VISUAL – LIDAR FUSION BASED DEEP LEARNING FRAMEWORK FOR AUTONOMOUS VEHICLE NAVIGATION

GROUP G06

BANDARA N.K.B.L.M. (E/15/034)
DEEN T.A. (E/15/070)
SANDARUWAN W.G.P.G. (E/15/324)

SUPERVISED BY

PROF. LILANTHA SAMARANAYAKE

Department of Electrical and Electronic Engineering

ABSTRACT

Autonomous vehicles have become a major topic of research interest in the recent past. This framework can be used in Robots that are used for applications such as delivery, security, warehouses, and home robots. Autonomous vehicles in agricultural fields can help to improve the living standard of the farmers. Not only their convenience at work in the field but also their personal lives since these vehicles can be used for different kinds of tasks from preparing the land for cultivation to harvesting.

The Robot developed by the team despite the COVID19 pandemic uses the Robot Operating System (ROS), which is a vastly used operating system for developing robots with Ubuntu OS. For global path planning Inertial Measurement Unit (IMU), Global Positioning System (GPS) and wheel encoders were used as inputs and an Unscented Kalman Filter (UKF) was developed for sensor fusion for more accurate results. Visual input was taken using a ZED^{TM} stereo camera for generating 3D point cloud maps and local path planning was done with the help of neural networks. A segmentation model was developed to detect the drivable area and an object detection model was used for obstacle avoidance.

Up to now, studies, simulation and hardware testing have successfully been conducted for vehicle navigation rough terrains avoiding dynamic obstacles.

ACKNOWLEDGEMENTS

The successful completion of this project would not have been possible without the proper assistance and guidance of many people whose names may not be all enumerated. Their contribution is sincerely appreciated and very gratefully acknowledged.

We would also like to acknowledge with much gratitude the support and guidance given by Prof. Lilantha Samaranayake (Professor, Department of Electrical and Electronic Engineering) throughout the completion of the project.

Furthermore, we would also like to acknowledge with much appreciation the support given by the non-academic staff of the Department of Electrical and Electronic Engineering. And finally, we would like to extend our sincere thanks to our families and all those mentioned/ non- mentioned who helped us in the successful completion of this project.

CONTENT

ABSTRACT………………………………………………………………… i

ACKNOWLEDGEMENT……………………………………………………. ii

CONTENT…………………………………………………………………. iii

LIST IF FIGURES…………………………………………………………. iv

LIST OF ABBREVIATION………………………………………………. v

1. **CHAPTER 01 INTRODUCTION** ……………………………….... 1
2. **CHAPTER 02 LITERATURE REVIEW……………………………. 2**
   - 2.1 Robot Operating System (ROS)... 2
   - 2.2 Unscented Kalman Filter…………………………………........ 3
   - 2.3 Stereo Camera System ……………………........................... 3
   - 2.4 Visual Perception ……………………………………………..... 4
3. **CHAPTER 03 THEORY…………………………………………..... 5**
   - 3.1 Unscented Kalman Filter.. 5
   - 3.2 Neural Network Local Path Planner…………………………….. 6
4. **CHAPTER 04 METHODOLOGY………………………………….. 8**
   - 4.1 Overview of Methofology……... 8
   - 4.2 Local Path Planner using sementation model……………….......... 11
   - 4.3 Printed Circuit Board Design ……………………...................... 12
5. **CHAPTER 05 RESULTS AND DISCUSSIONS**…………………... 13
   - 5.1 Uncentred Kalman Filter 13
   - 5.2 Simulation Environment 14
   - 5.3 Visual Perception 16
   - 5.4 Network Configuration 19
6. **CHAPTER 06 CONCLUSION**……………………………… …... 21
7. **CHAPTER 07 FUTURE WORKS**……………………………….... 22
8. **CHAPTER 08 REFERENCES**…………………………………… 23

LIST OF FIGURES

Figure 01: A typical ROS network configuration
Figure 02: Transformation data of Kalman filter, EKF and UKF
Figure 03: Architecture of a fully convolutional neural network
Figure 04: Finalized Robot
Figure 05: Robot with two SBCs
Figure 06: ZED Stereo Camera with Jetson Nano and Intel UP2 SBC
Figure 07: Printed Circuit board for the project
Figure 08: MATLAB Example with random noise
Figure 09: Noisy outputs of sensors.
Figure 10: UKF Position Estimation with EKF with noisy inputs
Figure 11: Modified Robot with ZED camera in GAZEBO environment
Figure 12: Created Simulated Agricultural Environment for testing
Figure 13: Generating 3D Point cloud map with ZED2 in Gazebo Simulated Environment
Figure 14: Autonomous Navigation testing in simulated environment
Figure 15: Autonomous Navigation with Obstacle Avoidance
Figure 16: Initial segmentation results and noise filtered image result
Figure 17: Line detected image using noise filtered gray image
Figure 18: Extracting points related to left and right boundaries of drivable area
Figure 19: Deep Neural Network testing in faculty road
Figure 20: YOLO detection result
Figure 21: Computer Networking Diagram

LIST OF ABBREVIATIONS

AC	- Alternating Current
API	- Application Programming Interface
CPU	- Central processing unit
DC	- Direct Current
DNN	- Deep Neural Network
EKF	- Extended Kalman Filter
GPS	- Global Positioning System
GPU	- Graphic Processing Unit
IMU	- Inertial Measurement Unit
IOU	- Intersection Over Union
L4T	- Linux for Tegra
LIDAR	- Light Detection and Ranging
ONNX	- Open Neural Network Exchange
ROS	- Robot Operating System
SBC	- Single Board Computer
SSH	- Secure Shell
UGV	- Unmanned Ground Vehicles
UKF	- Unscented Kalman Filter
USB	- Universal Serial Bus
UT	- Uncentred Transform
XML	- Extensible Markup Language
YOLO	- You only look once

CHAPTER 01 INTRODUCTION

Unmanned Ground Vehicles (UGVs) were very useful when automation of industrial and field works. During the last decade, people were trying to use the UGVs in various applications. Agriculture is one of the fields that UGVs are popular for various applications starting from land preparation to harvesting.

Previously, Cost of IMU was relatively high and building UGVs for affordable price was a difficult task but due to the availability of low-cost Inertial Measurement Unit (IMU), the inertial navigation systems have become suitable for cost- sensitive UGV applications. Although the low-cost sensors don't perform as well as high-grade sensors in terms of drift and noise in the measurement, they can still meet the requirements of UGV navigation system when integrated with the sensors such as Global Positioning System (GPS), magnetometer and Vision camera thanks to the complementary characteristics of these sensors [1].

However, the integration of GPS signals with Inertial Measurement Units has become a standard approach for position determination of UGV. By far the primary mechanism used to blend GPS data with inertial sensor measurement has been the Kalman filtering which used the predetermined and constant values of the process noise covariance and measurement noise covariance. When the state transition and observation models are nonlinear, extended Kalman and unscented Kalman filtering have to be used. Because the basic Kalman filter is limited to a linear assumption. When comparing extended vs unscented Kalman filters the Unscented Kalman filters were more accurate. [2]

This vehicle can be used as a multipurpose vehicle and with the further developments In Sri Lanka Inadequate, involvement of the technology in agriculture was a bit of a concern. This project can be used to improve the standards of living of the farmers. Since this can be easily optimized for field work from preparing the land to harvesting. This Project to develop a control system that can be implemented in commercially available vehicles such as tractors and combined harvesters. With user, friendly controls using mobile applications.

CHAPTER 02 LITERATURE REVIEW

Development of Autonomous vehicles with Unscented Kalman filters was started in the recent past. A lot of research has been going on. But we couldn’t find any major research related to Unscented Kalman filter implementation with the Robot Operating system. The specialty of the Kalman Filter is, Since the sensor fusion is done by the Kalman filter we can avoid the expensive sensors with higher accuracy. That will drastically reduce the cost of the robot.

2.1 Robot Operating System

The Robot Operating System (ROS) environment is an open-source meta-operating system. The system has been widely adopted by the scientific community, and it allows direct access to the latest developments in the field of robotics [3] [4].
The ROS framework is easy to implement in any modern programming language. Already implemented it in Python, C++, and Lisp, Also Java and Lua libraries in the experimental stage [4].
ROS currently only runs on Linux based platforms. It is a distributed computing environment. A running ROS system can comprise dozens, even hundreds of nodes, spread across multiple machines. Depending on how the system is configured, any node may need to communicate with any other node, at any time.

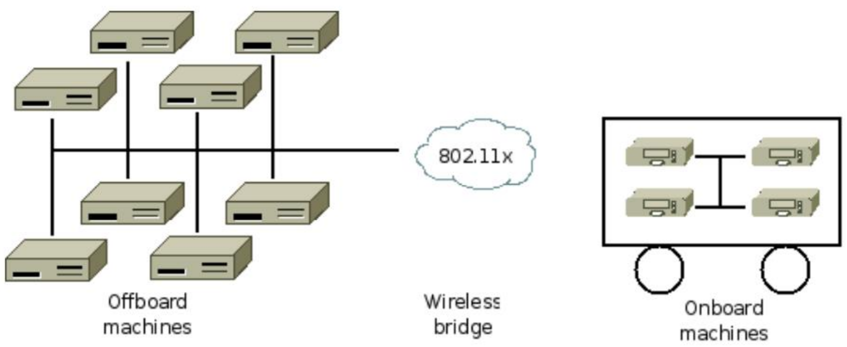


Figure 01: A typical ROS network configuration

2.2 Unscented Kalman Filter

1. Kalman Filter

   Sensor fusion is the process of combining multiple sensor readings to create a more useful measurement. For this Kalman filter is commonly used. Developed around 1960 mainly by Rudolf E. Kalman. It was originally designed for aerospace guidance applications. While it is the optimal observer for a system with noise This is only true for the linear case. [5]

This is not optimum for nonlinear systems. So, we have to go for an Extended Kalman filter or above approach.

   1. Kalman Filter will always work with **Gaussian Distribution.**
   2. Kalman Filter will always work with **Linear Functions.**

2. Extended Kalman Filter

The Extended Kalman Filter (EKF) has become a standard technique used in a number of nonlinear estimation and machine learning applications. For this linear approximation is done to the nonlinear function using Taylor series [5]. With the approximation it can work as a modified Kalman filter. Major drawback of this extended Kalman filter is approximation issues.

3. Unscented Kalman Filter

The UKF addresses the approximation issues of the EKF. The state distribution is again represented by a Gaussian Random Variable (GRV), but is now specified using a minimal set of carefully chosen sample points (Sigma Points). These sample points completely capture the true mean and covariance of the GRV, and when propagated through the true nonlinear system, capture the posterior mean and covariance accurately to the 3rd order (Taylor series expansion) for any nonlinearity. [2]

2.3 Stereo Camera System

Computer vision includes methods such as image acquisition, processing, analysis, and understanding [6]. Computer vision techniques attempt to model a complex visual environment using various mathematical methods. One of the purposes of computer vision is to define the world that we see based on one or more images and to restructure its properties, such as its illumination, shape, and color distributions. Stereo vision is an area within the field of computer vision that addresses an important research problem: which is the reconstruction of the three-dimensional coordinates of points for depth estimation. A stereo vision system consists of a stereo camera, namely, two cameras placed horizontally. The two images captured simultaneously by these cameras are then processed for the recovery of visual depth information [7].

2.4 Visual Perception (Neural Network and image processing)

Visual perception is one of the major aspects which are related with self-driving cars. Drivable area detection and obstacle detection are two main requirements of this. The most convenient method of developing these systems is machine learning. Neural networks are the special branch of machine learning used for this. For drivable area detection, fully convolutional neural networks are commonly used [8]. For obstacle detection, convolutional neural networks followed by dense layers can be used and vgg16, resnet50 and google net are few examples.[9]

In drivable area detection. After building and training, a neural network RGB data can be labeled using segmentation and this data can be used to identify and define a mathematical plane for differentiating roads. These segmentation data and depth map data use by ROS to optimize and develop the local path planner.[10]

In obstacle detection and avoidance. With obstacle detection, a set data about the obstacles on the frame and exact position of them can be taken. With that the lowest distance to collision can be calculated and those calculations can be used to avoid obstacles.

To improve the efficiency image processing techniques can be used and this can be achieved using python environment and 3rd party libraries such as OpenCV, TensorFlow and ONNX.

CHAPTER 03 THEORY

3.1 Unscented Kalman Filter

Consider propagating a random variable x (dimension L) through a nonlinear function, y= g(x). Assume it has mean $\bar{x}$ covariance P_x to calculate the statistics of y a matrix X was formed with 2L+1 Sigma vectors $X_{i.}$ (With Corresponding Weights W_i) according to following:

$X_0 = \bar{x}$

$X_i = \bar{x} + (\sqrt{(L+\lambda)Px})_i$ i= 1…., L

$X_i = \bar{x} - (\sqrt{(L+\lambda)Px})_{i-L}$ i = L+1, ….,2L

$W_0^{(m)} = \lambda/(L+\lambda)$

$W_0^{(c)} = \lambda/(L+\lambda) + (1-\alpha^2+\beta)$

$W_i^{(m)} = W_i^{(c)} = 1/\{2(L+\lambda)\}$ i= 1,2…….,2L

Where $\lambda = \alpha^2(L+k)$L is a scaling parameter. α determines the spread of the sigma points around $\bar{x}$ determines the spread of the sigma points around and is usually set to a small positive value (e.g., 1e-3). K is a secondary scaling parameter which is usually set to 0, and β is used to incorporate prior knowledge of the distribution of x (for Gaussian distributions, β =2 is optimal). $(\sqrt{(L+\lambda)Px})_i$ is the ith row of the matrix square root. These sigma vectors are propagated through the nonlinear function,

$Y_i = g(X_i)$ i= 0,2L

and the mean and covariance for Y are approximated using a weighted sample mean and covariance of the posterior sigma points,

$$\begin{aligned}\bar{\mathbf{y}} &\approx \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_i \\ \mathbf{P_y} &\approx \sum_{i=0}^{2L} W_i^{(c)} \{\mathcal{Y}_i - \bar{\mathbf{y}}\}\{\mathcal{Y}_i - \bar{\mathbf{y}}\}^T\end{aligned}$$

Note that this method differs substantially from general "sampling" methods, which require orders of magnitude more sample points in an attempt to propagate an accurate distribution of the state. The deceptively simple approach taken with the UT results in approximations that are accurate to the third order for Gaussian inputs for all nonlinearities. For non-Gaussian inputs, approximations are accurate to at least the second-order, with the accuracy of third and higher order moments determined by the choice of α and β. A simple example is shown in (Figure 1) for a 2-dimensional system: the left plot shows the true mean and covariance propagation using Monte-Carlo sampling; the center plots show the results using a linearization approach as would be done in the EKF; the right plots show the performance of the UT (note only 5 sigma points are required). The superior performance of the UT is clear.

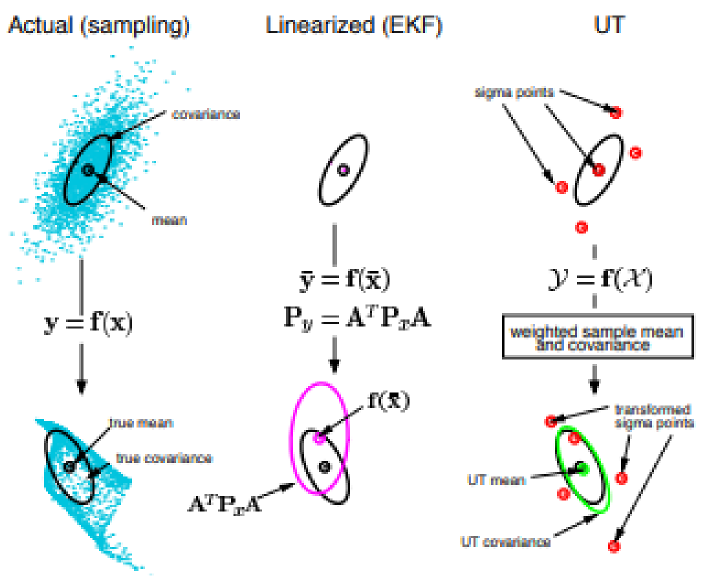

Actual (sampling)
Linearized (EKF)
UT
covariance
mean
sigma points
$\mathbf{y} = \mathbf{f}(\mathbf{x})$
$\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{P}_y = \mathbf{A}^T\mathbf{P}_x\mathbf{A}$
$\mathcal{Y} = \mathbf{f}(\mathcal{X})$
weighted sample mean and covariance
true mean
true covariance
$\mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{A}^T\mathbf{P}_x\mathbf{A}$
transformed sigma points
UT mean
UT covariance

Figure 02: Transformation data of Kalman filter, EKF and UKF

3.2 Neural Network and Local path planner

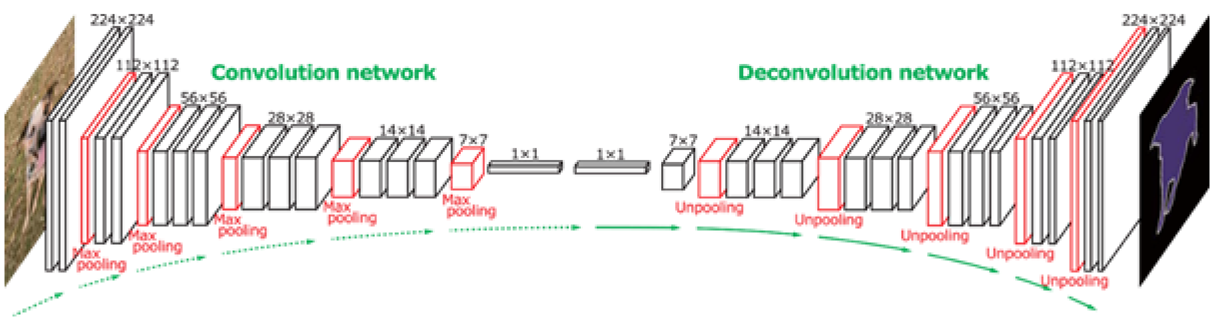


Figure 03: Architecture of a fully convolutional neural network

A fully convolutional neural network (UNET architecture) can be used to classify objects of a visual feed. It inputs an image in a form of 3D array and then it undergoes a series of operations such as convolution, pooling and up sampling to output an object classified image with different colors.

- **Least Mean Squares Method.**

After extracting the points that represent left and right margins of drivable area, Least Mean Square method is used to fit lines to obtain left and right margin gradients of drivable area.

$$X = (H^T H)^{-1} H^T Y$$

X - line parameters (gradient and intersection)

H - x coordinates followed by 1

Y - Y coordinates

Using the above equations, line parameters can be obtained so that the lines represent the borders of the drivable area.

CHAPTER 04 METHODOLOGY

4.1 Overview of Methodology.

The first task is to construct the vehicle to implement the project. The robot should be able to test in terrains so it can’t be too small. But it should be able to handle easily while the implementing and testing process.

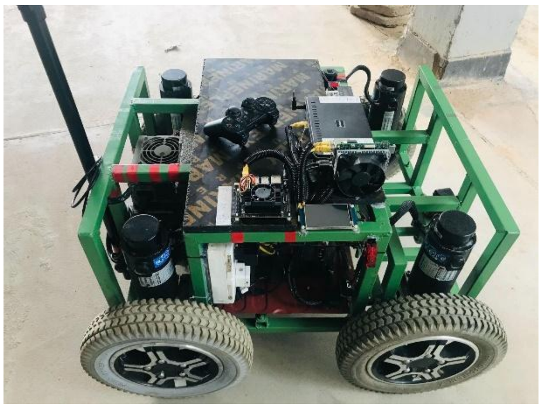


Figure 04: Finalized Robot

Then moved to implementing the robot electronic controlling system. For the initial implementation a motor controller, battery, raspberry pi3 and a PlayStation3 remote controller used as the main controller. A 90Ah Lead-Acid battery powered it.

While developing the robot algorithms it required more processing power. As a solution Raspberry pi3 SBC (Single Board computer) was replaced with Intel UP2 SBC. A Nvidia Jetson Nano SBC was implemented for Stereo Vision depth processing.

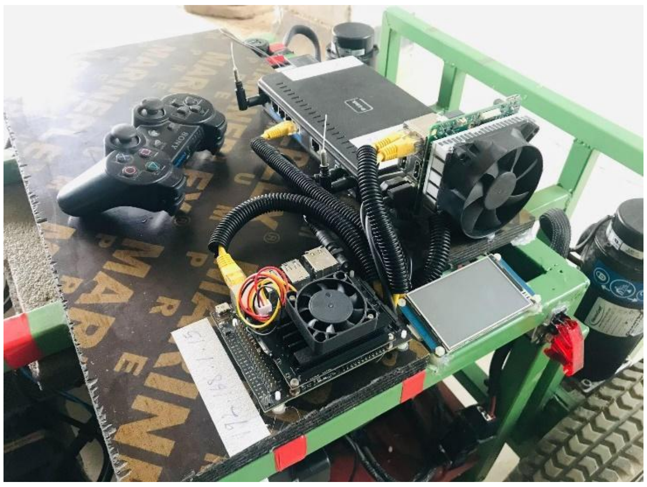


Figure 05: Robot with two SBCs

The Robot Operating System (ROS) was used as the software architecture. In the installation process, Ubuntu 18.04 for Intel UP2 and Nvidia Linux4Tegra Ubuntu (for the Nvidia Jetson Nano) flashed OS using ETCHER software using PC. Then installed ROS Melodic Morenia on boards.

In this project Python, C ++ and XML are used as programming languages. All sensor libraries and low-level programs written in C++ and high-level programs written in Python. XML is used to configure ROS launch files, Configuration Files and make Unified Robot Description Format (URDF) files.
Then the IMU and GPS sensors were implemented in the robot. Ublox NEO-7N-0-002 sensor used as GPS sensor and INVERSE SENSE MPU 9250 9-axis Motion-Tracking device used as IMU sensor. For the GPS sensor, homemade Enclosure was used for extra protection.
All sensor Inputs are converted to USB interface using STM32F4 and ATMEGA microcontrollers. It was a huge advantage for hardware driver compatibility with different computers. ROS_SERIAL was used to communicate with microcontrollers.

The sensors have process noise and measurement noise. For this Kalman filter is proposed for sensor fusion. Firstly, an Extended Kalman filter was used since the system is not linear. However, with the research we could find that unscented Kalman filters can be more accurate with higher nonlinear systems.
Implementation was started with small two sensor input MATLAB tutorials and got a better understanding about Unscented Kalman filters. Then the filter was developed with C++ and implemented in the robot.

For fast development and remote working, a simulated environment was required. GAZEBO simulator used with RVIZ visualization platform to develop and debug the codes and Algorithms.

LIDAR was used in robot navigation systems for obstacle avoidance purposes. In addition, it can be used to map outdoor environments but accuracy was small due to range limitations in sensors.

The LIDAR sensor was replaced later with the ZED^{TM} Stereo Camera which can be used as a LIDAR sensor. Also, the ZED^{TM} camera was used to generate 3D maps and 3D point cloud data for precise navigation in an outdoor environment.

This robot has a separate ROS node for each sensor and each node runs parallel when executing the main code. But ZED^{TM} Camera Node cannot execute on Intel ^{TM} UP2 SBC because it does not have a GPU for the Fast Calculation. A Nvidia Jetson Nano was used to execute the ZED camera Node. It has a Low-end GPU but it can work well when Only using ZED Camera Node. Ethernet interface was used to communicate between the Intel UP2 and NVIDIA^{TM} Jetson Nano. ZED^{TM} Camera was connected to the Jetson Nano via High-Speed USB 3.0 Interface.

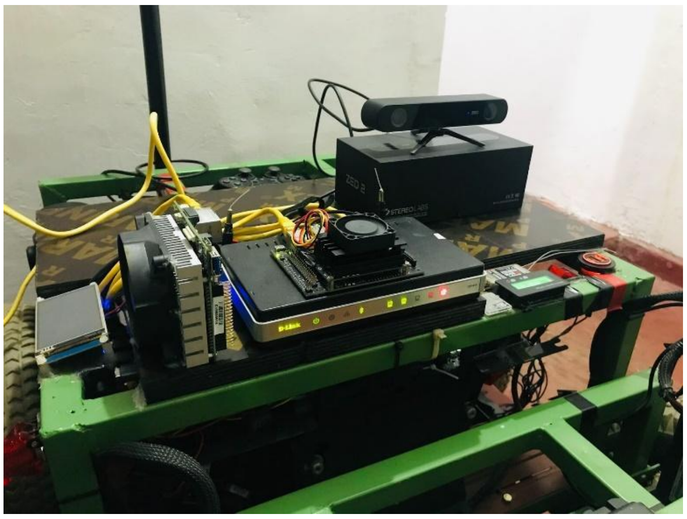


Figure 06: ZED Stereo Camera with Jetson Nano and Intel UP2 SBC

Networking between ROS Computers is important in this project. In Our project has three main computers to network,

1. Main Intel UP2 Computer which runs Robot Hardware, Navigation, Localization Controllers and Path planners.
2. Nvidia Jetson Nano which runs ZED Stereo Camera controllers for 3D Point cloud generation and map creation.
3. Intel Core i7 Desktop server which runs Deep Neural Networks for Steer command Prediction.

To get maximum data transfer speed between Intel Core i7 server computer, Jetson Nano and Intel UP2 we used Gigabit Ethernet between computers.
Additionally, a Laptop PC was used to connect all above three computers to execute command and state monitoring.
The Neural Network used the UNET model which uses InceptionV3 as the backbone. After that a suitable dataset for this project was created using the videos which were taken with a high-

resolution camera. For the labeling process ‘LabelMe’ software was used. After that the neural network was trained in google Colab^{TM} cloud computer using the dataset which was created earlier. And this process was done several times with slight modifications with the dataset until a highly accurate model was obtained.

Then the selected Keras models were converted to ONNX models to increase the Hardware Compatibility with CPUs and GPUs.
The segmentation models were implemented in the PC that was outside the vehicle and tested out for videos and camera feeds from ZED Camera streamed in a Local Area Network (Networked for ROS Communication and Video streaming with CAT6 Ethernet Cables)

Finally, for the DNN camera input, A ZED camera with 720p video recording capability was used and it was mounted on the robot with 60 CM height in ground for clear view.

4.2 Local Path Planner Using the Segmentation Model

- SEGMENTATION MODEL

Segmentation model was developed using the KERAS API in the TensorFlow framework. It is inspired by UNET architecture. But the encoder and the decoder of UNET are replaced by inceptionV3 neural networks that are constantly used in image classification tasks. The whole segmentation program was developed using a python environment and other than machine learning frameworks NUMPY and OPENCV libraries were used to handle the calculations and image operations.

After developing the and training the neural network, now the vehicle is capable of classifying road and background with separate colors. Then using those color details, a set of points in the 2D plane was extracted and those points are a 3D to 2D mapping of the visual data of the stereo camera. And those points represent the important spatial data about the environment and the drivable area.
After those points related to the left and right side of the drivable area were classified and they were used to calculate the angle between the horizontal axis of the robot and the road ahead. Also, the 2D coordinates were used to calculate the shift between the center of the robot and center of the drivable area. Both above details are needed as parameters to output a suitable steer angle. And a program that acts as a local path planner was developed and above data are the key parameters

that define the program. The whole purpose of the program is to output steer angles in such a way that the vehicle stays always in the middle of the drivable area.

The developed program is attached with ROS and in case it fails to predict a Steer angle the control will be given to the ROS GLOBAL PATH PLANNER and again it is possible to generate steer angles the control will be handed over to this program. It also happens if an obstacle is present in the drivable area.

4.3 Printed Circuit Board (PCB) Design

A printed circuit board is required because when a circuit is used in a project there can be loose connections and the circuit can be really large. These can be avoided using a printed circuit board.

This circuit board was designed using Altium Designer. Some components are integrated in the circuit board and some sensors can plug in to the circuit.

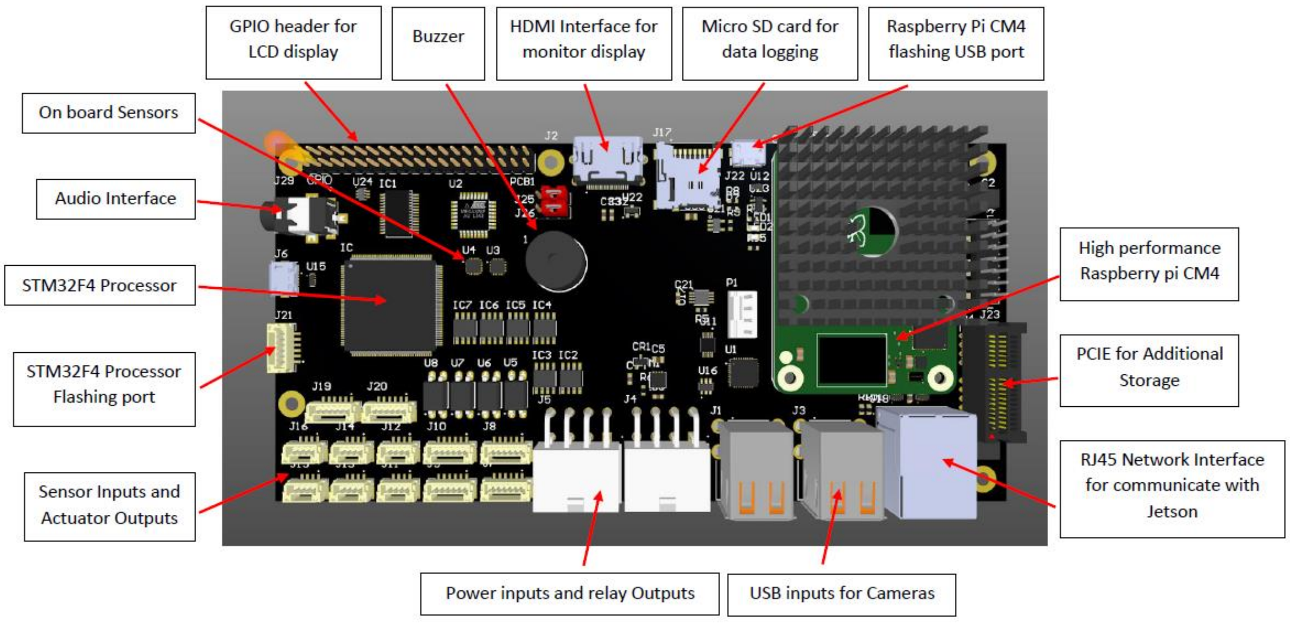

GPIO header for LCD display
Buzzer
HDMI Interface for monitor display
Micro SD card for data logging
Raspberry Pi CM4 flashing USB port
On board Sensors
Audio Interface
STM32F4 Processor
STM32F4 Processor Flashing port
Sensor Inputs and Actuator Outputs
High performance Raspberry pi CM4
PCIE for Additional Storage
RJ45 Network Interface for communicate with Jetson
Power inputs and relay Outputs
USB inputs for Cameras

Figure 07: Printed Circuit board for the project

CHAPTER 05 RESULTS AND DISCUSSIONS

5.1 Unscented Kalman Filter

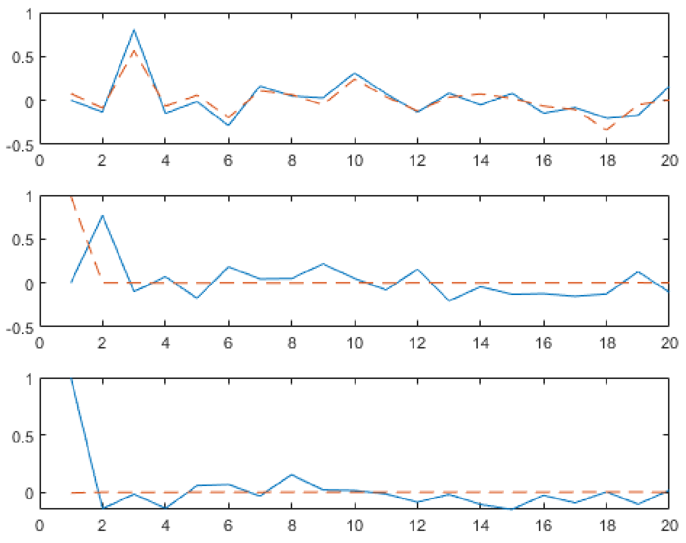


Figure 08: MATLAB Example with random noise

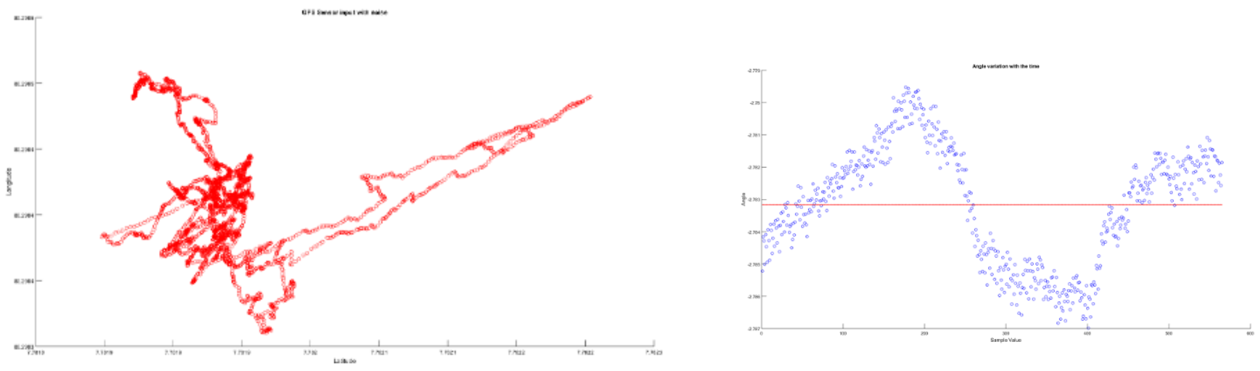


Figure 09: Noisy outputs of sensors

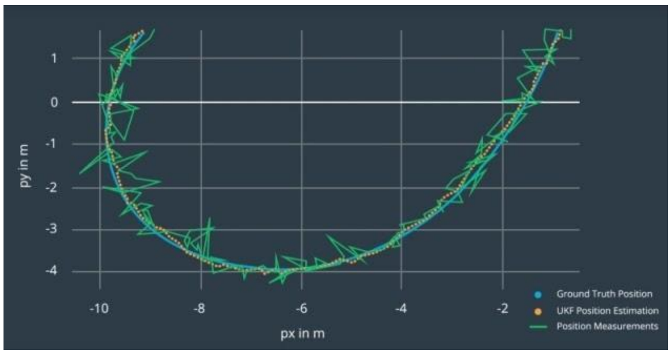


Figure 10: UKF Position Estimation with EKF with noisy inputs

Figure 10: Shows clearly that UKF gives a very good approximation of the position by taking noisy input signals. The sensors were tested without attaching them into the robot and movement was done referenced to the local frame.

5.2 Simulation Environment

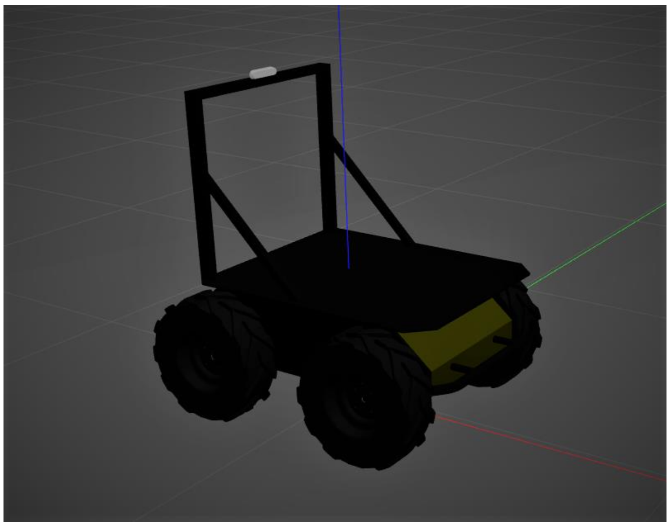


Figure 11: Modified Robot with ZED camera in GAZEBO environment

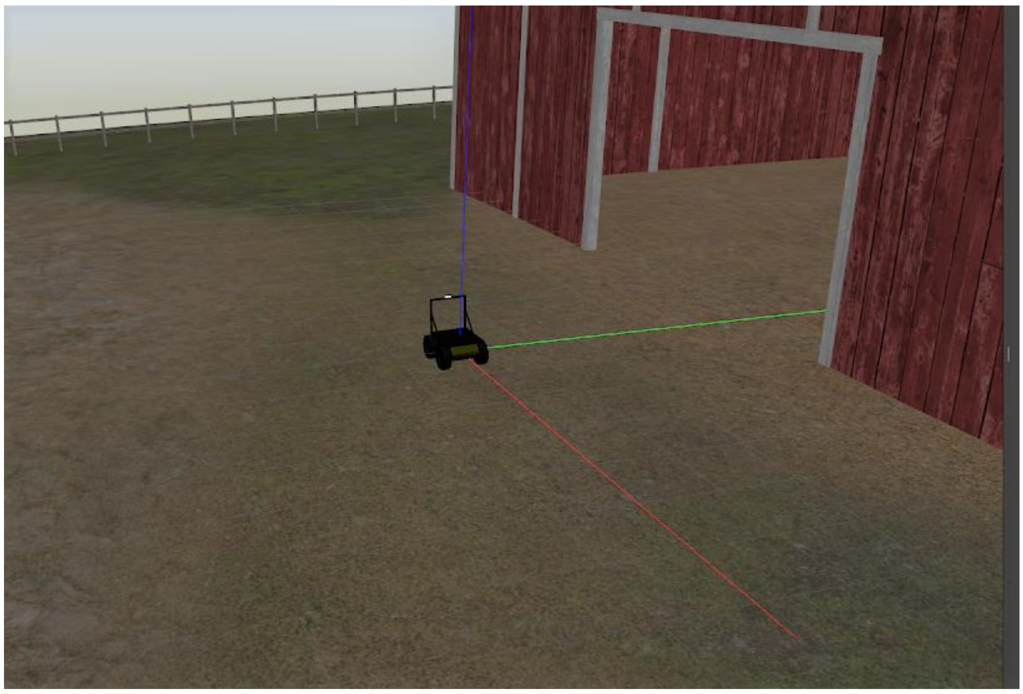


Figure 12: Created Simulated Agricultural Environment for testing

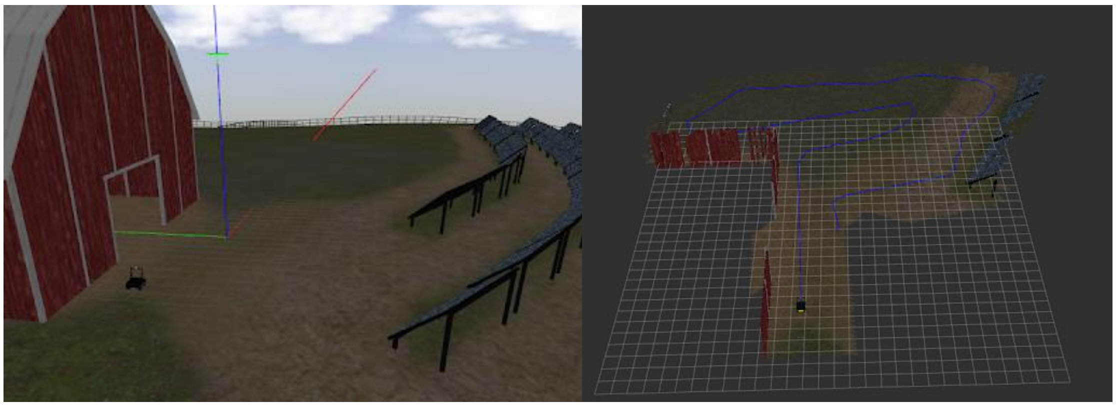
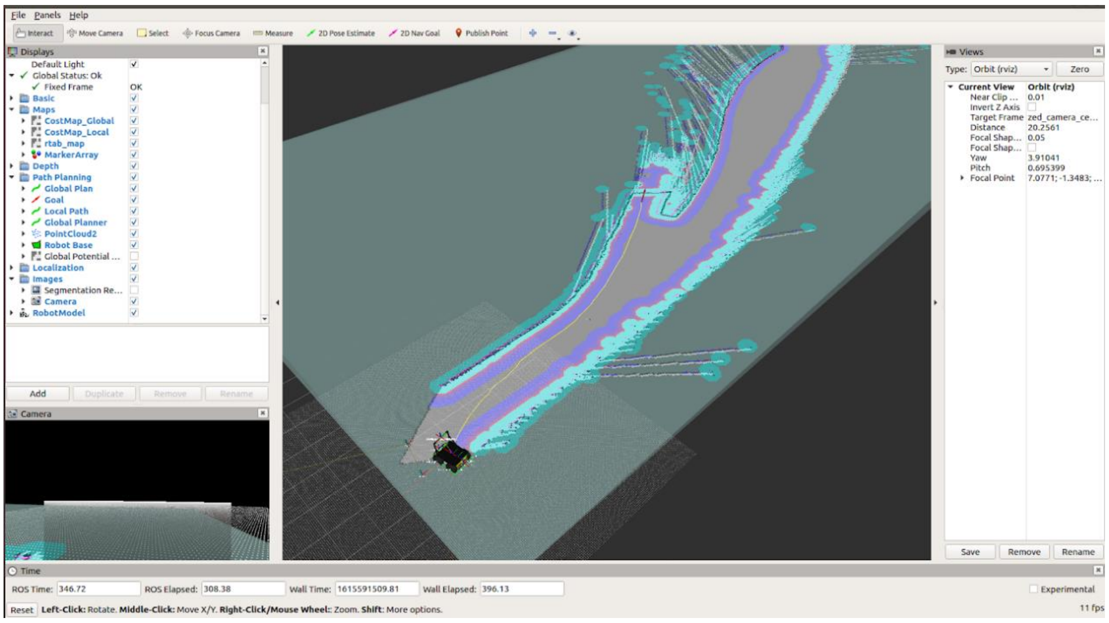


Figure 13: Generating 3D Point cloud map with ZED2 in Gazebo Simulated Environment



Displays

Default Light	✓
Global Status: Ok	
Fixed Frame	OK
Basic	✓
Maps	✓
CostMap_Global	✓
CostMap_Local	✓
rtab_map	✓
MarkerArray	✓
Depth	✓
Path Planning	✓
Global Plan	✓
Goal	✓
Local Path	✓
Global Planner	✓
PointCloud2	✓
Robot Base	✓
Global Potential ...	☐
Localization	✓
Images	✓
Segmentation Re...	☐
Camera	✓
RobotModel	✓

Camera

Views

Type: Orbit (rviz)

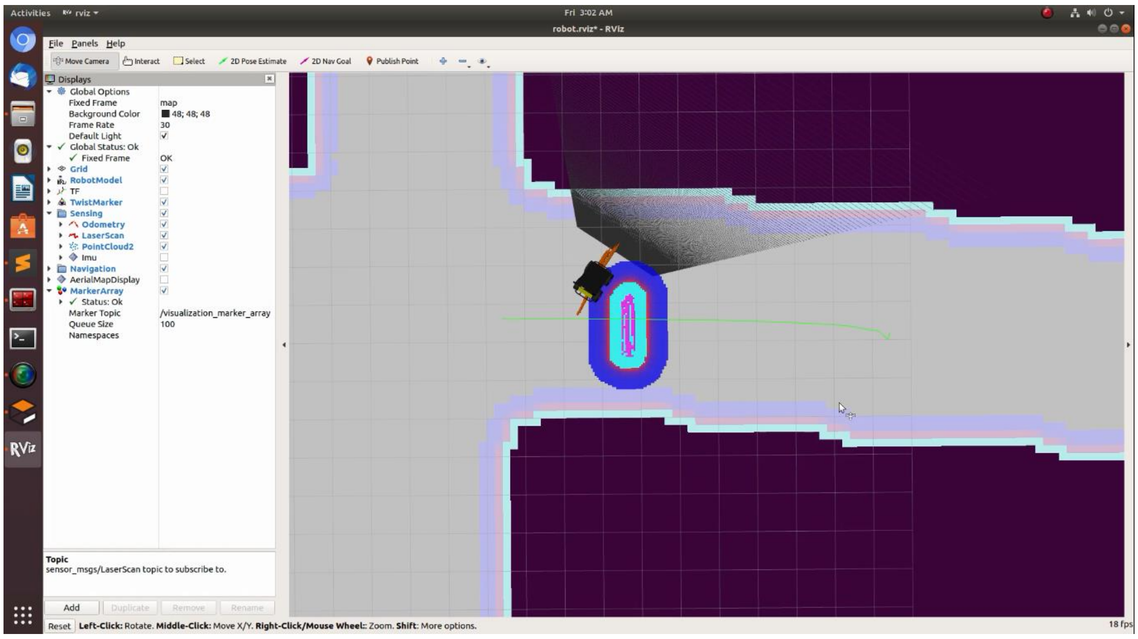
Current View	Orbit (rviz)
Near Clip ...	0.01
Invert Z Axis	☐
Target Frame	zed_camera_ce...
Distance	20.2561
Focal Shap...	0.05
Focal Shap...	☐
Yaw	3.91041
Pitch	0.695399
Focal Point	7.0771; -1.3483; ...

Time

ROS Time: 346.72 ROS Elapsed: 308.38 Wall Time: 1615591509.81 Wall Elapsed: 396.13 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 11 fps

Figure 14: Autonomous Navigation testing in simulated environment



Displays

Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
Global Status: Ok	
Fixed Frame	OK
Grid	✓
RobotModel	✓
TF	☐
TwistMarker	✓
Sensing	✓
Odometry	✓
LaserScan	✓
PointCloud2	✓
Imu	☐
Navigation	✓
AerialMapDisplay	☐
MarkerArray	✓
Status: Ok	
Marker Topic	/visualization_marker_array
Queue Size	100
Namespaces	

Topic
sensor_msgs/LaserScan topic to subscribe to.

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 18 fps

Figure 15: Autonomous Navigation with Obstacle Avoidance

As the initial phase of testing the developed systems, our team decided to use a simulated environment. In that simulated environment, drivable areas and obstacles could be created and an exactly similar virtual model of real-world hardware combination also could be simulated. Simply in the simulated environment, it was possible to build a virtual model of the same robot we had physically. It was a great advantage when it comes to power efficiency. A lot less power was used in simulations. Also, the damage to physical hardware was none when testing was conducted inside virtual environments. Also in this way, developers were able to test all the programs and if there were any bugs in programs, they could be easily debugged in simulation environments. After it was confirmed that a program is ready to use and bug free, then only the developers could implement them in the physical system. In that approach developed were able to guarantee the safety of valuable high-end hardware.

5.3 Visual Perception

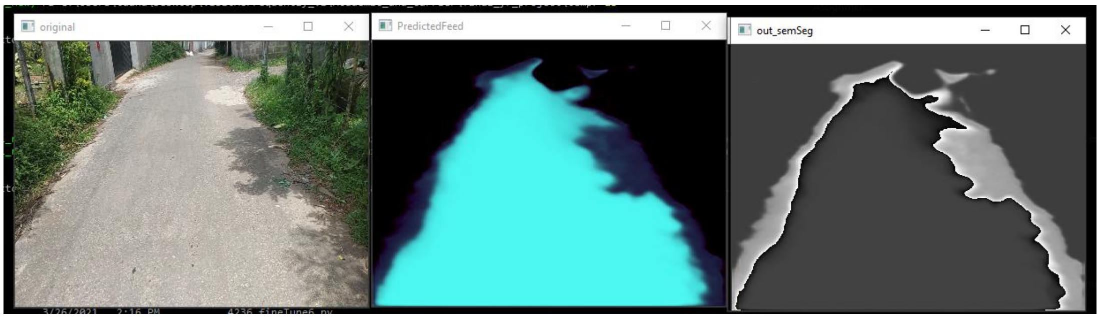


Figure 16: Initial segmentation results and noise filtered image result

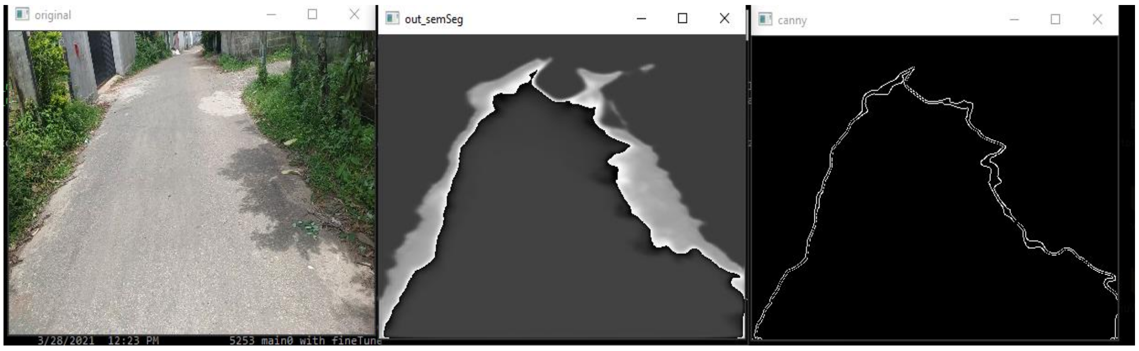


Figure17: Line detected image using noise filtered gray image

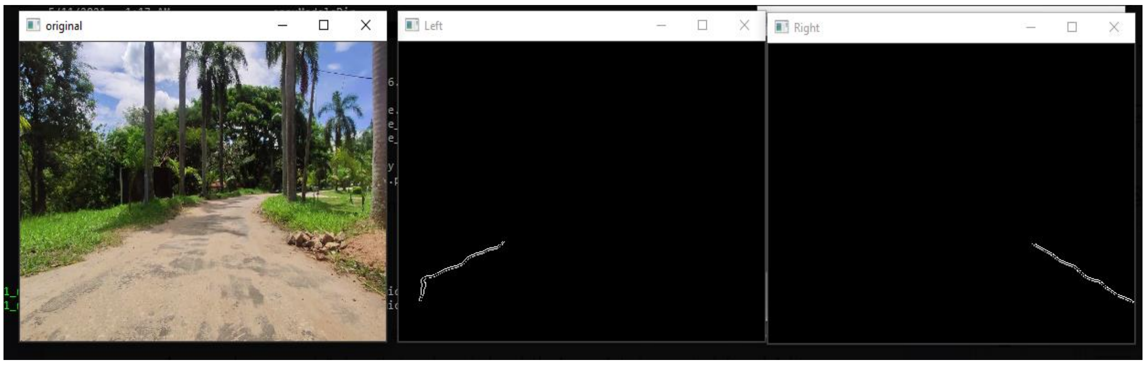


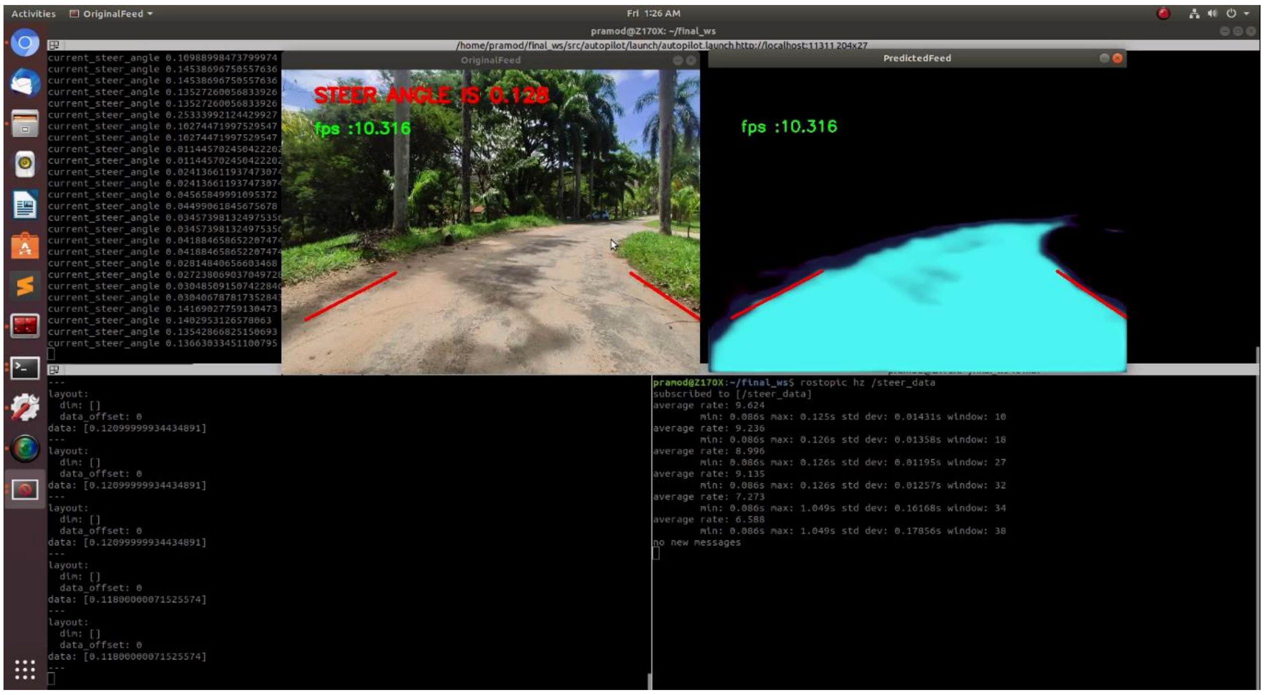
Figure 18: Extracting points related to left and right boundaries of drivable area

After successfully building the segmentation model any kind of drivable area could be detected with the color difference. But as shown in the results there is a little noise present in the segmented image. So, before any other image processing tasks, the noise had to be removed. For that an intensity thresholding function in the open computer vision library was used. After running it through the thresholding function, a gray image was obtained and that could be used to perform edge canny detection.

With edge canny detection the interested points of the frame could be filtered out and could be used to perform necessary calculations. A standard value for interested area with is crucial in steer result generation was selected with the testing. But if the road is too wide and if vehicle is in a junction, the interested area may be misidentified by the segmentation model. In such cases a method to pass the control to global path planner had to be developed. That part also completed successfully. Using the gradient data and offset data, the program was developed to generate steer angles.

After completing that part when the testing was conducted, sharp variations were noticed in the generated steer angles sometimes. The reason for the phenomena was poor estimation in segmentation model. Sometimes due to shadows and the backgrounds that is similar to the drivable area were main reasons for segmentation. Unfortunately, to eliminate those issues completely, a more accurate segmentation model was required to be built. But with the time limitations, it was not possible. So, interpolating the past steer angle data was a great help in predicting less variating steer angles. If a Steer angle is varying in higher variance that previous ones, the interpolated steer angle will be replaced instead of predicted steer angle.

After all the systems are implemented using the codes, a smooth flowing set of steer angles were generated and that system was implemented as the main local path planner of the vehicle.



```
current_steer_angle 0.10988998473799974
current_steer_angle 0.14538696750557636
current_steer_angle 0.14538696750557636
current_steer_angle 0.13527260056833926
current_steer_angle 0.13527260056833926
current_steer_angle 0.25333992124429927
current_steer_angle 0.10274471997529547
current_steer_angle 0.10274471997529547
current_steer_angle 0.011445702450422202
current_steer_angle 0.011445702450422202
current_steer_angle 0.024136611937473074
current_steer_angle 0.024136611937473074
current_steer_angle 0.04565849991095372
current_steer_angle 0.04499061845675678
current_steer_angle 0.034573981324975356
current_steer_angle 0.034573981324975350
current_steer_angle 0.041884658652207474
current_steer_angle 0.041884658652207474
current_steer_angle 0.02814840656603468
current_steer_angle 0.027238069037049728
current_steer_angle 0.030485091507422840
current_steer_angle 0.030406787817352843
current_steer_angle 0.14169027759130473
current_steer_angle 0.1402953126578063
current_steer_angle 0.13542866825150693
current_steer_angle 0.13663033451100795
```

```
---
layout:
  dim: []
  data_offset: 0
data: [0.12099999934434891]
---
layout:
  dim: []
  data_offset: 0
data: [0.12099999934434891]
---
layout:
  dim: []
  data_offset: 0
data: [0.12099999934434891]
---
layout:
  dim: []
  data_offset: 0
data: [0.11800000071525574]
---
layout:
  dim: []
  data_offset: 0
data: [0.11800000071525574]
---
```

```
pramod@Z170X:~/final_ws$ rostopic hz /steer_data
subscribed to [/steer_data]
average rate: 9.624
        min: 0.086s max: 0.125s std dev: 0.01431s window: 10
average rate: 9.236
        min: 0.086s max: 0.126s std dev: 0.01358s window: 18
average rate: 8.996
        min: 0.086s max: 0.126s std dev: 0.01195s window: 27
average rate: 9.135
        min: 0.086s max: 0.126s std dev: 0.01257s window: 32
average rate: 7.273
        min: 0.086s max: 1.049s std dev: 0.16168s window: 34
average rate: 6.588
        min: 0.086s max: 1.049s std dev: 0.17856s window: 38
no new messages
```

Figure 19: Deep Neural Network testing in faculty road

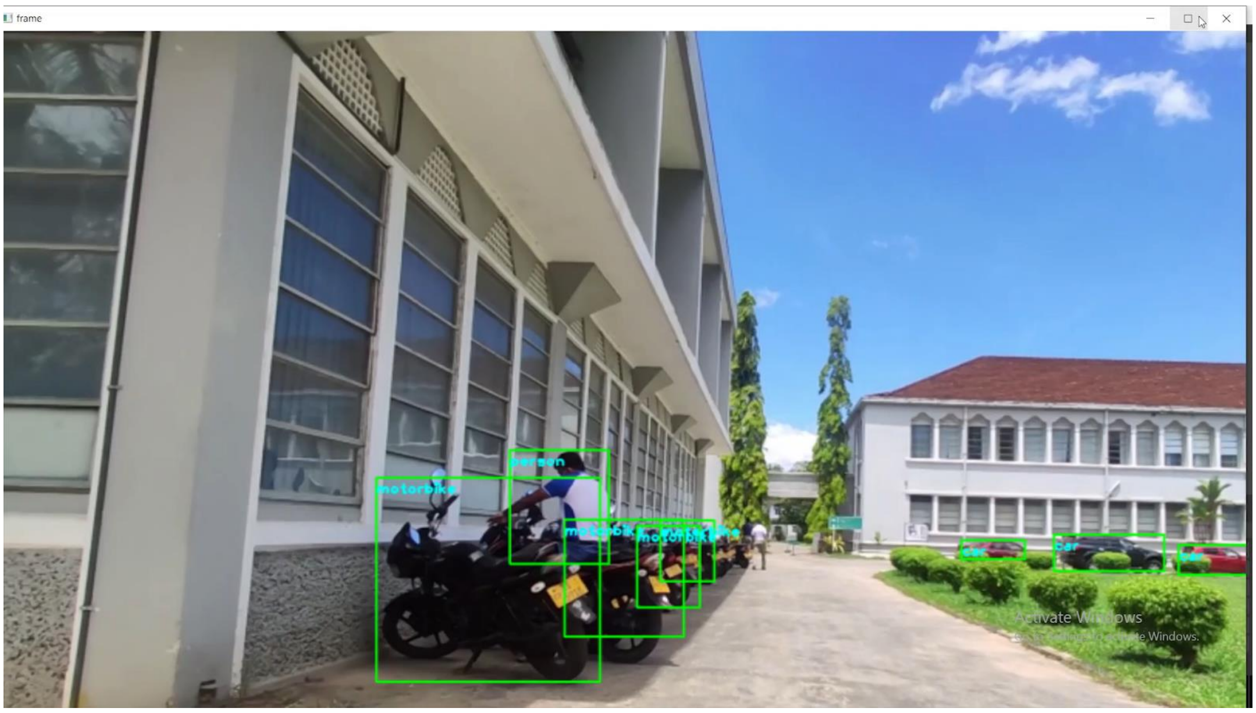


Figure 20: YOLO detection result

As another aspect of the visual perception, object detection and avoidance required to be accomplished. For that a YOLO-V4 model was trained on a dataset of vehicles, which are common obstacles related with self-driving tasks. After building the yolo model, the exact location of the obstacles within the drivable area could be obtained. The exact distances to those objects could be obtained using the depth map generated by the ZED camera. After fusing those data, a simple static obstacle avoidance system could be developed. When developing a static obstacle avoidance using YOLO and depth map, it was observed that the obstacle avoiding is not successful in case of dynamic objects and multiple objects. Therefore, after referring to some details about the ZED camera, our team found an inbuilt method of static and dynamic obstacle detection and that was optimized to perform the avoidance task. And it was faster and more accurate. Also, it worked better than the initial system. So, it was finally implemented as the obstacle avoidance system for the robot.

5.2 Network Configuration.

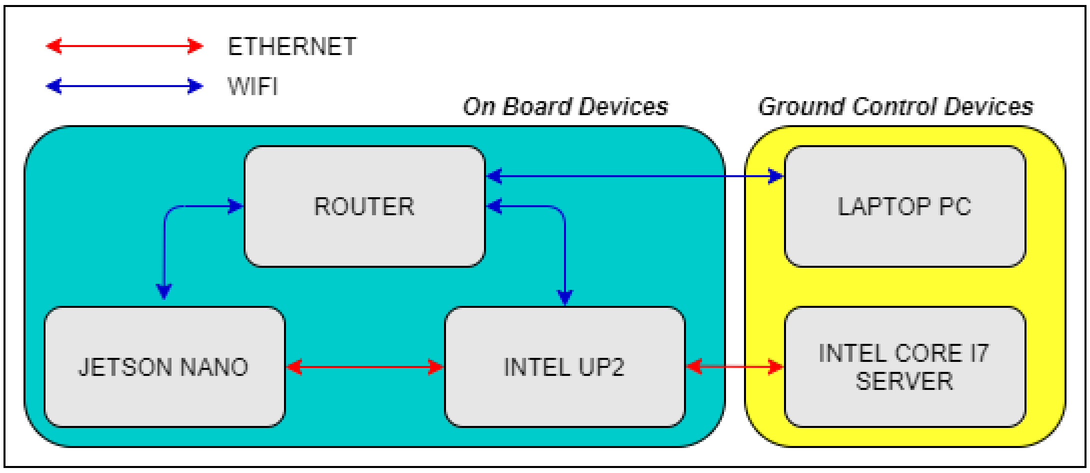


Figure 21: Computer Networking Diagram

For the Network static IP configuration was used. the configuration listed below,

- For Jetson Nano,

  Ethernet Interface: - 192.168.1.15

  WIFI Interface: - 192.168.70.12

- For Intel UP2,

  Ethernet Interface: - 192.168.1.10

  WIFI Interface: - 192.168.70.20

- For Intel Core i7 Server

  Ethernet Interface: - 192.168.1.20

A well-written ROS node makes no assumptions about where in the network it runs, allowing computation to be relocated at run-time to match the available resources. Deploying a ROS system across multiple machines is easy. Keep the following things in mind:

- Only need one master. Select one machine to run it on.

- All nodes must be configured to use the same master, via ROS_MASTER_URI.
- There must be complete, bi-directional connectivity between all pairs of machines, on all ports.
- Each machine must advertise itself by a name that all other machines can resolve.

When a ROS node advertises a topic, it provides a HOSTNAME: PORT combination (a URI) that other nodes will contact when they want to subscribe to that topic. It is important that the hostname that a node provides can be used by all other nodes to contact it. The ROS client libraries use the

Link to final videos: -
https://drive.google.com/drive/folders/1niAgvx93prHZrAyZlRQZjlMIXDn1qGUq?usp=sharing

CHAPTER 05 CONCLUSION

In this project, we have developed a fully functional Autonomous Vehicle. All the Algorithms are tested using simulations and prototype vehicle. These algorithms can be used for multiple purposes as examples security, warehouses, and delivery and home robots. This project was focused on making an Agricultural Multipurpose Autonomous Robot.

The Robot was built using Mild steel to obtain a strong and rigid structure. It weighs around 70kgs and can carry around 160kg. Brushless motors were used for wheels. In addition, a 90 Ah battery was used as the main power source.

Intel UP2 and a Jetson Nano board was used as the onboard computers and Intel core I7 Computer was used as the main server.
For the operating system of the robot, Robot Operating System with (ROS) which is vastly used for robot development OS used with Ubuntu OS.

Firstly, Manual control was implemented using a RF remote which has a 1km range. And Emergency Stop buttons were installed.

For the Autonomous part, an Inertial Measurement Unit, GPS and Wheel encoders were used with an Unscented Kalman Filter. Which can be used for highly nonlinear systems with low accurate sensors. And it gave reasonable results with the sensors.

Visual input was taken using a ZED stereo camera for generating 3D point cloud maps and Local Path Planning was done using Neural Networks. A segmentation model was developed to detect the drivable area and an object detection model is used for obstacle avoidance.
The PCB design was done but printing wasn’t possible due to the pandemic situation. Up to now, studies, simulation and hardware testing have successfully been conducted for vehicle navigation rough terrains avoiding dynamic obstacles.

CHAPTER 06 FUTURE WORK

Main objective of this project is to develop an Autonomous Multipurpose Agricultural vehicle. Up to now, developing Autonomous vehicles is finished and it can drive itself. PCB installation should be done as soon as we can manufacture it from Chinese PCB manufacturing company. Now we have to build specific hardware parts of the robot for agriculture. The development of neural networks also optimizes for the specific field.

Also, this control system should develop so that it can be implemented in the larger commercial vehicles such as Tractors and combined harvesters. Which will be really helpful for commercializing the product.

CHAPTER 06 REFERENCES

01. Xiaogang Wang, Jifeng Guo and Naigang Cui, "*Adaptive extended Kalman filtering applied to low-cost MEMS IMU/GPS integration for UAV,*" 2009 International Conference on Mechatronics and Automation, Changchun, 2009, pp. 2214-2218, doi: 10.1109/ICMA.2009.5246654.
02. Rudolph van der Merwe, E., 2021. The Unscented Kalman Filter for Nonlinear Estimation. [online] Seas.harvard.edu. Available at: <https://www.seas.harvard.edu/courses/cs281/papers/unscented.pdf> [Accessed 13 February 2021].
03. R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010.
04. B. H. Bodkin, Real-time mobile stereo vision [M.S. thesis], University of Tennessee, 2012
05. Kim, Y. and Bang, H., 2017. Introduction To Kalman Filter And Its Applications. [online] IntechOpen. Available at: <https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications> [Accessed 20 February 2021].
06. Cumming BG, DeAngelis GC. The physiology of stereopsis. Ann Rev Neurosci 2001; 24: 203–238.
07. Tweed D. Visual-motor optimization in binocular control. Vision Res 1997; 37(14): 1939–1951.
08. J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431-3440, doi: 10.1109/CVPR.2015.7298965.
09. Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.
10. B. Kang, Y. Lee and T. Q. Nguyen, "Depth-Adaptive Deep Neural Network for Semantic Segmentation," in IEEE Transactions on Multimedia, vol. 20, no. 9, pp. 2478-2490, Sept. 2018, doi: 10.1109/TMM.2018.2798282.
11. Xu, Qunshan & Jianghai, Zhao & Zhang, Chunxia & He, Feng. (2015). Design and implementation of an ROS based autonomous navigation system. 2220-2225. 10.1109/ICMA.2015.7237831.

12. Y. Huang, Y. Jing and Y. Shi, "Multi-sensor node fusion localization using unscented kalman filter in rough environments," 2018 Chinese Control And Decision Conference (CCDC), 2018, pp. 5476-5481, doi: 10.1109/CCDC.2018.8408085
13. Y. Li, M. Zhao, H. Li and G. Hao, "The Centralized Fusion Unscented Kalman Filter for Nonlinear System with correlated noise," 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2019, pp. 336-340, doi: 10.1109/IMCEC46724.2019.8983873.
14. R. A. Hamzah, A. M. A. Hamid and S. I. M. Salim, "The Solution of Stereo Correspondence Problem Using Block Matching Algorithm in Stereo Vision Mobile Robot," 2010 Second International Conference on Computer Research and Development, 2010, pp. 733-737, doi: 10.1109/ICCRD.2010.167.
15. S. Fazli, H. M. Dehnavi and P. Moallem, "A Robust Obstacle Detection Method in Highly Textured Environments Using Stereo Vision," 2009 Second International Conference on Machine Vision, 2009, pp. 97-100, doi: 10.1109/ICMV.2009.48.
16. Ki Yong Lee, Joon Woong Lee and Myeong Rai Cho, "Detection of road obstacles using dynamic programming for remapping stereo images to a top-view," IEEE Proceedings. Intelligent Vehicles Symposium, 2005., 2005, pp. 765-770, doi: 10.1109/IVS.2005.1505197.
17. Hara, Yoshitaka. (2017). Autonomous Navigation with ROS. Journal of the Robotics Society of Japan. 35. 286-290. 10.7210/jrsj.35.286.
18. Y. Song, T. Zhang, B. Li and H. Huang, "A Virtual Experiment Platform for 2D Robot Autonomous Navigation Algorithm System Based on ROS," 2018 IEEE International Conference on Information and Automation (ICIA), 2018, pp. 985-990, doi: 10.1109/ICInfA.2018.8812455.
19. R. Li, M. A. Oskoei, K. D. McDonald-Maier and H. Hu, "*ROS Based Multi-sensor Navigation of Intelligent Wheelchair,*" 2013 Fourth International Conference on Emerging Security Technologies, 2013, pp. 83-88, doi: 10.1109/EST.2013.35.
20. Nguyen, A. and Tran, Q., 2021. Autonomous Navigation with Mobile Robots using Deep Learning and the Robot Operating System. [online] arXiv.org. Available at: <https://arxiv.org/abs/2012.02417> [Accessed 20 March 2021].

G06

Project Title	VISUAL – LIDAR FUSION BASED DEEP LEARNING FRAMEWORK FOR AUTONOMOUS VEHICLE NAVIGATION		
Name of the supervisors	Prof. Lilantha Samaranayake		
Name of the students in the group	Telephone		email
	Mobile	*Home*	
Bandara N.K.B.L.M.	0713134121	0718881665	lakshanb@eng.pdn.ac.lk
Sandaruwan W.G.P.G.	0715875629		pramod.sandaruwan@eng.pdn.ac.lk
Deen T.A.	0763359039		e15070@eng.pdn.ac.lk