# Autonomous Precision Landing of UAV Digital Twins on Moving Platforms and River Data Analytics from UAV Imagery

by

Rezwana Ashrafi
17201043
Jahir Uddin
22341091
Suhail Haque Rafi
18201004
Mashiat Mamun Raidah
18101359

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
September 2022

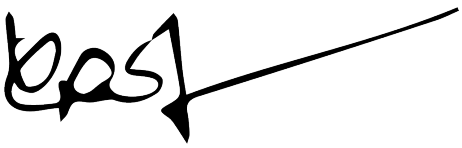
# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

Rezwana Ashrafi
17201043



Jahir Uddin
22341091

Suhail Haque Rafi
18201004

Mashiat Mamun Raidah
18101359

# Approval

The thesis/project titled "Autonomous Precision Landing of Unmanned Aerial Vehicles on Moving Platforms and River Data Analytic from UAV Imagery" submitted by

1. Rezwana Ashrafi(17201043)
2. Jahir Uddin(22341091)
3. Suhail Haque Rafi(18201004)
4. Mashiat Mamun Raidah(18101359)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 23, 2022.

**Examining Committee:**

Supervisor:
(Member)

Md. Khalilur Rhaman
Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

# Advisors

Supervisor:
(Member)

Md. Khalilur Rhaman
Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Advisor:
(Member)

Zubair Al Billal Khan
CTO & Co-Founder
STRAIGHT

# Abstract

UAVs have ignited our curiosity to understand the mechanism of flying and improve the way these vehicles fly to enable them to be of great benefit in a variety of applications such as exploration, rescue, 3D mapping, military use, and many other applications. But these applications of drones face limitations due to short battery life, human interventions necessary to replace the battery, power consumption, and communication distance. The purpose of our research is to mitigate the challenges faced by UAVs by proposing an autonomous quadcopter that is able to land precisely on a wireless charging station placed on a moving vehicle by motion tracking algorithm and increasing the accuracy of Helipad Detection and a Gazebo simulator capable of modelling an air traffic system for a swarm of drones by combining sensor values and control movement from the environment, as well as specifying the traffic system through path planning and collision avoidance. This study solves the short flight time of the soon to become one of the main components of the urban infrastructure air traffic system and ensures the variety of applications of these drones to be carried on for a safer and longer period of time and without any hazard. Aside from that, we will concentrate on factors such as water surface analysis, where we will look for water garbage in rivers and ocean bodies in Bangladesh.



**Keywords:** UAV, Digital Twin, Precision Landing, Trash Trap, Mapping, Detection, AI, River Image Processing

# Dedication

This study is a tribute to our parents, who have always provided for all of our necessities as we created our system and taught us that even the most difficult tasks can be completed if they are faced one step at a time.
We dedicate this research to everyone who put forth a lot of effort to assist us in completing this study.

# Acknowledgement

We acknowledge Md.Tahmid Rashid's guidence & mentoring and thank him from the core of our heart.

# Table of Contents

# List of Figures

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$APM$ ArduPilot Power Module

$COMP$ Coordinated Multi-Point Transmission

$ESC$ Electronic Speed Controller

$ETP$ Effluent Treatment Plant

$FAA$ Federal Aviation Administration

$FRP$ Fibre Reinforced Plastic

$UAS$ Unmanned Aerial System

$UAV$ Unmanned Aerial Vehicle

# Chapter 1

# Introduction

## 1.1 Thoughts Behind This Research

Lightweight quad-copters are becoming more common in a variety of applications, including topographic mapping, area monitoring, high-voltage line inspection, forest fire patrol, and so on. Drones will very certainly be utilized for continuous and autonomous operations in the future. All fully autonomous UAVs must have autonomous take-off and landing capabilities, which are both critical and problematic. In occupations that demand continuous flight operations, precision landing capabilities are critical for autonomous docking of aerial vehicles into a recharge station. UAVs are commonly powered by rechargeable lithium polymer batteries, which only last about 20-40 minutes[7], depends on the weather condition. The charging operation must be automated to produce a fully autonomous system. Bangladesh is a disaster prone country. Autonomous drone operation may help in rescue mission, surveying, river and ETP monitoring. This requires long flight time which requires more energy. The problem can be solved by implementing a autonomous docking solution so that it can land and recharge itself autonomously. The purpose of this study is to offer an autonomous vision based landing solution that can be integrated with existing drone flying system with the goal of analyzing UAV imagery and Digital Twin (DT). We implemented a Digital twin in this research to see the performance of sensors data. Also, by implementing DT we increased the performance of the drone by analyzing and reducing variance.
A unique state estimation methodology and predictive path tracking system, that allows us to monitor and anticipate an expected touchdown sequence position, with the accuracy and adaptability required to follow the vehicle also in bends of its course. The trajectory tracking approach applied for vehicle tracking using a prediction of its future direction is the most essential component of the suggested framework that helps the Quad-copter to touch down perfectly on a platform following a specific speed which is near to the UAVs' maximum speed and on a non-straight route. Simulating the air traffic system in a virtual environment is another aspect of our research. Commercial drone activities are becoming more common these days. Drones will soon be flying everywhere, dramatically boosting the quality of a wide range of services. According to the European drone outlook study [5], over 7 million drones will be in use in Europe by 2025 (200 thousand commercial and governmental drones, and about a thousand military drones). The Federal Aviation Administration (FAA) estimates that up to 3.17 million Unmanned Aerial Vehicles (UAVs) will

be in use in the United States by 2022, according to recent research. As a result, a standardized air traffic control system for UAVs that can be deployed in a real-world scenario is urgently needed. We need a simple model that can replicate an array of factors and operations utilizing each and every kind of drone with a variety of features, so we can examine and assess them before they are operationally tested. The simulator should be able to implement routing algorithms and collision detection approaches. The recommended simulator is based on the widely used Gazebo simulation platform. Furthermore, water is an essential resource for all life on Earth. When the water supply becomes contaminated because of pollution, it causes many health problems, Food chain contamination, and many more. Our target is to analyze the water surface pollution by image processing and data analysis using GIS mapping with an onboard CPU to detect the presence of trash and monitor the trash flow on the water surface.

## 1.2 Research Objectives

The major goal of this study is to provide dynamic charging stations to make UAVs more efficient while on the job. We will investigate dynamic precision landing, air traffic control, water trash detection, and wireless charging systems as part of our research. We will look at improving precision landing algorithms that will allow a drone to take off from a moving vehicle while maintaining communication with other drones. Also, our goal is to use UAVs to detect water trash so that we can notify authorities before it harms the environment. Our primary objectives are outlined as follows:

- One of our important targets is detecting the trash on the water surface. We will analyze the data and show the result on the spatial distribution map. If we look at the Earth's major pollution, we will notice that water pollution is one of them. In our country, mainly water from the surface gets polluted by the factories' garbage and with the water flow that garbage mixes with the river water and makes them polluted. The government of Bangladesh has made ETP (Effluent Treatment Plant) mandatory for all factories that use water as a raw or processing material, but most of the factories do not follow that. With this parameter of the drone, we will be able to detect the trash flow on the water surface. Also, we will be able to analyze where the industries do not maintain ETP and how badly that is affecting our water surface.

- Making a drone land precisely at a specific area has always been difficult, whether it's devising a mechanism for drone delivery, establishing a drone docking station, hovering a drone for monitoring, or attempting to land a drone in the back of a truck.We are addressing this challenge by proposing a unique precision landing algorithm that will allow the drone to land precisely.

- Incorporating autonomous technologies into heavily urbanized air transportation necessitates the creation of technologies and simulations to aid in ongoing research initiatives. We are building an Air Traffic Simulator System with the Gazebo Simulator Platform so that we can safely test UAVs in a real-world setting before deploying them. The ODE or bullet physics engines enable a

multi-robot simulation environment in Gazebo, which includes dynamics simulation. Gravity, contact forces, and friction are all taken into account by the simulator, making this simulation platform resemble a real-world environment. The user can add customized controls for simulating drones and senses, as well as alter the environment, using a plugin system.

# Chapter 2

# Related Work

## 2.1 Vehicle Tracking

In this paper [16], The author presents a case study of Yolov5 used to detect heavy goods vehicles at rest areas throughout the winter to estimate parking place occupancy in real-time. The author uses thermal network cameras because the icy circumstances and polar nights of much of the country pose some obstacles for picture identification in the winter. Because vehicle photos generally contain a lot of overlaps and cut-offs, the author used YOLOv5 and transfer learning to see if the front cabin and the back are good features for heavy goods vehicle recognition. The results indicate that the trained algorithms can reliably detect the front cabin of heavy goods vehicles, whereas detecting the rear cabin appears to be more difficult, especially when the vehicle is far away from the camera.
In this paper [17] The authors propose a multi-component reduction of the number of training examples in comparison to the initial YOLO-v5 design. The findings of this study's experiments also reveal that precision has greatly improved. The training results demonstrate a minor drop in characteristics from 7.28 million in the YOLO-v5-S profile to 7.26 million in the author's model, according to the author. Apart from that, when comparing the YOLO-v5-L/X profiles, the author reduced the recognition rate by 30 fps, and small object tracking performance was improved by 33percentage points as compared to the YOLO-v5-X feature in this study.
In this paper [14]The author explored the consequences and logic of several architectural and model adjustments performed to the famous YOLOv5 detection algorithm in terms of improving its small-object identification capacities, emphasizing its special demands and constraints, and suggesting prospective future study. The author has created an original YOLO-Z family of modeling techniques providing high quality with a 6 per cent increase in the ability to identify small objects while only raising prediction performance by around 3 ms. Based on these findings, existing technologies can be updated to accurate very small objects in situations where existing approaches cannot identify anything either. According to the author, this can improve an automated vehicle's detection sensitivity and perceptual resilience, resulting in superior planning and decision-making techniques, giving an autonomous racing car a significant competitive advantage.

## 2.2 Precision Landing

According to B.Caruso, M. Fatakdawala, A.Patil, G.Chen, and M.Wilde [15] (Demonstration of Inflight Docking), dynamic docking is not implemented yet. Autonomous static docking itself is a challenging issue. Currently, researchers are working on autonomous static docking and for future implementation, they are planning to work on autonomous dynamic docking. The author presented a vision/GPS-based docking control system for UAV Autonomous Aerial Refueling in this study [4]. An enhanced Kalman filter (EKF) approach for integrating GPS and machine vision (MV) sensor information to derive the accurate relative location of the UAV and the drogue has been proposed. The simulation results demonstrate that the proposed control strategy can precisely determine the relative location of the UAV and tanker and achieve safe docking. In this paper [13] the author's combined drones and deep learning technology. They used two drones for the demonstration. By using the UAV+CNN model, they realized the remote sensing recognition of parking spaces, which can largely inform people about which parking lot is available for them at the destination they are going to in advance. The authors of this study [18] Introduced an algorithm with two defining features:developed a difference convex automated system (DCA) to solution changes for UAV transmit beamforming and UAV-UE association, and (ii) utilized a deep Q-learning method to solve the challenges of CSI's inability to establish UAV positions. The two methods recommended resolving the problem. The deep Q-learning (DQL) technique is used in the first step to allow UAVs to learn the entire state of the system and account for such combined motion of all UAVs to manipulate their placements. In the second phase, the DCA continuously solves a concave approximation sub-problem of the original non-convex MINLP issue with the estimated, using computed UAV placements from the DQL approach. The problem's variables include transmitting directional antennas and UAV-UE connection.

## 2.3 Air Traffic Control System Simulator

In recent years, various studies on air traffic management systems were carried out by various scholars utilizing a variety of gaming engines and frameworks. The following are a few of them: Amjed Al-Mousa et al [9] presented the 'UTSim' simulator, which is based on the Unity framework. UTSim can simulate the physical characteristics of unmanned aerial vehicles, as well as movement, controls, connectivity, detection, and evasion in stationary and dynamic settings. UTSim was created with the goal of being simple to use. The user can choose the environment's features, the quantity and types of unmanned aerial vehicles in the environment, and the path planning and collision avoidance algorithm to be utilized. The simulator generates a log file containing a multitude of relevant data, including the number of sent and received messages, detected objects, and collisions of unmanned aerial vehicles. This article uses three scenarios to demonstrate the possibilities of UTSim and how it might help researchers in the subject of integrating unmanned aerial vehicles into urban air traffic. Although Unity 3D has many strengths that make it a great game development tool, in complex scenes, users might lose sight of some of the attached components:

- The built-in support for the PhysX physics engine in the Unity 5 engine has

several performance concerns and is missing some key features that are required to create an outstanding game app.

- From a graphics standpoint, the engine is behind. Unlike other game development engines, it does not provide a wide range of tools for creating stunning graphics.
- Licenses are required for the best visuals, deployment, and performance advantages. Furthermore, the use of graphics, buffering aid, stencil aid, and a plethora of other features raises production costs due to expensive licenses.
- In comparison to other engines, Unity's code is more reliable, and it comes with a wonderful architecture that boosts the game app's performance. However, the lack of source code makes it harder to identify, address, and resolve performance issues.
- The Unity engine consumes more memory, resulting in OOM failures and app debugging concerns.

Ziyi Zhao et al [10] introduce the Multi-agent Air Traffic and Resource Usage Simulation (MATRUS) framework, which intends to provide a quick assessment of various air traffic management strategies as well as the interaction among strategies, surroundings, and traffic conditions. It can also be utilized in the planning of a next-generation smart city to determine resource allocation and launch site placement. For a managed (centrally coordinated) and unmanaged (free-fly) traffic situation, extensive comparisons of UAS flight time, conflict ratio, and utilization of cellular communication resources are provided as a case study. This research, on the other hand, does not support evaluating the policies of UTM traffic management, elaborating airspace intrinsic capacity, or providing airspace traffic optimization at a lower level. Furthermore, the framework does not allow for the investigation of the impact of UAS detect and avoid techniques, the implementation of additional traffic management policies, or the handling of more complex traffic demand geographical distribution.
Sameer Alam et al [1] To investigate free flights, present the Air Traffic Operations And Management Simulator. The ATOMS concept, structure, capabilities, and applications are all described in this article. It's an intent-based simulator that divides the entire airspace into equal-size hyper-rectangular cells to consistently keep the intent reference points. It can simulate conventional and free-fly airspace operations and air navigation procedures from start to finish. To create precise trajectory predictions, the ATOMS models atmospheric and wind data. ATOMS employs a multi-agent-based modeling technique for modular design and easy integration of varied air transport subsystems. Airborne Separation Assurance, Cockpit Display of Traffic Information, weather avoidance, and decision support systems are only some sophisticated Air Traffic Management technologies prototyped in the ATOMS.. Experiments show that sophisticated ATM ideas support free-flying; nonetheless, more research and understanding of their intricate interactions in non-nominal settings is required.

## 2.4 Water Trash Trap Detection

Melisa A. Isgró et al. [22] employ regression analysis to calibrate empirical models for predicting water quality indicators based on in situ physicochemical parameters and spectral reflectance measurements acquired by the commercial Micasense RedEdge-MX Dual sensor. The multispectral photos are captured using the Micasense RedEdge-MX Dual Camera, acquiring information. Pix4D mapper, a Structure from Motion tool, was used to process the multispectral pictures. Based on the camera's EXIF metadata, the CRP to calculate absolute irradiance, and the DLS data to normalize each image for differences in incoming radiation during the conflict, this software does radiometric processing and calibration. The process's end products are ten single reflectance calibrated GEOTIFFs.They indicate that the reflectance value of each pixel on these maps ranged from 0.0 to 1.0.
The atmosphere barrier between the UAS and the ground is so thin that it may ignore this close-range remote sensing approach; it did not address atmospheric correction for UAS imagery. Picture segmentation thresholding is used in this paper [6] because it is a widely used image classification technique. It uses picture distribution statistics to extract objects from photos with varied gray level ranges for the target and background. Their article used an interpretive approach to retrieve data from beach detritus. The drone operated autonomously using DJI GS Pro, a ground station software. The whole mission procedure was completely automated, involving takeoff and landing, route planning, and computation of the proper spatial resolution of flying altitude. They selected the mapping and aerial image area patterns during the setup process. In this study [3], Using two alternative approaches in ArcGIS, employed high-resolution UAV pictures to determine lake borders. Lake boundaries are extracted by using a digitizing method and a maximum likelihood classification (MLC) method. MLC is a typical classification method with a generative model that assumes the picture features within each target class have a Gaussian distribution. So, Sample regions on the lake are determined for use as signature files in this manner. Using ArcGIS.Classified raster data of mine lakes are generated by using the signature file. The categorized data is passed through a majority filter three times to generate image contrast. And they use the lake boundaries transformed into vector data using a raster to polygon approach.
Paschalis Koutalakis et al [8] Present a low-cost unmanned aerial vehicle (DJI Spark) capable of obtaining reliable video images of a natural river. They evaluate the continuous frames taken by the UAV using three different software packages: Three image-based techniques are PIVlab, PTVlab, and KU-STIV (LSPIV, LSPTV, and STIV, respectively). The findings demonstrated that a low-cost UAV system could easily and successfully take a movie or a sequence of frames above rivers or moving water bodies. It can take Photographs with a multi-copter by directing the camera vertically downward. The study Using image-based techniques and a current streamflow meter was conducted along the same and additional cross-sections of the River. This allows for the analysis of surface velocity in various reaches and the testing of diverse natural water patterns.

## 2.5 Digital Twin

Digital twin is a virtual entity of a physical instance. According to Y. YU et al, [20] digital twins are more reliable, cheaper, and safer for on-site inspection. They also stated , modern technologies are capable of producing digital twins of existing assets. A digital twin is a way to inspect a entity's performance in real time and also make it possible to increase performance by adjusting errors. Another research was conducted [19], to study interference suppression of Digital Twin UAVs by adopting COMP technology. Another study by G. Nasos et al [11], deployed a digital twin to detect deviations from the UAVs expected behavious and detected potential bugs. S. Wen et al [23], developed a dynamic digital twin for an air-ground network in which DT served as an aggregator.

## 2.6 Wireless Charging

C. H. Choi and colleagues [2] suggest a completely automated wireless charging station. The station also allows for a less-than-ideal UAV landing on the platform, which is prevalent in real-world scenarios. Manual battery charging of quad-rotor UAVs might be totally eliminated with the suggested charging station. This study [12] discusses a rapid wireless charging port for an 85-kHz band 50-V 10-A inductive power transfer system constructed of fibre-reinforced plastic (FRP) and intended for outdoor use. S. Obayashi et al designed the charging port's distinctive frustum construction, which advantages of high coupling between transmit and receive coils, the lightweight of the receive coil, and misalignment prevention even in the gusty wind.

# Chapter 3

# Methodology

## 3.1 Calculations

We have calculated the total system before building the quad. It was necessary to make the quad stable and reduce costs. The calculations are given below:

### 3.1.1 Weight

- Total weight of quadcopter:
- 1 x Frame  400 gram
- 1 x Receiver  15 gram
- 1 x Flight controller 40 gram
- 1 x Battery 265gram (3s, other brand)
- 1 x Power distribution 68 gram
- 4 x ESC 4 x 28 = 112 gram
- 4 x Propeller 4 x 10 = 40 gram
- 4 x Motor 4 x 30.6 = 122.4 gram
- 1 x Rpi 1 x 46 = 46 gram
- 1 x gps 1 x 16 = 16 gram
- 1 x cmos camera 1 x 400 = 400 gram
- Others: 500 = 500 gram

Total weight of the quad-copter is 2024.4 gram.

### 3.1.2 Power

Total thrust of our 2306KV motors are 7032 gram. So, the weight lift ratio is 2024.4/7032 = 0.28
According to the motor specifications, the motors require 9.33 Ampere(50% throttle) continuous current. Using a linear ratio, the assumed usage is 9.33 * 0.28 = 2.6124 Ampere per motor. The total number of required ampere equals 10.4496 Ampere during a normal flight. We have used 3300 mAH battery. LiPo shouldn't discharge below 85%. This translates in 3300 * 0.85 = 2805 mAH for this battery.
The estimated flight time is = (2.805/10.4496) * 60 = 16 minutes.

## 3.2 3D CAD Design

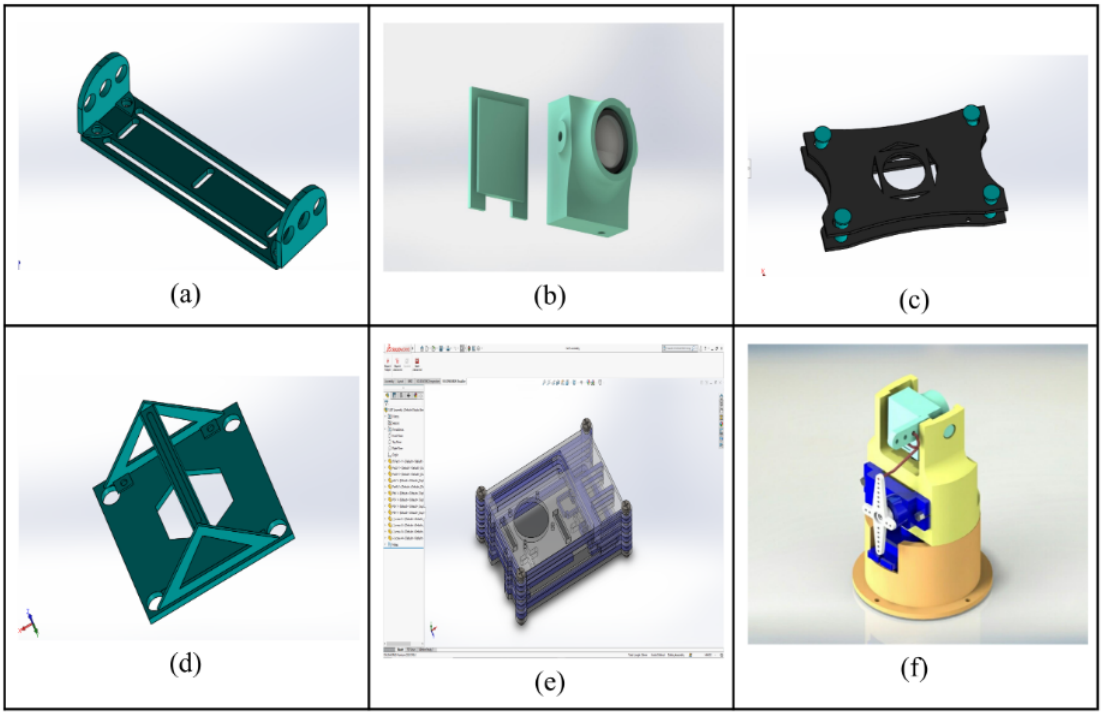



Figure 3.1: (a) Lipo Battery Mount, (b) CMOS camera box with lid, © Pixhawk Mount,(d) Servo Mount, (e) Raspberry Pi 4 Heat sink case, (f) Camera Mounted with Servo

We have included some of our SolidWorks designs in 3.1. We provided six different design images in all. We print those using our 3D printer after we finish our design. We couldn't use the 3D printer to print the weighty parts because there was a chance of them not printing correctly. As a result, we chose to print the lightest particles in the figure. We have designed this battery mount to add the least weight to our quadcopter. When tying up with the battery, you can place it anywhere in the quadcopter. We keep in mind that this battery mount provides the most air for cooling. We have designed our CMOS camera box with a lid that provides our camera extra protection with the flexibility to work with the servo. We have designed a lightweight Pixhawk mount that will be easy to place on the body with

the help of vibrator damping foam pads, and we designed it with enough space for the wires. When we activate the motor, the servo mount we construct here keeps it stable. We aim to build an amount that is exactly the right size for both the servo and the quadcopter.

A heat sink case was necessary because we are using a Raspberry Pi 4B for our quadcopter. Our heat sink is placed over the Raspberry Pi's chips and processor, and it helps transmit the heat generated on the chips and processor to the air. We have designed the servo so that the camera mount has the ability to move.

## 3.3 Hardware Architecture

We have used the PX4 Flight Controller board for its vast capacity to support different sensors and peripheral devices. It is directly connected to a ardupilot power module. APM is directly connected to 3300 mAH LiPo battery and it supplies regulated 5V to flight controller and 12V to electronic speed controller. Raspberry Pi is powered by 5V which comes from ESC. Raspberry Pi 4B is our main processing board in this design. It has a Logitech USB Camera connected to it that allows us to take good-resolution pictures. The script is executed on the Raspberry Pi and it detects AR Tags from the USB Camera feed. It also receives GPS and IMU Data from the Pixhawk Board. We have used the SE100 GPS Module that works perfectly with the flight controller. Later on, when it detects a tag, it sends appropriate commands to the PX4 and this instructs it to land. We have also connected a 2.4GHZ 9-channel radio controller that communicates through the RF, for manual control.

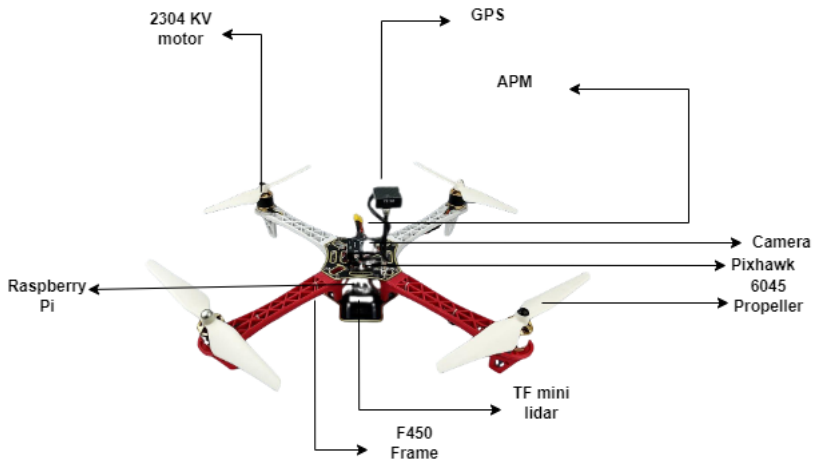



Figure 3.2: Quad Details

## 3.4 Power Distribution Board Design

We needed a custom Power Distribution Board (PDB) for our quadcopter. We tried to work on designing a custom PCB. We designed the PCB in the Proteas Software. The ESCs on the UAV require very high current passing through. This is why we made sure that the routes are thick enough to carry this current. Furthermore, we have also added some extra led along the routes to lessen the load on the PCB.

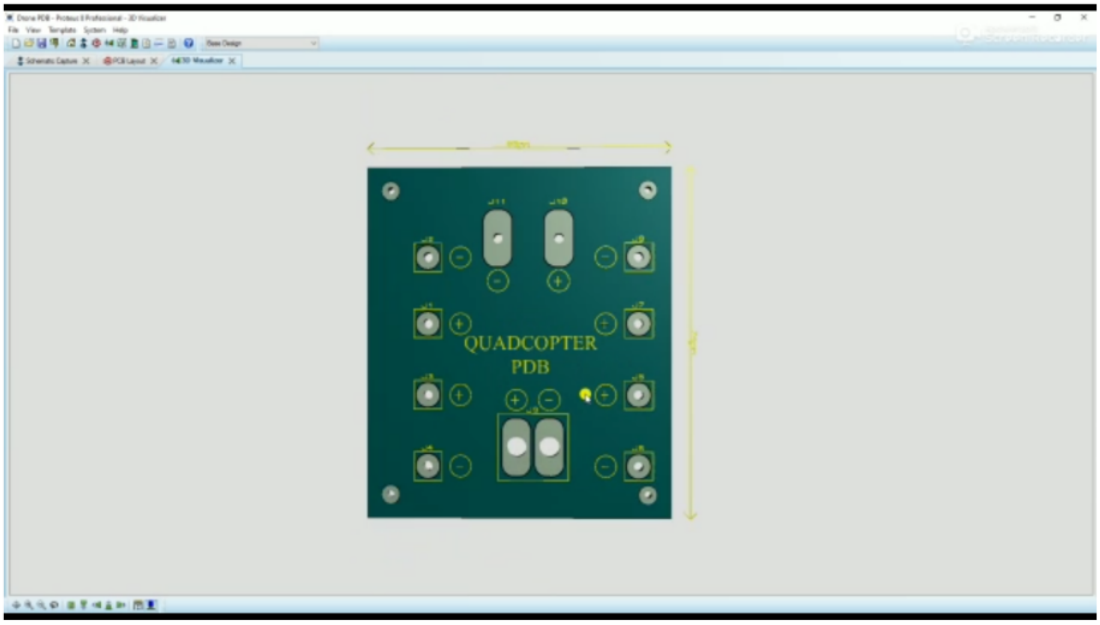



Figure 3.3: UAV Power Distribution Board Design

## 3.5 ROS Control Software Architecture

ROS is being widely used in the industry for its convenience and resources in robotics. We have designed our very own custom package for autonomous landing. The Landing Pad Node processes the camera feed and detects AR Tag using the OpenCV library. It is capable of detecting all 16 dictionaries of the Aruco Tag in different shapes and sizes. Whenever the landing pad i.e. the AR Tag has been detected it communicates with the Controller Node on the visual-data topic using MQTT Protocol.
When the Controller Node gets the localization information of the landing pad it calculates the PID values required for the landing and passes the data to the PX4 Firmware Node through the MAVLink Protocol on controller-data topic. The Raspberry Pi communicates with the Pixhawk using Serial Communication. The getting instructions from the Controller Node, the PX4 Firmware Node takes the necessary measures to navigate the UAV into a safe precision landing Here, the Controller Node and the PX4 Firmware Node act as both Publisher and Subscriber at different times. When Raspberry Pi is sending data to the Flight controller, the Controller Node is the Publisher and the PX4 Firmware Node is the Subscriber. Nevertheless,

the are times when Raspberry Pi needs data from the Pixhawk such as GPS data, IMU data etc. In those times, the PX4 Firmware Node works as the Publisher and Controller Node as the Subscriber.

### 3.5.1 Trajectory Projection

One of our applications is to precisely land the UAV on a running vehicle. For that, we followed two different approaches. One is tracking the helipad and another is tracking an AR Tag. But the problem is we can identify the vehicle but to approach the running vehicle it's important to know the speed of the vehicle without intercommunication. In real life, the challenge is to track a vehicle across different cameras and angles. It is challenging because in some cases the vehicle appears different because of the camera motion. For this problem, a robust tracker is beneficial. We solved this problem by using the optical flow algorithm. In this method, the vehicle is tracked utilizing Spatio-temporal image brightness fluctuations at the pixel level. The goal of this algorithm is to obtain a displacement vector for the object that will be monitored across the frame. We faced a problem while trying to predict the trajectory of a running vehicle. We noticed that we can detect small motions but we failed to detect large motions. We used the Lucas-Kanade approach to derive an equation for the velocity of certain spots to be tracked in order to address this problem. There is a function in OpenCV cv.calcOpticalFlowpyrLK(). We constructed a basic application to track the points using this single method. We start with the first frame and discover corner points, then use Lucas-Kanade optical flow to iteratively track those locations.

We work with another dataset for vehicle detection as we are working on precision landing on a running vehicle. We collected approximately 300 images, which was insufficient, but it resulted in an accuracy of 79 per cent. We trained this data into Yolov5. Earlier, we separated and labeled the 240 training images; for the remaining 60 images, we used them as validation and labeled them. We get 79 % accuracy after running the dataset through Yolov5, and the output of our trained data is shown in the figure.

Simulation is essential before creating a real-life prototype to assure a good system design as well as safety in deployment because developing and testing autonomous navigation algorithms on real-world UAVs is difficult and requires intensive resources. As a result of testing in simulations, risks are reduced when testing physically. Before deploying the entire system in the physical realm, visual-based autonomous landing algorithms must be thoroughly tested in simulated situations to catch potential vulnerabilities. Using the PX4 SITL, we tested our algorithms in the Gazebo Simulator environment. The Pixhawk flight stack is managed by SITL PX4, which also offers direct support for the prototype deployment procedure. For object detection and tracking, cameras have been widely utilized around the world. Deep learning algorithms provide good results, however, it is vulnerable to delivering low spatial resolution. It is computationally costly and limits the application's efficiency. This is why we have an alternative approach.

When dealing with vision-based tracking, this second approach focuses on lower-

ing the computing overhead. We've replaced Helipads with AR Tags [21] as docking pads. AR Tags are easily identifiable fiducial marker systems with their own IDs. So we can give each of our docking stations a unique ID. Even in rotating positions, system occlusion, and random overlap with an object, they are plainly apparent.
The landing pad is detected by comparing it to the AR Tag template. To discover features, the camera frame is compared to the template frame. A description is assigned to every feature in both templates. We determine the distance between the camera frame and the template using the Euclidean distance measurement formula. The correlation between the two frames is used to create a homography matrix.
The orientation of the landing pad is then determined by applying a perspective transform to the homography matrix. The landing pad's corners and centre are then calculated. As with the DeepSORT technique, the Kalman filter then utilizes a velocity model to track the location of the pad. It can reliably estimate the pad's location in the previous frame using this velocity model. The AR Tags' unique IDs allow us to numerically identify each station independently.
A control system is required to coordinate the peripheral devices to approach for landing. For this, we have used a PID controller that performs based on the object tracking data i.e. the centre of the landing pad from the camera frame and the angle, on top of the Pixhawk's built-in PID controller. The external PID controller minimizes the difference between the dimensions of the template frame and the camera frame.
At the moment, we are in the phase of testing this algorithm of the UAV physically and observing the results.

## 3.6 Precision Landing

Drone operation requires stability, reliability, and precision while navigating autonomously. For every flying vehicle, it's important to land or dock precisely on a charging pad or launching pad. When it comes to long-distance operations, it's a must-have recharging station. We, humans, make errors while doing something accurately. When it comes to drones or space, the situation is more complicated. In this study, we established vision-based precision landing both practically and in simulation. Previously, we stated we have built a quad-copter including a raspberry pi as a core processing and commanding module. For vision-based landing, we have used some components like a night vision pi camera attached to the raspberry pi, and a mini LiDar for ranging distance accurately.
There are three core modules in this project. Firstly, in the flight control module, we used an open-source Pixhawk flight controller for the physical drone. A flight controller is a module that controls drone movement in the air. Secondly, we used the Raspberry Pi 4 as a core processing module. The Raspberry Pi is a single-board microcomputer. Finally, we used an infrared-based night vision camera with manual focus. These three components are integrated together in the drone.

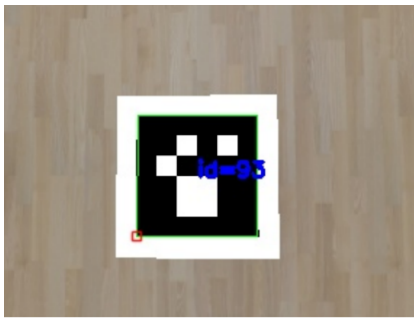



Figure 3.4: Marker Detection

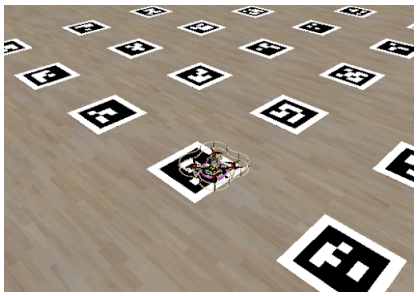


Figure 3.5: Landing on Target Marker

### 3.6.1 Digital Twin & Air Traffic System

An air traffic system is essential for safe air travel. Tracking and monitoring UAVs is vital for UAV navigation and making the sky safe for all flying objects.
For now, we have created an environment using an open-source java simulator. For UAV navigation, we have used PX4 firmware, and for control station and monitoring, we have used ground control software. We have used a custom quadcopter with necessary sensors such as GPS for positional data, imu, 3DR for two way communication etc. As we are using GPS for positional data, we can see every flying UAV on our ground control software. Also, we can easily define routes for UAVs. As a result, it'll be easier for UAVs to avoid collisions. Additionally, by using 3DR it is possible to intercommunicate between the UAVs which may help to synchronize and get necessary broadcasting data. Also, it is possible to stop any unauthorized flying objects or to track them and inform concerned authorities.
We will explain the simulation first, then we will move to the practical part. In the simulation, we used a robot operating system, a framework for robot control, PX4, an open-source drone firmware, Gazebo, a simulation environment, Rviz, to get feedback from subscribed topics, and RQT-graph to see the feedback from connected nodes. In the given figure, we can see all the published topics and connected nodes altogether, which we got from rqt. First of all, we will be giving an overview of the simulation. Later, we will deep dive into the simulation. For the landing platform, we have trained a model using the helipad dataset, which is discussed in part 28. To make the landing more precise, we have also used ArUco markers. An ArUco marker is composed of a black border and has a binary matrix inside which provides a unique identification number. This enables error detection and correction by extracting data from the marker. Internal matrix size is determined by marker size. OpenCV-

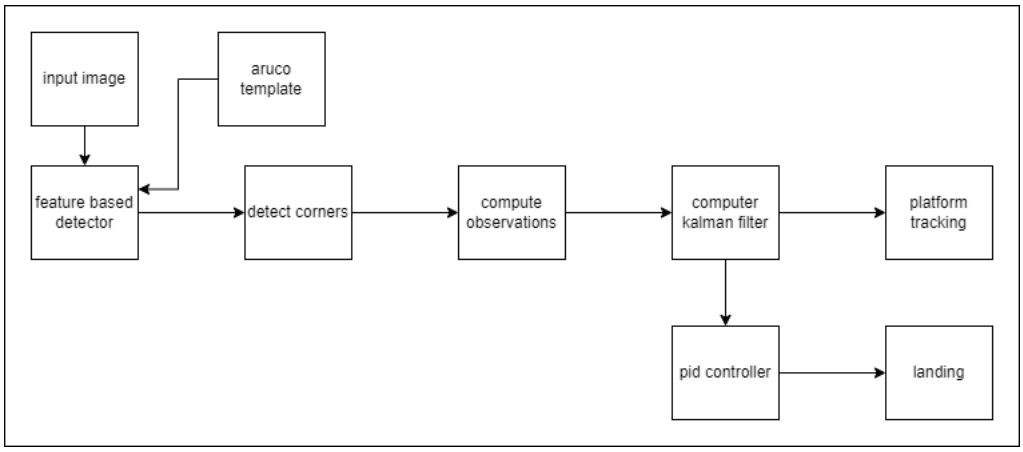



Figure 3.6: Simulation Pipeline

contrib has a library for ArUco markers, which makes it easier to code and detect. As the robot operating system comes with a pre-installed OpenCV dependency, the ArUco marker is also there. From the main camera, we are taking input from the environment. After getting the ArUco marker into a frame, the Rpi extracts its feature with respect to a given template, which we can see in 3.6. After extracting the feature, it detects the corners and computes observations. After computing, it reduces errors by applying a Kalman filter. By using the PID algorithm, it navigates to the target ArUco marker and lands. In 3.4, we have shown details of the total simulation pipeline. We have ROS nodes, topics, publishers, subscribers, actions, etc. We modeled the world in the Gazebo simulation environment using the Clover drone instance, ArUco map, and physics model. For coding, we used the Python code-base for easier integration. ROS nodes are individual processes that are written into the program. The nodes receive and send information. This information is organized in a specific way called a "topic." In 5.2, after initializing the simple off-board node, it subscribes to Gazebo and the main camera node, which captures aruco information from the world and publishes the message through TF. Aruco detects the node subscribes to that topic and gets the information. After extracting the feature, it again published the information through TF, and Mavros subscribed to that node. Mavros is a communication protocol between ROS and PX4 firmware. After getting the information through Mavros, the PX4 firmware gives the command to the drone to fly and land on the specific ArUco marker. During this time, it continuously receives messages from the subscribed thrust, control, and altitude nodes. By adjusting those parameters using the Kalman filter and PID algorithm, it reaches the target and lands smoothly.
In 3.4, we can see our main camera, which captures a frame of the marker, is detected and its unique identification number is shown. As shown in figure 4, the ability to detect ArUco markers individually helped us land on a specific target precisely, as shown in 3.5. Also, using camera calibration, we determined the distance between the main camera and the marker. With the help of a rangefinder(mini lidar), we may avoid collisions in swarm and obstacle detection scenarios. The rangefinder at the markers' z-axis helps us to adjust thrust and tune the PID algorithm to land safely.

Finally, we tested our simulation results in the real world. We equipped the drone

with a Raspberry Pi, a camera, pixhawk, GPS module, and telemetry and configured it properly. Additionally, we used a 10-channel 2.4GHz radio controller for manual override in the case of unwanted situations. We set it to script mode on switch 9. We planned a grid path-like simulation. The Rpi is configured to connect to the same LAN network as the laptop. By remotely accessing the RPi, we executed the landing script and switched to the script mode. It changed its altitude to 2.5 meters and looked for the marker. After seeing the marker, it analyzed the distance, adjusted its thrust, and landed. To make it safer and smoother, we used the SE100 GPS to get feedback from the GPS and took advantage of position hold mode.
We get real world sensor data from our quad through mavlink. Sensors are connected to Raspberry Pi. Raspberry Pi has raspbian os with ROS and other tools. It it connected with our ground control module via MavRos. MavRos is bridge between mavlink and ROS. It connects base station with other sensing module. After collecting the data such as: gyro and power data, it analyzes the data and thrust node estimates the thrust level. After estimating, PX4flow node adjusts the thrust level by tuning PID value. This way, DT helps us to reduce variance.

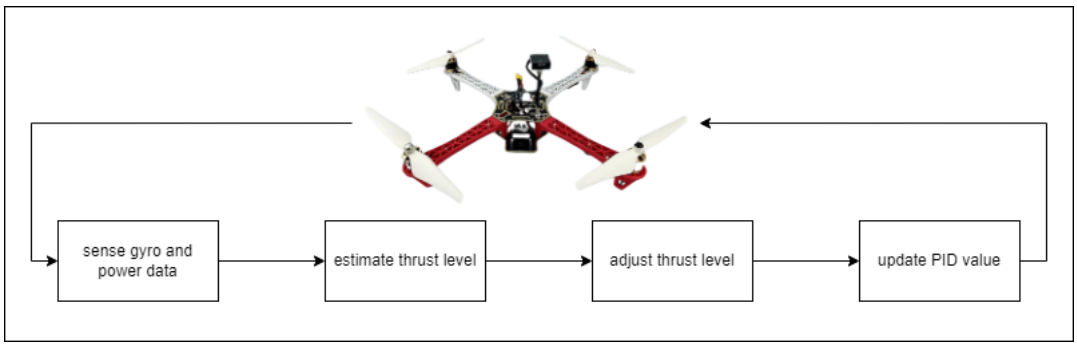



Figure 3.7: Digital Twin Architecture

## 3.7 Description of the Model and Data

Because of the vast variety and high volume of trash generated in manufacturing, manual waste sorting is time-consuming and labor-intensive; The quantity of output and residential waste increases as social productivity rises, lowering people's living standards and quality of life while also causing environmental damage. Thus, the implementation of automatic waste identification can be very useful. Because of its fast response time and excellent detection accuracy, we use the YoloV5 algorithm. The YoloV5 method treats target detection as a regression problem. Three fundamental structures are in the YOLOv5 network model, First one is the backbone, the second one is the feature pyramid network and the third one is the detection header. The backbone network is in control of feature extraction from various kinds of pictures at various scaling, the feature pyramid network is in control of fusing features from different kinds of scaling and transferring them to another network which is a detection network. The detection network is in control of trying to predict the object group in it to use input pictures and creating the object bounding box obtained by training the network model with the backpropagation algorithm.

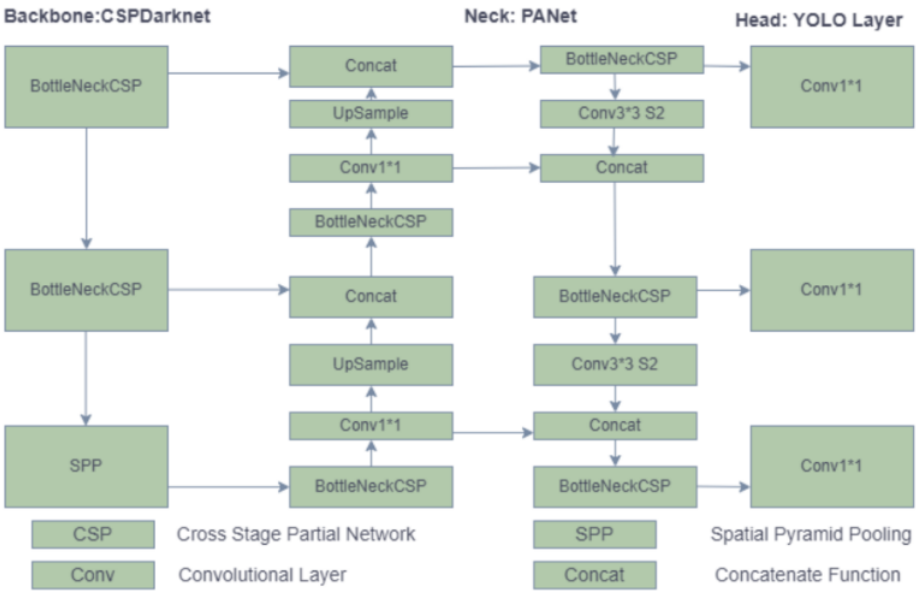



Figure 3.8: YOLOv5 Architecture

In the diagram above, the network architecture of YOLOv5 is depicted. The Cross-Stage Partial Network (CSPNet) on Darknet was first merged by YOLOv5, which led to the development of CSPDarknet as the primary building block. By integrating gradient changes in the feature map, CSPNet addresses recurrent gradient information problems, reduces model parameters and FLOPS, ensures fast and accurate inference, and reduces model size. The YOLOv5 utilizes a Path Aggregation Network (PANet) as neck to enhance the information flow. PANet implements a new Pyramid Network (FPN) feature to improve low-level feature propagation. Additionally, each feature level provides important information that reaches the following subnetwork thanks to the pooling of adaptive information, which connects with the feature grid to all other feature levels. PANet encourages accurate localization signals at lower levels, greatly improving the precision of object location. To conduct multi-scale prediction, the Yolo layer, which is at the core of Yolov5, generates three different sizes of feature maps (18, 36, and 72).

In order to land precisely, you must first detect the helipad. We created a secondary dataset by scraping photographs from the internet using Google. There are 650 photos in the dataset. We then divided the dataset into 80, 20 train and test groups. The model is trained using the YoloV5 model.

## 3.8 Data Acquisition and Calibration

We have made a dataset from surfing the internet. As we focused on images that are mostly taken from UAV or atleast from the high altitude similar to a UAV. We

have gathered around 200+ images for the model to work on. 160 of them were used for training and 40 were used for testing as per the 80-20 split.


Figure 3.9: Image Custom Dataset: Vehicles

### 3.8.1 Water Trash Detection

To capture images and videos of the trash on the water surface, we need a camera, and we are going to use a CMOS camera for that. The UAVs camera is a 20-megapixel 1-inch CMOS. The onboard computer is chosen based on the number of processes required to perform image processing after detecting the garbage. To detect the trash on the water surface, the data set here we used in this paper contains a large number of trash images, the majority of which we collected from the internet. There are two types of data categories in this set. one is labeled as trash and another as not trash. 20% of the images are selected as a set of validation, and the remaining 80% of images are used as the training set. We label the training set with the make sense tool and generate the corresponding XML file for training. Each of the pictures generates a text file with a similar title, with each line representing the label of a single object. The object class is the first column. Here the dataset only contains one type of class., and all classes are merged. Object frame's XY coordinates are in the second and third columns, and the coordinate position normalizes using the original image's aspect pixel values as a numerator. Column number, fourth and fifth display the object frame's normalized aspect pixel values.

## 3.9 Plotting Data on Map

As we know, A Geographic Information System (GIS) is a valuable tool for combining spatial and other data. It enables the spatial representation of information and analysis of integrated data, making resource development, environmental protection,

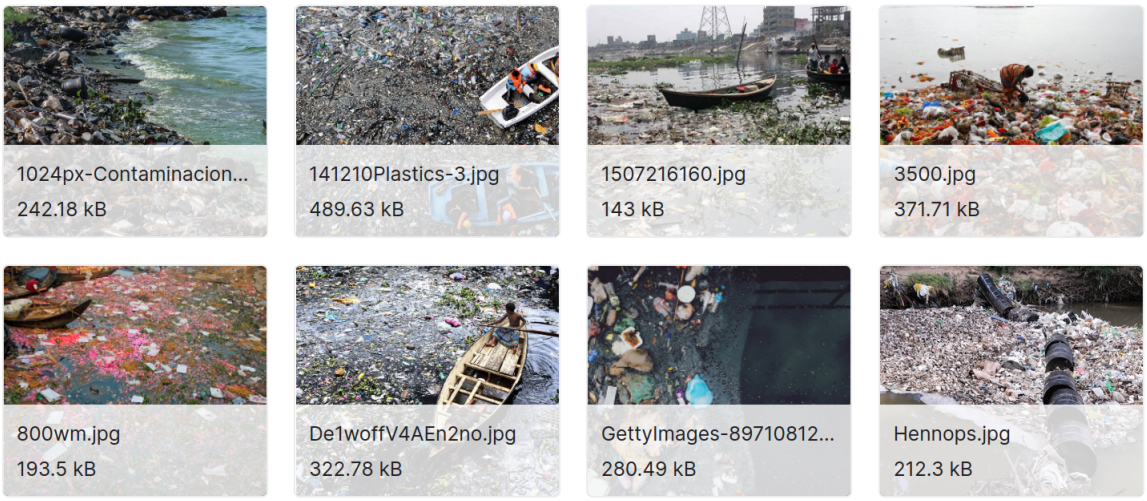



Figure 3.10: Image Custom Dataset: Water Trash Trap

and scientific research planning easier. GIS has advanced map-making skills that are valuable for communicating data analysis results. So we will use GIS mapping. For tracing and keeping the record of the location, we will use a Geocoordinate record. So, after analyzing the data, if there is any garbage or trash on the water surface, it will mark it as red in the digital image; otherwise, it will remain blue. Finally, we will show it in the spatial distribution map for better understanding.
We used one leafmap plotting backend because our main goal was to detect the garbage from the drone view, locate the garbage on the map with a circle marker, and measure the garbage area. We use Folium as the plotting backend to create an interactive map that functions in a Jupyter environment like Google Colab, Jupyter Notebook, or JupyterLab.

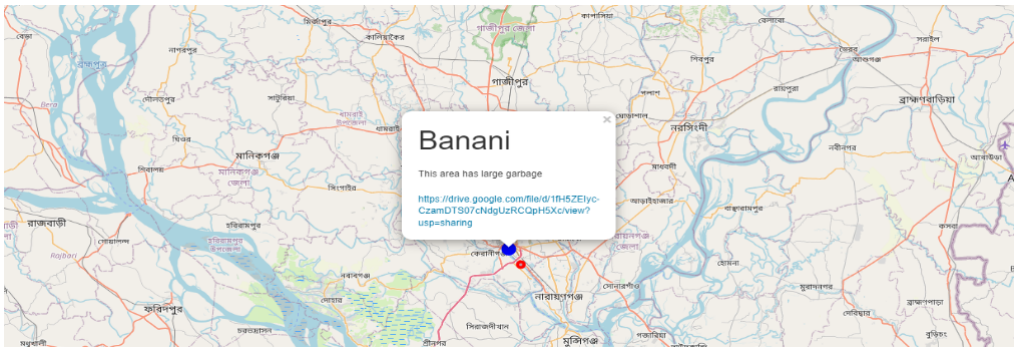



Figure 3.11: Trash Data on The Map

**Jupyter Notepad:** We use Jupyter Notebook as it is an open-source web application. Jupyter Notebook enables sharing of documents with real-time code, equations, visuals, and text. This technology has many applications, such as data transformation and cleaning,machine learning, simulation, statistical modeling, data visualization and many more. For our data visualization, we'll make use of this tool.

**Pandas:** Some of those tools are contained in the standard library, a toolbox that

is provided with the language. One of the significance of this class is the panda. It is a program that has evolved into a top open-source library for getting at and analyzing data from various sources.

**Numpy:** For Python-based scientific computing, Numpy is the core package.

**Folium:** When working on interactive leaflet maps,it is easy with folium to see data that has been modified .Python is used. Data connectivity to a map for choropleth visualizations is made possible by HTML, rich vector, and raster visualizations that are passed as markers on the map in folium. In addition, the library supports user-created tilesets utilizing Mapbox or Cloudmade API keys and provides several pre-built tilesets that leverage OpenStreetMap and Mapbox. Folium supports image, video, GeoJSON, and TopoJSON overlays. We utilized folium to locate trash on a map.

**Openstreetmap:** A popular choice for creating Slippy Map is Leaflet, a cutting-edge which is a mobile-friendly, open-source JavaScript library for interactive features.OpenStreetMap is used as a free global geographic database. It records each physical feature on the planet. The mappers who create the OpenStreetMap database collect data while traveling by car, bicycle, or foot along streets, using GPS receivers to record their every move.A set of transformable geometric properties as in points and lines that can be used for navigation or converted into maps are made using this information.OpenStreetMap has utilized a wiki-like system for this database. Any mapper can use it to edit or add features to any location, and every object's entire editing history is recorded. Because of this, deliberate vandalism can be overturned, keeping the data accurate, and OpenStreetMap does not store its data in an existing geographic information system. Instead, it uses its software and data model to facilitate crowdsourcing and provide the greatest flexibility. Compared to other geographic databases, Openstreetmap is updated more frequently. Downloads of the most recent data are always available. Anyone can manually fix errors in OpenStreetMap data. If any errors are found in the data, they can be corrected on the map quickly, and the database is immediately updated. While most other mappings only offer a few different style options, obtaining a customized map is expensive and time-consuming. The only restriction on the variety of mapping styles with OpenStreetMap is their own technical and cartographic capabilities. Additionally, several free map-rendering software programs and GIS programs support OpenStreetMap data.

**Mapbox:** Among other data sources, Mapbox Streets uses OpenStreetMap. All Mapbox users have access to Mapbox Streets, a set of vector tiles used in almost all of the different Mapbox template designs. The Mapbox Streets tileset will likely be included in any new style created using a Mapbox template in the Mapbox Studio style editor. The quickest method for using OpenStreetMap data in Mapbox Studio is to use Mapbox Streets as a source. Mapbox Streets is a customized tileset manufactured from OpenStreetMap data; it is updated as OpenStreetMap is edited. It is vital to use additional tools, such as Overpass Turbo, to extract OpenStreetMap data to add features to the map that Mapbox Streets does not support.

We fixed a frame for the drone image view to measure the garbage area. If trash is outside the frame when the drone takes the picture, the provided code will stitch the images from the exact location, create a complete view from those several images, and send it to the folium parameter. If the trash covers a large area, the circle maker will appear with a large radius and a different color on the map. These are some parameters we use.

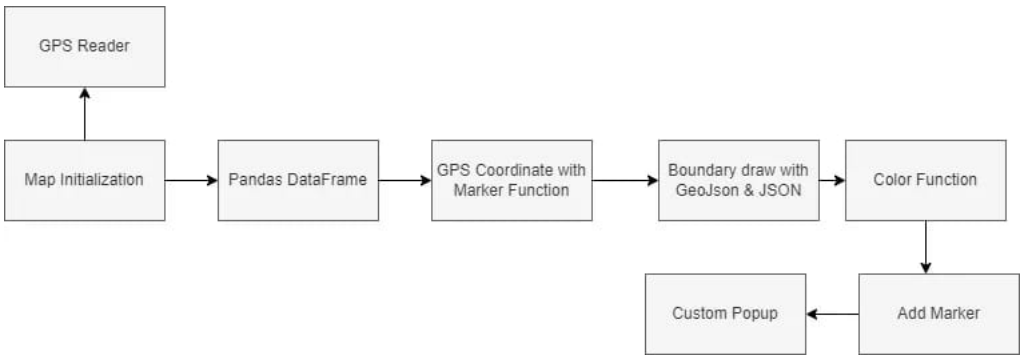



Figure 3.12: Map Plotting Workflow

Map() and Tiles: We employ the Map () function to create a map. To accomplish this, we must provide the function with the latitude and longitude data.The Map() function's "tiles" parameter defaults to being "OpenStreetMap". With this parameter, we can modify the Map's design. Alternate options include CartoDB Positron,Stamen Toner, CartoDB Dark Matter or Stamen Terrain. Adjusting the size using the width and height parameters, and also enlarging an image using the zoom start feature. We use the Circle() function to circle the coordinates. By entering parameters like radius and color, we can alter the output. We get GPS coordinates and the Marker() function to mark the coordinates' location. We furthermore placed the location name using the popup parameter. The data of garbage from the drone view is used. This data consists of area name and latitude and longitude from GPS coordinates and the measurement. We use the GPS coordinate data because we require a region's location on the Buriganga side. We use the Pandas library to load the data in Excel and look at some values after loading the data. The Pandas merge() function is used to combine these data. The Plugins package contains the MarkerCluster() function. We set a parameter based on the geographic data. We draw boundaries with the help of GeoJson in the Folium and a JSON file with the desired coordinates. We try to color the Map according to the size of the garbage patch. We use The Choropleth () function in this. Using each coordinate value, we use the MarkerCluster() function to map the locations of their accommodations. Then, we wanted to assign each lodging a different color based on how big the garbage area was. To do that, we require a color map. For various ranges, we use different colors. For instance, we use a red circle with a large radius for large area garbage, and we assign the minimum and maximum area coverage to various variables to achieve this.

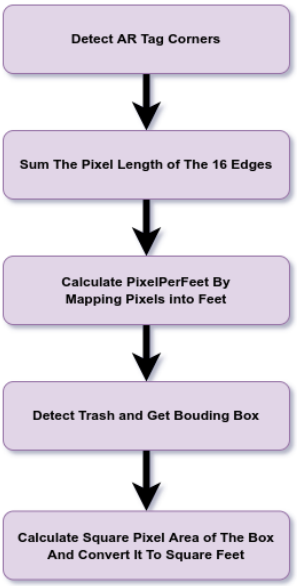



Figure 3.13: Pixel To Feet Method

## 3.10 Area Calculation for Trash Traps

After detecting trash with the help of YOLOv5, we are proposing a PixelToFeet method to approximate the area of the trash on the water surface. In this method, we will freeze the altitude of the UAV at a specific height and hover around the water body, maintaining that altitude. Before the start of hovering, the UAV will detect the landing pad of four 6X6_250 AR Tags and calculate the pixel per foot value from that Tag.

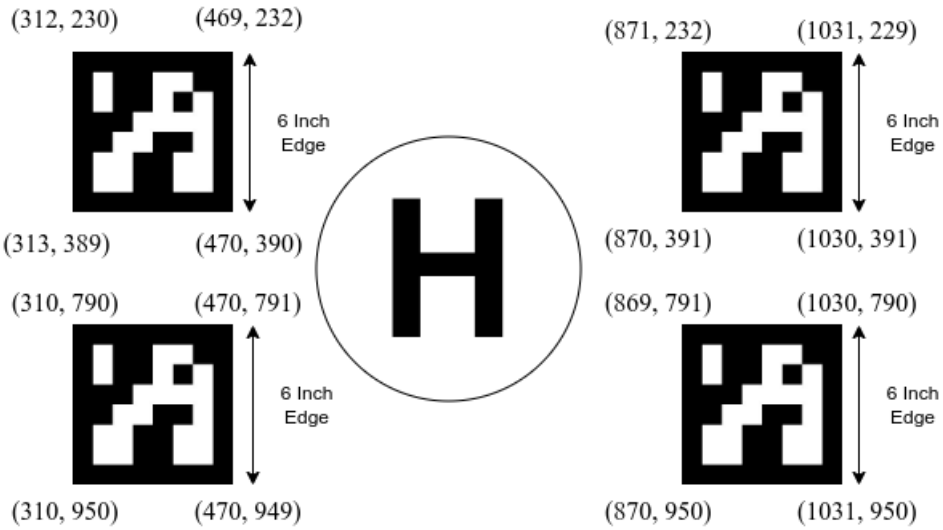



Figure 3.14: AR Tag Vertices From Landing Pad

For example, in a particular frame, the bounding box for the AR is detected at
Tag 0 [(312, 230), (469, 232), (313, 389), (470, 390)],
Tag 1 [(871, 232), (1031, 229)], (870, 391)], (1030, 391)],
Tag 2 [(310, 790), (470, 791), (310, 950), (470, 949),
Tag 3 [(869, 791), (1030, 790), (870, 950), (1031, 950)].

Now, we will calculate the Euclidean length of each edge of the tag and add them all together. The sum divided by 16 should be equal to 0.5. This is our value for pixels per foot. When the detection algorithm draws a bounding box around the trash, using the Pixel To Feet Method, we can easily calculate the area of the trash in feet and plot that information on the map.

## 3.11 Image Stitching

One other application is image stitching. So herein lies the problem. A UAV captures two-dimensional images of the three-dimensional world from roughly the same perspective, but by moving the UAV forward or backward. As we move the camera, the fields of view of the pictures overlap. We use this set of images to automatically create a bigger image covering a larger area in one single image that can be zoomed. We wish to stitch them together to make a panorama.

For this, first we apply the SIFT detector on the images so that we are able to extract features, and SIFT is ideal for this application. Then, based on the SIFT descriptor, we are going to match features between images and find some pairs of matching features.

We may get two matched feature points in these two images that have exactly the same local appearance, the same blobs, in the context of the SIFT detector. So we get a match, a perfect match, because they have the same appearance, but it just happens that they don't come from the same point in 3D. These are called outliers, and we need a method to distinguish them from the inliers. For this purpose, we used the RANSAC Algorithm.

After getting the valid matching features from RANSAC, we wish to work out the geometric relationship between pairs of images like the transformation that takes from one image to a different image, so that we are able to take a picture and warp it to the coordinate frame of the opposite image. That transformation is called homography. In particular, the transformation that we try to attain is named the Projective Transformation.

Once we are able to try this, we are able to actually warp the pictures to a standard coordinate frame. We simply select one of the images as the reference image and wrap all of the opposite images around the coordinate frame of that image. Next, we get a group of a stack of overlapping images. The problem remains that no two images are really captured with precisely the same exposure. Whether or not we managed to try and do that, it seems that the radiometric response of a

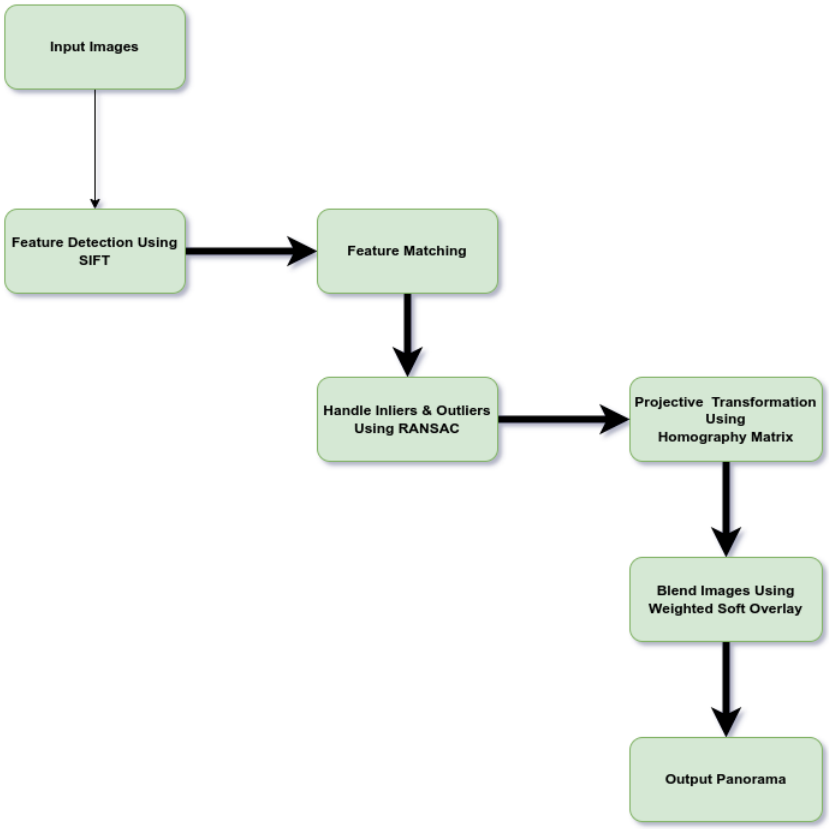



Figure 3.15: Image Stitching Workflow

lens varies slightly over the sector of view of the lens. This can be due to effects like vignetting. So it is nearly always absolutely necessary to have seams between overlapping images. To remove these seams, we need a blending algorithm, a simple algorithm that in an exceedingly literal sense removes these seams and creates one contiguous panorama of the scene. For every one, we are computing a weighting function. The weighted function we're using means that the load we assigned to a pixel is a function of its distance from the nearest boundary point. We will compute that by using the distance transform. The weight increases as our distance from the edge increases. The more we are inside a picture, the more confidence we have in the brightness of the process of blending. So a simple weighting function allows us to require these three images, which previously produced the seams, and after we blend them using that function, we get a panorama that appears fairly seamless.

# Chapter 4

# Evaluation


Figure 4.1: Training and Validation Losses

In This graph above, we can see that the losses are very minimal. The class loss on testing data accounts for only 0.297%, object loss for 1.56%, and box loss for 2.60%. Plotting the accuracy and recall values of the model in a graph of the confidence score threshold yields the Precision-Recall curve. Precision is a metric that helps us to determine how frequently the model forecast correctly. The recall is a metric that assists us to understand if the model predicted everything it should have predicted. Our model has a precision of 0.674 and a recall of 0.313.

If a model shows high recall but poor accuracy, it accurately identifies the majority of positive data but has a lot of false positives. For our case, the model has high accuracy but a poor recall means that it is correct when it identifies a sample as Positive, but only part of the positive samples is classified. Our precision-recall curve is trending higher because as the confidence rises, more forecasts and correct predictions are made. This can be further enhanced by data augmentation which will focus on in the next few weeks.
Here the recall being 77.5% and precision of 70.5% indicates that the was getting a very good amount of correct positive predictions and a majority of the positive samples are classified. Our precision-recall curve is rising because more forecasts

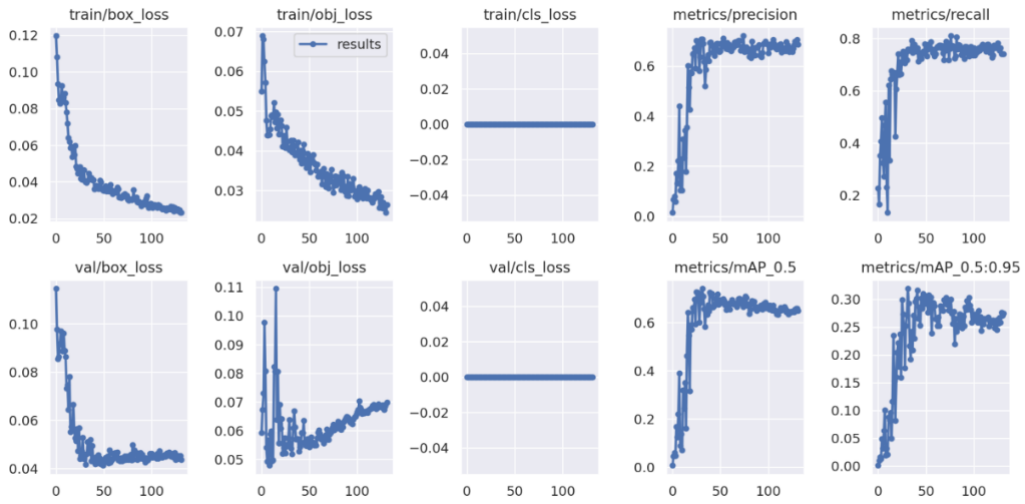



Figure 4.2: Different Evaluation Metrics for Vehicle Tracking

and correct predictions are made as confidence grows. We plan to strengthen this further via data augmentation.


Figure 4.3: Real-time Vehicle Tracking Results

From the image above we can see that the model perform very well in detecting vehicle on the road.

### 4.0.1 Helipad Detection

On the Helipad dataset, using YOLOv5, due to the weak dataset, we only acquired a 58% Mean Average Precision at 50% IOU after training the model. This model can be enhanced by adding more data and taking into account varied lighting situations.

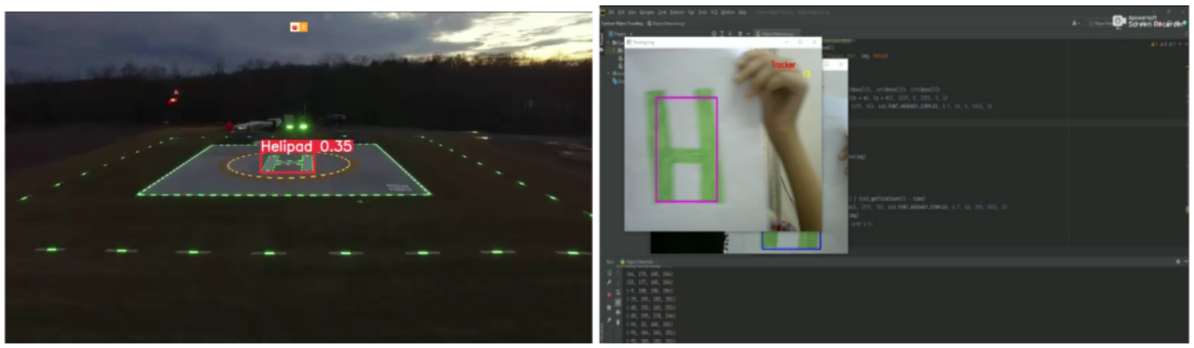



Figure 4.4: Helipad Detection

### 4.0.2 Water Trash Detection

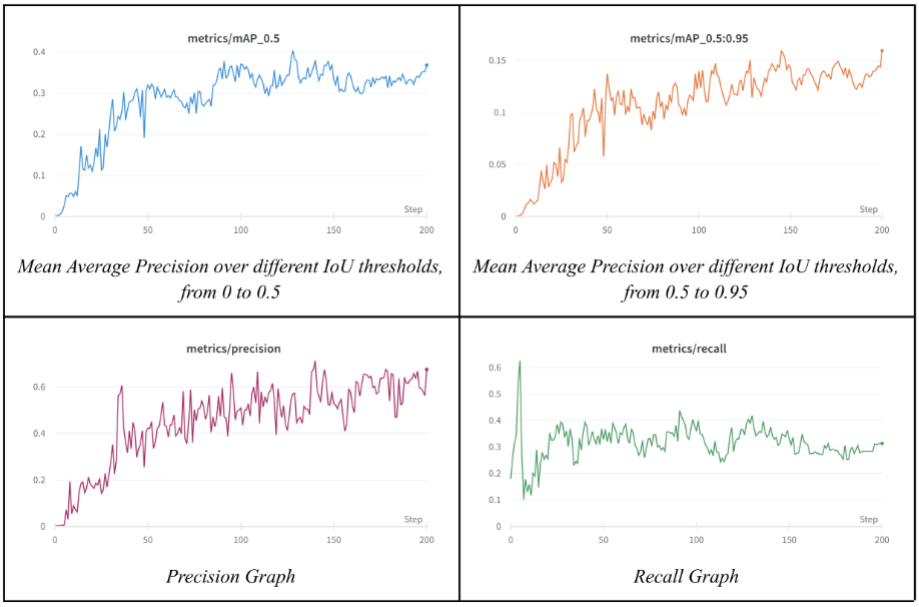



Figure 4.5: Different Evaluation Metrics for Water Trash Detection

As you can see in the above figure, the dataset gives a Mean Average Precision of 60% at 50% IOU and detects the trash on the water surface and detects the human as not trash. We plan to collect additional data with our drone in the future, which will improve the accuracy of the trash detection.

### 4.0.3 Landing & Digital Twin

In figure 4.6, we can see the drone's altitude while flying from GPS. It shows drone started to hover from ground and it went around 200mm vertically. By seeing the altitude variance, we can say that the drone struggled in the air to fly. After implementing DT, the altitude variance reduced significantly. We have plotted the range-finder's value in rqt-plot, getting from the range finder topic. It gives us the distance value in meters.

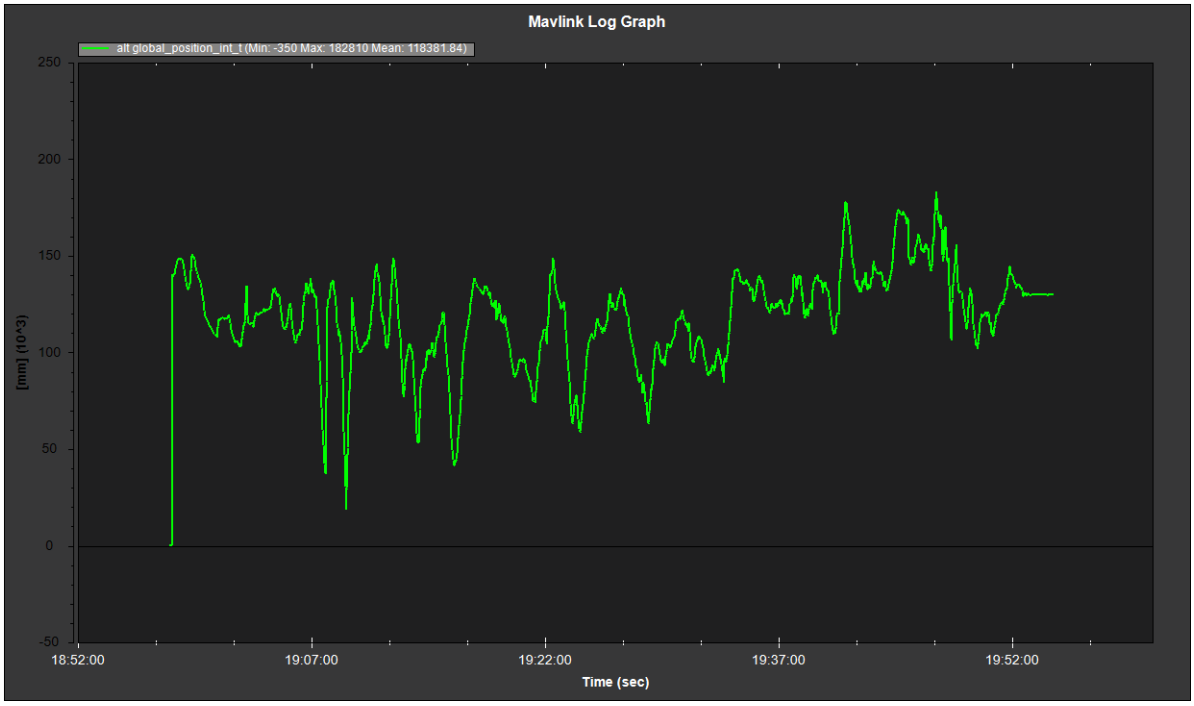



Figure 4.6: UAV altitude while flying

As we can see from the graph 4.7 the quad reached to 1.5 meter altitude and hovered for 4 seconds before it detected the marker and landed safely. The graph also shows that, the altitude variance reduced and the graph line became smooth. In the figure 4.8, we can see the drone's real-time location with its sensor data in the ground software's UI, such as: air pressure, live location, heading, speed, altitude, mode, etc., which helps us to manage and monitor air traffic. We have connected qgroundcontrol with ROS via MAVROS. Gazebo simulator is also connected with ground control software. This helped to visualize the physical entity in virtual environment.

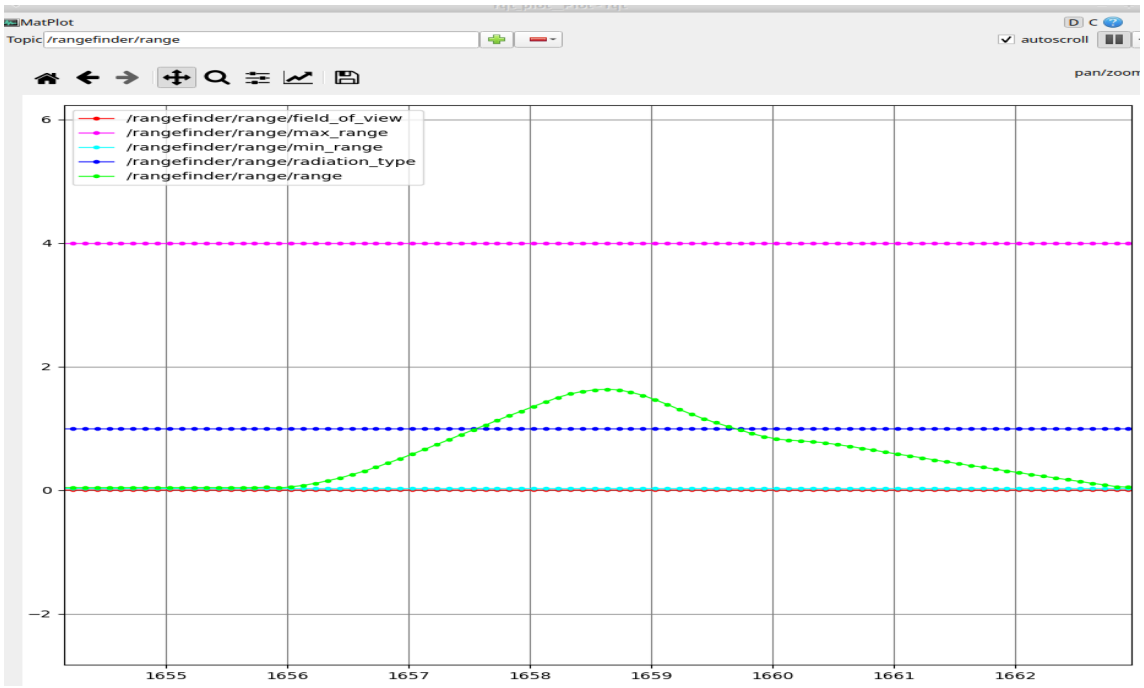



Figure 4.7: Rangefinder Analysis

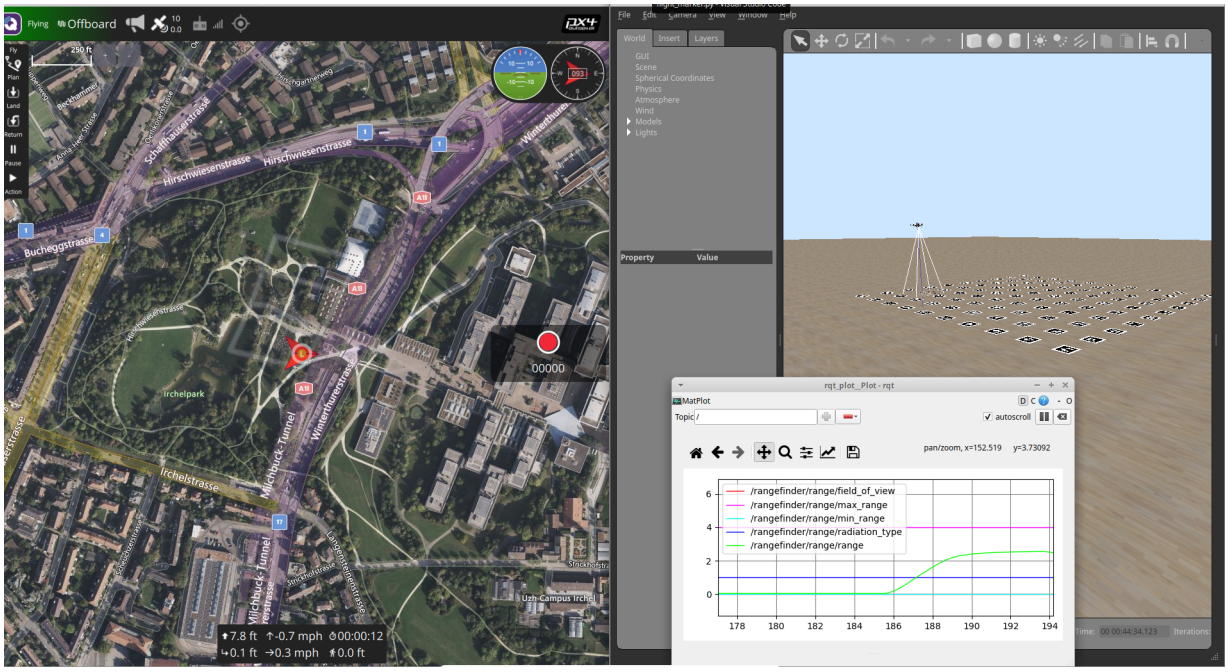



Figure 4.8: Air Traffic System

# Chapter 5

# Future Goals

## 5.1 ETP Monitoring

Bangladesh, as everybody knows, is a riverine country, facing river pollution for many years. Most of the river sides are covered in vast amounts of trash. The industries that do not adhere to the ETP are the main producers of this trash. Finding the industries that violate the ETP can make a remarkable change in river pollution.The most significant users of water in Bangladesh are the textile industries. Rivers and surface waters become contaminated when the textile is released as effluent.

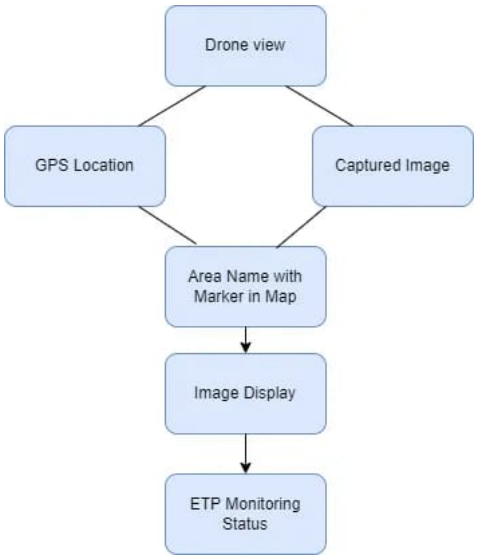



Figure 5.1: ETP Monitoring Workflow

To enforce the Effluent Treatment Plant or ETP in the Industries, the Department of Environment has worked. The government has established ETP as a requirement for receiving authorization for factory settings. According to the report, 52% of Bangladesh's industries have implemented ETP, while the remaining 48% have not. Smaller businesses frequently lack an effective ETP, and many factories discharge water directly. However, the factories that have ETPs do not operate continuously, and many of the ETPs—which are only used for display—are also inoperable. Our

future goal is to identify the industry sectors that improperly maintain ETP and discharge into river water and to mark those industries differently on the map so that the authority can get the information and take action.

## 5.2 Wireless Charging Station

The flight time of a UAV is limited due to the high current requirements of the motors. The energy required to operate a UAV is stored in a battery, often large and heavy. One UAV can only fly for a few minutes, even in a controlled environment with a fully charged battery. The flying time decreases even more if it carries additional weight, such as a camera or other gadget. In the future, we want to build a wireless charging station that doesn't require human involvement.

As we know, the energy delivery from a power source to an electrical load is wireless power transfer. Wireless transmission systems usually require two coils: one for transmission and one for reception—An electromagnetic loop antenna produces an oscillating magnetic field (copper coil). In this procedure, a specially designed transmitter transforms DC from a power supply to AC. The AC powered the transmitter's coil, which generates the magnetic field. When the receiving coil is near a magnetic field, the magnetic field can induce AC in the coil. The loop of the coil is the strength of magnetic field. Many loops in the coil can help build a strong field. A secondary wire inverts when it is introduced into a magnetic field. Because the poles' permanent magnet only works with AC, a DC input must be converted to an AC output before the coil at the transmitter can be energized. We'll focus on an inverter that will convert the DC input to the AC output. To minimize loss and increase output, we'll concentrate on the inverter.
When a drone's battery will reach less than 80% charged, it will seek a nearby charging station. After locating a nearby charging station, it will automatically fly to the station and land there using our unique precision landing algorithm. It will wait in the waiting station until the station becomes available if it is not free. Otherwise, it will land on the charging pad and begin charging. It will execute the previous command after completing the charge. If a new drone comes during this period, it will wait in the waiting station. We will utilize a scheduling technique for this. Additionally, We will create a priority queue based on the urgency of the drone's mission, which will assist in charging the drone on time.
For future work, we will work on algorithms and essential simulations for a successful wireless charging station.

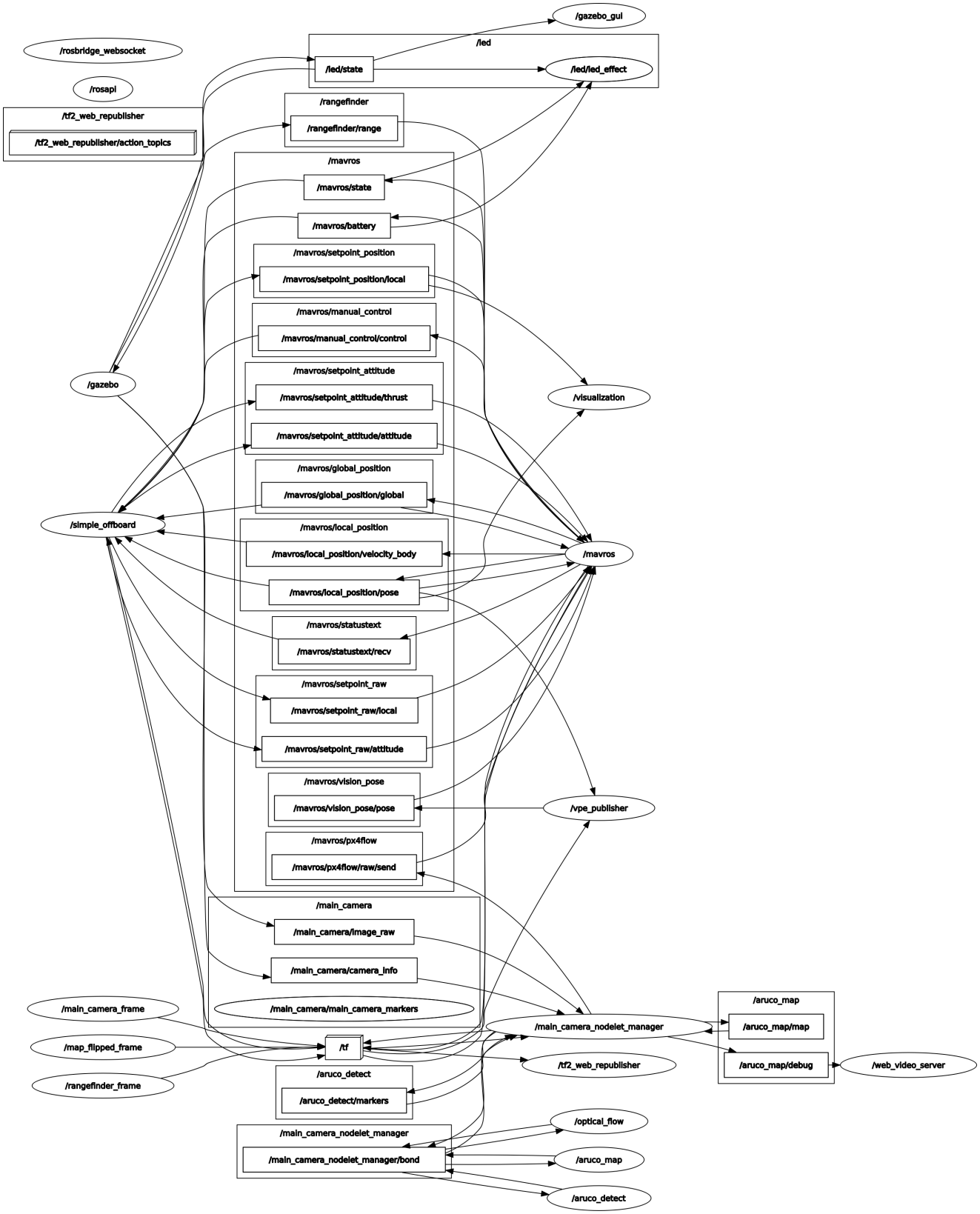



Figure 5.2: ROS Pipeline

# Chapter 6

# Conclusion

## 6.1 Conclusion

The future holds a lot of opportunities for autonomous UAVs. But it is also important to make sure that UAVs have the power to meet the rising demand for the applications of a swarm of drones. Keeping this in mind, we are working on this system for land drones autonomously with the precision of the wireless charging station, so that they can recharge and get back to the task that was assigned to them. The UAV Air Traffic Systems will allow us to increase the efficiency of a drone. We have proposed a dynamic precision landing algorithm. Along with making our environment safer we are working on detecting trash in rivers as well as monitoring effluent treatment plants.

# Bibliography


[1] S. Alam, H. A. Abbass, and M. Barlow, "ATOMS: Air traffic operations and management simulator," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 2, pp. 209–225, Jun. 2008.

[2] C. H. Choi, H. J. Jang, S. G. Lim, H. C. Lim, S. H. Cho, and I. Gaponov, "Automatic wireless drone charging station creating essential environment for continuous drone operation," in *2016 International Conference on Control, Automation and Information Sciences (ICCAIS)*, Ansan, South Korea: IEEE, Oct. 2016.

[3] M. A. Yucel and R. Y. Turan, "Areal change detection and 3D modeling of mine lakes using high-resolution unmanned aerial vehicle images," en, *Arab. J. Sci. Eng.*, vol. 41, no. 12, pp. 4867–4878, Dec. 2016.

[4] H. Zhu, S. Yuan, and Q. Shen, "Vision/GPS-based docking control for the UAV autonomous aerial refueling," in *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, Nanjing, China: IEEE, Aug. 2016.

[5] and Single European Sky ATM Research 3 Joint Undertaking, *European drones outlook study : unlocking the value for Europe*. Publications Office, 2017. DOI: doi/10.2829/085259.

[6] Z. Bao, J. Sha, X. Li, T. Hanchiso, and E. Shifaw, "Monitoring of beach litter by automatic interpretation of unmanned aerial vehicle images using the segmentation threshold method," en, *Mar. Pollut. Bull.*, vol. 137, pp. 388–398, Dec. 2018.

[7] A. M. Jawad, H. M. Jawad, R. Nordin, S. K. Gharghan, N. F. Abdullah, and M. J. Abu-Alshaeer, "Wireless power transfer with magnetic resonator coupling and sleep/active strategy for a drone charging station in smart agriculture," *IEEE Access*, vol. 7, pp. 139 839–139 851, 2019. DOI: 10.1109/ACCESS.2019.2943120.

[8] P. Koutalakis, O. Tzoraki, and G. Zaimes, "UAVs for hydrologic scopes: Application of a low-cost UAV to estimate surface water velocity by using three different image-based methods," en, *Drones*, vol. 3, no. 1, p. 14, Jan. 2019.

[9] A. Al-Mousa, B. H. Sababha, N. Al-Madi, A. Barghouthi, and R. Younisse, "UTSim: A framework and simulator for UAV air traffic integration, control, and communication," en, *Int. J. Adv. Robot. Syst.*, vol. 16, no. 5, p. 172 988 141 987 093, Sep. 2019.

[10] Z. Zhao, C. Luo, J. Zhao, Q. Qiu, M. C. Gursoy, C. Caicedo, and F. Basti, "A simulation framework for fast design space exploration of unmanned air system traffic management policies," in *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, Herndon, VA, USA: IEEE, Apr. 2019.

[11] N. Grigoropoulos and S. Lalis, "Simulation and digital twin support for managed drone applications," in *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2020, pp. 1–8. DOI: 10.1109/DS-RT50469.2020.9213676.

[12] S. Obayashi, Y. Kanekiyo, and T. Shijo, "UAV/drone fast wireless charging FRP frustum port for 85-khz 50-V 10-A inductive power transfer," in *2020 IEEE Wireless Power Transfer Conference (WPTC)*, Seoul, Korea (South): IEEE, Nov. 2020.

[13] T. Zhou and C. Huang, "UAV automatic docking technology based on deep learning," in *2020 International Conference on Computing and Data Science (CDS)*, Stanford, CA, USA: IEEE, Aug. 2020.

[14] A. Benjumea, I. Teeti, F. Cuzzolin, and A. Bradley, "YOLO-Z: Improving small object detection in YOLOv5 for autonomous vehicles," Dec. 2021. arXiv: 2112.11798 [cs.CV].

[15] B. Caruso, M. Fatakdawala, A. Patil, G. Chen, M. Wilde, ; P. Luong, F. Gagnon, L.-N. Tran, and F. Labeau, "Deep reinforcement Learning-Based resource allocation in cooperative UAV-Assisted wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, pp. 7610–7625, 2021.

[16] M. Kasper-Eulaers, N. Hahn, S. Berger, T. Sebulonsen, Ø. Myrland, and P. E. Kummervold, "Short communication: Detecting heavy goods vehicles in rest areas in winter conditions using YOLOv5," en, *Algorithms*, vol. 14, no. 4, p. 114, Mar. 2021.

[17] ——, "Short communication: Detecting heavy goods vehicles in rest areas in winter conditions using YOLOv5," en, *Algorithms*, vol. 14, no. 4, p. 114, Mar. 2021.

[18] P. Luong, F. Gagnon, L.-N. Tran, and F. Labeau, "Deep reinforcement learning-based resource allocation in cooperative UAV-assisted wireless networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 11, pp. 7610–7625, Nov. 2021.

[19] Z. Lv, D. Chen, H. Feng, R. Lou, and H. Wang, "Beyond 5g for digital twins of uavs," *Computer Networks*, vol. 197, p. 108 366, 2021, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2021.108366. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621003534.

[20] M. Mohammadi, M. Rashidi, V. Mousavi, A. Karami, Y. Yu, and B. Samali, "Quality evaluation of digital twins generated based on uav photogrammetry and tls: Bridge case study," *Remote Sensing*, vol. 13, no. 17, 2021, ISSN: 2072-4292. DOI: 10.3390/rs13173499. [Online]. Available: https://www.mdpi.com/2072-4292/13/17/3499.

[21] N. Svahn and P. Bergstedt, "A comparison between remote and physically co-located, plane and AR tag, as well as 2D and 3D supervision in a collaborative AR-environment," en, Ph.D. dissertation, 2021.

[22] M. A. Isgró, M. D. Basallote, and L. Barbero, "Unmanned aerial system-based multispectral water quality monitoring in the iberian pyrite belt (SW spain)," en, *Mine Water Environ.*, vol. 41, no. 1, pp. 30–41, Mar. 2022.

[23] W. Sun, N. Xu, L. Wang, H. Zhang, and Y. Zhang, "Dynamic digital twin and federated learning with incentives for air-ground networks," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 321–333, 2022. DOI: 10.1109/TNSE.2020.3048137.