

AUTONOMOUS VISION BASED NAVIGATION OF UAV USING DEEP NEURAL NETWORKS ON UNKNOWN TRAILS

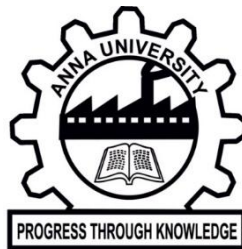
A THESIS

Submitted by

ARVINDH R

in partial fulfilment for the award of the degree of

**MASTER OF ENGINEERING IN
MECHATRONICS**



DEPARTMENT OF PRODUCTION TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY, CHENNAI

APRIL 2018

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this thesis titled “**AUTONOMOUS VISION BASED NAVIGATION OF UAV USING DEEP NEURAL NETWORKS ON UNKNOWN TRAILS**” is the bonafide work of **ARVINDH R (2016608004)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. A. RAJADURAI

Professor and Head

Department of Production technology

Madras Institute of Technology

Anna University, Chennai – 600 044

Dr. P. GANESH

Assistant Professor and Supervisor

Department of Production technology

Madras Institute of Technology

Anna University, Chennai – 600 044

Mr. Shashank Bhushan

Scientist ‘c’ and External supervisor

Centre for Artificial Intelligence and

Robotics (CAIR), DRDO,

CV Raman Nagar, Bangalore-560093

திட்டப்பணிச்சுருக்கம்

பலவிதமான பயன்பாடுகளால், ஆளில்லா வான்வழி வாகனம் உச்ச ஆராய்ச்சி ஆர்வமாக உள்ளன. சுற்றுச்சூழலில் மாற்றங்களை எதிர்கொள்ளவும், மீண்டும் திட்டமிடல், பார்வை மூலம் வழிநடத்துதல், மோதல் தவிர்த்தல் மற்றும் தொடர்பு இணைப்பு வரம்புகளுக்கு வெளியே செயல்பட முடியும் என ஆளில்லா வான்வழி வாகனத்திற்கு ஒரு அணுகுமுறை கட்டுப்பாட்டை வளர்த்துக் கொள்ள இந்த முன்மொழிவு நோக்கமாக உள்ளது. மற்றும் ஜி.பி.எஸ்ஸின் கீழ் உள்ள தொடர்பு இணைப்பு வரம்புகளுக்கு வெளியில் இயங்கக்கூடிய சூழல் நிலப்பரப்பு வழியாக ஒரு ஆளில்லா வான்வழி வாகனம் கட்டுப்பாட்டு அமைப்பை உருவாக்க இந்த திட்டம் உதவுகிறது. ஆழமான நரம்பு பின்னலமைப்பு போன்ற செயற்கை நுண்ணறிவு நுட்பத்தைப் பயன்படுத்தி, ஆளில்லா வான்வழி வாகனம் ஆனது அதன் தன்னியக்க நடவடிக்கைகளுக்கு தேவைகளை பூர்த்தி செய்ய திட்டமிடப்பட்டுள்ளது. ஆழமான நரம்பு பின்னலமைப்பு போன்ற இயந்திர கற்றல் நுட்பத்தை பயன்படுத்தி, ஆளில்லா வான்வழி வாகனம் ஆனது தேவையான நடவடிக்கைகளுக்கு தன்னியக்கமாக கட்டுப்படுத்தப்படலாம் மற்றும் அதன் செயல்திறனை சரி செய்ய மனிதத் தலையீடு தேவையில்லை. ஜி.பி.எஸ்ஸில் தெரியாத சுவடுகளைத் தொடர்ந்து பார்வையிடாமல் சுற்றுச்சூழல் உணர்கருவி மூலம் மட்டுமே தன்னியக்கமாக பயன்படுத்தப்படுகிறது. மாதிரியை பயிற்சி மற்றும் ஆழமான நரம்பு பின்னலமைப்பு வன்பொருள் சுழற்சியில் உருவகப்படுத்துதல், ஒரு முன்மாதிரி மற்றும் தேர்வுமுறை நிகழ்நேர சோதனை இந்த திட்டத்தில் நடத்தப்படும்.

ABSTRACT

Unmanned Aerial Vehicles (UAV) are in peak research interest because of its wide variety of applications. This proposal aims at developing a control for an UAV to navigate through complex terrain without human intervention or control, situational awareness, collision avoidance, on board information processing and can be able to operate outside the communication link limits under GPS denied environments. Using artificial intelligence (AI) technique such as Deep Neural Network, the UAV is programmed for its autonomous operations to satisfy the requirements. By using machine learning technique such as Deep Neural Network (DNN), the UAV can be controlled autonomously for the required operations and need not be required human intervention for tuning its performance. An existing DNN model such as TrailNet DNN is used for following the unknown trails on a GPS denied environment autonomously only through vision sensor (monocular camera) and the DNN model is trained using the manually collected dataset of trails. The project covers the collection of image dataset through manual flight, training the DNN model and implementation of autonomous control of DNN to the UAV.

ACKNOWLEDGMENT

I express my great heartfelt gratitude to our Dean **Dr. A. RAJADURAI**, Professor and Head of the Department, Production Technology, MIT campus, Anna University, Chennai for his constant encouragement and the involvement that he has shown on undergone the Phase II project.

I also express my deep sense of gratitude and indebtedness to my internal supervisor **Dr. P. GANESH**, Assistant professor, Department of production Technology, Madras institute of Technology, Anna University, Chennai and my external supervisor **Mr. SHASHANK BHUSHAN**, Scientist 'C', Centre for Artificial Intelligence and Robotics (CAIR), DRDO, Bangalore.

I acknowledge my sincere thanks to **Dr. MURALI KRISHNA**, Scientist 'F' and **Mr. ARUN PRAKASH RAO**, scientist 'D', CAIR, DRDO, Bangalore for their constant support on carrying out the project.

I acknowledge my sincere thanks to the project committee members **Dr. R. SIVARAMAKRISHNAN**, Associate Professor, **Dr. A. SIDDARTHAN**, Assistant professor, **Dr. C. NANDAKUMAR**, Assistant Professor, **Dr. P. KARTHIKEYAN**, Assistant Professor and **Ms. S. SRIDEVI**, Teaching fellow for the valuable suggestions during the review of project.

Last but not the least I am indebted to my parents, friends for their love and affection, which give me the strength for everything.

ARVINDH R

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT (TAMIL)	iii
	ABSTRACT	iv
	LIST OF FIGURES	ix
	LIST OF TABLES	x
	LIST OF SYMBOLS AND ABBREVIATIONS	xi
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 DEEP NEURAL NETWORKS	2
2	LITERATURE REVIEW	3
	2.1 OBJECTIVE OF THIS PROJECT	7
	2.2 NEED FOR THIS STUDY	7
	2.3 SCOPE OF THIS PROJECT	7
3	THEORETICAL BACKGROUND	8
	3.1 HARDWARE SELECTION	8
	3.2 SOFTWARE ARCHITECTURE	9
	3.2.1 System design	9
	3.2.2 Vision based processing	10
	3.3 METHODOLOGY	11
	3.4 IMPLEMENTATION	12

4	HARDWARE SETUP AND INITIAL FLIGHT TESTING	14
	4.1 FLIGHT CONTROLLER SETUP	14
	4.1.1 Assembling the UAV	15
	4.1.2 Calibration of IMU sensors on flight controller	15
	4.2 COMPANION COMPUTER SETUP	16
	4.2.1 Jetson TX1 setup	16
	4.2.2 Raspberry pi with Movidius NCS	16
	4.3 MAVROS NODE SETUP	17
	4.3.1 MAVROS	18
	4.3.2 MAVROS with companion computers and GCS	18
	4.4 GCS SETUP FOR INITIAL FLIGHT TESTING	19
	4.4.1 Manual mode	20
	4.4.2 Stabilized mode	20
	4.4.3 Position hold mode	20
	4.4.4 Mission mode	20
	4.4.5 Defining the waypoints on GCS	21
5	TRAINING AND TESTING OF DNN IN NVIDIA DIGITS	23
	5.1 INTRODUCTION	23
	5.1.1 Advantages of deep neural network	24

	5.2 TRAINING OF DNN IN NVIDIA DIGITS	24
	5.2.1 Dataset preparation	24
	5.2.2 Defining the DNN model	25
	5.2.3 Training the DNN model	26
	5.2.4 Testing of the model using DIGITS tool	27
6	SOFTWARE IN THE LOOP SIMULATION	29
	6.1 SITL ENVIRONMENT	29
	6.2 SOFTWARE ARCHITECTURE AND CONNECTIONS	29
	6.4 RUNNING THE DNN NODE ON SIMULATION ENVIRONMENT	31
7	RESULTS AND DISCUSSIONS	33
	7.1 SIMULATION RESULTS	33
	7.2 FLIGHT LOG DATA ANALYSIS	34
	7.3 OFF-BOARD TESTING OF UAV FOR AUTONOMOUS NAVIGATION USING DNN	35
8	CONCLUSION	36
	8.1 FUTURE WORK	36
	APPENDIX I	37
	APPENDIX II	38
	REFERENCES	39

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	System design.	10
3.2	Methodology of the project.	11
4.1	Fully assembled quadrotor UAV	15
4.2	Calibration of IMU sensors in flight controller.	16
4.3	Raspberry pi with Movidius NCS	17
4.4	Serial connection between pixhawk and Jetson TX1 carrier board	19
4.5	Waypoints loaded on GCS from MAVROS node.	21
4.6	Overview of initial flight testing	21
4.7	MAVROS node loaded on companion computer	23
4.8	Initial flight testing done on CAIR campus, DRDO using waypoints (video feed from on-board camera)	23
5.1	Preparing the dataset for training the DNN	25
5.2	Defining the DNN model on DIGITS	26
5.3	Caffe prototext file loaded in DIGITS tool	27
5.4	Trained model and its learning rate	27
5.5	Testing the model using a single test image on DIGITS	28
5.6	Predictions of single image input for trained model	28

6.1	Software In-The-Loop Simulation Model.	30
6.2	PX4 simulation environment with Gazebo	31
6.3	SITL of UAV with DNN node running under MAVROS	32
7.1	Given input video stream on DNN for SITL	34
7.2	Mag sensor inconsistency log data for yaw angular rate	35
7.3	Mag sensor inconsistency for yaw angle	35

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface
DNN	Deep Neural Network
FDM	Flight Dynamics Model
FFNN	Feed-Forward Neural Network
G	Acceleration due to gravity
GCS	Ground Control Station
GPIO	General Purpose Input Output
I²C	Inter Integrated Circuit
IMU	Inertial Measurement Unit
L	Lift force
MAV	Micro Aerial Vehicle
PID	Proportional Integral Derivative
RC	Remote Control
RMSE	Root Mean Square Error
S	Span of the wing
SITL	Software In-The-Loop
UAV	Unmanned Aerial Vehicle
V	Volume of the entire vehicle
W	Overall weight of the vehicle
A	Angle of Attack

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Unmanned Aerial Vehicles are an unrestrained rigid body in space having six degrees of freedom, three translating movements along longitudinal, lateral and vertical axis, and three rotational moments around the same axis. The rotational moments are commonly referred to as roll, pitch and yaw, and the controlled manipulation of these moments, using vehicle actuators, establishes the desired attitude. The traditional actuators on a plane are: aileron, elevator and rudder that respectively provide the roll, pitch and yaw. We also have the throttle, that controls the thrust produced by the motor. In this work, we use a Quadrotor UAV which have four propellers powered by electrical motors. UAV is any aircraft that does not have a pilot on board, and they can be autonomous, semi-autonomous or remotely controlled. There are a huge variety of sizes, forms and functionality, being classified mainly by functionality, sustainability and dimensions.

The demand for UAV has been grown for recent decades for diverse factors in both military and civilian applications like surveillance, search and rescue, missile launching etc. The flight dynamic of an aircraft is a complex nonlinear system. Classic control theories use a simplified mathematical model to optimize the response to make the control system acceptable. As dynamic systems models present a complex behaviour, solutions to these systems should be based on new concepts and advanced controlling technologies. Nowadays, a known classical controller, proportional, integral and derivative (PID) is widely used[1].

Due to relatively simple implementation and good performance, the PID has been largely used in all kinds of systems, even flight control ones.

The flight dynamics of an aircraft is a complex nonlinear system. Thus, it is not possible to model a high order differential equation for this control system. Classic control theories use a simplified mathematical model to optimise the response to make the control system acceptable. However, the individuality of each flight may demand constant recalibration of the parameters of the control system when using conventional system controls, due to changes in the flight characteristics as: payload, centre of gravity, flight conditions and environment. Moreover, several methods using AI techniques have been proposed to substitute the classical controller, like Deep neural networks. But some of these solutions require a high computational power, which is not always available.

1.2 DEEP NEURAL NETWORKS

The human brain is a source of intelligence and a truly remarkable parallel computer. The brain processes incomplete information obtained by the perception at an incredibly rapid rate. Nerve cells function about 10^6 slower than an electronic circuit gates, but human brains process visual and auditory information much faster than modern computers.

Inspired by biological nervous systems, many researchers have been exploring Deep neural networks. They model the brain as a continuous-time nonlinear dynamic system in connectionist architectures that are expected to mimic brain mechanisms to simulate intelligent behaviour.

An adaptive network, as the name indicates, is a network structure consisting of several nodes connected through directional links. Each node represents a process unit, and the links between nodes connected through directional links. Each node represents a process unit, and the links between nodes specify the casual relationship between the connected nodes.

CHAPTER 2

LITERATURE REVIEW

De Oliveira et al. (2017) used a Enhanced PID controllers used. Control using ANFIS algorithm for flight dynamics and genetic algorithm for attitude (roll, pitch and yaw) optimization. Constant recalibration and adaptation required due to varying environment. So, AI techniques can be used for adaptability.

Konam et al. (2017) designed a two layered Neural Network (NN) is used for navigation and obstacle avoidance. Vision based navigation by depth perception. Monocular camera with Kinect sensor used for depth perception. Off-board GPU is used for computation. Automatic Patch Pattern Labelling for Explanation (APPLE) algorithm is used for labelling each neuron in convolutional layers. Labelled neurons are used later for ranking the best output.

Sathyan et al. (2017) proposed a solution for Reducing the uncertainty in velocity and maneuver parameters using genetic and fuzzy algorithm. A special neuro-fuzzy technique is used. Objective is to find a conflict-free trajectory of an aircraft. Genetic algorithm is used for comparison of results from a fuzzy system.

Julian et al. (2017) proposed an optimal solutions are computed offline to guide autonomous UAVs. The problem of finding the fastest trajectory to a waypoint is framed as a Markov decision process and solved using discrete value iteration. neural network was trained from the MDP solution to accurately compress the MDP table with negligible loss. Flight results with the neural network guidance system are promising and show that neural networks have great

potential as on-board guidance systems or for generating optimal trajectories. Comparison of the trajectories generated mixed-integer linear programming under the same constraints as the MDP, negligible differences is found in the generated trajectories.

Smolyanskiy et al. (2017) proposed a control algorithm for autonomous navigation of a MAV to follow the unknown trails using Deep Neural Network. The designed control commands the MAV to follow a forest trail on a GPS denied environment using a DNN. The model was trained using a IDSIA forest trail dataset and deployment is done in a on-board GPU along with a flight controller for basic flight attitude control. It has the ability to generate environmental awareness with additional perceptual modules, including DNN-based object detection and obstacle detection via monocular visual odometry.

Muliadi et al (2017) compared the advantages of ANN over conventional control method especially for the UAV attitude controller. The ANN applied the learning mechanism to inversely mimicking the output of a system dynamics with respect to its input. In this paper, they found that it has a better capability in controlling the nonlinear systems. The comparison composed from the twin ANN model usage, the conversion into PWM values, the simultaneous numerical simulation has been done. The result of this paper shows that the ANN performance was better than the PID controller in handling the UAV altitude dynamics.

Canciello et al (2017) explains the large effect of wind due to the small scale, and the absence of aerodynamic data at low speed and Reynolds numbers are considered for designing controllers. Two nested loops, high-gain controller scheme is proposed to control attitude and both longitudinal and vertical speed. Two high-gain controllers have been considered for attitude and velocity control, and both their independent and joint use has been considered. An exponentially decaying term in the controllers and an integral action mitigate the drawbacks of classic high-gain control.

Atmeh et al (2016) paper proposed a neurodynamic system which is used to adaptively solve the problem of trajectory generation, planning path connecting a locus of points as a function of time. It was shown that by designing an NN with specific system properties and by training two FFNN a neurodynamic system with feedback can be used to adaptively generate complex trajectories. Two examples of trajectory generation were given, one of a circle and the other of a figure eight trajectories. It was noticed that the frequency of the intermediate trajectory generated by the RNN needed to be selected based on the frequency of the desired trajectory.

Schmidhuber et al (2015) paper proposes a Deep Learning in Neural Networks which is relevant for Supervised Learning, Unsupervised Learning, and Reinforcement Learning. A deep Credit Assignment Paths is used. Dynamic Programming is important for both deep Supervised Learning and traditional Reinforcement Learning with deep NNs.

Maravall et al (2015) proposed a Dual Feedforward / Feedback architecture has been used as the UAV's controller and the K-NN classifier using the grey level image histogram as discriminant variables has been applied for landmarks recognition. A solution to the problem of visual landmark recognition by means of a dual control architecture is proposed. So, that the convergence of the UAV's controller to a state of zero error has been achieved. As the controller's error signal, which is based on the difference between the current image captured by the UAV's on board camera and the target image.

Lee et al (2015) in this paper an Off-board computing cloud GPU is used for computation. NN can be used for real time object detection. Training based on the manually recorded data sets. Input from the real-time sensors and pre-input data sets are compared and output is generated.

Selma et al (2013) paper provides a solution in guiding an autonomous vehicle to a desired destination along a desired path in an environment characterized by a terrain and a set of distinct objects, such as obstacles traffic

lights and cars circulating in the trajectory. Where, ANFIS architecture has demonstrated a good performance in modelling the trajectory of an Unmanned Vehicle. The best feature of ANFIS is that it pre-processes all the data into several membership functions before mapping the data into an adaptive neuro structure. This pre-processing feature allows ANFIS to converge faster and better. To be able to have a basis for comparison, an ANN type controller is also designed.

Verbickas (2013) used A fixed wing UAV model with NN based vision navigation is used. A multi-layered Neural Network (NN) used. Support vector machine is used for basic flight dynamics. Stereoscopic (binocular) cameras used for 3D mapping of image edge feature based approach for object detection and navigation.

Rimal et al (2009) used a neural network plant model to assist in the controller training. In this paper the simulation done for the modelling capabilities of a state space neural network, to act as an observer for a non-linear process allowing a simultaneous estimation of parameters and states. This paper considered the simulation of identification of a nonlinear system dynamics using artificial neural networks approach.

Dujrene (2003) proposed a method is outlined which allows a base level application for applying an Artificial Intelligence method, Fuzzy Logic, to aspects of Control Logic for UAV flight. The approaches seen in most of the research included Expert Systems; Neural Networks, Fuzzy Logic, and Hybrid combinations of these and PID control schemes. the application was studied further to gain knowledge of the effects of aircraft speed also. This type of application will allow an Intelligent Agent approach to be implemented later. Attempts can be made to expand the use of AI techniques so a comparison can be done against more conventional flight controllers.

Chang-Su Park et al (2003) used a Model inversion control is augmented with neural network in control the airship. The input-output feedback linearization allows one to design a linear controller from a nonlinear system.

Neural network is used to compensate the nonlinearities resulting from this linearization. The augmented controller is applied to control the airship's velocity, pitch and yaw angle. A single augmented controller was successful in leaning and adapting to the two very different flight conditions.

2.1 OBJECTIVE OF THIS PROJECT

This project aims in developing an autonomous navigation control for a low flying quadrotor UAV using Deep neural network and to navigate autonomously through an unknown trail by using only vision sensor.

2.2 NEED FOR THIS STUDY

To overcome the drawbacks of conventional controllers and to provide a solution for autonomous navigation control of a complex non-linear system like UAVs. A machine learning algorithm (Deep Neural Network) is used in this project for providing a better solution for adaptability and for autonomous operation of a UAV without human intervention beyond the control limit.

2.3 SCOPE OF THIS PROJECT

This autonomous UAV will be potentially useful in surveillance, vision based navigation, military application, target detection etc. this potential use for its autonomy and decision making ability makes well suitable for military and commercial operations for various payload functions in the future.

CHAPTER 3

THEORETICAL BACKGROUND

3.1 HARDWARE SELECTION

The project requires a low altitude trail following aerial vehicle, so a quadrotor UAV can be used for low altitude flying and precise control. To ensure a robust control an inexpensive frame like DJI F450 frame with a brushless DC out runner motor along with a ten inch propellers. The motors are controlled by a electronic speed controllers.

The basic flight attitude and position is controlled by an open source pixhawk flight controller. For processing the DNN model requires higher computation power and massive parallelisation hence an on-board GPU such as Jetson TX1 is required as a companion computer. All the DNN nodes and communication nodes are run Jetson TX1 mounted on a Auvedia J120 carrier board. The companion computer is connected with the pixhawk flight controller using a UART communication protocol on telemetry 2 port of pixhawk and UART port in Jetson TX1.

Initially for data collection a raspberry pi 3 with raspberry pi camera is mounted on-board. Also, a GoPro camera stabilized with a gimbal is mounted on-board for collecting high resolution datasets. The companion computer is connected with pixhawk flight controller on UART using a MAVROS communication node for transferring the sensor data and control commands. A two 3500 Mah Li-Po battery is used to power the UAV system and companion computer.

3.2 SOFTWARE ARCHITECTURE

The software architecture of the UAV's autonomous control consists of a DNN node for autonomous navigation. the output of the DNN is converted into a control using a python script where the output from the SoftMax regression layer of DNN is converted into a telemetry control signal.

Based on the results from literatures surveyed the comparison of various DNN model is carried out and found that the TrailNet DNN outperforms well in autonomous navigation of UAV with better accuracy and autonomy. Hence the TrailNet model is trained for the Custom dataset collected through manual flight of UAV. The below table explains various DNN model comparison of its accuracy and autonomy.

Table 1: comparison of various DNN models

DNN	Number of layers	Time for training	Time required for processing	accuracy	autonomy
Giusti et al.	6	2 hr.	2 ms.	79%	80%
ResNet-18	18	10 hr.	19 ms.	92%	88%
mini AlexNet	7	4 hr.	8 ms.	81%	97%
SqueezeNet	19	8 hr.	3 ms.	86%	98%
TrailNet	18	13 hr.	11 ms.	84%	100%

3.2.1 System design

The whole software control nodes for autonomous navigation runs under the ROS environment on a Debian linux system of the companion computer. Here the companion computer is NVIDIA Jetson TX1 and ROS kinetic is used for all the software control. The camera is the only input for the DNN node. The output of the DNN control node is converted into telemetry signal by using a python script running under MAVROS node which uses a MAVLINK Protocol for communication. The autonomous control can be override by a signal from the

manual joystick or telemetry signal from the Ground Control Station. The system design of the UAV autonomous navigation control is shown in figure 3.1

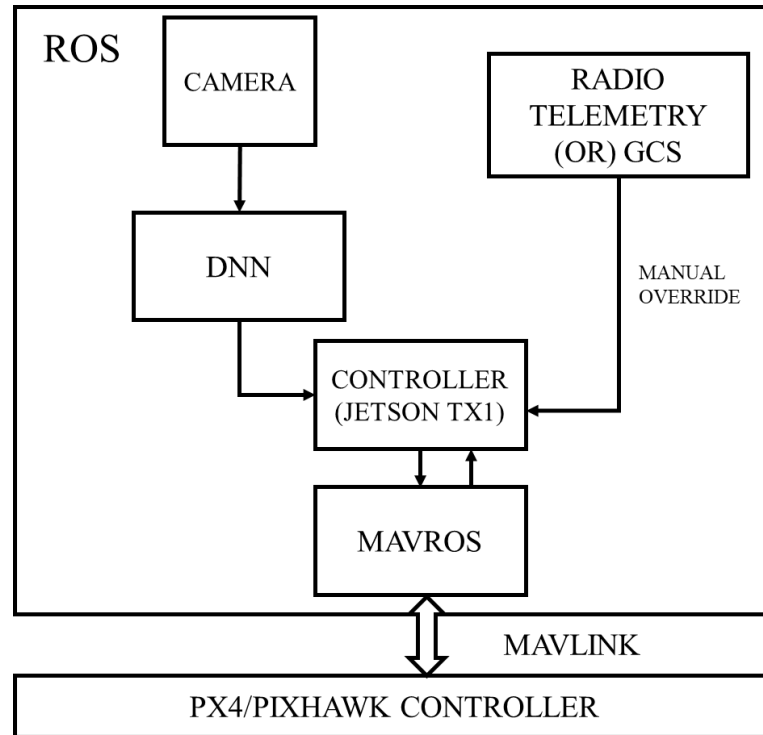


Figure 3.1 System design.

3.2.2 Vision based processing

The DNN node is processed in Jetson TX1 by the input from the vision sensor mounted in front of the UAV. The communication between the companion computer and flight controller is done using a MAVROS node through MAVLINK protocol at the baud-rate of 921600. The control commands from the companion computer and sensor data from the flight controller are transferred between each other on this communication link.

The DNN model used here for control is TrailNet DNN which is a modified version of ResNet-18. The two SoftMax layer is added at the end of the resnet layers which defines the DNN control for two outputs, one is for lateral offset where the translation of the camera view is given out and another one is view

orientation which defines the rotation of the camera from the original trail (Smolyanskiy et al. 2017).

3.3 METHODOLOGY

The methodology of the project is explained in the flow chart shown in Figure 3.2, where The DNN model is trained using the gathered vision sensor data from manual flight and then it is tested in simulation. After the model is implemented in the real-time UAV model for testing.

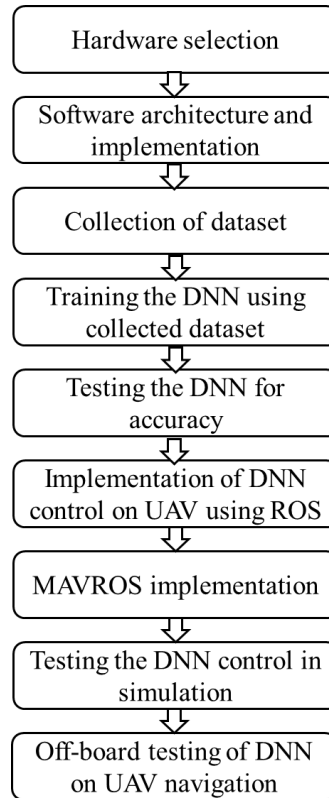


Figure 3.2 Methodology of the project.

Initially the dataset used for training the DNN model is collected using manual flight and through a pre-defined waypoint on the ground control station software. Then the model is trained using those collected dataset and some available dataset like IDSIA trail dataset. Optimization will be carried out after testing of the model for tuning its performance.

3.4 IMPLEMENTATION

The implementation of the DNN control for autonomous navigation of the UAV on unknown trails is explained as follows. The DNN used here is inferred under the ROS node which requires a higher computation. Hence a high-end GPU powered companion computers like Jetson TX1 or a Raspberry pi accelerated with a Movidius Neural Compute Stick (VPU) are used for on-board processing of the UAV system. The UAV's basic flight is controlled by a flight controller, here Pixhawk with a Custom PX4 flight stack is used to work under GPS denied environment.

The companion computer is linked with the pixhawk flight controller through a MAVLINK protocol which sends and receives the data between controller and companion computer which includes control commands and sensor data. These control node is running under a ROS node called MAVROS, where the sensor data and control commands is sent and received between companion computer and the flight controller at the baud rate of 921600. The MAVROS node is connected to a Ground control station through a UDP link on a specific IP address. The Ground control station is used to view, command and control the UAV remotely through a telemetry link and MAVROS link.

The TrailNet DNN node running in the companion computer outputs a two SoftMax layer of data having 6 outputs whereas, 3 of lateral offset for translation and 3 of view orientation for rotational correction. The DNN is trained as a classification problem and hence the output of three each will be predictions of left, centre and right of the trail. The output from the DNN is converted to control commands using the following relation.

$$\alpha = \beta_1 (y_{right}^{vo} - y_{left}^{vo}) + \beta_2 (y_{right}^{lo} - y_{left}^{lo}) \quad (3.1)$$

Where,

α – turning angle.

β_1 and β_2 - scalar angle parameters.

$$\text{Also,} \quad \varphi(y) = \begin{cases} y_{right} & \text{if } i = right \\ y_{left} & \text{if } i = left \\ 0 & \text{if } i = centre \end{cases} \quad (3.2)$$

The above relation equation 3.1 and 3.2 is used to convert the DNN predictions to turning angles. once the turning angle is known a waypoint is set somewhere in front of the UAV along with this turning angle produces the forward waypoint to move the UAV (Smolyanskiy et al. 2017). The new waypoint is then loaded on the pixhawk flight controller for the forward movement of the UAV. The above step will be followed for each prediction of the DNN and the view orientation and lateral offset is corrected to the path at the rate of 30 Hz.

CHAPTER 4

HARDWARE SETUP AND INITIAL FLIGHT TESTING

4.1 FLIGHT CONTROLLER SETUP

A flight controller (FC) is a small circuit board of varying complexity. Its function is to direct the RPM of each motor in response to input. A command from the pilot for the multi-rotor to move forward is fed into the flight controller, which determines how to manipulate the motors accordingly.

Most flight controllers also employ sensors to supplement their calculations. These range from simple gyroscopes for orientation to barometers for automatically holding altitudes. GPS can also be used for auto-pilot or fail-safe purposes. More on that shortly. With a proper flight controller setup, a pilot's control inputs should correspond exactly to the behaviour of the craft. Flight controllers are configurable and programmable, allowing for adjustments based on varying multi-rotor configurations.

Here a pixhawk flight controller is used for basic flight and executing flight commands from the companion computer. The initial setup and sensor calibration must be done for a proper and safe flight, which is discussed later in this chapter.

There are different versions of flight stack is available, here a custom version of PX4 flight stack is used on the pixhawk flight controller with a Q Ground Control(QGC) is used as a ground control station. The calibration of IMU sensor and compass is done using the ground control station.

4.1.1 Assembling the UAV

Assembling the UAV is done before the initial flight test shown in figure 4.1. The ESCs should be calibrated before connecting with the motors and the controller. Using a RC transmitter and receiver the ESC is programmed for a specific application. Then the flight controller is connected with all the sensors, RC receiver, GPS for initial flight planning and companion computer through a MAVROS node. The camera is directly connected to the companion computer for DNN input at 60Hz.



Figure 4.1 Fully assembled quadrotor UAV

4.1.2 Calibration of IMU sensors on flight controller

The calibration of IMU sensor is done in ground control station shown in figure 4.2, where the flight controller is connected with ground control station through a MAVLINK from a MAVROS node running on a companion computer or through a telemetry link of bandwidth 433MHz. each sensor of the flight controller is calibrated individually and the offset is updated in the flight controller through GCS.

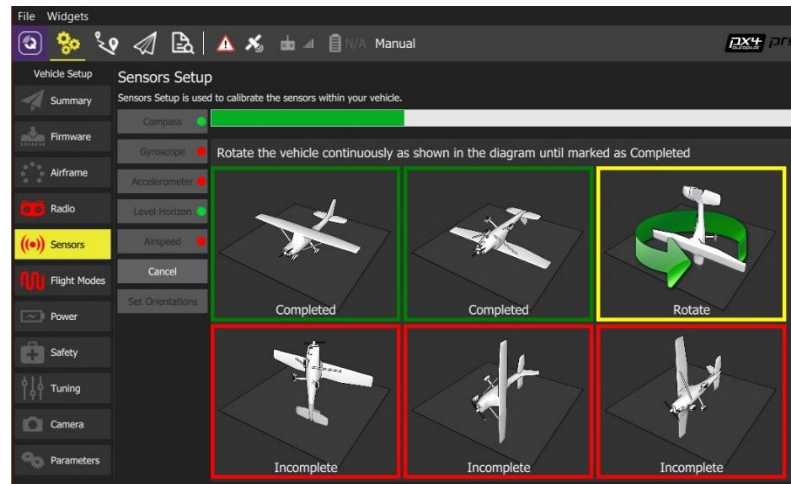


Figure 4.2 Calibration of IMU sensors in flight controller.

4.2 COMPANION COMPUTER SETUP

The companion computer need to be setup before mounting it on the UAV. the companion computers run on a Debian Linux OS. Requires ROS, Gstreamer, Caffe deep learning library, OpenCV and MAVROS for running a DNN node to control the UAV autonomously.

4.2.1 Jetson TX1 setup

NVIDIA Jetson TX1 is development computer with a GPU powered processor for embedded deep learning applications. Its requires an initial setup for on-board and off-board processing of DNN. The dependencies need to be pre-installed for processing the DNN real-time. the whole DNN processing and communication is done under ROS environment, hence ROS kinetic is installed for processing. It also requires Caffe deep learning library for processing the DNN, MAVROS for communicating control commands and gathering sensor data from the flight controller. Opencv is required for real-time processing of the images.

4.2.2 Raspberry pi with Movidius NCS

The companion computer can be either on-board or off-board. On-board processing unit is robust and doesn't have a delay in communication but off-

board processing has some delay on communication and requires a lesser proximity between the flight controller and the computer for efficient communication. Here, initially the quadrotor UAV which has been built for initial testing was unable to carry the load of the Jetson board due to low thrust of the motors and high battery power consumption of Jetson reduced the total flight time by less than 4 mins.

Hence as an alternative and for reducing the load and power consumption, a lower end computer like raspberry pi is used as a companion computer initially for sending flight commands and receiving sensor data through MAVROS. For running the DNN node on raspberry pi is not possible because of low performance in-built GPU available in Raspberry pi. Hence another device which accelerates the Deep learning applications called Movidius NCS shown in figure 4.3 is used for running the DNN node in raspberry pi. Which runs on the power obtained from raspberry pi's USB port.



Figure 4.3 raspberry pi with Movidius NCS

4.3 MAVROS NODE SETUP

The setup is for bridging the flight controller, companion computer and Ground Control Station, the MAVROS node setup is explained as follows.

4.3.1 MAVROS

The MAVROS provides communication driver for various autopilots with MAVLink communication protocol. Additionally it provides UDP MAVLink bridge for ground control stations and companion computers. The following are the features of MAVROS used in UAV control.

- Communication with autopilot via serial port, UDP or TCP.
- Internal proxy for Ground Control Station (serial, UDP, TCP)
- Plugin system for ROS-MAVLink translation
- Parameters and Waypoint manipulation tool
- OFFBOARD mode support
- Geographic coordinates conversions.

4.3.2 MAVROS with companion computers and GCS

The MAVROS is installed on the companion computer using ROS commands to receive the sensor data and send the control commands to the flight controller. Initially the MAVROS is installed on a linux platform using the ROS installation package or using the source installation.

The Ros launch command is used for launching the MAVROS node on the companion computer. Here the flight stack used is PX4 flight stack hence px4 launch file is used. The UART port of the companion computer is accessed through the serial pins of Tx and Rx. For Jetson TX1 the UART1 port shown in figure 4.4 is used for communicating with the flight controller through telemetry 2 port at the baud rate of 921600. And for the raspberry pi the GPIO 6 and 8 is used for connecting serially with flight controller. And the GCS is connected via UDP connection through a specific IP of the ground host computer.

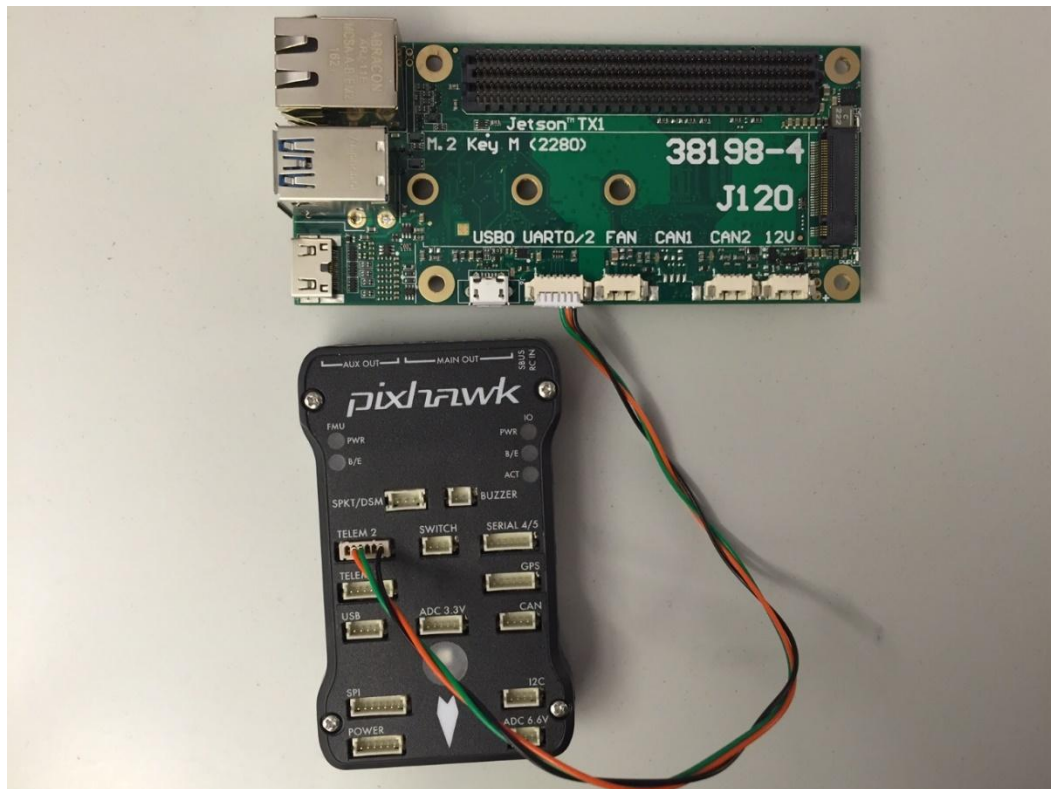


Figure 4.4 serial connection between pixhawk and Jetson TX1 carrier board

4.4 GCS SETUP FOR INITIAL FLIGHT TESTING

The ground control station is a software tool used for controlling, viewing and tracking a UAV with the help of telemetry link and MAVLink communication protocol. The initial flight testing is carried out for testing the flight stability, response to the command and sensor datalogging for flight failures.

The initial flight test has been carried for various modes of flight with the help of flight controller and ground control station. The various flight modes which has been carried out during the test flight is as follows

- (i). Manual mode
- (ii). Stabilized mode
- (iii). Position hold mode and
- (iv). Mission mode

4.4.1 Manual mode

When under manual control the roll and pitch sticks control the angle of the vehicle (attitude) around the respective axes, the yaw stick controls the rate of rotation above the horizontal plane, and the throttle controls altitude/speed. This mode was tested on the UAV for testing the transmitter accuracy and tuning the control.

4.4.2 Stabilized mode

During this flight control mode as soon as the control sticks released they will return to the centre dead zone. The multicopter will level out and stop once the roll and pitch sticks are centred. The vehicle will then hover in place/maintain altitude provided it is properly balanced, throttle is set appropriately, and no external forces are applied (e.g. wind). The craft will drift in the direction of any wind and you must control the throttle to hold altitude. This mode has been tested for flight controller stability and sensor calibration check.

4.4.3 Position hold mode

Position is an RC mode in which roll and pitch sticks control speed over ground in the left-right and forward-back directions (relative to the "front" of the vehicle), and throttle controls speed of ascent-descent. When the sticks are released/centred the vehicle will actively brake, level, and be locked to a position in 3D space compensating for wind and other forces. This position hold can be achieved with and without GPS lock. By using a PX4flow vision sensor the position hold has been achieved without GPS.

4.4.4 Mission mode

On the mission mode, the 3d waypoints are defined and the flight controller will follow the waypoints loaded through the ground control station or through the MAVROS node running on the companion computer.

4.4.5 Defining the waypoints on GCS

The waypoints can be loaded on to the flight controller through the ground control station shown in figure 4.5 and figure 4.6 or can be loaded on the MAVROS node running on the companion computer. The waypoints which is loaded through MAVROS node is defined on Appendix I. on alternate the waypoints can be created using the ground control station as mentioned as follows.

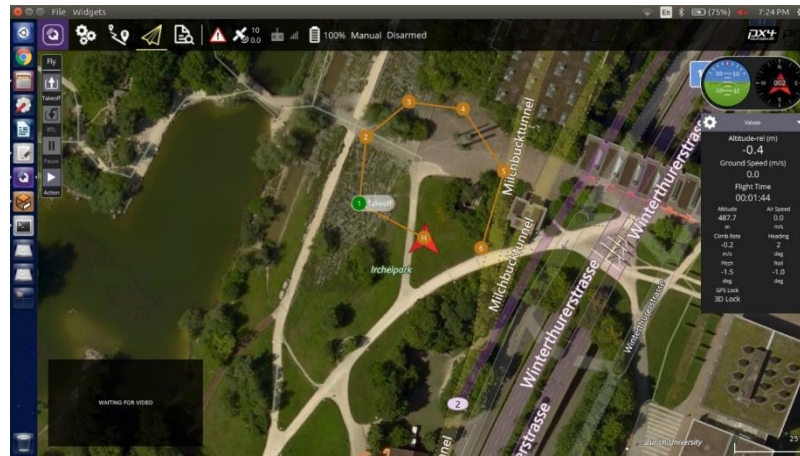


Figure 4.5 waypoints loaded on GCS from MAVROS node.

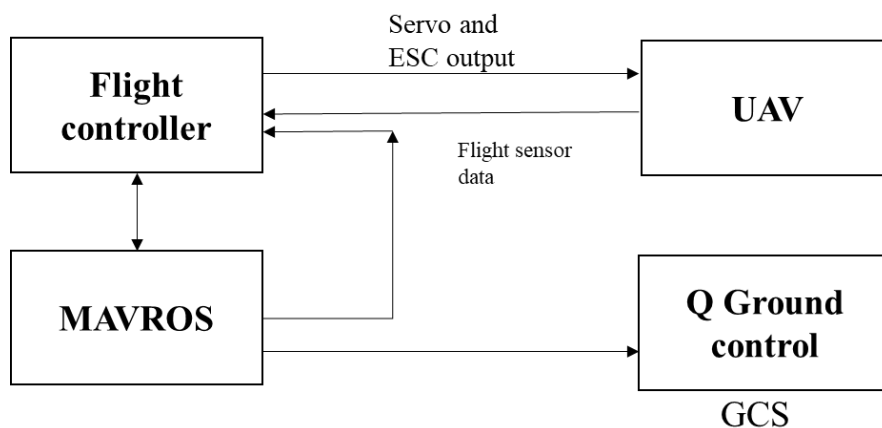
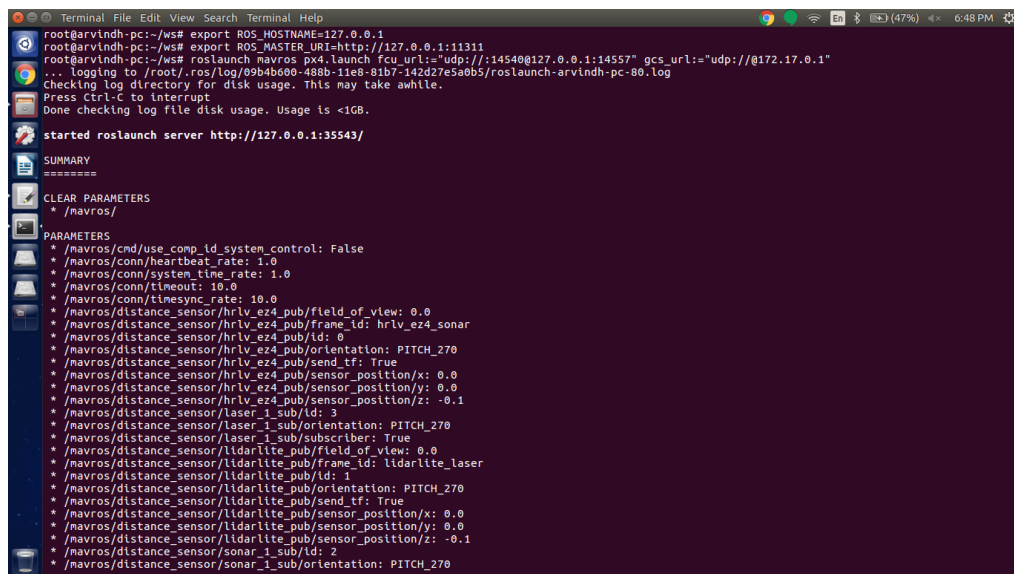


Figure 4.6 overview of initial flight testing

The defined waypoints should be loaded on the flight controller either by GCS or by MAVROS node shown in figure 4.7 and 4.8. After waypoints are defined, the mission will start from the given home position and hence the UAV will follow the given waypoints through 3D GPS lock positions. The fixed altitude is defined in each waypoint hence the UAV will hold those altitude and position and resist the change in position from the loaded waypoints affected by the environment.



```

root@arvindh-pc:~/ws# export ROS_HOSTNAME=127.0.0.1
root@arvindh-pc:~/ws# export ROS_MASTER_URI=http://127.0.0.1:11311
root@arvindh-pc:~/ws# roslaunch mavros px4.launch fcu_url:=udp://:14540@127.0.0.1:14557" gcs_url:=udp://:172.17.0.1"
... Logging to /root/.ros/log/09b4b600-488b-11e8-81b7-142d27e5a0b5/roslaunch-arvindh-pc-80.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://127.0.0.1:35543/

SUMMARY
=====
CLEAR PARAMETERS
* /mavros/

PARAMETERS
* /mavros/cmd/use_comp_id_system_control: False
* /mavros/conn/heartbeat_rate: 1.0
* /mavros/conn/system_time_rate: 1.0
* /mavros/conn/timeout: 10.0
* /mavros/conn/timesync_rate: 10.0
* /mavros/distance_sensor/hrlv_ez4_pub/field_of_view: 0.0
* /mavros/distance_sensor/hrlv_ez4_pub/frame_id: hrv_ez4_sonar
* /mavros/distance_sensor/hrlv_ez4_pub/id: 0
* /mavros/distance_sensor/hrlv_ez4_pub/orientation: PITCH_270
* /mavros/distance_sensor/hrlv_ez4_pub/send_tf: True
* /mavros/distance_sensor/hrlv_ez4_pub/sensor_position/x: 0.0
* /mavros/distance_sensor/hrlv_ez4_pub/sensor_position/y: 0.0
* /mavros/distance_sensor/hrlv_ez4_pub/sensor_position/z: -0.1
* /mavros/distance_sensor/laser_1_sub/id: 3
* /mavros/distance_sensor/laser_1_sub/orientation: PITCH_270
* /mavros/distance_sensor/laser_1_sub/subscriber: True
* /mavros/distance_sensor/lidarlite_pub/field_of_view: 0.0
* /mavros/distance_sensor/lidarlite_pub/frame_id: lidarlite_laser
* /mavros/distance_sensor/lidarlite_pub/id: 1
* /mavros/distance_sensor/lidarlite_pub/orientation: PITCH_270
* /mavros/distance_sensor/lidarlite_pub/send_tf: True
* /mavros/distance_sensor/lidarlite_pub/sensor_position/x: 0.0
* /mavros/distance_sensor/lidarlite_pub/sensor_position/y: 0.0
* /mavros/distance_sensor/lidarlite_pub/sensor_position/z: -0.1
* /mavros/distance_sensor/sonar_1_sub/id: 2
* /mavros/distance_sensor/sonar_1_sub/orientation: PITCH_270

```

Figure 4.7 MAVROS node loaded on companion computer

Figure 4.8 Initial flight testing done on CAIR campus, DRDO using waypoints (video feed from on-board camera)

CHAPTER 5

TRAINING AND TESTING OF DNN IN NVIDIA DIGITS

5.1 INTRODUCTION

A Deep Neural Network (DNN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of many highly-interconnected processing elements (neurons) working in unison to solve specific problems. DNNs, like people, learn by example. Learning in biological systems involves



adjustments to the synaptic connections that exist between the neurons.

5.1.1 ADVANTAGES OF DEEP NEURAL NETWORK

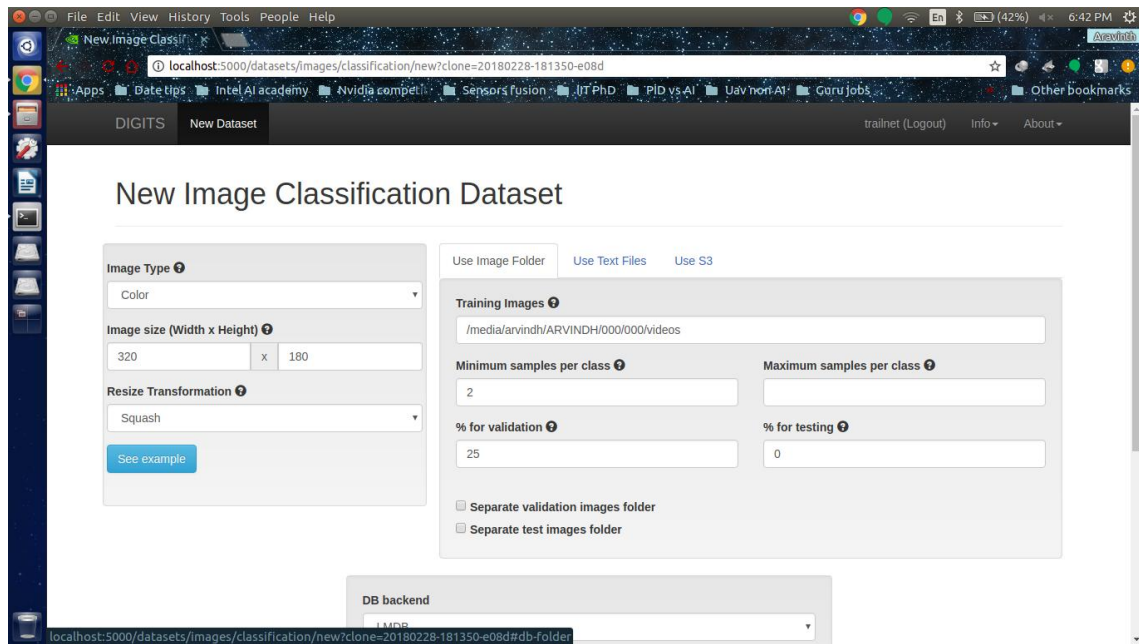
- (i). Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- (ii). Self-Organization: A DNN can create its own organization or representation of the information it receives during learning time.
- (iii). Real Time Operation: DNN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- (iv). Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

5.2 TRAINING OF DNN IN NVIDIA DIGITS

Training the DNN is done using the NVIDIA DIGITS, a GPU accelerated Deep Learning training tool. For training the trails on the DNN, it requires a huge dataset for better accuracy and prediction. Here the TrailNet DNN is defined as a classification problem rather than regression because of better prediction and clear output. The training has been carried out on a NVIDIA TITAN XP GPU accelerated Intel Xeon system with 16Gb of RAM memory. The training of the DNN has taken for 6 hours for reaching its accuracy of 93 percentage of training accuracy. As an initial part of training the DNN is dataset preparation which is explained as follows.

5.2.1 Dataset preparation

The dataset has been collected through a manual flight and through waypoints over the trail with the help of ground control station. A gimbal stabilized front facing GoPro camera is fixed under the UAV. the camera produces a high-resolution video of 4k, which must be converted into a 320 X 180 resolution of frames to input the images to DNN model for training. Hence the recorded video is converted into individual frames of desired resolution.



Since the DNN requires a huge dataset for better accuracy, along with the collected dataset an additionally available dataset of trails is used for improving the DNN accuracy. Hence the IDSIA trail dataset is used along with the collected dataset. For creating a dataset on NVIDIA DIGITS tool, initially a new dataset must have defined with the following parameters.

The resolution of the input image is converted from high resolution to 320x180 size for input of the DNN. The dataset collected from the manual flight and idsia dataset are to place in a directory to access for training. Here the dataset collected is separated as 75 percent for training and 25 percent for testing. The database backend is set LMDB method for training and the JPEG format is used for image formatting. Which is shown in figure 5.1

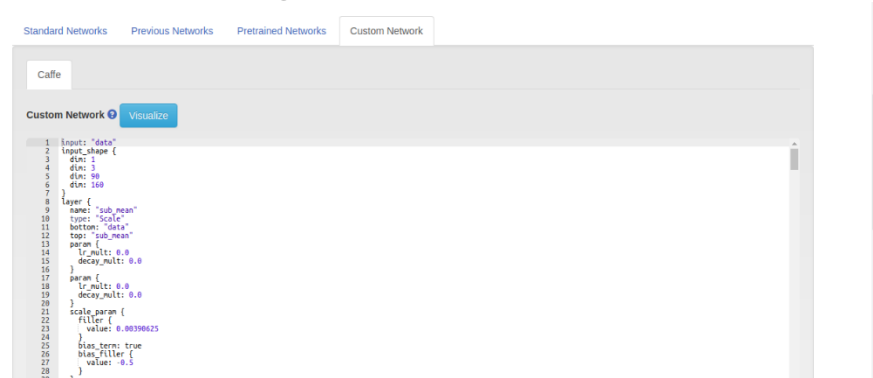
Figure 5.1 Preparing the dataset for training the DNN

5.2.2 Defining the DNN model

Once the dataset preparation is done, DNN model must be defined for training in DIGITS. For that it requires a new model definition which is defined in figure 5.2 as follows

Figure 5.2 Defining the DNN model on DIGITS

5.2.3 Training the DNN model



Once the model is defined and the prototext Caffe model file which has the DNN layers is defined and loaded on the model definition window. The Caffe file defines each layer of DNN model and can be visualized using the DIGITS tool. The defined prototext Caffe file is shown below in figure 5.3

New Image Classification Model

The screenshot shows the 'New Image Classification Model' configuration window with the following settings:

- Select Dataset:** A list of datasets is shown, with 'Train_001_002_004_007_009_Val_003_008' selected. Below the list, it says 'Done Feb 15, 11:02:00 PM'.
- Image Size:** 320x180
- Image Type:** COLOR
- DB backend:** Imdb
- Create DB (train):** 111363 images
- Create DB (val):** 4491 images
- Python Layers:**
 - Server-side file:** /data/trails/python-layers.py
 - ☐ Use client-side file
- Solver Options:**
 - Training epochs:** 20
 - Snapshot interval (in epochs):** 5.0
 - Validation interval (in epochs):** 2.0
 - Random seed:** [none]
 - Batch size:** 64 (multiples allowed)
 - Batch Accumulation:** (empty field)
 - Solver type:** Nesterov's accelerated gradient (NAG)
 - Base Learning Rate:** 0.001 (multiples allowed)
- Data Transformations:**
 - Subtract Mean:** None
 - Crop Size:** none

Figure 5.3 Caffe prototext file loaded in DIGITS tool

Once the model definition is done the DIGITS tool will load the datasets and starts the training. Training of each block of dataset has taken time depends on its size. Each dataset has different number of images. The overall training time of the model is six hours. Once the training is completed the DIGITS will display the training accuracy and learning rate on figure 5.4 as follows.

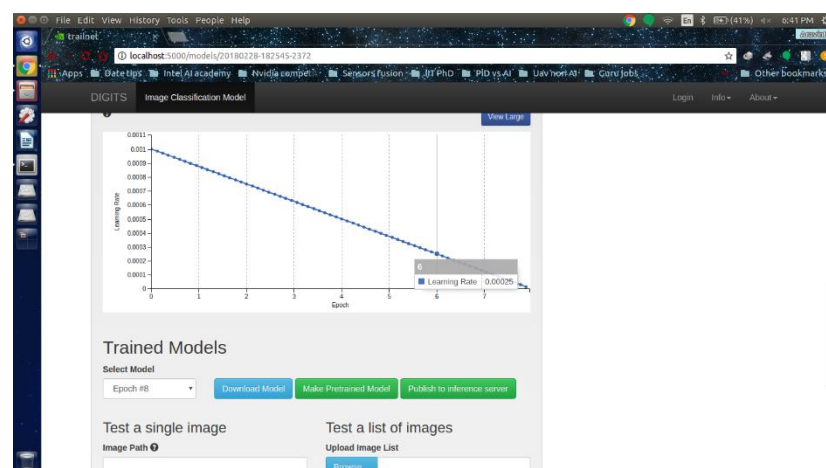


Figure 5.4 trained model and its learning rate

5.2.4 Testing of the model using DIGITS tool

Once the model is trained, it can be tested with a single image or multiple image inputs. using the test datasets for testing the accuracy of the model and its visualisations which is defined in figure 5.5

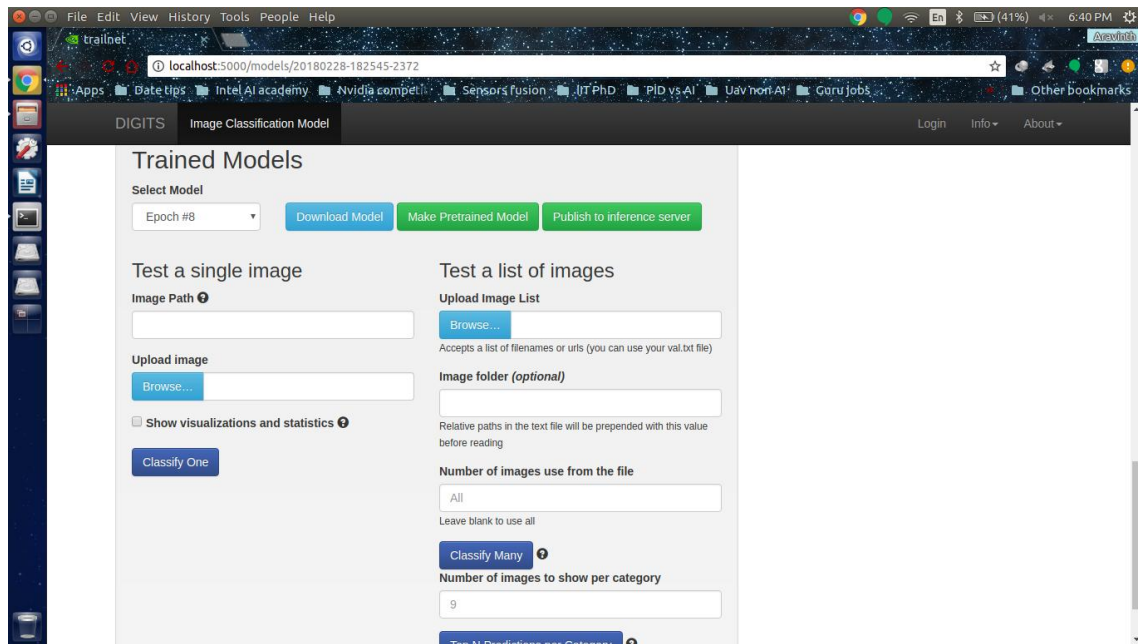
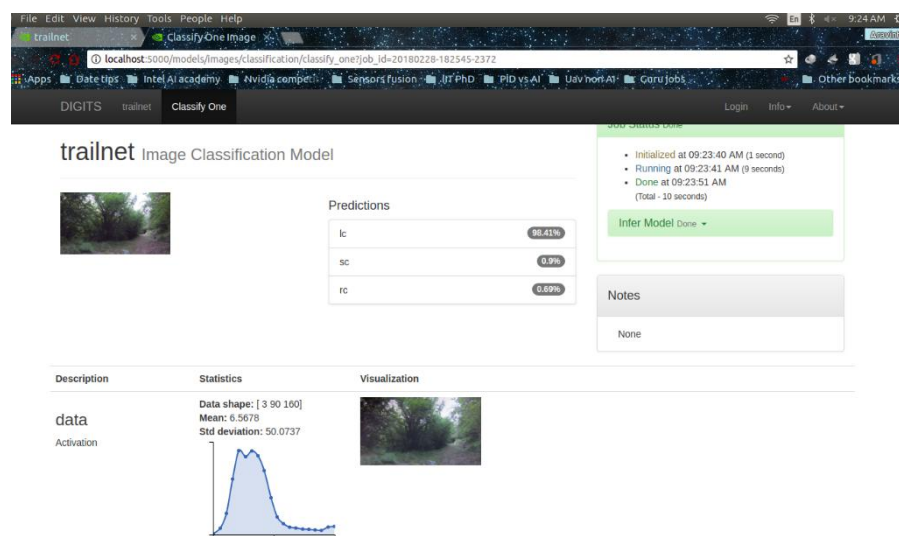


Figure 5.5 Testing the model using a single test image on DIGITS

the testing results in the following output, which classifies the given image as left, right or centre with percentage of predictions shown in figure 5.6

Figure 5.6 predictions of single image input for trained model



CHAPTER 6

SOFTWARE IN THE LOOP SIMULATION

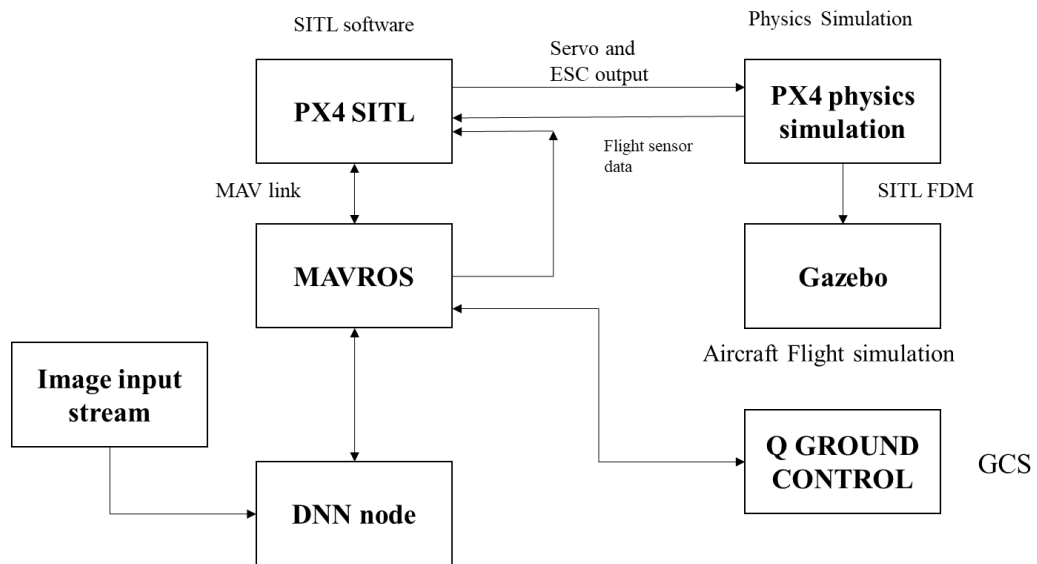
6.1 SITL ENVIRONMENT

The simulation environment can run the algorithm without any specific autopilot hardware. The software in the loop (SITL) simulation is a part of the autopilot code which gives a native executable to test the behaviour of the flight and the proposed solution. The advantages of this methodology are the early detection of bugs, design flaws, and the optimization of control software parameters. Thus, more mature designs are transferred to later design and test stages.

6.2 SOFTWARE ARCHITECTURE AND CONNECTION

The PX4 SITL is the main software, the software compiled and build is the same one that goes embedded in the UAV. The MAVROS is a portable console based app, that allows to simulate the radio control transmitter (RC Tx), supports the direct communication with the autopilot software using MAVLink, and can relay the real-time telemetry from the PX4 SITL to the ground control station (GCS), like a proxy.

The physics simulation software PX4 physics simulation is an open source flight dynamics model (FDM), fully configurable. The FDM is essentially the math model that defines the movement of an aircraft under the forces and moments applied to it using the various control mechanisms and from the forces of nature. As PX4 physics simulation does not have native graphical support, here Gazebo is used as a graphical viewer, a sophisticated and open flight simulator framework for use in research or academic environments, pilot training, and as an industry engineering tool, besides being a realistic desktop flight simulator. A Software In The Loop simulation model is defined in the Figure 6.1.

Figure 6.1 Software In-The-Loop Simulation Model.

6.3 PX4 SITL software

SITL can run PX4 on PC directly, without any special hardware. It takes advantage of the fact that PX4 is a portable autopilot that can run on a very wide variety of platforms. PC is just another platform that PX4 autopilot can be built and run on.

When running in SITL the sensor data comes from a flight dynamics model in a flight simulator. PX4 has a wide range of vehicle simulators built in, and can interface to several external simulators. by using the ros launch option the simulation environment for Software in-the-loop simulation will be started using a command. The PX4 physics simulation software will simulate all the required physics environment required for an UAV flight.

The MAVROS is a ROS node used for transferring control commands and sensor data between companion computer and flight controller, ground control station software which can be used for simulation of ground control and mapping the sensor data and Waypoint data of the UAV model. The below Figure 6.2 and

Figure 6.3 shows the PX4 simulation environment with gazebo and path of the UAV and sensor data respectively.

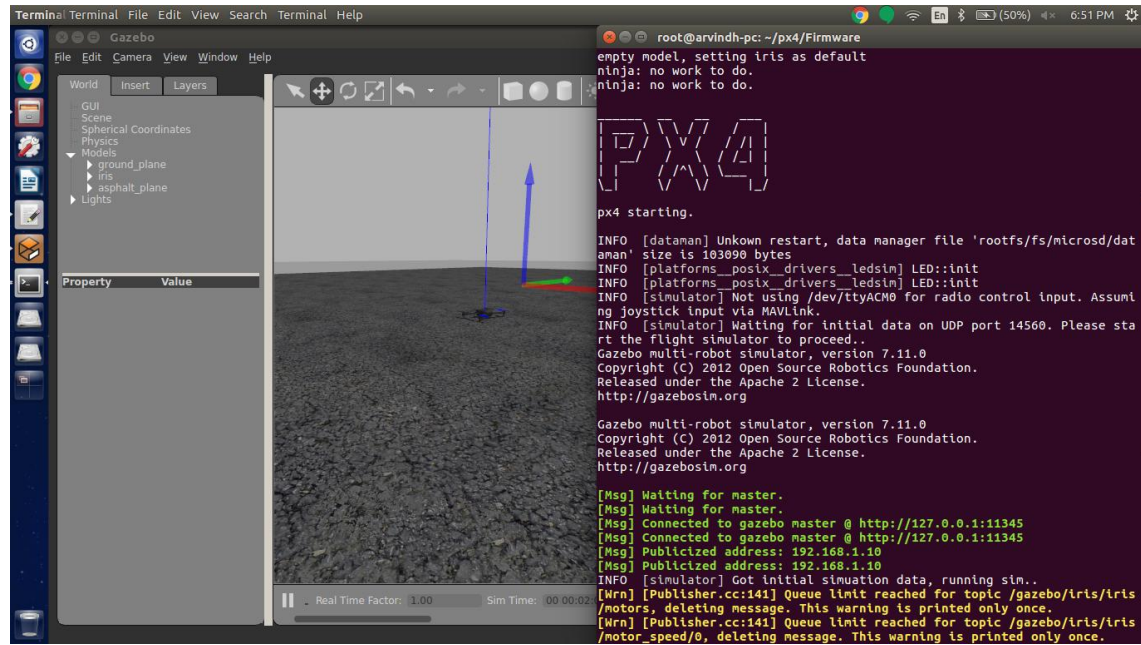


Figure 6.2 PX4 simulation environment with Gazebo

6.4 Running the DNN node on simulation environment

Once the PX4 simulation has started the gazebo window will open and the quadrotor UAV is loaded. After loading the modules, the MAVROS node must be started and the GCS is connected with a UDP link with the default url.

The DNN must be run in a virtual Docker environment. Each of the nodes such as PX4 node, MAVROS node, DNN node and Image publication node must be run in different Docker environments, so that each can access data from each other. Once all the modules are running on the virtual Docker containers. The DNN node has to be started separately on another Docker. The DNN node is started with the help of the Caffe_ros node. The Caffe_ros is a ROS node which converts the DNN predictions to the corresponding turning angles with the help of a python script. The python script for converting the DNN output to the corresponding angle is defined in the Appendix II. For deploying the DNN it requires a prototext file and

a caffe model binary file. The trained model of the DNN can be downloaded from the DIGITS tool after the training completion. After starting all the nodes, the simulation window is as follows in figure 6.3.

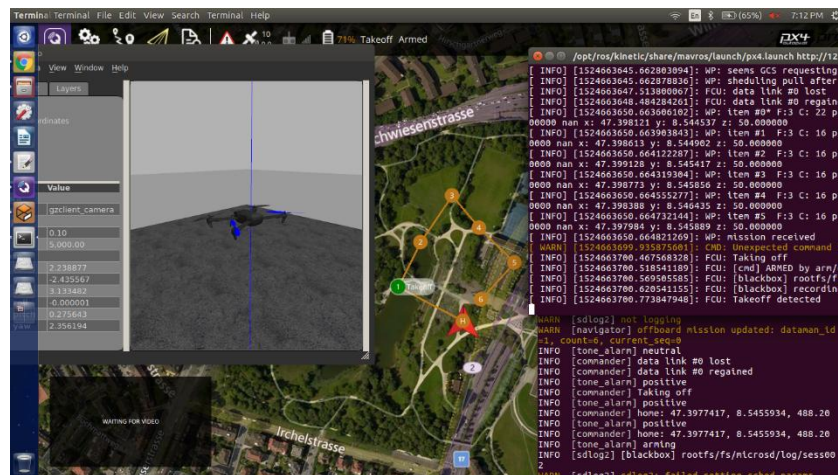


Figure 6.3 SITL of UAV with DNN node running under MAVROS

Once all the required nodes of ROS are started, the UAV can be armed using the commander take-off command. Then it will take off to an altitude of 2.5m and holds the position until it gets the command. Once the UAV is in position hold mode, the DNN node is started by giving a camera input or a saved video input as a DNN input. Then the UAV starts following the trails based on the given input from the video.

CHAPTER 7

RESULTS AND DISCUSSIONS

7.1 SIMULATION RESULTS

To test, the simulation environment will be configured to preliminary experimental tests. First, a flight mission with 7 points will be defined, using the ground control station. By adjusting this point, regardless of a system failure such as low battery, the UAV will activate the “fail safe” mode and will attempt to return to this point.

The “take off” point is the point that defines the route from the current position of the aircraft (Home point) to the defined point. Also, a desired altitude and a maximum pitch angle is set for an abrupt or softer take-off. The Waypoints has been loaded through MAVROS node command prompt and data has been logged in a log file.

Then the DNN node has been started and the mode switched to autonomous mode. Using the feed from a camera or image publication ROS node is used to transfer the recorded video to the DNN input. And the DNN is loaded into the simulation environment. Once it is loaded the `caffe_ros` node converts the DNN output to desired telemetry signal. The telemetry signal output from the DNN is similar to that of the signal from the RC transmitter. The DNN follows the trail based on the given input from the video feed and thus the output is shown in the following figure 7.1.

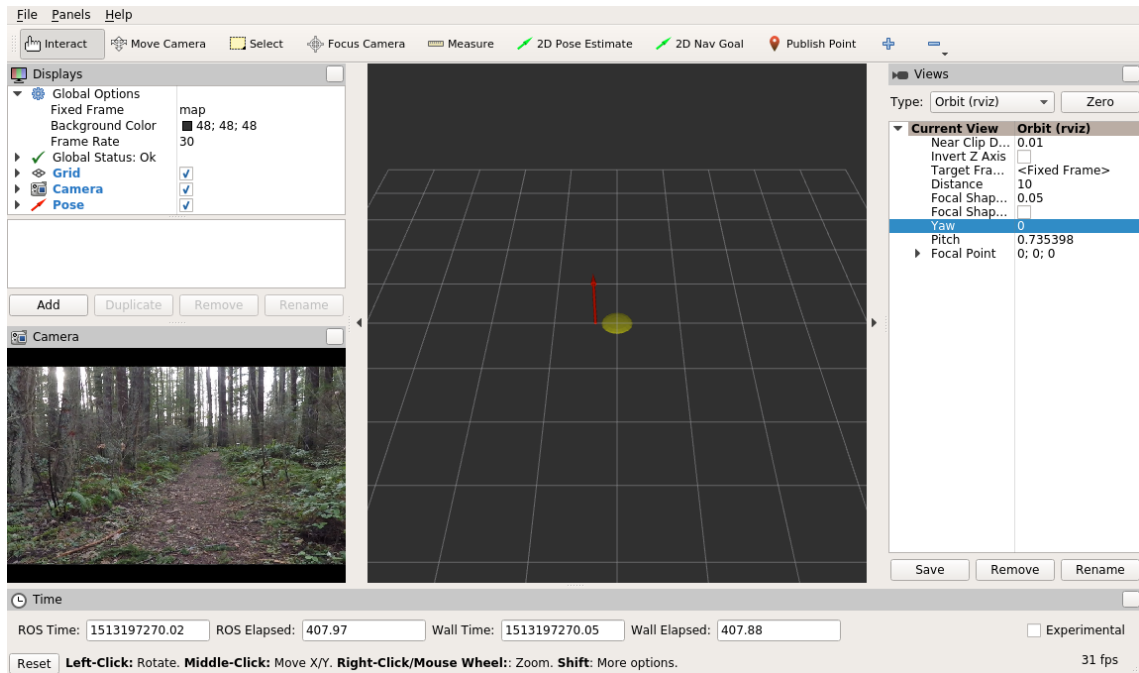


Figure 7.1 Given input video stream on DNN for SITL

7.2 FLIGHT LOG DATA ANALYSIS

The sensor data acquired from the manual flight of the model has been plotted as graph in Figure 7.1, Figure 7.2 as follows. During the test flight the UAV has crashed three times due to errors in the sensor calibration.

The flight controller has three magnetometer sensors one is a primary mag which is mounted on the GPS receiver and it has less interference with the environment. In addition to the two secondary mag sensors which should be disabled for a stable flight and In-consistency errors due to interference with the metal objects and other sensors. The sensor crash data has been logged and shown in the figure 7.2 and 7.3.





Figure 7.2 Mag sensor inconsistency log data for yaw angular rate

Figure 7.3 Mag sensor inconsistency for yaw angle

It is seen that from the above figure 7.2 and 7.3 the mag sensor inconsistency causes the UAV to continuously yaw on a direction. The data has been analysed and found that the error has been caused due to the interference of the secondary mag sensor, which makes the large angle difference with primary mag.

7.3 Off-board testing of UAV for Autonomous navigation using DNN

For testing the UAV in real time for autonomous operations, it requires an on-board companion computer. The Jetson TX1 has been mounted on the UAV and tested for initial flight. The flight's thrust was not enough to take-off with the Jetson TX1. Hence it has been tested on off-board by connecting the Jetson TX1 wirelessly using MAVLink 2 through Wi-Fi.

Where the Jetson TX1 is mounted on a ground robot which is controlled manually to maintain the proximity and the flight controller is connected with the off-board TX1 with MAVLink protocol using MAVROS node. The UAV followed the trail based on the input from the camera stream mounted on-board.

CHAPTER 8

CONCLUSION

This project proposed a new adaptive algorithm to ensure an optimal adjustment in trails followed by the UAV throughout the mission for its autonomous navigation without a GPS. The use of artificial intelligence techniques, such as Deep Neural Network (DNN), proved to be very efficient in searching for optimized parameters, even without the supervision of a human. The simulation results have shown an optimal performance of the DNN on unknown trails. And the real-time off-board testing of UAV for its autonomous navigation on the trails has shown that the off-board navigation has some delays in processing due to communication limits and vehicle has crashed three times due to sensor errors on yaw before achieving the stable flight.

8.1 FUTURE WORK

As a part of the future work, the on-board testing will be carried out for efficient results of the DNN predictions. A new high load carrying frame of hexa-copter (DJI F550) was assembled to take up the load of Jetson TX1 on-board for efficient processing. The initial flight testing with on-board Jetson TX1 has been done, the DNN control has to be tested in the upcoming weeks for its stable flight. A new flight controller called pixhawk cube will be used for reducing the sensor errors causing the UAV to yaw.

APPENDIX I

Waypoint data for collecting the sensor data from SITL simulation for training the Neural Network Model.

waypoint no.	x	y	z	pitch	altitude	speed
0	1	0	16	0	149.165085	584.409973
1	0	3	22	15	149.163986	28.110001
2	0	3	16	0	149.161697	99.800003
3	0	3	16	0	149.162659	100.730003
4	0	3	16	0	149.164581	100
5	0	3	16	0	149.163757	100
6	0	3	177	-2	0	0
7	0	3	16	0	149.163757	100

APPENDIX II

The python script used for converting the DNN output of view orientation and lateral offset to flight control signals of turning angle and waypoints is given below.

```
#include "px4_controller/px4_controller.h"

#include <boost/algorithm/string.hpp>

#include <mavros_msgs/ParamGet.h>

#include <mavros_msgs/ParamSet.h>

void PX4Controller::computeDNNControl(const float class_probabilities[6],
float& linear_control_val, float& angular_control_val)

{ float prob_sum = class_probabilities[0] + class_probabilities[1] +
class_probabilities[2];

    assert(prob_sum!=0);float left_view_p    = class_probabilities[0] /
prob_sum;

    float right_view_p    = class_probabilities[2] / prob_sum;

    prob_sum = class_probabilities[3] + class_probabilities[4] +
class_probabilities[5]; assert(prob_sum!=0);

    float left_side_p    = class_probabilities[3] / prob_sum;

    float right_side_p    = class_probabilities[5] / prob_sum;

    float current_turn_angle_deg =  dnn_turn_angle_*(right_view_p -
left_view_p) + dnn_lateralcorr_angle_*(right_side_p - left_side_p);

    current_turn_angle_deg = std::max(-90.0f, std::min(current_turn_angle_deg,
90.0f));float current_turn_angle_rad =
(float)angles::from_degrees((float)current_turn_angle_deg);

    turn_angle_ = turn_angle_*(1-direction_filter_innov_coeff_) +
current_turn_angle_rad*direction_filter_innov_coeff_;

    float turn_angle_rad = turn_angle_;
```

```

ROS_INFO("DNN turn angle: %4.2f deg.",
(float)angles::to_degrees(turn_angle_rad));

linear_control_val  = cosf(turn_angle_rad);

angular_control_val = sinf(turn_angle_rad);}

```

REFERENCES

- [1] De Oliveira H A and Rosa P F, 'Adaptive genetic neuro-fuzzy attitude control for a fixed wing UAV,' *Proc. IEEE Int. Conf. Ind. Technol.*, pp. 726–731, 2017.
- [2] Verbickas R, 'Convolutional Neural Network-based Vision Systems for Unmanned Aerial Vehicles,' *ProQuest Diss. Theses*, vol. MR94243, no. December, p. 138, 2013.
- [3] Konam S, Rosenthal, and Scherer, 'Vision-Based Navigation and Deep-Learning Explanation for Autonomy,' no. May, 2017.
- [4] Lee J, Wang J, Crandall, Sabanovic S, and Fox, 'Real-Time Object Detection for Unmanned Aerial Vehicles based on Cloud-based Convolutional Neural Networks,' 2015.
- [5] Sathyan A, Ernest N, Lavigne L, Cazaurang F, Kumar M, and Cohen K, 'A Genetic Fuzzy Logic Based Approach to Solving the Aircraft Conflict Resolution Problem,' *AIAA Inf. Syst. Infotech @ Aerosp.*, no. January, pp. 1–8, 2017.
- [6] Julian K D and Kochenderfer M J, 'Neural Network Guidance for UAVs,' *AIAA Guid. Navig. Control Conf.*, no. January, pp. 1–14, 2017.
- [7] Dujrene W R, 'Application of artificial intelligence techniques in uninhabited aerial vehicle flight,' *Digit. Avion. Syst. Conf.*, p. 8.C.3-8.1-6 vol.2, 2003.
- [8] Atmeh G and Subbarao K, 'A dynamic neural network with feedback for

trajectory generation,' *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 367–372, 2016.

- [9] Schmidhuber J, 'Deep Learning in neural networks: An overview,' *Neural Networks*, vol. 61, pp. 85–117, 2015.
- [10] Muliadi J and Kusumoputro B, 'Neural Network Control System of UAV Altitude Dynamics and Its Comparison with the PID Control System,' *J. Adv. Transp.*, 2017.
- [11] Rimal B P, Shin H, and Choi E, 'Simulation of Nonlinear Identification and Control of Unmanned Aerial Vehicle : An Artificial Neural Network Approach,' *Commun. Inf. Technol. 2009. Isc. 2009. 9th Int. Symp.*, pp. 442–447, 2009.
- [12] Maravall D, de Lope J, and Pablo Fuentes J, 'Vision-based anticipatory controller for the autonomous navigation of an UAV using artificial neural networks,' *Neurocomputing*, vol. 151, no. P1, pp. 101–107, 2015.
- [13] Selma B and Chouraqui S, 'Neuro-fuzzy controller to navigate an unmanned vehicle', *Springerplus*, vol. 2, p. 188, 2013.
- [14] Canciello G, Cavallo A, Amato E D, and Mattei M, 'Attitude and Velocity High-gain Control of a Tilt-rotor UAV', 2017.
- [15] Chang-Su Park, Hyunjae Lee, Min-Jea Tahk, and Hyochoong Bang, 'Airship control using neural network augmented model inversion,' *Proc. 2003 IEEE Conf. Control Appl. 2003. CCA 2003.*, no. 2, pp. 558–563, 2003.
- [16] Smolyanskiy N, Kamenev A, Smith J, and Birchfield S, 'Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness,' *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2017–September, pp. 4241–4247, 2017.