, A. K. Budhiraja, and P. Tokekar, "Algorithms for routing of unmanned aerial vehicles with mobile recharging stations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5720–5725.

[4] K. Wenzel, P. Rosset, and A. Zell, "Low-cost visual tracking of a landing place and hovering flight control with a microcontroller," *Journal of Intelligent and Robotic Systems*, vol. 57, pp. 297–311, Jan. 2010. DOI: 10.1007/s10846-009-9355-5.

[5] H. W. Ho and Q. Chu, "Automatic landing system of a quadrotor uav using visual servoing," Apr. 2013.

[6] S. Lange, N. Sünderhauf, and P. Protzel, "Autonomous landing for a multirotor uav using vision," *In Workshop Proceedings of SIMPAR 2008 Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots*, Jan. 2008.

[7] C. Martínez, P. Campoy, I. Mondragón, and M. A. Olivares-Méndez, "Trinocular ground system to control uavs," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3361–3367.

[8] Ma, Yan, *Coordinated landing and mapping with aerial and ground vehicle teams*, 2012. [Online]. Available: http://hdl.handle.net/10012/6993.

[9] A. Bachrach, S. Prentice, R. He, *et al.*, "Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1320–1343, 2012.

[10] D. Pebrianti, F. Kendoul, S. Azrad, W. Wang, and K. Nonami, "Autonomous hovering and landing of a quad-rotor micro aerial vehicle by means of on ground stereo vision system," *Journal of System Design and Dynamics*, vol. 4, pp. 269–284, Jan. 2010. DOI: 10.1299/jsdd.4.269.

[11] W. Kong, D. Zhang, X. Wang, Z. Xian, and J. Zhang, "Autonomous landing of an uav with a ground-based actuated infrared stereo vision system," Nov. 2013, pp. 2963–2970. DOI: 10.1109/IROS.2013.6696776.

[12] D. Pebrianti, F. Kendoul, S. Azrad, W. Wang, and K. Nonami, "Autonomous hovering and landing of a quad-rotor micro aerial vehicle by means of on ground stereo vision system," *Journal of System Design and Dynamics*, vol. 4, no. 2, pp. 269–284, 2010.

[13] B. Erginer and E. Altug, "Modeling and pd control of a quadrotor vtol vehicle," in *2007 IEEE Intelligent Vehicles Symposium*, 2007, pp. 894–899.

[14] H. Voos and B. Nourghassemi, "Nonlinear control of stabilized flight and landing control for quadrotor uavs," in *7th Workshop on Advanced Control and Diagnosis ACD 2009, Zielo Gora, Poland*, 2009.

[15] B. Ahmed and H. R. Pota, "Backstepping-based landing control of a ruav using tether incorporating flapping correction dynamics," in *2008 American Control Conference*, 2008, pp. 2728–2733.

[16] B. Ahmed, H. R. Pota, and M. Garratt, "Flight control of a rotary wing uav using backstepping," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 20, no. 6, pp. 639–658, 2010.

[17] S. M. B. Malaek, N. Sadati, H. Izadi, and M. Pakmehr, "Intelligent autolanding controller design using neural networks and fuzzy logic," in *2004 5th Asian Control Conference (IEEE Cat. No.04EX904)*, vol. 1, 2004, 365–373 Vol.1.

[18] "Artag photo webpage." (), [Online]. Available: https://commons.wikimedia.org/wiki/File:Comparison_of_augmented_reality_fiducial_markers.svg.

[19] "Ar track alvar wiki ros webpage." (), [Online]. Available: http://wiki.ros.org/ar_track_alvar.

[20] "Ardupilot webpage." (), [Online]. Available: https://ardupilot.org/. (18 Oct 2021 04:00:52).

[21] "Dronekit-python webpage." (), [Online]. Available: https://dronekit-python.readthedocs.io/en/latest/about/index.html. (18 Oct 2021 04:00:52).

[22] "Gazebo simulator website." (), [Online]. Available: http://gazebosim.org/. (accessed: 14.06.2010).