System Identification of a Micro Aerial Vehicle

Aman Sharma

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 20.10.2017

Supervisor

Prof. Arto Visala

Advisor

Prof. Arto Visala

Aalto University

Aalto University

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of the master's thesis

Author Aman Sharma

Title System Identification of a Micro Aerial Vehicle

Degree programme SpaceMaster

Major Space Robotics and Mechatronics **Code of major**

Supervisor Prof. Arto Visala

Advisor Prof. Arto Visala

Date 20.10.2017 **Number of pages** 67 **Language** English

Abstract

The purpose of this thesis was to implement an Model Predictive Control based system identification method on a micro-aerial vehicle (DJI Matrice 100) as outlined in a study performed by ETH Zurich. Through limited test flights, data was obtained that allowed for the generation of first and second order system models. The first order models were robust, but the second order model fell short due to the fact that the data used for the model was not sufficient.

Keywords MAV, System Identification, MPC, Quadcopter

Preface

I want to thank my parents for giving the opportunities in life that led me to this point. I would also like to thank my cousin, Radhika and my friends, Shikha, Richard, Christoph, Antvane, and Aleksandra for all their help and support.

Otaniemi, 24.4.2018

Aman Sharma

Contents

1 Introduction

Humans have persevered towards the art of flight from an early age. It has allowed us to connect to each other no matter the distance. In the last century we have gone from basic wooden planes to jets capable of flying at many times the speed of sound. This is due to development of newer aviation technology over the course of the years and due to a better understanding of the physical models and dynamics involved in aviation. These have caused a tremendous change in the capabilities of the modern airplanes and the pinnacle is still not reached.

The development of research in the aerospace sector following the first world war led to a boom in the industry. With universities offering degrees in the field and collaborating on research with industry, the field became available to anyone willing to learn. Due to this, there were more designers focused on improvement rather than mass production. Consequently, the industry saw a big and rapid improvement in aerospace technology [1]. A better understanding of the physics and aerodynamics involved and with the development of control theory, firms were able to create more efficient and better aircrafts at a much lower price. This grasp in the field of aerodynamics has also opened doors to aircrafts we could create to suit our needs. In figure 1, we see an extreme example of modern aerial vehicles that exist today: a C-130, capable of carrying immense loads and a nano-quadcopter used for research and development.

These innovations in aerospace technology has allowed us to develop aerial vehicles that are able to image landscapes, develop three dimensional maps of terrains, and even fly autonomously. The improvement of these technologies has, however, also caused the dynamics and physical models of these aircrafts to become more intricate. With an increase in the variability of sensors and payloads, modelling an accurate system to represent these aircrafts has become a difficult task as they lean more and more towards non-linearity.

1.1 Overview

In the recent years, there has been a surge in the personal and commercial drone industry, more specifically, drones with quadrotors. Even the low-cost drones come

(a) C-130 Hercules [2]

(b) Crazyflie Nano Quadcopter [3]

Figure 1: Various types of aircrafts

with basic sensors such as inertial measurments units (see section 3.3) and cameras capable of taking high-definition videos. The higher-end drones can be purchased with flight controllers capable of executing defined missions as well as target and path following. In 2016, the drone market grew from $1.71 to $2.80 billion for personal drones and from $2.36 to $3.69 billion for commercial drones. This means that since 2016 the revenue share for drones has grown by almost 34% and the unit sales has grown by almost 39% [4].

DJI, a drone manufacturer based in China, has managed to capture almost 70% of this market due to their affordable prices and due to a rapid provision of spare parts. For this project, their DJI M (Matrice) series M100 drone was utilized. Although this drone is programmer and development friendly and DJI provides documented SDKs and tutorials for it, there is still some critical information not provided, such as attitude dynamics and how the N1 flight controller works [5]. This thesis is written for the purposes of better understanding the system and dynamics of a quadcopter or drone. More specifically, it'll focus on the M100. The drone was used to test various sensors and how they change the physical model and the dynamics of the system.

1.2 Organization

In the following sections, the basic make-up and functionalities of quadrotor will be explained. Some fundamental concepts will be briefly overviewed while other, more specific and advanced concepts, will be presented in depth. The next sections will provide a quick overview of the different essential concepts involved when working with aerial vehicles while the background will give a more detailed presentation of specific context directly related to this thesis.

The first section will give an introduction in to the project, the second section will provide background and literature research regarding quadcopters, the third section will explore algorithms utilized in visual odometry, the fourth section will demonstrate the implementation of the topics discussed, and the final section will summarize the project. Objectives and requirements of the thesis can be read in section 1.4.

1.3 Motivation

The usage of drones in emerging technologies such as visual odometry, swarm robotics, 3D Mapping, etcetra, has given rise to a need to better understand this machinery. In order to stay near the pinnacle of current research and development of quadcopters, it is necessary to understand the basics and the advance methods that can be applied to these drones. This is where the first motivation for this thesis stems from.

The project was designed in order to develop a curriculum like thesis which can be used as a guide to teach studends about the various dynamics involved while working with quadcopters, eventually teaching them about the current technologies that exist today and how to implement them. This is where the second motivation of the thesis arises from.

While doing research on such a topic, various methods can be discovered and developed that can optimize some process. In addition, new technologies can be tested and improved through the collection and analysis of data. In this project, several new sensors and data collection methods are being tested in order to verify their effectiveness. This is where the third and final motivation of the thesis stems from.

1.4 Objectives

The first objective of this is to procure and assemble a suitable drone. The requirements for the drone were as follows:

1. Must be programmable and have some interactive interface
2. Must be modular so that parts (off the shelf or in house) can be added and removed quickly
3. Must be cost-effective and have off the shelf spare parts
4. Must be able to be fitted with a multitude of sensors and have some in-built sensors such as an inertial measurement unit, GPS, etcetra
5. Must be able to carry payloads of at least 2kgs

6. Must have a battery capable of powering the onboard sensors as well as a small onboard computer

These requirements are such because the main goal of the thesis is to teach students about robotics through drones as described below in the third objective.

The second objective of the thesis is to implement a robust controller on top of the N1 flight controller that the drone came with. As mentioned before, since DJI does not provide any open-source information about the controller, some adjustments have to be made when adding componenets such as an onboard controller. The flight controller that will be implemented is provide by ETH's Autonomous System's Laboratory and is based on the algorithm described in [5]. More information will be provided about the algorithm in the later sections.

The third and last objective was to develop a drone system that is capable of teaching students new to control systems especially in the field of robotics. The whole platform is made so that students can quickly see the affects of adding/removing sensors and payload and how it effects the drone's capabilities as well as how it changes the physical model of the drone.

2 Background

This section will explore the fundamental concepts of drone technology as well as dive in to the more detailed, project specific, conepts. Understanding the basics is necessary before diving in to the advance concepts. The section is written in such a way that one could work their way up from bottom.

2.1 Control Systems

In order to understand aerial vehicles, we must understand what a control system is. A control system regulates another device's or *plant's* behavior. There are two types of control systems, open-loop and closed-loop. In an open-loop system the action of the controller is independent of the output. A closed-loop or a feedback control system is the most common form of control system found. Here, a controller tries to match the output signal to an input signal using a feedback loop. Simple examples

are shown in figures 2 and 3 [6]. An open-loop toaster works off a timer. The process output or the color of the bread does not affect how long the timer will run. It is predefined to run for a specific time. In a closed-loop loop toaster model, the color of the bread directly affects the timer through the feedback loop, e.g. a person.

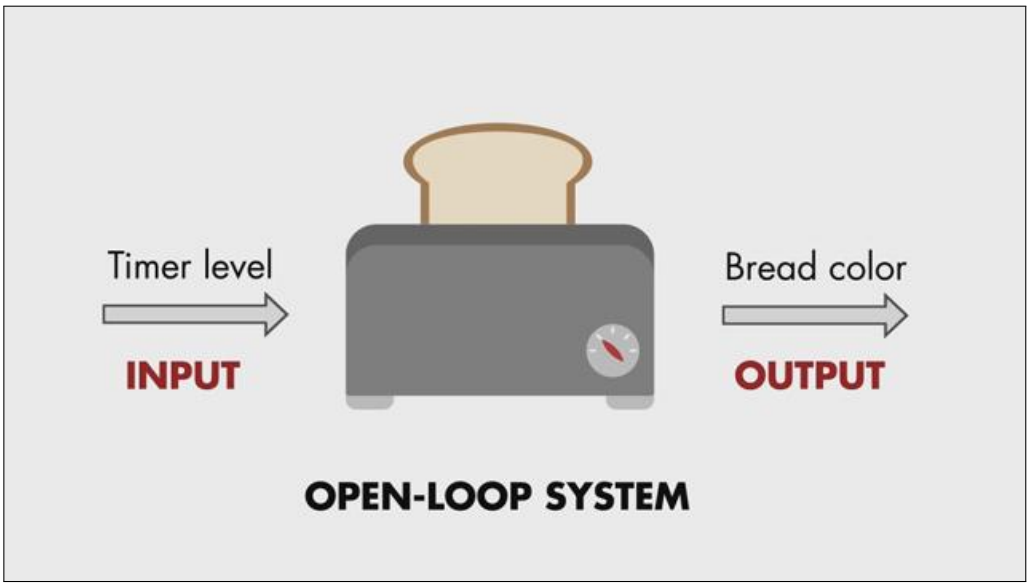


Figure 2: Open-Loop Control System

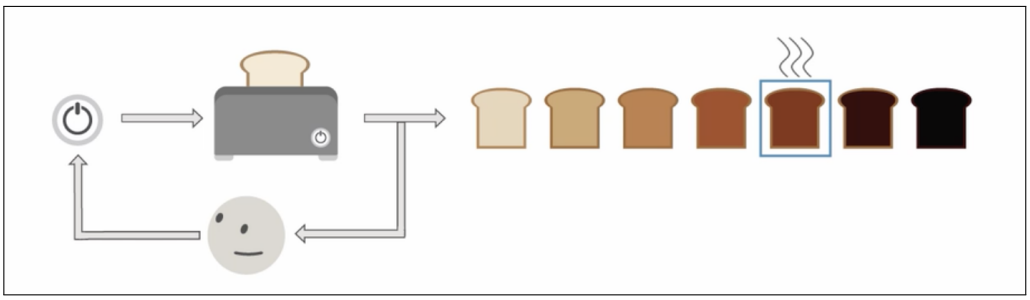


Figure 3: Closed-Loop Control System (Feedback Loop)

In an aircraft, the flaps, the ailerons, the rudder and the engine, are all controlled by the flight control system.

2.2 Control Theory

The history of control theory leads back to Aristotle (384-322 BC) as mentioned by Bennett in his first volume [7]. In it, he makes reference to Aristotle's book "*Politics*" quoting a verse from Book 1, Chapter 3 which states:

> *"...if every instrument could accomplish its own work, obeying or anticipating the will of others...if the shuttle weaved and the pick touched the lyre without a hand to guide them, chief workmen would not need servants, nor masters slaves"*

Here Aristotle is hinting towards the need to automate processes in such a way that they require no human input: a basic description of control theory. There are three core concepts of control theory: feedback, disturbances, and optimization [8, 9].

2.2.1 Feedback

The concept of feedback can be very intuitively understood through nature and even through our own experiences. In nature, evolution is driven by the concept of feedback. Over generations this constant feedback has improved the survivability of each species. In us, we utilize feedback for almost each and every task using our senses as "sensors" to provide our brain with relevant information [8, 9].

2.2.2 Disturbances

In every control system, the existence of disturbances is fundamental and goes hand-in-hand with feedback. For example, while driving a car, we usually try to maintain our heading so as to not swerve the car. However, a strong gust of wind can disturb this heading. Here we use the feedback from our sensors to correct the disturbance. These disturbances are inherent in almost every system and do not need to strictly controlled [8, 9].

2.2.3 Optimization

Optimization is the concept of receiving the best reward for the least amount of energy expended. In our previous car example, it would be optimal to correct the car's heading with the most minimal turn of the steering wheel. An over-correction can have lead to unwanted consequences. This concept is one of the more difficult concepts to achieve as it requires numerical solutions to the system which is most of the times non-linear [8, 9].

2.3 Linear and Non-Linear Control Theory

In addition to being categorized as open-loop or closed-loop, a system can be further subcategorized as linear or non-linear. A linear system follows the superposition principle, which relates to homogeneity and additivity. On the other hand a non-linear system does not follow the superposition principle. Most real-world physical systems, whether it is a cruise control system on a car or the actuators on a space telescope, are inherentely non-linear. In order to model these systems they either have to be linearized around a point of interest or one of several analysis techniques have to be applied in order to create a stable controller for these systems.

Almost, all the systems on aircrafts are non-linear due to the high amount of variablility present in the system. Non-linearity is also a hindrance in optimization as it is much more difficult to optimize highly non-linear systems. There is much research that is being devoted to this field since a small optimization can result in large cost-savings.

2.4 System Identification and Modeling

System identification is the method of representing dynamic systems in terms of a mathematical model using measurements of its input and output signal. System identification follows the following procedure [10]:

- Measure system behavior in time or frequency domain in order to estimate adjustable parameters in the system model
- Select a suitable model structure
- Select an estimation method to estimate the unknown parameters in the chosen model structure
- Simulate and validate the model in order to see if it fits well to the given criteria.

A model is a mathematical relationship between a system's input and out usually represented using differential equations, transfer functions, state-space models, or pole/zero gain models [10]. The model contains unknown parameters which are

tuned to fit the model to the behavior of the system. Usually, these equations are acquired by examining the primary physical laws governing the system and/or through experimental data. The benefit of system modeling is that it allows us to simulate and analyze the response of the system given certain inputs to a certain degree of accuracy. This helps in predicting the behavior of the system better. Since most systems are inherently non-linear, the non-linearity is linearized around some point of interest and simulated to see its response. Based on these point(s) of interest a system can have several different models. It all depends on which questions need to be answered.

2.4.1 Model Predictive Control

Model Predicitve Control (MPC) also knowns Dynamics Matrix Control (DMC), Generalizd Predictive Control (GPC), and Receding Horizon Control (RHC) utilizes a "dynamic model of the plant [which] is used to predict the effect of future actions of the manipulated variables on the output (thus the name "Model Predictive Control") [11]." In each sample, the objective is to minimize the predicted error thus opitimizing the manipulated variables which results in an optimal future action. As the sensors provide data to the controller the optimization is repeated for each of these time steps.

MPC is more useful in plants that require integration of multiple systems with different constraints and where the success of the plant depends on the fact that the contraints will be met and no violations will be allowed. Designing a controller for a single process is not only expensive but also yields in failure as it will not work with other processes [11].

Although MPC was originally designed for industries such as petro-chemical, it has found wide use in the aerospace industry and academia as well. The main reason being that through MPC, high-performance controllers capable of operating without human mediation are able to be designed. A high-level over view of MPC is shown in figure 4. The importance of MPC in this project comes from the Dynamic System Identification that can be performed using this method. This allows for a quick deduction of a system model when different payloads are added or removed from the copter. It is also a key part of the algorithm utilized in [5].

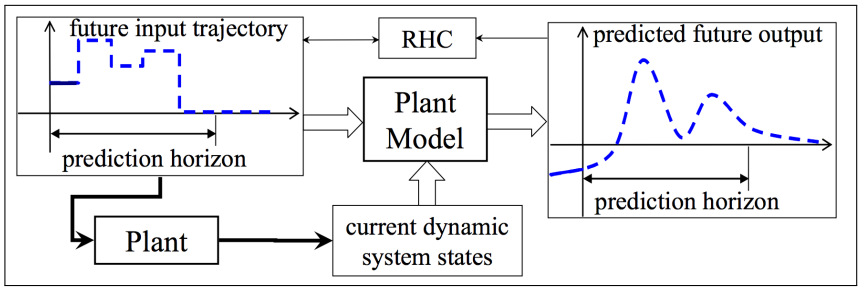
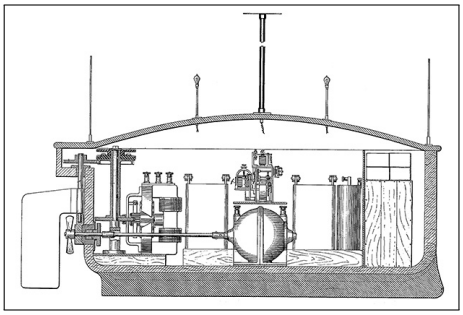


Figure 4: Model Predictive Control Flow Diagram



(a) Nicola Tesla's Radio Controlled Boat [15]

(b) Royal Air Force DH.82 Tiger Moth "Queen Bee" [16]

Figure 5: Examples of Early Radio-Controlled Vehicles

3 Quadrotors

3.1 History and Overview

This leads to the main topic of drones whose history and origination starts from war. The first instance of unmanned aerial warfare was in 1849 demonstrated by Austrian forces [12]. Following this, in 1898, Nikola Tesla, called the father of unmanned aerial technology, demonstrated a radio controlled boat as seen in figure 5a [12]. Then in 1935, a British warplane dubbed the DH.82 Queen Bee (see figure 5b) was equipped with a radio controller in its rear cockpit. This plane is where the term "drone" supposedly originated from [13]. The modern combat drone gained its face with the introduction of the MQ-1 Predator drone [14].

Modern commercial drones got their start in 2010 when Parrot introduced the AR Drone quadcopter which could be controlled with a smart device. A few years later, Jeff Bezos revealed an ambitious plan to deliver goods via quadcopters.

VTOL MAVs have seen a drastic rise in their usage in fields such as exploration, search and rescue, surveliance, etc. due to their small design and the ability to manuever in difficult areas. These MAVs are inherentCurrently, many low-cost MAVs are being implemented in research settings in order

3.2 Quadcopter Operation Theory

Quadcopter or quadrotors, as the name suggests, are comprised of four rotors which are placed equidistance for the center of mass of the quatcopter and from each other. The combination of these motors produce the thrust, the three torques, roll, pitch, and yaw that act on the copter. As can be seen in figure 6, the rotors are setup such that each rotor's neighbors counter-rotates compared to itself in order to cancel the gyroscopic effects and aerodynamic torques generated by each of the motors (rotors 1 and 3 rotate clockwise while rotors 2 and 4 rotate counterclockwise). This also allows the quadcopter to hover in one spot if equal and enough thrust is provided to the rotors [17]. Each motor M_i (for $i = 1,..., 4$) generates a force f_i which can be related to the angular velocity using equation (1).

$$f_i = k\omega_i^2 \tag{1}$$

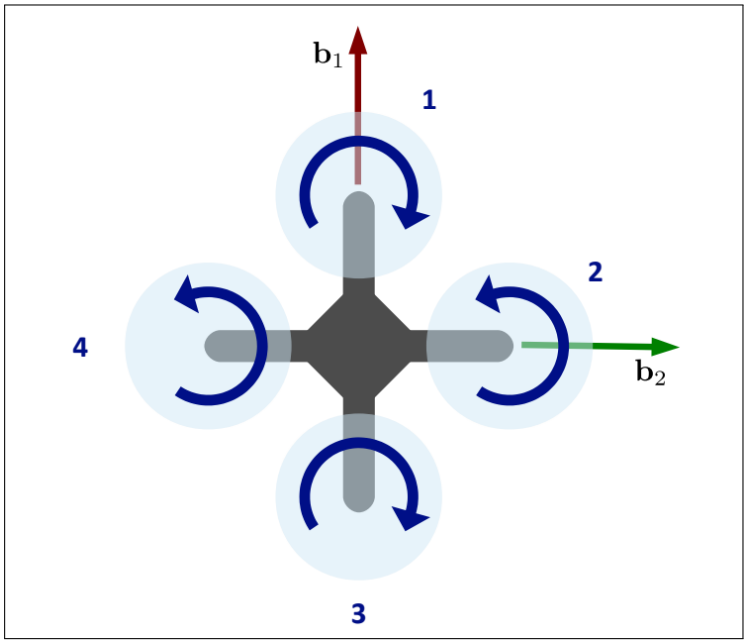


Figure 6: Quadcopter Rotor Setup [17]

The on-board flight controller, which is able to alter the rotational velocities of each of the rotors individually, controls the movement of the quadcopter in the roll, pitch, and yaw axis. The pitch torque is a function of difference $f_1 - f_3$, the roll torque $f_2 - f_4$, and the yaw is the sum of the reaction torques of all the motors as

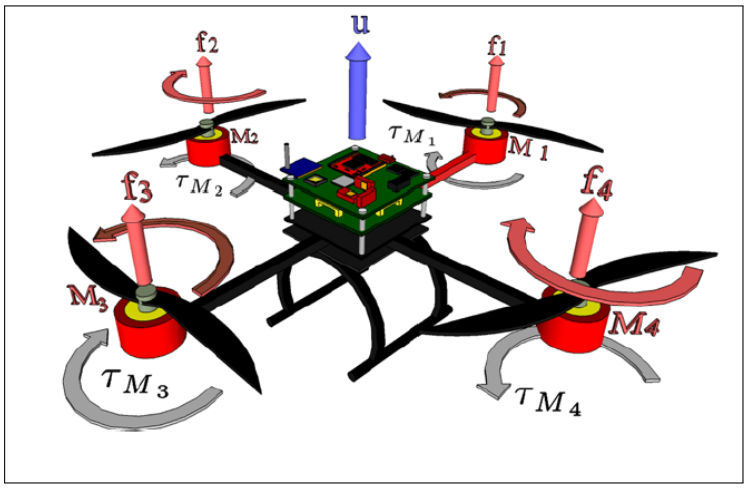


Figure 7: Quadcopter Forces and Torques

shown in equation (2). Referring to figure 6, movement about the yaw axis is done by increasing the velocities of rotors 1 and 3, while simultaneously decreasing the velocities of 2 and 4, or vice versa. Similarly, roll and pitch are induced by increasing or decreasing the velocities of the rotors with the same spin, for example, increasing rotor 1 and decreasing rotor 3 would induce movement in the positive pitch axis [17]. As mentioned during the introduction, aerodynamic drag (τ_{drag}) acts on the torque of motors such that it opposes the direction of the torque (see equation (3)) [18].

$$\tau_{yaw} = \sum_{i=1}^{4} \tau_{M_i} \tag{2}$$

$$I_{rot}\dot{\omega} = \tau_{M_i} - \tau_{drag} \tag{3}$$

Here, I_{rot} is the moment of inertia of the motor about its axis and τ_{drag} is defined as,

$$\tau_{drag} = \frac{1}{2}\rho A v^2 \tag{4}$$

where ρ is the density of the air, A is the area of the blade, and v is the velocity [18].

3.2.1 Quadcopter Dynamics Model

In order to understand the functions of a quadcopter a proper mathematical model must be derived in order to develop stabilization and trajectory methods. The dynamics can either be derived from the Newton-Euler method or the Euler-Lagrange method and are derived in both the inertial frame and the body frame (based on the center of mass of the copter).

In the inertial frame, the linear position of the quadcopter is defined on the (x, y, z) axes with ξ and the angular position is defined using the Euler angles with η. In equation (5), ϕ, θ, and ψ represent rotation about the x, y, and z axes, respectively.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{5}$$

In the body frame the linear and angular velocities are defined using vector V_b and ν_b as shown in equation (6) [19].

$$\mathbf{V_b} = \begin{bmatrix} v_{x,b} \\ v_{y,b} \\ v_{z,b} \end{bmatrix} \qquad \nu_\mathbf{b} = \begin{bmatrix} \omega_{x,b} \\ \omega_{y,b} \\ \omega_{z,b} \end{bmatrix} \tag{6}$$

Most sensors, such as the IMU, gather and output data in the body frame while others do so in the inertial frame. In order to changed the values from one frame to another there exists a rotation matrix $\mathbf{R}$ as shown in equation (7). Here $cx = \cos(x)$ and $sx = \sin(x)$. This matrix is orthogonal, therfore to go from inertial to body frame, the matrix can simply be inverted which is equal to its transpose ($R^{-1} = R^T$) [19].

$$\mathbf{R} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{7}$$

By treating the quadcopter as a rigid body, the newton-euler method can be used to derive the dynamics of the copter in the body-frame and they are given by:

$$m\dot{\mathbf{v}}_b + \nu \times (m\mathbf{v}_b) = \mathbf{R}^T\mathbf{G} + \mathbf{T}_b \tag{8}$$

Here, $m\dot{\mathbf{v}}_b$ is the force required to accelerate the mass and $\nu \times (m\mathbf{v}_b)$ is the centrifugal force and they are equivalent to gravity $\mathbf{R}^T\mathbf{G}$ summed with the total thrust, $\mathbf{T}_b$ [19].

These equations provide a high-level overview of the equations needed to properly model a quadcopter. Other factors such as gyroscopic forces, torque, etc. both in inertial and body-frame have to be derived.

3.3 Sensors

Most quadcopters today are equipped with a plethora of sensors. The personal drone market has exploited the area of vision based sensors. Now, even micro-aerial quadcopters come equipped with cameras capable of take high-definition pictures and videos. On the higher end of the quadcopter scale, there is the possibility to have sensors such as GPS, barometer, etc. and there is even the possibility of having object detection and object following. Well designed user-interfaces have made it

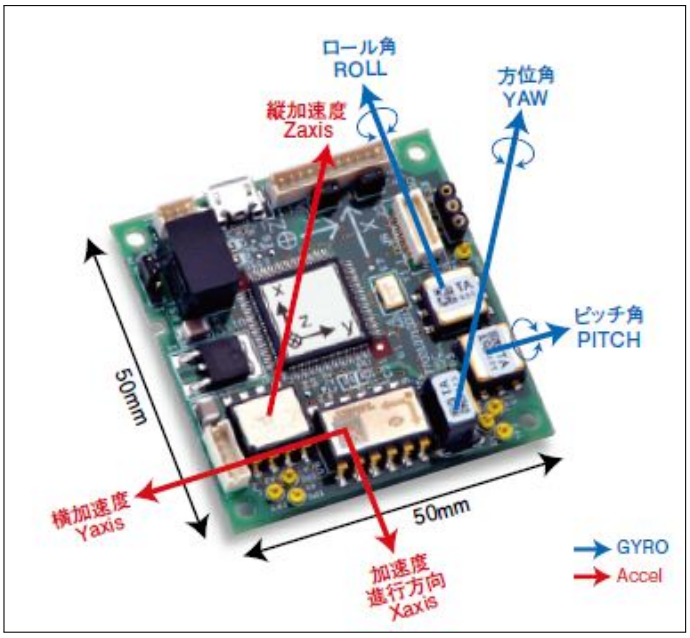


Figure 8: Inertial Measurement Unit [22]

possible for people to fully enjoy the benefits of these sensors, but underneath the "hood" there is many complex algortihms and small sensors running in order to provide the user with the best experience.

3.3.1 IMU

Inertial measurement units, abbreviated IMUs, are widely used in aircrafts, including UAVs, in order to detect the type, rate, and direction of a motion the aircraft performs [20]. Most IMUs used on UAVs, especially smaller drones, are known as MEMS IMUs (Micro-Electro-Mechanical System). They integrate multiaxis information from gyroscopes and accelerometers to provide precision position and motion information for stabilization and navigation purposes. They are able to detect the current rate of acceleration and as well as roll, pitch, and yaw. The IMU data is sent to a central computer or flight controller which fuses the data and calculates the current velocity and position from an initial frame of reference [20]. The output data is given in the body/sensor frame. In addition, sometimes a magnetometer and/or a pressure sensor is also included inside the IMU [21]. An example of an IMU used on small drones is shown below in figure 8.

3.3.2 Gyroscopes

Gyroscopes measure the angular velocity by measuring the rotational acceleration of the sensor. The output value is given in either degrees/sec, or rotations per minute (RPM). Most gyros used on drones currently have 3 degrees of freedom, meaning they are able to calculate rotation about all three axes (roll, pitch, and yaw). The gyroscopes manufactured with MEMS technology are known as vibrating structure gyroscopes. These utilize the principles of the Coriolis effect in order to measure the rate of rotation. An example is shown below in figure 9.

One of the biggest disadvantages gyros suffer from is drift. Since, the gyro data is integrated to receive rotation information, error is accumulated in the form of bias. This bias is usually measured over a long period of time, averaged, and then subtracted from the gyro data in order to compensate for it [23].

There are four types of MEMS gyros commonly used on drones which work off the same vibrating structure principle:

- Tuning Fork Gyroscopes
- Vibrating Wheel Gyroscopes
- Wine Glass Resonator Gyroscopes
- Foucault Pendulum Gyroscopes

3.3.3 Accelerometers

Accelerometers are also micro-electric mechanical systems which measure static acceleration forces, such as gravity, or dynamic acceleration forces caused by the movement or vibration of the accelerometer in either G-force or m/s^2. The static acceleration can be measured to produce the tilt of the device with respect to the gravity vector and the dyanmic acceleration can be measured to produce the way the device is moving. In a 3-axis accelerometer (most common in drones), the axes are setup such that they are orthongonal to each other [25]. An example of a 3-axis accelerometer sensor is shown below in figure 10.

Most accelerometers used in drones either utilize the piezoelectric effect which works measures a voltage proportional to acceleration produced by microscopic

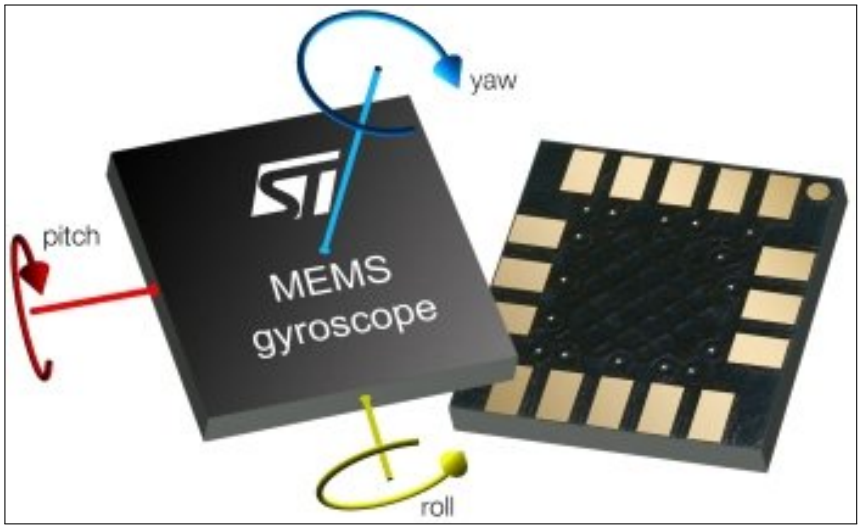


Figure 9: MEMS Gyroscope from Microchip [24]

crystal structures when they experience acceleration forces. The second method is by measuring capcitance between microscopic plates. The stress of acceleration forces causes one of the plates to move resulting in a change in capacitance. This capacitance can then be converted in to a voltage proportional to the acceleration [25].

Aside from the two types of accelerometer technologies described above, below are a few more types that the accelerometers may utilize [26].

The problem with accelerometers is their senstivitiy to vibrations. Aside from showing just the gravity vector values, it will also display forces that are used to move the UAV as well. Only on a long period of time is the accelerometer data reliable.

- Fiber Optic Accelerometers
- Hall Effect Accelerometers
- Heat Transfer Devices
- Magnetoresistive Devices
- Piezoresistive Accelerometers
- Strain Gage Accelerometers

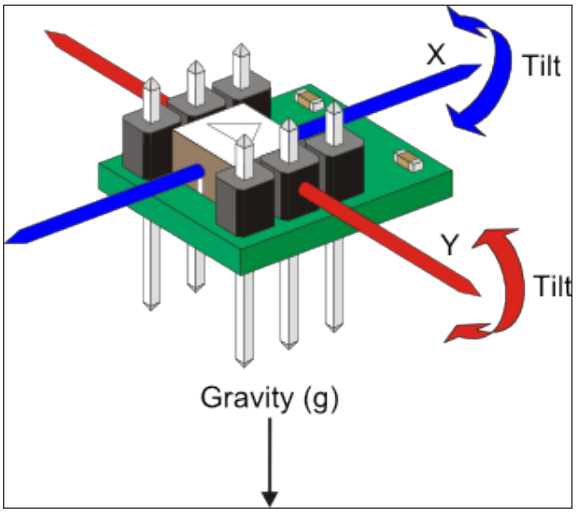


Figure 10: 3-axis Accelerometer

3.3.4 Global Positioning System (GPS)

GPS is another type of sensor that is available on more advanced drones such as the DJI M100 which is able to provide location data with in a few meters and even within a few centimeters an advanced GPS system. The GPS chip operates as a radio receiver receiving signals from the oribiting satellite and can determine position, speed, and time. Signals from three different satellites form three intersecting regions whose common points form the region in which the target can be found. The distance to these satellites must also be determined and this is done using the concept of Time of Arrival (ToA) which is the travel time of a signal from a transmitter to a receiver. By knowing the time and the speed of the radio signal (speed of light), the distance to the satellite can be deduced. There is some error correction that needs to be done since the radio signal does not fully travel at the speed of light and the clock on the receiver is not as accurate as the atomic clock on the satellite [27].

GPS is a widely applied sensor on a quadcopter when trying to execute autonomous flight. With GPS the drone is able to execute functions such as:

- Hover: allows the copter to hold its position at a certain location and altitude.
- Return to Home (RTH): this feature allows the user to record a home location to which the copter will return when this command is executed.
- Waypoint Following; this allows the copter to autonomously fly through present points. The copter is also able to execute other tasks when it reaches a certain waypoint.

GPS heavily advances the capabilites of a quadcopter while only changing the physical model slightly. Most GPS chips are not large in size and can fit on the quadcopter very easily without taking up too much payload weight. Figure 11 shows the GPS found on the DJI M100 drone.

3.3.5 Other Sensors

Below is a brief description of other sensors that many newer drones also contain:

- Pressure Sensor (Barometer): allows the copter to measure it's z-height by measuring the difference in pressure.

Figure 11: The GPS on the DJI M100

- Magnetometer: allows the drone to measure the magnetic field around it in order to determine its heading. It likes a compass for the drone by aligning it with the magnetic north [28].

3.4 Vision Based Sensors

In addition to the IMU, there is a realm of more complex sensors which can be added and are necessary for autonomously flight. Among these sensors are vision based sensors also known as passive sensors due to their ability to capture data without altering the environment (such as with laser and radar based systems) [29]. They are used in a variety of technologies and usually utilize a camera to determine orientation and position within a certain environemtn. These are more versatile than general camera systems in that the camera, light, and controller are all within a single container and they are able to offer benefits such as object detection and multi-point inspection [30].

These sensors generally operate within the realm of two modes, e.g. the monochrome mode and the color mode. In the monochrome mode, the intensity of each pixel is represented in black and white which in turn determines the brightness and shape of the target [30].

In color mode, each pixel's brightness and intensity is divided in to red, green, and blue (RGB). This allows for the identification of the target in a more accurate way since subtle differences of color on the target are more easily identifiable [30]. The next few subsections will explore some key underlying concepts and eventually discuss the concept of visual odometry.

3.4.1 Visual Odometry

Odometry is the process of determing your position by counting steps or rotations. In biological terms, it is knows path integration and mammals utilize for dead reckoning. Similar to the odometer in a car which measures the distance traveled by multiplying the rotation of the tire by its circumference, odometry in robotics refers to determining the distance and the trajectory of a robot. They are determined in a single vector form for each time t with a combination of the euler angles and cartesian coordinates $[x, y, z, \phi, \theta, \psi]$ [31].

Visual Odometry is defined as *"the process of estimating the robot's motion (translation and rotation with respect to a reference frame) by observing a sequence of images of its environment"* [32] and falls under the field of computer vision. Due to advancements in computer hardware, on-board real time vision processing allows vehicles and robots to navigate autonomously. It is similar to odometry in that it uses cameras to determine the trajectory of the robot. The cameras can be attached to the robot in various different schemes and they utilize the incoming pictures/video to recognize visual features. These visual features can then be tracked incrementally and position and orientation can be determined with respect to an inertial reference frame [33].

VO is classified under the umbrella term Structure from Motion (SFM) which is a technique of using 2-D image sequences in order to reconstruct 3-D structures [32]. When utilized with one camera it is called Monocular Visual Odometry and when utilized with two cameras it is called Stereo Visual Odometry (see section 3.4.3) [33]. The latter has advantages such as robustness, determination of exact trajectory, etc., while the former is more useful in smaller robots although the trajectory is unique up to a scale factor [33]. These two can then be further subdivided in to feature tracking, which is tracking the same features in adjacent frames, feature matching, which is matching features in a given set of frames, and optical flow techniques, which analyzes pixel intensity in specific regions in consecutive frames [32].

The three most widely used VO motion estimation techniques are 3D to 3D, 3D to 2D, and 2D to 2D. In 3D to 3D, 3D feature points are tracked and triangulated in sequential images and then the 3D distance between these points is minimized in order to determine camera transformation. 3D to 2D estimation works similarly to 3D to 3D except the variable that is minimized is the 2D re-projection error. 2D to 2D estimation takes advantage of the epipolar geometry in order to determine camera estimation [32]. This process requires extensive math which will not be reproduced here. A basic flow of how VO is implemented in given through the following steps [32]:

1. Use triangulation to reconstruct 3D points at time t, extracted from features in the right and left frames

2. Match them with them with the same features in the next frames through Feature Matching

3. Calculate transformation and use the one with least sum of square difference
4. Remove outliers through RANSAC to get a more precise transformation
5. Combine the estimated and global transformations
6. Repeat for each time step t_s

3.4.2 Monocular Vision

As the name suggests, monocular vision or visual odometry utilizes a single camera to obtain information about the environment. Unlike stereo vision, monocular vision works by first locating features in a first frame, then by reobserving and traingulating 3D points in a second frame, and by finally, calculating the change in camera pose in a third frame [32]. In [34], the use of monocular vision for localization and target tracking is explored.

One disadvantage of monocular vision is that it is difficult to determine the scaling between the first two frames. Therefore, in order to get accurate transformation between the two frames a calibrated predefined constant is used [32]. In simple terms, unlike stereo vision, where it can be determined that the trajectory changed by x and y meters, in monocular vision the trajectory can only be determined in x and y units. Monocular VO does have an advantage when using smaller robots, such as micro UAVs, since the objects are generally too far away from the camera. In this case, the stereo vision degenerates [33].

3.4.3 Stereo Vision

In order to better understand vision sensors, understanding stereo vision is crucial as it is a widely used in autonomous navigation in conjuction with LIDARs (Light Detection and Ranging).

Stereo vision works off the same principle as our eyes do. It utilizes two horizontally displaced cameras (see figure 12) to create two images which are separated by a distance called the baseline (the spatial difference between the two cameras). When compared with each other, the images produce a relative depth map in the form of a disparity map. A disparity map shows the difference in coordinates between the

same object on the left and right camera in terms of pixels. Since depth is inversely proportional to disparity, the scene depth at each of these points can be deduced.

Figure 12: Stereo Camera by ZED [35]

In figure 13 we see a simple model of the functions of a stereo camera. For a point of interest P, the coordinates $P(x, y, z)$ can be determined with the principle of similar triangles (y-axis points out of page) using equations equation (9). Here, xl and yl are the known x and y coordinates from the left camera and xr and yr are the known x and y coordinates from the right camera. For a camera with a known focal length, f, baseline, b and with 3 equations and 3 unknowns, we can solve x, y, and z as shown in equation (10) [36].

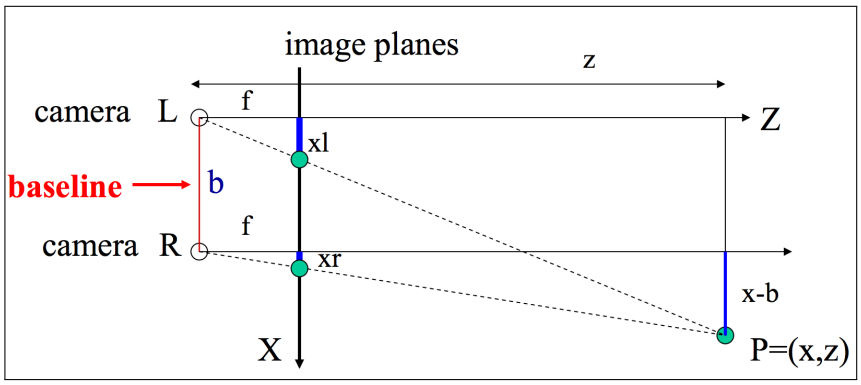


Figure 13: Simple Model of Stereo Vision [36]

$$\frac{z}{f} = \frac{x}{xl} \qquad \frac{z}{f} = \frac{x-b}{xr} \qquad \frac{z}{f} = \frac{y}{yl} = \frac{y}{yr} \tag{9}$$

$$z = \frac{f \cdot b}{xl - xr} = \frac{f \cdot b}{d} \qquad x = \frac{xl \cdot z}{f} = b + \frac{xr \cdot z}{f} \qquad y = \frac{yl \cdot z}{f} = \frac{yr \cdot z}{f} \tag{10}$$

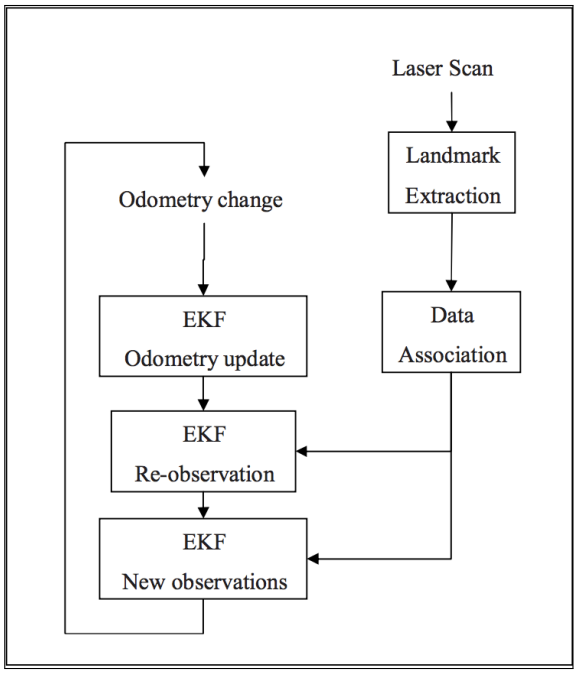
3.4.4 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping, commonly called SLAM, refers to a process utilized by vehicles and robots to build a map of an unknown evnvironment in order to locate themselves and navigate this environment [37]. It is more a set of algorithms which work inconjuction with each other in order to solve the SLAM problem [38]. The localization and mapping happens through the use of sensors which, as previously mentioned, allows the robot to determine its state which consists of its position and orientation. The mapping consists of tracking notable features in the environment and then using those features to navigate as well reset any localization error, also known as loop-closure [37].

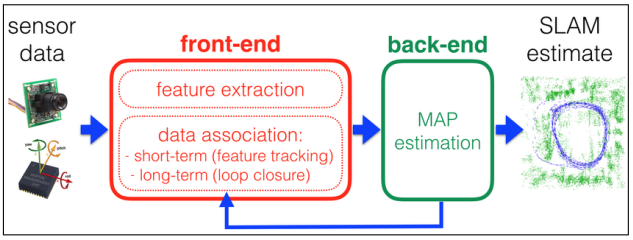
In the SLAM process, an estimator is typically implemented (Maximum a posteriori (MAP), Extended Kalman Filter (EKF), etc.), in order to estimate the position of the robot [37, 39]. The SLAM system typically consists of a front-end, which uses the incoming sensor data to build useable models, and a back end which draws from these models in order to provide useable information [37]. Figure 14a shows the process flow of a SLAM system implemented with an EKF while figure 14b shows the how the front and back end work with each other.

A more relevant area of SLAM in the field of computer vision that has gained popularity fairly recently is visual SLAM or vSLAM. vSLAM is combination of visual odomotery combined with a global map optimization and can be used both with stereo vision and as monocular vision (MonoSLAM). It has become more widely used due to the availability of inexpensive video sensors [32]. It differs from VO in such that it tries to build an optimal global map of the environment where as VO only considers a small portion of the environment at a time. Therefore vSLAM can be said to be comprised of the following five modules [40]:

- Initialization: global coordinate system is defined and an inital map is built using this coordinate system
- Tracking: feature tracking or matching is used to obtain 2D-3D relation between image and the tracked reconstructed map
- Mapping: unknown features of the environment are tracked



(a) Process Flow of a SLAM System [39]



(b) Front and Back End of a SLAM System [37]

Figure 14: SLAM System High-Level Flow

- Relocalization: occurs in case of failure in the tracking module
- Global Map Optimization: reduces acculumated error in the map estimation

There are other varieties of vSLAM that exist such as cvSLAM (Ceiling Vision) and FastSLAM. In cvSLAM, the camera is pointed towards the ceiling and features of the ceiling are tracked as proposed in [41]. In FastSLAM, the EKF is applied to each feature and each particle as described in [42].

3.5 Sensor Fusion

Individually, each sensor is not enough to provide a full scope of useful data to the drone. The data must be combined in order to produce some sense out of it. This is where the motivation behind sensor fusion arises from. In most cases, each sensor is unable to provide sufficient information due to limited spatial coverage (sensor only covers a certain area), limited temporal coverage (gyro data is reliable over longer periods of time), imprecision (each sensor has noise or bias), etc.

Inertial Measurment Units are a typical example of a sensor that might seem perfect on paper, but in real world applications, have very little tolerance for outside noise. They have been known to "lock-up" in conditions such as high-speed ($\geq$ 60kph) flying on a quadcopter with no rubber cushions to absorb the vibrations [43]. These vibrations can resonate at the same frequency as the micromechanical parts causing them to output incorrect data. It is valid to think that the IMU experiences lots of distrubances next to multiple motors rotating at many thousands of rotations per minute. Figure 15 shows the flow diagram for typical sensor fusion.

One way to overcome such noise is to add more sensors that can compensate for these disturbances and then fuse them using a software package or algorithm. This can provide us with estimated orientation data that is smooth and bias-free while also being computationally inexpensive [45]. There are four algorithms which are commonly used in UAV based applications. They are the complementary filter, the kalman filter, the Mahony, and the Madgwick Filter.

In order to output the orientation, the filter algorithm generally converts the incoming data in to either Euler angles, quaternions, or in to a direction cosine matrix. Of the three, quaternions are the easiest and least computationally expensive

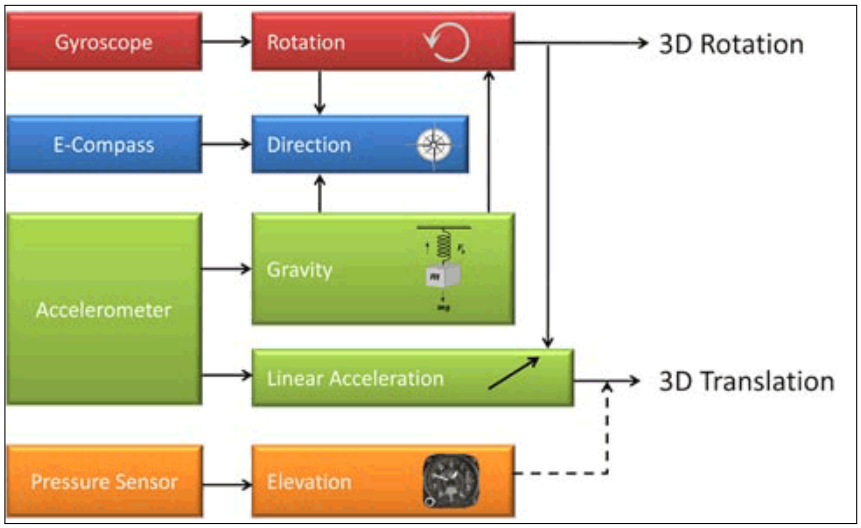


Figure 15: Sensor Fusion Flow Diagram for 6-DOF IMU [44]

because their lack of a singularity point ("gimbal lock" in Euler angles) and because they can be represented linear (DCM requires a 3x3 matrix) [45].

3.5.1 Complementary Filter

A Complementary Filter is a combination of a low-pass and high-pass filter. The low-pass filter filters the high frequence signals such as the vibration from an accelerometer and the high-pass filter filters the low-frequency signals such as the drift bias from the gyroscope [46, 47]. The simple form of the complementary filter is shown in equation (11)

$$angle = 0.98 \cdot (angle + data_{gyro} \cdot dt) + 0.02 \cdot (atan2(data_{acc})) \tag{11}$$

In figure 16, it is shown that the sensor data from the accelerometer is being filtered such that only low-frequency signals are allowed to pass. This is because the accelerometer data is noisy in the beginning and is only stable over a longer period of time. The gyro data on the other hand is reliable short-term but drifts away from the true value over a longer period of time.

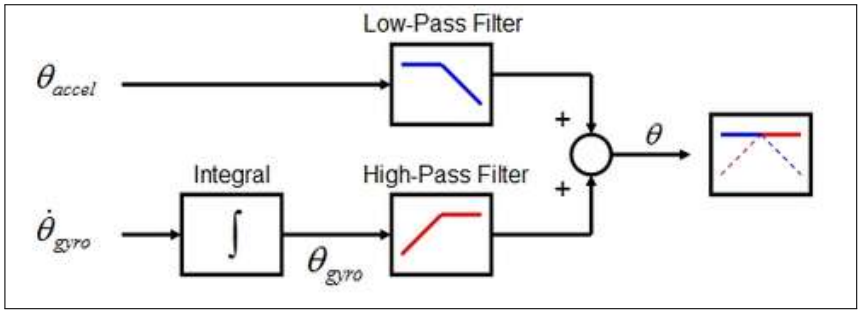


Figure 16: Complementary Filter

3.5.2 Mahony and Madgwick Filter

Mahony filter is a form of the complementary filter known as an explicit complementary filter. It corrects gyroscope bias by cross-multiplying the acceleration and magnetic field vectors (measured and estimated) and using the result to calculate error [48]. Madgwick filter is an optimized gradient descent algorithm which uses quaternions to calculate error in order to correct gyro drift [48].

3.5.3 Kalman Filter

In simple terms, a kalman filter uses information about the previous state of the system in order to predict the future state of the system. It is more on the complex side of the filter scale in terms of application and execution. The accuracy it provides is counteracted by it's computational needs. Understanding the kalman filter is necessary in order to understand how they are implemented on drones especially in our case. In the next few paragraphs, a brief and simple explanation of the filter is given.

In figure 17, the information flow of a kalman filter is given. The process starts out with the current state (represented in blue) $\hat{x}_{k-1}$ and the covariance matrix at the current state P_{k-1}. The covariance matrix tells us the correlation between each of the state variables to the other state variables. In other words, each element of of the matrix Σ_{ij}, states the degree of correlation between the i_{th} state variable and the j_{th} state variable and this relationship is symmetric. Next, the governing physical equations of the system are used to predict the future state (pink) using the prediction matrix F_k. Using matrix multiplication identity, we can derive the following equations [49]:

$$\hat{x}_k = F_k \hat{x}_{k-1} \tag{12}$$

$$P_k = F_k P_{k-1} F_k^T \tag{13}$$

In order to refine the prediction, a control matrix B_k and a control vector $\vec{u}_k$ are added which represent external changes that do not have direct influence on the states. Usually these are omitted in simple systems. The matrix Q_k which gets added to the covariance matrix represents the uncertainty. In real world system, the plant can go from its current state to many other states each inside a guassian distributed region with covariance Q_k. Another way of thinking of about this matrix is to treat as unknown noise with variance Q_k. The final result is the equations (14) and (15) [49].

$$\hat{x}_k = F_k \hat{x}_{k-1} + B_k \vec{u}_k \tag{14}$$

$$P_k = F_k P_{k-1} F_k^T + Q_k \tag{15}$$

Finally, we must update the prediction with our measurements from external sensors. The measurements from the sensors is represented through the matrix H_k and the covariance matrix (sensor noise or uncertainity) is represented with the matrix R_k with a mean of $\vec{z}_k$. However, the measured data represents a different gaussian region than the predicted data. This means that the future state can lie anywhere in these two regions. To get a more accurate grasp of where the future states might be, these two gaussian distrubtions can simply be multiplied in order to find a region where they overlap. This overlapped region is where the future state will most likely lie. After multiplication, the two aforementioned matrices in conjuction with state covariance matrix P_k produce K, also known as the kalman gain. Using this kalman gain we can update our predicted state to get the best estimated future state [49]. This future state is fed back in to the beginning of the loop and the cycle is repeated. An updated equation of the kalman filter is shown below.

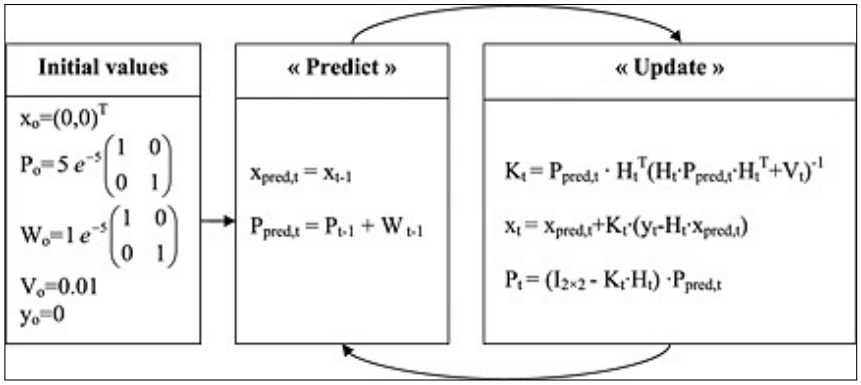


Figure 17: Kalman Filter Information Flow [49]

The kalman filter allows for the modelling of almost all linear systems accurately. There is an extension of this filter called the Extended Kalman Filter which is used for non-linear systems. In an EKF, the predictions and the measurements are linearized about their mean and implemented in a similar fashion.

For highly non-linear systems, the unscented kalman filter (UKF) and the sigma point kalman filter (SPKF) are used. They are both on the same level of complexity as the extended kalman filter but address some errors in the posterior mean and

covariance of the transformed Gaussian Random Variable (GRV) obtained through the first-order linearization of the non-linear system [50]. Another variation is iterated EKF (IEKF) which improves on the EKF by linearizes the measurement equation iteratively until convergence [32].

4 Algorithms

Vision Sensors utilize a multitude of problems and alogrithms in order to extract information out of the images they capture. In the next few sections, a few methods and algorithms will be explored which help a robot make sense of its environment and help it locate itself in the environment.

4.1 RANSAC

RANdom SAmple Consensus is a iterative algorithm used to estimate parameters for a model fit when many outliers are present in the data. It is widely used in computer vision to relate corresponding points in a stereo image, to compute similarity between two points in object detection, and for video stabilization. The RANSAC algorithm is highly robust (high accuracy) in estimating the parameters of the model even when the data is noisy [51].

The RANSAC algorithm can be broken down in to the following steps given M data points, K tolerance, and I iterations [51]:

1. Select N data points from M
2. Estimate model parameters using these points
3. Find number of data points in M not in N that fit the model within K and add them to X
4. If X is large enough exit
5. Otherwise, repeat I times

4.2 Bundle Adjustment

When working with feature based visual 3D reconstruction, the problem of bundle adjustment is always encountered. It is "*the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates.* [52]" The name is derived from the bundles of light that leave each feature and are captured by the camera and then are optimally adjusted [52]. The problem concerns itself minimizing some cost function usually through a non-linear least squares method. Figures 18 and 19 [53] provide an informative illustration about how bundle adjustment works. In figure 19, the figure on the left is before bundle adjustment and the figure on the right is after bundle adjustment.

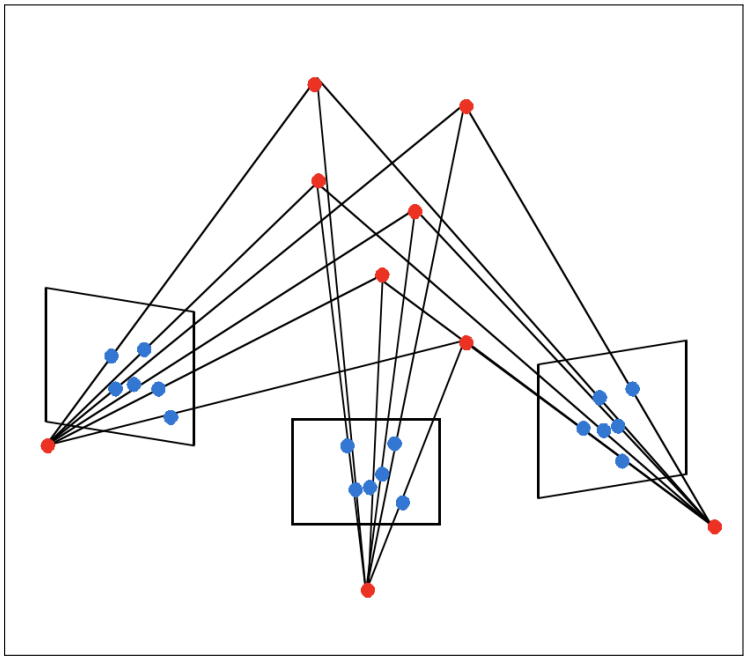


Figure 18: Capturing Bundle of Light-Rays for a Scene Point

4.3 Optical Flow

Optical Flow estimation is a technique to determine motion from adjacent frames. If it can be estimated between two video frames, the velocities of the objects in those frames or the camera can be determined. Through optical flow, a vector field can be constructed which displays the change in the pixel location of a target object. The use of optical flow has seen many recent developments in the field of UAVs and

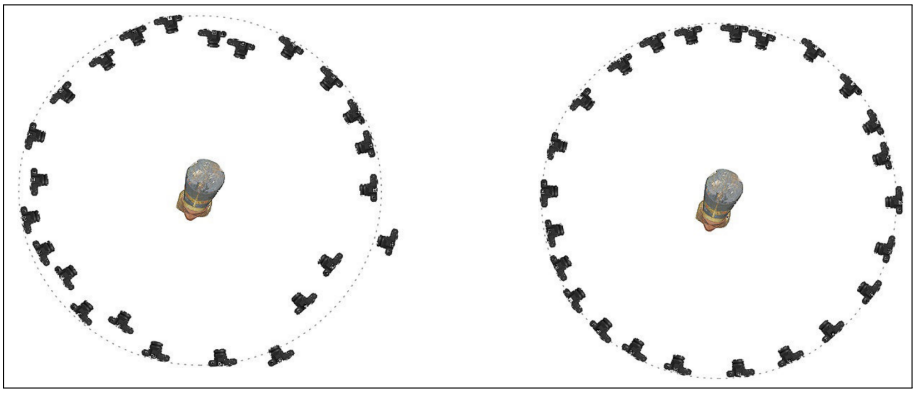


Figure 19: Before and After Bundle Adjustment

especially micro UAVs as it allows them to navigate in indoor environments where GPS might be degraded or denied and it is useful in obstacle avoidance [54]. It is a technique that is also utilized by the Guidance navigation system which was utilized for this project.

The most commonly used alogrithm for optical flow estimation is the Lucas-Kanade algorithm. It is a differential method which through the use of the following assumptions solves the optical flow problem for pixels in a neighborhood through the least squares technique. It works off of three assumptions: first is the "Intensity Coherence" assumption which states that the between consecutive frames the intensity of a pixel does not change. Second is the assumption neighboring pixels have similar motion. The final assumption is that the video is being sampled at a frequency fast enough to display motion over time. Other methods such as the Horn-Schunk method, block matching algorithms, feature-based detectors, etc. also exist [55].

4.3.1 Feature Detectors

Feature detection algorithms work by finding a unique feature in an image such as a corner or an edge that can be easily identified, tracked, and most importantly are repeatable in sequential images. Most optical flow algorithms use a combination of a differential method (Lucas-Kanade or Lucas-Kanade-Tomasi) with a feature detection and a feature descriptor algorithm for image matching and recognition, localization, and SLAM.

In the drone used for this project, the feature detector algorithm that was used was Feature from Accelerated Segment Test (FAST). FAST was first introduced in [56] by Edward Rosten and Tom Drummond and is one of the most commonly used feature detector algorithm for real time applications due to its computational effectiveness as shown in [55]. It works by placing 16 pixels around a central pixel and then compares the intensity of each of these pixels with the central pixel. Based on this analysis, the region is defined as either a corner, an edge, or uniform. A pixel is defined as a corner if a set of contiguous pixels have an intensity higher or lower than some given threshold. The biggest difference between FAST and other corner detectors is that it is derived using machine learning which cuts the computational cost of the algorithm drastically [56] and it must be used in conjuction with a feature

descriptor.

4.3.2 Feature Descriptors

A feature descriptors is a computers way of describing the features it detected in order to identify them in the next images. Most feature detectors either have a built in feature descriptor or work in conjunction with one in order to fully extract feature information from an image. The feature descriptor that was utilized in this project by the drone was Binary Robust Independent Elementary Features (BRIEF). It is again used for similar reasons as the FAST feature detector which is for it's computational efficiency and inexpensiveness.

Most feature descriptors create multi-dimensional vectors consisting of floating-point numbers to represent features. When tracking many different features, these vectors can occupy large chunks of memory. BRIEF, on the other hand, bypasses this steps by selecting unique pairs of x and y locations from an image patch and does intensity difference on these selected pairs. This reduces the computational cost heavily.

Other feature detector and descriptor algorithms also exist and they are summarized below. Comparisons done in [55, 57] show the difference between the different algorithms and are represented in figure 20 and table 1. In figure 20, the scale and orientation are represented by the size of the circle and lines, respectively. In FAST and HARRIS, the circles are small because these algorithms are not rotation invariant [57]. In table 1, image analysis was performed on a 378x240 pixel image and the time taken is shown. In the results, FAST takes the shortest time to extract features from the image.

- SIFT: Scale Invariant Feature Transform is a widely used feature detector and descriptor. It consists of 128-dim vectors which are used to track features. SIFT features are extracted using Difference of Gaussian (DoG) and are rotationally invariant

- SURF: Speeded Up Robust Features improves on SIFT's high use of time and computer efficiency by using box filters instead of DoG to extract features. It is however a little less robust than SIFT

- Harris: A corner detector that works by finding intensity difference for a displacement
- Shi-Tomasi: also a corner detector which improves on the Harris algorithm but works in a similar manner
- ORB: Orientated FAST and Rotated BRIEF is feature detector and descriptor which combines both FAST and BRIEF algorithms

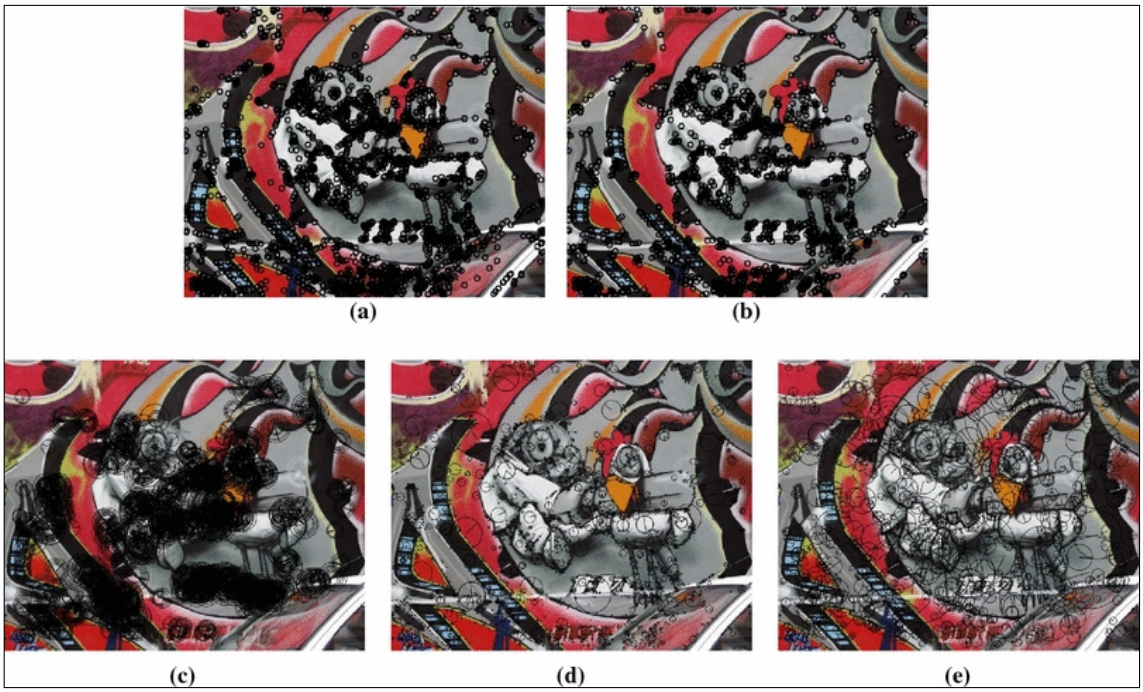


Figure 20: A Comparison Between Different Feature Detectors and Descriptors. a) FAST b) HARRIS c) ORB d) SIFT e) SURF [57]

Algorithms	**Shi-Tomasi**	**FAST**	**SIFT**	**SURF**
Time(s)	0.016	0.0041	0.0334	0.0236

Table 1: Time Comparison of different feature extractors on 378x240 px image [55]

5 Implementation

5.1 Overview

This section will concern with the implementation of the previous topics on the the DJI Matrice 100. It is a fully programmable modular quadcopter designed by China based company DJI. It has a wheelbase of 650mm with a total weight of 2355g with the battery and no other payload. It can carry a maximum take-off weight of up to 3600g which can include additional sensors, cameras, etc. As seen in figure 21, the base model comes with an expansion bay, a battery compartment, and a GPS module. The drone is popular among researchers due to its modularity and programmability.

Figure 21: DJI Matrice 100

5.2 Hardware and Software

In addition to the quadcopter several other pieces of hardware and software were also required in order to implement the work done in [5]. Most of the parts required were off the shelf although some were made in house. The software utilized was all open source.

5.2.1 DJI M100 Specifications

The DJI M100 is powered by the TB47D Intelligent Flight Battery with a capacity of 4500mAh and an output of 22.2V. It is used to power all the sensors on the copter including the onboard computer. The battery firmware is capable of displaying battery capacity in real-time to the user and has algorithms to calculate the distance the aircraft has travelled and how much battery capacity it will take to return to the home point. The copter also has XT30 and XT60 universal power ports which allow for external sensors to utilize the battery as a power source. In addition, there are several UART ports for serial communication with the onboard computer in order to send flight information data.

In order to fly the M100, a smart device capable of running the DJI Go Application is needed. The app displays the view from the camera as well as the location of the copter on a map (with GPS) and other relevant data. The smart device connects directly to the remote controller in one of its USB inputs.

The remote controller for the DJI M100 works in conjunction with the DJI Go Mobile App to fly the quadcopter. It allows the quadcopter to fly in three modes, Positioning (P), Attitude (A), and Function (F) mode. The F-mode is the one that is of interest because it allows for other applications to run on the copter.

Another application needed to test the copter is the DJI Assitant. The application allows for the user to change several parameters such as the baud rate, the frequency to publish other data such as velocity, acceleration, etc. But, most importantly it contains the DJI simulator which allows for the testing of self-written application in a simulator before being tested in a real environment. The copter simply needs to be connected to a computer running the application and all other operations remain the same.

5.2.2 DJI Software Development Kit

One of the main and crucial open source softwares used for this project were the DJI Software Development Kits (SDK), both for the onboard computer and for the groundstation. The SDKs are designed to allow registered developers to fully access the capabilites of the quadcopter by running their own applications (e.g. the one for this project) on the drone. The two SDKs that were used were the Onboard-SDK

(OSDK) for the onboard computer which was mounted on the copter and ROS-SDK which was used for the base station.

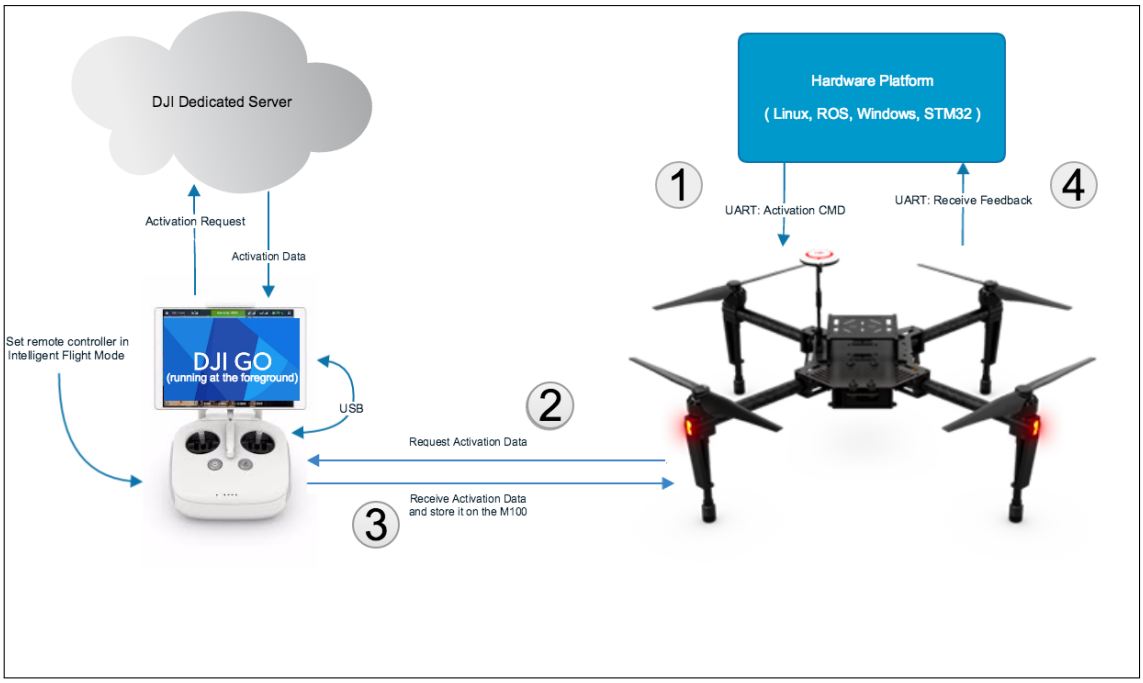
The OSDK provides a C++ library which can be used to interact with the DJI M100 through a serial interface on a Linux, ARM, or STM32 platform. It also comes with a ROS wrapper which makes integration in to ROS very easy. With the OSDK, the developers can control the copter without a remote controller, install third party sensors, and execute precise trajectories. Furthermore, through the OSDK, the copter's attitude, velocity, and position can be controlled on a low-level and GPS, RTK, compass, barometer, battery status, quaternions, linear acceleration, and angular rate data can be acquire at 200Hz. For this project, the OSDK was on a Intel NUC (see section 5.2.4) running Ubuntu 16.04.

The ROS SDK works in conjunction with the OSDK and allows developers to access all of ROS's capabilities through its messeages and services. By developing a ROS launch file for the app containing the valid developer key and baud rate (for the data), the app itself can then be run through ROS once it is verified by the DJI server. The sensor data including visual data from Guidance can also then be accessed for storage and analysis purposes. For this project, the Kinetic Kame flavor of ROS was run on top of the aformentioned linux operating system. The full validation cycle along with the process flow of OSDK can be seen in figure figure 22.

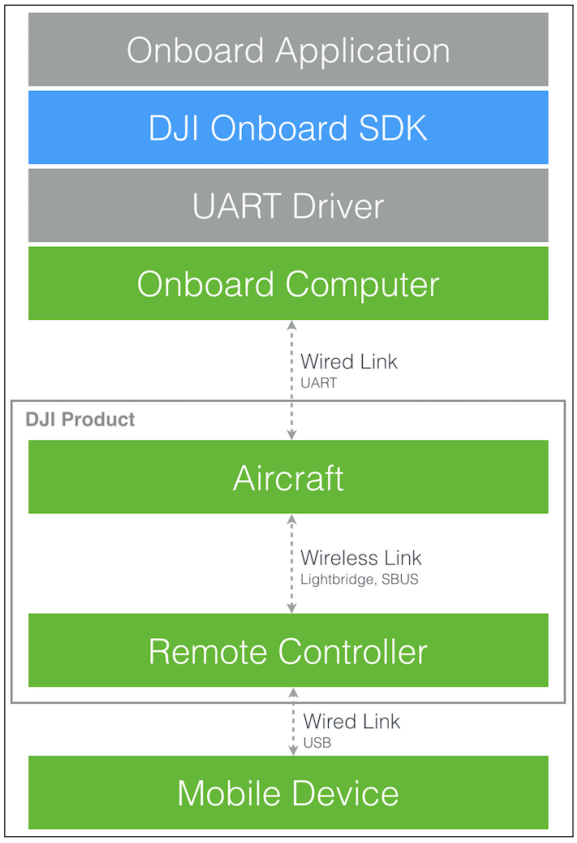
5.2.3 Guidance

Instead of using the Intel RealSense as used in [5], the Guidance system developed by DJI was instead used. The Guidance is vision tracking and collision avoidance system that provides almost 360° of obstacle avoidance. It consists of five stereo cameras units which are coupled with an ultrasonic sensor which help reduce the errors of vision-based algorithms. More specifically, each sensor unit consists of two mono-color global shutter cameras capable of capturing images in VGA resolution (640x480).

In addition, the Guidance system consists of a central processing module which is able to connect up to five sensor units. The module is made of a 6-DOF (gyro plus accelerometer) inertial sensor manufactured by TDK (MPU 6050) and an Altera Cyclone V SOC FPGA. The sensors are mounted on the copter so they face the



(a) Process of App Activation for Developer Application

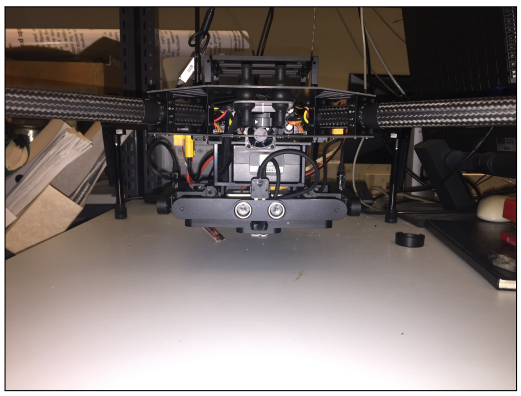


(b) Flow of Communication between OSDK and DJI M100

Figure 22: DJI Guidance

following directions: front, back, right, left, and downward. The mounted setup along with sensor modules are shown in figures 23a and 23b [58].

(a) Guidance Sensor Units along with Guidance Core



(b) DJI Matrice 100 with Guidance Vision System

Figure 23: DJI Guidance

The software side of Guidance works by processing 10 undistored and rectified 20Hz 8-bit grayscale image data captured at QVGA (Quarter-VGA: 320x240). This resolution is used because it is more optimal for real-time applications. The caputred images are remapped in to a visual odometry thread which is used for localization, and a visual mapping thread for obstacle avoidance system. For the former, FAST feature detector, BRIEF feature descriptor, and binary image mapping are utilized to extract and analyze features. For the second thread, a pair of stereo sensors are isolated through a multiplexer based on the direction of movement of the quadcopter. Figure 24 demonstrates the functionatilies described above. In the figure, the gray is system inputs, the blue are FPGA based hardware kernels, purple are ARM-based software kernels, and green are system outputs [58].

The Guidance core module contains algorithms for visual odometry, visual mapping, and for sensor fusion. The matching refinement is done through a combination of BRIEF, FAST, and a non-pyramid Lucas-Kanade algorithm. The 2D-2D motion estimation is done through the epipolar geometry, and the sensor fusion is done through a modified EKF. The visual odometer of the Guidance system is shown in figure 25 [58].

DJI also provides an open-source software development kit (SDK) which can be

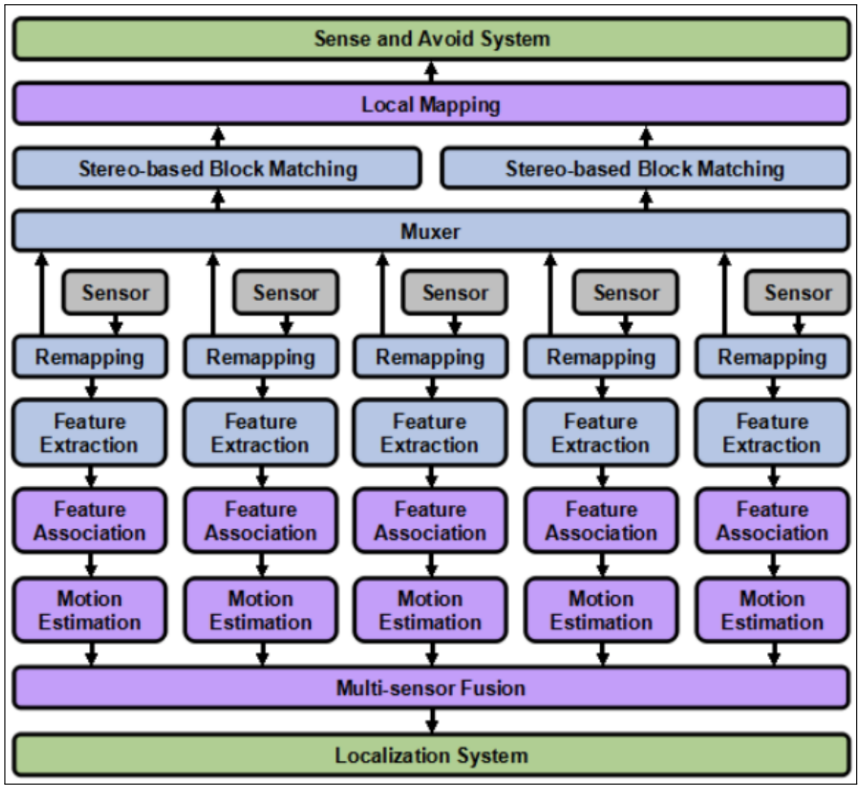


Figure 24: Block Diagram of Guidance's Functionalities [58]

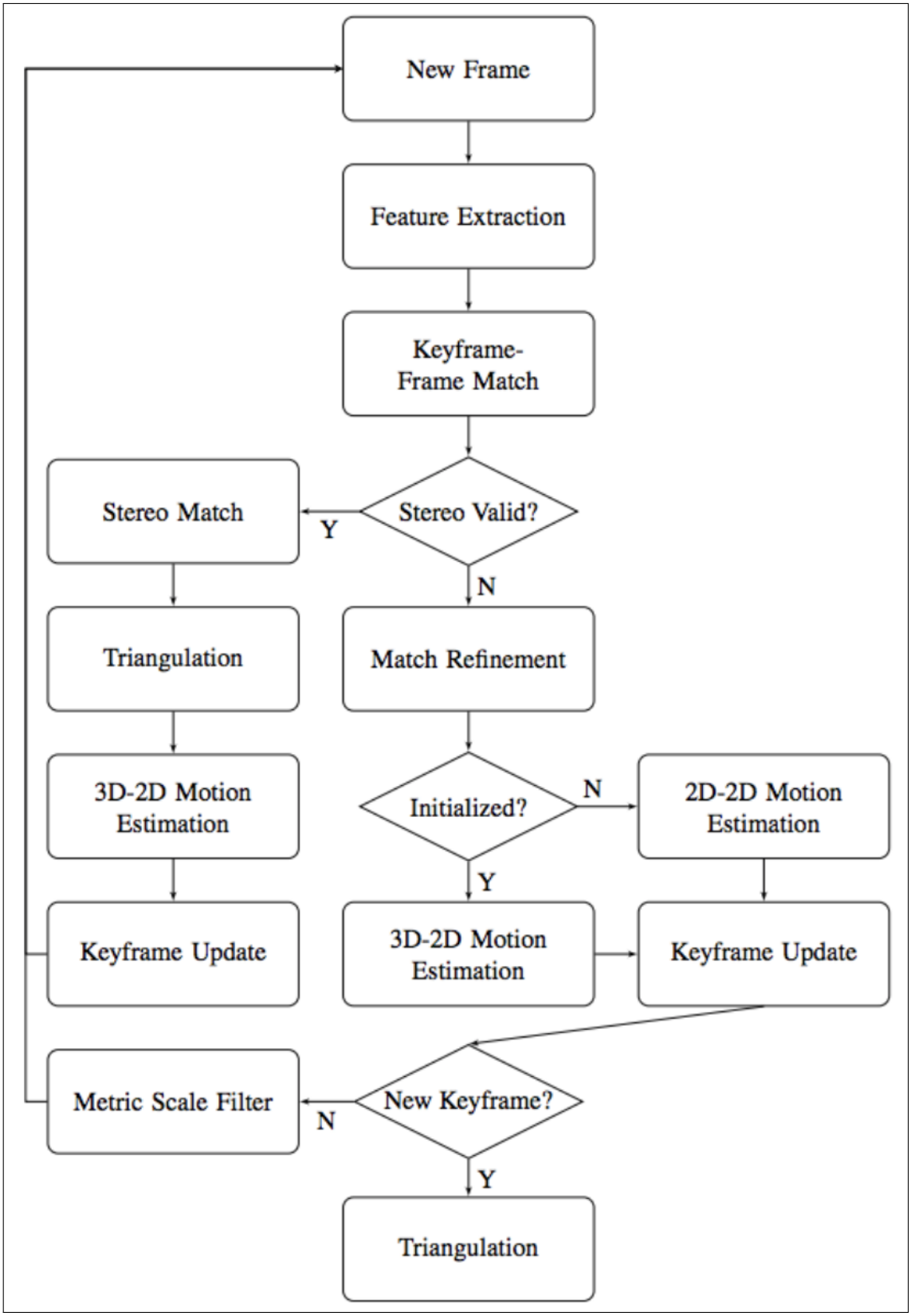


Figure 25: Block Diagram of Guidance's Functionalities [58]

used to obtain various types of raw or processed data from the quadcopter through a USB 2.0 or UART interface. These can then be used with ROS or OpenCV for further development. In addition, the sensors can be calibrated to user specification, low-speed data can be accessed via UART, bandwidth of the data flow can be adjusted, and camera exposure time can be set. Other data that can be obtained from the Guidance SDK is shown in table 2 [58].

Data	**Description**
Error Code	Enumerates error code which can be used for trouble shooting
Velocity Data	Velocity in body frame in **millimeter/sec** at max 10Hz
IMU Data	Acceleration data in **g** and gyroscope data in quaternions
Motion Data	Quaternion orientation and position and velocity in global frame
Ultrasonic Data	Obstacle distance in **m** and realiability of data. Both at 20Hz
Image (10 Channels)	QVGA 8-bit greyscale images at max 20Hz
Depth Map (2 Channels)	QVGA 16-bit depth images at max 20Hz
Disparity Map (2 Channels)	QVGA*2 bytes disparity images at max 20 Hz
Obstacle Distance (5 Channels)	0.1 - 20m distance data at max 20 Hz

Table 2: Available Data from Guidance SDK [58]

5.2.4 Onboard Computer

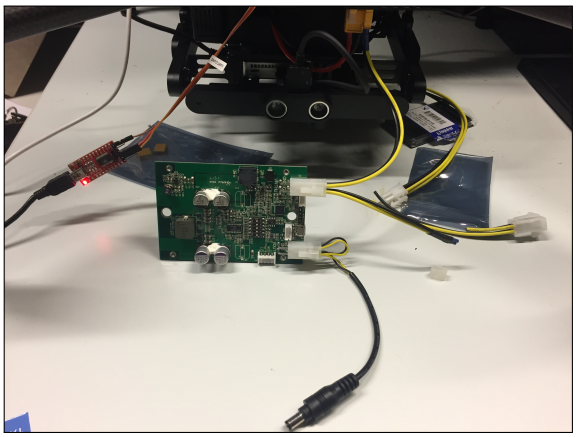
The next important piece of hardware that was utilized was an Intel Next Unit of Computing (NUC). This is a small portable computer that is used as the on-board computer for the quadcopter. An onboard computer is needed in order to process and store the incoming flow of images. Therefore, the specifications of the computer were created so to match the needed processing and storage power for the data. The

specs for the onboard computer used in this project are given in table 3.

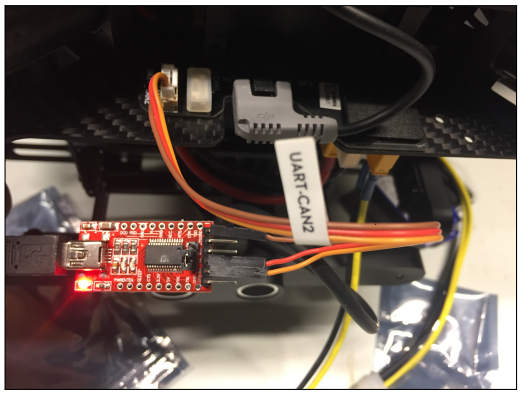
Item	Description
Processor	Intel i7-RYH
Memory	16GB
Storage	500GB SSD

Table 3: Intel NUC Specifications

The onboard computer receives its power directly from the copter's battery. The copter outputs a power of 20-26.1V at 1A from its XT30 and XT60 connectors, while the onboard pc has a power requirement of 12-19V. In order to ensure that the computer received the right voltage and current a DCDC converter was used (see figure figure 26a). This DCDC converter was designed specifically to power NUC devices and was the most compatible and safest product found on the market. The converter is capable of providing either 12V or 19V and this can be adjusted through a switch on the converter itself. As seen in the figure, the power from the copter goes in the Vin port and the power from Vout port can be directly plugged in to the NUC.



(a) Mini-Box DCDC NUC Converter [59]



(b) FTDI TTL to USB Converter

Figure 26: DJI Guidance

For communication with the copter an FTDI USB to TTL converter is needed. This is serial to USB converter which allows the computer to read the incoming data from the UART port. The configuration is shown in figure 26b. The UART cable

has four cables, Rx, Tx, Vcc, and Ground and they are for receiving, transmitting, power, and ground, respectively.

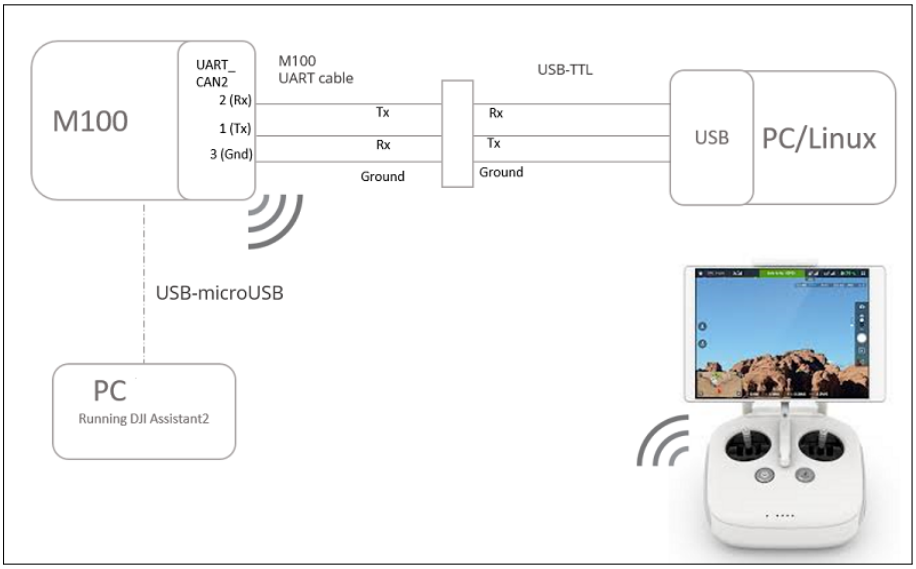


Figure 27: Setup of USB to TTL Chip

For the mounting of the NUC to the copter, there were two options available. The first one involved mounting the PC as is using velcro (for indoor testing this works well) while the other one involved using a homebrewed 3D printed case. With excess sensors and/or outdoor testing, it is necessary to remove the heavy manufactured casing of the computer and use a lighter one. The 3D printed case also makes it easier to mount the pc on to the copter. Itis shown in figure 28.

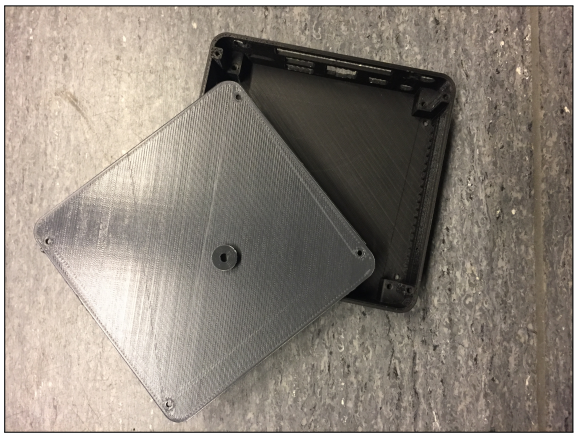


Figure 28: 3D Printed Case for the Intel NUC

5.3 Testing and Results

After the hardware setup, the software side was setup. This began by setting up a a third computer running a MacOS terminal (a linux machine could also be used). Both the NUC and the aformentioned computer were on the same network which allowed for the creation of a secure shell (SSH) connection with the NUC. Through this, remote commands can be sent to the NUC as long as the computer can access the same network as the ground computer running the terminal.

Next, ROS was setup on the NUC along with the Onboard SDK and ROS SDK, which were downloaded from DJI's Github repository. Additional pieces of software were also needed which were all installed from different Github repositories. In order to see if the hardware and software were working properly a sample mission was executed and the results were observed in the simulator.

Following this, a ros executable designed specifically for the copter and nuc was launched using the roslaunch command. This allowed us to launch the sdk client from which several different types of commands, such as takeoff, landing, drawing shapes, etc., can be sent to the copter. An example of the sdk client is shown in figure 29a. The data from these sample missions can be extracted using the rostopics that the sdk makes available. The data can be recorded in a bag file and played back using the rosbag command. The available topics are shown in figure 29b. The data is extremely useful while doing manual flights as it can be used for system identificaton. Some of the commands were then tested on the flight simulator to verify that the hardware was working properly and that the onboard computer was able to communicate with the quadcopter.

The next step was to do create a model for the copter through system identification. In order to do this, first a bluetooth game controller was needed to fly the quadcopter. The reason a game controller is used is because the inputs from the controller can be recorded and it is possible to see the roll, pitch, and yaw rate along with the thrust. For system identification the data which is needed as output is IMU data and velocity data which corresponds to topic /dji_sdk/imu and /dji_sdk/velocity. The data was recorded using rosbag and was parsed through using two MATLAB files, one for estimating roll and pitch dynamics and one for estimating yaw dynamics. Two experiments were run in order to record two sets of data, one for training and one

```
+------------------------ < Main menu > ------------------------+
| [1]  SDK Version Query         | [20] Set Sync Flag Test          |
| [2]  Request Control           | [21] Set Msg Frequency Test      |
| [3]  Release Control           | [22] Waypoint Mission Upload     |
| [4]  Takeoff                   | [23] Hotpoint Mission Upload     |
| [5]  Landing                   | [24] Followme Mission Upload     |
| [6]  Go Home                   | [25] Mission Start               |
| [7]  Gimbal Control Sample     | [26] Mission Pause               |
| [8]  Attitude Control Sample   | [27] Mission Resume              |
| [9]  Draw Circle Sample        | [28] Mission Cancel              |
| [10] Draw Square Sample        | [29] Mission Waypoint Download   |
| [11] Take a Picture            | [30] Mission Waypoint Set Speed  |
| [12] Start Record Video        | [31] Mission Waypoint Get Speed  |
| [13] Stop Record Video         | [32] Mission Hotpoint Set Speed  |
| [14] Local Navigation Test     | [33] Mission Hotpoint Set Radius |
| [15] Global Navigation Test    | [34] Mission Hotpoint Reset Yaw  |
| [16] Waypoint Navigation Test  | [35] Mission Followme Set Target |
| [17] Arm the Drone             | [36] Mission Hotpoint Download   |
| [18] Disarm the Drone          | [37] Enter Mobile commands mode  |
| [19] Virtual RC Test           |                                  |
+-------------------------------------------------------------------+
```

(a) SDK Client For DJI M100 [5]

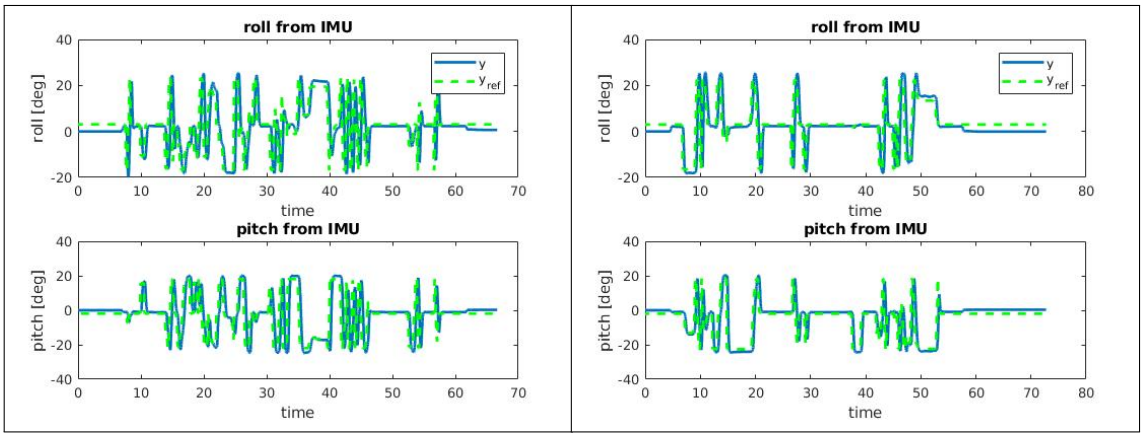
```
/dji_sdk/A3_GPS
/dji_sdk/A3_RTK
/dji_sdk/acceleration
/dji_sdk/activation
/dji_sdk/attitude_quaternion
/dji_sdk/compass
/dji_sdk/data_received_from_remote_device
/dji_sdk/drone_task_action/cancel
/dji_sdk/drone_task_action/feedback
/dji_sdk/drone_task_action/goal
/dji_sdk/drone_task_action/result
/dji_sdk/drone_task_action/status
/dji_sdk/external_position
/dji_sdk/external_transform
/dji_sdk/flight_control_info
/dji_sdk/flight_status
/dji_sdk/gimbal
/dji_sdk/global_position
/dji_sdk/global_position_navigation_action/cancel
/dji_sdk/global_position_navigation_action/feedback
/dji_sdk/global_position_navigation_action/goal
/dji_sdk/global_position_navigation_action/result
/dji_sdk/global_position_navigation_action/status
/dji_sdk/imu
/dji_sdk/local_position
/dji_sdk/local_position_navigation_action/cancel
/dji_sdk/local_position_navigation_action/feedback
/dji_sdk/local_position_navigation_action/goal
/dji_sdk/local_position_navigation_action/result
/dji_sdk/local_position_navigation_action/status
/dji_sdk/mission_event
/dji_sdk/mission_status
/dji_sdk/odometry
/dji_sdk/power_status
/dji_sdk/rc_channels
/dji_sdk/time_stamp
/dji_sdk/vc_cmd
/dji_sdk/velocity
/dji_sdk/waypoint_navigation_action/cancel
/dji_sdk/waypoint_navigation_action/feedback
/dji_sdk/waypoint_navigation_action/goal
/dji_sdk/waypoint_navigation_action/result
/dji_sdk/waypoint_navigation_action/status
/fcu/command/roll_pitch_yawrate_thrust
/keyboard/keydown
/rosout
/rosout_agg
/tf
```

(b) Different ROS Topics Available for Analysis

Figure 29: DJI Software Development Kit

for testing. Since the experiments were performed indoors, the quality and amount of data which could be recorded were limited.

First, the first order dynamics were estimated. The roll and pitch from the IMU for both data sets is shown in figure 30. They were obtained by reading the ROSbag files described earlier. In figure 31, the pitch and roll moments were fitted with a cubic spline fit with a 1-D estimation. The approximate fit for Experiment 1 came to be 71% for Experiment 1 and 65% for Experiment 2. Finally, in figure 32, the first order transfer function for the roll and pitch dynamics along with the gains can be seen. The use of MPC can be seen from the estimation of the gains. Similar analysis was done on the yaw rate as well.

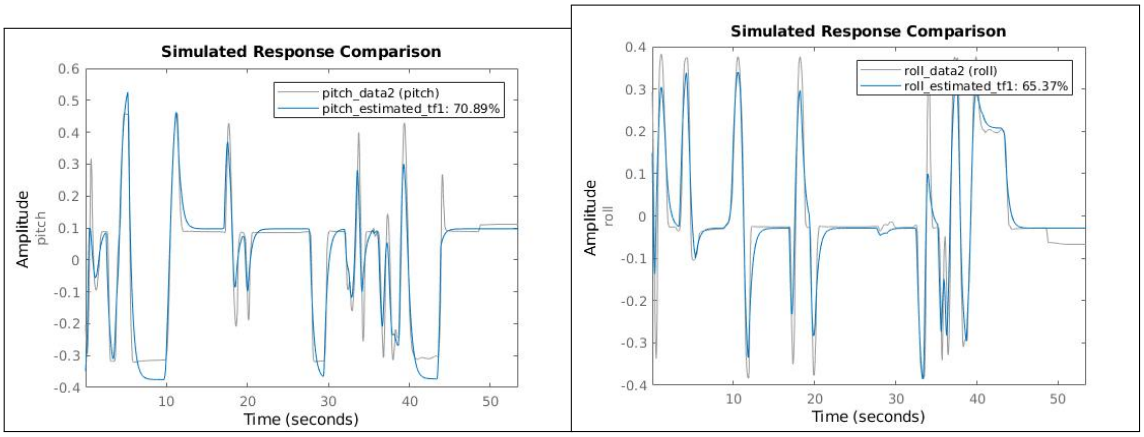


(a) Pitch and Roll Moments from Experiment 1

(b) Pitch and Roll Moments from Experiment 2

Figure 30: Pitch and Roll from the IMU for Both Experiments

Next, a second order estimation was done in order to improve the results. As can be seen the pitch was estimated to approximately 85%. A higher order model allows for a better fit, but this is not always the case. The second order transfer function and gains are shown in figure 33. Again, similar analysis was done on the yaw rate estimation.

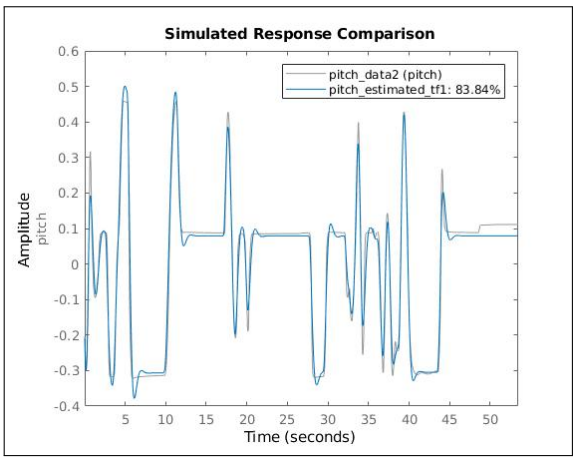


(a) Estimated Pitch Moment (b) Estimated Roll Moment

Figure 31: DJI Guidance

```
  From input "roll_{cmd}" to output "roll":
    3.015
  ---------
  s + 2.313

Continuous-time transfer function.

roll tau=0.432, gain=1.303
Pitch estimated transfer function is:

ans =

  From input "pitch_{cmd}" to output "pitch":
    3.158
  ---------
  s + 2.393

Continuous-time transfer function.

pitch tau=0.418, gain=1.319
```

Figure 32: First Order Transfer Function for M100



(a) Pitch Estimation Model Fit

```
ans =

  From input "pitch_{cmd}" to output "pitch":
          21.58
  ---------------------
  s^2 + 5.419 s + 20.02

Continuous-time transfer function.

pitch omega=4.475, gain=1.078 damping=0.606
```

(b) Estimated Pitch Moment Transfer Function

Figure 33: DJI Guidance

6 Summary, Conclusion, and Discussion

The drone market has exploded in the last decade with the availability of inexpensive drones and sensors. With the development of MEMS technology, many different sensors are available on the market today which can be easily implemented for data collection and for the study of drone dynamics. In the research field, drones have also been of special interest due to their capability of manuevering in difficult locations. Development of technologies such as VO and VSLAM have allowed researches to build autonomous systems that can work independently or with other drones.

The goal of this thesis was to setup a drone platform which can be used to teach about the changes in the dynamics of an aerial vehicle with the change of payload. This is useful because it allows for stable flights with a quick change in payload. It began with a thorough background research in to the development of a quadcopter and in to the fundamentals of the simple and complex sensors available today. Then a drone was selected which was easily programmable and highly modular. Once this was achieved, a solid research and understanding of the software and algorithms that power current autonomous drone systems was performed. Finally, the work demonstrated in [5] was implemented on a DJI M100.

As seen from the results, the estimation of the dynamics of drone can be performed rapidly with a high degree of confidence. For these data sets, the graphs are not too noisy due to the fact that the movement of the quadcopter was limited. As it is a powerful machine, flying in a small area impeded how much data could be obtained.

6.1 Further Work

The next step would be to fly the quadcopter outdoors or in a more spacious location in order to record a large quantity of data. By moving the copter around on all of its axis, a good diverse set of data can be obtained. With analysis on this a better model can be estimated for the copter. Another step would be to add more payload and see how the dynamics change not only due to weight, but due to the location of the payload as well.

In addition, the Guidance sensor can be utilized to extract visual data which can be used for autonomous flight as well as for VSLAM. Simpler filters, such as

the alpha-beta-gamma filter, can also be run on the data to analyze the difference between them and the onboard EKF. The onboard computer can also be switched out for an STM32 board, just as the one used in the Crazyflie (mentioned earlier), to see if it is possible to do similar system identification using this technique. Furthermore, different sensors and payloads should be added/removed to see the accuracy of the system identification script and to quickly assess the dynamics of the copter.

7 References

[1] G. Bugos, "The history of the aerospace industry," August 2001. [Online]. Available: http://eh.net/encyclopedia/the-history-of-the-aerospace-industry/

[2] U. Airforce, "C-130 hercules." [Online]. Available: http://www.af.mil/About-Us/Fact-Sheets/Display/Article/104517/c-130-hercules/

[3] Bitcraze, "Crazyflie." [Online]. Available: https://www.bitcraze.io

[4] A. Glaser, "Dji is running away with the drone market," Apr 2017. [Online]. Available: https://www.recode.net/2017/4/14/14690576/drone-market-share-growth-charts-dji-forecast

[5] I. Sa, M. Kamel, R. Khanna, M. Popovic, and R. Nieto, J. Siegwart, "Dynamic System Identification, and Control for a cost effective open-source VTOL MAV," Jan. 2017.

[6] Mathworks, "Understanding control systems." [Online]. Available: https://se.mathworks.com/videos/series/understanding-control-systems-123420.html

[7] S. Bennett, *A History of Control Engineer 1800-1930*, ser. IEE Control Engineering Series 8. London: Peter Peregrinus Ltd, 1979.

[8] N. Andrei, "Modern control theory - a historical perspective." [Online]. Available: http://www.eecs.tufts.edu/~khan/Courses/Fall2014/EE105/ModernControlHistory.pdf

[9] E. Fernandez Cara and E. Zuazua, "Control theory: History, mathematical achievements and perspectives." [Online]. Available: http://verso.mat.uam.es/web/ezuazua/documentos_public/archivos/personal/comites/ctsema.pdf

[10] Mathworks, "System identification overview." [Online]. Available: https://se.mathworks.com/help/ident/gs/about-system-identification.html?requestedDomain=www.mathworks.com#bsguia1-1

[11] C. Garcia E., D. Prett M., and M. Morari, "Model predictive control: Theory and practice a survey*," *Automatica*, vol. 25, pp. 335–338, May 1989.

[12] J. Desjardins, "Here's how commercial drones grew out of the battlefield," 2016. [Online]. Available: http://www.businessinsider.com/a-history-of-commercial-drones-2016-12?r=US&IR=T&IR=T

[13] B. Zimmer, "The flight of 'drone' from bees to planes," July 2013. [Online]. Available: https://www.wsj.com/articles/SB10001424127887324110404578625803736954968

[14] C. Dillow, "A brief history of drones," October 2014.

[15] J. Turi, "Tesla's toy boat: A drone before its time." [Online]. Available: https://www.engadget.com/2014/01/19/nikola-teslas-remote-control-boat/

[16] "Radio controls robot plane on pilotless flight," October 1935. [Online]. Available: https://goo.gl/qSSvyw

[17] P. Bouffard, "On-board model predictive control of a quadrotor," Technical Report, 12 2012.

[18] L. R. G. Carrillo, L. A. E. Dzul, R. Lozano, and P. Claude, *Quad rotorcraft control: vision-based hovering and navigation.* Springer, 2013.

[19] T. Luukkonen, "Modelling and control of quadcopter," Technical Report, 8 2011.

[20] H. Desa, M. Sofian, and A. Azfar, "Study of inertial measurement unit sensor," 09 2017.

[21] XSENS, "Imu inertial measurement unit." [Online]. Available: https://www.xsens.com/tags/imu/

[22] CUMATIX, "3-axis imu board." [Online]. Available: http://www.cumatix.com/en/3-axis-imu-board/

[23] V. E. N. Solutions, "Gyroscope." [Online]. Available: https://www.vectornav.com/support/library/gyroscope

[24] M. Engineer, "St has mems gyro for more than," August 2017. [Online]. Available: http://pic-microcontroller.com/st-has-mems-gyro-for-more-than/

[25] D. Engineering, "A beginner's guide to accelerometers." [Online]. Available: https://www.dimensionengineering.com/info/accelerometers

[26] I. GlobalSpec, "Accelerometers information." [Online]. Available: http://www.globalspec.com/learnmore/sensors_transducers_detectors/acceleration_vibration_sensing/accelerometers

[27] "How gps drone navigation works." [Online]. Available: http://www.droneomega.com/gps-drone-navigation-works/

[28] A. Hobden, "Quadcopters: Sensors." [Online]. Available: https://hoverbear.org/2015/06/04/quadcopters-sensors/

[29] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous Systems*, vol. 32, no. 1, p. 1–16, 2000.

[30] "What are vision sensors." [Online]. Available: http://www.keyence.com/ss/products/sensor/sensorbasics/vision/info/

[31] K. Daniilidis and J. Shi, "Lecture 39 visual odometry." [Online]. Available: https://www.coursera.org/learn/robotics-perception/lecture/ReEv0/visual-odometry

[32] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual slam: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, p. 289–311, 2015.

[33] A. Singh, "Visual odmetry from scratch - a tutorial for beginners," May 2015. [Online]. Available: https://avisingh599.github.io/vision/visual-odometry-full/

[34] B.-F. Wu, W.-C. Lu, and C.-L. Jen, "Monocular vision-based robot localization and target tracking," vol. 2011, 01 2011.

[35] J. Dove, "Zed stereo camera simulates human visual depth perception," Web Article, 5 2015. [Online]. Available: https://goo.gl/1XHFYu

[36] G. Bradski and A. Kaehler, *Learning OpenCV*, 1st ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media Inc., 9 2008, vol. 1.

[37] S. Riisgaard and M. Rufus Blas, "Slam for dummies."

[38] O. Haines, "An introduction to simultaneous localisation and mapping." [Online]. Available: https://www.kudan.eu/kudan-news/an-introduction-to-slam/

[39] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE TRANSACTIONS ON ROBOTICS*, vol. 32, no. 6, Dec 2016.

[40] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, Jun 2017. [Online]. Available: https://doi.org/10.1186/s41074-017-0027-2

[41] W. Jeong and K. M. Lee, "Cv-slam: a new ceiling vision-based slam technique," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[42] R. Havangi, "Intelligent fastslam: An intelligent factorized solution to simultaneous localization and mapping," *International Journal of Humanoid Robotics*, vol. 14, no. 01, p. 1650026, 2017.

[43] J. Lee, "How sensor fusion works," July 2016. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/how-sensor-fusion-works/

[44] Kionix, "Sensor fusion." [Online]. Available: http://www.kionix.com/sensor-fusion

[45] R. Valenti, I. Dryanovski, and J. Xiao, "Keeping a good attitude: A quaternion-based orientation filter for imus and margs," *Sensors*, vol. 15, no. 8, p. 19302–19330, Jun 2015.

[46] Robottini, "Kalman filter vs complementary filter," September 2011. [Online]. Available: http://robottini.altervista.org/kalman-filter-vs-complementary-filter

[47] P.-J. V. de Maele, "Reading a imu without kalman: The complementary filter," April 2013. [Online]. Available: http://www.pieter-jan.com/node/11

[48] T. Michel, H. Fourati, P. Geneves, and N. Layaida, "A comparative analysis of attitude estimation for pedestrian navigation with smartphones," *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2015.

[49] S. Pelland, G. Galanis, and G. Kallos, "Solar and photovoltaic forecasting through post-processing of the global environmental multiscale numerical weather prediction model," vol. 21, 05 2013.

[50] R. V. D. Merwe, E. Wan, and S. Julier, "Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004.

[51] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," vol. 24, pp. 381–395, 06 1981.

[52] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Bundle Adjustment — A Modern Synthesis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372. [Online]. Available: https://doi.org/10.1007/3-540-44480-7_21

[53] K. Amplianitis, "Bundle adjustment," December 2014. [Online]. Available: https://www2.informatik.hu-berlin.de/cv/vorlesungen/WS1415/material/WS_14_05_Bundle.pdf

[54] H. Chao, Y. Gu, and M. Napolitano, "A survey of optical flow techniques for robotics navigation applications," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 361–372, Jan 2014. [Online]. Available: https://doi.org/10.1007/s10846-013-9923-6

[55] Y. Zhang, T. Wang, Z. Cai, Y. Wang, and Z. You, "The use of optical flow for uav motion estimation in indoor environment," *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, 2016.

[56] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision – ECCV 2006 Lecture Notes in Computer Science*, p. 430–443, 2006.

[57] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2005.188

[58] G. Zhou, L. Fang, K. Tang, H. Zhang, K. Wang, and K. Yang, "Guidance: A visual sensing platform for robotic applications," *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2015.

[59] "Dcdc-nuc, 6-48v automotiove power supply for nuc, 12v or 19v output." [Online]. Available: http://www.mini-box.com/DCDC-NUC