Autonomous 3D Indoor Mapping Using Unmanned Aerial Vehicles

Undergraduate Student Project

by

Louie Isaniel Cachola Henson
2014-30227
B.S. Computer Engineering

Adviser:

Professor Charleston Dale Ambatali
Roxanne P. De Leon, ECE, MTM

University of the Philippines, Diliman

January 2021

Abstract

Autonomous 3D Indoor Mapping Using Unmanned Aerial Vehicles

Multirotor unmanned aerial vehicles (or drones as they are more commonly known) are quick and versatile, often proving to be a great asset for various purposes. It is because of this that a host of flight controllers have been developed using open-source technology

The project aims to provide search and rescue personnel a means to efficiently map out an area, lending them a smart tool for better planning and risk assessment. The goal of this project is to develop a drone that can autonomously provide the user with an accurate 3D map of indoor spaces. This would require a robust flight controller capable of accurate simultaneous localization and mapping (SLAM) and efficient 3D navigation and exploration. Performance of the UAV will be assessed in terms of exploration speed and mapping precision.

Contents

List of Figures

Chapter 1

Introduction

Unmanned aerial vehicles continue to be integrated into increasingly diverse environments and fields. From photography and cinematography, all the way to military applications. Similarly, mapping drones have found uses in fields such as archaeology and architecture, all the way to disaster risk reduction and management. They are especially useful for reducing the amount of risk that personnel experience by flying to places that pose a significant hazard to the personnel [1]. The problem, however, lies in mapping out indoor spaces, especially those where flight may be restricted by narrow paths and obstacles. Maneuvering through small spaces can prove to be difficult even for experienced pilots and could lead to a crash. Crashes are costly as the worst of them could require the replacement of the entire UAV. Furthermore, UAV flight times are restricted by their battery capacity. Hence, in order to maximize the use of the UAV, there must be a way to explore unknown spaces efficiently. One possible solution to this is the development of a mapping UAV with autonomous navigation and exploration functions. By limiting the amount of ways that the pilot can influence the actual movement, the risk of human error is limited [2]. Furthermore, we can employ various algorithms for more efficient exploration, as will be explored further in this document.

The development of a flight controller system capable of autonomous 3D indoor mapping can be divided into the development of several components; the flight controller board, the SLAM (simultaneous localization and mapping) algorithm, and the autonomous navigation function.

1.1 Flight Controller Design

The flight controller is, as its name suggests, the central device that controls how the UAV moves and reacts to certain stimuli. The flight controller contains multiple sensors including (but not limited to) a gyroscope, accelerometer, and barometer. These sensors allow the flight controller to get a sense of its orientation in 3D space. The flight controller will also be responsible for communicating with a base station and processing commands correctly. In order to make the UAV easy to mass produce and more affordable, production cost was prioritized while designing the flight controller. Several methods for designing the flight controller were found but one was seen to be the most cost-effective and flexible solution. Establishing a well functioning flight controller is important as it is essentially the cornerstone of the UAV and will influence work on both the autonomous navigation functions as well as the simultaneous localization and mapping (SLAM) functions.

1.2 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) is the process where a machine maps out its immediate environment and gets a sense of where it is in that environment. This is especially useful for autonomous exploration as it allows the UAV to formulate an obstacle-free trajectory to its target. Mapping often makes use of a lidar sensor [3] or one or more cameras[**OrbSlam2**] depending on the level of accuracy and speed that is required.

1.3 Autonomous Navigation

Autonomous navigation is the process where a machine navigates to a target location while avoiding obstacles. This is often used in tandem with SLAM and path finding algorithms such as Dijkstra's algorithm or even neural networks [3]. Methods of navigation include global and local navigation however Dowling et al. note that the best results can be found by combining the two [3].

Chapter 2

Review of Related Work

2.1 Flight Controller Board Design

As building a flight controller from scratch would prove to be akin to reinventing the wheel, a premade flight controller will be used as a base that this project will build upon. The base flight controller will be responsible for communicating with the electronic speed controllers (ESCs) and controlling the motors on a more basic level [4]. The corresponding diagram for the system is represented by figure 2.1. The flight controller makes use of a microcontroller that is connected to an inertial measurement unit (IMU). The

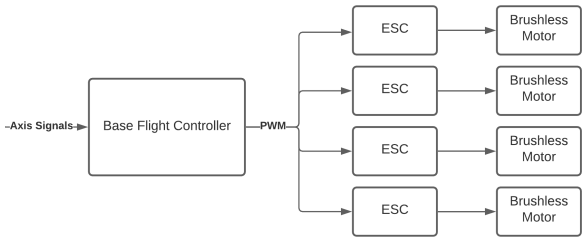

Figure 2.1: Diagram of base flight controller connected to ESCs and motors

IMU includes an accelerometer and gyroscope that allow the flight controller to discern its orientation in 3-dimensional space. Based on these readings, the flight controller will update the ESCs with the proper outputs. Based on figure 2.1, we will essentially be treating the base flight controller as a black box. The inputs would be a PPM signal containing the throttle, yaw, pitch, and roll signals [4]. The Pixhawk, another family of flight controllers, make use of MAVLink (Micro Air Vehicle Communication Protocol)

to deliver the necessary signals into the flight controller. MAVLink makes use of a UART connection to send instructions to the flight controller. S. Al-Kadhim demonstrates how to makes use of a Raspberry Pi to communicate with the Pixhawk [5]. This allows them to interface with the UAV in a more sophisticated manner and make use of the computer's higher processing power. Other known applications of onboard computers on UAVs is the application of neural networks for autonomous flight [6]. Meanwhile, Dowling et al. use a Raspberry Pi coupled with a lidar sensor to allow the UAV to autonomously navigate and map out its surroundings on a 2D plane [3]. This is shown in figure 2.2. The base flight controller itself is

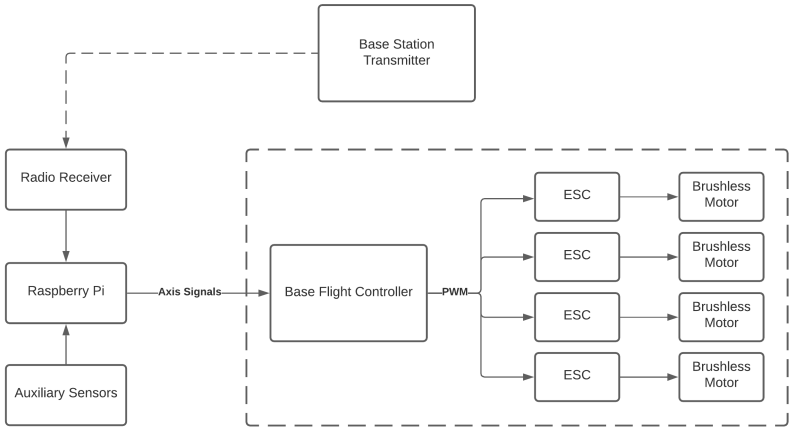

Figure 2.2: Diagram of flight controller system with raspberry pi

akin to a PID controller. The firmware allows us to manually tune the PID values depending on the user's preference. Saengphet et al. demonstrate a more robust way of tuning the PID values as compared to the more traditional heuristic methods. It involves conducting piloted test flights to see how the UAV responds to certain stimuli [7].

2.2 Simultaneous Localization and Mapping

An open-source SLAM API was proven to be functional by Mur-Artal et al[**OrbSlam1**]. The API makes use of a monocular camera as an input device and is able to reconstruct both indoor and outdoor environments in real time. This SLAM system was developed with ground based robots and cars in mind. It essentially works by selecting a number of points of interest in one frame of the video then matching these points to similar points of interest in the next frame. It then takes the difference in position of these point of interest and calculates depth from there.

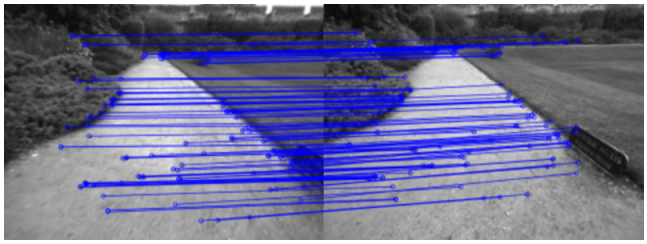

Figure 2.3: Frame by frame point matching by ORB SLAM 1[**OrbSlam1**]

The authors then further improved the accuracy in a second iteration[**OrbSlam2**]. Here, they expanded the input to stereo cameras and depth cameras as these seemed to provide much more accurate results. The authors found that using only a monocular camera was prone to failures as more rotations were used in the machine's exploration. They also found that using a depth camera or multiple cameras (as in the stereo camera) significantly improved the accuracy of depth perception. Another aspect that the authors improved

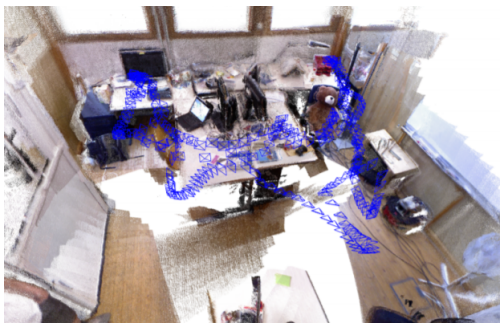

Figure 2.4: 3D reconstruction of a room by ORB SLAM 2 using a depth camera. Marked in blue is the trajectory of the camera[**OrbSlam2**]

upon is the fact that this version was developed to be an out-of-the-box SLAM solution. As a result, this version provides more support for varying systems and is much easier to use as compared to the previous version. However this version does have its lapses where processing power is concerned. Due to the heavy operations being performed, it would struggle to perform even on a Raspberry Pi 4.

Looking for a lightweight SLAM solution, FastSLAM 2.0 presents a SLAM system for low-power embedded architectures [**FastSlam2**], which would be more ideal for the purposes of this project. The authors used a 320x300 resolution camera connected to a Raspberry Pi 2B. While the system was able to function within the low-power specifications, it suffers in terms of accuracy. The system was significantly less ac-

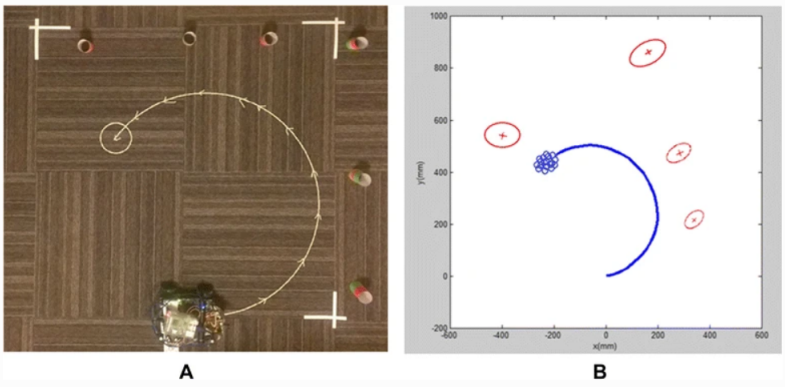

Figure 2.5: Top view of a robot navigating through a corner (A) and the corresponding Fast SLAM map (B). The robot's trajectory is denoted by the blue line while obstacles are denoted by the red figures.[**FastSlam2**]

curate than the SLAM systems discussed previously [**OrbSlam1**, **OrbSlam2**]. While there were no direct measured comparisons between ORB SLAM 2 and FastSLAM2.0, FastSLAM2.0 was only able to detect 75% of obstacles presented to it. As seen in figure 2.5, while navigating a corner of obstacles, the robot failed to detect 2 of the 6 obstacles. The authors claim that this was due to the low resolution of the camera used and that accuracy can be further improved by using an encoder with a higher resolution.

Finally, Forster et al. present a lightweight but accurate solution for localization and mapping with Semi-direct Visual Odometry (SVO) [**SVO2017**]. Primarily developed for use with UAVs and monocular cameras, SVO boasts a reduction in CPU usage of up to 79.45% compared to ORB SLAM 1. SVO is also up to 11.8 times faster while maintaining an accuracy that is 13.5 times better than that of ORB SLAM 1. A summary of the comparison between monocular ORB SLAM 1 and SVO may be seen in figure 2.6. SVO is faster because it needs to process less data. While ORB SLAM 1 and 2 extract features and descriptors from every frame, SVO only does this for a few key frames.

	Mean	St.D.	CPU@20 fps
SVO Mono	2.53	0.42	55 ±10%
ORB Mono SLAM (No loop closure)	29.81	5.67	187 ±32%

Figure 2.6: Comparison of and ORB SLAM 1 and SVO[**SVO2017**]

2.3 Autonomous Navigation

Autonomous navigation consists of two methods; global and local. Global navigation methods involve planning out the general trajectory of the robot without taking into account the local movement restriction of the robot. This entails planning out the overall path that the robot will take when reaching its goal. By contrast, local navigation methods involve planning the trajectory of the robot around its direct environment. This involves more minute movements and obstacle avoidance. According to Dowling et al., the best navigation results have been obtained by combining both of these methods[3]. The navigation system used by Dowling et al. mimics navigation systems used by ground based robots and adapts them to suit UAVs flying at a fixed height. Their system simulates all possible trajectories that the UAV can take and assigns a cost to each one. The UAV then takes the path with the smallest cost. While this method functions well for 2D mapping and navigation, this would be lacking for this project as we would need to navigate in 3D. Luckily, Zhou et al. present a method for performing fast UAV exploration in complex environments (FUEL)[**Fuel2020**]. It starts similarly by finding the most optimal global trajectory that is available to the UAV. It then continually recalculates as more and more regions are explored. FUEL is based on the frontier method of navigation where the UAV looks for the nearest unexplored region. The authors also introduce a frontier information structure (FIS) that contains information on the environment. The structure is then updated continuously and allows for high frequency planning. Based on the FIS, the system then generates a heirarchy of motions in three steps (course to fine). The heirarchical planner finds efficient global paths, selects a local set of optimal viewpoints, and from there, generates the best trajectory based on the minimum time. Figure 2.7 shows a test run by using the classical frontier method as a benchmark. As we can see, the FUEL method is more efficient, generating the shorter path. Alternatively, Xu et al. present

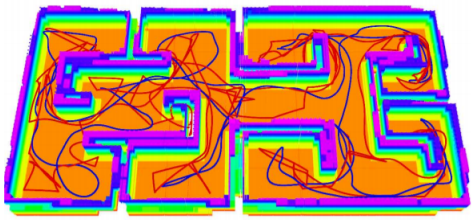

Figure 2.7: Path generated by frontier method (red) and FUEL method (blue)[**Fuel2020**]

a different method of navigation that deviates from the frontier method and instead adopts a method based on incremental sampling and a probabilistic roadmap [**ProbNav**]. In this method, nodes are incrementally added to the explored regions to generate the best viewpoints for the camera system. These viewpoints are

those that provide the system with the most data points, thereby mapping the most amount of regions in the least time. The probabilistic roadmap allows for rapid searching of alternative global and local paths that the UAV can take. However the authors found that the system was not able to locate all obstacles 100% of the time, raising concerns with regards to the safety of the UAV while navigating. The authors compared the method's performance to that of the frontier method. The results may be seen in figure 2.8. After 10 minutes of exploration, the probabilistic roadmap method was able to explore more regions than the frontier method.

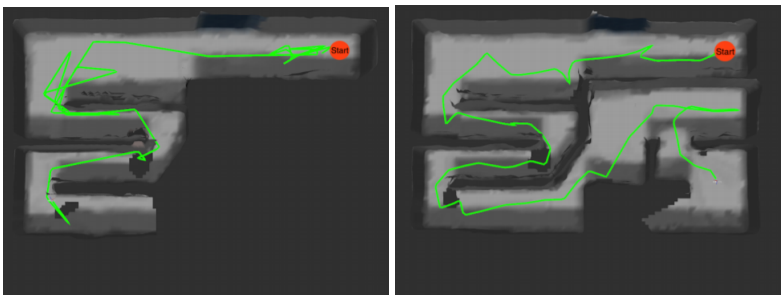

Figure 2.8: Regions explored and path generated by frontier method (left) vs proposed probabilistic roadmap method (right) after running for 10 minutes. [**ProbNav**]

Chapter 3

Objectives

This project aims to develop a low-cost and lightweight UAV capable of autonomous 3D indoor mapping. Given an enclosed indoor space, the UAV should be able to autonomously and completely explore and map out all enclosed regions while navigating at a speed of at least 0.5m/s. During flight, the UAV should be able to transmit the mapped data to the base station so that the user has real-time feedback of how the mission is progressing. On top of this, the UAV should also be able to function non-autonomously. The drone should be able to be piloted manually through a pilot interfacing with the base station.

Chapter 4

Preliminary Work

4.1 Flight Controller Design and Fabrication

Cost, size, and weight were the primary deciding factors for the flight controller design. For the base flight controller, the MultiWii was implemented as seen in figure 2.1. This allows for a much more flexible and cost effective solution. The microcontroller used was the Arduino Pro Mini, which provides a small footprint. It was then wired to send pwm signals to a 4-in-1 ESC board. A 4-in-1 ESC board was selected over individual ESC modules because the board is lighter, smaller, and is less costly. Furthermore, less wiring will need to be done, leading to easier mass production. The motors are 2400KV brushless motors suitable for 5 inch propellers. For the inertial measurement unit, the MultiWii was connected to an MPU6050 via an I2C interface. The MPU6050 includes both an accelerometer and a gyroscope. The arduino receives instructions from a Raspberry Pi 4. There are two options for communication between the Raspberry Pi and the Arduino; single wire PPM and PWM over multiple wires. Both methods were tested to find the method with the most stable signal. It was found that the signals gained from PWM over multiple wires was noisy. PPM gave a much more consistent signal and so it was chosen as the mode of communication. The Raspberry Pi will also be receiving data from a GY-87 module, which contains an accelerometer, gyroscope, barometer, and magnetometer which are all accessible via I2C protocol. This will aid the UAV with regards to SLAM and navigation. Aside from this, the Raspberry Pi is also connected to a GPS module which, while inoperable during indoor operations, will prove to be essential during outdoor missions especially when locating the UAV. For communication, the flight controller employs a LoRa RA-02 module,

which was chosen for its long range capabilities. The LoRa module will be communicating with a similar module connected to the base station. A diagram of the system may be seen in figure 4.1.

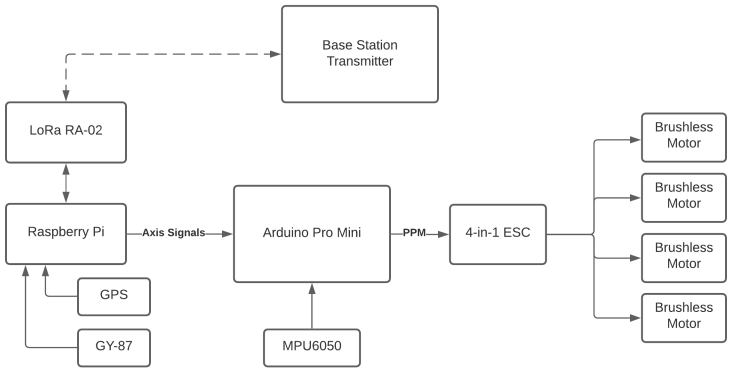

Figure 4.1: Flight controller diagram with ESC module and motors

The flight controller board has already been designed and fabricated to include the necessary components. This can be seen in figure 4.2. While quantitative tests have yet to be conducted, the flight controller has been verified to be able to run the motors at variable speeds. Further work needs to be done in order to refine the motor control. These include tuning and calibration, which will be done during the project proper.

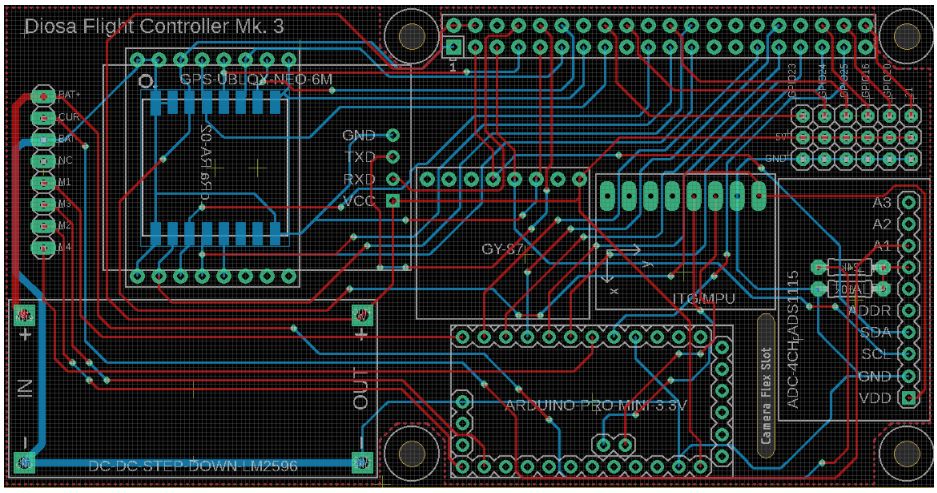

Figure 4.2: Flight controller board

The Raspberry Pi is running the latest version of Raspbian. It also uses ROS (robot operating system) for the UAV's processes. ROS is an open source robotics framework that was designed to be modular and

flexible, taking into account the multitude of processes that a robot needs to run. In the case of this project, the modularity allows for easier integration of different modules and functions. This in turn means that the source code will be easier to maintain as more and more functions are added.

Chapter 5

Methodology

This project can be divided into three milestones; flight controller calibration, integration of SLAM, and finally, integration of autonomous navigation system.

5.1 Flight Controller Tuning and Calibration

With the flight controller fabricated, it then needs to be calibrated and tuned. Fortunately, the MultiWii also comes with a GUI that allows users to calibrate and tune the sensors and PID values. The gyroscope and accelerometer may be calibrated automatically via the GUI however the PID values must be calibrated manually. Typical heuristic methods involve test flights where a pilot performs sharp maneuvers in order to see how the UAV reacts. They then come back to the interface and adjust the PID values based on trial and error. After calibration and tuning, the UAV should function similarly to an off-the-shelf quadcopter.

5.2 SLAM Integration

There are two outstanding choices for SLAM; Orb SLAM 2 [**OrbSlam2**] and SVO [**SVO2017**]. Orb SLAM 2 provides a powerful and fast out-of-the-box solution for SLAM however due to the processing power restrictions, SVO proves to be the more ideal system. Luckily, the developers have already provided a means to integrate SVO into the ROS framework. Therefore, applying SVO to the UAV should be a trivial

matter. Data from SVO will then be communicated via the RA-02 module back to the base station so that the pilot may have a real-time view of how the mapping operation is progressing.

5.3 Autonomous Navigation and Exploration

Two possible systems may be chosen for autonomous Navigation; the FUEL system[**Fuel2020**] that is built upon the frontier method of navigation and the method that uses a probabilistic roadmap as demonstrated by Xu et al.[**ProbNav**]. In order to see which navigation system is the best fit, both systems will be implemented and benchmark tests will be run. The systems will be compared in terms of exploration speed, efficiency (i.e. the tests will favor the method that generates the shortest flight path to completely explore a region), completeness (i.e. the method must not leave any region within the enclosed space unexplored), and ability to safely navigate around obstacles.

Chapter 6

Timeline

6.1 Gantt Chart

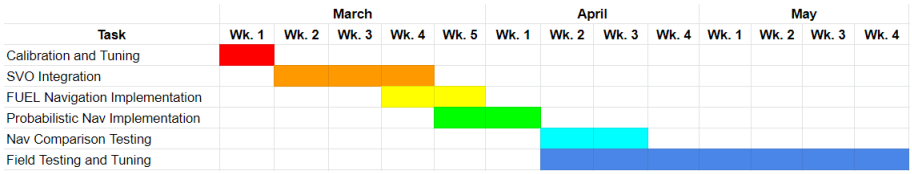

Figure 6.1: Project gantt chart

6.2 Halfway Deliverables

By the end of March, the following deliverables should be ready:

1. Flight controller achieves the following specifications:
    - UAV must be able to collect inertial data from the integrated IMU
    - GPS is able to obtain UAV's geographical coordinates and send it to the base station
    - Capable of sending and receiving data through integrated LoRa module
    - Should be able to perform stable non-autonomous flight at a fixed height for at least 10 minutes.
2. Operational SLAM system using SVO

- Should be able to perform SLAM in real-time
- Should be able to store map data on a global scale rather than just a local one (i.e. aside from being able to map out the UAV's current region, the drone must be able to store map data of other regions connected to it that the UAV has already explored)
- Should be able to transmit map data to base station

6.3 End of Project Deliverables

In addition to the deliverables listed above, by the end of the given time period, the project must have accomplished the following:

1. Autonomous exploration and navigation
    - The UAV should be able to generate a complete map of all reqions within an enclosed space (i.e. There must be no unexplored regions within the enclosed space after the UAV is done exploring).
    - The UAV should be able to autonomously navigate while flying at a speed of at least 0.5 m/s
    - After mapping a space, the UAV should be able to safely return to the origin

Bibliography

[1] X. Kuai, K. Yang, S. Fu, R. Zheng, and G. Yang, "Simultaneous localization and mapping (SLAM) for indoor autonomous mobile robot navigation in wireless sensor networks", in *2010 International Conference on Networking, Sensing and Control (ICNSC)*, 2010, pp. 128–132.

[2] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey", in *Robotics and Autonomous Systems, vol. 86*, 2016, pp. 13–28.

[3] L. Dowling, T. Poblete, I. Hook, H. Tang, Y. Tan, W. Glenn, and R. Unnithan, "Accurate indoor mapping using an autonomous unmanned aerial vehicle", 2018.

[4] Electronoobs, *Arduino MultiWii Flight Controller*, `http://www.electronoobs.com/eng_robotica_tut5_3.php`, 2017.

[5] S. Al-Kadhim, "Communicating with Raspberry Pi via MAVLink", 2019. DOI: `10.2139/ssrn.3318130`.

[6] N. Smolyanskiy, A. Kamenev, and J. Smith, *Project Redtail*, `https://github.com/NVIDIA-AI-IOT/redtail`, 2017.

[7] W. Saengphet, S. Tantrairatn, C. Thumtae, and J. Srisertpol, "Implementation of system identification and flight control system for UAV", in *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, 2017, pp. 678–683.