PROJECT REPORT

Entitled

"AUTONOMOUS AGRICULTURAL ROBOT (AGRIBOT)"

Submitted to the Department of Electronics Engineering
In Partial Fulfillment of the Requirement for the Degree of

Bachelor of Technology
(ELECTRONICS & COMMUNICATION)

: Presented & Submitted By :

Mr. PATEL DHRUV RAJENDRAKUMAR
(Roll No. U16EC053)
Mr. GANDHI MEET JAYENDRAKUMAR
(Roll No. U16EC056)
Mr. SHANKARANARAYANAN H
(Roll No. U16EC074)
B. TECH. IV (EC), 8th Semester

: Guided By :

Dr. A. D. DARJI
Associate Professor & Head, ECED.

(Year : 2019-20)

DEPARTMENT OF ELECTRONICS ENGINEERING
Sardar Vallabhbhai National Institute of Technology
Surat-395007, Gujarat, INDIA.

Sardar Vallabhbhai National Institute of Technology

Surat-395 007, Gujarat, INDIA.

ELECTRONICS ENGINEERING DEPARTMENT

CERTIFICATE

This is to certify that the **PROJECT REPORT** entitled **"AUTONOMOUS AGRICULTURAL ROBOT (AGRIBOT)"** is presented & submitted by Candidates **Mr. PATEL DHRUV RAJENDRAKUMAR, Mr. GANDHI MEET JAYENDRAKUMAR** and **Mr. SHANKARANARAYANAN H**, bearing **Roll Nos. U16EC053, U16EC056** and **U16EC074**, **of B.Tech. IV, 8th Semester** in the partial fulfillment of the requirement for the award of **B. Tech.** degree in **Electronics & Communication Engineering** for academic year 2019-20.

They have successfully and satisfactorily completed their **Project Exam** in all respect. We, certify that the work is comprehensive, complete and fit for evaluation.

Dr. Anand D. Darji
Associate Professor, Head, ECED,
Project Guide

PROJECT EXAMINERS:

Name of Examiner	**Signature with date**
1. **Dr. Z. M. PATEL**	_____
2.**Dr. P. J. ENGINEER**	_____
3.**Dr. R. N. DHAVSE**	_____

Dr. A. D. Darji
Associate Professor &
Head, ECED, SVNIT.

DEPARTMENT SEAL
(June - 2020)

ACKNOWLEDGEMENT

The success of this work required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of the seminar. We express our sincere gratitude to our guide Dr. A. D. Darji, without whom the work would not have been successfully completed. His encouragement proved very useful to us in every stage. He also guided us in the best way and motivated in each and every aspect for this project. His guidance paved the way for the successful completion of this project. We heartily thank him for his exemplary guidance, constant encouragement, and careful monitoring throughout the project.

We would also like to extend our gratitude to the Head of the Department Dr. A. D. Darji and all the faculty members of the Electronics Department for boosting the students' confidence levels and enlightening them with practical real-life situations with academic guidance and organization of projects.

We would also like to extend our sincere gratitude to Mr. Mahesh Birajdar (U16ME084) from the Mechanical Engineering for providing his valuable guidance and assistance in the development of the mechanical model of the Robot.

We would also like to extend our sincere gratitude to the TEQIP III team for understanding this project and providing the required funding for the same.

It's a great pleasure for us to present our project topic on "Autonomous Agricultural Robot (AGRIBOT)" which is trending and useful for serving futuristic applications. We mark our gratitude for everyone who helped and encouraged us with their efforts.

Dhruv Patel	Meet Gandhi	Shankaranarayanan H
(Roll No.: U16EC053)	(Roll No.: U16EC056)	(Roll No.: U16EC074)

ABSTRACT

Agriculture has always remained as an integral part of India. In fact, for the past few decades it has served as a major backbone for the Indian Economy. As the human population keeps on rising, the demand for the food also increases and so is the dependency on the farming industry. But the present scenario tells us otherwise. Every day we can see or hear the news of farmers committing suicide because of low yield, less rainfall, etc. Apart from this, there is a dearth of manpower created in this farming sector because people are moving to live in the cities and villages are becoming more and more urbanized. So, the farming sector which was once a backbone of our economy is now contributing only 17% to the Indian GDP (Gross Domestic Product).

On the other hand, the field of robotics has seen a tremendous development in the past few years. In the present scenario, new concepts like Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning are being incorporated with robotics to create autonomous systems for various applications like driving, farming, assembly line management, etc. Deploying such autonomous systems in the farming sector helps in many aspects like reducing manpower, better yield and nutritional quality of crops. So, in our project we are planning to design an autonomous agricultural robot which primarily focuses on weed detection.

The primary objective of this project is to design an autonomous agricultural robot specifically used for the detection of weed on the real-time basis without any human involvement. This will help to offer better and nutrients rich yield involving less man-power than conventional agriculture. This project can also be extended to design robot in various other applications involved in farming like weed removal, ploughing, harvesting, etc. in turn making farming industry more efficient.

The initial step is the system design of the robot which includes designing Robotic structure, embedded subsystems and Computer Vision approaches for Crop Weed classification. For designing of 3D mechanical model, a case study of MARIO(Mobile Autonomous Robot for Intelligent Operations) has been considered and the 3D model is simulated and developed

based on our design constraints. Then the next step is to interface various hardware and sensors in the robot. NVIDIA Jetson Nano is the primary processor and Arduino Mega is the secondary controller. All the sensors such as NEO-M8N (GPS Sensor), MPU-9265 (IMU), Raspberry V2 Camera have been interfaced with the Jetson board which acts like master controller while Arduino Mega being slave controller accepts commands from Jetson and drives the motors. Also, various computer vision approaches have been discussed for Crop Weed classification.

The next step involves the software and algorithms for testing the functionality of AGRIBOT. Data from various sensors is acquired and algorithms such as filtering the noisy data, sensor fusion etc. have been implemented for trajectory planning and autonomous navigation of the robot. Various Machine Vision Models are developed based on CNN (Convolutional Neural Network) which acts as robust feature extractors for our application of weed detection. The performance of these models has been tested on different datasets and accordingly the final Machine Vision model is selected. Our classification model Bonnet achieves a mean accuracy of 99.47 %, a mean iou of 98.03% and loss of 0.00348 units on the Bonn Dataset. The camera feed obtained during traversing the field is fed into this model and model predicts and classifies the crop and weed on real-time basis. Our model has an average latency of 2.5 fps on i7 + NVIDIA 940 MX. This makes it possible to deploy the model on an on-board processor for real-world application.

Because of the COVID-19 Pandemic, the further development in the hardware side wasn’t feasible so the robotic model which was developed, was visualized and tested in a simulated environment with the help of Gazebo. Software and algorithms tested on simulation can be deployed on an on-field robot on the basis of the results derived from this project which would be helpful for reviving the dwindling agricultural industry.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION CHAPTER-1

The term “Agriculture” is quite synonymous with our country India. The history of Agriculture in India dates back to the Indus Valley Civilization and even before that in some places of Southern India. In fact, it was in the Indus Valley Civilization where the system of irrigation was developed. The size and prosperity of the Indus civilization grew as a result of this innovation, which eventually led to more planned settlements making use of drainage and sewers. Sophisticated irrigation and water storage systems were developed by the Indus Valley Civilization, including artificial reservoirs at Girnar dated to 3000 BCE, and an early canal irrigation system from circa 2600 BCE. Archaeological evidence of an animal-drawn plough dates back to 2500 BC in the Indus Valley Civilization.

After India became independent from the British rule, Five-Year plans were carried out by the Planning Commission which focused more on food and cash crops supply. Land reclamation, land development, mechanization, electrification, use of chemicals fertilizers in particular, and development of agriculture oriented 'package approach' of taking a set of actions instead of promoting single aspect soon followed under government supervision. The Green revolution started in the late 1960s and addressed the problems of frequent famines, lack of self-sufficiency and lack of finances. In fact, agricultural sector is still one of the backbones of our Indian economy by contributing to about 17% to the Indian GDP.

But the present scenario is much different. Every day we can see or hear the news of farmers committing suicide because of low yield, less rainfall, etc. Apart from this, there is a dearth of manpower created in this farming sector because people are moving to live in the cities and villages are becoming more and more urbanized. At the current growth rate of the world population, it is really important to address the issue of the dwindling agricultural sector.

The field Robotics on the other hand evolved exponentially during the past few decades. Robotics deals with the design, construction, operation and use of robots, as well as computer systems for their control, sensory feedback and information processing. The main area of application of robots in agriculture today is at the harvesting stage. Emerging applications

of robots or drones in agriculture include weed control, cloud seeding, planting seeds, harvesting, environmental monitoring and soil analysis. According to a Verified Market Research, the agricultural robots market is expected to reach $11.58 billion by 2025.

1.1 Introduction to Agricultural Robot

An agricultural robot is a robot deployed for agricultural purposes. The application of our prime focus will be the weed monitoring through autonomously and tele-operation of robot. For designing such a robot, we will be using the ROS frame work and Deep learning and Machine Vision algorithms for crop-weed classification.

1.1.1 ROS for Agriculture

ROS stands for Robotic Operating System. It is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. To incorporate ROS in agriculture automation, it can be achieved making existing agriculture machines ROS-compatible, and/or by introducing ROS-compatible robots into agriculture. Making an agriculture machine ROS-compatible involves the following steps; developing its description model, ROS sensors drivers, and ROS controller interfaces. Also, it involves creating custom ROS packages to allow the robot to perform the required agriculture tasks. Introducing ROS compatible robots into agriculture involves a similar process; to modify the robot description model, to update sensor drivers and controller interfaces, and to create custom ROS packages to allow the robot to perform the needed agriculture tasks.
Once completed, then these ROS-compatible machines can utilize the features of ROS and other open-source technologies and features; such as software for navigation and motion planning, networking over vast open fields, HRI with agriculture workers, visualization and monitoring tools, and much more.

1.1.2 Deep Learning and Computer Vision for Agriculture

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. It is also known as deep

neural learning or deep neural network. Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding.

To extract meaningful phenotyping information from large-scale image datasets, a variety of computer vision, deep learning (DL) approaches have been utilized. In recent years, much attention has been paid to DL techniques, based on which computational algorithms and learning models were built to accomplish tasks such as vision-based feature selection, image object classification, and pattern prediction. With adequate training data, suitable learning algorithms, and well-defined predictive outcomes, the integration of computer vision, DL, and newly emerged analytic solutions (e.g., distributed computing) is leading to a step change for crop phenotyping research.

1.1.3 Embedded System Design for Agriculture

An embedded system is a combination of computer hardware and software, fixed in capability or programmable, designed for a specific function or functions within a larger system. Embedded systems are computing systems, but they can range from having no user interface (UI) -- for example, on devices in which the system is designed to perform a single task -- to complex graphical user interfaces (GUIs), such as in mobile devices. A typical embedded system design for an agriculture robot will look something like this: -

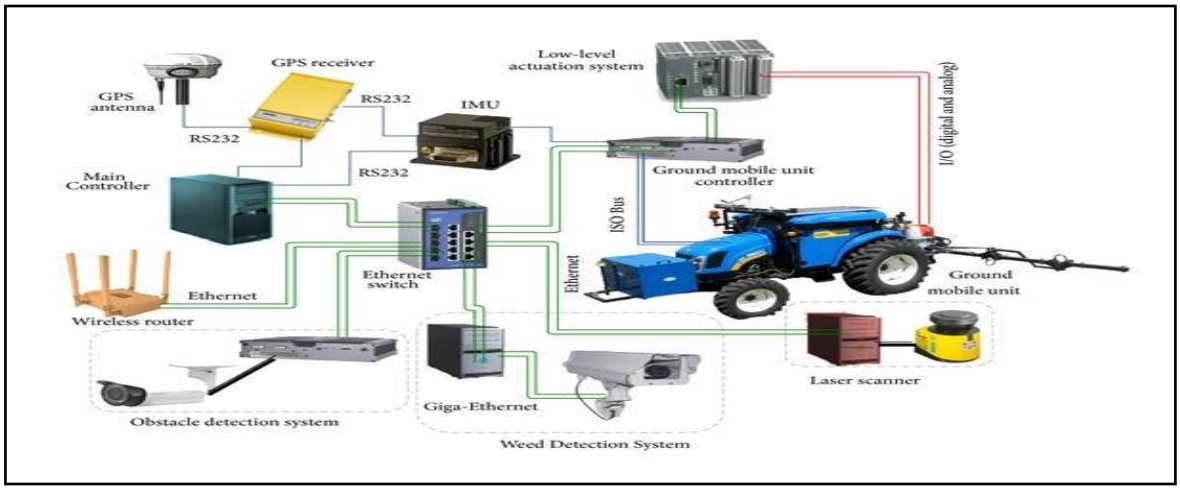


Fig. 1.1 Components of RHEA Project [16].

1.2 Recent Advances in Making Agriculture Autonomous.

The autonomous robots which are presently being used in agricultural are classified into 4 types on the basis of their applications. They are: -

1) Harvester
2) Weed Control
3) Seed Spraying
4) Crop Phenotyping

1.2.1 Harvester

Harvesting and picking is one of the most popular robotic applications in agriculture due to the accuracy and speed that robots can achieve to improve the size of yields and reduce waste from crops being left in the field.

Fig. 1.2(a) Strawberry Harvester designed by AGROBOT [15].

1.2.2 Weed Controller

Weeds are the unwanted plants that interfere with the use of land and water resources and therefore have an adverse effect on agriculture. Weeds compete in croplands with the beneficial and desired vegetation and pose a major problem. The losses caused by weeds are

greater than the losses caused by any other agricultural pest category. Weeds account for 45% of the total annual loss in agricultural production.

Fig. 1.2(b): RIPPA Robot developed by University of Sydney for weed removal [12].

1.2.3 Seed and Pesticide Spraying

With the help of both ground robots and UAVs it is possible to spray seeds and the required amount of pesticides for the plants. The seed spraying technique is also used for afforestation and it has been

1.2.4 Crop Phenotyping

Phenotyping, in general, is a process by which different traits of a plant, such as nitrogen consumption, yield, and dimensions, are assessed. Knowledge regarding these traits and the way by which they are affected by different factors are key to agricultural progress. For its importance, phenotyping has been performed by farmers for hundreds of generations. Until recent years, phenotyping strictly relied on manual measurements and “farmer intuition”. Nowadays, technological advances enable automatic, precise, high-throughput measurements as well as an exact analysis of the traits and factors that affect them, opening the door to a new age in agriculture.

1.3 Objective

The human population is growing day by day and so the demand for food keeps on increasing. The agriculture in India is currently dwindling because of poor farming techniques and agricultural practices, inadequate irrigation facilities, instability in the agricultural prices, agricultural indebtedness to name a few. Every day in newspaper there is at least 1 news which state about farmer committing suicide. Apart from this, presently the GDP growth rate of the Country has attained new lows and agriculture being one of the backbones of the Indian economy, also contributed to this slump.

So, in order to address these issues this project is being developed. As it is an autonomous system, the man-power required will be very much less as compared to the previous case, so it provides a cost-effective solution to the farmers. The primary objective of this project is to design an autonomous agricultural robot specifically used for the monitoring of weed on the real-time basis without any human involvement. By monitoring the weeds, the farmers are being ensured that whether the enough amount of nutrients and water are available to the plants or not on the daily basis and after that process can be done through less human resource.

So, the yield quantity and quality both increases which in-fact is beneficial to the consumer. Because of this the profit margin of the farmers also increases and less suicide cases would be reported in the newspapers. And finally, for the current economic situation prevailing in the country, if this kind of boost is given to the agriculture then the economic growth rate increases and GDP also increases which in-turn is beneficial for the whole country.

At the starting of the academic year 2019-2020, we proposed the design and deployment of Farm Robot which is able to autonomously navigate and perform weed control. However due to COVID-19 pandemic, we have shifted our approach to simulate the robot design and algorithms in an open source simulator. Through this project we have simulated an agricultural robot called as “AGRIBOT”. AGRIBOT - Autonomous Agricultural Robot is a four-wheel skid steering prototype that has been designed to use in the agricultural environment for automation of different tasks such as monitoring and classification between crop and weed. AGRIBOT is equipped with on-board sensors including GPS, IMU,

Compass and Camera. This report represents the approach for intelligent classification and automation with analysis of AGRIBOT through modelling, simulation and experimenting proposed algorithms using open source tools.

Chapter 2 provides the brief description about of the present scenario in the field of Robotics in Precision Agriculture and Computer Vision & Deep Learning practices. BoniRob and Mario Robots have been taken into consideration. Chapter 3 includes the system design which is further divided into hardware and software systems of AGRIBOT. Chapter 4 contains algorithms applied for analysis of AGRIBOT in simulated as well as real world cases. Chapter 5 reports the experimental results and last section sums up with conclusion of the project along with aspects which can be improvised for the future needs.

*****

LITERATURE SURVEY CHAPTER-2

Traditional Weed Management required methods like burying, cutting or uprooting. Several tools like finger weeders, brush weeders etc. were used which performed the above-mentioned operations. But these methods required human operation making it less effective and time taking as it caused damage to crops in case of intra-crop weeding [4]. This led to the demand of robotics and automation in the field of agriculture. Various Robots have been developed which navigated autonomously and performed various tasks like ploughing, crop monitoring, seeding through actuations in real time. The recent advances in the field of Modelling and Simulation, Sensor fusion and crop-weed classification with the help of computer vision and deep learning and how they provided the base for the development of AGRIBOT is thoroughly discussed in this section.

2.1 Modelling and Simulation

Nowadays, modelling & simulation are the major parts of scientific and engineering processes, especially in industries wherein heavy amount of resource might get wasted due to faulty systems. The same thing applies for the robotic industries in agriculture. Modelling and simulation play an important role in the performance analysis and in the development of advanced control algorithms for robotic systems. Design, testing, and validation of robotic systems ranging from indoor mobile robots to outdoor mobile robots, articulated industrial manipulators, underwater robotic systems, and humanoid robots would have never existed without proper modelling and simulation tools. Furthermore, simulation can be used as a tool to develop virtual environments for training operators as well as an educational tool for teaching and learning the basic concepts of robotic systems. Additionally, simulation provides low cost means of testing and experimentation, and makes controlling disturbances much easier compared with using real robotic systems [17].

Along with the advancement in powerful and affordable computing technologies in the last two decades, numerous proprietary and open source robotic modelling and simulation software have been developed for robotics application. Examples of such software are Open

Dynamics Engine (ODE), Robotic Toolbox for MATLAB, Microsoft Robotics Developer Studio (MRDS), Webots, Virtual Robot Experimentation Platform V-Rep and Gazebo. In the following table a comparison of different software has been provided [18].

Table 2.1 Comparison between general specifications of the selected simulation software for agricultural robotics [18].

Table 1 Comparison between general specifications of the selected simulation software for agricultural robotics

Simulation software	Developer	Physics engine	Supported operating systems	Prog language	CAD files support	API support	ROS support
Webots	Cyberbotics Ltd	Proprietary based on ODE	Linux, Mac OS, Windows	C++	WBT, VRML, X3D	C, C++, Python, Java, Matlab, ROS	Yes
Gazebo	Open Source Robotics Foundation	ODE, Bullet, Simbody, DART	Linux	C++	SDF/URDF, OBJ, STL, Collada	C++	Yes
Actin	Energid Technologies	Proprietary	Windows, Mac OS, Linux, VxWorks, and RTOS-32. (RTX and QNX Planned)	C++	SLDPRT, SLDASM, STEP, OBJ, STL, 3DS, Collada, VRML, URDF, XML, ECD, ECP, ECW, ECX, ECZ,	Not known	Yes
RoboDK	RoboDK	None	Linux, macOS, Windows, Android	Python	STEP, IGES, STL, WRML	C/C++, Python, Matlab	No
Morse	Academic community	Bullet	Linux, BSD*, Mac OS	Python	Unknown	Python	Yes
OpenRAVE	OpenRAVE Community	ODE, Bullet	Linux, Mac OS, Windows	C++, Python	XML, VRML, OBJ, Collada	C/C++, Python, Matlab	Yes
OpenHRP3	AIST	ODE, Internal	Linux, Windows	C++	VRML	C/C++, Python, Java	No
ARGoS	Swarmanoid project	Multiple-physics engines	Linux and Mac OSX	C++	Does not support	C++	Yes
V-REP	Coppelia Robotics	ODE, Bullet, Vortex, Newton	Linux, Mac OS, Windows	LUA	OBJ, STL, DXF, 3DS, Collada,URDF	C/C++, Python, Java, Urbi, Matlab/Octave	Yes

Selecting the most suitable simulation tool for a specific purpose like research, development or education can be difficult. The variety of simulation tools, features provided in above table, user-friendliness, open source support and dependency on external packages are some of the main considerations which can make it challenging to choose the most suitable robotic simulation tool.

Gazebo is a multi-robot simulation tool which has the capability of accurate and efficient simulation of a population of robots, sensors and objects in a 3-dimensional world. Gazebo generates realistic sensor feedback and has a robust physics engine to generate interactions between objects, robots and environment through URDF scripts. Furthermore, Gazebo provides high-quality graphics, and suitable programmatic and graphical user interfaces. Gazebo is offered freely as a stand-alone software, but has also been packaged along with Robot Operating System (ROS) as the simulation tool. ROS was originally developed by the

Stanford Artificial Intelligence Laboratory in Support of the Stanford AI Robot (STAIR) project. ROS is an open source robotic middleware that provides libraries and tools to help software developers produce robot programs. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing system, package management, and more user-friendly interfaces.

Along with these points, Gazebo fails at several points. While it can import 3D meshes, there are no editing options, which makes it difficult to alter and optimize models. Moreover, Gazebo interface has a number of issues and fails to follow established conventions. Several difficulties were also noted when installing dependencies for Gazebo and for many of its third-party models. It needs high power graphics in the computer too. While not necessarily severe by themselves, these issues together could have a negative impact on a research project. We have experimented our design through Gazebo simulator because of its wide acceptance in the industries and better open-source support.

There are many examples of robotics research projects wherein Gazebo/ROS has been used widely [17]. In this report we have taken into account of "BoniRob", "Rippa" and "Mario" robots. BoniRob is a public funded project, an autonomous agricultural robot that can autonomously perform repeating phenotyping tasks for individual plants. Authors have simulated robot in Gazebo and the system was developed in ROS framework. It includes reliable navigation, perception system, employed sensors and algorithms for robot [19]. MARIO - Mobile Autonomous Rover for Intelligent Operation, is a four-wheel active driving/steering (4WD4S) research prototype that has been designed to be used in the agricultural environment for automation of the agricultural tasks such as data collection, monitoring and inspections. Authors have analyzed performance and algorithms through modelling and simulated environment developed using Gazebo simulator and ROS.

2.2 Sensor Fusion

Despite a lot of research and developments of mobile robots in an unknown environment, a perfect solution to perform Autonomous tasks has still not been found in recent studies.

Nowadays researchers are majorly working on sensor fusion techniques to manipulate data precisely and carefully.

It is difficult for a mobile robot to estimate its exact location even if it has been equipped with a map of a specific environment. In addition, the exploration of unknown environment of the robot's location is more difficult. These difficulties are mainly caused by the accumulative error of the measured sensors, dynamics of robots and environmental changes.

The major reasons that decrease the accuracy of GPS sensor information are listed below: - First, there are such structural factors as time and position error of a satellite, signal refraction by the water vapor in the troposphere and the charged particles of the ionosphere, and various noises. Secondly, the geometric schematism between a GPS receiver and satellites is also an important factor. Thirdly, Selective Availability is an intentional degradation of accuracy intended to prevent the enemy from making tactical use of full accuracy of GPS. Last two reasons don't have any significant effect while the major error is due to the first reason only.

Authors have tested navigation algorithms using single GPS receiver and multi receivers. When using just one single GPS receiver, the reception impediment and noise resulted in serious inaccuracy of positioning. Therefore, they used three GPS receivers to make the positioning of GPS more accurate and reliable. They have also used camera sensor to compensate error. Fused data has been applied to compensate the tracking error between the real path and the target path through Kalman filter. Though the robot traced the path efficiently, there was error in co-ordinates shifting [20].

Likewise, Magnetometer is also an important sensor in the ground vehicle, unmanned aerial vehicles and even in satellites and submarines. Three- axis magnetometers are common sensors used nowadays. For the sake of environmental complexity, magnetometer measurements are corrupted by constant sources and time varying sources. The constant sources include hard irons error, null shift errors, soft irons errors, non-orthogonality and scale factor errors. While the time varying errors come from nearby electronics, such as current carrying wires and on-off transition of the ambient equipment. Hence calibration and error compensation of these errors have been a challenge for researchers. Many researchers

have tested algorithms such as TWOSTEP, least squares, fuzzy least squares, Kalman filtering and unscented Kalman filtering.

Three axis magnetometers are used to estimate the yaw angle, based on the preliminary knowledge of roll angle and pitch angle of the vehicles. Authors have tested algorithms for calibration and error compensation of yaw angle using a 9-axis sensor through a common microcontroller board. They had developed 2D and 3D models to simulate data in the MATLAB environment, while the measurement data come from laboratory experiments. Performance comparison was validated through simulation and efficiency of the proposed calibration method was demonstrated through simulation results. Performances of algorithms contain parameters like time, accuracy, reliability and noise measurement. They had tested least square method and Monte Carlo method for simulations in which Monte Carlo simulations could obtain higher accuracy slightly while it consumes more time to estimate [21].

2.3 Crop-Weed Classification:

In earlier time Robots did not had the capability to detect and classify objects at real-time [1]. Due to development in the field of imaging systems, robots were incorporated with cameras. Robots were equipped with colour cameras and spectral cameras for image acquisition. Colour cameras performed the task of segmentation of plants while the spectral cameras performed the task of classification of plants. Various image processing algorithms and the rise of deep learning has made it possible to classify into multi weed species and crops under outdoor illuminations [6]. The errors of the algorithm, when processing 666 field images, ranged from 2.1 to 2.9%. The ANN correctly detected 72.6% of crop plants from the identified plants, and considered the rest as weeds. It basically uses thresholding technique to segment the vegetation mask from the image and then the morphological features are given to an ANN network for classification [6].

Due to the rise in the development of CNNs, Encoder-Decoder segmentation architecture is used for training and classifying the vegetation mask. This mask along the bounding boxes is trained on a VGG 16 Network to classify the blobs extracted from the bounding box image

into crops and weeds [2]. It is trained on 1500 images and validated on 350 images and tested for 150 images acquired from sunflower field by a custom-built agricultural field robot. It achieves around 90% accuracy in classifying crops and weeds species [2].

The above segmentation method suffers from localization problems, new techniques were evolved which classified each pixel to a specific class (semantic segmentation). As referred from [8], a CNN is proposed based on the same encoder-decoder based segmentation architecture which provides classification in real-time and can operate at around 20Hz as shown in the 2.3(a).

It is a 14-channel network which can run on 20Hz and has less than 30,000 parameters. It is trained on Bonn dataset and multichannel representations along with RGB converge to 95% accuracy faster by 30% than the RGB counterpart. The 14-channel network processes each image frame in 44 ms whereas only RGB channel processes each image frame on the i7+GTX1080 Ti Tegra TX2 in 31ms but has a tradeoff with accuracy metric. The network is capable of running on 5Hz on a flying vehicle [8].

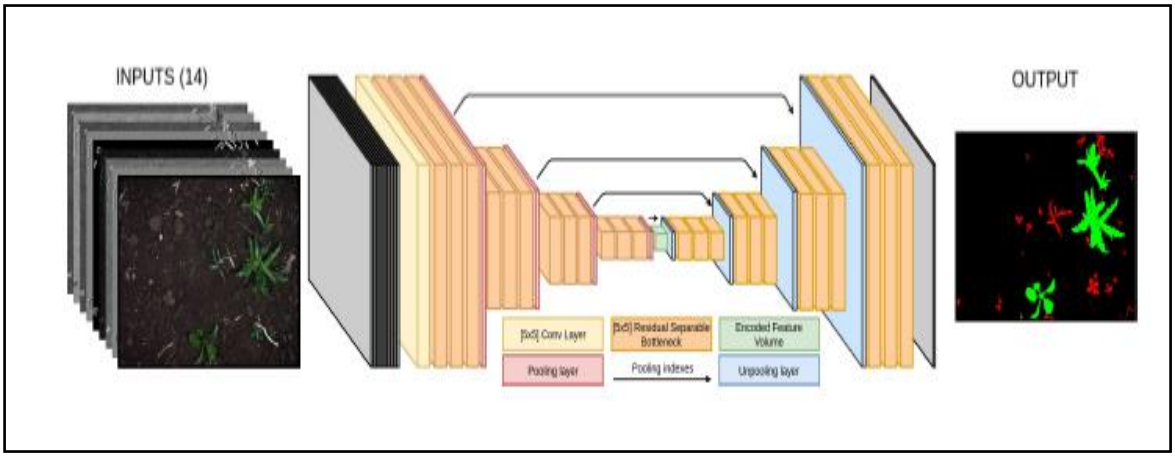


Fig. 2.3 Detailed architecture for semantic segmentation [8].

So, CNN based encoder-decoder based segmentation architecture with multi-channels is suitable for the current scenario which leads to less training time and greater accuracy for the segmentation and classification for multiclass species using semantic segmentation as it provides more information compared to bounding box approach.

*****

SYSTEM DESIGN CHAPTER-3

3.1 Introduction

Much research has been conducted in the development of mobile robotic systems to automate different agricultural tasks for harvesting, weeding, spraying and crop phenotyping. The existing trend within primary production is to use bigger, heavier vehicles to perform agricultural operations in the shortest possible time. An alternative is to let smaller, light weight mobile robots cooperate and serve a team of workers. Such a robotic system achieves high productivity, lower cost and lower soil compaction which will make the production and operations more sustainable. Therefore, developing a robotic system as a mobile base platform is required to augment and automate tasks in agriculture.

For the application of controlling the weed which are surrounding the crops in the field, these points must be taken into consideration. The terrain in which the vehicle will be driven is highly uneven. The Wheels should be rugged enough for driving in these conditions. The drive system should be able to take a zero radius turn if required. The track length of the vehicle may vary. The minimum value of the ground clearance of such robot should be 200mm so that the onboard modules as well as crops might not be damaged. On board power supply and connectivity must be available to the robot. It should be able to handle the real-time data and take decisions based on that in real-time scenario. Precise monitoring and detection of weed are more important so that farmers can easily take care afterward. The main processor should be capable of handling multiple data coming from different inputs at different frequencies.

Ground mobile robots can be categorized based on the locomotion system into legged, wheeled, tracked and hybrid robots. These require different mechanical and control system designs. Wheeled and tracked mobile robots are the most used in field service applications.

Wheeled Robots can use any number of wheels to navigate, with a minimum of one to maximum of any number as per requirement of the user.

The one wheeled robot is highly unstable and it requires extreme engineering and design techniques. The two wheeled robot is comparatively stable as compared to the single wheeled robot but for stabilization it requires its center of gravity to be nearer to the ground and the wheels kept parallel to each other. In a three wheeled system, the wheels are normally arranged in a triangular manner and are hence balanced. The front wheel acts as steering wheel, or most of the time just a balancing wheel while the rear wheels drive the robot. Four wheeled systems are more efficient compared to three or two wheeled systems because they are more stable on the rough surfaces as compared to three wheeled systems and cornering can be done more efficiently. The first two of the four wheels can be used to steer and the next two to drive the robot. Balancing a four wheeled robot is never an issue because the center of gravity is usually in the middle and that's why this system is commonly used in automobiles. So, a four wheeled system will suit perfectly for an agriculture robot.

The overview of AGRIBOT highlighting major systems is given below: -

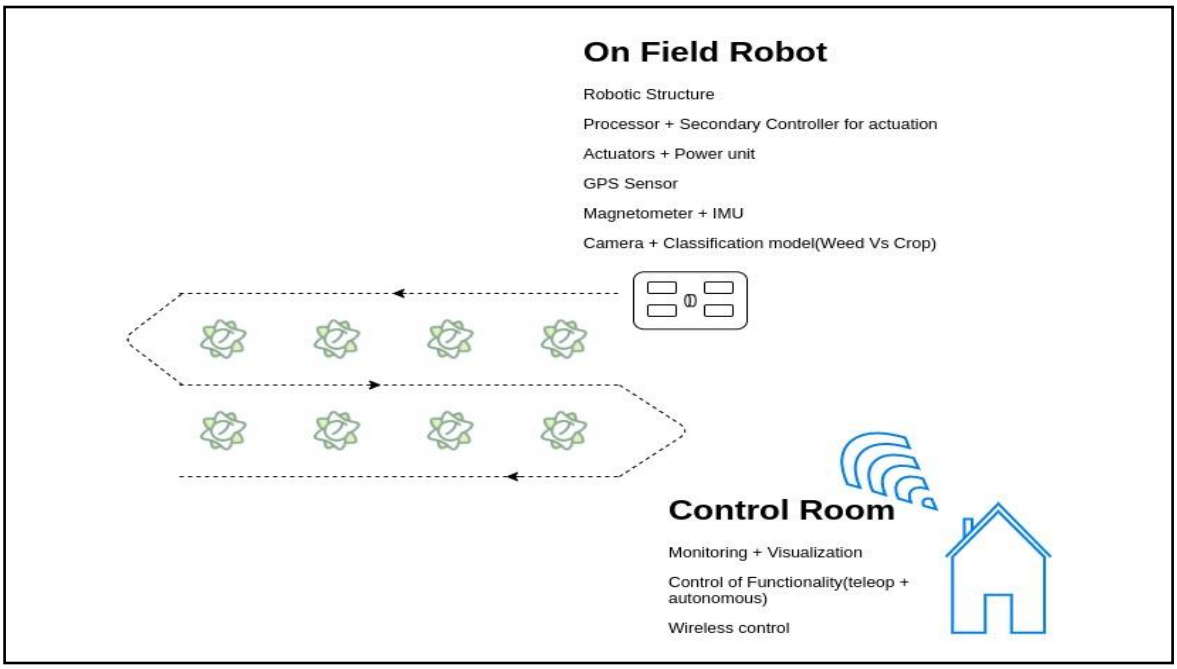


Fig. 3.1 Overview of AGRIBOT.

3.2 Mechanical Design

3.2.1 Design Overview

In regards to the robot mechanical design, several design criteria need to be considered such as body weight and size, strength, manufacturability, manufacturing time and cost. In the initial design process of this robot, it was very important to have a modular system for easy assembly and further maintenance. The driving mechanism consists of a DC motor, leg structure links, bearings and output shaft which drives the wheel. For this robot, four of these wheel modules attach to the chassis is shown in diagram [3.2 (a) and 3.2 (b)]. This design approach allows initial analysis, tests and validation on one-wheel module and also decreases the production time and cost for all four-wheel modules. The chassis structure is very simple and it is built from aluminum box section. Design is kept such a way that length between sides can be changed according to field requirements. Primarily we have considered the dimensions according to a sugarcane farm.

3.2.2 Robot Structure Design Using 3D CAD (Solidworks)

All the robot parts have been designed using 3D CAD to enable for further stress analysis and complex frame structure of robot. For this aim, the educational version of SolidWorks 2016 software has been used to design the 3D model of the robot. SolidWorks is a CAD modelling software and allows us to analyze stress and motion dynamics on given body. This design includes the parts which should be manufactured and also the other parts supplied from off the shelf. In designing the robot's body parts, assimilability with other parts such as motors and cameras has been considered. This 3D design allows us to study and analyze the mechanical interference between each part as joints and links. URDF of the robot has been generated using custom add-on available in CAD software. Physical criteria of robot body is also applied through this exporter. Then URDF scripts was used in ROS framework to simulate the robot for further functionality. Below are the diagrams taken from Solidworks during designing.

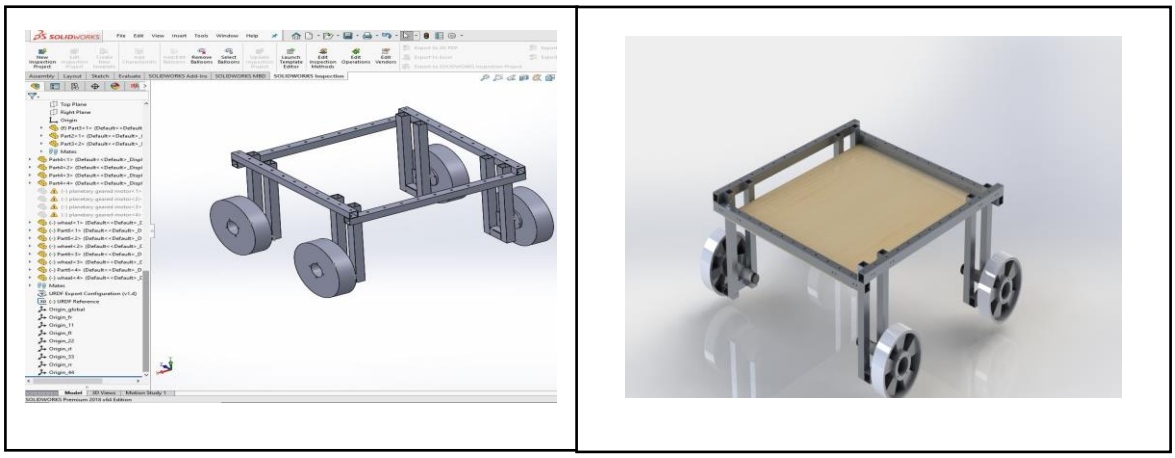


Fig. 3.2 (a) and (b) Solidworks Model of Agricultural Robot.

3.2.3 The Transformation Tree (TF)

The TF tree describes the various frames of the robot; such as the base frame, joint frames, arm frame, sensor frame, etc. and their relative relationships. The TF tree is needed to perform the complex robot kinematics operations; by combining the TF tree with the URDF model of the machine, all kinematics operations, forward or inverse, are then performed by ROS's TF package under the hood, saving the developers from the burden of performing these complex mathematics operations manually. We have used two packages from ROS libraries, robot state publisher and joint state publisher. These packages are responsible for all background coordinates transformations. Figure [3.2 (d)] shows the different frames and their axis.

The kinematic chain of the robot generated by URDF file is shown in figure [3.2 (c)]. This robot has 8 links and 9 joints. In URDF terminology multiple links can be connected to one link by joints and defined parent and child links. Driving joints are defined as continuous. Continuous joints have given to 4 joints so that all wheels can rotate freely around one axis. This robot definition allows to use Gazebo for dynamic simulation and RVIZ for visualization. This robot has four DOFs in total on robot local frame which provides three DOFs motion (V x, V y, and ω) in the world frame.

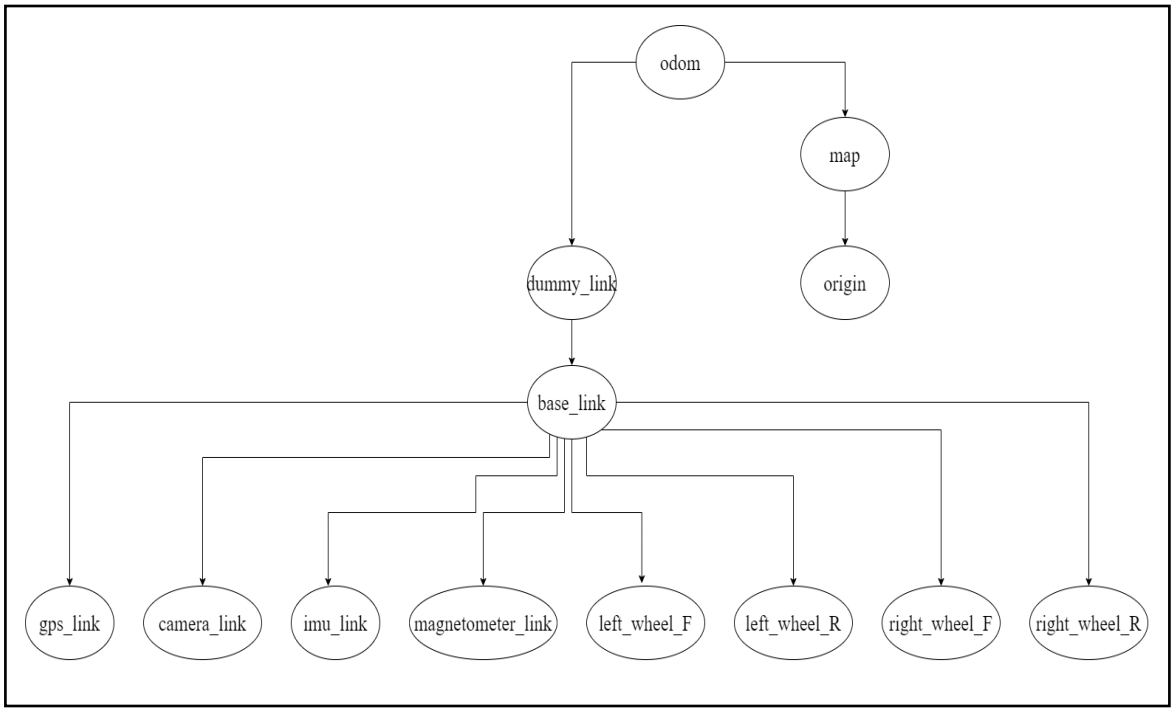


Fig. 3.2 (c) Kinematic Chain of the robot generated by the URDF File.

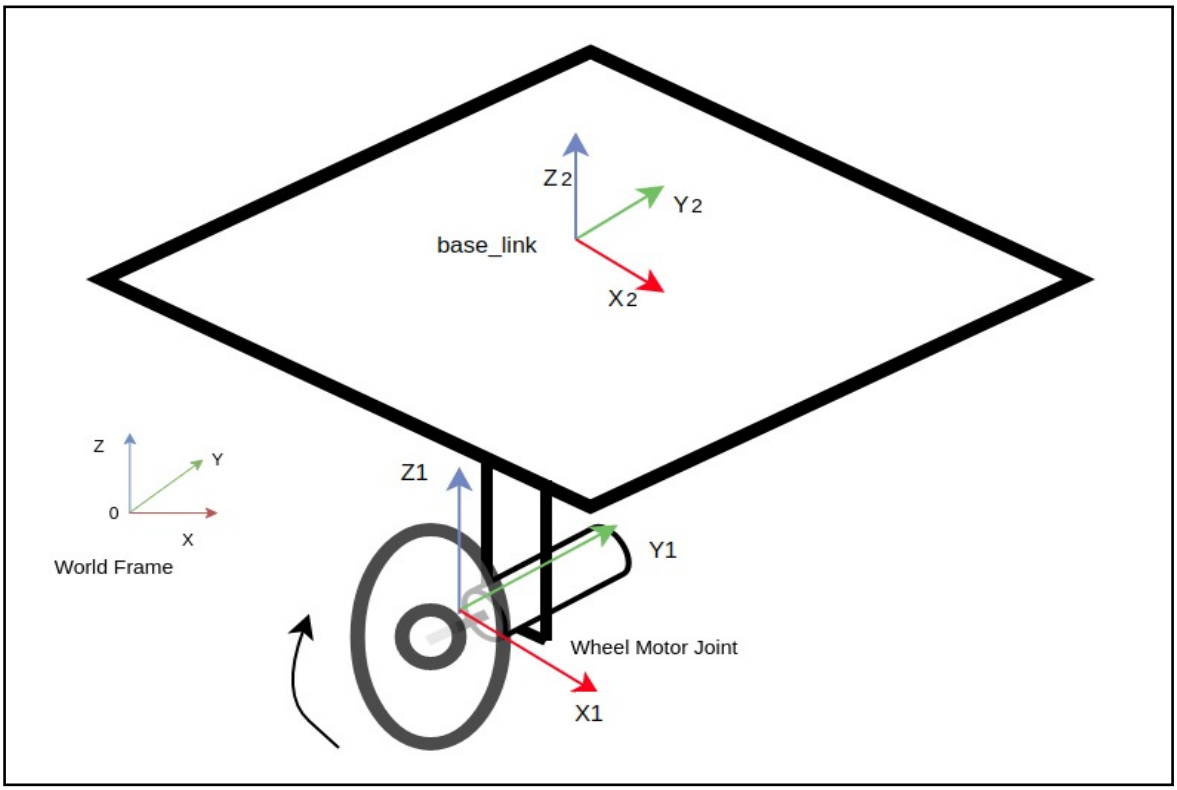


Fig. 3.2 (d) Example of different coordinate frames of Robot.

3.2.4 Kinematics of Drive System:

Generally, the locomotion control system of mobile robots can be divided into differential drive system, Skid-Steer driving, Ackermann driving system and omnidirectional system. Selection of the appropriate drive system depend on the type of application and the level of accuracy which is expected.

Differential Drive System

This is the most common control mechanism for robot builders. Velocity difference between two motors drive the robot in any required path and direction. Hence the name differential drive. Differential wheeled robot has two independently driven wheels fixed on a common horizontal axis.

There are three fundamental cases which can happen in a differential wheeled robot:

1. If the angular velocities are identical in terms of both values and direction, i.e. if both the wheels are driven at the same speed and same direction (either clockwise or anticlockwise) then the robot is more likely to follow a linear path, either forward or backward based on the motors spin.
2. If the angular velocities are identical in terms of values and opposite in direction, i.e. if both the wheels are driven in the same speed but in the opposite direction (One clockwise and other anticlockwise) then the robot tends to spin around its vertical axis. This complete turn capability is one of the greatest advantages of a differentially driven robot (i.e. zero radius turn).
3. If the angular velocities are different in terms of values (same or different direction), i.e. if the wheels are driven at different speeds in the same direction or opposite direction, then the robot makes a curve motion. Lastly, if one of the wheels rotate and the other stays still then the robot almost makes a 90° turn.

   One of the major disadvantages of the differential drive system control is that the robot does not drive as expected. It neither drives along a straight line nor turn exactly at expected angles, especially when DC motors are used. This is due to the difference in

the number of rotations of each wheel in a given amount of time. To overcome these problems dual-differential drive or Skid-Steer drive system can be used which can mechanically perform well in rough terrain.

Skid Steer Drive System:

Skid-steer locomotion is commonly used on tracked vehicles such as tanks, bulldozers, heavy agricultural machines, but is also used on some four- and six-wheeled robots. On these robots, the wheels (or tracks) on each side can be driven at various speeds in forward and reverse (all wheels on a side are driven at the same rate). There is no explicit steering mechanism--as the name implies steering is accomplished by actuating each side at a different rate or in a different direction, causing the wheels or tracks to slip, or skid, on the ground.

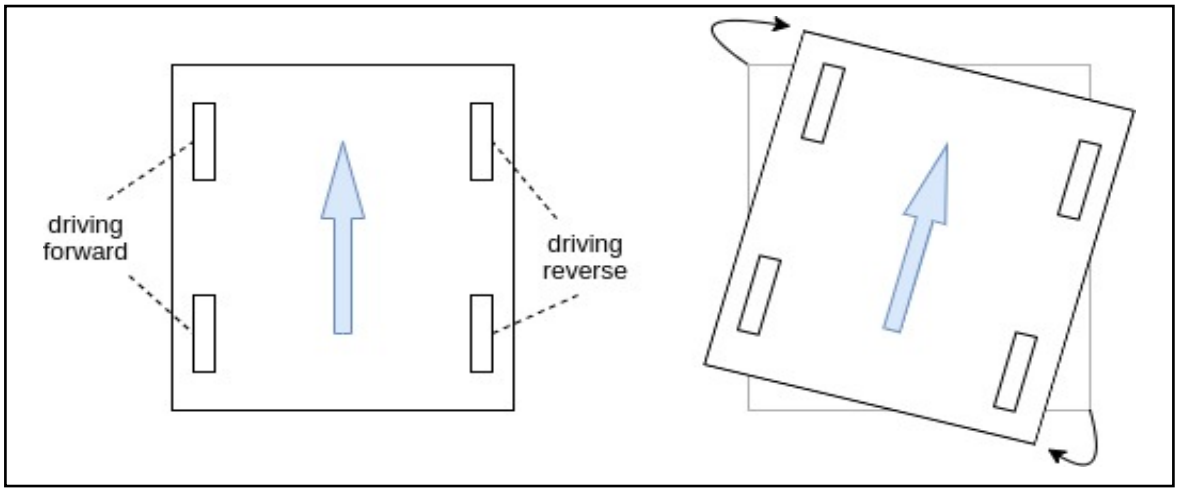


Fig 3.2 (e) Differential Dive System [23].

In the above left figure, the wheels on the left side are driven forward and the wheels on the right side are driven in reverse at the same rate. The result is a clockwise zero radius turn about the center of the vehicle shown in the right figure. In this sense Skid-steer is closely related to the differential drive system, but major drawback is exact location of robot cannot be identified as it skids during motion. In this project, we have implemented Skid-steer drive system as it provides increased traction on rough terrain which is main factor in our project. Other advantage is we don't need any steering mechanism and increased number of wheels

provides supports to heavy body of robot. So based on these constraints, we have used Skid Steer drive system to meet the requirements of the agricultural robot.

3.2.5 Standard kinematics Model

An ideal differential driven kinematics model cannot be applied for skid steering robots. So, the improvised Kinematic Schematic has been given in the figure below.

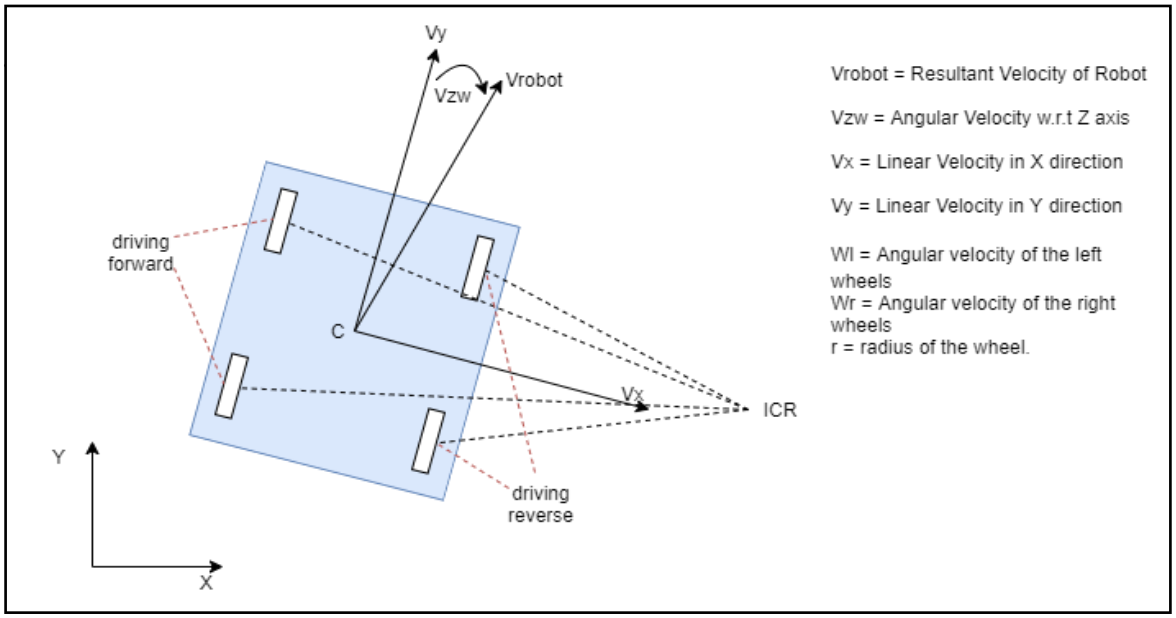


Fig. 3.2 (f) Kinematics model of Skid-Steering System [23].

A state space model of the system was developed with the following assumptions [23]: -

(1) the mass center of the robot is located at the geometric center (C) of the body frame;

(2) the two wheels of each side rotate at the same speed;

(3) the robot is running on a firm ground surface, and four wheels are always in contact with the ground surface.

The state vectors were the linear velocities in the X and Y direction and the angular velocity w.r.t the Z-axis.

The instantaneous radius of curvature derived from this model is given by: -

$$R = \frac{Wl+Wr}{Wl-Wr} \times \frac{Vx-(Wl)r}{Wz} \qquad (3.1)$$

Apart from that we derive an important quantity χ which is called as ICR co-efficient and depending upon its value we can differentiate between an ideal differential drive and the skid steering drive system where B is the lateral wheel base.

$$\chi = \frac{(Wl - Wr)r}{(Wz)B} \tag{3.2}$$

The above calculations are being used by GAZEBO skid-steer plugin which drives the robot. However different velocity commands are decided by path planner algorithm which will be discussed later in section 4.2.

3.3 Embedded System Design

The embedded system of the agricultural robot can be divided into five parts. They are central processing unit, control unit, power supply unit, peripheral circuitry and motors.

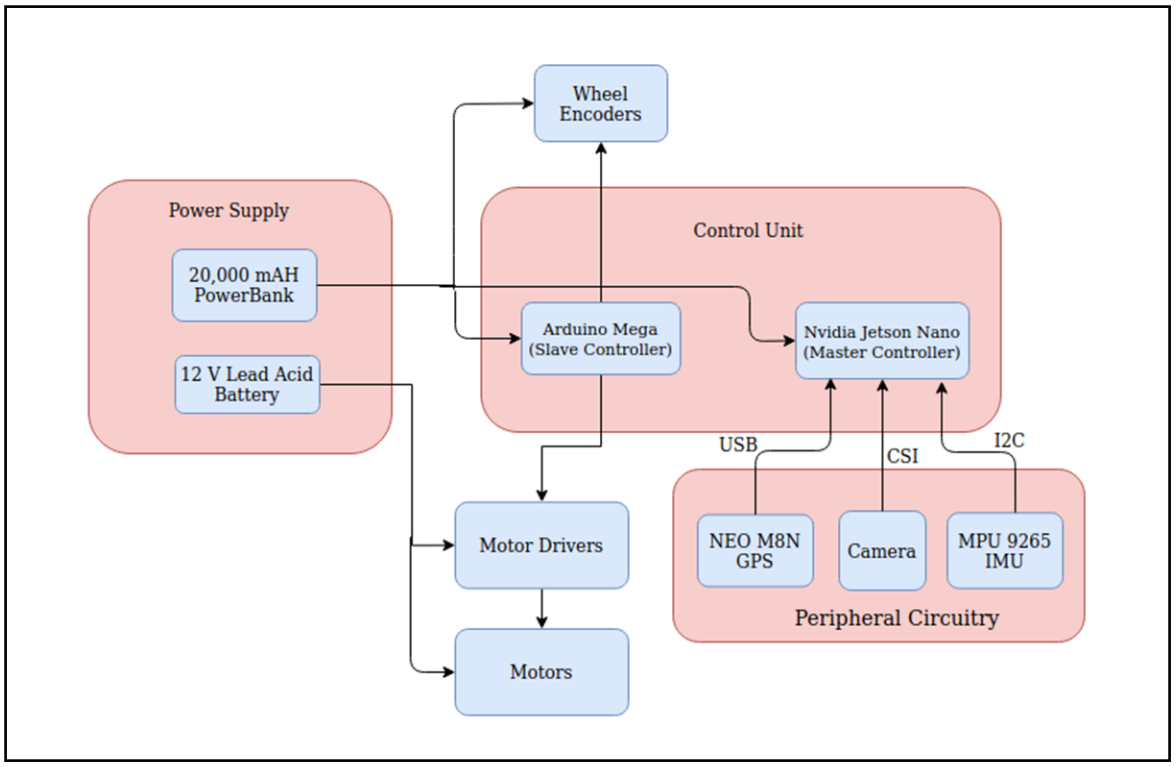


Fig. 3.3 (a) Embedded System Block diagram of Agricultural Robot.

3.3.1 Central Processing Unit

The Central Processing Unit includes a primary processor which is used for processing the data available from the GPS Sensor, Camera and IMU sensor. The primary processor used here is NVIDIA Jetson Nano.

NVIDIA Jetson Nano

The NVIDIA Jetson Nano Developer Kit is an embedded system-on-module (SoM) with quad-core ARM Cortex-A57, 4GB 64-bit LPDDR4 RAM and integrated 128-core NVIDIA Maxwell GPU. In terms of AI Performance, the Raspberry Pi Variants offer 21.5 GFLOPS while Jetson Nano offers 472 GFLOPS (i.e. 22 times faster). The Jetson Nano can handle the frame rate of 15-20 fps for running the UNet model whereas the Raspberry Pi counterparts can handle only maximum of 5 fps. Moreover, taking into account, computations on processing data from peripheral sensors like GPS & IMU, Jetson Nano is considered over Raspberry board for better processing.

Another point which is to be taken into consideration is the GPU of Jetson Nano. It has 128-core which will be highly useful to obtain segmentation and data level parallelism of the input image as compared to the Videocore GPU of Raspberry Pi. Therefore, the overall of performance of Jetson Nano is better than Raspberry Pi variants and thus it becomes the suitable choice for the main processor in the agricultural robot.

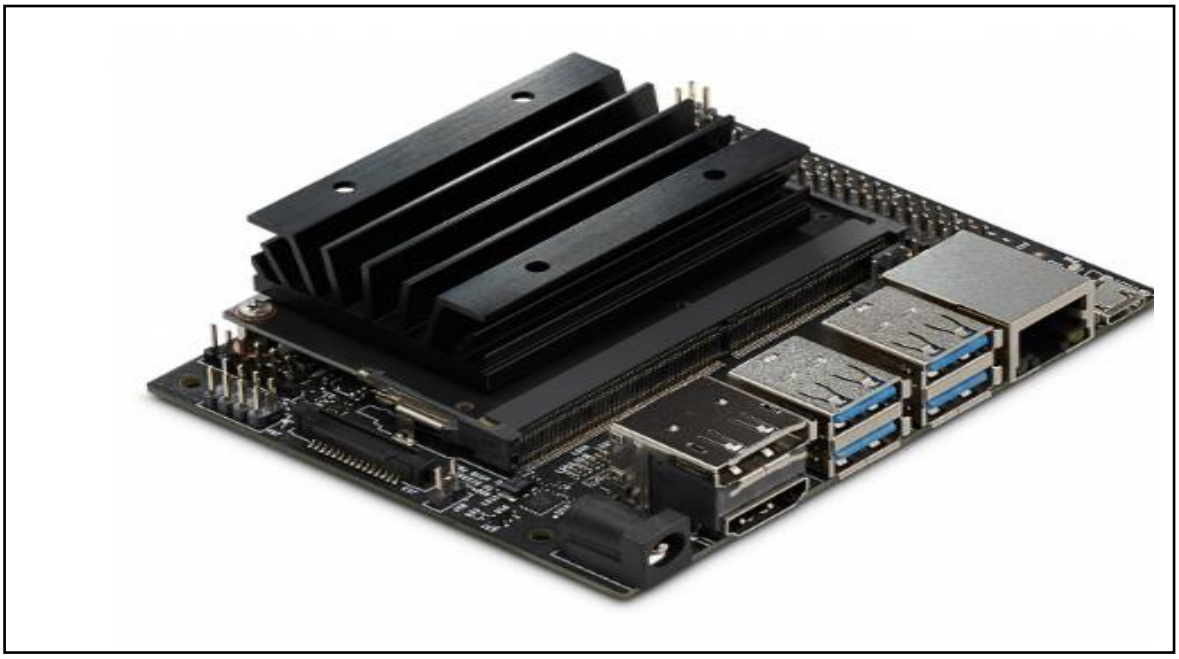


Fig. 3.3 (b) Jetson Nano Developer Kit [13].

The Jetson has been configured and setup remotely using the ssh protocol in the Linux terminal.

3.3.2 Control Unit

The Control unit consists of a slave controller, encoder and motor driver circuit. The secondary controller is responsible for driving the motors and taking the feedback from the motors via encoders. The secondary controller used in the agricultural robot is Arduino Mega 2560.

Arduino Mega 2560

Table 3.3 Comparison of different Microcontroller boards developed Arduino.

Pin Type	**Requirements**	**Arduino Uno**	**Arduino Mega**	**Arduino Due**
External Interrupt	4(Wheel Encoders)	2	6	54(All I/O Pins)
PWM	4(Motors)	6	15	12
Digital pins	4(Wheel Encoders) + 8(Motors)	14	54	54(All I/O Pins)

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. The alternatives of Arduino Mega 2560 are Arduino Uno and Arduino Due.

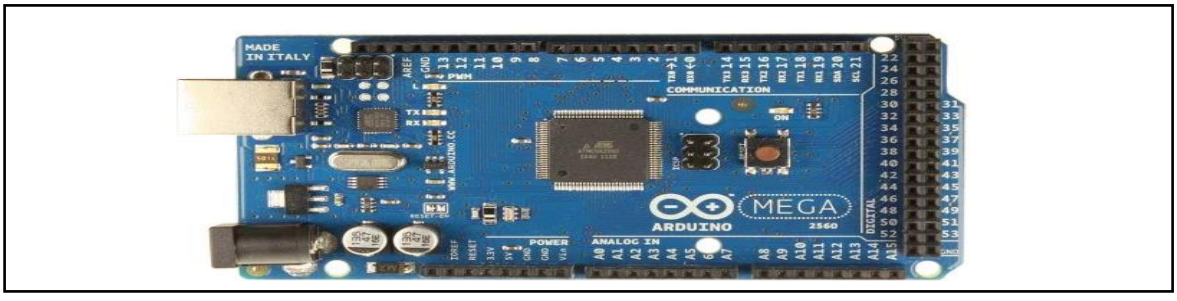


Fig. 3.3 (c) Arduino Mega 2560 Microcontroller board [25].

So, the Mega 2560 & Arduino Due meets the required specifications of the slave controller for the agriculture robot. The problem with Arduino due is that the maximum logic level input and output of each pins is only 3.3V. So, a Logic Level Controller has to be also used along with the Due which unnecessarily increases the complexity. Hence, Arduino Mega is chosen as the slave controller as it is able to handle computation callbacks of interrupts from wheel encoders as we don't require fast driving in our application unlike other mobile robots. The Jetson will give command to the Mega 2560 which in turn will the drive the motors as per requirement. It will take feedback from the motors which consists of encoders and try to drive the motors to the required speed.

Encoder

The encoder is used to translate rotary or linear motion into a digital pulse (signal) which used to control speed, direction, distance or position of the motor. The encoder is used to precisely control the displacement of the agricultural robot. The encoder used in this robot is in-built in the motor itself.

Resolution of the Encoder: - 1848 PPR.

Motor Driver

Motor driver acts as an interface between the motors and the control circuits. Motor require high amount of current whereas the controller circuit works on low current signals. So, the function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor. Rated current of the motor is 1-2 A and hence motor driver which can continuously drive load of 2 A and stall current of up to 15 A is required.

Motor

The primary focus of the agriculture robot is to remove the weed so a low RPM and High torque motor will meet the requirements. The motor which has been used is Planetary DC geared Motor with 24 RPM and a rated torque of 12.24 kg-cm and rated current of 0.9A.

3.3.3 Peripheral Circuitry

The Peripheral Circuitry has three components present in it. They are Raspberry Pi V2 Camera, MPU 9265 IMU and Neo-M8N GPS Module. These are directly interfaced with the primary processor (i.e.) using CSI, I2C and USB Interfaces respectively.

Raspberry Pi V2 Camera

The Raspberry Pi v2 Camera Module has a Sony IMX219 8-megapixel sensor. As compared to the previous version, the field of view has been increased in both the horizontal and vertical fields. One of the major advantages of using this camera than the cameras available with the USB Interfacing is the dedicated ports for MIPI CSI-2 camera interface which can be directly connected to the GPU of the Jetson Nano, therefore the load on the CPU will be less and thereby making it available for the other process. Also, the frame rate of USB Camera is very less compared to that of the V2 Camera for the same resolution. So, the Raspberry Pi V2 Camera Module meets the required specifications of the agricultural robot.

MPU- 9265 IMU

Fusing data acquired from IMU (Inertial Measurement Unit) sensor and GPS module the robot can be precisely located. The MPU 9265 is a 9 DOF (degrees of freedom) or a nine-axis IMU sensor, which means that it gives nine values as output: 3 values each from the accelerometer, gyroscope and magnetometer. The advantage of using a 9DOF system over the 6DOF system is that the best data from all three sensors, identifying and compensating for the flaws of some with the strengths of the others. It provides an accurate, low noise, smooth but responsive estimate of the device orientation angles, rotation speed, linear acceleration, Earth gravity, and geomagnetic fields; delivers sensor calibration error information; and is more resilient to magnetic interferences. The outcome greatly exceeds the sum of individual sensors' contributions, which is the essence of sensor fusion.

NEO-M8N GPS Module

The NEO-M8 series of standalone concurrent GNSS modules is built on the exceptional performance of the u-blox M8 GNSS engine in the industry proven NEO form factor. The NEO-M8 series provides high sensitivity and minimal acquisition times while maintaining low system power. The NEO-M8N provides best performance and easier RF integration. Sophisticated RF-architecture and interference suppression ensure maximum performance even in GNSS-hostile environments. The NEO-M8 series combines a high level of robustness and integration capability with flexible connectivity options. This makes NEO-M8 perfectly suited to industrial and automotive applications. So, NEO-M8N is used as the GPS Module in the agriculture robot.

3.3.4 Power Supply Unit

The Power Supply Unit has two components present in it. They are 12V lead acid battery and 20,000mAh power bank.

3.4 Modelling & Simulation of AGRIBOT

As we have seen kinematics and embedded system module in above subsections, in this subsection all the physical links and joints, as well as sensing and basic control interfaces of AGRIBOT is described. We have modeled and simulated model in gazebo simulator. The Gazebo simulator has provided dynamics simulation by considering gravity, friction, and contact forces provided by the included physics engines Open Dynamics Engine – ODE. Different types of Gazebo plugins have been enabled to develop control interfaces and sensing systems for AGRIBOT. Further sections are divided as:

1. World environment model description

2. Physical model description

3. ROS/GAZEBO plugins to model the sensors and control/hardware interfaces

Fig. 3.4 (a) shows the general structure and the required components in Gazebo to model a robotic system.

3.4.1 World Description

World is a general term to describe objects, global parameters and physics properties. By default, a world is defined by Gazebo with default required parameters. The objects in the world can be static or dynamic. Static objects such as building, lights or walls are defined by their visual and collision geometry. Dynamic objects such as robots are defined not only by visual and collision geometry but also by their inertia information. Objects can be created using standard geometric shapes, or inserted from the model database, or created and by any 3D modelling tools and imported to the Gazebo simulator environment. Scripts can also be added in sdf format for advance modelling.

Fig. 3.4 (b) shows a simulated environment in Gazebo with the set physics parameters such as gravity. Static objects as the modelled terrain, oak trees, house, lamp pole and agricultural environment (Farm) are also included in the simulated world environment. These objects were created using the sdf scripts and models were integrated from GAZEBO library.

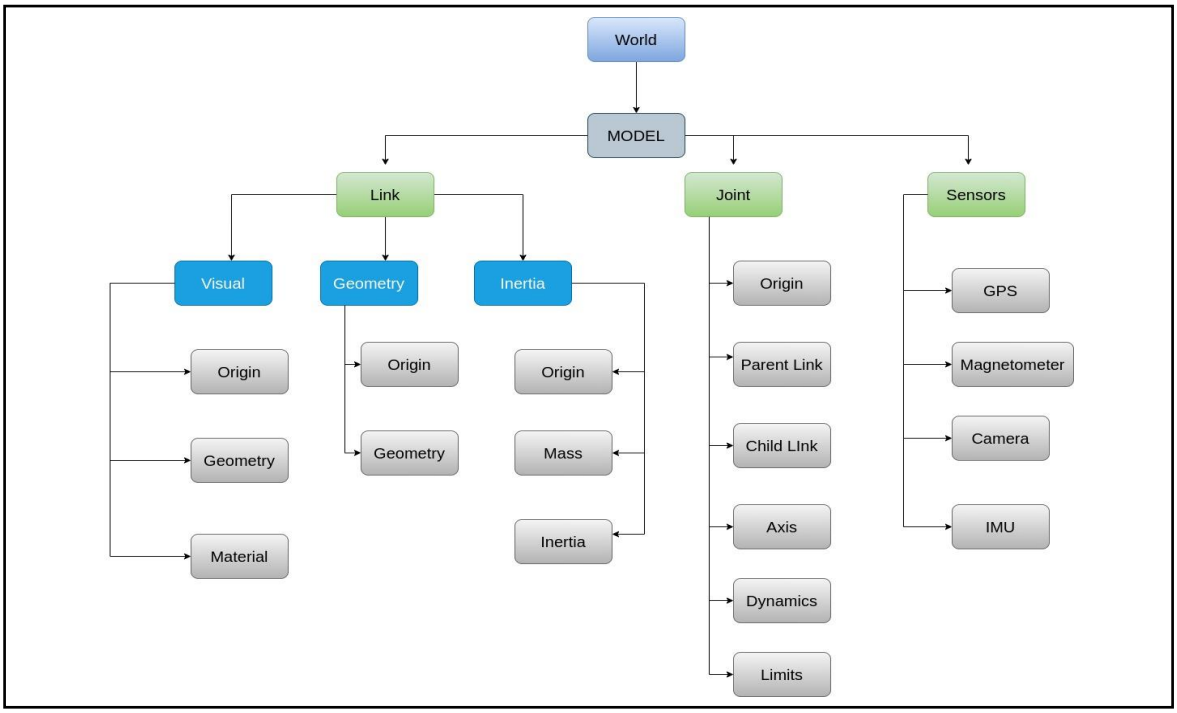


Fig. 3.4 (a) General structure and the required components in Gazebo to model a robotic system.

Fig. 3.4 (b) A simulated environment in Gazebo.

3.4.2 Physical Model Description

AGRIBOT simulated model: A robot, as a dynamic object in the world, consists of links that are connected to each other by joints. A link in Gazebo describes the kinematic and dynamic properties of a physical link in the form of visual/collision geometry and inertia information. A joint models kinematic and dynamic properties of a joint such as joint type, motion axes, and joint safety limits. All this information is described in the Universal Robotic Description Format - URDF file format so that we can frame design constraints.

URDF is an XML file format used by Gazebo and ROS to model all the components of a robot. To model AGRIBOT as a 4WDS platform, 10 links and 9 joints need to be defined as described earlier. The base_link describes the chassis and four links are connected to it by four fixed joints as driving links. Each of the driving links are connected to a wheel by a continuous joint through motor. Each sensor also should be defined as a physical link attached with a fixed type joint to the base_link. with no attached sensor. A virtual link called base_footprint is also needed by some of the of the ROS third party software such as navigation which is placed on the ground and it is sized as the footprint of the platform. Also, gazebo doesn't accept root link with inertia. Fig. 3.2 (e) in the preceding section presents the

kinematic schematics of the base_link and a wheel module with its links and joints. Fig. 3.4 (c) shows the kinematic diagram of AGRIBOT platform, showing the kinematic chain of all the physical and virtual links and joints, each joint with information of its placement with respect to the previous link and all calculations are being done by the TF package.

The physical geometry of each link needs to be defined in the form of visual and collision geometry. There are two ways to model the geometry of each link of a robot, inserting standard 3D shapes, or importing as mesh files. Physical geometry for a typical four wheeled robot can be modelled using a cubic box as the chassis and 4 cylindrical shapes as the wheels. Using continuous joints, wheels can be attached to the chassis. Inertial information of each link is essential, if a proper simulation is required. Inertial parameters define the mass, centre of the mass, and the moment of inertia tensor matrix for each modelled link. This information is obtained from SolidWorks through URDF exporter. The example of AGIRBOT URDF code to model the base_link has been added in Appendix A.

The alternative approach to define visual and collision geometry is to use mesh files in the format of COLLADA or STL files. These files can be generated by CAD tools such as Blender, SolidWorks or SketchUp. WE have used SolidWorks 2016 for generating CAD model. To provide colour information, gazebo material tag can be used. Collision geometry does not need material or colour information, therefore it can be represented as STL file format from Solidworks..

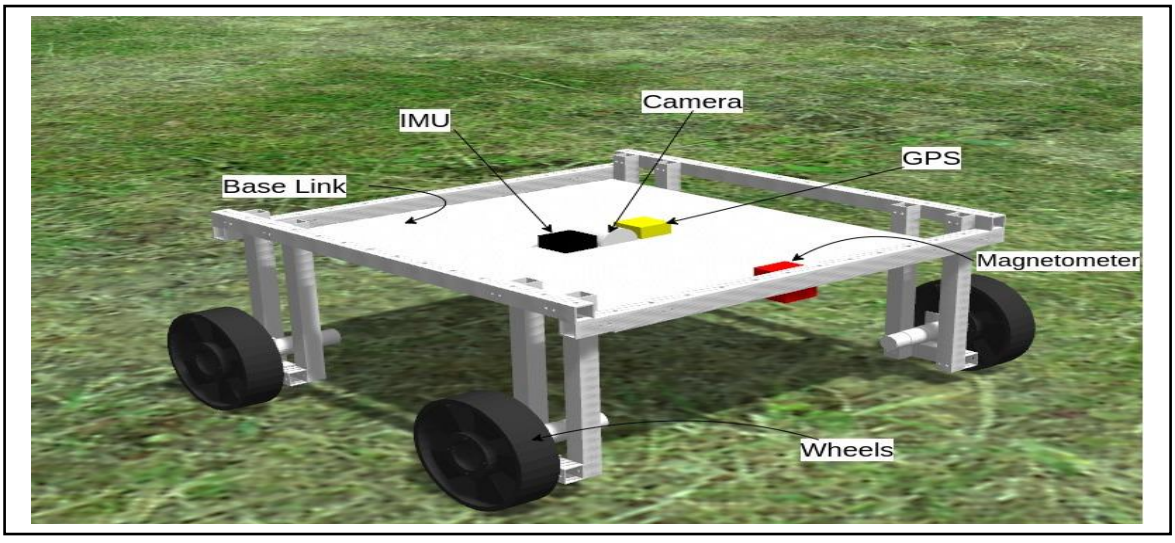


Fig. 3.4 (c) Kinematics Diagram of AGRIBOT Platform.

Joint Representation Joints are needed to connect the links to each other and form the kinematic and dynamic relationships between them. Joint element in the URDF file defines the kinematics and dynamics of the joint as well as joint type and safety limits. Fig. [tree figure] shows the tree structure of a two-link robot for better understanding of the different terms in joint element. The modeled continuous joint element that connects the driving link to wheels in AGRIBOT has also been added in Appendix A.

3.4.3 Sensor Modelling

With many different sensors on the robot, each sensor can be simulated independently as a Gazebo plugin and must be physically attached to the robot model as a link. These plugins are included in the URDF file. These plugins are typically C++ libraries loaded by Gazebo that have access to Gazebo' s API. In general, plugins are permitted by Gazebo to perform different kinds of task such as motion control or getting sensor data. These plugins output sensor information in form of standard ROS messages and services coded in libraries or own can be generated if needed. Common parameters, such as error characteristics and transformation frame of each sensor, can be defined as well as other parameters related to each sensor. By default, sensors in Gazebo have no noise and they sense the simulated environment perfectly. To make them more realistic, noise can be added. A first order Gaussian error model is typically used for all the sensors to add noise to the measurement taken from each sensor. This is modelled by setting the mean and the standard deviation of the Gaussian distribution. Each measurement of Y(t) at time t is given by:

$$Y = Y' + B + N_y \tag{3.3}$$

$$B' = -B/\tau + N_B \tag{3.4}$$

where in equations. (3.3) and (3.4), Y' is the raw measured value, B is the bias/offset, N_Y is additive noise that affects the measurement, and N_B defines the characteristics of random drift with time constant τ. This includes drift due to environmental changes, voltage changes and age of sensor [17].

Inertial measurement unit (IMU)

The inertial measurement unit (IMU) reports the robot body' s acceleration from a three-axis accelerometer, angular rates from a three-axis gyroscope and absolute orientation around the Z axis by a magnetometer. Integration of these measurements with other sensors provides a good reference for a localization system of robot. Noise and bias are the two types of disturbance that are applied to the angular rates and acceleration measurements of IMU. Four groups of parameters need to be set for the IMU model: angular rate noise and bias, acceleration noise and bias. Also, no noise or bias is added to the orientation measurement as it is considered a perfect value in the world frame. Noise is sampled and added from a Gaussian distribution by setting the mean and standard deviation of the Gaussian distribution. Bias is sampled once and will be added at the start of simulation. The simulated IMU on AGRIBOT is modelled with associated noise in Gazebo, the code in the Appendix A shows the required URDF code to model the IMU. Parameters for plugin has adopted from hector project.

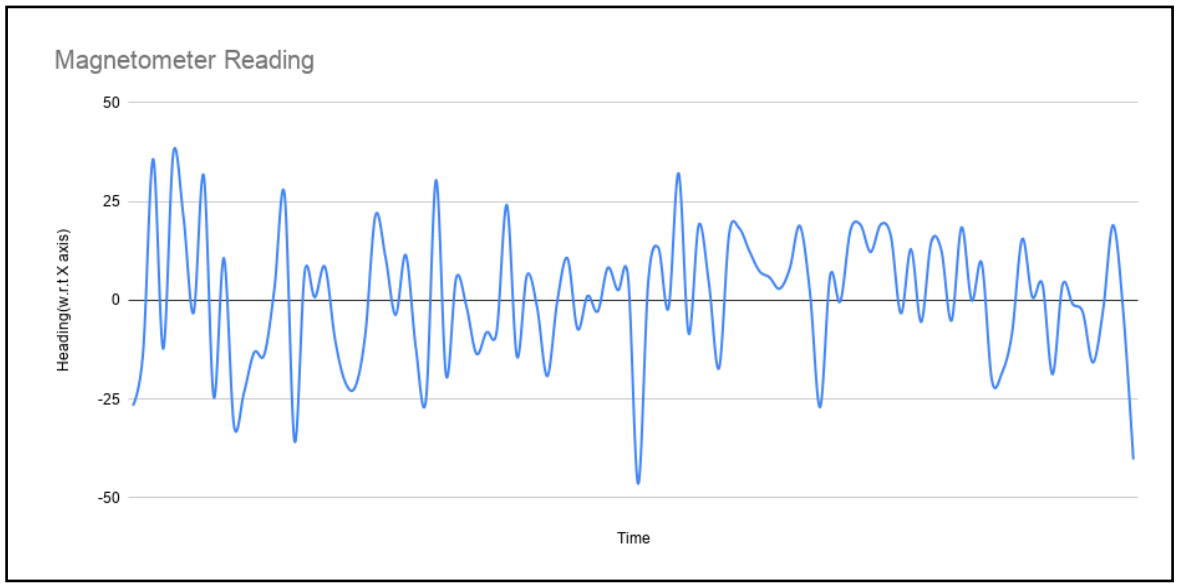


Fig. 3.4 (d) Magnetometer Reading simulated with the help of IMU Plugin on Gazebo.

Vision Camera (RPI-2 Camera)

Vision camera is a vision system to simulate the real physical world. Camera gives the ability to generate 3D images. Typically, a camera that is placed at the center of robot to capture

images of field. Through several pre-processing steps, images are being converted into video and DL model has been implemented to classify between crop and weed. All the required parameters such as frame rate, image width and height, output format, base line distance and noise parameters can be defined within Gazebo plugin through URDF. A Gaussian disturbance is sampled for each pixel individually and it is added to each colour channel of that pixel. Fig. 3.4 (e) shows the visualization of simulated camera by showing the image of an example scene in Gazebo. Parameters for plugin has adopted from hector project to model the simulated camera.

Fig. 3.4 (e) Visualization of simulated camera by showing the image of an example scene in Gazebo.

GPS

GPS sensor reports latitude, longitude and altitude of robot body from reference provide through parameters. Integration of this sensor with IMU, after converting it into XY frame helps us to locate robot in outdoor environment. Though real GPS sensors have around ±2m error in its data, we can use it with proper filters and methods like DGPS. For simulation

Prospective, Gaussian noise is used to model sensor. Noise is sampled and added from a Gaussian distribution by setting the mean and standard deviation of the Gaussian distribution. Appendix A code shows URDF modelling for GPS in ROS framework and parameters has adopted from hector project. Fig. 3.4 (f) shows the variation in GPS data for stationary pole at the starting position. As we can see from the given figure, the variation is roughly 1 to 2m from the exact location and it is random in nature.

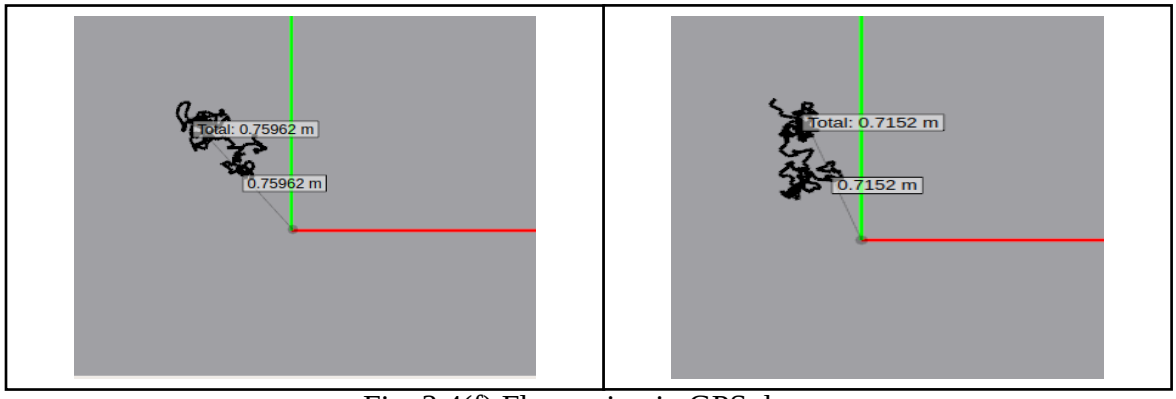


Fig. 3.4(f) Fluctuation in GPS data.

3.4.4 Controller for AGRIBOT

Control Plugin

To be able to control the motion of each joint in one degree of freedom, a Gazebo control plugin is required. Gazebo also needs to be interfaced with a robot middleware such as ROS to control each joint. A meta package called gazebo_ros_pkgs provides a set of ROS packages to interface with Gazebo. A set of packages called ros_control provide controllers, hardware interfaces and toolboxes to control joint actuators. Common controllers such as effort, position and velocity are provided by ros_control. These controllers (typically PID type) take joint states and set points as inputs and output effort, position or velocity. Each joint is interfaced with the relevant controller by the relevant hardware interface. The same developed software control system is used to control the physical robot, so that the control messages are written to the physical speed/position controllers to actuate control the actuation of the joints. The speed and position feedbacks are read from the encoders back to the control system. An overview of the low-level control system of simulated AGRIBOT is shown in Fig. 3.4 (f) By launching the ROS package containing the URDF file of AGRIBOT,

the simulated model is opened in the virtual world of Gazebo. This also loads the control interface and waits for all the controllers to be loaded.

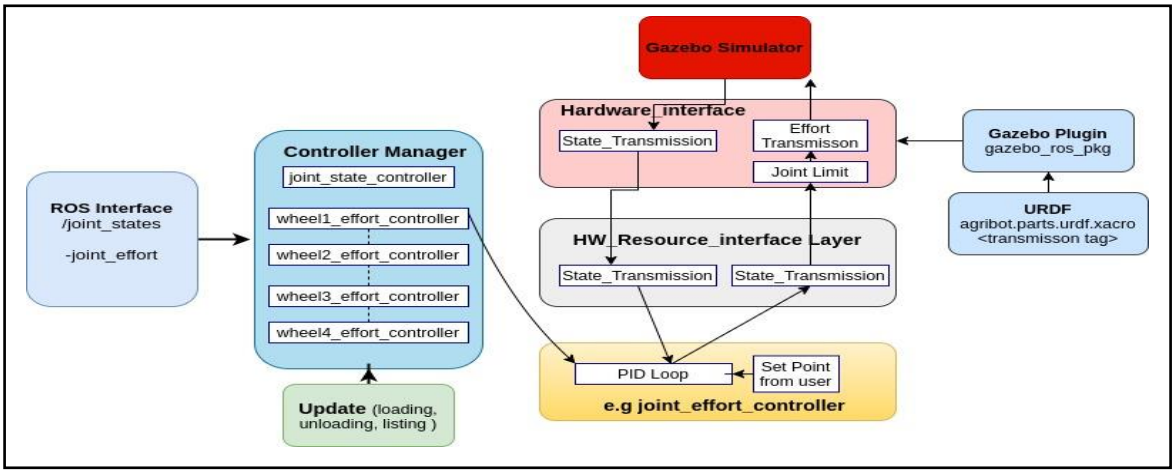


Fig. 3.4 (g) Low-level control system of simulated AGRIBOT.

The transmission tag in the AGRIBOT URDF file defines the type of command interface and the relationship between the joint and actuator. Controller manager provides the infrastructure to load, start, stop and unload the controllers in a real-time manner. A YAML configuration file is also needed which includes information of controllers such as joint controller type and parameters for PID. Hardware_interface provides position, velocity and effort interfaces between ROS and Gazebo. Our system has 5 controllers of which one provides joint states, four are effort controllers to control each of the driving joints with required torque.

3.5 Crop Weed Classification

3.5.1 Basic Understanding of CNN

In Deep Learning, CNN or ConvNets is one of the basic networks to perform the task of image recognition, classification, object detection etc.

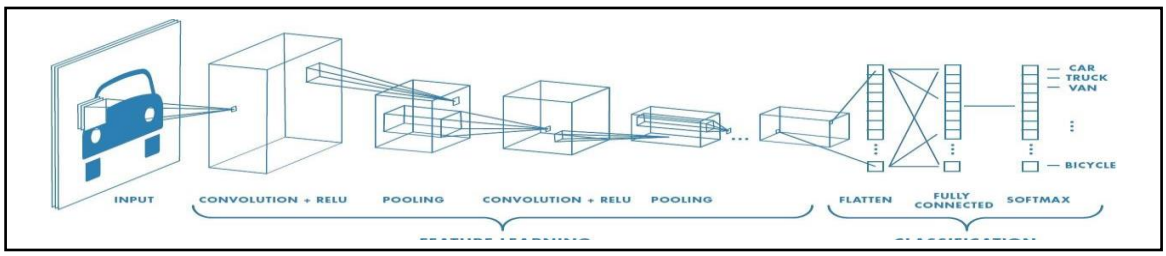


Fig. 3.5 (a) Basic CNN Model [13]

As shown in the Fig 3.5 (a), ConvNets basically are made up of Convolutional, Pooling, Batch Normalization and Fully Connected Layers.

Padding

There are 2 types of padding: Valid and same. In case of valid padding, no padding is there in the output. In case of same, padding is added in such a way that output size becomes equal to input size after convolution operation.

Here, n×n is the input size, f×f is the filter size and p is the padding size. In case of Valid padding, output size can be determined by equation (3.5). While in case of same, it is determined by equation (3.6) and padding size is determined by equation (3.7).

$$\text{Output Size} = (n - f + 1) \times (n - f + 1) \tag{3.5}$$

$$\text{Output Size} = n \times n \tag{3.6}$$

$$p = \frac{f-1}{2} \tag{3.7}$$

Stride

It is the no. of pixel shifts over the input image. Stride is the no. of pixels by which the filters move by at a time. If the stride value increases for a given image, the size of the resulting matrix decreases with it.

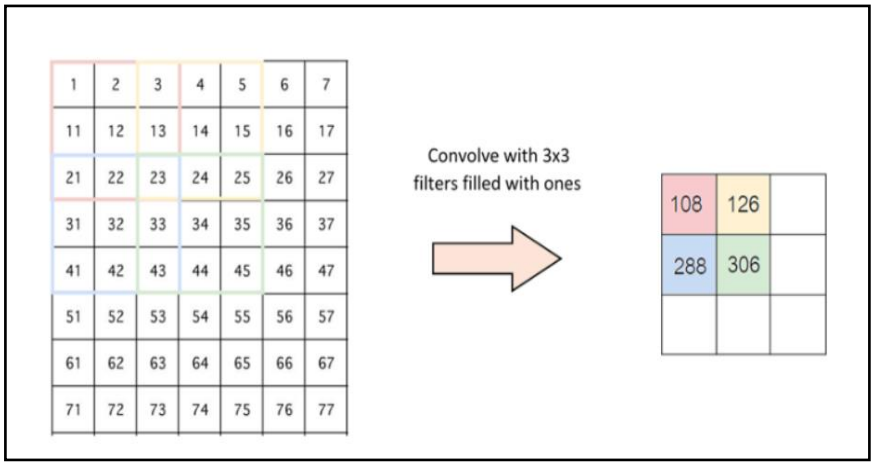


Fig. 3.5 (b) Simple Convolution using stride of 2 pixels [13]

Convolutional Layer

The task of Convolutional layers is to extract basic features like horizontal and vertical edges, area, perimeter etc. Various tasks like edge detection, sharpening, noise removal, blurring operations can be performed depending upon the filter mask taken.

$$\text{Input} = n \times n \times nc \tag{3.8}$$

$$\text{Filter} = f \times f \times nc \tag{3.9}$$

$$\text{Padding} = p \tag{3.10}$$

$$\text{Stride} = s \tag{3.11}$$

$$\text{Output} = \left(\frac{n+2p-f}{s} + 1\right) \times \left(\frac{n+2p-f}{s} + 1\right) \times n'c \tag{3.12}$$

Here, nc is the no. of channels in input and filter, n′c is the no. of filters, f is the filter size, s is the stride, p is the padding size.

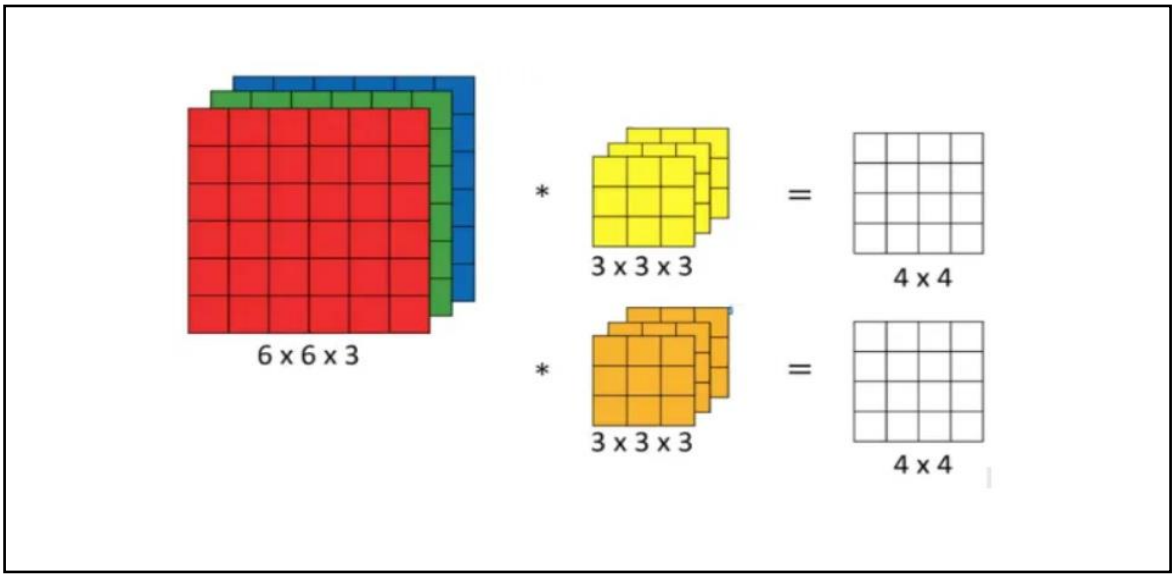


Fig. 3.5 (c) Convolutions over Volume [13]

As shown in the Figure 4.1.3, an RGB image is convoluted with 2 (3×3×3) filters and results in 2 (4×4) images. Dimension calculation can be done by the formulas shown above image. Each of the 3 channels of the filter is convoluted to its corresponding channel in the input

image and a 2D image is generated. Same is repeated for the second filter and hence the 3rd dimension of the output image depends on no. of filters.

Pooling Layer

There are 3 types of pooling: sum, average and max. It is basically used for down sampling but retaining the necessary features or information. It reduces the no. of parameters when the images are large by reducing the dimensions.

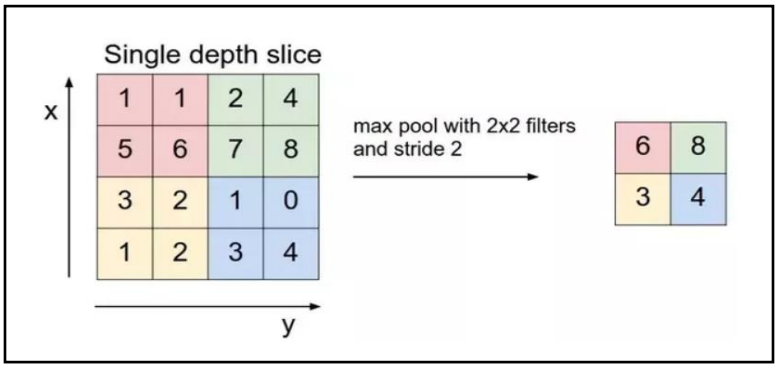


Fig. 3.5 (d) Max Pooling with stride of 2 using 2×2 filter [13]

Batch Normalization Layer

It basically normalizes the activations of each layer before providing as input to the next layer. This is done to reduce mean activation close to 0 and standard deviation value to 1. Apart from normalizing the inputs, it is necessary to normalize the outputs of each layer. This helps the network learn faster by having high learning rates. Weight initialization and Regularization are also its unique features. This overall helps to reduce the risk of overfitting.

Crop weed discrimination can be done using various ways which involve semantic segmentation, bounding box-based segmentation and classification and one-shot segmentation. All of these methods are discussed in detail below.

3.5.2 Approaches for Crop Weed Classification

Bounding Box Based Segmentation and Classification

Here, a Deep Learning based Approach allows a robot to perform an accurate weed/crop classification using a sequence of two Convolutional Neural Networks CNNs) applied to RGB images. The Network architecture shown and discussed here is taken from [9]. The

first network, based on an encoder-decoder segmentation architecture, performs a pixel-wise segmentation or semantic segmentation between vegetation and soil that enables to extract a set of connected blobs representing plant instances. Each plant is hence classified between crop and weeds by using the second network [9].

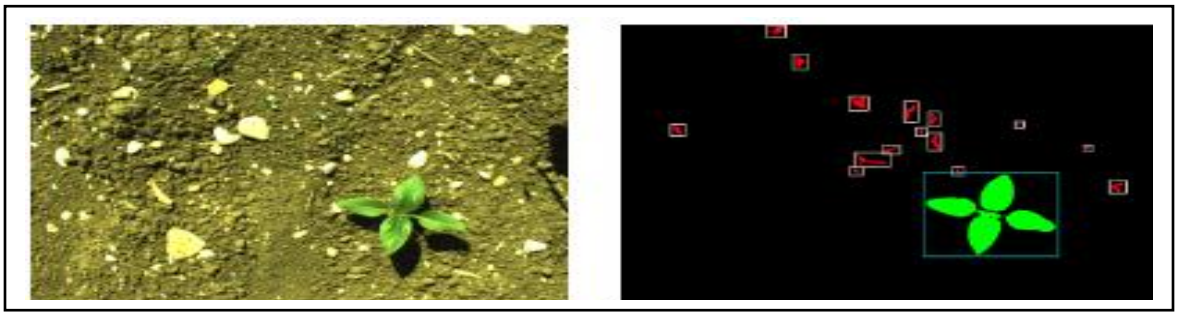


Fig. 3.5 (e) RGB image used by the robot. The second part shows label mask with bounding boxes where crops are coloured in green while weeds are in red colour [9].

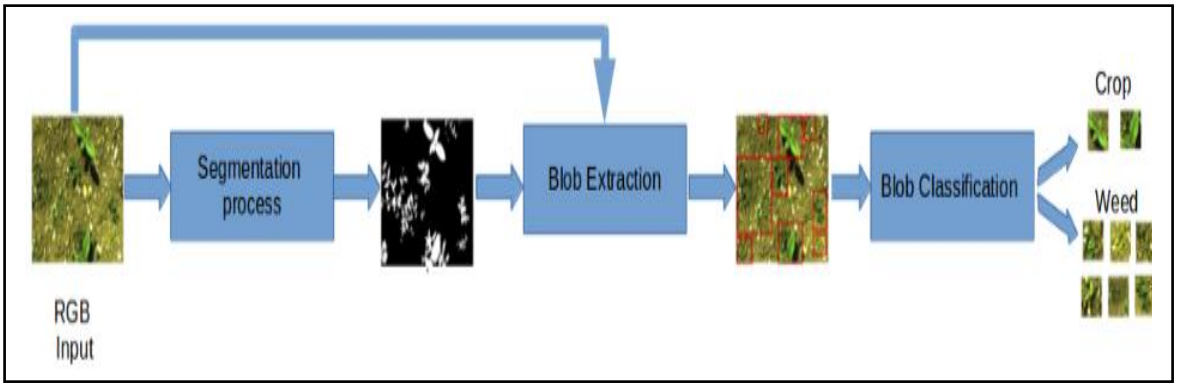


Fig. 3.5 (f) 1st step involves binary segmentation of RGB image. 2nd step involves extraction of blobs or boxes to be classified. 3rd step involves final classification of image blobs into crops and weeds [9]

Mainly this method involves 2 steps:

1. Background removal method using a deep pixel-wise segmentation to distinguish between soil and plants.

2. Deep CNN based classifier for crop/weed classification.

Basically, it reduces pixel-wise annotations of the dataset and hence giving faster operation in generation of datasets compared to pixel-wise annotated datasets.

As referred from [9], UNet semantic segmentation network, which is composed by a contracting encoder along with a symmetric expanding decoder. VGG16 is used by removing the fully connected layers and fine-tuning other layer in the implementation [9]. Expanding decoder is designed using 4 convolutional layers where each layer is composed of a batch normalization, 4 up sampling layers and a softmax pixel-wise classifiers.

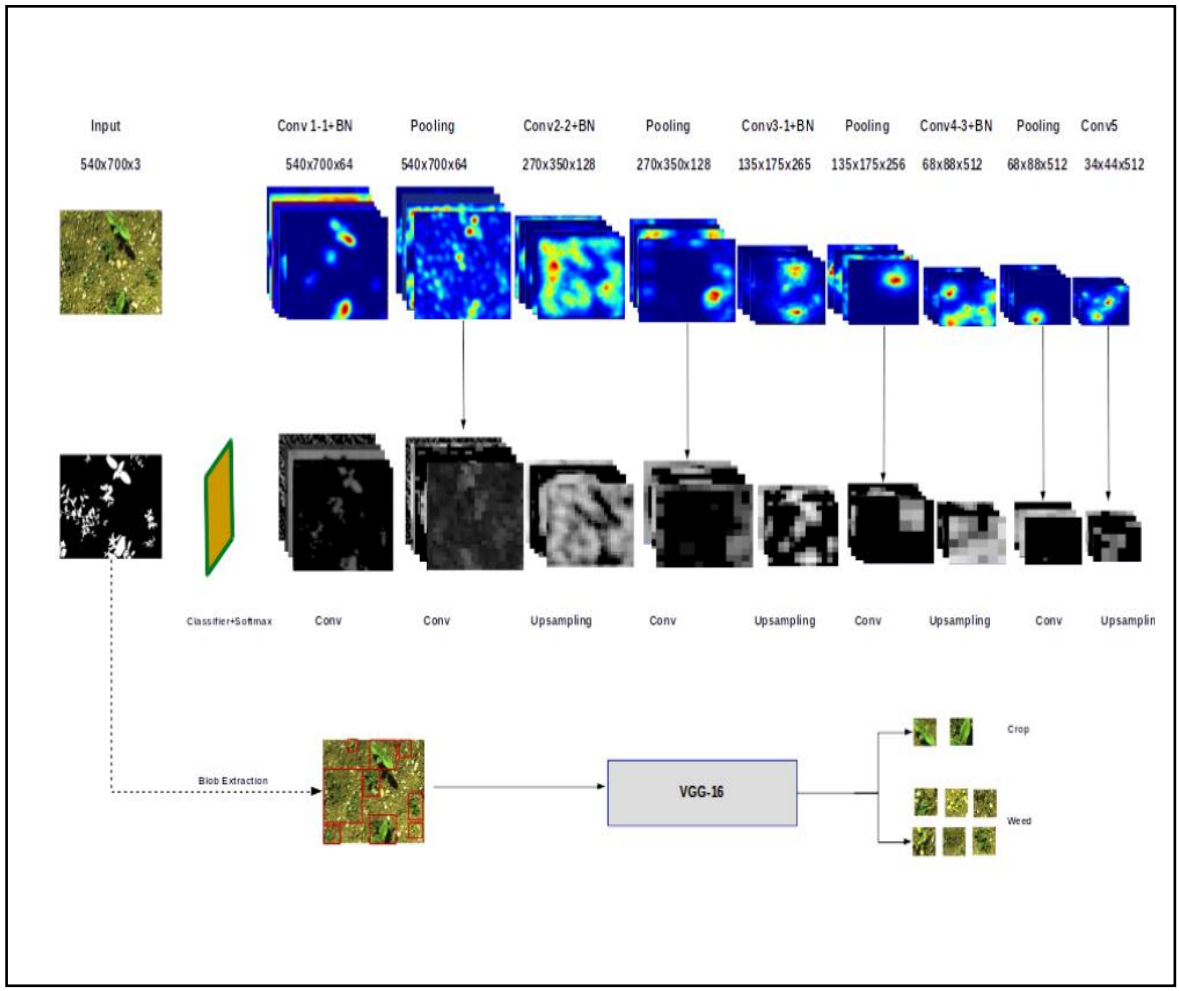


Fig. 3.5 (g) Classification Pipeline [9]

Between the contracting and expanding paths, the bottleneck is connected via a batch normalization layer and a dropout activation function. This first step allows us to train the network based on similarities between different plants instead of differentiating by using datasets which contains images of different types of fields, environmental conditions etc. The second step is the blob extraction. After obtaining the binary vegetation mask from the RGB image, morphological operator like dilation is applied to fill up the holes between

foreground regions (i.e. region with vegetation mask) and to increase its boundaries. Now, connected blobs are extracted and bounding boxes of each blob is determined in the RGB image. The third step involves the classification part. VGG16 Network architecture is used as encoder for object classification purpose. It consists of 13-convolutional layers with a 3×3 filter size. A max-pooling operation is done using a kernel of 2×2 with a stride of 2 for down-sampling. Batch normalization and a ReLU activation function are used too [9].

Algorithm Steps [9]:

1. *Input: RGB Image I_{RGB}*
2. *Result: A set of classified blobs B_C*
3. *$M \leftarrow$ Segmentation of Irgb using VGG-UNet*
4. *$C \leftarrow$ Contour Extraction (M) > set of contours belonging to connected regions*
5. *for i in range len(C) do*
6. *$B[i] \leftarrow BoundRect(C[i])$ > $B_M[i]$ is the bounding box around contour i*
7. *$B_{RGB}[i] \leftarrow (I_{RGB} \cap B_M[i])$ > $B_{RGB}[i]$ is the corresponding bounding box from RGB image.*
8. *$B_C[i] \leftarrow$ (classify $B_{RGB}[i]$ using VGG-16 into weed or crop)*
9. *end for*

Semantic Segmentation:

It basically is a task of classifying every pixel in an image into a class. However, it is different from instance segmentation which classify different objects of same class into different

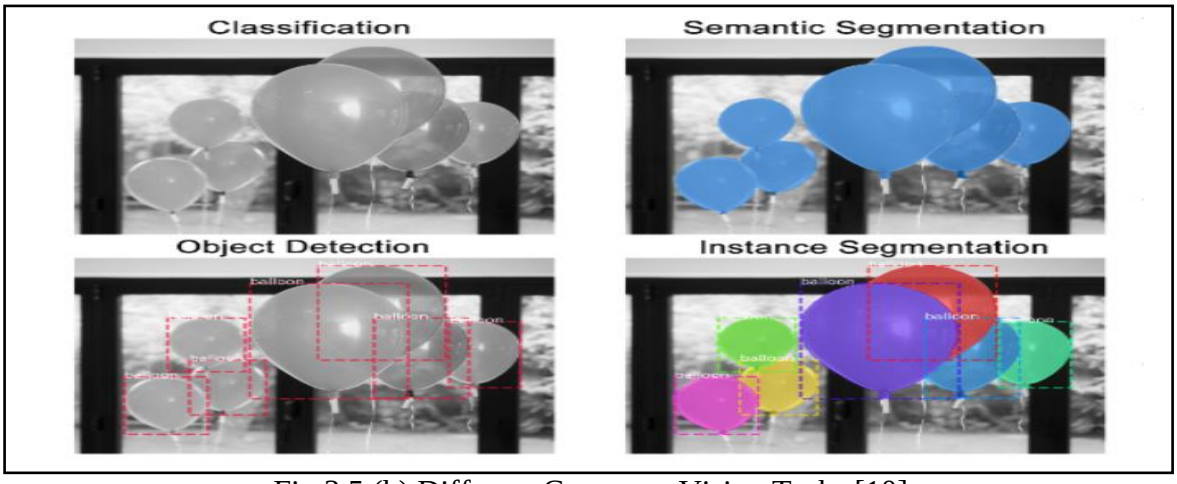


Fig.3.5 (h) Different Computer Vision Tasks [10]

Semantic Segmentation is chosen over the bounding box technique due to the following reasons.

1. Through Semantic segmentation more information is available as it's a pixel-wise segmentation and it's easy to perceive the information.
2. Bounding Box labelling for the dataset.
3. Availability of pixel-wise segmented datasets and fine classification of multiclass species.

One of the segmentation architectures is shown below here is UNet.

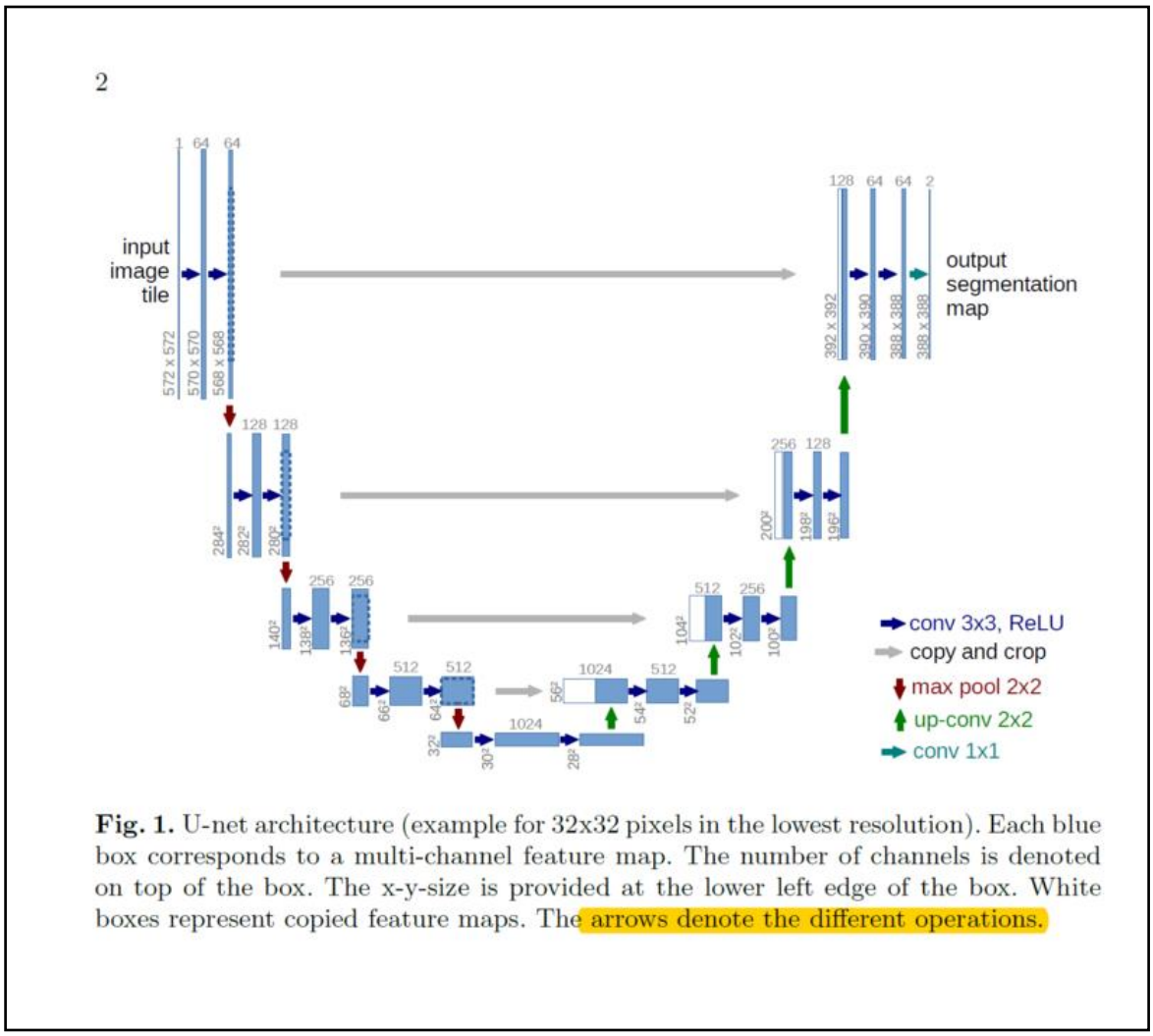


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Fig. 3.5 (i) UNet Architecture Top: No. of channels, Bottom: X-Y size [9].

The architecture can be broken in 2 parts: Encoder & Decoder. Encoder basically downsamples the image retrieving the important and contextual features of the image. It is basically stack of convolutional, max pooling followed by batch-normalization layers. The second path consists of symmetrical expanding part which is known as decoder which is used to enable precise localizations using up sampling by restoring the spatial dimensions of the image same as input.

*****

ALGORITHMS IMPLEMENTATION CHAPTER-4

A number of algorithms have been developed for autonomous traversing and monitoring of outdoor environment in agricultural domain. Majorly these algorithms consist of Kalman filtering & estimation of robot's position, mapping, path planning, classification algorithms for weed and crops etc. In this chapter, we have reported three algorithms which we have experimented for AGRIBOT in simulation.

- Filter for heading correction

- Trajectory Planning from starting to destination Point

- Crop & Weed Classification Model (Bonnet)

We have also included results in further section to discuss.

4.1 Filtering and Sensor Fusion

4.1.1 Filtering Magnetometer data:

Most of the algorithms fail to perform as expected due to high amount of noise in sensor raw data. So, first task performed was: filtering noise out of sensor data from magnetometer, so that we could estimate exact angle robot have to rotate before starting towards destination. Based on data shown in previous section, we have decided to implement moving median filter. Moving median filter works good for single type of data with single device. Statistically the moving average is optimal for recovering the underlying trend of the time series when the fluctuations about the trend are normally distributed. This filter is also expected to sustain spikes better than normal moving mean filter. Secondly, we have also tried the single dimensional Kalman filter to remove noise from magnetometer data.

1) Moving Median Filter

In filtering steps, data from magnetometer first stored in list of length n. From this list median is calculated and stored as a filtered output for further calculations. Now current list is updated using median and oldest data

$$data_t = \text{Median } (dm, dm\text{-}1, dm\text{-}2, \ldots\ldots, dm\text{-}n+1) \quad (4.1)$$

$$out(ø) = \text{Mean } (data_t + data_{t-1}) \quad (4.2)$$

Filtering Steps:

1 . Keep a running array of most recent n sensor values.

2 . Calculate the median of the n stored values.

3 . Use that median with previous median to calculate average value

4 . Store corrected values in previous median variable

5 . Publish final value (ø).

Below fig 4.1 (a) shows implementation of filter on simulated data taken from Gazebo.

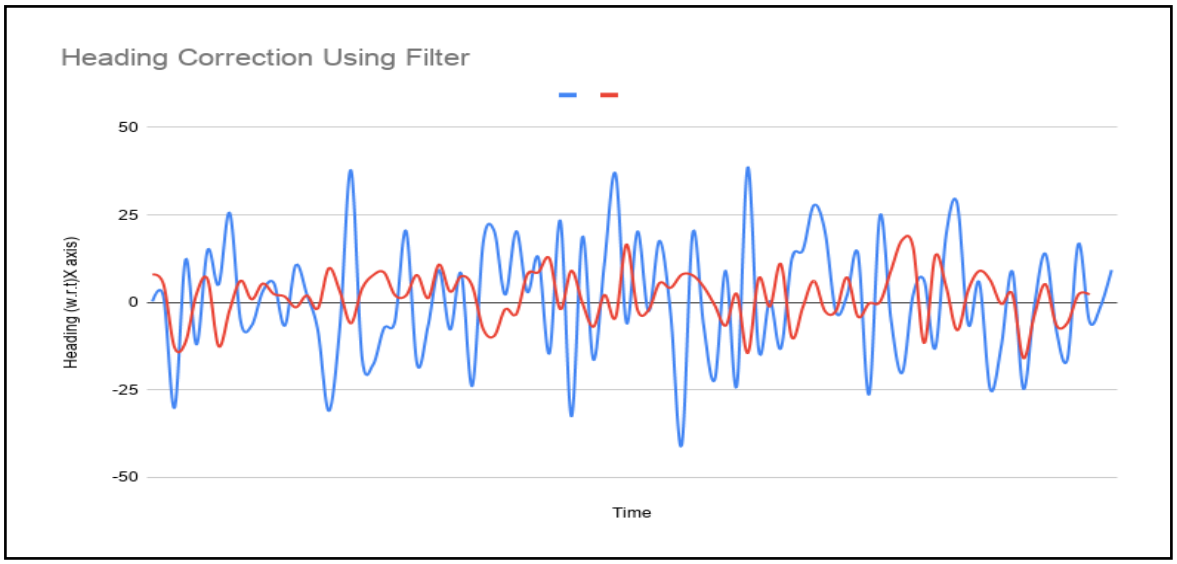


Fig 4.1 (a) Implementation of filter on data taken from Gazebo Simulation. Blue Colour represents Raw Data and Red Colour represents filtered data.

2) Single Dimensional Kalman Filter.

As we saw in previous section about noisy sensor data, it can be improved using approximations methods. Kalman filter works effectively with the uncertainty due to noisy data and some other external factor. The Kalman filter produces an estimate of the state of the system as an average of the system's predicted state and of the new measurement using a weighted average. The purpose of the weights is that values with better (i.e., smaller) estimated uncertainty are "trusted" more. The weights are calculated from the covariance, a measure of the estimated uncertainty of the prediction of the system's state. The result of the weighted average is a new state estimate that lies between the predicted and measured state, and has a better estimated uncertainty than either alone. This process is repeated at every time step, with the new estimate and its covariance informing the prediction used in the following iteration. This means that Kalman filter works recursively and requires only the last "best guess", rather than the entire history, of a system's state to calculate a new state. Figure 4.1 (b) shows the implementation of Kalman filter on simulated data taken from Gazebo.

Filter involves steps:

1. Initialize with error in estimation & error in measurement due to external factors
2. Calculate Kalman gain 'KG' as in equation 4.3.
3. Take data from raw measurement.
4. Now calculate estimated value of data using previous estimation and KG as in equation 4.4.
5. Calculate error in estimation for current iteration as in equation 4.5.
6. update error in estimate, previous error in estimation and previous estimation variable to repeat process.

$$\text{KG} = \frac{error\_est}{(error\_est + error\_mea} \tag{4.3}$$

$$\text{estimate\_data} = pre\_estimate + KG \times ((-1) \times RawHeading - pre\_estimate) \tag{4.4}$$

$$\text{error\_est} = (1 - KG) \times (pre\_error\_est) \tag{4.5}$$

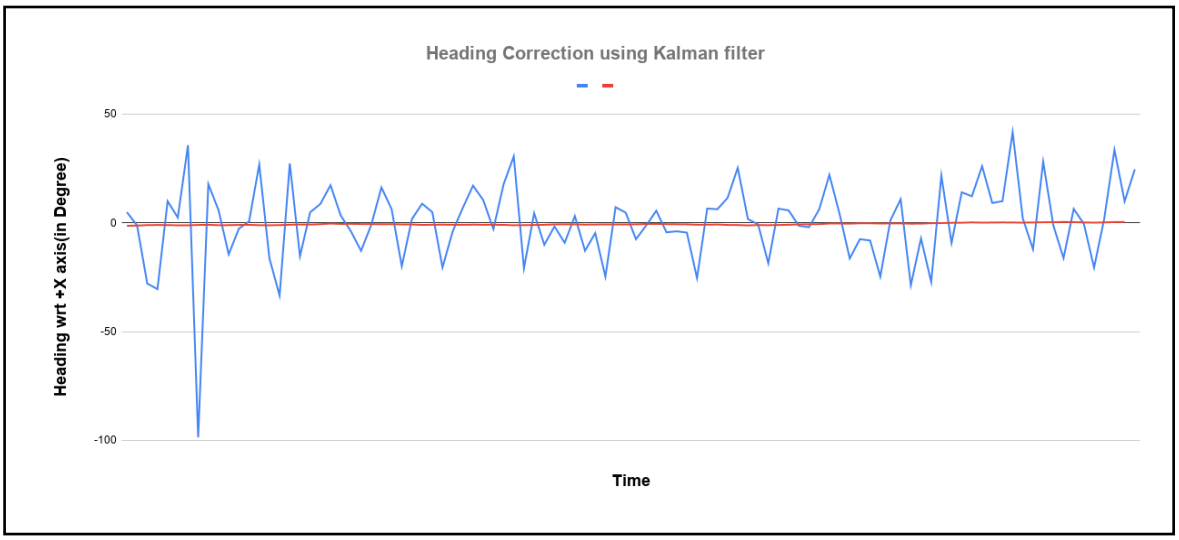


Fig. 4.1 (b) Implementation of Single dimensional Kalman Filter. Blue Colour represents Raw Data and Red Colour represents filtered data.

4.1.2 Calculation of parameters- Required angle(θ) and distance(d) to Destination

$\theta = \text{atan2}\,(Y'_1 - Y', X'_1 - X') \pm ø$ (4.6)

$d = ((X'_1 - X')^2 + (Y'_1 - Y')^2)^{\frac{1}{2}}$ (4.7)

Here, ø is the angle obtained from magnetometer data as mentioned in 4.1.1. As in the above equations, θ is calculated which is the required angle to rotate to align the robot's heading towards the destination.

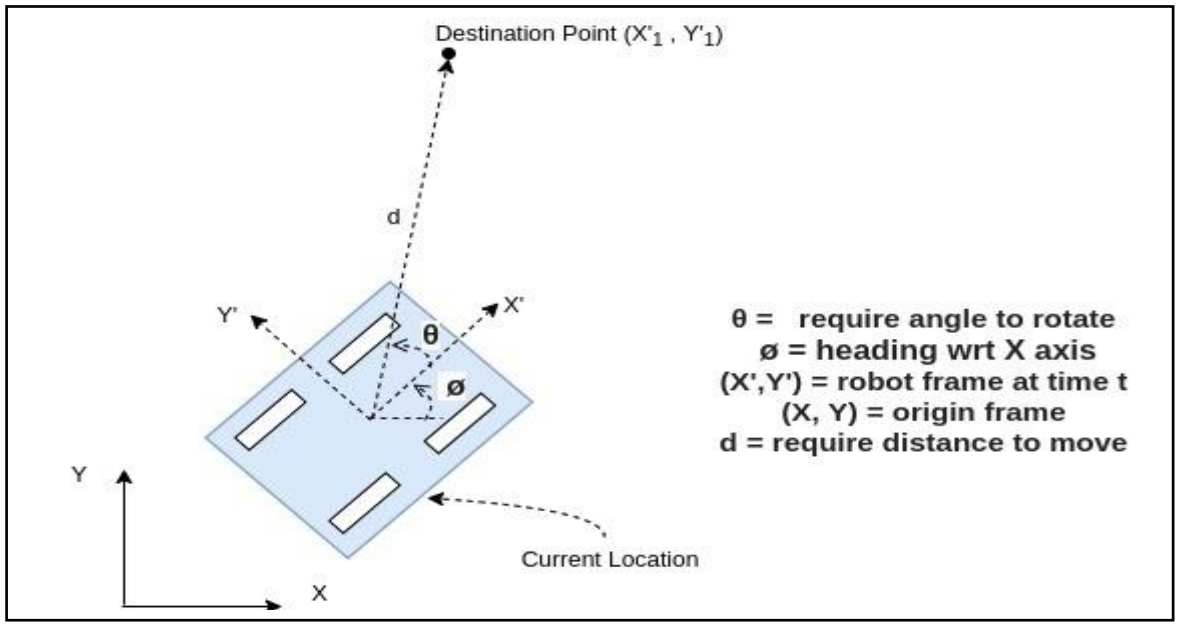


Fig. 4.1(c) Calculation of the required angle and distance to destination.

Required distance d to the destination is calculated simply by taking Euclidean distance between present position (X', Y') of the robot and destination coordinates (X'_1, Y'_1). The present position of the robot (X', Y') is regularly updated from GPS data.

4.2 Control and Navigation Algorithm

This node is responsible to drive the robot with a desired input speed. Through inverse kinematics desired speed information in the local frame in the form of linear and angular velocities are received and controller outputs driving speed for each wheel. These outputs are set points for the relevant ROS controllers to control the position, velocity or effort on each joint.

The AGRIBOT is designed to traverse according to

1. Required angle θ calculated from magnetometer and GPS readings (refer section 4.1.2)

2. Required distance d calculated to reach destination (refer section 4.1.2).

According to above points desire velocity is calculated and fed into the controllers

Steps for planning:

1. Get /distance and /angle data from sensors

2. Calculate error between desire angle and current angle wrt X axis

3. Now, Trajectory Planner implements PD controller on error in required angle θ and distance d to destination. If the (error in angle θ > threshold), first rotate the robot by only providing angular velocity and making linear velocity = 0.

4. Send Linear & Angular Velocities obtained by PD Controller to Skid steer controller.

5. The skid steer controller now calculates velocities of each wheel through inverse kinematics. Now, drive the robot with the required velocities.

6. Repeat the steps until the destination is reached.

Following block diagram shows pictorial view of whole path planning algorithm.

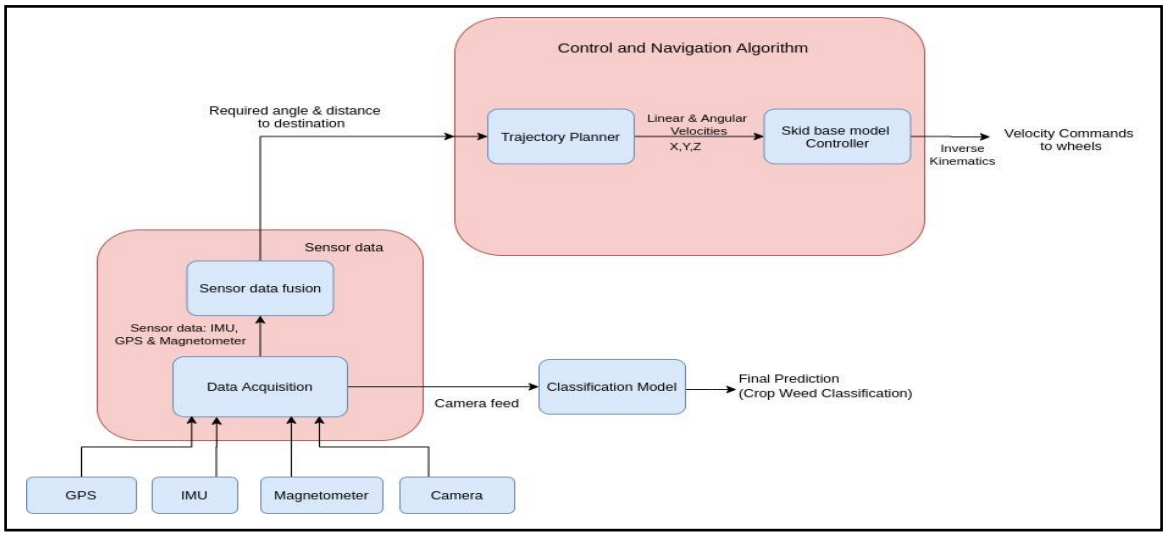


Fig. 4.2 (a) Pictorial View of Path Planning Algorithm.

4.3 Crop Weed Classification Models

4.3.1 Unet

The initial path is the encoder part which basically downsamples the important and contextual features of the image. It is basically a stack of convolutional, max pooling followed by batch-normalization layers. The second path consists of a symmetrical expanding part which is known as a decoder which is used to enable precise localizations using up sampling (transposed convolutions).

No. of trainable parameters: 23,54,785.

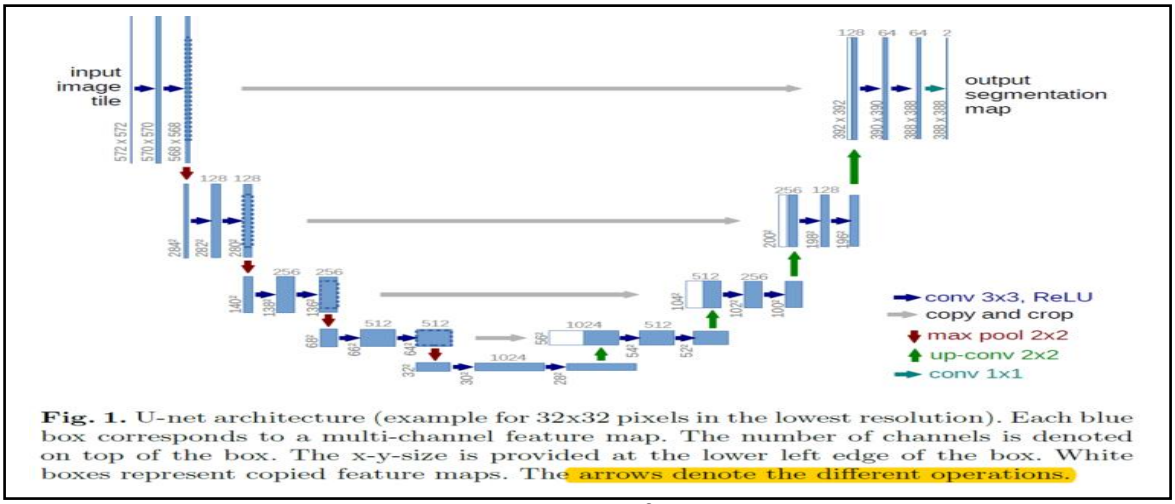


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Fig. 4.3(a) UNet Architecture Top: No. of channels, Bottom: X-Y size [9].

Convolutional, max pooling and batch normalizations are described in 4.1. Here, the up convolutions represent the transposed convolutions which up samples the image to the input dimensions.

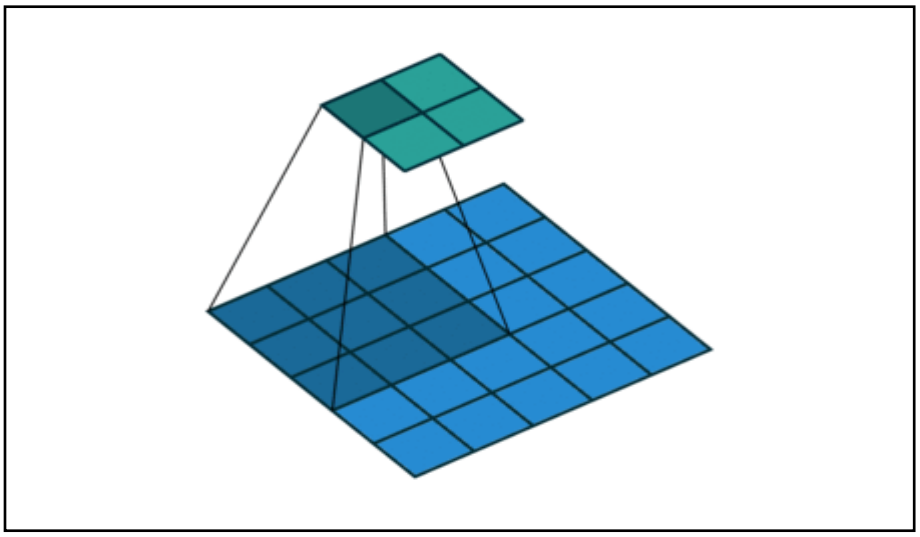


Fig. 4.3(b) Transposed Convolution. Input: 2×2, filter size: 3×3, output: 5×5 [1]

It is basically deconvolution which is being performed using convolution but increases the output dimensions.

4.3.2 Bonnet

It is also an end-to-end encoder-decoder semantic segmentation network. It is designed to meet target speed and efficiency as it targets narrower classes compared to general architectures which target over thousands of classes and may have millions of parameters. This bonnet architecture is computationally efficient for our task as it contains 30,000 parameters which is much lesser than other general architectures.

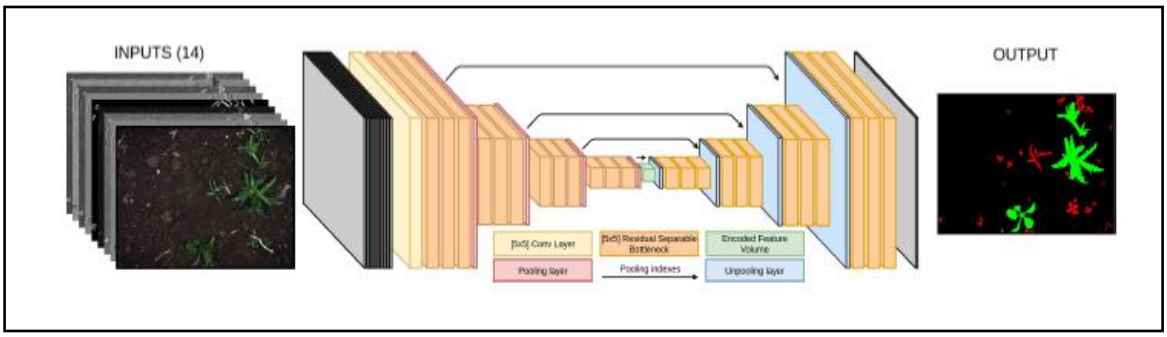


Fig. 4.3 (c) Bonnet Architecture [8].

The input to the model consists of multiple channels such as RGB, HSV, Excess Green (ExG), Excess Red (ExR), Color Index of Vegetation Extraction (CIVE), Normalised Difference Index (NDI) and is of 512×384 size. These multiple channels leads to learning of generalized and wide variety of features regardless of type of crop, lighting conditions etc. The model consists of less than 30,000 parameters.

No. of parameters(approx): < 30,000

Model architecture consists of Convolutional layer, Residual Separable Bottleneck Layers and Unpooling layer with shared indices.

Convolutional Layer: It consists of 5×5 convolutional layer followed by a Batch Normalization layer. Batch Normalization operation allows higher learning rates and better generalization over input. Activation function used is ReLU and zero padding is used along to avoid washing if edge information.

Residual Separable Bottleneck: It is built upon the ideas of 1.) residual connections 2.) bottlenecks 3.) separating convolutional layers. The residual connections solve the problem of vanishing gradients and help in training deeper networks.

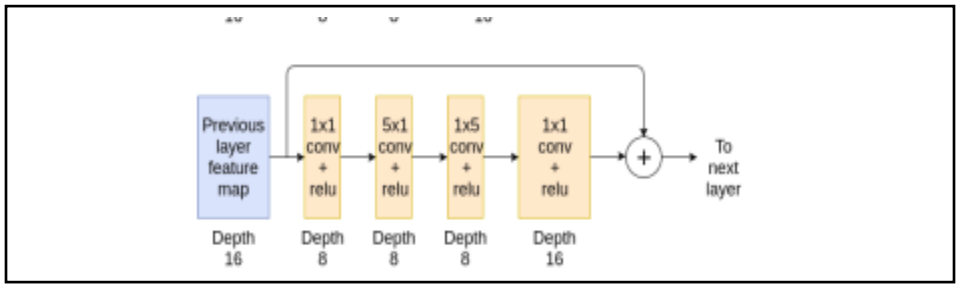


Fig. 4.3 (d) Residual Block [8]

The 1×1 convolutional layer reduce the depth by ½ so that computationally intensive 5×5 convolution which is done by 5×1 and 1×5 is not run over the whole depth. Again 1×1 convolutional layer is added at the end to increase/ expand the depth of the layer same as input. This reduces the GFLOPs from 6400 to 896 FLOPs drastically and hence reducing the parameters from 6400 to 896.

Unpooling with shared indices: The pooling indices in the encoder part are shared by symmetrical unpooling operation in the decoder. It allows to obtain spatial information of maximum activations in encoder part without implementing expensive transposed convolution. Pooling operation is [2×2] with a stride of 2.

Output: Last layer is a linear operation following a softmax activation of vector of length 3 per pixel corresponding to soil, crop and weed.

4.3.3 Segmentation Technique and Architecture Chosen

Semantic Segmentation is chosen over the bounding box technique due to the following reasons.

1. Bounding Box technique suffers from localization problem in case of overlapping.
2. Bounding Box labelling for the dataset.
3. Availability of pixel-wise segmented datasets and fine classification of multiclass species.

Segmentation Technique chosen: Semantic Segmentation.

Model Architecture chosen: Bonnet.

Datasets: **CWFID** by Sebastian Haug, Jörn Ostermann & **Bonn** by University of Bonn

Crop: - Carrot (CWFID)

Robot used to collect the dataset: - BONIROB.

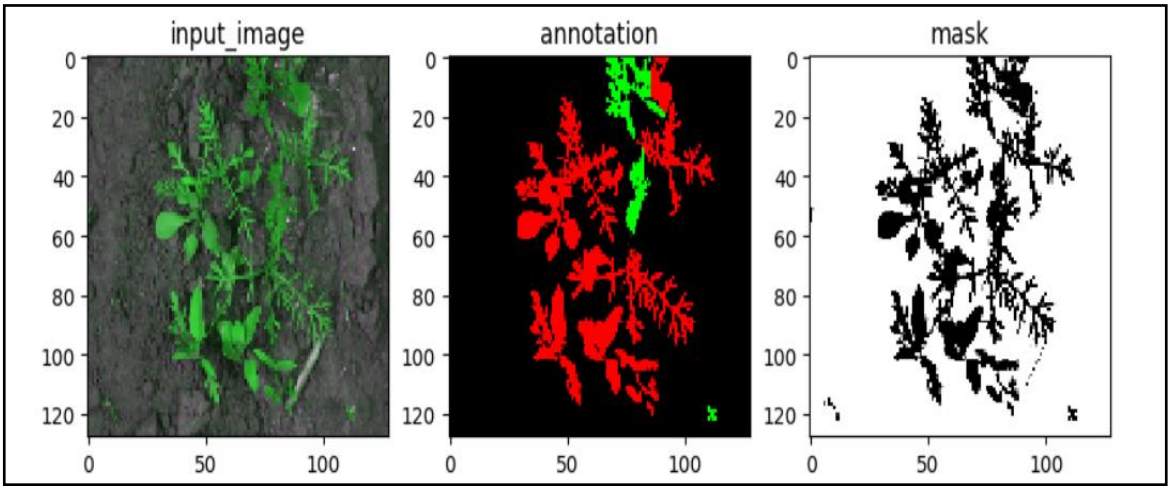


Fig. 4.3(e) Image from the CWFID dataset [10].

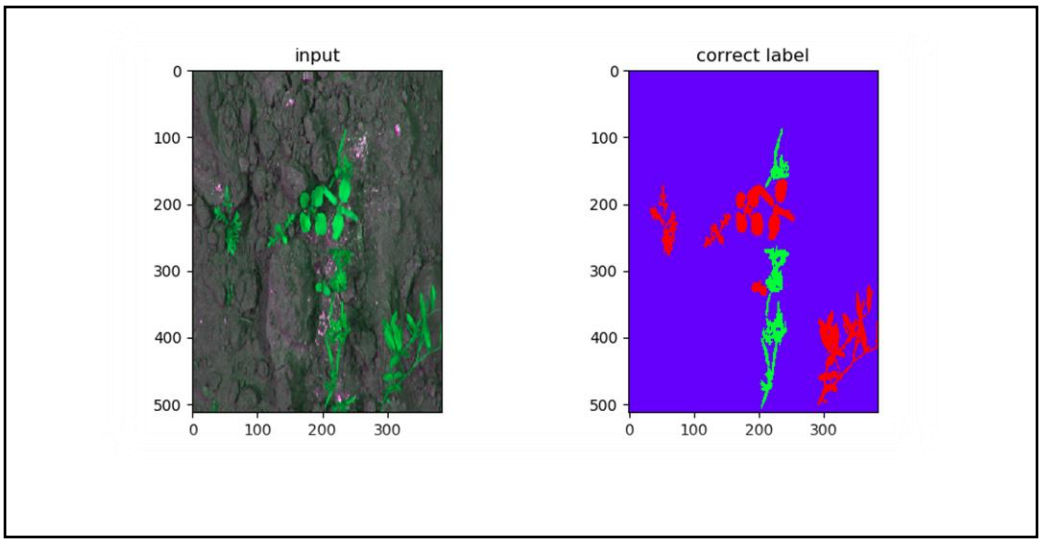


Fig. 4.3 (f) Image from Bonn dataset [24]

The no. of parameters in Bonnet is lesser by 100 times than UNet architecture. UNet architecture is generalized for semantic segmentation for over 1000+ classes. Whereas Bonnet architecture is designed specifically for faster and real-time operation and can perform really well on narrower space of classes.

Table 4.3 (a) Object-wise Test Performance. Trained in 70% Bonn, reporting 15% held-out Bonn, 100% Stuttgart and 100% Zurich from [8].

Dataset	Network	mAcc[%]	Precision [%]		Recall [%]	
			Weed	Crops	Weed	Crops
Bonn	I_{RGB}	86.84	83.63	81.14	91.99	80.42
	I_{RGB} + I_{NIR}	93.72	90.51	**95.09**	**94.79**	89.46
	I_1…. I_{14}(ours)	**94.74**	**98.16**	91.97	93.35	**95.17**
Zurich	I_{RGB}	45.51	59.75	19.71	45.52	23.66
	I_{RGB} + I_{NIR}	68.03	67.41	46.78	**65.31**	49.32
	I_1…. I_{14}(ours)	**72.08**	**67.91**	**72.55**	63.33	**64.94**
Stuttgart	I_{RGB}	46.05	42.32	42.03	46.10	25.01
	I_{RGB} + I_{NIR}	73.99	74.30	**70.23**	**71.35**	53.88
	I_1…. I_{14}(ours)	**76.54**	**87.87**	65.25	64.66	**85.15**

Table 4.3 (b) Bonnet's real-time performance on different devices. [8]

Input	FLOPS	Hardware	Preproc.	Network	Total	FPS
RGB	1.8G	i7+GTX1080Ti Tegra TX2 SoC	- -	31ms 190ms	31ms 190ms	32.2 5.2
All	2G	i7+GTX1080Ti Tegra TX2 SoC	6ms 6ms	038ms 204ms	44ms 204ms	22.7 4.7

Also due to it's multi-channel input, it tends to generalize more and perform much better over wider variety of data. It's capable of running at 4.7 FPS on Nvidia Jetson Tegra TX2. Also, in terms of performance it achieves mean accuracy of 94.74 % on Bonn dataset by University of Bonn. [8]

4.3.4 Performance Comparisons

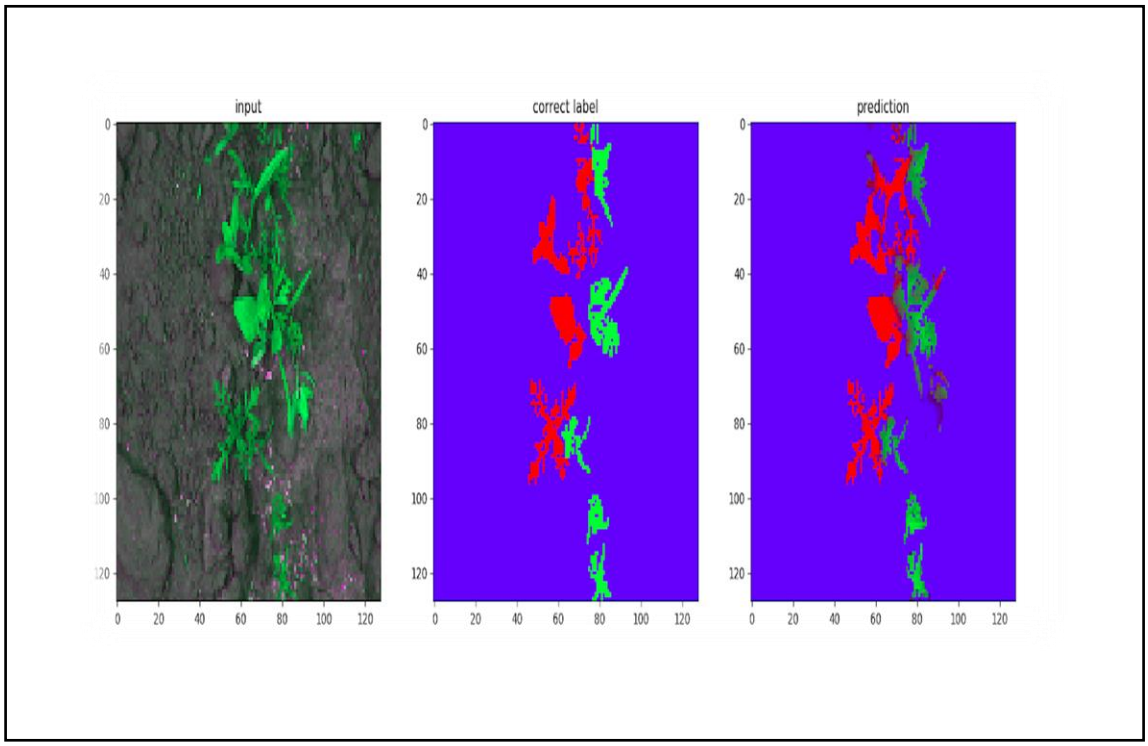


Fig 4.3(g) UNet's prediction on CWFID.

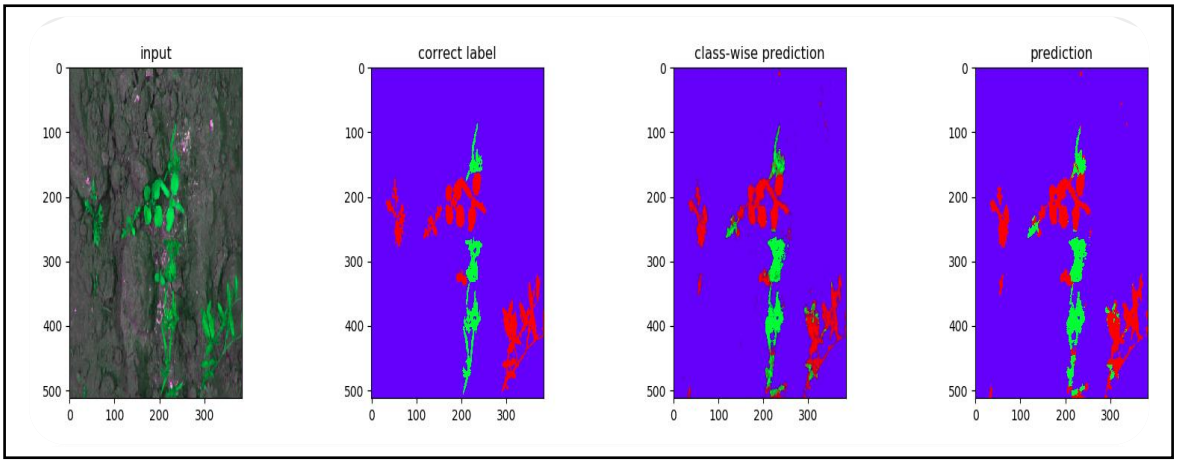


Fig 4.3(h): Bonnet's prediction on CWFID

Metrics Comparisons on CWFID Dataset:

Table 4.3 (c) Metrics Comparisons on CWFID Dataset.

Model	**Accuracy**	**Loss**
Bonnet	Train: 96.73% Test: 96.48%	Train: 0.01153 units Test: 0.0168 units
Unet	Train: 97.65% Test: 97.20%	Train: 0.0549 units Test: 0.0499 units

Looking at the metrics values of both models which are comparable, but in terms of class-wise predictions as shown from Fig. 4.4(a) and Fig. 4.4(b), Bonnet outperforms UNet in the same. Bonnet produces a better prediction result close to the ground truth.

The results shown here are obtained from training on CWFID which is a small dataset. But the results obtained are helpful in choosing the architecture for further training it into larger datasets and testing it into real-world images which require a high-end GPU.

Hence, we conclude from the results and choose Bonnet architecture for our further training on larger dataset of Bonn as it is computationally feasible for real-time and it's high performance on a narrower space of classes.

Now, further task is to train the Bonnet Architecture using Bonn dataset. There are a lot of loss functions for the semantic segmentation task on which the model can train on. We have tried some loss functions and figured out which works out best for our task.

Loss Functions:

1) Categorical cross-entropy Loss

It will compare the distribution of prediction in the output layer (1 for each class) with the ground truth, where the probability of true class is set 1 and 0 for other classes. Putting up in a different way, it is a one-hot encoded vector.

$$L\ (y, \hat{y}) = -\sum_{j=0}^{M} \sum_{i=0}^{N} (y_{ij} * \log(\hat{y}_{ij})) \qquad (4.5)$$

Here, y is the ground truth while that is the predicted value which is summed over all classes and averaged over the entire batch.

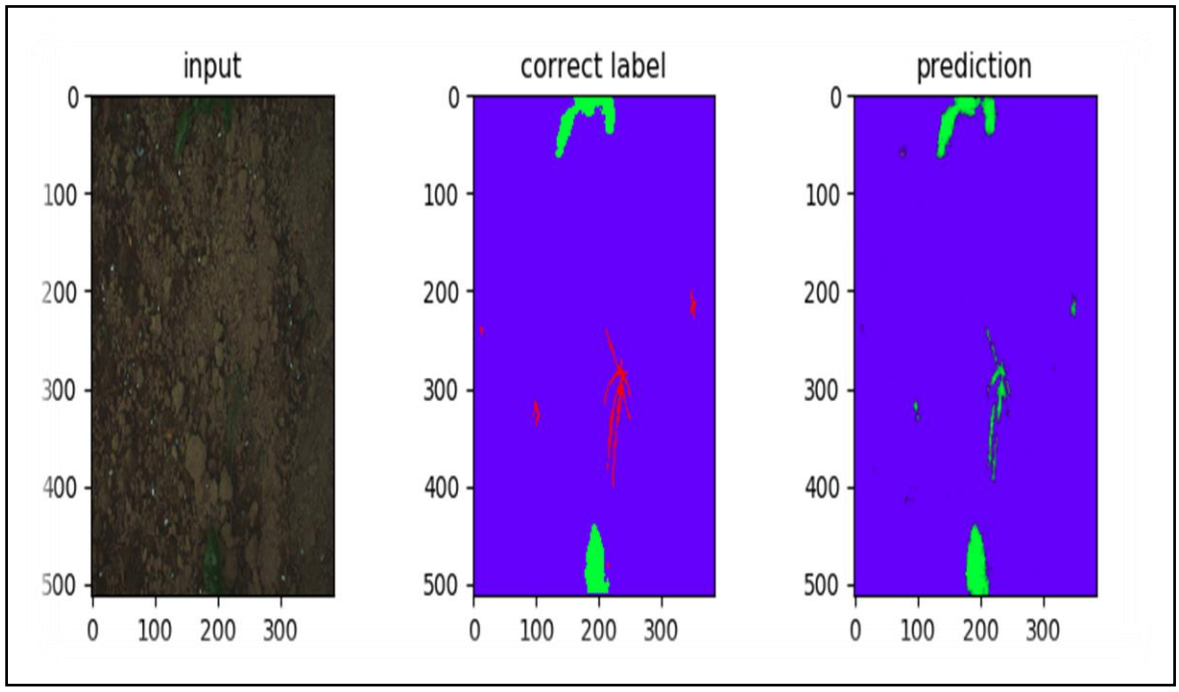


Fig. 4.3 (i) Bonnet's performance with Categorical cross-entropy on Bonn

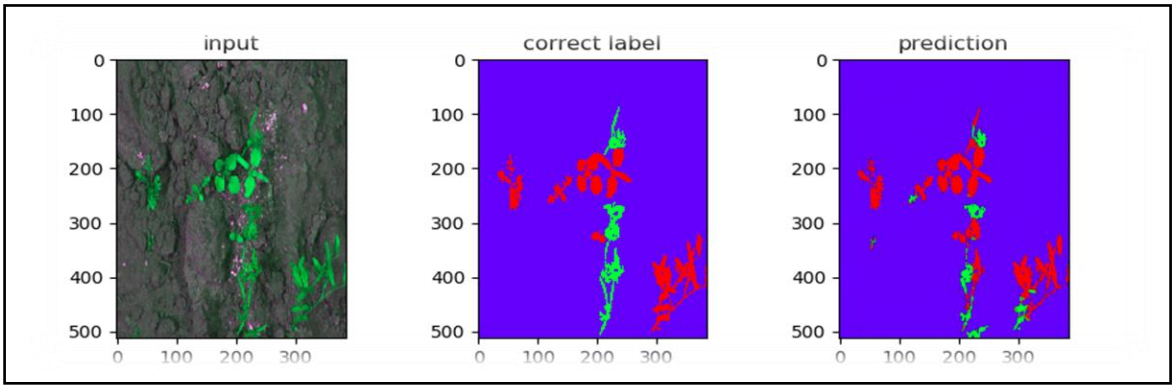


Fig 4.3 (j). Bonnet's performance with Categorical cross-entropy on CWFID

From the above predictions, it is clear that the loss function is more biased towards the class which is in majority in the image and any false predictions on classes with lesser frequency are weighed less. Hence, in the predictions, the pixels with minority classes in ground truth are predicted as majority classes as they were weighed more in the loss function. Hence, the above loss function is not effective.

2) Dice-Coefficient Loss:

It is a measure of overlap and is a popular function in image segmentation tasks. Dice coefficient ranges from 0 to 1, where 1 giving maximum or full overlap. Here p is ground truth and p is the prediction.

$$DL\ (p,p) = 1 - \frac{2pp+1}{p+p+1} \qquad (4.6)$$

Where $p \in \{0,1\}$ and $0 \leq p \leq 1$.

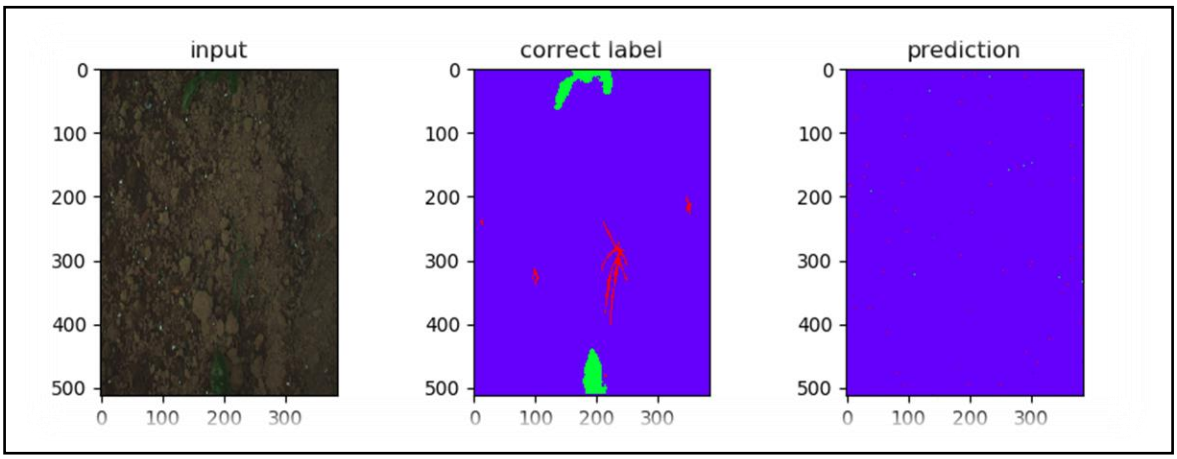


Fig 4.3 (k) Bonnet's performance with dice-loss on Bonn

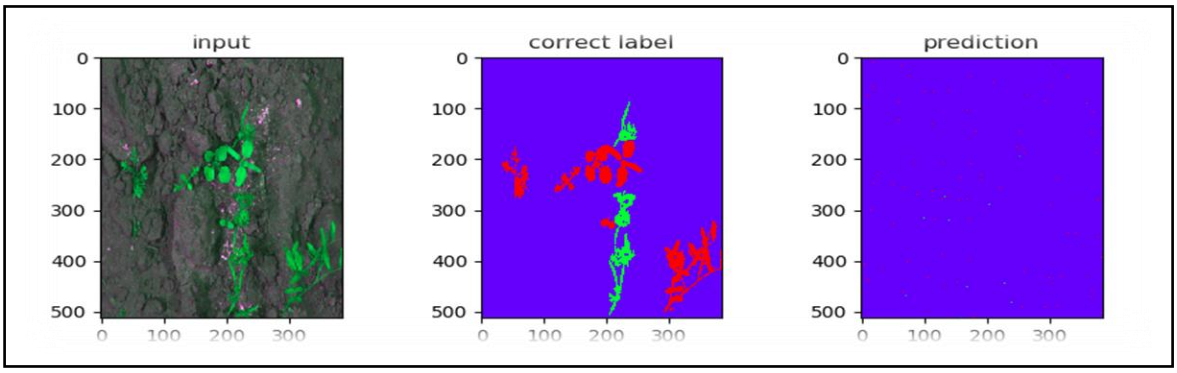


Fig 4.3 (l) Bonnet's performance with dice loss on CWFID

Due to some unknown reason, the dice-loss function does not seem to be working and the model predicts every pixel as soil class. Hence, it can't be used to train and optimize the final version of the model.

3) Weighted Categorical cross-entropy (WCCE) Loss:

From the conclusion drawn by using Categorical cross-entropy loss, it is clear that Categorical cross-entropy suffers from loss that weighs more on the major class in the image. To counter this, WCCE loss is used. Here, the losses from corresponding classes are weighed in inverse proportion to it's frequency of occurrence and summed up across all classes.

class_weights = [0.90,0.11,0.1]

Here, the above weight matrix is in order of Weed, Crop and Soil. Weed is given higher weight, Crop lower than weed and soil the least. These weights are given in inverse proportion of its frequency of occurrence in images to counter the problem by Categorical cross-entropy. These weights are multiplied with loss corresponding to each class and summed over all classes.

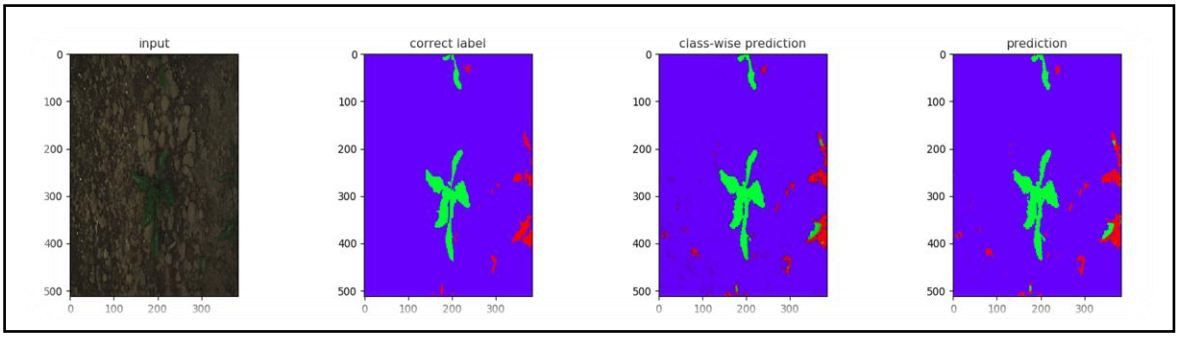


Fig. 4.3 (m) Bonnet's performance trained using WCCE on Bonn

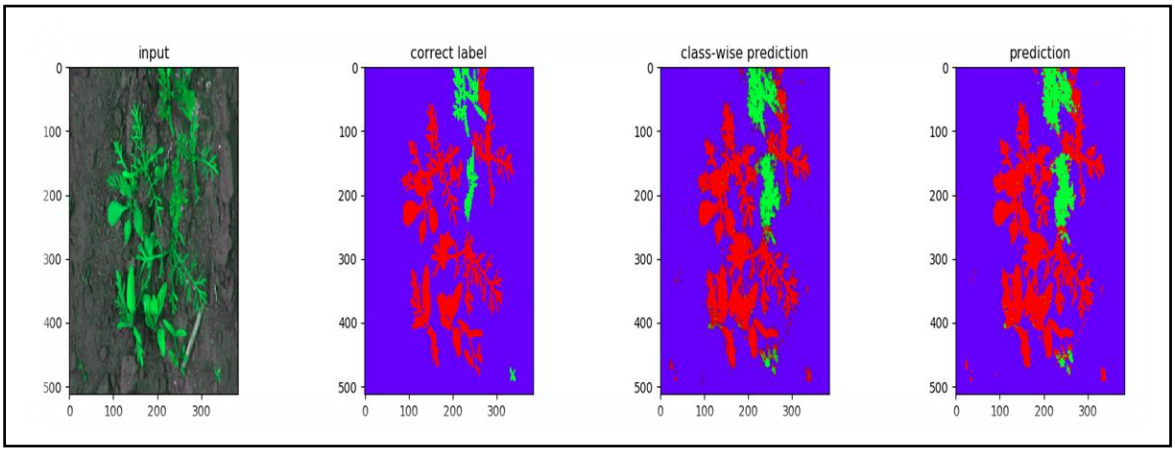


Fig. 4.3 (n) Bonnet's performance trained using WCCE on CWFID.

From the above figure, it can be seen that the class-wise prediction and prediction of the model is pretty close to the ground truth and the model performs better with this loss function than the loss functions discussed previously.

Loss Function used: WCCE

*****

RESULTS CHAPTER-5

This section provides the experimental results and performance of our robot through ROS third party software and custom open source tools. For some scenarios we have given input velocity from the keyboard to ensure functionality of the robot. The results from the experiment were recorded in a log file for further visualization with live performance, hence graphs were plotted.

5.1 Teleoperation

As from the below Fig 5.1, the smooth trajectory of the AGRIBOT is plotted and it can be seen that it is able to navigate through the crop rows in the field through remote or teleoperation with the help of camera vision.

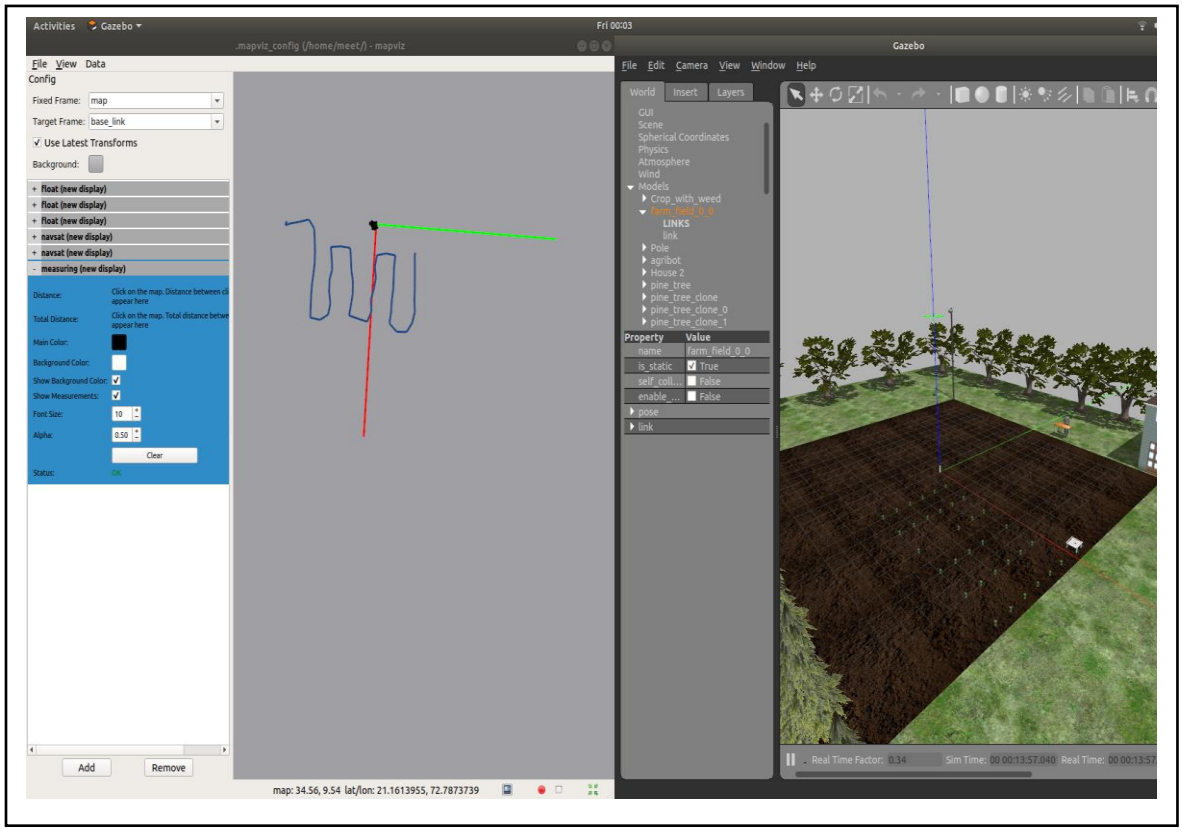


Fig. 5.1 Field traversing through teleoperation.

The Blue Line is the plotted trajectory, the Black Cluster is the starting point of field and on the left we have mapviz visualization and on the right, we have Gazebo simulator.

5.2 Autonomous Monitoring

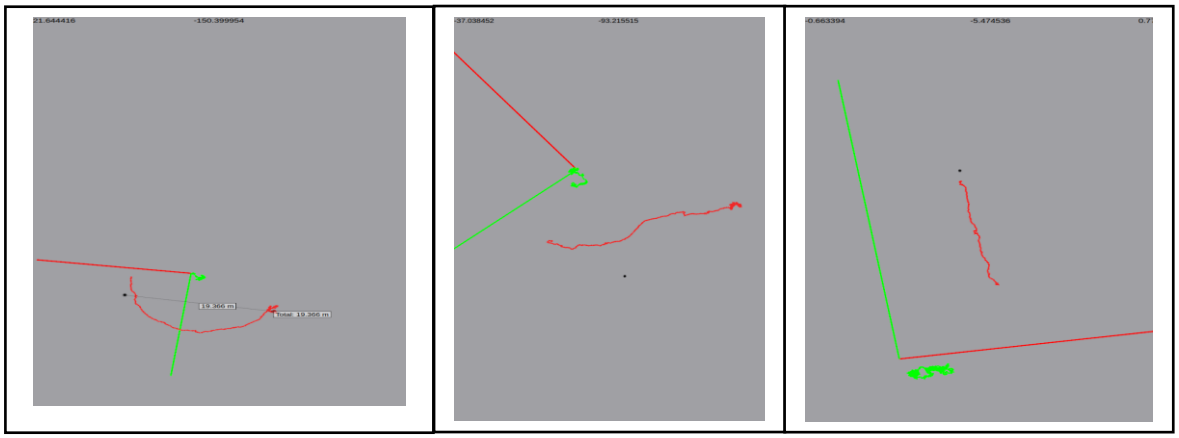


Fig 5.2(a): Testing of Trajectory Planner

The above fig 5.2(a) describes the Tests of the Trajectory Planner algorithm. The left case shows that if we control only one motion (i.e. linear or rotational) at a time then the resultant trajectory may fail to reach the destination point and the traversed trajectory may be a curve and not a straight line taking up more time to reach the destination. The middle case also describes the same but for a shorter distance. The right case shows that simultaneous control of both rotational and linear results in reaching the destination successfully and in the shortest path.

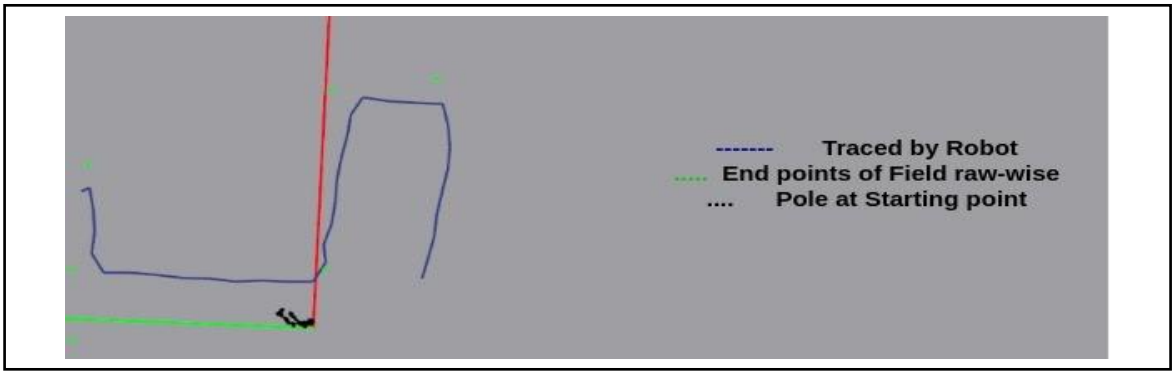


Fig. 5.2(b) Autonomous traversing of crop rows in the field

The right most point in the traced blue line is the starting point and the green points are the end points of field row-wise. Here, it is shown in the above fig that the AGRIBOT takes the turn before reaching the end points of field which can be tuned. For now, it's set to 1m which means the robot will take turn from the end points far by 1m.

5.3 Predictions on CWFID and Bonn Datasets

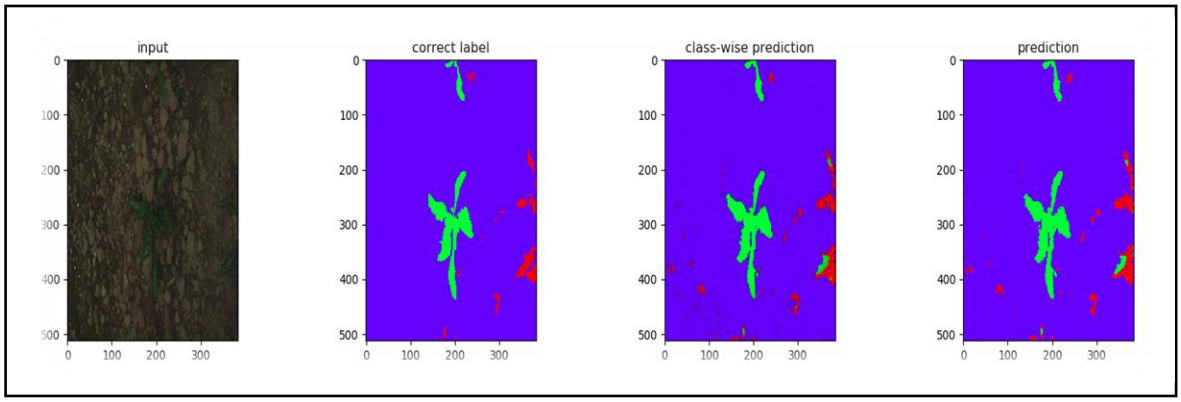


Fig. 5.3 (a) Prediction on Bonn dataset.

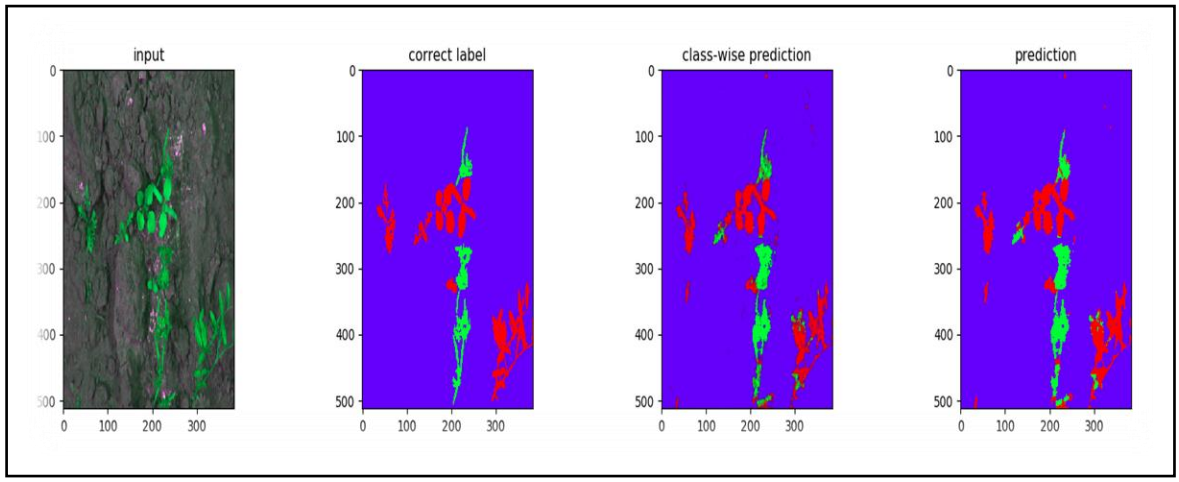


Fig. 5.3 (b) Prediction on CWFID dataset.

5.4 Metrics

```
Loss: 0.00348236609436323
Mean Accuracy: 0.994707358901611
Mean iou 0.9803453279771839
Precision (label 0): 0.3608780431852516 Recall (label 0): 0.6168240148136361
Precision (label 1): 0.8584504765210056 Recall (label 1): 0.9386781382643827
Precision (label 2): 0.9995336189298678 Recall (label 2): 0.9958416197867751
```

Fig. 5.4 Metrics results obtained on Bonn dataset.

Here, label 0 -> Weed, label 1 -> Crop, label 2-> Soil.

Table 5.4 (a) Bonnet's performance from [8]

Dataset	Network	mAcc[%]	Precision [%]		Recall [%]	
			Weed	Crops	Weed	Crops
Bonn	I_{RGB}	86.84	83.63	81.14	91.99	80.42
	$I_{RGB} + I_{NIR}$	93.72	90.51	**95.09**	**94.79**	89.46
	$I_1 \ldots I_{14}$(ours)	**94.74**	**98.16**	91.97	93.35	**95.17**
Zurich	I_{RGB}	45.51	59.75	19.71	45.52	23.66
	$I_{RGB} + I_{NIR}$	68.03	67.41	46.78	**65.31**	49.32
	$I_1 \ldots I_{14}$(ours)	**72.08**	**67.91**	**72.55**	63.33	**64.94**
Stuttgart	I_{RGB}	46.05	42.32	42.03	46.10	25.01
	$I_{RGB} + I_{NIR}$	73.99	74.30	**70.23**	**71.35**	53.88
	$I_1 \ldots I_{14}$(ours)	**76.54**	**87.87**	65.25	64.66	**85.15**

Table 5.4(b) Pixel Wise Test Performance. Trained in 70% Bonn, reporting 15% held-out Bonn, 100% Stuttgart and 100% Zurich from [8].

Dataset	Network	mIoU[%]	IoU[%]			Precision [%]			Recall [%]		
			Soil	Weed	Crops	Soil	Weed	Crops	Soil	Weed	Crops
Bonn	I_{RGB}	59.98	99.08	20.64	60.22	99.92	28.97	66.49	99.15	41.79	82.45
	$I_{RGB} + I_{NIR}$	76.92	99.29	49.42	82.06	99.88	52.90	84.19	99.33	**88.24**	97.01
	$I_1 \ldots I_{14}$(ours)	**80.8**	**99.48**	**59.17**	**83.72**	**99.95**	**65.92**	**85.71**	**99.53**	85.25	**97.29**
Zurich	I_{RGB}	38.25	96.84	14.26	03.62	96.95	14.96	03.78	96.88	35.35	45.86
	$I_{RGB} + I_{NIR}$	41.23	98.44	16.83	08.43	99.68	19.03	09.07	98.46	**51.27**	54.46
	$I_1 \ldots I_{14}$(ours)	**48.36**	**99.27**	**23.40**	**22.39**	**99.90**	**31.43**	**23.05**	**99.36**	47.79	**88.74**
Stuttgart	I_{RGB}	48.09	99.18	21.40	23.69	99.84	21.90	52.43	99.34	42.95	28.95
	$I_{RGB} + I_{NIR}$	55.82	98.54	23.13	45.80	99.85	25.28	68.76	98.69	**49.10**	57.84
	$I_1 \ldots I_{14}$(ours)	**61.12**	**99.32**	**26.36**	**57.65**	**99.86**	**37.58**	**68.77**	**99.45**	46.90	**78.09**

Although it's difficult to compare the results with the performance in [8], as the model from [8] is trained over 3 datasets namely: Bonn, Zurich and Stuttgart as in the above figure whereas we have trained using only Bonn dataset due to availability of the dataset.

5.4.1 Mean Accuracy

In terms of mean accuracy, our model achieves an accuracy of 99.47% as compared to its performance of 94.74% from table 5.4 (a) on Bonn Dataset. But the performance of a semantic segmentation can't be evaluated only on accuracy, we have calculated several other metrics too.

5.4.2 Mean IOU

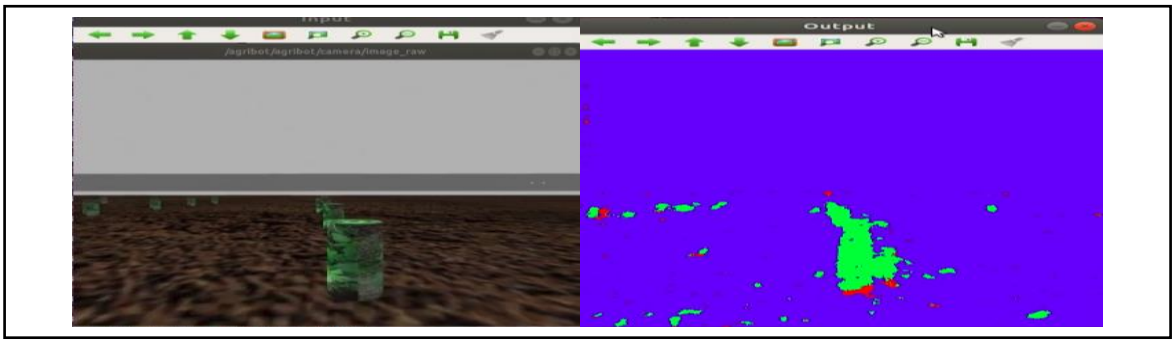
Our model performs at mean IOU of 98.03% as compared to its performance in [8] which is 80.8%. This difference might be because model in [8] is trained on wide range of 3 datasets and this score is generated on a part of one dataset i.e. Bonn dataset. Our model performs much better in this metric.

5.4.3 Precision & Recall

Here we are comparing our results from fig. 5.4 with table 5.4(b). In terms of precision and recall, the model performs quite similar for the Crop class. While the performance is terms of weed class, it is not upto mark as mentioned in [8]. Hence for the weed class, it still needs improvement. Although the prediction result of our model shown above on both datasets are quite close to the ground truth and hence the model is performing much better on the unseen data.

The model performance could be enhanced by training it with data of different crops by providing the ground truth. For providing the ground truth, one should pixel-wise segment the image into 3 classes where each class is represented by one of the R, G & B channels. This might require quite a huge amount of resources if we want to generate a large dataset on our own for our application. Instead we can take limited number of images from the agricultural field where we want to deploy our robot and prepare the dataset by preparing the pixel wise segmentation-based ground truth. Now we can fine tune our model with these data instead of training the whole model from scratch. These lead to training only high level and complex features instead of training the basic features like edges from scratch on limited amount of data.

5.5 Model Prediction in Gazebo Simulator



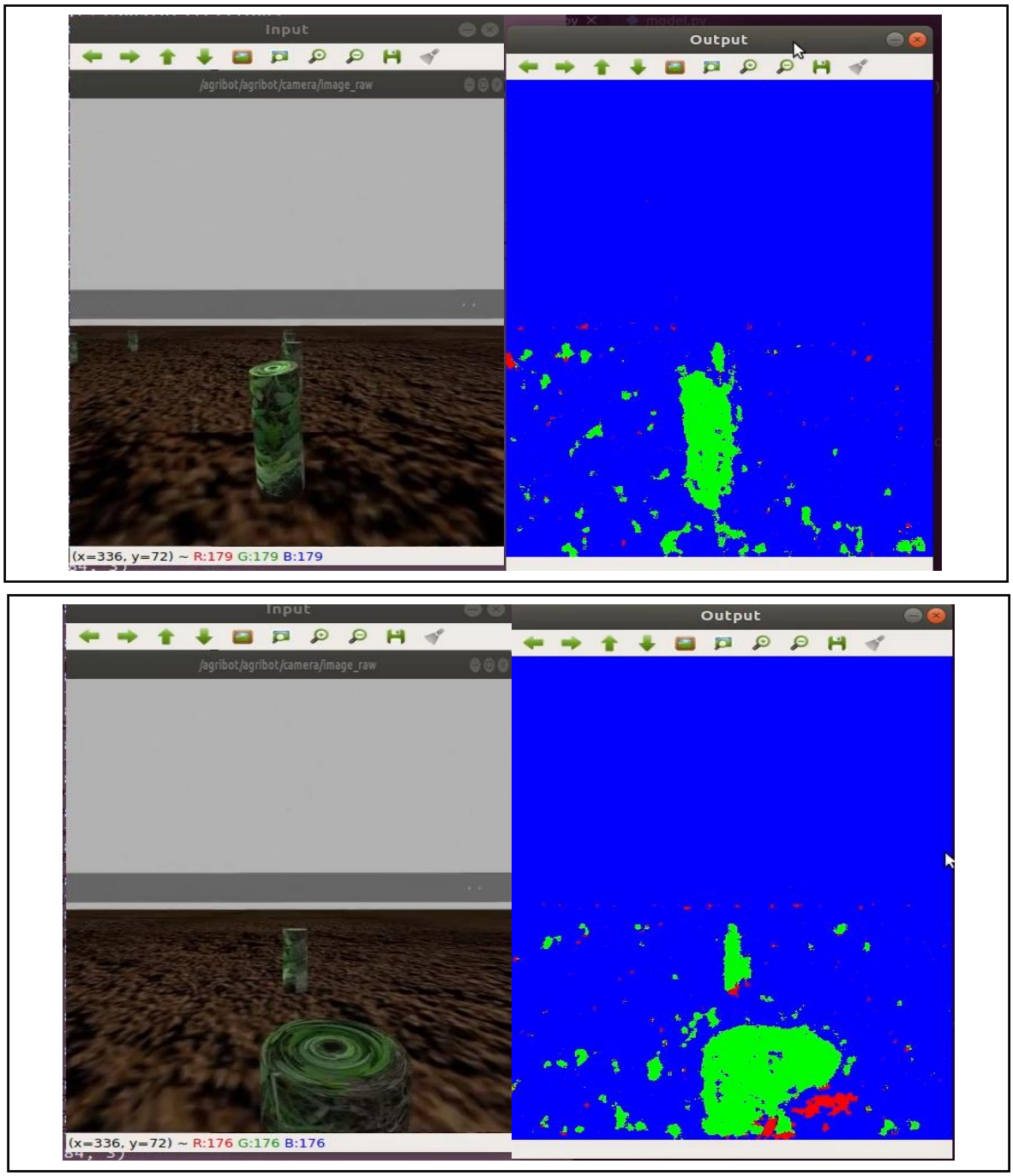
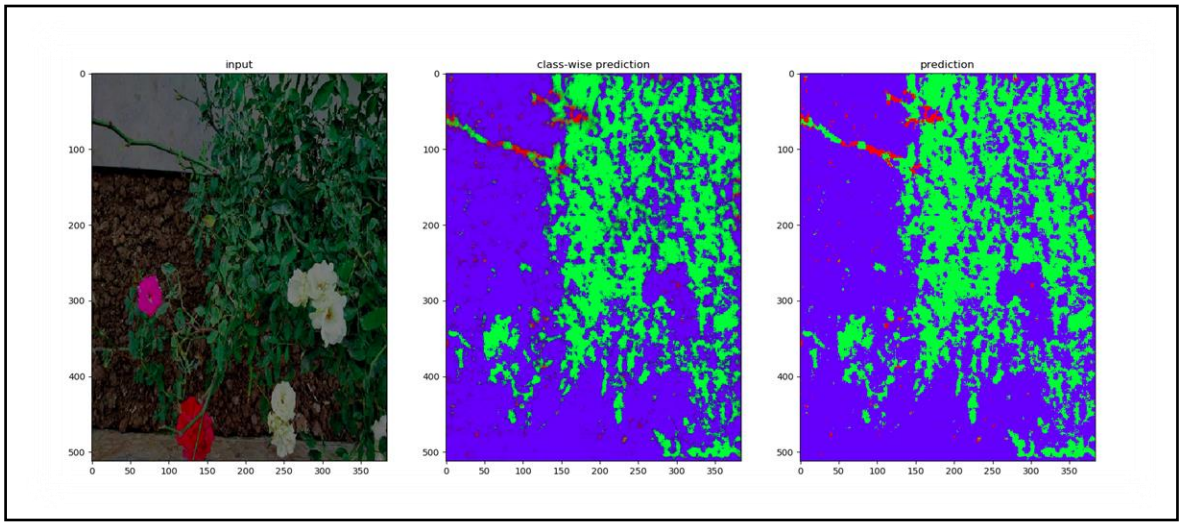
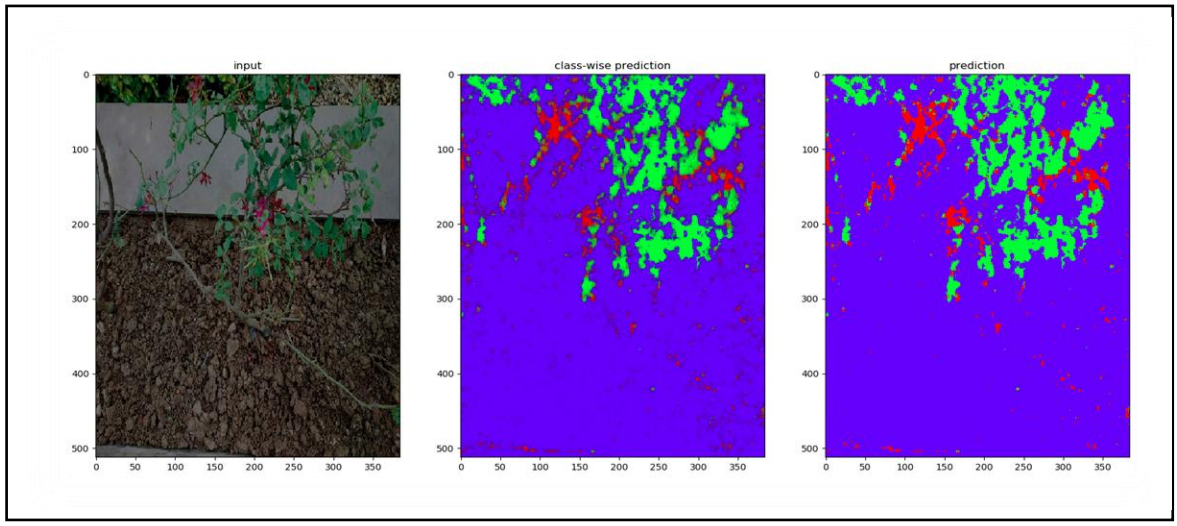


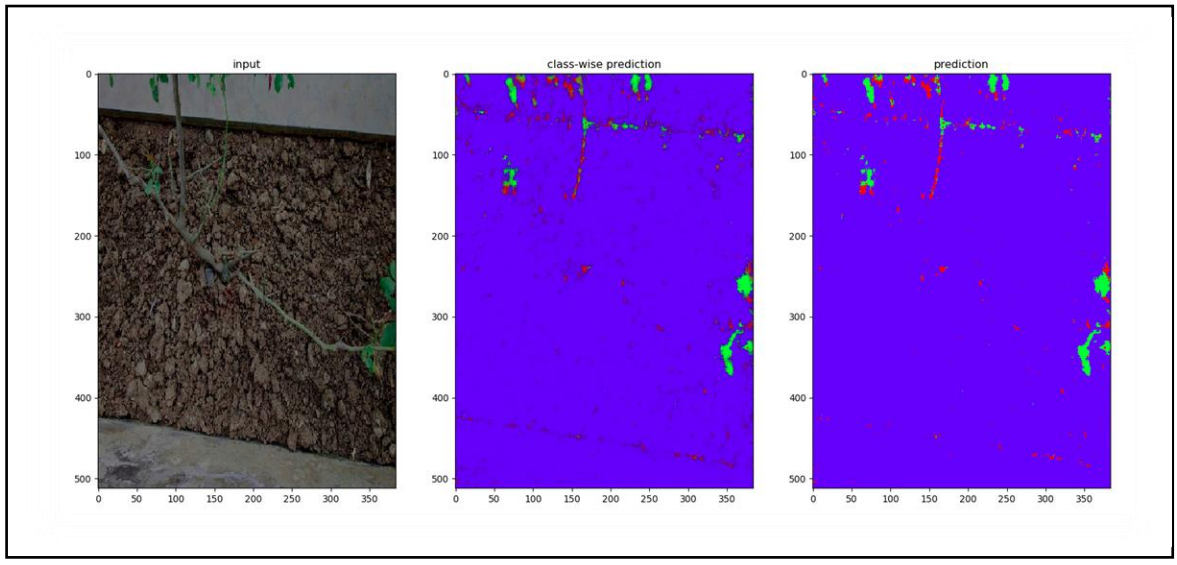
Fig. 5.5 Model Prediction in the Gazebo Simulator.

Here, the left window refers to video feed from camera in Gazebo simulation of our AGRIBOT. And the right window is the model prediction of Bonnet model on that camera feed.

5.6 Prediction on Images from Surrounding Farm







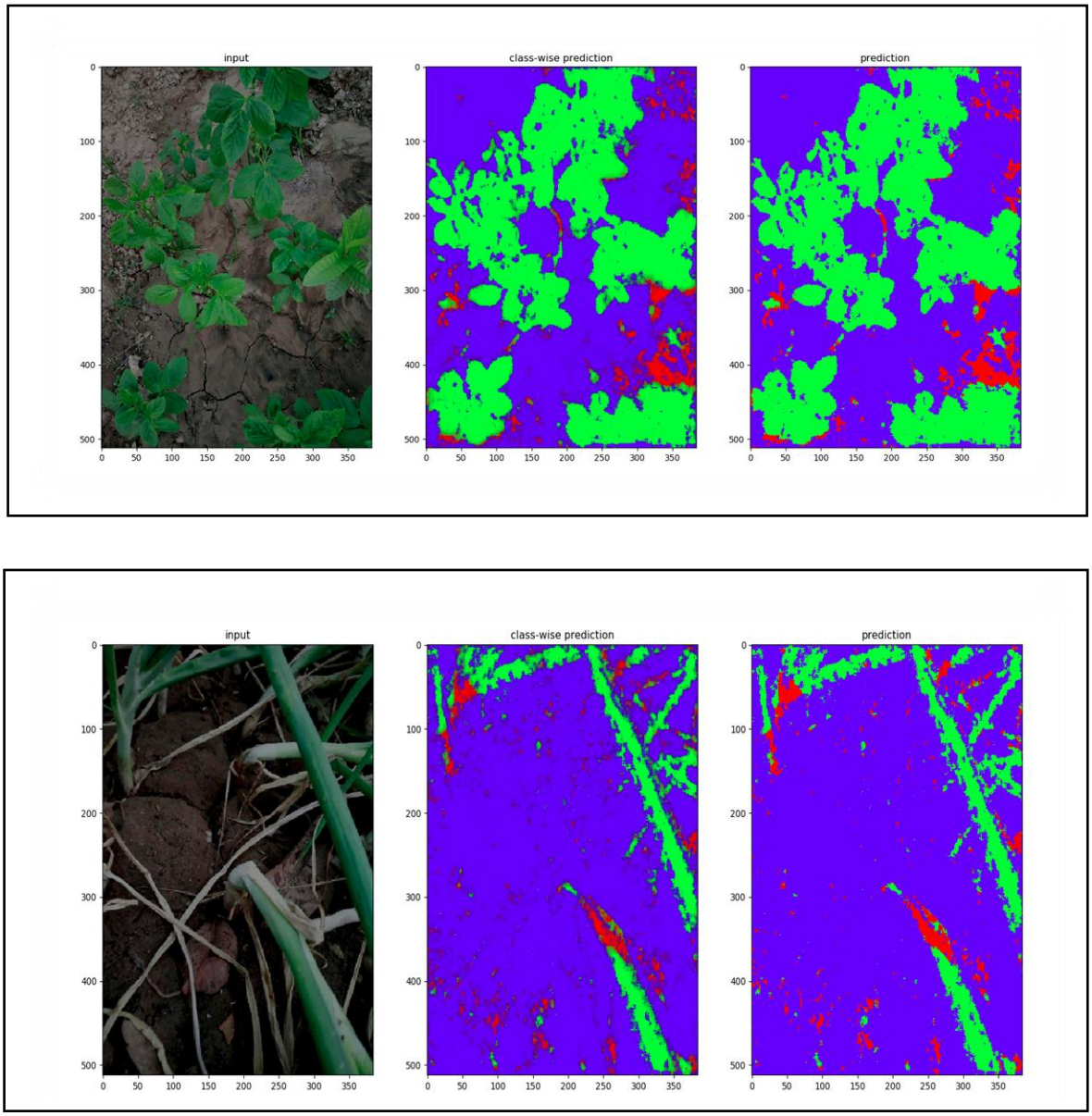


Fig. 5.6 Predicted Images from the nearby field.

From the above predictions, it can be concluded that the model has now correctly learned to recognize the vegetation mask in the image. It has also tried to classify features like less greenery vegetation, stick like parts of plants as weeds. The model has started to learn to classify weeds and the predictions can still be improved by providing more data to it.

*****

CONCLUSION CHAPTER-6

This project presents the idea of making an agriculture robot which is specifically used for the detection of weeds. It begins with the current situation of the Indian Agriculture and how robotics techniques are being used in the farming industry all over the world. Then the hardware system design consisting of both mechanical and electronics systems was designed by considering various key parameters presented by the field. The key highlights from the hardware system are: Skid-Steering System for the robot, why Jetson Nano is used as the primary processor and selection of various peripheral circuitries on basis of given constraints. Data from various sensors is acquired and is processed using various filtering techniques and sensor fusion to get important information like distance and angle to destination which is used for autonomous navigation of the robot. Simultaneous control of Linear and Rotational motion was done on the robot which was used to reach the destination. Then various approaches for crop vs weed classification were discussed and from that Semantic Segmentation approach was best suited for the agriculture robot. Two Semantic Segmentation architectures namely UNet and Bonnet were trained and tested on 2 different datasets: CWFID and Bonn dataset. Bonnet Model was selected over Unet due to its better metrics and performance and lesser number of parameters which makes it possible to run on a real-time for our application. In the Bonnet Model, we achieved a mean accuracy of 99.47 %, a mean iou of 98.03% and loss of 0.00348 units on the Bonn Dataset. Our model has an average latency of 2.5 fps on i7 + NVIDIA 940 MX. This makes it possible to deploy the model on an on-board processer like NVIDIA Jetson Nano for real-world application.

It performs very well over unseen data in both the datasets and its prediction is much closer to the ground truth. We have tested its performance on images from the surrounding farm. It predicts and segments the vegetation mask quite well and is able to classify less green and dry parts of plants as weeds.

For improving the autonomous navigation and making the trajectory of the AGRIBOT more smooth, various other techniques such DGPS (Differential GPS), implementation of predictive filters and control techniques can be tested. For minimizing the crop damage, data

from camera can be fused and used for precise navigation of the robot. For the machine vision part, the performance can be greatly improved by providing more images and their corresponding ground truths from different real scenarios to the classification model where the robot will be deployed. This will lead to precise weed detection in real case scenarios. Design of the AGRIBOT is experimented and validated on Gazebo simulator. Autonomous Navigation along with the classification of crop and weed on real-time basis in the simulated environment of the field was the goal of this project and it was achieved. This project would be a huge contribution for providing the boost in the diminishing field of agriculture.

*****

REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, "Deep Learning", MIT, USA: MIT Press, 2016. Accessed: 15th October, 2019. [Online]. Available: http://www.deeplearningbook.org

[2] Mulham Fawakherji, Ali Youssef, Domenico Daniele Bloisi, Alberto Pretto, Daniele Nardi, "Crop and Weeds Classification for Precision Agriculture Using Context-Independent Pixel-Wise Segmentation", in IEEE IRC 2019, The third IEEE Int. Conf. on Robotic Computing, Naples, Italy, 2019, doi: 10.1109/IRC.2019.00029

[3] Benjamin Schneiders, Gregory Palmer, Shan Luo, Karl Tuyls, "Fully Convolutional One-Shot Object Segmentation for Industrial Robotics", Univ. of Liverpool, Liverpool, UK, 2019.

[4] Jingyao Gai, "Plant Detection, Localization and Discrimination using 3D Machine Vision for Robotic Intra-row Weed Control", Graduate Thesis and Dissertations, Agricultural and Biological Eng., lowa State Univ., Ames, lowa, 2016.

[5] Paul J. Komi, "Plant Classification Combining Colour and Spectral Cameras for Weed Control Purposes", Ph.D. Thesis, Dept. Computer Vision, Loughborough Univ., Leicestershire, UK, 2008.

[6] Hong Y. Jeon , Lei F. Tian, Heping Zhu, "Robust Crop and Weed Segmentation under Uncontrolled Outdoor Illumination", Agricultural Research Service, Application Technology Research Unit, Wooster, USA, 2011.

[7] Mulham Fawakherji, Ali Youssef, Domenico Daniele Bloisi, Alberto Pretto, Daniele Nardi, "Crop and Weeds Classification for Precision Agriculture Using Context-Independent Pixel-Wise Segmentation", in IEEE IRC 2019, The third IEEE Int. Conf. on Robotic Computing, Naples, Italy, 2019, doi: 10.1109/IRC.2019.00029

[8] Andres Milioto, Philipp Lottes, Cyrill Stachniss, "Real-time Semantic Segmentation of Crop and Weed for precision Agricultural Robots Leveraging Background

Knowledge in CNNs", in 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 2018.

[9] Olaf Ronneberger, Phillip Fischer, Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS, Vol.9351: 234--241, 2015.

[10] Jorn Ostermann, Sebastian Haug, "A Crop/Weed Field Image Dataset for the Evaluation of Computer Vision Based Precision Agricultural Tasks", In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014.

[11] Mostafa Sharifi1, Matthew Samuel Young1, XiaoQi Chen1, Don Clucas1 and Christopher Pretty presented "Mechatronic design and development of a non-holonomic omnidirectional mobile robot for automation of primary production" at Co-gent Engineering (ISSN: 2331-1916) is published by Cogent OA, part of Taylor & Francis Group.

[12] ABC Science, ―RIPPA The Farmbot Exterminates Pests and Weeds‖ (14th May, 2018). Accessed on Oct. 17, 2019, (Online Video) Available: https://www.youtube.com/watch?v=XP7GoNKcTS4

[13] Nvidia, https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed Nov. 5th, 2019)

[14] Robot Platform, http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html (accessed Oct. 24th, 2019)

[15] AGROBOT, https://www.agrobot.com (accessed Oct. 24th, 2019)

[16] Sami Salama Hussen Hajjaj and Khairul Salleh Mohamed Sahari, ― "Bringing ROS to Agriculture Automation: Hardware Abstraction of Agriculture Machinery" presented at International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 3 (2017) pp. 311-316.

[17] Mostafa Sharifi, XiaoQi Chen, Christopher Pretty, Don Clucas, Erwan Carbon-Lunel presented "Modelling and simulation of a non-holonomic omnidirectional mobile

robot for offline programming and system performance analysis" in Simulation Modelling Practice and Theory 87 (2018) 155-169 by ELSEVIER

[18] Shamshiri R R, Hameed I A, Pitonakova L, Weltzien C, Balasundram S K, Yule I J, Et. al. Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison. Int J Agric & Biol Eng, 2018; 11(4): 15–31.

[19] Peter Biber, Ulrich Weiss, Michael Dorna, Amos Albert presented "Navigation System of the Autonomous Agricultural Robot " 'BoniRob"' " with Corporate Sector Research and Advance Engineering, Robert Bosch GmbH, Postfach 300240, 70442 Stuttgart, Germany.

[20] Jin-Hwan Joo, Dae_Han Hong, Yoon-Gu Kim, Ho-Geun Lee, Ki-Dong Lee, Suk-Gyu Lee presented "An Enhanced Path Planning of Fast Mobile Robot based on Data Fusion of Image Sensor & GPS" in ICROS-SICE International Joint Conference 2009 August 18-21, 2009, Fukuoka International Congress Center, Japan

[21] Zhiping Liu, Mingjing Zhu presented "Calibration and Error Compensation of Magnetometer" in 2014 26th Chinese Control and Decision Conference (CCDC).

[22] Lars Nieradzik. "Losses for Image Segmentation." github.io https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/ (accessed Feb 25. 2020)

[23] Wang T, Wu Y, Liang J, Han C, Chen J, Zhao Q. "Analysis and experimental kinematics of a skid-steering wheeled robot based on a laser scanner sensor" in Sensors (Basel). 2015;15(5):9681-9702. Published 2015 Apr 24. doi:10.3390/s150509681

[24] Nived Chebrolu, Philipp Lottes, Alexander Schaefer, Wera Winterhalter, Wolfram Burgard and Cyrill Stachniss presented "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields" in The International Journal of Robotics Research in 2017, Vol. 36(10) 1045-1052.

[25] Arduino, https://www.arduino.cc/ (accessed on Nov 5th. 2019)

ACRONYMS

ROS	Robotics Operating System
GUI	Graphical User Interface
GDP	Gross Domestic Product
AI	Artificial Intelligence
UAV	Unmanned Aerial Vehicle
HRI	Human Robot Interaction
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
TF	Transform
CAD	Computer Aided Design
URDF	Unified Robot Description Format
DOF	Degrees of freedom
RVIZ	Robot Visualization
IMU	Inertial Measurement Unit
GPS	Global Positioning System
PWM	Pulse Width Modulation

Appendix A

1) Example of Link Chain for Robotic Structure.

```
<link name="base_link">
    <inertial>
      <mass value="3.4672" />
      <origin xyz="0 0 0"
              rpy="0 0 0"  />
      <inertia
            ixx="0.045731"    ixy="-1.5439E-12"       ixz="-8.4448E-09"
            iyy="0.077236"       iyz="1.5494E-12"       izz="0.049938" />
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="1.57 0 3.14"/>
      <geometry>
            <mesh filename="package://agribot_description/meshes/base_link.STL"/>
      </geometry>
      <material name="White"/>
    </visual>
    <collision>
     <origin xyz="0 0 0" rpy="1.57 0 3.14" />
     <geometry>
            <mesh filename="package://agribot_description/meshes/base_link.STL"/>
```

```
        </geometry>
      </collision>
    </link>
```

2) Example of Joint for Robotic Structure.

```
<xacro:macro name="agribot_part" params="parent name translateX translateY translateZ
rotateX rotateY rotateZ color">
                <joint name="${parent}_to_${name}_joint" type="continuous">
                  <origin xyz="${translateX} ${translateY} ${translateZ}"
                        rpy="${rotateX} ${rotateY} ${rotateZ}" />
                  <parent link="base_link" />
                  <child link="${name}" />
                  <axis xyz="1 0 0" rpy="0 0 0"/>
                  <limit effort="100" velocity="100"/>
                  <joint_properties damping="0.0" friction="0.0"/>
                </joint>
</xacro>
```

3) IMU Plugin for Gazebo

```
<joint name="imu_to_base_link" type="fixed">
        <origin xyz="0 0.06 0" rpy="0 0 0" />
        <parent link="base_link"/>
        <child link="imu_link"/>
```

```
</joint>
<gazebo>
      <plugin name="imu" filename="libhector_gazebo_ros_imu.so">
          <serviceName>/imu/calibrate</serviceName>
         <robotNamespace>/agribot</robotNamespace>
          <updateRate>200.0</updateRate>
          <bodyName>imu_link</bodyName>
          <frameId>imu_link</frameId>
          <topicName>imu</topicName>
          <rpyOffset>0 0 0</rpyOffset>
          <xyzOffset>0 0 0</xyzOffset>
          <gaussianNoise>0.00000001</gaussianNoise>
          <accelOffset>0 0 -9.79999999586</accelOffset>
          <accelDrift>0.00000001 0.00000001 0.00000001</accelDrift>
          <accelDriftFrequency>0.00000001 0.00000001
0.00000001</accelDriftFrequency>
          <accelGaussianNoise>0.00000001 0.00000001
0.00000001</accelGaussianNoise>
          <rateDrift>0.0 0.0 0.0</rateDrift>
          <rateDriftFrequency>0.0 0.0 0.0</rateDriftFrequency>
          <rateGaussianNoise>0.0 0.0 0.0</rateGaussianNoise>
          <headingDrift>0.0 0.0 0.0</headingDrift>
          <headingDriftFrequency>0.0 0.0 0.0</headingDriftFrequency>
```

```
        <headingGaussianNoise>0.0 0.0 0.0</headingGaussianNoise>
      </plugin>
    </gazebo>
```

4) GPS Plugin for Gazebo.

```
<gazebo>
<plugin name="gps_controller" filename="libhector_gazebo_ros_gps.so">
<alwayson>true</alwayson>
<robotNamespace>/agribot</robotNamespace>
<updateRate>5.0</updateRate>
<bodyname>gps_link</bodyname>
<topicname>fix</topicname>
<velocitytopicname>fix_velocity</velocitytopicname>
<drift>5.0 5.0 5.0</drift>
<offset>0 0 0</offset>
<status> 0 </status>
<service> 1 </service>
<referenceLatitude> 21.1613108102 </referenceLatitude>
<referenceLongitude> 72.7869817026 </referenceLongitude>
<gaussiannoise>0.1 0.1 0.1</gaussiannoise>
```

<velocitydrift>0 0 0</velocitydrift>

<velocitygaussiannoise>0.1 0.1 0.1</velocitygaussiannoise>

</plugin>

</gazebo>

Appendix B

1) NVIDIA JETSON NANO SYSTEM-ON-MODULE

Maxwell GPU ◊

128-core GPU | End-to-end lossless compression | Tile Caching | OpenGL® 4.6 | OpenGL ES 3.2 | Vulkan™ 1.1 | CUDA® | OpenGL ES Shader Performance (up to): 512 GFLOPS (FP16)
Maximum Operating Frequency: 921MHz

CPU

ARM® Cortex® -A57 MPCore (Quad-Core) Processor with NEON Technology | L1 Cache: 48KB L1 instruction cache (I-cache) per core; 32KB L1 data cache (D-cache) per core | L2 Unified Cache: 2MB | Maximum Operating Frequency: 1.43GHz

Audio

Industry standard High Definition Audio (HDA) controller provides a multichannel audio path to the HDMI interface.

Memory

Dual Channel | System MMU | Memory Type: 4ch x 16-bit LPDDR4
| Maximum Memory Bus Frequency: 1600MHz | Peak Bandwidth:
25.6 GB/s | Memory Capacity: 4GB

Storage

eMMC 5.1 Flash Storage | Bus Width: 8-bit | Maximum Bus Frequency: 200MHz (HS400)
| Storage Capacity: 16GB

Boot Sources

eMMC and USB (recovery mode)

Networking

10/100/1000 BASE-T Ethernet | Media Access Controller (MAC)

Imaging

Dedicated RAW to YUV processing engines process up to
1400Mpix/s (up to 24MP sensor) | MIPI CSI 2.0 up to 1.5Gbps (per lane) | Support for x4 and x2 configurations (up to four active streams).

Operating Requirements

Temperature Range (T_j): -25 – 97C* | Module Power: 5 – 10W | Power Input: 5.0V

Display Controller

Two independent display controllers support DSI, HDMI, DP, eDP:

MIPI-DSI (1.5Gbps/lane): Single x2 lane | Maximum Resolution: 1920x960 at 60Hz (up to 24bpp)

HDMI 2.0a/b (up to 6Gbps) | DP 1.2a (HBR2 5.4 Gbps) | eDP
1.4 (HBR2 5.4Gbps) | Maximum Resolution (DP/eDP/HDMI): 3840 x 2160 at 60Hz (up to 24bpp)

Clocks

System clock: 38.4MHz | Sleep clock: 32.768kHz | Dynamic clock scaling and clock source selection

Multi-Stream HD Video and JPEG

Video Decode

H.265 (Main, Main 10): 2160p 60fps | 1080p 240fps

H.264 (BP/MP/HP/Stereo SEI half-res): 2160p 60fps | 1080p 240fps

H.264 (MVC Stereo per view): 2160p 30fps | 1080p

120fps VP9 (Profile 0, 8-bit): 2160p 60fps | 1080p

240fps

VP8: 2160p 60fps | 1080p 240fps

VC-1 (Simple, Main, Advanced): 1080p 120fps |

1080i 240fps MPEG-2 (Main): 2160p 60fps | 1080p

240fps | 1080i 240fps

Video Encode

H.265:2160p 30fps | 1080p 120fps

H.264 (BP/MP/HP): 2160p 30fps | 1080p 120fps

H.264 (MVC Stereo per view): 1440p 30fps | 1080p

60fps VP8: 2160p 30fps | 1080p 120fps

JPEG (Decode and Encode): 600 MP/s

Peripheral Interfaces

xHCI host controller with integrated PHY: 1 x USB 3.0, 3 x USB 2.0
| USB 3.0 device controller with integrated PHY | EHCI controller with embedded hub for USB 2.0 | 4-lane PCIe: one x1/2/4 controller
| single SD/MMC controller (supporting SDIO 4.0, SD HOST 4.0) |
3 x UART | 2 x SPI | 4 x I2C | 2 x I2S: support I2S, RJM, LJM, PCM,
TDM (multi-slot mode) | GPIOs

Mechanical

Module Size: 69.6 mm x 45 mm | PCB: 8L HDI | Connector: 260 pin SO-DIMM