A machine learning approach for UAV-based ground target geolocation in aerial images.

Project-I (EC4D001) Report submitted in partial fulfillment for the award of degree of

Bachelor of Technology

in

Electronics and Communication Engineering

by

Anurag Paul
20EC01045

Under the supervision of

Dr. Niladri Bihari Puhan

SCHOOL OF ELECTRICAL SCIENCES

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

Acknowledgement

I express my deep sense of gratitude and sincere regards to my supervisor Dr. Niladri Bihari Puhan. The friendly discussion helped immensely in selecting this topic and the generous encouragement throughout my dissertation work helped in continuing this project work. Sir has immensely assisted in providing all opportunities and facilities for the project work. His critical reviews aided in implementing the topic. I am thankful to sir for helping in this work.

I am indebted to my parents who have inspired us to face all the challenges and win all the hurdles in life. Finally, I would like to thank all those who directly or indirectly guided me during my work since inception.

Abstract

This report proposes geolocating a ground target with an unmanned aerial vehicle (UAV) in a simulation of urban environment. UAVs such as quadcopter and multirotor aircrafts have been widely used in recent years. Examples include information and intelligent warfare. For geolocating multiple targets of interest on the ground from an aerial platform, object detection with the trained network will be applied to the target pixel locations from a drone footage.

Within academia and industry, there has been a need for expansive simulation frameworks that include model-based simulation of sensors, mobile vehicles, and the environment around them. This report makes use of a quadcopter aircrafts simulation environment based on open-source AirSim framework. It is a popular community-built system that fulfills some of those needs. However, the framework required adding systems to serve some complex industrial applications, including designing and testing new sensor modalities, Simultaneous Localization And Mapping (SLAM), autonomous navigation algorithms, and transfer learning with machine learning models as required by a user.

This article describes the framework of this work and the communication way between different parts. Furthermore, we show the various applications and use cases the framework can serve.

Contents

List of Figures

Chapter 1

Introduction

With the continuous progress of technology, quadcopter aircrafts are widely used in security, rescue, plant protection, transportation, and other fields. Currently, there are two main test methods for quadcopter. The first way is to use a real quadcopter for the test. The other way is to use the current simulation environment for the test. Firstly, using an actual quadcopter for the test is a direct way to verify the aircraft. Undoubtedly, the effect is the most real. However, actual flights are inconvenient for the quadcopter test.

The main problems are as follows. One is the safety issue, an unverified quadcopter is directly used for the test can easily injure the people. Secondly, untested aircrafts are extremely easy to crash, resulting in great economic losses. The third is an efficiency problem. A real flight needs to consider many issues, such as assembly, maintenance, battery and so on.

These factors will greatly affect flight efficiency and extremely extend the development cycle. Therefore, many researchers use simulation environments for the quadcopter test. At present, many simulation platforms can be used for tests such as Gazebo [1] [figure 1.1], jMavSim [2] [figure 1.2] and so on. Among them, Gazebo is a feature-rich simulation platform, which is widely used in the robots, cars, and drone simulation.

However, some limitations, like inconvenient construction, are not real enough. Which is extremely important for the quadcopter simulation tests. Usually, the real environment is extremely complex and has a wide variety of objects. If the simulation environment is inconducive for building large complex scenes, it will greatly extend the development time. Next, if the simulation environment scenes are not real enough, it will have much negative effect on the experiment, especially in some visual tests.

Therefore, aiming at these shortcomings above, this thesis proposes a new simulation environment based on AirSim [3]. AirSim provides us with a more realistic simulation environment and rich data interface as shown in figure 1.3. This enhances the authenticity of the simulation. Compared to figure 1.1 and figure 1.2, AirSim's images are more realistic. Unreal Engine or many GitHub repositories have some ready-made scenes in application store that contain blocks, forest, snow mountains, and so on.

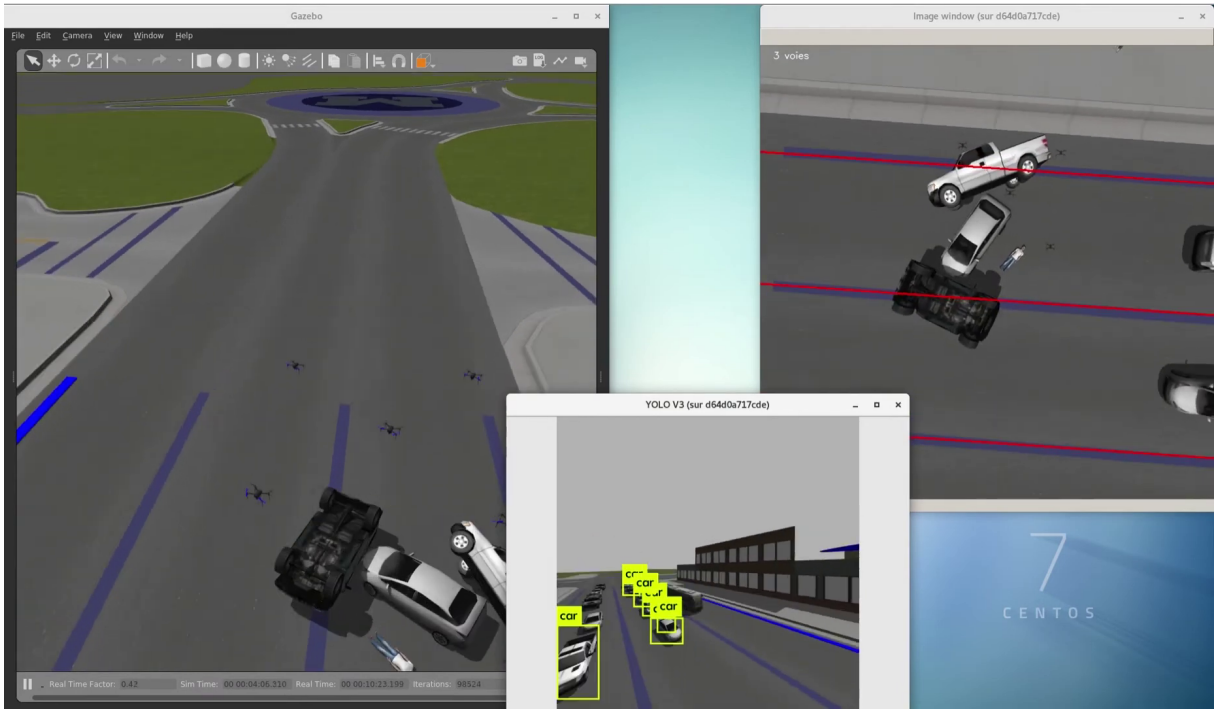

Figure 1.1: Gazebo drone fleet simulation.

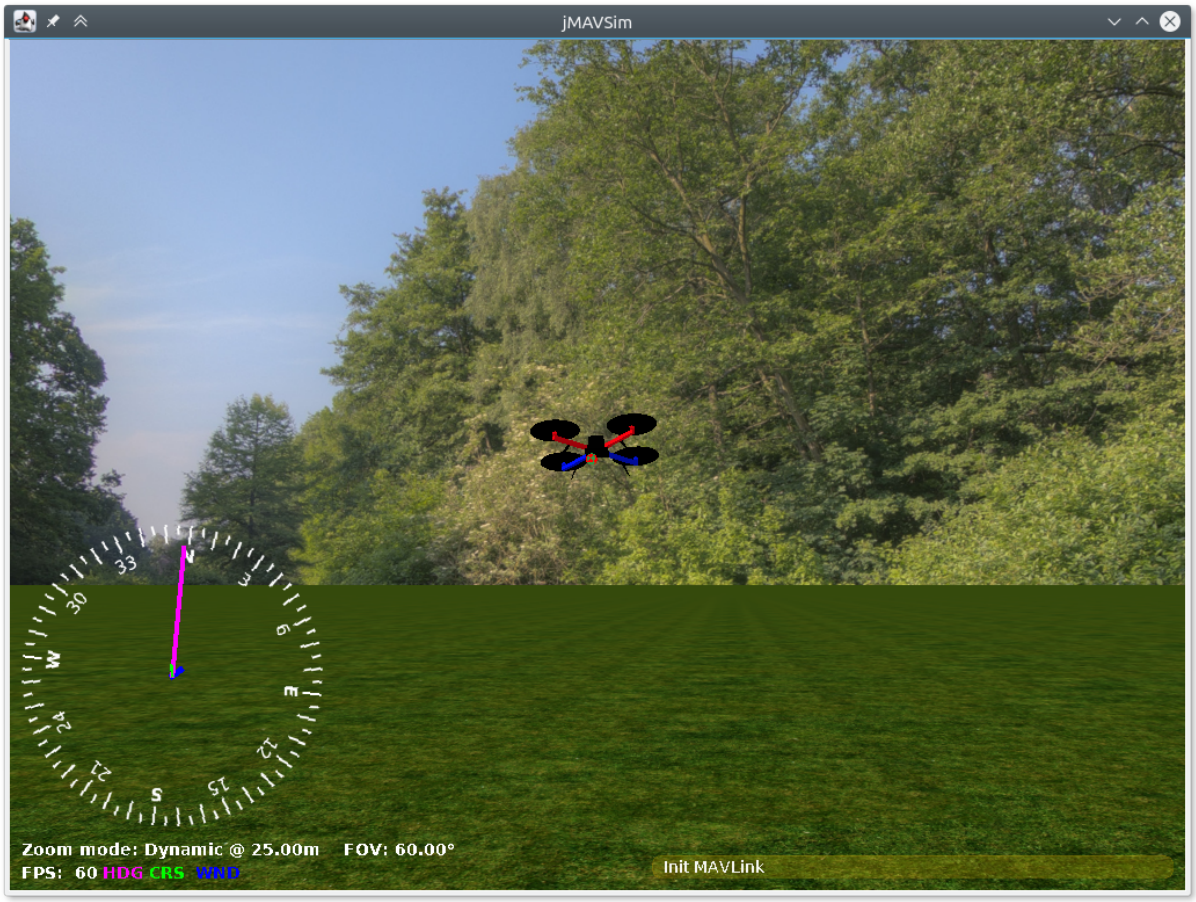

Figure 1.2: jMavSim, a simple multirotor simulator with MAVLink protocol support.

Figure 1.3: AirSimNH environment.

Chapter 2

Applications

Modern-day industrial applications, however, often require some form of flexibility in simulation. These applications include deciding ideal-sensor placement, validating sensor design parameters, testing autonomous robotic algorithms, generating training datasets for neural networks, and much more. Therefore, having a single, modular, and open-source framework allows for tackling the many requirements of such applications across several academic or industrial projects and takes less time to develop.

Commercial frameworks seem to provide the necessary tools to make these applications work. Most often, these commercial frameworks are designed for ADAS (Advanced Driver-Assistance System)/AD (Autonomous Driving) simulation that can do software-in-the-loop and even hardware-in-the-loop simulation of vehicles within detailed environments. However, to be more accessible to the general public, smaller companies, and academic researchers, opensource frameworks are desired, which can also be used for different application types than ADAS/AD.

Quadcopter can be used to set all kinds of sensors, and has low cost and high flexibility, thus can be applied to a variety of different tasks, package across target tracking, disaster rescue, crop monitoring, etc. Quadcopter can quickly spread, a big reason is that the development of the open source flight control, in the complicated products, four rotor quadcopter with its advantage of simple structure, convenient use and low cost, first came to the attention of the public.

Unmanned aerial vehicle learning in a virtual environment can greatly reduce the loss of the body and increase the speed of training. The virtual unmanned aerial vehicle needs to be transplanted to a real unmanned aerial vehicle after the virtual environment training is completed.

Chapter 3

Methodology

3.1 Simulation Framework Overview

The AirSim framework was designed for AI research and experimentation. While it focused on unmanned aerial vehicles, it was designed to be modular to accommodate new types of mobile platforms, sensors, and environments. In July 2022, Microsoft announced the end of its support to the original AirSim research project: it would evolve into a commercial closed-source version. This type of event further highlights the need for long-term supported, community-driven simulation frameworks.

The interaction between the AirSim Python library and the underlying Unreal Engine was used. AirSim allows recording of trajectories for vehicles; because the simulation is running in real-time, the data capturing through the API has to run in real-time, which was achieved. This enables the generation of datasets at a much faster speed than in real life because multiple simulations can run at the same time.

3.2 Virtual Environments

AirSim already provides basic weather simulation; hence the performance degradation of a sensor such as camera in rainy conditions regarding the intensity and range measurements can be tested. This influences this last form of quality attenuation. The proposed method is divided in to virtual and real environments. First we train the unmanned aerial vehicle to complete a specific task in a virtual environment. We can also use these sensors to get information about GPS and distance from the ground. Actions: There are six of them. The X-axis, Y-axis and z-axis and their positive and negative movement.

3.3 Realistic Environment

In order to transfer the model training in virtual environment to the real environment. We may use a programmable or manual unmanned aerial vehicle that can operate the

UAV through commands. The UAV connect with server. Because the virtual world cannot fully simulate the real world. Unmanned aerial vehicle will do a small amount of training in the real world so that the model can deal with the tasks in reality more effectively.

```python
import airsim
import sys
import time
from Drone_Auto import image_proc_algorithm
import keyboard  # using module keyboard

#connect and take off drone
client = airsim.MultirotorClient()# for car use CarClient()
client.confirmConnection()
client.enableApiControl(True)
client.armDisarm(True)
client.takeoffAsync()

# z of -20 is 20 meters above the original launch point.
z = -20

# Fly given velocity vector for 1 seconds
duration = .1
speed = 20
```

Figure 3.1: Maneuver Python snippet [4].

Table 3.1: Acumulated Data

Altitude	Longitude	Latitude	PixelX	PixelY	YLatitude
123.7985229	-122.140125	47.6419267	256	144	47.6419267
174.5480347	-122.140165	47.6419267	257	167	47.6419267
174.8162537	-122.140165	47.6419267	257	168	47.6419267
176.4992676	-122.140165	47.6419267	257	169	47.6419267
177.9062653	-122.140165	47.6419267	257	169	47.6419267
178.6100311	-122.140165	47.6419267	257	171	47.6419267
180.3651276	-122.140165	47.6419267	257	171	47.6419267
181.2619781	-122.140165	47.6419267	257	171	47.6419267
182.8026276	-122.140165	47.6419267	257	173	47.6419267
184.0067596	-122.140165	47.6419267	257	173	47.6419267
185.7004089	-122.140165	47.6419267	257	174	47.6419267
186.8738556	-122.140165	47.6419267	257	176	47.6419267
187.9985809	-122.140165	47.6419267	257	176	47.6419267
189.4416962	-122.140165	47.6419267	257	177	47.6419267
190.2579041	-122.140165	47.6419267	257	177	47.6419267
191.8865051	-122.140165	47.6419267	257	177	47.6419267
193.0978851	-122.140165	47.6419267	257	178	47.6419267

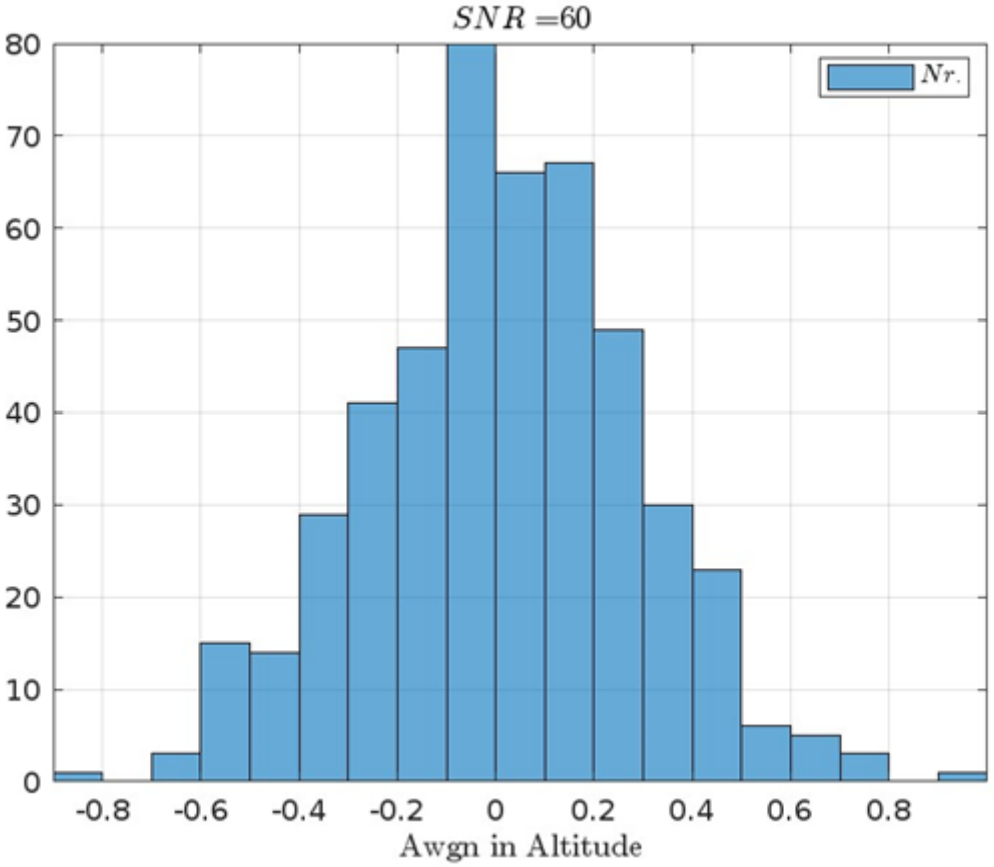

Figure 3.3: Δ Altitude

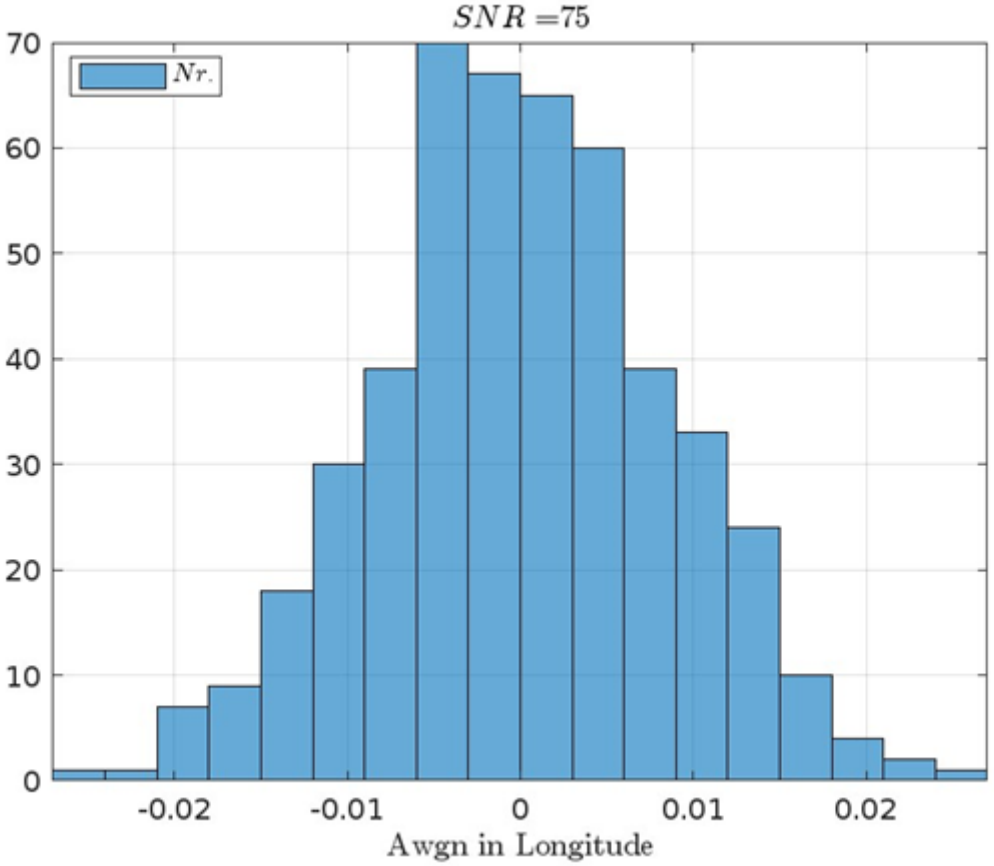

Figure 3.5: Δ Longitude

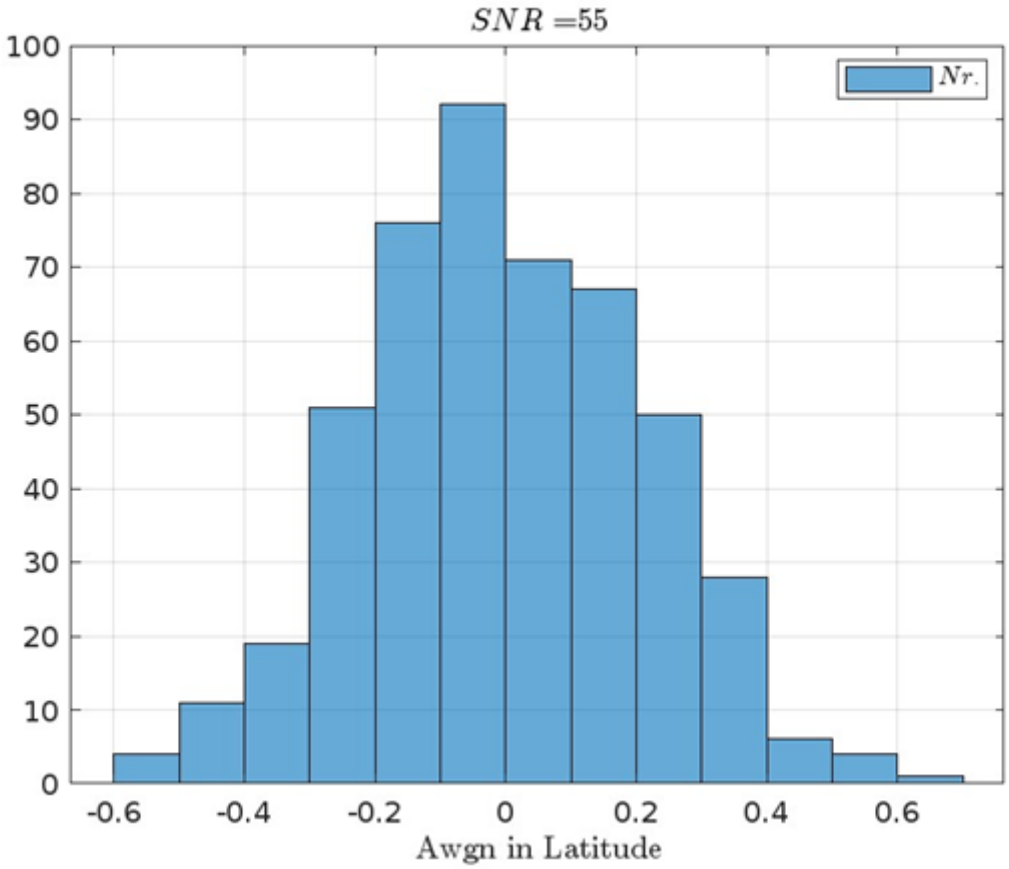

Figure 3.4: Δ Latitude

Haversine distance

$$D(x,y) = 2\arcsin[\sqrt{\sin^2((x_{lat} - y_{lat})/2) + \cos(x_{lat})\cos(y_{lat})\ sin^2((x_{lon} - y_{lon})/2)}]$$

Chapter 4

Results

Table 4.1: Distance Data

Sr. Nr.	Drone		Target		Distance(m)
	Latitude	Longitude	Latitude	Longitude	
1	47.64192669	-122.140165	47.64192669	-122.1401251	1.91 nm
2	47.63749272	-122.140165	47.64185602	-122.1384163	3.88 nm
3	47.63749273	-122.140165	47.64116615	-122.1384163	9.94 nm
4	47.63749272	-122.140165	47.641468	-122.140165	7.07 Å
5	47.63749253	-122.1401649	47.63749253	-122.1401649	1.7 mm

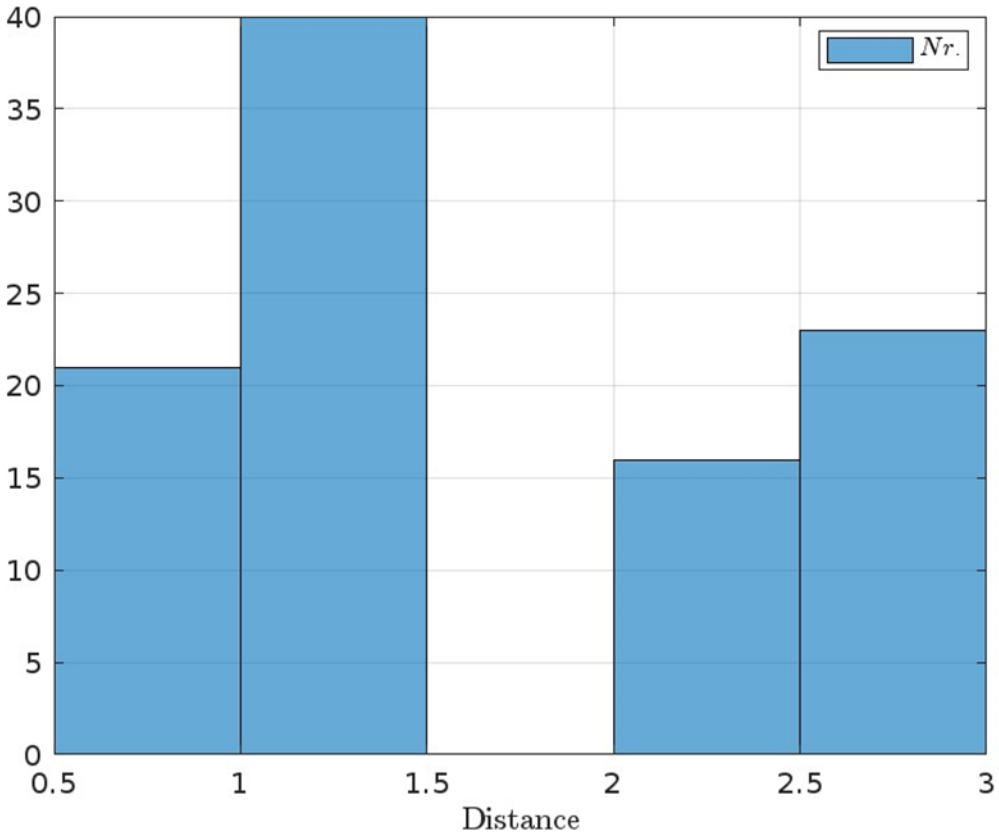

Figure 4.2: Δ distance.

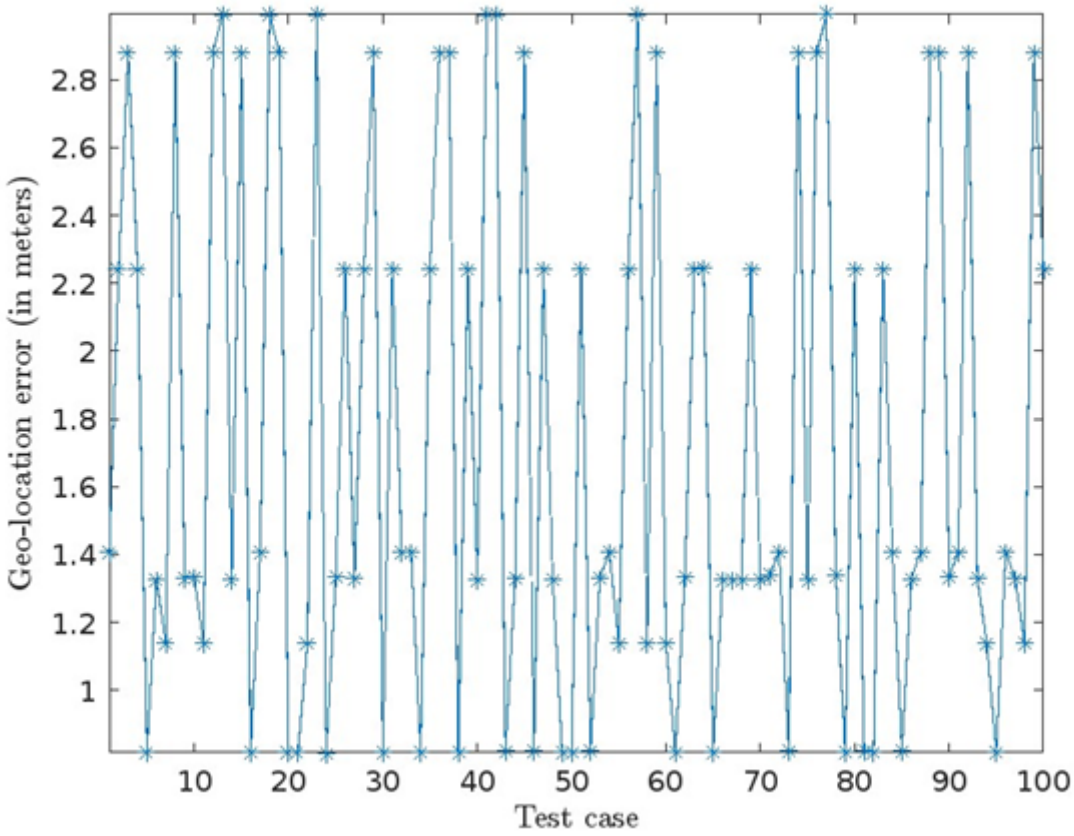

Figure 4.3: Geolocation-error (in meters).

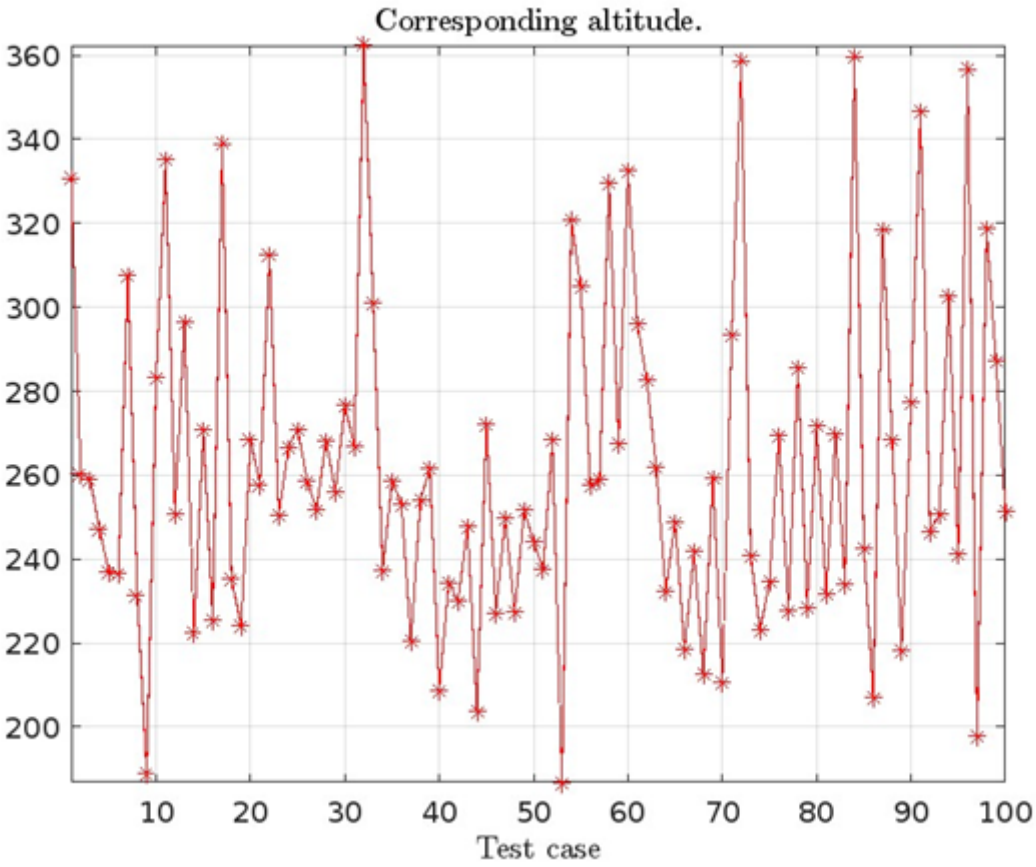

Figure 4.4: Altitudes

Future Work

Some more sets of images will be collected and correspondingly the n target locations will be acquired for training and testing. The images can be passed through a convolutional neural networks for detection of object detection such as litter on beaches or for coordinate location of objects. Moving vehicle detecting, tracking, and geolocating based on a monocular camera, a GPS receiver, and inertial measurement units (IMUs) sensors will be used. Vehicle detection and efficiency for small object detection in complex scenes would be investigated. COSYS-AIRSIM [5] may be used for more images.

Then, a visual tracking method based on filters, and a geolocation method to calculate the GPS coordinates of the moving vehicle may be implemented. Flight control method in terms of the previous image processing results would be introduced to lead the UAV that is following the moving vehicle. The framework should automatically supervise on target vehicles in real-world experiments, which would suggests its potential applications in urban traffic, logistics, and security.

References

[1] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[2] *jMAVSim*. URL: https://github.com/DrTon/jMAVSim.

[3] S. Shah, D. Dey, C. Lovett and A. Kapoor. "AirSim: high-fidelity visual and physical simulation for autonomous vehicles". In: *IEEE Transactions on Geoscience and Remote Sensing* (2017), p. 14.

[4] *Anurag Paul*. URL: https://github.com/AnuragPaul0/AirSim.

[5] Wouter Jansen et al. "COSYS-AIRSIM: A Real-Time Simulation Framework Expanded for Complex Industrial Applications". In: *2023 Annual Modeling and Simulation Conference (ANNSIM)*. 2023, pp. 37–48.