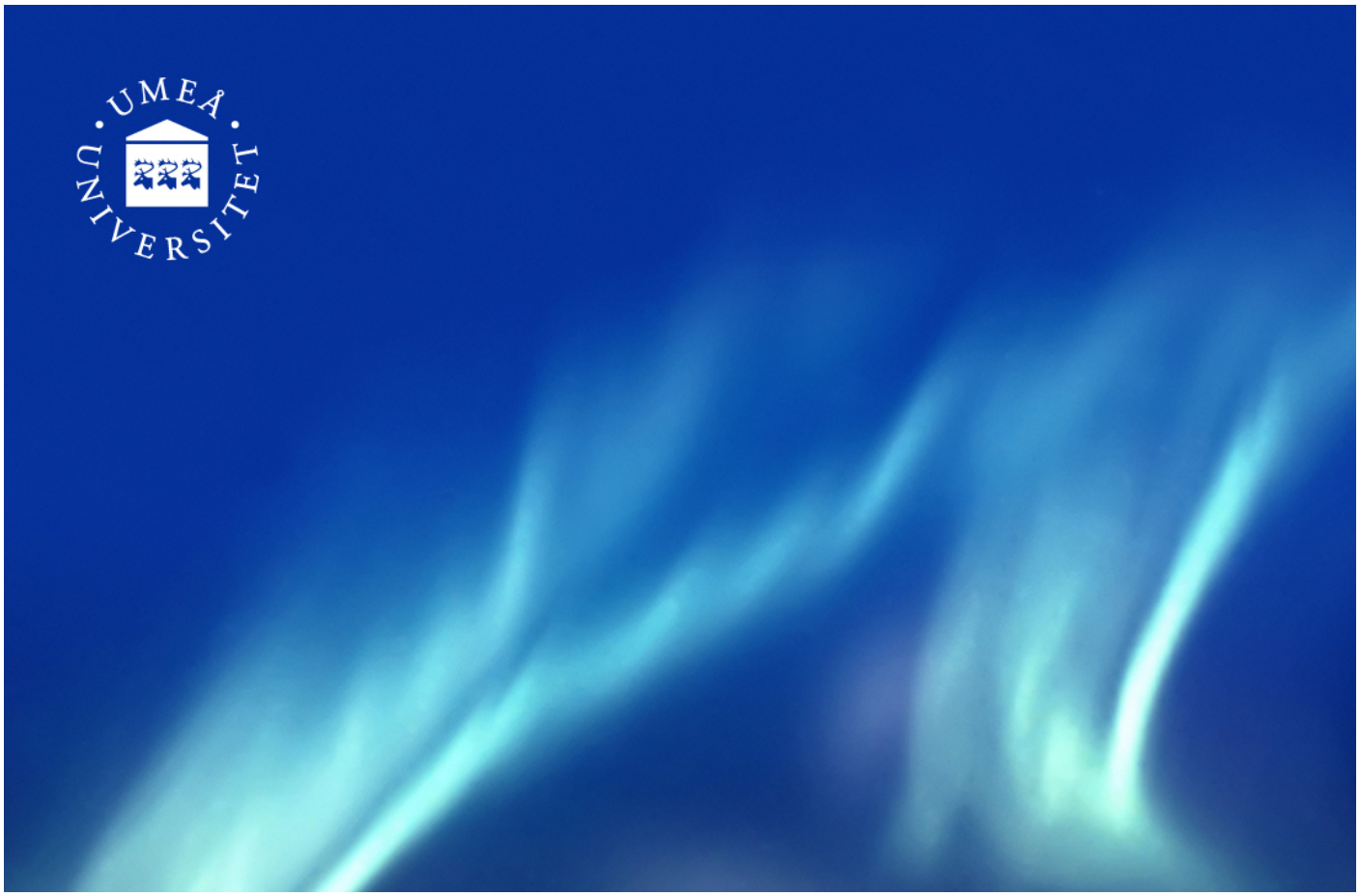


Quadcopter stabilization based on IMU and Monocamera Fusion

Arturo Pérez Rodríguez

Spring 2023
Master Thesis in Electronic Engineering Specialization Robotics and Control, 30 ECTS
Master of Science in Electronic Engineering Specialization Robotics and Control, 120 ECTS

This thesis has been carried out with the collaboration of Knightec AB.

Company:	*Knightec AB, 891 31 Örnsköldsvik*
Company Supervisor:	*Robert Ögren - Robert.Ogren@knightec.se*
Examiner:	*Leonid Freidovich - leonid.freidovich@umu.se*
University Supervisor:	*Sven Rönnbäck - sven.ronnback@umu.se*

Abstract

Unmanned aerial vehicles (UAVs), commonly known as drones, have revolutionized numerous fields ranging from aerial photography to surveillance and logistics. Achieving stable flight is essential for their successful operation, ensuring accurate data acquisition, reliable manoeuvring, and safe operation. This thesis explores the feasibility of employing a frontal monocamera and sensor fusion techniques to enhance drone stability during flight.

The objective of this research is to investigate whether a frontal monocamera, combined with sensor fusion algorithms, can be used to effectively stabilize a drone in various flight scenarios. By leveraging machine vision techniques and integrating data from onboard gyroscopes, the proposed approach aims to provide real-time feedback for controlling the drone.

The methodology for this study involves the Crazyflie 2.1 drone platform equipped with a frontal camera and a Inertial Measurement Unit (IMU). The drone's flight data, including position, orientation, and velocity, is continuously monitored and analyzed using Kalman Filter (KF). This algorithm processes the data from the camera and the IMU to estimate the drone's state accurately. Based on these estimates, corrective commands are generated and sent to the drone's control system to maintain stability.

To evaluate the effectiveness of the proposed system, a series of flight tests are conducted under different environmental conditions and flight manoeuvres. Performance metrics such as drift, level of oscillations, and overall flight stability are analyzed and compared against baseline experiments with conventional stabilization methods. Additional simulated tests are carried out to study the effect of the communication delay.

The expected outcomes of this research will contribute to the advancement of drone stability systems. If successful, the implementation of a frontal camera and sensor fusion can provide a cost-effective and lightweight solution for stabilizing drones.

Keywords: Machine Vision, Sensor Fusion, Unmanned Aerial Vehicles, Drone Stabilization, Frontal Monocamera

Acknowledgements

I would like to express my sincere gratitude to my thesis supervisors, Robert Ögren and Sven Rönnbäck, for their invaluable support, guidance, and expertise throughout this research journey. Their mentorship has been crucial in shaping this thesis.
I am thankful to Umeå University and Knightec for providing the necessary resources and environment that facilitated the successful completion of this research.
I would also like to acknowledge the contributions of my family, friends and colleagues for their assistance, encouragement, and stimulating discussions, which have been a source of motivation and inspiration.
Lastly, I extend my appreciation to all the professors I've met during these two years that have helped me to grow as an engineer, and as a person.
Thank you all for your invaluable support.

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface
CPU Central Processing Unit
DOF Degrees Of Freedom
EKF Extended Kalman Filter
GPS Global Positioning System
HT Hough Transform
IEEE Institute of Electrical and Electronics Engineers
IMU Inertial Measurement Unit
KF Kalman Filter
MAV Micro Air Vehicle
MCU MicroController Unit
MD Mahalanobis Distance
MEMS Micro Electro-Mechanical Systems
MSE Mean Square Error
NAV Nano Air Vehicles
PAV Pico Air Vehicles
PID Proportional-Integral-Derivative
RAM Random Access Memory
SD Smart Dust
SRAM Static Random Access Memory
SURF Speeded Up Robust Features
TCP Transmission Control Protocol
ToF Time of Flight
UAV Unmanned Aerial Vehicle
USB Universal Serial Bus
VM Virtual Machine

1 Introduction

Over the past few decades, we have witnessed tremendous advancements in technology, and one of the most significant developments has been the rise of Unmanned Aerial Vehicle (UAV), commonly known as drones. Drones have gained tremendous popularity over the past few years, revolutionizing various industries and providing unique opportunities for innovation and exploration. These UAVs have a wide range of uses, from entertainment to military applications. However, their impact has been most profound in industries such as agriculture, search and rescue, photography, mining, environmental monitoring, and surveillance [1].

The growing demand for drones is primarily due to their versatility, efficiency, and affordability. Drones are capable of accessing remote locations, performing complex tasks, and capturing high-quality data and imagery. In addition, they can be operated from a distance, which reduces risks to human life and makes them ideal for dangerous or challenging missions. As a result, drones have become indispensable tools in a wide range of industries and have opened up new possibilities for research, exploration, and development.

Due to this large variety of applications, many kinds of UAVs have been developed. UAVs can be classified in various ways, based on size, shape, capabilities, propulsion method and applications. According to the size and weight, drones can be classified from UAV with a maximum wing span of 61 m and weight of 15000 kg, to Smart Dust (SD) with a minimum size of 1 mm and weight of 5 mg. The drones between these two classes are the Micro Unmanned Air Vehicles (μUAV), Micro Air Vehicle (MAV), Nano Air Vehicles (NAV), and Pico Air Vehicles (PAV) [2].

According to the propulsion method, drones can be classified as fixed wings, single rotor, multi-rotor and hybrid. Fixed-wing drones use forward airspeed and static wings to generate the lift. single rotor, or helicopter, drones use a rotor to generate the vertical thrust, while multiple-rotor drones use multiple rotors. Fixed-wing drones are more energy efficient but need more space for take-off and landing than rotor drones. Hybrid models allow them to take off vertically while using wings to fly long distances. There are other propulsion systems, yet much less frequent. For instance, ornithopters flap their wings to mimic insects or birds [3][4].

One of the most popular types of drones is the quadcopter, which has four rotors and is capable of hovering and manoeuvring in tight spaces. Quadcopters have a unique design that allows them to be controlled with precision, making them ideal for capturing aerial footage and performing tasks that require accurate navigation. Moreover, quadcopters are relatively simple to operate and can be programmed to perform automated tasks, making them an excellent option for businesses and individuals looking to streamline their operations.

One of the main challenges in quadcopter design is maintaining stability during flight. Uncontrolled motion can lead to instability, causing the drone to crash or fly erratically. To address this challenge, researchers have developed various stabilization systems for quadcopters. These systems typically use a combination of sensors, including an IMU, which measures the drone's motion, and Global Positioning System (GPS), which provides location data. These sensors are used to adjust the speed and orientation of the drone's rotors, ensuring a stable flight.

However, GPS-based stabilization systems have some limitations. GPS signals can be affected by environmental factors such as interference, signal loss, obstruction, and multipath. Moreover, GPS sensors can be expensive and add weight to the drone, affecting its manoeuvrability and battery life. To overcome these limitations, researchers have explored the use of camera-based stabilization systems that rely solely on visual data.

This paper proposes a stabilization system for quadcopters that uses only the data from a frontal camera and IMU sensor. The system is based on computer vision techniques that allow the drone to sense its surrounding movement and adjust its motion accordingly. Specifically, it is used a combination of feature detection and motion estimation algorithms to track the drone's movement and orientation relative to the environment. This information is then used to adjust the speed and orientation of the drone's rotors, ensuring a stable flight.

This proposed system has several advantages over GPS-based systems. It is less affected by environmental factors, and more cost-effective. Moreover, the system can be used in indoor environments where GPS signals are usually unavailable. This system can contribute to the development of more efficient and reliable quadcopters for a wide range of applications. In addition, the use of a frontal camera, even though is more challenging than a facing-down camera, can reduce costs since many drones already need a front camera for their main purpose.

The rest of this thesis is organized as follows. A brief introduction to drones today is provided in this first part, and the problem is presented. The second section presents all the theories and algorithms that support this study. Part three is an explanation of similar works that have already been developed. Following this, section four shows in detail the hardware used in this work. The methods followed are explained in section five, followed by the development of the work, the results and the discussion. Finally, in section nine, a conclusion about the paper is developed.

1.1 Problem Statement

As devices such as CPUs, GPUs and cameras are becoming increasingly powerful, physically smaller and cheaper, computer vision has been steadily gaining ground in a wide range of applications, from robotic cells in production lines and analysis in medicine to agriculture and vehicle navigation. Parallelly, drones, and other forms of moving robotic systems, are becoming increasingly common in an almost equally wide range of applications. These systems typically use a suite of sensors to manoeuvre in conjunction with one, or several cameras to detect, register, and model the world around them. It could, however, be argued that their most powerful sensor is the camera and that this sensor is underutilised in these systems. By optimising how the system uses its available sensors it should be able to increase efficiency and performance simultaneously, whilst possibly reducing costs.

1.2 Aim and Objectives

If drones rely solely on the IMU for stabilizing, they tend to drift due to minor measurement errors and calculation deviations. In addition, should the disturbances be greater and/or more irregular, such as gusty winds, an engine with a greatly impaired ability or the like, the drone will drift more.

A solution to this is to use data from another sensor that can act as a reference, such as a camera. Using a camera with machine vision can provide data on the movement and velocity of the drone. So the challenge is to design a sensor fusion solution adapted to the provided drone that uses the IMU and a front camera (as observation) to estimate the local pose of the drone and stabilize it and compensate for disturbances.

The goals of the project are:

1. Designing a sensor fusion solution based on IMU and camera to control and stabilize the drone.
2. Testing the robustness against disturbances and comparing the performance using only the IMU sensor.

Such a system could be useful for scenarios where the vehicle requires already a frontal camera for its main purpose. Keeping a drone steady is important to record a video from a static position that is only accessible by a drone. For instance in theatres or different spectacles where a large and expensive structure would waste time and money just to record some video shots. Such a drone can record it without drifting in time and without any specialized sensors for stabilization, making it more cost-effective. By this method local position estimation could be also improved, allowing the drone to follow a track without diverging. This could be used for vigilance or crop supervision.

1.3 Test scenarios

To test the result, some case scenarios are defined. These scenarios will use only the IMU data and the sensor fusion with the camera.

- Take off and fly up straight to a certain altitude, and wait for 30 seconds steady.
- Flying to set position (X = 1m, Y = 1m) and stop.

Other more advanced test scenarios would be:

- Flying to the centre of the room and measuring the distance between the real centre and the drone position.
- Flying following a circular track and measuring how the drone trajectory diverges from the track.

1.4 Sustainable Development

The proposed stabilization system for quadcopters that uses only the data from the camera and IMU sensor can have significant implications for European sustainability objectives [5]. One of the main areas where this system can contribute is in the field of precision agriculture. Drones equipped with this system can provide farmers with real-time information about crop health, soil moisture, and other important variables. This can help farmers optimize their use of resources, reduce the use of pesticides and fertilizers, and increase crop yields, leading to more sustainable and environmentally friendly farming practices.

In addition, this system can also be used in disaster response and monitoring. During natural disasters such as floods and earthquakes, quadcopters equipped with this system can be deployed to assess the extent of the damage and provide information to rescue teams. This can help speed up the response time, minimize risks to human life, and reduce the impact of disasters on the environment.

Finally, this project can also contribute to the European sustainable objectives, in particular, *Goal 12: Responsible consumption and production*, by promoting the development of more energy-efficient and eco-friendly drones. By using only the data from the camera and IMU sensor, our system reduces the need for additional sensors and equipment, thereby reducing the weight of the drone and increasing its battery life. This can help reduce energy consumption and emissions and components required, leading to a more sustainable and environmentally friendly approach to drone technology.

Overall, this project has the potential to make significant contributions to the European sustainable objectives in various fields, including agriculture, disaster response, and the development of more eco-friendly drone technology.

1.5 Project limitations

All the computation and control will be made on a computer connected to the drone, this could lead to a worse performance due to the delay in communication but allow more powerful computation. The full implementation onboard of the drone will be for future work.

Other limitations could be the light conditions and background texture since machine vision techniques are sensitive to environments without much light and poor textured surfaces. The implementation will work under the assumption that the drone is the only object moving in the room, and that it is a small movement.

2 Related Work

UAVs and machine vision are two topics that have been really present in many pieces of research in the last years. Both of them are fields with a lot of potential, and the reduction in costs of electronic devices gives easy access to the hardware required to work on these topics. In addition, the reduction in size and energy consumption of cameras has brought the possibility to add them to micro UAVs.

The first project to be discussed is the work of Mori and Scherer [6], in which a system for detecting and avoiding frontal obstacles using a Parrot AR.Drone is developed. The AR.Drone, equipped with an onboard camera, is well-suited for visual-based obstacle detection.

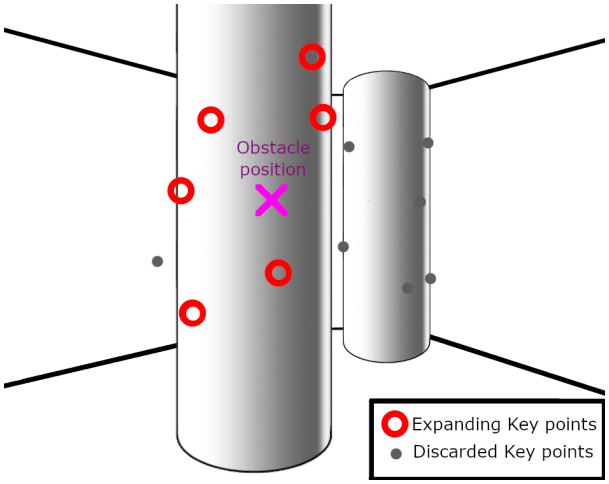


Figure 1: *Representation of detected key points on obstacles using the Speeded Up Robust Features (SURF) algorithm*

Obstacles are detected by using the SURF algorithm to identify key points in the camera images. These key points are then matched with the corresponding points in the next frame, and the size growth of these points is measured (Fig. 1). A reactive obstacle avoidance method enables the system to avoid obstacles without relying on deliberate path-planning techniques that could slow down the computation speed of the drone.

The quadcopter can avoid frontal obstacles by detecting the time to the collision of approaching obstacles. If the time to collision is considered too small, the quadcopter will fly sideways to avoid the obstacle. In addition to obstacle detection, the drone's current position and goal position are also known, and the yaw angle is controlled to achieve the goal.

This system offers several advantages over other obstacle-avoidance methods. Firstly, it is lightweight and does not require additional sensors since it only uses the onboard camera. This can reduce cost and complexity. Secondly, the reactive approach enables real-time obstacle avoidance without the need for path planning. Finally, the system can be adapted to different types of quadcopters, making it a versatile solution for obstacle avoidance in different scenarios.

Overall, the work of Mori and Scherer is an important contribution to the field of drone obstacle avoidance. The use of a reactive method and only relying on the onboard camera provides a lightweight and efficient solution that has the potential to improve safety and efficiency in various drone applications.

The SURF algorithm is also used by Mingkhwan and Khawsuk [7] to stabilize images captured from a camera on a multi-rotor. In this project, the drone is not controlled based on the camera. Instead, the recorded video is edited to remove the vibration caused by the drone's movement. To achieve motion compensation, 20 key points are extracted using the SURF algorithm, and the image frame is compensated in the opposite direction to offset camera motion by rotating and scaling it.

Crazyflie is a drone that has been utilized in many projects due to its versatility, lightweight, and cost-effectiveness. Lindgren's work [8] is an example of the drone's potential, where it is used in combination with the AI-deck and the multi-ranger deck to achieve autonomous target recognition and tracking.

The AI-deck enables the recording of video and the implementation of an onboard MobileNet-SSD object detector through the use of the GAP8 processor. Meanwhile, the multi-ranger deck uses five Time of Flight (ToF) sensors to provide millimetre-precision distance measurements to the drone's bottom, front, back, left, and right.

With these components, the drone is able to achieve stability in the air and detect humans. It then moves in the horizontal plane to keep the detected person in the desired position in the camera's field of view and at the desired distance. Stabilization is achieved through the use of the multi-range deck and the firmware in the Crazyflie, which implements a Kalman Filter.

Despite its successes, this project faced limitations. One of these was the performance of the detection model, which was not as accurate as expected. Additionally, the low computation speed of the images affected the control of the drone, resulting in oscillations in its movement. Nonetheless, this project showcases the potential of using Crazyflie and its broad deck variety to achieve autonomous target recognition and tracking, paving the way for further advancements in the field of drone technology.

Another project developed with Crazyflie is the one developed by Greiff, M. [9] where they research the dynamics and motion planning of this UAV. In addition, different positioning techniques are explored. There is only one of these methods that use no external components like cameras and anchors. This method uses a laser ranging sensor to measure the flight height and an optical flow sensor on the bottom of the drone to perform a position estimation of the centre of concentric circles printed on the floor. An Extended Kalman Filter (EKF) is performed to fuse the IMU and optical flow measurement, and the result obtained is good for the limited time of flight of the Crazyflie.

A similar project is [10] where a Parrot AR.Drone 2.0 also with a downward-looking camera implements optical flow and EKF to estimate distance and velocity. This estimation is done by finding the corners in an image and tracking them to the next frame. By measuring the distance between every two corners at one image and at the next frame, the expansion and contraction of the flow are obtained. The divergence estimated from this flow is used to control the landing.

3 Hardware platform

In this project, the Crazyflie quadcopter is used together with the AI-deck for video recording. Most of the computation and control is performed in a PC that communicates with the drone using the Crazyradio PA.

3.1 Crazyflie 2.1

The Crazyflie is an open-source quadcopter developed by Bitcraze AB that weighs approximately 27 g and has a motor-to-motor size of $92 \times 92 \times 29$ mm including motor mount feet. It is designed to be used indoors, with open-source firmware considered manageable and not too complex [11], and it allows the addition of more hardware for different purposes such as expansion decks. All of these make this device a great development platform for research. The flight time with full battery is around seven minutes [12].

The control of the device can be made through Bluetooth using the phone app provided by Bitcraze. Another option is using the Bitcraze Crazyradio PA, an open Universal Serial Bus (USB) radio dongle for a long-range radio connection with a 20 dBm power amplifier that provides a communication range of up to 1 km [13]. The Crazyradio PA allows connecting to the drone using the PC client installed on the Bitcraze Virtual Machine (VM). The Python Application Programming Interface (API) developed by Bitcraze also allows one to connect to the drone and control it easily using Python. There are four main dimensions of control when flying the Crazyflie: roll, pitch, yaw, and thrust [14].

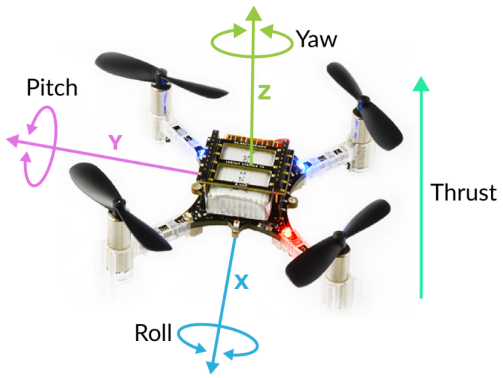


Figure 2: *Crazyflie 2.X frame and control variables (Based on [14] distributed under a CC BY 3.0 License)*

3.2 IMU and High Precision Pressure Sensor

Crazyflie 2.1 has a BMI088 IMU [15] onboard for detecting movements and rotations in 6 Degrees Of Freedom (DOF). This device is the combination of two sensors in one device: a triaxial 16-bit gyroscope and a triaxial 16-bit accelerometer. It is designed for harsh vibration environments, suppressing vibrations above a few hundred Hertzs making it a great option for drones. The output noise of the gyro is 0.014 $°/s/Hz$, and the output noise density of the accelerometer is 190 $\mu g/\sqrt{Hz}$ for Z-axis and 160 $\mu g/\sqrt{Hz}$ for X- and Y-axis. In Figure 3 the onboard gyro, ω, and the angle, θ, are plotted for two minutes. The. The angle is calculated as $\theta_{k+1} = \theta_k + \omega T_s$ with a sampling time, T_s, of 10 ms. It is repeated several times to calculate different angles. Only the first gyro, ω, is plotted since there is no observable differences among the gyro readings obtained. A drift in the angle over time is observed in all the cases.

To measure the pressure, Crazyflie uses the BMP388 digital sensor [16] onboard. This sensor consists of

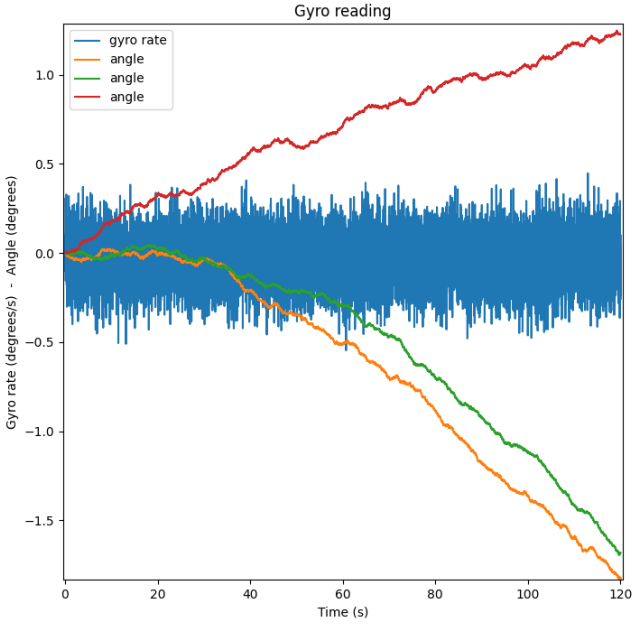


Figure 3: *Angle drift measured for two minutes*

a piezo-resistive pressure sensing element and it is suitable for altitude tracking in drone applications. The noise in pressure of the sensor without filtering is 1.2 Pa, and with the lowest bandwidth the noise becomes 0.03 Pa.

3.3 Onboard microcontrollers

For the main application, Crazyflie 2.1 uses an STM32F405 MicroController Unit (MCU) [17]. This unit has an ARM 32-bit Cortex-M4 Central Processing Unit (CPU), 192+4 kB of Static Random Access Memory (SRAM) and up to 1 MB of Flash memory. The radio communication and power management is controlled by an nRF51822 MCU [18], which uses the 32-bit ARM Cortex M0 MCU with 128kB available for application development and 16 kB of Random Access Memory (RAM). It allows multi-protocol 2.4 GHz radio communication.

In addition, the Crazyflie 2.1 counts with a micro-USB connector that enables charging the drone thanks to the onboard LiPo charger with 100 mA, 500 mA and 980 mA modes available. Finally, to save some data and settings between flights, the board has an 8 kB EEPROM [12].

3.4 Control system

Crazyflie turns the sensor signals into a state estimation. The base drone, without any additional deck, uses a complementary filter to estimate the attitude (roll, pitch, and yaw) from the gyroscope and accelerometer signals. This filter is efficient and provides great performance for simple cases like this. Using decks that provide velocity and/or position information (e.g. flowdeck, LPSdeck, Lighthouse deck), the Crazyflie perform an Extended Kalman Filter to estimate not only the attitude but also the position $[x, y, z]$ and velocity $[\dot{x}, \dot{y}, \dot{z}]$ [19].

Once the current state is estimated, the quadcopter uses Proportional-Integral-Derivative (PID) control to reach the desired state. This control consists of different PID [20] controllers in cascade, controlling the position, velocity, attitude and attitude rate (Fig. 4). For a High-Level Commander where the

set point is the desired position, all the PID controllers will be used. The difference between desired position signal and the current measured position is the input of the position PID. At the same time, the difference between the output of this PID and the measured velocity is the input of the velocity PID. And in the same way, the PID controllers for the attitude and attitude rate are concatenated. The attitude rate controller uses the gyroscope rates slightly filtered as feedback. The controller's final result is the desired thrusts that will be handled by the power distribution and sent to the motors [21].

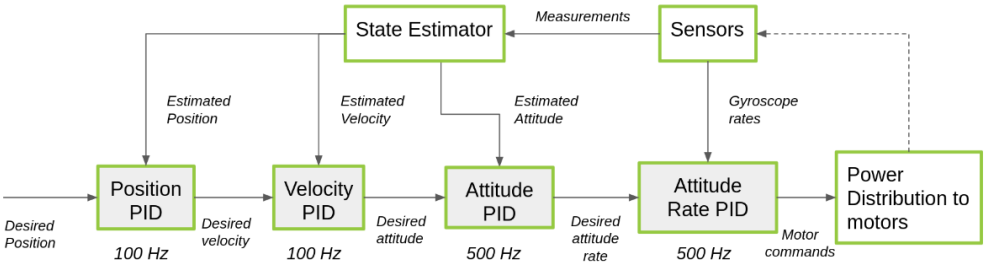


Figure 4: *Crazyflie 2.X control loops as cascaded PID ([21] distributed under a CC BY 3.0 License)*

It is also possible to control only the attitude using the two most inner-loop controllers. This would be the case where there is no additional deck that provides information for the position and velocity estimation.

3.5 Crazyflie AI-deck 1.1

AI-deck is a lightweight and low-power expansion deck developed by Bitcraze AB for the Crazyflie drone. This board extends the computational capabilities and enables complex artificial intelligence-based workloads to run onboard. The core used is GAP8 which allows this ultra-low power consumption. In addition, it uses a Himax HM01B0 monochrome camera with a 320x320 pixels resolution. The table 1 [22] shows the parameters of the camera used in the AI-deck. The ESP32 onboard adds WiFi connectivity, giving the possibility to stream images and handling control [23].

Table 1: *Himax HM01B0 monochrome camera parameters*

Pixel Array	320×320
Pixel size	$3.6\ \mu m$
Effective Focus Length	$0.66\ mm$
Horizontal Field of view	$87\ ^\circ$
Vertical Field of view	$87\ ^\circ$
Diagonal Field of view	$115\ ^\circ$

3.6 Python API

Bitcraze provides a library made in Python that gives high-level functions to communicate and control the Crazyflie. The library is asynchronous and based on callbacks for events. Nevertheless, certain classes have synchronous wrappers available that create a synchronous API by wrapping the asynchronous classes. This API has interfaces for radio, serial, USB, and Transmission Control Protocol (TCP) network connection. All communication links are identified using a Uniform Resource Identifier [24].

The library facilitates the establishment of logging configurations that enable the logging of firmware variables. A logging configuration comprises specific variables that need to be logged and an interval duration (in milliseconds) for transmitting the data to the host. Upon adding the log configuration to the firmware, the data is automatically transmitted at the configured time interval. The maximum length for a log packet is 26 bytes, and log configurations must have a time interval that is a multiple of 10 ms as the minimum period.

The library allows for run-time reading and writing of parameters to the firmware, which should only be done for data that is not continuously changing. Parameter values are updated only once after connecting and the updated values are sent back to registered callbacks. Parameters can be written from the host or during start-up, and there is experimental support for changing parameters within the firmware's app layer, but this may not update the host correctly.

The control setpoints are changed using the special API. The function $sent\_setpoint(\phi, \theta, \dot{\psi}, T)$ provided in the API allows setting the roll, pitch, yaw rate, and thrust to control the attitude. The thrust value corresponds to the mean force applied to the motors, with battery compensation applied to ensure independence from the battery voltage. Roll and pitch are measured in degrees, while yaw rate is measured in degrees per second. The quadcopter includes a safety function that will change the setpoint of roll, pitch and yaw rate to zero to stop accelerating if no new setpoint is applied after 500 ms. In addition, the power to the motors will be cut if no new setpoint is applied after two seconds [24].

There are other functions that allow setting the velocity or position setpoints, but this required additional positioning systems like Loco, Flowdeck or Lighthouse [24].

4 Theoretical Framework

This section serves as the conceptual foundation for the subsequent analysis and interpretation of this study. It presents a comprehensive overview of the key theoretical frameworks and perspectives that inform the research. Drawing from established theories and scholarly literature, this section establishes a framework that enables a systematic examination of the research problem and contributes to a deeper understanding of the subject matter.

4.1 Sensors

Sensors are devices that detect a physical phenomenon and convert these changes into an electrical signal. They play a crucial role in providing information about the environment to electronic systems. There is a huge variety of sensors, allowing one to get signals from all the different aspects of a system. Following this, a description of the most relevant sensors for this work will be provided.

4.1.1 IMU

IMU is a Micro Electro-Mechanical Systems (MEMS) that can be found in various applications such as Navigation Systems, Medical Rehabilitation, Sports Learning and Quality Control, among others. This sensor consists of an accelerometer and gyroscope. Typically each sensor has between two to three DOF for the X-, Y- and Z-axes, giving a total of four to six DOF with both sensors. Accelerometers measure acceleration, while gyros measure angular velocity.

Some IMUs also have magnetometers. These IMUs commonly use tri-axial sensors making a total of 9 DOF. The magnetometer is used to improve the measurement of the gyro, improving the drift issue that this type of sensor has. But magnetometer measurements might be disturbed if it is used in an environment surrounded by ferromagnetic metal, due to disturbances in the magnetic field [25].

A gyro has a tendency to drift over a long period of time, whereas an accelerometer is sensitive to any acceleration, for instance, when an object has a fast rotation or translational acceleration. Sensor fusion techniques like Complementary filters or Kalman Filters are commonly used to improve accelerometer and gyroscope data [25].

MEMS accelerometers can work either by measuring the displacement of the mass or by measuring the frequency of a vibrating element due to a changing mass. The displacement or vibration produces a change in the capacitance of the sensor. MEMS gyroscopes use the Coriolis effect that declares that in

a frame of reference rotating at angular velocity ω, a mass m moving with velocity v experiences a force F [26],

$$F = -2m(\omega \times v). \tag{1}$$

To measure this effect, rate gyros contain vibrating masses which vibrate along an axis. An additional vibration is induced along the perpendicular of this axis, displacing the mass from its origin when the gyro is rotated. This displacement is detected by capacitance changes [26].

4.1.2 Pressure sensor

Pressure sensors convert pressure into electrical signals using different sensing principles. These sensors are used in a large variety of fields such as hearing aids, touch screens, health monitoring, weight and height measurement, and weather forecasting. The sensing principle used by the sensor will depend on the pressure range, sensitivity, linearity, and response time of the task [27]. The main principles used are capacitive, piezoresistive, optical, resonant, and piezoelectric [28].

Piezo-resistive pressure sensors use a cavity in a material, usually silicon, to form a diaphragm. On the top of the diaphragm, four resistors are connected in a Wheatstone bridge circuit (Fig. 5). When pressure is applied to the diaphragm, the diaphragm is in a stress condition. These stresses are experienced by the resistors due to piezo-resistive effects [29].

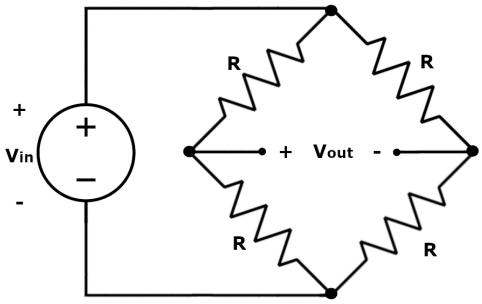


Figure 5: *Wheatstone bridge circuit*

4.2 Rigid Body Dynamics

The quadrotor model is formed by a frame with two arms crossing at a 90º angle. The centre of mass is located at the crossing point where the quadrotor's body lies. These arms have a propeller at each end, and the direction of rotation of the propellers in the same arm is paired and opposite to the direction of the propellers in the other arm. By rotating the propellers at the same speed and in opposite directions, the rotational torque can be cancelled. Since there are only four independently controllable actuators, the system is underactuated where there are only four DOF to be controlled directly. The control variables usually selected are vertical acceleration $\ddot{Z}$, roll acceleration $\ddot{\phi}$, pitch acceleration $\ddot{\theta}$ and yaw acceleration $\ddot{\psi}$. Roll, pitch and yaw motions can be obtained by small differences in the propellers' rotation speed while vertical motion depends on the sum of all propellers' speeds. By restricting the differences in propellers' rotation speed to small values, non-linear dynamics and motor saturation can be avoided [30].

Two frames of reference are defined, the global frame G that is static with respect to the world and represents the global pose, and a body frame B that moves and rotates with the drone. Both frames are in the same pose at the initial moment, with Z axis pointing up, X pointing to the front of the drone and Y pointing out to the left (Fig. 2).

The rotation matrix which defines the rotation between the global and the body frame is (2) [9],

$$ {}^{G}_{B}\boldsymbol{R} = {}^{B}_{G}\boldsymbol{R}^{T} = \begin{bmatrix} c_\phi c_\psi & s_\psi c_\phi & -s_\theta \\ c_\psi s_\theta s_\phi - s_\psi c_\phi & s_\psi s_\theta s_\phi + c_\psi c_\phi & c_\theta s_\phi \\ c_\psi s_\theta c_\phi + s_\psi s_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi & c_\theta c_\phi \end{bmatrix}, \tag{2} $$

Therefore,

$$ {}^{G}\begin{bmatrix} \ddot{p_x} & \ddot{p_y} & \ddot{p_z} \end{bmatrix}^T = {}^{B}_{G}\boldsymbol{R} \begin{bmatrix} \dot{u} & \dot{v} & \dot{w} \end{bmatrix}^T. \tag{3} $$

Where ${}^{G}[p_x, p_y, p_z]^T$ is the position in the global frame and ${}^{B}[u, v, w]^T$ is the velocity in the body frame.

From Newton's laws and the equation of Coriolis, (4) is obtained [31],

$$ m\,{}^{G}\frac{d\boldsymbol{v}}{dt} = m\left({}^{B}\frac{d\boldsymbol{v}}{dt} + {}^{B}_{G}\boldsymbol{\omega} \times \boldsymbol{v}\right) = \boldsymbol{f}. \tag{4} $$

Where ${}^{B}\frac{d}{dt}$ is the time derivative in the body frame, ${}^{G}\frac{d}{dt}$ is the time derivative in the global frame, ${}^{B}_{G}\boldsymbol{\omega}$ is the angular velocity of the body frame with respect to the global frame, m refers to the drone's mass and $\boldsymbol{f}$ is the total force applied to the quadrotor. In the system coordinates, where ${}^{B}\boldsymbol{v} = (u, v, w)^T$, ${}^{B}\boldsymbol{f} = [f_x, f_y, f_z]^T$ and ${}^{B}_{G}\omega = [p, q, r]^T$, (4) becomes

$$ \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}. \tag{5} $$

Using Newton's law for rotational motion and the equation of Coriolis it is obtained (6),

$$ {}^{G}\frac{d\boldsymbol{h}}{dt} = {}^{B}\frac{d\boldsymbol{h}}{dt} + {}^{B}_{G}\boldsymbol{\omega} \times \boldsymbol{h} = \boldsymbol{\tau}. \tag{6} $$

Where $\boldsymbol{h}$ is the angular momentum and $\boldsymbol{\tau}$ is the applied torque. (6) is easily resolved in body coordinates where ${}^{B}\boldsymbol{h} = \boldsymbol{I}^{B}_{G}\boldsymbol{\omega}$ where $\boldsymbol{I}$ is the constant inertia matrix defined as

$$ \boldsymbol{I} = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{bmatrix}. \tag{7} $$

Since the quadrotor is symmetric about all three axes, $I_{xy} = I_{xz} = I_{yz} = 0$. Defining ${}^{B}\boldsymbol{\tau} = [\tau_\phi, \tau_\theta, \tau_\psi]^T$, (6) can be rewritten in body coordinates as,

$$ \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{I_y - I_z}{I_x} qr \\ \frac{I_z - I_x}{I_y} pr \\ \frac{I_x - I_y}{I_z} pq \end{bmatrix} + \begin{bmatrix} \frac{1}{I_x}\tau_\phi \\ \frac{1}{I_y}\tau_\theta \\ \frac{1}{I_z}\tau_\psi \end{bmatrix}. \tag{8} $$

Assuming that ϕ and θ, and the Coriolis terms qr, pr and pq are small, (8) can be simplified to (9) [30],

$$ \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_x}\tau_\phi \\ \frac{1}{I_y}\tau_\theta \\ \frac{1}{I_z}\tau_\psi \end{bmatrix}. \tag{9} $$

Where F is the thrust or amount of upward force exerted between all the propellers.

Neglecting Coriolis terms and merging (3) (5) and simplifying gives the position model (10),

$$\begin{bmatrix} \ddot{p_x} \\ \ddot{p_y} \\ \ddot{p_z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \begin{bmatrix} \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) \\ \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ \cos(\phi)\cos(\theta) \end{bmatrix} \frac{F}{m}. \tag{10}$$

4.3 Control of Quadcopter

A common method to control quadcopters is PID control loops. PID controllers are used for their simple structure and easy implementation. These controllers are defined by the following equation,

$$u(t) = K_P[x_d(t) - x(t)] + K_I \int_0^t [x_d(\tau) - x(\tau)]d\tau + K_D \frac{d[x_d(t) - x(t)]}{dt} = C_{pid}\{x_d - x\}. \tag{11}$$

Where x_d denotes the setpoint or desired signal value, x refers to the actual signal value, and K_P, K_I, and K_D are the proportional, integral, and derivative parameters of the controller, respectively. This quadcopter is a system with 6 DOF, but only four variables are controlled. The total thrust F and the torques $\boldsymbol{\tau}$ created by the rotors are derived from the dynamic equations (9) and (10). While the total thrust F affects the acceleration in the direction of the z-axis, keeping the drone in the air, the torques τ_ϕ, τ_θ, τ_ψ contribute to the change in roll ϕ, pitch θ and yaw ψ respectively. The controllers for the drone are (12) [32],

$$\begin{aligned} F &= [g + C_F(z_d - z)]\frac{m}{\cos(\phi)\cos(\theta)}, \\ \tau_\phi &= I_x[C_\phi(\phi_d - \phi)], \\ \tau_\theta &= I_y[C_\theta(\theta_d - \theta)], \\ \tau_\psi &= I_z[C_\psi(\psi_d - \psi)]. \end{aligned} \tag{12}$$

Where the torque controllers are cascade PID controllers for the attitude and attitude rate implemented in the Crazyflie quadcopter firmware (Fig. 4). In this equations, ϕ, θ and ψ denote the estimated attitude signal in the figure while ϕ_d, θ_d and ψ_d denote the desired attitude signal.

4.4 Sensor Fusion

Sensor fusion is a method that allows one to extract information from several data sources and integrate them into a single signal. These algorithms are handy for guidance, navigation and control of vehicle applications, where there are multiple sensors getting similar information. This information is often integrated into a meaningful signal that can be used for control systems [33]. Sensor fusion provides more robust and reliable measurements, increases confidence and reduces ambiguity and uncertainty on the measured values [34].

4.4.1 Complementary Filter

Complementary filter is a computationally inexpensive sensor fusion approach that uses a low-pass filter, usually denoted as $G(s)$, and a high-pass filter, usually denoted as $[1 - G(s)]$, [35]. This technique can be used to estimate the attitude based on gyroscope and accelerometer readings. In this case, the accelerometer reading is introduced to compensate for the error of the angular rate derived from the random walk of the gyroscope. Using the quaternion derived from the gyro $\boldsymbol{q}_{\omega,t}$ and accelerometer $\boldsymbol{q}_{a,t}$ readings for a time t [36], the estimated attitude $\boldsymbol{q}_{est,t}$ applying the complementary filter will be defined by

$$\boldsymbol{q}_{est,t} = [1 - G(s)]\boldsymbol{q}_{\omega,t} + G(s)\boldsymbol{q}_{a,t}. \tag{13}$$

4.4.2 Kalman Filter

KF is a stochastic Sensor Fusion method that uses a mathematical model for filtering signals with uncertainty or sensors with noise and statistical models from an initial assumption to reduce the uncertainty. The filter uses a discrete-time algorithm and models noise from sensor signals to produce fused data. Estimated smoothed values of position, velocity, and acceleration can be achieved using this method [34].

The Kalman Filter consists of three phases: *Prediction Phase*, *Observation Phase* and *Fusion Phase*. In the *Prediction Phase* an estimator $\hat{\boldsymbol{x}}_{k+1|k}$ of the system state $\boldsymbol{x}$ for time $k+1$ is computed using the previous estimation $\hat{\boldsymbol{x}}_{k|k}$ (14), where $\boldsymbol{A}$ is a non-singular state transition matrix, $\boldsymbol{u}_k$ is the input vector, and $\boldsymbol{B}$ define the relation between the system state and the input. Then, the error covariance matrix $\boldsymbol{P}$ is predicted for discrete time $k+1$ (15) where $\boldsymbol{Q}$ is the covariance matrix [34] representing the process noise.

$$\hat{\boldsymbol{x}}_{k+1|k} = \boldsymbol{A}\hat{\boldsymbol{x}}_{k|k} + \boldsymbol{B}\boldsymbol{u}_k, \tag{14}$$

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{A}\boldsymbol{P}_{k|k}\boldsymbol{A}^T + \boldsymbol{Q}. \tag{15}$$

In the *Observation Phase*, a measurement from the environment $\boldsymbol{z_k}$ (16) is obtained, and it introduces the measurement noise $\boldsymbol{v}$ that is assumed to be white noise with the normal probability distribution of $p(v)(0, \boldsymbol{R})$, where $\boldsymbol{R}$ refers to the measurement covariance [37].

$$\boldsymbol{z}_k = H\boldsymbol{x}_k + \boldsymbol{v}. \tag{16}$$

Finally, in the *Fusion Phase*, the Kalman gain matrix $\boldsymbol{K}$ is computed (17) where $\boldsymbol{H}$ relates the measurements to the internal state. The system state prediction $\hat{\boldsymbol{x}}_{k+1|k+1}$ (18) is a linear combination of the state estimation $\hat{\boldsymbol{x}}_{k+1|k}$ and the *innovation*. The *innovation*, or *residual*, reflects the discrepancy between actual measurement and the predicted measurement $(\boldsymbol{z}_k - H\hat{\boldsymbol{x}}_{k+1|k})$ [37],

$$\boldsymbol{K}_{k+1} = \boldsymbol{P}_{k+1|k}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{k+1|k}\boldsymbol{H}^T + \boldsymbol{R})^{-1}, \tag{17}$$

$$\hat{\boldsymbol{x}}_{k+1|k+1} = \hat{\boldsymbol{x}}_{k+1|k} + \boldsymbol{K}_{k+1}(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k+1|k}). \tag{18}$$

When $\boldsymbol{R}$ approaches zero (17), the gain $\boldsymbol{K}$ weights the innovation more heavily, so the actual measurement $\boldsymbol{z}_k$ is more trusted and the predicted measurement $(\boldsymbol{H}\hat{\boldsymbol{x}}_{k+1|k})$ is less trusted. On the other hand, when the estimated error covariance $\boldsymbol{P}_{k+1|k}$ approaches zero, the gain $\boldsymbol{K}$ weights the residual less heavily, so the actual measurement is less trusted, while the predicted measurement is more trusted [37].

In addition, an outlier-rejection scheme can be applied in the Kalman filtering. This approach is grounded on the assumption that observations can be considered normal based on the measurement model, while abnormal outliers are generated by a different model. The outlier-rejection scheme aims to accept normal measurements and reject abnormal ones. A simple, but still good-performing, method for outlier-rejection is the Mahalanobis Distance (MD), also known as statistical distance. The MD (20) takes the variability of the variables into account, using the covariance of the innovation $\boldsymbol{S}$.

$$\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^T + \boldsymbol{R}, \tag{19}$$

$$d_M(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T S^{-1}(\boldsymbol{x} - \boldsymbol{y})}. \tag{20}$$

The outlier-rejection is implemented by calculating the MD between the estimation and the observation. In order to validate the observation, the χ^2 statistical test is applied to the distance. The fusion phase is only applied if the observation is accepted, otherwise, the observation is rejected and the system state $\boldsymbol{X}$ is not updated [38].

4.4.3 Extended Kalman Filter

Kalman filtering is used to estimate the internal state of a discrete-time controlled process that is governed by a linear stochastic difference equation. But the process to be estimated or the measurement related to the process could be non-linear. In these non-linear cases, the EKF linearizes about the current estimation and covariance to compute the estimation [39]. While the standard Kalman Filter is optimal for linear systems and always converges, the solution provided by the EKF is only approximate and may diverge if the consecutive linearizations are not a good approximation of the linear model in all the associated uncertainty domain [34] [40]. Consider the non-linear dynamics,

$$x_{k+1} = f_k(x_k) + w_k, \tag{21}$$

$$z_k = h_k(x_k) + v_k, \tag{22}$$

where w_k and v_k are white Gaussian, independent random noises with zero means and $f(*)$ and $h(*)$ are non-linear functions. To apply the EKF, linearizations around the estimated points are calculated [40],

$$F_k = \nabla f_k \mid_{\hat{x}_{k|k}}, \tag{23}$$

$$H_{k+1} = \nabla h_k \mid_{\hat{x}_{k+1|k}}, \tag{24}$$

where F_k and H_k denote the Jacobian matrices of $f(*)$ and $h(*)$. Same as in the Kalman filtering, there are three phases. In the *Prediction Phase* the internal state (25) and the covariance (26) are estimated,

$$\hat{x}_{k+1|k} = f_k(\hat{x}_{k|k}), \tag{25}$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k. \tag{26}$$

In the *Observation Phase*, the measure from the environment z_k (22) is obtained. Finally, in *Fusion Phase* the Kalman gain is computed (27) together with the predicted state (28) and the predicted covariance (29),

$$K_{k+1} = P_{k+1|k} H_{k+1}^T [H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1}]^{-1}, \tag{27}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}[z_{k+1} - h_{k+1}(\hat{x}_{k+1|k})], \tag{28}$$

$$P_{k+1|k+1} = [I - K_{k+1} H_{k+1}] P_{k+1|k}. \tag{29}$$

4.4.4 Sliding-mode-based observer

The sliding-mode-based observer, or sliding observer, is a deterministic observer that, contrary to the Kalman Filter, doesn't need assumptions about the stochastic nature of noise or estimating the error. This method avoids the heavy computations required for the Kalman filter [41] while also having measurement noise resilience.

This method ensures the convergence of a sliding surface defined to the origin in finite time [42]. J.P. Barbot et al.[43] present examples of Sliding mode observer designs for different systems for more information.

4.5 Hough Transform

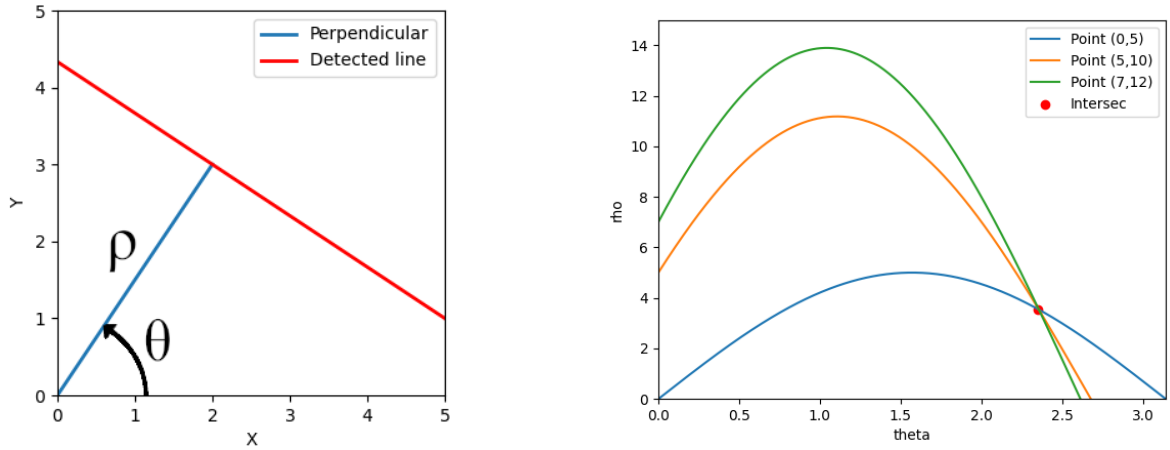
The Hough Transform (HT) algorithm parameterizes complex patterns of points in binary images. The HT converts a difficult global detection problem in image space into a more easily solved local peak detection problem in a parameter space [44]. This algorithm can be used to detect patterns such as straight lines, circles, or ellipses. Straight lines can be expressed with two parameters like the slope m and the y-intercept of the line c in the Cartesian coordinate system (30), or length ρ and orientation θ of the normal vector to the line from the image origin (Fig. 6a) in polar coordinate system (31),

$$f(x, y) = y - mx - c = 0, \tag{30}$$

$$f(x, y) = \rho(\theta) = x\cos(\theta) + y\sin(\theta), \tag{31}$$

where (x, y) defines a point in the image and $f(x, y)$ defines the family of lines that goes through that point. The HT in Cartesian coordinate system parametrization (m, c) presents a singularity with large slopes where $m \to \infty$. This issue is solved using polar parametrization (ρ, θ) [44].

For a given point (x_0, y_0), the family of lines that goes through it define a sinusoid in a plane $\theta - \rho$ with $(\rho, \theta) \in [0, \infty) \times [0, 2\pi)$. This sinusoid is calculated for all the points in the image, and if the curves of two different points intersect in the plane $\theta - \rho$, both points belong to the same line (Fig. 6b). A line can be detected by finding the number of intersections between curves. The intersection point (ρ, θ) defines the line parameters. All the intersection points with a number of intersections higher than a threshold are considered straight lines in the image [45].



(a) *HT parameters in polar coordinate system*

(b) *Intersection in the $\theta - \rho$ space of three family-line curves of Cartesian points in a straight line*

Figure 6: *Hough Transform*

4.6 Optical Flow

Optical flow is a fundamental concept in the field of computer vision and image processing, which refers to the pattern of motion in a sequence of images [46]. Specifically, it involves calculating the movement of pixels or features within an image or sequence of images. This information can be used for a wide range of applications, including object tracking and motion analysis [47].

The measurement of optical flow is a challenging problem, as it requires the computation of an approximation of the 2D motion field from spatiotemporal patterns of image intensity. Many techniques have been proposed over the years to estimate optical flow, ranging from traditional methods based on feature tracking to more modern Deep Learning-based methods [46].

Despite the challenges involved in measuring optical flow, it remains an important area of research in computer vision and image processing. Advances in machine learning, particularly deep learning, have led to significant improvements in the accuracy and robustness of optical flow estimation. As such, it is expected that optical flow will continue to play a critical role in many applications, from autonomous driving to robotics and beyond.

But great results can be also obtained without the necessity of complex AI methods. The classic optical flow techniques have been shown to perform well for certain tasks. These classic methods can be grouped into four categories [48]: Differential Techniques, Region-Based Matching, Energy-Based Methods, and Phase-Based Techniques.

Differential Techniques provide accurate results with fast algorithms. These techniques extract the information by making assumptions about spatial and temporal variations of the image intensity $I(\boldsymbol{x}, t)$, or filtered versions of the image. From the assumption that the intensity is conserved $\frac{dI}{dt} = 0$. the gradient constraint equation can be derived:

$$\nabla \boldsymbol{I}(\boldsymbol{x}, t) \cdot \boldsymbol{v} + \boldsymbol{I}_t(\boldsymbol{x}, t) = 0, \tag{32}$$

where $I_t(\boldsymbol{x}, t)$ is the partial time derivative of $I(\boldsymbol{x}, t)$ with respect to t and $\boldsymbol{v} = (u, v)^T$ [48]. But there are two unknown components of $\boldsymbol{v}$ in Eq. 32, constrained by only one linear equation. Horn and Schunk method introduces a smoothness assumption on the spatial variation of the optical flow in order to recover a unique solution of $\boldsymbol{v}$ [49]. On the other hand, Uras, Girosi, Verri and Torre method uses the second-order differential (33) to solve both components of $\boldsymbol{v}$. This second-order differential adds more constraints to calculating the 2D velocity,

$$\begin{bmatrix} I_{xx}(\boldsymbol{x}, t) & I_{yx}(\boldsymbol{x}, t) \\ I_{xy}(\boldsymbol{x}, t) & I_{yy}(\boldsymbol{x}, t) \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} + \begin{bmatrix} I_{tx}(\boldsymbol{x}, t) \\ I_{ty}(\boldsymbol{x}, t) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{33}$$

Lucas and Kanade method uses a weighted least-squares fit of local first-order constraints (32). This method intends to minimize (34) for each small spatial neighbourhood Ω.

$$\sum_{\boldsymbol{x} \in \Omega} \boldsymbol{W}^2(\boldsymbol{x}) [\nabla \boldsymbol{I}(\boldsymbol{x}, t) \cdot \boldsymbol{v} + \boldsymbol{I}_t(\boldsymbol{x}, t)]^2, \tag{34}$$

where $W(\boldsymbol{x})$ denotes a window function that gives more influence to the constraints at the centre than at the edges. The solution to $\boldsymbol{v}$ is given as,

$$\begin{aligned} \boldsymbol{v} &= [\boldsymbol{A}^T \boldsymbol{W}^2 \boldsymbol{A}]^{-1} \boldsymbol{A}^T \boldsymbol{W}^2 \boldsymbol{b}, \\ \boldsymbol{A} &= [\nabla I(x_1), ..., \nabla I(x_n)]^T, \\ \boldsymbol{W} &= diag[W(x_1), .., W(x_n)], \\ \boldsymbol{b} &= -[I_t(x_1), ..., I_t(x_n)]^T, \end{aligned} \tag{35}$$

Which is solved when $\boldsymbol{A}^T \boldsymbol{W}^2 \boldsymbol{A}$ is nonsingular [48]. According to J. L. Barron et al. [48], the most reliable methods are the first order, local differential method of Lucas and Kanade [50] and the local phase-based method of Fleet and Jepson [48].

4.7 SURF Algrithm

The SURF algorithm is a popular computer vision technique for detecting and describing features or "interest points" in images. The algorithm is designed to find points in the image that are well-defined positions, rich local image structure, and stable under various image domain perturbations. And then

extract a set of descriptors from these interest points that can be used to recognize these features in other images.

The algorithm is composed of two main steps: the detection and the description of features. In the detection step, SURF uses a scale-space representation of the image to detect scale-invariant features. The algorithm applies a series of kernels (i.e. box filters) of increasing size to the original image and analyzes the results to find points that are consistent across scales. These points are then further analyzed to determine their location, orientation, and scale [51].

In the description step, SURF computes a set of feature descriptors that are invariant to image transformations such as rotation, scaling, and changes in lighting. This is accomplished by three sub-steps: threshold comparison, non-maximal suppression, and interpolation of nearby data to find location, that captures the important characteristics of the image at that point. These descriptors are then used to match features across images or to recognize objects in an image.

The main advantages of the SURF algorithm are its speed and robustness to image transformations. The algorithm's high performance and accuracy are due to its use of an intermediate image representation known as the Integral Image and the Hessian-matrix approximation, which provides highly accurate results [51]. The algorithm also utilizes the scale-space concept and divides it into several octaves, where each octave contains a series of response maps computed by specific filters. As a result, the SURF algorithm has found widespread use in applications such as object recognition, image stitching, and augmented reality.

5 Method

5.1 Distance to the wall

To observe the drone's linear velocity, it is necessary to estimate the distance to the reference object. In this case, the drone is assumed to move in front of a wall that will work as a reference. Before taking off (Fig. 8a), the drone finds the line between the floor and the wall and calculates the distance to the wall according to the distance between the line and the centre of the picture.

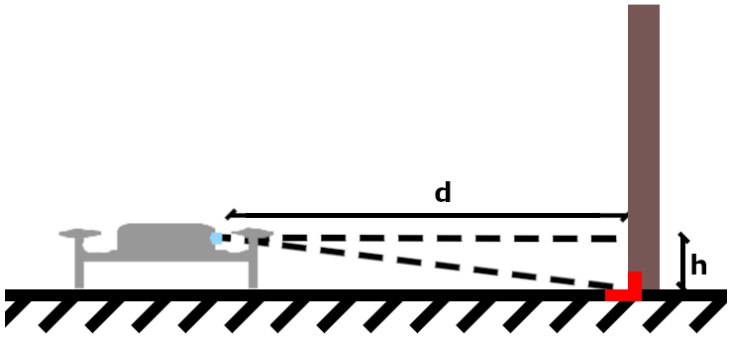
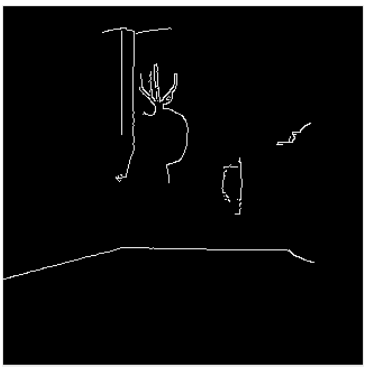


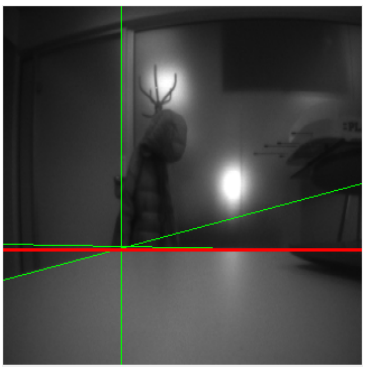
Figure 7: *Estimating distance to the wall*

To find the line between the floor and the wall, Fig. 8, first, the Canny edge detection filter is applied to the image to find the borders of all the objects in the image Fig. 8b. Then, Hough Transform is applied to the filtered image to find straight lines. This method finds lines that cross through a high amount of points. Once the lines in the picture are obtained, it is time to determine which of them corresponds to the line between the wall and the floor, Fig. 8c. By assuming that the floor has no horizontal lines and the camera is pointing straight to the wall, the line of interest is the closest horizontal line to the bottom of the image.

(a) *Image taken by the drone before taking off*



(b) *Image filtered with Canny edge detection filter*



(c) *Straight lines found by Hough Tramsform method*

Figure 8: *Method to find the line between the wall and the floor*

The distance to the wall is calculated using the pinhole camera model [52] that relates a three-dimensional space with a two-dimensional projection where the light rays from a point in the physical scene pass through a pinhole and project that point on the bi-dimensional camera sensor, Fig. 9,. The relation between the size and distance of the object in the real environment compared to the size in the projection is given by,

$$\frac{h_c}{d_c} = \frac{h}{d}. \tag{36}$$

Where d and h refer to the distance between the camera and the object and the height of the object respectively, and d_c and h_c refer to the focal length of the camera and the height of the object in the picture respectively.

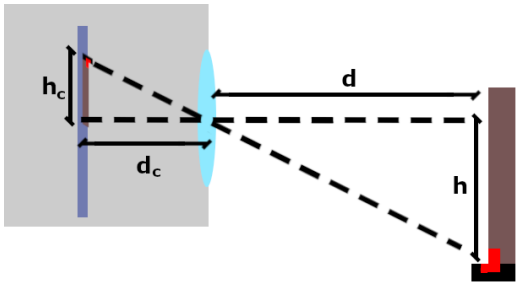


Figure 9: *Pinhole camera model*

Applying it to the case studied (Fig. 7), the height of the drone h, the focal length d_c and the size of each pixel s_{px} are known (Table 1). Hence the distance between the drone and the wall d is obtained using the linear relation with the distance in pixels h_{px} between the centre of the image and the centre of the line between the wall and the floor (37),

$$d = \frac{h}{h_{px} s_{px}} d_c. \tag{37}$$

The performance of this method has been empirically tested and observed that the real distance and the distance in pixels have a squared inversely proportional relationship instead of an inversely proportional relationship as was approximated previously due to the camera lens. (37) is modified to fit the data measured (38), obtaining a Mean Squared Error of 3.134 cm^2. Fig. 10 shows the estimation and real measurement using (38),

$$d = 100 \frac{h}{h_{px}^2 s_{px}} d_c. \tag{38}$$

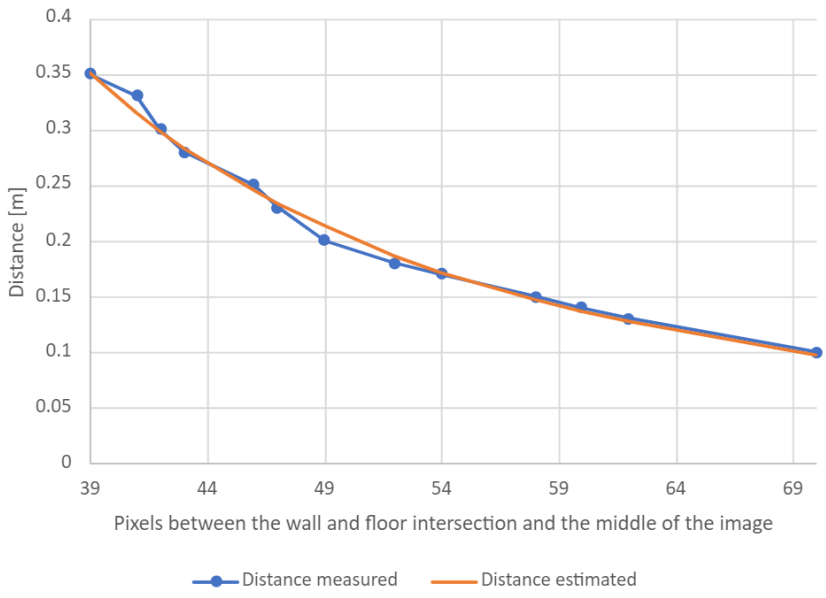


Figure 10: *Distance measured and the distance estimated to a wall*

This distance estimation is only valid for distances under 40 cm since the pixel variation over this distance is too small. Because of this, the noise detecting the line change greatly the distance estimated.

Better distance approximations could be made by calibrating the camera parameters [53], but it is not necessary for a low-quality camera as the AI-deck one.

5.2 Position, velocity and angle observation

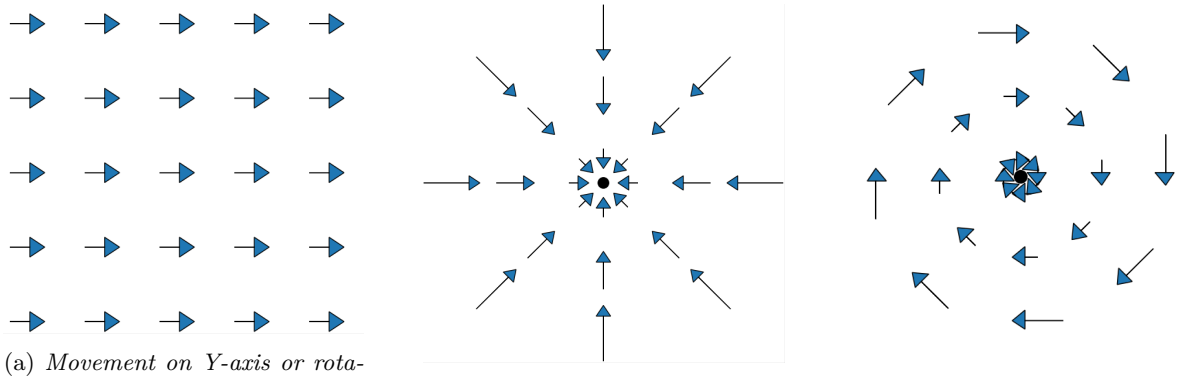
This section delves into the techniques used for observing the position, velocity and inclination of the drone, specifically through the use of machine vision. The section covers the various algorithms employed to process visual data, as well as the challenges that need to be overcome to obtain accurate measurements in a dynamic environment.

5.2.1 Optical flow method

This method consists in finding interest points or features in each frame and matching the features between two consecutive frames. The tools used to find and match features are from the library OpenCV. The first function used is *goodFeaturesToTrack*. This function identifies corners in an image that are suitable for tracking using the Lucas-Kanade optical flow algorithm [50]. The function takes an input image, a maximum number of corners to detect, a quality level, and a minimum distance between detected corners. It then returns the detected corners as a set of points, ranked by their "goodness" as determined by the minimal eigenvalue or the Harris function response [54].

Then, the function *calcOpticalFlowPyrLK* is used for computing the Lucas-Kanade optical flow algorithm [50] between two frames, which tracks a set of feature points across frames by estimating their displacement in the image plane. This will provide a set of directions and angle of displacement of the features between two consecutive frames. With this displacement, it is intended to distinguish among three different movements (Figure 11).

The first possible case (11a) corresponds to a movement parallel to the reference background, or a yaw variation, these two movements will trigger a horizontal image flow. The second case (11b) corresponds to a movement of approaching or moving away from the background, triggering a feature displacement outward or toward the centre respectively, with an increasing displacement from the centre to the edges of the image. The final case (11c) is a rotation movement that corresponds to a variation in the roll, triggering a rotation of the features centred in the middle of the image. There is an additional case similar to the first case where the drone moves vertically or there is a variation in the pitch, but this scenario is not considered since the altitude control doesn't use the sensor fusion method. All other image flow patrons will correspond to a combination of these basic cases.



(a) *Movement on Y-axis or rotation on Z-axis* (b) *Movement on X-axis* (c) *Rotation on X-axis*

Figure 11: *Optical flow expected on different movements*

Some assumptions must be done to quantify the real drone's movement. First, the reference object that is observed is assumed to be a wall perpendicular to the quadcopter and it must have enough texture details to be able to find features, the light conditions are controlled, without high contrast lights inferring on the wall, and the real perpendicular distance to the wall is known.

In the cases where there is a movement, the pixel displacement can be calculated as the mean displacement between all the features, while in the rotation cases, the angle is the mean of the angles calculated using the pixel displacement and distance to the centre of the image. To convert the distance in pixels to meters, (37) is used, but this time the distance to the wall d is known and the distance between two points on the wall h is calculated.

5.2.2 Strip method

For this method, a high-contrasted strip has been set on a wall. Each image is computed to find the vertical strip. All the vertical lines are detected by performing the Hough Transform over the image filtered by a Canny edge detection filter, similar to how it was done to detect the intersection between the wall and the floor. In this case, the Hough transform is calculated just around 10° from the vertical since it is assumed the drone will rotate small angles around 0°.

To find the strip among all the lines obtained, first, the lines are sorted by the ρ parameter of the Hough transform since this will sort them according to their position in the image. Now it is easier to check the distance between consecutive lines. To remove all the redundant lines, lines with a ρ closer than 5 pixels to any other line are removed. Then, the remaining lines are sorted by the θ parameter to find lines with similar inclinations. Finally, a possible strip will be two consecutive-in-θ lines that are close. In the beginning, if several possible strips are found, the closest strip to the middle of the image is taken as the real strip. In future estimations, when several strips are detected, the strip taken for the estimation is the closest one to the strip calculated in the previous iteration,

$$d(R, P) = \frac{|m \cdot p_x - p_y + c|}{\sqrt{m^2 + 1}}. \tag{39}$$

From the two lines that define the edges of the strip (represented in dark blue in Figure 12), three data are obtained: inclination φ, position d and width w. First, it is obtained the middle line (in light blue 12) in the strip as the mean of the Hough parameters. The inclination φ is the parameter θ of this middle strip. Then, the perpendicular distance between the middle line and the centre of the image is calculated (39), this distance is the position d of the strip. Finally, the perpendicular distances between each edge of the strip and the centre of the image are calculated. The absolute value of the distance between these two distances is the width w. The perpendicular distance between a point $P = (p_x, p_y)$ and a line $R, y = mx + c$, is given by (39).

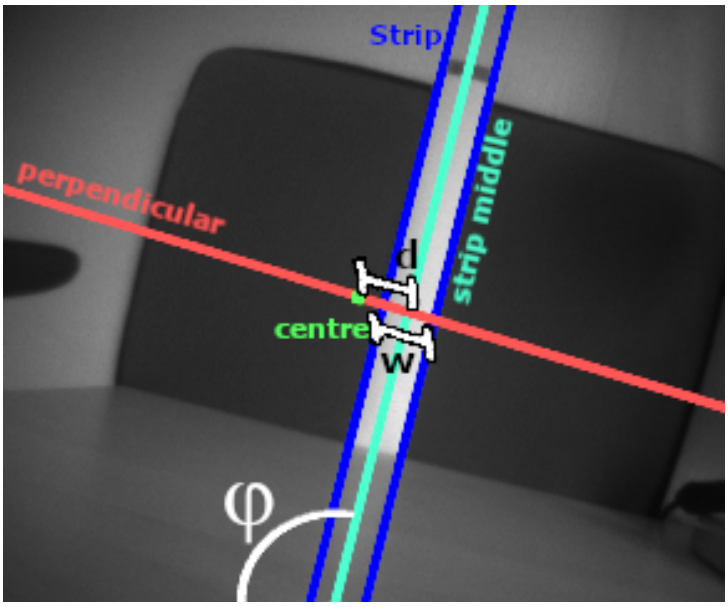


Figure 12: *Observed information from the camera*

The position, velocity and angle observation are derived from the three parameters of the strip. On the y-axis, the position is calculated as the difference between the current strip position d_k and the initial strip position d_0, while the velocity is the difference between the current strip position d_k and the previous strip position d_{k-1} divided by the time between the two frames. The roll, ϕ, of the drone is defined as the difference between the current strip inclination φ_k and the initial strip inclination φ_0.

On the x-axis, the position is calculated as the difference between the current strip width w_k and the initial width position w_0, while the velocity is the difference between the current strip width w_k and the previous strip width w_{k-1} divided by the time between the two frames. Unfortunately, the pitch can not be observed using this method.

To convert the distance in pixels to meters, (37) is used, but this time the distance to the wall d is known and the distance between two points on the wall h is calculated.

5.3 Sensor Fusion

To fuse the data from the image and gyro, a Kalman Filter is implemented. First of all, the state vector that defines the model must be derived. It is possible to simplify Eq. 10 assuming that the yaw is not changing, and linearize around pitch and roll equal to zero.

$$\begin{bmatrix} \ddot{p_x} \\ \ddot{p_y} \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \theta \\ -\phi \end{bmatrix} \frac{F}{m}. \tag{40}$$

Once the acceleration is obtained (40), the roll and pitch, and the linear velocity are obtained by integrating the angular velocity and the linear acceleration. With a small sampling time, the integration can be approximated using the sampling time, T_s, as $v_{k+1} \approx v_k + \dot{v}_k T_s$.

$$\begin{bmatrix} \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \theta_{k+1} \\ \phi_{k+1} \end{bmatrix} \approx \begin{bmatrix} \dot{x}_k + \ddot{x}_k T_s \\ \dot{y}_k + \ddot{y}_k T_s \\ \theta_k + \dot{\theta}_k T_s \\ \phi_k + \dot{\phi}_k T_s \end{bmatrix}. \tag{41}$$

Using (40) and (41), the state vector $\boldsymbol{X}_k$ of the filter for a discrete time k is obtained. To correct the drift in the gyro, a biases in x- and y-direction $[b_y, b_x]$ a added to the state,

$$\boldsymbol{X}_{k+1} = \begin{bmatrix} y_{k+1} \\ \dot{y}_{k+1} \\ \phi_{k+1} \\ b_{y|k+1} \\ x_{k+1} \\ \dot{x}_{k+1} \\ \theta_{k+1} \\ b_{x|k+1} \end{bmatrix} = \begin{bmatrix} y_k + \dot{y}_k T_s \\ \dot{y}_k - \phi_k T_s \frac{F}{m} \\ \phi_k + (b_{y|k} + \omega_{x|k}) T_s \\ b_{y|k} \\ x_k + \dot{x}_k T_s \\ \dot{x}_k + \theta_k T_s \frac{F}{m} \\ \theta_k + (b_{x|k} + \omega_{y|k}) T_s \\ b_{x|k} \end{bmatrix}. \tag{42}$$

For small attitude angles $\boldsymbol{u}_k = [\omega_{x|k}, \omega_{y|k}] \approx [\dot{\phi}_k, \dot{\theta}_k]$ as it is rate gyro readings, the system input.

To estimate the relative position and velocity of the drone, two Kalman Filter are implemented to fuse the data from the IMU sensor and the camera observation. Each Kalman Filter aims to estimate the movement on a different axis, x and y. The output expected from the filter is the drone's speed on the horizontal plane $[\dot{x}, \dot{y}]$. Using two different Kalman Filter for the two axes allows to validate and reject the observations independently, taking advantage of the fact that the estimation of the movement in each axis is independent.

5.3.1 Y-axis Kalman Filter

The input for the prediction is the roll rate gyro readings $\boldsymbol{U} = \omega_x \approx \dot{\phi}$ and the observation is the observed position, speed and inclination by the camera on the y-axis, $\boldsymbol{Z} = [y, \dot{y}, \phi]^T$.

The transition matrix $\boldsymbol{A}$, relation matrix $\boldsymbol{B}$ for the prediction phase and state vector $\boldsymbol{X}$ (18) are defined as

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & -T_s\frac{F}{m} & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix}; \ \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ T_s \\ 0 \end{bmatrix}; \ \boldsymbol{X} = \begin{bmatrix} y \\ \dot{y} \\ \phi \\ b_y \end{bmatrix}. \tag{43}$$

And the matrix $\boldsymbol{H}$ that relates the measurements to the inertial state (16) is defined as,

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{44}$$

5.3.2 X-axis Kalman Filter

There are only small variations in this filter compared to the filter developed for the y-axis. The input for the prediction is the pitch rate gyro readings $\boldsymbol{U} = \omega_y \approx \dot{\theta}$ and the observation is the observed position and speed by the camera on the x-axis, $\boldsymbol{Z} = [x, \dot{x}]^T$. In this case, the angle is not observed, hence the observation matrix must be modified.

Where $\omega_y = \boldsymbol{u}_k$ is the input vector of the filter. The transition matrix $\boldsymbol{A}$ and relation matrix $\boldsymbol{B}$ for the prediction phase and the state vector $\boldsymbol{X}$ (18) are defined as,

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & T_s\frac{F}{m} & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix}; \ \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ T_s \\ 0 \end{bmatrix}; \ \boldsymbol{X} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ b_x \end{bmatrix}. \tag{45}$$

And the matrix $\boldsymbol{H}$ that relates the measurements to the inertial state (16) is defined as

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{46}$$

5.4 Control

The first attempt to control the attitude (ϕ, θ, ψ) and elevation p_z is using PID control based on the position and attitude model. Crazyflie firmware provides a function $sent\_setpoint(\phi, \theta, \dot{\psi}, F)$ that takes the desired *roll*, *pitch*, *yaw rate* and *thrust* F and controls the drone using already tuned PID controllers. The attitude rate is measured with the gyro sensor onboard the aircraft, and from it, the attitude is estimated. Since we want to control the attitude to make the drone stay stable, this function can be used to set *roll*, *pitch* and *yaw rate* to zero.

5.4.1 Altitude Control

To control the altitude it is required to implement a new PID controller. The position p_z is estimated using the onboard pressure sensor. This estimation is provided already by the drone. The control function used is (12) for altitude control.

5.4.2 Attitude Control

Attitude control is done using a cascade control architecture with PID controllers, as presented the Figure 4. The Attitude PID and Attitude Rate PID are already integrated into Crazyflie firmware using the function $sent\_setpoint(\phi, \theta, \dot{\psi}, T)$, so only the position PID and velocity PID have to be implemented. The estimated velocity and position are obtained from the Kalman Filter developed previously. Two different cascade controllers are developed for the x and y axes, and the output of the velocity PID is introduced as parameters ϕ and θ in the function $sent\_setpoint(\phi, \theta, \dot{\psi}, T)$. No significant drift has been observed on the yaw, ψ, so it is not necessary to control it more than set the $\dot{\psi}$ parameter to zero in the previous function.

6 Development

This section covers the development process of quadcopter stabilization and how machine vision and sensor fusion were used to estimate the position and control the movement of the drone. The quadcopter was carefully selected to ensure that the drone could be programmable and teleoperable, and that the data obtained by it could be received by an external computer. Machine vision algorithms were used to analyze real-time images, allowing the quadcopter to estimate the relative position and velocity. Sensor fusion techniques were employed to combine data from multiple sensors for more accurate control and stabilization.

6.1 Research

The initial phase of this project focused on conducting research to identify the optimal hardware for the quadrotor. Several factors were considered during this process, including the potential for accidents and harm to individuals. After careful evaluation, it was recommended that a light drone be used since this would reduce the likelihood of damage or harm occurring. Additionally, small drones were deemed more suitable for use in enclosed spaces, further contributing to their appeal. Other requirements for the quadrotor were an embedded IMU, a front camera and the possibility of being easily programmed.

Several options for the quadrotor were considered, including Parrot Mambo FPV [55], Sky Viper [56], Drona Aviation Pluto [57], and Bitcraze Crazyflie [12]. Ultimately, the Bitcraze Crazyflie was selected due to its lightweight and the availability of a Python API. This feature allows the drone to receive all sensor readings and camera images, as well as to be controlled remotely from a computer. While Bitcraze does provide tools for modifying the drone's firmware, an external drone control was chosen to take advantage of the versatility and machine vision tools available in Python.

After acquiring the necessary hardware for the project, assembling, setting up, and connecting the drone to a computer did not present any significant challenges. However, a major obstacle was encountered when attempting to run the AI-deck, which is responsible for recording and sending images captured from the drone.

The primary issue with the AI-deck was that it needed to be flashed before it could be used, which was not an intuitive process. This caused some initial confusion and delay in the project's progress. Nonetheless, this challenge was overcome through research and consultation with technical experts. Overall, the process of assembling, setting up, and connecting the drone to a computer was relatively straightforward.

Many similar projects use optical flow and feature detection methods, even though most of them use a facing-down camera instead of a front camera. Facing-down cameras allow for easily detecting the horizontal movement of the drone in both X and Y-axes and the yaw rate using some optical flow methods.

One crucial aspect of optical flow methods is calculating the distance between the camera and the reference object to translate the distance in pixels into real-world distance. Facing-down cameras estimate the movement of the drone based on the floor's reference point, which makes it easy to obtain the distance to the floor using an embedded pressure sensor commonly used in drones.

In contrast to a facing-down camera, a front camera requires a different method to estimate the distance to reference objects, such as walls. One approach is to calculate the distance to the object before the take-off and estimate the position of the reference while flying. Optical flow methods in these systems allow for estimating movement on all three axes, including the attitude rate. However, in most cases, it is not possible to distinguish between pitch rate and movement in the Z-axis, or yaw rate, and movement in the Y-axis solely based on optical flow data.

Despite this challenge, many drones require a front camera for their primary function. Therefore, using it for stabilization would eliminate the need for additional sensors, simplifying the design and reducing the weight of the drone. However, it is important to note that using a front camera for stabilization may

not be as effective as using a facing-down camera due to the limitations of optical flow data.

Overall, while a front camera presents some unique challenges for estimating distance and movement, it remains an essential component of many drone applications. Utilizing a front camera for stabilization has the potential to simplify the drone design and reduce its weight, but careful consideration should be given to the limitations of optical flow data in accurately estimating movement on all three axes.

6.2 Drone's first steps

The Crazyflie can be controlled using the library provided by Bitcraze. This library gives basic instructions for adjusting the drone's attitude, which includes adjusting its roll (ϕ), pitch (θ), yaw rate ($\dot{\psi}$) and thrust (T). By using a PID controller, it is possible to regulate the drone's altitude. This is done by setting a desired altitude and using the pressure sensor as a feedback signal to calculate the thrust required to maintain that altitude. The technique used for regulating the altitude of the drone is explained in detail in Section 5.4.2 of the paper.

However, once the drone reaches the desired altitude, it is observed that it begins to drift in the horizontal plane. This drift is due to errors in the estimation of its roll and pitch angles, and can occur in both the X and Y axes, but it is not observed in the yaw. To counteract this drift, it is necessary to adjust the drone's roll and pitch angles according to the velocity and position of the drone.

The library's control loop and image rate operate at a frequency of 10 Hz, which provides a balance between control accuracy and computational efficiency. The gyro reading rate, which measures the drone's rotational velocity, is much faster at 100 Hz, allowing for precise control of the drone's orientation. However, when sending images, there is a delay of 360 ms, which can impact the drone's responsiveness to adapt to observed conditions.

6.3 Machine Vision

The first method (Section 5.2.1) involves finding points of interest or features in the image and matching them with the points found in the previous frame. This optical flow between two consecutive frames is used to observe the velocity and roll rate. However, the main problem encountered with this method is that the algorithm used to match points between two frames is not able to find enough quality points on the image. Most of the matches detected are grouped in a few clusters, Fig. 13, which does not provide information about the global image. Moreover, the matches are not consistent in time, making the prediction of movement unreliable, particularly for the movement on the X-axis. This is because it needs points from the entire image to compare the flow direction of pixels.

Figure 13: *Example Optical Flow matched points*

Although another algorithm such as the SURF could solve this issue, due to the lack of time and the uncertainty of whether a better algorithm could solve the problem, it was decided to look for a different approach. The strip method (Section 5.2.2) is an alternative approach that ensures the detection of the required features in the image. While this method differs more from a real-case scenario since not all environments will have a vertical strip to use as a reference, it provides the necessary information from the images to continue with the project. This method provides different observations, including the velocity on the X- and Y-axes, as well as the relative position and roll. This multi-observation approach can improve the estimation of the Kalman Filter, making it more reliable.

Both optical flow and strip methods bring a new problem since they can only calculate the distance in pixels in the image and require additional information to calculate the distance in the real world. Traditionally, this problem is solved using another sensor that can measure the distance between the drone and the reference captured by the camera. For instance, the stabilization systems where the camera used is pointing downwards take the altitude information from an onboard pressure sensor. In systems with front cameras, an ultrasonic front sensor could estimate the distance. However, the objective of this work is to eliminate all sensors that are not strictly necessary, so we calculated the distance using only the camera (Section 5.1).

The distance to the reference object is calculated when the drone has not taken off yet to ensure that the drone is completely horizontal and to have a precise knowledge of the altitude of the camera (since it will be the height of the drone). The accuracy of this calculation will drop for distances over 40 cm since a difference of one pixel can significantly change the distance estimate in that range.

6.4 Sensor fusion

The decision to implement the Kalman Filter in this study was primarily based on prior knowledge and experience using this method, and many similar works use Kalman Filtering in these cases. For simple cases where the drone only needs to remain stable around the initial conditions, it is possible to linearize the system, rendering the EKF unnecessary. However, for more advanced drone behaviours, EKF is required. Two Kalman Filters were implemented in this study (Section 5.3), one to estimate movement on the X-axis and another to estimate movement on the Y-axis.

There is a significant delay in the observations due to a communication delay of 350 ms and a computation delay of 100 ms. The communication delay is the time that passes from when a frame is taken until that frame is received in the computer. And the computation delay is the time required to process and obtain the parameters from a frame. As a result, the estimation in the Kalman Filter is done using an observation out of phase. To address this issue, a buffer is created to store the last IMU inputs. Before estimating the new state using the observation, a prediction of the state is computed using the closest IMU readings to the time when the observed image was taken. Although this method cannot completely eliminate the effect of the significant delay, it can mitigate it and improve the estimation.

6.5 Control

After estimating the drone's position and velocity, the control system can be implemented. The control system in this study is divided into three independent control loops. The first one is responsible for controlling the altitude, while the other two are cascade control loops for controlling the displacement on the X- and Y-axis, respectively.

The altitude control (Section 5.4.1) controls the thrust and uses the pressure sensor reading as feedback signal. The goal of this control loop is to develop a simple system that can keep the drone in the air to evaluate the primary focus of this study, the attitude control system. Although the pressure sensor readings are not always accurate and robust, the performance is sufficient to keep the drone stable at a constant altitude.

For the main control systems, the attitude controllers (Section 5.4.2), two cascade architectures are implemented, one for each axis to control. These controllers use the same architecture as the controllers

used by the Crazyflie quadcopter when using certain decks that provide position and velocity data. However, the frequency of the position and velocity controllers in this study is slower than the onboard controllers developed by Bitcraze. While the onboard controllers run at 100 Hz, the controllers in this study operate at a slower rate of 10 Hz. This slow rate, combined with the delay in observation, can significantly reduce the control system's performance.

6.6 Simulation

In order to study the performance of the system neglecting the effect of the delay, a simulation using Simulink has been suggested. This simulation used the Crazyfly model developed by Peter Corke [58]. The simulation uses a Kalman Filter with noisy real position, angle and angle rate data to estimate the position and velocity. The angle rate signal has an offset to simulate the measurement noise that usually produces the drift in these systems. Two cascade controllers are implemented to control the position and velocity since the model already controls the angle and angle rate as the real drone.

7 Results

Some of the test scenarios presented in Section 1.3 are performed and presented in this section. Some of the tests have been done in a simulated environment, while others were performed in a real environment using the Crazyflie quadcopter.

7.1 Results from Time Test

This test aims to study the computation and communication times in the different systems in order to find bottlenecks that can affect performance. The mean measured time during a test of the full system is represented in Table 2.

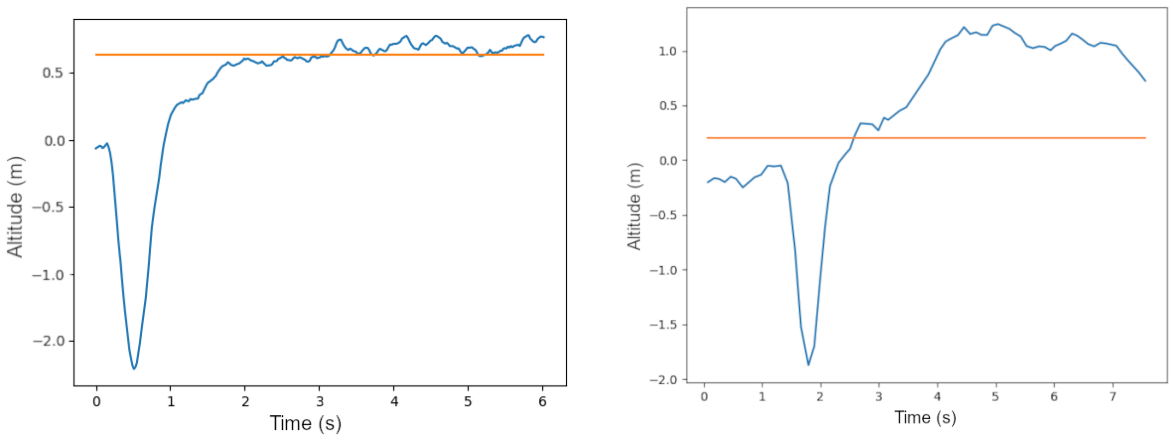
Table 2: *Time results table*

System	**Time** (s)
Kalman Filter Fusion	0.0029
Kalman Filter Prediction	6.73e-5
Control Loop	0.101
Feature extraction	0.0112
IMU communication rate	0.019
Camera communication delay	0.357
Camera communication rate	0.13

It is observed that all the systems can perform under the control loop time, but the camera communication delay. This large delay makes the system not controllable by current controllers.

7.2 Results from Altitude Tests

For the results in the altitude control, it is important to distinguish two different scenarios, since different behaviours have been observed depending on the battery level of the drone. In the first scenario, the drone is completely charged, resulting in proper altitude stabilization (Fig. 14a). In the second scenario, the battery lacks full charge, causing the drone to be incapable of maintaining a consistent altitude. (Fig. 14b).



(a) *Battery fully charged* (b) *Battery not fully charged*

Figure 14: *Real Scenario: Altitude Control*

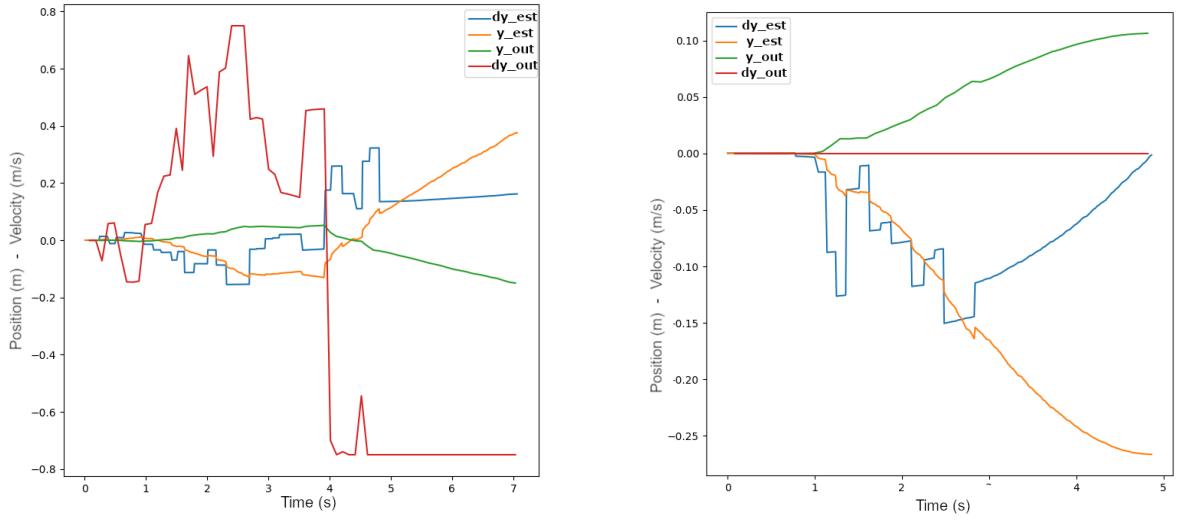
In Fig. 14 the blue line represents the pressure sensor reading, while the orange line represents the setpoint.

7.3 Results from Position Tests

Some of the tests in this section compare the results of flying with position and velocity control, in addition to the attitude and attitude rate controllers already integrated into the drone, or flying using only the attitude and attitude rate controllers onboard. In this first situation, the Kalman filter has been implemented to estimate the position and velocity thanks to the image and IMU data fusion, allowing position control. The second situation uses only the IMU data, and position and velocity can not be accurately estimated only with this information, hence the position can not be controlled directly.

The first test carried out on the real quadcopter consists of making the drone fly around 25 *cm* away from a wall with a vertical high-contrast strip pasted on it. In addition, a light pointing to the wall has been set up to keep high and constant brightening levels.

For this test, the drone is expected to fly up to 20 *cm* and stay still in the air (Fig. 15a) until the strip on the wall is not detected for 10 iterations in a row. The test is repeated with the deactivated position and velocity control system, just using the attitude and attitude rate controllers onboard (Fig. 15b).



(a) *Using position control* (b) *Without position control*

Figure 15: *Real Scenario: Stabilization test*

The orange and blue lines represent the estimated position and velocity by the Kalman Filter respectively, while the green and red lines are the output signal of the position and velocity controllers, respectively. In Figure 15b the velocity control signal is zero since the control is deactivated. In the first image (15a) the system tries to compensate for the error in position and velocity, but it becomes unstable and overcompensates the error, while in the second image (15b) the system does not try to compensate the error, so it just drifts away.

No ground truth data can be shown for this experiment since this would require external sensor equipment, flash camera, or good GPS-receiver, and there was no external ground truth equipment in this project. Instead, an internal estimation of the position and velocity are computed. Even when it is not possible to know exactly how good it is the estimation since there are no external sensors, the KF uses a covariance that tells within some confidence that the quadcopter is where it is estimated to be.

No further experiments were tested on the Crazyflie since it can not successfully pass the first scenario stabilizing the drone. The following tests are carried out on a simulated model on Simulink.

Three different scenarios are tested in the simulation using the parameters in Table 3. These parameters have been selected in order to fit the noise model used for the KF, behave with a similar drift when there is no additional control, and the sampling time used in the real drone. The offset and noise introduced in the angle rate simulate the IMU measurement error that produces the drift while the noise on the position and attitude of the KF simulated the error in the measurement of the position and inclination of

the strip. The communication delay added to the output of the Kalman Filter simulates the computation and communication delay between the drone and the computer. This delay is smaller than the delay measured in the real system to be able to control the system. Finally, the sampling time refers to the working rate of the Kalman Filter and position and velocity controllers. The global position and velocity are computed in the simulation, hence they are used in the graphics as a ground truth.

Table 3: *Simulation parameters*

Parameter	**Value**
Position X noise power KF	$1^{-5}\ m$
Position Y noise power KF	$1^{-5}\ m$
Roll noise power KF	$10^{-6}\ rad$
Roll rate offset	$-10^{-3}\ rad/s$
Pitch rate offset	$-10^{-3}\ rad/s$
Roll rate noise power	$5\ 10^{-7}\ rad/s$
Pitch rate noise power	$10^{-7}\ rad/s$
Communication delay	$0.11\ s$
Sampling time	$0.1\ s$

The noise frequency has been calculated using the maximum frequency of the system where the noise is applied [59] (47),

$$f_{noise} = 100\frac{f_{max}}{2\pi}. \tag{47}$$

In the first scenario, Fig. 16, the drone is expected to stay stable in the air for 30 s, using a setpoint of 0 (for position while using the position control and for attitude while using only the attitude control). In the results, it is observed how the drone stays stable around the initial point $(0, 0)$ compensating the noise and drift when the position is estimated and controlled (16a and 16b). On the other hand, the drone drifts over time (16c and 16d) when the position control is not introduced.

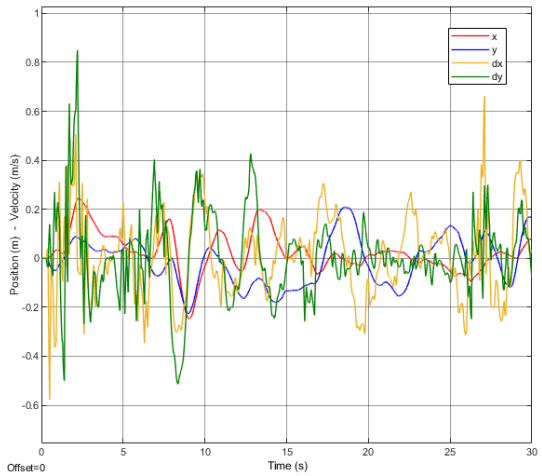
This experiment is repeated using a communication delay of 0.35 s instead of 0.11 s, Fig. 17. When the delay in communication increases, the system becomes unstable, oscillating more and more around the setpoint.

In the second scenario, Fig. 18, the drone gets a step setpoint of 1 m at time $t = 4\ s$ for X-axis and $t = 7\ s$ for Y-axis. This test can not be performed without position control. The drone follows the step setpoints successfully after 15 s.

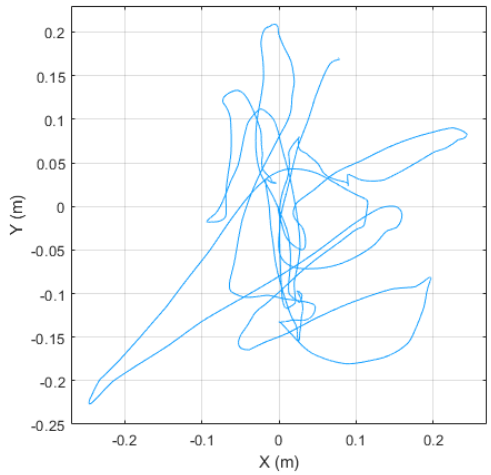
The final test consists of introducing a sinusoidal setpoint of frequency 1 Hz on X and Y-axes to make the drone perform a circular trajectory. The yaw is not changed in this or any other test, since the drone is supposed to point to the wall all the time. This experiment was repeated with a communication delay of 0.01 s and then, without the noise in the position and angle observed to observe their effect on the performance. To quantify the error the Mean Square Error (MSE) between the trajectory reference and the position is used. This error in the trajectory in the three cases is presented in Table 4. In Case A (19a) the system can follow the circular trajectory without becoming unstable. The error, in this Case, is the highest ($0.188\ m^2$).

In the second Case (19b), the communication delay between the position estimation and the control is reduced from 0.11 s to 0.01 s, highly improving the error at the beginning of the flight, since the quadcopter reaches the steady state faster, but the steady-state error is still high so the overall MSE is not that low. It is also important to have into consideration that even when the delay in communication is 0.01 s, the controllers and Kalman filter sampling time is 0.1 s. In the final Case (19c), The noise in the Kalman Filter is removed, obtaining the lowest MSE. Contrary to the previous Case, the system takes longer to reach a steady state, but the steady-state error is smaller since there is no noise in the position estimation.

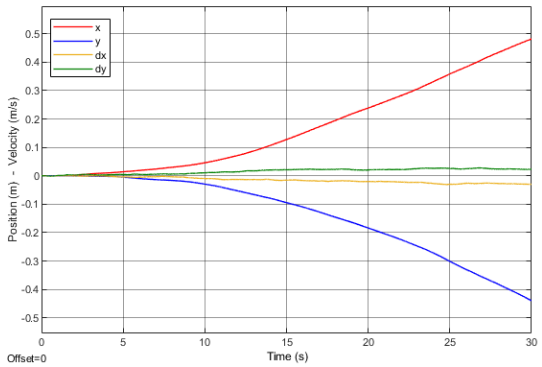
An additional experiment is carried out in the simulated model to test a system without position estimation and control, using only the velocity estimation and control, Fig. 20. This control method requires



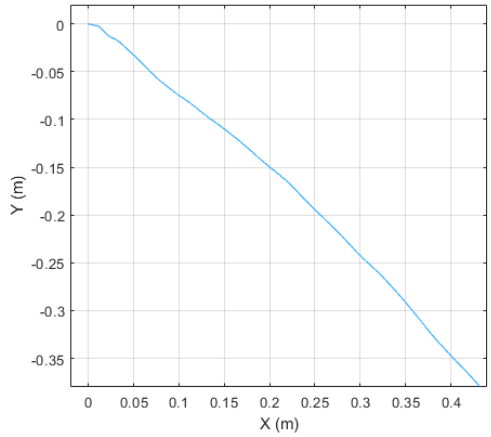
(a) *Position and velocity using position control*



(b) *Movement on XY plane using position control*



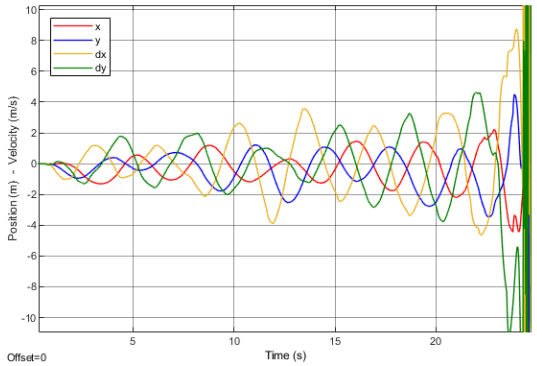
(c) *Position and velocity without position control*



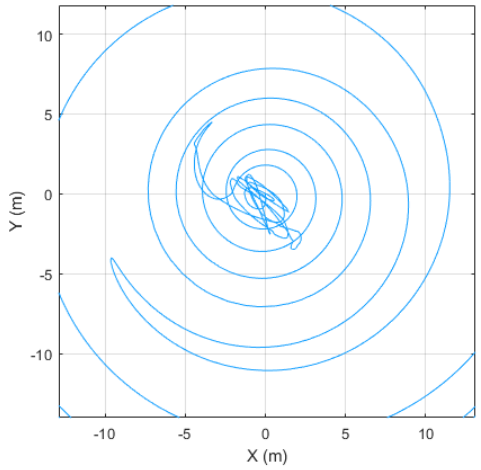
(d) *Movement on XY plane without position control*

Figure 16: *Simulation First Scenario: Stabilization test*

one less controller and a simpler Kalman Filter, but it can not compensate for all the drift.



(a) *Position and velocity with position control and 0.2 s of delay*

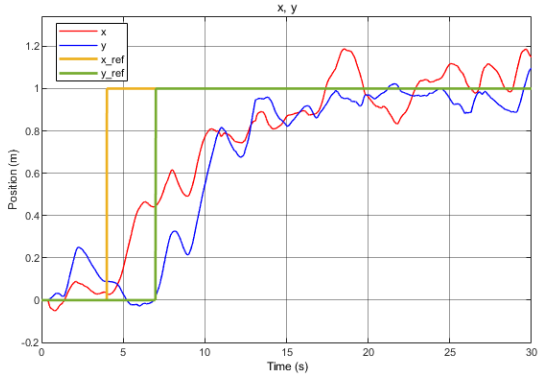


(b) *Movement on XY plane with position control and 0.2 s of delay*

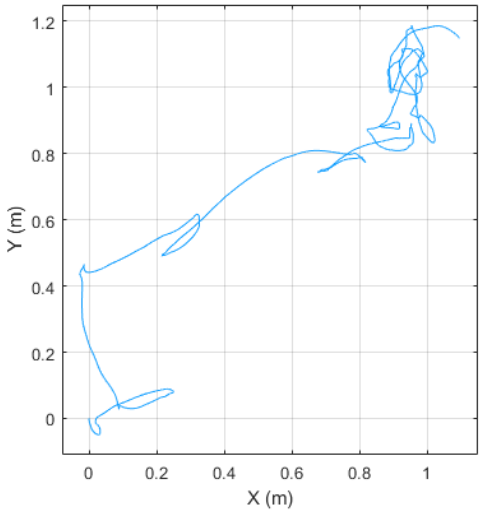
Figure 17: *Simulation First Scenario: Stabilization test with higher communication delay*

Table 4: *Circular trajectory test error*

Circular tests	**MSE** (m^2)
Case A	0.188
Case B	0.167
Case C	0.139

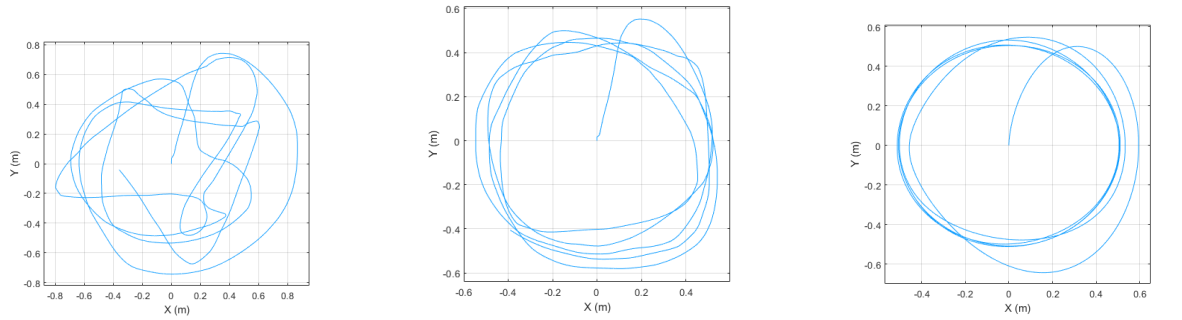


(a) *Position and velocity with position control and step setpoints*



(b) *Movement on XY plane with position control and step setpoints*

Figure 18: *Simulation Second Scenario: Flying to set position and stop*

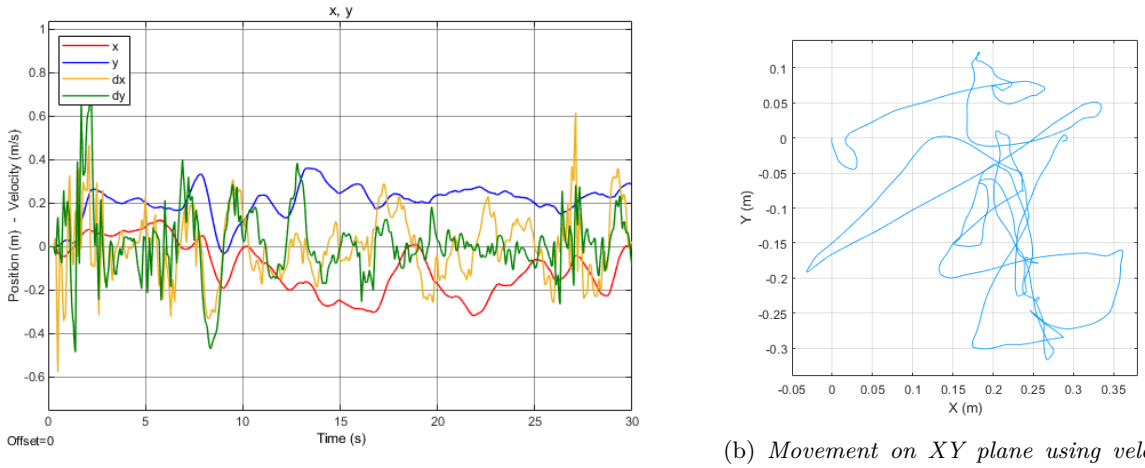


(a) *Case A: Communication delay of* 0.11 *s and noise*

(b) *Case B: Communication delay of* 0.01 *s and noise*

(c) *Case C: Communication delay of* 0.11 *s and no noise*

Figure 19: *Simulation Third Scenario: Circular trajectory*



(a) *Position and velocity using velocity control*

(b) *Movement on XY plane using velocity control*

Figure 20: *Simulation Velocity control: Stabilization test*

8 Discussions

All the ideas that have been built during this work, together with opinions after having experience with this specific hardware, discussions about the results obtained above and the ethical and social effects of this work will be commented on in this section.

8.1 Discussions on Hardware

The Crazyflie 2.1 platform, despite being an excellent option for research and development, has some drawbacks. First, the battery has a brief flying time, draining the whole charge in around 5 minutes while using the AI-deck. This is a common and understandable issue in this kind of micro quadcopter, but the real problem with it is that the thrust command does not completely adapt to the battery level, getting different thrust outputs with the same input and different battery levels. This makes that tuning a not-super-robust controller takes too much time since the battery level decays continuously.

The AI-deck is a great addition to the drone that allows real-time machine vision applications and sends the images recorded. But making it start working was a bit more complex than expected since the GAP8 processor must be flashed to start using it. The lack of experience flashing these products, the lack of the materials to do it and the fact that it was not completely clear if it was necessary or not to flash it, delayed the process of making it run the first time. Once some guidance was obtained from Bitcraze and the flashing tools (ARM-USB-TINY-H and ARM-JTAG-20-10) were acquired, booting the deck was fast and easy. In addition, the communication delay in sending the images through Wi-Fi has been a great limitation in the performance of the system developed in this work.

About the camera in the AI-deck, the quality and resolution are the ones expected for a camera of that size. But this camera seems too sensitive to light, making any spot where some direct light is received appears like a completely white region in the image. It is recommended to use this device indoors with a controlled light source.

8.2 Discussions on Methods

The method used to calculate the distance to the wall allows to estimate of the distance to the reference wall fast without the necessity of additional sensors. The main withdraw of this method is that even improving the image quality, the object reference can not be far away.

For the optical flow method, the algorithm used to find interesting points in the image is fast but quite limited in performance. It is hard to find points for this algorithm in some detailed environments, and many points will be detected in regions that became completely white due to the effect of direct light or reflections. All of this provides as a result clustered detected points that do not provide general information about the image. An alternative to this method could be the SURF algorithm since some projects have presented promising results using this algorithm [6] [7]. The main advantage of this method is that it is much more general than the Strip method that was finally implemented, even though the optical flow method is more sensitive to the background texture and light conditions.

As has already been highlighted, the Strip method performs well, but its practical use in real scenarios is limited. The environment needs to have some vertical and high-contrast structure that needs to be in front of the camera to make this method work properly, and in real cases, it is not possible to depend on these references. This method was meant to give the drone the information required to perform the sensor fusion, but it was never meant to be the final solution to this problem. Due to the time limitation in developing this project, no other method could be successfully developed.

About the sensor fusion, Kalman Filter has shown good performance for this case, allowing to estimate the position and velocity of the drone by using the position, velocity and angle low-frequency observation and an angle rate high-frequency input. The computation speed to perform the filter is high enough to do all the image computation and state estimation faster than the communication rate.

The control system, although simple, it is effective, performing a robust control even with noisy position and velocity estimations, and noisy and biased gyro readings. Although the communication and computation delay can be a problem, it can even lead to making the system unstable. To reduce the effect of the delay, other more advanced control systems could be implemented instead of a PID cascade control loop.

Another way of reducing the delay effect would be to perform the computations onboard. This would remove the communication delay but could increase the computation time since the quadcopter has much less computational power than a computer.

8.3 Discussions on Results

By studying the tests performed on the real quadcopter, that has a communication delay in the camera of 0.35 s, Fig. 15a, and the simulation with high communication delay, Fig. 16a, it becomes clear that the probable reason that makes the system unstable despite all the tuning tests is the high communication delay. In addition, Fig. 16 shows how the system can still keep the drone stable despite the drift introduced by the error in the gyro measurement. This system is stable in time and allows a delay in the position estimation equal to the measured time computation of the method developed.

Further experiments show how the error in the estimation and the delay in the communication affect the ability of the system to follow a track, Fig. 19. The results show how the system is able to follow a circular track when the communication delay is 0.11 s even when the position estimation is noisy. An improvement in the steady-state error can be achieved by reducing the error in the position estimation, while a faster response can be obtained reducing the communication delay.

The velocity control, Fig. 20, while being better than attitude control alone in eliminating drift, does not provide a significant advantage over position control, which completely eliminates drift.

8.4 Discussions on Social and Ethical considerations

The development of new technologies requires careful consideration of their social impacts, as well as their ethical implications. This project has the potential to impact society in various ways, and it is essential to analyze its effects and consider how to address any ethical concerns.

One of the most significant social impacts of the project is the potential for the technology to be used for surveillance purposes. With its camera, the quadcopter can monitor people's movements, raising concerns about privacy and surveillance. The ethical implications of using such technology must be considered, and measures must be taken to protect individuals' privacy. The images captured by the drone can be processed in real-time, so they are not saved. In addition, the data extracted from the pictures are exclusively about the drone's relative position, so no privacy is violated.

Another social impact of the project is its potential impact on the job market. The development of drones and related technologies could lead to the automation of certain jobs, particularly the jobs that require 4-D tasks (Dull, Dirty, Dangerous and Dear) [60]. This technology could be used for instance in the delivery and transportation sectors. While this could increase efficiency since the drone can follow more straightforward paths than a truck that needs to follow the road, it could also lead to job losses and unemployment. It is important to consider the potential impact on the workforce and take measures to mitigate any negative effects. A possible point of view on this aspect is that even if this device can risk some jobs like delivery men or women, it will create new positions to supervise, repair and design these drones. On the other hand, as there is a shortage of people in the current job market, the increased automation can be beneficial as it frees people to work in areas that lack people.

In addition to social impacts, the ethical effects of the quadcopter project are analyzed. Roboethics, an applied ethics that aims to develop scientific, cultural, or technical tools that can be shared by different social and technological groups and beliefs [61], provides a valuable framework for addressing ethical concerns related to the project. This project raises issues related to privacy, transparency, and

accountability, among others. It is important to ensure that the technology is developed and used in a way that aligns with ethical principles and values.

The IEEE Code of Ethics [62], a set of principles and guidelines established by the Institute of Electrical and Electronics Engineers (IEEE) to promote ethical conduct in the engineering profession, provides additional guidance on ethical issues related to technological developments. The code emphasizes the importance of prioritizing the safety and well-being of society and being transparent about any potential risks associated with technology development.

In conclusion, the social effects and impacts of the project are significant, and the ethical implications should be carefully considered in the development of a project like this. The principles of Roboethics and the IEEE Code of Ethics provide valuable frameworks for addressing ethical concerns related to the project. By considering the social and ethical implications of the project, we can ensure that the technology benefits society while minimizing potential negative impacts.

9 Conclusions

This work presented the implementation and evaluation of a stabilizing method for a quadcopter to eliminate the drift due to IMU measurement error. The quadcopter used was a Crazyflie 2.1 with an AI deck, and the sensor fusion algorithm combined camera information and IMU data to estimate the quadcopter's position and velocity. The control used was a PID cascade control to control the position, velocity, attitude, and attitude rate, and the Kalman Filter estimated the position and velocity.

The results showed that the Kalman filter worked well for this situation, and the quadcopter was successfully stabilized in a simulation. However, it was not possible to implement the system on the real drone due to the delay in the camera communication. The feature extraction was also found to be sensitive to background texture, light conditions, and the feature extraction algorithm used. The feature extraction algorithm used at the end, the strip method, worked well but is not useful for a real case.

Despite the inability to implement the system on an actual drone, this study has important implications for future research in the field of drone stabilization. The use of a sensor fusion algorithm like the Kalman Filter can improve the accuracy of drone positioning compared to using only the data from IMU sensor and can help stabilize the drone even in the presence of external disturbances. Moreover, using a PID cascade control can improve the drone's control performance, denying the drift effect and enabling it to follow a desired trajectory.

In conclusion, this study demonstrated the potential of sensor fusion algorithms like the Kalman Filter and Machine Vision methods for stabilizing quadcopters in the air. While the system was not successfully implemented on a real drone, the results obtained in the simulation showed promising results. Further research is needed to overcome the limitations encountered in this study and to fully exploit the potential of sensor fusion algorithms and Machine Vision for drone stabilization.

10 Future Work

Future work could focus on improving the feature extraction algorithm used to make it more robust to changing environmental conditions and more useful for real cases. Some possible solutions could be SURF algorithm or Machine Learning methods. Additionally, investigations could be carried out to find alternative camera systems with better image quality or communication protocols that can reduce the delay in-camera communication and allow for the implementation of the design on a real drone.

Other ways to implement this system could be using onboard computation to avoid communication delay, or exploring better control systems to mitigate the effect of the delay.

References

[1] G. N. Muchiri and S. Kimathi. A review of applications and potential applications of UAV. In *Proceedings of the Sustainable Research and Innovation Conference*, pages 280–283, 2022.

[2] Mostafa Hassanalian and Abdessattar Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99–131, 2017.

[3] Matúš Tkáč and Peter Mésároš. Utilizing drone technology in the civil engineering. *Selected Scientific Papers-Journal of Civil Engineering*, 14(1):27–37, 2019.

[4] Bas Vergouw, Huub Nagel, Geert Bondt, and Bart Custers. Drone technology: Types, payloads, applications, frequency spectrum issues and future developments. *The future of drone use: Opportunities and threats from ethical and legal perspectives*, pages 21–45, 2016.

[5] United Nations. Sustainable development goals. `https://www.un.org/sustainabledevelopment/blog/2015/12/sustainable-development-goals-kick-off-with-start-of-new-year/#`. Accessed: 2023-03-21.

[6] Tomoyuki Mori and Sebastian Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for Micro Unmanned Aerial Vehicles. In *2013 IEEE International Conference on Robotics and Automation*, pages 1750–1757. IEEE, 2013.

[7] Ekkaphon Mingkhwan and Weerawat Khawsuk. Digital image stabilization technique for fixed camera on small size drone. In *2017 Third Asian Conference on Defence Technology (ACDT)*, pages 12–19. IEEE, 2017.

[8] Ellen Lindgren. Target recognition and following in small scale UAVs. *Thesis, Uppsala Universitetet*, 2022.

[9] Marcus Greiff. Modelling and control of the Crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation. 2017, Thesis, Department of Automatic Control, Lund University. `https://lup.lub.lu.se/student-papers/search/publication/8905295`.

[10] Hann Woei Ho, Guido C. H. E. de Croon, and Qiping Chu. Distance and velocity estimation using optical flow from a monocular camera. *International Journal of Micro Air Vehicles*, 9(3):198–208, 2017.

[11] Josefine Möllerström and Max Nyberg Carlsson. Emulation of the Crazyflie 2.1 hardware for embedded control system testing. 2021,LUP Student Papers, Department of Automatic Control, Lund University.`https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9052405&fileOId=9052409`.

[12] Bitcraze AB. Crazyflie 2.1|Bitcraze. `https://www.bitcraze.io/products/crazyflie-2-1/`. Accessed: 2023-02-27.

[13] Bitcraze AB. Crazyradio PA|Bitcraze. `https://www.bitcraze.io/products/crazyradio-pa/`. Accessed: 2023-02-27.

[14] Bitcraze AB. Getting started with the Crazyflie 2.X.|Bitcraze. `https://www.bitcraze.io/documentation/tutorials/getting-started-with-crazyflie-2-x/`. Accessed: 2023-02-28.

[15] Bosch Sensortec. *BMI088 6-axis Motion Tracking for High-performance Applications*, 2018.

[16] Bosch Sensortec. *BMP388 Digital pressure sensor*, 2018.

[17] STMicroelectronics. *ARM Cortex-M4 32b MCU+FPU*, 2015.

[18] Nordic Semiconductor. *Multi-protocol Bluetooth Low Energy and 2.4GHz proprietary system-on-chip*, 2012.

[19] Bitcraze AB. State estimation|Bitcraze. `https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/state_estimators/`. Accessed: 2023-02-28.

[20] Michael A. Johnson and Mohammad H. Moradi. *PID Control*. Springer, 2005.

[21] Bitcraze AB. Controllers in the Crazyflie|Bitcraze. `https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/controllers/`. Accessed: 2023-02-28.

[22] Himax Imaging, Ltd. *Compact Camera Module*, 2019.

[23] Bitcraze AB. AI deck 1.1|Bitcraze. `https://www.bitcraze.io/products/ai-deck/`. Accessed: 2023-02-28.

[24] Bitcraze AB. The Crazyflie Python API explanation|Bitcraze. `https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/user-guides/python_api/`. Accessed: 2023-05-17.

[25] Norhafizan Ahmad, Raja Ariffin Raja Ghazilla, Nazirah M. Khairi, and Vijayabaskar Kasi. Reviews on various inertial measurement unit (IMU) sensor applications. *International Journal of Signal Processing Systems*, 1(2):256–262, 2013.

[26] Erik Grahn. Evaluation of mems accelerometer and gyroscope for orientation tracking nutrunner functionality. *Thesis, KTH Royal Institute of Technology*, 2017.

[27] Yaping Zang, Fengjiao Zhang, Chong-an Di, and Daoben Zhu. Advances of flexible pressure sensors toward Artificial Intelligence and health care applications. *Materials Horizons*, 2(2):140–156, 2015.

[28] William P. Eaton and James H. Smith. Micromachined pressure sensors: review and recent developments. *Smart Materials and Structures*, 6(5):530, 1997.

[29] S. Santosh Kumar and Amit Tanwar. Development of a MEMS-based barometric pressure sensor for micro air vehicle (MAV) altitude measurement. *Microsystem Technologies*, 26(3):901–912, 2020.

[30] H. C. T. E. Fernando, A. T. A. De Silva, M. D. C. De Zoysa, K. A. D. C. Dilshan, and S. R. Munasinghe. Modelling, simulation and implementation of a quadrotor UAV. In *8th International Conference on Industrial and Information Systems*, pages 207–212. IEEE, 2013.

[31] Randal W. Beard. Quadrotor dynamics and control. *Brigham Young University*, 19(3):46–56, 2008.

[32] Teppo Luukkonen. Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, 22(22), 2011.

[33] Jurek Z. Sasiadek. Sensor fusion. *Annual Reviews in Control*, 26(2):203–228, 2002.

[34] Wilfried Elmenreich. An introduction to sensor fusion. *Vienna University of Technology, Austria*, 502:1–28, 2002.

[35] Walter T. Higgins. A comparison of complementary and Kalman Filtering. *IEEE Transactions on Aerospace and Electronic Systems*, (3):321–325, 1975.

[36] Jin Wu, Zebo Zhou, Jingjun Chen, Hassen Fourati, and Rui Li. Fast complementary filter for attitude estimation using low-cost MARG sensors. *IEEE Sensors Journal*, 16(18):6997–7007, 2016.

[37] Greg Welch, Gary Bishop, et al. An introduction to the Kalman Filter. 1995.

[38] Raquel R. Pinho, R. S. Tavares João Manuel, and Miguel V. Correia. Efficient approximation of the Mahalanobis distance for tracking with the Kalman Filter. In *Computational Modelling of Objects Represented in Images*, pages 349–354. CRC Press, 2018.

[39] Gary Bishop and Greg Welch. An introduction to the Kalman Filter. *Proc of SIGGRAPH, Course*, 8(27599-23175):41, 2001.

[40] Maria Isabel Ribeiro. Kalman and Extended Kalman Filters: Concept, derivation and properties. *Institute for Systems and Robotics*, 43(46):3736–3741, 2004.

[41] Salvatore Carotenuto, Luigi Iannelli, Sabato Manfredi, and Stefania Santini. Sensor fusion by using a sliding observer for an underwater breathing system. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 7662–7667. IEEE, 2005.

[42] Suneel Kumar Kommuri, Jagat Jyoti Rath, and Kalyana Chakravarthy Veluvolu. Sliding-mode-based observer–controller structure for fault-resilient control in DC servomotors. *IEEE Transactions on Industrial Electronics*, 65(1):918–929, 2017.

[43] J. P. Barbot, M. Djemai, and T. Boukhobza. Sliding mode observers. *Sliding mode control in engineering*, 11:33, 2002.

[44] John Illingworth and Josef Kittler. A survey of the Hough Transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988.

[45] OpenCV. Hough Line Transform. `https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html`. Accessed: 2023-04-20.

[46] Deqing Sun, Stefan Roth, John P. Lewis, and Michael J. Black. Learning optical flow. In *Computer Vision–ECCV 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III 10*, pages 83–97. Springer, 2008.

[47] Kiran Kale, Sushant Pawar, and Pravin Dhulekar. Moving object tracking using optical flow and motion vector estimation. In 4^{th} *International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, pages 1–6. IEEE, 2015.

[48] John L. Barron, David J. Fleet, and Steven S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994.

[49] James J. Little and Alessandro Verri. Analysis of differential and matching methods for optical flow. *Massachusetts Institute of Technology*, 1988.

[50] Nusrat Sharmin and Remus Brad. Optimal filter estimation for Lucas-Kanade optical flow. *Sensors*, 12(9):12694–12709, 2012.

[51] Dimitris Bouris, Antonis Nikitakis, and Ioannis Papaefstathiou. Fast and efficient FPGA-based feature detection employing the SURF algorithm. In *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 3–10. IEEE, 2010.

[52] Scott Leorna, Todd Brinkman, and Timothy Fullman. Estimating animal size or distance in camera trap images: Photogrammetry using the pinhole camera model. *Methods in Ecology and Evolution*, 13(8):1707–1718, 2022.

[53] Juho Kannala and Sami S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006.

[54] OpenCV. Feature detection. `https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga1d6bb77486c8f92d79c8793ad995d541`. Accessed: 2023-05-09.

[55] Parrot. Mambo range documentation. `https://www.parrot.com/en/support/documentation/mambo-range`. Accessed: 2023-05-17.

[56] Sky Viper. Dash Nano Drone - Sky Viper. `https://sky-viper.com/dash/`. Accessed: 2023-05-17.

[57] Drona Aviation Pvt. Ltd. Pluto 1.2 DIY Nano Drone Kit. `https://www.dronaaviation.com/product/pluto-1-2/`. Accessed: 2023-05-17.

[58] Peter I. Corke and Oussama Khatib. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, volume 73. Springer, 2011.

[59] MathWorks. Band-Limited White Noise. `https://se.mathworks.com/help/simulink/slref/bandlimitedwhitenoise.html`. Accessed: 2023-05-17.

[60] Bernard Marr. The 4 Ds Of Robotization: Dull, Dirty, Dangerous And Dear. `https://www.forbes.com/sites/bernardmarr/2017/10/16/the-4-ds-of-robotization-dull-dirty-dangerous-and-dear/`. Accessed: 2023-06-21.

[61] Peter Kopacek. Ethical and social aspects of robots. *IFAC Proceedings Volumes*, 47(3):11425–11430, 2014.

[62] IEEE. IEEE Code of Ethics. `https://www.ieee.org/about/corporate/governance/p7-8.html`. Accessed: 2023-05-12.