Towards Autonomously Landing UAVs on a Moving Platform

Final Year Project Report

Timothy Masters

21969394

Supervisor:

Elliot Nicholls, Adjunct Senior Research Fellow

Director, Stealth Technologies

Professor Thomas Bräunl

UWA, School of Electrical, Electronic and Computer Engineering

Word Count: 6098

19th October 2020

THE UNIVERSITY OF WESTERN AUSTRALIA

Department of Mechanical Engineering

Abstract

Unmanned Aerial Vehicles (UAVs) present myriad opportunities for innovation and automation. Stealth Technologies seeks to integrate UAVs into a ground based autonomous vehicle platform for applications in security, defence, and more. For this innovation to be practical, then there needs to be a reliable and precise way for drones to land on the ground vehicle, ideally without needing to stop every time a drone needs to take off or land.

This project aims to build on past research to develop a robust system for the landing of a small hexacopter UAV on a moving platform, using onboard camera and image-processing to identify and follow the ground vehicle. It is proposed that an optical fiduciary tag system, AprilTag, is used to locate the vehicle and an Extended Kalmann Filter is used to provide accurate state estimation through sensor fusion of the onboard IMU, GPS, and AprilTag position estimate from the camera feed.

In this report, a proposed system architecture is presented, consisting of 3 hardware processor devices – a PixHawk flight controller and a Raspberry Pi 4 mounted to the UAV, and a ground station computer on the vehicle – as well as a number of Robot Operating System (ROS) nodes running on these devices. The viability of AprilTag fiduciary detection system running on various devices and over varying distances using an 8MP camera sensor with a fisheye lens downscaled to 640x480 resolution is explored and applied to the proposed design of a landing algorithm. The current progress towards implementing the entire architecture is shown, and a plan for further work to complete it is put forward.

Acknowledgements

Firstly, I would like to thank Elliot Nicholls for supervising me and allowing me to undertake such an interesting research project. Thanks also for all the advice you gave and the patience you showed me along the way. I would like to thank the rest of the team at Stealth Technologies too – Travis, Chao, Stuart, and Craig - for all the help given along the way, and all the mundane questions answered.

Thanks to Mum and Dad for supporting me through this year and indeed all my education. Cheers to Sahaj for keeping me motivated, and to Sam for sustaining me with many laughs.

Finally, I am grateful to Dad and Amy for giving the time to help proofread this document.

Contents

List of Figures

List of Tables

1 Introduction

The age of automation is here [1]. The robotic workforce is growing exponentially, and as computers become more intelligent, an increasing number of fields are opening up to automation. No longer is the robot limited to a sterile, predictable fabrication plant; cutting edge computing and intelligent systems mean that robotics is increasingly finding real world applications. Riding this wave is Perth based company Stealth Technologies. Leveraging technical expertise in machine learning, robotics, and autonomous driving, Stealth is developing the AxV autonomous electric vehicle platform for applications in security, perimeter testing, defence, and more [2].

1.1 Motivation

In recent years, Unmanned Aerial Vehicles (UAVs) have exploded in popularity, affordability, and utility. Primitive UAVs have been in use since as early as 1916 [3]. It was not until the early 2000s however that the now ubiquitous quadcopter saw a rise to prominence. From around 2005-2010, a number of advances in electronics technology brought about a revolution in micro UAVs, with cheap Lithium ion batteries, Inertial Measurement Units (IMUs), Global Positioning System (GPS) units, and flight controllers leading to a massive surge in popularity [4]. The ease of use the multirotor vehicle system with vertical take-off and landing capabilities, combined with the ability to hover in place precisely, allow for myriad applications. From agriculturalists to military users, the usefulness of this new vehicle has been proved time and time again.

Stealth Technologies aims to utilise this versatility to expand the capabilities of their ground based AxV platform with UAV integration. The UAVs may be equipped with a number of sensors – visual, thermal, LIDAR, and more – which together with the ability to cover rough terrain will greatly increase the effective sensing range of the ground vehicle. Additionally, UAVs may be employed to deliver payloads, for example delivering parcels to online shopping customers. The ability to launch and land potentially swarms of UAVs without stopping would be a massive boon to the usefulness of the AxV platform.

1.2 Objectives

The main objectives of this report are to research the viability and possible methods of integrating a UAV launch and land system into the existing AxV platform. Primary focus is directed at implementing the functionality to launch and land the UAV without stopping the ground vehicle, in particular investigating the capabilities and limitations of the AprilTag fiduciary detection system. This includes looking into hardware and software architectures that will streamline development and have a high chance of leading to an eventual successful product.

2 Literature Review

It must be noted that the landing of quadcopters on moving platforms is not a novel idea; it has been studied and demonstrated before numerous times. However there is little in the way of reliability reports and from what little video footage was found, the landings appeared extremely rough, involving simply shutting off the motors when above the target [5]. Clearly a feasible landing method must be gentle and have proven reliability to be used in a commercial product, to ensure longevity of the quadcopter and minimise maintenance costs.

2.1 Relevant Patents

In 2016, Ford was granted a patent on a UAV system to deploy from the trunk of a car [6]. As reported in the patent application, the primary utility would be in emergency scenarios – the drone could be employed to scout ahead and around the vehicle, finding the optimal, safest path for the car to take. Although as of 2020 Ford has not reported plans to integrate this into

any upcoming vehicles, the interest that such a huge player in the automotive industry has taken in this premise demonstrates the promise it holds.

Online commerce giant Amazon was in 2016 granted a patent for UAVs delivering packages to land on transportation vehicles for temporary transport, to increase range and efficiency of drone-enabled shipping. Whilst this is not exactly the same use case as Stealth, as the patent outlines the use of non-affiliated trucks as "hitchhiking" vessels, rather than UAV deployment from a central vehicle, it shows again that large industry players are showing interest in this area, and here particularly in the landing of UAVs on moving vehicles.

It should be noted that although this and other similar patents do not seem to conflict with Stealth's particular use case, the author recommends appropriate legal counsel be sought to ensure no infringement.

2.2 Scientific Literature

Araar, Aouf, and Vitanov [7] report successfully landing a quadcopter on a moving platform with an average error of 13 cm from the pad at speeds of approximately 2 km/h. They report difficulties with turbulence from the ground effect when landing, which were mitigated by introducing a minimum tracking altitude below which the vehicle steps into an automated landing protocol. This solution may result in worse accuracy performance at high speeds, especially if there is any variation in ground vehicle trajectory.

The primary method of locating the vehicle relative to the UAV was through the visual fiduciary tag system AprilTag, a theme common to many of the papers researched. AprilTag is a library of square, pixelated images similar to a QR code that may be detected by a camera and analysed to generate an estimated pose relative to the camera. A gimballed camera was used to track the tags. An array of progressively smaller AprilTags was used (Figure 1) so that the UAV may detect them from afar, but also keep some in field of view when coming in very close for landing. This technique could be easily adapted and used to improve both maximum and minimum detection range, though fewer tags would probably prove sufficient. The Extended Kalman Filter (EKF) was also compared with an Extended H∞ filter for state estimation, and it was found that the EKF outperformed the Extended H∞ in all situations. These filters are used to combine the data from multiple sensors into one best estimate of position.

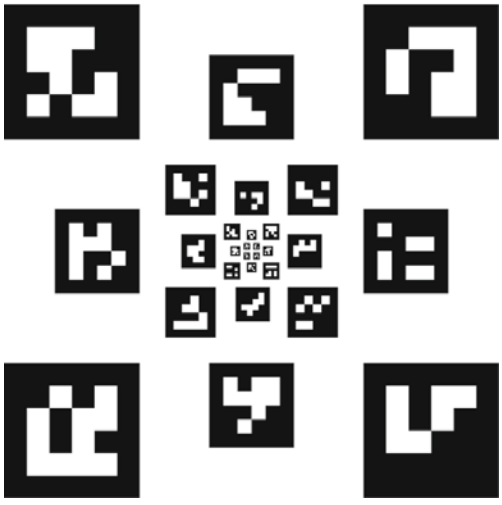

Figure 1: AprilTag array used on the landing pad by Araar, Aouf, and Vitanov [7]

Herisse et al. [8] report successful landing on a moving platform at walking speed using an "optical flow" method. This method does not use any fiduciary markers like AprilTags; it instead measures the speed at which the object it is looking at is moving relative to itself, and from that

deduces some positional data. The algorithm, while interesting, is both extremely complex to implement, and computationally expensive. With the availability of more practical options, this route was not explored further.

Borowczyk et al. [9] report successful landing of a DJI Matrice 100 quadcopter at up to 50km/h using a single large AprilTag, and transmitted GPS readings from the ground vehicle. A gimbal mounted camera was used to track the AprilTag from afar, and a fixed wide-angle lensed camera was relied on once the drone was close to the target. The combination of these two camera types adds significant complexity, but also allows for the best of both options – long detection distance from the narrow Field of View (FoV) gimballed camera, and up close reliable detection from the wide angle lensed camera.

A Proportional Navigation (PN) controller was employed to approach the ground vehicle. Once the UAV was close enough to the vehicle, this was switched to a Proportional Derivative (PD) controller for the final landing phase. As seen in the video of their demonstration, [5] the landing is quite rough at higher speeds as the quadcopters maximum speed was approached. This indicates that landing speeds should likely be kept well under the UAV's top speed to ensure successful landings.

Feng et al. [10] report on the somewhat novel approach of landing on a moving platform in an entirely simulated environment. Using a Kalman Filter on simulated noisy measurements, they were able to successfully land at up to 12 m/s by tracking an AprilTag with a gimballed camera. The control algorithm was Model Predictive Control (MPC), which is akin to a PID controller, but with the ability to predict changes in the system dependant variables caused by changes in the independent variables.

Olivares-Mendez, Mondragon, and Campoy [11] report on landing a single-rotor UAV on a stationary platform using "fuzzy logic" control. Fuzzy control works with probabilities of measurements being correct, rather than assuming true or false. This approach gave limited success, with average landing errors of 0.73m. This is unacceptably high for our application, so will not be considered.

Baca et al. [12] report landing a hexacopter on a ground vehicle moving at 15 km/h. A custom fiduciary marker was used rather than AprilTags, as this was part of an international competition. This marker provided much the same utility as AprilTags, so there is little point in investigating a customised approach. A fixed downwards facing camera with a fisheye lens was used. This required an additional processing step to undistort the image, but increased field of view substantially. A laser rangefinder was also used to great effect to assist in the landing phase. An Unscented Kalman Filter (UKF) was used for state estimation, and a nonlinear controller used to make the UAV track the predicted path of the ground vehicle. The nonlinear control is reported to give faster response than the linear PID approach, meaning the UAV can react faster to unexpected turbulence or wind interference.

Lee, Jung, and Shim [13] report using a colour based tracking system to land on a slow moving vehicle. Despite them finding some success in this method, the reliance on a unique colour to track introduces inherent limitations. A red box was used as the tracking marker, and it is noted that the presence of any other red objects in the scene may easily throw off the UAV. This severely limits its use for outdoor applications with inconsistent lighting and weather conditions, and thus it was decided not to pursue this method of tracking the ground vehicle.

Kim et al. [14] also utilised a colour thresholding based tracking system to land on a slow moving target. State estimation was done with a UKF, and a non-linear controller was used for the landing approach.

Paris, Lopez, and How [15] achieved landing on a platform moving at 4.7 km/h in turbulent wind conditions induced with leaf blowers, using a vertical AprilTag setup. Model Predictive Control is used to plan the landing trajectory, and a Boundary Layer Sliding Control (BLSC)

algorithm is implemented to combat the turbulence present. Due to the large amounts of turbulence created by fast moving vehicles, this may prove a valuable approach.

The approaches utilised by the various papers are summarised in Table 1 for the readers convenience.

TABLE 1: COMPARISON OF METHODS AND RESULTS OF PREVIOUS RESEARCH

Paper	**Ground Vehicle Speed (km/h)**	**Average Error (cm)**	**Detection method**	**State Estimator**	**Control Algorithm**
Araar, Aouf, and Vitanov [7]	2	13	AprilTag array, gimballed camera	EKF	PID
Herisse et al. [8]	Not reported	Not reported	Optical Flow	Not reported	PID
Borowczyk et al. [9]	50	Not reported	AprilTag, gimballed and fisheye cameras	EKF	PN and PID
Feng et al. [10]	43 (simulated)	15	AprilTag, gimballed camera	KF	MPC
Olivares-Mendez et al.	0	73	Custom fiduciary marker, fixed camera.	EKF	Fuzzy control, PD
Baca et al. [12]	15	Not reported	Custom fiduciary marker, fixed camera.	UKF	Trajectory prediction, Nonlinear control
Lee, Jung, and Shim [13]	Not reported	Tracking only, no landing	Colour thresholding, fisheye camera	EKF	PID
Kim et al. [14]	Not reported	Not reported	Colour thresholding, fixed fisheye camera	UKF	Nonlinear control
Paris, Lopez, and How [15]	4.7, plus induced wind	Not reported	AprilTag, static camera	EKF	MPC, BLSC

2.2 Proposed Approach

Based on the successes of the examined literature, it was decided to use AprilTag fiduciaries as the primary method of locating the landing target relative to the drone. Similarly, the best method for combining multiple sensor measurements (IMU, GPS & AprilTag detection) into one best position estimate was found to overwhelmingly be the EKF.

The landing process can generally be broken into 4 phases:

1. Autonomous flight using GPS navigation to the coordinates of the ground vehicle.
2. Once visual contact with the fiduciary is made, switch to a second control algorithm to position the drone above the vehicle using the AprilTag pose information.
3. Once near the vehicle, begin final descent phase to just above landing pad.
4. When drone is sufficiently close to the landing pad, motors are abruptly shut off.

Step 1 is trivial to implement, as all flight controllers have waypoint mission functionality inbuilt. For step 2, the accuracy of the algorithm is less important than the speed at which the UAV can close the distance to the ground vehicle. Proportional Navigation involves plotting a course such that the target stays at a constant bearing from the UAV. This in theory results in the shortest path to the target, as long as the target maintains a constant course. Due to the simplicity of the PN algorithm and its proven effectiveness [9], this is the leading candidate for step 2.

Step 3, the final descent, is the most critical phase. Due to the variety of algorithms used in the literature, and the lack of objective means of comparison (most did not include average error in landing), it is difficult to make an initial educated choice based off this data alone. The two methods that stand out are the simple PID control, and Model Predictive Control. PID would be simple to implement, and has proven success up to 50 km/h. Therefore this the leading candidate for an initial implementation of step 2. If this does not provide sufficient accuracy and reliability of landing, then the MPC and BLSC algorithms can be explored.

Shutting off the motors completely for the final landing may not seem the gentlest or most reliable landing method, but the challenges of landing on a moving platform necessitate it. A normal stationary landing involves a slow, steady descent for the final portion. When the flight controller senses contact is made with the ground (drone is no longer descending), the motors are slowly powered down. The pitch at which the drone must fly to maintain forward momentum means that a steady touchdown cannot be achieved on a moving landing; as soon as the front legs touch down, the pitch of the drone will change and thus lose speed, missing contact with the landing pad. Thus the best option is to achieve a very close approach to the target landing zone in step 3, then simply shut off the motors.

3 Hardware Selection

It was decided that before investing in a new UAV, existing hardware owned by UWA would be utilised to explore the viability of the concept without a large upfront investment.

3.1 UAV and Flight Controller

Two options of UAV were available to choose from. These were the DJI Matrice 100 quadcopter, and DJI F550 hexacopter. Both of these platforms have been demonstrated to be capable for this task in previous research [9] [10] [12]. The flight controller is a piece of hardware that runs the innermost control loops of the UAV control software. A defining difference between the Matrice 100 and the F550 is the flight controller used. The Matrice 100 uses a proprietary DJI flight computer integrated into the drone. This runs closed source software, designed to be controlled by the DJI Onboard SDK [16]. The disadvantage of this is that migrating to any other platform in the future would require a redesign as this SDK is designed exclusively for DJI products, primarily the Matrice series.

As the final Stealth Technologies UAV will have very sensitive applications potentially including private security, military, and national security it is worth also considering the potential danger of using closed-source foreign made products. In 2017, the U.S. Army issued a ban on all DJI products [17], and more recently, the U.S. Department of the Interior grounded its entire fleet of 800 Chinese made drones. There are fears that the data collected by these products may be making its way back to the People's Republic of China, where it can be accessed by the Chinese Communist Party under a national security law passed in 2017 [18].

Figure 2: UWA's DJI F550 Hexacopter

For these reasons, we decided to move forward with open-source software where possible and the DJI F550 hexacopter (Figure 2). Although the F550 frame is a DJI product, it has a PixHawk flight controller onboard running open-source software – either PX4 stack or ArduCopter. Many other suitable drones that could be purchased in the future also run this software – for example the VOXL m500 Development Drone, a U.S. developed UAV.

3.2 Onboard Computer

The flight controller is designed only to run the control algorithms essential to the drone's flight, and very little else. Therefore in order to perform any complex tasks not covered by the autopilot software – such as image processing, communication with a ground station, and the landing algorithm itself – there must be another processor onboard issuing commands to the flight controller. There are a variety of potentially suitable embedded processors for this task. The main computationally intensive task that needs to be accomplished by the processor is running the AprilTag algorithm at a sufficiently fast rate. The literature suggested a rate of 15-30 Hz [7] [9] [10], so 15 Hz was used a minimum acceptable detection rate criterion.

The F550 drone was already equipped with a Raspberry Pi 3B mounted to the bottom of the frame, so this was the first processor tested. At a resolution of 640x480, the Pi 3B was able to run the AprilTag detection algorithm at a rate of 9 Hz. As this was below the 15 Hz criterion decided upon previously, other solutions were investigated.

The NVIDIA Jetson product line was a very promising candidate. This includes the Jetson Nano and Jetson TX2, which contain similar CPUs to the Raspberry Pi 3 but with additional 128 and 256 core GPUs respectively, which should greatly improve the speed of any parallelisable process [19]. The NVIDIA Isaac SDK is an engine developed specifically for computationally expensive embedded processing utilizing the Jetson product range, for applications in edge AI and computer vision. There exists an Isaac "nodelet" for GPU-accelerated AprilTag detection, meaning that implementation on this platform is theoretically achievable. However, in practice the nodelet would only accept images in a YUYV format, whereas our camera would output an RGB format image stream. Upon attempting to modify the nodelet, it was discovered that the entire Isaac SDK is closed-source, despite being advertised as open-source. Additionally, the NVIDIA program used to flash the Operating System continually installed unworking OpenCV versions, so we were unable to create another nodelet to convert between RGB and YUYV image formats.

It is worth mentioning the possibility of offloading the resource-intensive image processing task to the ground vehicle, which already has powerful computational abilities. This would require compressing the image-stream and sending it over a Wi-Fi link to the vehicle, and then receiving the detected AprilTag poses over the same link. This was considered, but due to the potential instabilities and latencies involved in the Wi-Fi transmission, it was decided that the most reliable results would be found with an onboard processor.

Finally it was decided to upgrade the Raspberry Pi 3B to a Raspberry Pi 4. In some benchmarks, the model 4 has more than double the processing power as the 3B [20]. The core AprilTag algorithm ran with a steady detection rate of 19 Hz on this platform, passing the 15 Hz criterion.

3.3 Camera

There were two broad categories of camera type to choose from – ordinary or fisheye lens. An ordinary camera has a narrow field of view, and thus higher angular resolution. The fisheye lens greatly increases field of view, but at the cost of a lower angular resolution as each pixel covers a larger angle. The IMX219 camera module has a CSI interface, a port that both the Raspberry Pi and Jetson Nano had available, so this was selected. Without the fisheye lens option, the IMX219 module has a diagonal FoV of approximately 80°. With the fisheye lens, the diagonal FoV is more than doubled to 175°, seen in Figure 3.

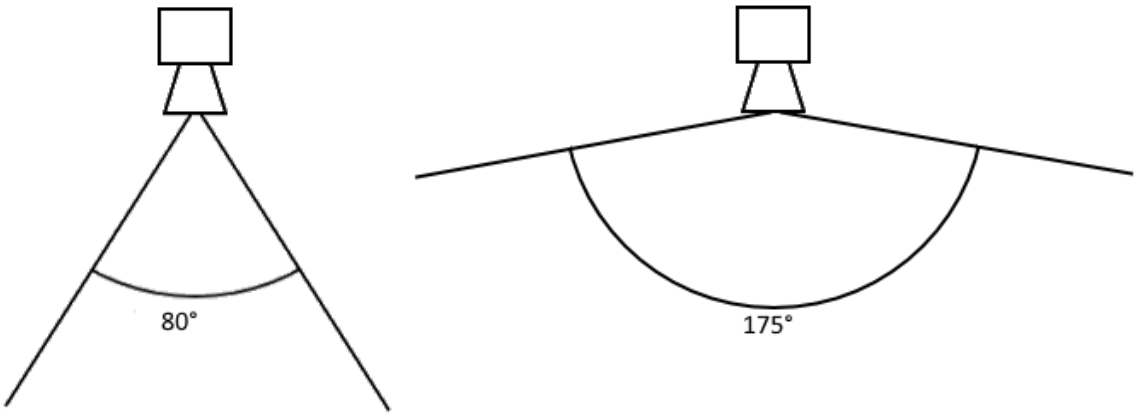

Figure 3: Comparing diagonal field of view of Raspberry Pi Camera V2 with and without fisheye lens

The advantage of a very wide FoV lies in the robustness of the tag detection up close. The closer the UAV is to the target, the more a slight sideways perturbation will change the angle between the camera and the target. With a small FoV camera, this may result in the AprilTag moving outside the camera's view frustum. This can be compensated for with a gimballed camera that actively tracks the AprilTag [9], at the cost of extra complexity in both the mechanical system and the software that applies the gimbal transformation to the detected AprilTag's pose.

Fisheye lenses also introduce significant distortion into the image, especially around the edges. This may cause issues with AprilTag detection and make the pose estimation less accurate. If distortion proves problematic, then undistort transformations can be applied to the image, using camera characteristics found by calibration programs [21]. Thus for the sake of simplicity and robustness, a fixed downwards-facing fisheye lensed IMX219 camera was chosen. In hindsight, a USB interfaced camera should have been used. This is because the CSI port, although officially

supported by all the embedded systems we used it on, had far less support in software and drivers than cameras that used the USB port. This is discussed in more depth in sections 3.2 and 4.3.

4 System Overview & Software Implementation

4.1 Software Selection – ROS

Robot Operating System (ROS) is a set of libraries and tools used in a wide variety of robotics applications [22]. Development is highly streamlined with pre-made modular "nodes" that can be run and utilized completely independent of user-written code. Each node is designed to work on a specific task and communicates with others through "topics". There are pre-existing nodes that will provide most of the functionality required from the software. Specifically, there is an AprilTag node that can perform real-time detection of AprilTags on an image stream published to a topic, and a MAVROS node which can provide two-way serial communication with the PixHawk flight controller through the MAVLink messaging protocol. Additionally, the Stealth Technologies ASV runs ROS, meaning establishing communication would be trivial once connected by Wi-Fi. All of this made ROS the obvious choice going forward.

4.2 System Overview

The proposed overview for the entire system can be seen in Figure 4, consisting of 5 discrete hardware components, and a number of abstracted software components within.

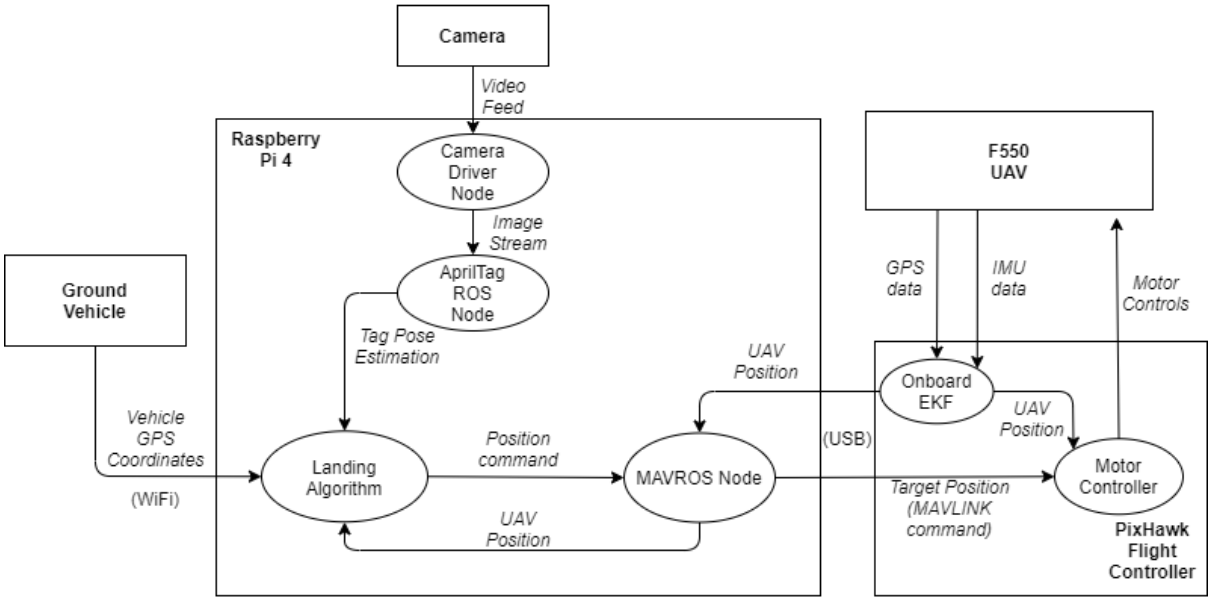

Figure 4: Overview of Proposed System Architecture

The PixHawk and F550 components are already implemented, as the PX4 flight controller software handles all communication with the UAV and internal calculations. The main area of focus is setting up the software that will be running on the Raspberry Pi. The AprilTag and MAVROS nodes need to be installed and set to read and publish on the correct topics. The MAVROS node also needs to be able to detect the Pixhawk device over USB and send and receive messages. This meant creating a Udev rule to assign the USB device a specific name and giving that name to the MAVROS node to use as the target address. The user account also had to be added to the dialout group to allow access to the USB port as a serial device.

4.3 Camera Driver

Due to the recent release date of the Raspberry Pi 4 (2019) and its radically different processor from previous generations, it does not support as much legacy software as the 3B. The only Ubuntu release fully supported and readily installable on the Pi 4 is 20.04. This meant that rather than using the older and more mature ROS Melodic as was done on the Raspberry Pi 3B, which is supported on Ubuntu 18.04, we were forced to use ROS Noetic. ROS Noetic was only released in May 2020, meaning it has far fewer packages available right out of the box. Fortunately the AprilTag and MAVROS nodes were available, but the RaspiCam node used to get the camera feed on the Pi 3B was not. This meant a camera driver node had to be written.

Leveraging online learning & tutorials [23], a simple node was created with OpenCV in C++ to publish the image stream and the camera info topic to the correct places, both required inputs for the AprilTag node. Unfortunately this node does not support any video modes other than 640x480, as calling the OpenCV functions to change camera resolution would instantly crash the node.

4.4 AprilTags

In order to maximise both detection range and up-close fidelity, it is proposed to use an AprilTag array similar to that shown in Figure 5 on the ground vehicle. The larger AprilTag (approx. 30cm) can be detected from further away providing increased range, whilst the small tag (approx. 8cm) allows for greater detection reliability and precision up close [7], as it will be less distorted on a fisheye lens, and less likely to be partially off frame.

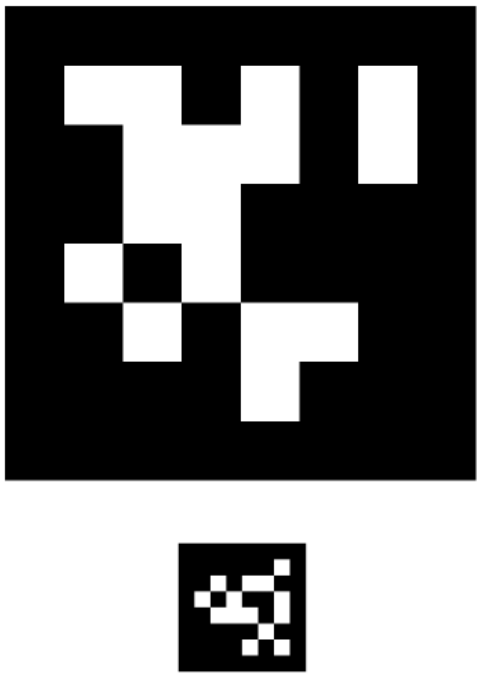

Figure 5: Proposed AprilTag Array

5 AprilTag Detection & Limitations

Angular resolution θ of each tag pixel is inversely proportional to distance d from the camera and linearly proportional to the width w of the AprilTag.

$$\theta \propto \frac{1}{d}, \qquad \theta \propto w$$

EQUATION 1

Therefore assuming detection will fail at a certain angular resolution threshold, it is expected that the maximum detection distance D should increase linearly with tag width:

$$D \propto w$$

EQUATION 2

In order to validate this expected relationship, AprilTags of various sizes were tested to determine their maximum detection distance from the camera. This was taken as the distance at which the detection would first begin to flicker as the AprilTag was slowly moved away from the camera. Results from initial testing using the IMX219 sensor with a fisheye lens running at 640x480 resolution, shown in Table 2 and Figure 6 confirms a linear relationship as expected, but with a small offset. This could be due to the fisheye distortion behaving differently at very close distances. As we are not concerned with using very small AprilTags at very close distances, this should not pose a problem.

AprilTag Size (cm)	**Maximum Reliable Detection Range (m)**
8.0 ± 0.05	1.9 ± 0.05
14.0 ± 0.05	3.1 ± 0.05
30.3 ± 0.05	6.1 ± 0.05

TABLE 2: APRILTAG DETECTION DISTANCE VS SIZE, USING FISHEYE LENS AT 640X480

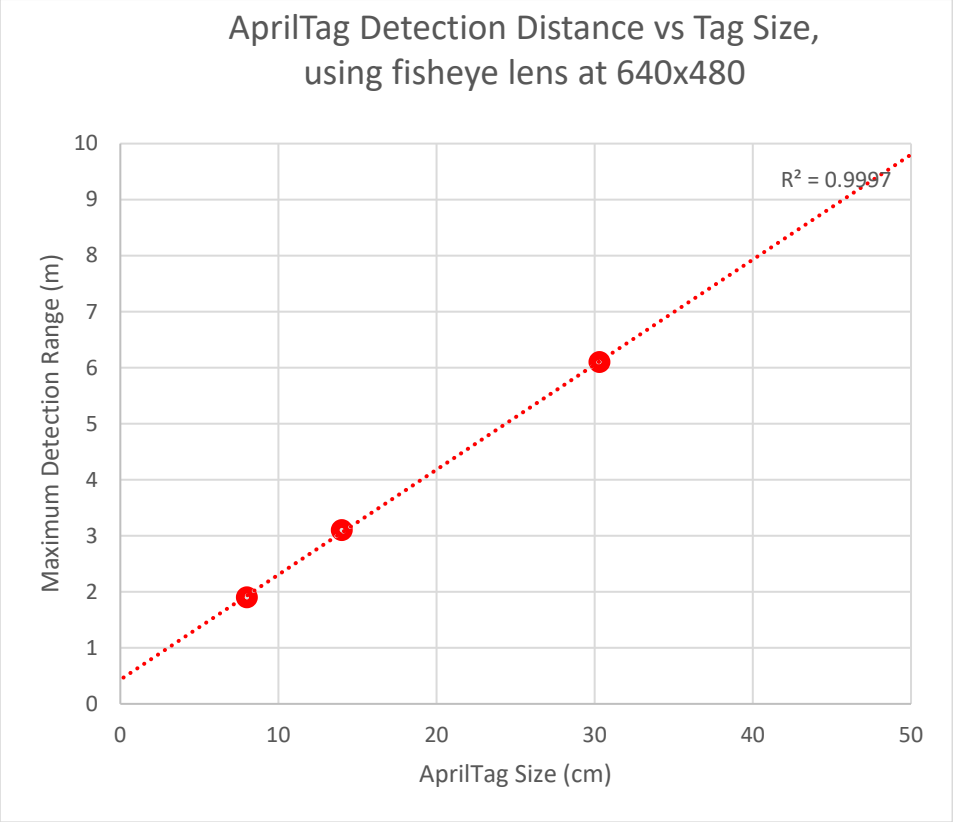

FIGURE 6: APRILTAG DETECTION DISTANCE VS SIZE, USING FISHEYE LENS AT 640X480

Using the relationship found, the minimum AprilTag size required for a certain desired maximum detection range may be estimated. Note that the usual limitations and uncertainties on extrapolation are in play here, but given the very high $R^2 = 0.9997$ value, this relationship should provide a sufficiently accurate estimate of required tag size.

Should longer detection range be required than the largest feasible AprilTag size permits, then either a smaller FoV camera can be selected, or the AprilTag algorithm can be run at a higher resolution. The effect of both can also be achieved by increasing the feed resolution, then cropping the image edges to the desired size. Increasing the resolution of the image scales the processing time required approximately linearly with the number of pixels in the image [24], so in theory a doubling of vertical and horizontal resolution would quarter the detection framerate. Assuming a similar angular resolution per pixel to distance relationship as observed in Figure 6, this would mean using resolution to increase the detection distance D by a factor would decrease framerate F by the square of that factor:

$$D \propto \frac{1}{F^2}$$

EQUATION 3

As the current detection rate is 19 Hz, in order to stay above the 15 Hz detection rate criterion, the resolution cannot be increased substantially if the Raspberry Pi 4 is to be kept as the processor. Using different processors would allow higher resolutions to be explored.

It is noted that the fisheye distortion is greatest around the edges of the image, but the AprilTags were detectable right up to the edge of the image, as seen in Figure 7. Thus the distortion should not impact detection ability sufficiently to warrant concern. The distortion may introduce an error into the estimated poses, especially near the edge. If this is found to be the case, further effort should be directed towards undistorting the image before passing to the AprilTag node.

Figure 7: AprilTag being detected near edge of FoV of fisheye lensed camera

6 Preliminary Flight Testing

6.1 Offboard Control with MAVROS

Before testing AprilTag tracking and landing onboard the UAV, it was necessary to first demonstrate simple offboard control commands. This involved using MAVROS to send the commands to the PixHawk using the MAVLink protocol and ensuring that the UAV performed the command as requested.

Connecting both the onboard Raspberry Pi and a laptop to a portable Wi-Fi network allowed Secure Shell (SSH) access onto the Pi to run a ROS node, which sent a take-off and position hold at 2m above take-off point command to the PixHawk. Upon receiving the command the UAV successfully took off, but extreme instability in the form of rapid oscillations and drift was observed when attempting to hold at the requested position. A recording demonstrating the instability can be viewed at https://photos.app.goo.gl/E6m1Kbb8JQDWeBYm6.

A common cause for instabilities is incorrect calibrations in the flight controller. The IMU and motor Electronics Speed Controllers (ESCs) were recalibrated, but the instability persisted. The flight controller PX4 software was replaced with the ArduCopter software, but this again did not resolve the instability.

Unfortunately, this instability made it infeasible to continue trials with the F550. In order to fix this and enable the F550 to continue to be used as a research platform by UWA in the future, it is recommended that the flight controller is replaced with the latest PixHawk unit. The internal IMU of the PixHawk being faulty is judged to be the most likely cause of this issue. However, this is not guaranteed to fix the problem; it may be that there is a problem with some other hardware – the frame, propellors, motors, or a combination of the above.

7 Conclusion

Although disappointingly an AprilTag assisted automated landing was not achieved, significant progress has been made in devising and implementing a robust and extendable framework for the control and vision processing of a multirotor device using ROS. The viability of multiple embedded computing platforms was evaluated, before settling on Raspberry Pi 4 and confirming its ability to run the AprilTag detection algorithm at above 15 Hz.

The capabilities of the AprilTag detection algorithm on a wide angle 640x480 image were investigated, and it was found that detection distance correlated linearly with tag width. This relationship can be used in the future to determine required AprilTag sizes depending on the distance requirements of the chosen landing algorithm.

It is recommended that the UWA F550 platform is upgraded to the newest PixHawk flight controller, in an effort to eliminate the extreme instabilities observed in the test flight. It is also recommended that Stealth Technologies invests in a new UAV platform as soon as possible, so that work may continue on this project.

8 Plan for Future Work

Stealth Technologies has elected to purchase the VOXL m500 Development Drone for continuation of this work. The m500 has a SnapDragon 820 mobile processor for low-power high performance computing, an integrated flight controller running the PX4 Stack, and ROS compatibility. This means that much of the work done so far should be transferrable to the new platform. The main outstanding work is deciding on and implementing the actual landing algorithm. An Extended Kalman Filter (EKF) needs to be run on the drone to fuse the position estimation given by the flight controller with the detected AprilTag pose to create a relative position estimate relative to the ground vehicle target. From the literature, there are then several approaches to the landing algorithm that could be employed. These include a combination of proportional navigation to approach the vehicle and a PID to land, or a model predictive controller with boundary layer sliding control system. These control algorithms need to be analysed in depth to determine which will give the best results for our system. Once the best approach is selected, it will need implementation as a ROS node.

Aside from the actual landing algorithm, there are hardware features that need to be considered too. There will need to be an automated system for securing the UAV to the vehicle after landing, and an automated method of charging the UAV onboard the vehicle so that it may embark on multiple missions after recharging. These solutions will need to be devised once the precision of the landing sequence is known, so they may be tailored to fit the uncertainties associated with it.

These tasks are expected to be continued by the author over the 2020-2021 Summer period, as well as worked on by a fellow MPE student who is currently also working on a thesis with Stealth Technologies.

Bibliography

[1] N. Wyman, "Forbes," 25 March 2019. [Online]. Available: https://www.forbes.com/sites/nicholaswyman/2019/03/25/the-age-of-automation-is-here-how-to-navigate-the-new-world-of-work/#548fb9024227. [Accessed October 2020].

[2] Stealth Technologies, "Stealth Technologies," 2020. [Online]. Available: https://www.stealthtechnologies.com.au/. [Accessed 2020].

[3] A. J. P. Taylor, Jane's Book of Remotely Piloted Vehicles, 1977.

[4] G. Hoffmann, D. Rajnarayan, S. Waslander, D. Dostal, J. Jang and C. Tomlin, "The Stanford Testbed of Autonomous Rotorcraft for Multi Agent Control (STARMAC)," in *23rd Digital Avionics Systems Conference*, Salt Lake City, 2004.

[5] A. P.-V. Nguyen, "Autonomous Landing of a Multirotor Aerial Vehicle on a High Velocity Ground Vehicle," [Video]. https://www.youtube.com/watch?v=ILQqD2xQ4tg&feature=youtu.be, 2016.

[6] Ford, "Google Patents," 9 August 2016. [Online]. Available: https://patents.google.com/patent/US9409644B2/en. [Accessed 2020].

[7] O. Araar, N. Aouf and I. Vitanov, "Vision Based Autonomous Landing of Multirotor UAV on Moving Platform," *Journal of Intelligent & Robotic Systems,* p. 369–384, 2017.

[8] B. Herisse, T. Hamel, R. Mahony and F.-X. Russoto, "Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow," *IEEE Transactions on Robotics,* pp. 1-13, 2011.

[9] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussié and J. LeNy, "Autonomous Landing of a Multirotor Micro Air Vehicle on a High Velocity Ground Vehicle," *IFAC-PapersOnline,* p. 10488–10494, 2017.

[10] Y. Feng, C. Zhang, S. Baek, S. Rawashdeh and A. Mohammadi, "Autonomous Landing of a UAV on a Moving Platform," *Drones,* pp. 2-34, 2018.

[11] "Autonomous Landing of an Unmanned Aerial Vehicle Using Image-Based Fuzzy Control," *IFAC RED-UAS,* pp. 79-86, 2013.

[12] T. Báča, P. Štěpán, V. Spurný, D. Hert, R. Pěnička, M. Saska, J. Thomas, G. Loianno and V. Kumar, "Autonomous Landing on a Moving Vehicle with an Unmanned Aerial Vehicle," *Journal of Field Robotics,* 2019.

[13] H. J. S. Lee and D. H. Shim, "Vision-based UAV Landing on the Moving Vehicle," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Arlington, VA, 2016.

[14] J. Kim, Y. Jung, D. Lee and D. H. Shim, "Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera," *2014 International Conference on Unmanned Aircraft Systems (ICUAS),* 2014.

[15] A. Paris, B. Lopez and J. How, "Dynamic Landing of an Autonomous Quadrotor on a Moving Platform in Turbulent Wind Conditions," Massachusetts, 2019.

[16] DJI, "DJI Onboard SDK Documentation," 2018. [Online]. Available: https://developer.dji.com/onboard-sdk/documentation/introduction/homepage.html. [Accessed July 2020].

[17] G. Mortimer, "sUAS News," 4 August 2017. [Online]. Available: https://www.suasnews.com/2017/08/us-army-calls-units-discontinue-use-dji-equipment/. [Accessed September 2020].

[18] M. McNabb, "Drone Life," 14 August 2020. [Online]. Available: https://dronelife.com/2020/08/14/the-ban-on-chinese-drones-part-2-what-happens-next/. [Accessed September 2020].

[19] NVIDIA Corporation, "NVIDIA Website," 2020. [Online]. Available: https://www.nvidia.com/en-au/autonomous-machines/embedded-systems/. [Accessed June 2020].

[20] G. Halfacree, "Medium," 24 June 2019. [Online]. Available: https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-4-73e5afbcd54b. [Accessed July 2020].

[21] OpenCV, "OpenCV Documentation," 2020. [Online]. Available: https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html. [Accessed 2020].

[22] Open Robotics, "ROS," 2020. [Online]. Available: https://www.ros.org/. [Accessed 2020].

[23] A. Addison, "Automatic Addison," 27 June 2020. [Online]. Available: https://automaticaddison.com/working-with-ros-and-opencv-in-ros-noetic/. [Accessed September 2020].

[24] April Robotics, "AprilTag Repository," 2020. [Online]. Available: https://github.com/AprilRobotics/apriltag. [Accessed 2020].

Appendix 1. Camera Driver ROS Node

```
#include <cv_bridge/cv_bridge.h>
#include <image_transport/image_transport.h>
#include <opencv2/highgui/highgui.hpp>
#include <ros/ros.h>
#include <sensor_msgs/image_encodings.h>
#include <camera_info_manager/camera_info_manager.h>

// Author: Addison Sears-Collins
// Website: https://automaticaddison.com
// Description: A basic image publisher for ROS in C++
// Date: June 27, 2020

// Expanded by Timothy Masters
// Masters Research Project, University of Western Australia
// September 2020

int main(int argc, char** argv)
{
    ros::init(argc, argv, "video_pub_cpp");
    ros::NodeHandle nh;  // Default handler for nodes in ROS

    // 0 reads from your default camera
    const int CAMERA_INDEX = 0;
    cv::VideoCapture capture(CAMERA_INDEX);
    if (!capture.isOpened()) {
      ROS_ERROR_STREAM("Failed to open camera with index " << CAMERA_INDEX <<
"!");
      ros::shutdown();
    }
    capture.set(cv::CAP_PROP_FRAME_WIDTH, 640.0);
    capture.set(cv::CAP_PROP_FRAME_HEIGHT, 480.0);
    capture.set(cv::CAP_PROP_FPS, 40.0);

    /** camera calibration information */
    camera_info_manager::CameraInfoManager cinfo(nh);
        // get current CameraInfo data
    cinfo.setCameraName("Cam0");
    cinfo.loadCameraInfo("file:///home/stealth/.ros/camera_info/Cam0.yaml");


    sensor_msgs::CameraInfo ci(cinfo.getCameraInfo());

    // Image_transport is responsible for publishing and subscribing to Images
    image_transport::ImageTransport it(nh);
    auto image_pub = it.advertiseCamera("camera/image_rect", 1);
    // Publish to the /camera/image_rect topic
```

```
    cv::Mat frame;//Mat is the image class defined in OpenCV
    sensor_msgs::ImagePtr msg;

    ros::Rate loop_rate(30);

    while (nh.ok()) {
      if(capture.isOpened()){
          // Load image
          capture.read(frame);
          // Check if grabbed frame has content
          if (frame.empty()) {
            ROS_ERROR_STREAM("Failed to capture image!");
            ros::shutdown();
          }
          // Convert image from cv::Mat (OpenCV) type to sensor_msgs/Image
(ROS) type and publish
          msg = cv_bridge::CvImage(std_msgs::Header(), "bgr8",
frame).toImageMsg();
          //pub_frame.publish(msg);
          image_pub.publish(*msg,ci);

          //cv::imshow("camera", image);//display camera feed direct to screen
          cv::waitKey(1); // Display image for 1 millisecond

          ros::spinOnce();
          loop_rate.sleep();
        }
    }

    // Shutdown the camera
    capture.release();
}
```