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

Anurag Paul
B.Tech in ECE
SES, IIT Bhubaneswar
Bhubaneswar, India
20ec01045@iitbbs.ac.in

Dr. Niladri Bihari Puhan
Supervisor
SES, IIT Bhubaneswar
Bhubaneswar, India
nbpuhan@iitbbs.ac.in

***Abstract*—The past few decades have witnessed the great progress of unmanned aircraft vehicles (UAVs) in civilian fields, especially in photogrammetry and remote sensing. 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.**

***Index Terms*—unmanned aircraft system (UAS), unmanned aerial vehicle (UAV), Geolocalization, sensors, digital twins.**

I. 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] [Fig. 1], jMavSim [2][Fig. 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.

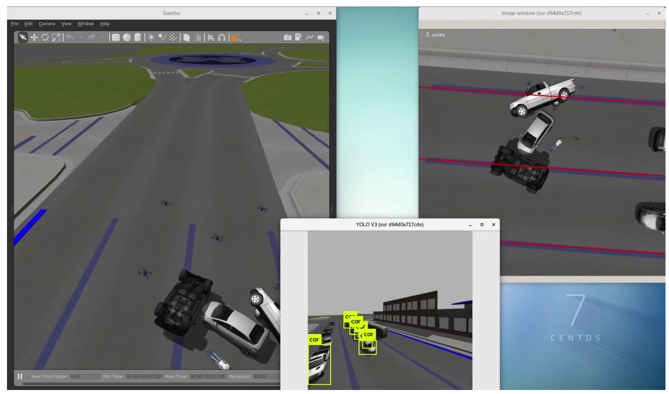


Fig. 1. Gazebo drone fleet simulation.

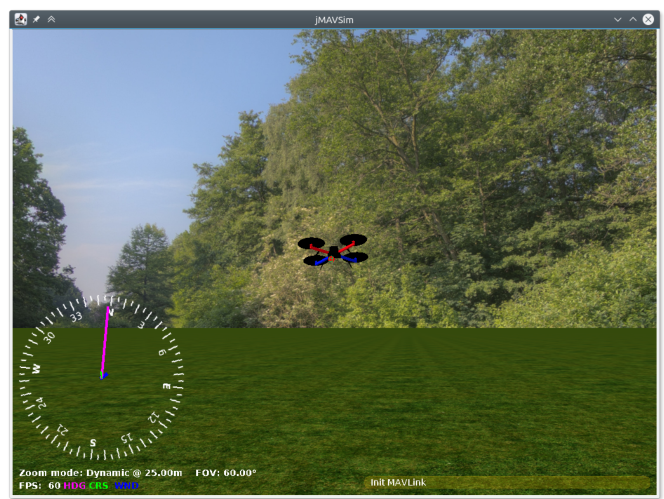


Fig. 2. jMavSim, a simple multirotor simulator with MAVLink protocol support.

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 simulation environment based on AirSim [3]. AirSim provides us with a more realistic simulation environment and rich data interface as shown in figure 3. This enhances the authenticity of the simulation. Compared to Fig. 1 and Fig. 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.

Fig. 3. AirSim wind turbine inspection environment [4].

This work focuses on UAV simulation with machine learning (ML)-based UAV target detection by discussing the key issues and difficult problems, and delineating areas of future development. The remainder of this work is organized as follows. Section II briefly summarizes the applications. Section III describes the methodology used.

II. 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.

III. Methodology

A. 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.

B. 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.

C. 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.

Maneuver Python snippet [5]

```
def rotate(t):
>     client.moveByVelocityZAsync(0,0,0,1, ···
```

```
    print(t)
while True:
    # '''
    l=0;u=0;f=0 # v
>   if keyboard.is_pressed('4'): ···
>   if keyboard.is_pressed('8'): ···
>   if keyboard.is_pressed('6'): ···
>   if keyboard.is_pressed('5'): ···
>   if keyboard.is_pressed('7'): ···
    if keyboard.is_pressed('9'):
        u+=fa; print('u+1')
    if keyboard.is_pressed('1'
    ) and d>0.1 and fa>0.1:
        fa-=1; print(fa); #f = f/fa .1 d
>   if keyboard.is_pressed('3'): ···
>   client.moveByVelocityBodyFrameAsync( ···
    # '''
>   if keyboard.is_pressed('s'): ···
>   if keyboard.is_pressed('a'): ···
>   if keyboard.is_pressed('w'): ···
>   if keyboard.is_pressed('d'): ···
>   if keyboard.is_pressed('q'): ···
>   if keyboard.is_pressed('e'): ···
>   if keyboard.is_pressed('n'): ···
    if keyboard.is_pressed('m'):
        t-=10; rotate(t)
>   if keyboard.is_pressed('r'): ···
```

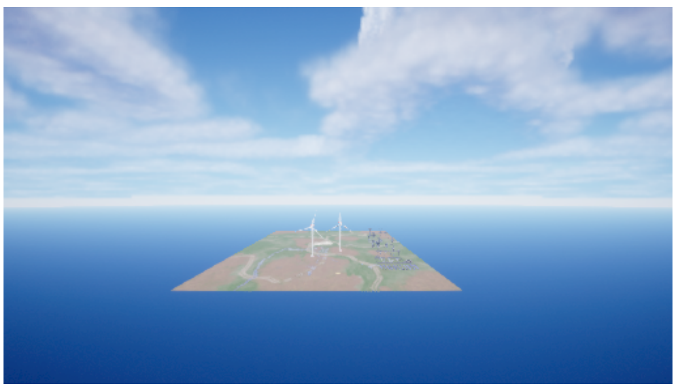


Fig. 4. 1 of 100 captured Images.

TABLE II
TARGETS AND THEIR PIXEL COORDINATES IN IMAGE 1 [FIG. 4]

Target number	PixelX	PixelY
1	236	190
2	190	205
3	349	215
4	291	171
5	257	186
6	309	193
7	274	196
8	246	189
9	208	171
10	128	216

Data: 1000

In this work, machine learning approached in static and video target geolocation. UAV data: The public, UAV-borne dataset for target geolocation is mainly visible data, and the image size is 512 × 288. There are only one multiple source data containing ground targets dataset.

IV. CHALLENGES

Target geolocation and tracking tasks in the UAV remote sensing video face many challenges, such as image degradation, uneven target intensity, small object size, and background complexity.

- ***Image degradation* [6].** The load that a mini-UAV platform carries is strictly limited in terms of weight, volume, and power. Rapid movement changes in the external environment (such as light, cloud, fog, rain, etc.) cause aerial images to be fuzzy and noisy, which inevitably leads to image degradation . High-speed flight or camera rotation also increases the complexity of object detection. Thus, it is necessary to carry out image pre-processing, such as noise reduction, camera distortion correction, etc., to ensure the effectiveness of the model.
- ***Uneven target intensity.*** The image acquisition equipment of a UAV typically uses a large aperture, fixed focal, and wide-angle lens. In addition, flexible camera movement results in an uneven density of captured objects. Some of them are densely arranged and overlap many times, so that it is easy to repeat detection. This could be infered from the image dataset target pixels with its corresponding pixel locations. Some are sparse and unevenly distributed, so that it is prone to missed detection.
- ***Target size.*** UAV remote sensing images can be acquired at different altitudes, yielding photographs containing any size of ground targets. This challenges the classical DL-based method. Ground objects in UAV remote sensing are primarily shown as images with an area smaller than 32 × 32 pixels.
- ***Real-time.*** Target geolocation or tracking in a video obtained by a drone needs to quickly and accurately locate moving ground objects, so real-time processing performance is highly essential.

TABLE I
SMALL SAMPLE OF ACUMULATED DATA

Altitude	Latitude	Longitude	PixelX	PixelY	Ylatitude
120.0288773	47.6463748721171	-122.142564917731	0	0	47.64146793
222.535171508789	47.6374925258878	-122.140164908372	208	171	47.63749253
224.43408203125	47.6374925270728	-122.140164908367	208	171	47.63749253
224.434707641601	47.6374925270732	-122.140164908366	208	171	47.63749253
224.739074707031	47.6374925272631	-122.140164908367	208	171	47.63749253
225.413589477539	47.637492527684	-122.140164908367	208	171	47.63749253
226.083831787109	47.6374925281023	-122.140164908368	207	171	47.63749253
226.480072021484	47.6374925283495	-122.140164908368	208	172	47.63749253
227.046447753906	47.637492528703	-122.140164908369	208	172	47.63749253
227.435272216796	47.6374925289456	-122.140164908369	208	172	47.63749253
227.864837646484	47.6374925292136	-122.140164908369	208	172	47.63749253
228.462814331054	47.6374925295868	-122.140164908369	208	172	47.63749253
228.972915649414	47.6374925299051	-122.140164908369	209	172	47.63749253
229.439010620117	47.637492530196	-122.140164908368	209	172	47.63749253

Haversine distance:

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

V. RESULTS

A. Cross validation

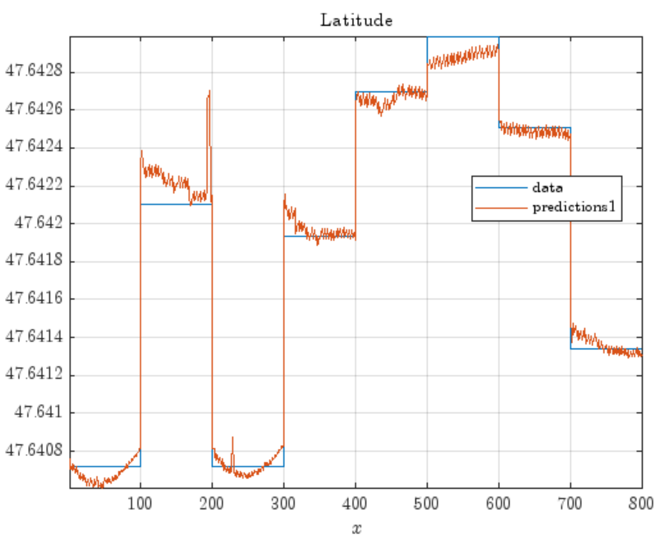


Fig. 5. Testing with training data.

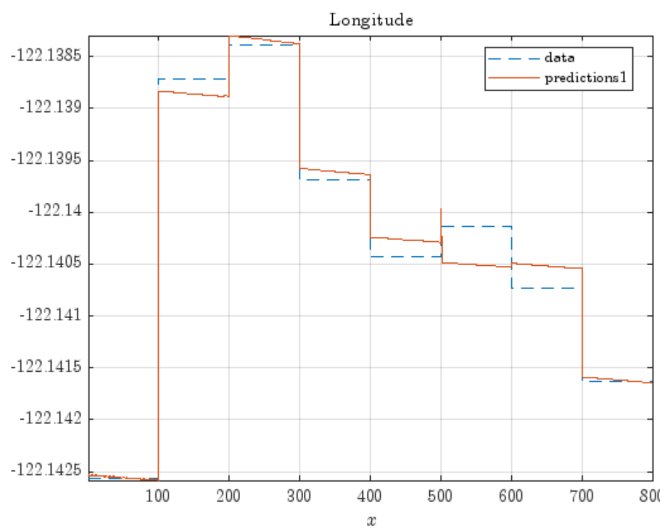


Fig. 6. Corresponding tested parameter.

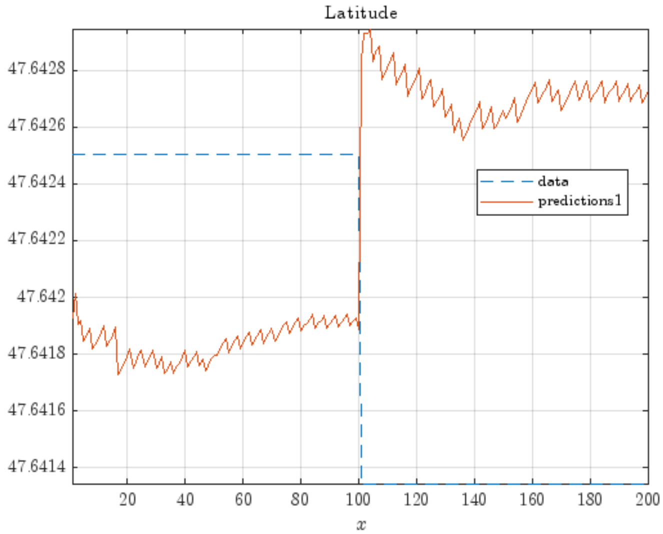


Fig. 7. Cross validation: 1 of 5 parameter test.

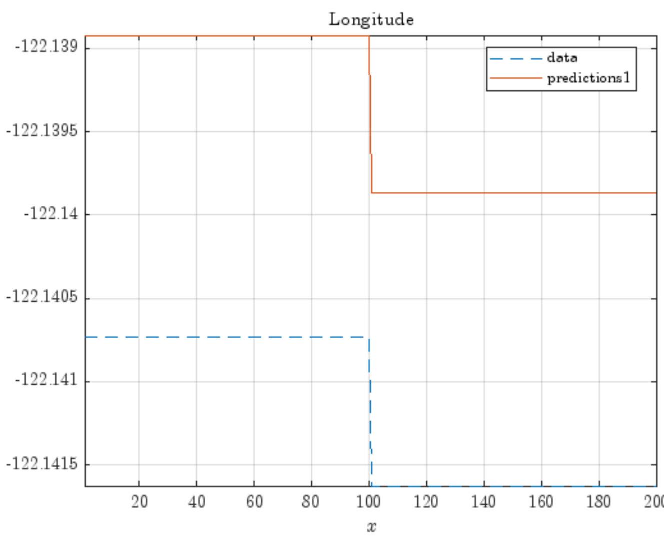


Fig. 8. Corresponding Cross validation: 2:10 data set for testing.

TABLE III
CROSS VALIDATION RESULT

	Minimum	Maximum
Distance(m)	547.4	682.9
Position	200	200

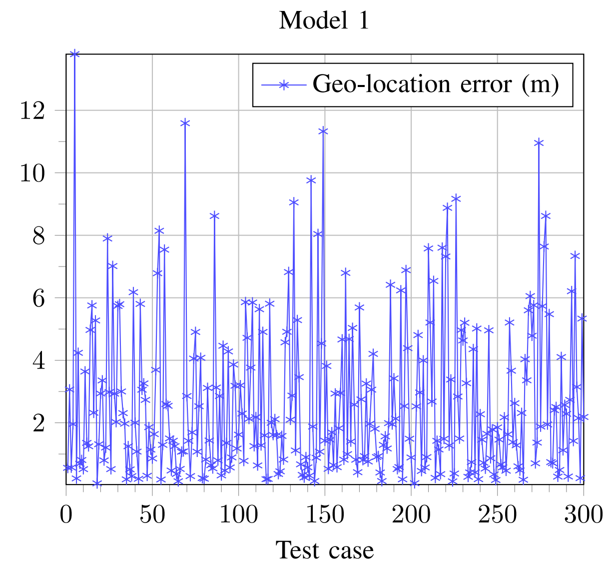
B. Without cross validation

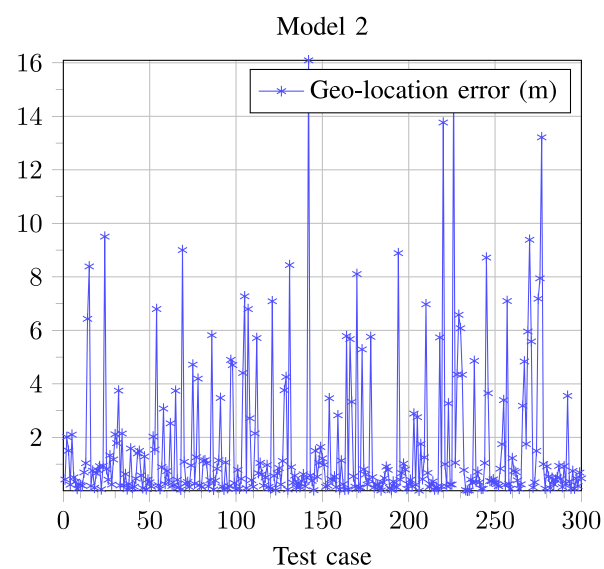
TABLE IV
TRAINING AND TESTING

	Training	Testing
Latitude	700×6 table	300×6 table
Longitude	700×6 table	300×6 table

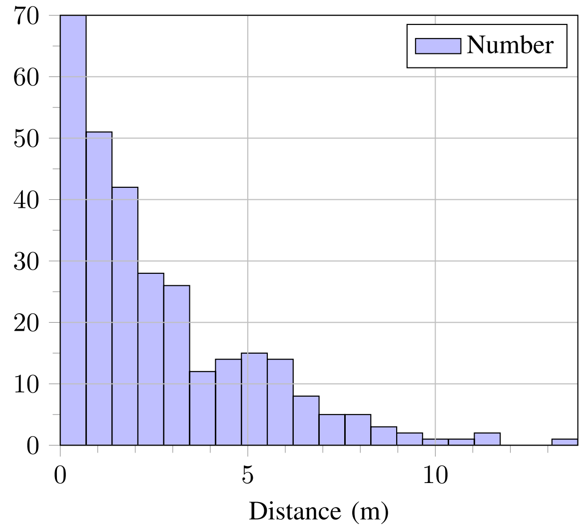
TABLE V
MODELS WITH PROPERTIES AS PER MATLAB'S FITRGP() FUNCTION

Properties	Model 1	Model 2
KernelFunction	-	ardsquaredexponential
FitMethod	-	sr
PredictMethod	-	fic
Standardize	-	1
Sigma	0.002	0.002

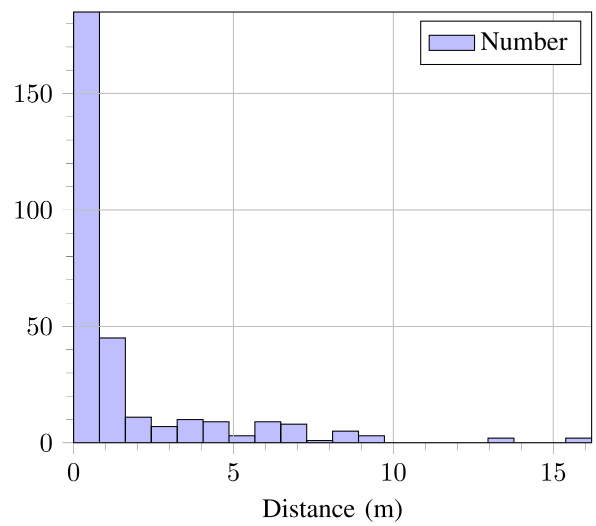




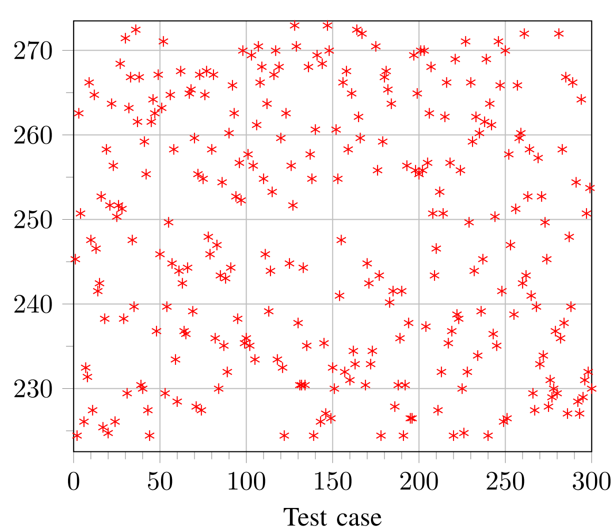
Model 1



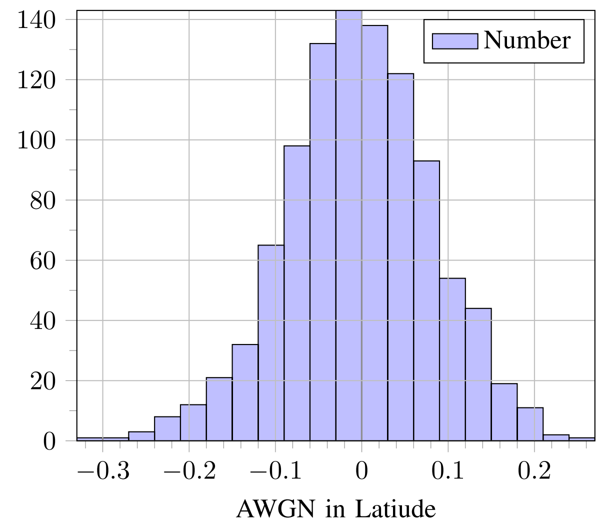
Model 2



Corresponding altitude

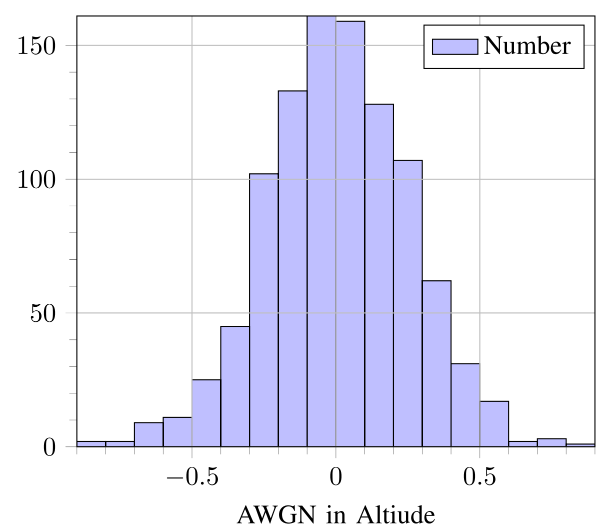


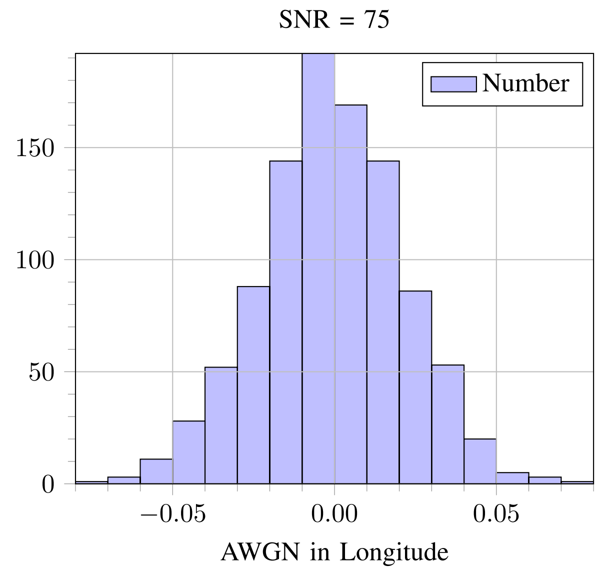
SNR = 55

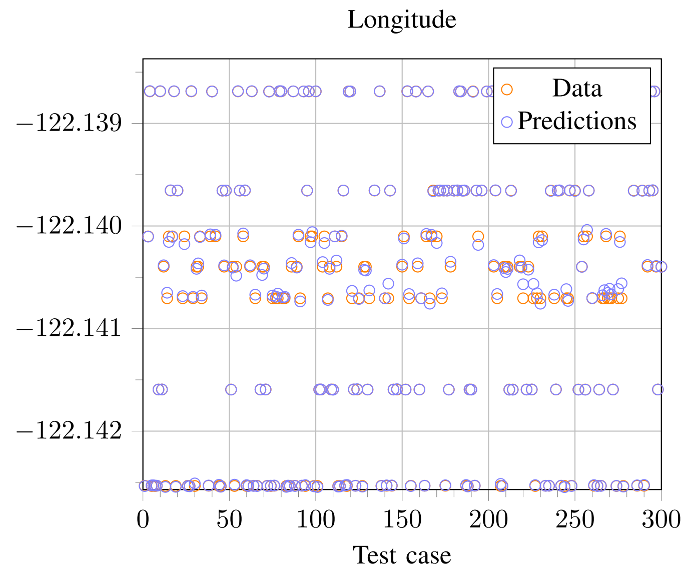


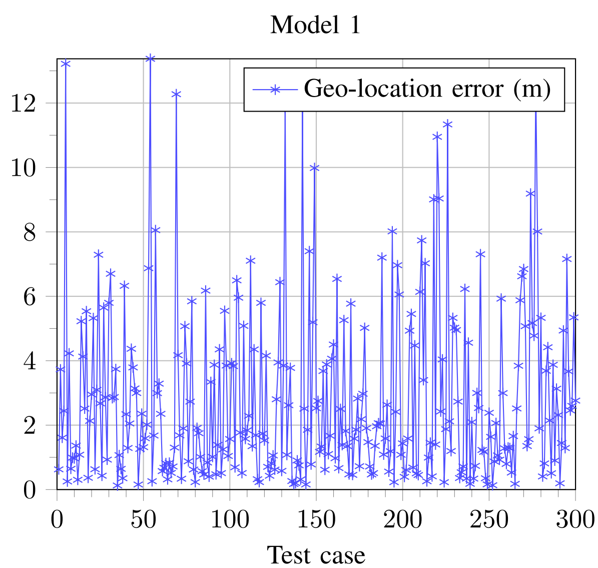
C. Without cross validation but adding noise

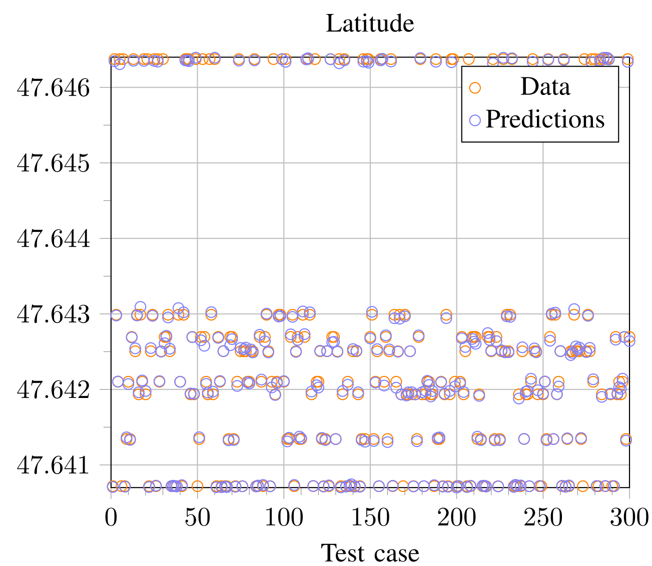
SNR = 60

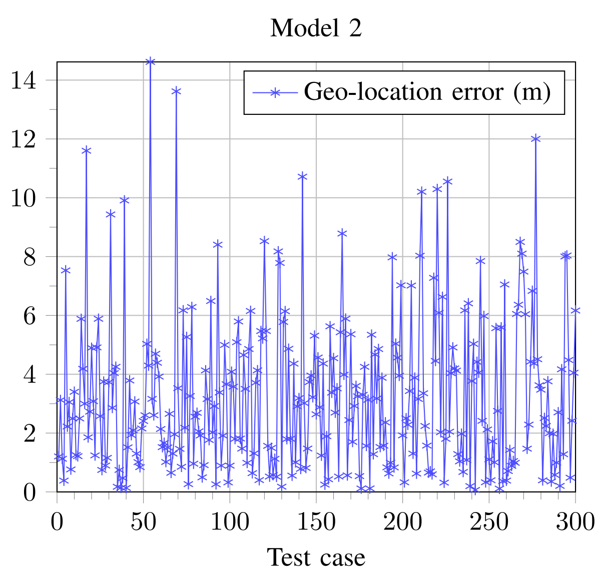


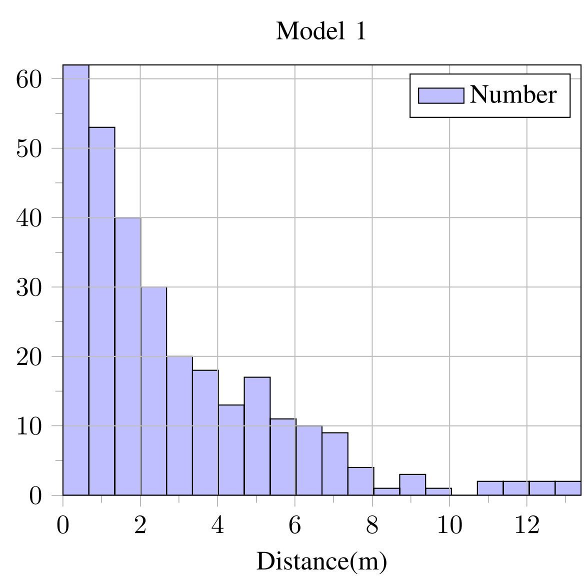

SNR = 75
Number
AWGN in Longitude
150
100
50
0
−0.05
0.00
0.05

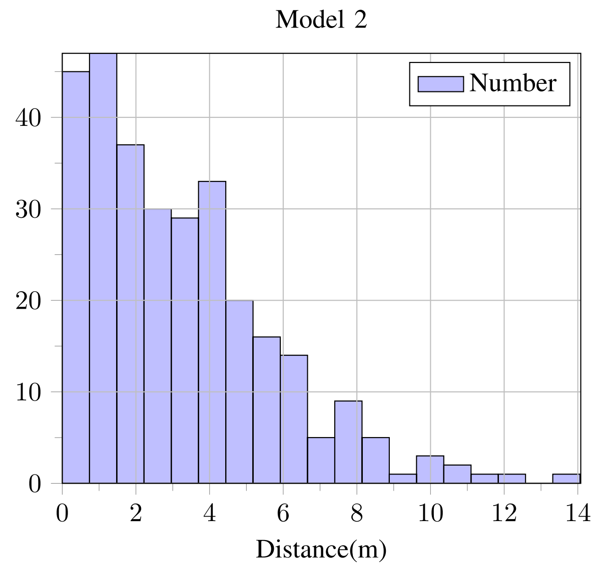

Longitude
Data
Predictions
−122.139
−122.140
−122.141
−122.142
0
50
100
150
200
250
300
Test case


Model 1
Geo-location error (m)
12
10
8
6
4
2
0
0
50
100
150
200
250
300
Test case


Latitude
Data
Predictions
47.646
47.645
47.644
47.643
47.642
47.641
0
50
100
150
200
250
300
Test case


Model 2
Geo-location error (m)
14
12
10
8
6
4
2
0
0
50
100
150
200
250
300
Test case


Model 1
Number
60
50
40
30
20
10
0
0
2
4
6
8
10
12
Distance(m)



Distance(m)	Model 1	Model 2
Maximum	13.3737	14.6186
Minimum	0.1261	0.0669

VI. Future Work

More sets of images may be collected and correspondingly the n target locations be acquired for training and testing. The images can be passed through a convolutional neural networks for 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 [7] 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.Geo-location error can be checked for more targets and / or acquired data.

Acknowledgments

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 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.

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] Sai Vemprala et al. *ChatGPT for Robotics: Design Principles and Model Abilities*. Tech. rep. MSR-TR-2023-8. Microsoft, February 2023. URL: https://www.microsoft.com/en-us/research/publication/chatgpt-for-robotics-design-principles-and-model-abilities/.

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

[6] Xin Wu et al. “Deep Learning for Unmanned Aerial Vehicle-Based Object Detection and Tracking: A survey”. In: *IEEE Geoscience and Remote Sensing Magazine* 10.1 (March 2022), pp. 91–124. ISSN: 2373-7468. DOI: 10.1109/mgrs.2021.3115137. URL: http://dx.doi.org/10.1109/MGRS.2021.3115137.

[7] 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.