ROS Based Teleoperation and Docking of a Low Speed Urban Vehicle

A Thesis
Presented to
The Academic Faculty

By

Zulfiqar Haider Zaidi

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology
August 2020

ROS Based Teleoperation and Docking of a Low Speed Urban Vehicle

ACKNOWLEDGMENTS

I would like to thank Dr. Bras for the incredible opportunity he gave me when I joined the Sustainable Design and Manufacturing lab, along with his support and advice along this 2-year journey of completing research and building incredible projects.

I have a great deal of gratitude towards my committee members, Dr. Mazumdar and Dr. Simmons, who have taken time to read and improve the work in this thesis.

To the great people that have come and gone (or remain) in the SDM lab, I would like to express feelings of friendship and happiness. I cannot imagine having been grouped with more intelligent and proactive people, while still down to earth and always willing to have a good laugh or talk about meaningful events.

I would like to thank my family, who have always encouraged me to push through and overcome obstacles in my road.

Finally, I would like to thank my friends, who offered me incredible support throughout my time at Georgia Tech. I can honestly say that none of this would be possible without their support, guidance, and motivation.

TABLE OF CONTENTS

LIST OF TABLES vii

LIST OF FIGURES viii

Summary xiii

1 Introduction 1

1.1 Motivation 1

1.2 Objectives 2

1.3 Thesis Structure 4

2 Literature Review 5

2.1 Types of Teleoperated Systems 5

2.1.1 Master – Slave 5

2.1.2 Supervisory – Subordinate 6

2.1.3 Partner – Partner 7

2.1.4 Teacher – Learner System 7

2.1.5 Autonomous System 8

2.2 Current Applications of Teleoperation 8

2.2.1 Space Applications 9

2.2.2 Military/Defense Applications: 9

2.2.3 Security Applications 10

2.2.4 Toxic Environments 10

2.2.5 Underwater Vehicles 10

2.2.6 Forestry and Mining Industry 10

2.2.7 Telesurgery 11

2.2.8 Telepresence 11

2.3 Review of Teleoperation related Literature 12

2.4 Teleoperated Vehicles in Industry 17

2.5 Different Levels of Autonomy for Cars 18

2.6 Summary 20

3 Teleoperated System Description 21

3.1 General Teleoperation Scheme: 22

3.1.1 Basics of ROS Network 23

3.1.2 Issues in using ROS for teleoperation over the 4G LTE network 24
3.1.3 Potential Solutions for building a Remote ROS network 25
3.1.4 Chosen Solution for establishing a Remote ROS network 28
3.1.5 General Architecture of the Teleoperation System 28
3.2 Control Station/ Local Operator Station Environment 30
3.2.1 Operator Station Information Flow 30
3.2.2 Logitech 27 Racing Wheel 31
3.2.3 Graphical User Interface 34
3.3 Remote Vehicle Environment 35
3.3.1 Remote Environment Information Flow 36
3.3.2 Steering Wheel Control 38
3.3.3 Gas Pedal Control 38
3.3.4 Brake Pedal Control 39
3.3.5 Ignition and Gear Control 40
3.3.6 Camera Angle Control: 40
3.4 Summary 40
4 Auxiliary Systems 42
4.1 Video Feed Processing 42
4.1.1 Pixel-Wise Image Segmentation or Semantic Segmentation 42
4.1.2 Object Detection and Tracking 48
4.1.3 Steering Wheel Overlay 64
4.2 LIDAR Data 65
4.2.1 Data from the LIDAR 66
4.2.2 Mounting position of LIDAR 67
4.2.3 Safety System based on LIDAR 67
4.2.4 Number of Obstacles based on LIDAR data 70
4.2.5 Heading (Angle) at which each obstacle is present 71
4.2.6 Width of each Obstacle using LIDAR Data 71
4.2.7 LIDAR Feedback to Operator 73
4.3 Speed of Golf Cart 74
4.4 Summary 76
5 Experimental Results 77

5.1 Testing Setup using Human Operator 77
5.1.1 Setup and Results for Forward Trajectory Task 77
5.1.2 Setup and Results for Reverse Trajectory Task 78
5.2 Quantifying Delay in Communication 80
5.2.1 Delay in Communication from Remote Vehicle to Control Station 80
5.2.2 Delay in Communication from Control Station to Remote Vehicle 99
5.2.3 Effects on Teleoperation during Periods of Extended Communication Delay (> 1 seconds) 108
5.3 Summary 109
6 Autonomous Docking 111
6.1 Differential GPS 111
6.1.1 Basics of DGPS 111
6.1.2 DGPS Used in our System 112
6.2 QR Tracking 113
6.2.1 Reasons for using QR code as a Visual Marker 113
6.2.2 Visual Servoing based on QR Code Tracker 114
6.3 Docking System 115
6.4 Experimental Setup and Results 116
6.4.1 Issues with the Docking Procedure 119
6.4.2 Illustration of Issues with Final Orientation of the Vehicle w.r.t the Wall 120
6.5 Summary 121
7 Conclusions and Future Work 123
7.1 Summary 123
7.2 Achievements 124
7.3 Improvements 126
7.3.1 Improvements in Teleoperation System 126
7.3.2 Improvements in Docking System 127
7.4 Future Work 128
REFERENCES 130

LIST OF TABLES

Table 3-1: Control Signals read from the racing wheel and sent to the remote vehicle 32

Table 4-1: Classes of Object detected by the object detection network.. 52

Table 4-2: Table showing the details of the information published about detected objects 54

Table 4-3: RPLIDAR A1 sample point data information 66

Table 4-4: Example Data used to show steps followed to calculate obstacle width 72

Table 5-1: Forward trajectory tasks results 78

Table 5-2: Reverse trajectory task results.. 79

Table 6-1: Passed and Failed tests for docking procedure 119

LIST OF FIGURES

Figure 2-1: Master - Slave System (Swanson, 2013) 6

Figure 2-2: Supervisory – Subordinate System (Swanson, 2013) 6

Figure 2-3: : Partner - Partner (Swanson, 2013) 7

Figure 2-4: Teacher - Learner System (Swanson, 2013) 8

Figure 2-5: Fully Autonomous System (Swanson, 2013) 8

Figure 2-6: In the predictive display system setup, the vehicle states are predicted and camera images are enriched by the predicted vehicle position before they are displayed to the operator (Chucholowski, Buchner, Reicheneder, & Lienkamp, 2013) 13

Figure 2-7: Teleoperated driving within a narrow passage using visualization of the remote vehicle in the virtual environment (Hosseini & Lienkamp, 2016) 15

Figure 2-8: The virtual vehicle surroundings reconstructed using the transmitted occupancy grid map, seen within HMD from (a) operator's point of view and (b) perspective view (Hosseini & Lienkamp, 2016) 16

Figure 2-9: "Free Corridor" approach in animation (Tang, Vetter, Finkl, Figel, & Lienkamp, 2014) 17

Figure 3-1: General Teleoperation Configuration (Stanford Artificial Intelligence Laboratory et al., 2016) 22

Figure 3-2: Several components together on a ROS network (Stanford Artificial Intelligence Laboratory et al., 2016) 23

Figure 3-3: Local ROS network (Stanford Artificial Intelligence Laboratory et al., 2016) 24

Figure 3-4: Remote ROS network (Stanford Artificial Intelligence Laboratory et al., 2016) 25

Figure 3-5: Cloud-based solution for Remote ROS network 26

Figure 3-6: VPN connectivity overview 27

Figure 3-7: General system architecture for the Teleoperation System 29

Figure 3-8: Local Operator Station Nodes and Information Flow.. 30

Figure 3-9: Logitech G27 Racing Wheel 31

Figure 3-10: Data extraction from the Logitech G27 Racing Wheel 32

Figure 3-11: Correspondence between different parts of Logitech G7 Racing Wheel and various control signals.. 33

Figure 3-12: Control station set up in lab 34

Figure 3-13: Snapshot of GUI at the operator station 35

Figure 3-14: Remote Environment Information Flow.. 36

Figure 3-15: Feedback Control for the steering wheel 38

Figure 3-16: Gas Pedal mapping from Control Station to the Remote Vehicle 39

Figure 3-17: Brake Pedal mapping from Control Station to the Remote Vehicle.................................... 40

Figure 4-1: An example of Segmented Image... 43

Figure 4-2: A low resolution segmentation prediction map that illustrates the pixel-wise segmentation concept (Jordan) 43

Figure 4-3: Segmentation Network Architecture (Long, Shelhamer, & Darrell, 2015)............................ 45

Figure 4-4: Residual Learning: A building block (He, Zhang, Ren, & Sun, 2016)...................................... 46

Figure 4-5: Architecture of ResNet18 (He, Zhang, Ren, & Sun, 2016) 46

Figure 4-6: Basic Architecture of FCN 47

Figure 4-7: Classes detected by semantic segmentation network along with the color codes 48

Figure 4-8: Using Object detection to locate and detect items in an Image 49

Figure 4-9: Representation of bounding box using Boundary Coordinates 50

Figure 4-10: MobileNtV2 architecture (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018) 51

Figure 4-11: Schematic Architecture of the SSD network used in our system 51

Figure 4-12: Output Image with Bounding Boxes and prediction confidence 53

Figure 4-13: Area of Intersection (Left); Area of Union (Right) 61

Figure 4-14: Representation of IoU matrix 62

Figure 4-15: Flowchart explaining how i) Matched detections ii) Unmatched Detections iii) Unmatched Trackers are added or removed from the Good Tracker list 64

Figure 4-16: Ax example image with steering guidelines overlaid on the image being transmitted to the operator 65

Figure 4-17: An Image of RPLIDAR A1 system 65

Figure 4-18: The array with distance (m) and heading angle (degrees) 66

Figure 4-19: Position of LIDAR w.r.t. the golf cart 67

Figure 4-20: Graphical illustration of virtual envelope in front of golf cart 69

Figure 4-21: Illustration of sample points inside the safety envelope 70

Figure 4-22: An example showing 3 obstacles (as there are 3 groups of non-zero values) 71

Figure 4-23: A triangle explaining the Cosine Rule 72

Figure 4-24: Width of an Obstacle based on Law of Cosines 73

Figure 4-25: An image showing the LIDAR data being transmitted to the operator 74

Figure 4-26: Schematic for the Hall effect sensor circuit to measure RPM 75

Figure 5-1: Route for Forward Trajectory Task 77

Figure 5-2: Route for Reverse Trajectory Task 79

Figure 5-3: Lag in Video Feed at 11 am 81

Figure 5-4: Box plot for Video Feed latency at 11 am 82

Figure 5-5: Lag in Video Feed at 1 pm 83

Figure 5-6: Box plot for Video Feed latency at 1 pm 83

Figure 5-7: Lag in Video Feed (seconds) at 3 pm 84

Figure 5-8: Box plot for Video Feed latency at 3 pm 85

Figure 5-9: Lag (seconds) in Video Feed at 5 pm 85

Figure 5-10: Box plot for Video Feed latency at 5 pm 86

Figure 5-11: Lag (seconds) in Video Feed at 7 pm 87

Figure 5-12: Box plot for Video Feed latency........ 87

Figure 5-13: Lag (seconds) in Video Feed at 9 pm 88

Figure 5-14: Box plot for Video Feed latency at 9 pm 88

Figure 5-15: Variability in Video Feed latency over the day 89

Figure 5-16: Box plot for latencies in Video Feed at different times of the day........ 90

Figure 5-17: Lag (seconds) in LIDAR data feed at 11 am........ 91

Figure 5-18: Box plot for latency in LIDAR data feed 91

Figure 5-19: Lag (seconds) in LIDAR data feed........ 92

Figure 5-20: Box plot for latency in LIDAR data feed 92

Figure 5-21: Lag (seconds) in LIDAR data feed........ 93

Figure 5-22: Box plot for latency in LIDAR data feed 94

Figure 5-23: Lag (seconds) in LIDAR data feed........ 94

Figure 5-24: Box plot for latency in LIDAR data feed 95

Figure 5-25: Lag (seconds) in LIDAR data feed........ 96

Figure 5-26: Box plot for latency in LIDAR data feed 96

Figure 5-27: Lag (seconds) in LIDAR data feed........ 97

Figure 5-28: Box plot for latency in LIDAR data feed 97

Figure 5-29: Variability in LIDAR Data Feed latency over the day 98

Figure 5-30: Box plot for latencies in LIDAR Data Feed at different times of the day........ 99

Figure 5-31: Lag (seconds) in Control Station data feed........ 100

Figure 5-32: Box plot for latency in Control Station data feed 100

Figure 5-33: Lag (seconds) in Control Station data feed at 1 pm 101

Figure 5-34: Box plot for latency in Control Station data feed at 1 pm 101

Figure 5-35: Lag (seconds) in Control Station data feed at 3 pm 102

Figure 5-36: Box plot for latency in Control Station data feed at 3 pm 102

Figure 5-37: Lag (seconds) in Control Station data feed at 5 pm 103

Figure 5-38: Box plot for latency in Control Station data feed at 5 pm 104

Figure 5-39: Lag (seconds) in Control Station data feed at 7 pm 104

Figure 5-40: Box plot for latency in Control Station data feed at 7 pm 105

Figure 5-41: Lag (seconds) in Control Station data feed at 9 pm 106

Figure 5-42: Box plot for latency in Control Station data feed at 9 pm 106

Figure 5-43: Variability in Control Station data feed latency over the day 107

Figure 5-44: Box plot for latencies in Control Station data feed at different times of the day 108

Figure 6-1: Schematic diagram of DGPS unit (Allen, Wolfe, Judy, Haukkala, & James, 2001) 112

Figure 6-2: Signal Flow schematic in our DGPS 113

Figure 6-3: Flowchart explaining the QR code tracking process 115

Figure 6-4: Docking system based on DGPS and QR tracking 116

Figure 6-5: Setup to validate the docking procedure 117

Figure 6-6: QR code mounted on the wall serving as a visual marker 118

Figure 6-7: Example of a passed test 118

Figure 6-8: Example of a failed test 119

Figure 6-9: Angle α of the vehicle w.r.t wall 120

Figure 6-10: Variation in final orientation of vehicle w.r.t. the wall 121

Summary

In recent years, 4G LTE technology has provided us with higher than ever transfer speeds over the cellular networks, permitting streaming of video and other high bandwidth services. On the other hand, there has been a rapid development and an explosion of interest in frameworks for robot software development, particularly ROS. Though there have been many studies which have leveraged 4G LTE network as the mode of communication when studying teleoperations, a very few studies have used 4G LTE network with ROS framework for building teleoperated systems.

Therefore, this study seeks to build a teleoperated system using the ROS framework which employs the 4G LTE network for communication. For this purpose, a prototype system is built using a remote-controlled low speed urban vehicle that hosts a multimedia link between the vehicle and the control station. The operator drives the vehicle remotely primarily based on processed video feed and LIDAR data. The vehicle is also equipped with safety systems to avoid collisions. The teleoperated system built is tested by asking an experienced driver to complete certain tasks while driving the vehicle remotely.

Moreover, this study also intends to build an autonomous docking procedure for the vehicle. A docking procedure based on differential GPS and video feedback is built that allows the vehicle to autonomously dock itself into a charging station. The procedure provides a proof of concept solution for the autonomous charging/fueling of self-driving cars.

1 Introduction

1.1 Motivation

Teleoperation is a field that started in the middle of twentieth century. Since then it has evolved on a very fast pace. Teleoperation is a very broad field that combines wide array of disciplines and is providing benefits to several industries ranging from sociology to space exploration. Though it may encompass a plethora of fields, at its roots teleoperation is simply a system that encompasses a master, or operator station; a slave, or remote environment; and a communications link that bridges both sides.

Teleoperation has been significantly improved over the years by making improvements in the operator station, the remote environment, and the communications link. With time, more complicated system has been developed on the master side due to advances in user interfaces and augmented reality leading to a higher situational awareness (SA). Similarly, the slave side has also evolved considerably allowing the operator much more minute control over the remote environment. All this while, vast leaps have been made in communication technology that has allowed for long-range control and real-time feedback.

Communication technology has vastly evolved since the time teleoperated systems were first introduced. Originally, the teleoperation was conducted through wired connections between the master and slave systems. As telecommunication systems evolved int the radio frequency spectrum, it gave the ability to teleoperated systems to be controlled wirelessly. The extent and range of teleoperation depended upon the communication infrastructure. The advent of internet marked the democratization of long-range communications. With time the data transfer rates increased and latencies in the system decreased which improved the teleoperated systems. However, the “last mile” implementation of wireless systems still depended on installing hardware that have short range limitations, like Wi-Fi.

With the emergence of 4G cellular technology, the “last mile” problem can be solved by employing these cellular technologies on the remote environment side. In chapter 2, we will look at studies that have already looked at the potential of this technology in teleoperated systems. Many investigations claim a successful implementation of a teleoperated system based on 4G communication. However, this does not mean that this technology is reliable and effective in controlling a teleoperated system. Hints of this can be found in authors’ conclusion which recommend development of communication loss protocols and adding some level of autonomy to the system.

Over the last few years, a lot of open-source robotics software platforms have been developed. These frameworks allow for rapid development by providing tools and libraries that provide hardware abstraction, message-passing, libraries, visualizers, package management and much more. Development of teleoperated systems around these frameworks would be the next logical step in evolution of teleoperation. One such widely used robotics development software platform is ROS which stands for Robot Operating System). It is the intent of this study to develop a teleoperation system architecture that is built around 4G cellular technology and ROS.

The other focus of this thesis is to develop a docking system for autonomous vehicles. There has been an explosion of interest and research in the development of autonomous vehicles. Autonomous vehicles will be a reality in near future. The autonomous vehicles will have to be charged/fueled autonomously to be truly autonomous. This requires development of a system that enables the vehicle to dock into a charging station. This study’s second focus would be the development of such a system.

1.2 Objectives

The primary objective of this research is to build a teleoperated system that allows an operator to control a vehicle remotely over a cellular network. To leverage the benefits that come from using a robotics software framework, it is our objective to build the entire system around ROS. With every teleoperated system there is a possibility of communication loss or limited situational awareness that might result in an accident. Therefore, this research also focuses on the development of safety systems for collision avoidance. Moreover, this study also focuses in employing latest advances in the field of computer vision to increase the situational awareness of the operator.

SAE International's Standard J3016 define six level of automation for automakers and policy makers to use to classify a system's sophistication (SAE, 2018). The six levels of autonomy are range from no autonomy to fully autonomous vehicles. These levels of autonomy will be further discussed in Chapter 2. The objective of this thesis is not to build a Level 5 autonomous vehicle. We are building a system which can be operated by a human operator remotely. It provides a solution for Level 2 and upwards automated vehicle to be controlled by a human who is not physically present in the car themselves if a scenario arises which the automated system cannot deal with by itself. Moreover, as the system is built around ROS, we can use the capabilities and tools provided by ROS for further development work on the vehicle.

Moreover, use of ROS would allow our system architecture to be much simpler as compared to building our own communication channel. In the study, Evaluation of Teleoperation System Performance Over a Cellular Network, (Sepulveda, 2016) a custom web application to connect remote vehicle to the control station was built. This made the system architecture much more complex. If all the communication protocols are handled by ROS, then the system architecture will be much simpler.

Another important consideration is to keep the overall cost of the system as low as possible. Low speed vehicles are conventionally much cheaper than high speed vehicles. They are usually used in golf courses, resorts, hotels, college campuses, airports, and industrial facilities. These vehicles are used because of their low initial cost and maintenance expenses. Thus, any system which is built for these types of vehicles should be low-cost and use easily available commercial parts. If the system is very expensive, it will make it unfeasible to be deployed in low speed urban vehicles.

The secondary objective of this research is to build an autonomous docking system for an autonomous vehicle based on vision and location data. This will allow the vehicle to be charged automatically without requiring human aid to plug it in.

1.3 Thesis Structure

With the intent of achieving the objectives mentioned in the previous section, this thesis is structured in the following way. Chapter 2 focuses on providing relevant information about teleoperation, and the research performed previously. Chapter 3 focuses on the development of the teleoperation system architecture that was built in this thesis. Chapter 4 focuses on the additional systems that were incorporated to increase the safety of the teleoperated system as well as improve the situational awareness (SA) for the operator. Chapter 5 focuses on the experimental results obtained by asking different users to operate the system. Chapter 6 focuses on the development of an autonomous docking system to dock the vehicle autonomously into the charging station. Chapter 7 concludes the thesis by summarizing the work done and detailing possible future research avenues.

2 Literature Review

Teleoperation refers to the control of a device or machine remotely. (Sheridan, 1995) defines teleoperation as follows:

A teleoperator is a machine enabling a human operator to move about, sense and mechanically manipulate objects at a distance. It usually has artificial sensors and effectors for manipulation and/or mobility, plus a means for the human to communicate with both. Most generally, any tool which extends a person's mechanical action beyond his reach is a teleoperator. (Sheridan, 1995)

2.1 Types of Teleoperated Systems

Based on the type of control structure used, teleoperation can be divided into following sub-categories (Swanson, 2013):

- Master – Slave
- Supervisory – Subordinate
- Partner – Partner
- Teacher – Learner
- Fully Autonomous

A brief description of them is as follows:

2.1.1 Master – Slave

Every teleoperation system consists of an operator environment and a remote environment, which are linked together using a communication protocol.

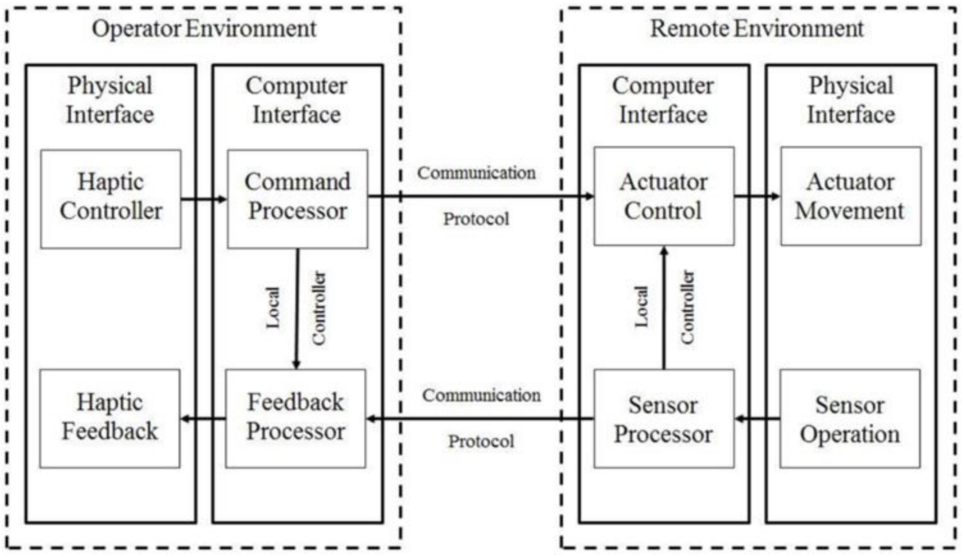


Figure 2-1: Master - Slave System (Swanson, 2013)

The controller inputs the commands through the physical interface to the command processor that transmits the commands to the actuator controller present on the remote environment. It, then commands the actuator movement that generates a reaction in the remote environment. Simultaneously, the sensory feedback is collected on the remote environment and is transmitted to the operator. Based on the feedback, the operator generates a new command and the cycle continues.

2.1.2 Supervisory – Subordinate

Unlike Master – Slave system, this type of control structure allows different levels of autonomy.

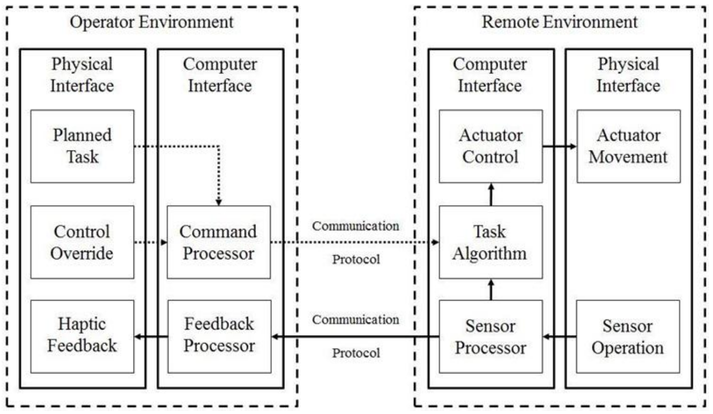


Figure 2-2: Supervisory – Subordinate System (Swanson, 2013)

There are computational resources present on the remote side that allows execution of preplanned tasks. Continuous feedback is provided to the operator environment. The feedback ensures that the remote system is working correctly. In case a corrective measure is needed, the operator can send instructions to the task algorithm to complete.

2.1.3 Partner – Partner

This configuration is different in the sense that the actuation takes place in the operator environment instead of the remote environment. The remote environment helps the operator by sending sensing and corrective signals to the operator environment.

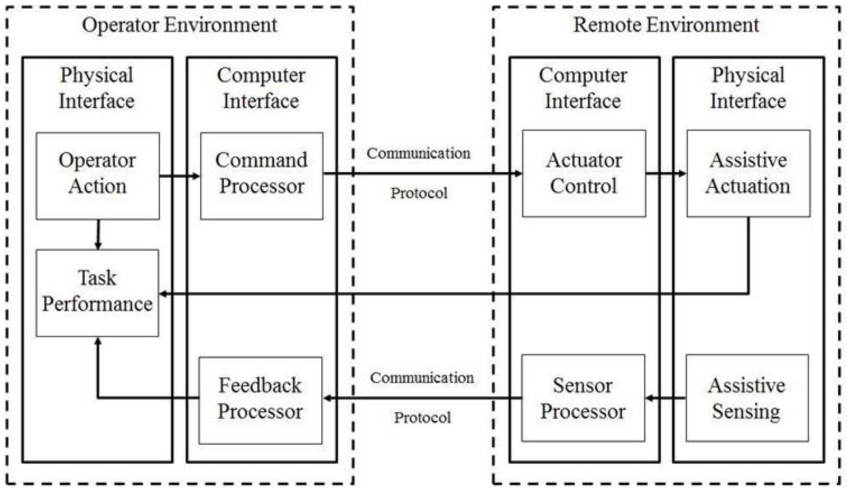


Figure 2-3: : Partner - Partner (Swanson, 2013)

2.1.4 Teacher – Learner System

Teacher – Learner system is a mix between Master – Slave and Supervisory – Subordinate system. The difference is in the fact that in this system the remote environment has a learning algorithm block that progressively assumes more control of the remote side and eases the strain of the operator.

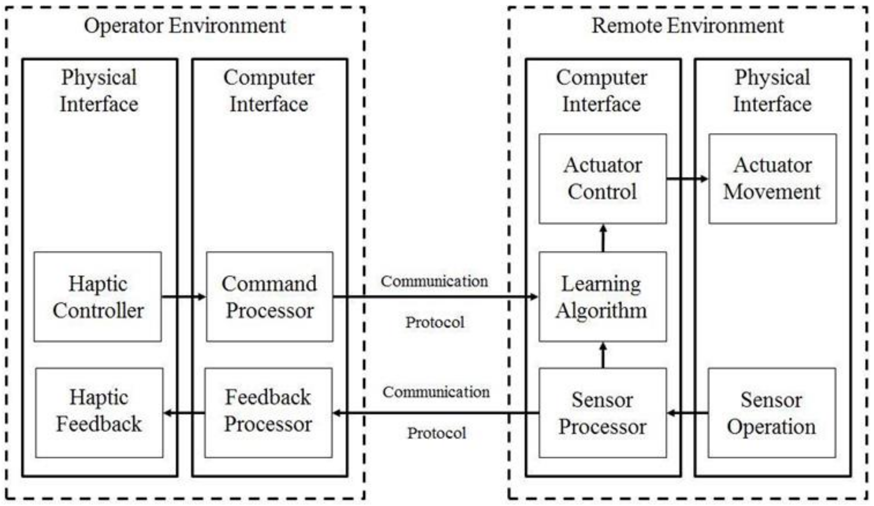


Figure 2-4: Teacher - Learner System (Swanson, 2013)

2.1.5 Autonomous System

In fully autonomous systems, the remote environment can complete tasks on itself without any supervision from the operator. However, sensory feedback is still available to the operator.

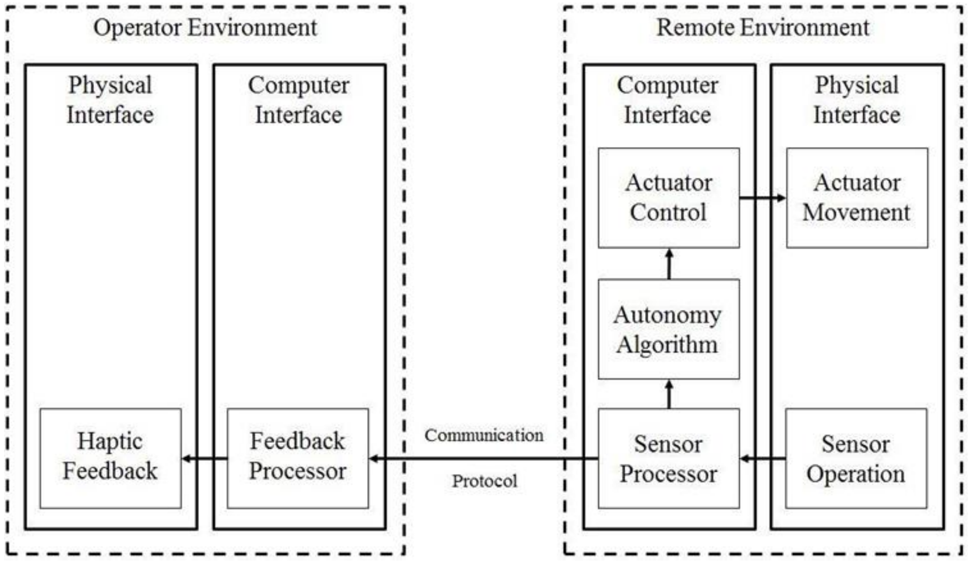


Figure 2-5: Fully Autonomous System (Swanson, 2013)

2.2 Current Applications of Teleoperation

Teleoperated machines have been in existence for a long time. Raymond C Gortez filed a patent for a master-slave device that handled radioactive material from a safe distance in the year 1949 (United States of America Patent No. US2632574A, 1949). In this section we will take a brief look at the current applications of teleoperated machines. We will then take a more detailed look at the latest research carried out in teleoperated ground vehicles and teleoperated cars. We will also look at studies that investigated the concept of enhancing telepresence (feeling of being present in the remote environment).

2.2.1 Space Applications

Space is a very appropriate environment to deploy teleoperated machines and vehicles. The presence of a human to operate a machine in space either requires too many resources (Manned mission to moon) or is simply not possible (exploration of sun), hence teleoperated vehicles are apt for this purpose. In a survey published on teleoperation, (Lichiardopol, 2007) has categorized the space applications into following categories:

i) Space Exploration robots: It includes vehicles sent to space to explore planets, moons, universe etc. Space exploration vehicles can be classified as:
   - Landing Robots (e.g. Curiosity (NASA), Spirit (NASA), Lunokhod I (Russia))
   - Exploration Robots (e.g. Voyager (NASA), Parker Solar Probe (NASA))
   - Deep Space Observers (e.g. Hubble Telescope (NASA))

ii) Satellites: These are used for communication, weather forecast, GPS and much more

iii) Outer-Space Robot Arms: There are robotic arms present on International Space Station to perform tasks outside the space station.

2.2.2 Military/Defense Applications:

Military for a long time has used teleoperated vehicles to minimize human exposure in risky situations. Some of the known applications are Unmanned Air Vehicles (UAV) for reconnaissance and long-range strike missions. One example of such a UAV is US Airforce Predator that can be remotely piloted through radio and satellite links. It can also fly autonomously based off GPS and internal navigation.

Military has also developed Unmanned Ground Vehicles (UGV) for reconnaissance, route-clearing, and land mine detections. Military UGVs are often suppled with vehicle localization equipment (GPS, LIDAR etc.) and advanced sensors such as stereovision cameras to provide best possible feedback in dangerous situations (Lichiardopol, 2007).

2.2.3 Security Applications

Police and law enforcement agencies are increasingly using teleoperated vehicles and machines to deal with situations dangerous for humans. Various police departments around the world have deployed teleoperated robots for bomb disposal, surveillance etc. (Lichiardopol, 2007)

2.2.4 Toxic Environments

Teleoperated robots are often deployed in toxic or radioactive environments that are unsafe for humans. Pioneer Robot developed by Redzone Robotics of Pittsburgh was used in Chernobyl to examine the interior of the destroyed nuclear reactor. (Cui, Tosunoglu, Roberts, Moore, & Repperger, 2003)

2.2.5 Underwater Vehicles

Deep sea operations were one of the first areas where teleoperation techniques were used. They are used in surveying, inspections, oceanography and different manipulation and work tasks, which were traditionally performed by divers. They are usually tethered to a surface ship and controlled using video feedback and joysticks (Lichiardopol, 2007).

2.2.6 Forestry and Mining Industry

Mining and forestry industry have unique hazards associated with them that makes them a dangerous place for humans. Hazards such as falling trees, rough terrain, and caving in mines, have necessitated the development of teleoperated vehicles and machines to be used in such environments. One noticeable example is the “Work-Partner”, a centaur like robot developed by Helsinki University of Technology. It is able perform a variety of forestry tasks and it has a very well-developed human-machine interface

(Suomela, 2001). Based on their applications, (Lichiardopol, 2007) sub-categorizes the teleoperated machines in the forestry and mining industry into following categories:

- Excavating Machines: It includes machines such as excavators, drillers, crushers etc. that are operated by humans remotely.
- Exploratory robots: These allow to remotely monitor and survey mines and ores from a safe location. An example of such a robot is the Groundhog robot developed by Robotics Institute at CMU (Ferguson, et al., 2003).
- Rescue Robots: Allow for search and rescue in case of mine collapses or in other hazardous circumstances. An example of such a robot is the Cave Crawler developed by CMU (Baker, et al., 2004)

2.2.7 Telesurgery

Teleoperation has also been used in medical field allowing surgeons to perform very delicate surgeries. In this scenario, the word "remote" can mean either a large distance or just a difference in scales. The world's first telesurgery was performed in 2001. It was conducted by a team of doctors in NY, USA using the ZEUS robotic system (by Intuitive Surgicals). The patient was in Strasbourg, France. The surgery was a success. Since then, there have been many studies carried out in this field that have mainly been focused on maximum possible latency time, haptic feedback, visual feedback, and 3D reconstruction (Choi, Oskouian, & Tubbs, 2018)

2.2.8 Telepresence

Telepresence in the context of teleoperation application means replacing a human with a robot that is controlled by a human operator from another location. Typical example of this are teleconference robots such as HeadThere Giraffe Telepresence robot and Pebbles Teleconferencing robot. They are also finding uses in nursing/care and education industry.

2.3 Review of Teleoperation related Literature

In their paper, *Investigating Remote Driving over the LTE Network*, (Liu, Kwak, Devarakonda, Bekris, & Iftode, 2016) studied the human behavior in remote driving setup. To conduct this study, they built a scaled remote driving prototype and tested the driver's performance under various network conditions. They concluded that the current LTE network is not fully feasible for remote driving at this time. The main takeaway from their study was that irregular delay in communications is the most detrimental factor in achieving successful remote driving. They found that negative effects of irregular delay can be mitigated by video frame arrangement strategy that regulates delay magnitude to achieve a constant delay that resulted in a smoother display. This resulted in a better driving performance, measured in terms of lap times, and driving mistakes (maintaining lane). They also concluded that driving experience for the drivers based on NASA's Task Load Index with maximum regular delay (348 ms) is not significantly different as compared to their experience with no delay.

In their thesis, *Evaluation of Teleoperation System Performance Over a Cellular Network*, (Sepulveda, 2016) tested the hypothesis "Can a vehicle can be controlled over a 4G LTE network streaming HD video feedback?". After several tests, and through means of statistical analysis, the hypothesis was overwhelmingly rejected and insight into what and why the system presented issues were discussed.

In their paper, *A System Design for Teleoperated Road Vehicles*, (Gnatzig, Chucholowski, & Lienkamp, 2013) modified an Audi Q7 car so that it can be driven remotely over cellular data. They concluded that delay of up to 500 ms should be unproblematic for driving at speeds of up to 30 km/h. Although their testing scenario had no moving obstacles on the track.

In another paper, *Prediction Methods for Teleoperated Road Vehicles*, (Chucholowski, Buchner, Reicheneder, & Lienkamp, 2013) proposed predictive models to mitigate the effects of delay in teleoperated systems. In their systems, they estimated the future position of the car based on previously received data from the car, the car parameters, and the current input by the controller to the car. Based on the output of the predictive model, the images from the car are enriched before being presented to the operator. Their strategy can be summarized using the flow chart in Figure 2-6:

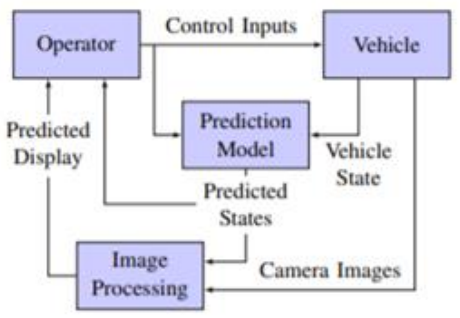


Figure 2-6: In the predictive display system setup, the vehicle states are predicted and camera images are enriched by the predicted vehicle position before they are displayed to the operator (Chucholowski, Buchner, Reicheneder, & Lienkamp, 2013)

In their paper*, Human Performance Issues and User Interface Design for teleoperated robots*, (Chen, Haas, & Barnes, 2007) conducted an extensive study where they measured the effects of various factors on remote operation of robots. They concluded that with reduced Field of View, drivers have more difficulty in judging the speed of the vehicle, time-to-collision, perception of objects, locations of obstacles, and the start of a sharp curve, loss of peripheral vision. Proposed solutions were to use wider FOV or changeable FOV, but they might have an undesirable side effect of perceived speed increase and motion sickness. They also studied the effects of multiple cameras. Use of multiple cameras can lead to attention switching and change blindness. They also concluded that for ground vehicles, even though direct view (egocentric view) is better in terms of driving performance (less mistakes and more accurate driving), users still preferred exocentric view (God's view or aerial view) from an aerial vehicle. Also, integration from different camera sources may be difficult for the operator. Effects of depth perception were also studied. Authors concluded that ability to measure depth is very helpful for the remote operators. Effects of frame rate on user's performance were also investigated. The authors recommend a minimum frame rate of 10 Hz. Detrimental effects of time delays were also studied. In their study, the negative effects of time delays started manifesting after delays exceeded beyond 170 ms. They also arrived at a similar conclusion as Liu et al. that variable delay is more detrimental as compared to regular but longer delays. Authors concluded that for driving like tasks the delay should not exceed beyond 170 ms. Another conclusion from this study was that robot-to-operator delay is more detrimental than operator-to-robot delay. Authors also studied the effects of motion on user performance. Motion degrades accuracy of the drivers and caused severe motion sickness in the subjects. They recommended building display systems that mitigate the effects from vibrations.

In the paper, *The effect of bandwidth on operator control of an unmanned ground vehicle*, French et al. (French, Ghirardelli, & Swoboda, 2003) recommended a minimum frame rate of 8 FPS for unmanned ground vehicles. They further concluded that frame rate above 8 FPS might not greatly improve performance.

In the paper, *Adaptive Automation for Human-Robot Teaming in Future Command and Control Systems*, (Parasuraman, Barnes, & Cosenzo, 2007) proposed an adaptive automation scheme for human robot teaming. The study aimed to present techniques for human robot partnering in systems such as US army's future combat system (FCS). Humans would be involved in supervisory control of uninhabited vehicles with occasional need for manual intervention. They proposed an adaptive automation scheme that is invoked based on one of the following things: critical events; operator performance measurement; operator physiological assessment; operator modelling; and hybrid methods that combine or more of the previous methods. An automation matrix that decides how to implement automation and what tasks to automate. The matrix provides the framework for determining what is automated and the impact of such automations. Soldier tasks define the first level of the matrix. The next level of the matrix addresses the potential automation approaches. The final component of the matrix provides weightings for factors such as task importance, task connectedness, and expected workload. A simple algorithm then combines the weights into a single number that represents the overall priority for development of that automation. From this prioritized list, various automation strategies (i.e., adaptive or adaptable) can be evaluated.

In the paper, *Predictive Safety Based on Track-Before-Detect for Teleoperated Driving Through Communication Time Delay*, (Hosseini & Lienkamp, 2016) argued that human teleoperator will be late to react to an upcoming collision or hazard due to communication delay. Based on this observation, a novel safety system is proposed. Their safety algorithm predicts the future trajectories of dynamic objects in the vehicle surroundings using a stereo vision-based track-before-detect approach and reacts autonomously to the predicted hazards through speed control. After each intervention, the human operator is informed about the autonomous intervention of the vehicle by a Human-Machine-Interface (HMI), having the ability to override this intervention. Therefore, the human still has the final decision power. Results of the test drives show an overall increase of the safety by reduction of Time-To-Collision as well as an improvement of the acceptance of teleoperated driving through the reduction of the overall triggered deceleration during driving in urban areas.

In the paper, *Enhancing Telepresence during the Teleoperation of Road Vehicles using HMD-based Mixed Reality*, (Hosseini & Lienkamp, 2016) used head mounted display to illustrate the 360^o vehicle surroundings to the operator as a mixture of real and virtual environments. The mixed reality images were constructed using data from cameras and LIDAR. An example of such images is in Figures 2-7 and 2-8.

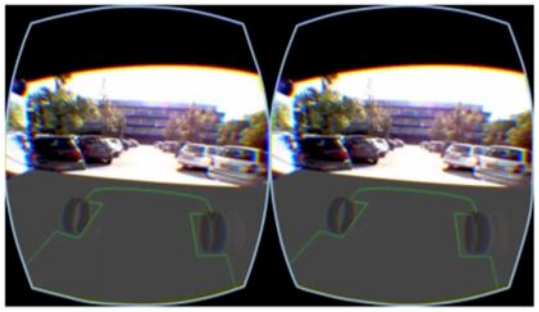


Figure 2-7: Teleoperated driving within a narrow passage using visualization of the remote vehicle in the virtual environment (Hosseini & Lienkamp, 2016)

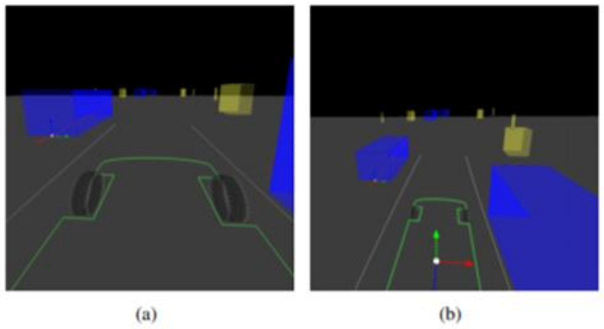


Figure 2-8: The virtual vehicle surroundings reconstructed using the transmitted occupancy grid map, seen within HMD from (a) operator's point of view and (b) perspective view (Hosseini & Lienkamp, 2016)

Two different scenarios were tested to measure the performance of the mixed reality system as compared to the conventional human machine interaction techniques. In first case, to test how precise can the operators drive the vehicle, participants were asked to drive as close as possible to a truck from different directions. According to the results, authors found that this mixed reality approach resulted in: a) more precise control (measured by comparing the distance from the parked truck); b) better situation awareness (measured by comparing the successful collision detection rate between the two scenarios; c) reduced workload (using the NASA Task Load Index). Second test focused on urban driving scenario. Participants were asked to drive on a 1.2 km long track with dynamic and static obstacles mimicking the inner-city conditions three times using the conventional and mixed reality HMI techniques. Metrics such as: a) Directional stability (how frequently lanes were changed) b) Lateral stability (measured by measuring steering wheel reversal rate) c) Workload assessment (using NASA TLX) didn't show a significant difference between using conventional human machine interaction techniques or using the mixed reality system.

In the paper, *Teleoperated Road Vehicles – The "Free Corridor" as a safety strategy approach*, (Tang, Vetter, Finkl, Figel, & Lienkamp, 2014) addressed the problem of what path the vehicle needs to follow in case of communication loss. The safety concept "Free Corridor" presents a solution to the above-mentioned problem. Here, a predicted path and a predicted braking distance are blended onto the video images. In case of a connection loss, an emergency brake is activated, and the vehicle undergoes a previously defined trajectory by the operator. The task of the operator is to keep the “Free Corridor” constantly free of obstacles. It should consequently be possible to bring the vehicle safely to a safe state. In Figure 2-9, the yellow planes show the predicted trajectory of the sides of the car. The red plane shows the distance or point at which the front bumper will come to a stop

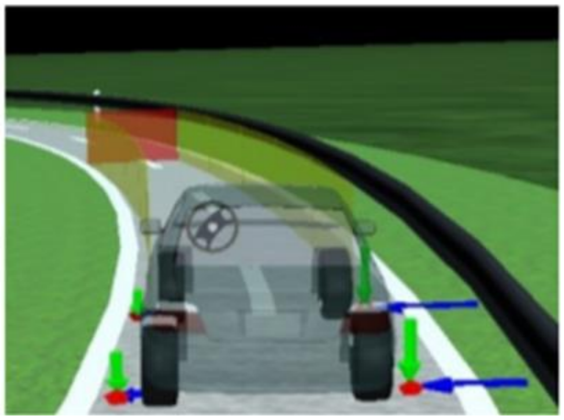


Figure 2-9: "Free Corridor" approach in animation (Tang, Vetter, Finkl, Figel, & Lienkamp, 2014)

2.4 Teleoperated Vehicles in Industry

Nissan has worked on teleoperation system for their vehicles. In their scenario, if autonomous cars are stuck in some real-world situation which they cannot get out of, a human in a call center can take over. So, it is semi-autonomous solution. They call it “Seamless autonomous mobility” (Davies, Nissan's Path to Self-Driving Cars? Humans in Call Centers, 2017).

Phantom Auto is a startup which provides Software Development Kits (SDK) to car makers that would allow the car to be operated and monitored remotely. They argue that current AI technology is unable to drive a car completely autonomously. An autonomous vehicle would face many scenarios which would require human intervention. Therefore, instead of having a human aboard the vehicle, teleoperation would allow a single human to complete monitor or control multiple vehicles from a remote control station (Ohnsman, 2018).

Another startup working on teleoperation is called Designated Driver. Designated driver plugs into 4G LTE network of both AT&T and Verizon using four cellular radios. They stream back feeds from 4 cameras mounted on top of the car. On 4G connection they claim to have dropped the latency below 100 milliseconds. They also plan to adjust the resolution of the video feed based on connection quality. On control station side, a bank of six screen displays video feeds from various cameras and the area of map as well. They also plan to incorporate torque feedback into the system (Davies, Wired, 2019).

Starsky is a start-up that focuses on teleoperated, semi-autonomous trucks. Humans are still in the loop. They can drive the truck remotely from a location close to their home (Korosec, 2019).

In June 2017, at Mobile World Congress Shanghai, China Mobile, SAIC Motor, and Huawei Technologies jointly demoed the world's first 5G-based remote driving technology with a consumer car. In the test, the driver was located over 30 kilometers away from the vehicle. Several high-definition video cameras installed in the vehicle sent multiple real-time HD video feeds to the driver, providing him with a 240-degree view of the vehicle's surroundings over a high-bandwidth 5G network. (Including peripheral vision, without turning their head, an average person has a binocular visual field of around 180-190 degrees). Control signals for the steering wheel, gas pedal, and brakes were also transmitted over the 5G network, which provided the ultra-low latency needed to support instant response to different roadside conditions. From his remote position, the driver was always able to maintain full control over the vehicle (Huawei Demonstrates 5G-based Remote Driving with China Mobile and SAIC Motor, 2017).

2.5 Different Levels of Autonomy for Cars

SAE International’s Standard J3016 define six level of automation for automakers and policy makers use to classify a system’s sophistication (SAE, 2018). The six levels of autonomy are defined as follows:

i) Level 0 (No Automation): No automation added to the car. The human at the wheels controls the steering, acceleration, brakes, and traffic. All the old cars are an example of such kinds of system.

ii) Level 1 (Driver Assistance): Under certain conditions, the car controls either the steering or the vehicle speed, but not both simultaneously. However, the driver performs all other aspects of driving and has the responsibility to take over if the assistance system fails. Adaptive cruise controls in the car is an example of such types of systems.

iii) Level 2 (Partial Automation): The automation system in the car can steer, accelerate, and brake in certain conditions. However, the driver performs all tactical maneuvers such as responding to traffic signals, changing lanes, overtaking, and looking for potential hazards on the road. To indicate that the driver is paying attention, the driver is required to keep their hands on the steering wheel. Examples of such kinds of system are Volvo pilot assist, Mercedes Benz driver assistance systems, Audi Traffic Jam assist.

iv) Level 3 (Conditional Automation): The automation system in the car can most aspects of driving, including the monitoring environment. If a scenario arrives that the automation system cannot deal with it, it asks the driver to take over the control of the vehicle. An example of such a system is Audi Traffic Jam Pilot.

v) Level 4 (High Automation): The car can operate without human input and oversight but only under certain conditions and in specific geographical areas. The driver may not be involved in such a type of car in certain geographical area.

vi) Level 5 (Full Automation): The automated system can drive a car fully by itself in any area or condition that a human driver could negotiate. As of now, there is no car on the market that can achieve Level 5 autonomy. However, many companies including Google, Tesla, Apple, Zoox, Argo AI are working on such driverless cars.

In Section 1.2, we discussed that the objective of our system is not to build a completely autonomous vehicle. Rather, our objective is to provide a solution in which a human can take over the control of the vehicle if required for Level 2 and upwards autonomous vehicles. Most autonomous vehicles companies plan to use teleoperation to some degree. Some use-case scenarios are as follows:

- Waymo launched its robotaxi service in 2018. In their system, if a car is unsure about what to do, it can ping a human for advice (Davies, Wired, 2019).
- California approved driverless testing without backup drivers, but only if the vehicle can be operated remotely (Associated Press, 2018).

2.6 Summary

In this chapter, the basics of teleoperation are discussed. Various teleoperation architectures such as Master-Slave, Supervisory-Subordinate, Partner-Partner, Teacher-Learner and Fully Autonomous. Applications of teleoperation such as space applications, military/defense applications, applications in toxic environment, underwater applications, medical applications are discussed.

Relevant literature is discussed. Studies that focus on remote driving over LTE networks are presented. Moreover, studies focusing on various aspects of teleoperation like safety, telepresence, user interface, effects of communication delay etc. are also discussed.

Lastly, teleoperated vehicles developed by industry are also mentioned.

3 Teleoperated System Description

There is a plethora of options when it comes to implementing a teleoperated system. The main objectives and constraints of our system were as follows:

- The teleoperation system should work over the 4G LTE network.
- The system should be built around ROS. ROS is an open-source, meta-operating system for robots. Using ROS allows us to build, write and run code across multiple computers. Moreover, ROS provides us with services including hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management (Stanford Artificial Intelligence Laboratory et al., 2016)
- Use of low-cost and commercially available equipment. This will reduce the overall cost of implementation for the current system as well as any future teleoperated system.
- Any 4G LTE modems that are used should be commercially available. Moreover, any data plan that is purchased should be commercially available with no preferential treatment while connecting to the network or while being assigned bandwidth.
- Safety systems should be added so that the vehicle can be brought to a safe stop in case of emergency situations such as imminent collision or communication loss.

Several configurations can exist when designing a teleoperated system. One thing common in all configurations is an Operator-Remote environment system connected through a communications link. The general configuration of the system can be seen in Figure 3-1.

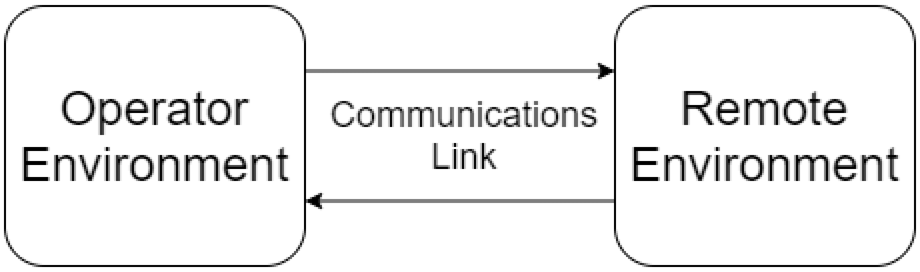


Figure 3-1: General Teleoperation Configuration (Stanford Artificial Intelligence Laboratory et al., 2016)

Based on the objectives and design constraints a teleoperation system was designed using the 4G LT network as the communication medium between the two environments. In chapter 2, various configurations for the teleoperated system were presented. Our system is a basically a Master – Slave system while incorporating some safety systems from the Supervisory – Subordinate system.

We will divide the description of the system into three parts:

- General Teleoperation Scheme: This will describe the general framework and programs used to achieve teleoperation.
- Local Control Station: This section describes the controls situated on the local station, and how that information is confirmed. It will also shed light on the information that the operator is exposed to while driving the vehicle.
- Remote Vehicle Station: It will discuss the controls situated on the remote vehicle and provide some insights into how the system was built.

3.1 General Teleoperation Scheme:

We had two constraints for the teleoperation scheme.

i) The communications between the remote vehicle and the local operator station should be provided by the 4G LTE network. The 4G LTE modem that is used should be commercially available.

ii) Both the systems on the control station and the remote environment should be built around ROS. This constraint will limit our options.

First, let us look at a brief description of ROS network.

3.1.1 Basics of ROS Network

ROS is designed for distributed computing, where several connected components (robots, computer, mobile device etc.) are networked. This can be illustrated with diagram in Figure 3-2:

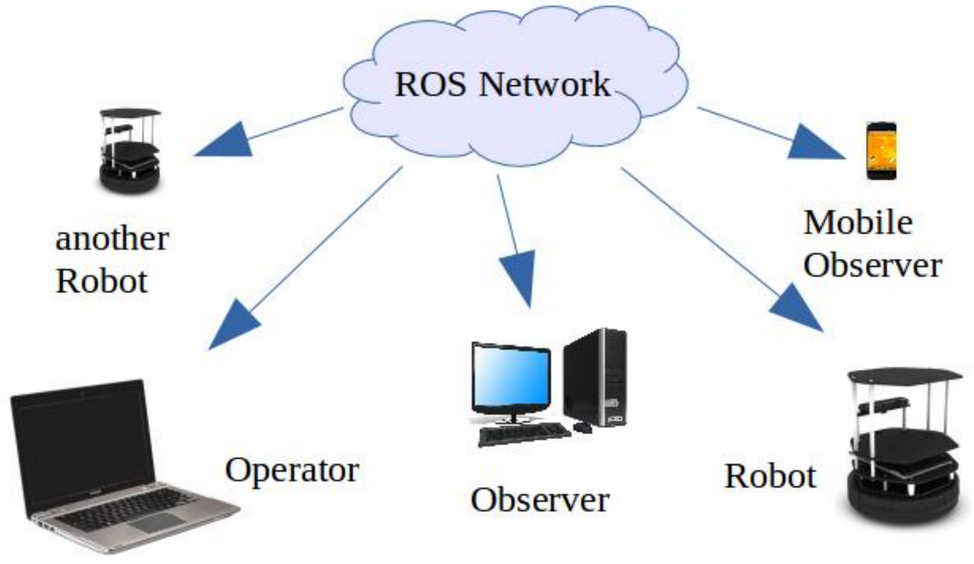


Figure 3-2: Several components together on a ROS network (Stanford Artificial Intelligence Laboratory et al., 2016)

As all these devices share the same network, they form a local network on this modem, and this local network is referred *local ROS network.* All the components have the same public IP address, which is the IP address of the modem. As a security feature, any remote query (from the internet) to any of the component is blocked at the modem. However, all the components within the local network have complete visibility to each other. Figure 3-3 illustrates the local ROS network.

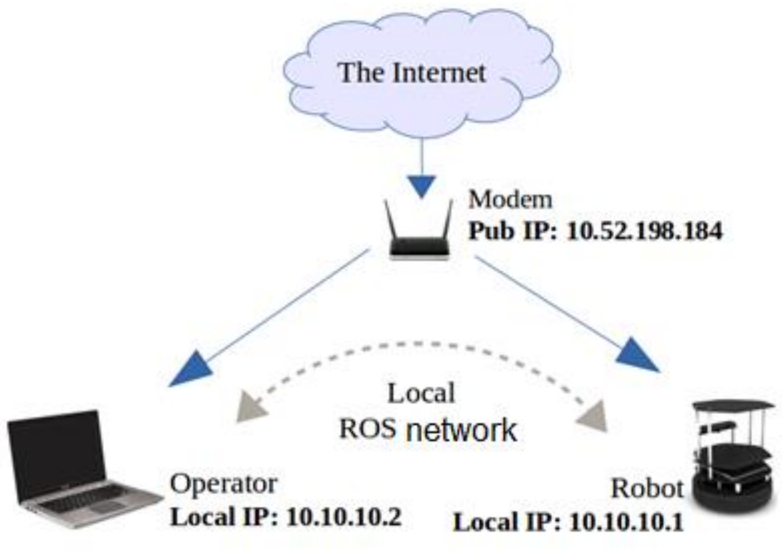


Figure 3-3: Local ROS network (Stanford Artificial Intelligence Laboratory et al., 2016)

Within the local ROS network, every component is identified with a unique IP address as can be seen in Figure 3-3.

3.1.2 Issues in using ROS for teleoperation over the 4G LTE network

As the local ROS network does not let any query from the remote source (internet) to pass through, we cannot establish a connection between two local ROS networks. In our scenario, the remote vehicle and the local operator station can each form a local ROS network. Each local ROS network is connected to the internet separately and need to be connected to each other remotely. Such a network is called remote ROS network. It can be illustrated by the diagram in Figure 3-4.

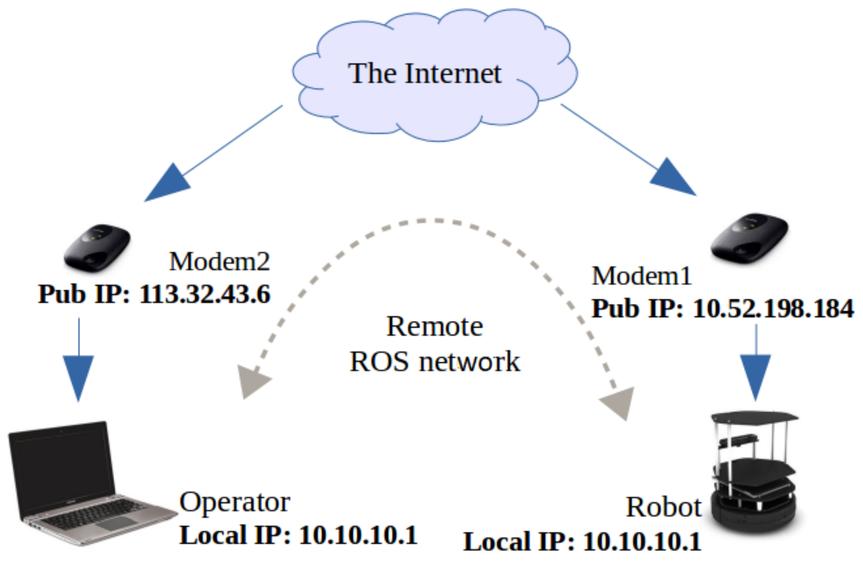


Figure 3-4: Remote ROS network (Stanford Artificial Intelligence Laboratory et al., 2016)

Figure 3-4 also shows the challenges that are faced while setting up a remote ROS network. We cannot use local IP addresses to connect to components in the other local ROS network as the components are on separate local networks. The public IP addresses cannot be used as these IP addresses refer to the modem addresses rather than the components themselves.

3.1.3 Potential Solutions for building a Remote ROS network

To overcome this problem, there are following three options:

1. **Port Forwarding (PF):** PF is a computer networking technique that is usually used to make services on a host residing on a local private network available to hosts on the external network by remapping the destination IP address and port number of the communication to an internal host. ROS documentation prefers this method because if port forwarding is successfully established, all the messages being published by components on separate local ROS networks get passed through to each other without any extra conversion required.

   However, it also has a few downsides, which are as follows:

i) Port forwarding requires beforehand knowledge of all public IP addresses of the networked components. Dynamically changing IP addresses pose problems. In our case, the commercially available modem we used did not have an option of setting up a static IP address.

ii) Port forwarding is a complex multi-layered process that involves configuring settings with the ISP, modem/router, operating system configuration, setup within ROS itself. Such an elaborate setup makes troubleshooting of the problem difficult.

iii) Port forwarding needs to be allowed and supported by the ISP. The modem used did not have port forwarding support.

2. **Cloud Based Solution:** Cloud based solution builds a remote ROS network by transferring the contents of the ROS messages called “rosmsg” through the cloud. The message first needs to be converted to web format and then needs to be converted back to the ROS message on the other end. This means that each message being transferred through the cloud would have to undergo a two-way data conversion. The diagram in Figure 3-5 illustrates this concept.

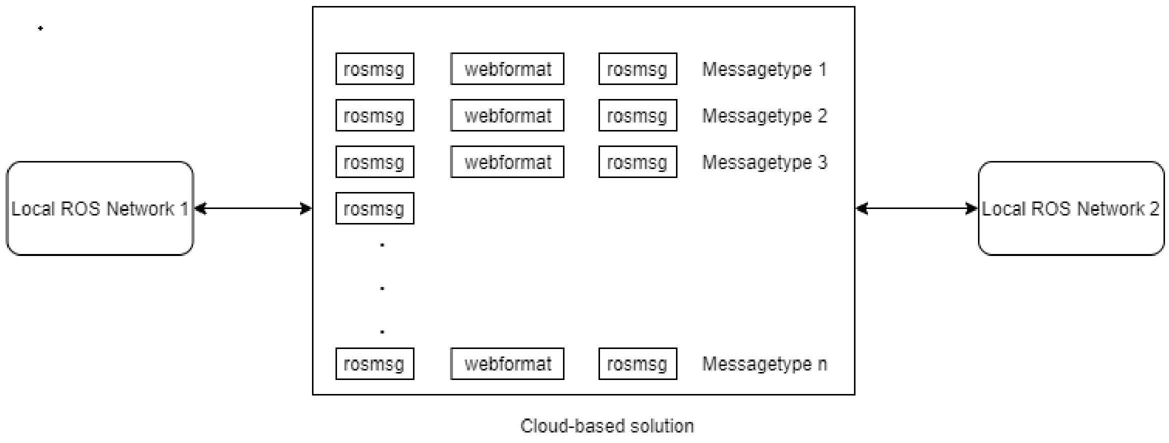


Figure 3-5: Cloud-based solution for Remote ROS network

However, there are few challenges with this solution:

i) Each type of ROS message will require an individual bridge to transfer it through the cloud. For complex application such as ours, it will require a lot of individual rosmsg-webformat-rosmsg bridges to be coded.

ii) Moreover, with a research project like ours, we are constantly adding and removing new sensors to the project. If a cloud-based solution is used, each new sensor will require a new bridge to be built. It is intuitive to say that this will increase the development time of the project therefore it is infeasible.

3. **Virtual Private Network (VPN):** The third option is to create a virtual private network. VPN is another computer networking technique that extends a private network across a public network and enables users to send and receive data across shared or public networks as if the devices were connected to a private network. This concept can roughly be illustrated by the diagram in Figure 3-6.

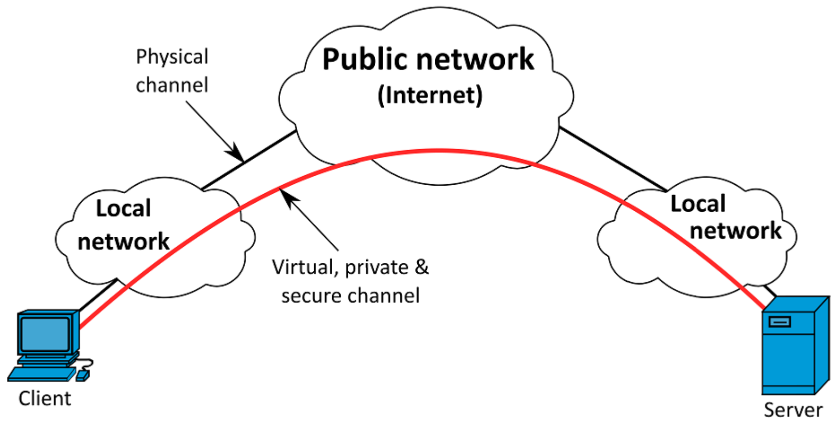


Figure 3-6: VPN connectivity overview

By using VPN, a virtual local network is created on which all the components are connected. In essence, we now have a local ROS network. As a local ROS network has the property of complete visibility between each of the connected components, we can now send and receive ROS messages between any component of the system. The only requirement is installing a VPN software on each computer on the local ROS network. Each computer should be connected to the same VPN.

The one downside to this approach is:

1. As all the data must go through the VPN, there might be an additional latency added to the system.

3.1.4 Chosen Solution for establishing a Remote ROS network

All three options were evaluated. The option of port forwarding was not viable as the 4G LTE modems we have did not have a static IP address provided by the ISP. Moreover, the modem and the cellular service provider did not provide port forwarding support. The cloud-based solution was not chosen as it required to write code to create a bridge whenever a new message type is added to the system. This would have significantly increased the development time of the project. The last option was the most viable option and hence it was selected as the final solution to overcome the problem of stablishing a Remote ROS network.

The VPN software used was Cisco's Anymobile Connect VPN software. Each computer that was part of the system had this VPN software running and connected to the same private network.

3.1.5 General Architecture of the Teleoperation System

As discussed before, the teleoperation system is built around the 4G LTE system and ROS meta operating system. The general architecture of the teleoperated system is depicted in Figure 3-7.

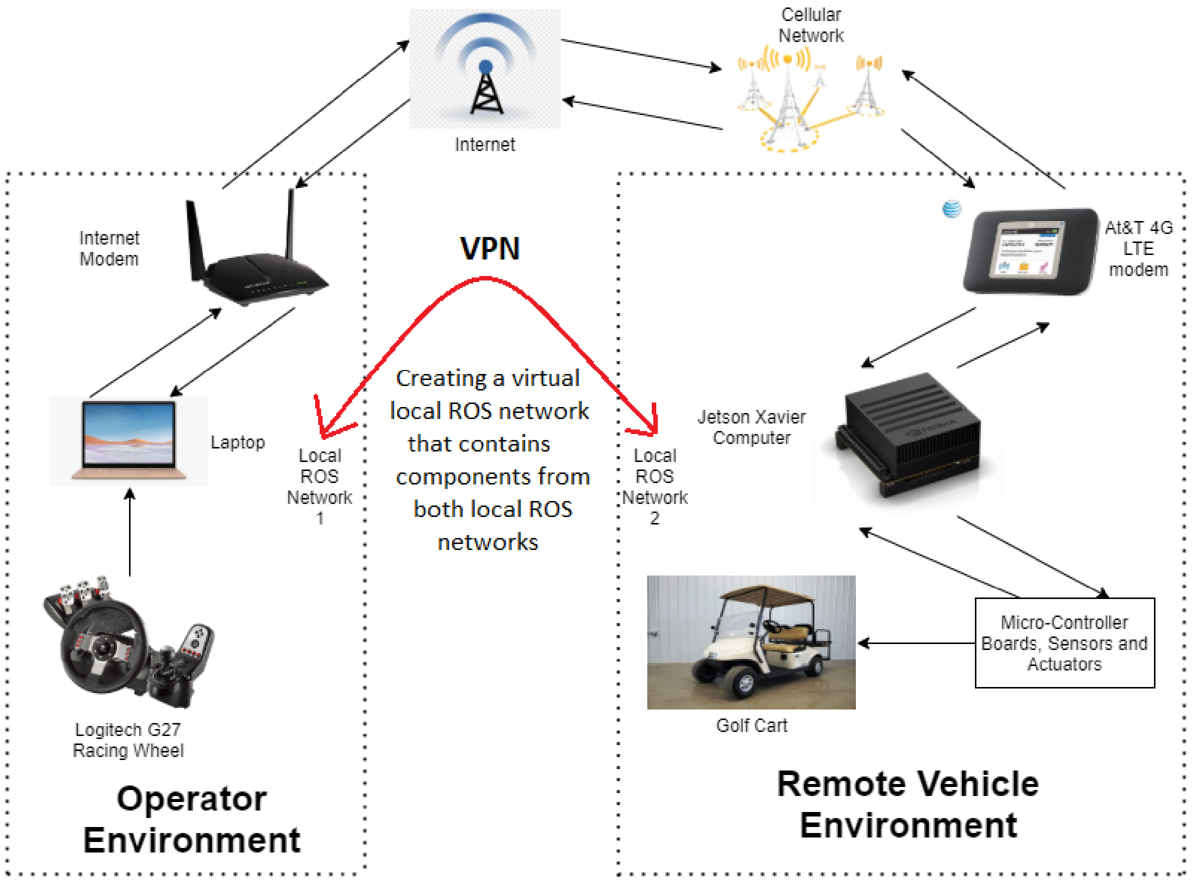

Cellular Network
Internet
VPN
Internet Modem
At&T 4G LTE modem
Creating a virtual local ROS network that contains components from both local ROS networks
Laptop
Local ROS Network 1
Local ROS Network 2
Jetson Xavier Computer
Micro-Controller Boards, Sensors and Actuators
Golf Cart
Logitech G27 Racing Wheel
Operator Environment
Remote Vehicle Environment

Figure 3-7: General system architecture for the Teleoperation System

The operator environment represents the main interface of the system with the operator and is where all the control signals are generated. Whereas the remote vehicle environment is the actual vehicle to be controlled where the feedback signals are generated and sent back to the operator station. The remote vehicle environment is connected to the internet through 4G LTE network. The local operator station is connected to the internet through the fiber optic. The two environments make up two local ROS networks which are connected to create a single virtual ROS network. In the subsequent sections, further details would be provided for each subsystem.

3.2 Control Station/ Local Operator Station Environment

The operator station or the control station is the environment that generates all the signals to control the remote vehicle. The setup for the control station resembles a typical automobile cockpit including a steering wheel, gas and brake pedals, gears for forward and reverse motion, and screens that provide feedback from the remote vehicle.

We will now illustrate the incoming and outgoing information to the control station using the diagram in Figure 3-8.

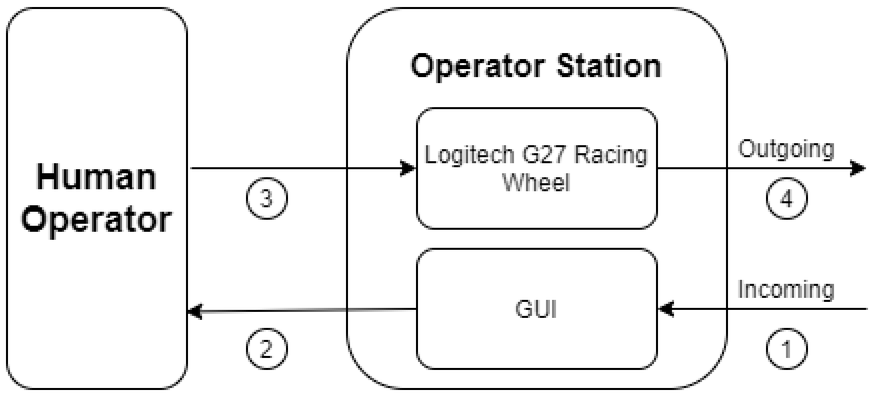


Figure 3-8: Local Operator Station Nodes and Information Flow

The local operator station captures information from the human operator, convert it into control signals that are sent to the remote vehicle. It also receives video feed and other auxiliary information from the remote vehicle and displays it to the human operator.

3.2.1 Operator Station Information Flow

- Incoming (Line 1): This is composed of the video feed from the remote vehicle; LIDAR data from the remote vehicle; auxiliary information from the sensors such as vehicle speed, steering wheel angle etc.

- Outgoing (Line 4): This is composed of the control signals captured from the human operator to the remote vehicle.

Now, let us look at the two main parts that make up the control station i.e. the Logitech G27 Racing Wheel and the GUI.

3.2.2 Logitech 27 Racing Wheel

“Logitech G27 Racing Wheel” was used as a control station. The gaming wheel is shown in Figure 3-9.

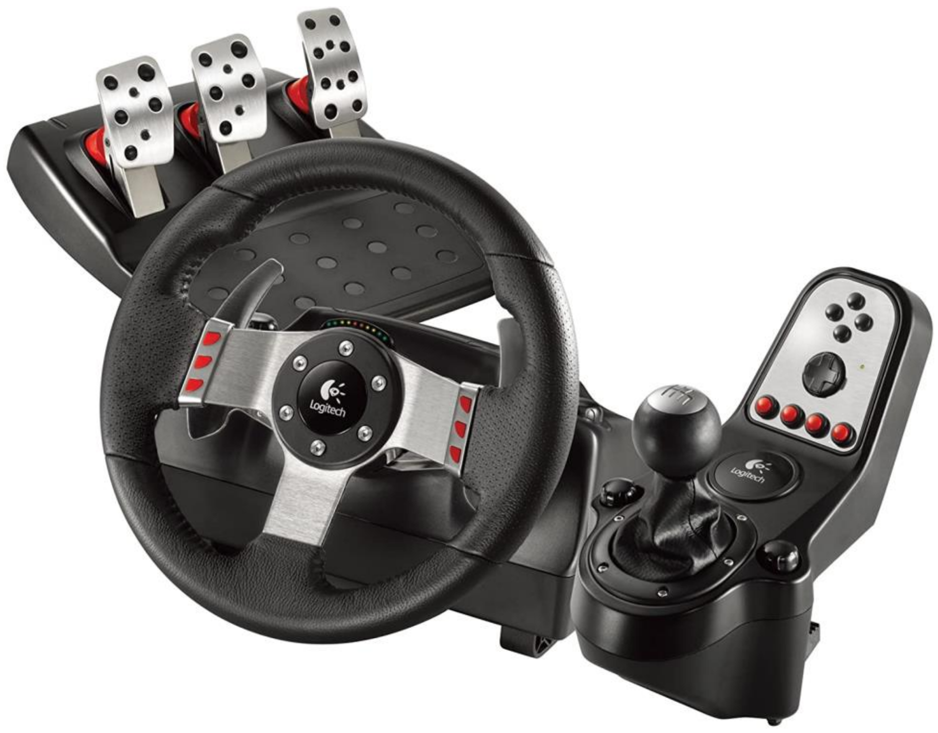


Figure 3-9: Logitech G27 Racing Wheel

A software tool has been provided by Logitech to get the data from the wheel. But, since the tool is only available in Windows, we needed to write our own code to extract the data in a Linux environment. The use of Linux operating system was necessary as our system is built around ROS, which is compatible with Ubuntu. Therefore, python scripts were written to extract data from the Logitech wheel and send it over the ROS network.

A script called "client.py" using Python 3.6 was used to read the data from the Logitech wheel. As ROS is only compatible with Python 2.7, another script called "server.py" was written in python 2.7 that communicated with the first script and published the data received from the racing wheel over the ROS network. The flowchart in Figure 3-10 explains the data flow.

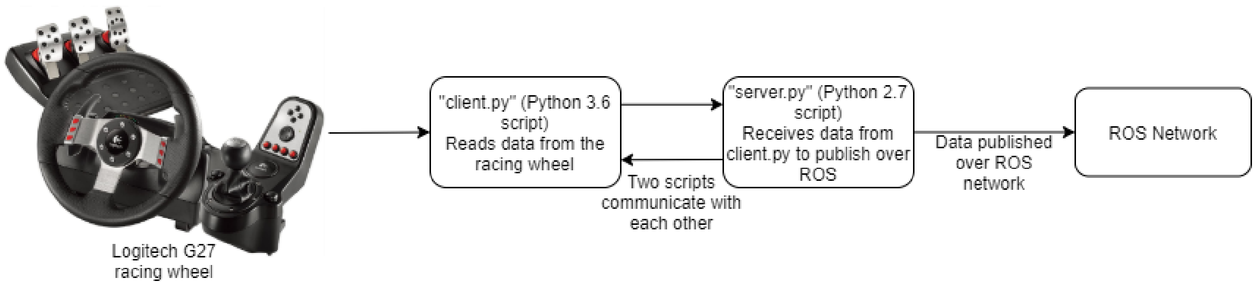

Logitech G27
racing wheel
"client.py" (Python 3.6 script)
Reads data from the racing wheel
"server.py" (Python 2.7 script)
Receives data from client.py to publish over ROS
Two scripts communicate with each other
Data published over ROS network
ROS Network

Figure 3-10: Data extraction from the Logitech G27 Racing Wheel

Table 3-1 shows the data we read from the racing wheel that is used to control the remote vehicle.

Table 3-1: Control Signals read from the racing wheel and sent to the remote vehicle

Control Signals	**Range of Values**	**Corresponding Hardware on the G27 Racing wheel**
Gas	0 (Not pressed) – 1 (Fully Pressed)	Right Foot Pedal
Brake	0 (Not pressed) – 1 (Fully Pressed)	Left Foot Pedal
Steering Wheel Position	0(Extreme Left) – 0.5 (Straight) – 1(Extreme Right)	Steering wheel
Gear	Forward (F), Neutral (N), Reverse (R)	1, 2, 3, 4, 5, 6 correspond to F
		Neutral correspond to N
		Reverse correspond to R

Camera Angle Position	75° (Extreme Left) – 135° (Straight) – 195° (Extreme Right)	Left and Right Paddle; Dpad

From table 4, we can see that we can control all the controls on the golf cart from the control station. The Logitech G27 racing wheel is labelled in the diagram in Figure 3-11 to show which part corresponds to which control signal.

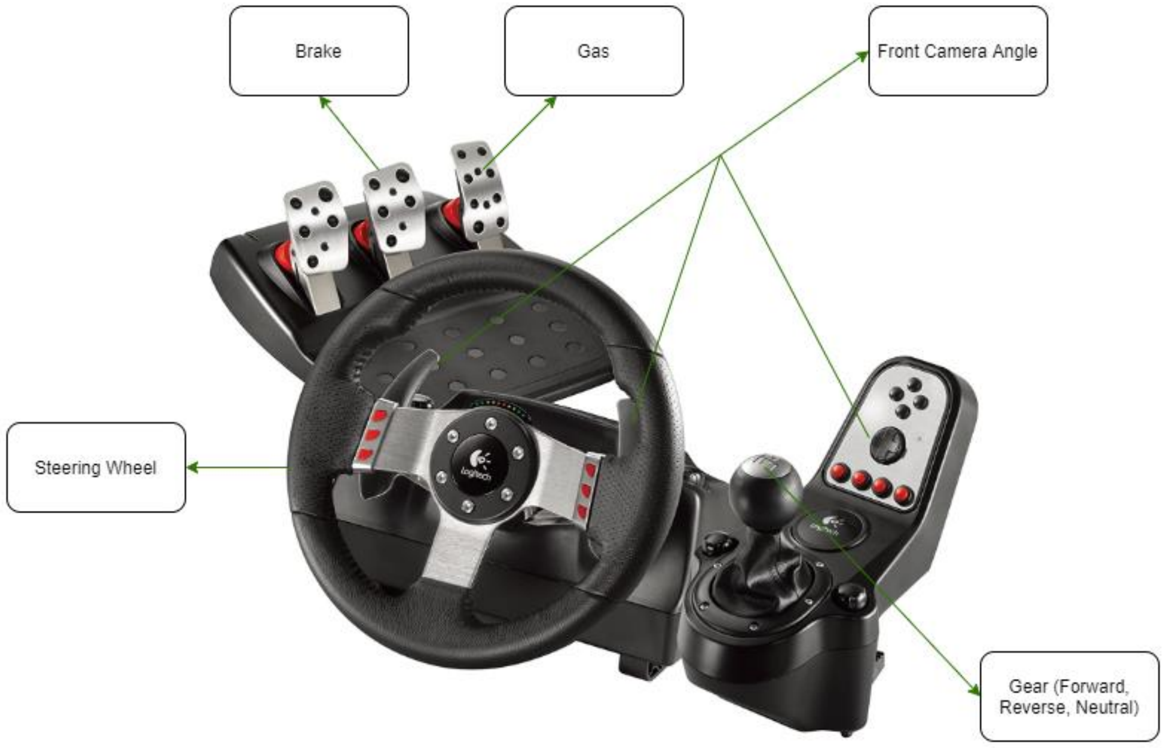


Figure 3-11: Correspondence between different parts of Logitech G7 Racing Wheel and various control signals

Figure 3-12 shows an image of the control station set up in our lab.

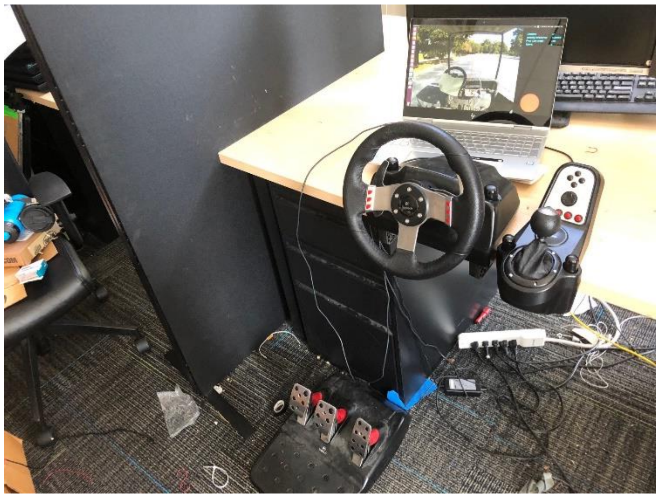


Figure 3-12: Control station set up in lab

3.2.3 Graphical User Interface

All the information from the remote vehicle is displayed to the operator using a graphical user interface.

The GUI contains the following information:

1. Video feed in one of two formats:
   - i) Semantically segmented video feed
   - ii) Video feed with object tracking
2. LIDAR data from the remote vehicle
3. Vehicle speed information
4. Steering wheel angle

A snapshot of the Graphical User Interface is as follows:

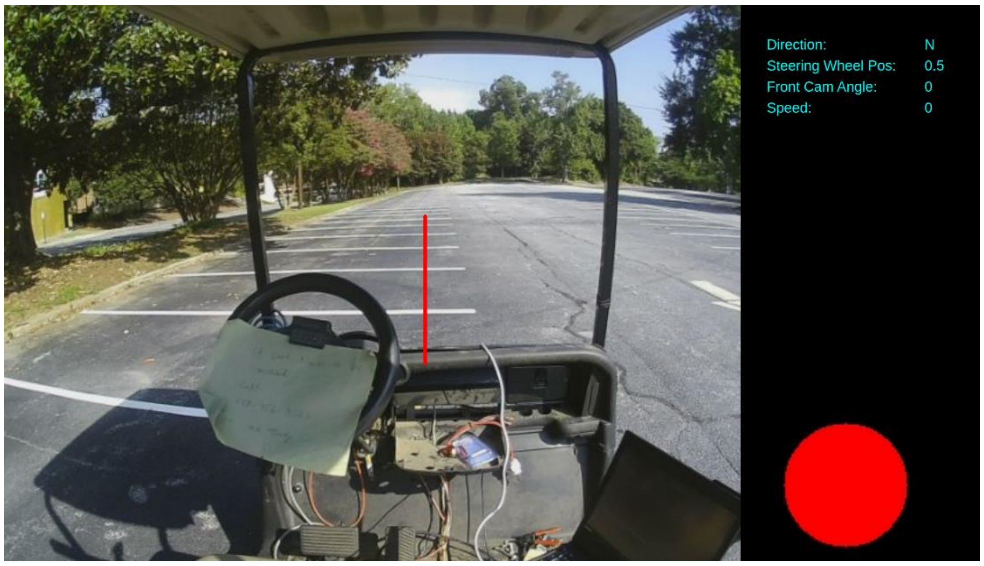


Figure 3-13: Snapshot of GUI at the operator station

More details of each of the subsystems will be provided in the next chapter.

3.3 Remote Vehicle Environment

To create a better understanding of the remote vehicle environment, let us look at a visual representation of the information flow in Figure 3-14.

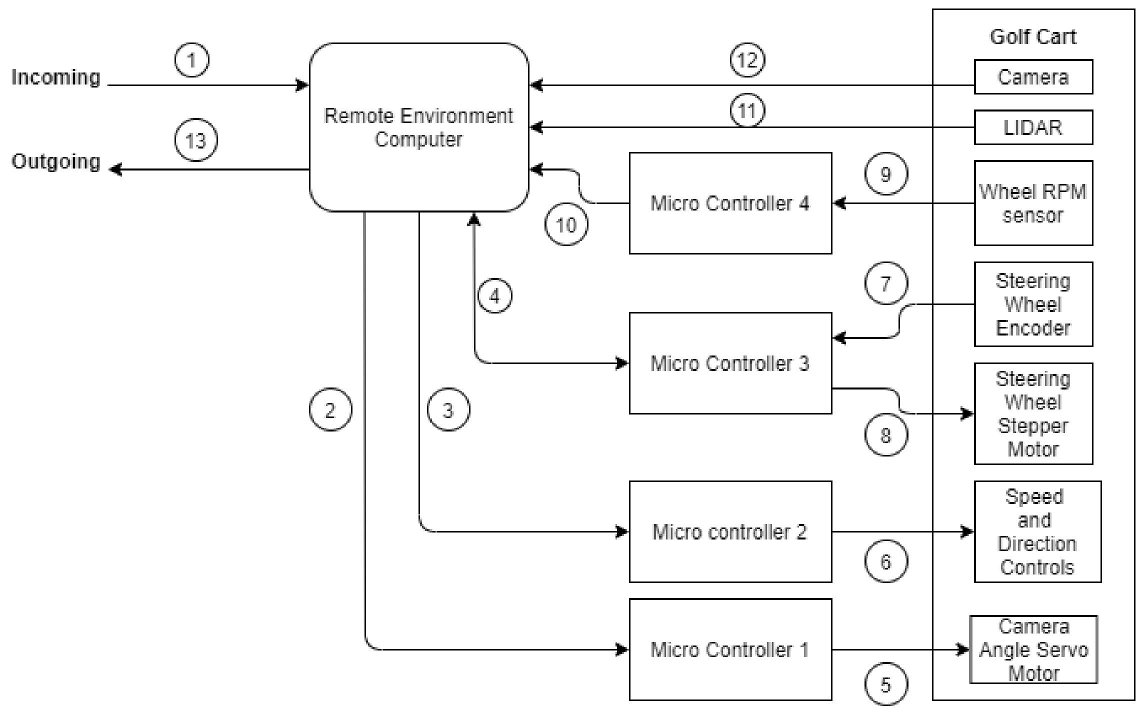


Figure 3-14: Remote Environment Information Flow

3.3.1 Remote Environment Information Flow

We can briefly describe the information that is presented in each signal line.

- **Incoming Signals (Line 1):** Here, the incoming information is composed of all the information that the operator sends from the control station to control the vehicle. All the control signals i.e. gas pedal signal; brake pedal signal; steering wheel angle; direction control; front facing camera angle; that are gathered from the G27 Racing Wheel are contained in this message.
- **Front Facing Camera Angle, Lines 2 and 5:** The remote environment computer receives the angle that the front facing camera should be at from the local operator station. The micro-controller that controls the servo camera on which the front facing camera is mounted, subscribes to this information, and moves the servo motor to that angle.

- **Speed and Direction Controls, Lines 3 and 6:** The remote environment computer receives the throttle signal, brake signal and the direction signal (Forward, Neutral, Reverse) from the local operator station. The micro-controller subscribes to this information, and then sends the corresponding signals to the built-in golf cart controller.
- **Steering Wheel Controls, Lines 4, 8 and 7:** The remote environment computer receives the steering wheel angle from the control station. The micro-controller that controls the steering wheel position subscribes to this information. It compares this angle with the current angle of the steering wheel that it receives from the encoder mounted on the steering column (line 7). It then performs a simple feedback control to minimize the error between the desired angle and the current angle.
- **Wheel RPM sensor, Lines 9 and 10:** The wheel of the golf cart has a hall effect sensor mounted on it. The micro-controller connected to that sensor measures the RPM of the wheel and calculates the linear velocity of the vehicle. The micro-controller then publishes this linear velocity information on the ROS network. The details of the workings of the sensor will be discussed in the net chapter.
- **LIDAR sensor, Line 11:** The LIDAR is connected to the remote environment computer. The information from the laser scan is processed by the remote environment computer and is published on the ROS network. The details of this processing will be discussed in the next chapter.
- **Camera/ Video Feed, Line 12:** The camera captures the video feed and sends it to the remote environment computer. The computer processes the video feed and publishes it on the ROS network. The details of the processing will be discussed in the next chapter.
- **Outgoing Signals, Line 13:** The outgoing information is composed of all the feedbacks captured from the vehicle that the operator station that the operator requires to reconstruct the remote

environment and take the appropriate actions. This information includes: Processed video feed; processed LIDAR data; vehicle speed and vehicle's steering wheel angle.

Now, let us briefly discuss the various control systems that were mounted on the golf cart that enabled us to control it from the local operator station.

3.3.2 Steering Wheel Control

To control the steering wheel on the golf cart a stepper motor and an encoder was mounted on the steering wheel. The encoder provided the feedback necessary to complete the closed loop control. The desired position of the steering wheel is provided by the Logitech G27 racing wheel, the current position is measured by the encoder mounted on the golf cart, and the stepper motor provides an actuation mechanism.

The simple feedback control implemented can be illustrated using the diagram in Figure 3-15.

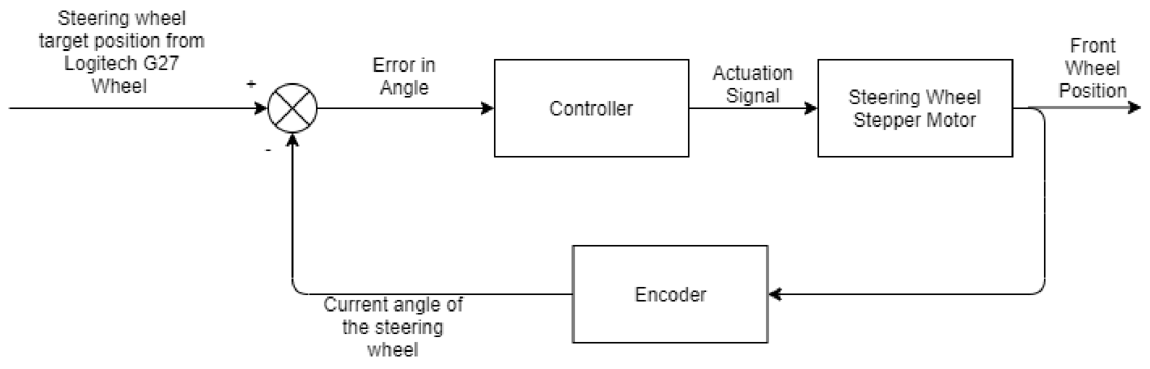

Steering wheel target position from Logitech G27 Wheel
+
-
Error in Angle
Controller
Actuation Signal
Steering Wheel Stepper Motor
Front Wheel Position
Encoder
Current angle of the steering wheel

Figure 3-15: Feedback Control for the steering wheel

3.3.3 Gas Pedal Control

For gas pedal control a simple mapping was performed between the pedal on the control station and the gas pedal on the remote golf cart. On the control station, the value for the gas pedal goes from 0 (not pressed) to 1 (fully depressed). For the golf cart, by pressing the gas pedal we can change the voltage from

0.5 Volts (not pressed) to 5 Volts (fully pressed). A simple linear mapping was performed between the two ranges of values as illustrated in Figure 3-16.

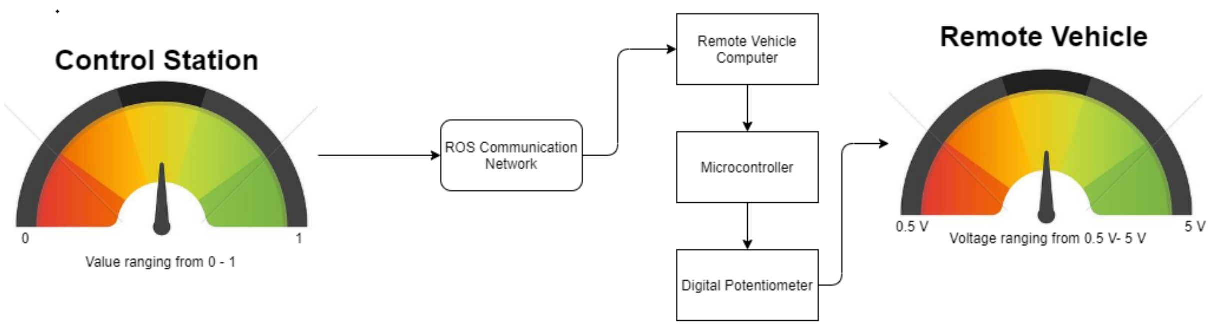


Figure 3-16: Gas Pedal mapping from Control Station to the Remote Vehicle

This linear mapping was achieved using a digital potentiometer. The digital potentiometer was supplied with 5V. The output voltage from the potentiometer depends on the resistance value that is set. By changing the resistance values through a micro-controller, the output voltage can be set within the 0.5V – 5V range.

3.3.4 Brake Pedal Control

Like the gas pedal control, a simple linear mapping was performed for the brake pedal. The value for the brake pedal on the control station goes from 0 (not pressed) to 1 (fully pressed). For the golf cart, by pressing the brake pedal, we can change the voltage from 0.5 Volts (not pressed) to 5 Volts (fully pressed). A simple linear mapping with the help of a digital potentiometer was performed between the two value ranges as illustrated in Figure 3-17.

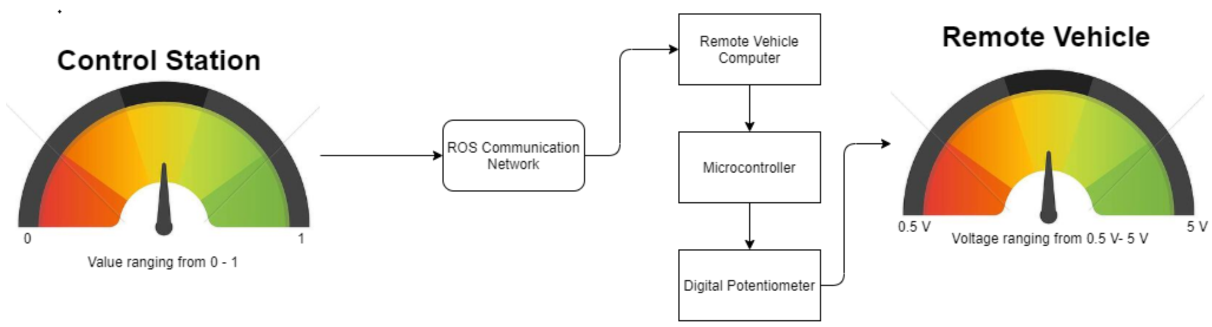


Figure 3-17: Brake Pedal mapping from Control Station to the Remote Vehicle

3.3.5 Ignition and Gear Control

The ignition and gear control were similarly mapped. Any forward gear (1st, 2nd, 3rd, 4th, 5th, 6th) on the racing wheel was mapped to the golf cart's forward gear, the neutral gear on the racing wheel was mapped to golf cart's neutral gear, and the racing wheel's reverse gear was mapped to golf cart's reverse gear.

3.3.6 Camera Angle Control:

The front facing camera on the golf cart was mounted on a servo motor that gives us the ability to pan it. The camera can be panned from the control station using the paddles on the steering wheel. The left paddle starts panning the camera towards the left, and the right paddle starts panning towards the right. Releasing the paddle will cause the camera to come back towards the center. The camera pans up to 60^o on each side. We can also control the camera using the Dpad on the control station. The difference between the Dpad and the paddle control is that Dpad allows us to maintain the camera angle. Releasing the Dpad will not bring the camera back to the center. This will allow the operator to look in a direction for as long as they require. The Dpad pans the camera in 5^o increments.

3.4 Summary

Recapping this chapter, the general teleoperation scheme for the system is presented. A comparison between various ROS based remote teleoperation schemes is discussed. The VPN based system is chosen as the final solution. The general architecture for the system is discussed. Information flow and details about the local operator station are presented. Likewise, the information flow and the details about the remote vehicle environment are presented.

4 Auxiliary Systems

In the previous chapter we discussed that the remote environment computer processes video feed, LIDAR data and measures vehicle’s linear speed. The details of these systems will be discussed in this chapter.

The main feedback that the operator gets is the video feed. Let us discuss the details of how the video feed is processed by the remote environment computer before sending it to the local control station.

4.1 Video Feed Processing

The video feed from the camera is processed in two ways. The user is either shown a pixel-wise segmented video feed or the video feed in which the objects are being tracked. In this section the details of both these methods are provided.

4.1.1 Pixel-Wise Image Segmentation or Semantic Segmentation

Let us look at a brief explanation of what does the term semantic segmentation refers to.

4.1.1.1 Brief Introduction of Semantic Segmentation

Semantic Segmentation is based on image recognition, except the classification occurs at pixel level as compared to classifying entire images as with image recognition. The goal of semantic segmentation is to label each pixel of the image with a corresponding class it represents. As each pixel is being classified, this task is referred to as **dense prediction.** An example of segmented image can be seen in Figure 4-1:

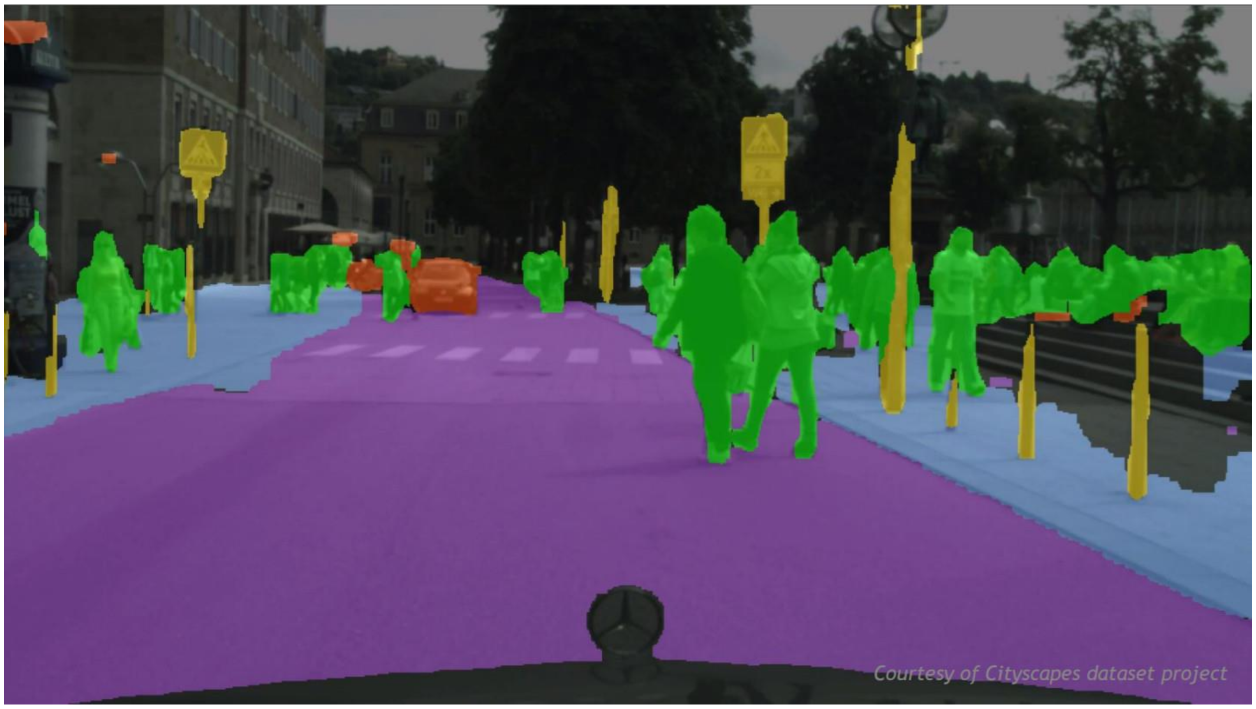
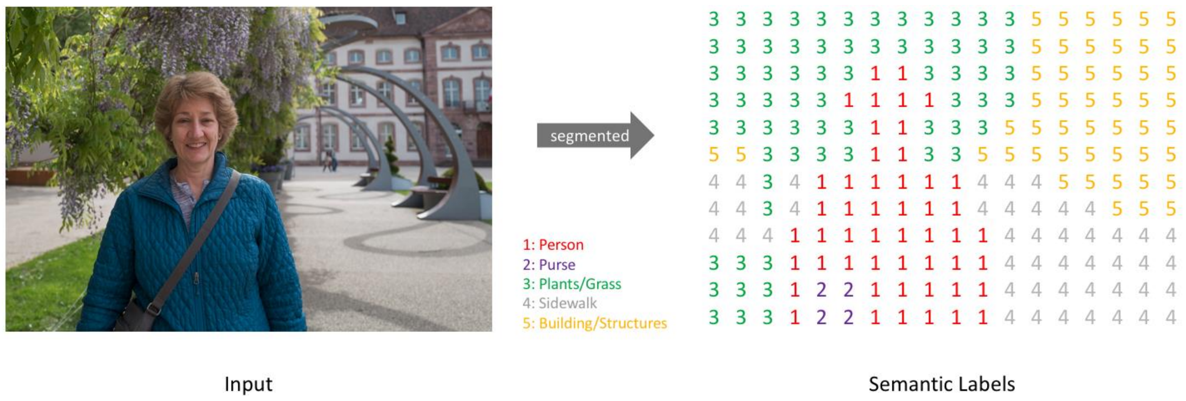


Figure 4-1: An example of Segmented Image

The goal of semantic segmentation is to take as input either a colored RGB image (height x width x 3) or a greyscale image (height x width x 1) and output a segmentation map where each pixel contains a class label represented as an image (height x width x 1). This concept can be illustrated with the image in Figure 4-2:



3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

Figure 4-2: A low resolution segmentation prediction map that illustrates the pixel-wise segmentation concept (Jordan)

Recently, semantic segmentation has been increasingly used in self-driving cars as semantic segmentation not only tells what objects are in the image, but also where, in those images, the objects are located on a pixel level. Hence, using semantic segmentation the systems are better able to make the sense of their surroundings.

4.1.1.2 Brief Explanation of Semantic Segmentation used in our System

The semantic segmentation used in our system is based on the paper "Fully Convolutional Networks" by (Long, Shelhamer, & Darrell, 2015). Fully Convolutional Networks (FCNs) owe their name to their architecture, which is built only from locally connected layers, such as convolution, pooling, and up sampling. Note that no dense layer is used in this kind of architecture. This reduces the number of parameters and computation time. Also, the network can work regardless of the original image size, without requiring any fixed number of units at any stage, given that all connections are local. Usually models that have fully connected layers work with specific size of input images as the size of fully connected layer is fixed. But, in case of Fully Convolutional Networks (FCN), there is no fully connected layer. Hence, they can work with any input size. To obtain a segmentation map (output), segmentation networks usually have 2 parts:

- Downsampling path: capture semantic/contextual information
- Upsampling path: recover spatial information

A general architecture of FCN can be understood from the image in Figure 4-3.

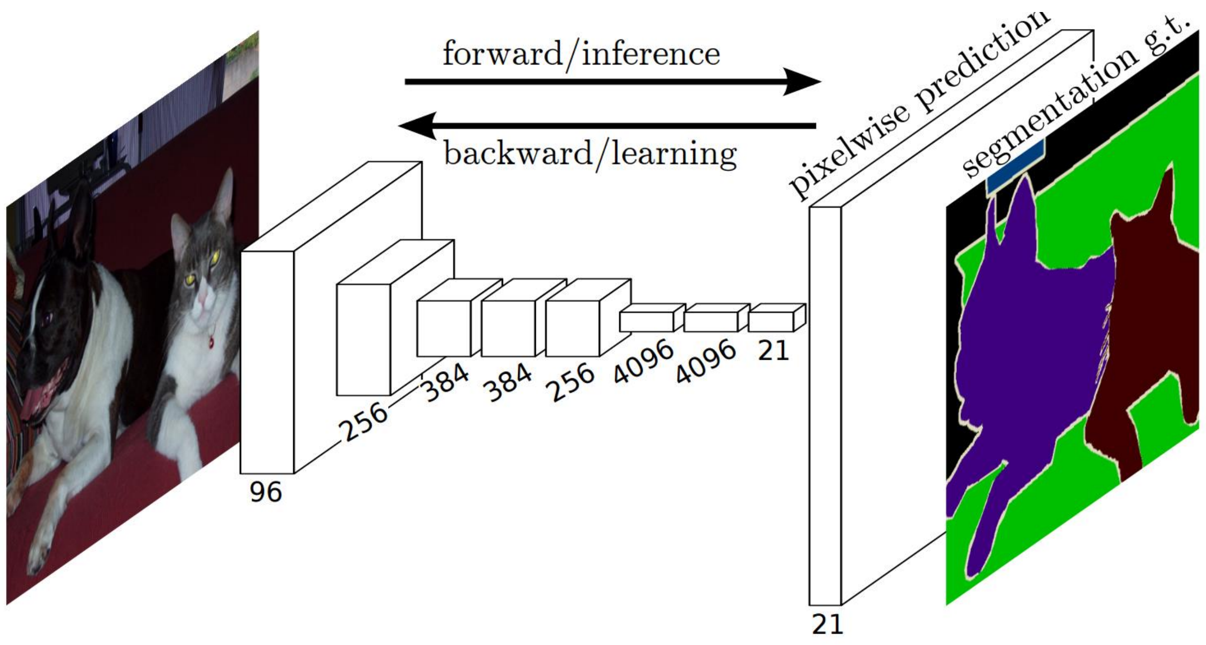


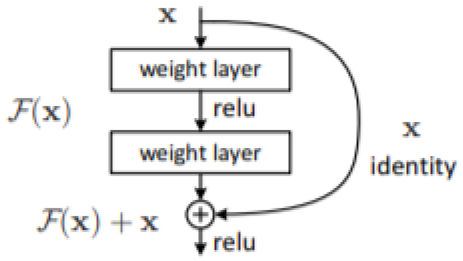
Figure 4-3: Segmentation Network Architecture (Long, Shelhamer, & Darrell, 2015)

As you can see in the image in Figure 4-3, the fully connected layer is replaced by a convolutional layer that predicts class for each pixel.

In the model we used, Resnet18 network was used as a backbone of the FCN model. The downsampling (what classes exist in the image) was achieved using the Resnet18 neural network. Let us look a brief look at the Resnet18 neural network model.

4.1.1.3 ResNet-18

This neural network was proposed by (He, Zhang, Ren, & Sun, 2016). ResNet-18 model is an 18-layer convolutional neural network. ResNets were proposed to solve the issue if performance saturation and/or degradation when training deeper layers, which is a common problem among other CNN architectures. Residual networks avoid this issue by implementing an identity shortcut connection, which skips one or more layers and learns the residual mapping of the layer rather than the original mapping. The building block of the residual learning network is illustrated in Figure 4-4.



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4-4: Residual Learning: A building block (He, Zhang, Ren, & Sun, 2016)

The complete architecture of the ResNet18 CNN is illustrated in Figure 4-5.

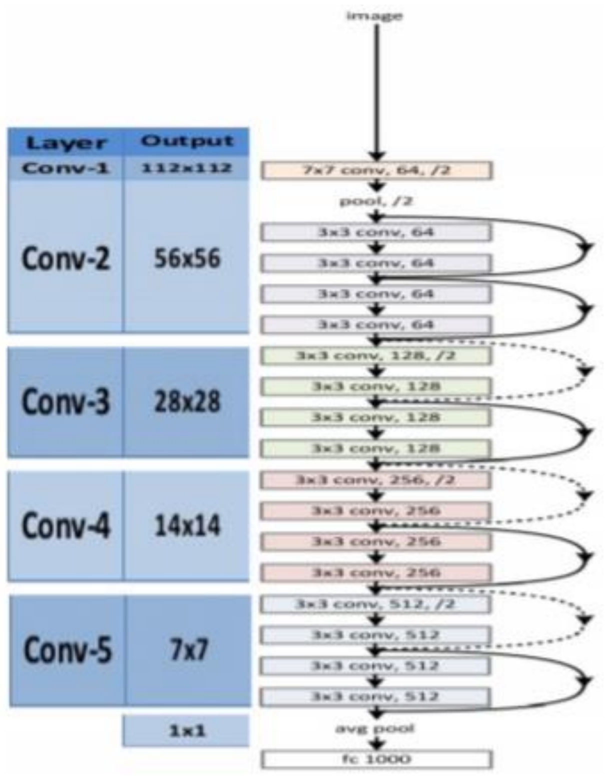


Figure 4-5: Architecture of ResNet18 (He, Zhang, Ren, & Sun, 2016)

In semantic segmentation, the last fully connected (FC) layer is replaced with a 1x1 convolutional layer.

We will take a brief look at this process in the next section.

4.1.1.4 Fully Convolutional Network (FCN)

In the previous section, we saw the architecture of ResNet-18 that is used to extract image features (downsampling path). The total number of channels are then transformed into the number of categories using a 1x1 convolutional layer. Finally, a transposed convolutional layer is used to transform the height and width of the feature map to the size of the input image. The final output has the same height and width as the input image and has a one-to-one correspondence in spatial positions. The final output channel contains the category prediction of the pixel corresponding to the spatial position. Architecture of Fully Convolutional Network is illustrated in Figure 4-6.

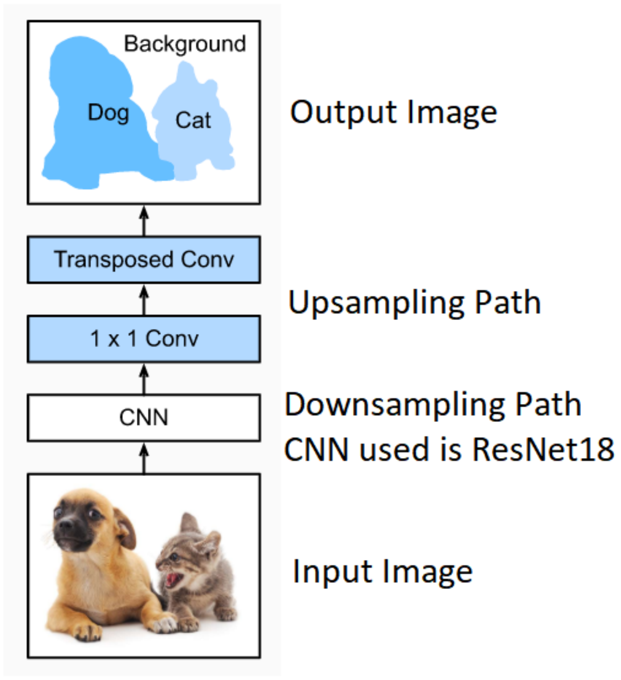
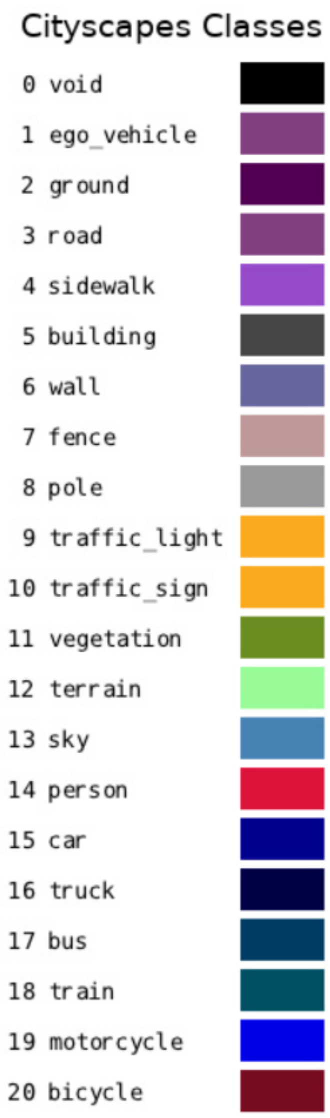


Figure 4-6: Basic Architecture of FCN

4.1.1.5 Classes Detected by the Semantic Segmentation Network

The semantic segmentation network that we use can detect 20 different classes. The classes along with their color codes are shown in figure 4-7.



Cityscapes Classes

0 void	⬛
1 ego_vehicle	🟪
2 ground	🟪
3 road	🟪
4 sidewalk	🟪
5 building	⬛
6 wall	🟦
7 fence	🟫
8 pole	⬜
9 traffic_light	🟧
10 traffic_sign	🟧
11 vegetation	🟩
12 terrain	🟩
13 sky	🟦
14 person	🟥
15 car	🟦
16 truck	🟦
17 bus	🟦
18 train	🟦
19 motorcycle	🟦
20 bicycle	🟥

Figure 4-7: Classes detected by semantic segmentation network along with the color codes

The second method by which the video was tracked is called object tracking. The details of this method will be discussed next.

4.1.2 Object Detection and Tracking

In this method, objects are detected in the video feed. The objects that are detected are continuously tracked. We will now discuss the details of the detection and tracking algorithms that were used.

4.1.2.1 Object Detection

Object detection is the term used to describe computer vision related tasks that involve identifying objects in photographs. It combines image classification and object localization. *Image classification* is the technique of predicting the class of one object in an image. *Object localization* refers to the process of identifying the location of one or more objects in an image and drawing a bounding box around them. *Object Detection* combines these two tasks, localizing and classifying one or more objects in an image. Figure 4-8 showcases the above process.

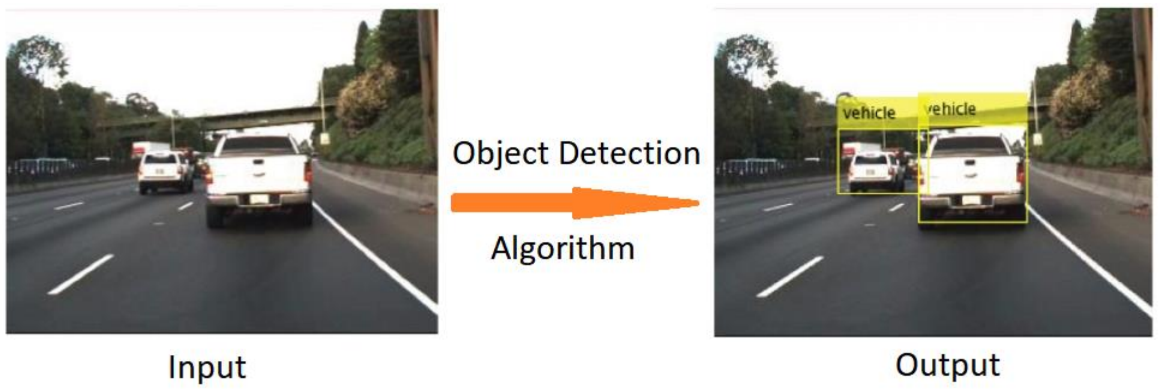


Figure 4-8: Using Object detection to locate and detect items in an Image

Bounding box can be represented by the pixel coordinates of the x and y lines that constitute its boundaries. An example of bounding box coordinates is shown in Figure 4-9.

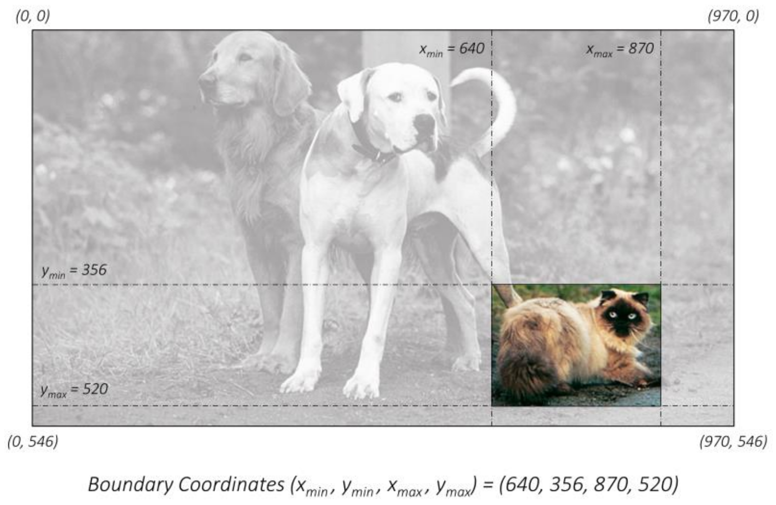


Figure 4-9: Representation of bounding box using Boundary Coordinates

The boundary coordinates of a box are simply (x_min, y_min, x_max, y_max).

In our system, we use the SSD-Mobilenet-v2 model to detect objects in our image. This model can detect 91 different classes. SSD stands for "Single-Shot-Detector". Now, we will look at a brief description of Single-Shot-Detectors.

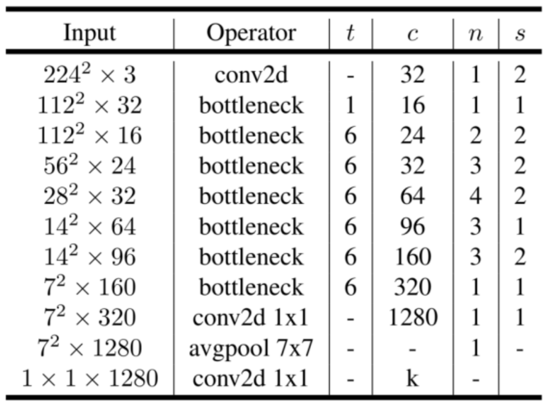
4.1.2.1.1 Single Shot Detectors (SSD)

SSDs are called SSD because they need to take a *one single shot* to detect multiple objects within an image. The older technique was known as Regional Proposal Network (RPN) that needed two shots, one for generating region proposals, and the other for detecting the object in each proposal. Hence, SSD are much faster as compared to the two-shot RPN based approaches.

SSD is a purely convolutional neural network can be divided into following parts:

- Base Neural Network: A base neural network that is used for feature extraction. This base network is a standard Convolutional Neural Network without the final fully connected classification layers. This base network provides a feature map that is used in the next part. In our system the base neural network is the network “MobileNetV2” (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018).

The architecture of MobileNetV2 can be described using the table in Figure 4-10.



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 4-10: MobileNtV2 architecture (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018)

The final fully connected layer is removed, so that instead of getting a prediction we get a feature map on which further convolutions can be applied.

- Additional Convolution Layers: To the base network, additional convolutional layers are added to provide high-level feature maps.
- Prediction Convolution Layers: These are the convolutions that are applied on the high-level feature maps obtained by applying additional convolutions to the output of base network. These prediction convolutions locate and identify the objects in the image detector.

The overall architecture of the Singleshot Detector that we are using can be described using the diagram in 4-11.

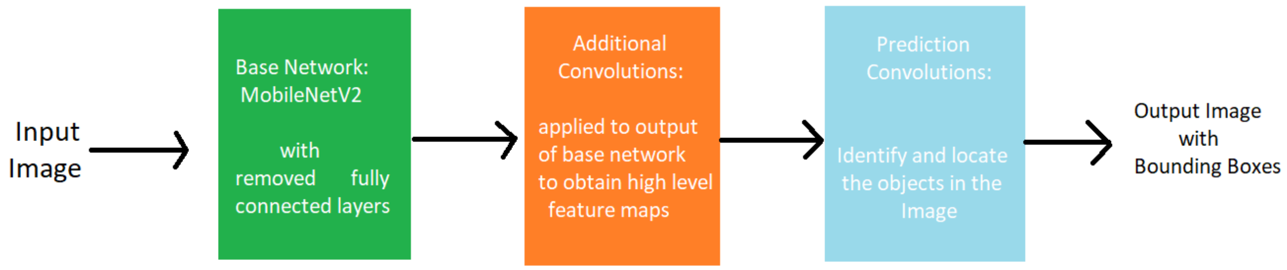

Input Image
Base Network: MobileNetV2
with removed fully connected layers
Additional Convolutions:
applied to output of base network to obtain high level feature maps
Prediction Convolutions:
Identify and locate the objects in the Image
Output Image with Bounding Boxes

Figure 4-11: Schematic Architecture of the SSD network used in our system

The details of the neural network used are much more intricate than the brief description mentioned here. These details can be accessed at the reference mentioned. We used this neural network architecture with pre-trained weights. This network can detect **91** different classes. The classes that can be detected are enlisted in Table 4-1.

Table 4-1: Classes of Object detected by the object detection network

Unlabeled	Person	Bicycle	Car	Motorcycle
Airplane	Bus	Train	Truck	Boat
Traffic light	Fire hydrant	Street sign	Stop sign	Parking meter
Bench	Bird	Cat	Dog	Horse
Sheep	Cow	Elephant	Bear	Zebra
Giraffe	Hat	Backpack	Umbrella	Shoe
Eyeglasses	Handbag	Tie	Suitcase	Frisbee
Skis	Snowboard	Sports ball	Kite	Basketball bat
Baseball glove	Skateboard	Surfboard	Tennis racket	Bottle
Plate	Wine glass	Cup	Fork	Knife
Spoon	Bowl	Banana	Apple	Sandwich
Orange	Broccoli	Carrot	Hot dog	Pizza
Donut	Cake	Chair	Couch	Potted plant
Bed	Mirror	Dining table	Window	Desk

Toilet	Door	Tv	Laptop	Mouse
Remote	Keyboard	Cell phone	Microwave	Oven
Toaster	Sink	Refrigerator	Blender	Book
Clock	Vase	Scissors	Teddy bear	Hair dryer
Toothbrush				

An example of the output image from our system is shown in Figure 4-12.

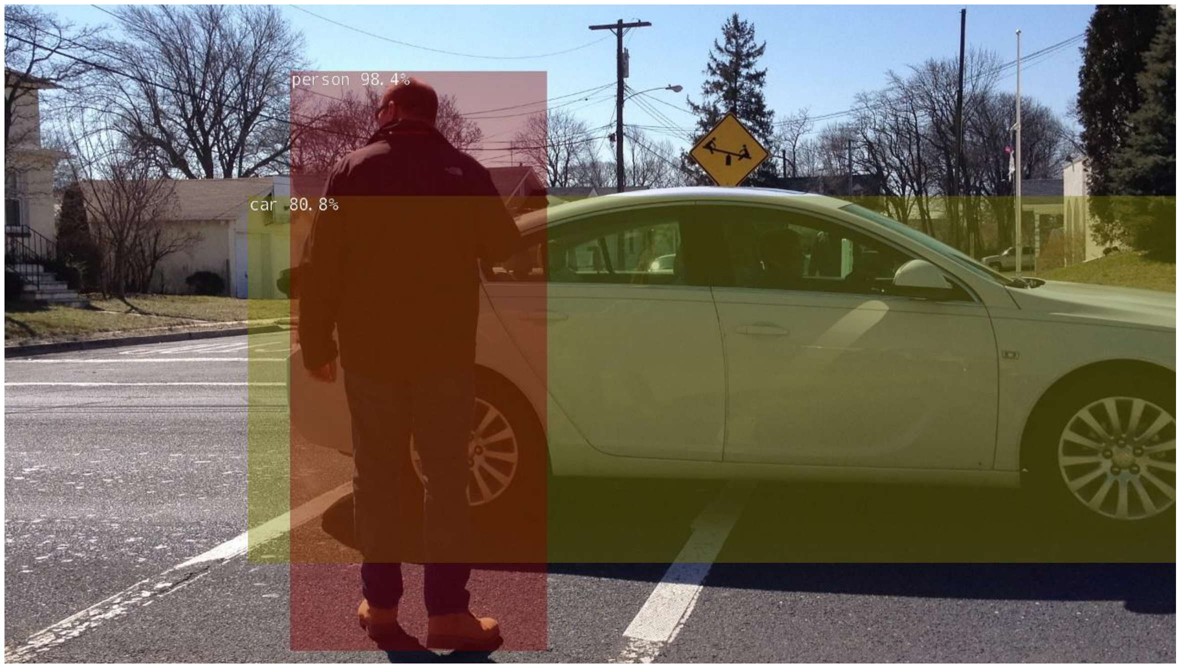


Figure 4-12: Output Image with Bounding Boxes and prediction confidence

4.1.2.1.2 Output of Object Detection

The output of the object detection algorithm detects and locates the objects in the image. Table 4-2 describes the information that is published about each detected object.

Table 4-2: Table showing the details of the information published about detected objects

Information about Detected Objects	Example
Object Class	for e.g. person/car/cup etc.
Prediction Confidence	Expressed using percentage e.g. 80%/90% etc.
Bounding Box	(x_{min}, x_{max}, y_{min}, y_{max}) coordinates to know the bounding box coordinates

In Figure 4-12, two objects, car and a person are detected. So, for each detected object, we have the information described in Table 4-2.

4.1.2.1.3 Issues with Object Detection Only

With object detection, we are detecting objects in each frame. If multiple objects are being detected in an image, then there is no way to connect the current detections to the detections in the previous frame. Moreover, if the objects get occluded behind something, we are no longer able to detect it. Also, if for some reason like a sudden variation in lighting, it is possible that the object detector might miss a detection in an image.

Object tracking can be used on top of object detection to keep track of the moving objects in a video. This will also help us to create trajectories of the detected objects in the video stream which will help us with collision avoidance.

Moreover, whenever there are moving objects in the videos, there would be scenarios when the appearance of the object is not that clear. In that scenario, detection alone would fail while tracking will succeed as it accounts for the motion model and history of the object. Some scenarios in which, object detection fails and object tracking works are as follows:

1. **Occlusion:** If an object or an obstacle gets temporarily occluded from the view, object detection will fail to detect it.
2. **Illumination Variation:** When driving outside there can be sudden changes in illumination. This might cause object detection to temporarily fail.
3. **Motion Blur:** As the camera is mounted on a moving vehicle, objects might get blurred due to the motion of the object or the camera. Hence, the object detector might fail.

There could be some other similar scenarios in which the detector might fail to detect an obstacle or an object.

4.1.2.2 Object Tracking

Kalman filtering was used to track objects. Kalman filtering has following properties that make it beneficial for our use:

1. Prediction of Object's Future Position: This particular property allows us to track object even in the case that we fail to detect the object for some reason. We will update the object's position based on its past motion.
2. Correction of Predictions based on New Measurements: This allows us to correct the position of the object based on latest measurements from the detection algorithm.
3. Reduction of Noise caused by Inaccurate Detections: If for some reason, the bounding box being detected by the object detection algorithm is bouncing around a lot, Kalman filtering helps us in reducing noise and keeping the object tracking smooth.
4. Association of Multiple Objects to their Tracks: Kalman filtering tracking also helps us in associating the objects being detected to their tracks.

4.1.2.2.1 Kalman Filter Equations

The Kalman Filter can be broken down into two steps. One is the prediction step and the other is the update phase. Let us first look at the prediction phase.

4.1.2.2.1.1 Prediction phase

In the prediction phase, next state of the object is predicted based on state transition matrix. The following equations are used in the prediction phase:

$$\bar{x} = Fx$$

Where

$$x = state\ mean = \begin{bmatrix} x_{min} \\ \dot{x}_{min} \\ x_{max} \\ \dot{x}_{max} \\ y_{min} \\ \dot{y}_{min} \\ y_{max} \\ \dot{y}_{max} \end{bmatrix}$$

x_{min}, x_{max}, y_{min}, y_{max} are the coordinates of the bounding box.

$$\bar{x} = predicted\ state = \begin{bmatrix} \bar{x}_{min} \\ \bar{\dot{x}}_{min} \\ \bar{x}_{max} \\ \bar{\dot{x}}_{max} \\ \bar{y}_{min} \\ \bar{\dot{y}}_{min} \\ \bar{y}_{max} \\ \bar{\dot{y}}_{max} \end{bmatrix}$$

The state transition matrix is initialized as

$$F = state\ transition\ matrix = 8 \times 8\ matrix = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

From the above matrix, you can see that we are assuming a constant velocity model where we update the bounding boxes coordinates based on the previous position and the velocity. The above matrix translates to following equations:

$$\bar{x}_{min} = x_{min} + \dot{x}_{min} dt$$

$$\bar{\dot{x}}_{min} = \dot{x}_{min}$$

As you can see, we are assuming a constant velocity model as velocity remains same. Moreover x_{min}, x_{max}, y_{min} and y_{max} are updated independent of each other. We can get similar equations for the others

$$\bar{x}_{max} = x_{max} + \dot{x}_{max} dt$$

$$\bar{\dot{x}}_{max} = \dot{x}_{max}$$

$$\bar{y}_{min} = y_{min} + \dot{y}_{min} dt$$

$$\bar{\dot{y}}_{min} = \dot{y}_{min}$$

$$\bar{y}_{max} = y_{max} + \dot{y}_{max} dt$$

$$\bar{\dot{y}}_{max} = \dot{y}_{max}$$

We set the value of dt as 1 in our code.

$$dt = 1$$

Also, an update for state co-variance matrix is made based on the state transition matrix and process covariance.

$$\bar{P} = FPF^T + Q$$

Where

$$P = state\ covariance\ matrix = 8 \times 8\ matrix$$

$$\bar{P} = Updated\ state\ covariance\ matrix\ (8 \times 8\ matrix)$$

The matrix P is initialized as follows in the code:

$$P_{initial} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

$$F = state\ transition\ matrix\ (as\ described\ above)$$

$$Q = process\ covariance\ matrix$$

The matrix Q is initialized as follows:

$$Q_{initial} = \begin{bmatrix} 0.25 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 1 \end{bmatrix}$$

Now, let us take a brief look at the update phase equations.

4.1.2.2.1.2 Update Phase Equations

Based on the predictions made in the previous step, and the new measurements, we can update the tracks.

In the first step, we calculate a quantity called the residual, represented by y (4 x 1 vector).

$$y = z - H\bar{x}$$

Where

$$z = measurement = 4 \times 1\ vector = \begin{bmatrix} x_{min} \\ x_{max} \\ y_{min} \\ y_{max} \end{bmatrix}$$

Note that the new measurement does not have any velocity values. This is because the detector only outputs the bounding box coordinates, and not any velocities.

$$H = measurement\ function\ matrix = 4 \times 8\ matrix = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Basically, the H matrix is just extracting a 4x1 vector of position values of the bounding boxes from the 8x1 predicted state matrix.

$$\bar{x} = predicted\ state\ (8 \times 1\ vector, as\ described\ before)$$

In the next step, we calculate the Kalman gain, K

$$K = \bar{P}H^T(H\bar{P}H^T + R)^{-1}$$

Where

$$K = Kalman\ Gain\ matrix = 8 \times 4\ \ matrix$$

$$\bar{P} = Predicted\ state\ covariance\ matrix = 8 \times 8\ matrix\ (calculated\ in\ the\ prediction\ phase)$$

$$H = measurement\ function\ matrix = 4 \times 8\ matrix\ (as\ described\ before)$$

$$R = measurement\ noise\ covariance = 4 \times 4\ matrix$$

R matrix is initialized as follows:

$$R_{initial} = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

A low value in the diagonal as compared to the diagonal values in the P matrix means a more reliable measurement. Higher values in the diagonal means that we have unreliable measurements.

After calculating the Kalman gain, we can update the state

$$x = \bar{x} + Ky$$

Where

$$x = updated\ state = 8 \times 1\ state\ vector$$

$$\bar{x} = predicted\ state = 8 \times 1\ predicted\ state\ vector$$

$$K = 8 \times 4\ Kalman\ gain\ matrix$$

$$y = 4 \times 1\ residual\ vector$$

After updating the state, we update the state covariance matrix

$$P = (I - KH)\bar{P}$$

Where

$$P = 8 \times 8\ updated\ state\ covariance\ matrix$$

$$I = 8 \times 8\ identity\ matrix$$

$$K = 8 \times 4\, Kalman\ gain\ matrix$$

$$\bar{P} = 8 \times 8\, predicted\ state\ covariance\ matrix$$

These equations are the basis of our object tracker. In the tracker, we can see that the state vector is updated based on new measurements. Now, in case of multiple objects being tracked and multiple objects being detected, the problem of assigning a new detection to an existing tracker arises. We achieve this using the Hungarian method or the Kuhn-Munkres algorithm. The details of the algorithm will be discussed next.

4.1.2.2.2 Hungarian Algorithm or the Kuhn-Munkres Algorithm

In the event, that there are multiple detections, we need to assign each detection to an existing track. This is achieved by using the intersection over union (IoU) of a tracker bounding box and detection bounding box as a metric. Intersection over union is defined as follows

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union}$$

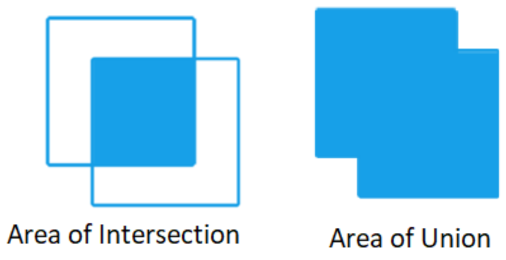


Figure 4-13: Area of Intersection (Left); Area of Union (Right)

IoU between each detection and existing track is calculated. This gives us an IoU matrix where each entry records the IoU value between the corresponding detection (row) and track (column).

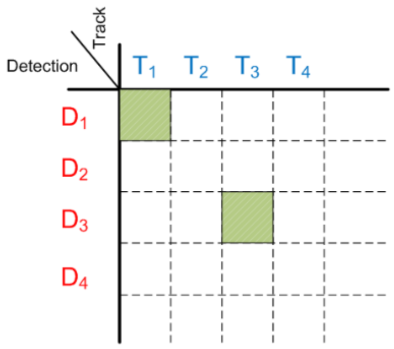


Figure 4-14: Representation of IoU matrix

Based on this IoU matrix, we can assign each detection to an existing track using an algorithm called linear assignment. The linear assignment problem is also known as minimum weight matching in bipartite graphs. Let the problem be described by a matrix C, where each entry C(i, j) is the cost of matching vertex i (of the first set) and the vertex j (of the second set). The goal is to find complete assignment between the first and second set with the minimum cost.

In order to formally state this, let X be a Boolean matrix where X(i, j) = 1 if and only if row i is assigned to column j. The optimal assignment has following cost

$$min \sum_{i} \sum_{j} C_{i,j} X_{i,j}$$

Such that each row is assigned to at most one column and each column is assigned to at least one row. We use a built-in python library to achieve this assignment. The cost matrix used is the negative of the IoU matrix as the linear assignment algorithm minimizes the costs, and we need to find an assignment that maximizes the IoU between detections and tracks.

$$Matched\ indices = Linear\ assignment(-IoU\ matrix)$$

4.1.2.2.3 Tracking Pipeline

Based on linear assignment results, we also keep a list of unmatched detections and unmatched trackers. When an object enters the frame for the first time, it is not assigned to any existing track, and it is referred to as an unmatched detection. Similarly, any match that has an IoU of less than *0.33* signifies an existence of an unmatched track. This usually happens if the object leaves the frame and the previously established track has no more detection to associate with. This could also happen if the object being tracked gets occluded for some reason. In these scenarios, the track is referred to as an unmatched track. Matched detections, unmatched detection and unmatched tracks are defined as follows:

$$Matched\ Detection\ and\ Tracker = A\ detection\ and\ tracker\ are\ matched\ if\ they\ are\ of\ same\ class\ and\ IoU > 0.33$$

$$Unmatched\ Detection = A\ detection\ that\ doesn't\ get\ matched\ to\ any\ tracker\ i.e.\max{(IoU)} < 0.33$$

$$Unmatched\ tracker = A\ tracker\ that\ doesn't\ get\ matched\ to\ any\ detection\ i.e.\max(IoU) < 0.33$$

Now, we have three different things i) Matched detections and trackers ii) Unmatched detections and iii) Unmatched trackers.

To build a consistent system to keep track of the detected objects, two new design parameters are introduced. These parameters are as follows:

$$Minimum\ Hits = Number\ of\ conecutive\ matches\ needed\ to\ establish\ a\ new\ tracker$$

$$Maximum\ Age = Number\ of\ consecutive\ unmatched\ detections\ before\ a\ track\ is\ deleted$$

These parameters can be tuned to improve the performance of the tracker.

Based on the matched detections, unmatched detections, unmatched trackers, minimum hits, and maximum age, we built a pipeline to build an object tracker that can consistently track multiple objects. The objects that are being tracked are maintained in a list called "Good Tracker List".

The pipeline can be explained using the flow chart in Figure 4-15.

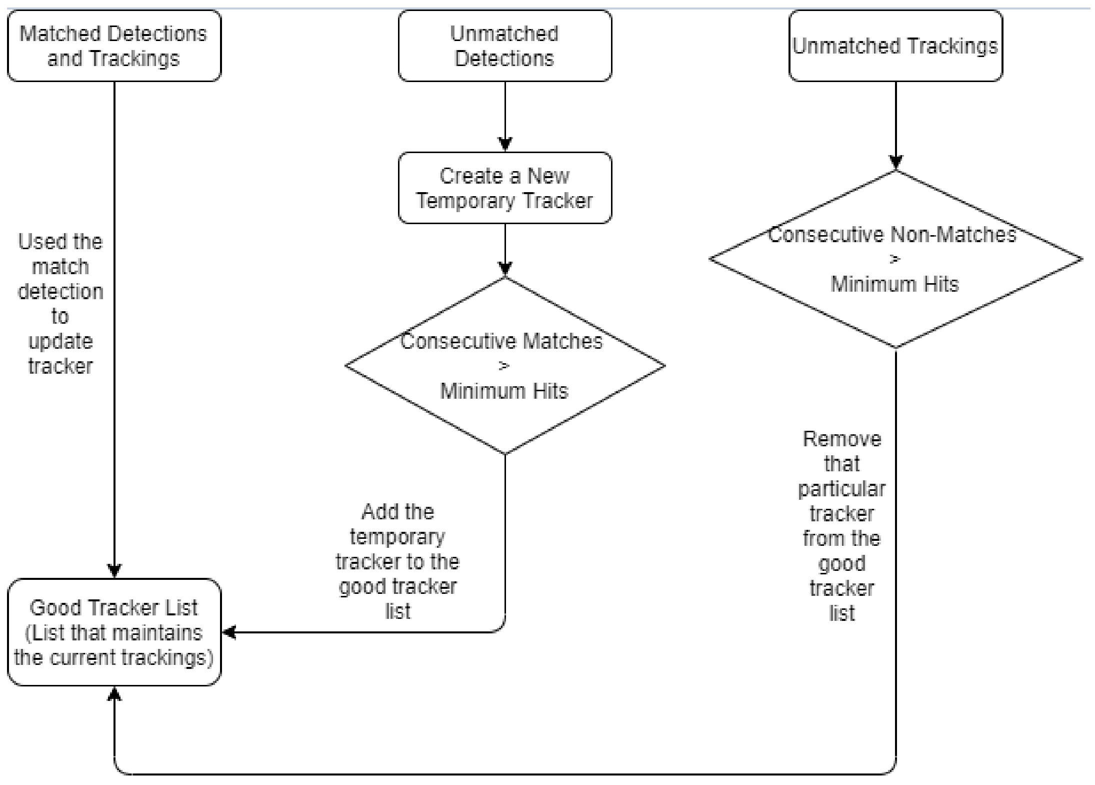

Matched Detections and Trackings
Unmatched Detections
Unmatched Trackings
Create a New Temporary Tracker
Consecutive Non-Matches > Minimum Hits
Used the match detection to update tracker
Consecutive Matches > Minimum Hits
Remove that particular tracker from the good tracker list
Add the temporary tracker to the good tracker list
Good Tracker List (List that maintains the current trackings)

Figure 4-15: Flowchart explaining how i) Matched detections ii) Unmatched Detections iii) Unmatched Trackers are added or removed from the Good Tracker list

Using the mechanism described above we can keep a track of multiple objects at a time.

4.1.3 Steering Wheel Overlay

An overlay was added on the screen that overlaid guidelines on the video feed being transmitted to the operator. The guidelines that are overlaid on the video based on the steering wheel angle. It helps the operator in steering the car. An example of image being transmitted to the operator can be seen in Figure 4-16.

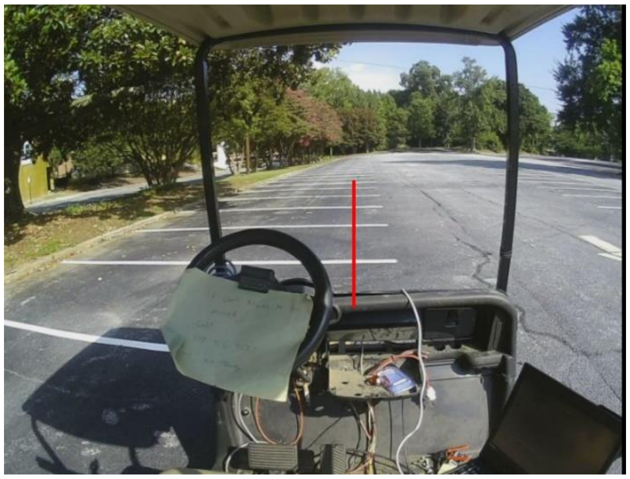


Figure 4-16: Ax example image with steering guidelines overlaid on the image being transmitted to the operator

After the video feed, the second most important feedback data that the user uses to make decisions is the LIDAR data. The laser scan from the LIDAR is processed by the remote environment computer, and this processed data is then sent to the operator. The details of this processing will be discussed next.

4.2 LIDAR Data

A 2D LIDAR was also mounted on the front of the golf cart. “RPLIDAR A1” which is an economical 2D LIDAR was used. It is a 360^o omnidirectional laser range scanner. It has a range of between 0.15 - 12 meters. The nominal scanning frequency of the radar is 5.5 Hz, but it can be increased up to 10 Hz. The distance resolution is usually less than 0.5 mm (or < 1% of the distance). The angular resolution is typically 1^o.

Figure 4-17: An Image of RPLIDAR A1 system

4.2.1 Data from the LIDAR

For each sample point data, we get the following information, as per Table 4-3.

Table 4-3: RPLIDAR A1 sample point data information

Data Type	**Unit**	**Distance**
Distance	m	Measured distance value between the rotating core and the sampling point
Heading	Degree	Heading angle of the measurement
Quality	Level (0-1)	Quality of the measurement
Start Flag	(Boolean)	Flag of a new scan

We get this data in the form of a 360 x 1 array where the array index corresponds to the angle and the numerical value at that index refers to the distance of the sample point. Therefore, we can know both the distance and the heading. This can be illustrated using the diagram in Figure 4-18.

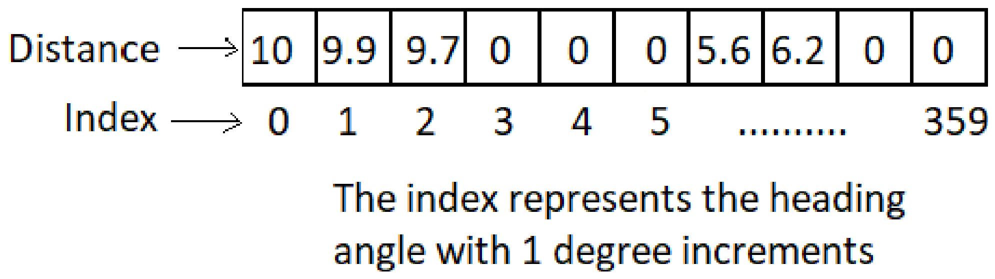

Distance
10
9.9
9.7
0
0
0
5.6
6.2
0
0
Index
0
1
2
3
4
5
....
359
The index represents the heading angle with 1 degree increments

Figure 4-18: The array with distance (m) and heading angle (degrees)

A value of 0 in the array means that the sample point is not in the measuring range (0.15 - 12m) of the LIDAR. In our system, we assume that a value of 0 means that the sample point is in free space (distance > 12 m).

4.2.2 Mounting position of LIDAR

The LIDAR is mounted on front of the golf cart. Therefore, we are only interested in the scan of the area that corresponds to 180° in front of the golf cart. The position of the laser scanner with respect to the golf cart can be seen in the diagram in Figure 4-19.

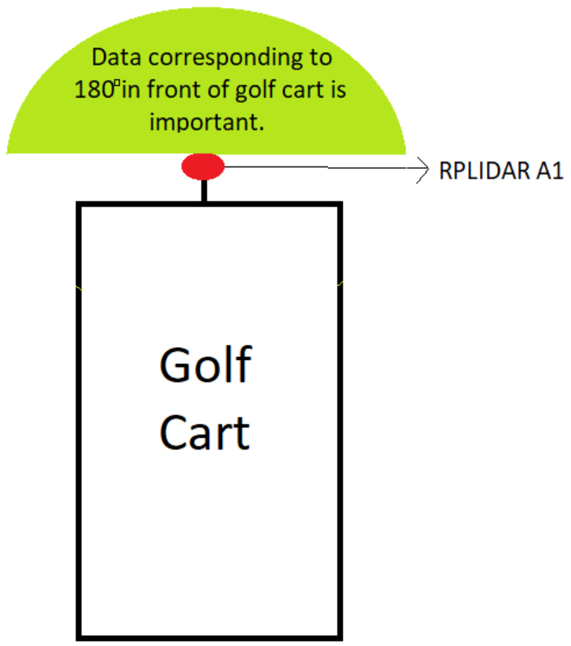

Data corresponding to 180° in front of golf cart is important.
RPLIDAR A1
Golf Cart

Figure 4-19: Position of LIDAR w.r.t. the golf cart

4.2.3 Safety System based on LIDAR

We introduce a safety system for the golf cart based on the LIDAR scan. The LIDAR scan gives us a point cloud in 2D of the area surrounding the sensor. As the LIDAR is mounted in front of the golf cart, the data from the 360° scan corresponding to the area in front of the golf cart is of importance to us (the area marked green in figure 26). Using this data, we can implement a simple safety system that will allow us to stop the golf cart, if the operator accidentally drives the golf cart into an obstacle.

Our safety system is based on the following approach:

- Create a virtual envelope in front of the golf cart.
- If any sample point from the laser scan is inside that virtual envelope, let the controller know that there is an imminent collision, so brakes could be applied.

The virtual envelope that is created is given by following equation:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$y \geq 0$$

$$a = 0.85m$$

$$b = 2m$$

This gives us an ellipse that has a major axis of 2 m along y-axis and a minor axis of 1.7 m along the x-axis. The x-axis is aligned with the front bumper of the golf cart and y-axis is perpendicular to it. A graphical illustration of this envelope can be seen in Figure 4-20.

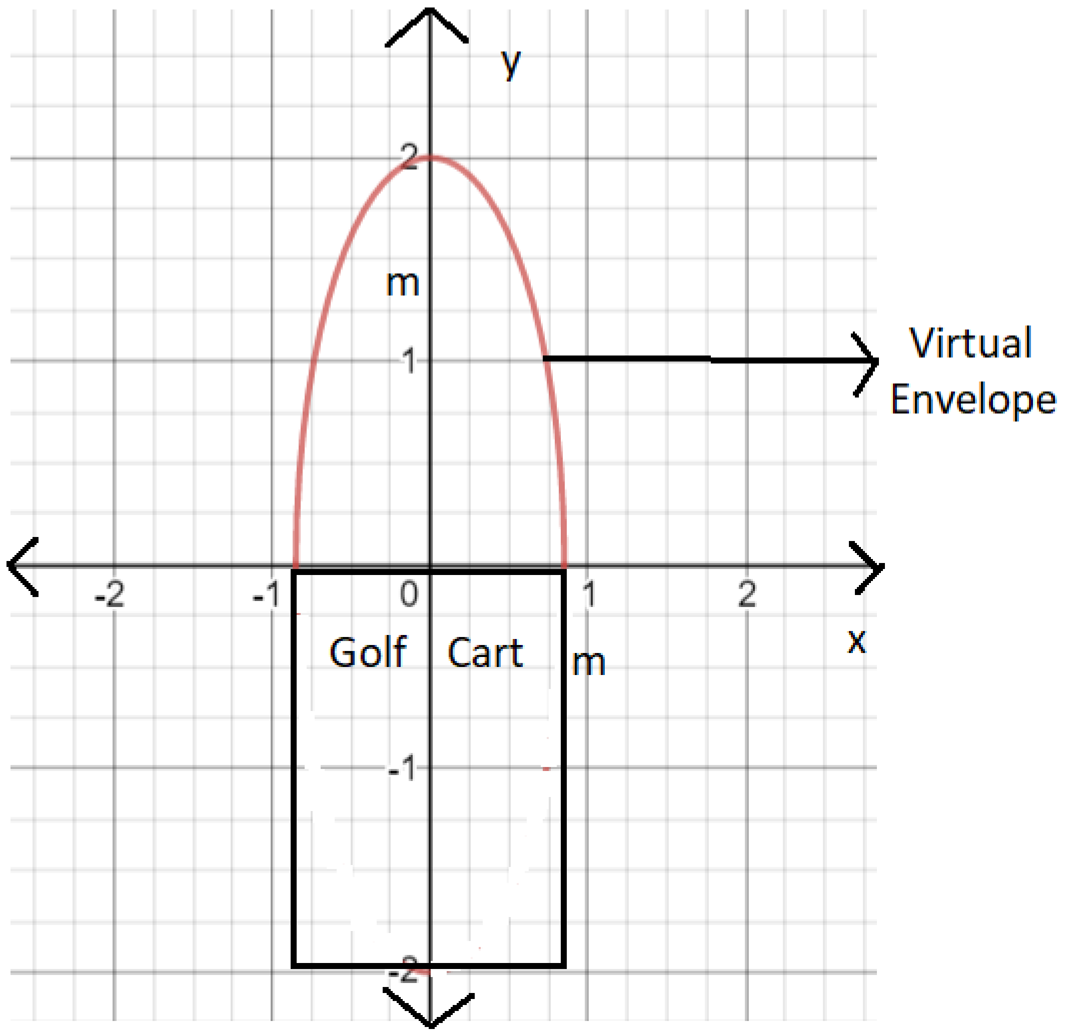


Figure 4-20: Graphical illustration of virtual envelope in front of golf cart

If the actual LIDAR scan finds any sample points inside this virtual envelope, the controller is alerted that there is an obstacle inside the virtual envelope. Brakes are applied and the vehicle is brought to a safe stop.

Figure 4-21 illustrates this concept.

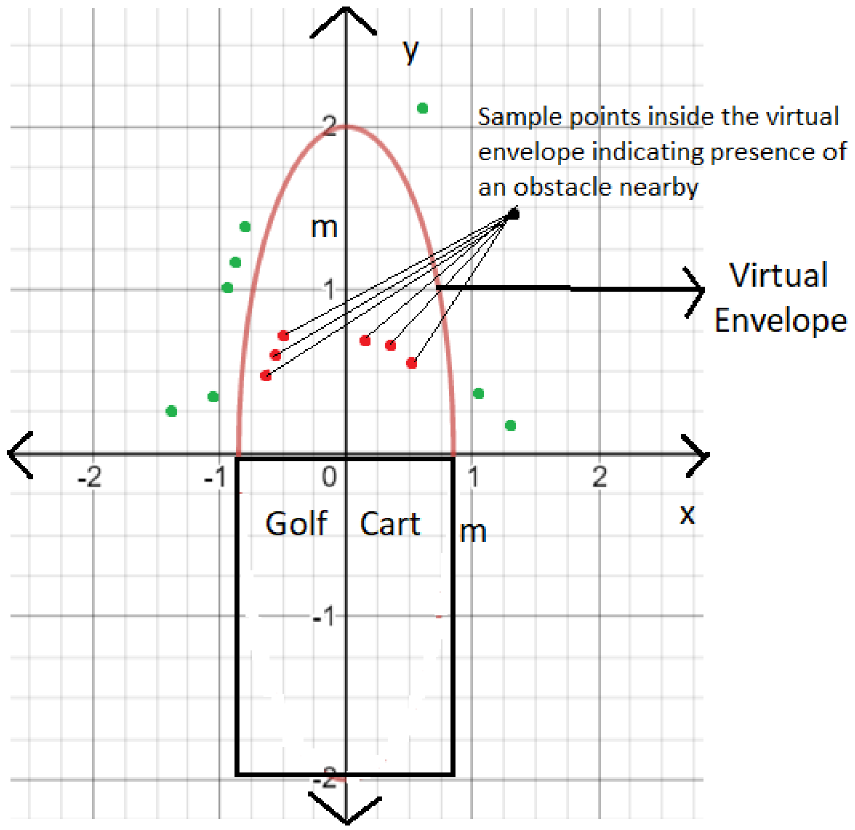

y
Sample points inside the virtual envelope indicating presence of an obstacle nearby
Virtual Envelope
m
Golf Cart
m
x
-2
-1
0
1
2
-1
-2
1
2

Figure 4-21: Illustration of sample points inside the safety envelope

As you can see in the above figure, if there is sample point that is detected inside the safety envelope, we know that a collision is imminent, and the vehicle is brought to a stop.

Based on the data from the LIDAR, a method was devised to calculate the number of obstacles and the approximate width of those obstacles.

4.2.4 Number of Obstacles based on LIDAR data

Based on the data coming from the LIDAR a simple algorithm was developed to count the total number of obstacles in front of the LIDAR. As we saw before, we have the data in the form of an array. All the consecutive non-zero members of the array are grouped into as one obstacle. The total number of such groups in the array gives us the total number of obstacles. This simple approach can be explained using the diagram in Figure 4-22.

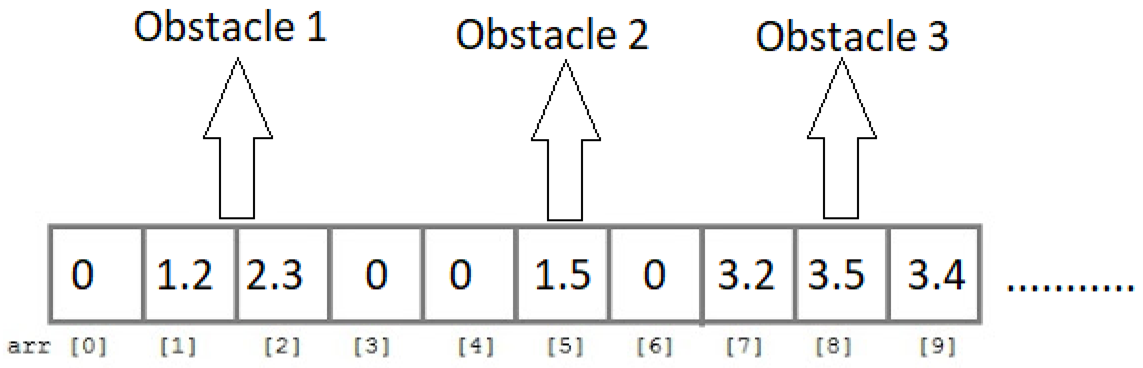


Figure 4-22: An example showing 3 obstacles (as there are 3 groups of non-zero values)

We also used the data from the LIDAR to approximate the heading and width of each obstacle.

4.2.5 Heading (Angle) at which each obstacle is present

The heading (angle) of each obstacle with respect to the LIDAR is also determined by simply taking the average of the first and last heading angle for each obstacle. For e.g. if in figure 30, for obstacle 3, say the indices 7, 8 and 9 correspond to -83^o, -82^o and -81^o respectively, then the heading (angle) for obstacle would simply be the average of the first (-83^o) heading angle and last (-81^o) heading angle equating to -82^o.

4.2.6 Width of each Obstacle using LIDAR Data

The width of the obstacles was calculated using the law of cosines. The Law of Cosines (or the cosine rule) says that

$$c^2 = a^2 + b^2 - 2ab\cos C$$

Where

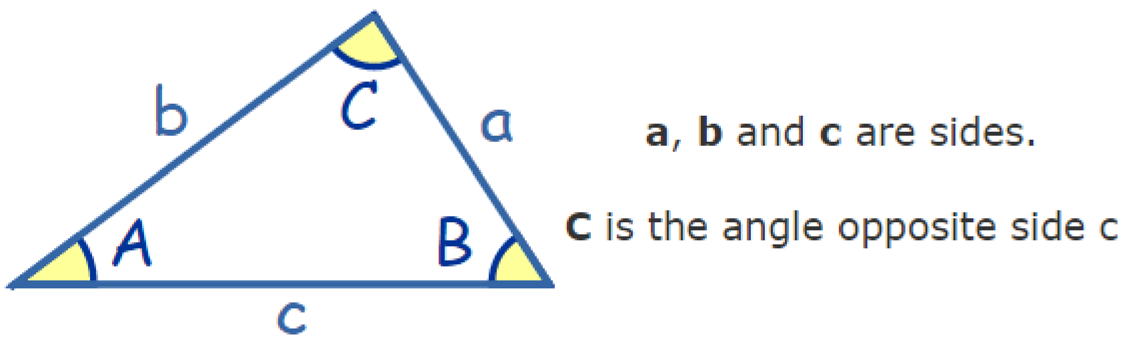


Figure 4-23: A triangle explaining the Cosine Rule

Using the law of cosines, we can approximate the width of the obstacles. An example calculation is as follows:

Say we have the data in Table 4-4 for an obstacle.

Table 4-4: Example Data used to show steps followed to calculate obstacle width

Array Index	Corresponding Heading Angle	Distance (m)
13	-77°	4.67
14	-76°	4.34
15	-75°	4.28
16	-74°	4.87
17	-73°	3.90

Using the first and the last entry we can create a triangle and use the law of cosines to determine the obstacle width.

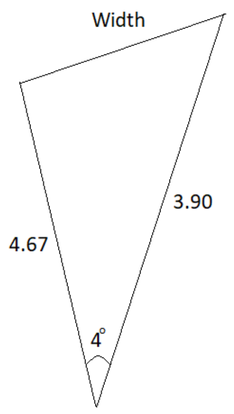


Figure 4-24: Width of an Obstacle based on Law of Cosines

The width of the obstacle can then be calculated using the Cosine Law:

$$Width = \sqrt{4.67^2 + 3.90^2 - 2(4.67)(3.90)\cos 4^o} = 0.83\ m$$

If there is a single sample point for an obstacle, then we do not calculate its width.

4.2.7 LIDAR Feedback to Operator

Hence, based on LIDAR data, we determined and transmitted to the user, the following data:

- Is there an imminent collision based on virtual safety envelope and LIDAR data? If yes, the controller stops the car and the user is informed.
- The total number of obstacles present in front of the vehicle.
- At what distance is each obstacle present.
- At what angle is each obstacle present.
- Width of each obstacle in front of the vehicle.

Based on the above data, the image shown in Figure 4-25 is created and displayed to the operator. In the image, each obstacle is represented by a red circle. The size of the circle is proportional to the size of the obstacle in front of the vehicle. The location of the red circle in the image corresponds to the location of the obstacle with respect to the vehicle.

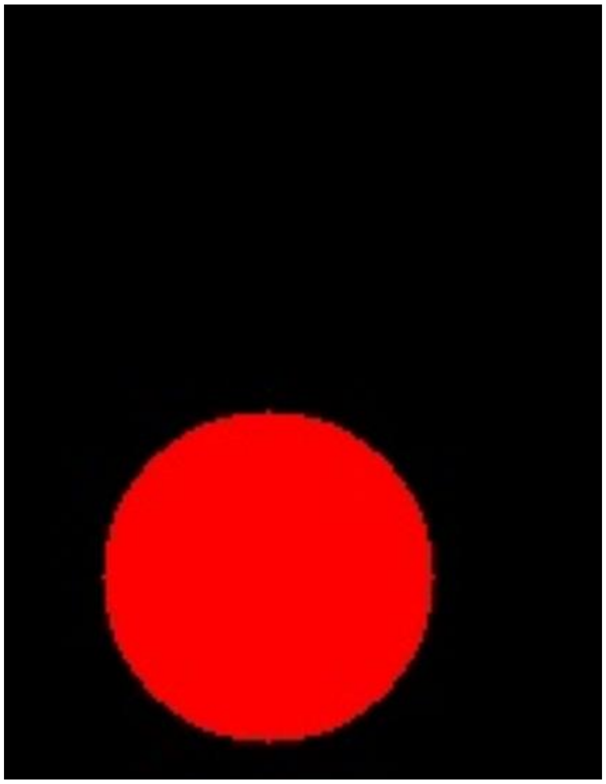


Figure 4-25: An image showing the LIDAR data being transmitted to the operator

Now, we will explain the RPM sensor that was built using the Hall-Effect sensor ad magnets.

4.3 Speed of Golf Cart

The golf cart has a built-in mechanism to measure speed. But, since we cannot temper with the built-in controller, we do not have access to that data. Therefore, a simple method to measure speed of the golf cart was built.

The speed measuring device built was based on hall-effect sensor. A hall-effect sensor is a device that is used to measure the strength of the magnetic field. Its output voltage is directly proportional to the magnetic field strength through it.

Based on this principle, a simple rotational speed measurement unit can be built. Whenever the magnetic field passes through the hall effect sensor, it gives an output voltage which can be detected using a micro-controller. If we count the number of spikes and keep track of time, we can measure the rpm of the vehicle which can be used to find speed. The schematic of the circuit is shown in Figure 4-26.

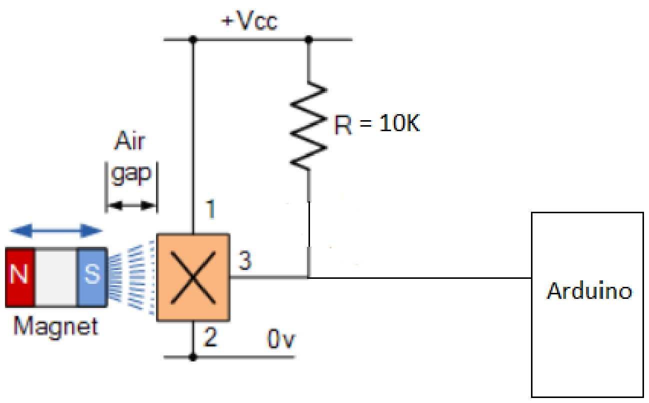


Figure 4-26: Schematic for the Hall effect sensor circuit to measure RPM

Magnets were mounted on the rotating shaft. Each time the magnet passed in front of the sensor it would send a high voltage signal to the Arduino. If we count the total number of spikes in one second, we can calculate the angular speed of the wheels. In our system, two magnets were used, so *two ticks signal a single rotation.* The rotational speed of wheels was converted to linear velocity using the following simple formula:

$$v = \frac{2\pi}{60}.r.N$$

Where

$$v = linear\ velocity$$

$$r = radius$$

$$N = number\ of\ revolutions\ per\ minute$$

The calculated speed of the vehicle was published over the ROS network.

4.4 Summary

In this chapter, the details of the auxiliary systems that were added to assist with the remote driving are discussed. Following auxiliary systems were added:

- Processed Video feed:
    - User has the option to see a video stream that is made up of semantically segmented images. In this type of segmentation, each pixel is classified. The semantic segmentation was achieved using a Fully Convolutional Network.
    - The other option the user has is to see a video stream with object detection and tracking. Object detection was achieved using a Single-Shot detector, and tracking was achieved using a Kalman filter.
    - A steering wheel direction overlay was also added to assist the operator.
- LIDAR Data and Safety system:
    - Using a simple 2D LIDAR mounted on front of the cart, following information about obstacles was calculated:
        - Total number of obstacles in front of vehicle
        - Distance at which each obstacle is present
        - Heading (angle) at which each obstacle is present
        - Width of each obstacle
    - A safety system was built which would automatically stop the vehicle if there is an obstacle present in front of the vehicle.
- Speed of vehicle was measured using magnets and hall-effect sensors.

The objectives of all the systems is to provide the operator with as much information as possible about the remote vehicle environment. Increasing telepresence, would make operating the vehicle easier.

5 Experimental Results

In this chapter, we will discuss the experimental results from teleoperation testing. The teleoperation system was tested with a human operator. The operator was an experienced human driver who was asked to perform various tasks. Moreover, tests were also performed to quantify delay in communications. Results from those tests will also be discussed in this chapter.

5.1 Testing Setup using Human Operator

An experienced human driver was asked to perform two tasks. The first task was to drive the vehicle in forward gear around a parking lot. The second task was to drive the vehicle remotely in the reverse gear from one parking spot to another.

5.1.1 Setup and Results for Forward Trajectory Task

The operator was asked to drive the vehicle in forward gear around the path shown in Figure 5-1.

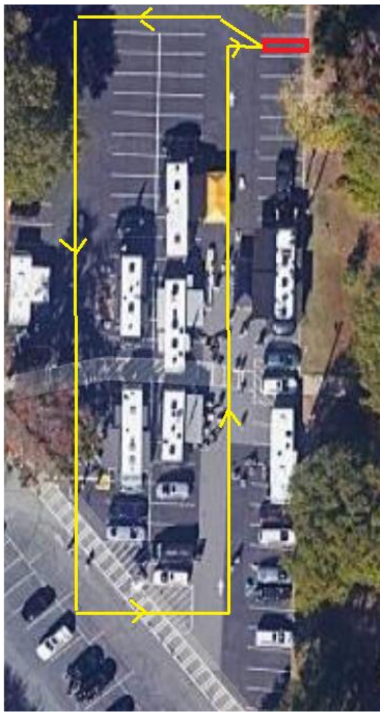


Figure 5-1: Route for Forward Trajectory Task

The test would be categorized as a PASS if the operator managed to drive the vehicle around the parking lot and managed it to park it in the same spot again. To compare the performance with in-situ driving the same task was also performed by the same driver while driving the vehicle in-situ. The comparison of performance in two scenarios is in Table 5-1.

Table 5-1: Forward trajectory tasks results

	In-situ Driving	**Remote Driving**
Pass/Fail	Pass	Pass
Drive Time	101 seconds	160 seconds
No. of Stops	3	6

Where,

- Pass/Fail: Whether the driver was able to accomplish the task successfully i.e. was the driver able to complete the route and park the vehicle successfully into the parking spot.
- Drive Time: The total time required by the driver to complete the task
- No. of stops: The number of times the driver stopped the vehicle while completing the task.

5.1.2 Setup and Results for Reverse Trajectory Task

The operator was asked to drive the vehicle in reverse gear around the path shown in Figure 5-2.

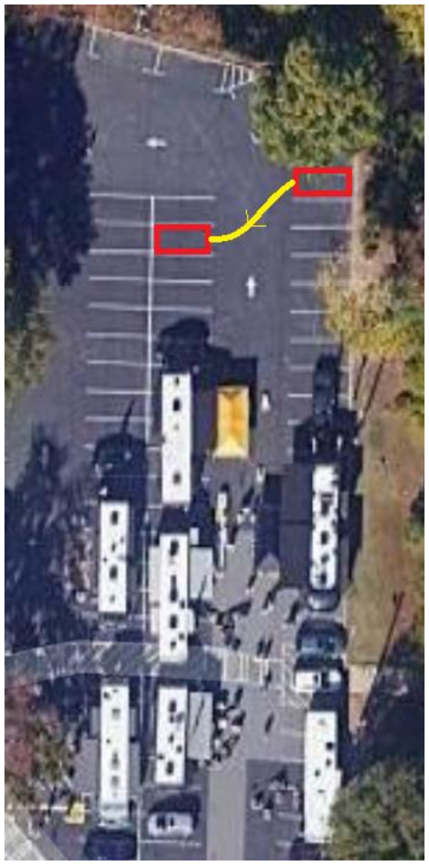


Figure 5-2: Route for Reverse Trajectory Task

Like the forward trajectory task, the test would be categorized as a PASS if the operator managed to park the vehicle successfully in the parking spot. To compare the performance with in-situ driving the same task was also performed by the same driver while driving the vehicle in-situ. The comparison of performance in two scenarios is in Table 5-2.

Table 5-2: Reverse trajectory task results

	In-situ Driving	**Remote Driving**
Pass/Fail	Pass	Pass
Drive Time	20 seconds	40 seconds
No. of Stops	1	3

Where,

- Pass/Fail: Whether the driver was able to accomplish the task successfully i.e. was the driver able to complete the route and park the vehicle successfully into the parking spot.
- Drive Time: The total time required by the driver to complete the task
- No. of stops: The number of times the driver stopped the vehicle while completing the task.

5.2 Quantifying Delay in Communication

Some data was also collected and analyzed to quantify the delays in communication. As described in chapter 3, two-way communication is going on between the control station and the remote vehicle.

1. Communication from remote vehicle to control station
    a. Video feed
    b. Lidar data feed
2. Communication from control station to remote vehicle
    a. Comprises of data from the G27 racing wheel to drive the vehicle

Data to quantify latency in each of the above communication channel was collected.

5.2.1 Delay in Communication from Remote Vehicle to Control Station

The delay in communication was measured for both the video feed and LIDAR data.

5.2.1.1 Video Feed Latency

At a given moment the operator is either watching the front camera stream or the reverse camera stream. Hence, only one video stream is being transmitted over the cellular network. The video was being transmitted at 24 fps. The latency was measured by comparing the time stamp a particular frame was published with the time stamp it was received at.

The latency tests were performed at different times of the day to account for the variability in network traffic during different times of the day. The tests were carried out on a Wednesday at 11 am, 1 pm, 3 pm 5 pm, 7 pm and 9 pm.

- **Video Feed Latency at 11 am:**

Figure 5-3 shows the lag (seconds) in video feed over a course of about 3500 frames.

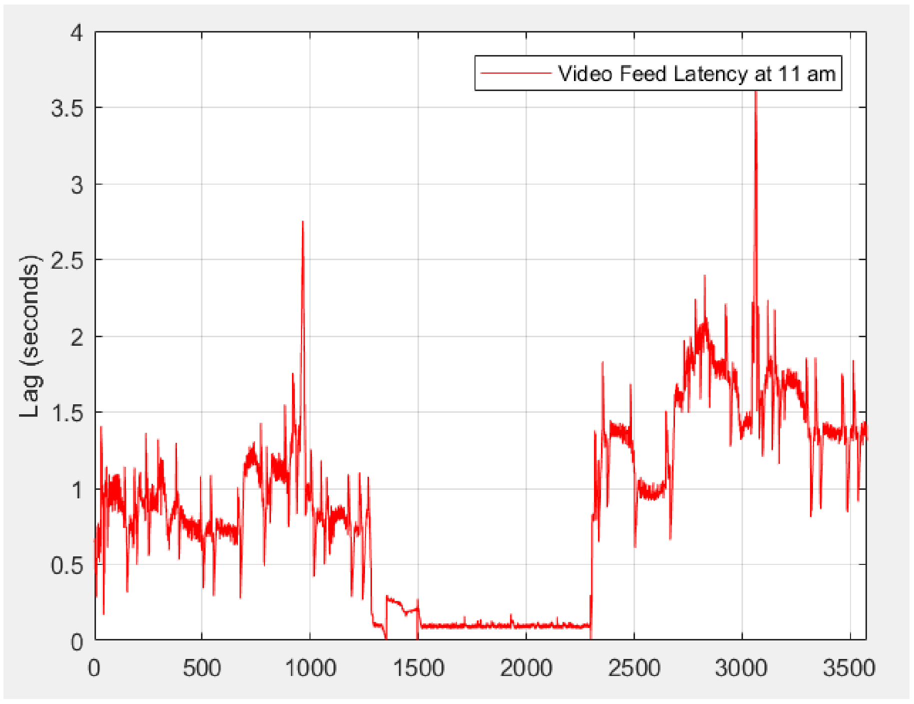


Figure 5-3: Lag in Video Feed at 11 am

A box plot for the same data can be seen in Figure 5-4.

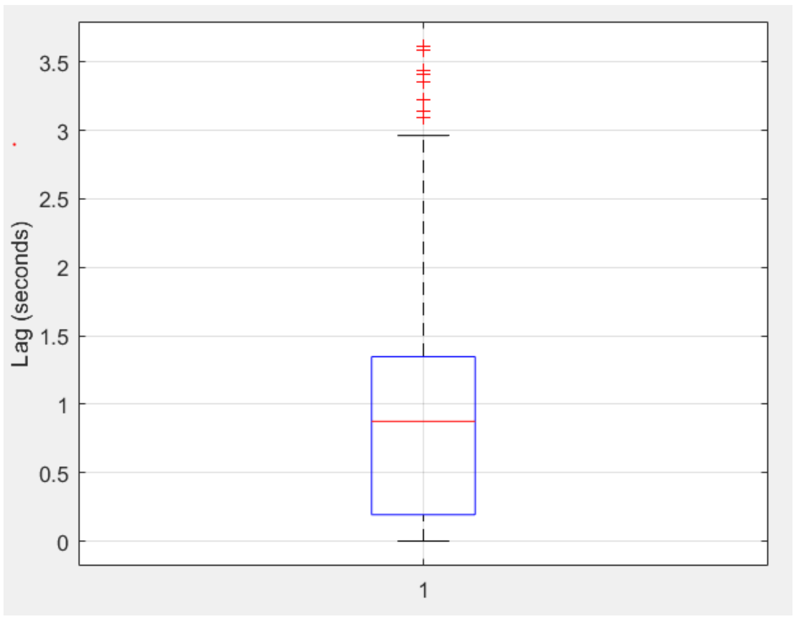


Figure 5-4: Box plot for Video Feed latency at 11 am

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.875\ seconds = 875\ ms$$

$$Standard\ Deviation = 0.602\ seconds = 602\ ms$$

- **Video Feed Latency at 1 pm:**

Figure 5-5 shows the lag (seconds) in video feed over a course of about 3500 frames.

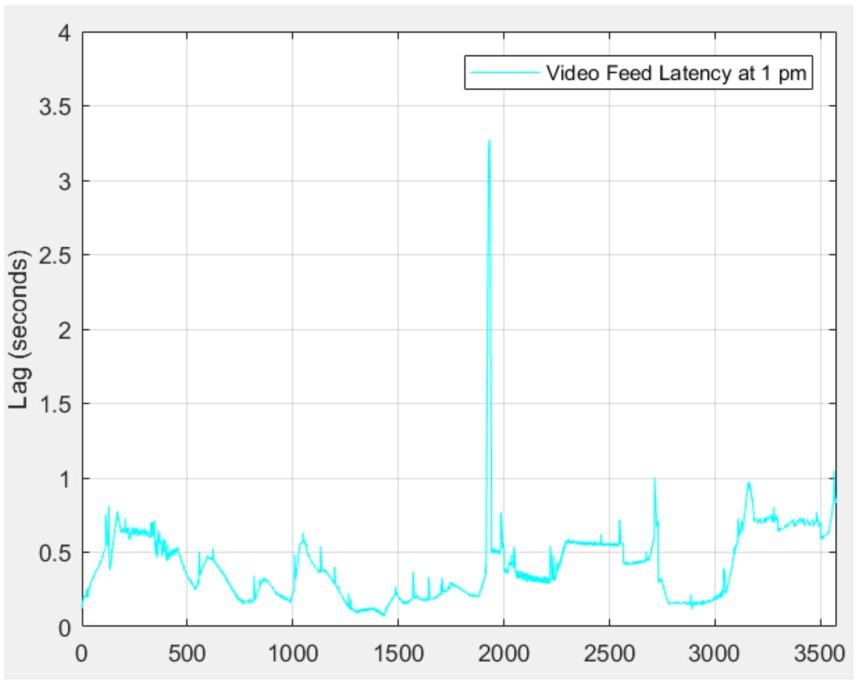


Figure 5-5: Lag in Video Feed at 1 pm

A box plot for the same data can be seen in Figure 5-6.

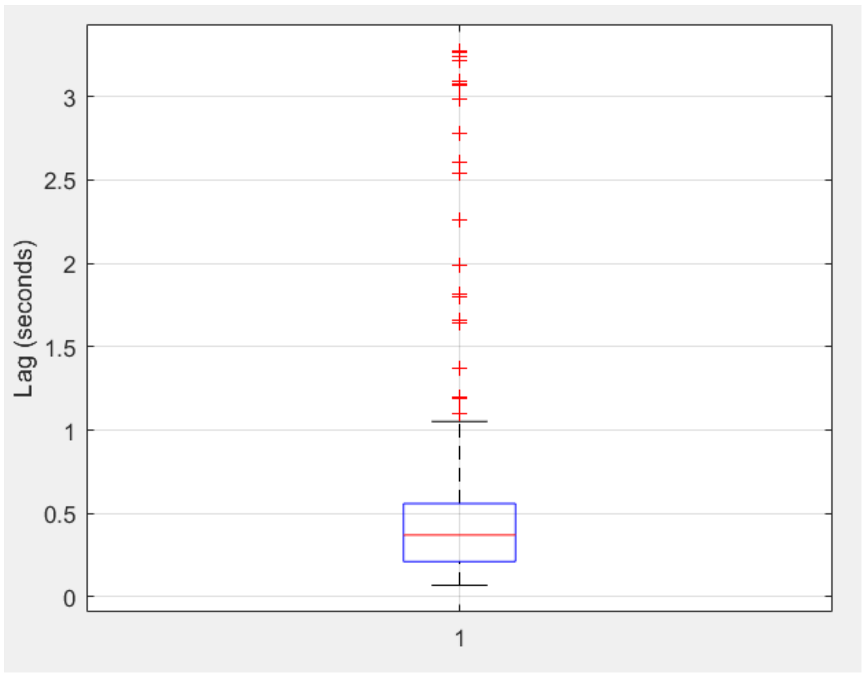


Figure 5-6: Box plot for Video Feed latency at 1 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.411\ seconds = 411\ ms$$

$$Standard\ Deviation = 0.263\ seconds = 263\ ms$$

- **Video Feed Latency at 3 pm:**

  Figure 5-7 shows the lag (seconds) in video feed over a course of about 3500 frames.

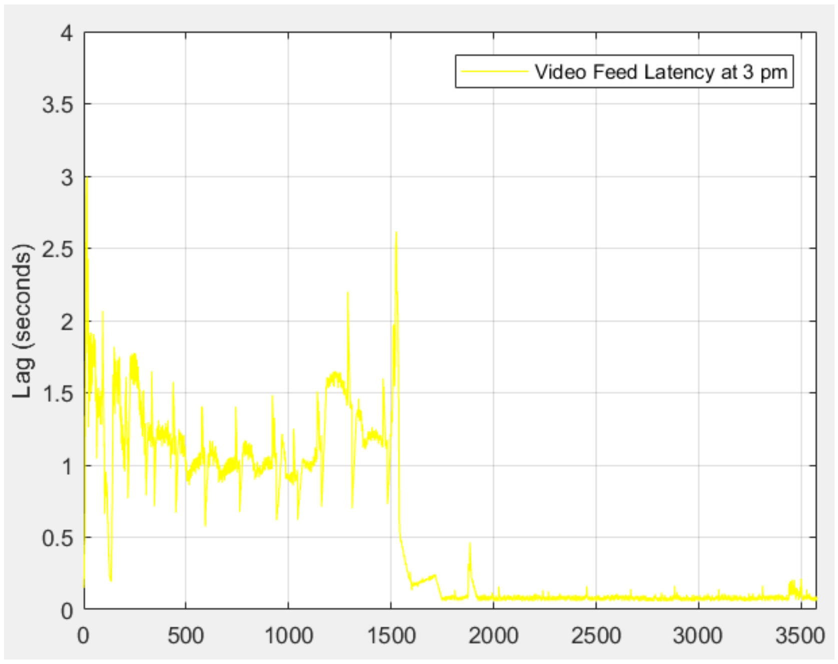


Figure 5-7: Lag in Video Feed (seconds) at 3 pm

A box plot for the same data can be seen in Figure 5-8.

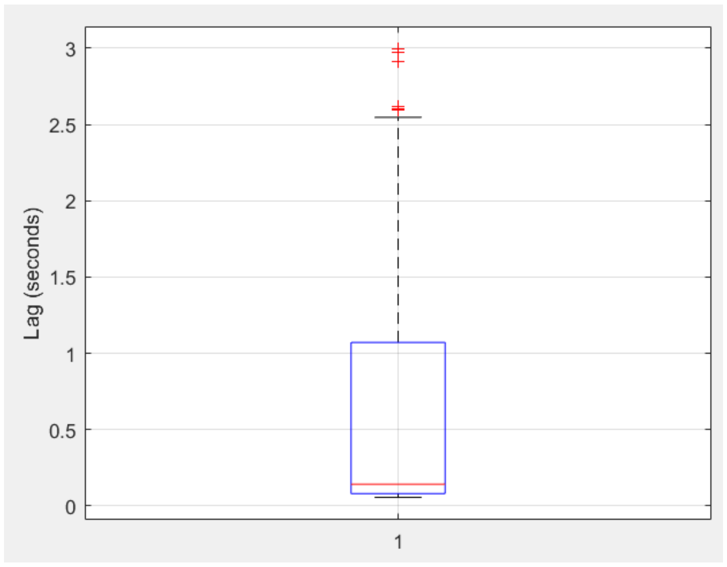


Figure 5-8: Box plot for Video Feed latency at 3 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.567\ seconds = 567\ ms$$

$$Standard\ Deviation = 0.581\ seconds = 581\ ms$$

- **Video Feed Latency at 5 pm:**

Figure 5-9 shows the lag (seconds) in video feed over a course of about 3500 frames.

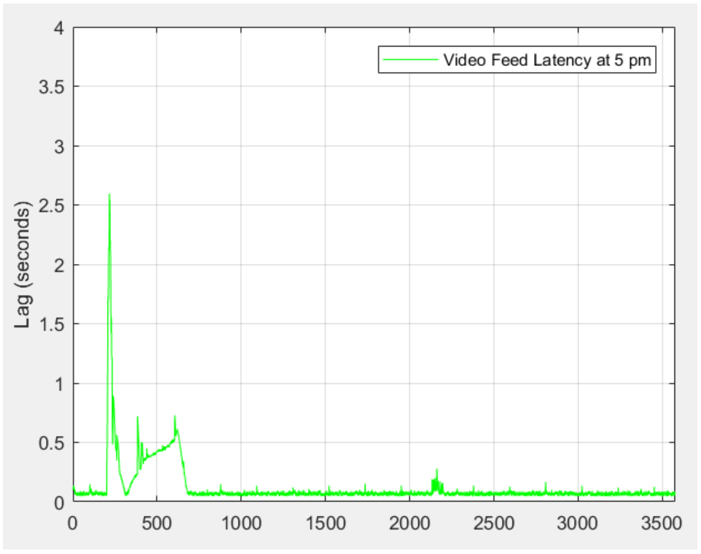


Figure 5-9: Lag (seconds) in Video Feed at 5 pm

A box plot for the same data can be seen in Figure 5-10.

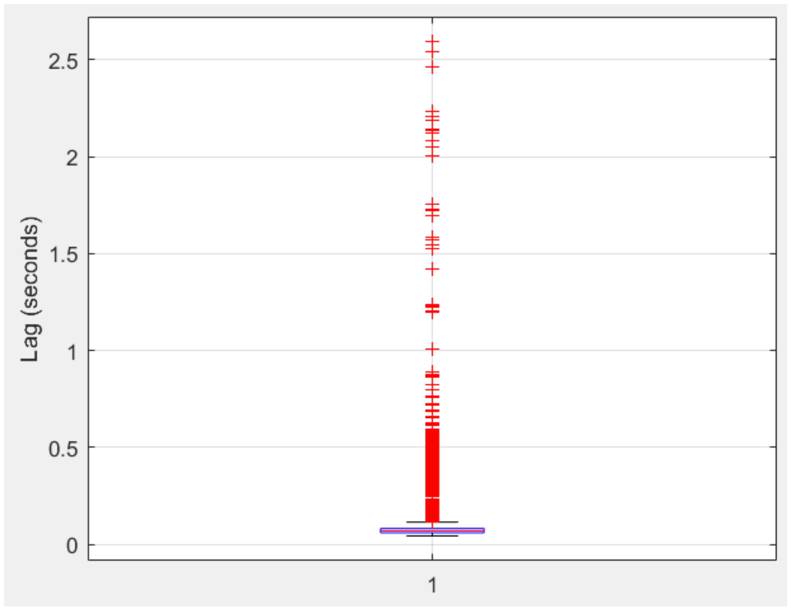


Figure 5-10: Box plot for Video Feed latency at 5 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.121\ seconds = 121\ ms$$

$$Standard\ Deviation = 0.195\ seconds = 195\ ms$$

- **Video Feed Latency at 7 pm:**

Figure 5-11 shows the lag (seconds) in video feed over a course of about 3500 frames.

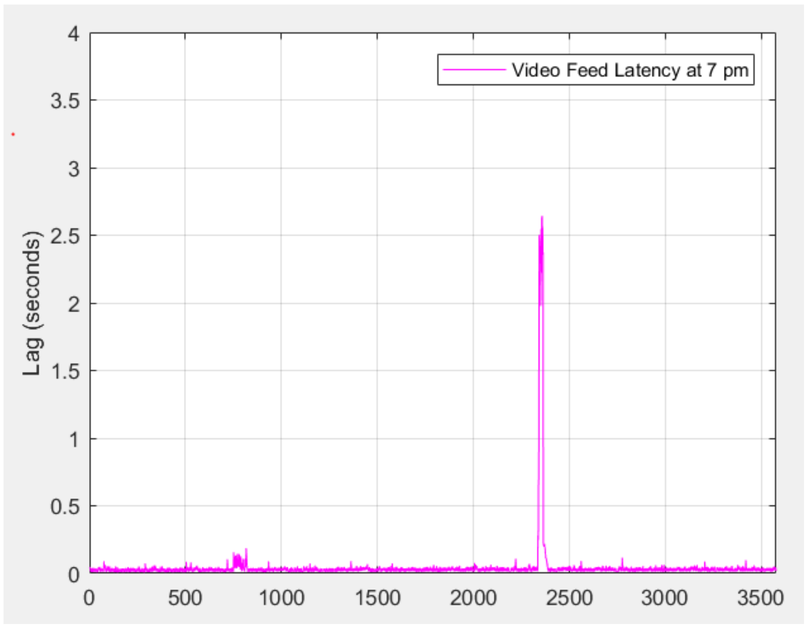


Figure 5-11: Lag (seconds) in Video Feed at 7 pm

A box plot for the same data can be seen in Figure 5-12.

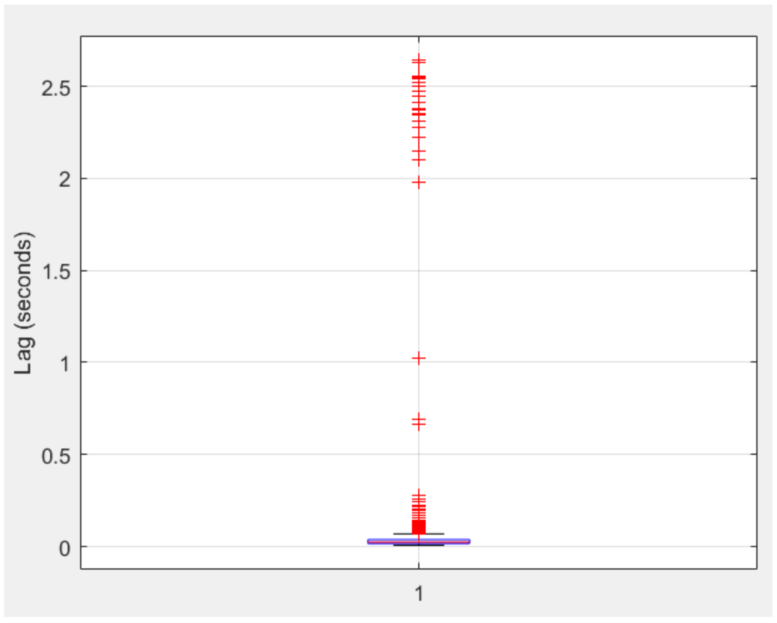


Figure 5-12: Box plot for Video Feed latency

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.045\ seconds = 45\ ms$$

$$Standard\ Deviation = 0.189\ seconds = 189\ ms$$

- **Video Feed Latency at 9 pm:**

Figure 5-13 shows the lag (seconds) in video feed over a course of about 3500 frames.

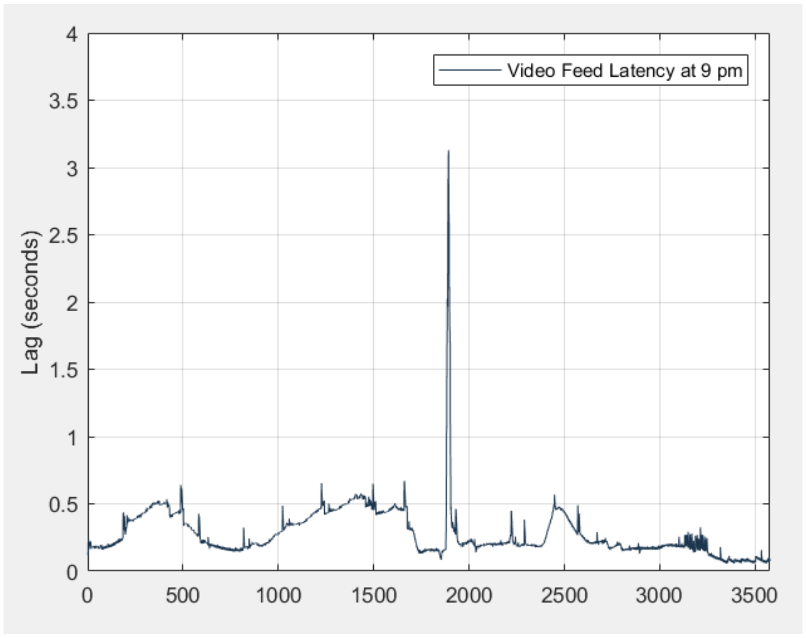


Figure 5-13: Lag (seconds) in Video Feed at 9 pm

A box plot for the same data can be seen in Figure 5-14.

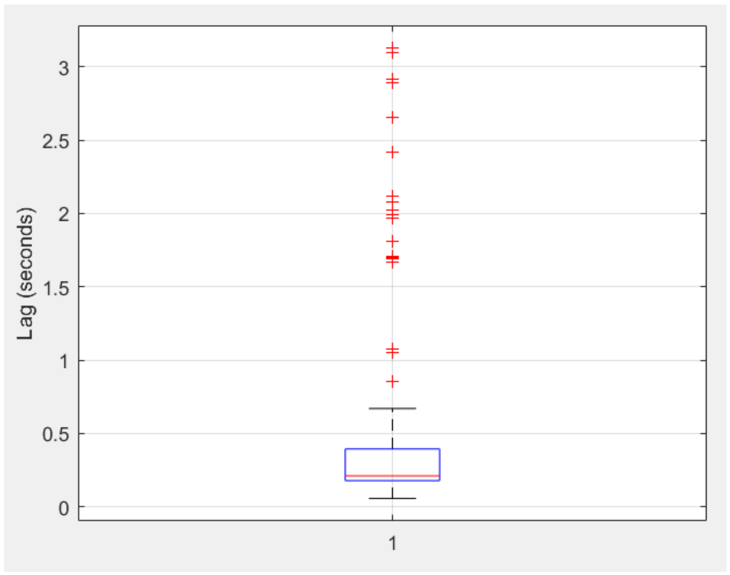


Figure 5-14: Box plot for Video Feed latency at 9 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.280\ seconds = 280\ ms$$

$$Standard\ Deviation = 0.197\ seconds = 197\ ms$$

- **Variability in Latency in Video Feed over the course of the day**

  To better visualize the variability in video feed latency over the course of the day, all the latency plots are plotted on the same graph and shown in Figure 5-15.

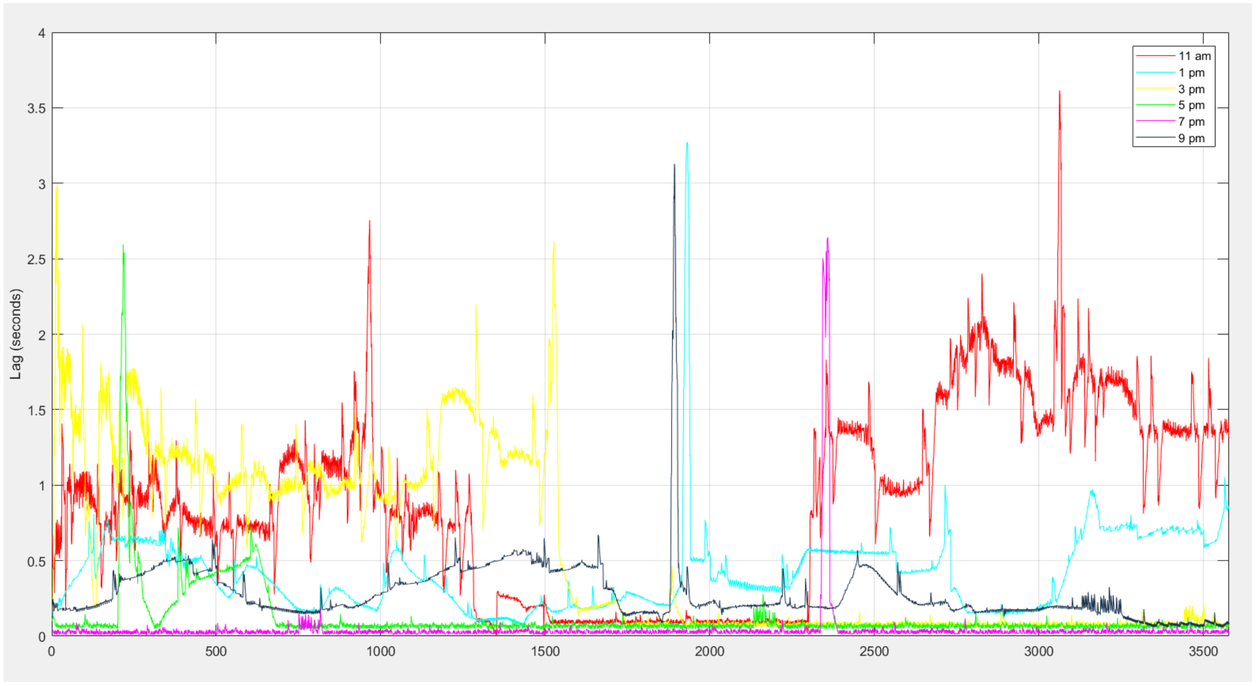


Figure 5-15: Variability in Video Feed latency over the day

Similarly, the box plots can also be plotted on the same figure for better comparison.

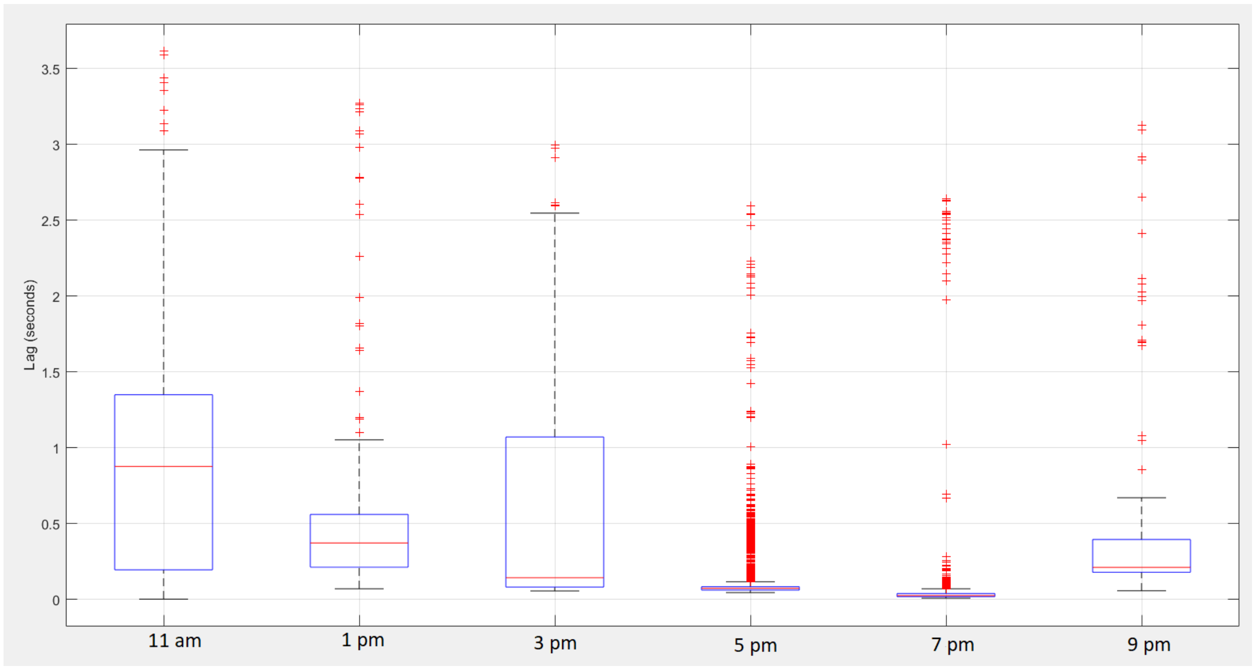


Figure 5-16: Box plot for latencies in Video Feed at different times of the day

As expected, the latency in video feed is much higher at times when network traffic is high (i.e. during work hours). As soon as the work hours finished, the latency in video feed dropped considerably. Unfortunately, our system has no control over the network traffic, or the bandwidth allocation provided by the network operator.

5.2.1.2 LIDAR data Latency

The LIDAR data is being published at a rate of about 5 Hz. The latency was measured by comparing the time stamp a particular message was published with the time stamp it was received at.

The latency tests were performed at different times of the day to account for the variability in network traffic during different times of the day. The tests were carried out on a Wednesday at 11 am, 1 pm, 3 pm 5 pm, 7 pm and 9 pm.

- **Latency in LIDAR data feed at 11 am**

  Figure 5-17 shows the latency in LIDAR data feed at 11 am.

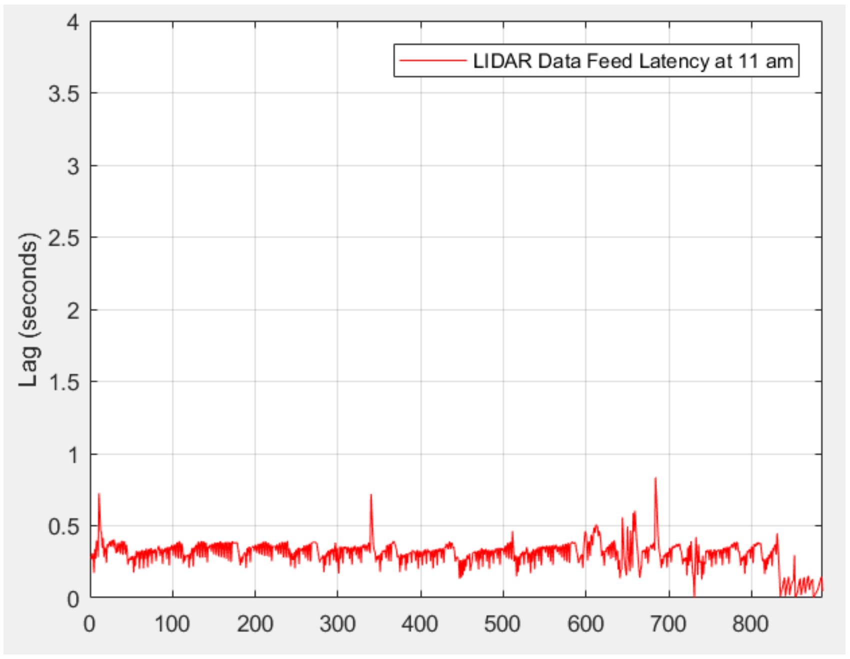


Figure 5-17: Lag (seconds) in LIDAR data feed at 11 am

A box plot for the same data can be seen in Figure 5-18.

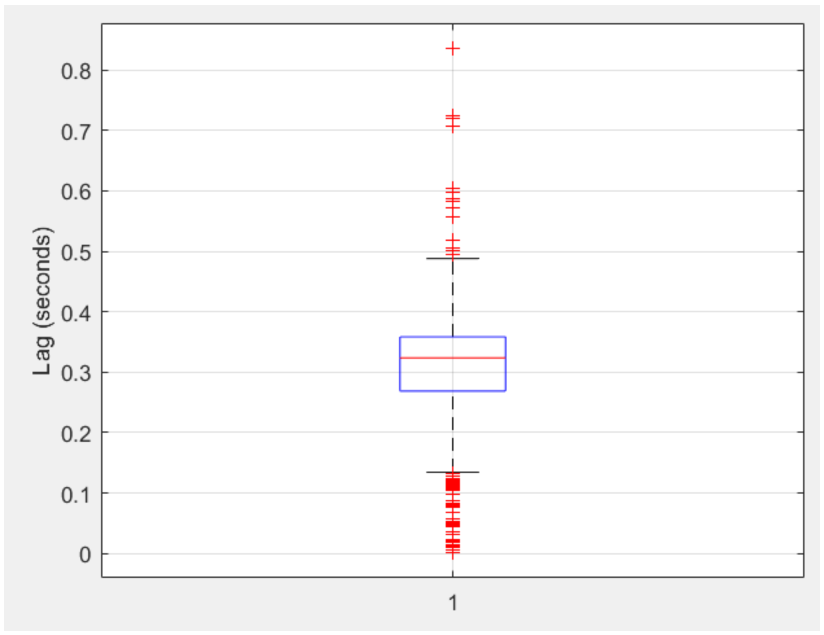


Figure 5-18: Box plot for latency in LIDAR data feed

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.307\ seconds = 307\ ms$$

$$Standard\ Deviation = 0.091\ seconds = 91\ ms$$

- **Latency in LIDAR data feed at 1 pm**

Figure 5-19 shows the latency in LIDAR data feed at 1 pm.

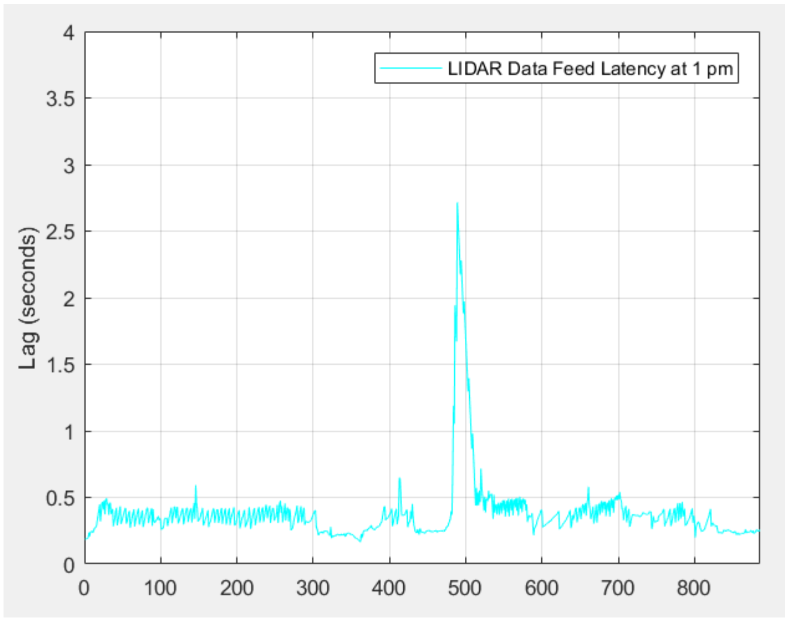


Figure 5-19: Lag (seconds) in LIDAR data feed

A box plot for the same data can be seen in Figure 5-20.

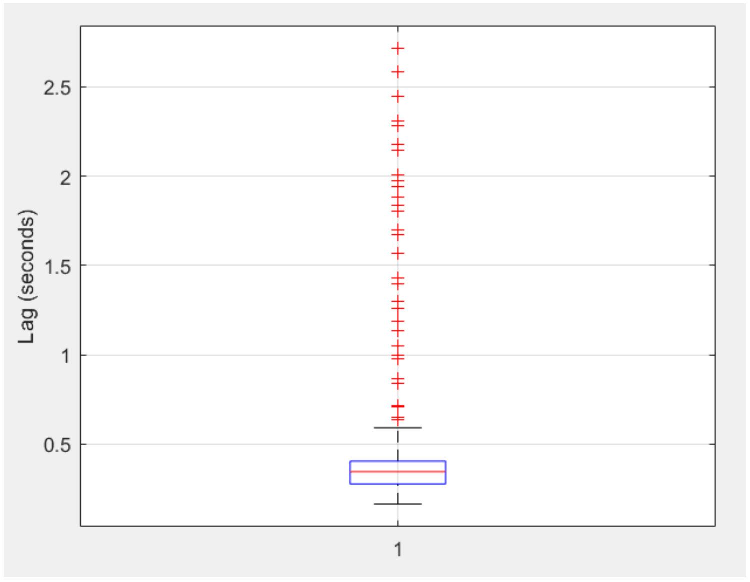


Figure 5-20: Box plot for latency in LIDAR data feed

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.384\ seconds = 384\ ms$$

$$Standard\ Deviation = 0.263\ seconds = 262\ ms$$

- **Latency in LIDAR data feed at 3 pm**

Figure 5-21 shows the latency in LIDAR data feed at 3 pm.

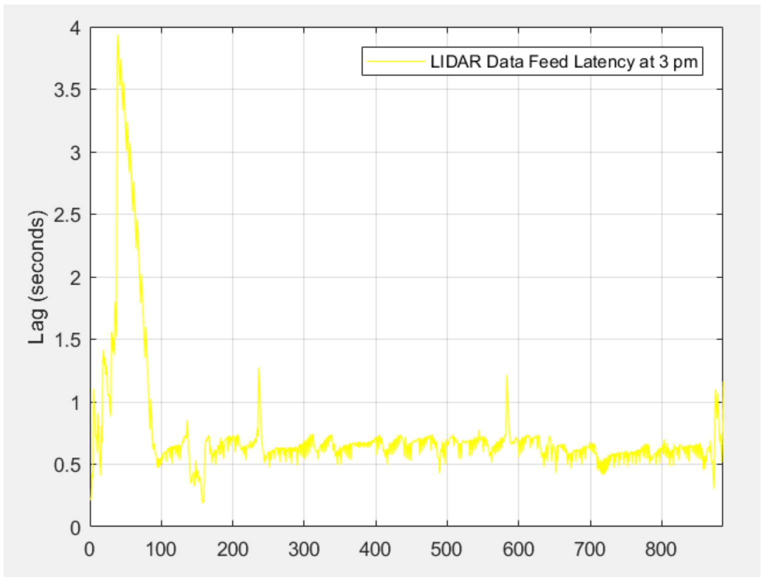


Figure 5-21: Lag (seconds) in LIDAR data feed

A box plot for the same data can be seen in Figure 5-22.

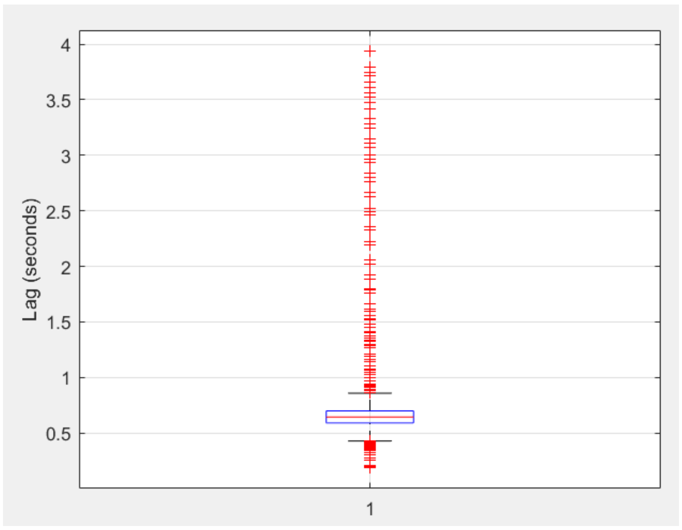


Figure 5-22: Box plot for latency in LIDAR data feed

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.748\ seconds = 748\ ms$$

$$Standard\ Deviation = 0.490\ seconds = 490\ ms$$

- **Latency in LIDAR data feed at 5 pm**

Figure 5-23 shows the latency in LIDAR data feed at 5 pm.

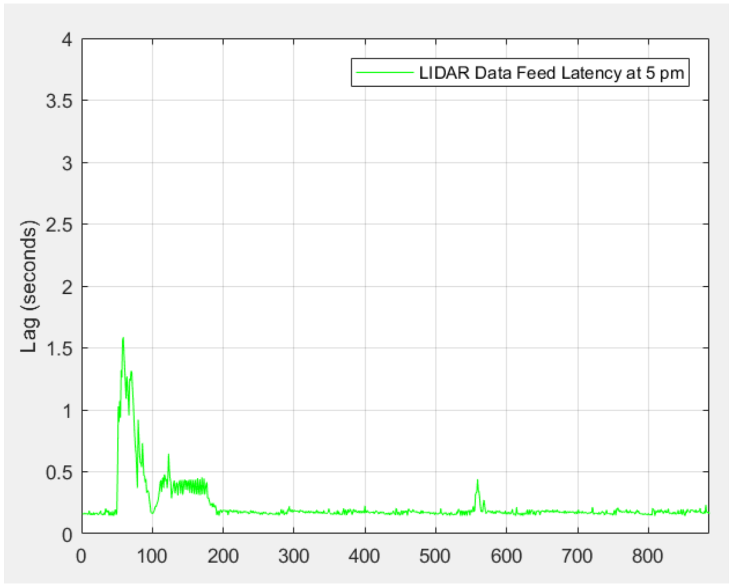


Figure 5-23: Lag (seconds) in LIDAR data feed

A box plot for the same data can be seen in Figure 5-24.

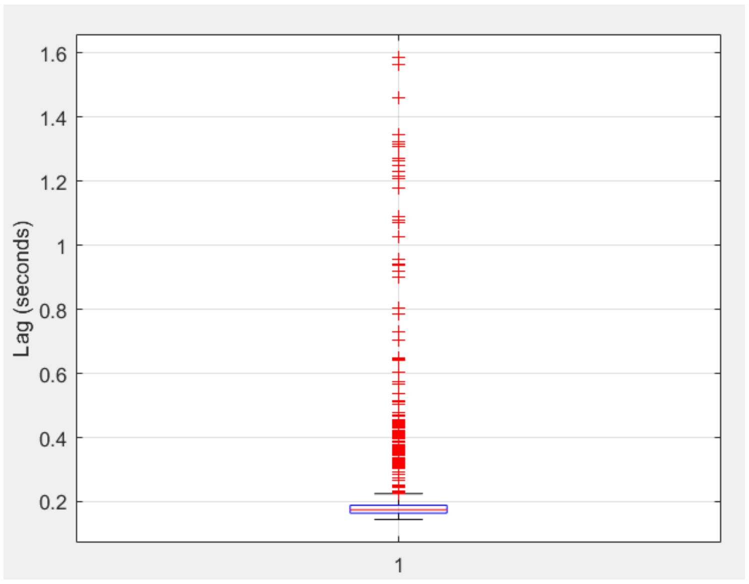


Figure 5-24: Box plot for latency in LIDAR data feed

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.228\ seconds = 228\ ms$$

$$Standard\ Deviation = 0.185\ seconds = 185\ ms$$

- **Latency in LIDAR data feed at 7 pm**

Figure 5-25 shows the latency in LIDAR data feed at 7 pm.

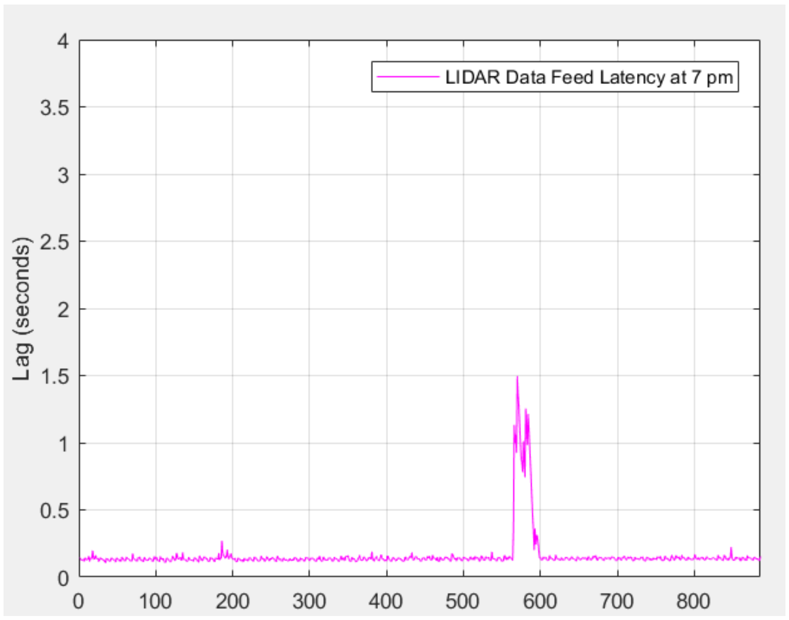


Figure 5-25: Lag (seconds) in LIDAR data feed

A box plot for the same data can be seen in Figure 5-26.

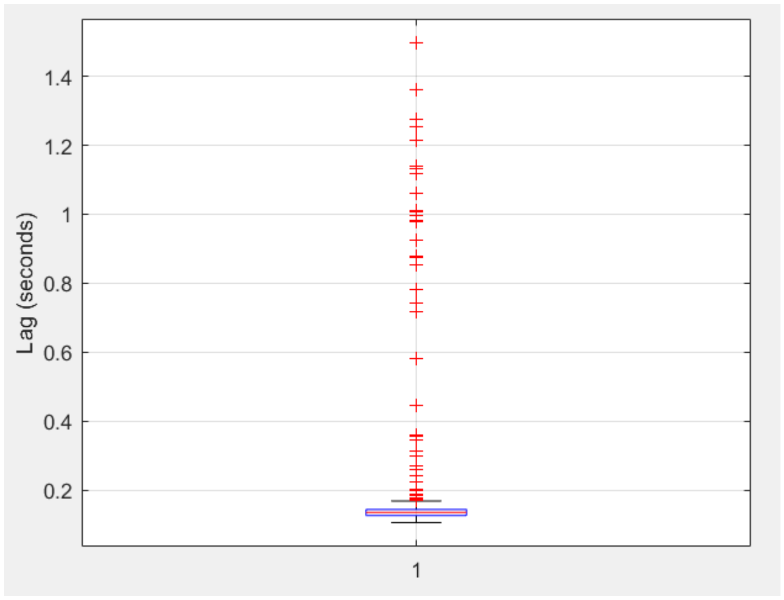


Figure 5-26: Box plot for latency in LIDAR data feed

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.161\ seconds = 161\ ms$$

$$Standard\ Deviation = 0.148\ seconds = 148\ ms$$

- **Latency in LIDAR data feed at 9 pm**

Figure 5-27 shows the latency in LIDAR data feed at 9 pm.

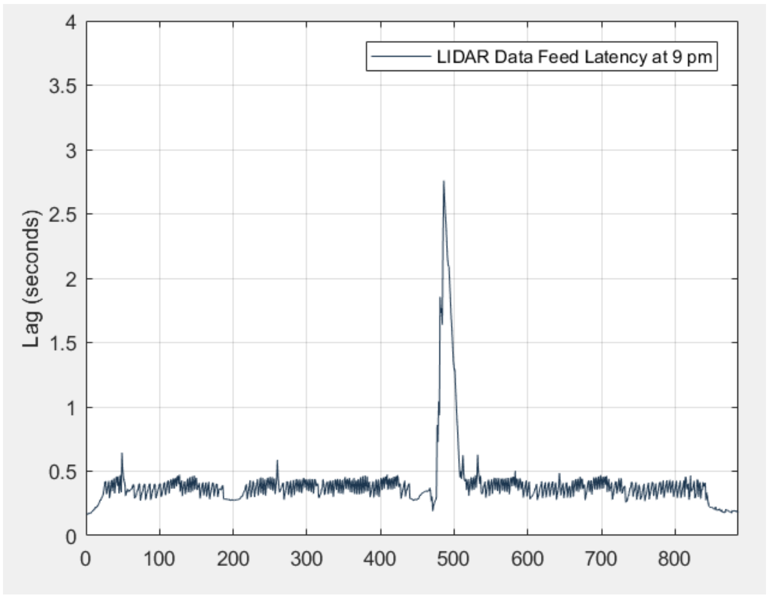


Figure 5-27: Lag (seconds) in LIDAR data feed

A box plot for the same data can be seen in Figure 5-28.

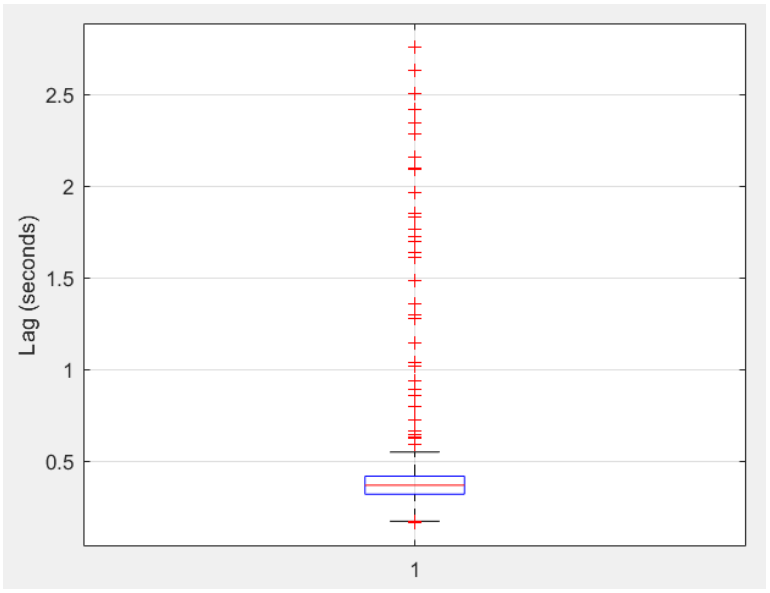


Figure 5-28: Box plot for latency in LIDAR data feed

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.403\ seconds = 403\ ms$$

$$Standard\ Deviation = 0.264\ seconds = 264\ ms$$

- **Variability in Latency in LIDAR Data Feed over the course of the day**

To better visualize the variability in LIDAR data feed latency over the course of the day, all the latency plots are plotted on the same graph and shown in Figure 5-29.

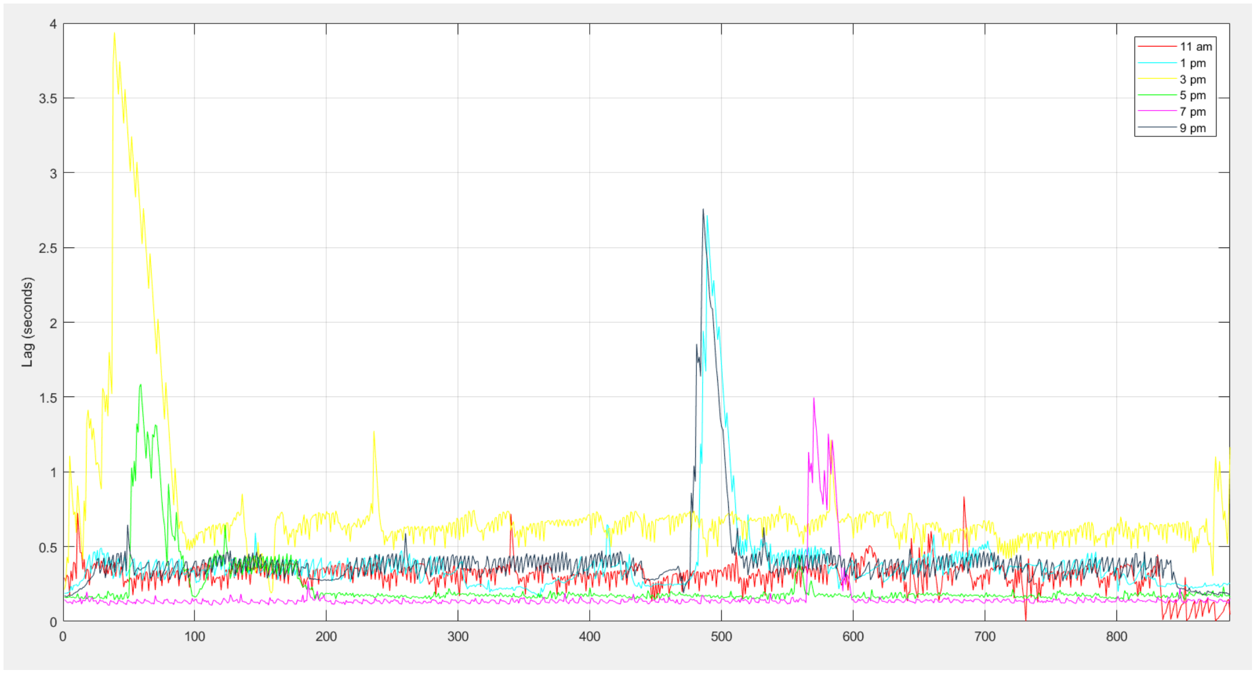


Figure 5-29: Variability in LIDAR Data Feed latency over the day

Similarly, the box plots can also be plotted on the same figure for better comparison.

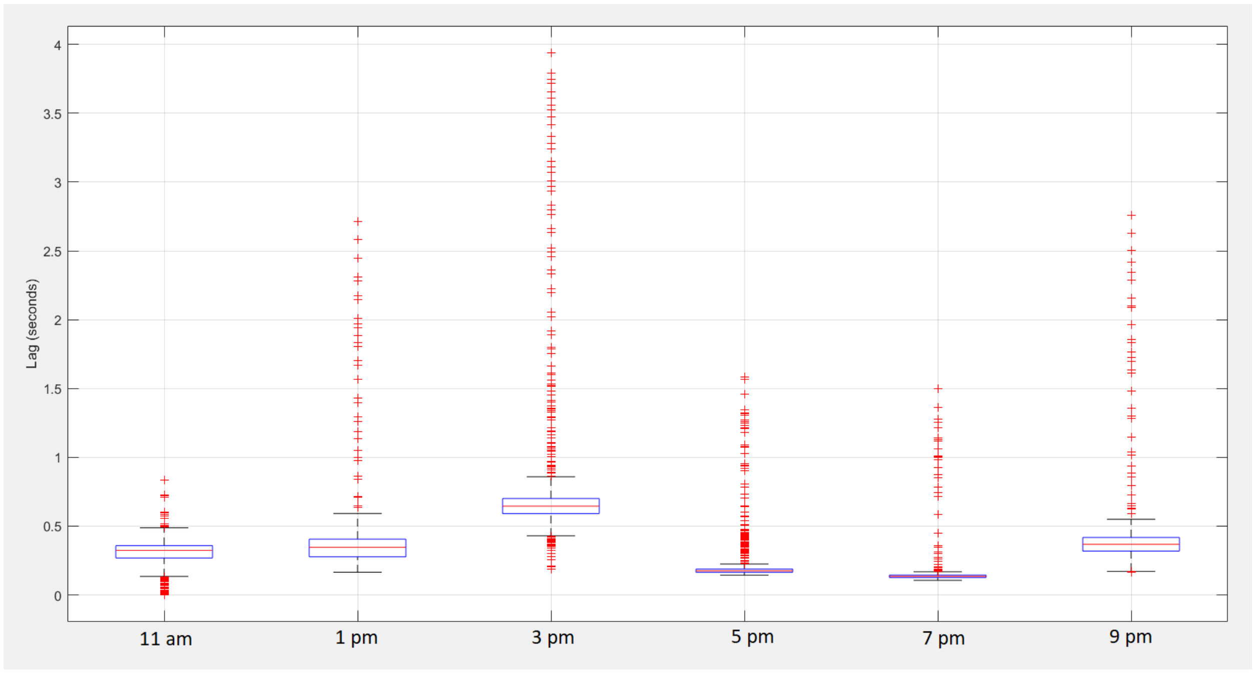


Figure 5-30: Box plot for latencies in LIDAR Data Feed at different times of the day

As expected, the latency in LIDAR data feed is much higher at times when network traffic is high (i.e. during work hours). As soon as the work hours finished, the latency dropped considerably. Unfortunately, our system has no control over the network traffic, or the bandwidth allocation provided by the network operator.

5.2.2 Delay in Communication from Control Station to Remote Vehicle

The latency in transmission of control signals from the control station to the remote vehicle was measured by comparing the time stamp a particular message was published with the time stamp it was received at.

The latency tests were performed at different times of the day to account for the variability in network traffic during different times of the day. The tests were carried out on a Wednesday at 11 am, 1 pm, 3 pm 5 pm, 7 pm and 9 pm.

- **Control Station data feed Latency at 11 am:**

  Figure 5-31 shows the lag (seconds) in control station data feed at 11 am.

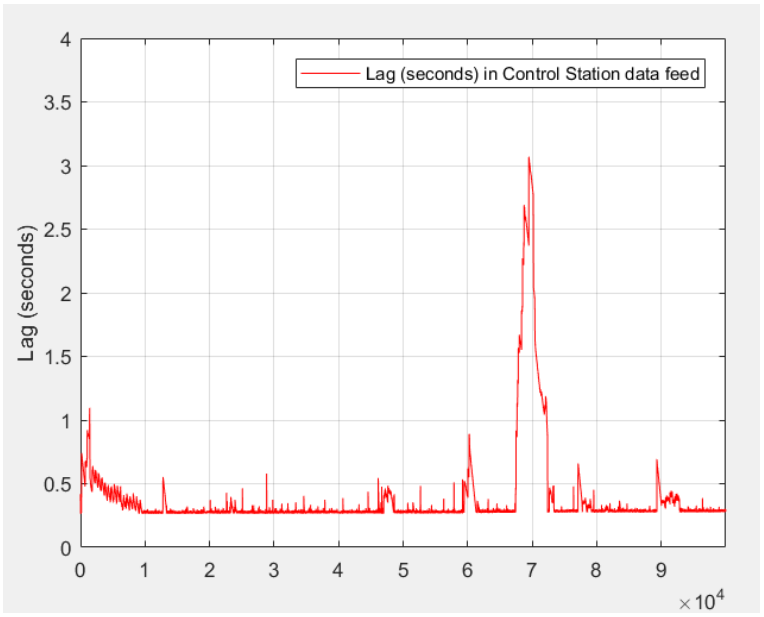


Figure 5-31: Lag (seconds) in Control Station data feed

A box plot for the same data can be seen in Figure 5-32.

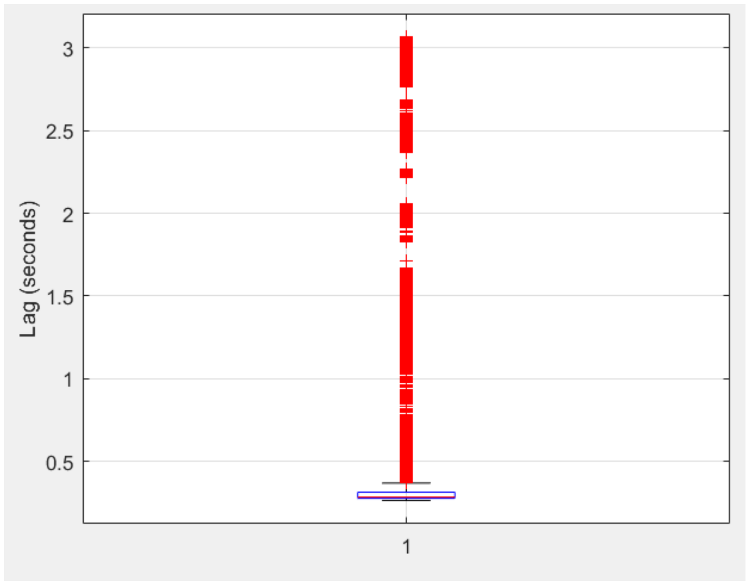


Figure 5-32: Box plot for latency in Control Station data feed

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.388\ seconds = 388\ ms$$

$$Standard\ Deviation = 0.363\ seconds = 363\ ms$$

- **Control Station data feed Latency at 1 pm:**

Figure 5-33 shows the lag (seconds) in control station data feed at 1 pm.

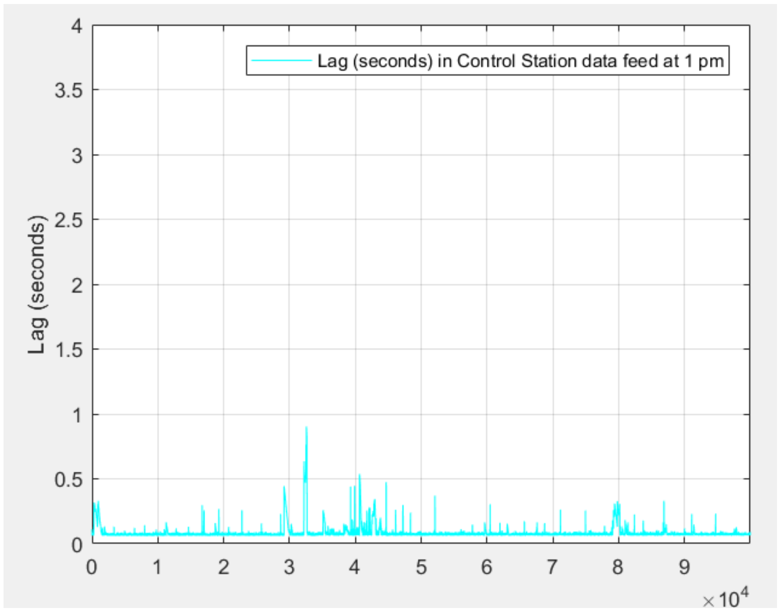


Figure 5-33: Lag (seconds) in Control Station data feed at 1 pm

A box plot for the same data can be seen in Figure 5-34.

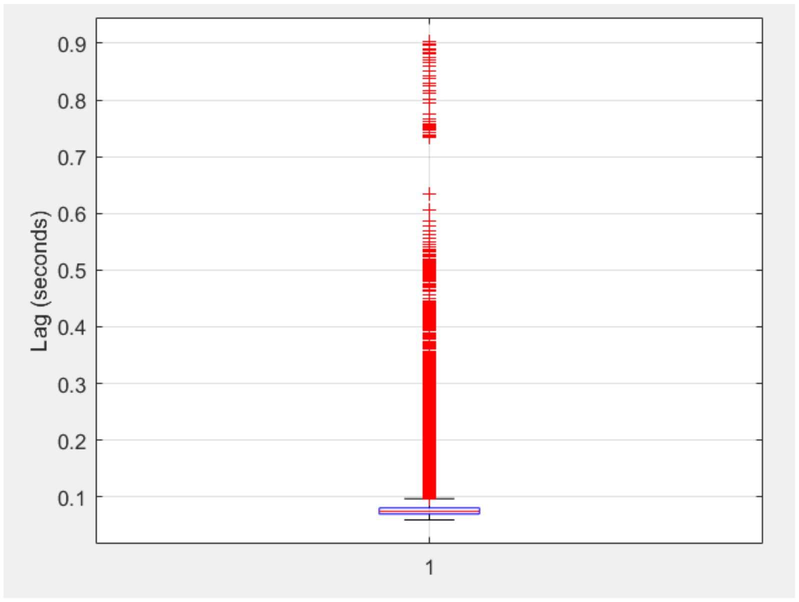


Figure 5-34: Box plot for latency in Control Station data feed at 1 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.087\ seconds = 87\ ms$$

$$Standard\ Deviation = 0.056\ seconds = 56\ ms$$

- **Control Station data feed Latency at 3 pm:**

Figure 5-35 shows the lag (seconds) in control station data feed at 3 pm.

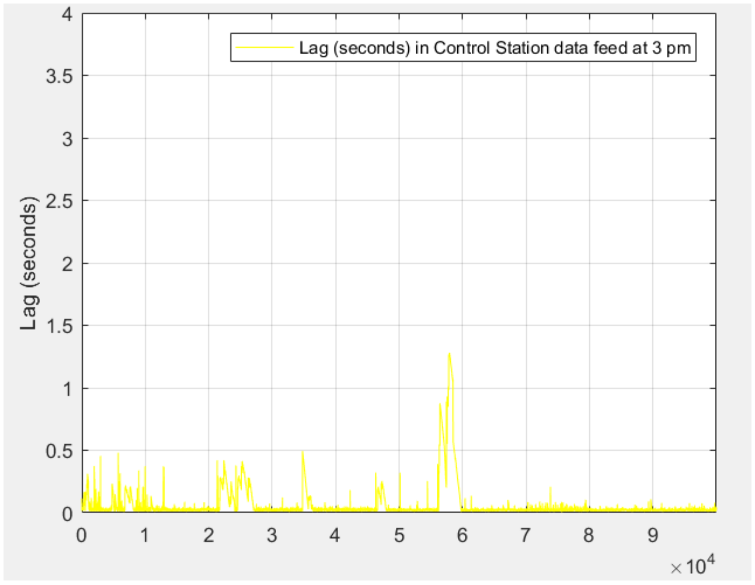


Figure 5-35: Lag (seconds) in Control Station data feed at 3 pm

A box plot for the same data can be seen in Figure 5-36.

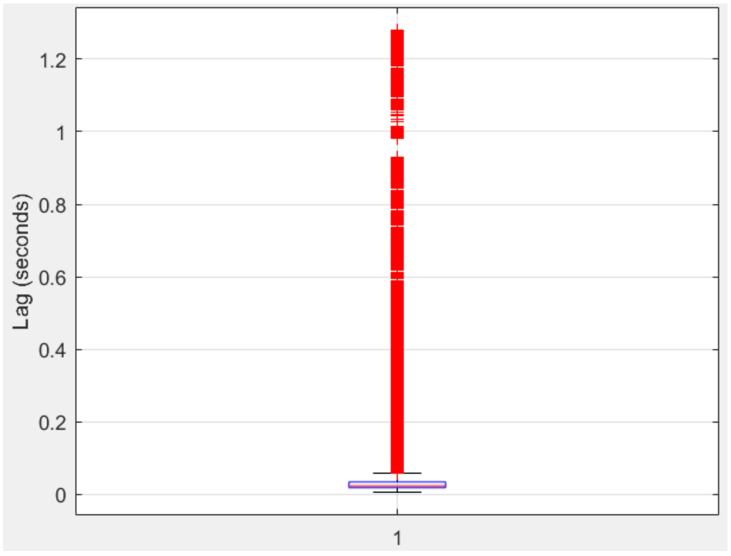


Figure 5-36: Box plot for latency in Control Station data feed at 3 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.064\ seconds = 64\ ms$$

$$Standard\ Deviation = 0.139\ seconds = 139\ ms$$

- **Control Station data feed Latency at 5 pm:**

Figure 5-37 shows the lag (seconds) in control station data feed at 5 pm.

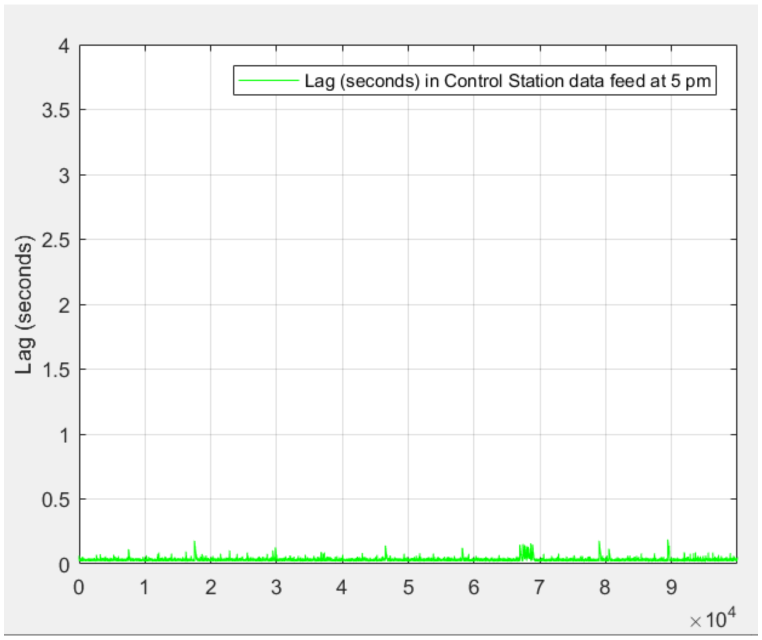


Figure 5-37: Lag (seconds) in Control Station data feed at 5 pm

A box plot for the same data can be seen in Figure 5-38.

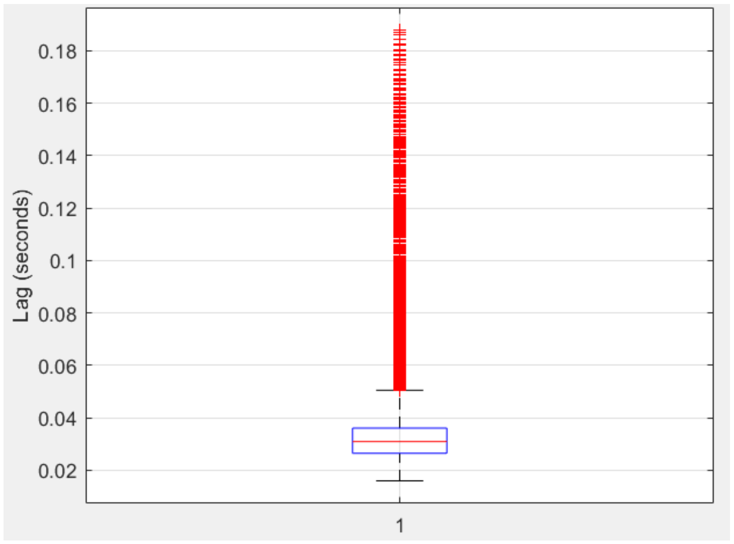


Figure 5-38: Box plot for latency in Control Station data feed at 5 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.034\ seconds = 34\ ms$$

$$Standard\ Deviation = 0.015\ seconds = 15\ ms$$

- **Control Station data feed Latency at 7 pm:**

Figure 5-39 shows the lag (seconds) in control station data feed at 7 pm.

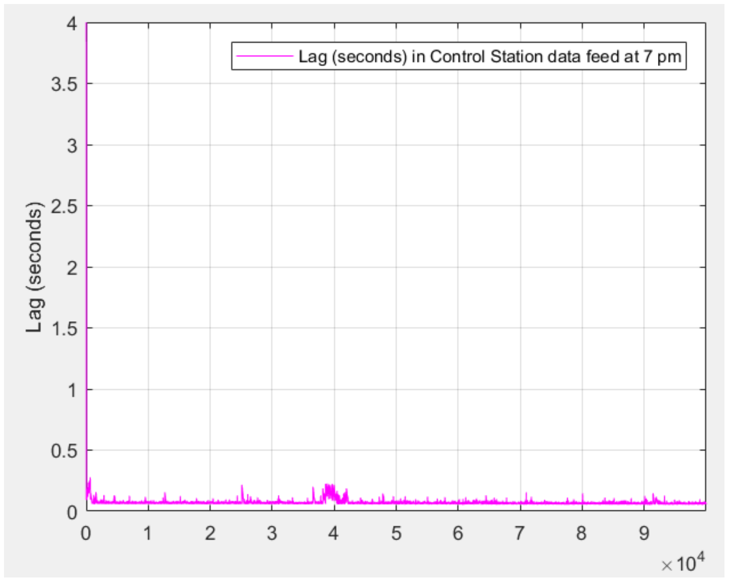


Figure 5-39: Lag (seconds) in Control Station data feed at 7 pm

A box plot for the same data can be seen in Figure 5-40.

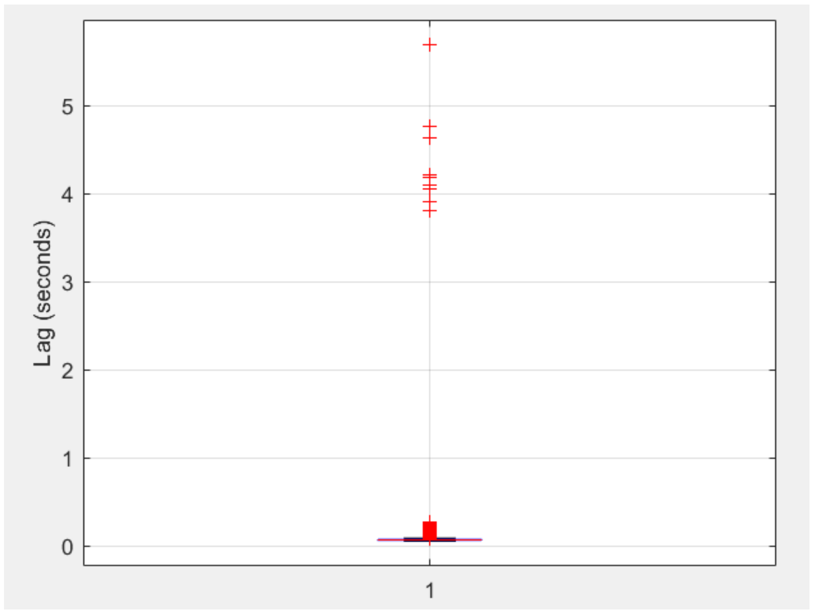


Figure 5-40: Box plot for latency in Control Station data feed at 7 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.073\ seconds = 73\ ms$$

$$Standard\ Deviation = 0.075\ seconds = 75\ ms$$

- **Control Station data feed Latency at 9 pm:**

  Figure 5-41 shows the lag (seconds) in control station data feed at 9 pm.

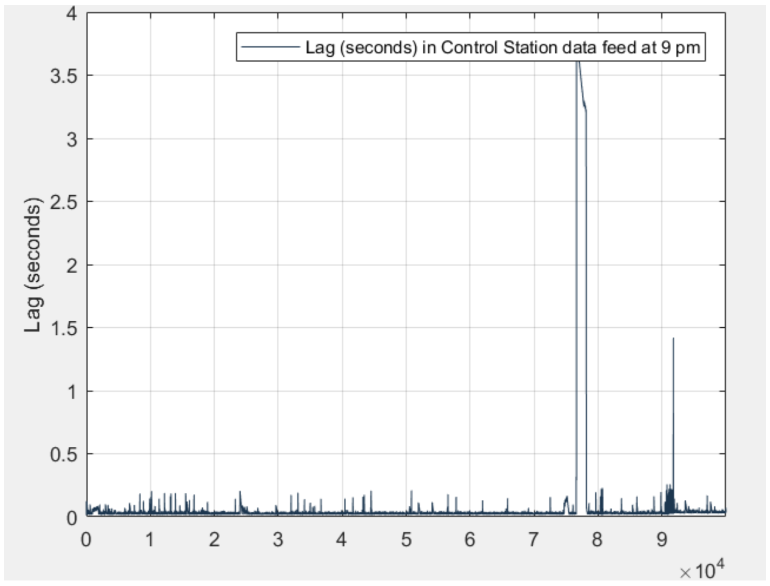


Figure 5-41: Lag (seconds) in Control Station data feed at 9 pm

A box plot for the same date can be seen in Figure 5-42.

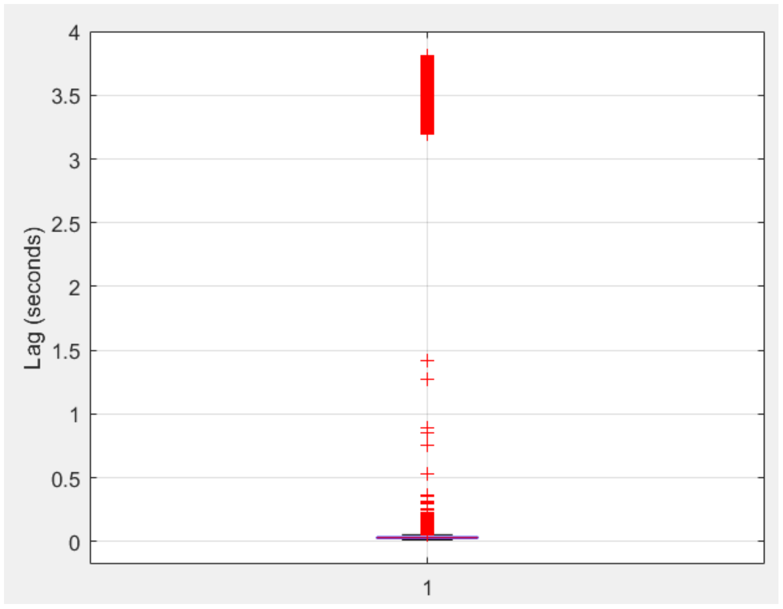


Figure 5-42: Box plot for latency in Control Station data feed at 9 pm

The average delay and the standard deviation are as follows:

$$Average\ Delay = 0.086\ seconds = 86\ ms$$

$$Standard\ Deviation = 0.418\ seconds = 418\ ms$$

- **Variability in Latency in Control Station Data Feed over the course of the day**

  To better visualize the variability in Control Station data feed latency over the course of the day, all the latency plots are plotted on the same graph and shown in Figure 5-43.

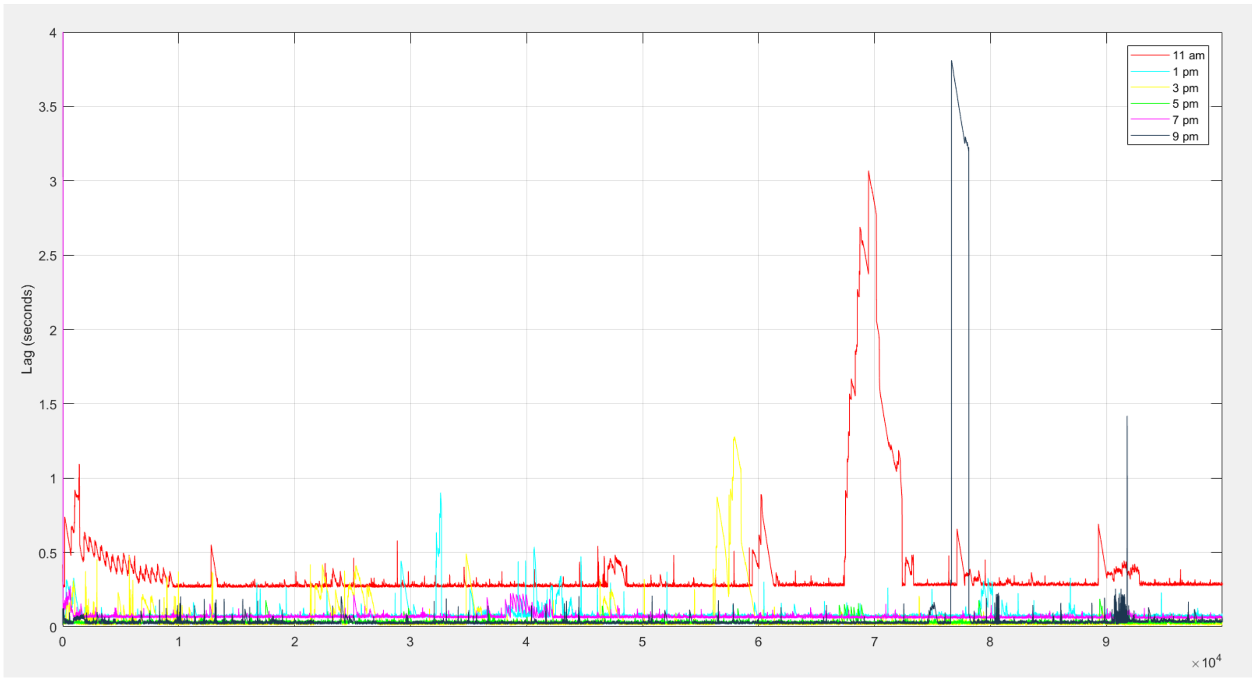


Figure 5-43: Variability in Control Station data feed latency over the day

Similarly, the box plots can also be plotted on the same figure for better comparison.

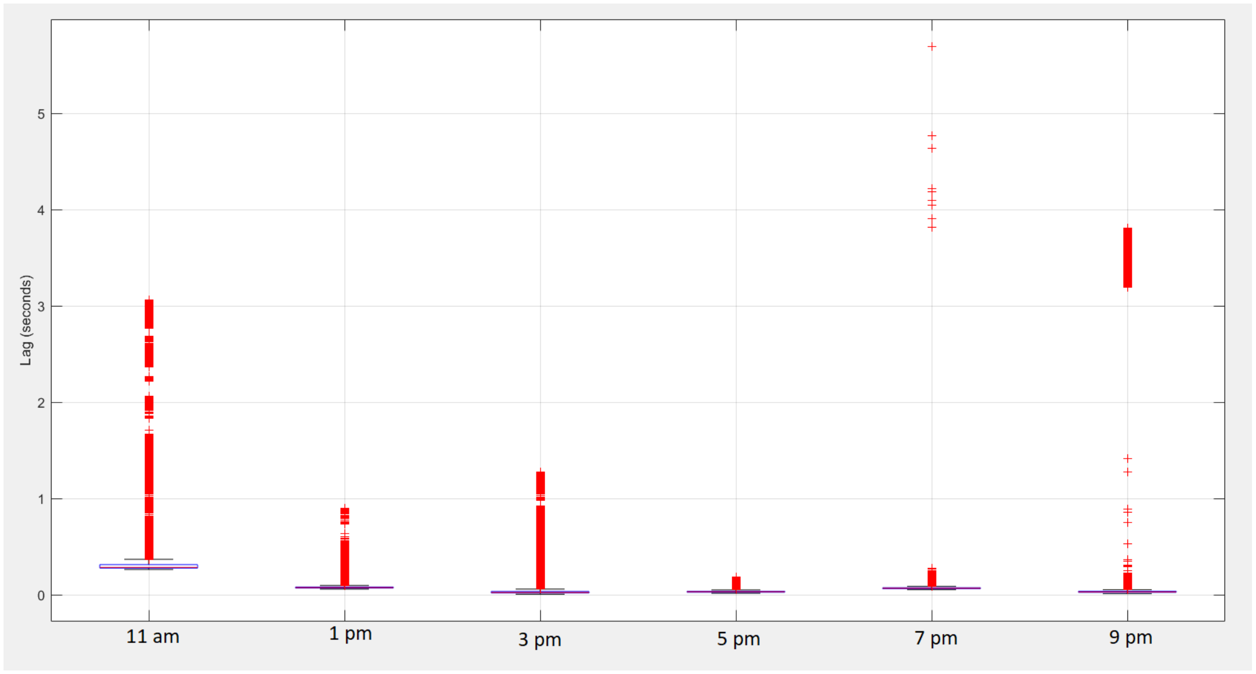


Figure 5-44: Box plot for latencies in Control Station data feed at different times of the day

As expected, the latency in LIDAR data feed is much higher at times when network traffic is high (i.e. during work hours). As soon as the work hours finished, the latency dropped considerably. Unfortunately, our system has no control over the network traffic, or the bandwidth allocation provided by the network operator.

5.2.3 Effects on Teleoperation during Periods of Extended Communication Delay (> 1 seconds)

From the plots and the data given in sections 5.2.1 and 5.2.2, we are aware that the latency in two-way communication is usually not more than 0.5 seconds, especially during the off-peak hours of the day. However, from the individual plots, we can still see that sometimes there are spikes in the graphs which indicate an extended delay in communication. Although less likely, this can happen even during off-peak hours. The effect of such a communication delay on each component of the teleoperated system is enlisted as follows:

- **Effects on Control Station**

    - The video feed that the operator is watching gets stuck. The video feed resumes as soon as the latest video frame is received from the remote vehicle.
    - Similarly, the LIDAR data feed gets stuck. The feed resumes as soon as the new data is received from the remote vehicle.
- **Effects on Remote Vehicle**
    - The remote vehicle stops receiving the control signals from the control station. It keeps on maintaining the values stored in the last successfully received control signal message. However, if an obstacle comes in front of the vehicle, the safety system gets activated and the vehicle stops.

5.3 Summary

In this chapter, a series of tests were conducted to validate the working of the system. The tasks were:

- Forward trajectory task
- Reverse trajectory task

The tasks were performed by an experienced driver. The operator was first asked to perform the task while driving in-situ. The operator was then asked to perform the same task while driving the vehicle remotely. This allowed to do a side-by-side comparison of in-situ driving and remote driving.

Data was also collected to quantify the latency in communication. Latency was measured for following data streams.

- Latency in communication from remote vehicle to control station
    - Latency in video feed
    - Latency in LIDAR data feed
- Latency in control signals feed from control station to remote vehicle

The data was collected during different times of the day to account for variability in network conditions at different hours of the day. The data shows that the communication delay is much less during the off-peak hours of the day as compared to the peak work hours.

6 Autonomous Docking

An autonomous docking system was developed for the vehicle so that the vehicle can dock automatically into a charging station. The docking system was built around the Differential GPS system and visual QR code tracking. We will first these two subsystems and then we will look at how they were tied together.

6.1 Differential GPS

A differential global positioning system (DGPS) is an enhancement to the ordinary GPS which provides improved location accuracy. The nominal GPS is accurate to about 15 meters, whereas the DGPS provides an accuracy of up to 1-3 cm.

6.1.1 Basics of DGPS

DGPS is a method of improving the accuracy of the receiver by adding a local reference to augment the information available from the satellites. Differential GPS consists of two units:

i) Stationary Unit: A stationary unit is set up at a known location. The stationary unit then measures the GPS location and compares it with the actual known location and computes the error.

ii) Rover Unit: Rover unit measures its GPS location and then adjusts the value based on the error value calculated by the stationary unit. The basic underlying premise of the DGPS is that any two receivers that are relatively close together will experience similar atmospheric error

This concept can be illustrated using Figure 6-1.

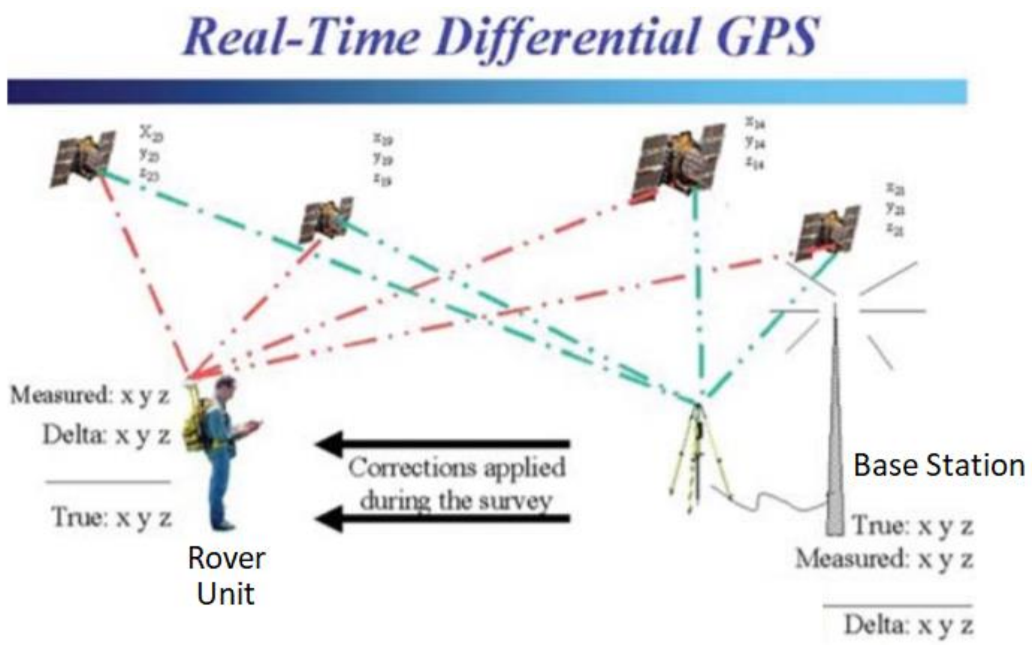


Figure 6-1: Schematic diagram of DGPS unit (Allen, Wolfe, Judy, Haukkala, & James, 2001)

6.1.2 DGPS Used in our System

We use “Ublox C94-M8P” differential global positioning system in our system. The details of how the DGPS was set up can be found at (Bezdek, 2020). A brief description is as follows:

i) Base Station: The base station was fixed at a permanent location. As we did not know the true accurate coordinates of that point, we used the survey-in mode in the Ublox software to determine the true location of the base station. When working in Survey-in mode the Base Station will determine its own position by building a weighted mean of all valid 3D position solutions.

ii) Rover Station: Rover was mounted on the vehicle and was set up so that it could adjust the location of the rover based on the errors calculated by the base station. The base station and the rover communicated with each other using the radio signals.

iii) Data extraction from the Rover: As all the communications of our overall system is based on ROS, we needed to communicate the calculated location of the rover to the ROS network. For that purpose, an Arduino was connected to the rover, which read the location values and published it to the ROS network.

The signal flow can be illustrated using the diagram in Figure 6-2.

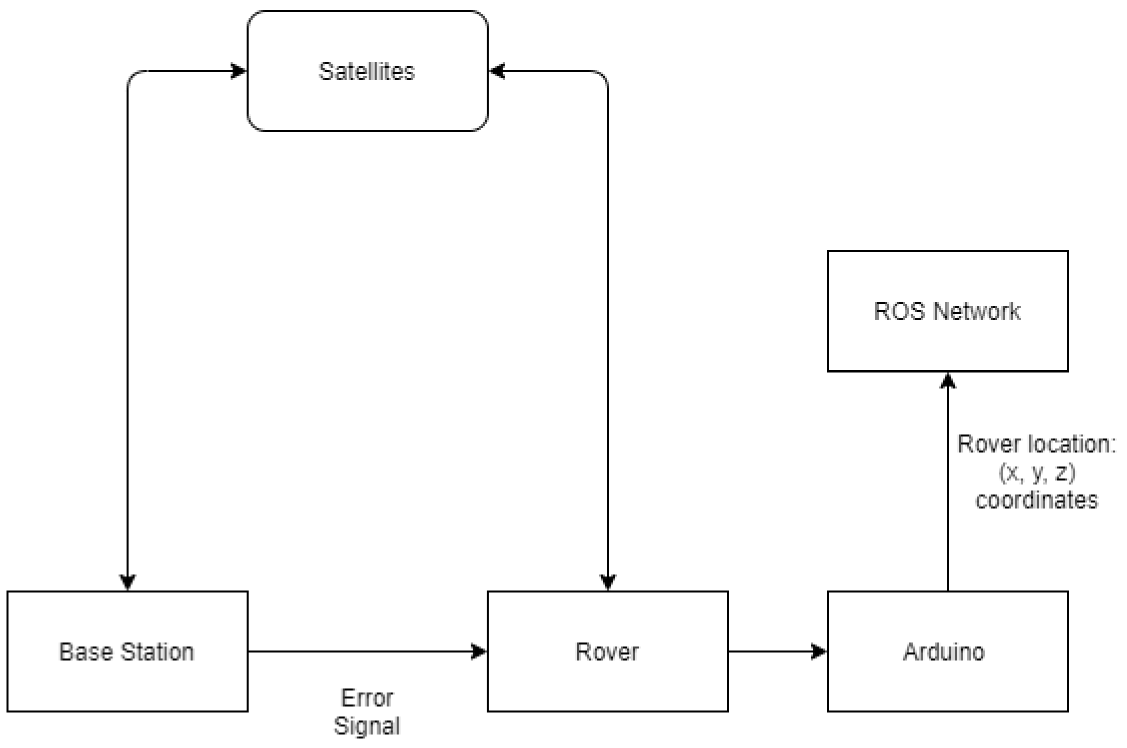


Figure 6-2: Signal Flow schematic in our DGPS

Hence, using the DGPS we were able to know the location of the rover with an accuracy of up to 1-3 cm. This location could be used to move the vehicle to a particular location.

6.2 QR Tracking

6.2.1 Reasons for using QR code as a Visual Marker

To make a system work off visual feedback, we need a visual marker. In real-world scenario, the visual marker would be some feature points of the docking station. However, we are free of any such constraints as we do not have any particular model of charging/docking station that our system needs to detect.

Therefore, QR codes were chosen as visual markers. QR codes have an advantage of easy detection and fast readability. Moreover, there are many libraries that can be used to detect QR codes. By using QR codes as a "visual marker" allowed us to focus on the docking procedure, instead of focusing on the problem of feature detection in images.

6.2.2 Visual Servoing based on QR Code Tracker

A python library called "pyzbar" was used to detect the QR code. The pyzbar library enabled us to read the text encoded by the QR code as well as the location and the size of the QR code box in the image. This enabled us to perform visual servoing to plug the vehicle into the charging port. Our QR code tracking script performed the following functions:

i) Look for the QR code in the video feed.

ii) If a QR code is found, decode it, and read the data. If the QR code reads "Charging Station", it means it is the correct QR code and the vehicle can dock here.

iii) Start reading the position and size of the QR code and publish that information over the ROS network.

iv) The visual servoing controller subscribes to the published information and tries to move the cart in such a way so that the current position and size of the QR code becomes equal to the desired values.

The above process can be represented by the flowchart in Figure 6-3.

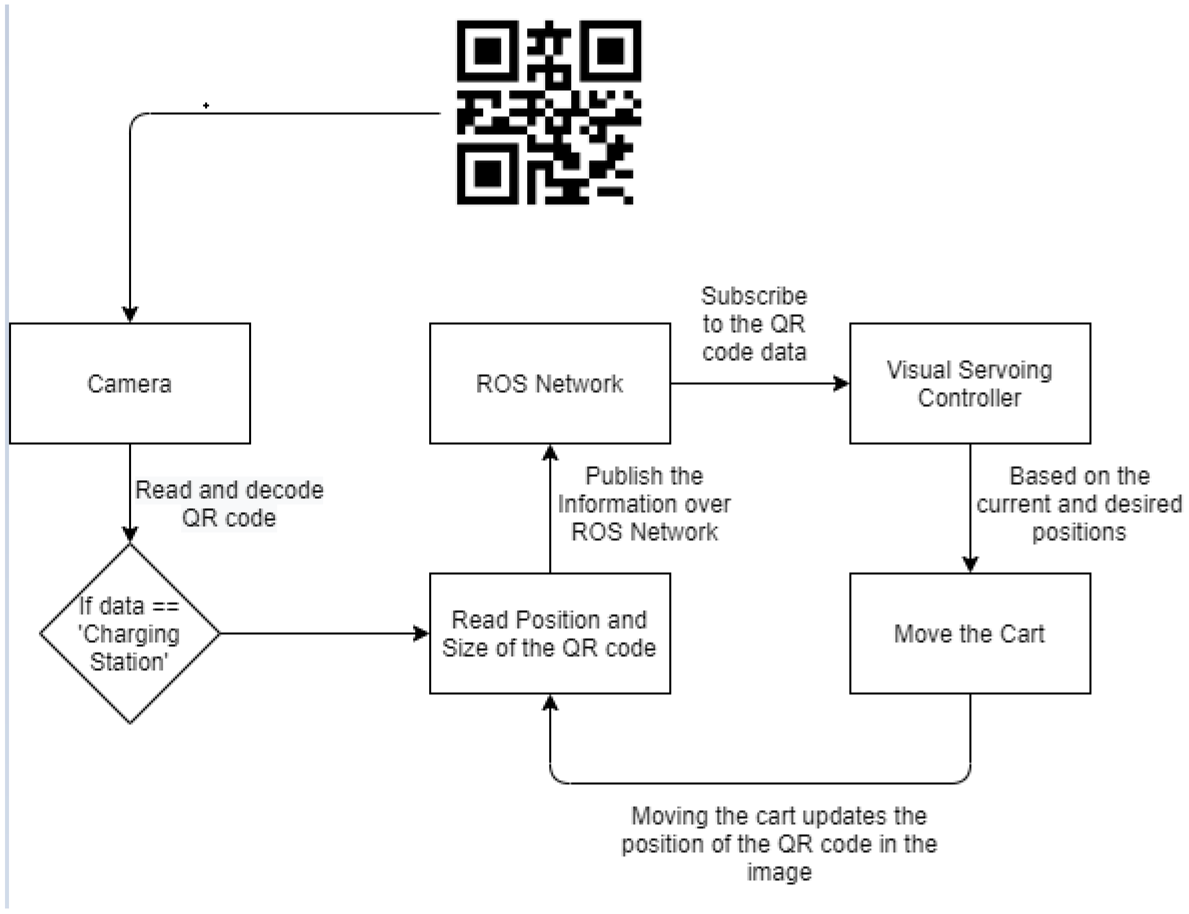

Camera
Read and decode QR code
If data == 'Charging Station'
Read Position and Size of the QR code
Publish the Information over ROS Network
ROS Network
Subscribe to the QR code data
Visual Servoing Controller
Based on the current and desired positions
Move the Cart
Moving the cart updates the position of the QR code in the image

Figure 6-3: Flowchart explaining the QR code tracking process

The docking system was created by tying together the DGPS and the QR tracking subsystems.

6.3 Docking System

The docking system was created by combining the DGPS and the QR tracker together to create an autonomous docking system. The vehicle first moves close to the charging station based on the DGPS. Once, it is close enough, the visual QR tracker takes over and docks the cart into the charging station.

The overall system can be explained by the diagram in Figure 6-4.

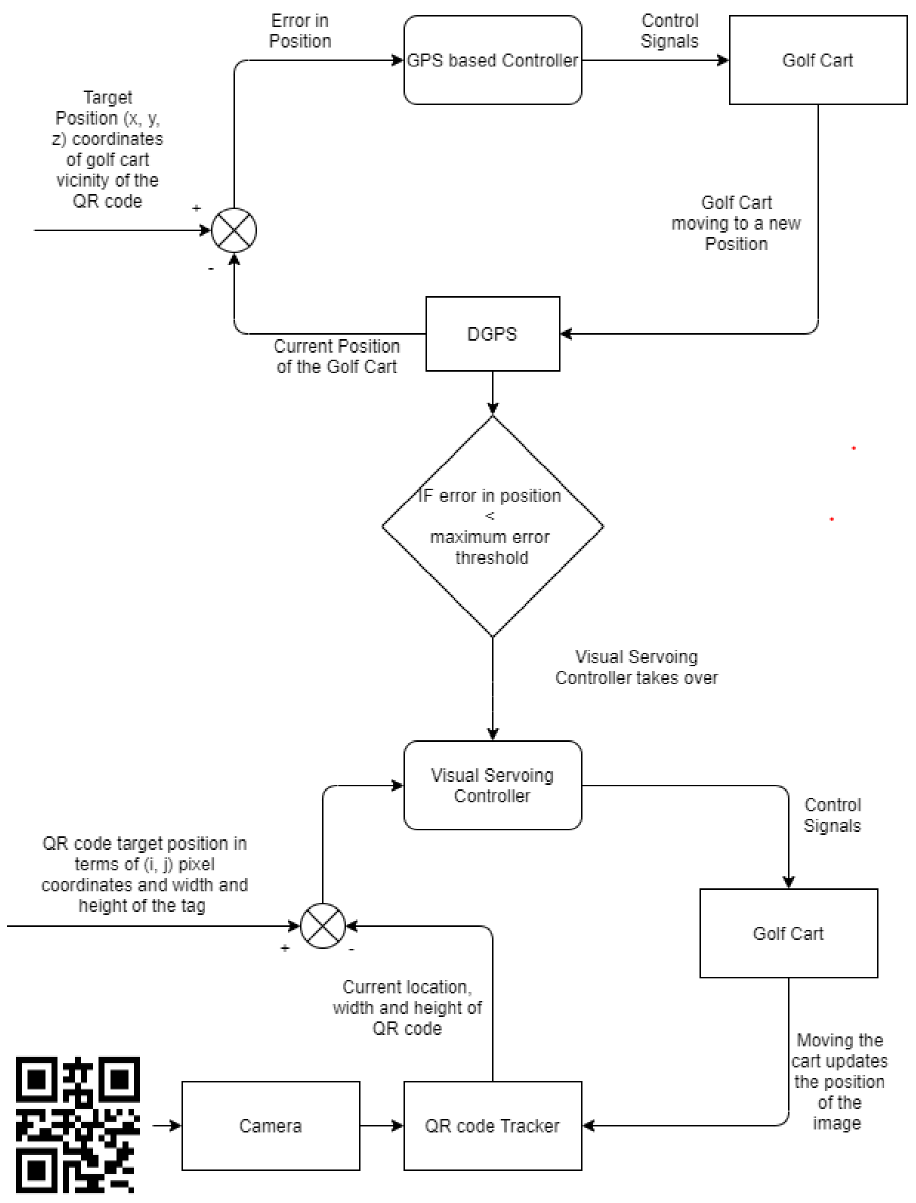

Error in Position
GPS based Controller
Control Signals
Golf Cart
Target Position (x, y, z) coordinates of golf cart vicinity of the QR code
+
-
Golf Cart moving to a new Position
DGPS
Current Position of the Golf Cart
IF error in position < maximum error threshold
Visual Servoing Controller takes over
Visual Servoing Controller
Control Signals
QR code target position in terms of (i, j) pixel coordinates and width and height of the tag
+
-
Golf Cart
Current location, width and height of QR code
Moving the cart updates the position of the image
Camera
QR code Tracker

Figure 6-4: Docking system based on DGPS and QR tracking

6.4 Experimental Setup and Results

A test was conducted to validate the working of the procedure.

The vehicle was to be parked/docked inside the two white lines in the image shown in Figure 6-5.

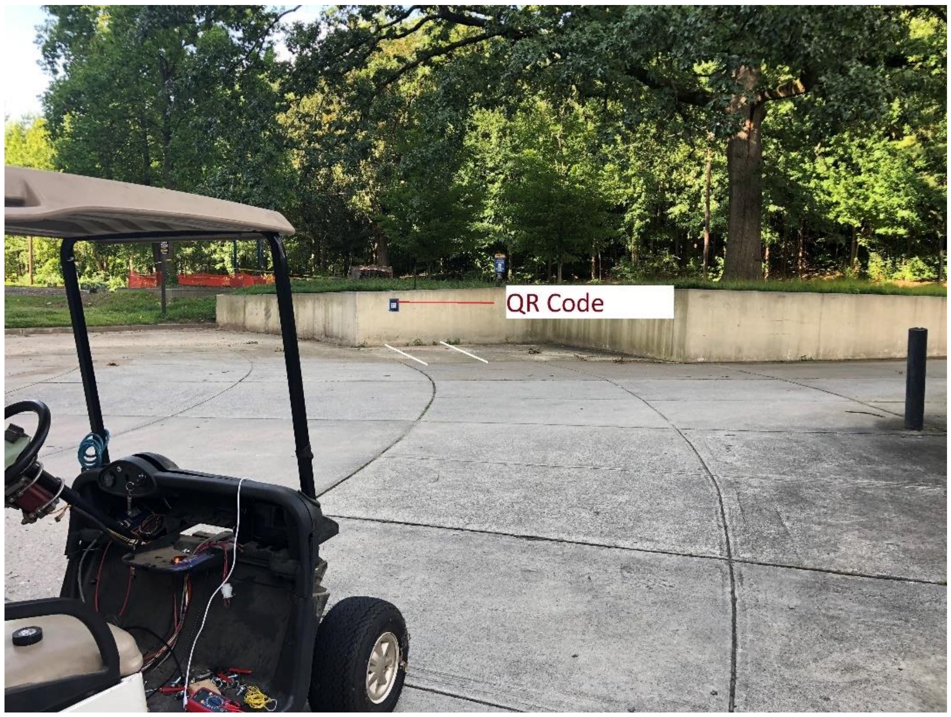


Figure 6-5: Setup to validate the docking procedure

In the above image, the vehicle starts off at a distance from the QR code. It is brought closer to the QR code based on the GPS location. Once the vehicle is close enough to the QR code that the camera can read it, the visual servoing algorithm takes over and brings the vehicle to its final position.

A closer look at the QR code can be seen in Figure 6-6.

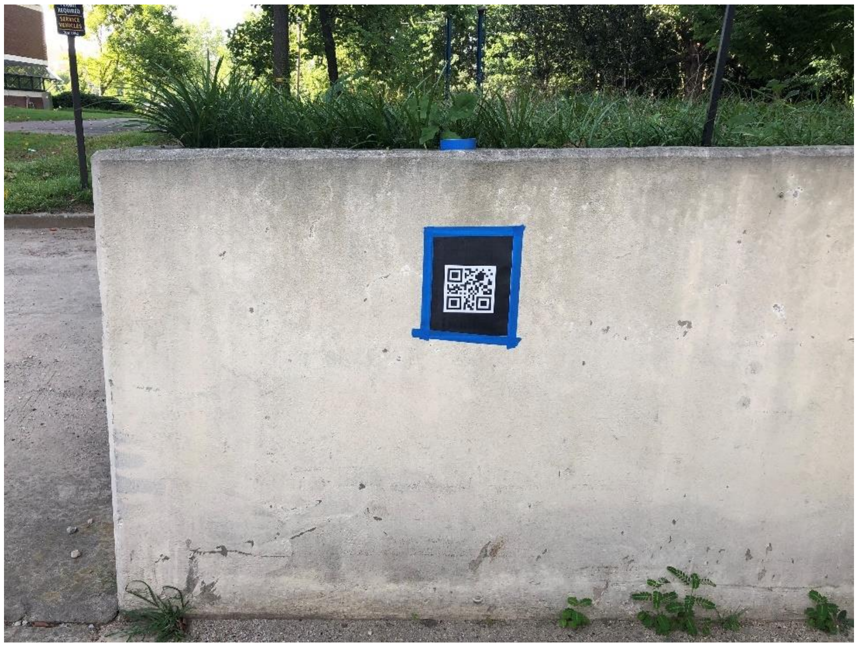


Figure 6-6: QR code mounted on the wall serving as a visual marker

The results were categorized as PASS/FAIL based on the vehicle's final position. If the vehicle was inside the blue lines the test was regarded as PASS, and if the vehicle was outside the lines the test was regarded as a FAIL.

Example of passed and fail tests can be seen in the images in Figure 6-7 and Figure 6-8 respectively.

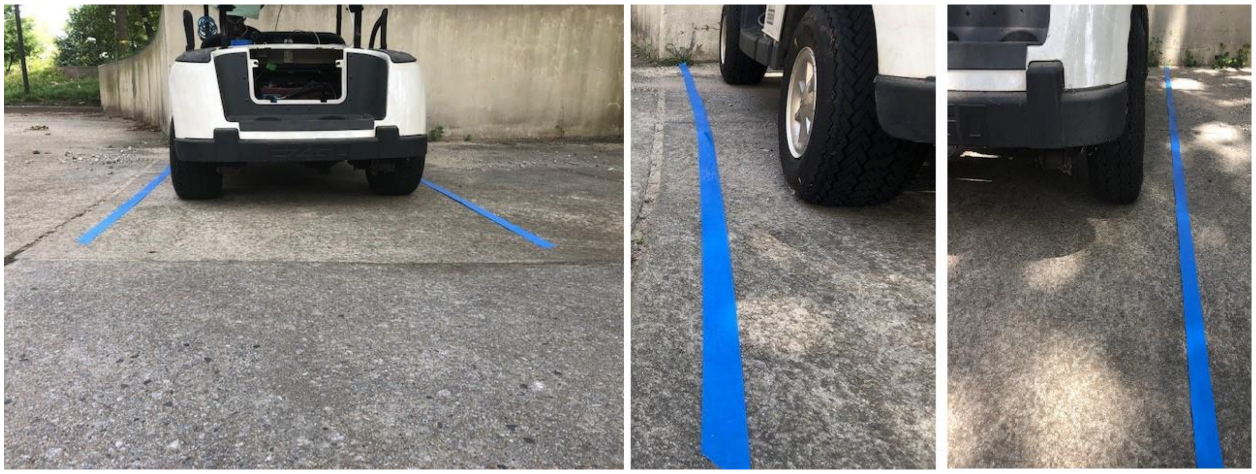


Figure 6-7: Example of a passed test

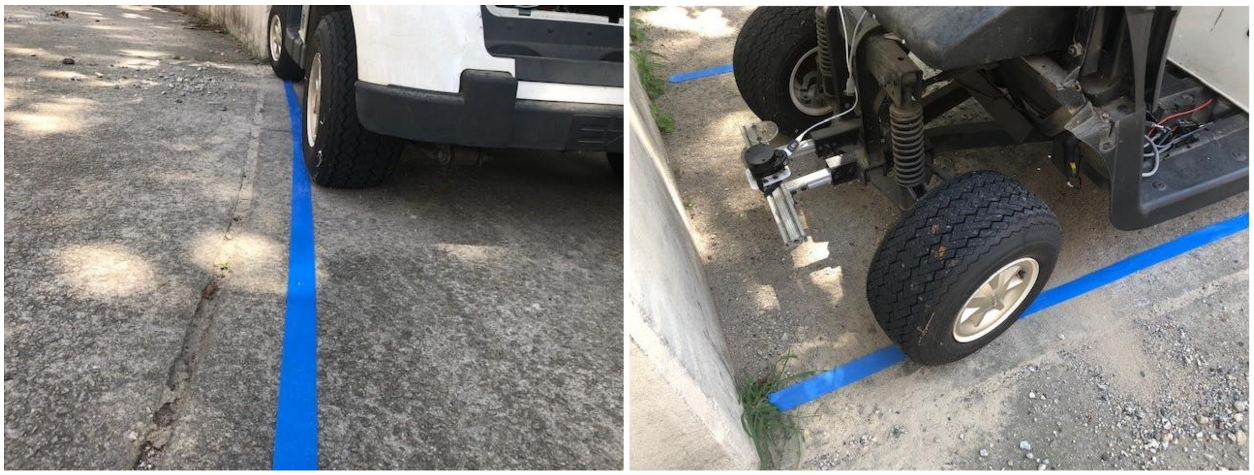


Figure 6-8: Example of a failed test

A total of 10 tests were performed. The results are in Table 6-1.

Table 6-1: Passed and Failed tests for docking procedure

Total Tests	10
Passed Tests	8
Failed Tests	2

As you can see in the above table, the docking system in its current setup is partially successful.

6.4.1 Issues with the Docking Procedure

The docking system described in this chapter have a few performance issues. The issues observed by the author are enlisted below

- The docking system algorithm does not take the dynamics of the vehicle in account. It just works based off the GPS and visual feedback. Hence, most of the times the final orientation of the car is not completely perpendicular to the wall. Dynamics of the vehicle should be incorporated into the system to ensure control over the final orientation.

- The visual servoing algorithm tries to center the QR code in the camera. It requires the vehicle is moved left or right based on the readings from the camera. As the vehicle is unable to achieve purely left or right motion, the final orientation is not completely perpendicular to the wall.
- The visual servoing algorithm works off the 2D camera. Hence, the pose of the QR code in 3D is not calculated which again results in a final orientation that is at some angle with the wall.

Hence, in its current form *the docking system built in this study will not be accurate enough to achieve the accuracy required for plugging the vehicle into a charging port.*

6.4.2 Illustration of Issues with Final Orientation of the Vehicle w.r.t the Wall

In section 6.4.1, it was mentioned that the final orientation of the vehicle was usually at some angle with the wall. In this section, we will illustrate this issue. In Figure 6-9, the angle α can be used to describe the angle offset of the vehicle with respect to the wall.

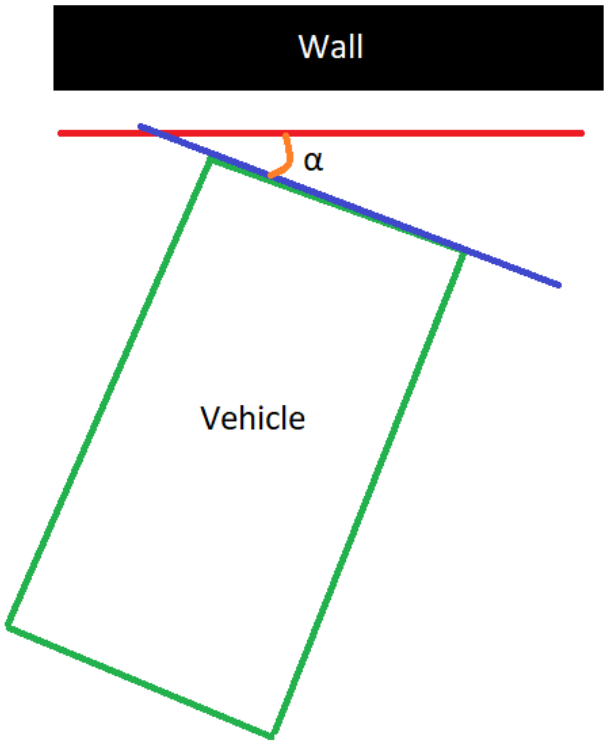


Figure 6-9: Angle α of the vehicle w.r.t wall

The angle, α in our experiments had a value in the range of 2°-20°. In Figure 6-10, we can see the angle α during different runs of the experiment.

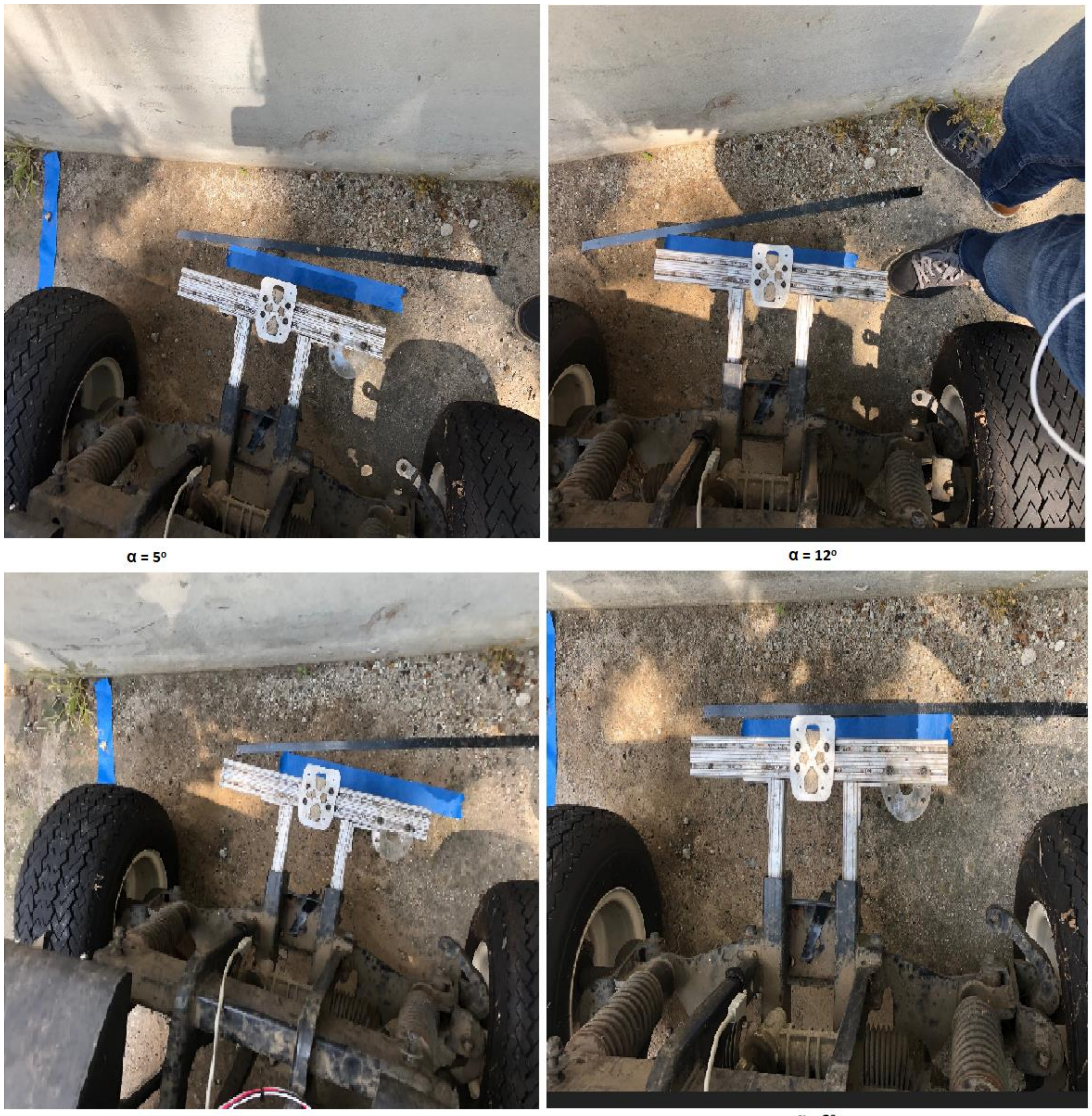


Figure 6-10: Variation in final orientation of vehicle w.r.t. the wall

6.5 Summary

In summary, a docking system was built to autonomously dock the vehicle into a docking station. The system was built using feedback from a differential GPS unit and visual feedback. The vehicle moves closer to the docking station based off the readings from the differential GPS. Once, it gets close enough to be able to read the QR code, the visual servoing algorithm takes over and attempts to dock the vehicle.

The system was tested by creating an experimental setup. The system was partially successful as the procedure enabled the vehicle to be inside the boundary lines of the simulated docking station. However, the final orientation of the vehicle was usually not perpendicular to the wall. As a result, the system in the current state would be not accurate enough to enable the vehicle to plug itself into a charging port.

The system could be vastly improved by incorporating the dynamics of the vehicle, and by using a stereoscopic camera.

7 Conclusions and Future Work

In this work, a cellular teleoperated system built around ROS framework to drive a low speed urban vehicle remotely is built and tested. Safety systems for collision avoidance were also incorporated into the system. Moreover, latest advances in the field of computer vision were employed to increase the situational awareness of the operator.

As a secondary objective, an autonomous docking system was built for a vehicle based on vision and location data. This allowed the vehicle to be charged/parked automatically without requiring a human intervention.

In what is left of this thesis, a summary of the information detailed heretofore will be reviewed along with relevant conclusions pertain to the completed experiments. Finally, future works related to this investigation and how the current systems may be improved are discussed.

7.1 Summary

Teleoperation has grown at huge leaps since its inception spanning across many disciplines and science fields. On top of this, each field has seemingly unbounded space to continue growing. One field that is of particular interest is the use of cellular network as a communication medium between the control station and remote vehicle. Improvements in cellular technology has led to consistently higher data transfer rates and decreased latencies. This has happened to such an extent that it is now possible to stream large amounts of data which in turn allows for exploring new uses such as teleoperation. Moreover, in the last few years, a lot of open-source robotics software platforms have been developed for e.g. ROS. These frameworks allow for rapid development by providing tools and libraries that provide hardware abstraction, message-passing, libraries, visualizers, package management and much more.

Thus, a system was designed and built around the ROS framework and used the 4G LTE network for communication between the control station and remote vehicle. Additionally, to increase the situational awareness of the operator, techniques from computer vision were employed to present the operator with an augmented video feed. Moreover, data from a 2D LIDAR was also processed and presented to the user visually. A safety system based on the data from the LIDAR was also incorporated. It stopped the vehicle in case of an imminent collision. A speed measuring device was also built and mounted on the vehicle.

The system was tested by asking an experienced driver to perform a few tasks. The performance of the driver while driving the vehicle remotely was compared with the in-situ driving. Moreover, data was logged to quantify latency in communication between control station and remote vehicle, and latency in communication between remote vehicle and remote station.

A docking system to autonomously dock/park the vehicle based off the location data from GPS and visual feedback was also built. The system was partially successful. It was usually able to bring the vehicle within the marked boundaries. However, the final orientation of the vehicle was not completely perpendicular to the wall. This means that currently the system is not accurate enough to be able to dock the vehicle into a charging port.

7.2 Achievements

Our objective was to build a teleoperated system around ROS that uses 4G LTE network for communication purposes. Our aim was to provide a solution that uses human telepresence rather than their actual presence inside car to take over the control of a car in a scenario that a Level 2 or upwards automated vehicle cannot get out of itself.

The teleoperated system built in this study is built around ROS and it achieves the objective of being controlled remotely over the 4G LTE network. Moreover, ROS handles the communication protocols, therefore we did not require to create our own web application for connecting control station to the remote vehicle. Additionally, we were also successful in our objective of using low-cost commercially available equipment. The total cost of our project on parts was less than $1,000. Also, all the equipment was easily commercially available. This makes our system feasible to be deployed in low-cost low speed urban vehicles.

Moreover, as the system is built around ROS, we can use the capabilities provided by ROS for further development work on the vehicle. Some of the capabilities provided by ROS that can be integrated into our system are as follows (South West Research Institute, n.d.):

- Interfaces for sensors such as LIDAR, camera, IMU, GPS, vehicle CAN bus etc.
- Real-time vehicle controller
- Multiple path-planners
- Perception technologies such as terrain classification, lane following, obstacle detection and recognition
- Persistent world model capable of fusing a priori map and terrain data with sensed information
- Various autonomy modes including teleoperation, dismount following, convoying, and fully autonomous navigation
- Simulation and data visualization tools

The system in its current state is unfit for teleoperating on an ordinary road as we still experience delays in communications that can be as large as 3 seconds to 4 seconds. Therefore, it cannot be deployed in high speed vehicles operating on roads and highways. However, the system in its current state can still be used in areas with low vehicle and human traffic. An example of such a scenario could be parking and remote repositioning of a golf cart which was the objective of (Sepulveda, 2016).

We were also able to build a docking procedure for our vehicle using location and visual data. The docking procedure was a partial success as it allowed the vehicle to be parked between two lines with a success rate of 80% over 10 runs.

7.3 Improvements

In this section, we will briefly discuss some improvements that can be implemented to improve the working of both systems.

7.3.1 Improvements in Teleoperation System

Author feels certain improvements can be made to improve the performance of the teleoperated system.

7.3.1.1 Predicting Network Congestion and achieving consistent time delay

The overall latency is dictated primarily by how congested or busy the cellular network is. Unfortunately, the teleoperated system has no control over the amount of people connecting to the cellular network. Also, as mentioned in the literature review, multiple studies have pointed that a constant delay in communication is better than variable delay even if the delay is greater. Therefore, the author recommends building a system that predicts the network congestion and achieves a consistent time delay.

7.3.1.2 Predicting Loss of Communication and bringing the vehicle to a safe stop

Although, network congestion is an issue but even a more major concern is communication loss between the vehicle and control station. Although, the safety systems are incorporated into the system, but they are only activated if a collision is imminent. However, the author recommends building an additional safety system that monitors the communication channels and is triggered if it predicts that there is an imminent loss of communication. Ideally, such a system would be able to bring the vehicle to a safe stop until the communication is established again with the control station.

7.3.1.3 Adding parallel connections

The author also recommends adding parallel or redundant connections. The aim of implementing redundancies in the system would be to i) try to reduce connection loss by having a backup system ii) decrease bandwidth constraint on the system by adding an additional pipeline through which data can flow. This would also allow to send additional data to the operator that will help in increasing the situational awareness of the operator.

7.3.1.4 Incorporate object tracking system into the LIDAR based safety system

Currently, the safety system is triggered based on the data from the LIDAR. However, a more robust system can be built if the already applied object tracking system is incorporated into the safety system. The safety system would check for a potential collision by comparing the trajectories of the obstacles in the video feed with the trajectory of the vehicle. If a collision is imminent, it will bring the vehicle to a safe stop.

7.3.1.5 Increase Situational Awareness of the Operator by adding multiple cameras

Additionally, to improve the situational awareness of the operator is to add more cameras to the system. However, adding more cameras will result in an increase in communication latencies as more data would need to be transferred. A trade-off analysis between negative effects in performance caused by communication delay, and positive effects in performance caused by enhances in situational awareness would be required to determine the optimum number of cameras, and the resolution and FPS for each camera. Another option would be to stitch together images from multiple cameras and provide the operator with a 3D view of the surroundings.

7.3.2 Improvements in Docking System

As mentioned in Chapter 6, the docking system in its current state is partially successful. Although, it can dock/park the vehicle within the boundaries most of the times, the final orientation of the vehicle is not completely perpendicular to the wall. Hence, in its current state the system is not accurate enough to dock itself into a charging port.

Several improvements can be made to the docking system.

7.3.2.1 Incorporating dynamics of the car in the Docking System

The docking system algorithm does not take the dynamics of the vehicle in account. It just works based off the GPS and visual feedback. The visual servoing algorithm tries to center the QR code in the camera. It requires the vehicle is moved left or right based on the readings from the camera. As the vehicle is unable to achieve purely left or right motion, the final orientation is not completely perpendicular to the wall. Dynamics of the vehicle should be incorporated into the system to ensure control over the final orientation.

7.3.2.2 Use of Stereoscopic Camera to estimate the pose Visual Marker

The visual servoing algorithm works off the 2D camera. Hence, the pose of the QR code in 3D is not calculated. Therefore, it was observed in testing that most of the times the vehicle is in a final orientation that is at some non-zero angle with the wall.

Use of stereoscopic camera would allow a better estimate of the pose of the visual marker. Consequently, the final orientation of the vehicle could be corrected if required.

7.4 Future Work

We have built the ability to control a vehicle through a computer. Moreover, we also have installed multiple sensors for i.e. cameras (with object detection ability), LIDAR, speed measuring sensors, GPS. This opens a lot of avenues for future work related to this project.

The vehicle can be made more autonomous with all the sensors mounted on it. By adding more cameras and LIDARs to the vehicle, and by performing sensor fusion, the vehicle can be made to drive autonomously. In such a scenario, while teleoperating the vehicle, the operator would have much less workload. The operator will only mark points on the map, and the vehicle will autonomously drive from the current point to the marked point on the map. If the vehicle gets stuck somewhere, or in a case of emergency, the operator can remotely take over the control of the vehicle and drive it.

A lot of other possibilities also arise if autonomous driving capabilities are added to a low speed urban vehicle. As an example, in our school, a lot of low speed urban vehicles are used as service vehicles. If multiple such vehicles have autonomous driving capabilities added to them, they can communicate and collaborate with each other to perform tasks. Moreover, the communication capabilities will allow the vehicles to communicate with the control station. This whole setup would allow a single operator to operate a whole fleet of service vehicles.

Vehicle platooning is another avenue that can be explored with our current setup. Multiple low speed vehicles that are being teleoperated by a single operator can be platooned. Such an application would be very useful in airline industry for transporting airline baggage.

Communication and collaboration are not only limited to ground vehicles only. The teleoperated vehicle we have built can also communicate and collaborate with aerial vehicles like drones. For example, a teleoperated vehicle can serve as a moving charging base on which multiple aerial drones can land and recharge themselves.

Future avenues for the docking system also exist. We observed that the docking system in its current state is not accurate enough to achieve docking into a charging port. However, it is accurate enough to get the vehicle in close vicinity of the charging/docking station. A system can be built in which the vehicle communicates with a robotic arm (or some other device) at the docking/charging station to let it know that it is in close vicinity of the docking station. The robotic arm or some other mobile device than makes the final fine adjustments that would allow the vehicle to dock itself into the station.

REFERENCES

Allen, L., Wolfe, D., Judy, C., Haukkala, E., & James, R. (2001). NDGPS Network Enhancments in FY01. *Proceedings of the 14th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS)*, (pp. 2722-2734). Salt Lake City.

Associated Press. (2018). Retrieved from https://www.foxnews.com/auto/california-oks-autonomous-car-testing-without-backup-drivers

Baker, C., Morris, A., Ferguson, D., Thayer, S., Whittaker, C., Omohundro, Z., . . . Thrun, a. S. (2004). A Campaign in Autonomous Mine Mapping. *IEEE Conference on Robotics and Automation.* IEEE.

Bezdek, A. (2020). *Differential GPS Configuration On Golf Cart.* Atlanta.

Chen, J. Y., Haas, E. C., & Barnes, a. M. (2007). Human Performance Issues and User Interface Design for Teleoperated Robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Volume 37, Issue 6*, IEEE.

Choi, P. J., Oskouian, R. J., & Tubbs, R. S. (2018). Telesurgery: Past, Present, and Future. *Cureus, Volume 10, Issue 5*.

Chucholowski, P. M., Buchner, S., Reicheneder, J., & Lienkamp, a. M. (2013). Prediction Methods for Teleoperated Road Vehicles. *Conference on Future Automotive Technology.* München.

Cui, J., Tosunoglu, S., Roberts, R., Moore, C., & Repperger, D. W. (2003). A Review of Teleoperation System Control. *Florida Conference on Recent Advances in Robotics.* Boca Raton: FCRAR.

Davies, A. (2017, January 05). *Nissan's Path to Self-Driving Cars? Humans in Call Centers*. Retrieved from Wired.com: https://www.wired.com/2017/01/nissans-self-driving-teleoperation/

Davies, A. (2019, 03 26). *Wired*. Retrieved from wired.com: https://www.wired.com/story/designated-driver-teleoperations-self-driving-cars/

Ferguson, D., Morris, A., Haehnel, D., Baker, C., Omohundro, Z., Reverte, C., . . . Thrun, a. S. (2003). An Autonomous Robotic System for Mapping Abandoned Mines. *Advances in Neural Information Processing Systems (NIPS).*

French, J., Ghirardelli, T. G., & Swoboda, a. J. (2003). The effect of bandwidth on operator control of an unmanned ground vehicle. *I/ITSEC.* Florida.

Gnatzig, S., Chucholowski, F., & Lienkamp, T. T. (2013). A System Design for Teleoperated Road Vehicles. *International Conference on Informatics in Control, Automation and Robotics* (pp. 231-238). Reykjavík: Springer.

Gortez, R. C. (1949). *United States of America Patent No. US2632574A.*

He, K., Zhang, X., Ren, S., & Sun, a. J. (2016). Deep Residual Learning for Image Recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770-778). Las Vegas: IEEE.

Hosseini, A., & Lienkamp, a. M. (2016). Enhancing telepresence during the teleoperation of road vehicles using HMD-based mixed reality. *IEEE Intelligent Vehicles Symposium* (pp. 1366-1373). Gothenburg: IEEE.

Hosseini, A., & Lienkamp, a. M. (2016). Predictive Safety Based on Track-Before-Detect for Teleoperated Driving Through Communication Time Delay. *Intelligent Vehicle, IEEE Symposium.* IEEE.

Huawei Demonstrates 5G-based Remote Driving with China Mobile and SAIC Motor. (2017, 06 28). Retrieved from huawei.com: https://www.huawei.com/en/news/2017/6/5G-based-Remote-Driving

Jordan, J. (n.d.). *An overview of semantic image segmentation.* Retrieved from Jeremy Jordan: https://www.jeremyjordan.me/semantic-segmentation/

Korosec, K. (2019, August 14). *TechCrunch*. Retrieved from techcrunch.com: https://techcrunch.com/2019/08/14/self-driving-truck-startup-starsky-launches-hutch-its-api-and-nerve-center/

Lichiardopol, S. (2007). *A Survey on Teleoperation.* Eindhoven: Technische Universiteit Eindhoven.

Liu, R., Kwak, D., Devarakonda, S., Bekris, K., & Iftode, a. L. (2016). Investigating Remote Driving over the LTE Network. *Automotive UI.* Association for Computing Machinery.

Long, J., Shelhamer, E., & Darrell, a. T. (2015). Fully Convolutional Networks for Semantic Segmentation. *Computer Vision and Pattern Recognition.* Boston: IEEE.

Ohnsman, A. (2018, 06 5). *Forbes*. Retrieved from forbes.com: https://www.forbes.com/sites/alanohnsman/2018/06/05/when-your-robot-car-gets-stumped-this-startup-wants-to-steer-it-out-of-trouble/#5d29695a7fca

Parasuraman, R., Barnes, M., & Cosenzo, K. (2007). Adaptive Automation for Human-Robot Teaming in Future Command and Control Systems. *The International C2 Journal, Voume 1, Number 2*.

SAE. (2018, 12 11). Retrieved from https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, a. L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4510-4520). Salt Lake City: IEEE.

Sepulveda, R. (2016). *Evaluation of Teleoperation System Performance Over a Cellular Network.* Atlanta: Georgia Institute of Technology.

Sheridan, T. B. (1995). Teleoperation, telerobotics and telepresence: A progress report. *Control Engineerinfg Practice, Volume 3, Issue 2, 3*(2), 205-214.

South West Research Institute. (n.d.). Retrieved from SWRI: https://www.swri.org/robot-operating-system-ros

Stanford Artificial Intelligence Laboratory et al. (2016). *Robotic Operating System*. Retrieved from ROS: https://www.ros.org

Suomela, J. &. (2001). COGNITIVE HUMAN MACHINE INTERFACE OF WORKPARTNER ROBOT. *IFAC Proceedings Volumes, 34*, 51-56.

Swanson, K. D. (2013). *Identification of Stability Thresholds in Time-Delayed Vehicle Operation.* The Pennsylvania State University.

Tang, T., Vetter, P., Finkl, S., Figel, K., & Lienkamp, a. M. (2014). Teleoperated Road Vehicles – The “Free Corridor” as a safety strategy. *Applied Mechanics and Materials*, 1399-1409.