Energy Efficient Path Planning for Aerial Photography Based Disaster Response

B. Tech. Final year project Report

By

Mangu Singh	**(Roll No. B190641EE)**
Aayush Patidar	**(Roll No. B190458EE)**
Akshay Nirmal	**(Roll No. B190939EE)**
Piyush Agarwal	**(Roll No. B190889EE)**

Under the guidance of

Rakesh R Warier

Department of Electrical Engineering

NATIONAL INSTITUTE OF TECHNOLOGY CALICUT

NIT Campus P.O., Calicut - 673601, India

2022

CERTIFICATE

This is to certify that the Project entitled “Energy Efficient Path Planning For Deep Learning Based Aerial Surveying” is a bonafide record of the work done by Piyush Agarwal, Mangu Singh, Aayush Patidar, Akshay Nirmal, under my supervision and guidance, in partial fulfilment of the requirements for the award of Degree of Bachelor of Technology in Electrical and Electronics Engineering from National Institute of Technology Calicut for the academic year 2023.

Dr. RAKESH R WARIER
Assistant Professor
Dept. of Electrical Engineering

Dr. PREETHA P
Associate Professor & Head
Dept. of Electrical Engineering

Place: NITC
Date: 09.12.2022

ACKNOWLEDGEMENT

The accomplishment of this report benefits from the help and direction from our project guide **Dr. RAKESH WARIER**, Assistant Professor, Department of Electrical Engineering to whom we express our heartfelt gratitude. I also pay a deep sense of gratitude to **Dr PREETHA P**, Associate Professor and Head Dept. of Electrical Engineering for encouraging and providing us with the opportunity to prepare this project. I would also like to extend our gratitude to all those who have helped and supported us.

Piyush Agarwal

Mangu Singh

Aayush Patidar

Akshay Nirmal

ABSTRACT

In this project deep learning based aerial surveying of an area has been performed.For aerial inspection a UAV is needed. The primary goal is to manage the UAV flight path in such a way that the energy consumed by the UAV should be as minimal as possible. After the efficient path planning the data can be collected and then the data can be processed. With the help of processed data it can be concluded that if the area is affected by a calamity.

Coverage path planning is the operation of finding a path that covers all the points of a specific area. Thanks to the recent advances of hardware technology, Unmanned Aerial Vehicles (UAVs) are starting to be used for photogrammetric sensing of large areas in several application domains, such as agriculture, rescuing, and surveillance. However, most of the research focused on finding the optimal path taking only geometrical constraints into account, without considering the peculiar features of the UAVs, like available energy, weight, maximum speed, sensor resolution, etc. This paper proposes an energy-aware path planning algorithm that minimises energy consumption while satisfying a set of other requirements, such as coverage and resolution. The algorithm is based on an energy model derived from real measurements. Finally, the proposed approach is validated through a set of experiments.Unmanned aerial vehicles (UAVs) are frequently adopted in disaster management. The vision they provide is extremely valuable for rescuers. However, they face severe problems in their stability in actual disaster scenarios, as the images captured by the on-board sensors cannot consistently give enough information for deep learning models to make accurate decisions. In many cases, UAVs have to capture multiple images from different views to output final recognition results. In this paper, we desire to formulate the fly path task for UAVs, considering the actual perception needs. A convolutional neural networks (CNNs) model is proposed to detect and localise the objects, such as the buildings, as well as an optimization method to find the optimal flying path to accurately recognize as many objects as possible with energy awareness. The simulation results demonstrate that the proposed method is effective and efficient, and can address the actual scene understanding and path planning problems for UAVs in the real world well.

CONTENTS

List of Abbreviations

CNN– Convolutional Neural Networks

UAV– Unmanned Aerial Vehicle

DL– Deep Learning

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

The unmanned aerial vehicle (UAV) is often used in disaster management. For example, after a disaster, people can use the UAV to detect the disaster area to find the damaged buildings and residents; and in the disaster recovery stage, the UAV can be used to assess the severity of the disaster, the feasibility, and cost of the reconstruction. They are utilised to assist people in disaster reconstruction planning. However, a serious problem with UAV is that their battery is limited and cannot be used for long-term or large-scale survey tasks. Therefore, how to develop a feasible flight path for efficient and accurate target detection has become a very important issue. This task mainly consists of an important aspect of energy aware path planning.

1.1 OBJECTIVES

1. Search and Rescue Operation
2. To Design energy efficient UAV trajectory for covering area of interest:
   Here we will design an energy efficient trajectory for covering areas of interest by using several techniques. For example taking path where the turns ratio is minimum.
3. To Use trained CNN for processing UAV collected images:
   Now our task is to use a trained CNN algorithm to process UAV collected images.We explore the use of different CNN architectures and using different hyperparameters.

1.2 UAV Remote Sensing and CNN

To begin with UAVs. Unmanned Aerial Vehicle – UAV popularly known as drone, is an airborne system or an aircraft operated remotely by a human operator or autonomously by an onboard computer. There are two broad classes of UAVs – Fixed wing and Rotary based.
For surveying purposes we are combining both UAV and remote sensing technology.*Remote sensing* is the process of detecting and monitoring the physical characteristics of an area by measuring its reflected and emitted radiation at a distance (typically from satellite or aircraft). Special cameras collect remotely sensed images, which help researchers "sense" things about the Earth. Cameras on satellites and aeroplanes take images of large areas on the Earth's surface, allowing us to see much more than we can see when standing on the ground.And sonar systems on ships can be used to create images of the ocean floor without needing to travel to the bottom of the ocean are some example UAV remote sensing.

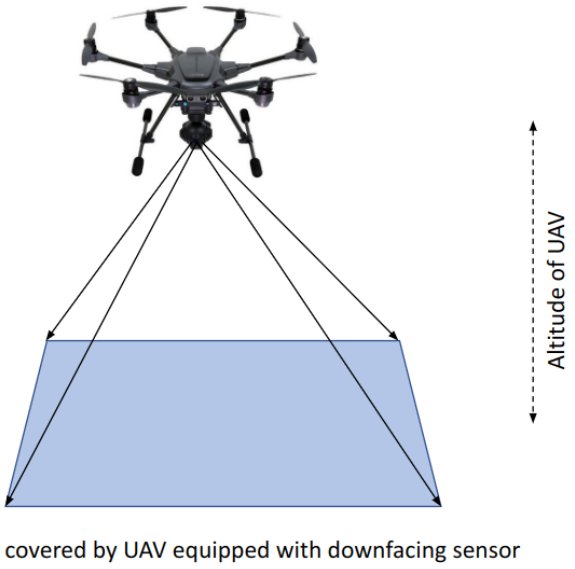


Fig. 1.1: UAV remote sensing

UAV combined with remote sensing technology have been creating new vistas in global scenarios to acquire the geospatial data on land resources and environment. The imagery obtained from UAVs can immensely support many applications ranging from large-scale mapping, urban modelling to vegetation structure mapping. Specifically in the NE region of our country with limited connectivity and difficult terrain conditions, the local planning and developmental activities can be greatly improved by the UAV survey.

The remote-sensing community is always committed to developing remote-sensing methods for improving the performance of aspects, such as preprocessing, segmentation, and classification. Neural networks, the basis of deep learning (DL) algorithms, have been used in the remote sensing community for many years.The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs, it can learn the key features for each class by itself.convolutional neural networks can extract informative features from images, eliminating the need of traditional manual image processing methods.

1.3 NEED of ENERGY EFFICIENCY

Battery-powered UAVs are the most convenient and inexpensive. However, the energy of battery-powered UAV is limited and can only provide a short flight time. Therefore, for a large monitored area where the nodes to be visited are distributed, we hope that the UAV can finish as many tasks as possible in the duration of flight. How to reduce the energy consumption of the UAV and make it collect data from as many sensor nodes as possible in the duration of flight?

That has become an important challenge which UAV-assisted data collection methods have to confront in practical applications.

If the energy efficiency of the UAV can be improved, more tasks will be completed and more sensor nodes will be visited in the duration of flight. This can also reduce the number of UAVs and the cost of input in an application scene where multiple UAVs alternately or collaboratively collect data. Conversely, if the energy efficiency of the UAV is not high, the recharge cycles of the UAV will increase and data collection will be delayed, which results in not only collecting data not in time but also losing data (e.g., for a sensor node, the new sensed data can erase the old data that has not been collected).

Improving the energy efficiency of a UAV has important practical significance in other aspects. For example, when the UAV is used to search and rescue the trapped people in an earthquake and other disaster sites, it can make the UAV visit as many search-and-rescue points as possible in the duration of flight and save the search-and-rescue time and more lives.

CHAPTER 2

METHODOLOGY

2.1 ENERGY MODEL:

Given the large variety of drones, each with specific physical characteristics, like weight, type of power supply, propellers, etc., deriving a general parametric energy model that can be used to predict the energy consumption in different operating conditions is a hard task. In this work, we propose a method that can be used to model and analyse the energy consumption of a specific drone as a function of its speed and operating conditions.

In this section, we present the drone energy-consumption model for a UAV that needs to traverse a set of waypoints in a surveyed area to perform data collection tasks. The UAV is a quadrotor with quad-core 64-bit, 2.56 GHz processor and 3300 mAH Li-Po battery. We used the energy model proposed by Lige Ding to formulate the energy consumption problem. As the energy consumption of a drone depends on the drone speed, the energy consumption model needs to consider the different flight stages including acceleration, deceleration, hovering, and turning. The consumed power is calculated by multiplying the supply voltage and the current, which is measured by an energy measurement module. The experiments conducted by Lige Ding provided a better understanding of the energy performance of a drone.

2.1.1 Effect of velocity:

The UAV was commanded to fly in a straight line at different constant speeds. The power consumption measured at 2 m/s, 4 m/s, 6 m/s and 8 m/s were 242W, 245W, 246W and 268W, respectively.

Power consumption during acceleration and deceleration was recorded and can be used to calculate energy consumption when UAV accelerates and decelerates. Similar readings of power consumption were observed during the acceleration and deceleration phases as observed in effect of velocity. As velocity increases, power consumption during acceleration increases and vice versa.

Energy consumption model for flying straight When a UAV flies from a starting point to a target point, it goes through three phases: acceleration, uniform speed, and deceleration. We can calculate energy consumption at different speeds and distances E(v,d)

$$E(v,d) = \int_0^{t_1} P_{acc}dt + \int_{t_1}^{t_2} P(v)dt + \int_{t_2}^{t_3} P_{dec}dt \quad (1)$$

2.1.2 Effect of Turning:

In this experiment, the power consumption of the UAV was recorded when it rotated at four different angles 45, 90, 135, 180. It was assumed that an angular speed of ω_{turn} (2.07 rad/sec)

would require P_{turn} (260 W) for the turn. The energy consumption at turning angle Δθ can be calculated by

$$E_{turn} = P_{turn} \frac{\Delta\theta}{\omega_{turn}} \quad (2)$$

where E_{turn} denotes Energy consumption during turn, Pturn denotes Power consumption during turn, Δθ denotes turning angle, !turn denotes angular velocity during turn.

When a UAV travels for a short distance, it is not able to achieve a uniform speed because it directly goes from the accelerating phase to decelerating phase. The 1st equation can be used to evaluate the optimal drone speed. When the distance is too short, the drone will take its whole time accelerating and decelerating, and when the distance between two-way points is large enough, drones can go through all flight phases and reach an optimal speed. Once the path length is ideal, the drone can remain at the optimal speed.

2.2 CNN:

CNNs are most frequently used for image classification, such as detecting highways in satellite photos or categorising handwritten characters and numbers. In addition to these more commonplace tasks, CNNs also excel at signal processing and image segmentation.

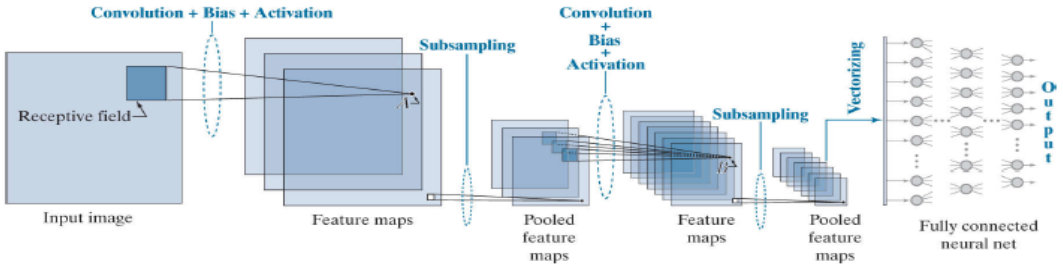


Fig. 2.1: CNN architecture

Receptive field selects a region of pixels in the input image. The receptive field is moved over the image and, at each location, forming a sum of products of a set of weights and the pixels contained in the receptive field.The number of spatial increments by which a receptive field is moved is called the stride. Our spatial convolutions in previous chapters had a stride of one, but that is not a requirement of the equations themselves.When repeated for all locations in the input image, this results in a 2D set of values that we store in the next layer as a 2-D array, called a feature map.

Max pooling operation calculates the maximum value for patches of a feature map, and uses it to create a downsampled (pooled) feature map. Translating the image by a small amount

would not significantly affect the values of most pooled outputs (Translational Invariance). Could result in loss of accurate spatial information

The nature of the learned features is determined by the kernel coefficients. Note that the contents of the feature maps are specific features detected by convolution. For example, some of the features emphasise edges in the character. The pooled features are lower-resolution versions of this effect. If you look at each map carefully, you will notice that it highlights a different characteristic of the input. For example, the map on the top of the first column highlights the two principal edges on the top of the character. The second map highlights the edges of the entire inner region, and the third highlights a "blob-like" nature of the digit, almost as if it had been blurred by a low pass kernel. The other three images show other features. Although the pooled feature maps are lower-resolution versions of the original feature maps, they still retained the key characteristics of the features.

2.3 PATH PLANNING:

When we talk about the energy efficient path planning algorithm, the first thing we can consider for the path planning is that the distance travelled by the UAV should be minimum and second thing we can consider is to minimise the number of turns as the figures given below figure a has more number of turns as compared to figure b. So the second path planning would be better.

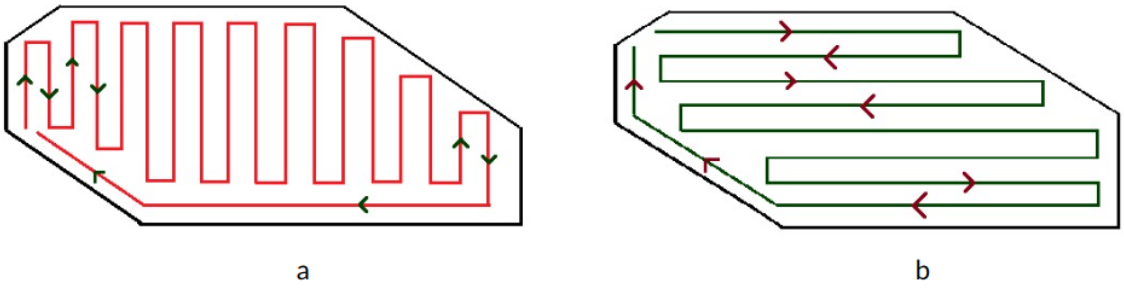


Fig. 2.2: Two ways of path planning for an area

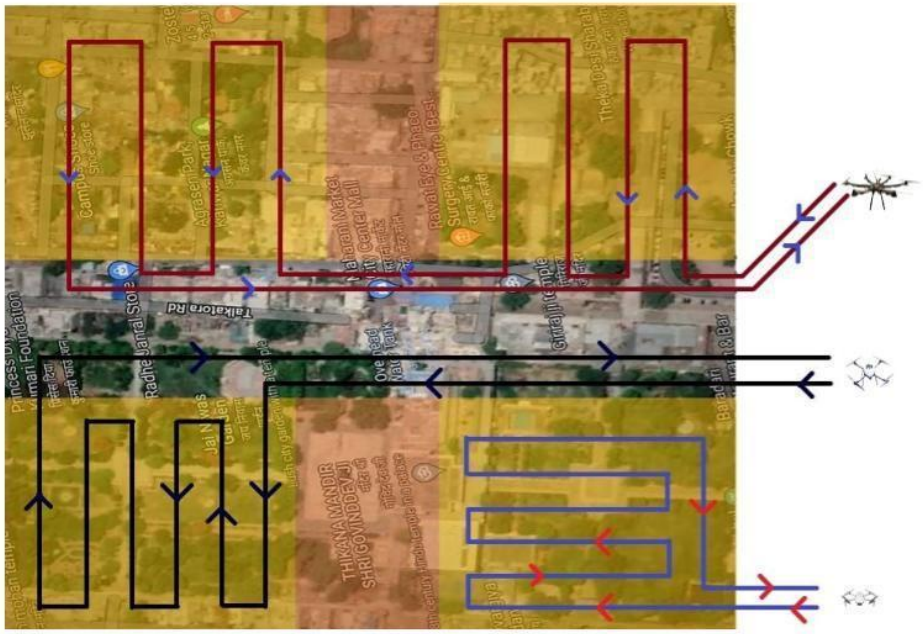


Fig. 2.3: Collective data collection by different types of UAVs

For an example we want to do a survey of the above given area.We only need to monitor the area which is in yellow. We are not allowed to enter into the red part so we can say that the red part is a restricted area. For collecting image data from the yellow part we have three UAVs in total. So our task is to combine UAVs in such a way so that the energy consumed by them is minimal.

Hexacopter UAV, DJI inspire 1 UAV, DJI F450 UAV all of them have different power capabilities, different number of sensors along with different load batteries. The Hexacopter UAV is larger in size and has more flight time so it monitors more area as compared to the others, as we can see from the figure also. Similar kinds of behaviours are followed by the DJI inspire1 and the DJI F450 also. In comparison to DJI F450, DJI inspire1 has more flight time so it is monitoring farther areas.

CHAPTER 3

Simulations and Discussions

3.1 Search and Rescue:

The real power that drones provide to rescue operators is the easy access to aerial data of a large area, which gives the rescue team the ability to expedite the process of finding a missing person, where every second count.Drones can carry different types of payloads that can be used in different situations. Two of the popular payloads are the 4K wide-angle camera and the thermal camera that are extensively used during search and rescue missions.HD video from a drone is not that useful in a search and rescue mission, since the resolution is lost when looking at a still image. On the other hand, a high resolution still image can provide valuable information to the people on the ground looking for the missing person. This is why cameras that can capture high resolution still images are preferred.Thermal cameras are also used in search and rescue missions, especially during night time missions.

3.2 Image Data Collection:

AIDER (**A**erial **I**mage **D**ataset for **E**mergency **R**esponse applications): The dataset construction involved manually collecting all images for four disaster events, namely Fire/Smoke, Flood, Collapsed Building/Rubble, and Traffic Accidents, as well as one class for the Normal case.

The aerial images for the disaster events were collected through various online sources (e.g. google images, bing images, youtube, news agencies websites, etc.) using the keywords ”Aerial View” or ”UAV” or”Drone” and an event such as Fire”,”Earthquake”,”Highway accident”, etc. Images are initially of different sizes but are standardised prior to training. All images were manually inspected to first contain the event that was of interest and then to have the event centred at the image so that any geometric transformations during augmentation would not remove it from the image view. During the data collection process the various disaster events were captured with different resolutions and under various conditions with regards to illumination and viewpoint. Finally, to replicate real world scenarios the dataset is imbalanced in the sense that it contains more images from the Normal class. This subset includes around 500 images for each disaster class and over 4000 images for the normal class. This makes it an imbalanced classification problem.

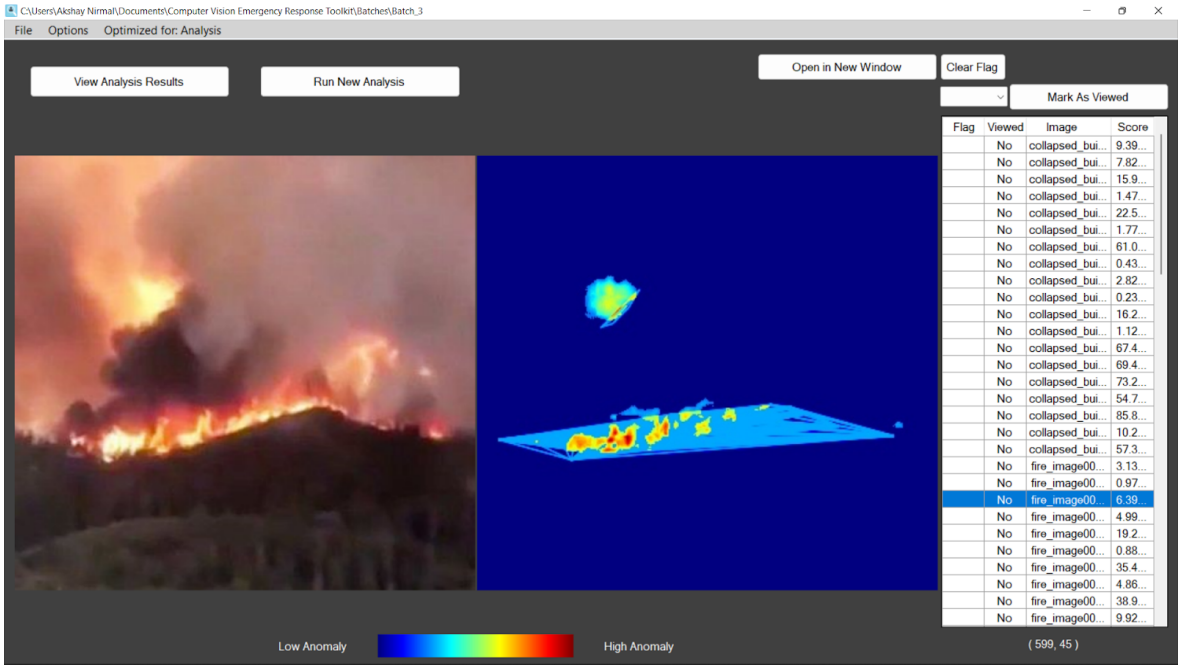
It is advised to further enhance the dataset that random augmentations are probabilistically applied to each image prior to adding it to the batch for training. Specifically there are a number of possible transformations such as geometric (rotations, translations, horizontal axis mirroring, cropping and zooming), as well as image manipulations (illumination changes, colour shifting, blurring, sharpening, and shadowing).

This dataset is associated with the following publications: C. Kyrkou and T. Theocharides, "EmergencyNet: Efficient Aerial Image Classification for Drone-Based Emergency Monitoring Using Atrous Convolutional Feature Fusion," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 1687-1699, 2020, doi: 10.1109/JSTARS.2020.2969809. C. Kyrkou and T. Theocharides, "Deep-Learning-Based Aerial Image Classification for Emergency Response Applications Using Unmanned Aerial Vehicles," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 2019, pp. 517-525, doi: 10.1109/CVPRW.2019.00077.

Fig. 3.1: Sample images of disasters such as fire, flood, building collapse and road accidents

3.3 Image Analysis

For image analysis software named as Computer Vision Emergency Response Toolkit is used.The Computer Vision Emergency Response Toolkit is an application that allows users to detect features of interest in high-resolution photos obtained from UAV fly-overs. Users can run an analysis job by selecting a single image, a set of images, or a folder of images through the interface provided. Upon completion of the analysis of the image(s) provided, users can view the features of interest side-by-side with the original image, also through the interface provided. The ability to view jobs and their respective image heat map pairs is available anytime, even while a job is running. In addition, there is a checkbox feature provided that enables users to keep track of the images they have reviewed and what is left to be done.



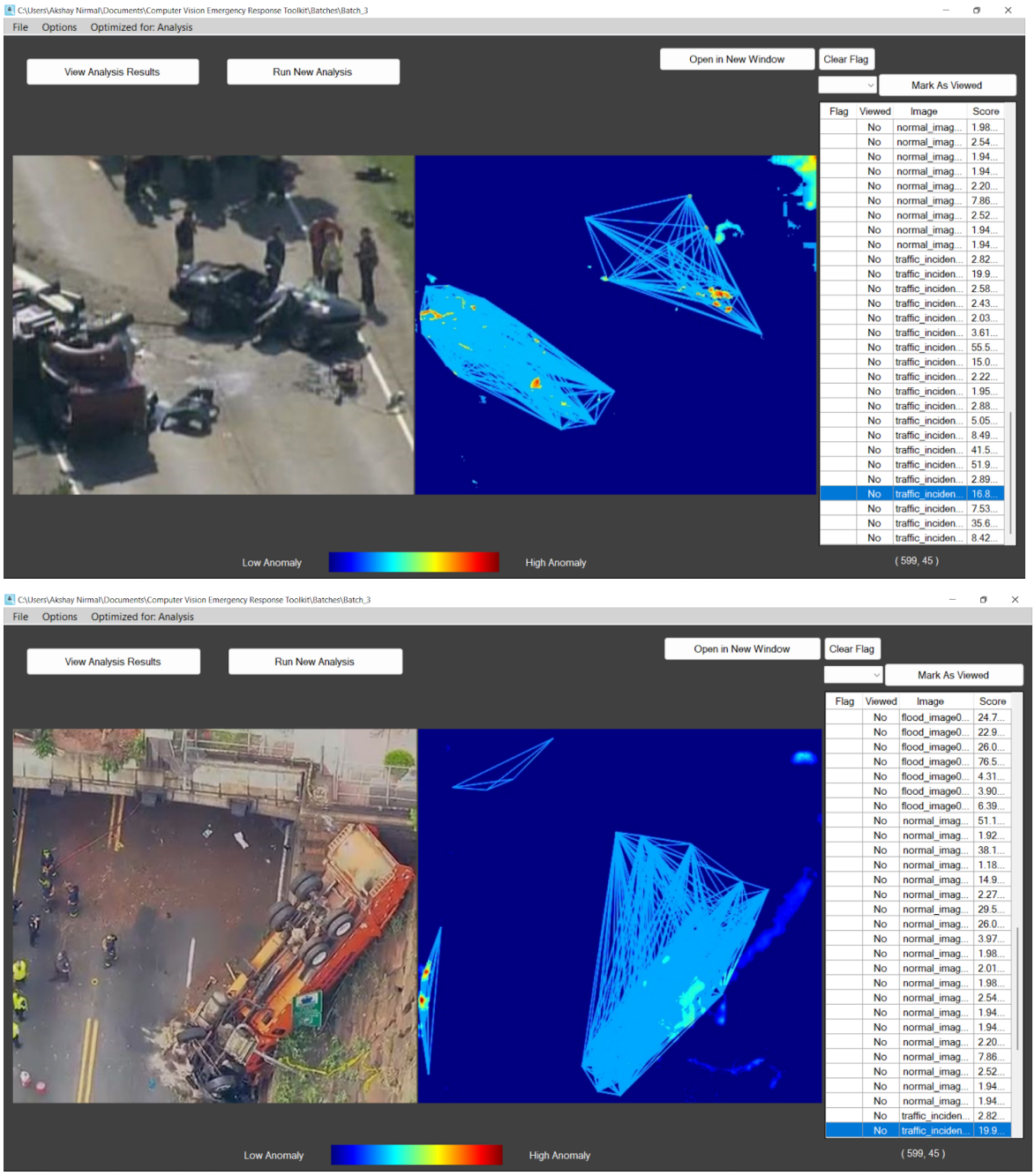


Fig 3.2: Sample images obtained after the analysis

CHAPTER 4

Pluto 1.2

4.1 Introduction

Pluto aims at providing an ultimate user experience in terms of flight experience as well as tinkering. We believe that tinkerers have great ideas. Some of which can even change the world. PlutoX is a great prototyping tool for implementing our ideas. Tinkering is an interesting hobby and the development tool plays a very important role in making it an engrossing experience. A basic development tool will provide a platform where the tinkerer has to begin from scratch every time and gradually start implementing their ideas, which surely is time consuming. However, if a development tool provides an effortless platform for the tinkerers to directly develop their ideas on, without the need of worrying about the rest of the system, it will definitely be more productive. With this perspective, we do not want our tinkerers to waste their energy in figuring out how to get the drone to fly but to focus only on what they want to do and achieve with the drone . For this purpose, we have done all the work required for Pluto to fly and hence, tinkerers do not need to worry about that. Even without any tinkering, a user can easily fly Pluto. The entire system has been designed in such an elegant way that the tinkerers can directly start implementing their ideas by coding into the platform provided to them.

4.2 Hardware

By hardware, we essentially refer to electronics. The flight controller of PlutoX is called PrimusX. Primus X comes loaded with many features.

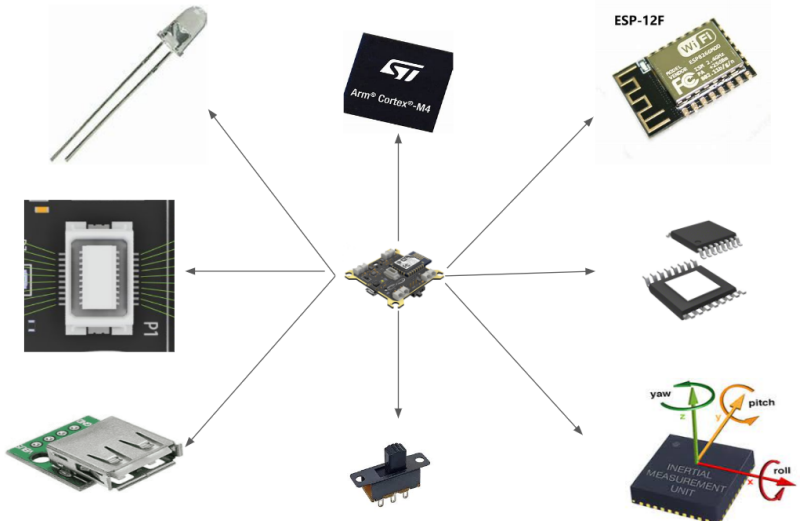


Figure 4.1
PRIMUSX Components

Microcontroller: PrimusX has a 32 bit ARM Cortex-M4 processor. It has 256Kb of flash. Approximately 178 Kb is available for user code and 78kb is used by drone's firmware.

Wireless module: PrimusX uses ESP12F module for wireless communication. It generates a hotspot for connecting your computer and your phone. This helps in wirelessly flashing your code onto PlutoX and also for flying it with your phone using Pluto Controller app.
Inertial Measurement Unit (IMU): PrimusX has a 9-axis IMU with three inbuilt core sensors - Rate-gyro, Accelerometer and Magnetometer. These three sensors are crucial as they provide the information regarding Pluto's current orientation, velocity and acceleration. Primus also comes with a Barometer sensor located at the bottom side. It measures pressure along with the temperature. The working of each of these sensors is explained in detail in their respective projects.
Motor drivers: PrimusX comes with 8 motor drivers M1 to M8. Four of these drivers, M1 - M4 are reversible drivers (H-bridge) and the other four, M5 - M8 are unidirectional drivers. All the motor ports are labelled on the Primus X board.
LEDs: PrimusX contains three sets of LEDs:

1. Power LED: Which indicates whether PlutoX is powered ON or not.
2. Status LEDs: These are the typical RGB LEDs which are useful for indicating status as per the requirements.
3. Charging LED: This turns on when PlutoX is charging. When the charging is finished, the LED dims.

USB port: PrimusX is mounted with a USB port for charging the battery. It can also be used for DFU based flashing.
Switch: At the back side of Primus X is the ON-OFF slide switch.
Camera port: PlutoX comes with a camera module which contains its own WiFi and UART for communication with the microcontroller. The ESP12F also contains its WiFi and communicates with the microcontroller by UART. This could cause trouble in communication with two channels communicating simultaneously. When Pluto is switched ON, the ESP12F therefore checks for any connection on the camera port. If the port is not connected with the camera module, the ESP module works as usual. If the camera port is connected, the ESP module detects it and shuts itself down, with the camera module taking over all the tasks carried out by the ESP12F module.
RC/LED port: This port is under development. An external RC can be connected to Pluto. This port can be used to connect the receiver of the controller. One way communication is possible through this port. This port can also be used to connect addressable LEDs.
Unibus: PlutoX is all about tinkering. For this purpose, PrimusX comes with a 20 pin unibus for integrating external hardware like sensors.

4.3 Software

All the hardware is of no use unless coupled with the software, and this combination together makes PlutoX extraordinary. It was mentioned earlier that the tinkerers do not have to worry about programming PlutoX for ordinary flights. PlutoX runs on Magis, which provides stable flight.

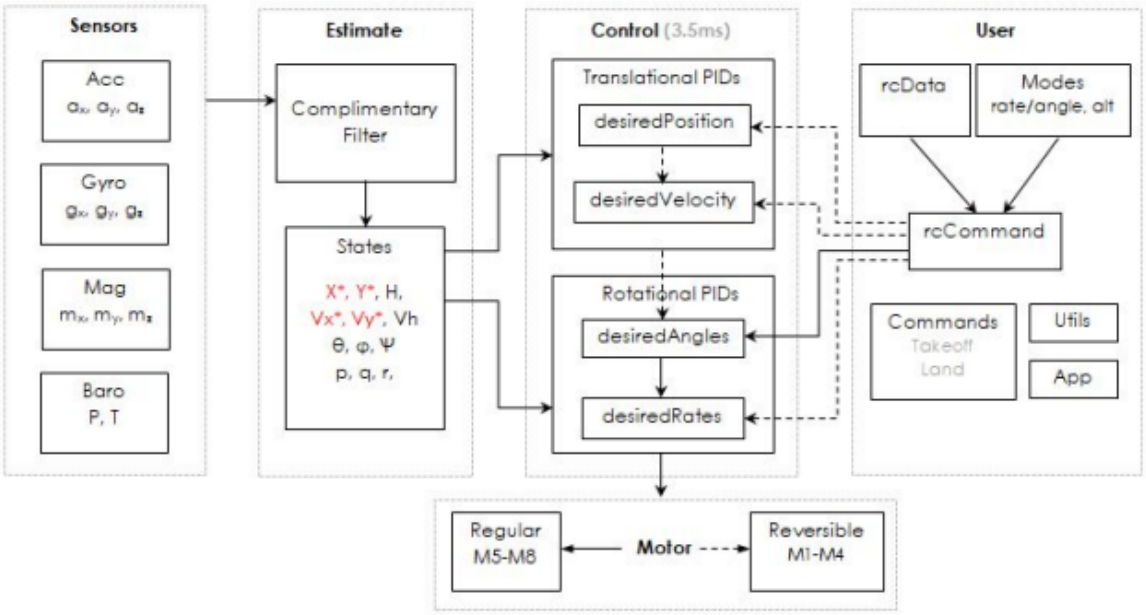


Figure 4.2
Working of Magis

Sensors Block: In order to control any system, it is required to know the current condition of that system. Condition could refer to a number of variables that a system can have. The smallest set of variables which completely describe a system are called state variables. The value of these state variables can be obtained by using sensors.
It is important to know that all the data provided by each of these sensors is very noisy. Such raw data can not be directly utilised in estimation of the states of the drone. The data provided in Sensor Block is forwarded to Estimate Block where further processes take place. Thus, the Sensor Block will contain all the raw data provided by each sensor. Including this block as a header while coding gives access to all the data contained in this block. So, for example, the current reading of temperature or pressure can be accessed by using the "Sensor.h" header.

Estimate Block: The data received from the Sensors Block is raw, containing a lot of noise. This noise is filtered out by the complementary filter and the filtered data is used for estimating the values of state variables. The values of certain state variables are not obtained directly through the sensors. These values are then estimated in this block by using other state variables' estimations as they are interdependent.
Estimate block contains all the filtered data regarding the current angle, rate, position and velocity, which is then fed to the Control Block. The access to all the filtered data can be gained by using the header "Estimate.h". This data can be utilised in programming new features. For example, suppose a feature where in PlutoX should not tilt beyond a specific angle of roll. The limiting angle of roll can be mentioned based on choice but it will have to be compared with the current angle of roll. In order to get the current angle of roll, the data can be accessed from the Estimate Block and that will be done by using the header "Estimate.h".

User Block: This block represents the input provided by the user or the pilot. A pilot can use the Pluto Controller app for flying PlutoX. Using the app, the pilot will be controlling the pitch, roll, yaw and thrust movements of PlutoX. Let us understand the purpose of this block by taking an example of pitch. Suppose, the pilot wants to pitch forward. So the pilot moves the right joystick in the front direction. The magnitude of the forward movement of the joystick will be taken as the data for rcData. This data is used to generate rcCommand. It is the modified rcData after applying low pass filters, exponential, etc. rcData is converted into a user-friendly range in rcCommand. rcCommand is further given as input data to the Control Block and the further process takes place over there.

Control Block: Control Block, as the name suggests, controls the movement of the drone across all the axes, both translational and rotational. This is achieved by using PIDs. This block receives input data from Estimate Block and from User Block. The Estimate Block provides the current values of the state variables of PlutoX. The User Block provides the desired values of the state variables as received from the user.

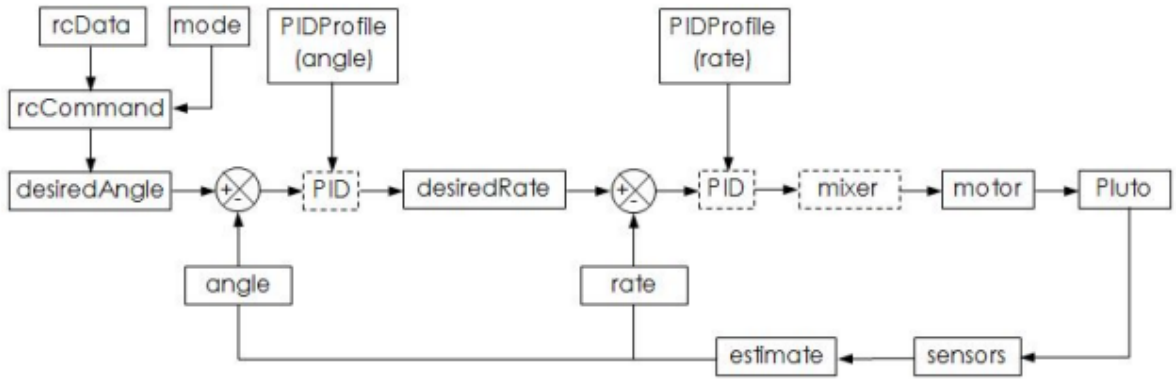


Figure 4.3
Control Block Diagram

In order to understand how Control Block works, let us take an example. Imagine a user wants to move PlutoX in the forward direction i.e. to pitch forward. When it is stable in the air before providing any pitch input, the Estimate Block provides the particular values of state variables at that instant. As soon as the user pitches forward, the rcData generates rcCommand considering the flight mode, and this rcCommand generated is forwarded to the specific part in the Control Block. Control Block generates the desired position at which PlutoX has to be taken to, the desired translational velocity at which PlutoX should move, the desired angle at which PlutoX should incline at and the desired rate at which PlutoX will change its angles. It should be noted that the 'desired' values are generated by considering the current values of the state variables and input from the user, i.e. essentially it generates the correction to the current error. So if PlutoX is pitching forward at 10° and the rcCommand is to tilt it to 30°, then the desired angle value would

be 20° and based on the desired angle, it will further generate the desired rate of change of angle to reach there. It can be understood by the following control loop diagram, which takes into account angle mode as the flight mode.

4.4 Project Example With Pluto 1.2

Objectives:

- Learning the usage of Cygnus and its APIs.
- What is debugging?
- How to use debug APIs like LED, print.

Problem statement: To detect the movement of PlutoX in vertical direction (upwards and downwards) by using LEDs on board and printing the velocity values on the Pluto Monitor.

Explanation: Debugging is the process of locating and removing the bugs in the software or hardware. In the case of PlutoX, tinkerers would program the ideas and flash them on the drone. In certain cases, the code might not work as expected or might show some errors. In such cases, debugging APIs can be used to identify the bug and solve them. The debugging APIs include LEDs, print and graph. LEDs can be used for visual confirmation of a task, print can be used to display the values on the screen of the computer and graph can be used for plotting the graph of values on the screen. Using these APIs, it becomes easier to understand the issue with the written program and to debug it. This project aims at understanding a basic application of debug APIs.

Approaching the problem:

- Creating a new project in the Cygnus IDE.
- Coding
- Building the Project
- Flashing the code
- Running the project

Project Takeaway:

This project helped in learning the use of LEDs for indicating the movement of PlutoX, monitoring the velocity values and plotting its graph as well. This can be used for debugging purposes in the future codes. For example, while programming the drone for performing a specific task such as rolling left and right based on certain input, program a specific LED for a particular movement. Suppose the red LED is programmed with rolling right. During flight, if this task does not function properly, check whether the red LED is ON or not, which will indicate whether the system of PlutoX is rolling right or not. Correspondingly, the values for the same can be obtained on the monitor and can also be plotted on a graph for understanding the working and to rectify any bugs in the code i.e. debug the code.

CHAPTER 5

Energy Model

5.1 Introduction

Given the large variety of drones, each with specific physical characteristics, like weight, type of power supply, propellers, etc., deriving a general parametric energy model that can be used to predict the energy consumption in different operating conditions is a hard task. In this work, we propose a method that can be used to model and analyse the energy consumption of a specific drone as a function of its speed and operating conditions.

5.2 Specifications

To derive an energy model suited for the analysis, we performed a set of experiments aimed at understanding how the energy consumption is affected by the different operating conditions, such as speed, horizontal and vertical accelerations. The drone used for the experiment is a Pluto 1.2 nano drone controlled by a Pluto controller and Magis software autopilot board.The Pluto 1.2 weighs about 80gm , and is equipped with four 850 Kv motors and powered with a 3S lipo battery 3.7v,600 mAh.

The maximum voltage your 3.7V lithium ion battery can have is 4.2V, also known as the maximum safe voltage/charge cut-off Voltage. When the battery is completely discharged, it will have a voltage of 3V, also known as the minimum safe voltage. Here's the deal:Most lithium-ion battery cells have a voltage of 3.7V because of their cell chemical properties, they are ternary lithium batteries. A single ternary lithium battery has 3.7V as its nominal voltage.The maximum speed of the drone is 3m/s and maximum range is 80m.

5.3 Mathematically proposed energy model

Hence for covering any area the Total energy consumed by UAV can be given by the following expression

$$E = E_{Take-off} + E_{lan} + N \times E_{90^o turn} + D \times E_{/m} - - - (i)$$

$E_{Take-off}$ = Average Energy consumed in Take-off

E_{lan} = Average Energy consumed in landing

N = Number of 90 degree turns

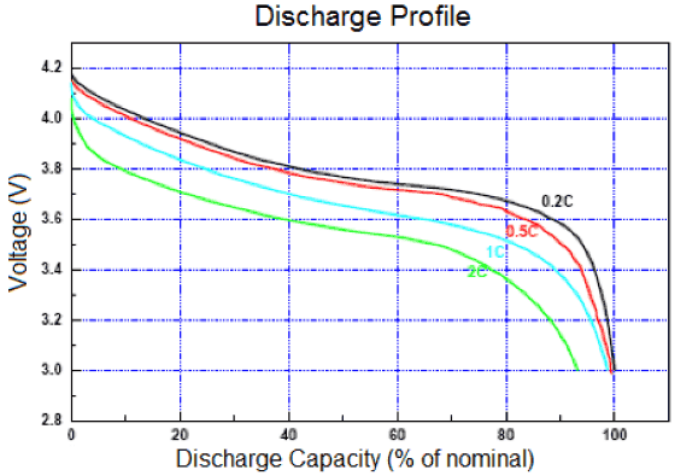
E_{90^o}turn =Energy consumed in 90 degree turns

D = Total distance travelled by UAV

$E(/m)$ =Average Energy consumed per meter

5.4 Li-Po batteryEnergy Calculation

The Lipo battery of 600mAh and 1C is used for flying the drone.So for the calculation of energy consumed for per 0.1 V drop the Lipo battery discharge curve is as following



Discharge: 3.0V cutoff at room temperature

Figure 5.1

The C-rate is the unit battery experts use to measure the speed at which a battery is fully charged or discharged. For example, charging at a C-rate of 1C means that the battery is charged from 0-100% in one hour. A C-rate higher than 1C means a faster charge; for example, a 3C rate is three times faster, so a full charge in 20 minutes. Likewise, a lower C-rate means a slower charge: C/5 (or 0.2C) would be five times slower than 1C, amounting to a five-hour charge.
Energy of a Li-Po battery at a given voltage can be written as following

$$\mathbf{E_{li-po}} = \mathbf{C} \times \mathbf{V} - - - \text{(ii)}$$

$\mathbf{C}$ = Energy Capacity

$\mathbf{V}$ =Volage of the battery

Hence by the eq(ii) the energy consumed for V2 to V1 drop will be

$$\mathbf{E_{v1-v2}} = \mathbf{C1V1} - \mathbf{C2V2} - - - \text{(iii)}$$

$\mathbf{E_{v1-v2}}$ =Energy consumed for the drop of voltage from V1 to V2

$\mathbf{C1}$ =Energy Capacity at V1 voltage

$\mathbf{C2}$ = Energy Capacity at V2 voltage

For 0.1V drop energy calculation
V1= 4V
V2= 3.9V

C1= 581.1 mAh
C2 = 528 mAh

$$E = C1V1 - C2V2$$
$$= ((581.1 \times 4) - (528 \times 3.9))\left(\frac{3,600}{1,000}\right)$$
$$= 954.7J$$

Similarly:

Voltage Drop	Energy Drop (kJ)
4v to 3.9v	0.954
3.9v to 3.8v	1.257
3.8v to 3.7v	1.3608

Table 5.1

5.5 Experiments

5.5.1 Takeoff and landing average energy:-

In the first experiment energy consumed for takeoff and landing was calculated. For that first the drown was flown In a first experiment, we programmed the drone to take off and to land subsequently in altitude hold mode. Repeated the activity until the battery voltage is dropped by 0.1V . Continued the experiment for further 0.1drop and so on.The energy was calculated by taking the data of voltage drop v/s no. of take offs and landings.

Voltage Drop	Number of Flights	Time Taken	Total flight time	Average Time Taken
4.0v to 3.9v	8	4.22, 4.8, 4.62, 4.52, 5.07, 4.62, 4.62, 5.12, 4,49	36.96	4.62
3.9v to 3.8v	13	4.58, 4.84, 4.15, 4.29, 4,51, 4.40, 3.12, 4.51, 3.51, 3.60, 2.60, 3.19, 3.50	50.81	3.908
3.8v to 3.7v	16	5.10, 4.68, 3.13, 4.94, 3.55, 6.56, 4.00, 5.41, 6.08, 4.08, 2.96, 4.15, 4,.87, 3.80, 4.01, 5.57	72.88	4.555

Table 5.2

Exp No.	Voltage Drop	Energy Drop	No. of Flights	Average energy per

		(kJ)		flight
I	4v to 3.9v	0.954	8	0.11925
II	3.9v to 3.8v	1.257	13	0.09669230769
III	3.8v to 3.7v	1.3608	16	0.08505
IV	4 to 3.9v	0.954	10	0.0954
Total		4.5258	47	0.09629361702

Table 5.3

Overall Energy for Take-off and land = 0.09629361702 kJ = 96.29361702 J

5.5.2 Yaw turn Energy calculation:-

In the second experiment energy consumed for Yaw turn was calculated. We programmed the drone to take off , to take multiple turns afterwards landing. Imitated the activity until the battery voltage dropped by 0.1V . Continued the experiment for further 0.1drop and so on.The energy was calculated by taking the data of voltage drop v/s no. of take offs, numbers of turns and landings.

Exp No.	Voltage Drop	Energy Drop (kJ)	No. of Flights	No. of yaw turns	Energy only in turns (kJ)	Energy per turn (kJ)
I	4v to 3.9v	0.954	5	25	0.35775	0.01431
II	3.9v to 3.8v	1.257	11	39	0.1933846154	0.004958579882
III	3.8v to 3.7v	1.3608	13	39	0.25515	0.006542307692
IV	3.9v to 3.8v	1.257	10	38	0.2900769231	0.007633603239
Total		4.8288	39	141	1.073348936	0.007612403803

Table 5.4

Overall Energy per turn = 0.007612403803 kJ = 7.612403803 J

4.5.3 Straight line Energy Calculation:-

For Calculation of straight line path energy calculation we made the drone take off then as soon as it was stable in the air we made it go forward and backward than land. We repeated this for the voltage drop of 0.1v and so on for the other drops.

Exp No.	Voltage Drop	Energy Drop (kJ)	No. of Flights	Total distance travelled	Energy only in turns	Energy per metre (kJ)
I	4v to 3.9v	0.954	4	200	0.477	0.002385
II	3.9v to 3.8v	1.257	5	215	0.7735384615	0.003597853309
III	3.8v to 3.7v	1.3608	7	263	0.76545	0.002910456274
IV	3.8v to 3.7v	1.3608	6	230	0.8505	0.003697826087
	Total	4.9326	22	908	2.814140426	0.003099273596

Table 5.5

Overall Energy for per metre = 0.003099273596 kJ = 3.099273596 J

CHAPTER 6

UAV automation Apis and Code

6.1 Introduction:

The User code execution starts when the drone is in developer mode. The developer mode can be controlled using the Pluto Controller app using the button,
There are four basic functions:
i. plutoInit()

This function is executed immediately after the drone powers up. This function is particularly useful for initialising hardware.
ii. onLoopStart()

This function is executed once, after the user turns the developer mode on. This function can be useful for initialising some variables
iii. plutoLoop()

This function is executed in a loop, along with the drones internal primary stabilisation code. By default this loop runs every 3.5ms. Using APIs, you can modify this loop frequency. This will not affect the drones internal stabilisation loop.
iv. onLoopStop()

This function is executed once, after the user turns off the developer mode. While using the APIs it's important to note the frame of reference. In the following figure we briefly explain some basic angles and axis of the drone.

6.2 Basic Terminology

When you are trying to control a drone, there are many variables to choose from. These variables in state-space theory are referred to as state variables. A State variable can be defined as the smallest set of variables that fully describe the system(drone) and its response to any given set of inputs. In case the system is a rigid body, there are 12 state variables that are used. These state-variables are: Position: X, Y, Z These three variables represent position of the drone with respect to the Inertial frame.

Velocity: Vx, Vy, Vz These three variables represent velocity of the drone with respect to Inertial frame.

Rotation rate: p, q, r These three variables represent the rotational rates of the drone with respect to body-axis.

Angles: Roll(φ), Pitch(θ), Yaw(ψ) There three variables represent the orientation of the drone with respect to inertial frame. These three angles are non-commutative. Thus the sequence of rotation matters. This representation is called 3-2-1 Euler representation. As seen in the figure below:
Roll represents rotation about X-axis, Pitch represents rotation about Y-axis, Yaw represents rotation about Z-axis.

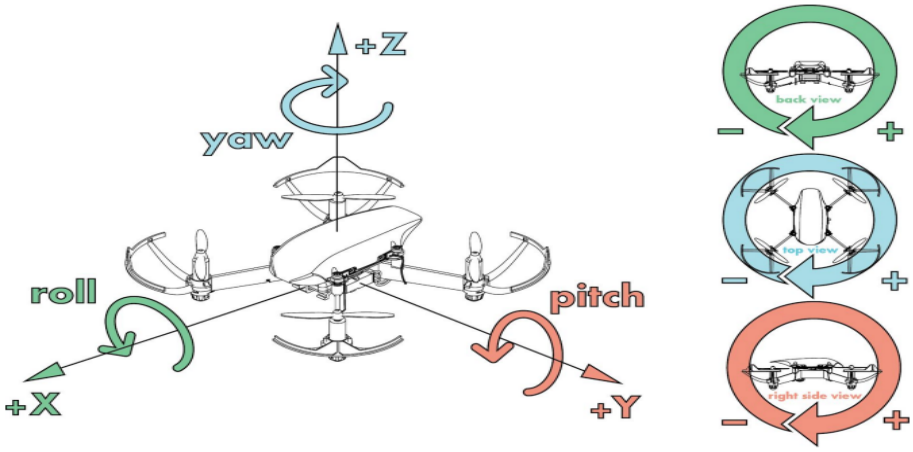


Figure 6.1
Representation of Yaw, Pitch and Roll

6.3 UAV automation code using Magis software

Magis continuously runs inside the hardware of Pluto 1.2. Even while programming Pluto 1.2 for tinkering, the core software which results in the seamless flight still continues to run. Thus, one does not need to program the drone again or does not need to flash Magis again after testing. The entire tinkering program will run simultaneously along with Magis.

While programming an idea,It is not required to make changes in Magis. A separate loop has been provided for the program our idea. This loop is called just before the Control loop. The default frequency of calling is 3.5ms, which can be changed if needed. The entire program can be coded in this loop by the tinkerer and when the loop is called, the code will run. Let us understand the loop in a better way.

Turning the drone on Yaw-axis:

```
// Do not remove the include below
#include "PlutoPilot.h"
#include "Control.h"
#include "Sensor.h"
#include "User.h"
#include "Utils.h"
#include "Math.h"
#include "Estimate.h"

//The setup function is called once at Pluto's hardware startup
int32_t YAW_angle, height, angle, Pitch_angle;
```

```
PID pitchanglePID; //This is a structure of pid variable.
uint8_t P, I, D;
Interval T5;
int time;
bool flag, t;
void plutoInit()
{
//Add your hardware initialization code here
}
//The function is called once before plutoLoop() when you activate developer mode
void onLoopStart()
{
//Do your one time stuff here
	LED.flightStatus(DEACTIVATE);/*Disable the default led behaviour*/
	T5.reset();
	time = 0;
	angle = 0;
	height = 0;
	flag = false;
}
//The loop function is called in an endless loop
void plutoLoop()
{
//Add your repeated code here
	t = T5.set(1000,true);
	int x = Acceleration.getNetAcc();
	if(x<2 && (!FlightStatus.check(FS_CRASHED))) /*Condition for free fall*/
	{
		Command.arm(); /*Arm the drone*/
	}
	height = Position.get(Z);
	YAW_angle = Angle.get(AG_YAW);
	Pitch_angle = Angle.get(AG_PITCH);
	if(height > 290 and !flag) {
		flag = true;
		time = 0;
	}
	if(t){
		time++;
	}
```

```
        if(flag and time >= 1) {
                angle = 90;
        }
        DesiredAngle.set(AG_YAW, angle);
        DesiredPosition.set(Z, 300);
        Monitor.println("Current Height:", height);
        if(flag) Monitor.println("Time:", time);
        Monitor.println("Current YAW angle:", YAW_angle);
        Monitor.println("Current PITCH angle:", Pitch_angle);
}
//The function is called once after plutoLoop() when you deactivate developer mode
void onLoopFinish()
{
    //Do your cleanup stuff here
}
```

Pitching the drone forward :

```
// Do not remove the include below
#include "PlutoPilot.h"
#include "Control.h"
#include "Sensor.h"
#include "User.h"
#include "Utils.h"
#include "Math.h"
#include "Estimate.h"


int32_t YAW_angle, height, angle, Pitch_angle;
PID pitchanglePID;  //This is a structure of pid variable.
uint8_t P, I, D;
Interval T5;
int time;
bool flag, t, prt;

//The setup function is called once at Pluto's hardware startup
void plutoInit()
{
//Add your hardware initialization code here
}
```

```
//The function is called once before plutoLoop() when you activate developer mode
void onLoopStart()
{
//Do your one time stuff here
	height = 0;
	flag = false;
	T5.reset();
	t = true;
	prt = false;
	time = 0;
	angle = 0;
}
//The loop function is called in an endless loop
void plutoLoop()
{
//Add your repeated code here
	t = T5.set(1000,true);
	int x = Acceleration.getNetAcc();
	if(x<2 && (!FlightStatus.check(FS_CRASHED))) /*Condition for free fall*/
	{
		Command.arm(); /*Arm the drone*/
	}
	height = Position.get(Z);

	Pitch_angle = Angle.get(AG_PITCH);
	DesiredPosition.set(Z, 300);
	DesiredAngle.set(AG_PITCH, angle);
	if(height > 190 and !flag) {
		flag = true;
		time = 0;
	}
	if(flag){
		time++;
	}

	if(flag and time >= 2 and t) {
		angle = 200;
		t = false;
	}
```

```
        PIDProfile.get(PID_PITCH, &pitchanglePID);
        //You can access the individual gains as follows
        P = pitchanglePID.p;
        I = pitchanglePID.i;
        D = pitchanglePID.d;
        prt = true;
        if(prt){
                Monitor.println("height:", height);
                Monitor.println("pitch angle:", Pitch_angle);
                Monitor.println("P:", P);
                Monitor.println("I:", I);
                Monitor.println("D:", D);
                Monitor.println("time:", time);
        }
}
//The function is called once after plutoLoop() when you deactivate developer mode
void onLoopFinish()
{
//Do your cleanup stuff here
}
```

Flipping the drone Backward:

```
 // Do not remove the include below
#include "PlutoPilot.h"
#include "Control.h"
#include "Sensor.h"
#include "User.h"
#include "Utils.h"
#include "Math.h"
#include "Estimate.h"

int32_t YAW_angle, height, angle, Pitch_angle;
PID pitchanglePID; //This is a structure of pid variable.
uint8_t P, I, D;
Interval T5;
int time;
bool flag, t, prt;

//The setup function is called once at Pluto's hardware startup
```

```
void plutoInit()
{
//Add your hardware initialization code here
}
//The function is called once before plutoLoop() when you activate developer mode
void onLoopStart()
{
//Do your one time stuff here
	height = 0;
	flag = false;
	T5.reset();
	t = true;
	prt = false;
	time = 0;
}

//The loop function is called in an endless loop
void plutoLoop()
{
//Add your repeated code here
	t = T5.set(1000,true);
	int x = Acceleration.getNetAcc();
	if(x<2 && (!FlightStatus.check(FS_CRASHED))) /*Condition for free fall*/
	{
		Command.arm(); /*Arm the drone*/
	}
	height = Position.get(Z);
	Pitch_angle = Angle.get(AG_PITCH);
	DesiredPosition.set(Z, 300);
	if(height > 290 and !flag) {
		flag = true;
		time = 0;
	}
	if(flag){
		time++;
	}

	if(flag and time >= 2 and t) {
		Command.flip(BACK_FLIP);
		t = false;
```

```
        }

        PIDProfile.get(PID_ROLL, &pitchanglePID);
        //You can access the individual gains as follows
        P = pitchanglePID.p;
        I = pitchanglePID.i;
        D = pitchanglePID.d;
        prt = true;
        if(prt){
                Monitor.println("height:", height);
                Monitor.println("pitch angle:", Pitch_angle);
                Monitor.println("P:", P);
                Monitor.println("I:", I);
                Monitor.println("D:", D);
                Monitor.println("time:", time);
        }
}
//The function is called once after plutoLoop() when you deactivate developer mode
void onLoopFinish()
{
//Do your cleanup stuff here
}
```

Hence from the above examples we can conclude that we can control the drone by programming instead of controlling it manually.

CHAPTER 7

Simulation Validation

7.1 Experimental Values

For Energy consumption calculation we are taking an area of 150mx100m in that first it is flown parallel to the length and energy calculated by the proposed Energy Model.

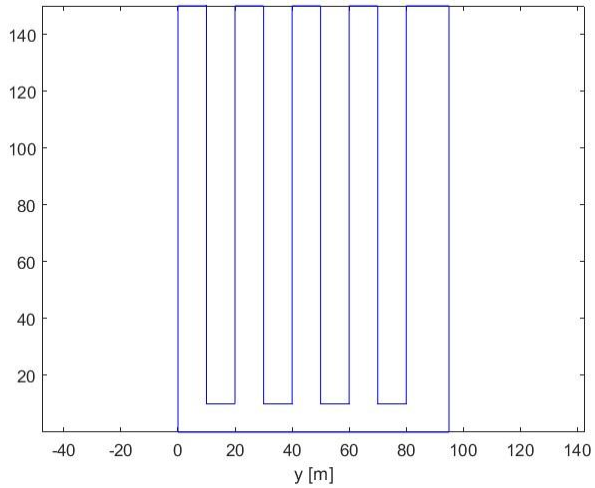


Fig. 7.1: Path along the length

By the energy Model

$$E_T = E_{take-off} + E_{lan} + N \times E_{90^o turn} + D \times E_{/m}$$

Here

$$E_{take-off} + E_{lan} = 96.29J$$

$$E_{90^o turn} = 7.612J$$

$$E_{/m} = 3.099J$$

$$D = 1{,}610m$$

$$N = 20$$

$$\begin{aligned} E_(T) &= E_(take-off) + E_(lan) + N * E_(90^o\ turn) + D * E_(/m) \\ &= 96.29 + (20 \times 7.612) + (1{,}610 \times 3.099) \\ &= 5{,}237.920 \end{aligned}$$

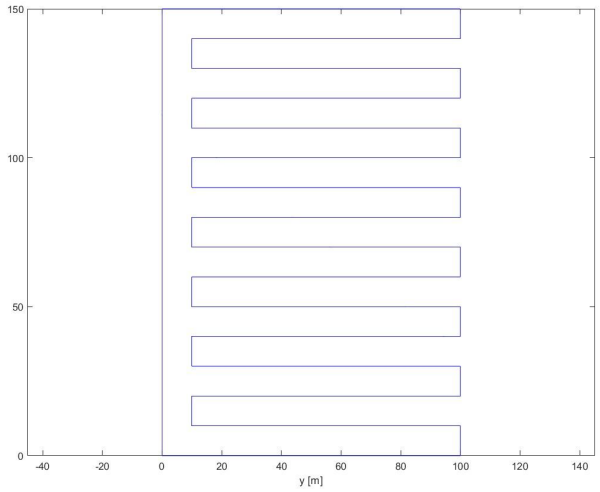


Fig. 7.2 : Path along the width

For Energy consumption calculation we are taking an area of 150mx100m . Now it is flown parallel to width and energy calculated by the proposed Energy Model.

$$E_T = E_{take-off} + E_{lan} + N \times E_{90^o turn} + D \times E_{/m}$$

Here

$$E_{take-off} + E_{lan} = 96.29J$$

$$E_{90^o turn} = 7.612J$$

$$E_{/m} = 3.099J$$

$$D = 1{,}760m$$

$$N = 32$$

$$\begin{aligned} E_(T) &= E_(take-off) + E_(lan) + N * E_(90^o\ turn) + D * E_(/m) \\ &= 96.29 + (32 \times 7.612) + (1{,}760 \times 3.099) \\ &= 5{,}794.114J \end{aligned}$$

7.2 Simulation values

For Energy consumption calculation we are taking an area of 150mx100m in that first it is flown parallel to the length and once parallel to width, then energy calculated by the matlab code.

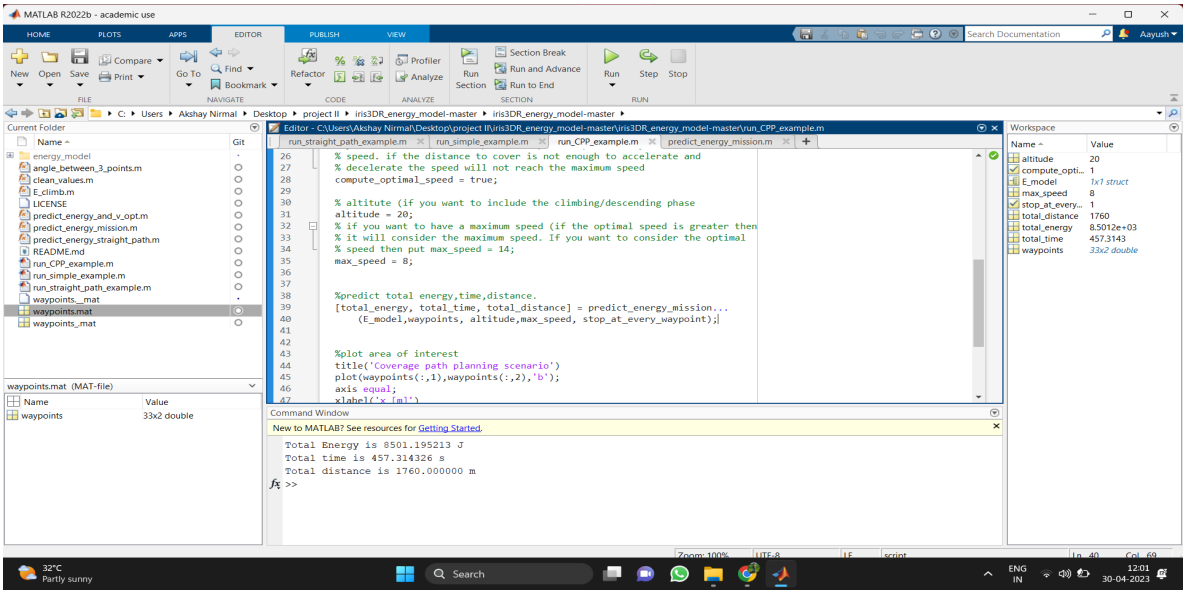


Fig. 7.3: Simulation results along the width

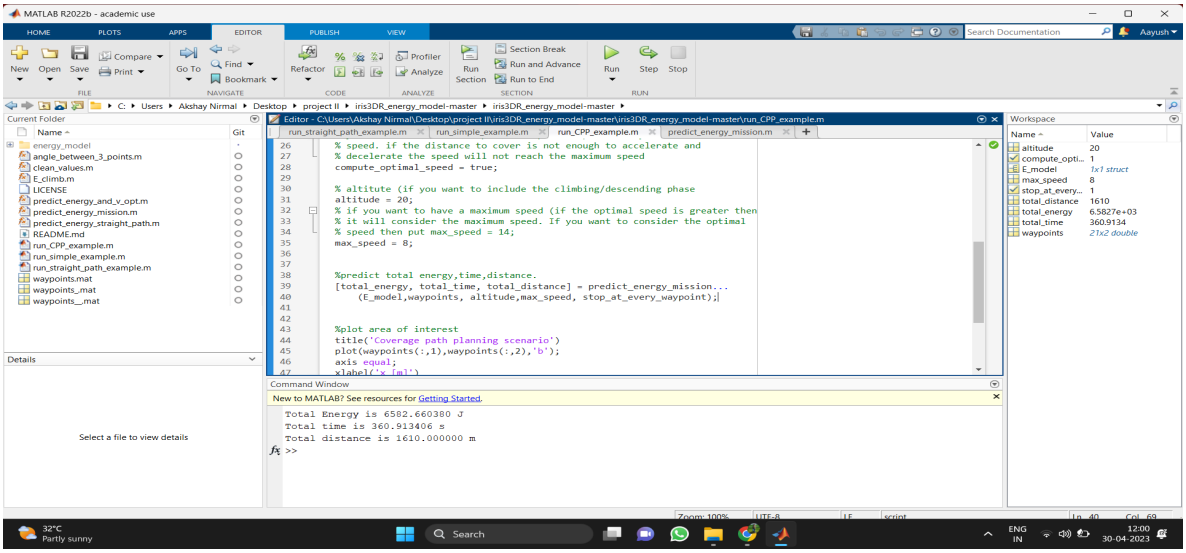


Fig. 7.4: Simulation results along the length

7.3 Validation:

To validate the energy model, we have taken a rectangular area of dimension 150mx100m and simulated the drone Pluto 1.2 over the area for the surveying purpose. When the number of turns are minimised then the total energy consumed also decreases. For flying it parallel to the length, energy consumed is **6586.660 J** and the number of turns are **20** . But while flying it parallel to the width of energy consumed is **8501.195J** and the number of turns are **32.** While flying parallel to width with less number of turns the energy consumed is **22.57 %** less than flying parallel to length with more number of turns.

CHAPTER 8

Future Scope

The future scope of UAV path planning is vast and involves the development of advanced algorithms that can optimise the path planning process. Some of the areas that hold great promise for future in UAV path planning include:

1. Autonomous path planning: Researchers may develop algorithms that enable UAVs to plan their own paths autonomously, without any human intervention. Such algorithms may incorporate machine learning techniques that allow the UAV to learn from past experiences and optimise its path planning based on environmental conditions.
2. Multi-UAV path planning: As the number of UAVs in use increase, there will be a need to develop algorithms that can coordinate the movements of multiple UAVs. This will require the development of sophisticated algorithms that can factor in the position and movement of other UAVs in the airspace.
3. Dynamic path planning: Dynamic path planning involves generation paths that can adapt to changing environmental conditions. This could include accounting for wind and weather patterns, as well as obstacles that may appear unexpectedly.
4. Collaborative path planning: Collaborative path planning involves developing algorithms that can coordinate the movement of UAVs with other vehicles or systems, such as ground-based robots or satellites. This requires sophisticated algorithms that can factor in the movements and objectives of other vehicles or systems.

Overall, the future scope of UAV path planning is broad and offers significant potential for innovation and development. As the use of UAVs continues to increase across various industries, the need for efficient and effective path planning algorithms will only grow, making this a critical area of research for the future.

CONCLUSION

In this project we have discussed and Experimented a broad idea of doing Energy Efficient Path Planning for Aerial Photography Based Disaster Response .We have so far given a drone path planning model that takes into account the overall amount of energy used to complete a task. The drone's acceleration, deceleration, hovering, and turning are the foundation of the energy model. From the data gathered from UAV surveying, we can use the data for multiple objectives using CNN and deep learning algorithms as mentioned in the report.

Here, we used Pluto 1.2 drone for the surveying purpose and calculated the energies for possible movements such as yaw angle and energy taken for the per metre distance. Later the model was verified with the simulation results. Which gave us the expected results that keeping the number of turns minimum helps in reducing the energy consumption for the same area.

REFERENCES

1. Sindhuja Sankaran. A Path Planning Method with Perception Based on Sky Scanning for UAVs. Article on mdpi. 24 January 2022.
2. Lucas Prado Osko, Jose Marketo Junior. A Review on Deep Learning in UAV Remote Sensing. ScienceDirect. October 2021.
3. Carmelo di Franco, Giorgio Buttezzo. Energy Aware Coverage Path Planning of UAVs. IEEE. 2015.
4. Aashish Mishra, Calculating the Energy, Run time and Charge of a LiPo battery, Circuit Digest, December 2021.
5. Adafruit, LiPo Battery, Discharge Curve and voltages.
6. Pluto Drone APIs Documentation.
7. Projects with Pluto Drone using Cygnus IDE.