UAV precision landing using image recognition and control

Bachelor Thesis

In partial fulfillment of the requirements for the degree

"Bachelor of Science in Engineering"

Study program:

Mechatronics - Electrical Engineering

Management Center Innsbruck

Supervisors:

Dr. Eric Coyle

Assessor:

Dr. Daniel McGuiness

Author:

Mathis Abe

2110602025

Declaration in Lieu of Oath

„I hereby declare, under oath, that this bachelor thesis has been my independent work and has not been aided with any prohibited means. I declare, to the best of my knowledge and belief, that all passages taken from published and unpublished sources or docments have been reproduced whether as original, slightlychanged or in thought, have been mentioned as such at the corresponding placesof the thesis, by citation, where the extend of the original quotes is indicated.“

Place, Date

Signature

Acknowledgement

I would like to express my sincere gratitude to the RobotX team at Embry-Riddle Aeronautical University. Dr. Eric Coyle provided invaluable support with his expertise and took the time to thoroughly consider many of the problems we encountered. I am also deeply thankful for the assistance I received from the team members of RobotX, particularly Sagar Sarkar and Isaac Kay, who were instrumental in discussing concepts and testing the system.

Many thanks to the Austrian Marshall Plan Foundation for their support, which enabled me to write my thesis abroad in the US. I also appreciate the help I received from Dr. Daniel McGuiness, who facilitated the thesis process from the side of the MCI.

Abstract

This thesis presents a comprehensive approach to autonomous precision landing of an Unmanned Aerial Vehicle (UAV) using image recognition and control systems. The primary objective is to enable a UAV to accurately land on a floating helipad over water, a task complicated by environmental factors such as reflections of the sun on the water and wind. The proposed solution integrates an onboard computer, a downward-facing camera, and image processing algorithms to detect and navigate to the landing pad. The system's robustness and accuracy were validated through extensive testing in various conditions, including simulations and real-world scenarios. The results indicate that the developed system is close to meeting the precision and reliability requirements necessary for competitive applications, such as the RobotX competition.

Keywords: UAV, Precision landing, Image recognition, Autonomous control, Floating helipad

Kurzfassung

In dieser Arbeit wird ein Ansatz zur autonomen Präzisionslandung eines unbemannten Luftfahrzeugs (UAV) unter Verwendung von Bilderkennungs- und Steuerungssystemen vorgestellt. Das Hauptziel ist es, ein UAV präzise auf einer in Wasser schwimmenden Platform zu landen, eine Aufgabe, die durch Umweltfaktoren wie Sonnenreflexionen auf dem Wasser und Wind erschwert wird. Die dargestellte Lösung integriert einen Bordcomputer, eine nach unten gerichtete Kamera und Bildverarbeitungsalgorithmen zur Erkennung und Navigation zum Landeplatz. Die Robustheit und Genauigkeit des Systems wurde durch umfangreiche Tests unter verschiedenen Bedingungen, einschließlich Simulationen und realen Szenarien, validiert. Die Ergebnisse zeigen, dass das entwickelte System den Anforderungen an Präzision und Zuverlässigkeit, die für Wettbewerbe wie RobotX erforderlich sind, nahezu gerecht wird.

Schlüsselwörter: UAV, Präzisionslandung, Bilderkennung, Autonome Steuerung, Schwimmender Landeplatz

Contents

1 Introduction

1.1 Motivation

In recent years, UAV technology has become cheaper and more accessible for consumers and companies. This leads to a more widespread usage with ever more demanding tasks such as mapping, surveillance, or delivery. The need for fully autonomous operations is also increasing with a growing number of systems being deployed. One critical factor here is the ability to land on a designated target with accuracy and reliability. Traditionally, autonomous landings are often only purely guided by GPS. However, this becomes problematic in environments with poor GPS coverage or if the target position is not very precisely known.

This is where image recognition can greatly improve the success rates, as the GPS coordinates then only have to be accurate enough to view the target. Furthermore, a purely GPS-guided landing is nearly always inferior compared to visual guidance in terms of accuracy. In addition, if the target is moving, a continuous update of the position instead of a single precise coordinate is needed.

Figure 1.1: Descent of the UAV on land.

One application where precision landing using image recognition is often implemented is the autonomous landing and recharging for UAVs [1]. One benefit of automating these processes lies in the reduction of human labor. Furthermore, as many ships are nowadays equipped with UAVs to help with reconnaissance or to aid in dangerous missions, landing those on floating platforms over water is ever more important.

Developing a reliable system to solve these tasks comes with various technical difficulties. A fast and accurate image recognition and tracking system is needed in order to locate the landing spot relative to the UAV. Furthermore, a robust controller is required in order to use the location of the landing pad to maneuver the drone, while the communication between the components must be consistent.

Aside from the relevance in the market of today, the task of autonomously landing on a floating platform is also part of the biennial international student competition "RobotX".

This fully autonomous contest involves both a UAV and an Unmanned Surface Vessel (USV) to work in tandem in order to complete all challenges. The Embry-Riddle Aeronautical University has been taking part in RobotX for many years. The USV and UAV used for this can be seen in Figure 1.2. The UAV performing a landing using the system discussed in this thesis is depicted in Figure 1.1.

Figure 1.2: USV with UAV on top.

1.2 Related Work

The challenge of vision-based control for UAVs can be mostly divided into two approaches: Rule-based methods and machine learning. When using rule-based methods, predefined rules for feature extraction and classification are hard-coded by the user.

For this approach, a large subset of similar work is based on the detection of AprilTags or ArUco Markers. A lot of effort has been put in to make the detection of these markers efficient and reliable, therefore providing a very robust way of detecting the position of the landing pad. However, on the controls side, the same challenges still exist.

In their research, Artur Khazetdinov et al. present an approach for precision UAV landing with the use of visual sensory data. This method utilizes ArUco markers to signal the position of a stationary landing pad. However, the system was only simulated and not tested in real-world conditions [2].

In contrast, when using machine learning, the system is trained on a dataset and learns the features and classifications by itself, often with guidance in the learning process. This is beneficial when the target does not have clear features that can be detected with relatively simple rules. However, large amounts of training data are required to adequately train the network.

Dominik Pieczyński et al. developed a deep-learning-based approach using regression to locate the position of the landing along with a metric for the uncertainty of the presence of a landing pad [3].

A combination of the rule-based approach and machine learning is used by Miguel Moreira et al. due to the use of custom features, including a triangle shape and an "H" in conjunction with a small ArUco marker. This helps achieve greater precision by using different features at various altitudes, therefore gaining good visibility at high altitudes while still maintaining precision close over the target [4].

In addition, some research focuses on landing on moving targets such as rovers or buses. Chang Wang et al. presents a vision-based deep-reinforcement-learning approach in order to land on a moving rover. Automatic curriculum learning is used to improve the success rates of landing in various conditions [5].

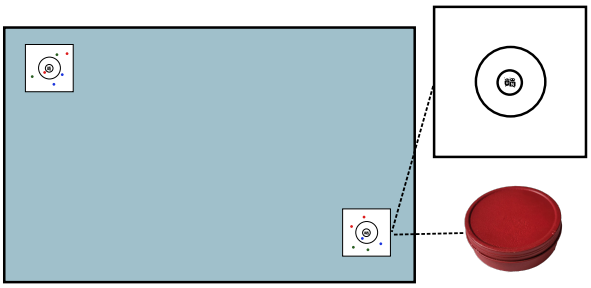
A lot of the research focuses on the detection and landing over land rather than water. However, research by Carlos Castillo et al. details an approach for landing a UAV on a moving vessel over water. This approach uses radio location to estimate the relative position [6], and therefore does not make use of image recognition as a means of guiding the UAV.

In general, not a lot of research has gone into landing on platforms over water, especially when combined with vision-based target detection. The specific task for this thesis involves targeting a slowly moving, yet primarily stationary object, complicated by a background of water, which can hinder the effectiveness of the image recognition algorithm due to visible glare. Furthermore, risk dramatically increases in comparison to testing on land, as crashing in water bears the eventuality of losing or destroying the UAV.

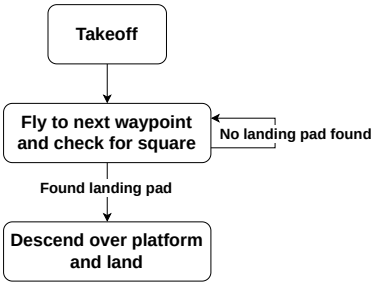
1.3 Goal

The goal of the thesis is to develop a system to autonomously detect and land a UAV on a floating helipad over water using vision-based methods. This will be used for the international Maritime RobotX competition. The goal of the competition is also to pick up and drop-off tins located on two different platforms. However, the pickup and drop-off of the tins exceeds the scope of this thesis. Waypoints with the estimated location of those landing pads are assumed to be provided by the USV. The UAV has to take off, fly to the approximated waypoints, check whether a landing pad is in view, and land on it if this is the case. Depending on the mode specified by the user, the UAV should also land on the closest tin to the center of the platform.

An outline of the task is shown in Figure 1.3. Subfigure (b) shows the procedure that is used as a high-level overview. The area in which the targets lie is specified by the event organizer. However, the USV will be able to give an approximation of the location of the individual platforms, therefore drastically simplifying the search of the objects. Multiple tins will be located on each platform.



(a) Overview of the task to be done. The floating helipads are located over water and have small colored tins on top.



(b) Diagram of the process.

Figure 1.3: Overview of the task at hand.

1.4 Organization of the Thesis

The thesis discusses the development and testing of an implementation for the autonomous detection and landing of a UAV on a floating helipad over water. After this introduction, the methodology applied to facilitate communication, controls, and image recognition is presented. Next, the results are divided into multiple tests: on a scaled-down target by holding the UAV by hand, in simulation, over land, and over water. The thesis ends with a short summary and an outlook over proposed future work.

2 Methodology

This chapter describes the methodology for precision landing on the target as described in the introduction. The approach includes a description of the UAV and its payload, communication between the different components such as flight controller and onboard computer, image processing used to identify the target, and the implemented controls.

2.1 UAV and payload

The UAV can be seen in Figure 2.1, and is custom-built based on the "Tarot Iron Man 650" frame. With the onboard computer and camera attached, the mass is 3.9 kg. Including the propeller guards, the drone has a size of 90 cm x 90 cm x 35 cm. The Pixhawk V6X running PX4 is used as a flight controller, as this is the newest model in the Pixhawk flight controller family. In order to maintain the desired attitudes, the flight controller is equipped with a compass, accelerometer, and gyroscope. Furthermore, to hold the current position, sensor fusion is used to combine the position estimation by the Internal Measurement Unit (IMU) with the GPS and barometer. The flight controller is equipped with redundant sensors and is therefore automatically able to detect sensor failures.

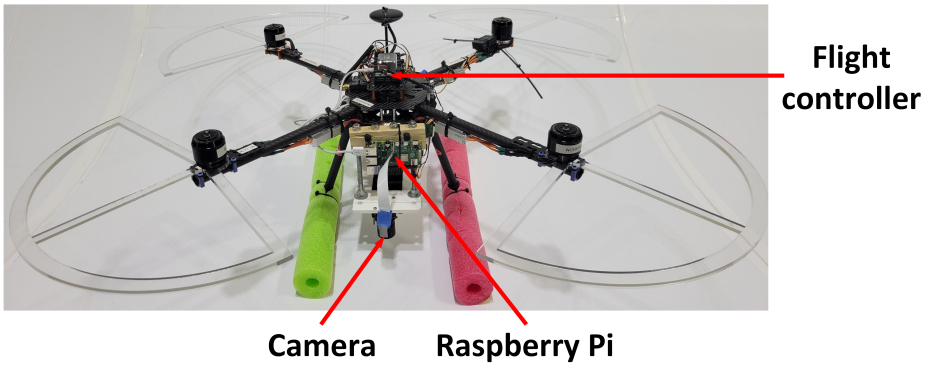


Figure 2.1: UAV with camera payload and onboard computer.

To accomplish the task of precision landing, the UAV needs to be equipped with an onboard computer to handle processing of the image and controls. A camera is also needed to capture the landing pad or tin. The camera is oriented straight down, as the UAV will be approximately level with respect to the horizon when trying to descend on the target. An onboard computer is used for planning and processing of images since

the flight controller does not have the necessary processing power. The Raspberry Pi 4B is used for this task along with a Raspberry Pi HQ camera and a wide angle lens. The Raspberry Pi is chosen as the computing platform due to its low price and relative ease of use. The choice of camera module is made because the Raspberry Pi HQ camera is natively supported by the Raspberry Pi and also has a high resolution of $12.3\,\mathrm{MP}$ along with a wide field of view with the correct lens. A wide field of view of $65\,^{\circ}$ by $52\,^{\circ}$ is required as this decreases the minimum altitude at which features of a certain size can still be tracked, thus making the landing more accurate. At an altitude of $10\,\mathrm{m}$, the area visible in the image spans $12.74\,\mathrm{m}$ by $9.75\,\mathrm{m}$. With the target size of $2\,\mathrm{m}$ by $2\,\mathrm{m}$. This is deemed large enough to be able to locate the helipad, as the initial waypoint can be off by at least $3.88\,\mathrm{m}$, while still allowing the target to be fully in view.

In order to provide ample buoyancy in the event of a crash over water, the UAV is equipped with pool noodles to guarantee the ability to recover the drone. All the pieces combined can float $4.8\,\mathrm{kg}$ leaving $0.9\,\mathrm{kg}$ of flotation after accounting for the computing and camera payload.

2.2 Communications

Figure 2.2 shows the most important systems components to this mission, as well as the way in which they are connected. The Unmanned Surface Vessel (USV) is not part of the UAV and represents the autonomous boat from which the UAV receives the estimated GPS positions of the landing platforms. The Ground Control Station (GCS) is used to monitor the UAV while testing, using 2 data links. Messages from the flight controller are received via a $915\,\mathrm{MHz}$ two-way link. This enables changing parameters on the flight controller, and also provides telemetry data. The second $2.4\,\mathrm{GHz}$ link is used to communicate with the onboard computer over a Wi-Fi network using Robot Operating System (ROS version 1) messages. This link is also used to monitor the image processing results and other messages concerned with the onboard computer.

The rest of the components represent the UAV. As illustrated, the camera is connected directly to the onboard computer, where the image processing takes place. The flight controller and the onboard computer are connected via MAVROS, which is a ROS implementation of the lightweight MAVLink messaging protocol for UAVs. This enables the onboard computer to access much of the sensor data, position estimation, and states of the flight controller. All commands sent by the onboard computer to the flight controller are also sent via this protocol.

Communication between the USV and UAV is not in the scope of this report, but will be done using ROS messages connected over a $2.4\,\mathrm{GHz}$ link, as this enables the use of Wi-Fi. This frequency also provides sufficient speed and allows for a longer range than a $5\,\mathrm{GHz}$ network.

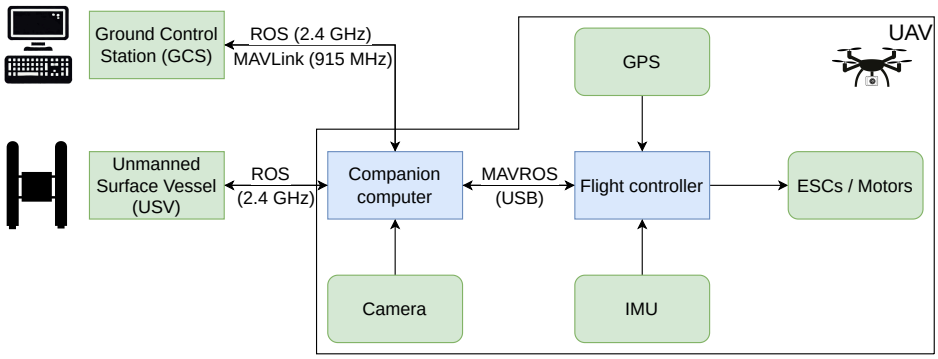


Figure 2.2: Basic components and connections on and to the UAV.

2.2.1 ROS

The communication between the onboard computer and flight controller, as well as the USV and GCS to the onboard computer is based on ROS. ROS is a framework designed to simplify communications in robotic applications. ROS relies on nodes, which are programs, and handles communication between those with topics and services. Nodes can publish a topic, and other nodes are able to subscribe to the relevant topics. This enables asynchronous communication. Services, on the other hand, are a more direct form of communication in which a request is sent, and the node waits for a response [7].

Communication between the flight controller and the onboard computer is conducted via MAVROS. MAVROS is a ROS package that enables communication with the flight controller using the MAVLink protocol. ROS is used because this allows for easier integration with the rest of the software running on the Raspberry Pi. The topics and nodes used for this code are illustrated in Figure 2.3.

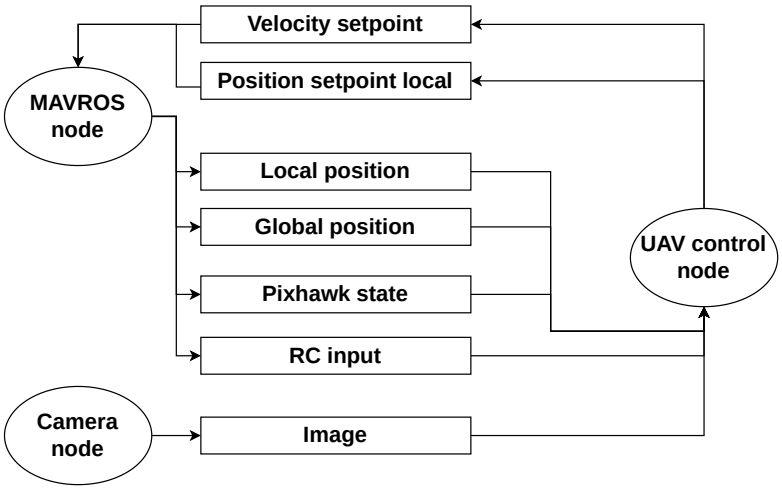


Figure 2.3: ROS nodes and topics. Nodes are round, topics are rectangular.

The "UAV control node" is the main program and receives various messages from the MAVROS node, as well as the image from the camera. The local and global positions

are used for waypoint navigation. The calculated waypoints are then published as a local position setpoint. These waypoints are required for the UAV to navigate to the estimated target positions.

The Pixhawk state is necessary, as the UAV control node needs to know and switch the flight controller state depending on the current step of the process. One such example is switching to "Loiter" if the target has been lost. This switch is done using a service call.

To ensure safe testing, the remote pilot in command needs to have the power to always kill the running script and take over control. Relying on the state for this is too slow as this is only updated at a frequency of 1 Hz. For this reason, the "UAV control node" also subscribes to the Remote Control (RC) channels coming from the transmitter. This is published at a much higher frequency of 30 Hz and enables efficient takeover of the UAV by killing the script.

2.2.2 States

As shown in Figure 2.8 (b) of the Introduction, the landing process can be divided into two subprocesses:

- Flying to a waypoint and checking for landing pad
- Descending over target

These two procedures are discussed here in more detail.

Waypoint navigation

Figure 2.4 shows the details of the process for flying to a waypoint. The UAV starts on the ground and must be armed via remote control by the operator before takeoff. The waypoint list is provided by a user during testing, but will be provided by the USV during competition. At each waypoint, the UAV stops for 3 seconds to check for any landing platforms in the image. If they are detected, the UAV flies over the closest one by setting a waypoint and then switches to the descent procedure. If none are detected, the UAV flies to the next waypoint.

Landing on helipad

There are two modes for landing on the helipad. The user can choose to either land on one of the tins or in the middle of the inner concentric circle. In both cases most steps remain the same. Figure 2.5 (a) shows the states through which the code runs during this step. In order to separate the different procedures, a state machine is used. The UAV first uses visual features of shrinking size to guide the descent down to an altitude

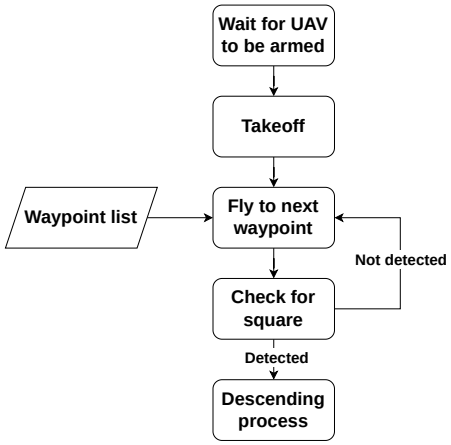


Figure 2.4: Process of flying to waypoints to search for the landing pads.

of 1.5 m (green area). These visual features are discussed in more detail in 2.5. Once an altitude of 1.5 m has been reached, the UAV actively tries to maintain this altitude above the target (blue area).

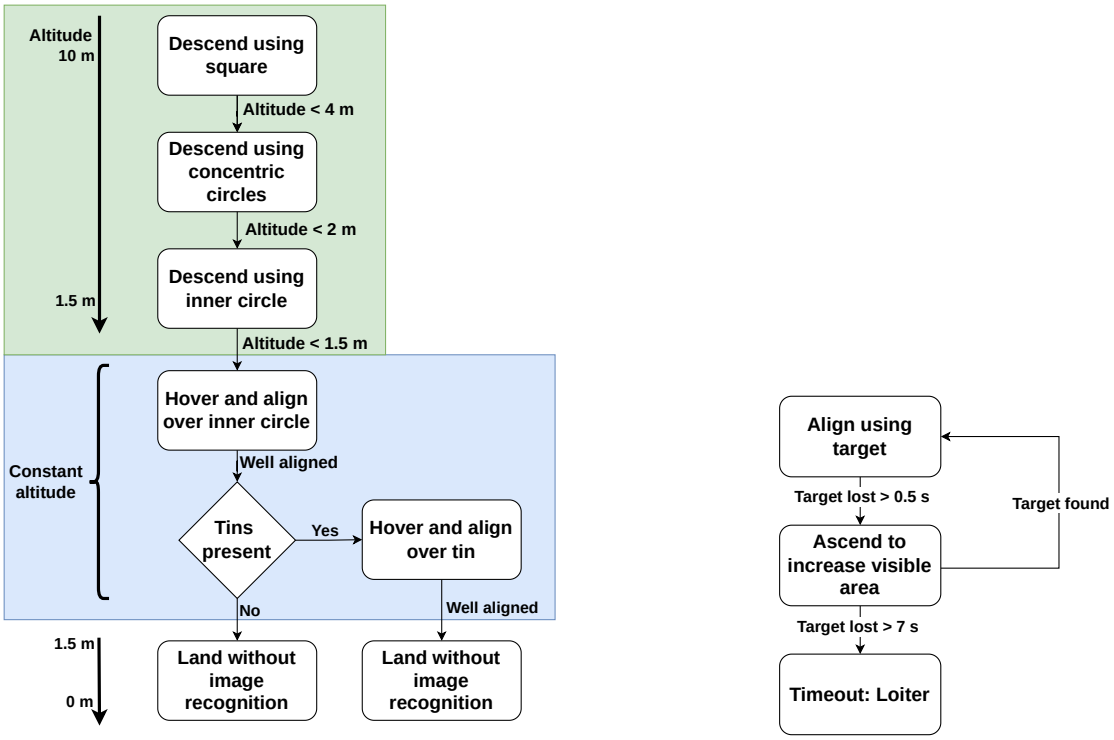
Depending on a variable set by the user, the UAV then either attempts to land on the inner circle or the closest tin relative to that inner circle. Figure 2.5 (a) illustrates the process if landing on tins is enabled. If the goal is to only land in the center of the platform, the UAV simply lands after being well aligned with the inner circle. If landing on a tin is enabled, the UAV switches the reference point to the closest tin relative to the inner circle. In the case that none are present, the UAV lands just as before only based on the inner circle.

Landing the last 1.5 m without image recognition is implemented, as the target is otherwise easily lost if the UAV is hit by a gust at low altitude. In this way the landing can also be done faster, which reduces the impact of sensor drift or wind.

If the target is lost at any step during the descent from 10 m to 1.5 m, the UAV follows the steps depicted in Figure 2.5 (b). To increase the chances that the target is in the frame, the drone first ascends at a constant speed. If the target has been lost for too long, however, the UAV loiters. This may be changed later so that the UAV flies to the next waypoint or to shore. For the purpose of simplifying testing, this has been left to loiter.

2.3 Controlling the UAV

Several modes are used to control the UAV. For flying to the correct waypoint, the desired position in the ENU (East North Up) coordinate system is sent to the flight controller. The flight controller manages the internal controls necessary to arrive at this position. The following sections will discuss the controls for descending over the target.



(a) Process of descending and landing on landing pad.

(b) Procedure when target is lost.

Figure 2.5: State diagrams for descending and landing on target.

2.3.1 Descent process

Depending on the stage of the descent process, different controls are used. When descending from $10\,\mathrm{m}$ to $1.5\,\mathrm{m}$ (Figure 2.5 (b) green area), the vertical velocity is kept constant and a controller is used to align the UAV in the horizontal plane. Once the UAV attempts to keep the altitude constant, an active altitude controller is also needed (Figure 2.5 (b) blue area).

The position estimation generated by the image recognition procedure is used as the error for the control algorithm. An overview of the controls used is illustrated in Figure 2.6. This diagram is divided into the controls implemented on the Raspberry Pi, the Pixhawk flight controller, and finally the UAV, which includes components such as ESCs, motors, propellers, the air frame and external influences.

The process for calculating the velocity setpoints in the horizontal plane is as follows: First, image processing is performed to estimate the position of the target relative to the UAV in meters. The distances in the horizontal plane are then used as the error and fed into the P controller, which outputs the horizontal velocity setpoint. This setpoint is then used by the flight controller to control the horizontal motion of the UAV. To ensure safe operation, the maximum speeds of the UAV commanded by this process are limited to a maximum of $0.5\,\mathrm{m\,s^{-1}}$.

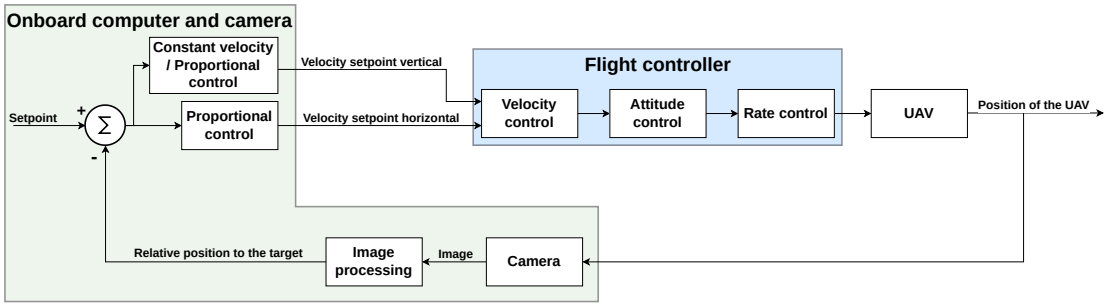


Figure 2.6: Controls implemented on the onboard computer as well as the flight controller.

The vertical controller can have one of the following states:

- Constant descend
- Constant ascend
- Proportional control

The proportional control is needed to hover at a constant altitude as the copter otherwise slowly drifts up or down even if the vertical velocity setpoint is zero. The error used for the proportional altitude controller is the altitude estimation based on the image. How this error is measured is discussed in more detail in Section 2.5.

Constant descend is used for most of the descent process illustrated in (Figure 2.5 (b) green area) except for when the target is lost. In this case, constant ascend is used to increase the visible area again.

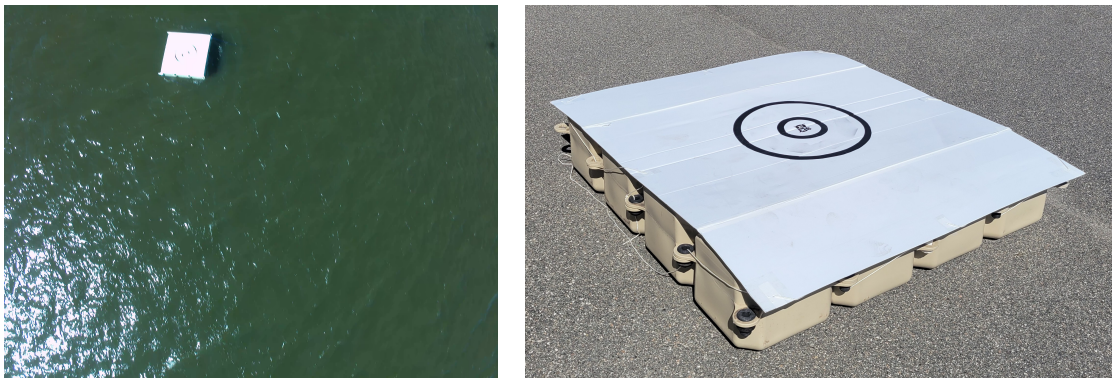
2.4 Target

The landing pad design is specified by the competition organizer. Table 2.1 lists the dimensions of various features. The background color is not clear but is assumed to be either light gray or white. It is important to note that the background color was white in the 2022 RobotX competition. The landing pad is placed on floating dock units for flotation. To be able to conduct the tests, a replica was built and can be seen in Figure 2.7 (b). Due to the size and weight of the floating dock platform, it can be assumed that the angle of the target relative to the horizon is stable in calm waters.

Table 2.1: Known dimensions of the landing pad

Measurement	Dimension
Total landing pad	2 by 2 m
Inner circle diameter	0.24 m
Outer circle diameter	0.72 m

The target has different visual features of varying sizes that are used to guide the descent. However, based on the altitude of the UAV, some features may not be detectable. This can be seen in Figure 2.7 (a), where the concentric circles are not clearly visible, which was captured at an altitude of approximately 15 m.



(a) Image taken from UAV. The circles are notably not visible.

(b) Landing pad mounted on floating dock.

Figure 2.7: Landing pad on a floating dock and image captured from flying UAV.

2.5 Image recognition

A vital and also the most challenging part of this task, is the reliable detection of the target. Special care must be taken to develop a robust algorithm that works in various lighting conditions and is unlikely to detect incorrect targets. As described earlier, because different features are visible at different altitudes, an approach must be used that decides on the correct feature to track.

Figure 2.8 provides an overview of the different features used for image processing during the descent process. The exception here is the "Land" step, which does not require any image recognition. The following sections go into more detail regrading each particular state.

For all image recognition tasks, a rule-based approach is utilized without the use of machine learning. Machine learning was not chosen due to the lack of available training data, as flying over water with a deployed helipad requires significant effort and has a long setup time. It is highly likely that over 1000 training images would be required, both over land and water in differing lighting conditions. This process would require waiting for the correct water conditions, possibly taking months. Getting approval for flying over water from the county also caused wait times of about a month. Aside from that, with the target features clearly defined, a hard-coded approach was deemed feasible. Machine learning is typically more computationally intensive, resulting in lower frequencies and complicating the controls.

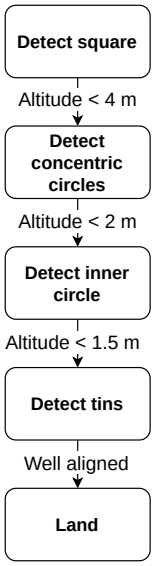


Figure 2.8: Overview of the states used for image recognition during the descending process.

2.5.1 Pinhole camera physics

As most of the transitions between states depend on the altitude, an accurate way of estimating the altitude above the target is needed. Since GPS does not have the accuracy required for this task, the altitude is calculated from known features on the target. What the current known feature is that is used as a reference depends on the current state. It is always one of the following:

- The $2\,\mathrm{m}$ by $2\,\mathrm{m}$ platform
- The $0.72\,\mathrm{m}$ diameter outer circle
- The $0.24\,\mathrm{m}$ diameter inner circle

The detection of the pixel size and location of these features is described in more detail in Sections 2.5.3, 2.5.5, and 2.5.8. Estimation of the altitude is possible because the camera can be approximated as a pinhole camera. The principles of a pinhole camera can be seen in Figure 2.9. As is obvious from the figure, the perceived size of the object in the image is dependent on the height h over the target. As the UAV descends lower, the target therefore appears bigger and details such as circles and tins are easier to identify. At a certain altitude, the features are also too large and therefore do not fit in the image. That is, the target may fill out more than the entire image.

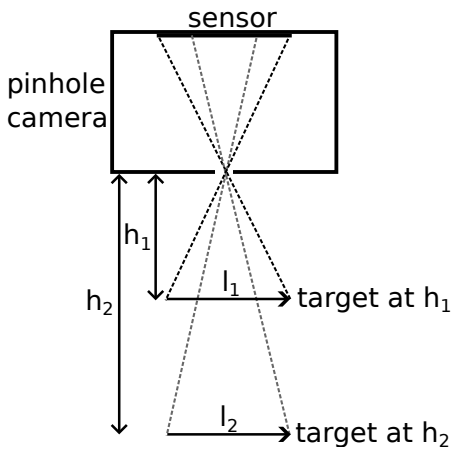


Figure 2.9: The camera can be approximated as a pinhole camera. The size of the object on the image is dependent on the height.

To calculate the altitude, the following parameters are needed:

- fov_h - horizontal field of view of the camera
- l_m - length of the object in meters
- l_{px} - length of the object in pixels
- w_{px} - total width of the image in pixels

The formula for calculating the altitude is given by equation 2.1.

$$h = \frac{\frac{l_m \cdot w_{px}}{l_{px}}}{2 \cdot \tan(\frac{fov_h}{2})} \tag{2.1}$$

This altitude estimate is then used to determine which feature is the most reliable to track at the current height. Furthermore, altitude is used to constrain the search for features to a range of sizes estimated from this measurement. This can help speed up the processing and reduce false positives.

Moreover, camera distortion can be ignored, as the target is centered most of the time. The only exception being the very start, when the target may be located close to the edge of the frame. However, the image processing algorithm is robust against distortion at this stage due to the relative ease of detecting a square.

2.5.2 Position estimation

To estimate the relative error to the target in meters, the center of the target must be first detected. What feature is used as the target depends again on the current state. It is always one of the following:

- The square platform
- The concentric circles
- The inner concentric circle
- The closest tin to the center

The next subsections discuss the procedure for detecting these features. After this is achieved, the relative position is converted from the error in pixels relative to the image center to an error in meters dependent on the height of the UAV. In order to reduce complexity, the pitch and roll angle of the UAV are not taken into account. Incorporating these measurements would introduce more noise and might thus require more advanced filtering. Some shortcomings of this method are discussed in a later section.

To smooth out any noise due to imperfect or faulty detections as well as external factors such as wind gusts, a 5-point running weighted average filter is used for both the altitude estimation and the horizontal error. The weight of this filter for each individual measurement depends on the total distance to the target d and the time difference Δt since this measurement was taken. This is done because the measurements are assumed to be more precise when the UAV is close to the target. Furthermore, a more recent measurement is assumed to be more accurate as the UAV is in motion. The final weight is therefore the product of a factor w_{distance} and a factor w_{time} as shown in equation 2.2.

$$\begin{aligned} w_{dist} &= \begin{cases} 10 - d & \text{if } 10 - d > 2 \\ 2 & \text{otherwise} \end{cases} \\ w_{time} &= e^{-\Delta t} \\ w &= w_{dist} \cdot w_{time} \end{aligned} \tag{2.2}$$

This weight is applied to each measurement, and the weighted average over all five samples is used for any further calculations.

2.5.3 Detection of square

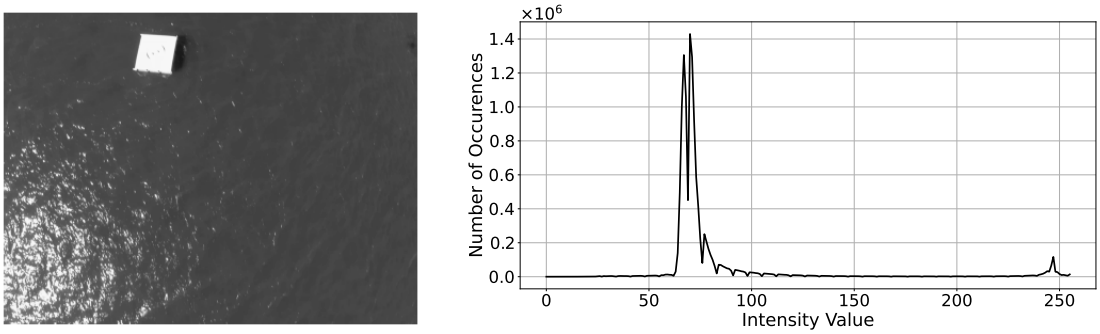
As the landing pad is bright and surrounded by darker water, square detection can be performed by looking for a bright object of approximately the right size and shape. The specifics of these conditions are discussed in Subsection 2.5.3. The method of detecting the square described here is used both for descending over the landing pad and for initially detecting the presence of a landing pad.

Bimodal image

To understand the thresholding process for detecting the square, one must first understand the approximately bimodal nature of the expected image.

In a bimodal image, the histogram shows, as the name implies, a bimodal distribution. This means that two peaks that roughly represent two normal distributions can be detected and sufficiently separated. Figure 2.10 shows this distribution of an image captured from the UAV over water. As the histogram only contains the intensity and thus brightness of the individual pixels, a grayscale image is sufficient.

However, in a typical image, the two peaks that make up the fore and background are approximately equal in size. As is obvious from Figure 2.10 (a), this is not the case here, with the peak close to 250 being much smaller in amplitude and width. This peak represents the landing pad and is much smaller due to the target occupying up much less than half of the image.



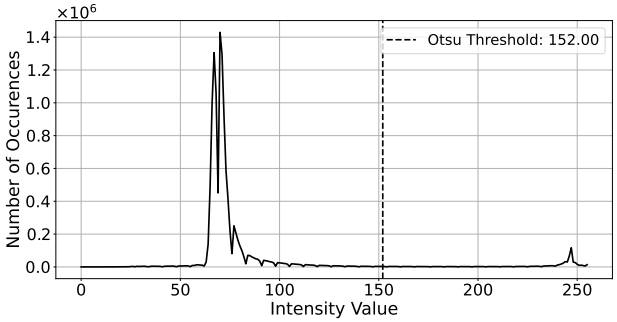
(a) Grayscale image. (b) Histogram of blurred grayscale image.

Figure 2.10: Grayscale image and corresponding histogram.

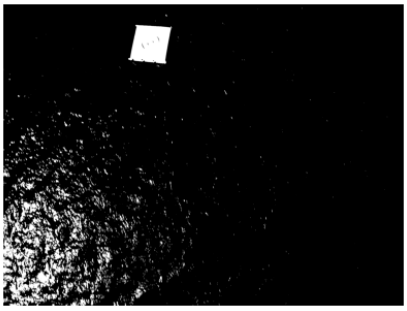
Otsu's method

The challenge of thresholding lies in finding a suitable intensity threshold that can be used to dissect the image into foreground and background. For bimodal images, Otsu's method is a popular choice. A detailed description of this algorithm can be found in [8]. Otsu's method splits the histogram into two classes, based on the intensity of the grayscale pixels. The global threshold used for this is computed by minimizing the within-class variance.

The result of using Otsu's method on the UAV images is illustrated in 2.11. The bright spots in the water caused by the sun are picked up in the thresholding process and make the reliable detection of the square more difficult.



(a) Histogram with threshold indicator.



(b) Thresholded image.

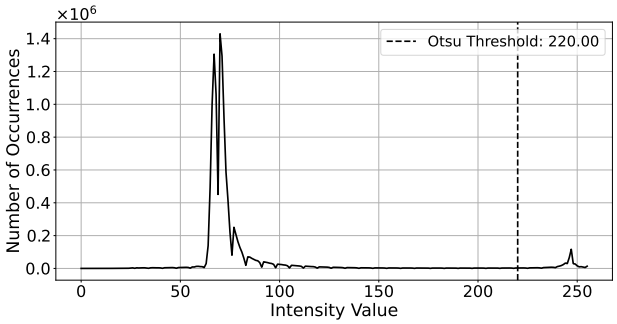
Figure 2.11: Result of using unmodified Otsu's method for thresholding.

A way of mitigating this problem is to modify Otsu's method to produce a threshold closer to the second peak caused by the platform. This can be done by adjusting the function for the within-class variance. This function is minimized to find the optimal threshold. Equation 2.3 shows the original form. By multiplying $w_1(t) \cdot \sigma_1^2(t)$ by a factor x greater than 1 the weight of the second class variance can be increased. Thus, when minimizing for $\sigma_w^2(t)$ a higher threshold will be computed that is closer to the second peak. The final modified equation is shown in 2.4.

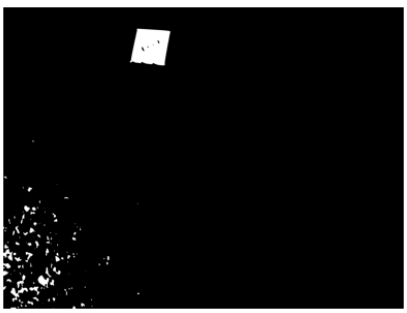
$$\sigma_w^2(t) = w_0(t) \cdot \sigma_0^2(t) + w_1(t) \cdot \sigma_1^2(t) \tag{2.3}$$

$$\sigma_w^2(t) = w_0(t) \cdot \sigma_0^2(t) + w_1(t) \cdot \sigma_1^2(t) \cdot x \tag{2.4}$$

Figure 2.12 shows the result of setting the factor $x = 50$. The noise caused by the reflection of the sun is drastically reduced and can be filtered out more easily.



(a) Histogram with threshold indicator at a higher value.



(b) Thresholded image using the different modified threshold value.

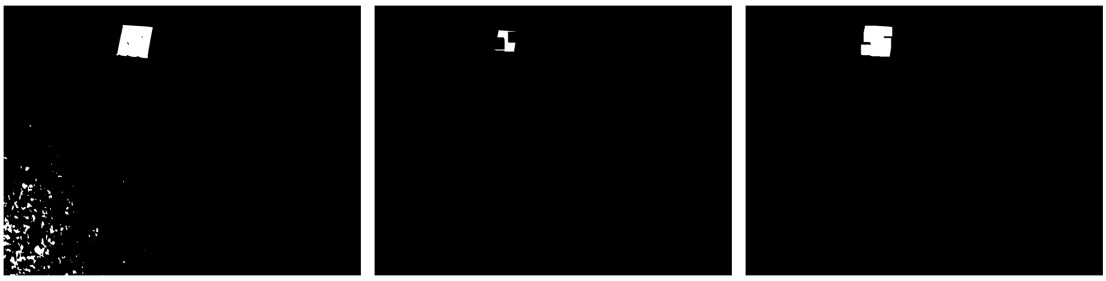
Figure 2.12: Result of using modified Otsu's method for thresholding.

Otsu's method was chosen over other alternative options due to the relative ease of implementation and the robustness compared to other thresholding techniques in terms

of changing lighting conditions. Other methods include global and adaptive Gaussian thresholding [9].

Morphological Operations

To further reduce small unwanted objects in the image, first an erosion and then a dilation is applied. Both of these operations use a kernel $\frac{1}{38}$ of the image width to compute either the local minimum for erosions or the local maximum for dilations. This local extreme is then applied to the anchor point of the kernel, typically being the center. This results in a closing operation for erosions and an opening operation for dilations. More details are found in [10]. The result of these operations can be seen in Figure 2.13.



(a) Thresholded image. (b) Result of erosion. (c) Result of dilation.

Figure 2.13: Result of morphological operations with a kernel size of $\frac{1}{38}$ of the image width.

Contour approximation

To locate the square and check if it has the right dimensions, contours must be fitted to the image, after the morphological operations have been performed. The contours are then checked for:

- The approximately correct size of the area
- Having four corners
- All lines being of similar length

The correct size of the area of the square is determined in the following way: Initially, the size is estimated using the altitude provided by the flight controller. This measurement often shows deviations from the actual altitude of up to $2\,\mathrm{m}$. To account for these inaccuracies, a tolerance range of $\pm 3.5\,\mathrm{m}$ is added to the estimated size. This tolerance range is specifically designed to be greater than the potential error in the altitude measurement, ensuring that the correct size of the square can still be detected even if the altitude estimation is not precise, as the actual size lies withing the range of possible calculated sizes.

Once the square is detected, this altitude is used as the new altitude reference for the square detection process while still adding the same tolerance.

The requirement for all lines to be of similar length is checked as shown Equation 2.5. The length of the longest and shortest line is calculated, and the difference is divided by the length of the longest line. The contour is only considered valid if this value is less than 0.2. This value has proven as a good match due to empirical results.

$$\alpha = (l_{longest} - l_{shortest})/l_{longest}$$
$$\text{Valid if } \alpha < 0.2 \tag{2.5}$$

The contours, after filtering using the parameter constraints explained above, are superimposed on the original grayscale image in Figure 2.14. The center of the resulting contour is then used to estimate its position.

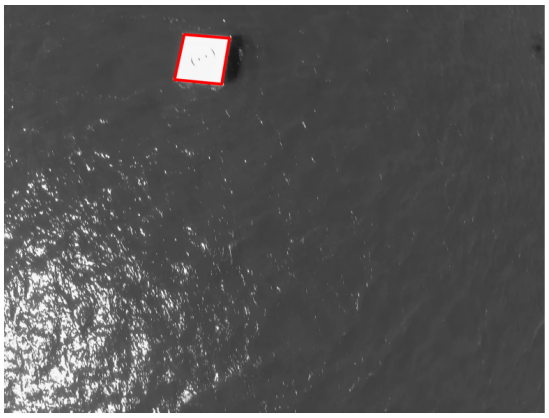


Figure 2.14: Fitted contours using thresholded image and checking if the contours have the right parameters. For easier viewing the contour is superimposed on the grayscale image.

2.5.4 Checking for the presence of landing platforms

When initially searching for a platform, the same approach of thresholding and contour approximation is used to check for the presence of a landing pad. To ensure robust detection, this algorithm is run repeatedly over a chosen period of 3 s to minimize the effect that noise or wind gust may have. The result is a list of positions relative to the copter where one or both landing pads are located. To check whether two landing pads have been detected, the distribution of locations in the horizontal plane is tested for bimodality with the dip test [11]. Depending on the result, the location of one or both of the platforms is saved.

If two platforms are detected, the algorithm selects the closer one as the first target and saves the location of the other helipad. If only one target is visible, this becomes the

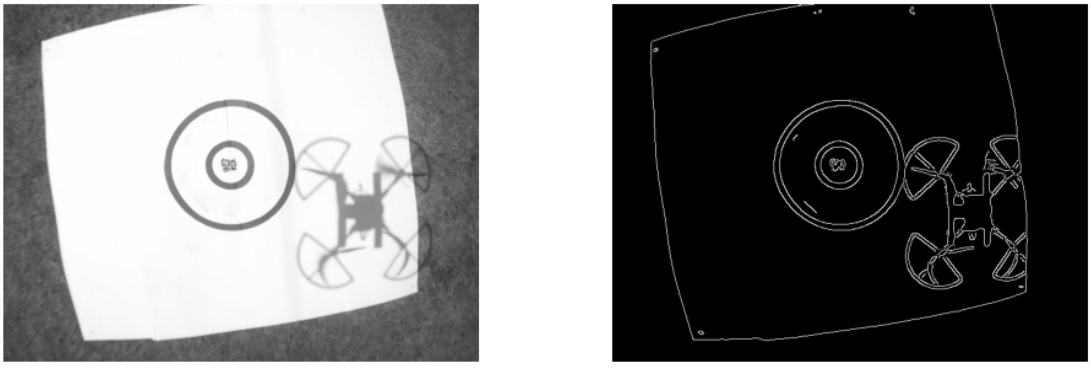
primary target. The UAV then flies to this target and starts descending on the basis of the continuously updated relative position of the square.

2.5.5 Detection of circles

Once the UAV has successfully descended lower over the target using the square as a reference, the square may fill out most or even all the image and therefore can no longer be used as a guide for descent. For this reason, one or both concentric circles must be used in order to estimate the relative position. Here canny edge detection is used, to find a set of possible contours. The Hough Circle Transformation is then used to find the most likely circles from the edge detection.

Canny edge detection

The Canny edge detection algorithm uses multiple steps to find sharp gradients in the image. This is done by first applying a 5x5 Gaussian filter to reduce the influence of noise in the image. Next, the first derivative in both the horizontal and vertical directions is then computed. More steps are taken to remove unwanted pixels that are not part of an edge. The process is described in more detail in [12]. The result of applying this algorithm to an image of the landing pad is visible in Figure 2.15. The algorithm runs on a grayscale image, as it is based on the intensity of the pixel values.



(a) Grayscale image of the landing pad. (b) Result of canny edge detection.

Figure 2.15: Canny edge detection performed on a grayscale image.

Hough Circle Transformation

The Hough Circle Transform is based on the Hough Transform and is used to detect circular objects in images. This algorithm works by testing how well a circle of a certain radius fits the edge points in the image. The algorithm searches for circles between r_1 and r_2. The two radii r_1 and r_2 are calculated based on the estimated height of

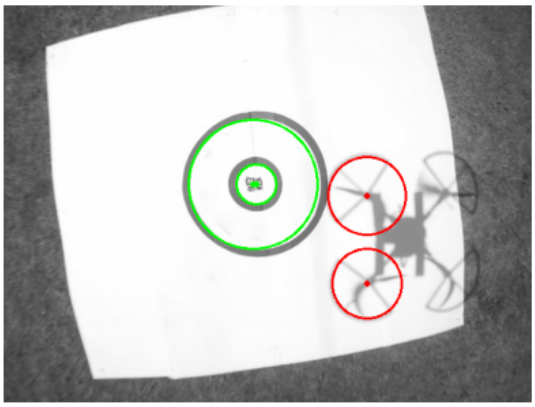


Figure 2.16: Result of Hough Circle Transform used to detect the concentric circles. The red circles are unwanted. The green ones have a similar center point.

the UAV. Being able to limit these radii to a relatively small range vastly speeds up the process of finding suitable circles, and thus increases the measurement frequency. A more detailed explanation of the Hough Circle Transform can be found in [13].

2.5.6 Detecting concentric circles

To detect concentric circles, the Hough Circle Transform is used with different settings for the minimal and maximal radii. Figure 2.16 shows the detected circles . To ensure that the correct circles are used as a reference for the landing pad location, one of the small and big circles needs to have approximately the same center. The center point of the larger circle is then used as a reference for the control algorithm. This method of selecting correct circles based on the center points allows for lower thresholds for the Hough Circle Transformation, which makes it more robust to noise and distortions in the image. Higher thresholds tend to pick up fewer circles, reducing both false and true positives. This is especially problematic in challenging lighting conditions.

2.5.7 Detecting inner circle

The inner circle is detected in nearly the same way as the concentric circles, but with a higher threshold for the Hough Circle Transformation and lower range of radii. This higher threshold means that the circle has to be closer to a perfect circle to be recognized as valid. This is used to try and enforce no more than 1 circle is detected. Although this makes this less robust than the method for concentric circle detection, the noise and distortions also have less influence since the UAV is closer to the target, and therefore the feature fills out more of the image.

2.5.8 Detection of tins

To be able to land on a tin, the UAV hovers at an altitude of $1.5\,\mathrm{m}$ and aligns itself over the closest tin relative to the center of the inner concentric circle. This requires both the tin and the inner circle center to be in frame, as the distance between the two is used as a criterion for selecting the right tin.

The procedure involves the following steps:

1. Calculating which tin is the closest to the inner circle center.
2. Save the distance to the center of the circle and the position of the tin.
3. Check for tins again and select tins with an approximately matching distance to the center of the circle.
4. Of the selected tins, choose the one closest to the previous position in the image frame as a reference.
5. Save the new position and use the tin to align the UAV.
6. Repeat from Step 3 until the UAV is well aligned.

As both the inner concentric circle along with the tin need to always be visible, the altitude at which the UAV can align is constrained to a minimum height. Ample space around the necessary features is required in order to counter wind gusts that can shift the visible area of the camera. This is why a relatively high altitude of $1.5\,\mathrm{m}$ is used. Figure 2.17 shows the detection of the tins along with the closest selected as a reference for the alignment. The red arrow points to the desired location.

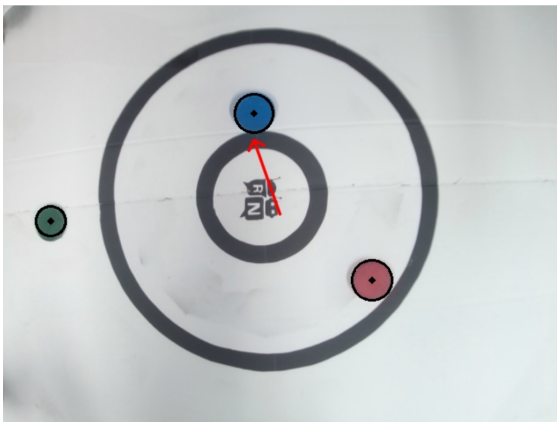


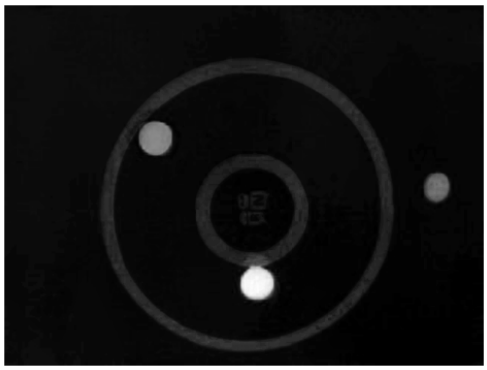
Figure 2.17: The closest tin to the center is selected.

HSV color model

The detection of the tins is based on the fact that they are the only colorful objects on the platform, the rest of the image being a mix of gray, white and black. Thus, the image is converted into the hue, saturation, value (HSV) color model. In contrast to the use

of the red green blue (RGB) or cyan magenta yellow key (CMYK) color model, the use of HSV allows for a dissection based on the apparent intensity of the color.The Hue in HSV specifies, as the name implies, the perceived color of the pixel, while the Value represents the brightness. The Saturation is the relevant parameter, as this contains the purity of the color. Perfect white and black have a saturation of zero, while a more full or vivid color has a higher value. This means that the saturation channel is a good fit to detect colorful objects and is thus used for the tin detection. Figure 2.18 shows the original image along with the grayscale image of the Saturation channel. The tins stand out, with the concentric circles only being slightly visible. However, this image was recorded inside and therefore does not show problems due to imperfect lighting. Challenges related to detecting the tin in less optimal lighting conditions are discussed in a later chapter.

(a) RGB image of landing with tins.



(b) Saturation channel of HSV image.

Figure 2.18: Image captured indoors by holding the UAV over the landing pad. In the saturation channel, the tins clearly stand out as they are the only colorful objects.

3 Results

This chapter explains the results observed from the testing. As validating image recognition code on a flying UAV is coupled with significant effort and risk, this part of the code is first tested using a scaled-down platform over water while holding the drone by hand. The controls are validated in a simulation environment to reduce the risk of failure due to logic errors. Testing code in simulation is also a faster way to verify new features, as there is close to no setup time.

3.1 Offline testing and tuning

Figure 3.1 shows a test setup for verifying the image processing without flying the UAV. The target is scaled-down by a factor of 5 so that holding the UAV at a height of $2\,\mathrm{m}$ is equivalent to $10\,\mathrm{m}$ when flying. This means that the full range of altitudes can be tested at scale.

Figure 3.1: Collecting sample data on a scaled-down platform to validate the image recognition algorithm.

The captured video stream and raw data from the flight controller are recorded as a Rosbag. This enables testing of the code via playback of the captured data. The data recorded during those tests are then used to tune the image recognition algorithm. The key values that needed to be tuned are:

- Factor by which the Otsu's thresholding technique is adjusted. See equation 2.4.
- Threshold for the Canny edge detection
- Threshold for the Hough Circle Transformation
- Tolerances for the size estimation of the square, concentric circles, and tins

3.2 Testing in simulation

The algorithm must also be tested in simulation. For this, a Gazebo environment provided by the RobotX competition organizer is modified and used. The simulation contains a 3DR IRIS quadcopter with a downward-facing camera and landing pads placed in the water. Figure 3.2 shows a landing using this setup.

3.2.1 Landing on the center circle

Figure 3.4 shows the estimate of target position performed by image recognition. Subfigure (a) plots the altitude vs. the total distance to the target in m in the horizontal plane for a typical trial. Due to the x-axis displaying the distance to the target rather than a position, the UAV never reaches zero and cannot cross into negative distances.

As can be seen, the UAV is already well aligned at the start of the descent. The dashed and dotted lines represent the closest edge of the platform to the center and the outer circle, respectively. Note that these are used as a reference for the reader and do not exist other than lines on the landing pad itself.

In general, the distance to the center of the target can be divided into the three categories listed below and illustrated in Figure 3.3.

- Safe landing below $87.5\,\mathrm{cm}$ to the center of the target.
- Safe or unsafe landing depending on the position of the UAV between $87.5\,\mathrm{cm}$ to $128.5\,\mathrm{cm}$.
- Unsafe landing beyond $128.5\,\mathrm{cm}$ to the center of the target.

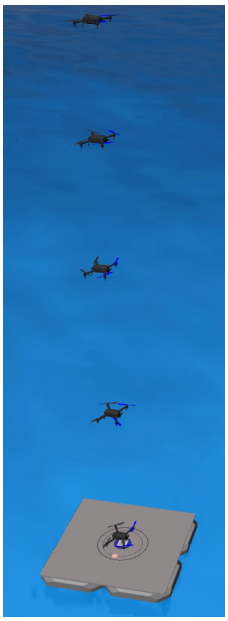


Figure 3.2: Photo series of UAV landing on the platform.

These distances correspond to the landing gear being 25 cm apart. Therefore, one leg of the landing gear can fall off at precisely 87.5 cm if the landing gear is parallel to the edge of the platform and in the middle of the edge. The other extreme is a landing on the corner where either both landing gears have to be halfway on the platform or the outer landing gear is on the very corner of the landing pad. This both comes out to be 128.5 cm for both cases.

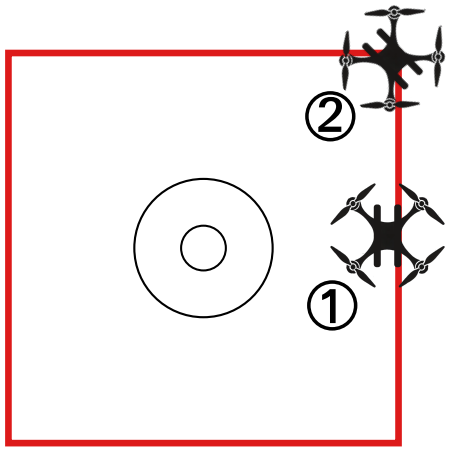
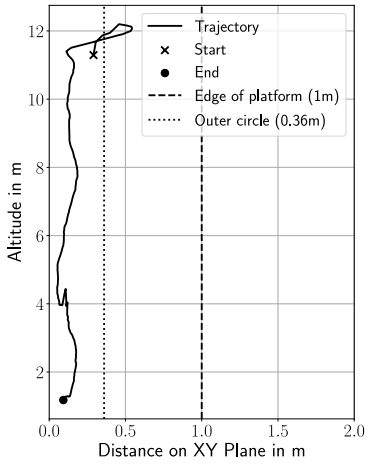
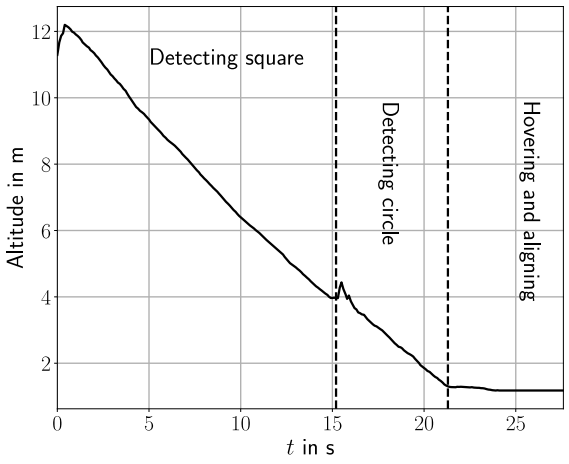


Figure 3.3: Edge cases relevant for a safe UAV landing. Edge case 1 represents the border of a safe landing at a distance to the center of 87.5 cm. Between a distance of 87.5 cm to 128.5 cm (edge case 2), the landing is potentially safe or unsafe depending on the position of the UAV. Everything beyond 128.5 cm will fall off the platform and is therefore unsafe.

The edge of the platform is used in all figures as a reference, but should be noted that a landing at a distance of 1 m can already be unsafe.



(a) Altitude vs. distance in x-y plane plotted. At the start of the ascent, the UAV is already well aligned.



(b) Altitude vs. time. The different states are marked by vertical lines.

Figure 3.4: Position measurements produced by the image processing algorithm.

As can be seen in Figure 3.4 (a), the algorithm keeps the UAV well aligned, staying always within the limits of the outer circle. Subfigure (b) plots altitude vs. time. The UAV uses the square as a reference until $t = 15\,\mathrm{s}$. When switching to the concentric circle, a small peak can be seen as the size of the features does not match exactly the expected one. After the UAV reaches an altitude of $1.5\,\mathrm{m}$, the UAV aligns itself, which starts at $t = 21\,\mathrm{s}$. This results in the altitude graph flattening out. The graph ends at $t = 28\,\mathrm{s}$ as image recognition is not used for the final unguided part of the landing.

3.2.2 Landing on a tin

As explained above, the approach for landing on a tin is nearly the same. However, because of the need to align first with the center of the platform and then align over a tin, the hover phase is longer. This becomes clear in Figure 3.5. When switching to using the tin as a reference, the distance to the target peaks at $t = 39\,\mathrm{s}$. In this specific example, the duration of the hovering process has increased by approximately $8\,\mathrm{s}$ from being well aligned with the inner circle at $39\,\mathrm{s}$ to being aligned with the tin and landing at $47\,\mathrm{s}$. While the specific time by which the hovering process is increased fluctuates, it is always longer because of the inherent nature of aligning twice which is required for landing on a tin. This can be problematic when wind conditions complicate the alignment in the first place.

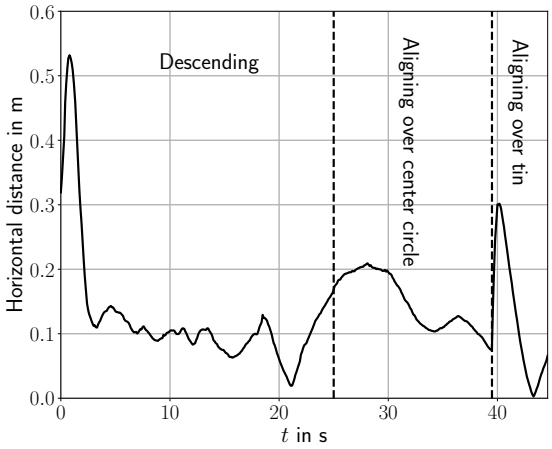


Figure 3.5: Distance to object of interest vs. time. The object of interest switches from the middle of the platform to a tin at the second dashed line.

3.3 Real-time testing

3.3.1 Testing on land

After the underlying logic and image recognition have been proven in simulation, multiple flights are conducted both to land on the tin and to land on the center circle, to determine the accuracy and effectiveness of the process.

The first such testing was tuning of the P controller used for maintaining horizontal position. For this, the P controller for both axes in the horizontal plane is tuned until oscillations occur. Figure 3.6 shows the result of a poorly tuned P controller where the gain is set too high. The position estimate captured from the camera is relative to the distance from the target along the east-west axis. The peak at $t = 50\,\mathrm{s}$ is the result of the tracking switching to a tin instead of the inner circle.

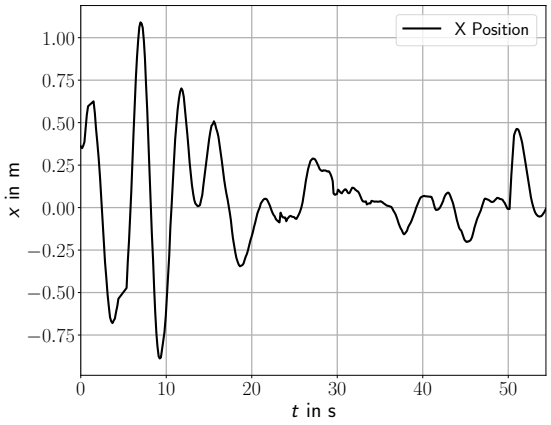


Figure 3.6: Distance to the landing pad in x. x refers to the east-west axis, with east being positive x. Oscillatory behavior can be observed especially at the beginning of the alignment process.

However, it must be noted that tuning based on the camera position estimation has a few issues. As described earlier, the pitch and roll angle of the UAV are neglected to reduce noise and complexity. This means that when the UAV is not level, the estimated position has increased error. Figure 3.7 illustrates this problem. Although the UAV is, in this example, almost directly over the target, the error gathered from the image suggests that the target is to the right of the copter.

The position estimation combined with the controller acts as a negative feedback loop. If the target is detected, as shown in the figure, on the right side of the image, the drone tries to accelerate in this direction. However, in order to accelerate, the UAV has to bank in this direction, thus increasing the received error to the target.

This means that potential slight oscillations that can be observed in the plot of the position relative to the target are not actually real oscillations but rather an artifact of the way the position is measured. An external way to measure the position would be greatly beneficial in tuning the controller.

The tuning procedure is not needed for the altitude controller as the gain used for the simulation also proved to be working well in real-world testing.

Landing on the circle center

A total of 7 attempts were made to evaluate the precision of the algorithm. Wind speed was not measured, but estimated to be around $1\,\mathrm{m\,s^{-1}}$ from north to east. During

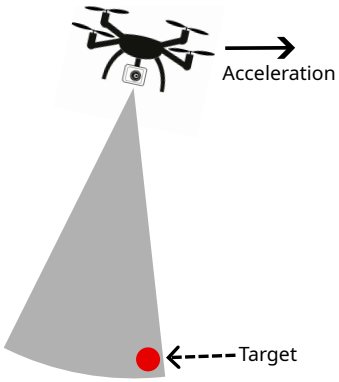


Figure 3.7: The position estimation returns poor results if the UAV is not level relative to the horizon, as the camera is not pointed straight down as assumed in the calculations.

Table 3.1: Results of landing on the center circle. All distances are measured in cm from the camera to the center of the inner circle. A positive distance northward means the UAV landed north of the center, a negative distance means the UAV landed south of the center.

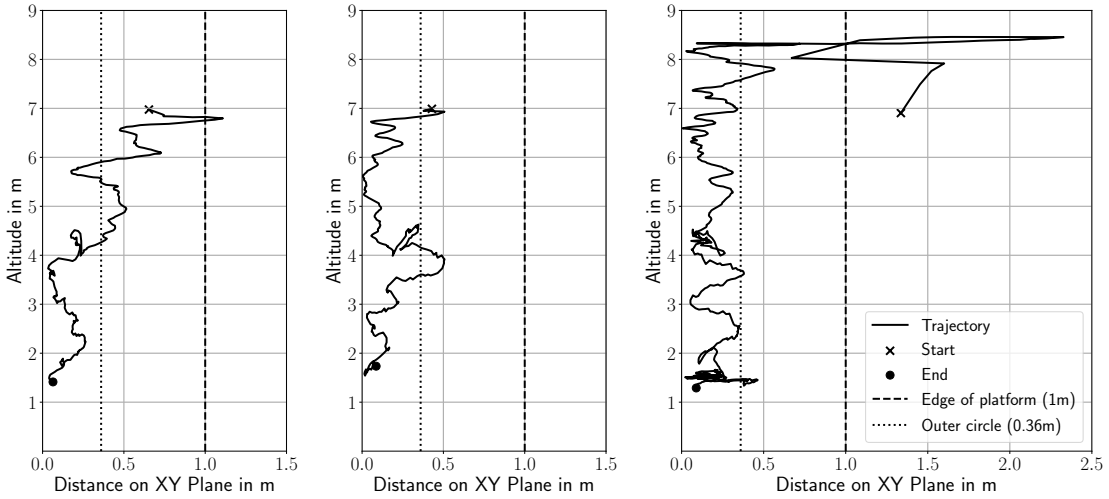
Attempt	Dist. northward	Dist. eastward	Total dist.
1	−26 cm	5.5 cm	26.58 cm
2	−26 cm	−1 cm	26.02 cm
3	3.5 cm	−4 cm	5.32 cm
4	10 cm	−5 cm	11.18 cm
5	8 cm	9 cm	12.04 cm

testing, issues with flying precisely to the first waypoint to start detecting the target forced the premature stop of two attempts. This had to be done as the UAV is flown in a netted flight cage, and the altitude at which the procedure can be safely tested is limited to the roof of the cage at about 9 m. The inability to reliably fly to the desired altitude of 6 m to 7 m is most likely a result of bad GPS vertical position estimates, as the flight cage is located right next to a building. This is also supported by the fact that when testing over water, the flight altitude appeared much more consistent.

The distances after landing of the 5 attempts where the UAV reached the desired waypoint are measured from the center of the camera to the center of the inner circle. Table 3.1 shows the results of these tests. The weather was sunny, which resulted in a high exposure of the image. However, this did not prove to be a problem as the features were reliably tracked. In terms of accuracy, the UAV was able to land within the outer circle for all 5 attempts. The average accuracy is 16.23 cm, and all tries stayed well within the safe limit of 87.5 cm. As the inner circle has a radius of 12 cm, the UAV landed inside 2 out of 5 times inside this radius. These test therefore indicated a strong change this approach can work over water as none of the attempts were close to an unsafe distance from the center.

The recorded position estimate during the landing procedure produced by the image recognition algorithm is depicted in Figure 3.8. Attempt number 5 shows the greatest

fluctuation in the estimation of the distance to the target. One reason for this could be the fact that there were more gusts during the descent, but further tests would be needed to validate this. Observations from other tests show, that the larger starting error is unlikely to influence the fluctuations later in the descent.



(a) Attempt 1 (b) Attempt 2 (c) Attempt 5

Figure 3.8: Altitude vs. distance to center of landing pad for three different landing attempts.

Landing on a tin

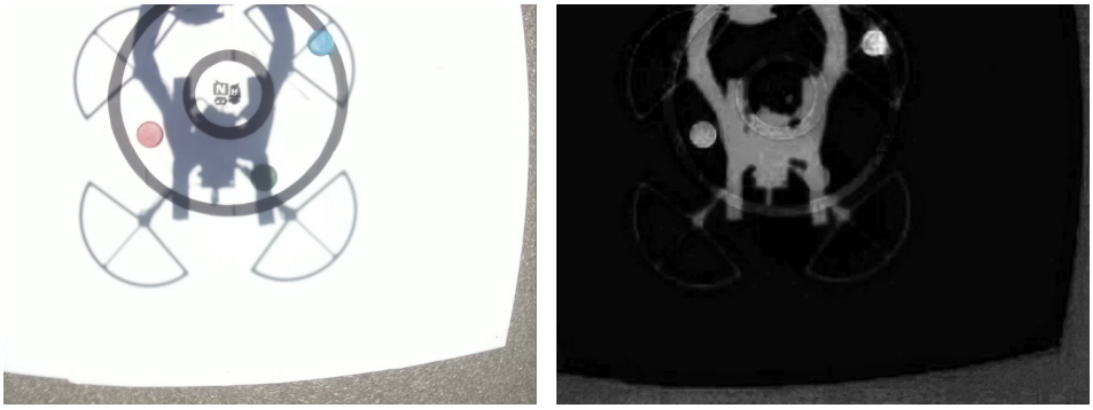
To test the landing on a tin procedure, three tins of different colors, blue, green, and red, were placed at different distances relative to the center of the landing pad. This was performed in calm weather with an estimated wind speed of $1\,\mathrm{m\,s^{-1}}$ from the North.

Seven attempts were carried out with an average accuracy of $22.31\,\mathrm{cm}$ from the selected tin. In attempts 2 to 7, the selected tin was also the closest to the center of the platform. However, in the first attempt, the algorithm initially selected the correct tin for tracking, but later switched to a nearby one due to poor detection of the primary tin. The issue was that the tin intended for detection was only intermittently detected, leading to inconsistent results. The distance measured in the table is the distance from the tin it attempted to land on, and not the one it initially saw.

Moreover, the algorithm proved to be very dependent on lighting conditions, as the image was overexposed and the Hough Circle Transformation, combined with the Canny edge detection for the saturation channel, failed to perform consistently. The overexposed image is shown in Figure 3.9 together with the corresponding saturation channel. As is evident, the color information is lost due to the shadow of the drone appearing as colorful as the colored tins themselves. The reason for this was an incorrect aperture setting, with the aperture being too wide open.

Table 3.2: Results of landing on a tin. All distances are measured in cm from the camera to the center of the tin. A positive distance northward means the UAV landed north of the center, a negative distance means the UAV landed south of the center.

Attempt	Dist. northward	Dist. eastward	Total dist.
1	6 cm	6 cm	8.49 cm
2	−15 cm	−25 cm	29.15 cm
3	−32 cm	3 cm	32.14 cm
4	−14 cm	−14 cm	19.80 cm
5	−20 cm	−12 cm	23.32 cm
6	−2 cm	22 cm	22.09 cm
7	7 cm	−20 cm	21.19 cm



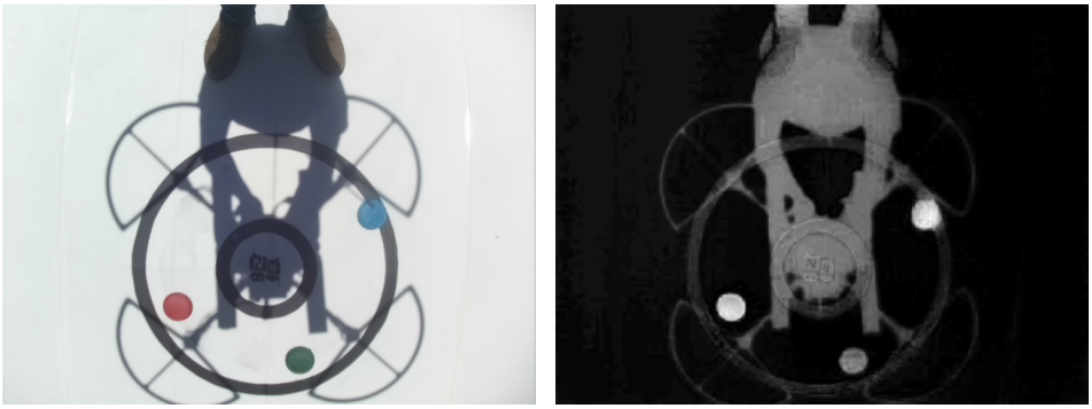
(a) Original image (b) Saturation channel

Figure 3.9: Image captured by holding the drone over the target. This is the aperture setting used for the tests shown in Table 3.2. As the aperture is too open, much of the color detail is lost.

In contrast to this, Figure 3.10 shows an image taken with a more closed aperture. Although the green tin still does not appear clearly in the saturation channel, the red and blue tins stand out. Brighter spots in the saturation channel correspond to a higher saturation value.

With this adjusted aperture setting, the tests were conducted again. Due to the red and blue tin standing out much better, the Canny edge detection, as well as the Hough Circle Transformation, can be tuned to be more robust. With these adjustments, 12 more test flights were conducted. Two of those needed to be interrupted prematurely due to the UAV flying too close to the top of the flight cage. However, both of these failures occurred before the target could be detected. The remaining 10 flights are listed in Table 3.3.

In attempt number 6, the algorithm once again accidentally switched to tracking the wrong tin partway through the alignment process. As described before, the distance



(a) Original image (b) Saturation channel

Figure 3.10: Image captured by holding the drone over the target. This is the aperture setting used for the tests shown in Table 3.3. Due to the image not being overexposed, the color information is retained much better.

Table 3.3: Results of landing on a tin. All distances are measured in cm from the camera to the center of the tin. A positive distance northward means the UAV landed north of the center, a negative distance means the UAV landed south of the center.

Attempt	Dist. northward	Dist. eastward	Total dist.
1	11 cm	26 cm	28.23 cm
2	22 cm	−22 cm	31.11 cm
3	22 cm	−12 cm	25.06 cm
4	−4 cm	−5.5 cm	6.80 cm
5	8 cm	−8 cm	11.31 cm
6	18 cm	−5 cm	18.68 cm
7	5 cm	1 cm	5.10 cm
8	40 cm	20 cm	44.72 cm
9	3 cm	12 cm	12.37 cm
10	24 cm	2 cm	24.08 cm

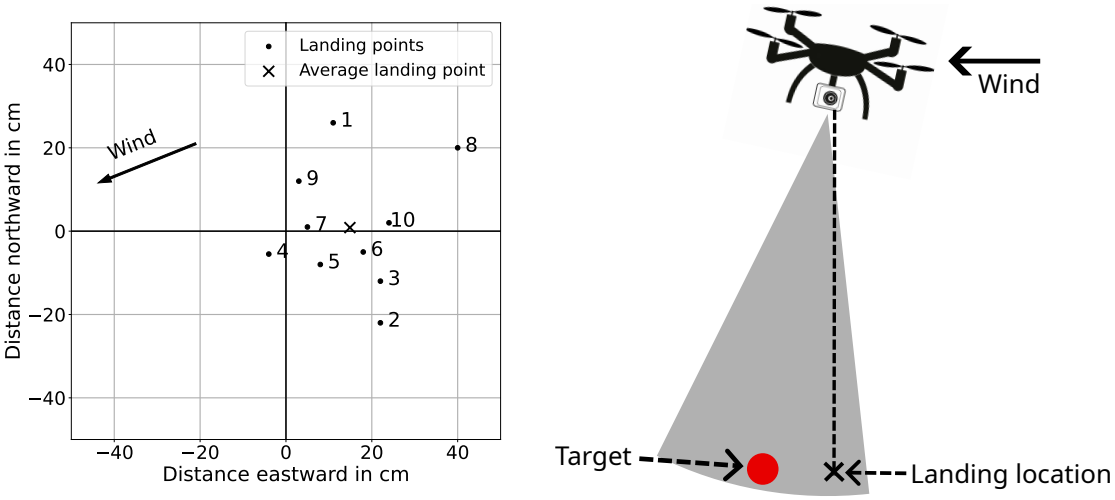
measured is relative to that second incorrectly tracked tin and not to the one closest to the center.

The average distance from the desired landing location is 20.75 cm. The wind speed was estimated to be around $3\,\mathrm{m\,s^{-1}}$ from East-northeast (ENE). The average landing accuracy therefore decreased from 22.31 cm to 20.75 cm even though the wind increased from $1\,\mathrm{m\,s^{-1}}$ to $3\,\mathrm{m\,s^{-1}}$ compared to the trial listed in Table 3.2. This is a strong indicator that the more reliable visual tracking decreased the average error.

Figure 3.11 (a) shows a map of landings relative to this desired position at (0,0). It is notable that the landing points are grouped up towards the east of the target. The reason for this lies in the way the landing is performed. As described above, the UAV hovers at an altitude of 1.5 m and aligns over the target before landing comparatively

quickly without using image recognition. This final landing part can be assumed to be approximately straight down.

If the UAV has to bank against wind in order to hold its position, the target may wrongfully appear to be directly below the UAV. This problem is illustrated in Figure 3.11 (b). This discrepancy likely explains the general offset observed in the landing positions.



(a) Map of landing locations relative to the target.

(b) UAV banking against wind, seemingly aligned if only the captured image is taken into account.

Figure 3.11: Location of the landing and a depiction of the UAV banking against wind. The wind speed is estimated to be around $3\,\mathrm{m\,s^{-1}}$ from ENE.

3.3.2 Testing over water

In order to test over water, the landing pad is mounted on top of floating dock pieces. The contraption is placed in a tidal creek. During testing, the direction of flow of the water remained approximately constant. The helipad is kept steady with the help of two anchors that are arranged in a V-shape against the current. Most of the movement due to changes in flow or other factors such as wind is therefore minimized. Figure 3.12 shows this setup with the UAV on the target.

The UAV was started from an elevated dock on land and guided to a manually estimated position of the floating platform. A total of 12 flights were conducted. Table 3.4 shows the attempts with the corresponding outcome and 3.5 provides a summary grouped by results.

Only two flights landed on the platform. Four attempts had to be interrupted, as the estimated waypoint was too far away from the helipad and therefore not in view of the camera. In addition, difficulties with the connection between the onboard computer and flight controller caused a failsafe event initiated by the flight controller. The failsafe on the flight controller is set to automatically use the manual control input provided by the

Figure 3.12: Test setup on water. The landing target is mounted to floating dock units and held in place by two anchors. The anchors are attached to the blue and pink ropes attached to the corners of the floating dock.

remote control when communication to the onboard computer is lost. This led to one attempt accidentally landing on the platform due to the low position of the throttle stick, as well as one that had to be aborted. In terms of image processing, the detection of the landing platform as a square failed 3 times due to issues with glare on the water. Moreover, the inner circle tracking failed an additional two times, resulting in the target being lost and automatically ascending.

Because of various problems with landing on the platform itself, landing on a tin was not tested over water. The following subsections discuss the different categories mentioned in Table 3.5 in the same order as listed there.

Table 3.4: Results of landing on the floating helipad. The attempts are listed in order with the outcome listed on the right-hand side.

Attempt	Result
1	Did not fly far enough
2	Flew too far
3	Flew too far
4	Inner circle detection failed
5	Flew too far
6	Square detection failed (glare)
7	Lost connection (flight controller to onboard computer)
8	Landed successfully (29 cm)
9	Inner circle detection failed
10	Detected two squares, aborted
11	Accidental landing (lost connection, distance 43 cm)
12	Square detection failed

Table 3.5: Results of landing on the floating helipad. This table is a summary of Table 3.4. The categories are not mutually exclusive, as one landing on the helipad was caused by a lost connection between onboard computer and flight controller. This landing therefore appears both in "Landed on helipad" and "Lost connection between Raspberry Pi and flight controller".

Outcome	Number of attempts
Landed on helipad	2
Manually estimated waypoint not close to the platform	4
Lost connection between Raspberry Pi and flight controller	2
Detection of square failed	3
Tracking of the inner circle failed	2

Successful landings

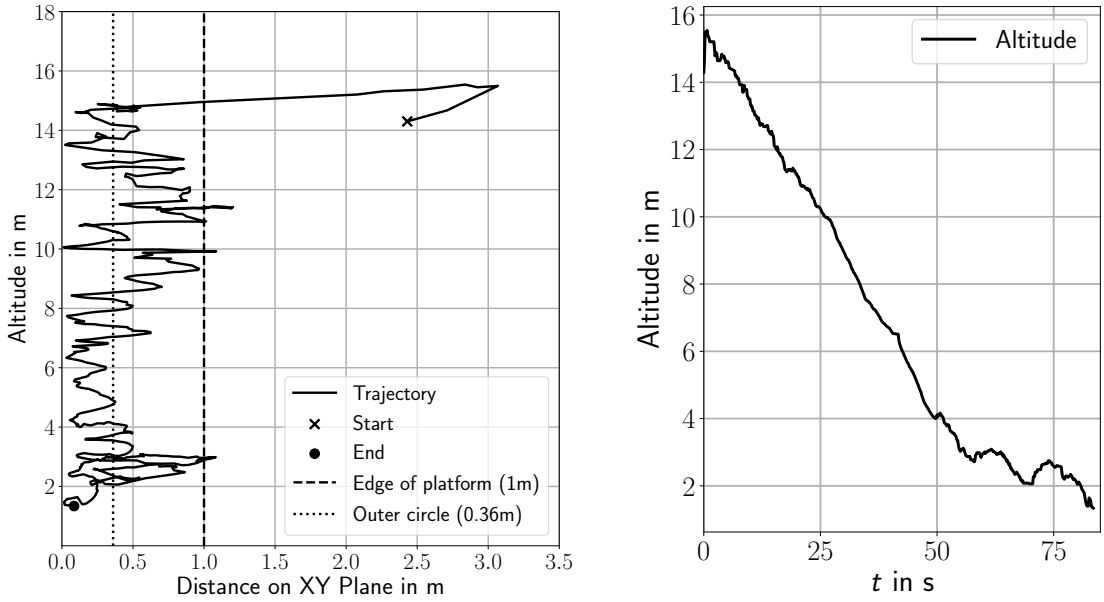
Of the two landings on the helipad, only one was planned, the other being a result of the lost connection between the computer onboard and the flight controller. This only truly successful landing is discussed here in more detail.

Figure 3.13 shows the landing path taken by the UAV as calculated by the camera position estimation. The UAV starts the descent at $15\,\mathrm{m}$ and aligns itself over the platform. However, during the descent from $15\,\mathrm{m}$ to $1.5\,\mathrm{m}$ the distance to the center point of the platform is often much greater than that seen in attempts over land, such as illustrated in Figure 3.8. This is likely a combination of the platform moving on water and the higher wind speeds experienced of about $4\,\mathrm{m\,s^{-1}}$ on the day of testing. A slower detection of the target features does not appear to be the cause of this problem, as the rate at which the detection is run can be seen in Table 3.6, both for the tests over land with much lower errors during descent, and the landing over water.

In Figure 3.13 (b), the altitude is increasing twice at $55\,\mathrm{s}$ and $62\,\mathrm{s}$ due to a temporary loss regarding the tracking of the inner circle and the therefore initiated ascent.

Table 3.6: Frequencies of the image processing algorithm depending on the feature and the test location. No notable difference can be observed, with the variation lying within the measurement inaccuracy due to a small window size of about $2\,\mathrm{s}$ to $10\,\mathrm{s}$ depending on the feature measured.

	Feature	Image processing frequency
Testing over land	Square	7.5 Hz
	Concentric circles	9.8 Hz
	Inner circle	9.9 Hz
	Tins	6.5 Hz
Testing over water	Square	7.7 Hz
	Concentric circles	9.5 Hz
	Inner circle	10.2 Hz
	Tins	not measured



(a) Altitude vs. distance to center of landing pad.

(b) Altitude vs. time plot.

Figure 3.13: Path captured by the camera on the successful landing attempt.

Manually estimated waypoint not close to the platform

Failure to reach the correct waypoint is due to difficulties in estimating the position from land purely by visual means. For future testing, this is not expected to be a problem, as the UAV will receive the waypoints from the USV. The object location done by the USV has an accuracy of roughly 20 cm. Therefore, only a failure in correctly deciding which object corresponds to the platform could result in the waypoint being too far away.

Lost connection between Raspberry Pi and flight controller

It is not clear what caused the loss of connection between the onboard computer and the flight controller. This behavior could not be replicated in the flight cage. Possibilities for this include overheating of the onboard computer or a loose USB connection between the two components. Further testing is needed to verify this.

Failure of square detection

Although water glare is expected, the intensity and density of the glare pattern compared to scaled-down testing increased significantly. Figure 3.15 demonstrates the reason why a higher altitude causes more glare. With an increase of the altitude, a greater number of wavelengths that reflect the sunlight are in view of the camera. Therefore, the individual spots of glare occur less separated and more as a continuous area. This significantly increases the difficulty of filtering out the glare using thresholding and morphological operations.

The altitude the UAV is flown at increased from 2 m for scaled-down testing to about 15 m. The initially planned altitude of 10 m was overshot due to the greater height of the launch position relative to the target. Figure 3.14 shows the increase in glare from scaled-down testing to flying at 6 m and 15 m.

(a) Less glare visible during scaled-down testing. Altitude 2 m.

(b) Glare at 6 m.

(c) Glare at 15 m.

Figure 3.14: Glare seen at different altitudes. Although the sun's angle of incidence differed between scaled-down testing and water testing, the points of glare observed at various altitudes were still significantly more widely spaced.

(a) Glare at low altitude.

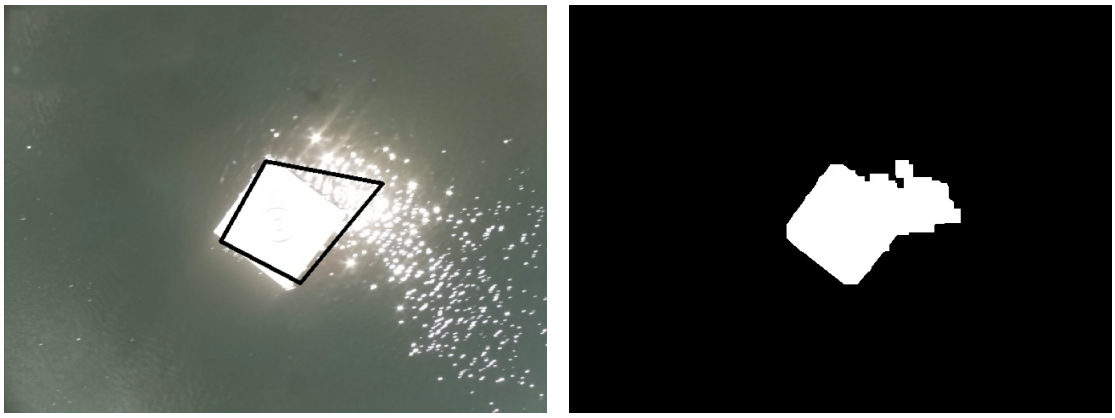
(b) Glare at high altitude.

Figure 3.15: Cause of glare on water. With increasing altitude, more wavelengths are visible, appearing as a brighter area. A low altitude also results in bright spots, but those are spaced further apart, appearing less bright.

Initially, because the sun was nearly directly overhead, the glare appeared right next to the helipad. This distorted the detected shape of the helipad as depicted in Figure 3.16. In order to detect this as a valid landing platform, the parameters discussed in Section 2.5.3 need to be less strict.

Specifically, the size of the area is changed from the expected size at the current altitude ± 3.5 m to ± 5 m and the formula for the accepted angles is modified as illustrated in Equation 3.1.

$$\alpha = (l_{\text{longest}} - l_{\text{shortest}})/l_{\text{longest}}$$
$$\text{Valid if } \alpha < 0.4 \tag{3.1}$$



(a) Glare appears right next to the helipad. This distorts the detected contour of the helipad.

(b) Thresholded image with morphological operations applied.

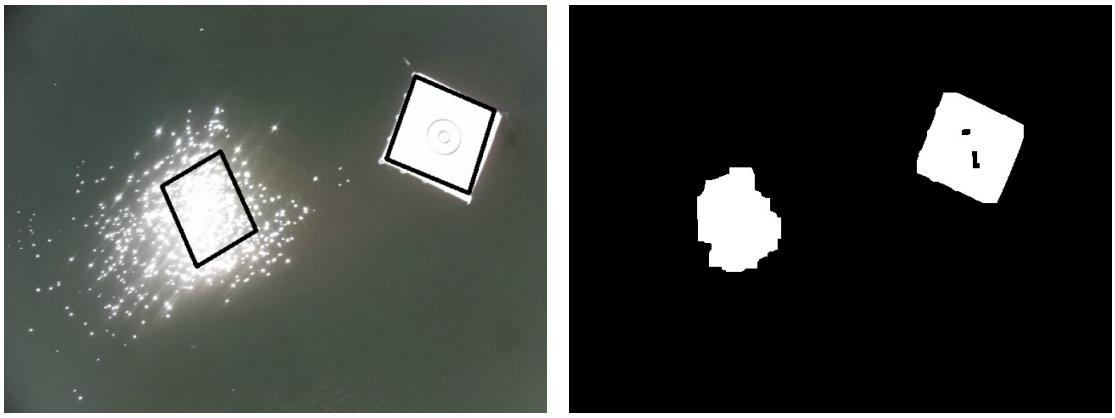
Figure 3.16: Original image and the result of thresholding. The glare and platform are directly adjacent.

However, this becomes an issue when the glare and the platform are not adjacent but still within the same field of view. Figure 3.17 shows the glare incorrectly detected as the landing pad due to less strict parameters concerning contour detection. Therefore, solving this issue purely on the basis of changing the parameters associated with the shape does not prove to be feasible. Moreover, depending on the intensity of the Sun, it cannot be assumed that the landing platform is the object with the greatest size after thresholding and performing morphological operations.

Figure 3.18 shows the histogram of the landing platform in water at 15 m cropped with no glare. The two peaks that correspond to the water at 75 and the platform at 255 are clearly visible. However, Figure 3.19 shows a very similar distribution for the glare cropped from the same image. Both histograms have sharp peaks at 255, making it impossible to filter out the glare by choosing a threshold closer to 255.

Failing to track the inner circle

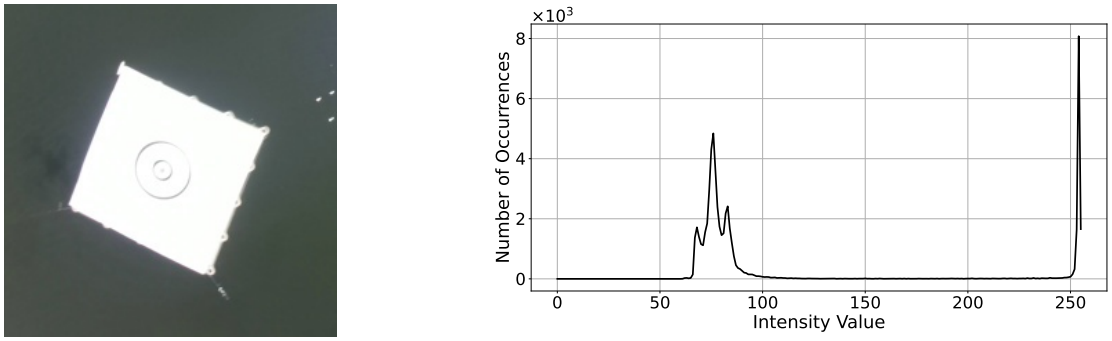
Two tries failed due to insufficient tracking of the inner circle. The difficulties detecting the inner circle originate from it appearing distorted in some frames. The distortions appear to be caused by vibrations or movement of the UAV in combination with the rolling shutter effect [14]. The distorted inner circle, combined with the higher Hough Circle Transformation threshold compared to the concentric circle detection threshold, leads to no detection of that feature. The reasons for having to use the higher threshold are mentioned in subsection 2.5.7.



(a) Image of platform and glare on the water. The black boxes indicate detected features. The glare is incorrectly detected as the helipad.

(b) Corresponding image after thresholding. The glare in the water is still clearly visible.

Figure 3.17: Original image and the result of thresholding. The glare and the platform are not right next to each other.



(a) Cropped out landing platform in the water without glare.

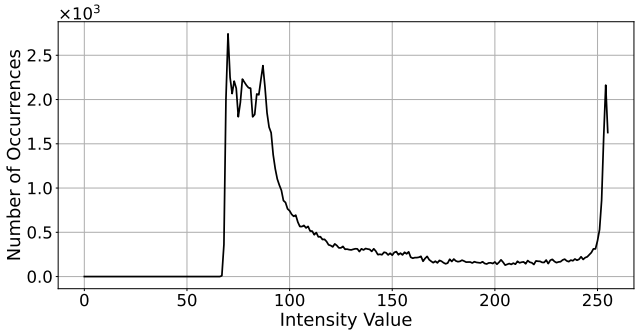
(b) Histogram of the grayscale picture.

Figure 3.18: Cropped out image of the platform over water. The histogram shows two peaks corresponding to the white target close to 255 and the water at around 75 in value.

Figure 3.20 shows the difference between two frames taken $\frac{1}{30}$ s apart. Although 3.20 (b) shows distortion, this occurs much less in 3.20 (c). The UAV has stayed close to stationary in between the two frames as the image remains largely the same except for the distortions that are most notable on the inner circle. This leads to 3.20 (b) not being detected while the inner circle is detected in 3.20 (c) again.

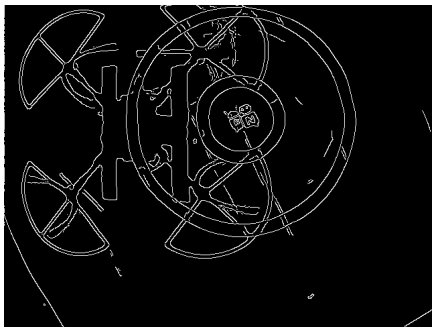
Due to insufficient time, no improvements could be made and tested again over water. However, possible fixes for the two issues found with image processing are discussed in the outlook.

(a) Cropped out section of glare on the water.



(b) Histogram of the grayscale picture.

Figure 3.19: Cropped out image of the glare seen on the water. The histogram shows two peaks corresponding to the glare close to 255 and the water at around 75 in value.



(a) First image captured by the camera with Canny Edge detection applied. This image is not cropped and purely for reference purposes.

(b) First image cropped to only show the inner circle. The inner circle is distorted as is apparent when compared to the red dashed circle.

(c) Second image captured $\frac{1}{30}$ s after the first image and also cropped. Much less distortion is visible.

Figure 3.20: Images with Canny Edge detection performed taken $\frac{1}{30}$ s apart. The inner circles are cropped out to show the distortion more clearly. The red dashed circle is only for reference. The images are captured at $1.4\,\mathrm{m}$.

4 Conclusion and outlook

4.1 Conclusion

This thesis presents a vision and controls system to allow autonomous UAV landing on a floating platform over water. To accomplish this, a tracking algorithm was developed in order to locate the target. This relative location is then used to control the UAV with respect to this platform.

In the simulated environment, testing was successful both for landing on tins and in the inner concentric circle. However, when transitioning to the real world, various difficulties emerged, such as overexposure and wind that influence both the reliability and accuracy of the landing. Over land, after some adjustments, the landing process was still very successful with an average accuracy of $16.23\,\mathrm{cm}$ for landing on the inner concentric circle and $20.75\,\mathrm{cm}$ for landing on a tin with a wind speed of $1\,\mathrm{m\,s^{-1}}$ and $3\,\mathrm{m\,s^{-1}}$ respectively.

Over water, various new problems arose, resulting in only two landings on the helipad out of 12 attempts. While this is in large part due to incorrectly estimated waypoints, the image recognition failed due to multiple reasons as well. The reflection caused by the water made successful detection of the square difficult.

4.2 Reflection and outlook

Throughout this research, several key insights and achievements were made.

The initial goal of the thesis was realized in large parts, except reliability over water. Sensors, flight controller, onboard computer, and camera were successfully integrated. The vision-based tracking system performs well for most of the environment where excessive glare is not present. On land, the UAV is able to descend on the target, showing the ability to land both on tins and the inner circle.

The following section discusses potential improvements and future research based on the findings of this thesis.

Adjusting the exposure time to be shorter or aperture setting to be more closed at high light intensity has the possibility of reducing the problems experienced with detecting the square platform. This could help create greater separation in the histogram between

the peak corresponding to the glare and the peak corresponding to the platform. It can be expected that the pixels representing the platform will retain a high intensity over most of the area, while the glare might show darker spots with a shorter exposure time or a more closed aperture. The darker spots correspond to parts of waves that are at the wrong angle to reflect light into the camera. This then allows for more efficient erosion of the glare from the outside and inside.

One option to fix the inconsistent inner circle detection is to use a lower Hough Circle Transformation threshold in conjunction with a tracking system for this feature. The lower threshold would allow for the detection of the inner circle even when it is slightly distorted. However, this would also lead to false positives being detected, such as propeller guards or other circular objects. The tracking system can then be used to select the most probable circle as the new reference. This probability can be based on the location of the detected objects relative to the previously detected position of the target, as both the UAV and the helipad are not expected to move much in between frames. This assumption of negligible movement is made as the frequency at which the detection runs is shown to be around $10.1\,\mathrm{Hz}$ as shown in Table 3.6.

Moreover, using a camera with a global shutter would eliminate nearly all distortion caused by vibrations, as the whole image is captured at once. Specifically, the use of the "Raspberry Pi Global Shutter Camera" would also allow for a higher dynamic range compared to the "Raspberry Pi High Quality Camera", therefore capturing more detail in both bright and dark parts of the image [15]. This is important as this would result in more detail of the glare being preserved, thus showing darker areas in between the bright spots. However, the lower resolution of global shutter cameras compared to rolling shutter cameras at the same cost is a drawback for other applications on the same UAV requiring a high resolution.

Another approach to mitigate inconsistent inner circle detection is the use of a 90° instead of a 65° lens. This would allow the use of the concentric circle detection down to $1.27\,\mathrm{m}$ with the same size of area in view as previously at $2\,\mathrm{m}$ with the 65° lens. Therefore, the uncertainty based on the altitude of the UAV before initiating the final landing procedure without image recognition could be reduced. Furthermore, because the area would be equivalent to the one previously in frame at $2\,\mathrm{m}$, image recognition based on the inner circle would no longer be required. With the use of the concentric circle detection as the last step, the algorithm would therefore be more robust due to the higher Hough Circle Transformation thresholds of this step. In testing, detection of the helipad location also never failed when using concentric circle detection.

This would also decrease the altitude needed at the beginning of the descent from $10\,\mathrm{m}$ to $6.37\,\mathrm{m}$. However, this is not expected to reduce glare in the image as the same number of wavelengths are still in view.

These improvements have the potential to improve the image recognition and therefore make landing more robust.

Bibliography

[1] A. S. Nair, P. A. Jeyanthy, L. Ramesh, G. M. Kurian, and S. R. Mohamed, "Autonomous precision landing with uav and auto charging," in *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, 2021, pp. 463–466.

[2] A. Khazetdinov, A. Zakiev, T. Tsoy, M. Svinin, and E. Magid, "Embedded aruco: a novel approach for high precision uav landing," in *2021 International Siberian Conference on Control and Communications (SIBCON)*, 2021, pp. 1–6.

[3] 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, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197624000228

[4] M. Moreira, F. Azevedo, A. Ferreira, D. Pedro, J. Matos-Carvalho, Á. Ramos, R. Loureiro, and L. Campos, "Precision landing for low-maintenance remote operations with uavs," *Drones*, vol. 5, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2504-446X/5/4/103

[5] C. Wang, J. Wang, C. Wei, Y. Zhu, D. Yin, and J. Li, "Vision-based deep reinforcement learning of uav-ugv collaborative landing policy using automatic curriculum," *Drones*, vol. 7, no. 11, 2023. [Online]. Available: https://www.mdpi.com/2504-446X/7/11/676

[6] C. Castillo, A. Pyattaev, J. Villa, P. Masek, D. Moltchanov, and A. Ometov, *Autonomous UAV Landing on a Moving Vessel: Localization Challenges and Implementation Framework*, 09 2019, pp. 342–354.

[7] ROS Documentation, "Robot operating system (ros)," accessed: 2024-06-21. [Online]. Available: https://www.ros.org/

[8] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[9] OpenCV Documentation, "Tutorial: Image thresholding," accessed: 2024-07-10. [Online]. Available: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

[10] ——, "Erosion and dilation," accessed: 2024-05-21. [Online]. Available: https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

[11] J. A. Hartigan and P. M. Hartigan, “The Dip Test of Unimodality,” *The Annals of Statistics*, vol. 13, no. 1, pp. 70 – 84, 1985. [Online]. Available: https://doi.org/10.1214/aos/1176346577

[12] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[13] R. O. Duda and P. E. Hart, “Use of the hough transformation to detect lines and curves in pictures,” *Commun. ACM*, vol. 15, no. 1, p. 11–15, jan 1972. [Online]. Available: https://doi.org/10.1145/361237.361242

[14] D. Rieck, “Mechanical, rolling, and global shutter cameras for drones,” 2021, accessed: 2024-06-21. [Online]. Available: https://medium.com/@douglasrieck/mechanical-rolling-and-global-shutter-cameras-for-drones-78460b54e79d

[15] Raspberry Pi Foundation, “Raspberry pi global shutter camera,” accessed: 2024-07-03. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-global-shutter-camera/

List of Figures

List of Tables