DEVELOPMENT OF A COOPERATIVE UNMANNED AERIAL VEHICLE FOR AGRICULTURAL OPERATIONS

by

Efe Oğuzhan Karcı

B.S., Mechanical Engineering, Marmara University, 2020

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Mechanical Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor, Assist. Prof. Sinan Öncü, for his invaluable guidance and support throughout my research and education. His insightful feedback and encouragement were essential in helping me to develop and refine my ideas. I would also like to thank Assoc. Prof. Evren Samur and Assoc. Prof. Volkan Sezer for their participation in my jury.

I am grateful to my colleagues in the Smart and Autonomous Mobility Laboratory, Mustafa Kangal, Yağız Koç, and Ender Yağcılar, for their valuable support, helpful comments, and suggestions during my thesis period. I would like to thank entire staff of Department of Mechanical Engineering in Boğaziçi University.

This work was supported by the UTOPIA (Automated Open Precision Farming Platform) Project, co-funded by TUBITAK 1071 (Project No: 120N785) and the European Union's Horizon 2020 research and innovation programme under grant agreement No 862665 ERA-NET ICT-AGRI-FOOD. This work was also supported by Boğaziçi University Research Fund (Grant Number 16883) through the MARE (Multi-Agent Robot Environment) Project.

Finally, I am indebted to my family, and Ceyda who provided unwavering support and encouragement throughout the entire process. Without their love and support, this study would not have been possible.

ABSTRACT

DEVELOPMENT OF A COOPERATIVE UNMANNED AERIAL VEHICLE FOR AGRICULTURAL OPERATIONS

This study presented the design of a quadcopter unmanned aerial vehicle and its cooperative behavior with another ground vehicle. The first step was to determine the hardware and software requirements for the quadcopter. The quadcopter was modelled using a computer-aided design program, taking into account its software and hardware requirements to provide input for dynamic modelling. This three-dimensional model provides the information needed by the software, such as the quadcopter's center of gravity and moment of inertia. ZED mini camera, global positioning system, optical distance sensor and inertial measurement unit were used as sensors in the quadcopter. The velocity and position information obtained from the sensors was processed and utilized by the robot's operating system. The Extended Kalman Filter used to process sensor data was compared theoretically with the Kalman Filter and the reasons for its preference were explained. During the studies conducted via ROS, the MAVLink protocol was used to communicate with the Pixhawk autopilot, which controls the unmanned aerial vehicle. The selected filtering algorithms were tested in the simulation environment created on Gazebo and RViz. After the simulation, the testing phase began. The velocity and position data of the quadcopter were processed and sent to the site manager in the test area. The information was then used to simulate various scenarios.

ÖZET

TARIMSAL OPERASYONLAR İÇİN İŞ BİRLİĞİNE DAYALI BİR İNSANSIZ HAVA ARACININ GELİŞTİRİLMESİ

Bu çalışmada, dört pervaneli bir insansız hava aracı'nın tasarımının yanı sıra başka bir kara aracı kullanılarak iş birliğine dayalı hareketi gerçekleştirilmiştir. Öncelikle dört pervanelinin donanımsal ve yazılımsal gereklilikleri belirlenmiştir. Dört pervaneli hava aracı, dinamik modellemeye girdi sağlamak amacıyla yazılımsal ve donanımsal gereklilikleri göz önünde bulundurularak bilgisayar destekli tasarım programı yardımı ile çizilmiştir. Bu üç boyutlu model ile, dört pervaneli hava aracının ağırlık merkezi ve eylemsizlik momenti gibi yazılım tarafında ihtiyaç duyulan bilgileri elde edilmiştir. Dört pervanelide sensör olarak ZED mini kamera, global konumlama sistemi, optik mesafe ölçüm sensörü ve atalet ölçüm birimi kullanılmıştır. Sensörlerden çekilen hız ve konum bilgileri, robot işletim sistemi üzerinden işlenerek kullanılır hale getirilmiştir. Sensör verilerini işlemek için kullanılan Genişletilmiş Kalman Filtresinin teorik olarak Kalman Filtresi ile karşılaştırılması yapılmış ve tercih edilme nedenleri açıklanmıştır. ROS üzerinden yapılan çalışmalar esnasında MAVLink protokolü kullanılarak, insansız hava aracını kontrol eden Pixhawk otopilot ile haberleşme sağlanmıştır. Seçilen filtreleme algoritmaları Gazebo ve RViz üzerinde oluşturulmuş simülasyon ortamında test edilmiştir. Simülasyonun ardından, test aşamasına geçilmiştir. Hava aracının işlenen hız ve konum bilgileri test alanında bulunan üçüncü bir bilgisayara gönderilmiş ve bu bilgilerin işlenmesi sonucu farklı senaryolar gerçekleştirilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

f	Process model
h	Measurement model
K_k	Kalman gain at the time step k
$\hat{P}_{k-1}^{+}$	Posteriori covariance at the time step k-1
$\hat{P}_{k}^{+}$	Posteriori covariance at the time step k
$\hat{P}_{k}^{-}$	Priori covariance at the time step k
R_k	Measurement covariance at the time step k
S_k	Innovation covariance at the time step k
u_k	Control input vector at the time step k
v_k	Measurement noise vector at the time step k
w_k	Process model noise vector at the time step k
x_k	State vector at the time step k
$\hat{x}_{k-1}^{+}$	Posteriori mean at the time step k-1
$\hat{x}_{k}^{+}$	Posteriori mean at the time step k
$\hat{x}_{k}^{-}$	Priori mean at the time step k
$\tilde{y}_k$	Innovation with linear measurement model at the time step k
z_k	Measurement at the time step k
υ_k	Innovation with non-linear measurement model at the time step k

LIST OF ACRONYMS / ABBREVIATIONS

CAD	Computer Aided Design
CAE	Computer Aided Engineering
CCW	Counterclockwise
CPU	Central Processing Unit
CW	Clockwise
DC	Direct current
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
GNSS	Global Navigation Satellite Systems
GPIO	General Purpose Input/Output
GPS	Global Positioning System
HLC	High Level Controller
ICSP	In-Circuit Serial Programming
I^2C	Inter-Integrated Circuit
I^2S	Inter-IC Sound
IMU	Inertial Measurement Unit
KF	Kalman Filter
LiPo	Lithium-ion polymer
LLC	Low Level Controller
MATLAB	Matrix Laboratory
MAVLink	Micro Air Vehicle Link
PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation
RC	Radio Controller
ROS	Robot Operating System
RTK	Real Time Kinematic
RViz	Robot Operating System Visualization
SCA	Serial Clock

SDA	Serial Data
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
UGV	Unmanned Ground Vehicle
URDF	Unified Robot Description Format
USB	Universal Serial Bus

1. INTRODUCTION

In the dynamic landscape of technological innovation, unmanned aerial vehicles, popularly known as drones, have emerged as a groundbreaking frontier with the potential to reshape numerous industries. Among the various designs of UAVs, quadcopters have stood out for their unique capabilities and versatility. Characterized by their four-rotor design, these compact flying machines blend intricate aerodynamic principles with advanced computer control systems, demonstrating profound possibilities for both civil and commercial applications. Figure 1.1 presents the quadcopter system that has been devised specifically for this thesis.

Figure 1.1. Quadcopter system devised for this thesis.

1.1. Motivation and Background

Quadcopters, as a subset of the broader UAV category, have been chosen as the subject of focus for this thesis due to their distinct mechanical simplicity and maneuverability, compared to other multi-rotor or fixed-wing UAVs. Despite their seeming simplicity, the underpinning technology that facilitates their flight and control mechanisms is a fusion of diverse areas such as robotics, computer science, electronics, and aerodynamics. This

multidisciplinary aspect of quadcopters underlines their adaptability and expansive use-cases.

A quadcopter's control system is generally partitioned into a high-level controller (HLC) and a low-level controller (LLC). The high-level controller, often also referred to as the “flight planner”, deals with more strategic aspects of the flight, such as defining waypoints, managing mission trajectory, and dealing with global positioning. It is essentially concerned with where the drone needs to go and plots a course for reaching the destination based on pre-defined algorithms and sensor inputs. To accomplish this goal, tracking_pid package has been selected to serve as the high-level controller.

On the other hand, the low-level controller, also known as the “attitude controller,” manages the immediate, tactical aspects of the flight. It ensures stability and controls the drone's attitude (its orientation in space) by manipulating the speeds of the individual rotors. This fine-tuning allows the drone to maintain its balance, respond to immediate environmental influences like wind, and make the precise movements necessary to follow the path set out by the HLC. The commercial flight controller, PX4, has been selected to function as the low-level controller. This thesis not only seeks to delve into the fundamental control systems of quadcopters but also aims to explore and enhance our understanding of cooperative behavior. Cooperative behavior between Unmanned Ground Vehicles (UGVs) and UAVs is a critical aspect in the realm of autonomous systems, yielding significant advancements in terms of operational efficiency and situational awareness. This collaborative approach allows these unmanned vehicles to share sensor data and coordinate their movements. Furthermore, cooperative behavior can provide new possibilities for intricate tasks and missions. For example, a team of UAVs and UGVs could collaboratively perform tasks such as search and rescue, environmental monitoring, or infrastructure inspection more effectively than either could manage alone.

In Figure 1.2, there is a groundbreaking smart agriculture system, UTOPIA, which stands at the forefront of innovation by integrating advanced robotics, cloud computing, and data analytics to revolutionize traditional farming practices. The diagram illustrates the integrated workflow of a smart agriculture system, highlighting the core functions of the Site Manager in managing and relaying information between the user, cloud services, and field

operations. At the heart of the system, the Site Manager is responsible for the acquisition, fusion, and processing of sensor data, which is tagged with locational and environmental information to support agricultural decisions. Each component of the system, from the user interface to the heterogeneous mobile robots, is interconnected through the ROS (Robot Operating System), facilitating a modular yet cohesive agricultural management solution. The system's architecture enables seamless communication between the cloud service, labeled as UTOPIA Cloud Service, and on-ground components, ensuring real-time data exchange and operational coordination. The UTOPIA Cloud Service is tasked with key functions such as field tracking, plant recognition, harvest prediction, and providing climate and weather condition information, crucial for precision agriculture.

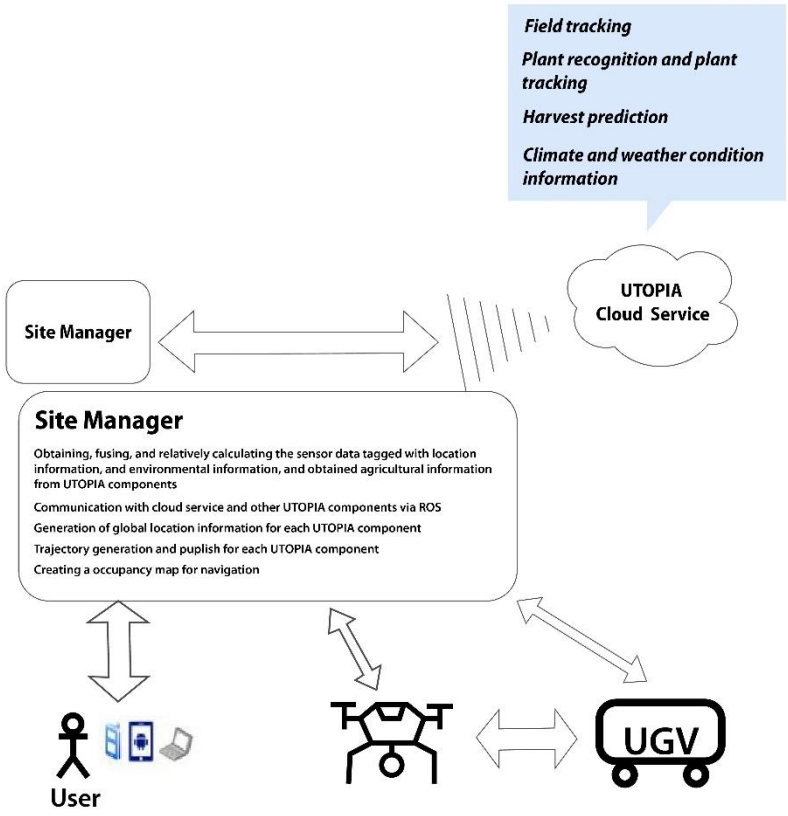

Figure 1.2. The UTOPIA Smart Agriculture Framework.

In Figure 1.3, there is an operational flow and control systems for both aerial and ground-based robotic components within the UTOPIA smart agriculture framework. Both UAVs and UGVs are equipped with an array of sensors for collecting environmental and agricultural data, which they send to the Site Manager after tagging, showcasing an efficient

data flow that enhances the accuracy of farm monitoring and intervention. Central to both UAV and UGV operations is the Low-Level Controller, with Pixhawk and Roboteq controllers respectively, which act as the brains of the operation by executing commands and managing real-time adjustments during flight or navigation. There is a synergy between the user's strategic input and the technological dexterity of UAV and UGV to achieve a high degree of automation in smart farming practices.

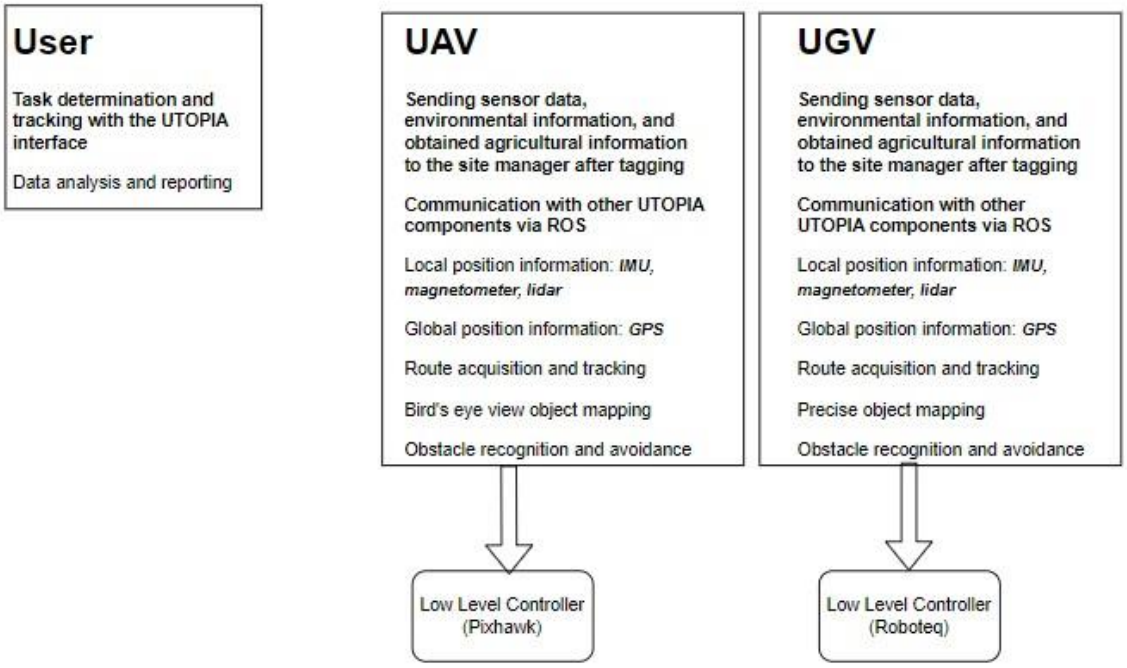

Figure 1.3. Integrating user interface, UAV, and UGV for enhanced precision agriculture.

1.2. Problem Statement and Objectives

UAVs, specifically quadcopters, have seen extensive applications in diverse fields due to their unique maneuverability and versatility. However, the full potential of quadcopters is yet to be realized, especially regarding the robustness of their control systems and the efficacy of their localization capabilities. The development and implementation of a system that enables this cooperation effectively and robustly are imperative to advancing the field of autonomous systems. The goals of the dissertation are outlined as follows:

- To construct a fully functional quadcopter framework incorporating the necessary hardware elements.

- To employ PX4 as the low-level controller. PX4, due to its open-source nature, extensive community support, and compatibility with various hardware, offers a flexible and reliable choice for the low-level controller.
- To implement Sensor Fusion using an Extended Kalman Filter. The EKF is a well-established method for sensor fusion, which will be employed to process data from various onboard sensors. This implementation will aim to enhance the quadcopter's state estimation accuracy, providing a more reliable foundation for the control actions and improving the overall system's robustness.
- To establish cooperative behavior between the UAV and a UGV. By sharing sensor data and other relevant information between the UAV and a UGV, cooperation will be realized. This strategy aims for leading to more efficient and resilient operation.
- To simulate scenarios using Gazebo, Robot Operating System Visualization (RViz), and Matrix Laboratory (MATLAB). The proposed sensor fusion method, and cooperative behavior will be tested and validated through simulations using Gazebo and RViz, aided by MATLAB.
- To demonstrate the validity of simulations experimentally.

1.3. Outline of the Thesis

The summary of the thesis structure is provided below:

- Chapter 2 provides a review of the literature on the application areas of UAVs, previous studies on quadcopter modeling and control, including the aspects of cooperative behavior and sensor fusion.
- Chapter 3 is dedicated to an exploration of the quadcopter's hardware design, discussing the critical components and architecture that contribute to its functionality and performance.
- Chapter 4 is focused on the mathematical generation of quadcopter trajectories, providing an understanding of how these flight paths are formulated and implemented.
- Chapter 5 is centered around the concept of sensor fusion in the quadcopter, explaining how various sensory data inputs are integrated for enhanced navigational accuracy and performance. Furthermore, it provides a mathematical exposition of the Extended Kalman Filter as part of the discussion.

- Chapter 6 delves into the concept of cooperation between the quadcopter and UGV, exploring how these two entities collaborate to enhance situational awareness and navigation.
- Chapter 7 focuses on the simulation setup and presents the results obtained, providing a comprehensive analysis of the quadcopter's performance and behavior under various simulated scenarios.
- Chapter 8 encompasses the test results and in-depth discussions, analyzing the outcomes of practical experiments conducted to evaluate the quadcopter's performance under sensor fusion.
- Chapter 9 concludes the thesis by summarizing the findings and implications, while also highlighting potential directions for future research and development.

2. LITERATURE SURVEY

Within the realm of agriculture, the implementation of innovative, regulated solutions such as UAVs is gaining traction [1]. UAVs, often referred to as drones, are aircraft systems that operate without an onboard human pilot [2]. UAVs have the capacity to be outfitted with a diverse array of sensors that can significantly contribute to various applications [3]. The trend towards smaller electronics, computing systems, and sensors has paved the way for novel possibilities in the realm of remote sensing applications [4].

A set of motion primitives was implemented, using the low-level controllers provided by the PX4 firmware, which are employed for controlling the drone in autopilot mode. Upon activation, motion primitives are transformed into a sequence of MAVLink messages - a communication protocol for unmanned vehicles - and transmitted to the drone through the User Datagram Protocol (UDP) [5]. The Pixhawk comes with a gyroscope, accelerometers, a magnetometer, and a barometer, and also supports connectivity with an external GPS module. Its role involves processing sensor readings and signals from the Radio Controller (RC), generating flight data, and managing each motor individually. Additionally, Pixhawk has the capability to interface with other devices via a protocol known as MAVLink, specifically designed for UAV applications [6].

Determining pose is crucial for various navigational operations, encompassing localization, mapping, and control. The method employed primarily hinges on the accessible onboard sensors. In aerial navigation, these must be judiciously selected due to payload constraints. The extensive time required for computation in response to the fast dynamics of aerial vehicles poses challenges for their control [7]. A design incorporating an Extended Kalman Filter is employed for the purpose of sensor fusion. It is demonstrated that the quadcopter can navigate along predetermined routes in unfamiliar environments, utilizing its camera and onboard sensors [8]. The EKF algorithm can be effectively implemented in the Pixhawk, enabling it to accommodate further optimization processes of attitude detection. The EKF algorithm offers notable advantages, such as its straightforward formulation. It features a relatively quick rate of convergence for state estimation. Moreover, once the filter achieves stability, the estimation accuracy remains uninfluenced by the initial

value [9]. EKF operates by consistently refreshing a linear approximation centered around the prior state estimate and by representing state densities through Gaussian distributions [10].

In the majority of autonomous scenarios, the initial operational phase involves target detection via various sensors and the determination of their actual locations, which facilitates the execution of subsequent operations. UAVs and UGVs have the potential to augment the efficiency of surveillance, particularly in aspects of crowd detection and localization [11]. Cooperative localization explores how the positions of robots operating in a group can be determined more precisely through data exchange among themselves. To improve localization in multi-robot teams, positions of robots relative to each other within the team can be utilized instead of markers in the environment. Some insightful approaches have been suggested regarding cooperative localization [12, 13]. In these studies, some robots in the group have been fixed to serve as markers throughout the entire navigation duration and coordinated to provide a localization infrastructure for the other robots. Both of these studies utilize a centralized strategy where the main robot controls the movement task. The main robot observes the other robots and updates its estimates as a result of the cooperative localization process. The primary goal here is to ensure accuracy in these estimates. This way, the main robot can navigate within its environment and map the area. A decentralized approach has also been presented to accomplish cooperative localization [14]. In this study, each robot updates its own result using the estimate step of the distributed Extended Kalman filter in conjunction with data received from proprioceptive sensors. Moreover, as each robot can communicate and exchange information with one another, these filters are used to improve cooperative localization. The information exchange occurs when one robot encounters another in the environment. As robots benefit from each other during the cooperative localization process, their estimates become interdependent. The approach using the Extended Kalman filter produces consistent estimates only when the information used is independent or when cross-correlation information is provided. On the other hand, an approach based on distributed maximum a posteriori estimation has also been employed [15]. This approach has a computational cost of $O(n^2)$ and is resilient to sudden malfunctions, such as the breakdown of some robots. To achieve a fully distributed approach, the method requires simultaneous communication. For this, each robot needs to communicate with others in each iteration. These requirements may hinder the application of this approach

when cooperative localization of groups composed of numerous robots is desired. In some techniques used in the update phase of cooperative localization, robots do not use information related to all other robots in the group. An example of this concept can be the data collection strategy in wireless sensor networks [16]. These techniques do not provide a common estimate and end up producing an approximate solution to the problem. In some studies aiming to produce a common estimate, partial information has been used to cope with time and space complexity [17, 18]. For instance, in one presented study, group movement was facilitated by each robot adjusting its estimate using only a subset of robots [19]. In recent studies, cooperative localization has been carried out using the covariance intersection algorithm. This proposed algorithm consistently combines data in situations where the cross-correlation among estimates is unknown [20]. The strategy behind this algorithm allows for dealing with cross-correlation without the need to record the correlations between the robots' estimates. In situations where each robot only needs to record its estimates and associated uncertainties, it's possible to perform cooperative localization using covariance intersection consistently and with lower complexity. In one study, an approximate algorithm has been presented to carry out cooperative localization dependent on covariance intersection [21]. In this study, a group composed of three robots was utilized, with each robot recording only its own estimate and uncertainty. The results have been compared with an Extended Kalman filter approach using the matrix recorded by each robot. The Extended Kalman filter was used as a benchmark in these comparisons. Although the localization error of the covariance intersection-based approach was larger than that of the Extended Kalman filter-based approach, the significant reduction in complexity during the recording and processing of information justifies its usage.

3. HARDWARE DESIGN

Quadcopters, otherwise known as quadrotors or drones, are characterized by their X-shaped design, with four motors placed symmetrically around the body. This setup provides a perfect blend of stability, maneuverability, and lifting capacity. The heart of quadcopter hardware design revolves around these components: flight controller, motors, propellers, Electronic Speed Controllers (ESCs), batteries, and optionally, payloads such as cameras or sensors.

In the construction of a quadrotor UAV intended for agricultural use, careful consideration has been given to its functionality as well as its design compatibility with the operational process. The body, which is accommodating various devices such as sensors, processors, and controllers, was made from 2 mm thick aluminum and it was shaped with the assistance of laser cutting. Aluminum, being a lightweight metal, has been chosen to prevent undue load on the motors due to weight. Additionally, the property of aluminum to absorb energy robustly is a significant advantage. This characteristic will reduce the potential damage to the technological components on the quadrotor UAV in the event of a collision.

In the fabrication of the UAV's arms, both the need for lightweight construction to prevent undue burden on the body, and the requirement for robustness to support the rotors, were considered. As a result, a composite design was employed. To construct the arms, a carbon fiber tube with an inner diameter of 16 mm and a thickness of 1 mm was fit over an aluminum tube of 1 mm thickness and 14 mm inner diameter, utilizing an interference fit method. This design leveraged the lightness of aluminum while benefiting from the strength of carbon fiber.

The use of a 3D printer was instrumental in the creation of the UAV's landing legs, battery holder, and fasteners. The use of 3D printing enabled the formation of high-complexity fasteners according to design requirements. Furthermore, it facilitated the design of the battery holder and UAV’s landing legs with appropriate dimensions and structures. By using carbon fiber filament, structures were produced that were strong relative to their weight, and uniformly shaped.

While installing electronic speed control devices to drive the rotors of the quadrotor UAV, careful consideration has been given to the rotation direction of the propellers. Electronic speed controllers and their corresponding motors have been numbered. According to this numbering system, motors numbered 1 and 3 have been set to rotate counterclockwise, while motors numbered 2 and 4 are set to rotate in a clockwise direction.

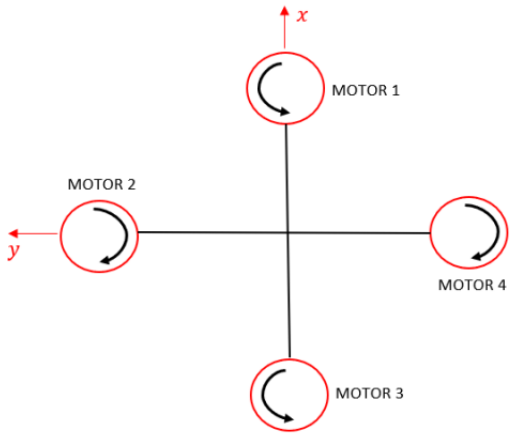

Figure 3.1. Rotation directions of propellers.

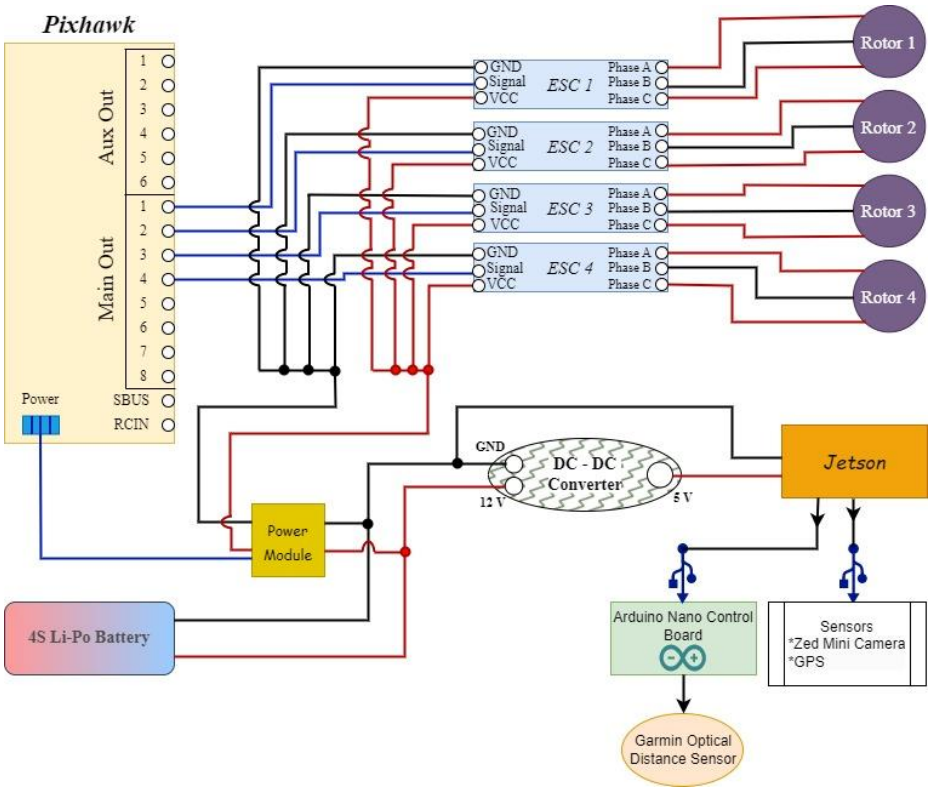

Figure 3.2. Electrical circuit design.

The design of the UAV's electrical schematic, as depicted in Figure 3.2, has been completed. To capture the operating voltage of the Jetson, which is powered by a 4S LiPo battery, a DC-DC converter has been incorporated into the circuit. The Pixhawk, which governs the control of the UAV, and the electronic speed controllers that manage the rotors, also derive their operating voltages from a power module connected to the 4S Li-Po battery.

Table 3.1. List of components.

No.	Components	Amount
1	Adjustable 5V DC-DC Converter	1
2	Garmin Lidar-Lite V3 Optical Distance Sensor	1
3	Arduino Nano Control Board	1
4	Nvidia Jetson Nano	1
5	Telemetry (for ground station and rover)	2
6	Radio Controller Receiver and Transmitter	1
7	Brushless Motor	4
8	Drone Arm	4
9	Motor Holder	4
10	Drone Arm Fastener	8
11	CW Propeller	2
12	CCW Propeller	2
13	Battery Holder	1
14	Camera Holder	1
15	ZED Mini Camera	1
16	Distancer	4
17	Aluminum Plate	5
18	4S Li-Po Battery	1
19	Landing Legs	4
20	Electronic Speed Controller	4
21	Pixhawk	1
22	Buzzer	1
23	Reach M+ RTK-GNSS Module	1

Initially, the motors were attached to the motor holders. This was achieved by passing the motor wires through the motor holders. Next, the landing legs were attached to the motor holders. The drone arms were then pushed into the motor holders. Bolts and nuts were used to prevent the twist of drone arms inside the motor holders. Two aluminum plates were put together, and the drone arm fasteners were attached to them. Then, the battery holder was attached to the assembled aluminum plates. Two more thin plates were put together and attached on top of the drone arm fasteners. Next, distancers were attached to another thin plate. These distancers also were screwed onto the aluminum plates located in the middle of quadcopter frame. Then, the Pixhawk, RC receiver module, buzzer, telemetry, and GPS module were placed to the top aluminum plate with double-sided tape. Electronic Speed Controllers were put on the bottom plate considering the lengths of the cables. The ESC cables were connected to the motor cables with solder. After this, all other sensors are placed onto the quadcopter frame and assembly was finalized.

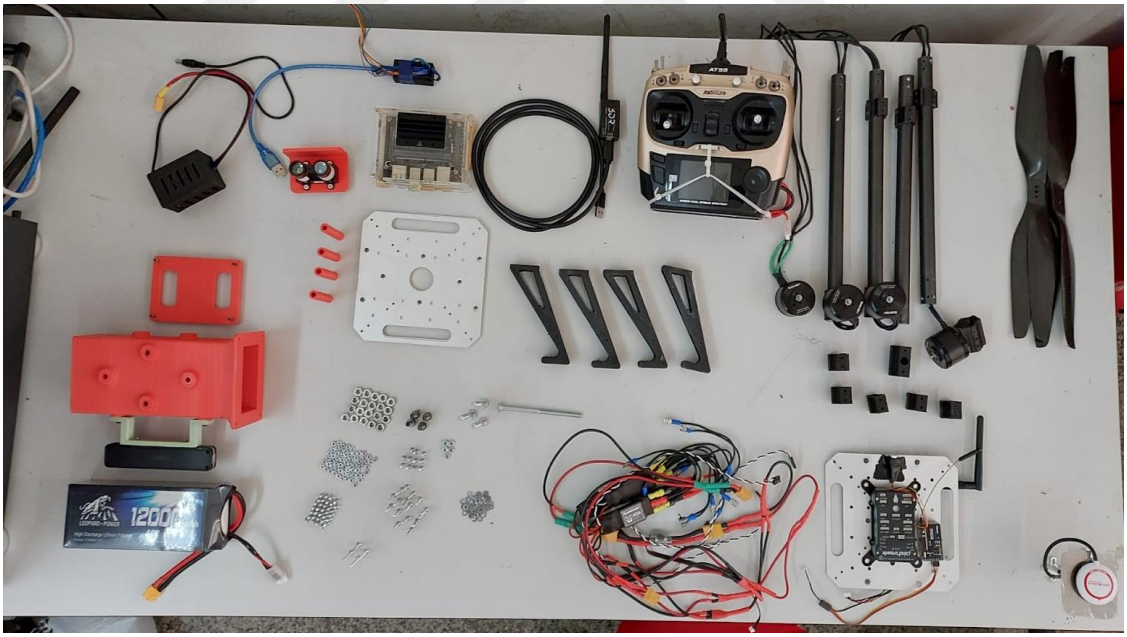

Figure 3.3. Hardware components.

The sensors mounted for use on the UAV include the following:

- Garmin-Lidar Lite V3 Optical Distance Sensor: This sensor emits a light pulse that travels to the target object and returns to the sensor, facilitated by laser technology. The time taken for the light pulse to complete this round-trip journey is measured, and

this information is used to calculate the distance to the object. The Lidar-Lite V3 is capable of measuring distances up to 40 meters (131 feet) with an accuracy of +/- 2.5 centimeters (1 inch).

Figure 3.4. Garmin Lidar-Lite V3 optical distance sensor.

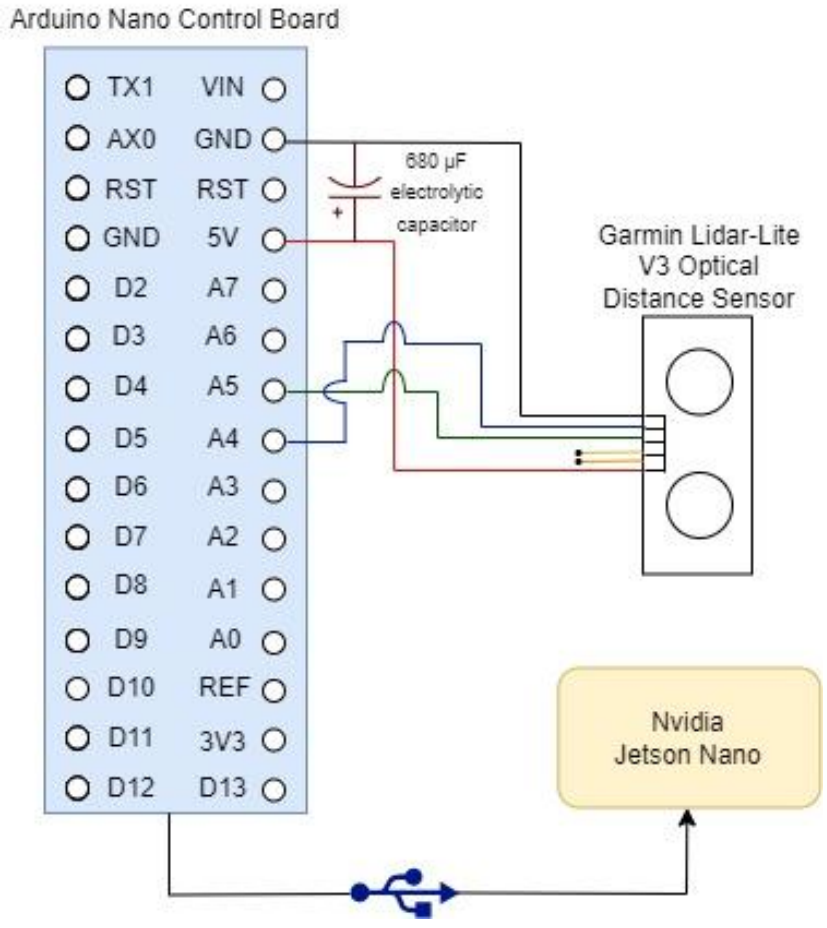

Figure 3.5. Electrical connection schematics of Garmin Lidar-Lite V3.

Table 3.2. Wiring color code.

Wiring Color	Description
Black	Power ground (-) connection
Blue	I2C SDA connection
Green	I2C SCA connection
Yellow	Mode-control connection
Orange	Power enables (internal pull up)
Red	5 Vdc power (+) connection

- ZED Mini Camera: The ZED Mini Camera is a compact stereo camera that utilizes two 2-megapixel sensors to capture 3D images and videos. The camera uses passive stereo imaging to generate a depth map of the area, enabling real-time capture of 3D images and videos. The ZED Mini Camera boasts a field of view of 90 degrees horizontally and 60 degrees vertically. It is equipped with an integrated Inertial Measurement Unit.

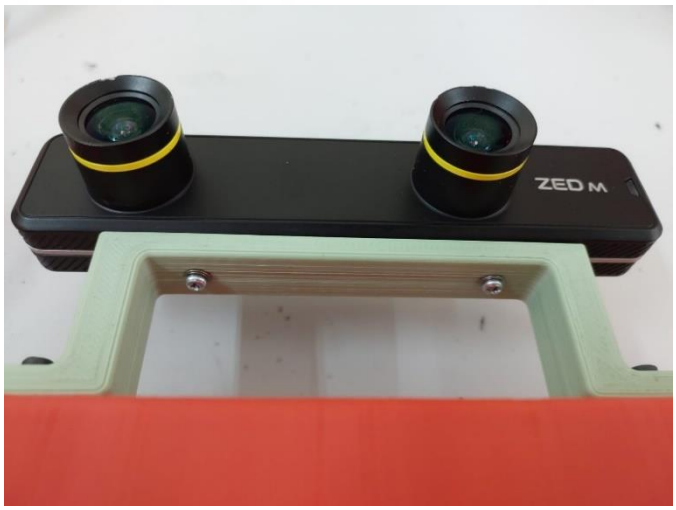

Figure 3.6. ZED mini camera.

- Reach M+ RTK-GNSS Module: The Real-Time Kinematic (RTK) module is a positioning device that observes and corrects prevalent errors in existing satellite navigation systems. It incorporates a 9DOF IMU with an update rate between 5 Hz and 14 Hz. By measuring location with up to 14 mm precision while in motion, it is far more accurate than standard GPS devices.

Figure 3.7. Reach M+ RTK-GNSS module.

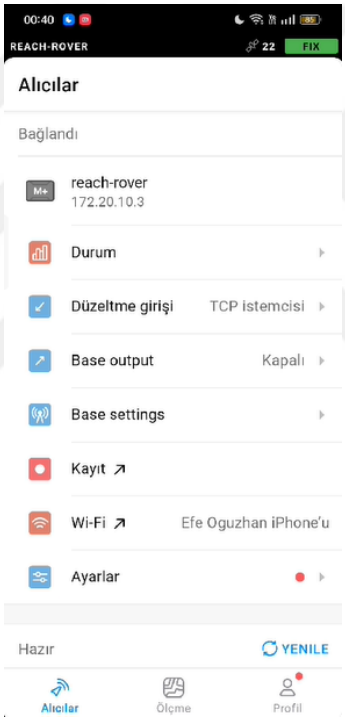

Figure 3.8. Connecting reach module to the internet via mobile hotspot.

A progressive localization approach is implemented using Emlid devices as GPS data receivers. These devices consisted of both base and rover configurations, which reliably delivered fix data, as visualized through our mobile application interface. To facilitate this, we utilized ROS to develop a custom node, aptly named RosNMEADriver. This node effectively parsed NMEA sentences from the GPS, ensuring valid checksums, and translated them into relevant ROS messages for position, velocity, and time reference. These messages were then published to specific ROS topics, such as tcpfix, tcpvel, and tcptime. To bridge

the gap between traditional longitude and latitude data and the more robot-friendly Cartesian x-y coordinates, we employed the robot_localization package.

The controllers mounted for operation on the UAV include the following.

- Nvidia Jetson Nano Development Board: The Nvidia Jetson Nano serves as a digital controller for both aerial and land vehicles. Essentially, it is a small computer designed for embedded system developers. It includes a 4-core CPU unit as well as a 128-core GPU unit. It has numerous pins and USB connection points that provide various communication protocols such as GPIO, I2C, I2S, SPI, and UART.

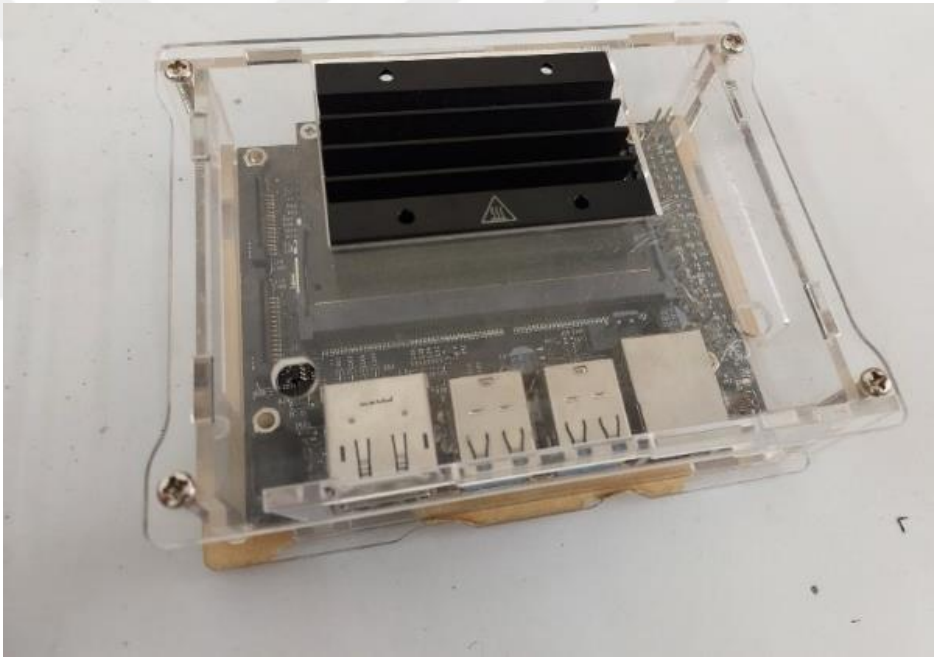

Figure 3.9. Nvidia Jetson Nano development board.

- Pixhawk: Pixhawk is an open-source autopilot system designed for UAVs and other robotic vehicles. It's a high-performance, versatile platform that provides a range of features for navigation, control, and telemetry. Pixhawk employs a combination of sensors, including accelerometers, gyroscopes, magnetometers, barometers, and GPS to ensure accurate measurements of the vehicle's position, speed, and direction. It also includes a powerful microcontroller that runs a real-time operating system, providing a range of control algorithms and functions for flight and mission planning.

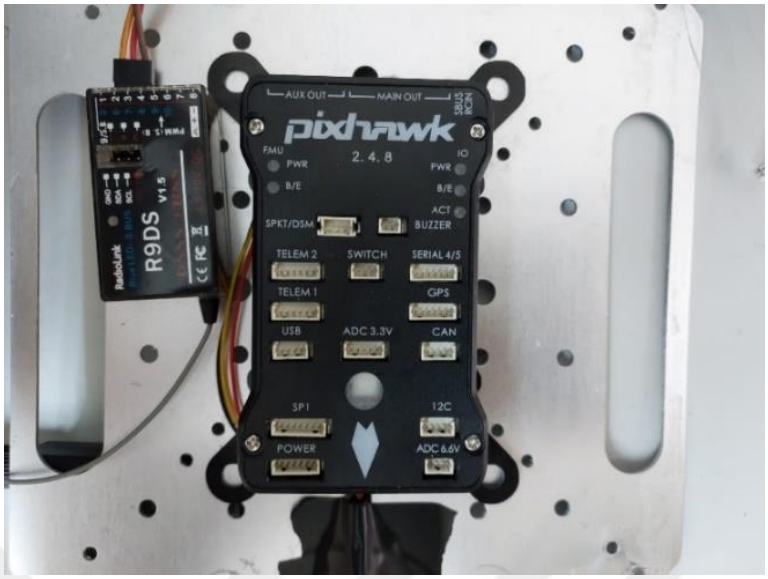

Figure 3.10. Pixhawk.

- Arduino Nano Control Board: The Arduino Nano is a microcontroller board based on the Atmega328. It features 14 digital input/output pins (of which 6 can be used as PWM outputs), 8 analog inputs, a 16MHz crystal oscillator, a USB socket, an ICSP connector, and a reset button.

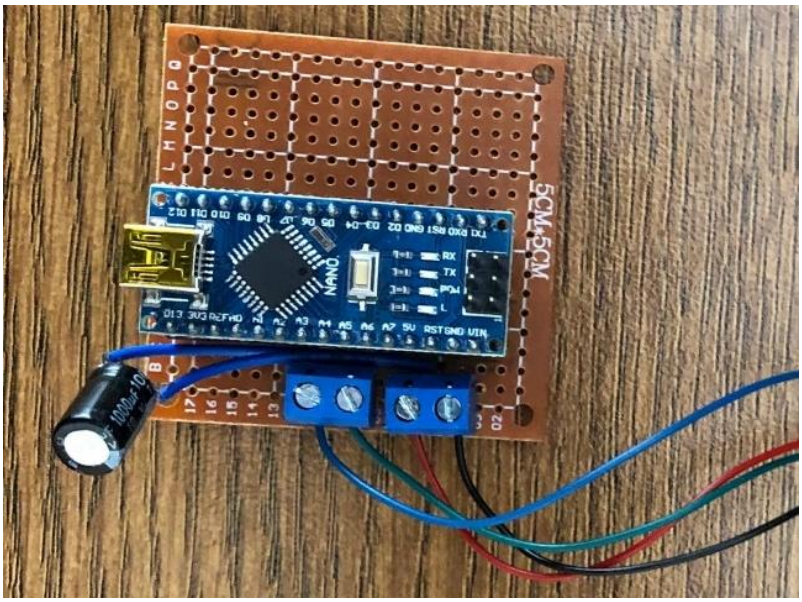

Figure 3.11. Arduino Nano control board.

The connections of the sensors and controllers mentioned above are depicted in Figure 3.12.

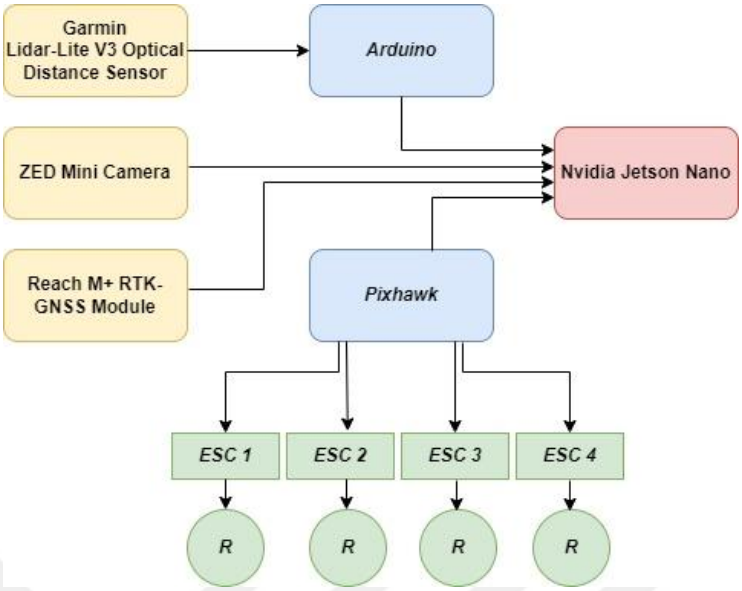

Figure 3.12. Connection diagram of sensors and controllers.

For each sensor, a sensor mounting bracket was designed and printed by a 3D printer. The sensors were mounted in accordance with the CAD model design. To prevent the formation of a magnetic field, an aluminum plate has been placed beneath the GPS receiver. A remote-control module was connected to the Pixhawk controller and configured with QGroundControl to allow manual control when necessary. The Nvidia Jetson Nano card was placed in a box and positioned between aluminum plates. This arrangement was designed to optimize access to the sensors and other electronic components and to protect them from potential external threats while the vehicle is airborne.

4. TRAJECTORY GENERATION

The importance of trajectory generation for drones is fundamentally undeniable, serving as a vital component that ensures both the operational success and safety of these unmanned aerial vehicles. The trajectory generation for a drone is a sophisticated process that blends the computational capability of MATLAB with the dynamic coordination of the Robot Operating System (ROS).

At the heart of this process lies a MATLAB-generated code, designed to communicate with a ROS node titled “ryobi_waypoint_chooser.py”. This communication takes place through two ROS topics: “/waypoints” and “/waypoints_checker”, utilizing “nav_msgs/Odometry” and “std_msgs/Bool” message types respectively. The former is crucial in assigning waypoints during the UGV's keyboard-controlled operation by sending this information to MATLAB, while the latter confirms the successful assignment of these waypoints. Distinctively, the MATLAB code gives the Euclidean distance between points for the usage of discretization, while maintaining a constant trajectory velocity. This data is then utilized by the MATLAB function, “generateTrajectoryFromData”. By employing the position and orientation information derived from the waypoints, alongside the Euclidean distance and trajectory velocity data, this function conducts the necessary calculations to generate the trajectory required for the UAV.

The given function calculates a discretized trajectory from raw x, y, and theta data, at a constant velocity (vel), with a certain discretization distance (d). It then computes the first and second derivatives of x, y, and theta with respect to time and consolidates the data in a full trajectory format.

Let’s denote the input vectors as

$$X = [x_1, x_2, \dots, x_n], \tag{4.1}$$

$$Y = [y_1, y_2, \dots, y_n], \tag{4.2}$$

$$TH = [th_1, th_2, \dots, th_n]. \tag{4.3}$$

The Euclidean distance ‘D’ between successive calculated points (x, y) is calculated as

$$D_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}\ for\ i = 1\ to\ i = n - 1. \quad (4.4)$$

The process could be represented as shown in Figure 4.1. The algorithm then discretizes the data. It starts from the first point and goes through the trajectory until the sum of the Euclidean distances ‘d’ exceeds the specified value. It then records that point, and the procedure is repeated.

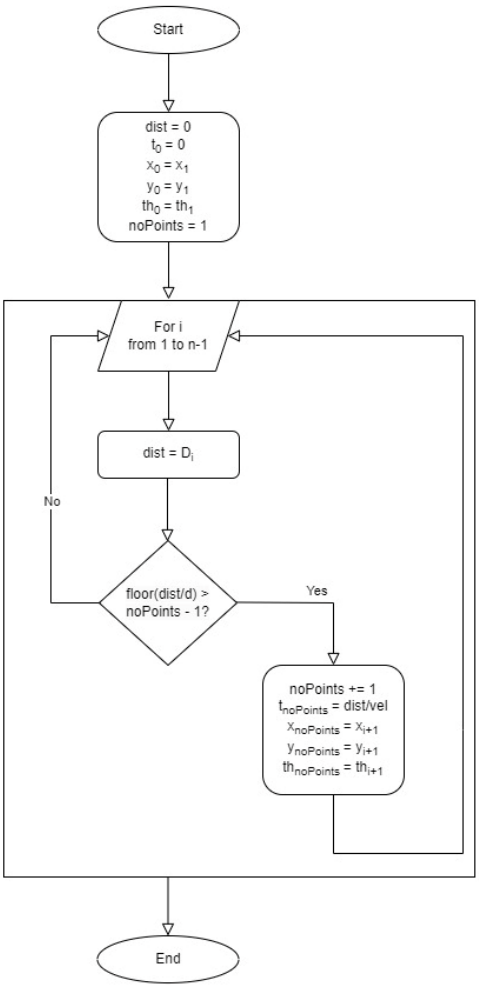

Figure 4.1. Pseudocode chart representing the discretization.

After that, the function calculates the first and second derivatives with respect to time of x, y, and th. This can be represented as

$$dx = \Delta x\ /\ \Delta t, \qquad dy = \Delta y\ /\ \Delta t, \qquad dth = \Delta th\ /\ \Delta t, \quad (4.5)$$

$$ddx = \Delta dx\ /\ \Delta t, \qquad ddy = \Delta dy\ /\ \Delta t, \qquad ddth = \Delta dth\ /\ \Delta t. \quad (4.6)$$

In Equation (4.5) and Equation (4.6), Δ denotes the change in the values (difference) between successive time steps. Finally, all these values are combined into the final trajectory as

$$trajectory = [t, x, y, th, dx, dy, dth, ddx, ddy, ddth]. \quad (4.7)$$

The information that is obtained from the "generateTrajectoryFromData.m" function is utilized in another function called "Traj_plot.m". This function is responsible for generating a set of visualizations to represent the robot's movement in a more understandable manner.

Table 4.1. Received waypoints for Task 1.

No.	Received Waypoints		No.	Received Waypoints	
	X	Y		X	Y
1	1.98	1.04	19	1.46	5.80
2	1.92	1.55	20	1.80	5.98
3	1.88	1.68	21	2.12	6.19
4	1.78	1.68	22	2.22	6.26
5	1.37	1.77	23	2.58	6.49
6	1.25	1.79	24	2.95	6.71
7	0.63	1.94	25	3.38	6.90
8	0.51	2.14	26	3.55	6.97
9	0.35	2.73	27	3.98	6.99
10	0.32	2.89	28	4.50	6.99
11	0.42	3.35	29	5.09	7.02
12	0.52	3.81	30	5.29	7.03
13	0.62	4.22	31	5.76	7.06
14	0.75	4.72	32	6.23	7.07
15	0.89	5.26	33	6.63	7.08
16	0.93	5.43	34	7.13	7.11
17	0.95	5.52	35	7.19	7.12
18	1.26	5.69	36	7.64	7.06

- The function generates a 2D trajectory of the robot, plotting y-coordinates against x-coordinates in a 2D view. This gives a broad view of the robot's path across the 2D space.

- It provides time plots for the x and y coordinates, as well as the orientation of the robot (th) at different points in time. These time plots illustrate how the position and orientation of the robot evolve over time.
- The function also shows velocity time plots for the robot. These are represented by the first derivatives of the x-coordinate, y-coordinate, and orientation of the robot with respect to time (dx, dy, dth). Essentially, they represent the velocity in the x and y directions, and the angular velocity of the robot over time.
- Lastly, the "Traj_plot.m" function displays the acceleration time plots. These are depicted as the second derivatives of the x-coordinate, y-coordinate, and orientation (ddx, ddy, ddth) over time. These plots effectively show the acceleration in the x and y directions, as well as the angular acceleration of the robot over time.

These plots together provide a comprehensive understanding of the robot's trajectory, its speed, and its acceleration, both in terms of linear movement and rotational movement. Understanding the robot's movement throughout its trajectory is crucial for assessing its overall performance and identifying any deviations from the intended path.

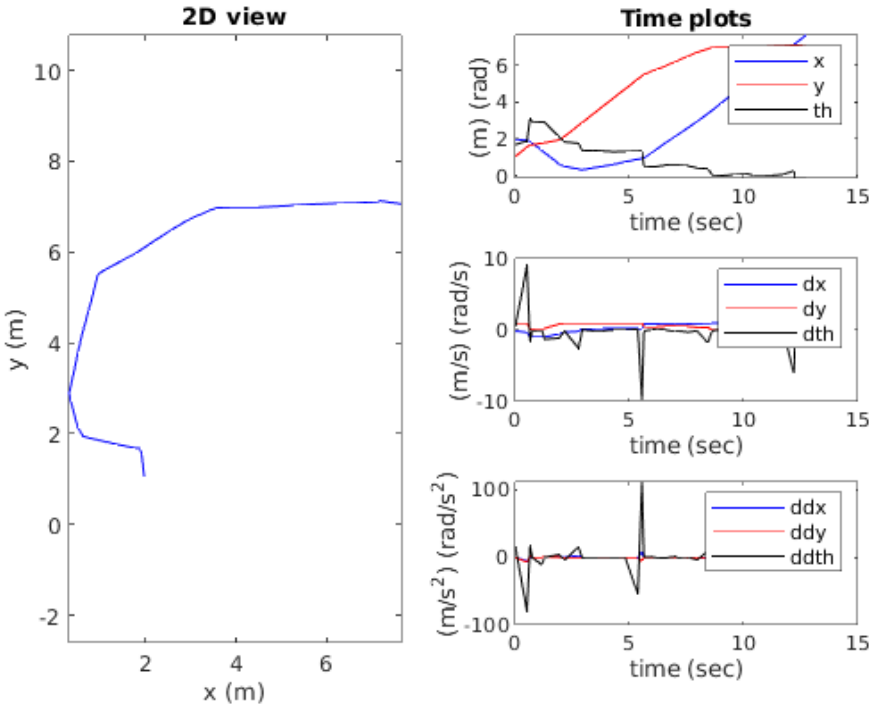

Figure 4.2. Robot's trajectory, velocity, and acceleration in terms of linear movement and rotational movement while tracing selected waypoints for Task 1.

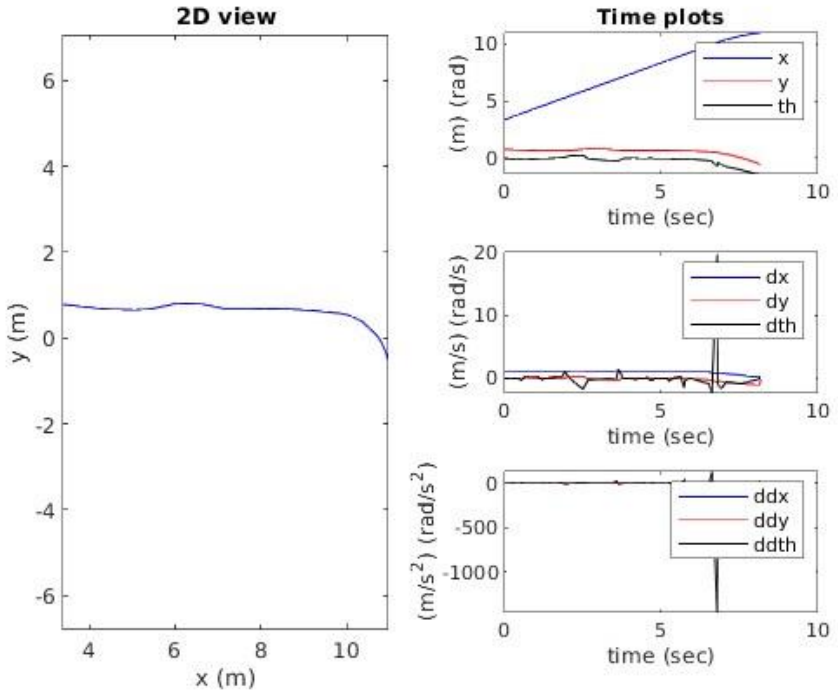

Figure 4.3. Robot's trajectory, velocity, and acceleration in terms of linear movement and rotational movement while tracing selected waypoints for Task 2.

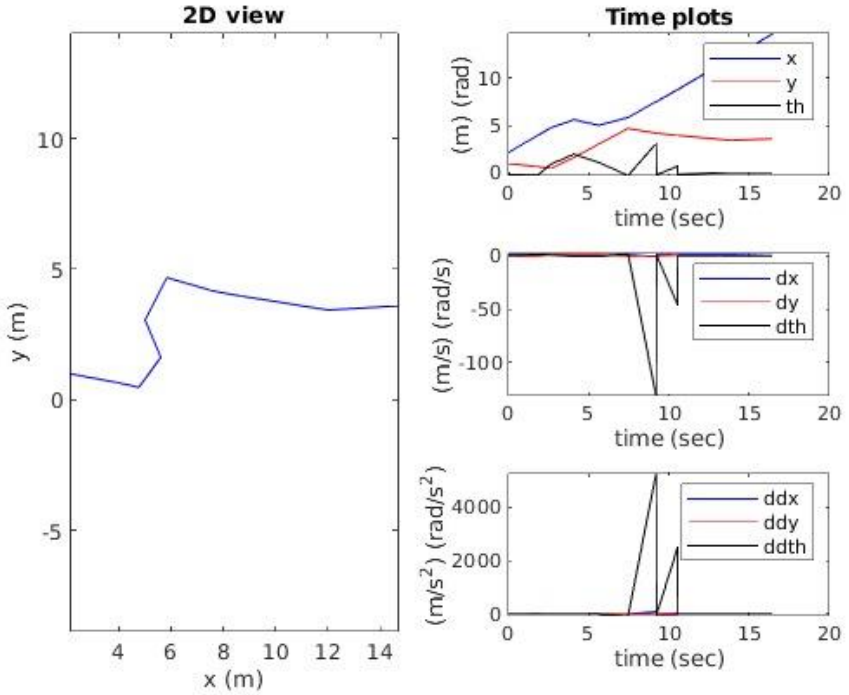

Figure 4.4. Robot's trajectory, velocity, and acceleration in terms of linear movement and rotational movement while tracing selected waypoints for Task 3.

After going over the first scenario where the drone follows the path set by the ground vehicle, we now shift our attention to a different method of path planning. In the second scenario, the drone will operate independently, tracing paths such as figure-8, circle, hexagon, ellipse, and line. These paths are set by a mathematical formula, not influenced by any external vehicle. Moreover, these cases will be used to demonstrate how the Extended Kalman Filter can be utilized to improve the drone's localization accuracy, considering the trajectories that have been assigned.

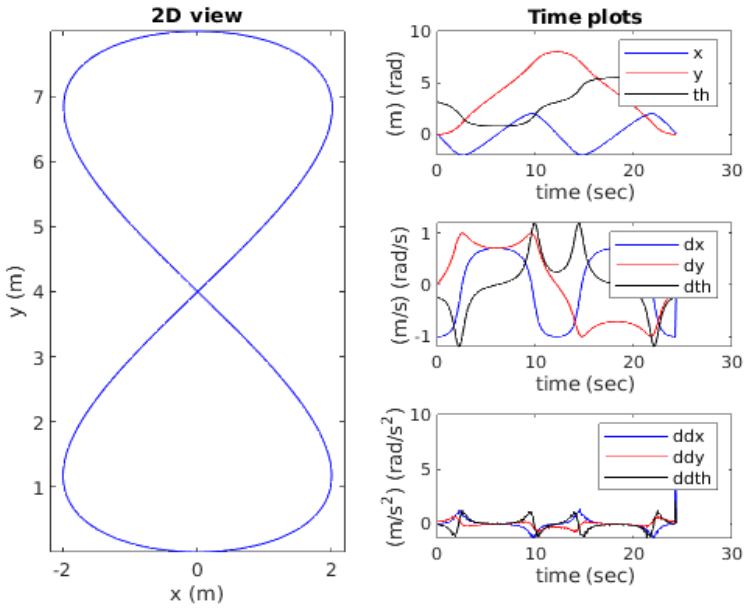

Figure 4.5. Robot's trajectory, velocity, and acceleration in terms of linear movement and rotational movement while tracing 8-figure.

In the 8-figure trajectory generation portion of the implemented MATLAB code, an array of parameters is initially set. These parameters, specifically the rotation offset, the offset for x and y coordinates, and the size of the figure in x and y dimensions, provide a basis for the generation of a continuous trajectory. Subsequently, a continuous trajectory is computed. This operation is performed for each value of t within the range from 0 to 2π. The x and y coordinates of the trajectory are determined using a combination of trigonometric functions including sine and cosine, with the parameters set previously. Following this, the orientation at each point along the trajectory is computed. This calculation employs the atan2 function, which is commonly used to return the four-quadrant inverse tangent. A discrete trajectory is then generated by applying the function “generateTrajectoryFromData.m”.

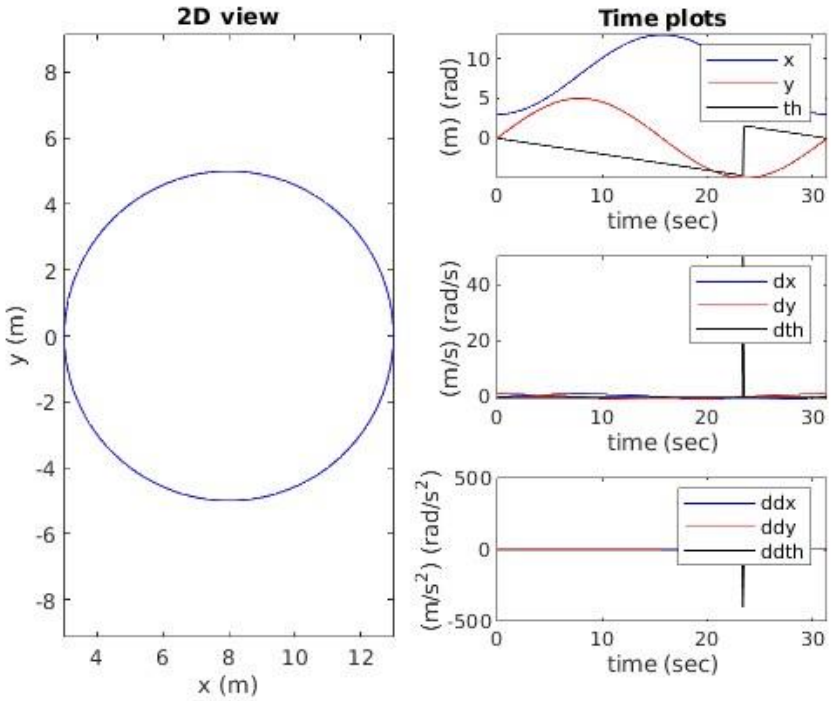

Figure 4.6. Robot's trajectory, velocity, and acceleration in terms of linear movement and rotational movement while tracing circle.

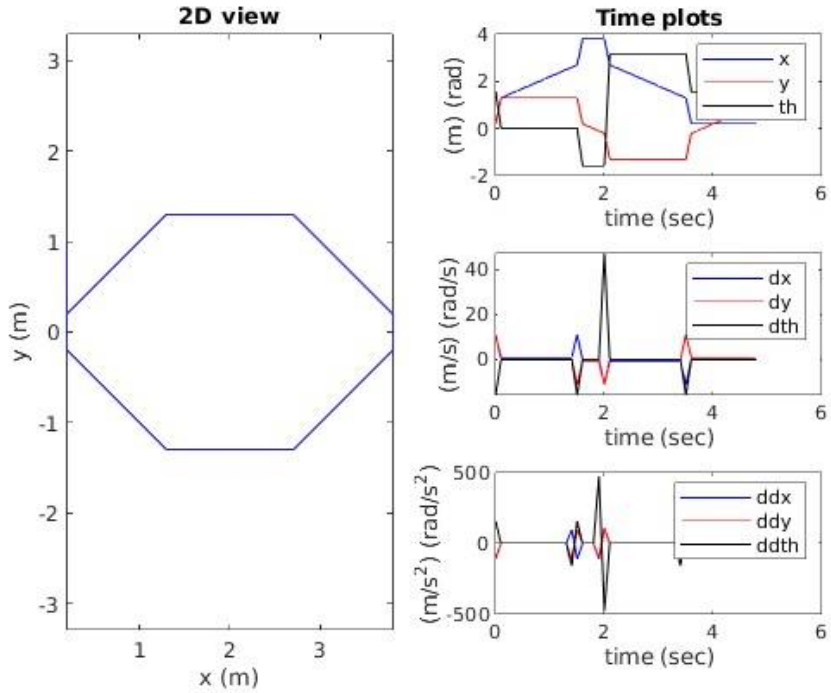

Figure 4.7. Robot's trajectory, velocity, and acceleration in terms of linear movement and rotational movement while tracing hexagon.

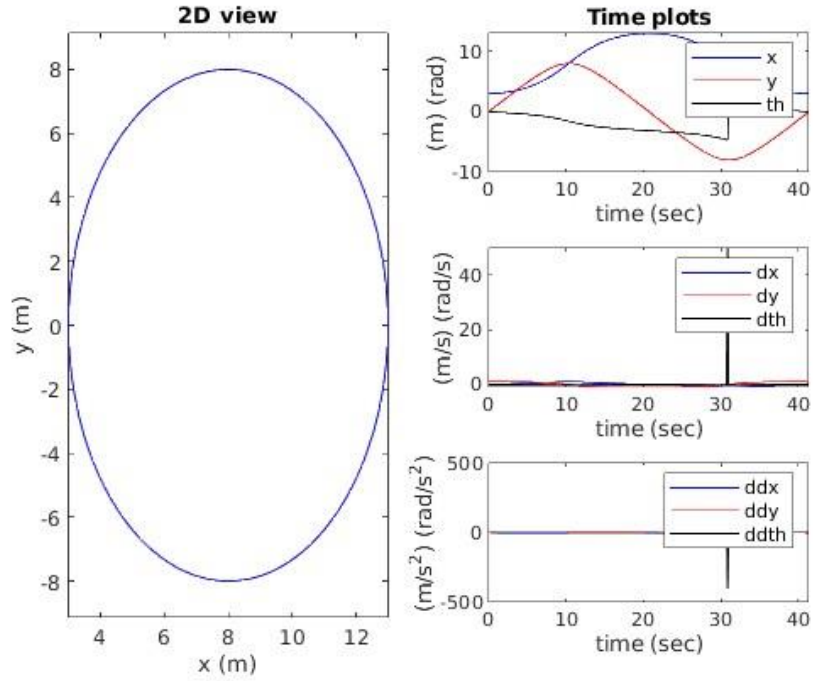

Figure 4.8. Robot's trajectory, velocity, and acceleration in terms of linear movement and rotational movement while tracing ellipse.

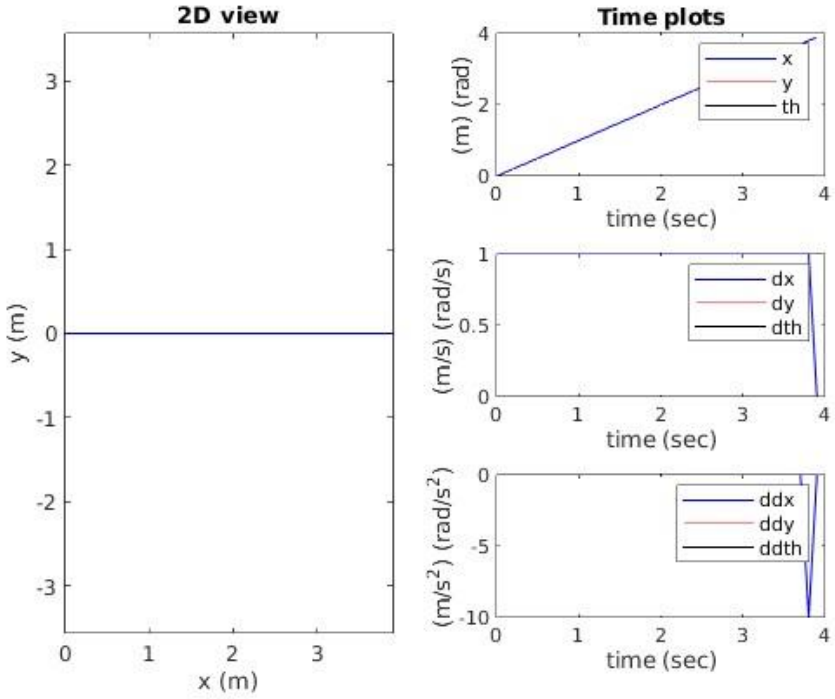

Figure 4.9. Robot's trajectory, velocity, and acceleration in terms of linear movement and rotational movement while tracing line.

The inputs for the function “generateTrajectoryFromData.m” are the previously computed x and y coordinates, the orientation, and the pre-determined distance between points and velocity. The output of this function is a discrete trajectory, which aligns with the continuous one but has distinct, evenly spaced points.

Generated trajectories are graphically represented using the Traj_plot function, and subsequently stored in a CSV file. This final step allows for the visual assessment of the trajectory and the preservation of the trajectory data for future use.

5. SENSOR FUSION

Sensor fusion is like putting together pieces from different data sources to create information that is more reliable and accurate than the separate pieces. When we blend sensor data in such a way, the combined result gives us a better picture than what each sensor could give us on its own.

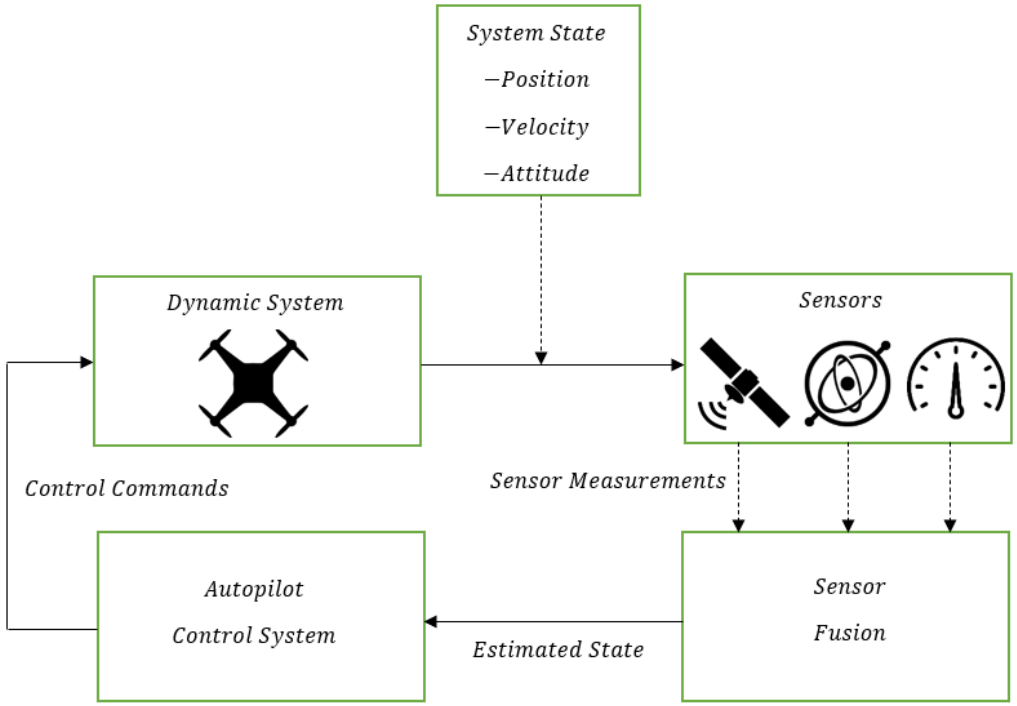

Figure 5.1. Sensor fusion schematics.

The Extended Kalman Filter is a version of the Linear Kalman Filter specifically designed to work with non-linear systems. It does this by treating the non-linear system as if it were linear, using an initial rough estimate around a particular state and its uncertainty. This estimated state and uncertainty are continually refined or updated as more data is processed, keeping the approximation relevant. Process model is given as

$$x_k = f(x_{k-1}, u_k, w_k). \tag{5.1}$$

Measurement model is given as

$$z_k = h(x_k, v_k), \tag{5.2}$$

where

- x_k: State vector.
- u_k: Control input vector.
- $w_k \sim N(0, Q_k)$: Process model noise vector.
- $v_k \sim N(0, R_k)$: Measurement noise vector.

The Extended Kalman Filter does not assure system stability, and it might deviate if the present state significantly differs from the actual state. Essentially, this means that the calculated state needs to stay near the real state; otherwise, the filter could go off track. In other words, the errors cannot become too big, and the initial conditions need to be nearly accurate.

The estimation process starts with the initial condition, denoted as $\hat{x}_0^+$, which represents our best guess of the system's starting state. Our goal is to advance time to find $\hat{x}_1^- = E[x_1]$, which is an expectation of x_1. We can achieve this by using the system's non-linear model. This can be illustrated as

$$\hat{x}_1^- = f(\hat{x}_0^+, u_1, 0), \tag{5.3}$$

or, in general, it can be given as

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+, u_k, 0). \tag{5.4}$$

In many fields, especially in control systems, signal processing, and estimation theory, understanding the accuracy of an estimated state is crucial. By carrying this covariance forward in time, we can continuously update our understanding of the system's state accuracy, which is essential for dynamic systems and real-time applications. Initially, P_0^+ represents the error covariance for the estimated state of x_0 and can be represented as

$$P_0^+ = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]. \tag{5.5}$$

In advancing from P_0 to P_1^- as shown in (5.6), we utilize a linear transformation technique to navigate the complexities of the system's potentially nonlinear dynamics. It is pivotal for its ability to linearly approximate the impact of both state changes and process noise on the system's evolution. The use of Jacobian matrices allows us to elegantly handle the non-linearities by linearizing the state transition and noise processes. It provides a pragmatic yet theoretically sound framework for predicting system behavior over time, especially when dealing with the inherent uncertainties and complexities of real-world dynamic systems. Equations can be formulated as

$$P_1^- = \nabla f_x P_0^+ \nabla f_x{}^T + \nabla f_w Q_1 \nabla f_w{}^T, \tag{5.6}$$

$$P_k^- = \nabla f_x P_{k-1}^+ \nabla f_x{}^T + \nabla f_w Q_k \nabla f_w{}^T, \tag{5.7}$$

where the Jacobian Matrices are given as

$$\nabla f_x = \left.\frac{\partial f}{\partial x}\right|_{x=\hat{x}_{k-1}^+}, \nabla f_w = \left.\frac{\partial f}{\partial w}\right|_{x=\hat{x}_{k-1}^+}. \tag{5.8}$$

Comparison of prediction steps regarding Linear Kalman Filter and Extended Kalman Filter can be demonstrated as shown in Figure 5.2, and Figure 5.3.

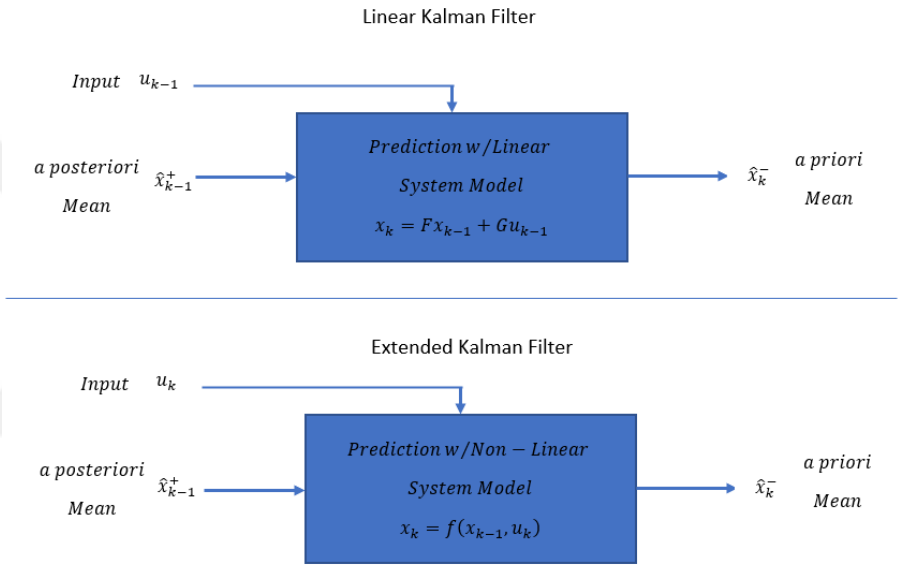

Figure 5.2. Prediction step for state comparing KF and EKF.

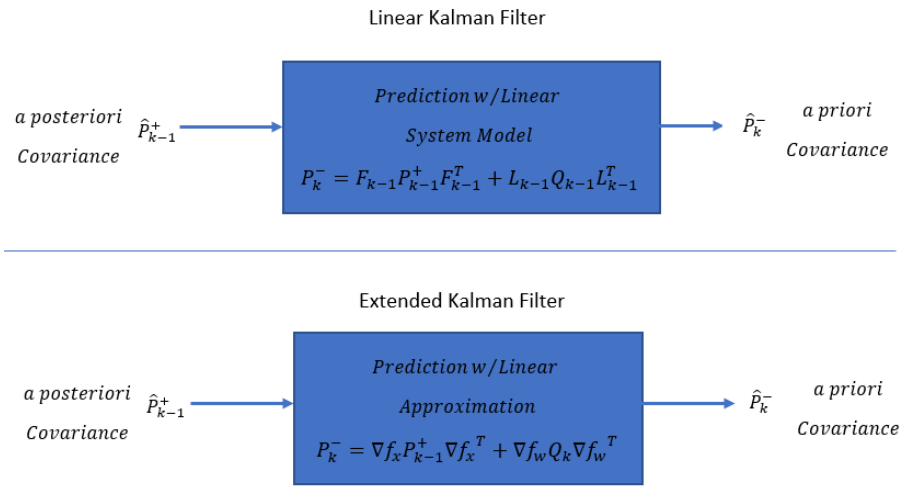

Figure 5.3. Prediction step for covariance comparing KF and EKF.

The Jacobian matrix ∇f_x is an $(m \times n)$ matrix for the function $f = f(x)$ whose elements are the partial derivatives of the m -outputs of the function with respect to n -inputs. This relationship can be expressed as

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\right), \tag{5.9}$$

$$\nabla f_x = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}, (\nabla f_x)_{ij} = \frac{\partial f_i}{\partial x_j}. \tag{5.10}$$

The process model state Jacobian matrix F_{k-1} is simply the Jacobian of the process model $x_k = f(x_{k-1}, u_k, w_k)$ with respect to the state vector. It is evaluated about the current best state estimate x_{k-1}. So, the matrix is a $(n \times n)$ matrix where n is the number of states. The mathematical representation can be given as

$$F_{k-1} = \nabla f_x = \nabla f_x = \left.\frac{\partial f}{\partial x}\right|_{x=x_{k-1}}. \tag{5.11}$$

The process model noise Jacobian matrix L_{k-1} is simply the Jacobian of the process model $x_k = f(x_{k-1}, u_k, w_k)$ with respect to the noise vector. It is evaluated about the current best state estimate x_{k-1}. So, the matrix is a $(n \times I)$ matrix where n is the number of states and I is the number of noise components. The mathematical expression can be demonstrated as

$$L_{k-1} = \nabla f_w = \nabla f_x = \left.\frac{\partial f}{\partial w}\right|_{x=x_{k-1}}. \tag{5.12}$$

The Kalman Filter corrects for state estimation errors by feeding back a weighted term based on observed measurement errors. Based on the current a priori state estimate $\hat{x}_k^-$ using the non-linear measurement function, the measurement $\hat{z}_k$ can be predicted [22].

$$\hat{z}_k = h(\hat{x}_k^-, 0). \tag{5.13}$$

We will define the innovation vector υ_k to be the difference between the predicted measurement $\hat{z}_k$ and the true measurement z_k that can be written as

$$\upsilon_k = z_k - h(\hat{x}_k^-, 0). \tag{5.14}$$

The innovation covariance S_k is defined [22].

$$S_k = E[\upsilon_k \upsilon_k^T]. \tag{5.15}$$

This term given in Equation (5.15) can be calculated as

$$S_k = \nabla h_x P_k^- \nabla {h_x}^T + \nabla h_\upsilon R_k \nabla {h_\upsilon}^T, \tag{5.16}$$

where the Jacobians are given as

$$\nabla h_x = \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}_k^-}, \nabla h_v = \left.\frac{\partial h}{\partial v}\right|_{x=\hat{x}_k^-}. \quad (5.17)$$

Comparison of measurement innovation steps regarding Linear Kalman Filter and Extended Kalman Filter can be demonstrated as shown in Figure 5.4, and Figure 5.5.

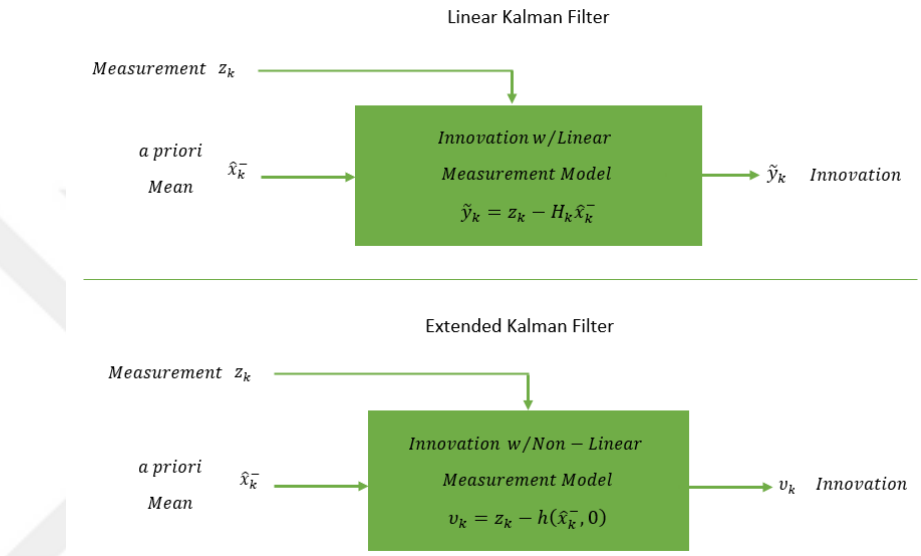

Figure 5.4. Comparison of KF and EKF in terms of measurement innovation.

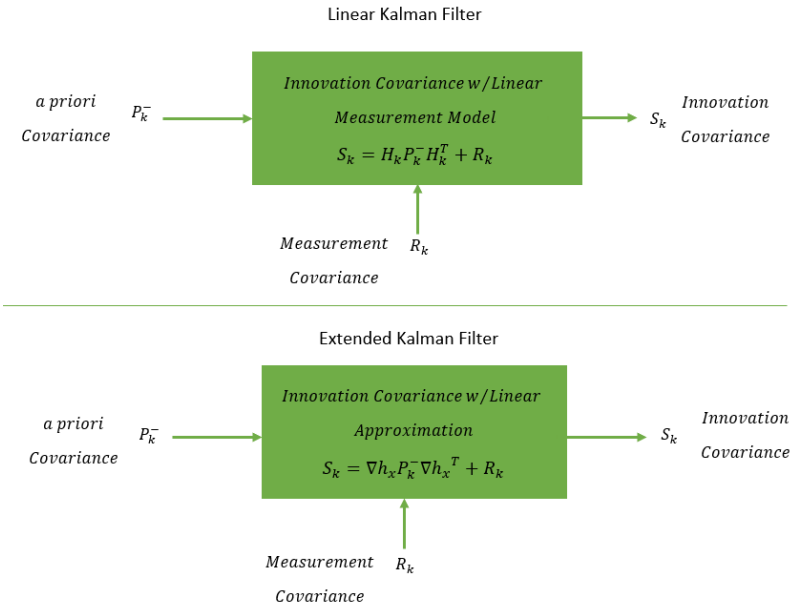

Figure 5.5. Comparison of KF and EKF in terms of innovation covariance.

The a priori state estimate $\hat{x}_k^-$ can be updated with the current measurement innovation information υ_k to form the a posteriori state estimate $\hat{x}_k^+$ using the Kalman gain as

$$K_k = P_k^- \nabla h_{x_k}^T S_k^{-1}, \tag{5.18}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \upsilon_k. \tag{5.19}$$

The a priori state error covariance matrix P_k^- can also be updated with the same information to form the a posteriori state error covariance matrix P_k^+.

$$P_k^+ = \left(I - K_k \nabla h_{x_k}\right) P_k^-. \tag{5.20}$$

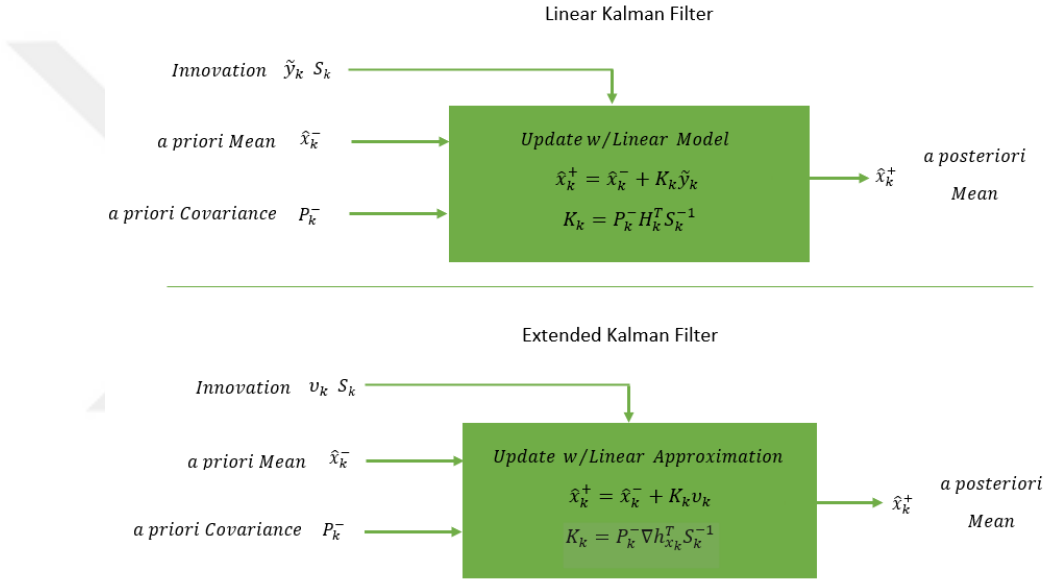

Figure 5.6. Comparison of KF and EKF in terms of update step for state.

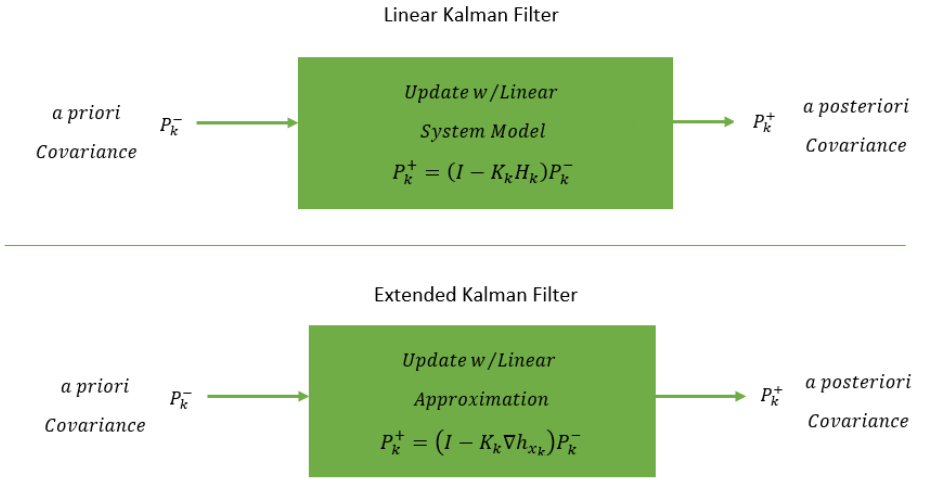

Figure 5.7. Comparison of KF and EKF in terms of update step for covariance.

Comparison of update steps regarding Linear Kalman Filter and Extended Kalman Filter can be demonstrated as shown in Figure 5.6 and Figure 5.7.

We have calculated the Jacobian matrices analytically from the equations using differentiation. It is also possible to calculate this numerically straight from the model using numerical differentiation. It can be mathematically captured as

$$P_k^+ = \left(I - K_k \nabla h_{x_k}\right) P_k^-. \tag{5.21}$$

Function of $y = f(x, u)$ can be given as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_l \end{bmatrix}\right). \tag{5.22}$$

Inputs can be defined [22].

$$X_0 = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad X_i = X_0 + \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix} \qquad \Delta x_j = \begin{cases} \Delta x, & j = i \\ 0, & i \neq j \end{cases}, \tag{5.23}$$

$$U_0 = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_l \end{bmatrix} \qquad U_i = U_0 + \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \vdots \\ \Delta u_l \end{bmatrix} \qquad \Delta u_j = \begin{cases} \Delta u, & j = i \\ 0, & i \neq j \end{cases}. \tag{5.24}$$

Outputs can be defined.

$$Y_0 = f(X_0, U_0), \tag{5.25}$$

$$Y_{x_i} = f(X_i, U_0), \tag{5.26}$$

$$Y_{u_i} = f(X_0, U_i), \tag{5.27}$$

$$\nabla f_x|_{(X_0, U_0)} \approx \frac{1}{\Delta x}\left[\left(Y_{x_1} - Y_0\right) \cdots \left(Y_{x_n} - Y_0\right)\right], \tag{5.28}$$

$$\nabla f_u|_{(X_0, U_0)} \approx \frac{1}{\Delta u}\left[\left(Y_{u_1} - Y_0\right) \cdots \left(Y_{u_l} - Y_0\right)\right], \tag{5.29}$$

$$\nabla f_x = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \qquad \nabla f_u = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial u_1} & \cdots & \frac{\partial f_m}{\partial u_l} \end{bmatrix}. \tag{5.30}$$

Calculation of Jacobian matrix ∇f_x for function $y = f(x, u)$ and given conditions (X_0, U_0) can be described as follows:

- Select appropriate perturb amount ∇x.
- Calculate initial state $Y_0 = (X_0, U_0)$.
- For each element in the state vector $i \in [1, \cdots, n]$, calculate the perturbed vector as

$$X_i = X_0 + [\Delta x_1 \quad \Delta x_2 \quad \cdots \quad \Delta x_n]^T, \Delta x_j = \begin{cases} \Delta x, & j = i \\ 0, & i \neq j \end{cases}. \tag{5.31}$$

- For each element in the state vector $i \in [1, \cdots, n]$, calculate the perturbed output as

$$Y_{x_i} = (X_i, U_0). \tag{5.32}$$

- For each element in the state vector $i \in [1, \cdots, n]$, calculate the corresponding Jacobian column as

$$(\nabla f_x)_i = \frac{1}{\Delta x}\left(Y_{x_i} - Y_0\right). \tag{5.33}$$

Calculation of Jacobian matrix ∇f_u for function $y = f(x, u)$ and given conditions (X_0, U_0) can be described as follows:

- Select appropriate perturb amount ∇u.
- Calculate initial state $Y_0 = (X_0, U_0)$.
- For each element in the state vector $i \in [1, \cdots, l]$, calculate the perturbed vector as

$$U_i = U_0 + [\Delta u_1 \quad \Delta u_2 \quad \cdots \quad \Delta u_l]^T, \Delta u_j = \begin{cases} \Delta u, & j = i \\ 0, & i \neq j \end{cases}. \tag{5.34}$$

- For each element in the state vector $i \in [1, \cdots, l]$, calculate the perturbed output as

$$Y_{u_i} = (X_i, U_i). \tag{5.35}$$

- For each element in the state vector $i \in [1, \cdots, l]$, calculate the corresponding Jacobian column as

$$(\nabla f_u)_i = \frac{1}{\Delta u}\left(Y_{u_i} - Y_0\right). \tag{5.36}$$

The Jacobian transformation is a linear approximation of the non-linear transformation (it works quite well if the system is approximately linear at the linearization point). Large uncertainties can lead to linearization error effects [22]. Linear approximation is most accurate near the linearization point. Linear approximation of uncertainty using Jacobian tends to under-estimate the error (leading to overconfident estimates).

6. COOPERATIVE BEHAVIOUR

Cooperative behavior is a clever system used in robotics, especially for ground-based robots called Unmanned Ground Vehicles and flying robots known as Unmanned Aerial Vehicles, or drones. This system helps these two kinds of robots to work together to figure out where they are and understand their surroundings better. The ground robot and the drone share information about their locations and what they can see around them. Because the drone can fly, it sees a big picture from above and helps correct the ground robot's understanding of its position. The ground robot, on the other hand, provides detailed views from the ground level and can help the drone know where it is when the drone can't connect to GPS. This teamwork makes both robots work better and makes their operations more accurate and adaptable. This shows why cooperative behavior is very important in the world of robotics.

Figure 6.1. UGV that is used during cooperative behavior.

The kinematics of the UGV have been designed based on the principle of differential drive. On the chassis, spring-loaded shock absorbers, each rated at 1500 lbs, have been deployed on both the right and left rear wheels. This arrangement aims to enhance the UGV's mobility in rough terrain conditions and achieve a more stable ride. The motive wheels, corresponding to the rear part of the Unmanned Ground Vehicle, are two 300mm diameter wheels, designed to adapt to off-road conditions, and are driven by Motiontech 450W 2400

RPM motors. To prevent deviation from the vehicle's route, the caster wheels are installed on both the left and right side of the vehicle's front part, aiming to ensure stable movement in various terrains. These wheels play a crucial role in maintaining the UGV's orientation. Considering the vehicle's turn center to be the midpoint of the common axis of the rear wheels, this aspect has been factored in during the vehicle's design and weight distribution considerations. The designs were carried out using CAD modeling software, and the position of the center of gravity has been carefully monitored.

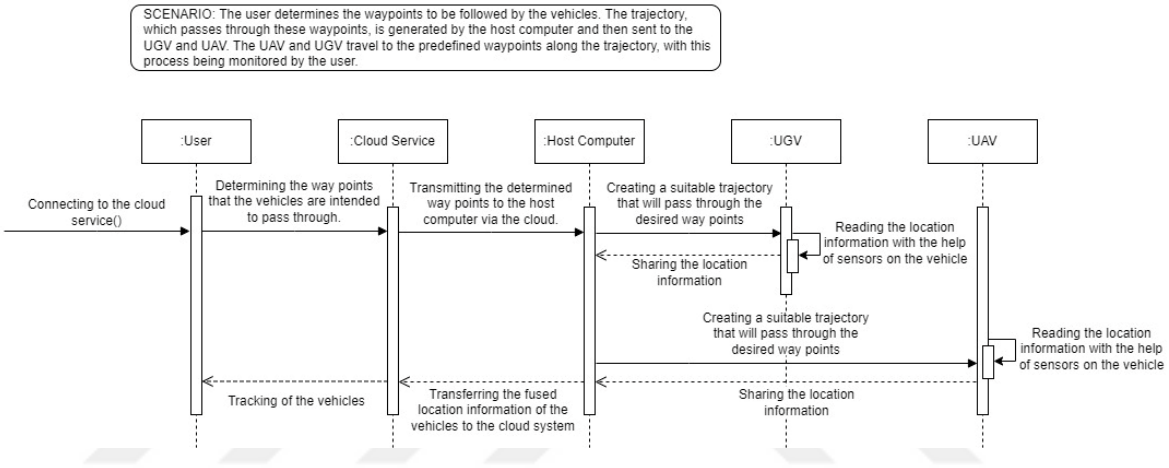

Figure 6.2. Sequence diagram of scenario demonstrating cooperative behavior.

In the scenario, the user first determines the route points, or waypoints, through which the vehicles are intended to pass. These waypoints are then transmitted to the main computer via cloud technology. The main computer generates a suitable trajectory based on these points, ensuring the vehicles will pass through the desired locations.

The generated trajectory is sent to both the UGV and UAV, which then start moving towards the predefined waypoints. Throughout this process, the user keeps a track on the vehicles, effectively monitoring the real-time progress of the trajectory traversal.

Fused location information of the UGV and UAV is transferred to the cloud system. This data is primarily acquired through sensors mounted on the vehicles, which continuously read the location information. This vital data is then shared and made accessible as needed.

The process iteratively continues, constantly updating the location data and enabling the vehicles to navigate through all the desired waypoints accurately, making this an efficient solution for vehicle routing and tracking.

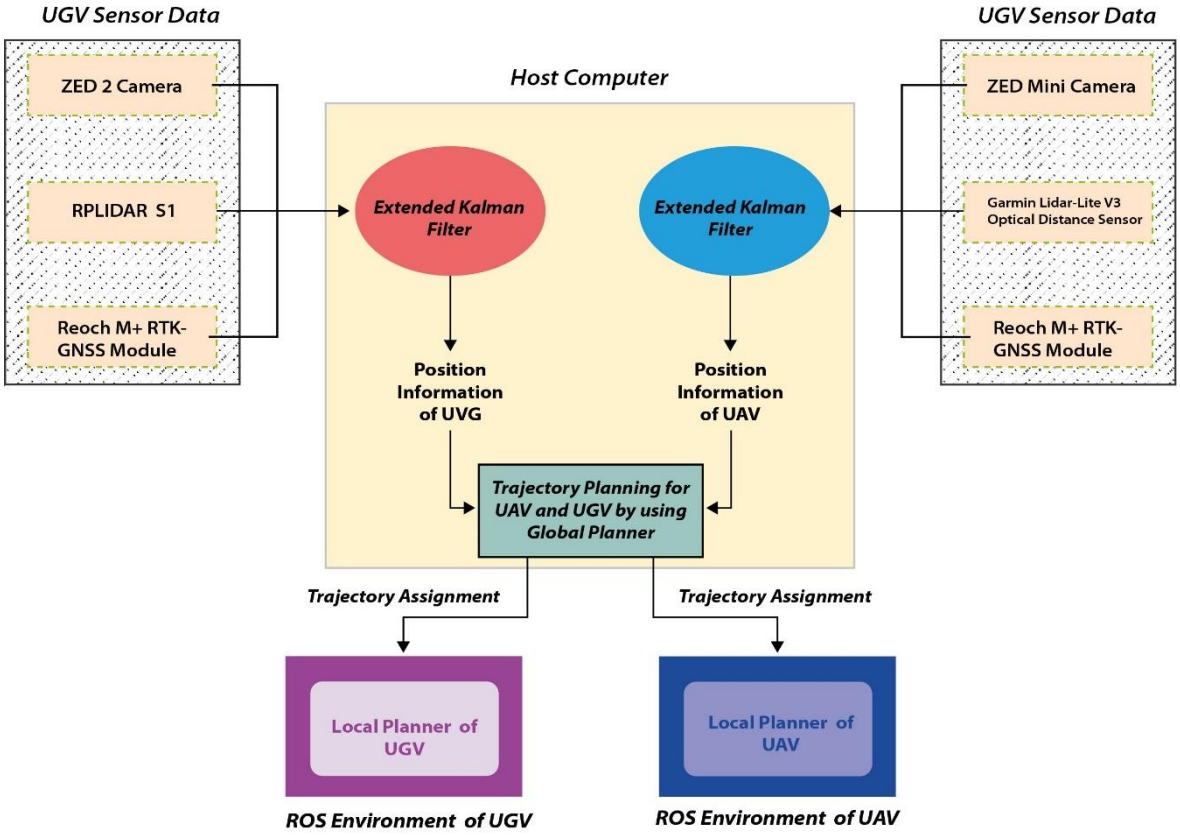

Figure 6.3. Operation algorithm of UAV and UGV.

As seen in Figure 6.3, each vehicle publishes data from its own sensors to the host computer. At the host computer, sensor data is processed using an Extended Kalman Filter (EKF), and position information is determined. Accordingly, a roadmap can be created for each vehicle.

A TP-Link access point device is employed to augment the range of an Everest router. This technical adaptation allowed seamless integration of multiple devices, such as UAV, UGV, and a side computer, into a unified network. This allowed the computer to get sensor data from the UAVs and UGVs easily. Additionally, this enhanced connectivity was pivotal in interlinking the Emlid base and rover GPS receivers.

Figure 6.4. TP-Link access point device and Everest router.

In Table 6.1., a comprehensive list of all devices used, along with their respective types and IP addresses, is presented. Additionally, Secure Shell (SSH) connection was specifically employed to establish a remote connection between the side computer and both the UGV and UAV.

Table 6.1. Types of devices and their IP addresses.

Device Type	IP Address
UAV	192.168.10.126
UGV	192.168.10.227
Side computer	192.168.10.163
Emlid base module	192.168.10.210
Emlid rover module	192.168.10.132

An access point that supports Secure Shell was chosen for a couple of important reasons. Firstly, a secure connection is provided by SSH, ensuring that the UGV and UAV can be linked safely to the side computer. More crucially, SSH is required so that the robots can be controlled remotely, and sensor information can be received on the side computer. Additionally, using this network, SSH connections between the side computer and both the UGV and UAV can be established over distances of up to 30 km.

Table 6.2. Specifications of 5GHz 300Mbps 23dBi Outdoor CPE.

HARDWARE FEATURES	
Interface	10/100Mbps Ethernet port
Button	RESET: Restore the device to its factory defaults
Power Supply	24V passive PoE adapter included
ESD Protection	15KV (Estimation is based on shielded CAT5e(or above) cable with an integrated grounding wire.)
Lightning Protection	Up to 6KV (Estimation is based on shielded CAT5e(or above) cable with an integrated grounding wire.)
Operating Temperature	-40 C to 70 C
Operating Humidity	10% to 90%
Certification	CE, FCC, RoHS, IP65
WIRELESS FEATURES	
802.11 Standards	11a/n

7. SIMULATION SETUP AND RESULTS

The simulation begins with the design of the UAV and UGV in SolidWorks, a solid modeling computer-aided design and computer-aided engineering (CAE) software program. Here, the physical properties of the vehicles, such as their shape, size, and the interconnection of their parts, are defined. The CAD models are used to generate Unified Robot Description Format (URDF) models, which are XML files describing the physical properties, visual geometry, collision geometry, and kinematic properties of the robots, including their joints, links, and physical parameters like inertia tensors and centers of mass. This URDF file is crucial for Robot Operating System, as it provides a comprehensive specification of the robot's physical aspects and how they relate to one another.

Figure 7.1. UAV modelled in SolidWorks.

Gazebo is used to create a realistic simulation environment for the UAV and UGV. These robots are tested in a risk-free, virtual world, enabling validation of control algorithms, navigation strategies, and interaction behaviors. In the simulation, the UGV is manually controlled via keyboard inputs. URDF models, generated from the initial SolidWorks designs of the UAV and UGV, are loaded into this simulated Gazebo environment. In this space, the UAV and UGV interact with the virtual world following the laws of physics. Waypoints are marked during this process, which the UAV will subsequently follow. This manual control allows for increased flexibility and adaptability to the surrounding environment. Simultaneously utilizing Gazebo and RViz offers the dual advantage of simulation and visualization. As waypoints are defined and adjusted for the UAV through

manual control of the UGV, these changes are dynamically reflected in the RViz environment.

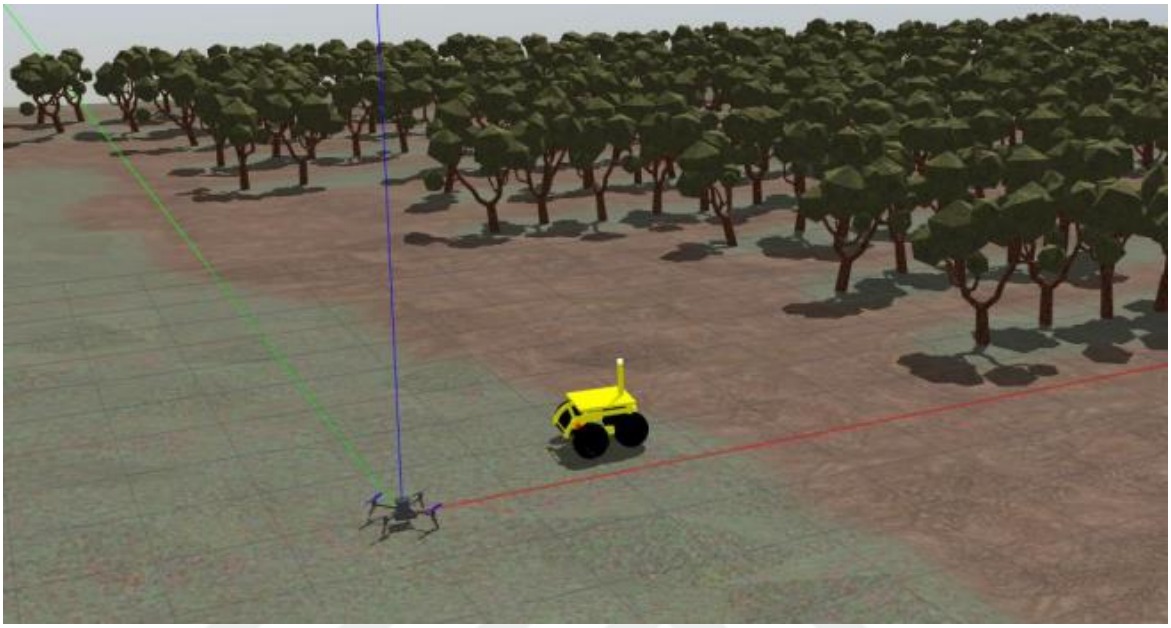

Figure 7.2. Simulation environment created in Gazebo.

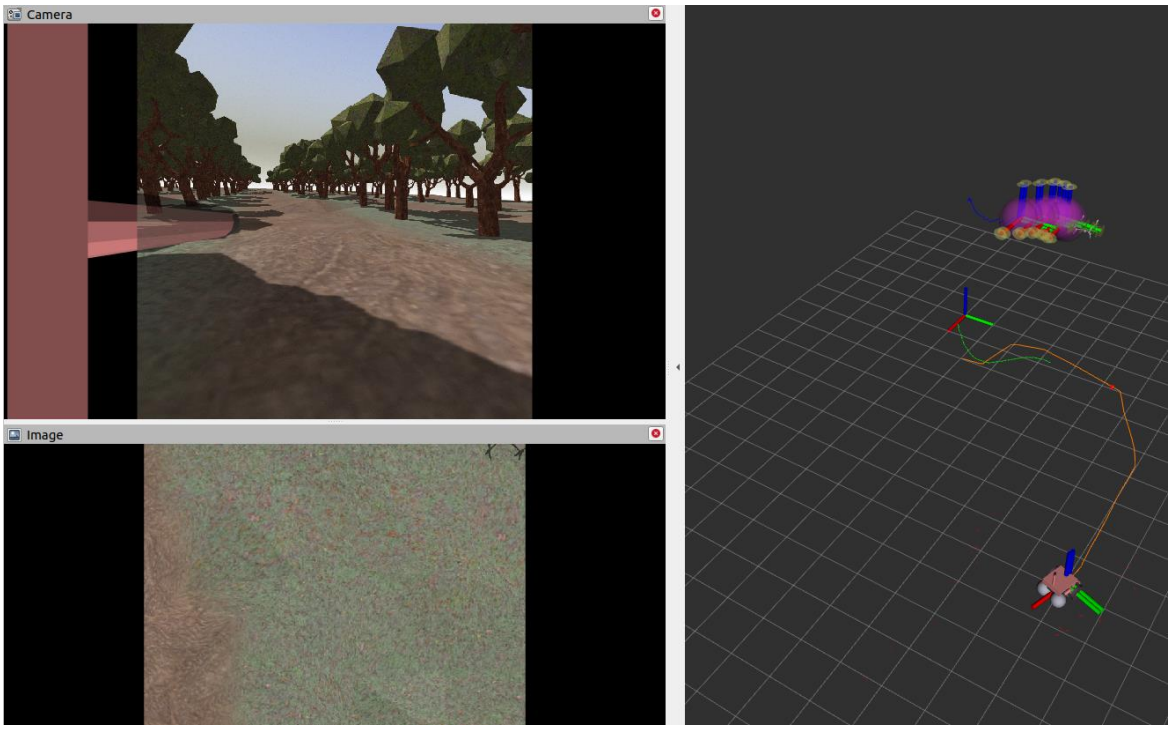

Figure 7.3. Simulation environment created in RViz.

The UAV is controlled via MAVROS, a ROS package that provides MAVLink functionality, offering a simple and unified interface to control drones. The UAV follows the trajectory outlined by the waypoints marked by the UGV. It uses sensor data to ensure it is following the correct path. This sensor data is processed using an Extended Kalman Filter, which is a recursive algorithm used to estimate the state of a dynamic system affected by random noise. This filtering process ensures a smooth, accurate trajectory.

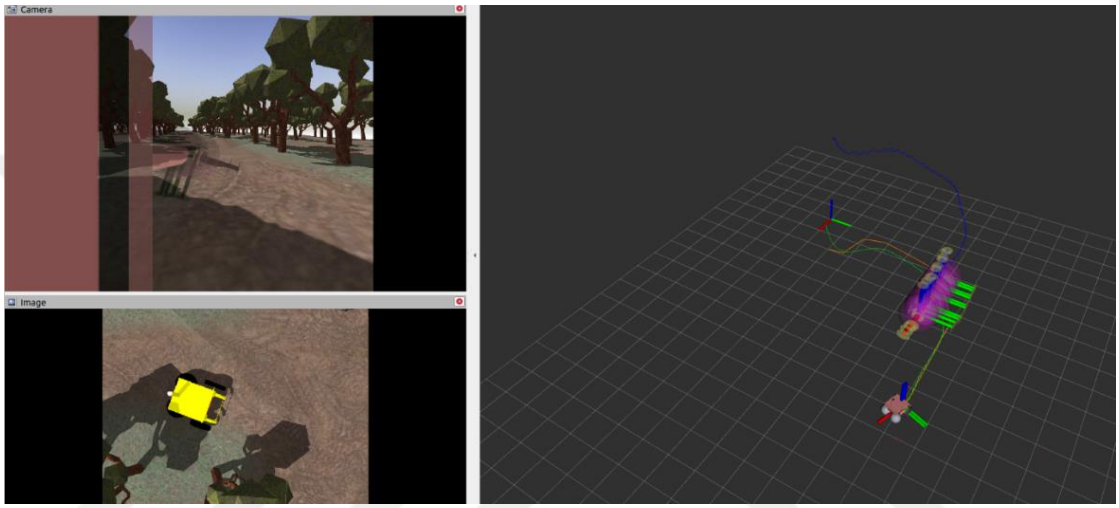

Figure 7.4. Realization of the first scenario.

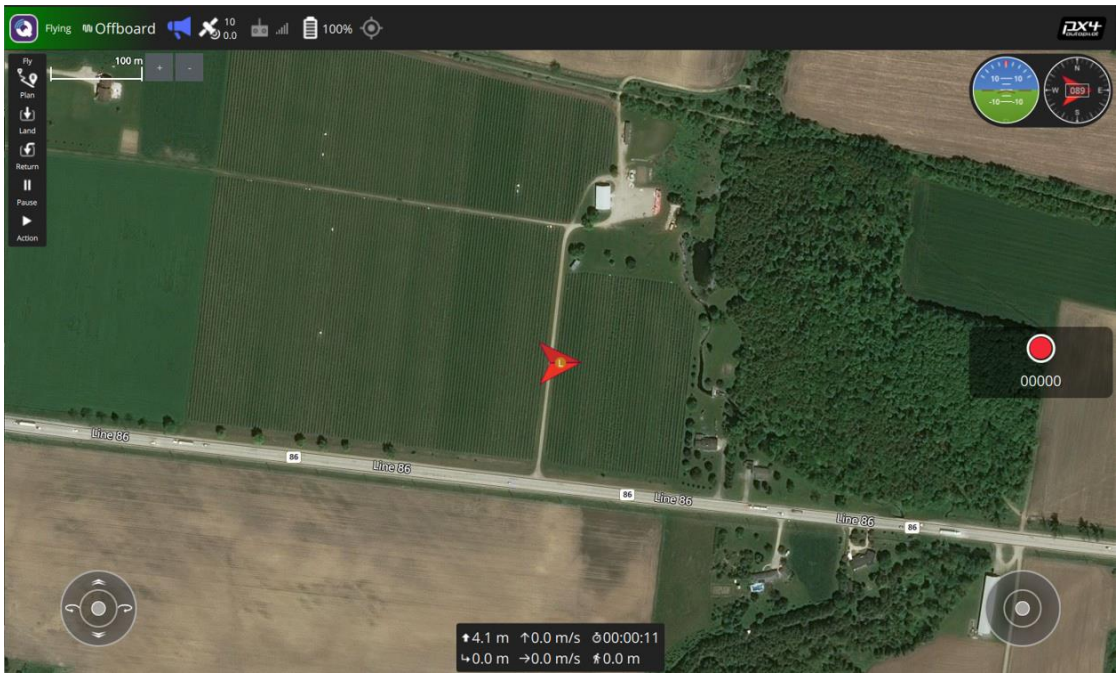

Figure 7.5. QGroundControl used for offboard mode.

The UAV is equipped with a depth camera, which it uses to visually locate the UGV during the simulation. In addition, QGroundControl is used for flight control and mission planning of the UAV. It provides full flight control and mission planning for any MAVLink enabled drone and displays real-time data about the vehicle's performance.

The low-level control of the UAV is managed by Pixhawk, an open-source, high-performance autopilot hardware suitable for fixed-wing, multirotor, and other platforms. This interacts with the JointVelocityController, a ROS controller that controls the velocity of each individual joint (or in this case, each propeller). This is achieved by sending desired velocities to the joints and employing a Proportional-Integral-Derivative (PID) controller to minimize the error between the desired and actual velocities. The trajectory for the UAV is generated and published in MATLAB, a high-level language and interactive environment that enables us to perform computationally intensive tasks. The trajectory data is then transferred to ROS, which manages the flight control system.

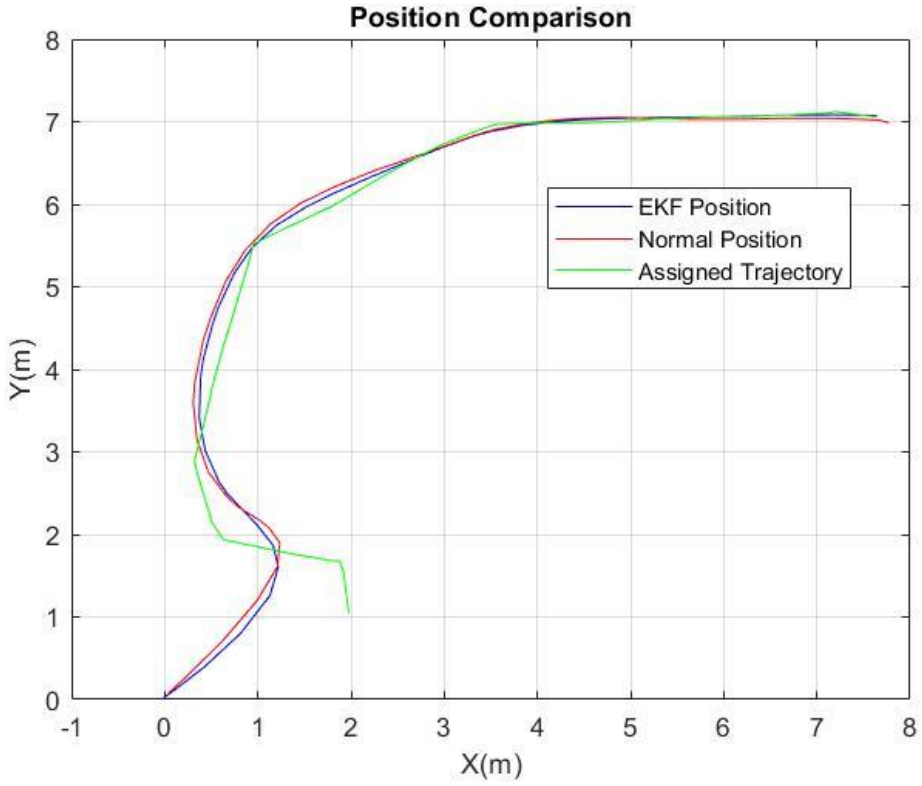

Figure 7.6. Position estimation with and without EKF while drone is tracing the chosen waypoints for Task 1.

Table 7.1. Comparison of position information regarding raw and EKF-processed sensor data in waypoint navigation for Task 1.

Point Nr.	Coordinate Axes and Euclidean Distances (m)	EKF	Normal	Assigned
1	X	0.508	0.466	0.683
	Y	4.522	4.535	4.472
	d	0.182	0.226	-
2	X	1.742	1.704	1.802
	Y	6.092	6.146	5.981
	d	0.126	0.192	-
3	X	3.578	3.571	3.555
	Y	6.899	6.912	6.970
	d	0.075	0.060	-
4	X	6.227	6.227	6.227
	Y	7.066	7.031	7.073
	d	0.008	0.042	-
5	X	7.192	7.192	7.192
	Y	7.084	7.040	7.123
	d	0.039	0.083	-

To quantify the accuracy of the estimated trajectory against the assigned reference path, the Euclidean distance formula is employed. It can be written as

$$d = \sqrt{\left(X - X_{ref}\right)^2 + \left(Y - Y_{ref}\right)^2}. \tag{7.1}$$

A comparative analysis was conducted by selecting five critical points from both the Extended Kalman Filter (EKF) processed dataset and the unfiltered dataset, to assess the deviation from the assigned trajectory. The chosen data points, listed in a tabular format, provide the X and Y coordinates that reveal the positional discrepancies, offering a clear evaluation of the path tracking performance with and without the application of the EKF. The calculated errors, represented as Euclidean distances, serve as a metric to evaluate the improvement in localization accuracy when the EKF is applied, as opposed to the baseline data which lacks such filtering.

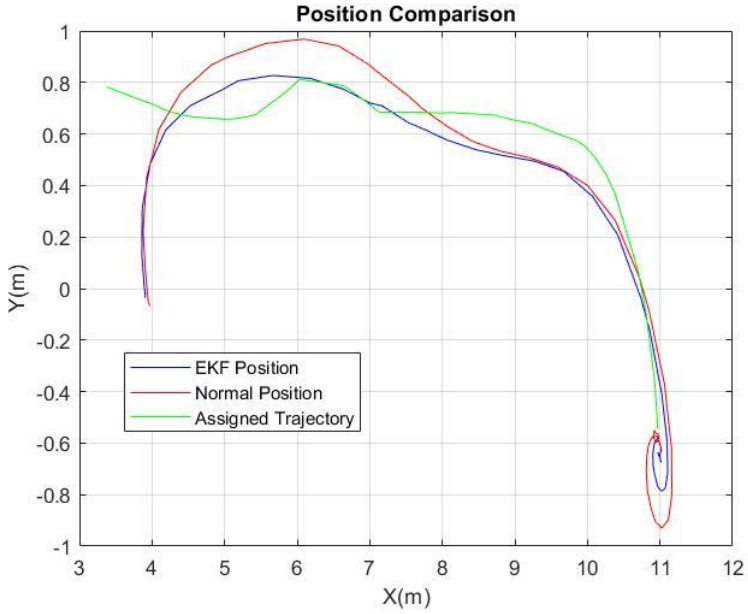

Figure 7.7. Position estimation with and without EKF while drone is tracing the chosen waypoints for Task 2.

Table 7.2. Comparison of position information regarding raw and EKF-processed sensor data in waypoint navigation for Task 2.

Point Nr.	Coordinate Axes and Euclidean Distances (m)	EKF	Normal	Assigned
1	X	5.229	5.229	5.229
	Y	0.812	0.922	0.666
	d	0.146	0.256	-
2	X	6.622	6.830	6.642
	Y	0.775	0.897	0.788
	d	0.024	0.218	-
3	X	9.667	9.670	9.963
	Y	0.456	0.458	0.552
	d	0.311	0.308	-
4	X	10.990	11.050	10.930
	Y	-0.359	-0.358	-0.361
	d	0.061	0.121	-
5	X	10.15	10.19	10.38
	Y	0.323	0.333	0.371
	d	0.235	0.194	-

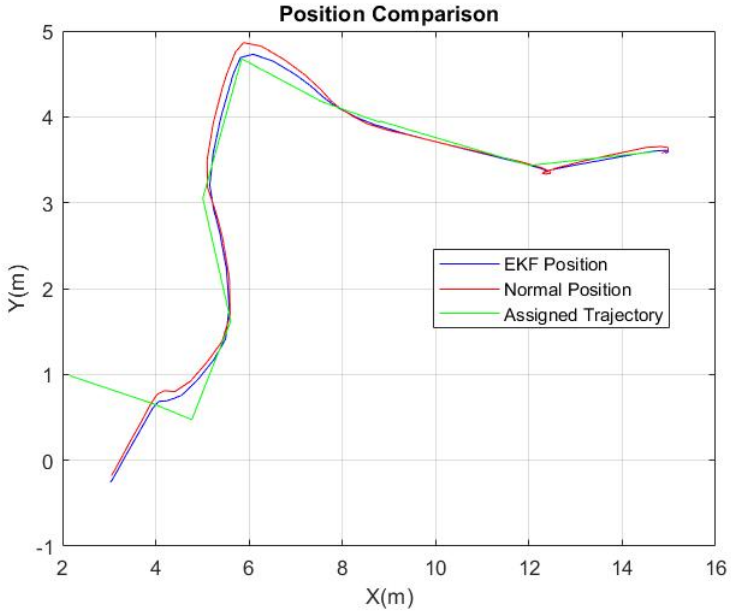

Figure 7.8. Position estimation with and without EKF while drone is tracing the chosen waypoints for Task 3.

Table 7.3. Comparison of position information regarding raw and EKF-processed sensor data in waypoint navigation for Task 3.

Point Nr.	Coordinate Axes and Euclidean Distances (m)	EKF	Normal	Assigned
1	X	4.549	4.525	4.772
	Y	0.754	0.842	0.473
	d	0.359	0.444	-
2	X	6.683	6.756	6.535
	Y	4.594	4.665	4.474
	d	0.190	0.292	-
3	X	8.825	8.812	8.866
	Y	3.887	3.871	3.940
	d	0.067	0.088	-
4	X	5.318	5.352	5.140
	Y	2.767	2.766	2.740
	d	0.180	0.214	-
5	X	14.04	14.02	14.03
	Y	3.543	3.585	3.554
	d	0.012	0.034	-

The simulation is a complex interplay of mechanical design, physics, controls, sensor fusion, and trajectory tracking. Its modular structure provides flexibility and adaptability, with the potential to incorporate more advanced algorithms for improved performance and functionality.

In the MATLAB code called “positionget.m”, comparisons between position data derived from two different sources, one being unfiltered GPS coordinates and the other being coordinates processed by an Extended Kalman Filter, are performed. The purpose is to assess the efficacy of the EKF in improving the accuracy of the position data.

Subscribers to the ROS topics “/quad1/odometry/abs” and “/quad1/odometry/gps” are created. These subscribers enable the code to receive messages about the odometry data for the quadrotor, derived from the EKF and the GPS, respectively.

The final “Pos_array” and “Pos_ekf arrays” contain a history of the positions reported by the GPS and EKF, respectively, at each time the position changed. These arrays can be used to analyze and compare the trajectory followed by the quadrotor according to each source of position data. The EKF utilized in this scenario combines position information from GPS, accelerometer, and magnetometer Gazebo plugins to estimate the quadrotor's state.

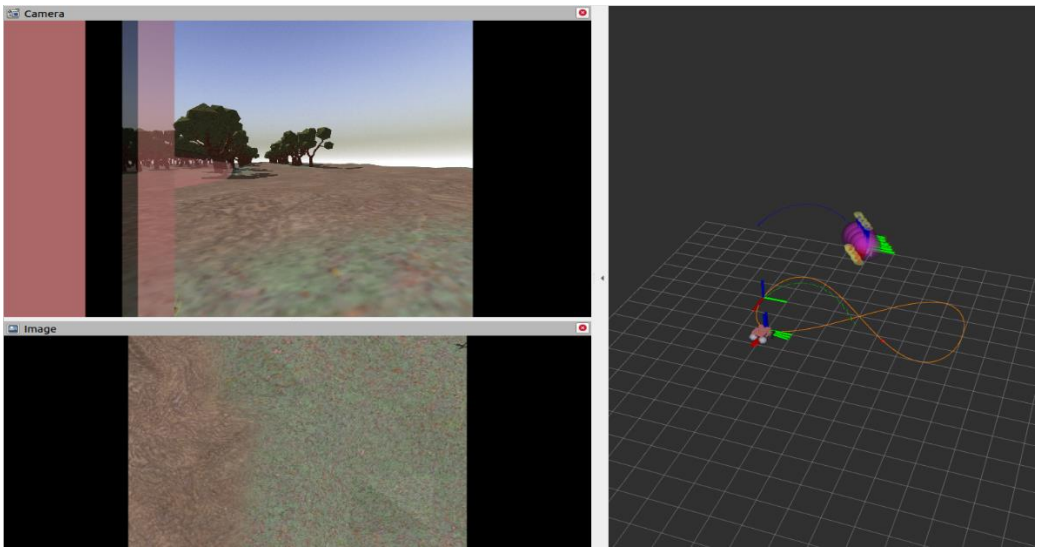

Figure 7.9. Realization of the figure-8 scenario.

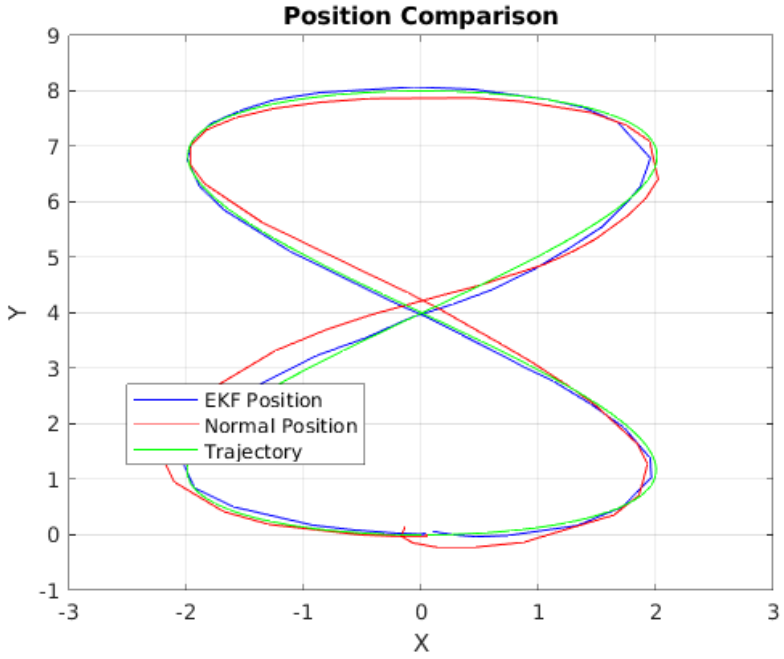

Figure 7.10. Position estimation with and without EKF while the drone is tracing 8-figure.

Table 7.4. Comparison of position information regarding raw and EKF-processed sensor data collected while drone is tracing 8-figure.

Point Nr.	Coordinate Axes and Euclidean Distances (m)	EKF	Normal	Assigned
1	X	-0.055	-0.111	0.000
	Y	-2.941	-2.967	-3.000
	d	0.080	0.116	-
2	X	1.490	1.559	1.695
	Y	-1.000	-1.000	-1.000
	d	0.205	0.136	-
3	X	0.913	0.854	0.900
	Y	-3.921	-4.061	-3.950
	d	0.032	0.119	-
4	X	-0.957	-0.995	-0.944
	Y	-3.972	-3.901	-4.002
	d	0.033	0.113	-
5	X	-1.103	-1.115	-1.082
	Y	-0.168	-0.112	-0.242
	d	0.076	0.133	-

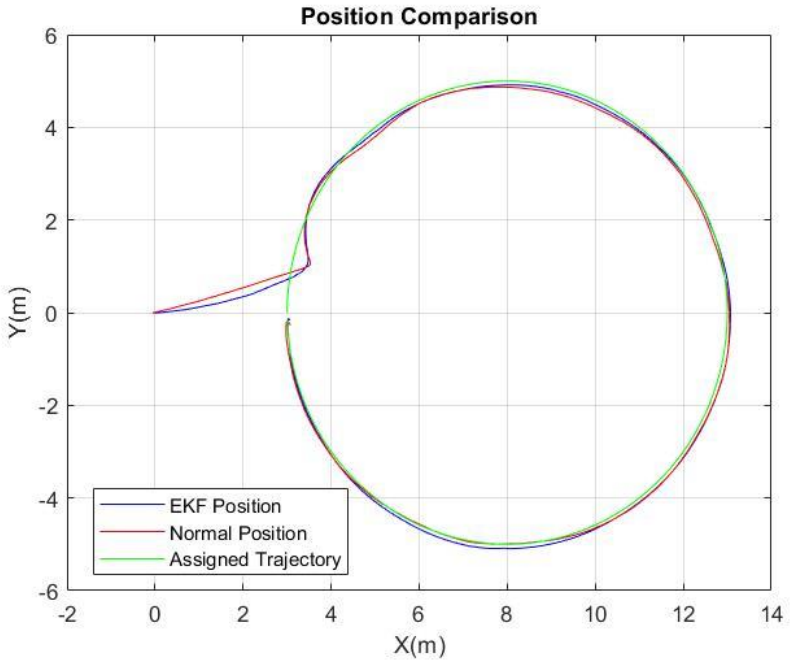

Figure 7.11. Position estimation with and without EKF while the drone is tracing circle.

Table 7.5. Comparison of position information regarding raw and EKF-processed sensor data collected while drone is tracing circle.

Point Nr.	Coordinate Axes and Radius (m)	EKF	Normal	Assigned
1	X	3.479	3.523	-
	Y	1.103	1.028	-
	r	4.654	4.594	5.000
2	X	3.787	3.793	-
	Y	2.867	2.824	-
	r	5.096	5.067	5.000
3	X	5.118	5.177	-
	Y	3.982	3.942	-
	r	4.916	4.849	5.000
4	X	9.125	9.037	-
	Y	4.784	4.743	-
	r	4.915	4.855	5.000
5	X	12.600	12.530	-
	Y	1.914	1.962	-
	r	4.982	4.937	5.000

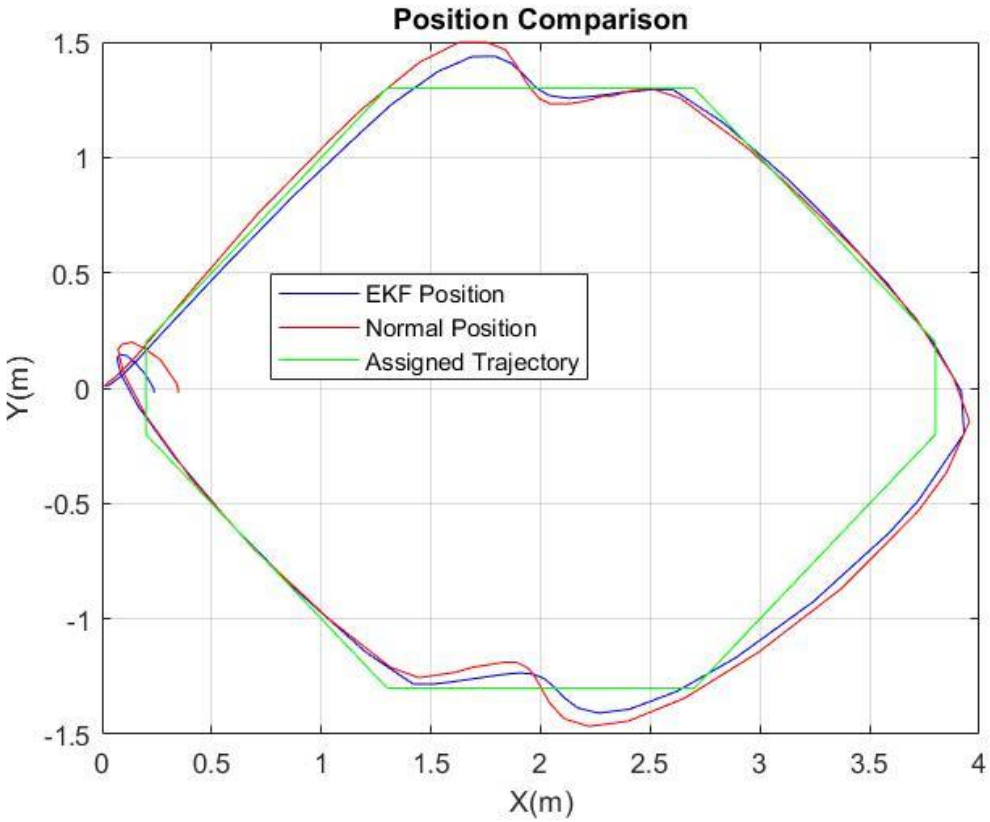

Figure 7.12. Position estimation with and without EKF while the drone is tracing hexagon.

Table 7.6. Comparison of position information regarding raw and EKF-processed sensor data collected while drone is tracing hexagon.

Point Nr.	Coordinate Axes (m)	EKF	Normal	Assigned
1	X	1.752	1.752	1.752
	Y	1.438	1.499	1.300
2	X	3.926	3.955	3.800
	Y	-0.144	-0.144	-0.144
3	X	2.220	2.220	2.220
	Y	-1.398	-1.466	-1.300
4	X	1.889	1.889	1.889
	Y	-1.235	-1.188	-1.300
5	X	2.700	2.700	2.700
	Y	1.233	1.212	1.300

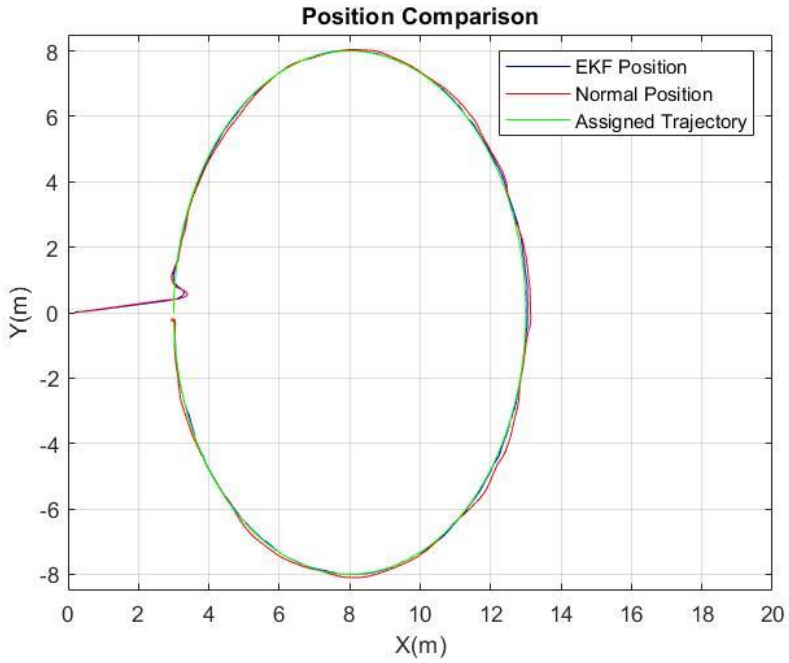

Figure 7.13. Position estimation with and without EKF while the drone is tracing ellipse.

Table 7.7. Comparison of position information regarding raw and EKF-processed sensor data collected while drone is tracing ellipse.

Point Nr.	Coordinate Axes and Euclidean Distances (m)	EKF	Normal	Assigned
1	X	13.040	13.140	13.000
	Y	-0.214	-0.189	-0.197
	d	0.046	0.143	-
2	X	12.350	12.440	12.340
	Y	-4.000	-4.187	-3.970
	d	0.032	0.239	-
3	X	3.274	3.213	3.275
	Y	-2.618	-2.721	-2.616
	d	0.002	0.122	-
4	X	4.790	4.815	4.782
	Y	6.075	5.970	6.120
	d	0.046	0.154	-
5	X	11.530	11.560	11.510
	Y	5.736	5.853	5.694
	d	0.045	0.166	-

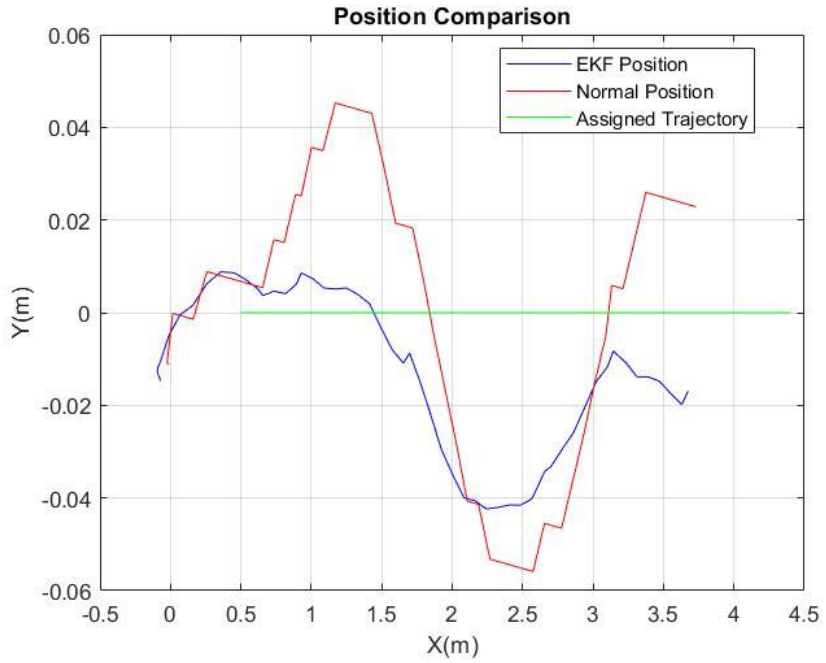

Figure 7.14. Position estimation with and without EKF while the drone is tracing line.

Table 7.8. Comparison of position information regarding raw and EKF-processed sensor data collected while drone is tracing line.

Point Nr.	Coordinate Axes (m)	EKF	Normal	Assigned
1	X	1.171	1.171	1.171
	Y	0.005	0.045	0.000
2	X	2.575	2.575	2.575
	Y	-0.040	-0.056	0.000
3	X	1.720	1.720	1.720
	Y	-0.011	0.018	0.000
4	X	1.001	1.001	1.001
	Y	0.007	0.036	0.000
5	X	3.213	3.213	3.213
	Y	-0.010	0.005	0.000

The Root Mean Square Error (RMSE) was utilized as a statistical measure to further assess the magnitude of the positional errors from the selected points, offering a comprehensive view of the overall trajectory tracking accuracy. It can be expressed as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}\,. \tag{9.2}$$

The efficacy of the EKF became evident through its lower RMSE values across all shapes, indicating a more precise adherence to the assigned trajectories and demonstrating its robustness in error correction.

Table 7.9. Peak deviations and RMSE analysis of drone navigation in autonomous waypoint navigation and autonomous geometric flight modes.

Shape	Flight Mode	Sensor Data Type	Root Mean Square Error (m)	Peak Deviation (m)
Task 1	Autonomous Waypoint Navigation	EKF	0.106	0.182
		Normal	0.142	0.226
Task 2	Autonomous Waypoint Navigation	EKF	0.188	0.311
		Normal	0.228	0.308
Task 3	Autonomous Waypoint Navigation	EKF	0.201	0.359
		Normal	0.260	0.444
Figure-8	Autonomous Geometric	EKF	0.106	0.205
		Normal	0.124	0.136
Circle	Autonomous Geometric	EKF	0.158	0.346
		Normal	0.191	0.407
Hexagon	Autonomous Geometric	EKF	0.103	0.138
		Normal	0.149	0.199
Ellipse	Autonomous Geometric	EKF	0.038	0.046
		Normal	0.169	0.239
Line	Autonomous Geometric	EKF	0.019	0.040
		Normal	0.037	0.056

In the following analysis, we present a comprehensive table that summarizes the key metrics of drone performance in autonomous operations. This table focuses on two critical aspects: Peak deviations and RMSE (Root Mean Square Error), which are crucial for assessing the accuracy and reliability of drone navigation. These metrics are evaluated under

two distinct flight modes: Autonomous Waypoint Navigation and Autonomous Geometric Flight Modes. The mean and standard deviation for each mode provides insights into the consistency and precision of the drones under varying operational conditions.

Table 7.10. Mean and standard deviations of drone navigation in autonomous waypoint navigation and autonomous geometric flight modes.

Flight Mode	Metric	Sensor Data Type	Mean (m)	Standard Deviation (m)
Autonomous Waypoint Navigation	RMSE	EKF	0.165	0.052
		Normal	0.210	0.061
	Peak Deviation	EKF	0.284	0.092
		Normal	0.326	0.110
Autonomous Geometric Flight Modes	RMSE	EKF	0.085	0.056
		Normal	0.134	0.060
	Peak Deviation	EKF	0.155	0.127
		Normal	0.207	0.131

8. TEST RESULTS AND DISCUSSION

In our preliminary setup for drone assembly, the careful alignment of the rotors onto the drone arms was a critical factor. Ensuring the rotors' flatness is paramount for optimal flight performance, as any tilt or deviation from the intended angle can cause imbalances during flight, leading to inaccurate results or crashes. To establish the precise leveling of the rotors, we employed the use of a water level ruler. This highly sensitive tool allowed us to ensure that each rotor was perfectly horizontal, and thus, properly set for the tests. This meticulous approach not only bolstered the reliability of our results but also laid a strong foundation for any subsequent testing phases. The drone's performance in flight following this calibration process will be a testament to the accuracy and effectiveness of this method.

To further evaluate the lifting capability of the drone's rotors, a standalone test setup was also created. An identical rotor and electronic speed controller (ESC), mirroring the components used in the drone assembly, were connected. This process involved carefully soldering the wires of two components together. The propeller was affixed in a reversed position for this test. To ensure a safe clearance between the rotor and propeller, two washers were used as spacers. This arrangement was designed to reflect the downward force exerted by the rotor when in operation, simulating the lift that each rotor could generate. After placing the assembly onto weighing scale, total weight was set to zero.

Figure 8.1. Experimental setup to measure lift generated by each rotor at full throttle.

The assembly was then placed on a weighing scale, with the expectation that the downward thrust created by the spinning propeller would register as an increase in weight, effectively indicating the lifting capacity of each rotor. The tests were conducted with two different propellers, one shorter and the other longer. The results from these tests provided valuable insight into the influence of propeller length on the overall lifting capacity of the rotor, crucial information for optimizing our drone's flight performance.

Table 8.1. Variation of lift values according to propeller type.

No.	Weight	Length	Pitch	Center Hole Diameter	Lift (Each)
1	12 g	30,48 cm (12 in.)	12,97 cm (5,5 in.)	4 mm	1,0 kg
2	20 g	36,00 cm (14 in.)	12,97 cm (5,5 in.)	4 mm	1,5 kg

The Sunnysky x4110s KV:400 rotor was chosen for our quadcopter after careful consideration of its specifications and our requirements. Having a drone of 3.2 kg requires bigger motors, in other words a low KV motor is needed. Since the drone's total weight is approximately 3.2 kg, it falls well within the rotor's recommended lift weight for a 4-axis configuration. This ensures optimal performance and thrust-to-weight ratio for our application. The rotor's max thrust of 3720g further assures that each rotor can provide more than enough lift for the quadcopter, granting it stability and agility during maneuvers.

Table 8.2. Specifications and recommended lift weights for the selected rotor.

Parameter	Value
Rotor model	Sunnysky x4110s KV:400
Weight of rotor	165g
Max thrust	3720g
Recommended lift (4-axis)	3.0 - 4.8kg
Recommended lift (6-axis)	4.8 - 7.2kg
Recommended lift (8-axis)	6.0 - 10.0kg
Drone weight	3.2kg approx.

Figure 8.2. Experimental setup for measuring current correlation with different lift values.

In the experimental setup, a consistent voltage of 16.3 Volts, analogous to the output of a standard 4S LiPo battery, was supplied using a power source to simulate real-world conditions for UAV operations. The rotor RPMs were modulated via a radio controller, interfacing with a Pixhawk flight controller that translated these manual inputs into commands for the electronic speed controller (ESC). The ESC, in turn, regulated the single rotor mounted on the scale. The propeller was affixed to the rotor in reverse orientation to ensure that the thrust generated could be directly measured as a downward force, allowing for precise quantification in gram-force (gf) based on the varying current inputs. This configuration facilitated the collection of data to determine the correlation between current supplied and lift produced, under the controlled voltage scenario, providing critical insights into the efficiency and performance of the propulsion system under test. The data demonstrates a progressive increase in thrust as the current increases. For instance, at a low current of 0.3 A, the thrust produced is 24 gf, whereas at a higher current of 4.8 A, the thrust significantly increases to 663 gf. This trend suggests a direct relationship between the current supplied and the thrust generated, highlighting the efficiency and response of the propulsion system under the specified voltage condition.

Table 8.3. Variation of lift values according to the given current under 16.3 volts.

No.	Amps (A) vs. Thrust (gf) under 16.3 Volts		No.	Amps (A) vs. Thrust (gf) under 16.3 Volts	
	Amps (A)	Thrust (gf)		Amps (A)	Thrust (gf)
1	0.3	24	24	2.6	413
3	0.5	70	26	2.8	440
5	0.7	111	28	3.0	467
7	0.9	145	30	3.2	487
9	1.1	185	32	3.4	515
11	1.3	220	34	3.6	544
13	1.5	248	36	3.8	561
15	1.7	285	38	4.0	582
17	1.9	315	40	4.2	606
19	2.1	340	42	4.4	623
21	2.3	377	44	4.6	650
23	2.5	400	46	4.8	663

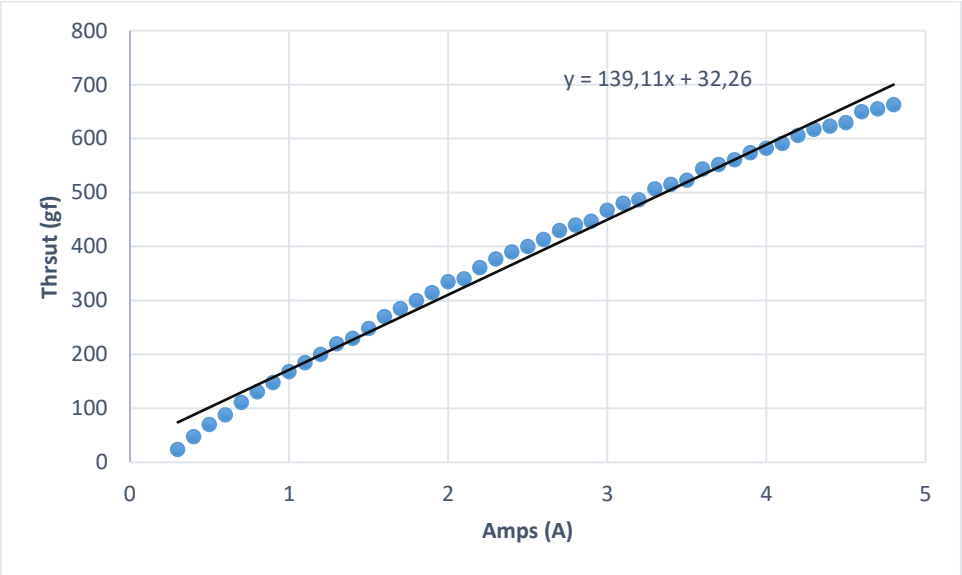

Figure 8.3. Amps (A) vs. Thrust (gf) under 16.3 Volts.

In our case, we leverage the odometry data from the ZED Mini camera for the Extended Kalman Filter (EKF). To achieve this, we utilize the parameters provided in the

“common.yaml” file located in the 'params' directory of the “zed-wrapper” folder in the “zed-ros-wrapper”. The configuration file designates “quad1/camera_link”, a component we defined in our drone's URDF, as the “base_frame”. We then define “map_frame” as “quad1/map” and “odometry_frame” as “quad1/odom”. Enabling the “positional tracking from start” feature allows us to maintain the origin of the global frame at the start point of the vehicle's movement, even if the vehicle stops and restarts, on the RViz visualization tool. We make modifications in our launch file, “zedm.launch”, where we set “base_frame” as “quad1/base_link” as set in the URDF. We input the X, Y, Z positions and orientation of the camera with respect to the “base_frame”. We source these values from the URDF file generated from the drone model we built in SolidWorks. Given that the camera is placed facing downward, we set the pitch angle orientation to the equivalent of pi/2, which is 1.5708. This launch file is used when we run the camera. We also write a node named “quad_tf_to_odom”, which enables the necessary position and orientation transformation between “quad1/base_link” and “quad1/odom' data”. To transfer odometry data according to the necessary position and orientation to “quad1/map”, we write the “quad_tf_odom_to_map” node. Furthermore, we write two more nodes named “quad_zed_odom_tf” and “quad_zed_imu_tf”. In “quad_zed_odom_tf”, we transform the ZED's odometry information into the quadcopter's odometry information, and in “quad_zed_imu_tf”, we transform the ZED's IMU information into the quadcopter's IMU information. Lastly, we configure the Fixed Frame and TF Prefix settings on RViz and add the image topic of the ZED mini camera to obtain the camera view on RViz. We also define the IMU, Odometry, and axes information on RViz.

The ZED mini camera is used to obtain position information for the X and Y coordinates, while data from the Garmin optical sensor will be used to acquire altitude information in the Z direction. Configuration matrices are fine-tuned to optimize the ZED mini camera's performance, minimizing the discrepancies between the estimated state and the actual state by adjusting for the sensor's inherent noise and biases. The ZED mini camera is equipped with an intelligent algorithm that autonomously configures its covariance matrices, adapting to environmental variables and sensor noise levels to optimize performance.

Table 8.4. Configuration matrices for odometry data, pose data, and IMU data.

Parameters	odom0_config	pose0_config	imu0_config
X position	True	True	False
Y position	True	True	False
Z position	False	False	False
Roll orientation	False	False	True
Pitch orientation	False	False	True
Yaw orientation	True	True	True
X velocity	False	False	False
Y velocity	False	False	False
Z velocity	False	False	False
Roll angular velocity	False	False	True
Pitch angular velocity	False	False	True
Yaw angular velocity	False	False	True
X acceleration	False	False	True
Y acceleration	False	False	True
Z acceleration	False	False	True

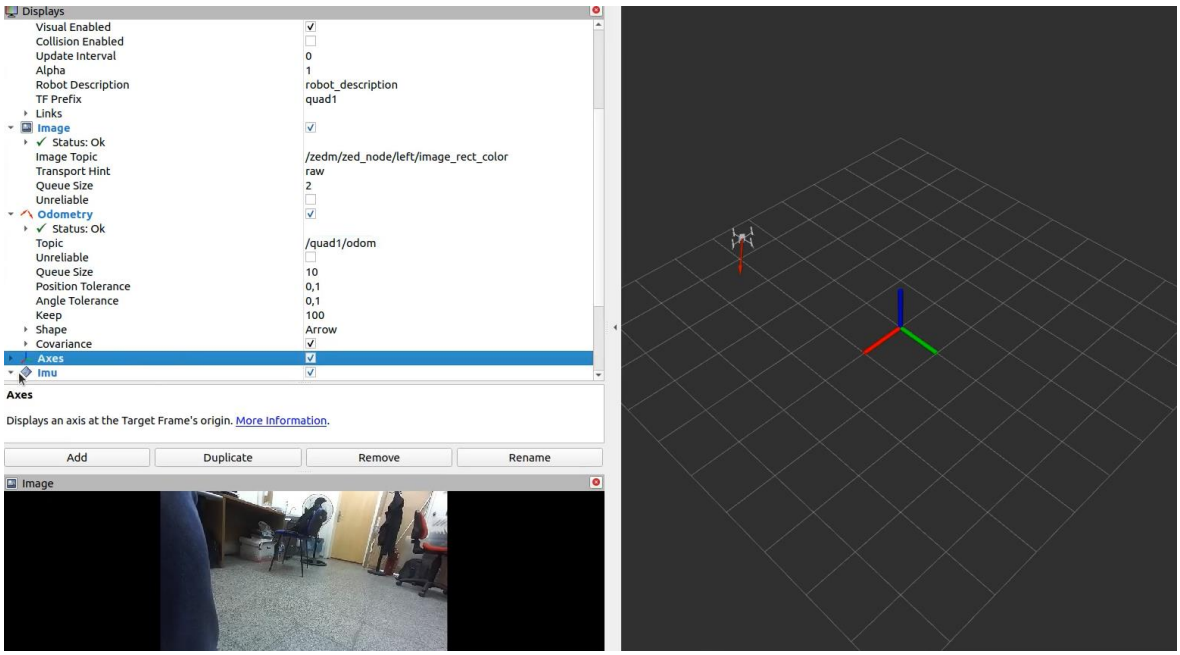

Figure 8.4. Image view of ZED mini camera on RViz.

In the process of implementing an Extended Kalman Filter for sensor fusion in our robotics system, we employ the ZED mini camera as a key sensor, exploiting its built-in accelerometer, gyroscope, and Inertial Measurement Unit. The ZED mini camera provides orientation information in the form of quaternions, a mathematical construct particularly useful for representing orientations and rotations of objects in three dimensions. However, these quaternions are then subjected to transformations in order to translate them into yaw, pitch, and roll across various coordinate frames. This step is imperative to maintaining consistency in the representation of data across different sensors, as the EKF requires a common data format to yield accurate results. During the data preparation phase, message types like Float64 are appropriately defined and utilized to ensure the smooth transmission of information into the EKF. The process of configuring the EKF revolves around managing various sensors and determining their relationships with the data they produce. To elaborate, the odometry data provided by the ZED mini camera is treated as a relative change in both position and orientation, the pose data as an absolute measure of position and orientation, while the data from the IMU is interpreted as a relative change.

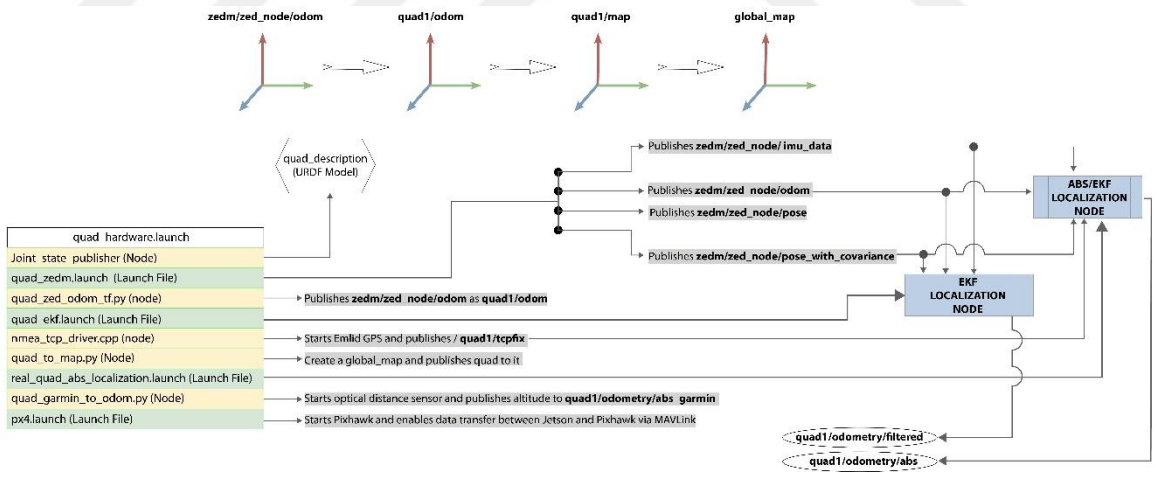

Figure 8.5. Overview of the Extended Kalman Filter implementation.

Setting up the correct configuration matrices forms a crucial part of this procedure. These matrices guide the extraction of accurate sensor information and are instrumental in their fusion within the EKF. However, defining the covariance matrices, which is a key component of the Kalman filtering process, is not handled manually. Instead, the EKF automatically integrates the covariance data derived from the ZED mini camera, thus ensuring optimal performance and reducing the risk of human-induced error. Overall, the

sensor fusion implemented using the EKF and the ZED mini camera enables our robotic system to effectively synthesize diverse sensor data, allowing it to navigate its environment with enhanced precision and reliability.

In Figure 8.5, the network of nodes integral to the operation of the Extended Kalman Filter is illustrated. Various sensors, including the ZED mini camera, the optical distance sensor by Garmin, and the Emlid GPS, are activated by the corresponding launch files and nodes. Initialization of the Pixhawk drone controller also depicted. The transformation of coordinate frames is demonstrated. Odometry data sourced from the ZED mini camera is first transformed to the drone's “base_link” frame. Subsequently, it is transformed to a dedicated map frame for the drone and then to a global map frame. Two EKF localization nodes are employed. The first one is fed by sensors included in the ZED mini camera, while the second is augmented with data from the GPS. Superior odometry results are expected to be produced by the latter due to the broader sensor input.

Figure 8.6. Developed drone controlled via Radio Controller.

The drone developed for this research was primarily controlled using a radio controller. During its flights, several external disturbances were encountered, potentially compromising the robustness of the radio control. Nevertheless, despite these challenges, the drone managed to hover and land safely.

To evaluate the performance of the Extended Kalman Filter (EKF), it was initially intended for the drone to undertake autonomous flight. However, this proved to be a complex endeavor due to the intricate nature of the control algorithms and trajectory-following tasks. The risk associated with ensuring control during autonomous flight rendered it an unsafe approach.

To address this limitation and still achieve data collection goals, another method was employed. Drone was controlled manually. The UGV was tasked with following a predefined trajectory assigned by the drone. As the UGV navigated its course, it collected sensor data and processed it through the EKF. This setup provided a valuable benchmark. The UAV's trajectory served as a reference against which the position data derived from the UGV's EKF could be compared.

Figure 8.7. Drone mounted on the UGV.

Moreover, for a comprehensive analysis, two sets of EKF results were charted. The first derived from the drone's EKF using both the ZED mini camera and GPS data, while the second omitted GPS data. These comparative graphs effectively showcased the performance

distinction between the two configurations, offering insights into the enhanced accuracy potentially achieved with the incorporation of GPS.

In the test, a straight line measuring 9 meters in length was marked on the field to establish the trajectory for the unmanned ground vehicle. The UAV, equipped with a ZED Mini camera that faces downwards, is designed primarily for aerial surveillance and data acquisition. Due to the risks associated with autonomous flight, the UAV is mounted securely on the UGV, enhancing safety and ensuring continuous operation in challenging environments. The UGV, outfitted with a forward-oriented ZED 2 camera, compensates for the limited field of view of the UAV's downward-facing camera by providing essential forward positional information. Sensor fusion techniques are employed, integrating data from the Emlid GPS and the ZED 2 camera. 9 meters line was instrumental in defining the start and end points of the vehicle's path. The vehicle was autonomously controlled and followed this precisely marked line. During its journey, the ZED 2 camera and GPS were actively engaged in data collection. The data collection commenced in ROS with the drone positioned at the starting point of the line. The drone then autonomously traveled the entire 9 meters along the x-axis.

Figure 8.8. Field and experimental setup used for the test.

The gathered data unveiled findings that aligned closely with our expectations, bearing minor discrepancies. Notably, the EKF data, which was created by fusing both GPS readings and ZED 2 camera sensor inputs, showcased a superior performance than the EKF data derived solely from the ZED 2 camera sensors. The deviation observed can be attributed to variations in GPS accuracy. Ideally, the GPS system should operate in “fixed mode,” a specific mode demonstrating enhanced accuracy. However, due to the positioning of the GPS base and rover modules, as well as the presence of magnetic and physical interferences, rover module operated in “float mode”. Nevertheless, even under these conditions, the performance of the fusion with GPS data was superior to the fusion relying solely on the accelerometer and gyroscope. It is important to contextualize the data presented. The figures and tables that follow are key to interpreting the experiment's findings. A comparative analysis was conducted by selecting five critical points from both the EKF processed dataset and the EKF processed data which also used GPS, to assess the deviation from the assigned trajectory. The chosen data points, presented in a tabular format, provide the X and Y coordinates to highlight the positional discrepancies. This comparison offers a clear evaluation of the path tracking performance, illustrating the differences between using EKF processed data alone and EKF processed data enhanced with GPS inputs.

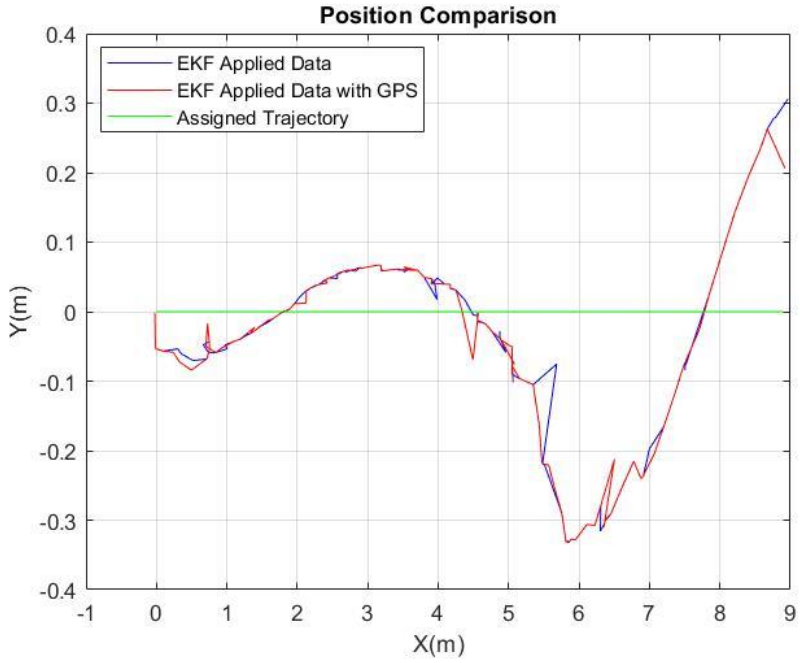

Figure 8.9. Position estimation of trial 1 with and without GPS while drone is tracing a line.

Table 8.5. Comparison of EKF-processed position data with and without GPS collected while drone is tracing first line trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	0.500	0.500	0.500
	Y	-0.084	-0.069	0.000
2	X	3.186	3.186	3.186
	Y	0.067	0.066	0.000
3	X	5.600	5.600	5.600
	Y	-0.245	-0.235	0.000
4	X	5.840	5.840	5.840
	Y	-0.330	-0.332	0.000
5	X	8.900	8.900	8.900
	Y	0.306	0.206	0.000

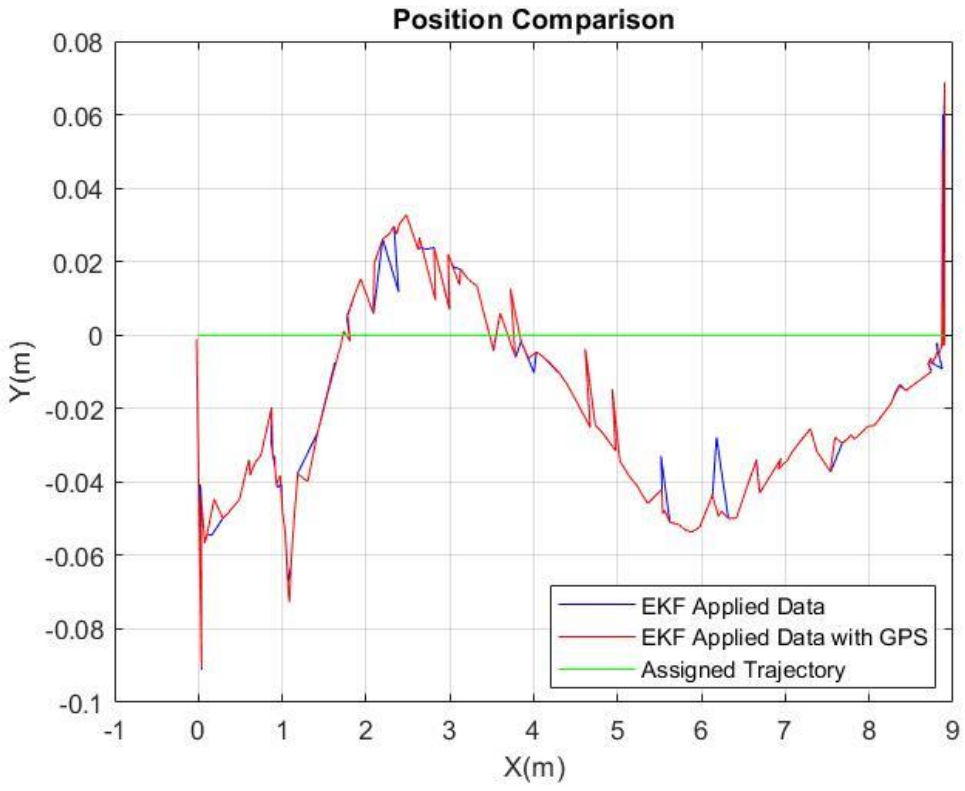

Figure 8.10. Position estimation of trial 2 with and without GPS while drone is tracing a line.

Table 8.6. Comparison of EKF-processed position data with and without GPS collected while drone is tracing second line trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	0.038	0.038	0.038
	Y	-0.091	-0.090	0.000
2	X	1.080	1.080	1.080
	Y	-0.066	-0.073	0.000
3	X	2.333	2.333	2.333
	Y	0.030	0.029	0.000
4	X	6.200	6.200	6.200
	Y	-0.027	-0.049	0.000
5	X	8.900	8.900	8.900
	Y	0.068	0.069	0.000

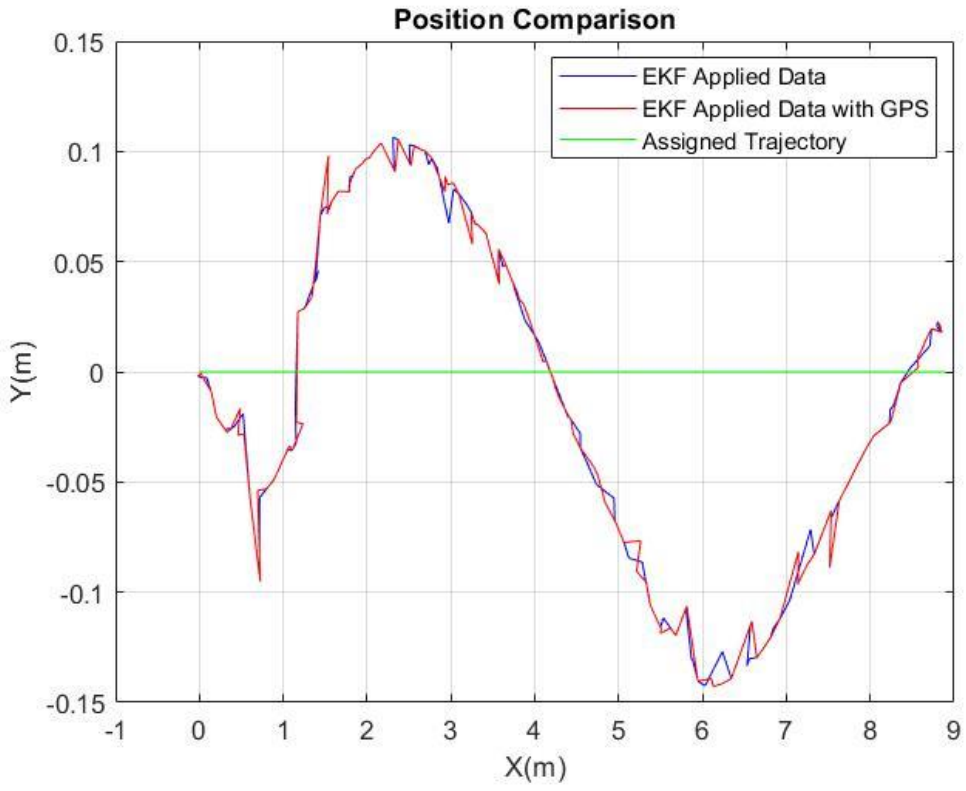

Figure 8.11. Position estimation of trial 3 with and without GPS while drone is tracing a line.

Table 8.7. Comparison of EKF-processed position data with and without GPS collected while drone is tracing third line trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	0.072	0.072	0.072
	Y	-0.092	-0.095	0.000
2	X	2.334	2.334	2.334
	Y	0.107	0.091	0.000
3	X	6.030	6.030	6.030
	Y	-0.142	-0.139	0.000
4	X	7.510	7.510	7.510
	Y	-0.066	-0.089	0.000
5	X	8.800	8.800	8.800
	Y	0.023	0.022	0.000

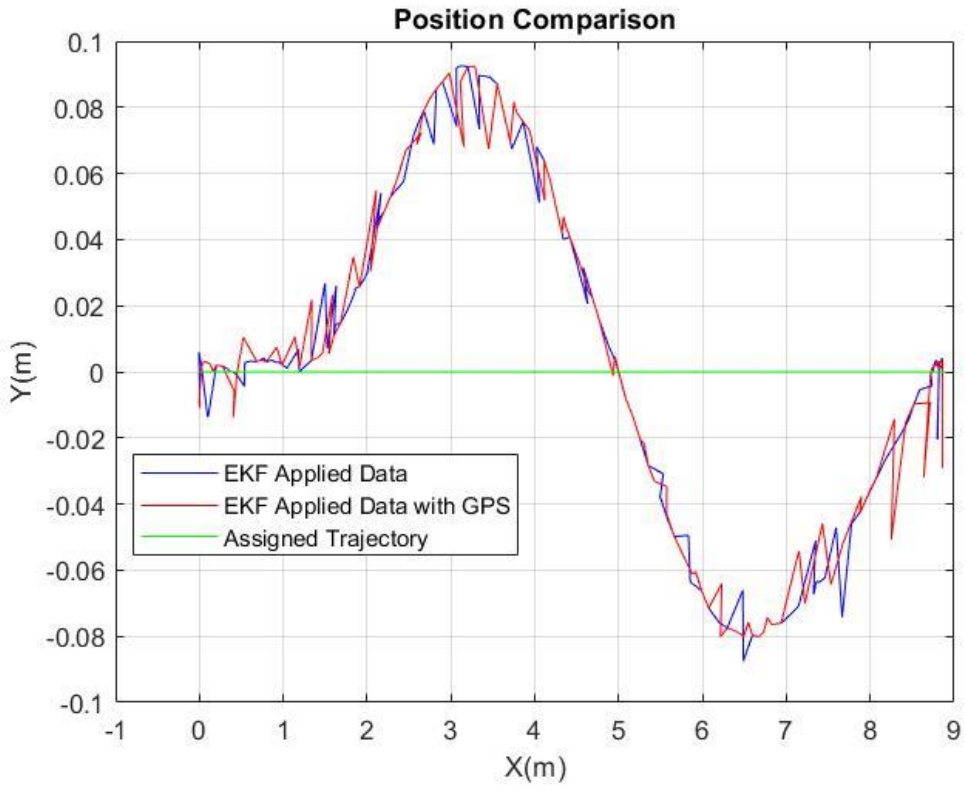

Figure 8.12. Position estimation of trial 4 with and without GPS while drone is tracing a line.

Table 8.8. Comparison of EKF-processed position data with and without GPS collected while drone is tracing fourth line trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	2.100	2.100	2.100
	Y	0.054	0.030	0.000
2	X	3.200	3.200	3.200
	Y	0.093	0.092	0.000
3	X	4.340	4.340	4.340
	Y	0.040	0.047	0.000
4	X	6.500	6.500	6.500
	Y	-0.088	-0.080	0.000
5	X	8.250	8.250	8.250
	Y	-0.026	-0.051	0.000

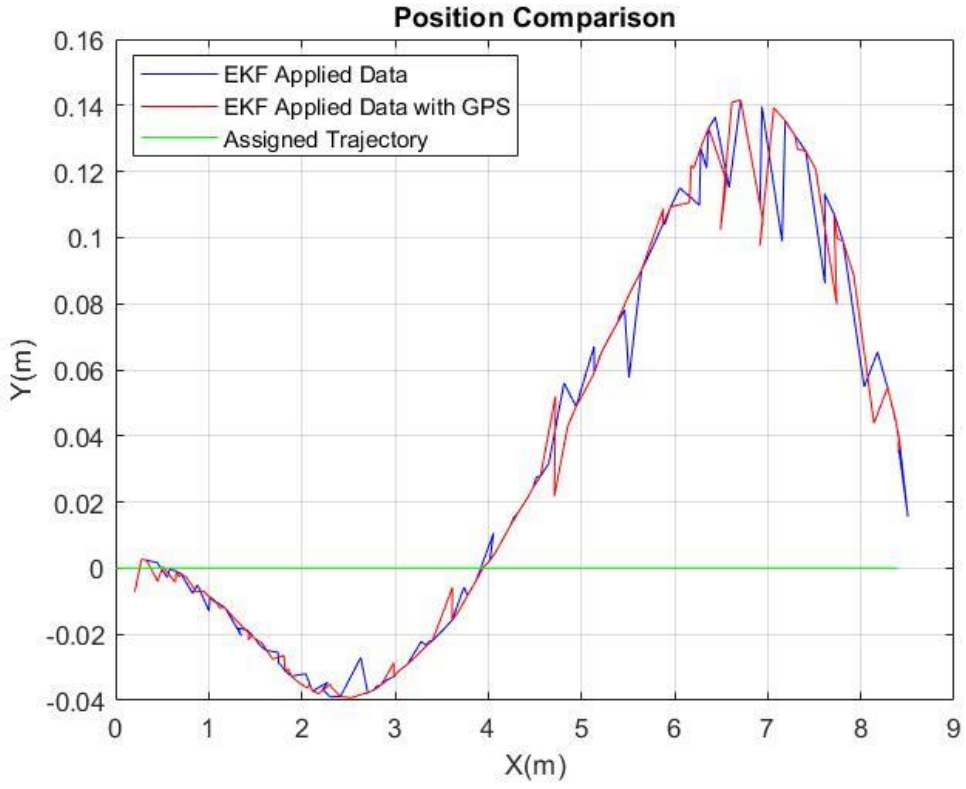

Figure 8.13. Position estimation of trial 5 with and without GPS while drone is tracing a line.

Table 8.9. Comparison of EKF-processed position data with and without GPS collected while drone is tracing fifth line trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	2.630	2.630	2.630
	Y	-0.027	-0.038	0.000
2	X	4.800	4.800	4.800
	Y	0.056	0.043	0.000
3	X	6.900	6.900	6.900
	Y	0.140	0.098	0.000
4	X	7.100	7.100	7.100
	Y	0.098	0.139	0.000
5	X	7.700	7.700	7.700
	Y	0.107	0.080	0.000

In the subsequent experiment, the robot was guided along a U-shaped trajectory. This path was designed to assess the filter and sensors' performance during directional changes. Initially, the trajectory led the robot in one direction, then it made a turn and followed a parallel path in the opposite direction, effectively creating a U-shaped movement pattern.

Notable deviations were observed during the turning points, which are believed to be linked to the PID control settings for lateral and longitudinal orientation. These settings led to minor inconsistencies in the trajectory during the sensor data recording phase. The analysis effectively illustrates the variance in path tracking performance between using EKF processed data on its own and when enhanced with GPS inputs. A thorough analysis was performed by selecting key points for comparison from datasets processed with the Extended Kalman Filter (EKF), both with and without the integration of GPS. This analysis aimed to determine variations from the planned route. The selected points are detailed in a table, showing the X and Y coordinates to pinpoint where positional variances occurred. Such a side-by-side comparison provides an insightful look into how the path adherence varies when comparing the EKF-processed data in isolation to that supplemented with GPS data.

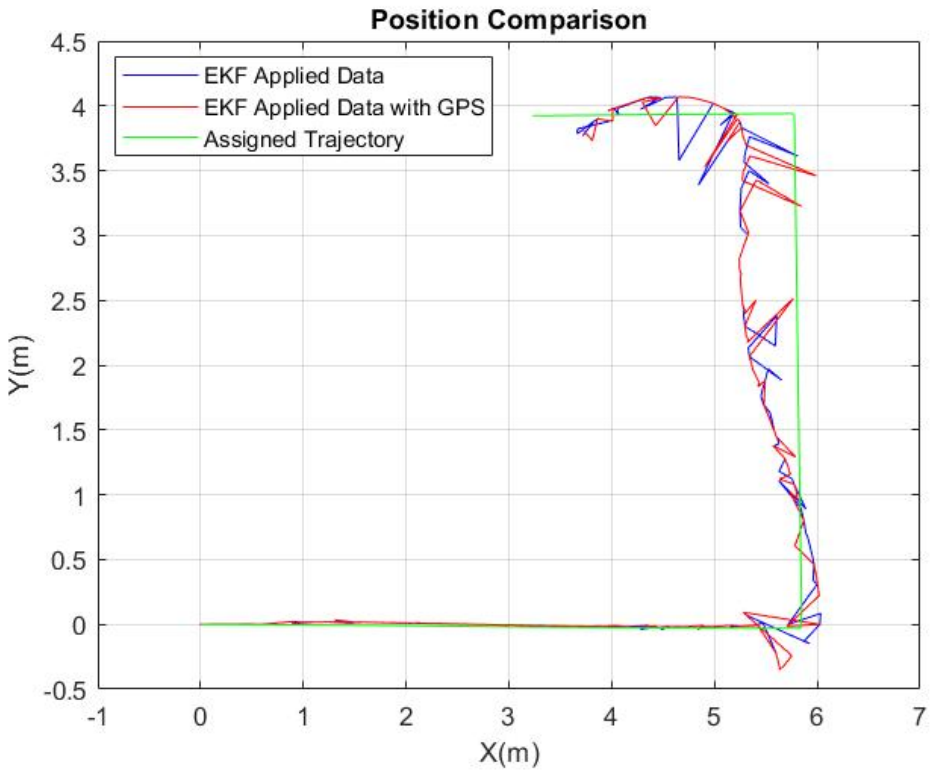

Figure 8.14. Position estimation of trial 1 with and without GPS while drone is tracing a U−shaped figure.

Table 8.10. Comparison of EKF-processed position data with and without GPS collected while drone is tracing first U-shaped trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	4.500	4.500	4.500
	Y	-0.016	-0.019	-0.027
2	X	5.327	5.329	5.809
	Y	2.100	2.100	2.100
3	X	5.254	5.331	5.793
	Y	3.000	3.000	3.000
4	X	5.288	5.279	5.785
	Y	3.500	3.500	3.500
5	X	4.300	4.300	4.300
	Y	4.068	4.061	3.931

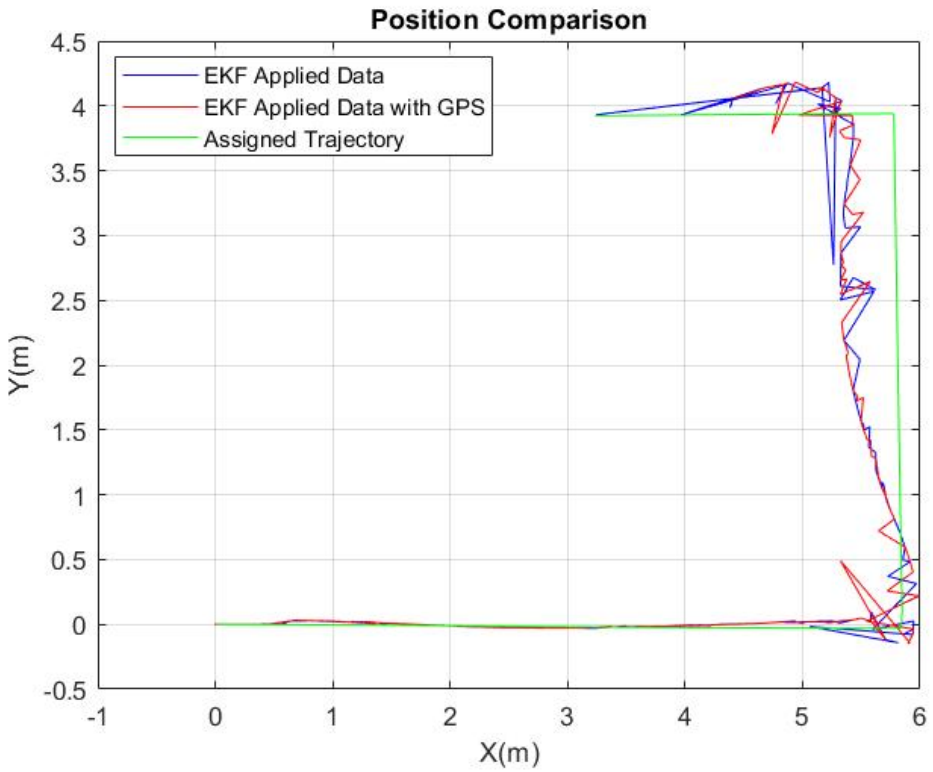

Figure 8.15. Position estimation of trial 2 with and without GPS while drone is tracing a U−shaped figure.

Table 8.11. Comparison of EKF-processed position data with and without GPS collected while drone is tracing second U-shaped trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	3.200	3.200	3.200
	Y	-0.030	-0.023	-0.020
2	X	5.727	5.943	5.850
	Y	0.400	0.400	0.400
3	X	5.325	5.375	5.799
	Y	2.700	2.700	2.700
4	X	5.350	5.520	5.789
	Y	3.200	3.200	3.200
5	X	4.800	4.800	4.800
	Y	4.167	4.176	3.940

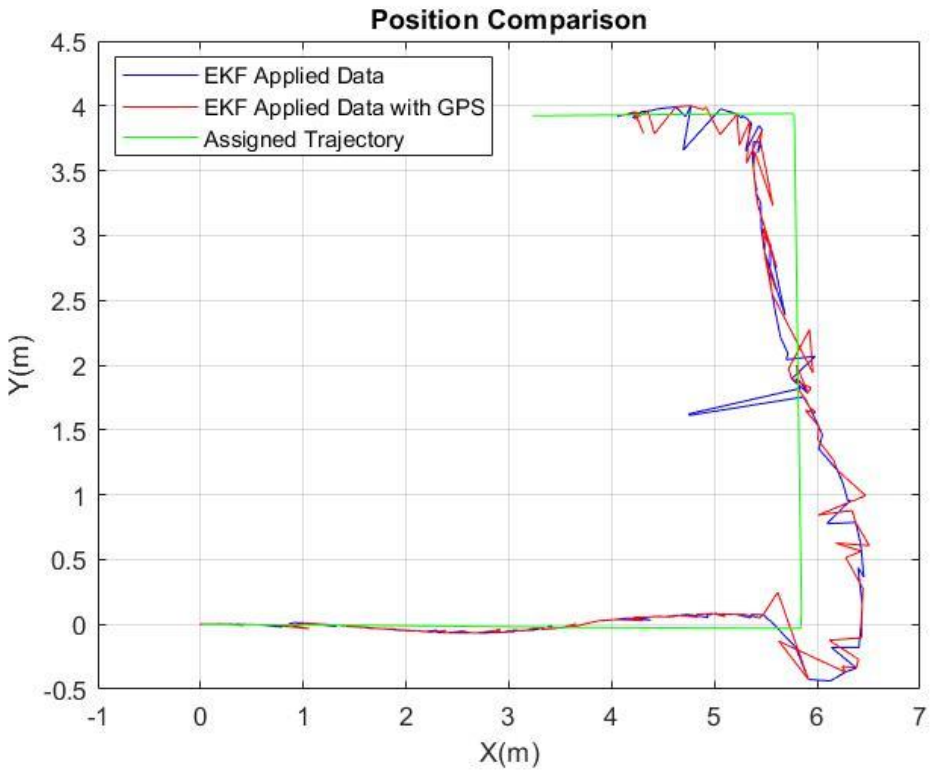

Figure 8.16. Position estimation of trial 3 with and without GPS while drone is tracing a U−shaped figure.

Table 8.12. Comparison of EKF-processed position data with and without GPS collected while drone is tracing third U-shaped trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	2.550	2.550	2.550
	Y	-0.066	-0.051	-0.017
2	X	6.438	6.277	5.840
	Y	0.500	0.500	0.500
3	X	6.300	6.470	5.830
	Y	1.000	1.000	1.000
4	X	5.449	5.422	5.789
	Y	3.200	3.200	3.200
5	X	4.400	4.400	4.400
	Y	3.971	3.785	3.931

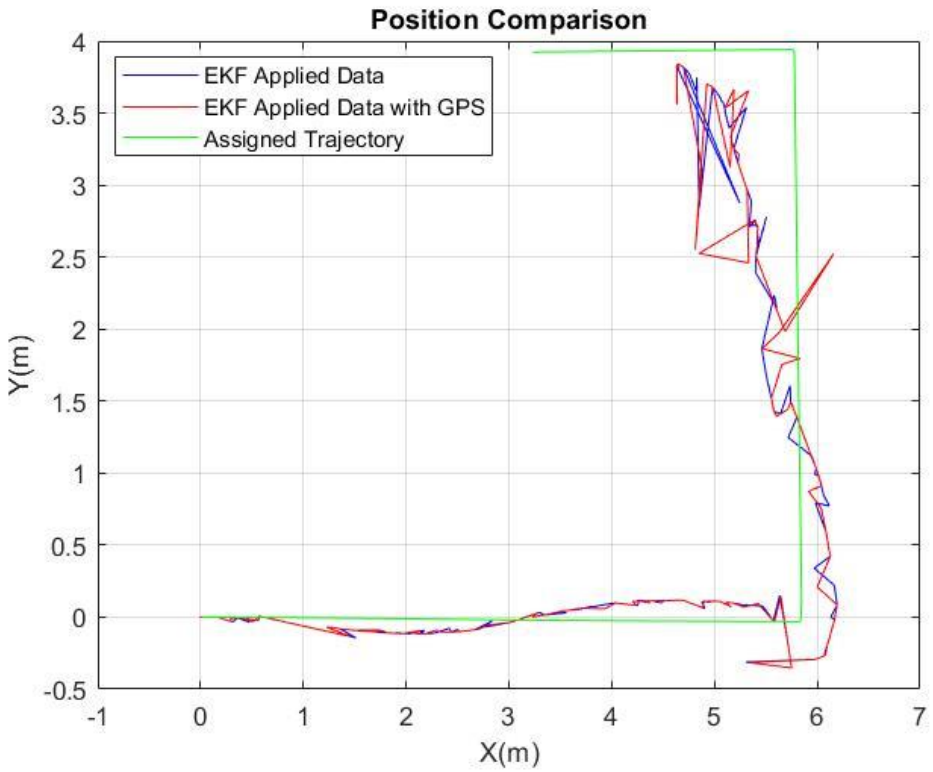

Figure 8.17. Position estimation of trial 4 with and without GPS while drone is tracing a U−shaped figure.

Table 8.13. Comparison of EKF-processed position data with and without GPS collected while drone is tracing fourth U-shaped trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	2.270	2.270	2.270
	Y	-0.111	-0.089	-0.014
2	X	4.900	4.900	4.900
	Y	0.107	0.114	-0.028
3	X	6.167	5.999	5.850
	Y	0.210	0.210	0.210
4	X	5.358	5.318	5.793
	Y	2.900	2.900	2.900
5	X	5.188	5.238	5.789
	Y	3.200	3.200	3.200

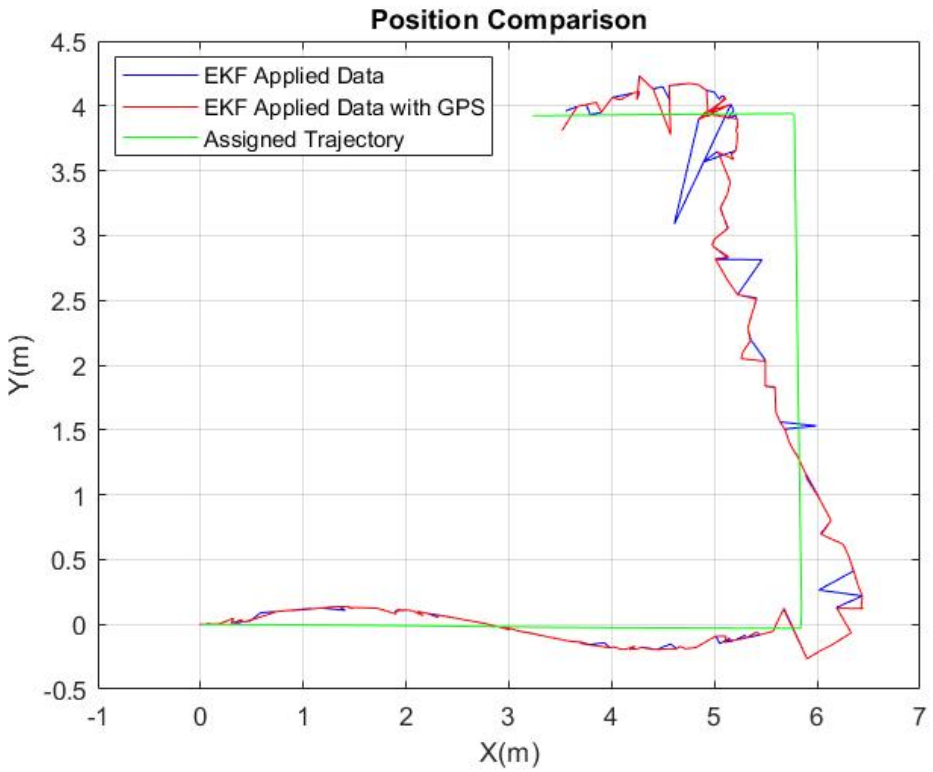

Figure 8.18. Position estimation of trial 5 with and without GPS while drone is tracing a U−shaped figure.

Table 8.14. Comparison of EKF-processed position data with and without GPS collected while drone is tracing fifth U-shaped trial.

Point Nr.	Coordinate Axes (m)	EKF	EKF with GPS	Assigned
1	X	1.410	1.410	1.410
	Y	0.108	0.137	-0.001
2	X	4.570	4.570	4.570
	Y	-0.151	-0.190	-0.027
3	X	6.016	6.399	5.850
	Y	0.280	0.280	0.280
4	X	5.496	5.259	5.815
	Y	2.050	2.050	2.050
5	X	4.600	4.600	4.600
	Y	3.090	3.770	3.930

Nevertheless, these deviations, which were confined within a 0.4-meter range, were deemed minimal. It must be noted that these results were obtained even though the GPS was found to be operating in float mode. Overall, the findings from this experiment were considered both consistent and promising for future endeavors.

Table 8.15. Peak deviations and RMSE analysis of experimental test results.

Shape	Test No.	Sensor Data Type	Root Mean Square Error (m)	Peak Deviation (m)
Line	1	EKF	0.234	0.330
		EKF with GPS	0.208	0.332
	2	EKF	0.066	0.091
		EKF with GPS	0.062	0.090
	3	EKF	0.095	0.142
		EKF with GPS	0.072	0.139
	4	EKF	0.066	0.093
		EKF with GPS	0.064	0.092
	5	EKF	0.094	0.140
		EKF with GPS	0.088	0.139
U-shaped	1	EKF	0.397	0.539
		EKF with GPS	0.379	0.520
	2	EKF	0.311	0.474
		EKF with GPS	0.252	0.424
	3	EKF	0.374	0.598
		EKF with GPS	0.390	0.640
	4	EKF	0.368	0.601
		EKF with GPS	0.340	0.551
	5	EKF	0.415	0.840
		EKF with GPS	0.368	0.553

In Table 8.16, we delve into the essential metrics that define drone performance in automated tasks. It concentrates on two main elements: Peak Deviations and RMSE, which

are key to determining the accuracy and dependability of drone navigation. These parameters are examined across two distinct navigation tests: line trajectory and u-shaped trajectory. By presenting the average and standard deviation for each test type, we gain a clear understanding of the drones' accuracy and uniformity in these autonomous navigation scenarios.

Table 8.16. Mean and standard deviations calculated for peak deviations and RMSE analysis of experimental test results.

Shape	Metric	Sensor Data Type	Mean (m)	Standard Deviation (m)
Line	RMSE	EKF	0.111	0.070
		EKF with GPS	0.099	0.062
	Peak Deviation	EKF	0.159	0.099
		EKF with GPS	0.158	0.100
U-shaped	RMSE	EKF	0.373	0.039
		EKF with GPS	0.346	0.056
	Peak Deviation	EKF	0.610	0.138
		EKF with GPS	0.538	0.078

9. CONCLUSION

This study describes the design of a drone, which incorporates a range of tools and sensors, including the ZED mini camera, Garmin optical distance sensor, and Emlid GPS. This was not just an ordinary drone. The quadrotor is characterized by its four rotors. One objective is to use the Extended Kalman Filter to interpret data captured by the sensors.

For testing purposes, the tools RViz, Gazebo, and ROS were used. The URDF model was created using Solidworks and integrated into ROS. The model's availability proved invaluable as it is used in simulations that reflect potential real-world operations.

Understanding and implementing the Extended Kalman Filter (EKF) is crucial. Detailed investigations were conducted into the theory, with mathematical explanations supporting its operations. The advantages of EKF over the standard Kalman Filter were highlighted.

For the hardware aspect, detailed discussions about the components necessary for building a drone from the ground up were provided. The component analysis is considered essential for anyone embarking on their drone creation journey. Furthermore, the drone's design is suitable for flight using a radio controller. That verifies the technical aspects of the design.

The planned path of the drone, or trajectory, was another focus of the study. This thesis addressed methods for describing particular movements and patterns of drones. In the first simulation scenario, a UGV was used. This was important for cooperative behaviour. The UGV determined the waypoints, which were then followed by the drone. In another simulation scenario, the drone executed geometric shapes. The use of the Extended Kalman Filter to combine sensor information in simulations resulted in a significant improvement in odometry accuracy. In real-world applications, the performance of the EKF was further enhanced by incorporating GPS data.

In the experiments conducted, the accuracy of the UGV's trajectory was assessed by measuring the displacement of a rear wheel. The purpose of the measurements was to validate whether the assigned trajectory was compatible with what the UGV was following as a path. It should be noted, however, that the experiments were conducted in a different environment from the one in which the initial adjustments were made. This change in location could have caused minor deviations in the measurements, highlighting the importance of environmental factors in the interpretation of the results.

However, discrepancies were noted. Minor discrepancies were found between real-world tests and computer simulations. Such deviations are expected due to the inherent unpredictability of real-world conditions compared to controlled simulations. Possible reasons for the issue may vary from inaccuracies in the sensor readings to unanticipated external factors. The simulation results showed a slight improvement over the experimental outcomes, but the differences were not significant. The results consistently showed that implementing EKF with GPS produced better outcomes in object-free environments compared to EKF without GPS, confirming the GPS's ability to enhance navigational precision.

In conclusion, this study provides valuable insights into the integration of sensors and filters, particularly the EKF, in drone development. This study contributes to the advancement of future drone technologies. Real-world experiments can be conducted by achieving autonomous flight. This provides an opportunity to evaluate how the performance of the Extended Kalman Filter (EKF) changes over different drone configurations.

REFERENCES

1. Islam, N., M. M. Rashid, F. Pasandideh, B. Ray, S. Moore and R. Kadel, "A Review of Applications and Communication Technologies for Internet of Things and Unmanned Aerial Vehicle Based Sustainable Smart Farming", *Sustainability*, Vol. 13, No. 4, p. 1821, 2021.

2. Muchiri, G. and S. Kimathi, "A Review of Applications and Potential Applications of UAV", *Sustainable Research and Innovation Conference*, Nairobi, Kenya, pp. 280-283, 2022.

3. Matese, A. and S. F. Di Gennaro, "Practical Applications of a Multisensor UAV Platform Based on Multispectral, Thermal and RGB High Resolution Images in Precision Viticulture", *Agriculture*, Vol. 8, No. 7, p. 116, 2018.

4. Salamí, E., C. Barrado and E. Pastor, "UAV Flight Experiments Applied to the Remote Sensing of Vegetated Areas", *Remote Sensing*, Vol. 6, No. 11, pp. 11051-11081, 2014.

5. Desai, A., T. Dreossi and S. A. Seshia, "Combining Model Checking and Runtime Verification for Safe Robotics", *Runtime Verification: 17th International Conference*, Seattle, USA, pp. 172-189, 2017.

6. Lamping, A. P., J. N. Ouwerkerk and K. Cohen, "Multi-UAV Control and Supervision with ROS", *Aviation Technology, Integration, and Operations Conference*, Georgia, USA, p. 4245, 2018.

7. Garcia, S., M. E. López, R. Barea, L. M. Bergasa, A. Gómez and E. J. Molinos, "Indoor SLAM for Micro Aerial Vehicles Control Using Monocular Camera and Sensor Fusion", *International Conference on Autonomous Robot Systems and Competitions*, Bragança, Portugal, pp. 205-210, 2016.

8. Huang, R., P. Tan and B. M. Chen, “Monocular Vision-Bases Autonomous Navigation System on a Toy Quadcopter in Unknown Environments”, *International Conference on Unmanned Aircraft Systems*, Denver, USA, pp. 1260-1269, 2015.

9. Feng, L. and Q. Fangchao, “Research on the Hardware Structure Characteristics and EKF Filtering Algorithm of the Autopilot PIXHAWK”, *Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control*, Harbin, China, pp. 228-231, 2016.

10. Bugallo, M. F., S. Xu and P. M. Djurić, “Performance Comparison of EKF and Particle Filtering Methods for Maneuvering Targets”, *Digital Signal Processing*, Vol. 17, No. 4, pp. 774-786, 2007.

11. Minaeian, S., J. Liu and Y. J. Son, “Vision-Based Target Detection and Localization via a Team of Cooperative UAV and UGVs”, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 46, No. 7, pp. 1005-1016, 2015.

12. Kurazume, R., S. Nagata and S. Hirose, “Cooperative Positioning with Multiple Robots”, *IEEE International Conference on Robotics and Automation*, San Diego, USA, pp. 1250-1257, 1994.

13. Rekleitis, I., G. Dudek and E. Milios, “Multi-Robot Collaboration for Robust Exploration”, *Annals of Mathematics and Artificial Intelligence*, Vol. 31, pp. 7-40, 2001.

14. Roumeliotis, S. I. and G. A. Bekey, “Collective Localization: A Distributed Kalman Filter Approach to Localization of Groups of Mobile Robots”, *IEEE International Conference on Robotics and Automation*, San Francisco, USA, Vol. 3, pp. 2958-2965, 2000.

15. Nerurkar, E. D., S. I. Roumeliotis and A. Martinelli, “Distributed Maximum a Posteriori Estimation for Multi-Robot Cooperative Localization”, *IEEE International Conference on Robotics and Automation*, Kobe, Japan, pp. 1402-1409, 2009.

16. Ahmadinia, M., H. Alinejad-Rokny and H. Ahangarikiasari, "Data Aggregation in Wireless Sensor Networks Based on Environmental Similarity: A Learning Automata Approach", *Journal of Networks*, Vol. 9, No. 10, p. 2567, 2014.

17. Karam, N., F. Chausse, R. Aufrere and R. Chapuis, "Localization of a Group of Communicating Vehicles by State Exchange", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, pp. 519-524, 2006.

18. Prorok, A., A. Bahr and A. Martinoli, "Low-Cost Collaborative Localization for Large-Scale Multi-Robot Systems", *IEEE International Conference on Robotics and Automation*, Minnesota, USA, pp. 4236-4241, 2012.

19. Mourikis, A. I. and S. I. Roumeliotis, "Performance Analysis of Multirobot Cooperative Localization", *IEEE Transactions on Robotics*, Vol. 22, No. 4, pp. 666-681, 2006.

20. Julier, S. J. and J. K. Uhlmann, "A Non-Divergent Estimation Algorithm in the Presence of Unknown Correlations", *American Control Conference*, Albuquerque, USA, Vol. 4, pp. 2369-2373, 1997.

21. Carillo-Arce, L. C., E. D. Nerurkar, J. L. Gordillo and S. I. Roumeliotis, "Decentralized Multi-Robot Cooperative Localization Using Covariance Intersection", *IEEE/RSJ International Conference on Intelligent Robot and Systems*, Tokyo, Japan, pp. 1412-1417, 2013.

22. Udemy and Dumble S., "Advanced Kalman Filtering and Sensor Fusion", https://udemy.com/course/advanced-kalman-filtering-and-sensor-fusion/, accessed on May 5, 2022.

APPENDIX A: ROS VISUALIZATION TOOLS

Transform tree used to share information about the relative and absolute positions and orientations of different coordinate frames over time are given in this appendix. Also, rqt_graph providing a visual representation of how nodes interact with each other in the ROS computation graph is given in this appendix.

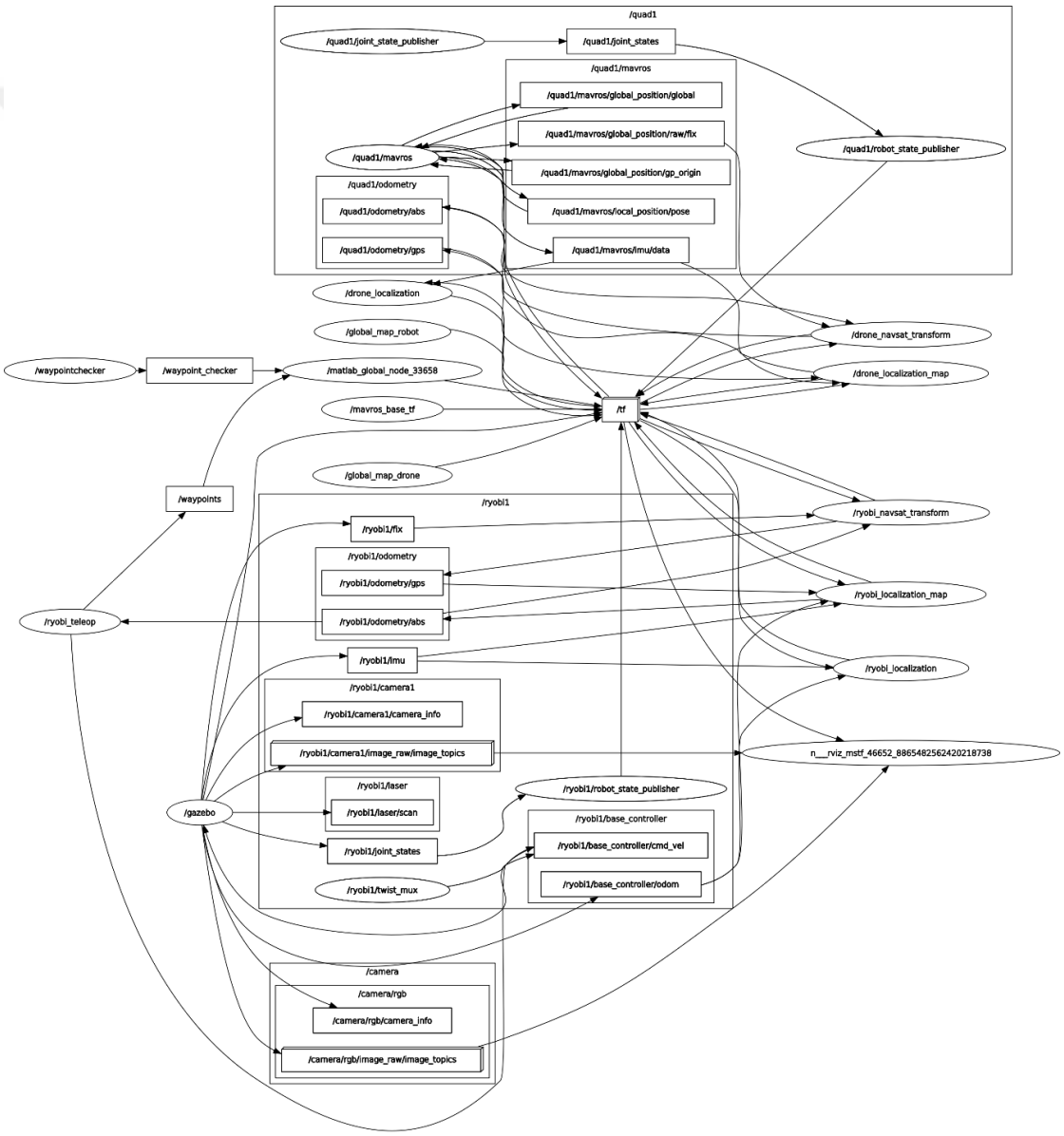

Figure A.1. Rqt_graph illustrating the node and topic interactions within the ROS computational framework of the simulated robotic environment.

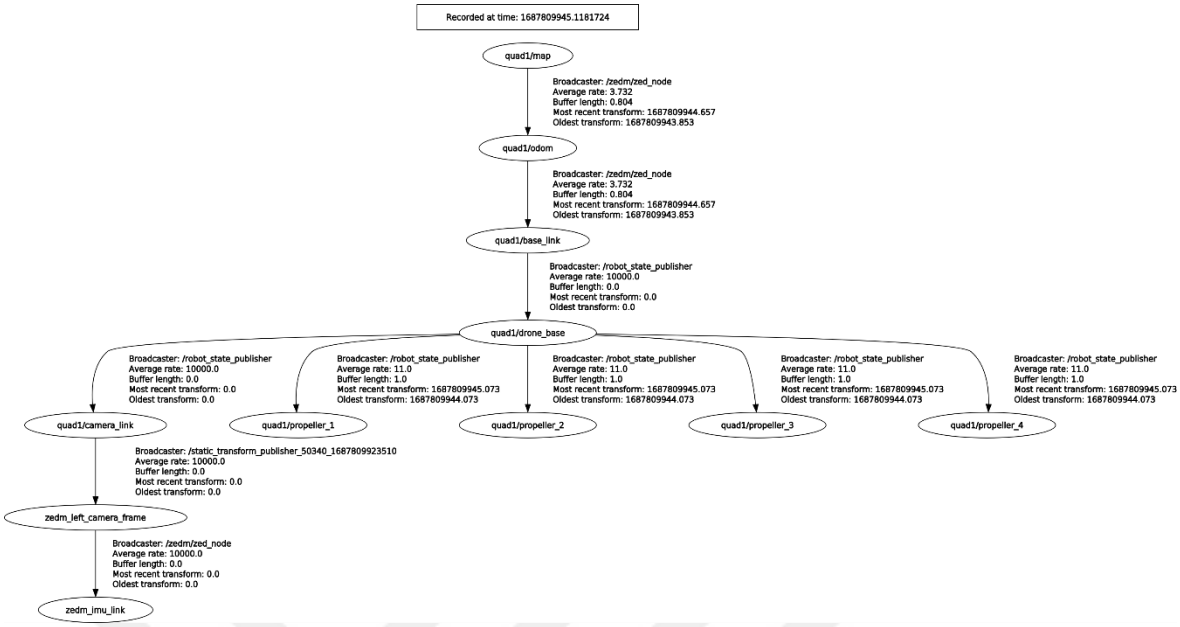

Figure A.2. Tf tree depicting the coordinate frames for the physical robot system.

APPENDIX B: ROTOR TECHNICAL SPECIFICATIONS

This appendix presents the detailed technical specifications of the rotor.

Table B.1. Sunnysky X4110S 400KV datasheet.

Prop (inch)	Volts (V)	Amps (A)	Thrust (gf)	Watts (W)	Efficiency (g/W)	Load temperature in 100% throttle
15*5.5	22.2	1.9	500	42	11.85	53°C
		3.2	750	71	10.56	
		4.9	1000	109	9.19	
		6.9	1250	153	8.16	
		9.2	1500	204	7.34	
		11.4	1750	253	6.91	
		14.2	2000	315	6.34	
		18.8	2430	417	5.82	
	25	1.7	500	43	11.76	
		2.9	750	73	10.34	
		4.4	1000	110	9.09	
		6.1	1250	153	8.20	
		7.9	1500	198	7.59	
		9.9	1750	248	7.07	
		12.5	2000	313	6.40	
		15.6	2300	390	5.90	
		18.8	2600	470	5.53	
		22.3	2840	558	5.09	
17*5.8	22.2	1.6	500	36	14.08	75°C
		2.9	750	64	11.65	
		4.4	1000	98	10.24	
		6.1	1250	135	9.23	
		8.1	1500	180	8.34	
		10.3	1750	229	7.65	
		13.1	2000	291	6.88	
		16.4	2300	364	6.32	
		19.9	2600	442	5.89	
		27.7	3130	615	5.09	
	25	1.4	500	35	14.29	
		2.5	750	63	12.00	
		3.9	1000	98	10.26	
		5.4	1250	135	9.26	
		7.2	1500	180	8.33	
		9.3	1750	233	7.53	
		11.4	2000	285	7.02	
		13.7	2300	343	6.72	
		18.4	2600	460	5.65	
		23.3	3000	583	5.15	
		33.3	3600	833	4.32	

Table B.2. Specifications of Sunnysky X4110S 400KV.

Parameter	Value
Item Brand	Sunnysky
Item No	Sunnysky X4110S 400KV
Stator Diameter	41mm
Stator Thickness	10mm
No. of Stator Arms	12
No. of Rotor Poles	14
Motor KV	400
No-Load Current(A/10V)	0.6A
Motor Resistance	Ω107m
Max Continuous Current	45A/30S
Max Continuous Power	1125W
Rotor Diameter	47.5mm
Shaft Diameter	4mm
Max Li-Po Cell	6S
Weight	165g

APPENDIX C: TECHNICAL DRAWINGS OF UAV COMPONENTS

Detailed technical drawings are presented below, showcasing the dimensions of key components, each critical to the overall functionality and structure of the system.

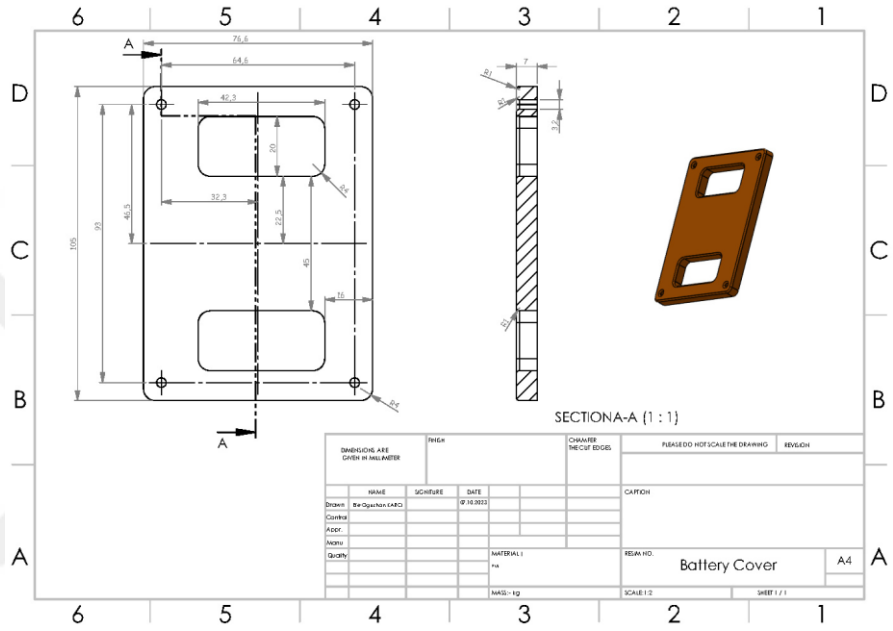

Figure C.1. Battery cover.

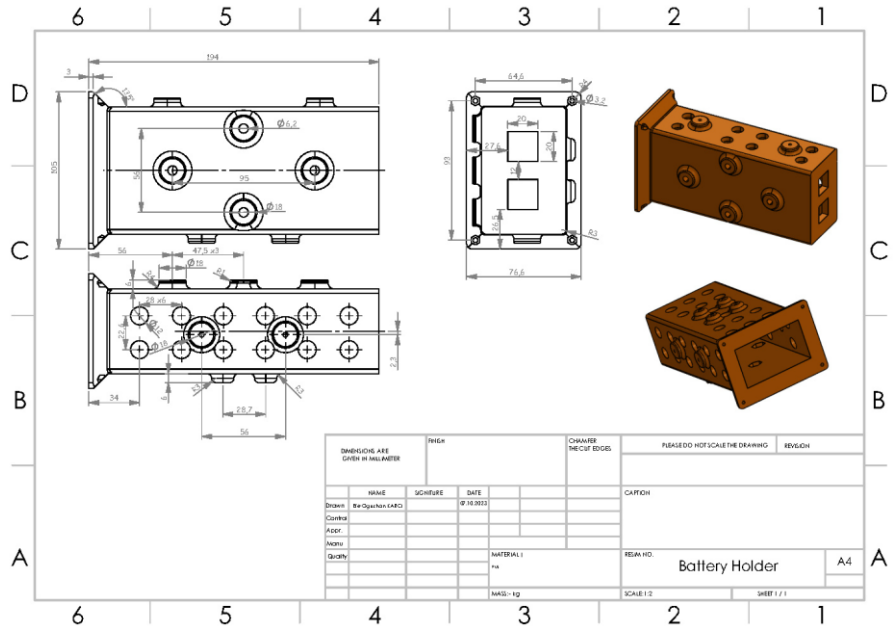

Figure C.2. Battery holder.

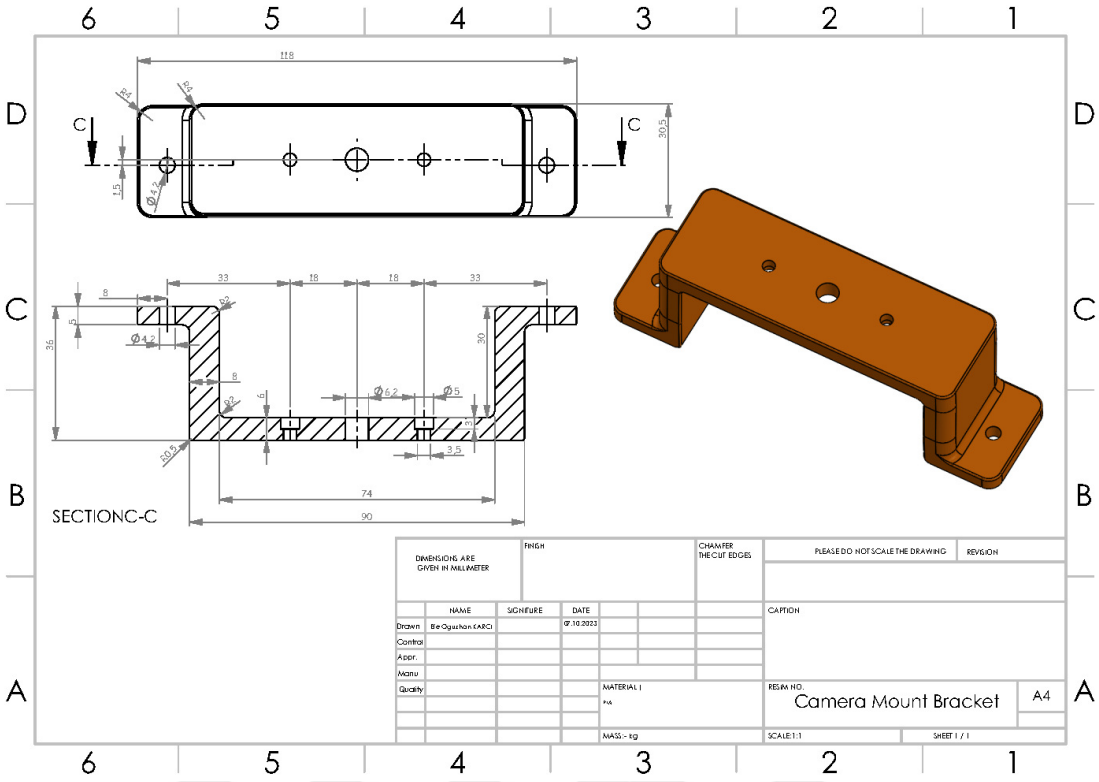

Figure C.3. Camera mount bracket.

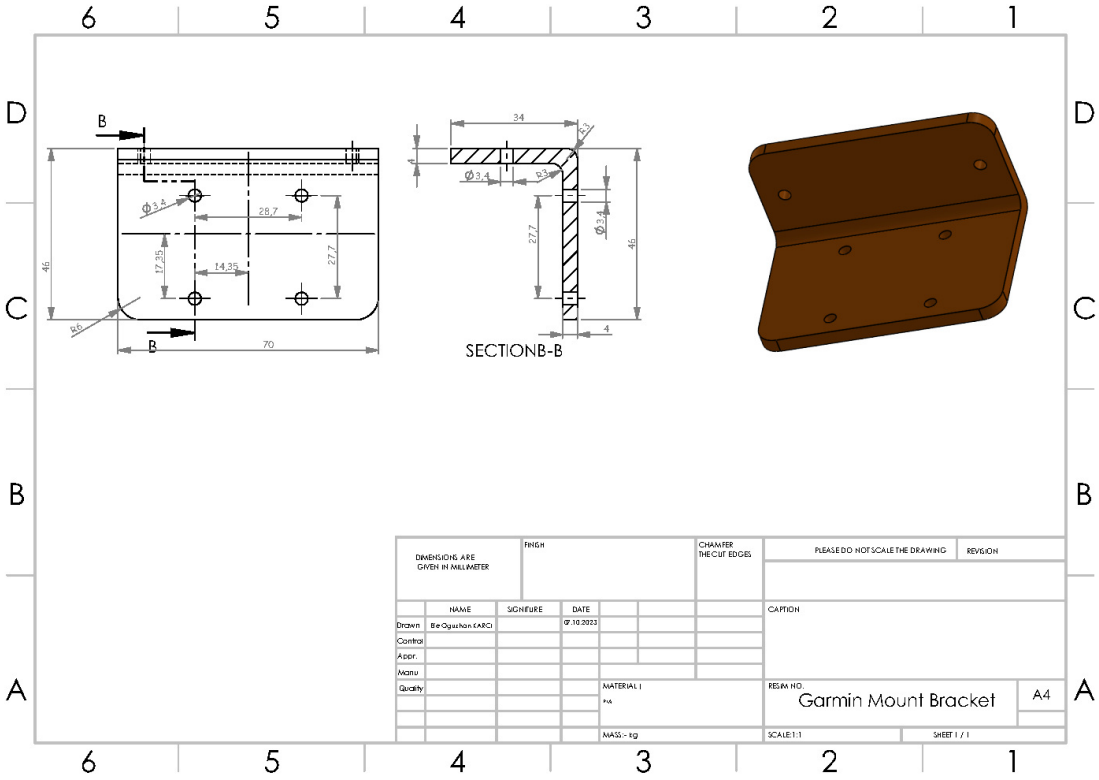

Figure C.4. Garmin mount bracket.

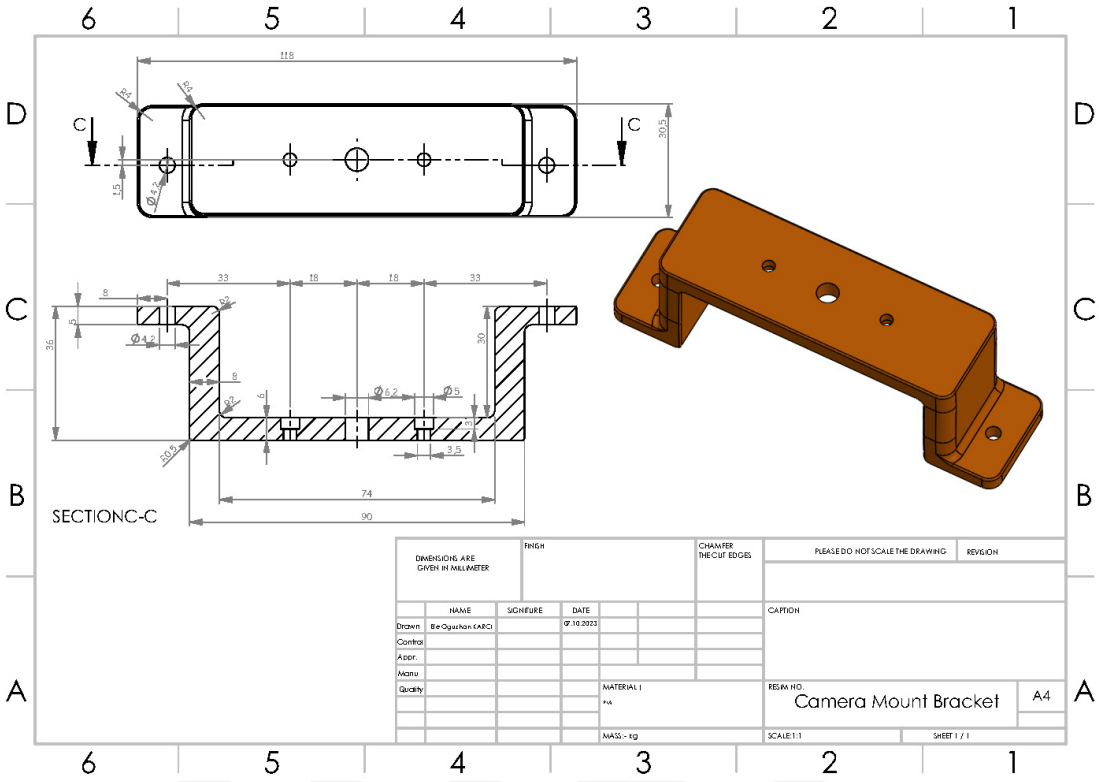

Figure C.5. Drone base plate.

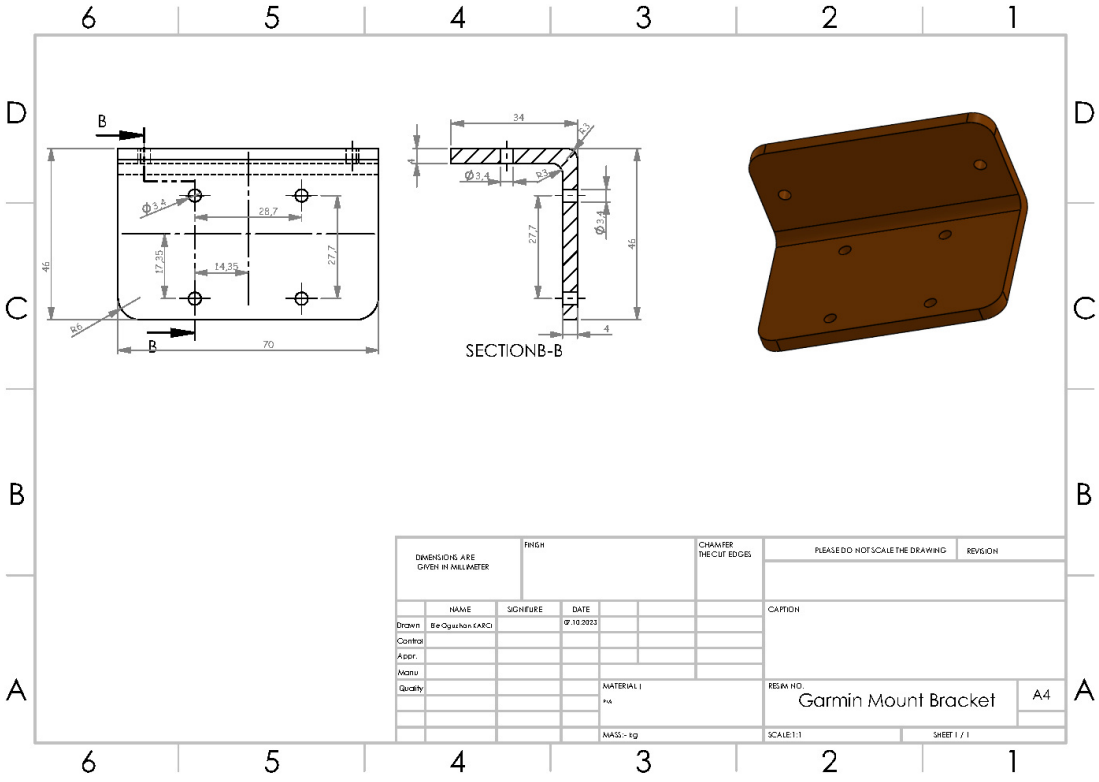

Figure C.6. Motor mount bracket.

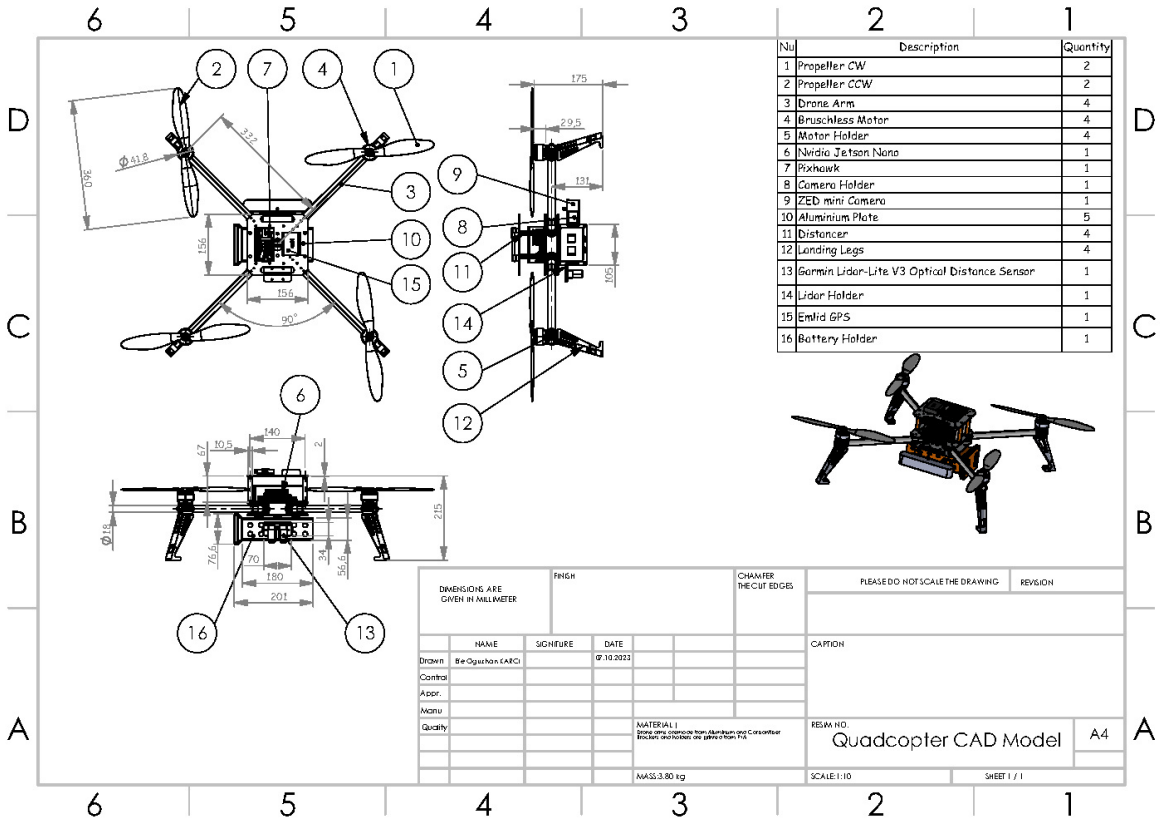

Figure C.7. Drone assembly.