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
Professor Roxanne De Leon

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 survey or reconnaissance missions. However, learning to pilot a UAV can prove to be difficult and expensive. The goal of this project is to develop a drone that can not only provide the user with an accurate 3D map of indoor spaces but do so *autonomously*. 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 Tables

List of Figures

Chapter 1

Introduction

As technology develops, unmanned aerial vehicles continue to be integrated into increasingly diverse environments and fields. From photography and cinematography, all the way to military applications. Similarly, 3D mapping has 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. This necessitates the development of more complex ways of controlling the UAV that would effectively lower the learning curve that new pilots have to face. Autonomous functions also lower the risk of human error by limiting the amount of ways that the pilot can influence the actual flight. 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, the autonomous exploration function, and the mobile radio controller that allows the pilot to interface with the UAV.

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 [1] or one or more cameras[2] 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 [1]. Methods of navigation include global and local navigation however [1] notes that the best results can be foound by combining the two.

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. [3] shows how to build a basic flight controller. 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 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, as seen with [3] would be a PPM signal containing the throttle, yaw, pitch, and roll signals. The Pixhawk, another family of flight controllers as demonstrated by [4] 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. [4] makes use of a Raspberry Pi to communicate with the Pixhawk. This allows them to interface with the UAV in a more sophisticated manner and make use of the computer's higher processing power. [5] demonstrates how to use an onboard computer to apply neural networks for intelligent flight control. Meanwhile, [1] uses a Raspberry Pi coupled with a lidar sensor to allow the UAV to autonomously navigate and map out its surroundings on a 2D plane. This is shown in figure 2.2. The base flight controller itself is a PID

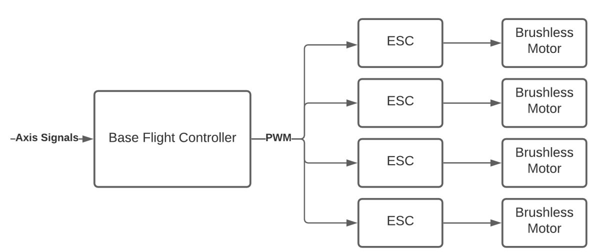

Figure 2.1: Diagram of base flight controller connected to ESCs and motors

controller. The firmware allows us to manually tune the PID values depending on the user's preference.

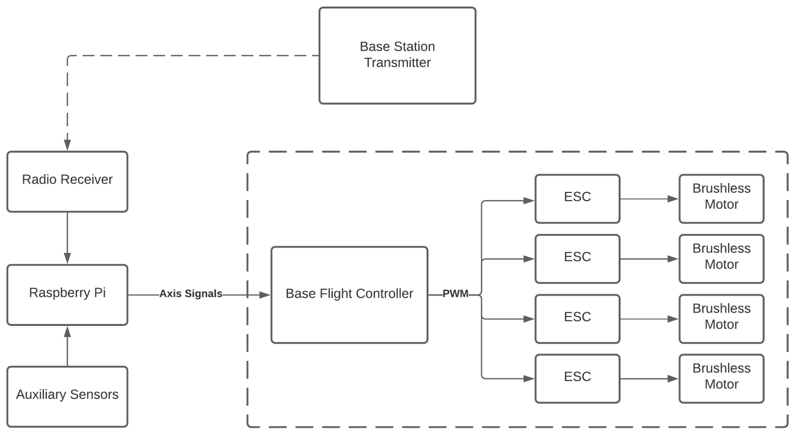

Figure 2.2: Diagram of flight controller system with raspberry pi

[6] demonstrates a more robust way of tuning the PID values as compared to the more traditional heuristic methods.

2.2 Simultaneous Localization and Mapping

An open-source SLAM API was proven to be functional in [7]. 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. The authors then further improved the accuracy in [2]. 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 multiple cameras (as in the stereo camera) significantly improved the accuracy of depth perception. Another aspect that the authors improved 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, [8] presents a SLAM system for low-power embedded architectures, 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 accurate than the SLAM systems seen in [7] and [2]. 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, [9] presents a lightweight but accurate solution for localization and mapping. [9] makes use of semi-direct visual odometry (SVO). Primarily developed for use with UAVs and monocular cameras, SVO boasts a reduction in CPU usage of up to 79.45% compared to [2]. SVO is also up to 11.8 times faster while maintaining an accuracy that is 13.5 times better than that of [2]. SVO does away with feature extraction and matching procedures by estimating the movement of the camera system using the pixel intensities per frame.

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 [1], the best navigation results have been obtained by combining both of these methods. The navigation system used by [1] 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, [10] presents a method for performing fast UAV exploration in complex environments (FUEL). 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. Alternatively, [11] presents a different method of navigation that deviates from the frontier method and instead adopts a method based on incremental sampling and a probabilistic roadmap. 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.

Chapter 3

Problem Statement and Objectives

Chapter 4

Methodology

Chapter 5

Timeline

5.1 Gantt Chart

5.2 Halfway Deliverables

Chapter 6

Experimentation and Results

Bibliography

[1] 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.

[2] R. Mur-artal and J. Tardos, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras", 2017. DOI: 10.1109/TRO.2017.2705103.

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

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

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

[6] 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.

[7] R. Mur-artal, T. Juan, and J. Montiel, "ORB-SLAM: a Versatile and Accurate Monocular SLAM System", 2015. DOI: 10.1109/TRO.2015.2463671.

[8] A. Serrata, S. Yang, and R. Li, "An intelligible implementation of FastSLAM2.0 on a low-power embedded architecture", 2017. DOI: 10.1186/s13639-017-0075-9.

[9] C. Firster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems", 2017.

[10] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "FUEL: Fast UAV Exploration using Incremental Frontier Structure and Hierarchical Planning", 2020.

[11] Z. Xu, D. Deng, and K. Shimada, "Autonomous UAV Exploration of Dynamic Environments via Incremental Sampling and Probabilistic Roadmap", 2020.