DIGITAL MAP BASED NAVIGATION SYSTEM FOR AUTONOMOUS VEHICLE WITH DGPS LOCALIZATION

by

Balasubramaniam Ramakrishnan

Submitted in partial fulfillment of the
requirements for the degree of
Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
August 2012

DALHOUSIE UNIVERSITY

DEPARTMENT OF MECHANICAL ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "DIGITAL MAP BASED NAVIGATION SYSTEM FOR AUTONOMOUS VEHICLE WITH DGPS LOCALIZATION" by Balasubramaniam Ramakrishnan in partial fulfillment of the requirements for the degree of Master of Applied Science.

Dated: August 27, 2012

Supervisor: ______________________

Readers: ______________________

____

DALHOUSIE UNIVERSITY

DATE: August 27, 2012

AUTHOR: Balasubramaniam Ramakrishnan

TITLE: DIGITAL MAP BASED NAVIGATION SYSTEM FOR AUTONOMOUS VEHICLE WITH DGPS LOCALIZATION

DEPARTMENT OR SCHOOL: Department of Mechanical Engineering

DEGREE: M.A.Sc. CONVOCATION: May YEAR: 2013

Signature of Author

Dedicated to my parents for accepting and

supporting my decisions and

encouraging in my endeavours.

Table of Contents

List of Tables

List of Figures

Abstract

Autonomous Vehicles (AV) can navigate itself from point 'A' to point 'B' without the aid of humans. Research on autonomous vehicles were primarily focused on the localization, navigation and path planning schemes. This led to numerous methods in each of the fields of focus. This research focuses on creating a scheme for the autonomous vehicle to navigate using minimal sensors and get maximum data/information from the map. At first a digital map contains various structures and each has an associated database. This database contains the details of the environment. At present these data are manipulated for use by humans and for this map to be used with autonomous vehicle require more sensors. This work designs maps for use with autonomous vehicle and navigates using differential GPS (dGPS) of high accuracy for localization. Then the vehicle gets path and directions from digital map and navigates using multiple waypoints that are provided by the path. Finally, the scheme is tested and demonstrated through simulation and test results.

List of Abbreviations and Symbols Used

AV- Autonomous Vehicle

GPS-Global Positioning System

dGPS- differential Global Positioning System

DARPA-Defense Advanced Research Projects Agency

WGS-World Geodetic System

SBAS-Satellite Based Augmentation System

WAAS-Wide Area Augmentation System

θ- ‘Theta’ Steering angle

GNSS-Global Navigation Satellite System

Acknowledgements

First I would like to thank my supervisor Dr. Ya-Jun Pan of the Mechanical Engineering Department for the opportunity and her support and guidance. Like to thank Dr. Marek R.Kujath and Dr. Jason Gu for their part in my supervisory committee, and also like to thank GIS staffs Ron Fritz and Raymond Jahncak for sharing their knowledge in GIS. Thanks to members of ACM Group (Leslie, Trent, and Xiang) for all the help by lending a hand for testing and valuable suggestions. Thanks to all the people who helped me along the way with my learning experience.

Chapter 1

Introduction

As soon as the automobiles came into existence it has been a human dream to have a vehicle that can drive itself. The invention of global positioning system brought the dream closer to reality. Advancement happened when Defense Advanced Research Projects Agency (DARPA) Challenge was put forth we will see in the history section. And recently automobile companies were looking in to manufacturing vehicles with autonomous capabilities. The next generation GPS III will have both low frequency and high frequency transmission which enable the high accuracy GPS signal to reach both indoors and tunnels and structures as such [3]. The automobile is a major engineering contribution of the last century. It has become a part of life for most people. The constant improvements in cabin technology has increased driver distraction and it can be overcome by letting the vehicle drive itself.

1.1 History

Some of the events that unfolded leading to this research.

1.1.1 GPS/dGPS

In 1960, the GPS got its start when United States Government, Department of Defense (DOD- US), NASA, Department of Transport (DOT) wanted to create a satellite based three dimensional positioning. In 1968, Secretary of Defense established Defense Navigation Satellite System (DNSS) program to consolidate each military services to make a joint single system. DNSS started a steering group that came up with concept *NAVSTAR* GPS. This program was developed by the GPS Joint Program Office (JPO) and it continues to oversee the development of hardware (satellite and ground equipment) and majority of military receivers. In 1999, US Government announced modernization of civil GPS by adding to new frequencies namely L2C and L5. This gave birth to civilian differential GPS. The primary GPS was opened to

Figure 1.1: GPS Satellite System (Garmin Website)

public use after the military got hold of the dGPS for their survey and defence applications. The earlier GPS had compounded problems starting with coverage. Since there were less satellites at the initial time period and the inaccuracy in the quality of signal signal reflection and others cases made very unreliable. When the GPS was made available to public and the lower cost made its use in abundance pushed the US military to add more satellites to cover more land mass as well make sure there is a satellite at any given time. The GPS system of satellites as shown in Fig. 1.1 provides coverage around the world at an given time with at least one satellite available. Eventually the dGPS was made available to public when additional frequency for defence purpose were added to it. The accuracy of a standalone GPS is 10 m, and the accuracy dGPS is in terms of single digit meters to few centimetre depending on the type used.

1.1.2 Autonomous Vehicle

The autonomous vehicle research began with the focus on making a vehicle to drive itself when the US military decided to have a autonomous ground vehicle fleet. The research was held within the fence of military in the early stages during 1980's. The military opened the research for civilians, university research and automobile enthusiasts in late 80's and early 90's. There research gave out lot of innovative features (sensor development) that are standard in todays vehicles. The first test of the civilian research came out in 1993.

1.1.3 DARPA Challenge

Defense Advanced research Projects Agency (DARPA) is a US government agency that funds and overseas many advanced technology development for military and some civilian use. DARPA decided to take autonomous vehicle research up by a notch in early 2000's.

Grand Challenge : Grand Challenge is the first challenge put forth by DARPA to the Universities and companies involved in this area of research. The objective of this challenge was to have more active involvement in developing non remote control, fleet of autonomous vehicle that follow a predetermined path in the desert for a specified distance of about 200 km. The result none of the participants actually completed the course but only one vehicle travelled the furthest of about 11-12 km and won that years award [4]. The next challenge in 2005 had about 23 qualified for the

(a) Oshkosh Truck Corporation, University of Parma

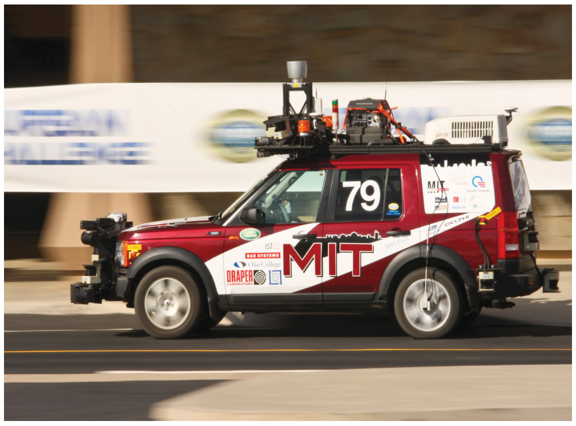
(b) Carnegie Mellon

Figure 1.2: Autonomous Vehicles in Grand Challenge

final and this time most of the vehicles travelled greater distance than 11 km mark for previous year record, also five of the participants completed the entire length of the challenge. This success led to the next challenge putting autonomous vehicle in urban environment. Some of the vehicles participated in grand challenge are shown Fig. 1.2. Fig. 1.2 (a) finished last in the challenge; it had sensors like GPS/IMU, multi direction vision among other sensors [5]. Fig. 1.2 (b) finished third; this is one of two vehicles from the same team; the other finished second. Both vehicles used laser scanner, stereo camera, GPS and inertial sensors to name a few and seven Intel processors for control.

(a) Stanford and Virginia Tech AVs

(b) Carnegie Mellon AV


(c) MIT AV

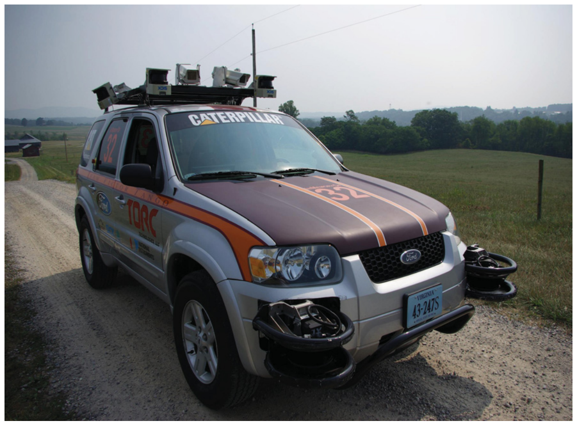

(d) Virginia Tech AV

Figure 1.3: Autonomous Vehicles in Urban Challenge

Urban Challenge: After the success of the Grand challenge the DARPA decides to take the vehicles from desert to urban environment. After the 2005 challenge the

DAPRA announces urban challenge in two years time and also announces a higher award price. Now the number of participants has gone up and DARPA decides to filter them setting certain standards. Some of the vehicles participated are shown in Fig. 1.3. This challenge involved an secluded urban layout for testing with the traffic involved to simulate the urban driving scenario [3].

Fig. 1.3 (a) shows vehicles in controlled intersection in perpendicular direction of travel negotiating each other. The vehicle on the foreground to the left is stanford racing controlled by seven Intel processors. This AV uses GPS, inertial and vision to mention some of the sensors onboard [6]. This finished the course in second place. Fig. 1.3 (b) carnegie mellon team which won the challenge. The vehicle has cameras, radars and lidars sensor onboard for localization and with 500,000 lines of code to control [7]. Fig. 1.3 (c) shows the MIT vehicle that finished fourth. Has a roof mounted sensor system include GPS, vision, lidar and inertial [8]. Fig. 1.3 (d) shows the virginia tech vehicle that finished third, has GPS with IMU, laser range finders and camera [9].

1.2 Applications of AV

The application is to have a vehicle that needs no driver but be able to make decisions for itself and travel between destinations avoiding any accidents. The other application would be as an autopilot; enabling long distance travel without any fatigue related accidents, tourist transport services and other transit programs. Same system can be used for cargo transport and of course for security patrolling with guys as lookout or camera controlled by remote.

1.3 Literature Review

The autonomous vehicle research as mentioned in the introduction started by military, the GPS was developed by military as mentioned earlier in this chapter and was not available for civilian use for some time. When the AV research was opened for private and academic researches the GPS got integrated from start as a primary mode of localization but due to low accuracy and transmission errors it wasn't used alone. The increase in availability made the GPS a prominent part of most researches. The research was separated in to two based on the type of GPS used. In the following sections we review some of the work in the field.

1.3.1 GPS Based Navigation

The recent work uses a low cost GPS assisted with the map aided vision, in which the GPS errors is corrected using the inertial sensors and the camera feed is compared to the 3D map form US Geographical Survey Digital Elevation Maps (DEMs) and the sensor fusion is done using Kalman filter [10]. Next work is on Unmanned Arial Vehicle (UAV) localization using sensor fusion (GPS and inertial) by state dependent Riccati Equation (SDRE) [11]. The next work by Chirdpong Deelertpaiboon and Manukid Parnichkun [12] is focused on the sensor fusion. This work uses the following sensors for localization of the intelligent vehicle, GPS, Compass and camera. The work proposes a fuzzy based sensor fusion algorithm for localization of an intelligent vehicle by correcting translational error of latitude and longitude in easily available maps such as Google Earth. The following work is based on GPS, 3D-GIS and laser. The 3D-GIS and the laser are used when the GPS is not available. The laser (real) data is matched with the 3D-GIS laser (virtual) data, using the Iterative Closest point algorithm (ICP) for navigation [13]. The next work is based on multiple sensor fusion for navigation in the urban area during GPS outage. The paper introduces multi-path reflection approach for the navigation [14]. These works show that the GPS alone cannot do the job at hand and this work is intended to use only GPS for localization and that why the choice of sensor for this work is dGPS.

1.3.2 dGPS Based Navigation

The dGPS cost is higher and also its recent compare to GPS so the number of research using is less compared to GPS research. Here are some of more relevant works. This work is based on carrier-phase based differential GPS and SLAM based laser scanner localization. In this work the authors have used Extended Kalman filter method for slam for places where there is GPS outage [15]. The next work is a Matlab based simulation using the dGPS and aerial images from the laser detection and ranging (LADAR), light detection and ranging (LIDAR) [16]. The following work is based on realtime dGPS and laser range finder integration. A fuzzy logic steering is used for the concept of human driving [17]. The following work uses extended Kalman filter for the integration of the sensors. The work is based on using dGPS in line with odometer is obtained by using dGPS and sensors. Laser is used for curb detection for the purpose of navigation and obstacle avoidance [18]. The works discussed here are for robot like vehicles and some are for car-like vehicles. These works does not use the digital map and the intended use of the is not the same. This work is focused on road transport vehicles.

1.3.3 Path Following Methods

There are many methods that are developed and used for following path that are generated for reaching the destination. Some of the path planning methods also act as following methods. Some of the path following methods are discussed here. The landmark based navigation is the primary method used for mobile robot navigation. The landmark can either be a waypoint or an obstacle [19]. Multiple waypoint trajectory is a simple and effective method widely used. This method uses waypoints at certain intervals that guide the vehicle towards the desired path. The work by Hashima and Lu [20] uses this method with time and orientation using geometric approach. The other work similar by Chen and Fraichard [21] uses waypoint with feedback system. The wall following method is the one where the vehicle uses sensors like sonar to find a structure like wall and move along side the wall by repetitively scanning for the presence of the wall. The next work uses kinematic model of a car-like vehicle with sliding and slipping and uses closed loop control to compute the heading [23]. Since the proposed work has the path available as waypoints and the input to the test vehicle being steering and speed the a simple waypoint path following is used.

1.4 Thesis Contribution

The literature discussed above showed some of the focus methods and the trend is using multiple sensors to obtain the non-changing environment and as well the environment that are mobile in some cases. The map can provide the environment data for known environment by basically designing the map to provide path avoid obstacles. Thus minimizing the hardware requirement and/or minimizing the workload on the sensors by utilizing them in certain on demand conditions. This work focusses on designing the digital map for autonomous vehicle and use as the source of data required to make decisions for navigation. The system architecture is built around the digital map. The map is used as the source of data for making decisions in creating path. The data required for navigation is provided by the user, the map and GPS sensor. The software is built using the vehicles base platform - the MRDS, and this enabled this work to share the computation between two processors. The intention is to minimize the number of sensors used for acquiring the environmental data there by minimizing the computation power required. The sensors used for this work are dGPS and laser range finder. The system proposed here can be applied to any transport vehicles. It can be used as a transit system in airports, institutions, and industries. Note that lot of time were committed to programming, debugging and experimental testing.

1.5 Thesis Organisation

This thesis is organized in the following order. Chapter II discusses the problem formulation and discusses the features of map and its construction. Chapter III looks in detail about the current scheme and the control, navigation. Chapter IV shows the simulation results and results analysed. Chapter V discusses the hardware setup and the test results are analysed. Chapter VI summarizes the the works and concludes.

Chapter 2

Problem Formulations

This chapter discusses the basic components of the system, and explains the basic components that make the proposed system. The first section explains the science of cartography and how its being implemented. The subsection explains projection systems used that is relevant to this work, because there are a lot of different systems for different purpose and locations. This is the standard that define maps and its unit of measure. The second section discusses about the dGPS and how it works, types and the advancements coming in near future. The third section discuss the availability of GPS at present. The fourth section discusses Microsoft Robotic Developer Studio (MRDS) and how it works. Also discuss how MRDS is different and beneficial. Finally we elaborate the problem and present the assumptions made going forward towards this work.

2.1 Digital Map

Digital Maps are created using satellite images of topography as the base and building layers to represent different structures as different polygon layers. The database contains all the attributes and informations that may be required by the user. The maps are calibrated by taking multiple geographic surveys using dGPS. These maps have different coordinate system, this is discussed later in this chapter. The navigation map are also called road map are developed by two companies namely *Navteq* & *TeleAtlas*. The coordinate system used in a road map is WGS and the map used for creation of the map used in this work. The maps used for other government purposes use geographic coordinate system. In the early years of GPS usage the map coordinate system (geographic) was considered to be same that of WGS. Later it was found that there is a difference between geographic coordinate and WGS are different. The difference is a constant value and was included while the map is converted for road map, hence now the map position coincides with that of a WGS. Fig. 2.1 shows the

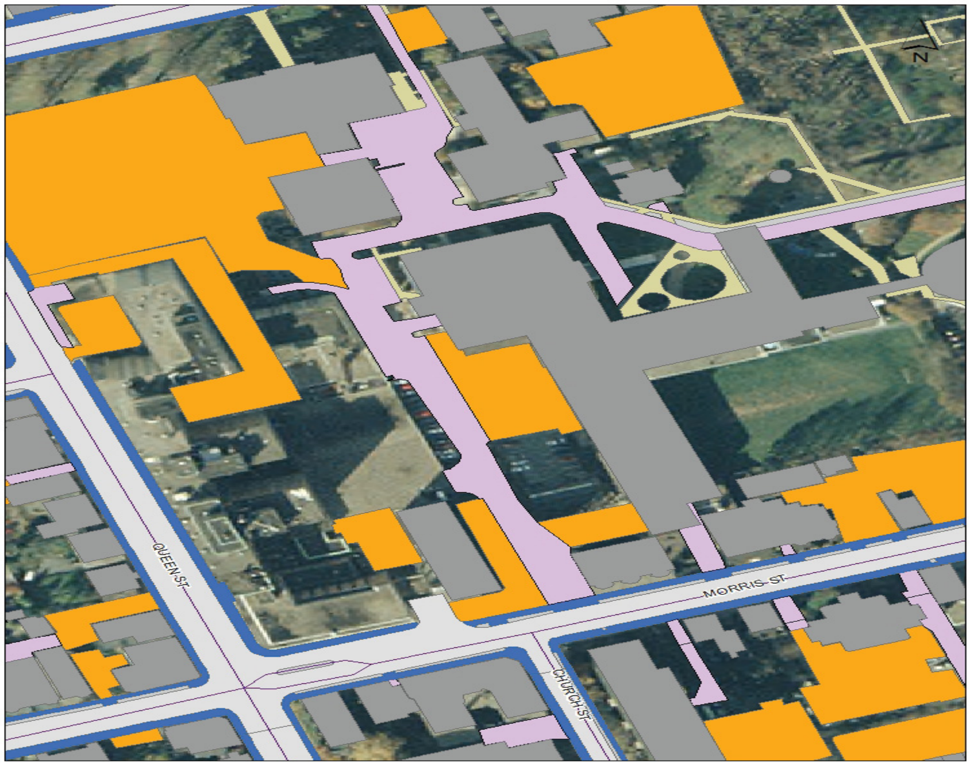


Figure 2.1: Map Layers Representing Different Structures

layers of the map taking shape. The satellite image is used to create different layers which in turn create a database representing each component and its properties. In this Fig. 2.1, different colour polygon represent buildings, parking lots, different type of road and sidewalks. The map view of the normal consumer map uses this layer to represent places. Fig. 2.2 shows the legends of Fig. 2.1. The grass and other not in use land mass are usually represented by green polygons not shown on this map. The line with street names is the feature that contains address information and is used for generating path. In a normal road map the road is a line that falls with in the boundaries of road polygon and it is not accurate and not suitable for use with AV as the primary source. The road line feature represents both side of the road, this is a problem as the AV will need median scanning sensor which have to be active all time to check AV from entering opposing lane. So for AV the map needs to be converted to assist the AV. In this work a map for AV is created and put to test.

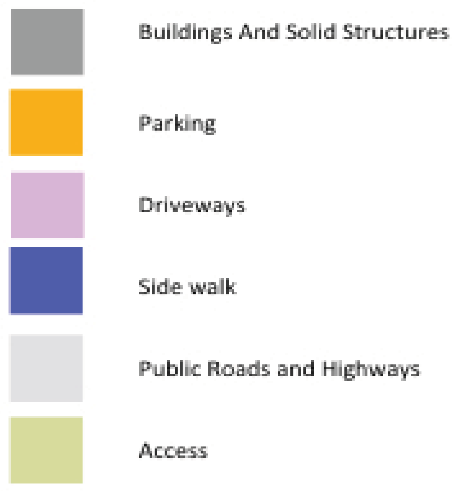


Figure 2.2: Map's Different Structures and Respective Colours

Coordinate System

Table 2.1 shows the difference between the position overlay of GPS in different coordinate systems. The coordinate system defines the way a map is laid out. The

Table 2.1: Coordinate System Overlay Differences

Different Coordinate Systems	
WGS84	Space or Satellite orbit based reference
ITRF2000	1 cm offset from the WGS84
NAD 83	2.2m offset from WGS84

map can be based on the view from satellite (the true shape and dimensions) and the map that is created with assumption earth is flat. The prior map is created using the distance in two planes North South and East West. The maps created using the true shape include the height or elevation and the different diameters along poles and equator. The latter method uses distances from a landmark and does not consider the different distances along pole and equator. The GPS system is satellite based and works with the map created alike. To get the distance in KM or Miles from the WGS map needs converting degrees to the preferred units (KM/mile). There are few methods to convert but the preferred and most used for converting distance between two degrees (WGS) to units of length is Haversine Equation (Code 2.1 in Matlab). The Haversine equation or formula uses difference of two latitudes and longitudes and computes the distance by multiplying the radius of earth as shown in Code 2.2 and the output unit of measure depends on the unit of the radius. This formula is used throughout the calculation at various stages.

Code 2.1: Haversine Equation used in Matlab

```
%harversine Formula
    dxcr=rutx(2)-xc(1);%del-long
    dycr=ruty(2)-yc(1);%del-lat
    cam=sind(dycr/2)^2+cosd(yc(1))*cosd(ruty(2))*sind(dxcr/2)
      ^2;
    ccm=2*atan2(sqrt(cam),sqrt(1-cam));
    crdm=earthm*ccm;
 %end hf
```

Code 2.2: Haversine Equation in MRDS

```
  private Double HS(double[,] r, double[,] t)
  {
   int lh = r.Length;
   double reflat = r[1, 0]; double reflong = r[0, 0];
   double tlat = t[1, 0]; double tlong = t[0, 0];
   double dlat = tlat - reflat;
   double dlong = tlong - reflong;
   double ea = Math.Pow(Math.Sin((dlat/2)*(Math.PI/180)),2) +
Math.Cos(reflat*(Math.PI/180))*Math.Cos(tlat*(Math.PI/180)) *
Math.Pow(Math.Sin((dlong/2)*(Math.PI/180)),2);
   double ec = 2*Math.Atan2(Math.Sqrt(ea), Math.Sqrt(1 - ea));
   double mdis = earthm *ec;
   return mdis;
  }
```

2.2 dGPS Sensor

The addition of two frequencies for civilian use led to rapid development of GPS service and also lowered the cost of stand-alone GPS hardware. The purpose of these additional frequency is to provide correction and give the user a more precise position. These differential corrections can be obtained by different ways which are discussed later. The differential corrections provided compensation for ionospheric delay and associated issues in the GPS that were reducing its accuracy. Further in 2007 the satellites were replaced by newer once which had components to generate L5 frequency messages and various other precise outputs [3].

There are two ways of getting augmentation, also known as differential correction hence dGPS. The two types are absolute and relative. The absolute gets a global correction value in the range of a meter up to 0.2 m. The relative gets correction from the local station that provide correction in relation to its position. The accuracy of this method is in the range of centimetres. The signal range of the station is limited and varies with hardware and location.

Absolute methods; Local and Regional that uses satellite as differential source called Space Based or Satellite Based Augmentation System (SBAS). The SBAS system developed and used in North America available world wide is Wide Area Augmentation System (WAAS). The accuracy of this system ranges from 2 m to 0.2 m, and it depends on the quality of signal and the receiver. The disadvantage of this system is its inaccuracy with altitude. The accuracy of altitude is up to 10m. Fig. 2.3 show the method by which this system gets the differential. It uses two sources, the signal from the orbiting satellites, which requires three or more satellites. The second source is the geo-stationary satellite and this provides the differential by calibrating its position using base station. This system is used in autopilot systems in aeroplanes during flight, without altitude measurement from the GPS. This system is also used for cartography; due to the tectonic plate movements and other natural influence changing the landmass the satellites position (geo-stationary) and the digital map needs calibration. This system is used to setup the other differential method described next.

Relative methods include Code Based and Carrier Phase Based (Fig. 2.4). This method requires a satellite based reference GPS station and uses the same hardware with the capability to connect to the reference station. Fig. 2.4 shows the general operation of this system. The system is also referred as Real Time Kinematic (RTK). The GPS gets differential from the reference station which is a stationary source that gets its position from a satellite. This system produces accuracy in centimetres the range depends on the source accuracy and hardware capability. Limitation is the

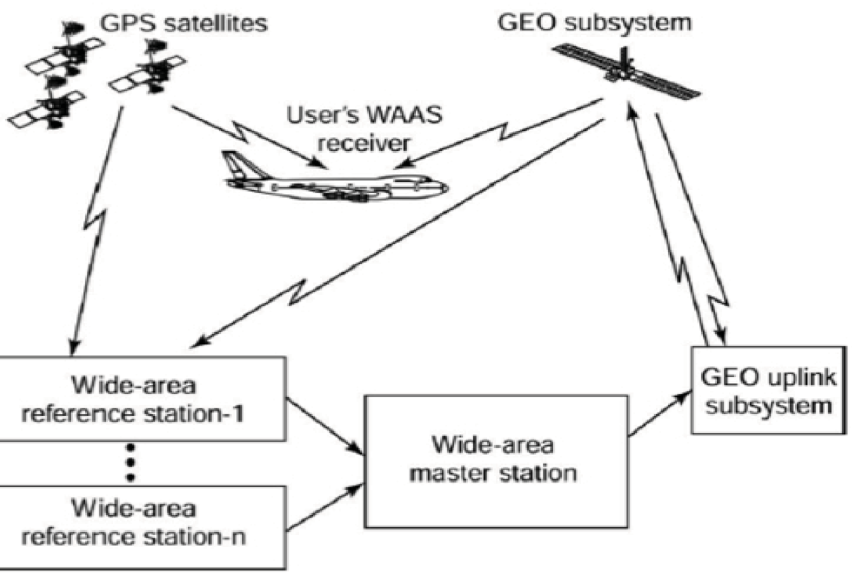


Figure 2.3: Wide Area Augmentation System

reference station should be set up and number of reference stations needed for longer range.

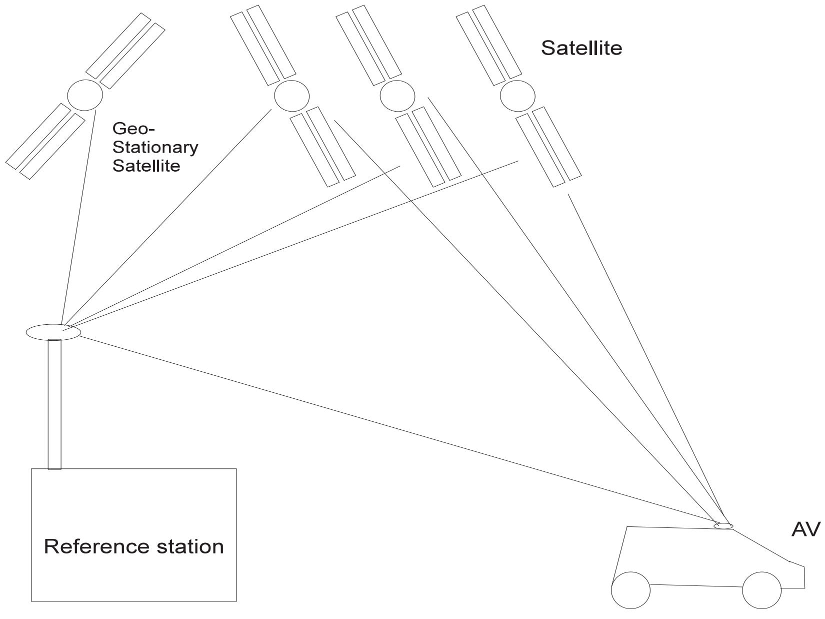


Figure 2.4: Code/Carrier Based System

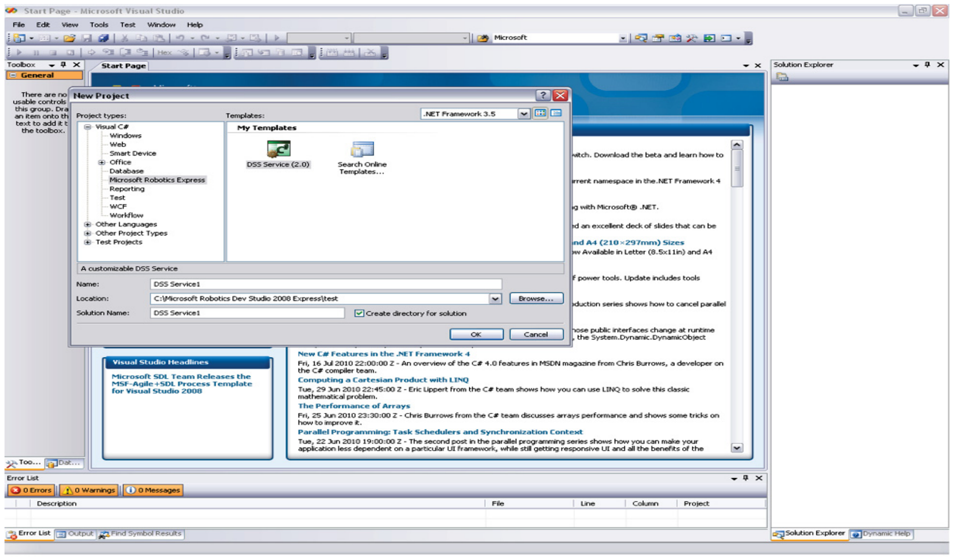
GPS III program is underway, to be available for civil usage in 2030 envisioned to provide sub meter accuracy, the launch of the first GPS III satellite was planed for 2013 [3].

2.3 GPS Availability

Availability of GPS at a place is calculable since the orbits of GPS satellites are well known. At any given time the position can be calculated using software. The GPS outage i.e, the unavailability of the GPS service, occurs in few places when a satellite is taken offline for maintenance. This usually occur over remote patches over sea, and this was observed based on one satellite taken out for service. An extensive study shows the availability of GPS Services and outage for different number of satellites at the same time [3]. The outage time in case for one satellite taken offline is observed to be in the range of 10 minutes. For three satellites offline the outage can last up to an hour and the occurrence is very rare.

2.4 Microsoft Robotic Developer Studio

MRDS is a relatively new software development by Microsoft team in Australia 2006-2007. MRDS is a developer platform developed by Microsoft for controlling and managing robots. This platform uses technologies such as Decentralized Service (DSS) and Coordinated Concurrent Runtime (CCR). This enables developer to have different program (services) running at the same time in same node or different nodes. The test vehicle uses this software technology for its operation. The software is like an operating system in itself that runs the programs called services. These services in a manifest run within MRDS node. The advantage of this platform is it enables concurrent and asynchronous operation. It means not all the services need to be operational when the manifest is started, each service can be activated when required and different services can be runt alongside each other without interference until the communication between services is required. The services are created using C Sharp (C#) in this work. Fig. 2.5 shows the initial MRDS project creation selector in visual studio. The service may be created within C# but it has some unique syntax and way of working. The services can also be created using C++ and visual basic but the capability of these services is limited [2]. The MRDS consist of the following parts service identifier, contract identifier, state, service handler, main port, notification port and partners as shown in Fig. 2.6.



Project types: Visual C# (Windows, Web, Smart Device, Office, Database, Microsoft Robotics Express, Reporting, Test, WCF, Workflow), Other Languages, Other Project Types, Test Projects

Templates: .NET Framework 3.5

My Templates: DSS Service (2.0), Search Online Templates...

A customizable DSS Service

Name: DSS Service1

Location: C:\Microsoft Robotics Dev Studio 2008 Express\test

Solution Name: DSS Service1

[✓] Create directory for solution

Figure 2.5: MRDS Service Creation

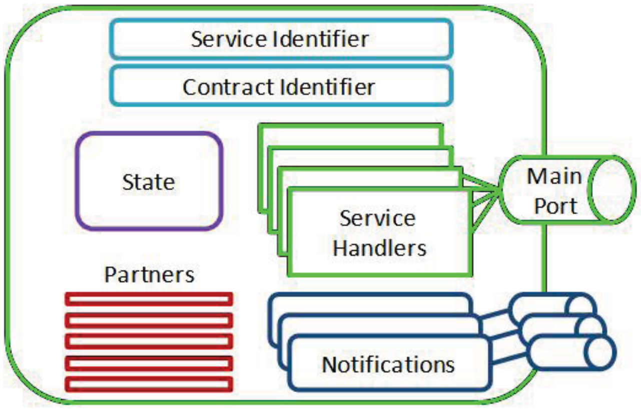
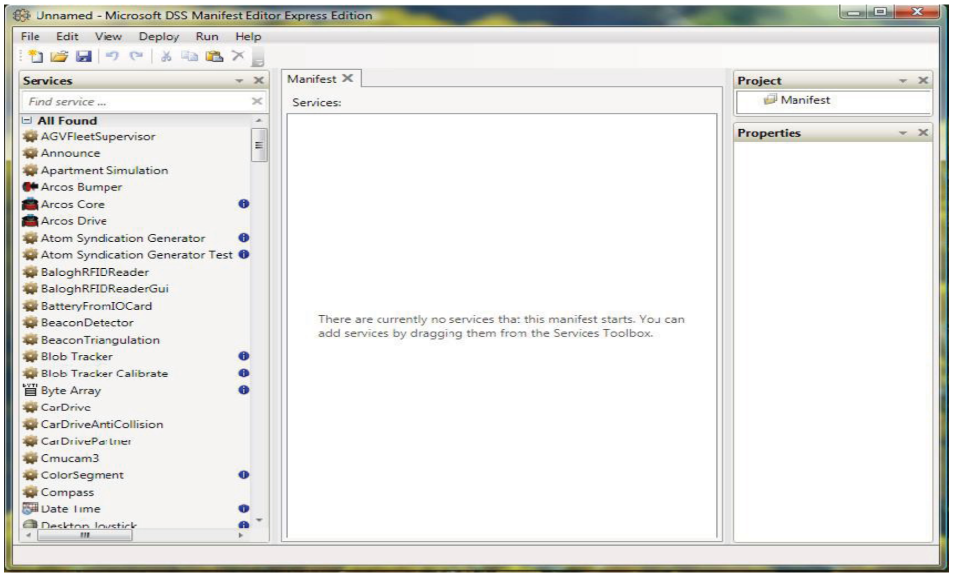


Figure 2.6: Structure of MRDS [2]

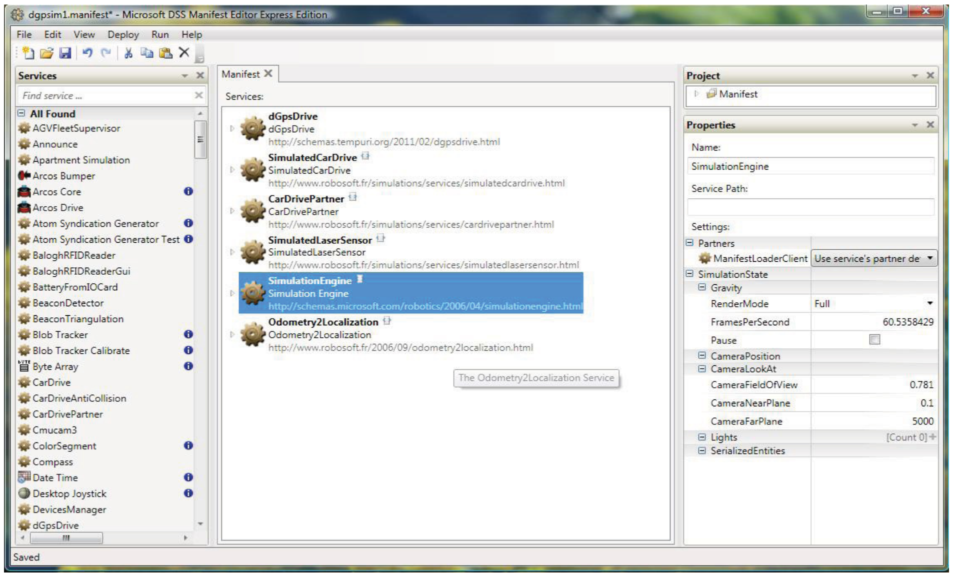
The service identifier is a unique identification used by the manifest to identify the specific service. This identifier helps to have more than one service in same name with different versions or users. The contract identifier does have the same functionality of the service identifier but here the contracts of each service have a unique identification. The following are parts of a service: the state is the general state of the service. The state of a service can be set by the service to give out specific information. The state also gives out any error in communication or in starting the service. The service handlers are handlers for each port and partners. Main port is the port that defines the port in which service is running this is used for creating a connection to this service. The notification port is the one where the results that are to be sent to another service or for display are declared. MRDS manifest is created using MRDS Manifest editor. The parts of the manifest editor is shown in Fig. 2.7. The left pane contains all the services available for use listed in alphabetical order. This also shows the services created by user. The centre pane holds the services that are added to the new manifest that will be created in this process. The right pane contains the properties of each service and the property can be modified and named or change node.



There are currently no services that this manifest starts. You can add services by dragging them from the Services Toolbox.

Figure 2.7: Manifest Editor Window

Fig. 2.8 shows a sample manifest with multiple services added and also shows the properties pane and the properties that can be set. Theses properties are unique to each service and it is available if the service is created to allow changes, but the service path and the name can be edited to have multiple instance of same service to run at a different path (node). The services that are add with specific path runs at the specified path and not where the manifest is running. The manifest are run using the dssnode Fig. 2.10, a console based runtime environment. For convenience a batch file is created to make he manifest run in the console. The manifest only initiates those services. The MRDS provides a web based interface (Fig. 2.9) to monitor what



Services:

dGpsDrive
dGpsDrive
http://schemas.tempuri.org/2011/02/dgpsdrive.html

SimulatedCarDrive
SimulatedCarDrive
http://www.robosoft.fr/simulations/services/simulatedcardrive.html

CarDrivePartner
CarDrivePartner
http://www.robosoft.fr/simulations/services/cardrivepartner.html

SimulatedLaserSensor
SimulatedLaserSensor
http://www.robosoft.fr/simulations/services/simulatedlasersensor.html

SimulationEngine
Simulation Engine
http://schemas.microsoft.com/robotics/2006/04/simulationengine.html

Odometry2Localization
Odometry2Localization
http://www.robosoft.fr/2006/09/odometry2localization.html

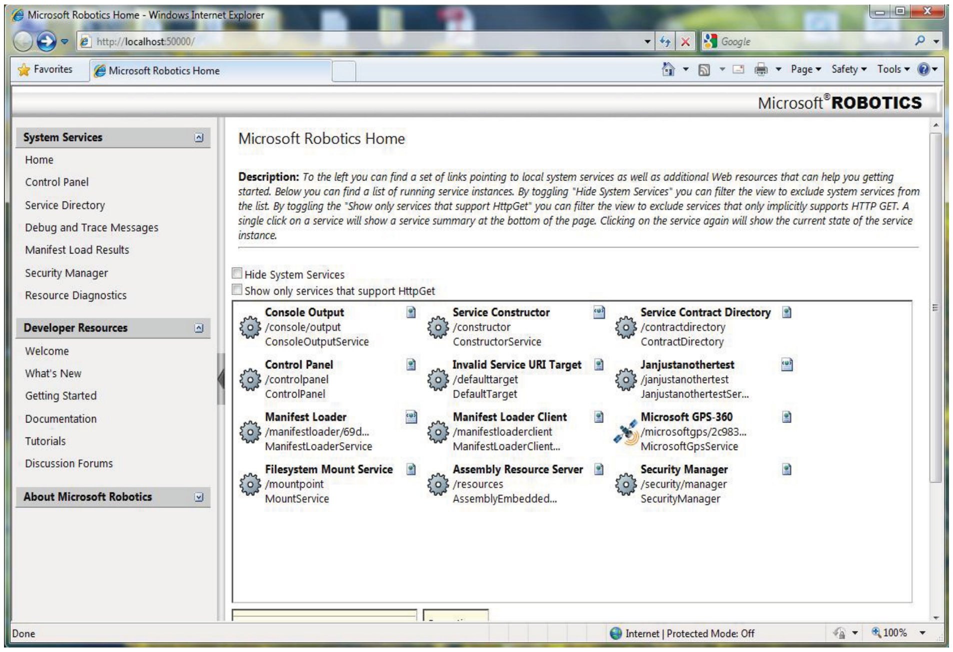
The Odometry2Localization Service

Name: SimulationEngine
Service Path:
Settings:

Partners	
ManifestLoaderClient	Use service's partner de
SimulationState	
Gravity	
RenderMode	Full
FramesPerSecond	60.5358429
Pause	☐
CameraPosition	
CameraLookAt	
CameraFieldOfView	0.781
CameraNearPlane	0.1
CameraFarPlane	5000
Lights	[Count 0]
SerializedEntities	

Figure 2.8: Manifest with Sample Services

services and manifests are active and where they are active. The different nodes can be remotely monitored using the same. The specifics related to this work are discussed in the next chapter.



Microsoft Robotics Home

Description: *To the left you can find a set of links pointing to local system services as well as additional Web resources that can help you getting started. Below you can find a list of running service instances. By toggling "Hide System Services" you can filter the view to exclude system services from the list. By toggling the "Show only services that support HttpGet" you can filter the view to exclude services that only implicitly supports HTTP GET. A single click on a service will show a service summary at the bottom of the page. Clicking on the service again will show the current state of the service instance.*

☐ Hide System Services
☐ Show only services that support HttpGet

Console Output
/console/output
ConsoleOutputService

Service Constructor
/constructor
ConstructorService

Service Contract Directory
/contractdirectory
ContractDirectory

Control Panel
/controlpanel
ControlPanel

Invalid Service URI Target
/defaulttarget
DefaultTarget

Janjustanothertest
/janjustanothertest
JanjustanothertestSer...

Manifest Loader
/manifestloader/69d...
ManifestLoaderService

Manifest Loader Client
/manifestloaderclient
ManifestLoaderClient...

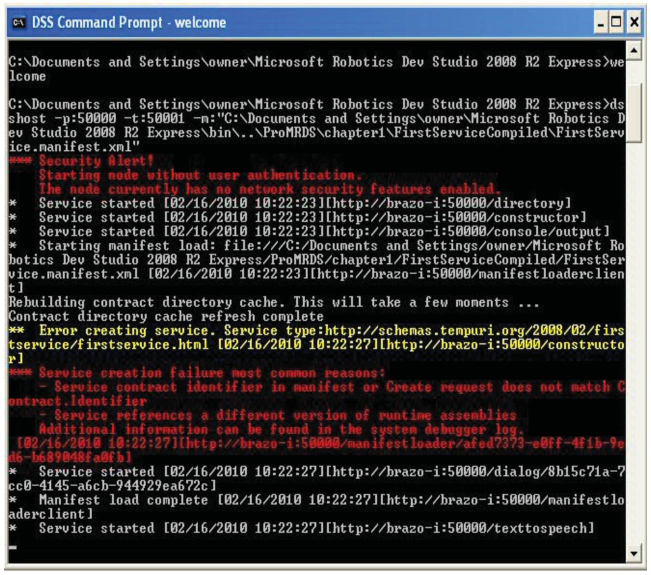
Microsoft GPS-360
/microsoftgps/2c983...
MicrosoftGpsService

Filesystem Mount Service
/mountpoint
MountService

Assembly Resource Server
/resources
AssemblyEmbedded...

Security Manager
/security/manager
SecurityManager

Figure 2.9: Web-inerface to Monitor Services



```
C:\Documents and Settings\owner\Microsoft Robotics Dev Studio 2008 R2 Express>we
lcome

C:\Documents and Settings\owner\Microsoft Robotics Dev Studio 2008 R2 Express>ds
shost -p:50000 -t:50001 -m:"C:\Documents and Settings\owner\Microsoft Robotics D
ev Studio 2008 R2 Express\bin\..\ProMRDS\chapter1\FirstServiceCompiled\FirstServ
ice.manifest.xml"
*** Security Alert!
    Starting node without user authentication.
    The node currently has no network security features enabled.
*   Service started [02/16/2010 10:22:23][http://brazo-i:50000/directory]
*   Service started [02/16/2010 10:22:23][http://brazo-i:50000/constructor]
*   Service started [02/16/2010 10:22:23][http://brazo-i:50000/console/output]
*   Starting manifest load: file:///C:/Documents and Settings/owner/Microsoft Ro
botics Dev Studio 2008 R2 Express/ProMRDS/chapter1/FirstServiceCompiled/FirstSer
vice.manifest.xml [02/16/2010 10:22:23][http://brazo-i:50000/manifestloaderclien
t]
Rebuilding contract directory cache. This will take a few moments ...
Contract directory cache refresh complete
**  Error creating service. Service type:http://schemas.tempuri.org/2008/02/firs
tservice/firstservice.html [02/16/2010 10:22:27][http://brazo-i:50000/constructo
r]
*** Service creation failure most common reasons:
    - Service contract identifier in manifest or Create request does not match C
ontract.Identifier
    - Service references a different version of runtime assemblies
    Additional information can be found in the system debugger log.
 [02/16/2010 10:22:27][http://brazo-i:50000/manifestloader/afed7373-e0ff-4f1b-9e
d6-b689048fa0fb]
*   Service started [02/16/2010 10:22:27][http://brazo-i:50000/dialog/8b15c71a-7
cc0-4145-a6cb-944929ea672c]
*   Manifest load complete [02/16/2010 10:22:27][http://brazo-i:50000/manifestlo
aderclient]
*   Service started [02/16/2010 10:22:27][http://brazo-i:50000/texttospeech]
_
```

Figure 2.10: MRDS Console

2.5 Problem Formulation

The purpose of various sensors in an AV is to provide details (data) about its surrounding for navigation and its location. The location is provided in most cases by a GPS but due to the low accuracy as discussed earlier it is always accompanied by other sensors to improve on the location information. The cost of dGPS is higher and there is less research done using this system. When adding more sensors each sensor output needs to be monitored and integrated to make it work. This demands more computation. The AV's in DARPA challenge (Fig. 1.3) had a rack of computers in the boot for processing the sensor data in realtime. The digital map is a database of information that relates to any given point on the map. The proposed system works by creating a digital map for AV. The concept is the digital map provides the data required to navigate using a dGPS of higher accuracy.

The objective of the work is to determine if the data from the tailored map is enough to navigate the vehicle. The targeted application of the design of this system is for a short commute like an university transit (Fig. 2.11). The components of the work include a car like steering vehicle, a dGPS, a laser range finder, and a map. The laser does not interfere with the calculation of steering, it slows down and stops on obstacle in the path. The vehicle steers using its front wheels only and hence the system is intended to work with any vehicle with similar dynamics.

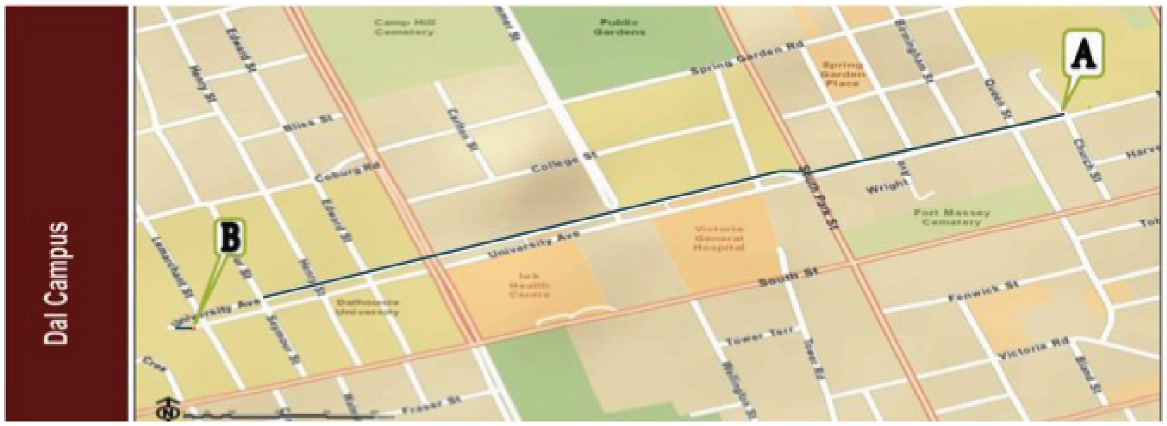


Figure 2.11: Sample Application

Assumptions

Use of dGPS with no GPS loss and the accuracy of the position is in the order of 0.6 m. The path is created by map at the start and it does not compute path when AV is in motion. The direction (side) of travel in road is based on the path from the map based on the start and destination positions only. The front scanning laser range finder is used only for collision avoidance. This research is a study of proof of concept of standalone use of map based decision-making using dGPS navigation. For MRDS the assumption is that there are no data loss and delays in communication between the services and the sensors.

2.6 Summary

First we saw what makes the digital map. The potential of the digital map has not been explored extensively so far. In this research the map is developed for AV and its out come is studied. The dGPS properties and its types are discussed in the second section and it has been used in researches in combination with other sensors similar to all non-differential GPS cases. The last but one section discussed the work platform and its properties. The problem is defined and intention is formulated for this work. Then the assumptions are put forward that aid this work.

Chapter 3

Digital Map based Navigation System Design

This chapter explains the creation of the proposed system and operations. The last chapter discussed the components of the system and the reason for the proposed system. Section 3.1 explains how the proposed system operates. Section 3.2 discusses the graphical user interface and the available interactions. Section 3.3 discuss the software usage in creating the basis of the testing apparatus.

3.1 System Architecture

This section explains the overall work of the proposed scheme. The way in which this work is setup is shown in Fig. 3.1, the software layer that encompass this work is MRDS. The services and how they are connected to the hardware is shown in Fig. 3.2. The service with the Graphical User Interface (GUI) is created for this work and it contains the map which allows for interaction. The GUI communicates with the vehicle and the GPS through MRDS and the map database. Each of these components are explained in this chapter.

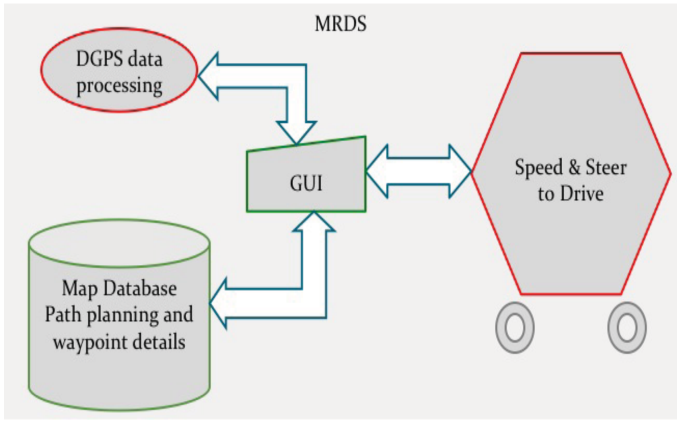


Figure 3.1: System Archictecture

Each work requires a process plan and the process by which the workflow occurs in this work is explained using the flowchart Fig. 3.3. The manifest is started and

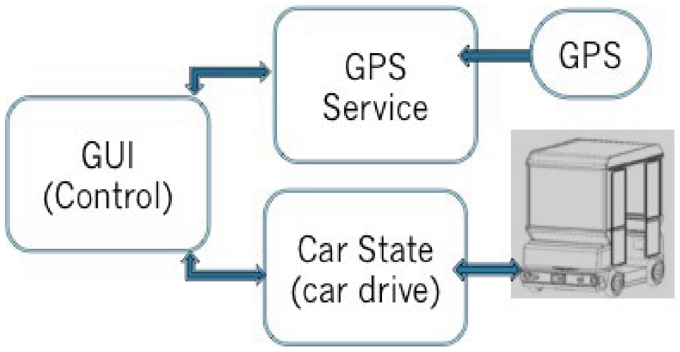


Figure 3.2: Services and Hardware Communication

the initialization is done as explained later in this chapter. After the initialization this is the flow of work that is been used in this research. At first the user selects a

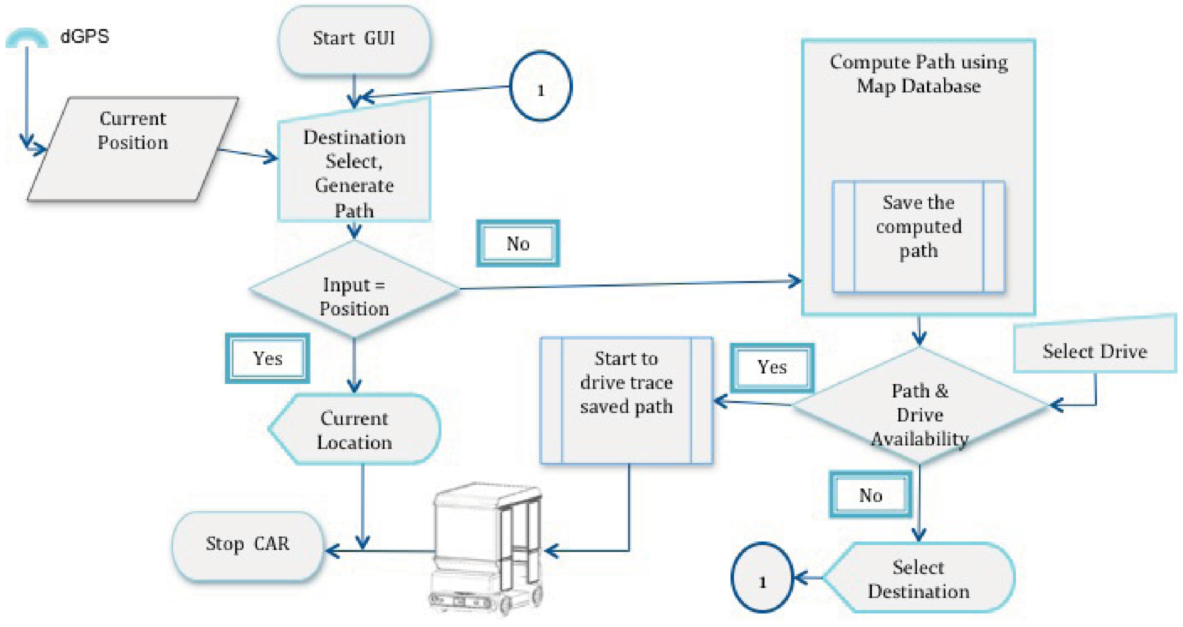


Figure 3.3: Process Flow

destination on the map and the position of the preferred destination is taken along with the current position from the dGPS for path planning. If the user selects the path planning without selecting a destination the system will return an error message and ask to pick a destination. Check if the current position is same as destination if found true the vehicle does not move and if the positions are not same then the control goes to path planning and if the positions start and end are available then the path is planned. The user selects to drive then control checks for the availability of a path if false then displays select destination. If the condition is true then the path following method explained next is used to trace the path to the destination while checking with the current position and when the current position is same as the destination the AV stops.

3.2 User Interface

This is designed for the user interaction; the man machine interface/ graphical user interface for user to provide inputs to the system like destination and certain controls for the user. The windows form is used for creating the interface, but there are certain differences from a normal windows application. The pane which displays the map is from the plug-in tools provided with ArcGIS developer kit. In order to use the control the license initializer is required for establishing the control with the ArcGIS architecture and the availability of ArcGIS installed in the device that is being used. The methods used for making this plug-in controls to work as a standalone is advanced programming that requires direct hardware level coding for authorization and stable working. The other user controls are from basic toolkit in form designer.

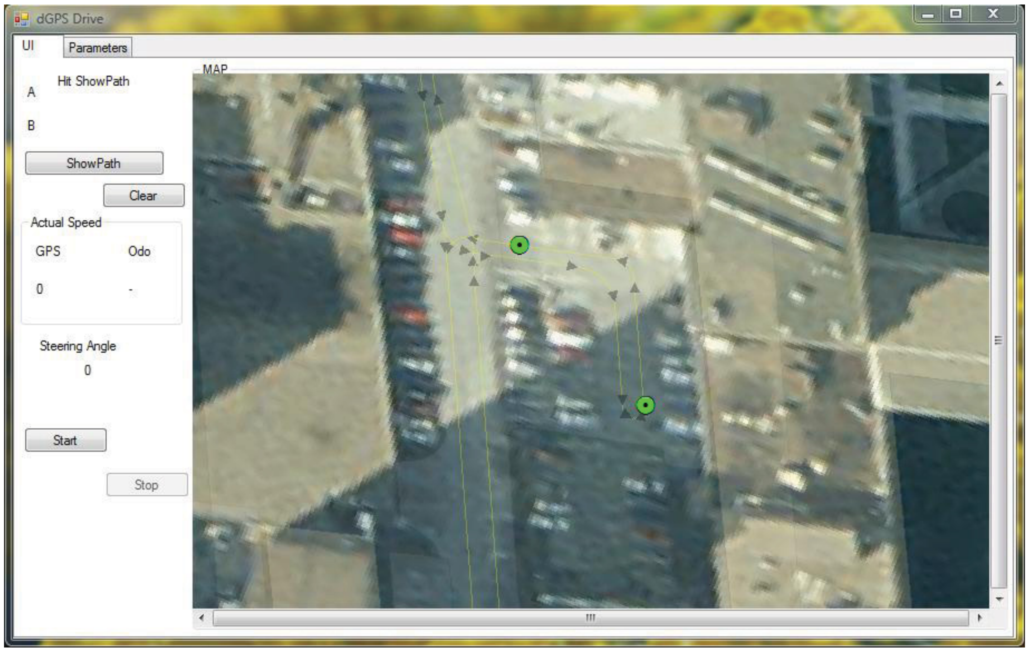
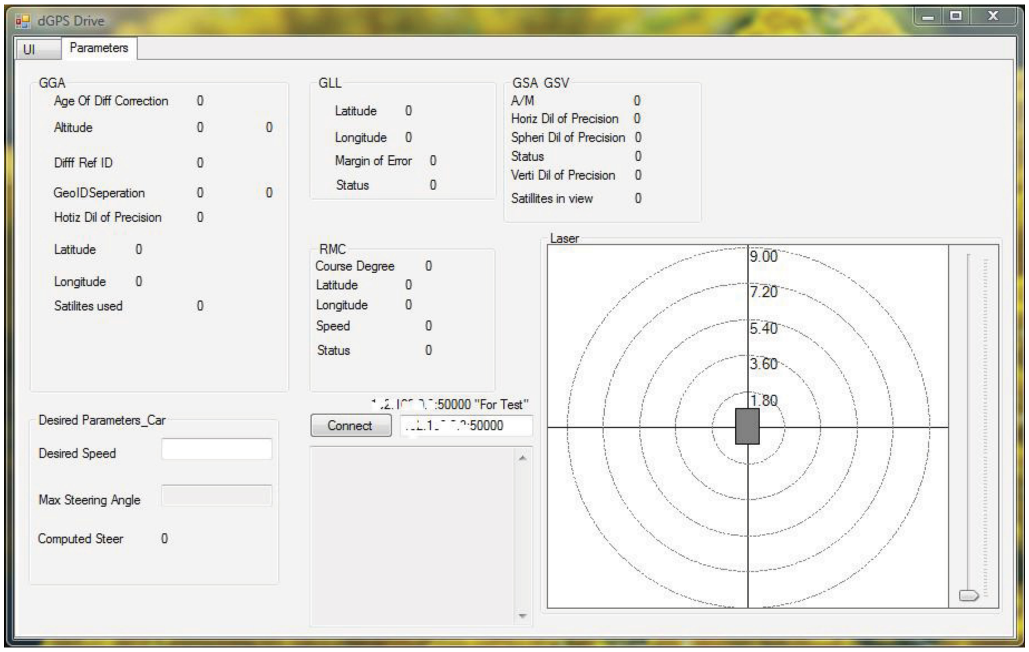


Figure 3.4: Graphical User Interface

The map pane has two controls in this work, one: the pan, zoom, and two: select using single mouse click Fig. 3.4. This single mouse click is used to select destination and get the values of the point of selection in map units. The pan and zoom allows the map to be moved to locate the destination. The position values are displayed in the text box and the map refreshes to display the point selected with the marker. The position of the GPS is displayed as static member in the map and could not be updated since the source is read from MRDS service and not ArcGIS plug-in tools. The buttons provide user to initiate or terminate events. The first button on the top near the test boxes is used to compute the path for the given positions. On successful creation of path a message is displayed for conformation since the path is not displayed on the map, the reason being the programming requirement for graphics hardware control. The two buttons in the lower left are for initiating drive and stop. The drive button makes the program to calculate the steering and send drive commands to the vehicle until it reaches the destination. The stop can be used if the user decides to stop at a prior point.



dGPS Drive

UI | Parameters

GGA

Age Of Diff Correction	0	
Altitude	0	0
Difff Ref ID	0	
GeoIDSeperation	0	0
Hotiz Dil of Precision	0	
Latitude	0	
Longitude	0	
Satilites used	0	

GLL

Latitude	0
Longitude	0
Margin of Error	0
Status	0

GSA GSV

A/M	0
Horiz Dil of Precision	0
Spheri Dil of Precision	0
Status	0
Verti Dil of Precision	0
Satillites in view	0

RMC

Course Degree	0
Latitude	0
Longitude	0
Speed	0
Status	0

Desired Parameters_Car

Desired Speed	
Max Steering Angle	
Computed Steer	0

[redacted]:50000 "For Test"

Connect [redacted]:50000

Figure 3.5: Graphical User Interface Communication

The second tab on the UI is used for communication with the other services and initializing other services Fig. 3.5. The text box is used to enter the remote port where the hardware is located that is to be controlled by this manifest. The devices in the remote port must be powered on and must be available for remote control. The onboard computer has a manifest that turns the devices on and is used for controlling the vehicle form the onboard computer and the remote host. The large text box below displays the messages from the devices if the connection is success or failure. On successful connection the GPS values are displayed in the NMEA message display sections. The laser range finder displays dynamic readout on a successful connection. The max steer values are read from the vehicle state and the steering angle from the car is read from the localization service.

3.3 Software Development

The Map is developed using the ArcGIS software. This is a widely used professional map development tool, used by government for managing various geographic resources. Map created for this work is started using this satellite image in Fig. 3.6 by creating the road feature with respect to the navigable regions in the parking lot. The road feature is drawn using the line tools in ArcGIS.

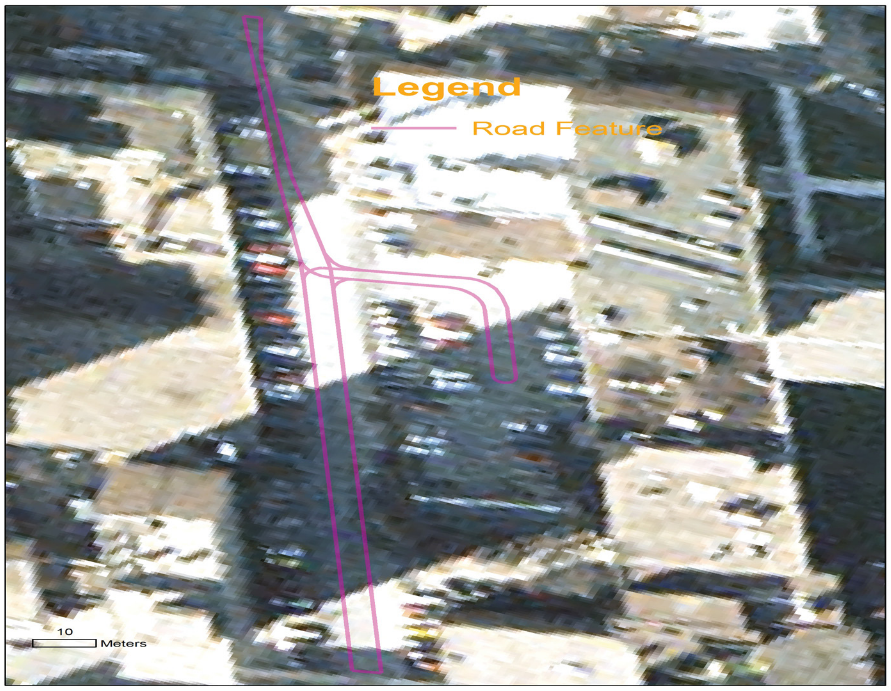
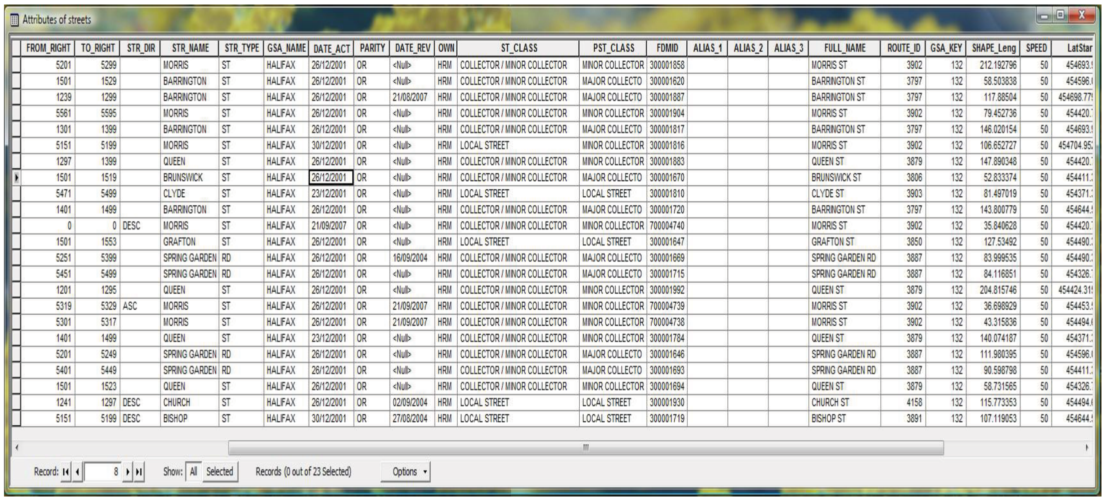


Figure 3.6: Map with Road Feature

The road feature must be connected by the vertex of the preceding line so it will be able to generate a path. In the event of disconnected network the path cannot be created. The map shows the road feature that is used in this work Fig. 3.6. The next step is to create the dataset which is the feature that is used in path planning. To create the network dataset the road feature needs to be provided with certain details. All the features created in the map have an attribute table in Fig. 3.7.


Attributes of streets

FROM_RIGHT	TO_RIGHT	STR_DIR	STR_NAME	STR_TYPE	GSA_NAME	DATE_ACT	PARITY	DATE_REV	OWN	ST_CLASS	PST_CLASS	FDMID	ALIAS_1	ALIAS_2	ALIAS_3	FULL_NAME	ROUTE_ID	GSA_KEY	SHAPE_Leng	SPEED	LatStar
5201	5299		MORRIS	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	300001858				MORRIS ST	3902	132	212.192796	50	454693.
1501	1529		BARRINGTON	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001620				BARRINGTON ST	3797	132	58.503838	50	454596.
1239	1299		BARRINGTON	ST	HALIFAX	26/12/2001	OR	21/08/2007	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001887				BARRINGTON ST	3797	132	117.88504	50	454698.77
5561	5595		MORRIS	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	300001904				MORRIS ST	3902	132	79.452736	50	454420.
1301	1399		BARRINGTON	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001817				BARRINGTON ST	3797	132	146.020154	50	454693.
5151	5199		MORRIS	ST	HALIFAX	30/12/2001	OR	<Null>	HRM	LOCAL STREET	MINOR COLLECTOR	300001816				MORRIS ST	3902	132	106.652727	50	454704.95
1297	1399		QUEEN	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	300001883				QUEEN ST	3879	132	147.890348	50	454420.
1501	1519		BRUNSWICK	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001670				BRUNSWICK ST	3806	132	52.833374	50	454411.
5471	5499		CLYDE	ST	HALIFAX	23/12/2001	OR	<Null>	HRM	LOCAL STREET	LOCAL STREET	300001810				CLYDE ST	3903	132	81.497019	50	454371.
1401	1499		BARRINGTON	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001720				BARRINGTON ST	3797	132	143.800779	50	454644.
0	0	DESC	MORRIS	ST	HALIFAX	21/09/2007	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	700004740				MORRIS ST	3902	132	35.840628	50	454420.
1501	1553		GRAFTON	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	LOCAL STREET	LOCAL STREET	300001647				GRAFTON ST	3850	132	127.53492	50	454490.
5251	5399		SPRING GARDEN	RD	HALIFAX	26/12/2001	OR	16/09/2004	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001669				SPRING GARDEN RD	3887	132	83.999535	50	454490.
5451	5499		SPRING GARDEN	RD	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001715				SPRING GARDEN RD	3887	132	84.116851	50	454326.
1201	1295		QUEEN	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	300001992				QUEEN ST	3879	132	204.815746	50	454424.31
5319	5329	ASC	MORRIS	ST	HALIFAX	26/12/2001	OR	21/09/2007	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	700004739				MORRIS ST	3902	132	36.698929	50	454453.
5301	5317		MORRIS	ST	HALIFAX	26/12/2001	OR	21/09/2007	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	700004738				MORRIS ST	3902	132	43.315836	50	454494.
1401	1499		QUEEN	ST	HALIFAX	23/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	300001784				QUEEN ST	3879	132	140.074187	50	454371.
5201	5249		SPRING GARDEN	RD	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001646				SPRING GARDEN RD	3887	132	111.980395	50	454596.
5401	5449		SPRING GARDEN	RD	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MAJOR COLLECTO	300001693				SPRING GARDEN RD	3887	132	90.598798	50	454411.
1501	1523		QUEEN	ST	HALIFAX	26/12/2001	OR	<Null>	HRM	COLLECTOR / MINOR COLLECTOR	MINOR COLLECTOR	300001694				QUEEN ST	3879	132	58.731565	50	454326.
1241	1297	DESC	CHURCH	ST	HALIFAX	26/12/2001	OR	02/09/2004	HRM	LOCAL STREET	LOCAL STREET	300001930				CHURCH ST	4158	132	115.773353	50	454494.
5151	5199	DESC	BISHOP	ST	HALIFAX	30/12/2001	OR	27/08/2004	HRM	LOCAL STREET	LOCAL STREET	300001719				BISHOP ST	3891	132	107.119053	50	454644.

Record: 8 Show: All Selected Records (0 out of 23 Selected) Options

Figure 3.7: Road Feature Attribute Table

This attribute table for a road feature has the address the speed limits and the type of road (navigable 'oneway'). The title of the columns that need to be used for path should follow formats for the application to create the proper output. This needs to be set properly depending on the requirement for the pathplannig to work. Once the attribute table is prepared then the network dataset can be created. The ArcCatalog is a support software of ArcGIS, in the location of the road feature file select the file and from the context menu select the created new network data. This creates the network data for the selected feature and is shown in Fig. 3.8. that include the direction of the network. This means the path is created in directions that can be driven on the correct side of the road.

The start point and the end point determines the path direction. The satellite image shown in the first two figures were calibrated to use with WGS and the image is stretched to match the accuracy of the GPS. Fig. 3.9 is a clear image that is used to show the results for easy visibility. This does not change the values of the position and the results in anyway, but the there is a small offset of the image in Fig. 3.9.

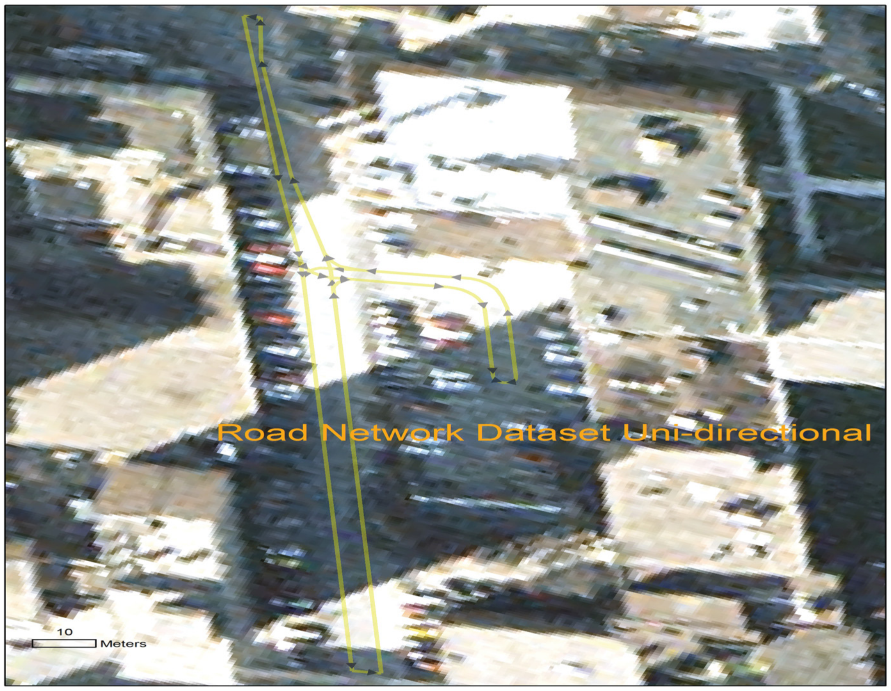


Figure 3.8: Map with Road Network Dataset

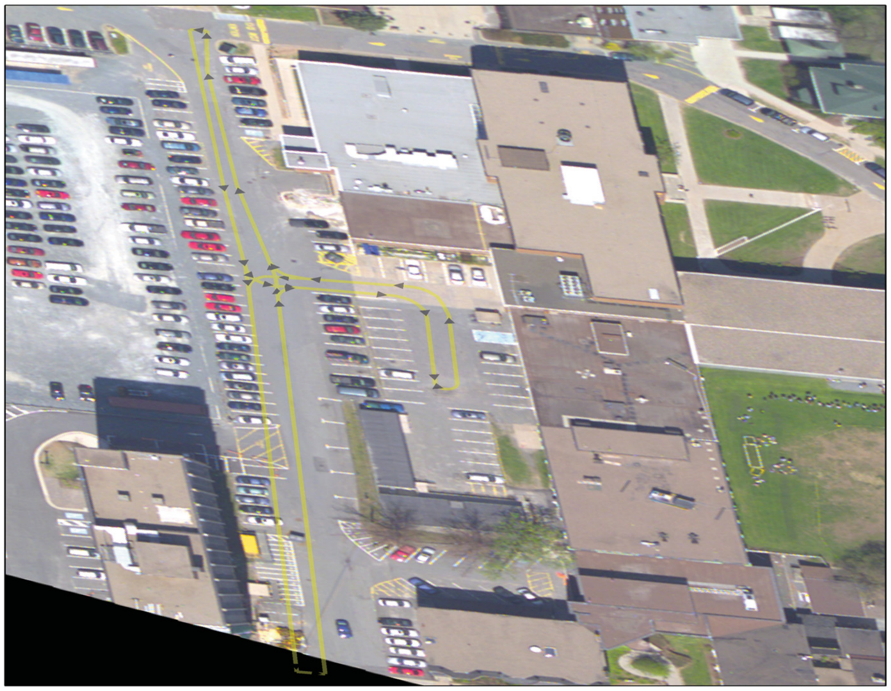


Figure 3.9: Map with Road Feature with clear Image Background

Throughout this work the satellite view of the map is used.

The MRDS is the brain of the vehicle used in this work. The GUI created for this work is built within MRDS. A service is created for connecting to the GPS service and connect to vehicle and control from this interface. Fig. 3.4 shows the UI in which the map is displayed and can be interacted with. The current position is obtained from the dGPS and the user inout is the destination. The map although works from within the service it does not use any MRDS component. The map is operated as a simple C# program component since the tools are not available for use with MRDS. The map generates path on valid start and end and is saved, and these waypoints are received into the main service for navigating vehicle (Code 3.1). The UI 'Forms' are by default not supported by MRDS, but they can be modified to use with MRDS adding Code 3.2 in the form and adding Code 3.3 in the main service. A communication port between the form and the main service must be created for form to send events to the main-service and vice-versa.

Code 3.1: Recieves Waypoints

```
       /// <summary>
       /// receives new route as string[]
       /// </summary>
     void RouteHandler(rt _r)
        {
            rout = _r.rou;
        }
```

Code 3.2: Form as MRDS Component in Form

```
using Microsoft.Ccr.Core;
using Microsoft.Dss.Core;
using Microsoft.Dss.ServiceModel.Dssp;
```

Code 3.3: Form as MRDS Component in Main Service

```
using System.Windows.Forms;
using Microsoft.Ccr.Adapters.WinForms;
```

The map components are added using the tools provided to create plug-ins for the ArcGIS software and its working requirement are discussed in last chapter. The map

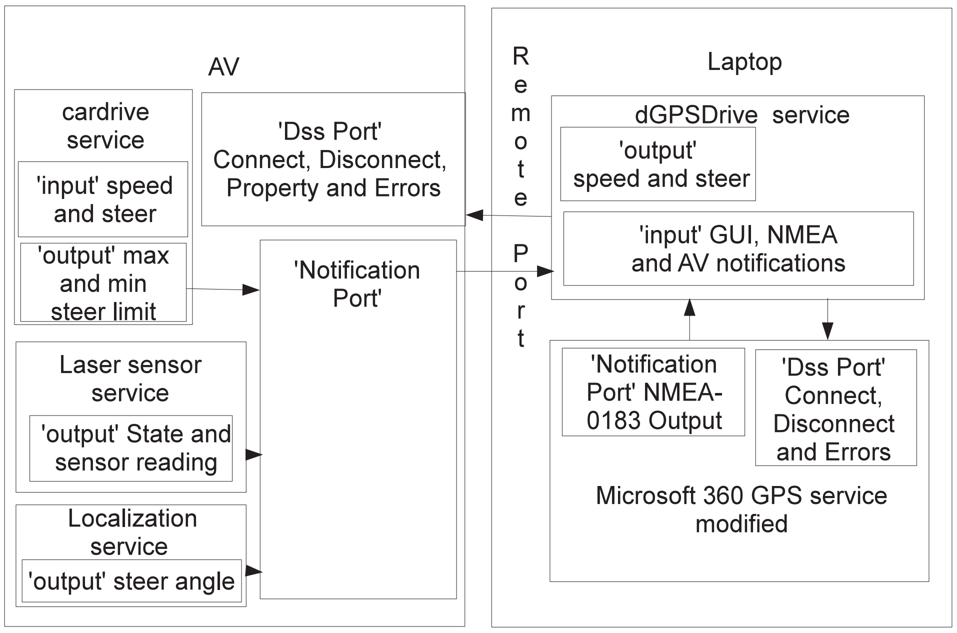

AV
cardrive service
'input' speed and steer
'output' max and min steer limit
'Dss Port' Connect, Disconnect, Property and Errors
'Notification Port'
Laser sensor service
'output' State and sensor reading
Localization service
'output' steer angle
Remote Port
Laptop
dGPSDrive service
'output' speed and steer
'input' GUI, NMEA and AV notifications
'Notification Port' NMEA-0183 Output
'Dss Port' Connect, Disconnect and Errors
Microsoft 360 GPS service modified

Figure 3.10: Services Developed and Data Transmitted using MRDS Ports

display component, the path planning component, the interaction on the map and mouse input for destination are based on the some examples from the developer help file of ArcGIS. For this work the path planning component is modified to provide the waypoints as string for use with the main service. The path planning code inputs are modified to read one value from the mouse click and the other from the dGPS as shown in Code 3.4. The first part within the block region is the code that works with the map architecture and sets the position input as the feature to use for path planning. The path as waypoints are sent to the main service using MRDS in the last part of the code.

Code 3.4: Path Planning Event

```
private void PathPlng_Click(object sender, EventArgs e)
{
 #region set currentgpspoint as geometry in stops
 // Open the feature workspace, input feature class, and
//network dataset
IWorkspaceFactory workspaceFactory =
 new ShapefileWorkspaceFactoryClass();
IFeatureWorkspace featureWorkspace =
workspaceFactory.OpenFromFile(SHAPE_WORKSPACE, 0)
 as IFeatureWorkspace;
// sets stops as input fc
IFeatureClass inputStopsFClass =
 featureWorkspace.OpenFeatureClass(INPUT_STOPS_FC);
// for the second row update
ITable table;
table = inputStopsFClass as ITable;
IRow row;
int j = 0;
// find the field to be updated
 j = table.FindField("Shape");
//get the row of objectID(OID) "0"
row = table.GetRow(0);
//update value of the field in the present row
IPoint crnptn = new PointClass();
crnptn.Y =Convert.ToDouble(_clat);
crnptn.X=Convert.ToDouble(_clong);
row.set_Value(j,crnptn as IGeometry);
cLat = crnptn.Y.ToString();
cLon = crnptn.X.ToString();
savedCurrentPointText = cLat + "," + "\n " + cLon ;
label34.Text = savedCurrentPointText;
//store the assigned value
 row.Store();
#endregion
// set stopsfc with new input
// and save the created featureclass for the solveroute()
rout= _rc.SolveRoute();
int r = 0;
 while (r < rout.Length)
 {
   _rpt.WriteLine(rout[r]);
   r++;
 }
_rpt.Close();
rt _rt = new rt();
_rt.rou = rout;
_formPort.Post(_rt);
}
```

After the path planning is done the normal C# operations ends, from here the working code is MRDS. Although the C# is used for programming MRDS the workings are different and MRDS is a programming tool on its own. The previous chapter discussed the ports and the working of the MRDS, here we discuss the communication of each services. The modified GPS service is added as a default partner as shown in Code 3.5. This is helpful as in the manifest there is no need of adding any other services to use the developed system. The services that are required to operate the vehicle and the onboard sensors are added as silent partners (not visible in the manifest) and are connected using the GUI. The MRDS allows services to run on different node (computer) that can be controlled from one which initiates the manifest. In this case the laptop is what controls the vehicles services that run in the vehicle but controlled by laptop. The node must be provided to connect to the specific remote location.

Code 3.5: Setting Partners

```
 // Partner Ports
 private cardrive.CarDriveOperations _carDrivePort;
 private cardrive.Get _cardriveget = new cardrive.Get();
 private laser.LaserSensorOperations _laserPort;
 private laser.LaserSensorOperations _laserNotif =
  new Robosoft.RobuBOX.Generic.
Devices.LaserSensor.Proxy.LaserSensorOperations();
 private loc.LocalizationOperations _locPort;
 private loc.LocalizationOperations _locNotify =
 new loc.LocalizationOperations();
```

```
  /// <summary>
  /// MicrosoftGpsService partner
  /// </summary>
[Partner("MicrosoftGpsService",Contract=gps.Contract.
   Identifier
, CreationPolicy = PartnerCreationPolicy.UseExistingOrCreate)]
  gps.MicrosoftGpsOperations _microsoftGpsServicePort =
new gps.MicrosoftGpsOperations();
  gps.MicrosoftGpsOperations _microsoftGpsServiceNotify =
new Microsoft.Robotics.Services.Sensors
.Gps.Proxy.MicrosoftGpsOperations();
```

The GUI second tab is designed to enter the node for connection. Once the node is known the service are initialized as shown in Code 3.6. When this is active the code request the partners for subscription and if successful reads the state of the service. If there is an error a subscription fail message is sent to the interface to notify user. Also some of the parameters are read from the sensor defaults that are used in this work so not to exceed the physical limits of the device. In this code the GPS connection does not use the uri like the other services because the device is connected to the laptop (localhost) and by default all the services are run in the localhost unless specified. The method IEnumerator is a looping task so these stay active once they are turned on.

Code 3.6: Connecting to Services

```
#region connection to other services
// string uri = "localhost:50000";// for sim
//string uri = "IP:50000";//for test
/// <summary>
/// Lauch services connections
/// </summary>
void ConnectHandler(Connect c)
 {
  _robotUri = "http://" + c.uri + "/directory";
  SpawnIterator<string>(_robotUri, ConnectToCarDriveHandler);
  SpawnIterator<string>(_robotUri,
     ConnectToLocalizationHandler);
  SpawnIterator<string>(_robotUri, ConnectToLaserHandler);
  SpawnIterator(Connect2GpsHandler);
 }
#region Connect and Subscribe to GPS
// request subscription for connecting and getting data from
   GPS
IEnumerator<ITask> Connect2GpsHandler()
{
....
.....
....
}
*Refer Appendix
#endregion

#region Connection and get to CarDrive
// request subscription for connecting to car and getting
   state
IEnumerator<ITask> ConnectToCarDriveHandler(string robotUri)
{
....
.....
....
}
*Refer Appendix
 #endregion

 #region Connection and subscription to laser
// request subscription for connecting and getting data from
   Laser
IEnumerator<ITask> ConnectToLaserHandler(string robotUri)
{
```

```
....
.....
....
}
*Refer Appendix
#endregion

 #region Connect To Localization Handler
// request subscription for connecting and getting
//data from Localization
IEnumerator<ITask> ConnectToLocalizationHandler(string
   robotUri)
{
....
.....
....
}
*Refer Appendix
#endregion
#endregion
```

The connection to the service if successful all the services keep running but does not interfere. Now the focus is on driving the vehicle to follow the path. The GUI triggers the start of drive control. Since the partnership is active, the command required to move the vehicle is send speed and steer values. The syntax Code 3.7 is a one time event, that means that it should be in a loop for the vehicle to have continuous motion. This loop is controlled by the calculation of steering angle. The steering angle calculation is dependent on the current position (dGPS) of the vehicle and the distance to the waypoint. The dGPS feed is high frequency and the values cannot be read on demand but it is continuously feed. The initial tests showed the input speed of the GPS data is too much for the program to handle and the response was chaotic. So the loop speed is now set by the timer (Code 3.8) for thread and this manages the GPS update interval and the drive command is managed at the same rate.

Code 3.7: Send Drive Value to Vehicle

```
// calls the drv
SpawnIterator<double, double>(speed, steer, drv);

/// <summary>
/// Sends spd and str to drive car
/// d is speed and r is steer sent after calculation
// when this is called
/// </summary>
/// <param name="d"></param>
/// <param name="r"></param>
 IEnumerator<ITask> drv(double d, double r)
 {
  cardrive.DriveRequest req = new cardrive.DriveRequest();
  req.Speed = d; req.Steer = r;
yield return Arbiter.Choice<DefaultSubmitResponseType,
W3C.Soap.Fault>(_carDrivePort.Drive(req),
delegate(DefaultSubmitResponseType response)
  {           },
delegate(W3C.Soap.Fault f)
  {
   LogError("Error sending drive consign");
  });        }
```

Code 3.8: Timer and Loop Control

```
//TimeSpan interl =
//TimeSpan.FromMilliseconds(500);// half second
TimeSpan interl =
 TimeSpan.FromMilliseconds(250);// quater second
// in the loop
// waits for the time interval before continuing to next step
    _rset.WaitOne(interl);
```

3.4 dGPS based Localization and Navigation

The dGPS is used in this work is of paramount importance. The GPS has lower accuracy and as discussed in previous chapters it is not suitable for this work. The proposed method depends on the dGPS accuracy. The navigation method used is waypoint path following. This method is chosen because the path provides with waypoints and can be used with this method directly. The vehicle input parameter for driving is speed and steer. Since this work is a prototype and tested in a parking lot access way the speed is set in the program and is modified to observe behaviour. The speed can be read from the path component and can be used for future works. The calculation for steering is done using the following method. The length or distance between the car and the first waypoint (first goal) is calculated. The distance between the car and the path start point and the distance between the path start to the first goal are used. This gives the steering angle for the vehicle. The vehicle travels a set distance before the waypoint index increments. Fig. 3.11 shows the method with a set of waypoints for explanation. The angle θ is given by Equation (3.1)

$$\theta = \cos^{-1}(((l_{23})^2 + (l_{12})^2 - (l_{31})^2)/(2 \times l_{23} \times l_{12})) \tag{3.1}$$

.

Code 3.9 is used to calculate the angle, it gets the three points two waypoints and the current position. The calculated angle is then compared with the maximum permissible angle that the vehicle can take and if the angle calculated is large then the angle is set to the max. value. In this case the max. angle is 0.57 rad but as an added safety its set to 0.56 rad in the program.

Code 3.9: Calculate Angle

```
private Double tt(double[,] a, double[,] c, double[,] r)
 {
 int l = c.Length;
 double lac = Math.Sqrt(Math.Pow(a[0, 0] - c[l - 2, 0], 2) +
Math.Pow(a[1, 0] - c[l - 1, 0], 2));
 double laro = Math.Sqrt(Math.Pow(a[0, 0] - r[0, 0], 2) +
Math.Pow(a[1, 0] - r[1, 0], 2));
 double lcro = Math.Sqrt(Math.Pow(c[l - 2, 0] - r[0, 0], 2) +
Math.Pow(c[l - 1, 0] - r[1, 0], 2));
 double t = Math.Acos(((Math.Pow(lac, 2) + Math.Pow(laro, 2) -
Math.Pow(lcro, 2)) / (2 * lac * laro)));
 _tofile.WriteLine("ang calc:" +t.ToString() );
 if (Math.Round(t, 2) > 0.56)
  {
   t = 0.56;
  }
 else if (Math.Round(t, 2) < -0.56)
  {
   t = -0.56;
  }
  return Math.Abs(t);
 }
```

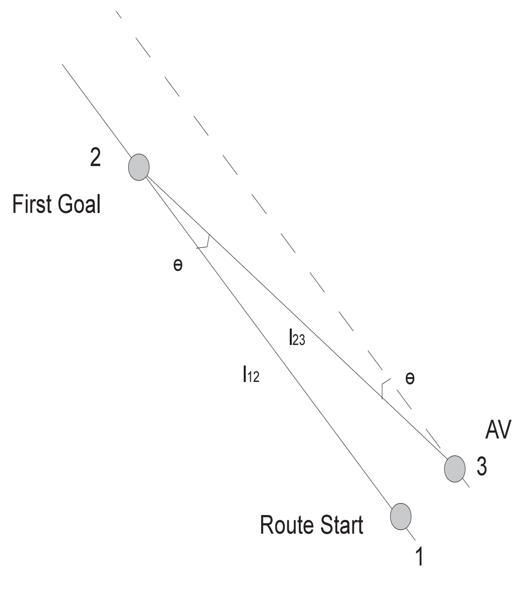


Figure 3.11: Steering angle calculation

The continues line represents part of the path (first segment) and the dotted parallel line is the vehicle's orientation. The line connecting goal and the car is to represent the distance used during the steering angle calculation. The points represent the the waypoints and the car initial position. The point 1 Route start is created by the path planning which is the nearest point on the network from the actual position. Point 3 is the actual position of AV and point 2 is the position of the first waypoint the target. The distance between the points are measured using the haversine equation discussed in the previous chapter. This method is put to test in the simulation and the next chapter demonstrates the results.

3.5 Summary

This chapter explained what the infrastructure of the system and how it's being implemented and used. The development of the program was discussed and the parts of important codes were explained. The chapter also explained the path following and navigation method used.

Chapter 4

Simulation Studies

The simulations conducted for the proposed system use GUI for path generation and Matlab for testing the path following part of the design. The path is generated using the vehicle and dGPS in the test location with the help of GUI. In the path generation process start position is not exactly in line with map path component. So the path generated the first point is closest point from the AV. The start position is replicated in Matlab by the user input using graphical input method to simulate the possible starting position of the actual vehicle. The two driving scenarios are discussed in the following sections.

4.1 Simulation Assumptions

The purpose of the simulation is to test the path following algorithm used. This is to ensure the vehicle doesn't behave chaotic when doing tests. For the purpose of simulation the following assumptions are made. The model chosen is a particle based for vehicle is a differential drive as shown in Fig. 4.1. The new position is calculated using the current position, velocity and heading angle as shown in Equation (4.1) and the same in Matlab (Code 4.1). The angle calculated by Equation (3.1) is relative and to get the new position the absolute angle is used. The value of v in the Matlab is in degrees as the plot values and waypoints are in degrees. Finally initial heading of the vehicle is as the same direction of the path.

$$x_{n+1} = x_n - v \times \cos\omega \quad \& \quad y_{n+1} = y_n + v \times \sin\omega \tag{4.1}$$

Code 4.1: Position Update in Matlab

```
xc(l+1)=xc(l)- 0.000006*cos(pteta(l));
yc(l+1)=yc(l)+ 0.000006*sin(pteta(l));
```

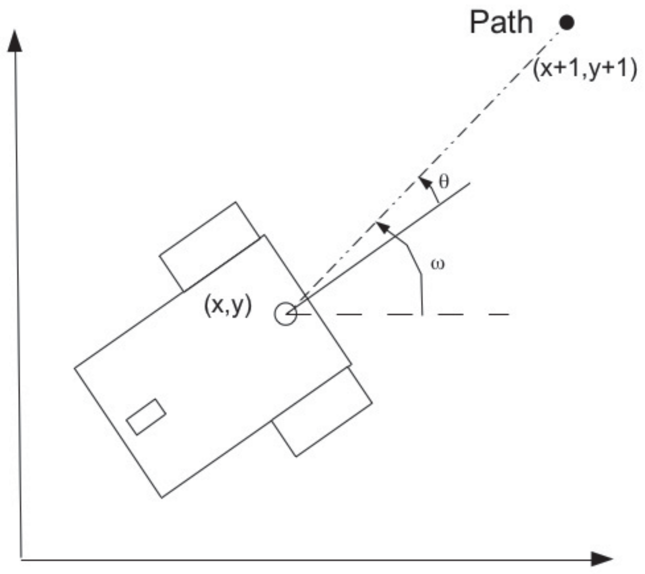


Figure 4.1: Simulation Model

4.2 Simulation: Straight Line Case

The first simulation is the case with a straight line path. The AV need not be in line with the path, but it merges into the path and reach destination. This simulation case has two start positions, starting on both sides for path. Fig. 4.2 shows the path with waypoints marked in 'o' and the path taken by the car denoted as '+'. Fig. 4.3 shows the path and the path taken by the car when it started on the right side of the path. The simulation result from Fig. 4.2 shows the case where the car's initial position is to the left of the path. The car locks on to the first waypoint as its goal and computes the steering to reach the goal. The actual car has a maximum steering angle input of 0.57 radians or 32.66 degrees and so in the simulation the computed angle is limited by the set limit of 0.5 radians. As you can see the coordinates of the plot are degrees since the unit of GPS and map as explained in the coordinate system in chapter II. Also the plot does not show the true length (decimal values) of the values used for calculation. For better understanding the same is plotted using meters in Fig. 4.4 where its easy to see the distance between waypoints. This is

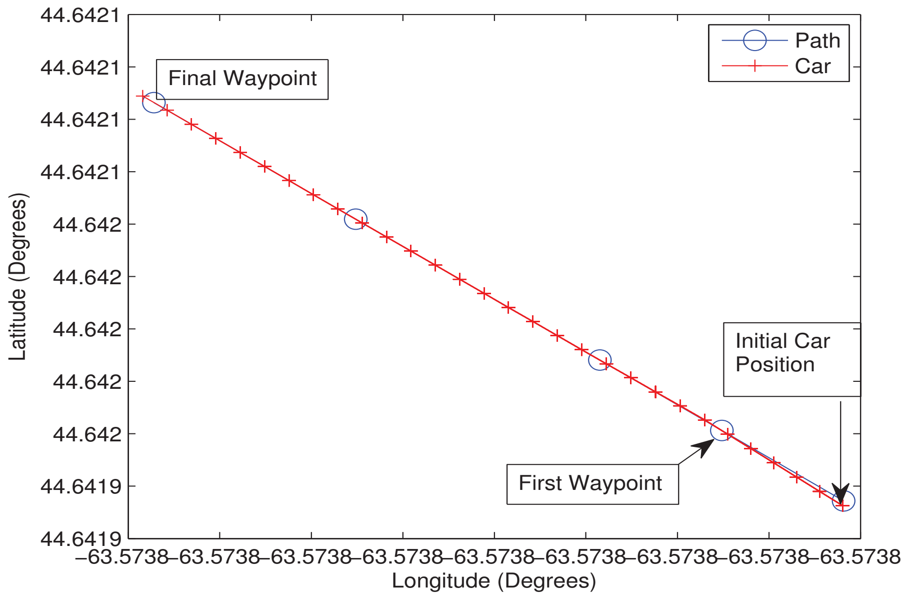


Figure 4.2: Straight Line case with the Start Position on the Left

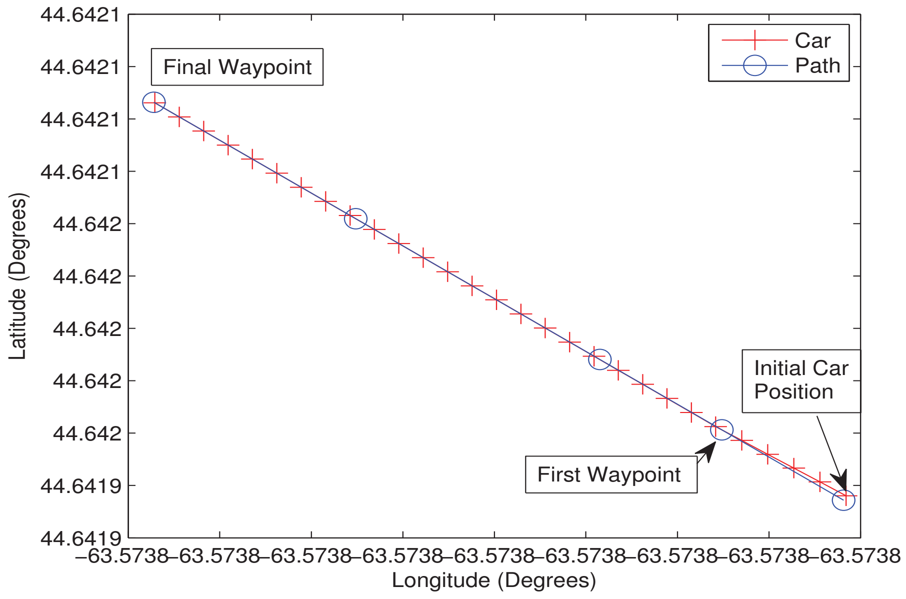


Figure 4.3: Straight Line case with Start Position on the Right

achieved by using Haversine equations.

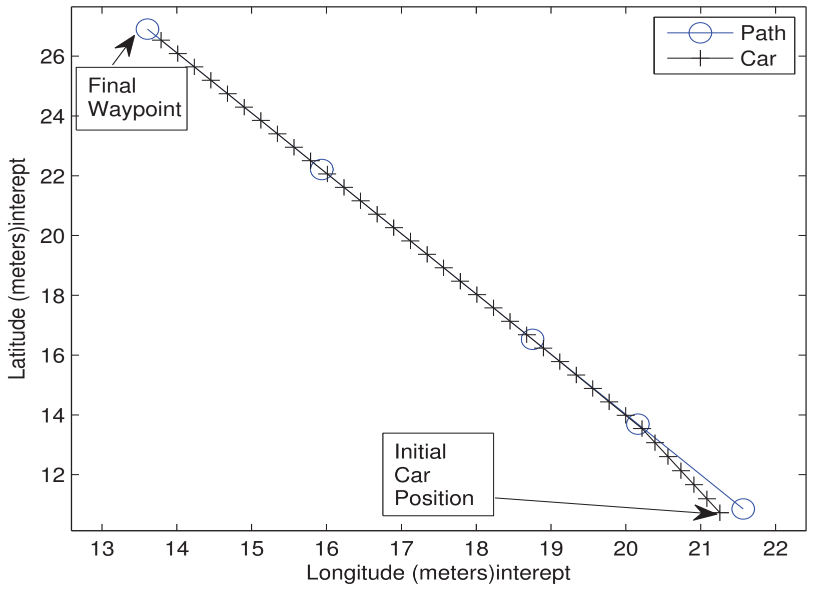


Figure 4.4: Straight Line case in Meter

4.3 Simulation: Curvilinear Case

This case is to simulate the AV following a path with a turn. The simulation parameters are same as before and the start points are defined in the same way.

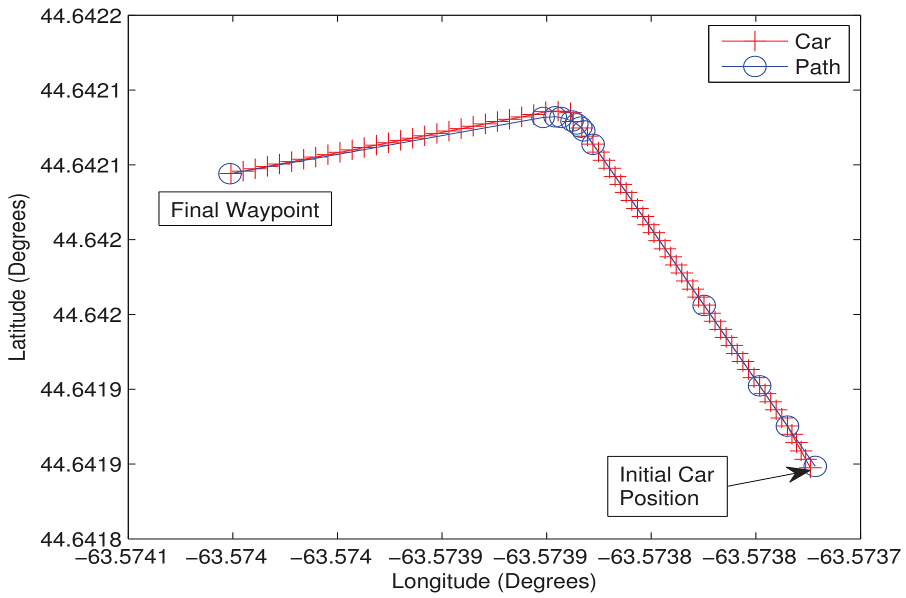


Figure 4.5: Curvilinear case with the Start Position Left of Path

The path consist of a left turn. Here the turn is defined by more number of waypoints at close proximity. This multiple waypoints help the AV navigate in a curve. Fig. 4.7 shows the path of the car and the path to be followed. In this work the speed of the car is same for straight line and curve since the speed is low. In general the speed for turn is for AV is split by range in speed to radius of the turn this can be applied for future works if it involves variable speeds. The simulation of the path with curvilinear profile is shown in Fig. 4.5 is for the AV starting in the left side of the path. Similar to the straight line test the AV merges in to the path and follows through the turn but since the waypoints are closer representing the turn, the AV position deviates from the path and then tracks back to fall in with the path before reaching the destination. Fig. 4.6 shows the result of the simulation for AV

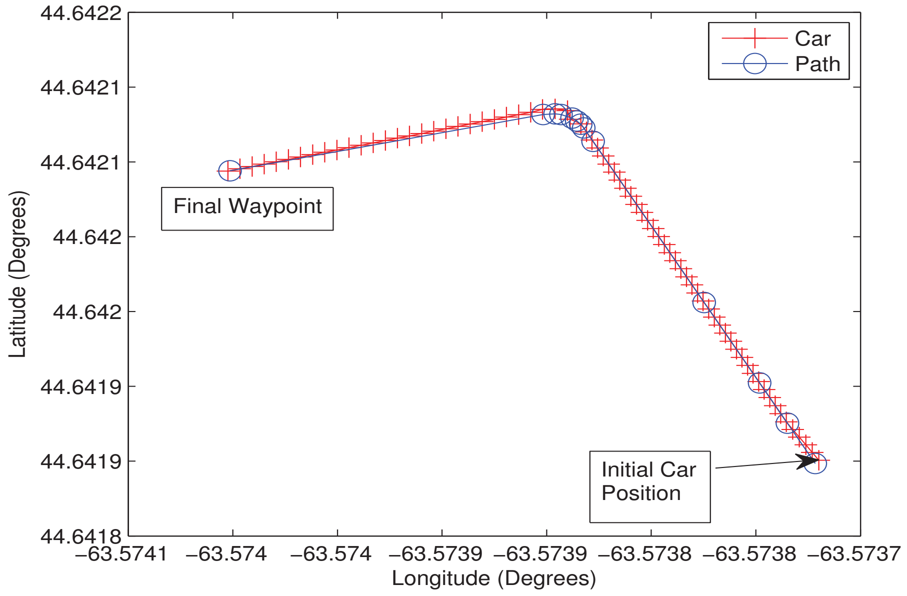


Figure 4.6: Curvilinear case with the Start Position Right of Path

start position on the right to the path. It shows similar behaviour that of the turn simulation with start in left of the path. The AV merges in to the path until it gets to the turn where it deviates due to the closeness of the waypoints and merges in to the path.

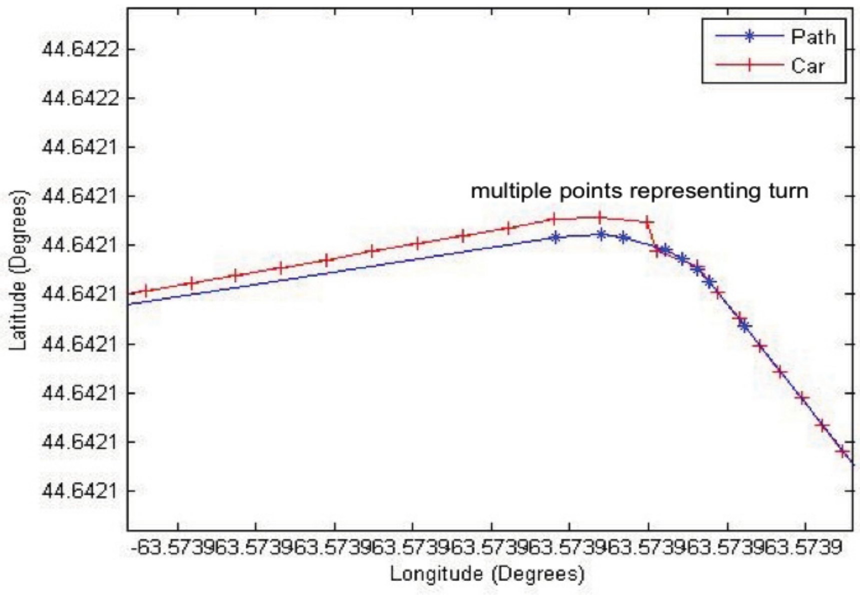


Figure 4.7: Zoomed Profile of the Turning Point

Fig. 4.7 shows the zoomed view of the simulation in the path at the curve and the path taken by the simulated AV. The deviation of the AV can be clearly seen here.

4.4 Simulation Results

The Simulation results show that the path following algorithm works with the waypoints generated by the proposed system. The results show the position of AV steadily proceed and merge with planned path. This is because the method used to calculate the next position is theoretical. However in the experiment the position of AV is expected to deviate from the path but closely following and reaching the waypoints. The waypoints represent a small point and the AV's GPS position need not exactly reach the point and this work focuses on the method of using a digital map and the dGPS sensor. The program checks for the nearness of the position of AV to the waypoint as shown in Code 4.2. The while loop calculates the steering and computes the next position and is controlled by the distance 'crdm' of AV to waypoint and the waypoint counter.

Code 4.2: Main Loop Condition

```
while (crdm > .11 || ru<rl )
```

```
.
.
.
         crdm=earthm*ccm;
end %end while
```

4.5 Summary

Based on the simulation results, the experiment is proceeded with the same path following algorithm. The difference would be with the position of AV as it is from the dGPS itself in the experiment and there could be more position points of AV as a result. The next chapter explains the experimental setup and experiment in detail.

Chapter 5

Experimental Studies

In this chapter the experimental setup and the results are presented. The hardware section details the hardware used for the work. The software section details the software that runs the hardware and the control.

5.1 Hardware Setup

The vehicle (Fig. 5.1) used is an electric vehicle with onboard computer that controls all the hardware (motor, sensors).

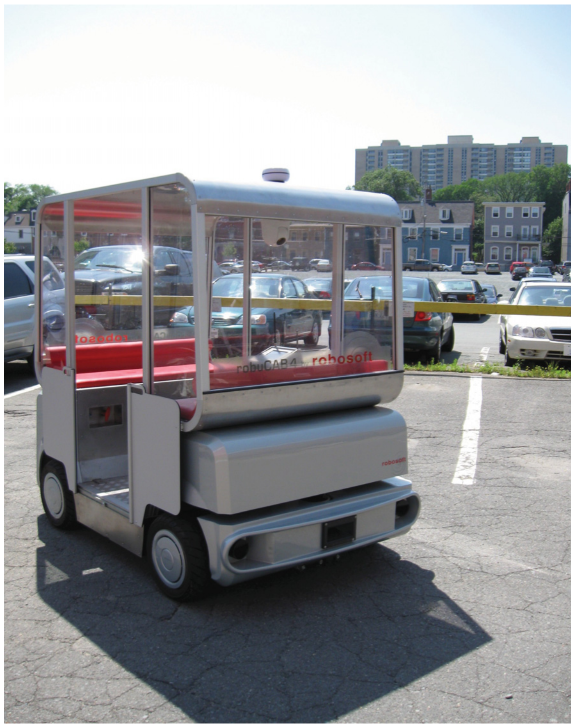


Figure 5.1: RobuCAB the Test Vehicle (ACM Lab @ Dalhousie University)

The vehicle is powered by a bank of battery. This power is used for the onboard embedded computer and the sensors and electric motor. The vehicle can be driven as a differential drive and car like drive. Each wheel has an electric motor that drives the wheels, and it also has four wheel steer. For this work the vehicle is set to drive like car (front wheel steer). The sensors onboard the vehicle are front facing laser range finder, front facing sonar, pan tilt camera. The dGPS used in this work is powered separately and its antenna is placed on the roof of the vehicle and connected to the laptop. The laptop is intel Centrino core 2 Duo"2 GHz" with Windows Vista on"3 GB" RAM. The laptop is connected to vehicle using vehicle's onboard access point and controlled remotely. The input from user is high level for navigation input is speed in meters per second and steer in radians. The maximum speed of the vehicle is 15 kph, and the maximum steering limit is in the range of -0.57 to 0.57 radians, turning radius is 3610 mm, and 2200×1260×19500mm in dimension [24].

The differential GPS sensor (Fig. 5.2) is a hemisphere GPS R110 receiver with antenna. This sensor can be used with different differential source like SBAS, code based, carrier phase based and so on. The differential source used for this work is SBAS differential with WAAS. The unit receives NMEA 0183(Table 5.1)"GGA, GLL, GSA, GST, GSV, RMC, RRE, VTG, ZDA" messages at the rate of 1Hz to 20 Hz. The power source can be anything between 8 and 36 VDC, for this work 18 VDC is used.

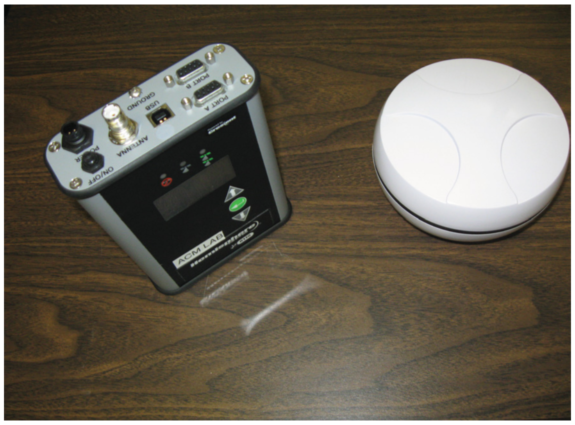


Figure 5.2: dGPS Antenna and Hemisphere Receiver-R 110

Table 5.1: Sample NMEA Messages [1]

Message	Description/Key Data
GPGGA	GPS fix data
GPGLL	Geographic position - latitude/longitude
GPGSA	Global Navigation Satellite System (GNSS) Degree Of Precision and active satellites
GPGST	GNSS pseudo range error statistics
GPGSV	GNSS satellite in view
GPRMC	Recommended minimum specific GNSS data
GPRRE	Range residual message
GPVTG	Course over ground and ground speed

5.2 Software Implementations

Operating System: The AV has an embedded computer running Windows XPe [24] and the sensors and the drive motors that are connected to the car are operated by MRDS services. The control laptop that has the map database and the designed GUI controller is Windows Vista home edition.

Interface Platform: MRDS is the programming language used in creating the control service and the software runtime manages the manifest created containing services required. MRDS acts as a second operating system in AV.

Digital Map: The map used in this work is designed in a map development platform the ArcGIS which is being used by the digital map providers for creating the different layout, features, and other data tweaks. The software ArcGIS also provides a developer kit for .net which helped in this work for integrating the map controls with the user interface.

5.2.1 Location

The test vehicle is not street legal in Canada so the test was conducted in the parking and drive way in the sexton campus, Dalhousie University. The map was modified as discussed in chapter III and the road map is created within the parking area. The AV is driven from the lab through narrow hallway on to the ramp lift out to the parking lot using the onboard joystick that is activated using a service.

5.2.2 Test Procedure

The “RobuCABIntegratedApplicationGUI” manifest is the one that turns on the sensors and monitors the low level controls onboard the test vehicle. The same manifest is used to start onboard drive control and transfer control to the remote computer. In this case the laptop is used to control connected to the vehicle using vehicles wireless access point. The dGPS sensor is mounted on vehicle but its powered by a separate battery source. First the battery is connected to the dGPS sensor and is connected to the laptop, the dGPS sensor is then turned on. The GPS takes few seconds to get differential lock which give the needed accuracy. Then in the vehicle the remote control service is turned on and in the laptop the UI control service is started up. The second tab in the UI is selected and the connection to vehicle and its onboard sensor (laser) services are connected to the control UI service. The UI prints the state of the connection and if successful shows the laser scan and the GPS strings. Switch to the first tab and select the destination, the coordinates of the selected point is displayed in side panel.

5.3 Experimental Results

Here we see the results for two different cases. The tests are conducted with parameter changes and the output is analyzed for each result.

Table 5.2: Tests: Straight Line Case Conditions

Fig	Speed ‘m/s’	Sampling Time	Total Distance ‘m’
5.3	0.7	750‘milliseconds’	16.67
5.4	0.5	‘500 milliseconds’	17.37
5.5	0.3	‘500 milliseconds’	15.54
5.6	0.7	‘250 milliseconds’	18.12
5.7	0.5	‘250 milliseconds’	16.73
5.8	0.3	‘500 milliseconds’	18.07
5.9	0.3	‘500 milliseconds’	18.23

Straight Line Case: The test is conducted for a straight line path. The position of the AV need not be aligned on to the route line. As discussed in last chapter the map produces the path from the nearest point of the AV on the route feature to the nearest point of the destination on route feature. Table 5.2 gives a quick look at the parameters for each result that is explained next.

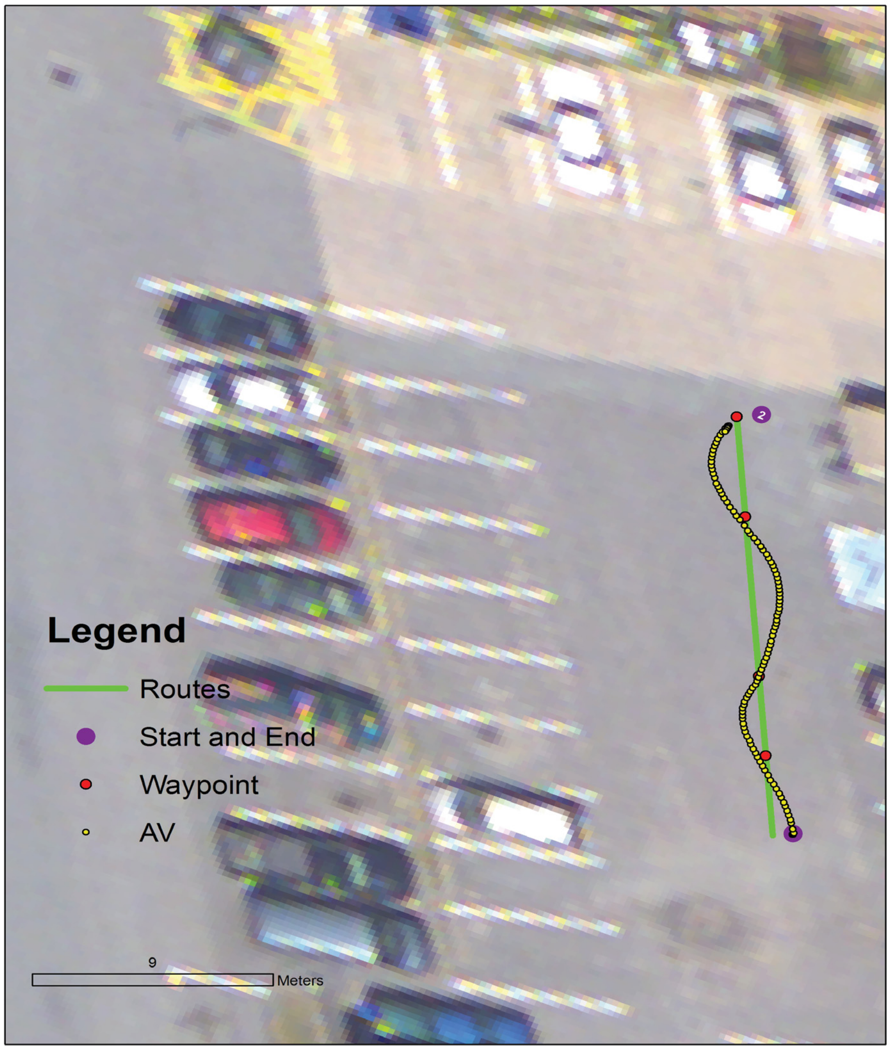


Figure 5.3: Straight Line - Test Result 1

The result shown in Fig. 5.3 has four members and they are the start and end points, the path, the waypoints, the AV position from dGPS sensor. The parameters that are modified for tests are time and speed. The time is the parameters that controls the position update and execution of the loop. The speed is the input speed for the AV. For this result the speed is set to "0.7 m/s" and the time is "750 milliseconds". The movement of the AV was discrete for this sampling time. The total distance from the beginning of the path to the end is “16.67 m”. The distance between the initial AV position and the beginning of the path (parallel offset) is “0.614 m”. The AV takes a curve like path along the length of the path, this is because the start offset and of the combination of the speed used made the AV go further off for each computation. Still the AV reached the destination, the knowledge from this test was used to tune parameters for future tests. The AV using the path following algorithm discussed in previous chapter computes the steering input for each position update. The program uses distance between the waypoint and the current position for incrementing the target waypoint.

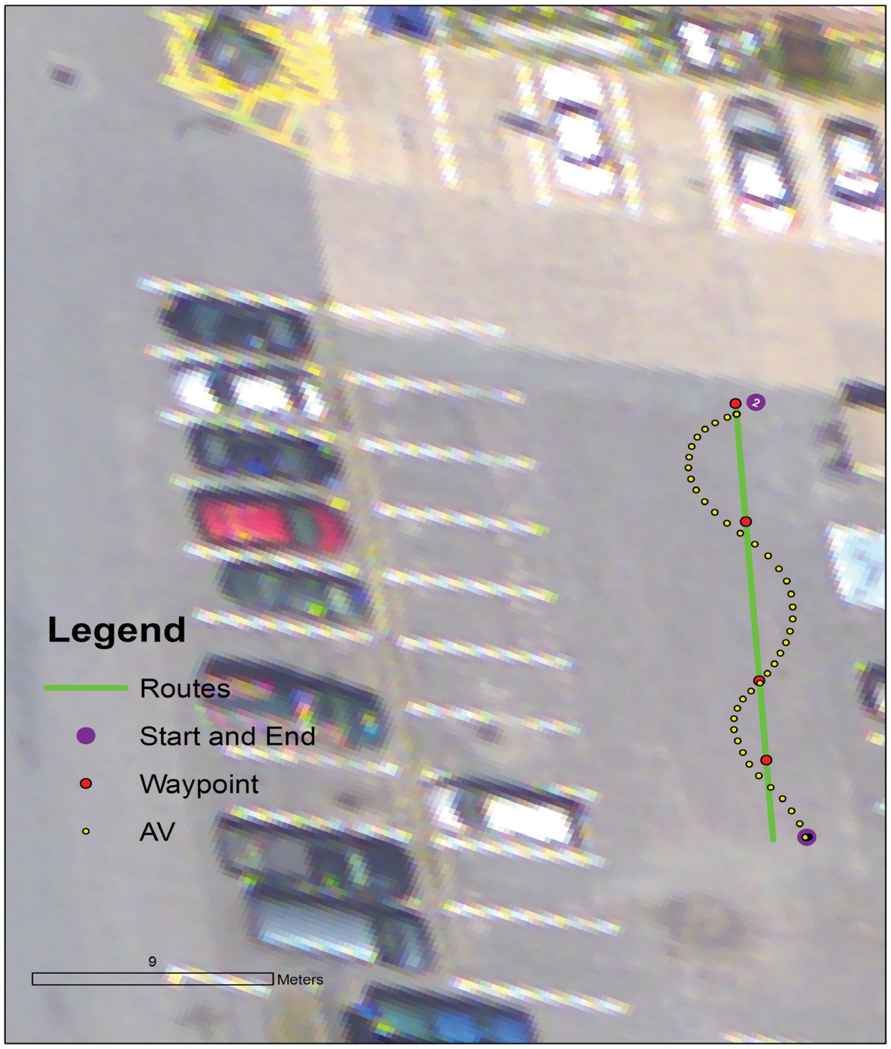


Figure 5.4: Straight Line - Test Result 2

The distance makes the vehicle to stop little distance before the actual destination point selected. The AV navigates between the waypoints and since the start point is at an offset and the AV turns towards the first waypoint and the trend continues till the last waypoint. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “0.880 m”.
Fig. 5.4 shows the result for the second test with the legends in the image, the start position is offset a lot and this influenced the path taken by the AV. For this result the speed is set to “0.5 m/s” and the time is “500 milliseconds”. The total distance from the beginning of the path to the end is “17.37 m”. The distance between the initial AV position and the beginning of the path is “0.949 m”. The start point offset makes this test behave like the first result and navigates in curve toward each of the waypoint till the last waypoint. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “1.50 m”. The result shows the values used in this test makes the AV navigate in large deviations to the path but reaches the final waypoint. The parameters for the following test were changed to make the AV follow the path closely.

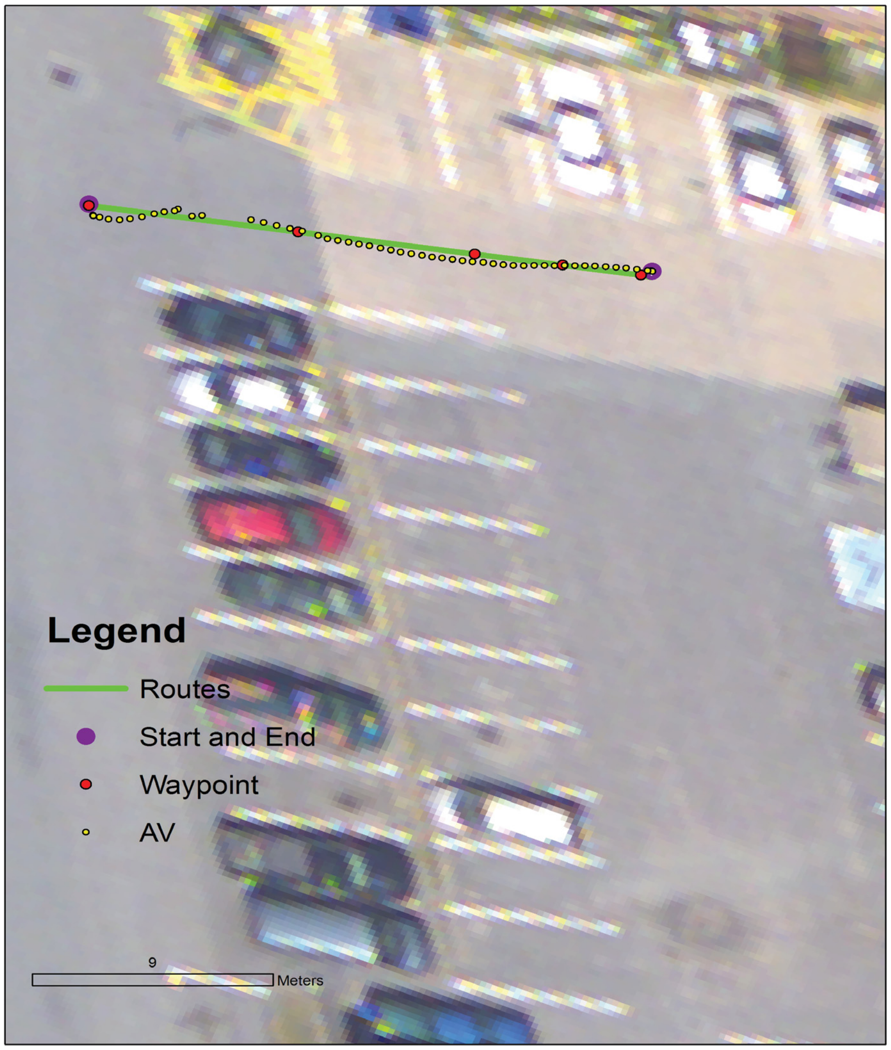


Figure 5.5: Straight Line - Test Result 3

The result shown in Fig. 5.5 was a test made to the north of the other tests and this was done to check the continuity of the path feature along the location prior to the curvilinear case. For this result the speed is set to “0.3 m/s” and the time is “500 milliseconds”. The total distance from the beginning of the path to the end is “15.54 m”. The distance between the initial AV position and the beginning of the path is “0.0238 m”. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “0.420 m”. This result shows that the values used makes AV follow the path fairly close. The reason for some missing AV position is unknown and this happened only once, but reached the destination without showing any physical symptoms during test.

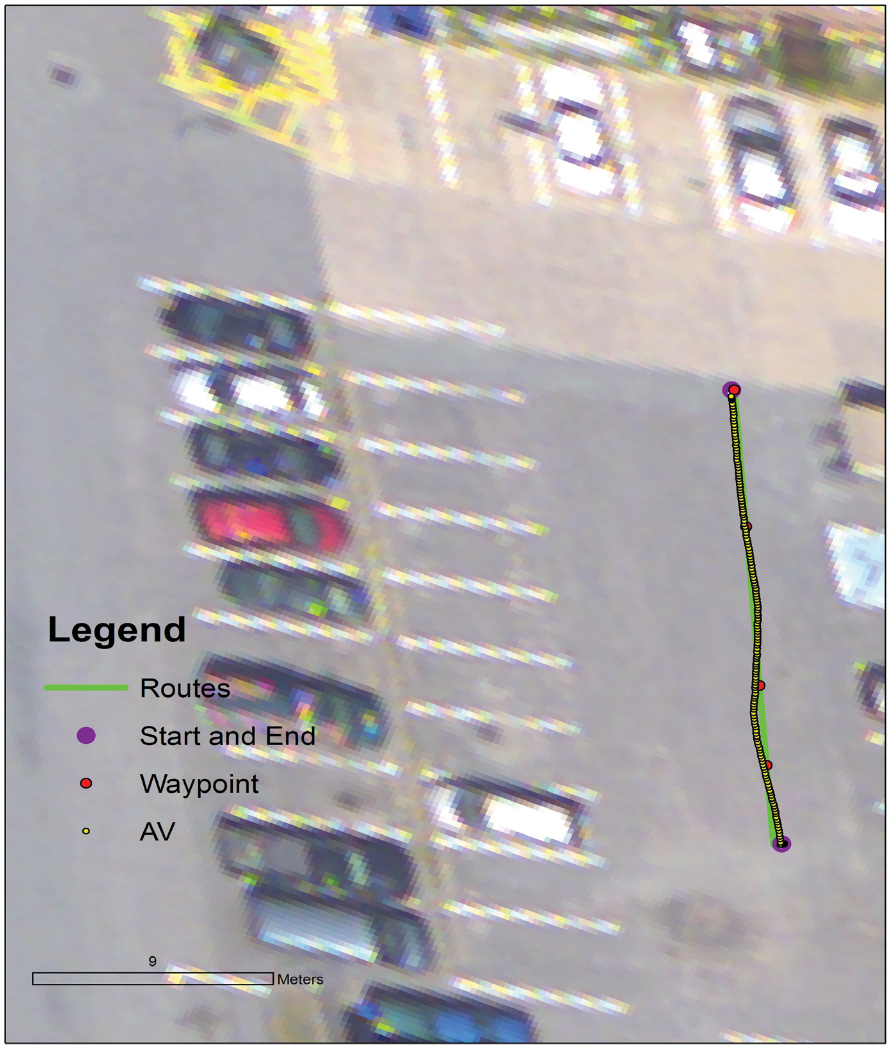


Figure 5.6: Straight Line - Test Result 4

The result shown in Fig. 5.6 was tested in the regular location with different parameters deduced from the previous result success. For this result the speed is set to “0.7 m/s” and the time is “250 milliseconds”. The total distance of the path from start of the path to the end is “18.12 m”. The distance between the initial AV position and the beginning of the path is “0.208 m”. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “0.277 m” and it is very close to the path. The values used in this test made the AV navigate smooth compared to the previous tests. Although there is a small offset at start the AV merges in to the path nearing the destination.

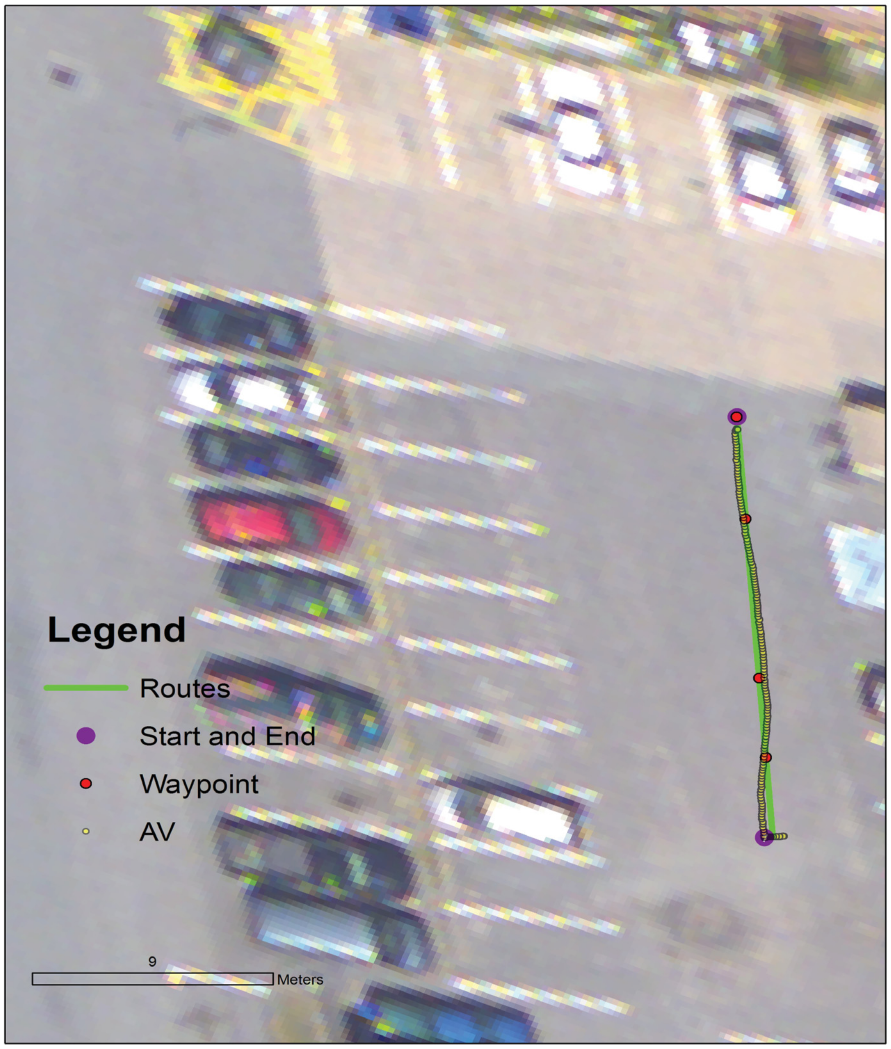


Figure 5.7: Straight Line - Test Result 5

Fig. 5.7 shows the result for different set of parameters. For this result the speed is set to “0.5 m/s” and the time is “250 milliseconds”. The total distance from the beginning of the path to the end is “16.73 m”. The distance between the initial AV position and the beginning of the path is “0.234 m”. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “0.301 m”. This test shows the AV starting on the left side of the route feature and smoothly merging with the path. This test showed a very smooth drive and the sampling time helped the tracking. In spite of the open loop path following method used; the results show very close tracking of the path.

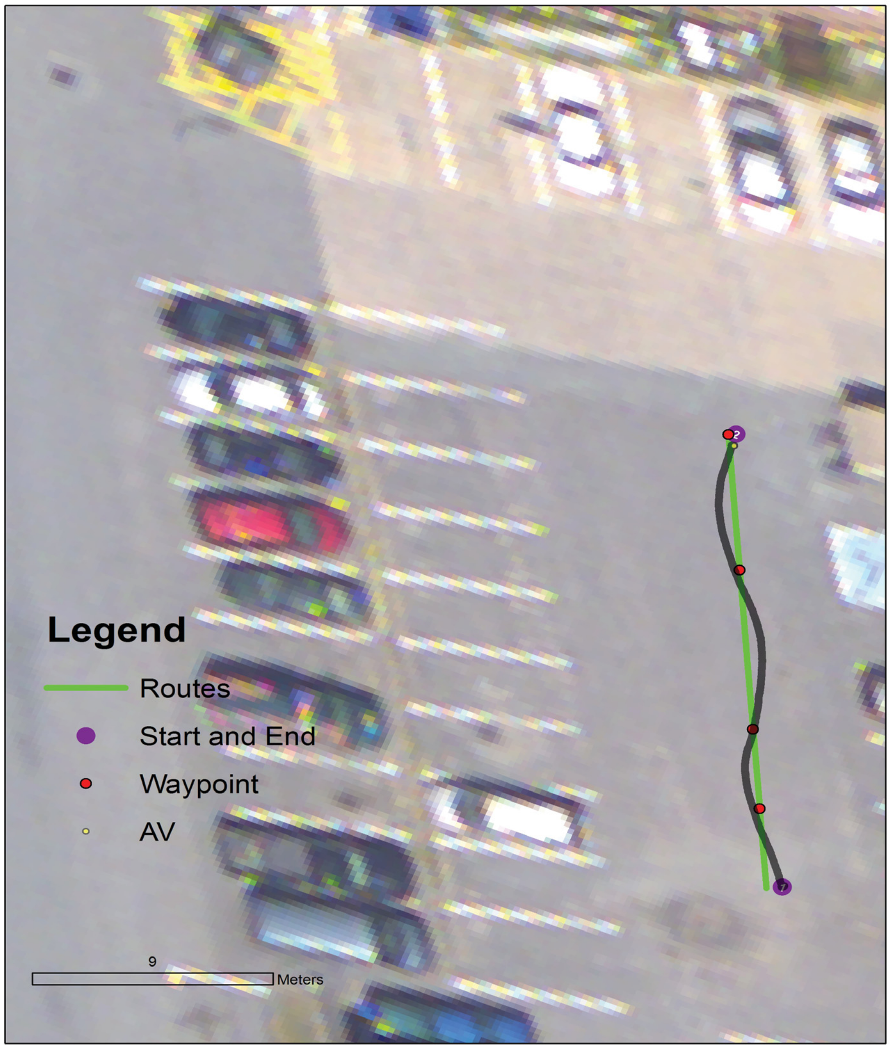


Figure 5.8: Straight Line - Test Result 6

The result shown in Fig. 5.8 were done to with different parameters for behaviour analysis. For this result the speed is set to “0.3 m/s” and the time is “500 milliseconds”. The total distance from the beginning of the path to the end is “18.07 m”. The distance between the initial AV position and the beginning of the path is “0.448 m”. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “0.471 m”. In this test the AV was positioned initially at an offset and with a lower speed and relatively higher time value (less number of cycles). Theses were done to obtain some parameters for the curvilinear test.

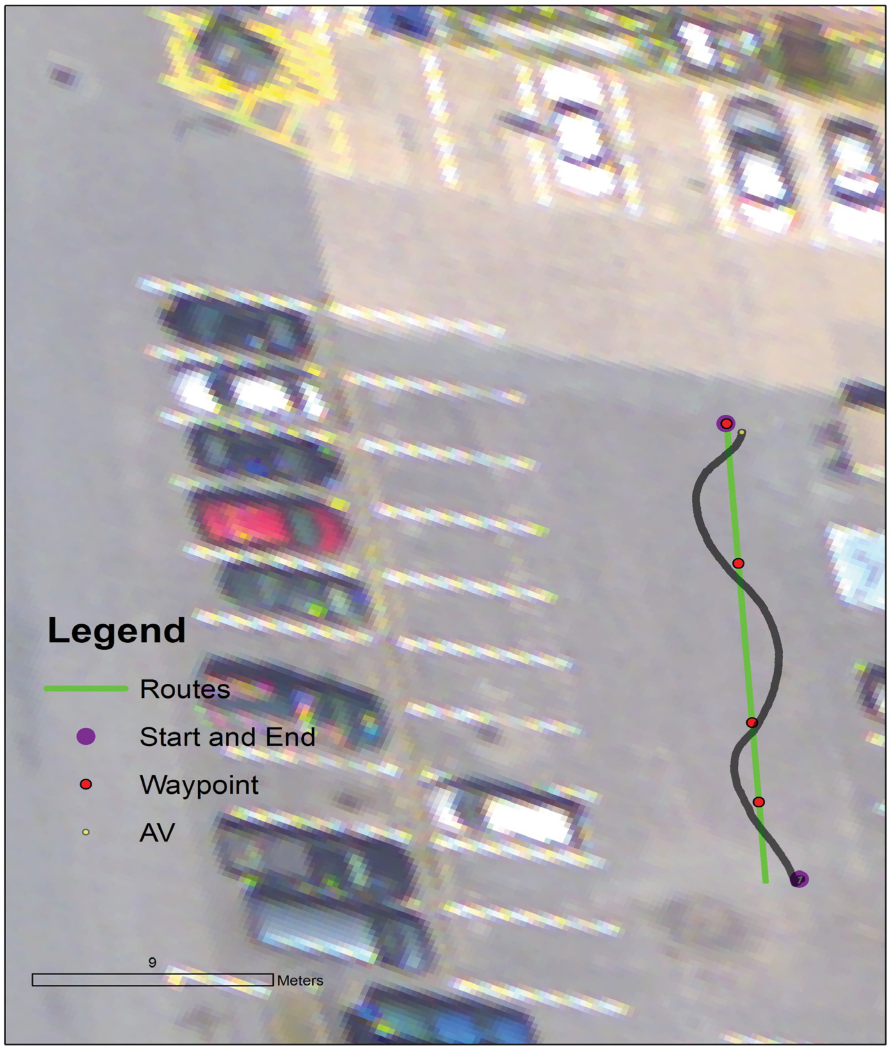


Figure 5.9: Straight Line - Test Result 7

Fig. 5.9 shows another test result with the same parameters like the last but the initial offset is higher distance from the path. For this result the speed is set to “0.3 m/s ” and the time is “500 milliseconds ”. The total distance from the beginning of the path to the end is “18.23 m ”. The distance between the initial AV position and the beginning of the path is “0.819 m ”. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “1.096 m ”. There is a huge offset at the start and the AV path behaves as observed in previous test with similar beginning. The other observations from these test is that the distance between waypoints and the the number of waypoints change the behaviour of the path of AV.

Curvilinear Case: Table 5.3 shows the conditions used for each of the tests discussed later in this part.

Table 5.3: Tests: Curvilinear Case and Conditions

Fig	Speed 'm/s'	Sampling Time	Total Distance 'm'
5.10	0.35	500'milliseconds'	25.97
5.11	0.25	250'milliseconds'	28.03
5.12	0.25	250'milliseconds'	33.63

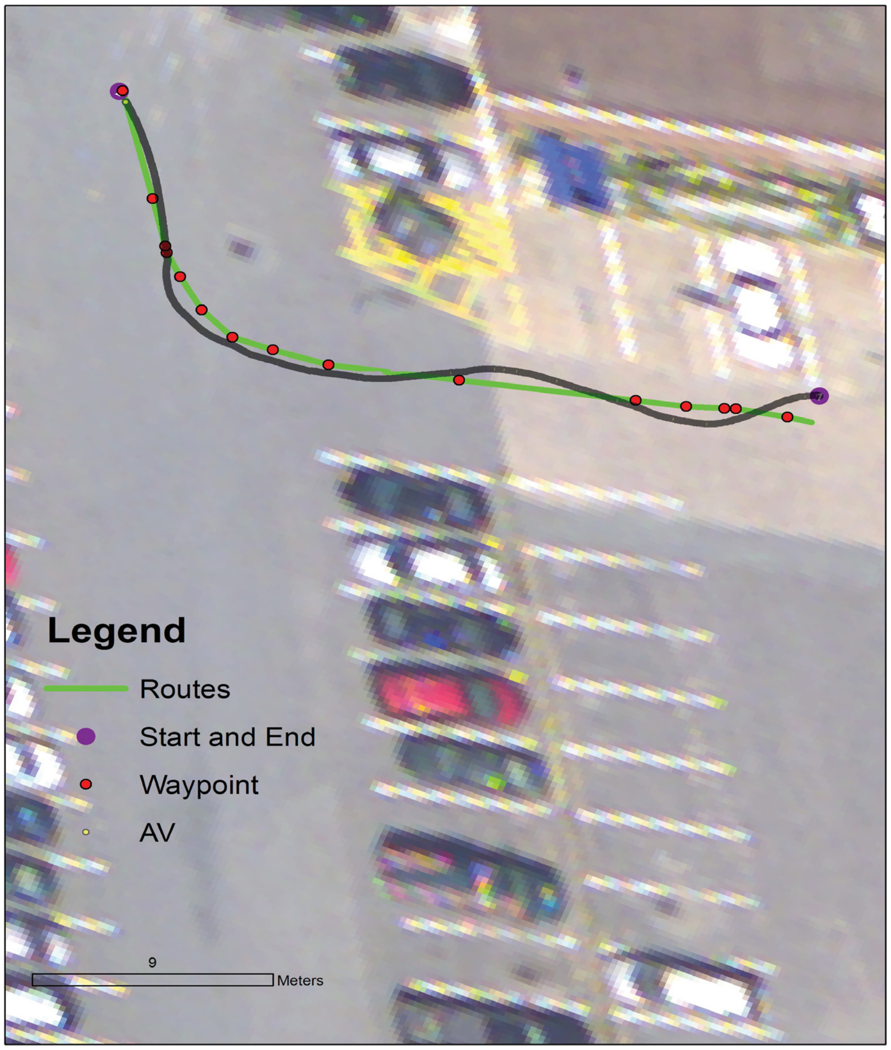


Figure 5.10: Test Result for Curvilinear case '1'

Fig. 5.10 has the result for a path with curvilinear path turning right. The start point is near the white car on the map. For this result the speed is set to "0.35 m/s" and the time is “500 milliseconds”. The total distance from the beginning of the path to the end is “25.97 m”. The distance between the initial AV position and the beginning of the path is “1.147 m”. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “0.630 m” at the turn and the offset along primary straight is “0.683 m”. Using the knowledge from the previous results; this test parameters are set. The AV reached the destination spot-on.

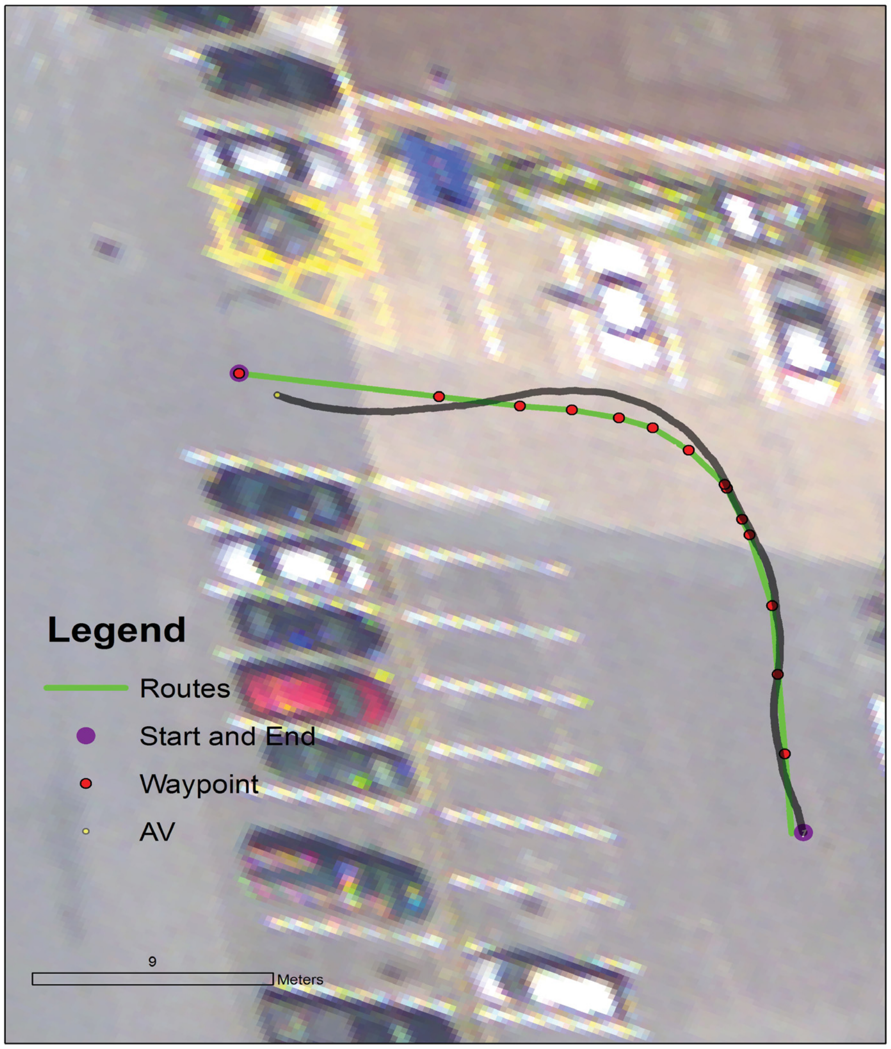


Figure 5.11: Test Result for Curvilinear case ‘2’

Fig. 5.11 shows the result for a curvilinear case with the following parameters. For this result the speed is set to “0.25 m/s” and the time is “250 milliseconds”. The total distance from the beginning of the path to the end is “28.03 m”. The distance between the initial AV position and the beginning of the path is “0.339 m”. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “0.881 m” at the curve. The start position is close to the path and the AV tracks it closely and as expected in the curve has the offset. The AV stopped “1.002 m ” ahead of the destination as service vehicle reversed in close to the destination and the laser scanner stopped the vehicle and the user terminated the drive.

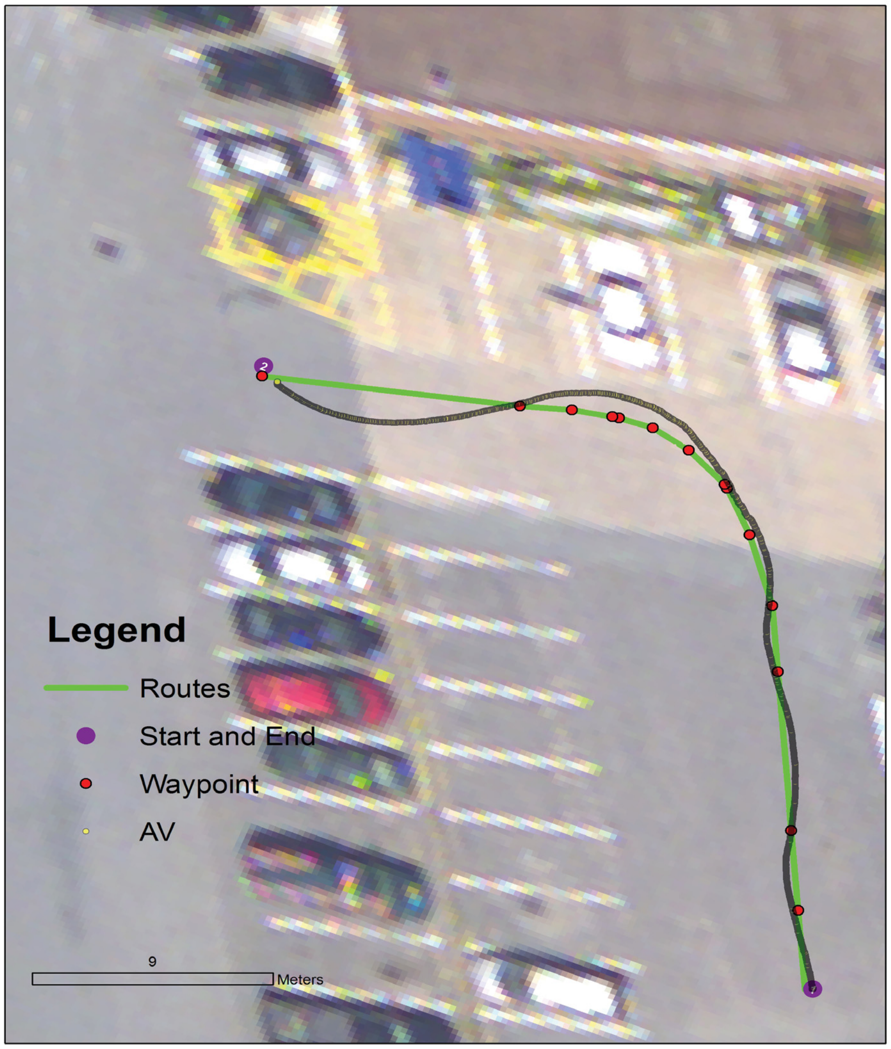


Figure 5.12: Test Result for Curvilinear case ‘3’

Fig. 5.12 has the start and end points, the path, the waypoints, the AV position from dGPS sensor. For this result the speed is set to “0.25 m/s” and the sampling time is “250 milliseconds”. The total distance from the beginning of the path to the end is “33.63 m”. The distance between the initial AV position and the beginning of the path is “0.235 m”. The furthest offset of AV from route measured using the measuring tool in ArcGIS is “0.928 m” at the turn and an offset of “1.102 m” after the turn is made. The behaviour of the AV is similar to that of the previous test but this reached the destination as there were no obstruction that was in the last case.

5.4 Comparisons

The result analysis is based on the data that was saved during test. The difference of the test result from that of simulations is discussed here. The path following method used in this work is open loop. That means the navigation does not get feedback form the vehicle at every step. The simulation results show the AV’s position merging to the path irrespective of the deviation of the starting position. This is because the position is calculated using the known values. The test uses the GPS position and the interval of update to compute the steering angle alone and not compute the position ahead. The simulation for turn shows some similarity at the turn as in simulation the distance of the AV falls further from the waypoint and falls off the path before merging back. The methods used in both the simulation and the test, the result behaviour are as expected.

5.5 Summary

The results shows that the proposed method is a viable design to be used for an AV. The digital map in this work is designed for an AV delivers the data needed for this work but has potential to provide more data. The dGPS sensor used in this work has produced repeatable results and the accuracy kept the AV from driving into the building or onto the opposing lane. The objective is achieved.

Chapter 6

Conclusions and Future Work

This chapter includes the conclusions in section 6.1 where the last three chapters are summarized. The potential future work is discussed in the section 6.2.

6.1 Conclusions

The objective of this work is to create a digital map for use with AV and use one sensor for localization and navigation. The digital map for AV is created and is tested here using the dGPS. The path following used in this work is simple waypoint based navigation. The proposed system is tested using the AV and the sensors described in this work hardware section. The test results show that system successful at obtaining repeatable results. The simple waypoint navigation scheme works with the map path data and GPS. The accuracy of GPS used is 0.6 m and the vehicle is is bigger than the area of accuracy of the GPS making it suitable for open loop path following. The proposed scheme of using map designed for AV and using dGPS alone for localization and navigation is a success and any path following methods using waypoints can be used for more accurate following or correcting the deviations along the path.

6.2 Future Work

Further the study can be made for obstacle negotiation, that is to drive around the obstacle using other sensors if the map permits. This work used laser range finder to avoid collision. The AV stopped when it detected obstacle and started continuing on clear. The future work can be done to navigate around the obstacle using the map as a guide as to direction to which the turn can be made. Some of the options are discussed below.

6.2.1 Hardware Upgrade and Use

The ability of the proposed system navigating with minimal sensors is proved with the test results. But the potential to use different sensors for purpose other than localization is there. The use of colour sensor or camera to look for signal when approaching the intersections is another test that can be done. The vehicle used in this work is not legal to be used in street but if the vehicle is able then the sensor can be activated for such a situation. For instance there can be some sensor to avoid running in to intersection. If there is a vehicle in front of AV with this proposed system the AV would stop until the vehicle in front moves, but if the AV is in an intersection as the first vehicle a camera or a light colour sensor can be activated on demand for providing go/ no-go. Side scanning sonar can be incorporated to use in parallel parking and the parking location details provided by map like time at which the spot can be used or not so on.

6.2.2 Map

This section discusses the possibilities of the digital map in future works. In this work the path is computed at the start and it doesn't change until the next cycle of user input. The digital map can be integrated as a service and can be used as a dynamic map path planning component. This change will enable the method of navigation used in this work to use only one waypoint as target as that waypoint will be changing with the dynamic path planning. The other simpler possibility would be to use the digital map to provide some clues for other sensors on board. For instance the map can be used to activate camera or light sensor when the vehicle approaches intersection with signal. More customization can be done to the map (to the road feature) based on the vehicle dynamics like the radius of turn and the length. The map created like this need not have a possible path to such a location by avoiding a non maneuverable situation like eighteen wheeler in a short curved roads.

6.2.3 Control Possibilities

This work uses open loop waypoint control with the GPS and speed varied, the distance between AV and waypoints for increment through the waypoints. The path following method used in the following work [20] can be used as the direct closed loop following scheme for this system. Any control methods that use the waypoint for the reference can be put to use with the proposed system. The control would remove the deviations that are observed in some of the results shown in this work.

Bibliography

[1] *Hemisphere GPS, GPS Technical Reference: Part No. 875-0175-000 Rev. D1, Hemisphere GPS Precision GPS Applications*, 2008.

[2] K. Johns and T. Taylor. *Professional Microsoft Robotics Developer Studio*. Wiley Publishing Inc, 2008.

[3] Elliott D. Kaplan and Christopher Hegarty. *Understanding GPS: Principles and Applications, Second Edition*. Artech House mobile communications series. Artech House, Boston, 2005.

[4] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[5] http://www.terramax.com/technology/thebrains.cfm.

[6] http://cs.stanford.edu/group/roadrunner//old/technology.html.

[7] http://www.tartanracing.org/tech.html.

[8] http://grandchallenge.mit.edu/technology.shtml.

[9] http://www.me.vt.edu/urbanchallenge/vehicle.htm.

[10] Vishisht Gupta. *Vehicle localization using low accuracy GPS, IMU and map-aided vision*. PhD thesis, Penn State University, 2009.

[11] A. Nemra and N. Aouf. Robust ins/gps sensor fusion for uav localization using sdre nonlinear filtering. *IEEE Sensors Journal*, 10(4):789 –798, Apr 2010.

[12] Chirdpong Deelertpaiboon and Manukid Parnichkun, editors. *Fusion of GPS, Compass, and Camera for Localization of an Intelligent Vehicle*, volume 5. International Journal of Advanced Robotic Systems, 2008.

[13] C. Wuthishuwong, C. Silawatchananai, and M. Parnichkun, editors. *Navigation of an intelligent vehicle by using stand-alone GPS, compass and laser range finder*. IEEE International Conference on Robotics and Biomimetics, ROBIO, IEEE, Feb 2009.

[14] A. Soloviev and F. Van Graas. Use of deeply integrated gps/ins architecture and laser scanners for the identification of multipath reflections in urban environments. *IEEE Journal of Selected Topics in Signal Processing*, 3(5):786 –797, Oct 2009.

[15] Mathieu Joerger and Boris Pervan, editors. *Measurement-Level Integration of Carrier-Phase GPS and Laser-Scanner for Outdoor Ground Vehicle Navigation.* Journal of Dynamic Systems, Measurement, and Control, ACME, 2009.

[16] S. Shair, J.H. Chandler, V.J. Gonzalez-Villela, R.M. Parkin, and M.R. Jackson, editors. *The Use of Aerial Images and GPS for Mobile Robot Waypoint Navigation*, volume 13. IEEE/ASME Transactions on Mechatronics, Dec 2008.

[17] Bing-Fei Wu, Tsu-Tian Lee, Hsin-Han Chang, Jhong-Jie Jiang, Cheng-Nan Lien, Tien-Yu Liao, and Jau-Woei Perng. Gps navigation based autonomous driving system design for intelligent vehicles. In *International Conference on Systems, Man and Cybernetics, ISIC*, pages 3294 –3299. IEEE, Oct 2007.

[18] Seung-Hun Kim, Chi-Won Roh, Sung-Chul Kang, and Min-Yong Park. Outdoor navigation of a mobile robot using differential gps and curb detection. In *International Conference on Robotics and Automation, 2007*, pages 3414 –3419. IEEE, April 2007.

[19] *Autonomous Mobile Robots Sensing, Control, Decision Making and Applications.* Taylor and Francis Group, LLC, 2006.

[20] Mohd Sani Mohamad Hashim. Multiple waypoints trajectory planning with specific position, orientation,velocity and time using geometric approach for a car-like robot. Australasian Conference on Robotics and Automation, Dec 2009.

[21] Gang Chen and Thierry Fraichard. A real-time navigation architecture for automated vehicles in urban environments. Proceedings of the Intelligent Vehicles Symposium, Istanbul, Turkey, June 2007. IEEE.

[22] Balaji Sethuramasamyraja. Gps based waypoint navigation for an autonomous guided vehicle - bearcat lll. Master's thesis, University of Cincinnati, 2003.

[23] Chang Boon Low and Danwei Wang, editors. *GPS-Based Path Following Control for a Car-Like Wheeled Mobile Robot With Skidding and Slipping*, volume 16. IEEE Transactions ON Control Systems Technology, Mar 2008.

[24] RobuCAB. *User's Manual Index:D.*

[25] W.B. Zavoli and S.K. Honey. Map matching augmented dead reckoning. In *36th Vehicular Technology Conference*, volume 36, pages 359 – 362. IEEE, May 1986.

[26] http://www.cs.cmu.edu/red/red/h1ghlander.html.

[27] E. North, J. Georgy, M. Tarbouchi, U. Iqbal, and A. Noureldin. Enhanced mobile robot outdoor localization using ins/gps integration. In *International Conference on Computer Engineering Systems, ICCES*, pages 127 –132, Dec 2009.

[28] Maria Gini and Richard Voyles, editors. *Distributed Autonomous Robotic Systems 7.* Springer, 2006.

[29] J.B. Bullock and E.J. Krakiwsky. Analysis of the use of digital road maps in vehicle navigation. In *IEEE Position Location and Navigation Symposium*, pages 494 –501. IEEE, April 1994.

[30] Jing Peng, M.E. El Najjar, C. Cappelle, D. Pomorski, F. Charpillet, and A. Deeb. A novel geo-localisation method using gps, 3d-gis and laser scanner for intelligent vehicle navigation in urban areas. International Conference on Advanced Robotics, ICAR, pages 1 –6, June 2009.

Appendix A

Sample MATLAB Code

Both straight line and curvilinear path were tested using the following Matlab code used for simulation. The file containing the waypoints is changed for different case. Note: The sample code plotting the path and simulating the AV's path in meters. The haversine equation is used to convert the degrees coordinates to meter for the meter simulations. It's omitted here for simplicity.

```
%File: newcarctrld6.m
%Change route file for turn and straight line
%-----------------File reads-------------------------
clear all; clc; close all;
[ruti,rutx,ruty]= textread('C:\*****\Balu\******\
MATLAB\rut_Oct18T1.txt', '%f %f %f ');
%---------------constants-------------------------
i=1; k=1; earthm=(6371000);%( meters)
%------------------------------------------------
plot(rutx,ruty,'-o','MarkerSize',10);
xlabel('Longitude (Degrees)'); ylabel('Latitude (Degrees)');
hold on;
rl=length(ruti);
while k<rl
 difyy=ruty(k+1)-ruty(k);%del-lat
 difxx=rutx(k+1)-rutx(k);%del-long
 am(k)=sind(difyy/2)^2+cosd(ruty(k))*
cosd(ruty(k+1))*sind(difxx/2)^2;
 cm(k)=2*atan2(sqrt(am(k)),sqrt(1-am(k)));
 rdm(k)=earthm*cm(k);
 k=k+1;
end
 tot=sum(rdm);
 %gets initial av point


[xc(1),yc(1)]=ginput(1);% long, lat
plot(xc,yc,'-+r');
lj=legend('Path','Car');set(lj);
hold on;
%Calculates the distance(m)between rutpt(2)first goal and car position
%hf1
   dxcr=rutx(2)-xc(1);%del-long
   dycr=ruty(2)-yc(1);%del-lat
   cam=sind(dycr/2)^2+cosd(yc(1))*cosd(ruty(2))*sind(dxcr/2)^2;
   ccm=2*atan2(sqrt(cam),sqrt(1-cam));
   crdm=earthm*ccm;
 %end hf
 ru=1;l=1;
     while (crdm > .11 || ru<rl )
     carpt=[xc(l) yc(l)];
     if(ru==rl)
     angpt=[rutx(ru) ruty(ru)];
     rorgpt=[rutx(ru-1) ruty(ru-1)];
     else
     angpt=[rutx(ru+1) ruty(ru+1)];
     rorgpt=[rutx(ru) ruty(ru)];
     end
  %angle
     lac(l)=sqrt((angpt(1)-carpt(1))^2+(angpt(2)-carpt(2))^2);
     laro(l)=sqrt((angpt(1)-rorgpt(1))^2+(angpt(2)-rorgpt(2))^2);
     lcro(l)=sqrt((carpt(1)-rorgpt(1))^2+(carpt(2)-rorgpt(2))^2);
     teta(l)=acos((lac(l)^2+laro(l)^2-lcro(l)^2)/(2*lac(l)*laro(l)));
   if teta(l)> 0.5
        teta(l)= 0.5;
   elseif teta(l)<-0.5
```

```
       teta(l)=-0.5;
   end
  %end angle
     rslop(l)=((angpt(2)-rorgpt(2))/(angpt(1)-rorgpt(1)));
     const(l)=angpt(2)-(rslop(l)*angpt(1));
     tor(l)=atan2((angpt(2)*pi/180)-(rorgpt(2)*pi/180),
  ((angpt(1)*pi/180)-(rorgpt(1)*pi/180)));
      ed(l)=((xc(l)*rslop(l))+yc(l)*(-1)+const(l))/
sqrt((rslop(l))^2+(-1)^2);
       if ed(l) >0
              teta(l)=-teta(l);
           else
              teta(l)=teta(l);
        end
      pteta(l)=pi-(tor(l)+teta(l));
      xc(l+1)=xc(l)- .000006*cos(pteta(l));
      yc(l+1)=yc(l)+ .000006*sin(pteta(l));
     %hf2
       dxa(l)=angpt(1)-xc(l);%del-long
       dya(l)=angpt(2)-yc(l);
       dam(l)=sind(dya(l)/2)^2+cosd(yc(l))*cosd(angpt(2))*
       sind(dxa(l)/2)^2;
       dcm=2*atan2(sqrt(dam(l)),sqrt(1-dam(l)));
       drdm(l)=earthm*dcm;
      %end hf
      if (drdm(l)<0.7)
                 while(rdm(ru)<.25 && ru<rl-1
                 ru=ru+1;
               end
              ru=ru+1;
         end
      l=l+1;
     %hf3
       dxcr=angpt(1)-xc(l);
       dycr=angpt(2)-yc(1);
       cam=sind(dycr/2)^2+cosd(yc(1))*cosd(angpt(2))*sind(dxcr/2)^2;
       ccm=2*atan2(sqrt(cam),sqrt(1-cam));
       crdm=earthm*ccm;
     %end hf
    plot(xc,yc,'-+r')
   end
plot(xc,yc,'-+r',rutx,ruty,'-o','MarkerSize',10);
hold on;
```

Appendix B

Sample C# Code

This is the C Sharp code that was used for testing with AV.

B.1 The Main program that makes the service

```
/*--------------------------
* Connects to car and gps and controls the car using map data
* and gps position
* Created for my research -Balu --*/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using Microsoft.Ccr.Core;
using Microsoft.Dss.Core.Attributes;
using Microsoft.Dss.ServiceModel.Dssp;
using Microsoft.Dss.ServiceModel.DsspServiceBase;
using W3C.Soap;
using ds = Microsoft.Dss.Services.Directory;
// added as partners
using gps = Microsoft.Robotics.Services.Sensors.Gps.Proxy;
// for connecting to car service
using laser = Robosoft.RobuBOX.Generic.Devices.LaserSensor.Proxy;
using loc=Robosoft.RobuBOX.Generic.Processing.Signal.Localization.
Proxy;
using cardrive = Robosoft.RobuBOX.Generic.Devices.CarDrive.Proxy;
// for windows form
using System.Windows.Forms;
using Microsoft.Ccr.Adapters.WinForms;
//for saving file
```

```
using System.IO;

using submgr = Microsoft.Dss.Services.SubscriptionManager;
//ESRI_ for map
using ESRI.ArcGIS.esriSystem;
// for timer to use with while
using s = System.Threading;

namespace dGpsDrive
{
/// <summary>
/// Main Service
/// </summary>
[DisplayName("dGpsDrive")]
[Description("dGpsDrive service (ControlUI for Autonomus driving
RobuCab with dGPS)")]
 [Contract(Contract.Identifier)]
public class dGpsDriveService : DsspServiceBase
{
   private static LicenseInitializer m_AOLicenseInitializer =
new LicenseInitializer();
#region State and Partners
//state
/// <summary>
/// Sercice State
/// </summary>
[ServiceState]
  private dGpsDriveState _state = new dGpsDriveState();
 // Partner Ports
  private cardrive.CarDriveOperations _carDrivePort;
....
* Explained in main text
```

```
 #endregion
#region Service Port
/// <summary>
/// Main service port
/// </summary>
  [ServicePort("/dGpsDrive", AllowMultipleInstances = false)]
  private dGpsDriveOperations _mainPort = new dGpsDriveOperation
s();
  #endregion
#region Members
 //for laser drop
 string _laserSubMgr = string.Empty;
 string _laserSubscriber = string.Empty;
 //for loc drop
 string _localizationSubMgr = string.Empty;
 string _localizationSubscriber = string.Empty;
 // for gps drop
 string _gpsSubMgr = string.Empty;
 string _gpsSubscriber = string.Empty;
 string _robotUri = string.Empty;
 // handle to the main service
 Form1 _form1;
 //handle for form1 to send message to this
 Form1port _Form1Port = new Form1port();
 // based on dcdsupervisor
 private string[] rout;
 StreamWriter _datas; StreamWriter _GPSLog;
 StreamWriter _tofile;StreamWriter _gga;
 string dt = string.Format("{0:yyyy-MM-dd_hh-mm}", DateTime.Now);
 private Double slop; private Double cons;
 private Double[] ruti = null; private Double[] rutlong = null;
 private Double[] rutlat = null; Double earthm = 6371000;
 private Double[,] refpt;
 private Double[,] angpt;
 private Double[,] carpt;
 Double Speed;
 Double Steer;
 Double angle;
 private readonly s.AutoResetEvent _rset =
new System.Threading.AutoResetEvent(false);
  #endregion
 //TimeSpan interl = TimeSpan.FromMilliseconds(500);//half second
 TimeSpan interl = TimeSpan.FromMilliseconds(250);//quater second
#region Creation and Start
/// <summary>
/// Service constructor
/// </summary>
 public dGpsDriveService(DsspServiceCreationPort creationPort)
       : base(creationPort)
        {       }
  /// <summary>
  /// Service start
  /// </summary>
protected override void Start()
 {
//ESRI License Initializer generated code.
m_AOLicenseInitializer.InitializeApplication(new esriLicenseProduct
Code[] { esriLicenseProductCode.esriLicenseProductCodeArcEditor,
esriLicenseProductCode.esriLicenseProductCodeArcInfo },
 new esriLicenseExtensionCode[] {
esriLicenseExtensionCode.esriLicenseExtensionCodeNetwork,
esriLicenseExtensionCode.esriLicenseExtensionCodeTracking });
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
 // create instance of form
WinFormsServicePort.Post(new RunForm(userwin));
 base.Start();
 // TT - Added code to create a default State if no
 // config file exists
 if (_state == null)
 {
  _state = new dGpsDriveState();
   _state.Log = true;
 }
Console.WriteLine("Starting dGPSDrive");
//Interleaves
MainPortInterleave.CombineWith(new Interleave(
 new TeardownReceiverGroup(),
 new ExclusiveReceiverGroup(),
 new ConcurrentReceiverGroup(
Arbiter.Receive<DsspDefaultLookup>(true, _mainPort, DefaultLookup
Handler),
Arbiter.Receive<DropfromForm1>(true,_Form1Port,DropForm1
Handler),
Arbiter.Receive<ResetLoc>(true,_Form1Port,ResetLocHandler),
Arbiter.Receive<Connect>(true, _Form1Port, ConnectHandler),
Arbiter.Receive<srt>(true,_Form1Port,driveHandler),
Arbiter.Receive<stp>(true,_Form1Port,stopHandler),
Arbiter.Receive<rt>(true, _Form1Port, RouteHandler)
 )));
 _tofile = new StreamWriter(@"C:\**********\******\
***\dGpsDrive\*********\dGpsDrive\runtimelog\" + dt +
"_strlog.txt", false);
_GPSLog = new StreamWriter(@"C:\****\****\Desktop\
*******\Network_*****\p2****\NewWGS84\" + dt + "_GPSLog.txt",
false);
 _datas = new StreamWriter(@"C:\**********\******\*****\
dGpsDrive\*********\dGpsDrive\runtimelog\" + dt + "_datas.txt",
false);
 _gga = new StreamWriter(@"C:\**********\******\*****\
dGpsDrive\*********\dGpsDrive\runtimelog\" + dt + "_gga.txt",false);
 }
```

```
#endregion

#region Dss Handlers
//dgpsdriveoperations
/// <summary>
/// Get handler explecit interleave as concurrent
/// </summary>
/// <param name="get"></param>
/// <returns></returns>
[ServiceHandler(ServiceHandlerBehavior.Concurrent)]
public virtual IEnumerator<ITask> GetHandler(Get get)
{
get.ResponsePort.Post(_state);
  yield break;
 }
/// <summary>
/// Replace handler explecit interleave as exclusive
/// </summary>
/// <param name="replace"></param>
/// <returns></returns>
[ServiceHandler(ServiceHandlerBehavior.Exclusive)]
public virtual IEnumerator<ITask> ReplaceHandler(Repalce replace)
  {
     _state = replace.Body;
     replace.ResponsePort.Post(DefaultReplaceResponseType.Instance);
     yield break;
    }
/// <summary>
/// Drop service handler explecit interleave as teardown
/// </summary>
/// <param name="drop"></param>
/// <returns></returns>
[ServiceHandler(ServiceHandlerBehavior.Teardown)]
public virtual IEnumerator<ITask> DropHandler(DsspDefaultDrop
drop)
{
 bool error = false;
 if (_localizationSubMgr != string.Empty)
 {
  //unsubscribe to localization
 submgr.DeleteSubscription delete = new submgr.DeleteSubscription
();
 delete.Body = new submgr.DeleteSubscriptionMessage
(_localizationSubscriber);
  ServiceForwarder<submgr.SubscriptionManagerPort>
(_localizationSubMgr).Post(delete);
yield return Arbiter.Choice<DefaultDeleteResponseType, W3C.Soap.
Fault>(
delete.ResponsePort,
  delegate(DefaultDeleteResponseType response)
 {
 _localizationSubMgr = null;
 _localizationSubscriber = null;
 },
delegate(W3C.Soap.Fault fault)
 {
LogError("Error deleting subscription to localization");
  error = true;
  });
  }
 if (_laserSubMgr != string.Empty)
  {
   //unsubscribe to laser
 submgr.DeleteSubscription delete = new submgr.
DeleteSubscription();
  delete.Body = new submgr.DeleteSubscriptionMessage
(_laserSubscriber);
 ServiceForwarder<submgr.SubscriptionManagerPort>(_laserSubMgr)
```

```
.Post(delete);
 yield return Arbiter.Choice<DefaultDeleteResponseType, W3C.Soap.
Fault>(
 delete.ResponsePort,
 delegate(DefaultDeleteResponseType response)
 {
  _laserSubMgr = null;
  _laserSubscriber = null;
  },
 delegate(W3C.Soap.Fault fault)
  {
 LogError("Error deleting subscription to laser");
 error = true;
 });
}
if (_gpsSubMgr != string.Empty)
 {
//unsubscribe to GPS
submgr.DeleteSubscription delete = new submgr.
DeleteSubscription();
delete.Body = new submgr.DeleteSubscriptionMessage
(_gpsSubscriber);
ServiceForwarder<submgr.SubscriptionManagerPort>(_gpsSubMgr)
.Post(delete);
yield return Arbiter.Choice<DefaultDeleteResponseType, W3C.Soap.
Fault>(
delete.ResponsePort,
delegate(DefaultDeleteResponseType response)
 {
  _gpsSubMgr = null;
  _gpsSubscriber = null;
  },
 delegate(W3C.Soap.Fault fault)
  {
```

```
  LogError("Error deleting subscription to laser");
  Console.WriteLine("Error deleting subscription to laser");
  error = true;
 });
 }
if (!error)
{
 LogInfo("Service dropped");
 DirectoryDelete();
Shutdown();
drop.ResponsePort.Post(DefaultDropResponseType.Instance);
 }
else
  {
  LogError("Service dropping failed");
  drop.ResponsePort.Post(new W3C.Soap.Fault());
  }
  yield break;
  }
 #endregion
 #region Notifications from Sensors
 #region GPSNotifications handler
/// <summary>
/// gganhandler
/// </summary>
/// <returns>ggaN</returns>
 IEnumerator<ITask> gganhandler(gps.GpGgaNotification ggaN)
 {
 if(ggaN.Body.IsValid==false)
 {
  SpawnIterator(stopcar);
  Console.WriteLine(" GPGGA is not valid");
 yield break;
  }
```

```
//send gga to form ,read from partner
 for (; ; )
 {
 WinFormsServicePort.Post(new FormInvoke(delegate()
   {
_form1.gganotify(ggaN.Body.AgeOfDifferentialCorrection,
ggaN.Body.AltitudeMeters, ggaN.Body.AltitudeUnits, ggaN.Body.
DiffRefStationId,
ggaN.Body.GeoIdSeparation, ggaN.Body.GeoIdSeparationUnits,
ggaN.Body.HorizontalDilutionOfPrecision,
ggaN.Body.Latitude, ggaN.Body.Longitude, ggaN.Body.SatellitesUsed);
    }));
  _gga.WriteLine(ggaN.Body.Longitude.ToString("##.########") +',’
+ ggaN.Body.Latitude.ToString("##.########"));
  carpt = new double[,] { { Math.Round(ggaN.Body.Longitude, 8) },
 { Math.Round(ggaN.Body.Latitude, 8) } };
  yield break;
  }
 }
/// <summary>
/// gllnhandler
/// </summary>
/// <returns>gllN</returns>
IEnumerator<ITask> gllnhandler(gps.GpGllNotification gllN)
 {
 //send gll to form ,read from partner
   for(;;)
   {
    WinFormsServicePort.Post(new FormInvoke(delegate()
     {
  _form1.gllnotify(gllN.Body.Latitude, gllN.Body.Longitude,
gllN.Body.MarginOfError, gllN.Body.Status);
     }));
  yield break;
  }
 }
/// <summary>
/// gsanhandler
/// </summary>
/// <returns>gsaN</returns>
IEnumerator<ITask> gsanhandler(gps.GpGsaNotification gsaN)
 {
 //send gsa to form ,read from partner
  for (; ; )
 {
 WinFormsServicePort.Post(new FormInvoke(delegate()
   {
_form1.gsanotify(gsaN.Body.AutoManual, gsaN.Body.Horizontal
DilutionOfPrecision, gsaN.Body.SphericalDilutionOfPrecision,
gsaN.Body.Status, gsaN.Body.VerticalDilutionOfPrecision);
    }));
yield break;
 }
 }
/// <summary>
/// gsvnhandler
/// </summary>
/// <returns>gsvN</returns>
IEnumerator<ITask> gsvnhandler(gps.GpGsvNotification gsvN)
{
//send gsv to form ,read from partner
 for (; ; )
  {
  WinFormsServicePort.Post(new FormInvoke(delegate()
   {
 _form1.gsvnotify(gsvN.Body.SatellitesInView);
   }));
  yield break;
```

```
    }
 }
/// <summary>
/// rmcnhandler
/// </summary>
/// <returns>rmcN</returns>
IEnumerator<ITask> rmcnhandler(gps.GpRmcNotification rmcN)
{
//send rmc to form ,read from partner
for (; ; )
  {
WinFormsServicePort.Post(new FormInvoke(delegate()
   {
 _form1.rmcnotify(rmcN.Body.CourseDegrees, rmcN.Body.Latitude,
rmcN.Body.Longitude, rmcN.Body.SpeedMetersPerSecond, rmcN.Body
.Status);
   }));
 yield break;
  }
 }
/// <summary>
/// vtgnhandler
/// </summary>
/// <returns>vtgN</returns>
IEnumerator<ITask> vtgnhandler(gps.GpVtgNotification vtgN)
{
//send vgt to form ,read from partner
for (; ; )
{
WinFormsServicePort.Post(new FormInvoke(delegate()
    {
   _form1.vtgnotify(vtgN.Body.SpeedMetersPerSecond);
    }));
    yield break;
   }
 }
  #endregion
#region car odo and loc handler
/// <summary>
/// Occures when Steering angle updated
/// </summary>
/// <param name="lostr">CarSteeringAngle</param>
void carodostr(loc.Update lostr)
  {
    WinFormsServicePort.Post(new FormInvoke(delegate()
    {
    _form1.locstr(lostr.Body.Theta);
     }));
   }
#endregion
/// <summary>
/// laser notification Occure when laser is updated
/// </summary>
/// <param name="update">laser values</param>
IEnumerator<ITask> Laserhandler(laser.Update update)
 {
laser.Update temp;
while (_laserNotif.P3.Test(out temp))
   {
   update = temp;
   }
WinFormsServicePort.Post(new FormInvoke(delegate()
  {
 _form1.robotViewControl1.UpdateLaser(update.Body);
  }));
yield return Arbiter.Receive<DateTime>(false, TimeoutPort(100),
   delegate(DateTime d)
  {
```

```
Activate(Arbiter.ReceiveWithIterator<laser.Update>(
 false, _laserNotif, Laserhandler));
   });
   yield break;
  }
#endregion

#region Internal Handler
//from RoboGenGUI
/// <summary>
/// Launch Form
/// </summary>
/// <returns>Form initialized</returns>
Form1 userwin()
  {
   _form1 = new Form1(_Form1Port);
   return _form1;
  }
#region connection to other services
// string uri = "localhost:50000";// for onboard
//string uri = "IP:50000";//for remote
/// <summary>
/// Lauch services connections
/// </summary>
void ConnectHandler(Connect c)
 {
  _robotUri = "http://" + c.uri + "/directory";
  SpawnIterator<string>(_robotUri, ConnectToCarDriveHandler);
  SpawnIterator<string>(_robotUri, ConnectToLocalizationHandler);
  SpawnIterator<string>(_robotUri, ConnectToLaserHandler);
  SpawnIterator(Connect2GpsHandler);
 }
#region Connect and Subscribe to GPS
/// <summary>
/// Connect and Subscribe to Gps
/// </summary>
/// <returns>calls gps handlers</returns>
IEnumerator<ITask> Connect2GpsHandler()
 {
  bool success = false;
 WinFormsServicePort.Post(new FormInvoke(delegate()
  {
     _form1.Status("Subscribing initialize to gps");
     }));
//Create a subscription message to subscribe to the MGps-360
Service
  gps.Subscribe msg = new Microsoft.Robotics.Services.Sensors.
Gps.Proxy.Subscribe();
   //sets notif port
   msg.NotificationPort = _microsoftGpsServiceNotify;
   //Post the message for subscribe
    _microsoftGpsServicePort.Post(msg);
   // Wait for a response
 yield return Arbiter.Choice(
msg.ResponsePort, delegate(SubscribeResponseType response) {
 success = true; LogInfo("Subscribe Success gps"); },
 delegate(Fault fault) { success = false; LogInfo("error
connecting gps" + fault); }         );
       if (!success)
       yield break;// subscription failed

// Add receivers to the main interleave for each of the possible
// notification messages from the GPS Service.
 MainPortInterleave.CombineWith(new Interleave(
        new ExclusiveReceiverGroup(),
        new ConcurrentReceiverGroup
    (
     Arbiter.ReceiveWithIterator<gps.GpGgaNotification>
(true, _microsoftGpsServiceNotify, gganhandler),
     Arbiter.ReceiveWithIterator<gps.GpGllNotification>
(true, _microsoftGpsServiceNotify, gllnhandler),
     Arbiter.ReceiveWithIterator<gps.GpGsaNotification>
(true, _microsoftGpsServiceNotify, gsanhandler),
     Arbiter.ReceiveWithIterator<gps.GpGsvNotification>
(true, _microsoftGpsServiceNotify, gsvnhandler),
     Arbiter.ReceiveWithIterator<gps.GpRmcNotification>
(true, _microsoftGpsServiceNotify, rmcnhandler),
     Arbiter.ReceiveWithIterator<gps.GpVtgNotification>
(true, _microsoftGpsServiceNotify, vtgnhandler)
    )));
WinFormsServicePort.Post(new FormInvoke(delegate()
   {
     _form1.Status("Gps Subscription Successful");
    }));
  }
  #endregion
#region Connection and get to CarDrive
/// <summary>
/// Connect to Car Drive service and Get Car State
/// </summary>
IEnumerator<ITask> ConnectToCarDriveHandler(string robotUri)
 {
 bool error = false
 WinFormsServicePort.Post(new FormInvoke(delegate()
 {
  _form1.Status("Connection to CarDrive service Initializing");
 }));
 //Inform to which sercice to connect
ServiceInfoType info = new ServiceInfoType(cardrive.Contract.
Identifier);
 //Declare a port to the node/machine where the service to
 connect is running, giving URI
 ds.DirectoryPort remotePort = ServiceForwarder<ds.DirectoryPort>
(new Uri(robotUri));
 //Declare the request
ds.Query query = new ds.Query(new ds.QueryRequestType(info));
//post
 remotePort.Post(query);
 //listen for a response
yield return Arbiter.Choice<ds.QueryResponseType, W3C.Soap.Fault>
(query.ResponsePort,
   delegate(ds.QueryResponseType response)
 {
 //instanciate the service port
_carDrivePort = ServiceForwarder<cardrive.CarDriveOperations
>(response.RecordList[0].Service);
  WinFormsServicePort.Post(new FormInvoke(delegate()
   {
   _form1.Status("Connection to CarDrive service SUCCEDED");
   }));
 },
  delegate(W3C.Soap.Fault fault)
 {
  error = true;
WinFormsServicePort.Post(new FormInvoke(delegate()
  {
   _form1.Status("Connection to CarDrive service FAILED");
   }));
 });
 if (!error)
  {
yield return Arbiter.Choice<cardrive.CarDriveState,
W3C.Soap.Fault>(_carDrivePort.Get(),
delegate(cardrive.CarDriveState response)
   {
WinFormsServicePort.Post(new FormInvoke(delegate()
    {
   // reads from state propertise
   _form1.carodo(response.CurrentSpeed);
    string minst = response.Properties.MinSteer.ToString();
    string maxst = response.Properties.MaxSteer.ToString();
    _form1.carstrmax( minst + "," + maxst);
    }));
   },
   delegate(W3C.Soap.Fault fault)
    {
    LogError("Error getting car drive state!");
     });
   }
 }
 #endregion

#region Connection and subscription to laser
/// <summary>
/// Connect to laser of the current car/sim
/// </summary>
IEnumerator<ITask> ConnectToLaserHandler(string robotUri)
{
  Boolean error = false;
 //Inform to which sercice to connect
WinFormsServicePort.Post(new FormInvoke(delegate()
   {
    _form1.Status("Connection to Laser service Initializing");
   }));
//Change "laser" by the alias of the service to connect
ServiceInfoType info = new ServiceInfoType(laser.Contract.Identifier);
//Declare a port to the node/machine where the service to
connect is running, giving URI
 ds.DirectoryPort remotePort = ServiceForwarder<ds.DirectoryPort>
(new Uri(robotUri));
```

```
  //Declare request
  ds.Query query = new ds.Query(new ds.QueryRequestType(info));
  //post
  remotePort.Post(query);
  //listen for a response
yield return Arbiter.Choice<ds.QueryResponseType, W3C.Soap.Fault>
(query.ResponsePort,
delegate(ds.QueryResponseType response)
  {
 error = false;
 //instanciate services port: receive the two intance of laser
 _laserPort = ServiceForwarder<laser.LaserSensorOperations>
(response.RecordList[0].Service);
 },
delegate(W3C.Soap.Fault fault)
 {
  error = true;
 _laserPort = null;
WinFormsServicePort.Post(new FormInvoke(delegate()
    {
    _form1.Status("Connection to Laser service FAILED");
    }));
   });
 if (error == false)
  {
  //Subscribe
yield return Arbiter.Choice<SubscribeResponseType, W3C.Soap.Fault
>( _laserPort.Subscribe(_laserNotif),
 delegate(SubscribeResponseType response)
   {
//Make the link between the message receive and the handler
  _laserSubMgr = response.SubscriptionManager;
  _laserSubscriber = response.Subscriber;
Activate(Arbiter.ReceiveWithIterator<laser.Update>(false,
_laserNotif, Laserhandler));
WinFormsServicePort.Post(new FormInvoke(delegate()
   {
   _form1.Status("Connection to Laser service SUCCEDED");
    }));
   },
  delegate(W3C.Soap.Fault fault)
  {
 WinFormsServicePort.Post(new FormInvoke(delegate()
  {
   _form1.Status("Connection to Laser service FAILED");
   }));
  });
  }   }
 #endregion

#region Connect To Localization Handler
/// <summary>
/// Connect To Localization of the current car
/// </summary>
/// <param name="robotUri"></param>
/// <returns></returns>
IEnumerator<ITask> ConnectToLocalizationHandler(string robotUri)
 {
    bool error = false;
   //Inform to which sercice to connect
 WinFormsServicePort.Post(new FormInvoke(delegate()
       {
       _form1.Status("Connection to Laser service Initializing");
       }));
  ServiceInfoType info = new ServiceInfoType(loc.Contract.Identifier);
  //Declare a port to the node/machine where the service
 to connect is running, giving URI
   ds.DirectoryPort remotePort = ServiceForwarder<ds.DirectoryPort>
```

```
(new Uri(robotUri));
//Declare request
 ds.Query query = new ds.Query(new ds.QueryRequestType(info));
//post
  remotePort.Post(query);
//listen for a response
yield return Arbiter.Choice<ds.QueryResponseType, W3C.Soap.Fault>
(query.ResponsePort,
delegate(ds.QueryResponseType response)
 {
  error = false;
 _locPort = ServiceForwarder<loc.LocalizationOperations>
(response.RecordList[0].Service);
 },
 delegate(W3C.Soap.Fault fault)
  {
   error = true;
   _locPort = null;
WinFormsServicePort.Post(new FormInvoke(delegate()
   {
  _form1.Status("Connection to Loc service FAILED");
   }));
 });
if (error == false)
 {
//Subscribe with the odometry
yield return Arbiter.Choice<SubscribeResponseType,
W3C.Soap.Fault>(
 _locPort.Subscribe(_locNotify),
 delegate(SubscribeResponseType response)
  {
//Make he link between the message receive and the handler
 _localizationSubMgr = response.SubscriptionManager;
 _localizationSubscriber = response.Subscriber;
 Activate(Arbiter.Receive<loc.Update>(false, _locNotify, carodostr));
WinFormsServicePort.Post(new FormInvoke(delegate()
   {
  _form1.Status("Connection to Loc service SUCCEDED");
  }));
 },
delegate(W3C.Soap.Fault fault)
 {
  WinFormsServicePort.Post(new FormInvoke(delegate()
  {
  _form1.Status("Connection to Loc service FAILED");
 }));
});
 } }
 #endregion
 #endregion
 #region DriveCar
 /// <summary>
/// receives new route as string[]
/// </summary>
void RouteHandler(rt _r)
 {
  rout = _r.rou;
 }
/// <summary>
/// gets button event from form to drive car
/// </summary>
/// <param name="st"></param>
void driveHandler(srt st)
 {
SpawnIterator<double,double>(0,0,drv);
int rl = rout.Length;
int datart = (rl - 1) / 3;
int ct = 0;
```

```
ruti = new double[datart];
rutlat = new double[datart];
rutlong = new double[datart];
for (int s = 0; s < rl - 1; )
 {
ruti[ct] = Convert.ToDouble(rout[s]);
rutlong[ct] = Math.Round(Convert.ToDouble(rout[s + 1]), 8);
rutlat[ct] = Math.Round(Convert.ToDouble(rout[s + 2]), 8);
 s = s + 3; ct = ct + 1;
 }
int ril = ruti.Length;
int j = 0;
double[] rdis = new double[ril - 1];
while (j != ril - 1)
{
double[,] firpt = { { rutlong[j] }, { rutlat[j] } };
double[,] secpt = { { rutlong[j + 1] }, { rutlat[j + 1] } };
rdis[j] = HS(firpt, secpt);
j = j + 1;
}
double tdis = 0;
foreach (double q in rdis)
 {
 tdis += q;
_datas.WriteLine(q.ToString() + ":rutdist each" + "\n");
}
_GPSLog.WriteLine("OutLoop:" + carpt[0, 0].ToString() + ":long,lat:" +
carpt[1, 0].ToString() + "\n");
angpt = new double[,] { { rutlong[1] }, { rutlat[1] } };
refpt = new double[,] { { rutlong[0] }, { rutlat[0] } };
double crdm = HS(carpt, angpt);
_datas.WriteLine(crdm.ToString() + ",:(crdm) Car2G1dis_m" + "\n");
_datas.WriteLine(ril.ToString() + ":ril=length of rut index" + "\n");
_datas.WriteLine(rl.ToString() + ":rl=length of rut" + "\n");
```

```
 int ru = 0;
while (((Math.Round(crdm, 2) > 0.11) && ru != ril - 1) || (ru < ril - 1))
 {
_GPSLog.WriteLine(carpt[0, 0].ToString() + ":long,lat:" +
carpt[1, 0].ToString() + "\n");
// Calculates the angle defined by the 3 points.
 if (ru == ril)
 {
 //set last point (angle) to be current
 //and last but one to be org
angpt = new double[,] { { rutlong[ru] }, { rutlat[ru] } };
refpt = new double[,] { { rutlong[ru - 1] }, { rutlat[ru - 1] } };
}
 else if (ru < ril - 1)
 {
 angpt = new double[,] { { rutlong[ru + 1] }, { rutlat[ru + 1] } };
 refpt = new double[,] { { rutlong[ru] }, { rutlat[ru] } };
  }
_datas.WriteLine(angpt[0, 0].ToString() + "," +angpt[1, 0].ToString() +
 ":angpt" + "\n");
_datas.WriteLine(refpt[0, 0].ToString() + "," + refpt[1, 0].ToString() +
 ":refpt" + "\n");
_datas.WriteLine(carpt[0, 0].ToString() + "," + carpt[1, 0].ToString() +
 ":carpt" + "\n");
// angle calc
angle = tt(angpt, carpt, refpt);
//for determining the new position
slop = (angpt[1, 0] - refpt[1, 0]) / (angpt[0, 0] - refpt[0, 0]);
cons = angpt[1, 0] - (slop * angpt[0, 0]);
double tor = Math.Atan2(((angpt[1, 0] * (Math.PI /180)) - (refpt[1, 0]*
(Math.PI/180))),((angpt[0, 0]*(Math.PI/180))-(refpt[0, 0] *
(Math.PI/180))));
double ed = ((carpt[0, 0] * slop) + (carpt[1, 0] * (-1)) + cons) /
Math.Sqrt((Math.Pow(slop, 2)) + (Math.Pow((-1), 2)));
```

```
_datas.WriteLine(ed.ToString() + ":ed" + "\n");
//if ed is -ve the car is on the right side of path and should turn Left
//and if +ve the car is on the left and should turn Right
//-ve steer turns Right and +ve Turns Left
if (ed > 0)
 {
angle = -angle;
  }
Steer = Math.Round(angle, 2);
Speed = 0.35;//m/s //Speed = 0.5;
SpawnIterator<double, double>(Speed, Steer, drv);
_tofile.WriteLine(ru.ToString() + ":RU waypoint counter" + "\n");
 _tofile.WriteLine("Steer:"+Steer.ToString() + "Speed:" +Speed.
ToString() +"\n");
// waits for the time interval before continuing to next step
 _rset.WaitOne(interl);
 double drdm = HS(carpt, angpt);
_datas.WriteLine(drdm.ToString() + ":dist b/w car & ru+1" + "\n");
if ((Math.Round(drdm, 1) < 0.7) && (ru < ril - 1))
{
 while (Math.Round(rdis[ru], 2) < 0.27))
     {
      ru = ru + 1;
      }
    ru = ru + 1;
   }
   crdm = HS(carpt, angpt);
  _datas.WriteLine(crdm.ToString() + ":crdm loop check" + "\n");
  _datas.WriteLine(ru.ToString() + ":RU waypoint counter" + "\n");
   double dwpt = HS(refpt, carpt);
  _datas.WriteLine(dwpt.ToString() + ":dist b/w 'ru' w
aypoint and car" + "\n");
Application.DoEvents();
if (Math.Round(dwpt, 3) > Math.Round(Math.Round(
HS(refpt, angpt) * 1.1, 3)) && (ru < ril - 1))
   {
   ru = ru + 1;
   }
 if (Math.Round(HS(refpt, carpt),3) > Math.Round(
Math.Round(HS(refpt, angpt)*4,3)))
    {
      SpawnIterator<double, double>(0, 0, drv);
      _datas.WriteLine("stopcar!!");
      SpawnIterator(stopcar);
     break;
    }
  }
      SpawnIterator<double, double>(0, 0, drv);
      SpawnIterator(stopcar);
_tofile.WriteLine("Steer:"+Steer.ToString() +",Speed:"+Speed.
ToString() +"\n");
 _datas.Close();
  }
private Double HS(double[,] r, double[,] t)
  {
   int lh = r.Length;
   double reflat = r[1, 0]; double reflong = r[0, 0];
   double tlat = t[1, 0]; double tlong = t[0, 0];
   double dlat = tlat - reflat;
   double dlong = tlong - reflong;
   double ea = Math.Pow(Math.Sin((dlat / 2) * (Math.PI / 180)), 2) +
Math.Cos(reflat * (Math.PI / 180)) * Math.Cos(tlat * (Math.PI / 180))*
Math.Pow(Math.Sin((dlong / 2) * (Math.PI / 180)), 2);
 double ec = 2 * Math.Atan2(Math.Sqrt(ea), Math.Sqrt(1 - ea));
 double mdis = earthm * ec;
  return mdis;
 }
private Double tt(double[,] a, double[,] c, double[,] r)
```

```
 {
 int l = c.Length;
double lac = Math.Sqrt(Math.Pow(a[0, 0] - c[l - 2, 0], 2) +
Math.Pow(a[1, 0] - c[l - 1, 0], 2));
double laro = Math.Sqrt(Math.Pow(a[0, 0] - r[0, 0], 2) +
Math.Pow(a[1, 0] - r[1, 0], 2));
double lcro = Math.Sqrt(Math.Pow(c[l - 2, 0] - r[0, 0], 2) +
Math.Pow(c[l - 1, 0] - r[1, 0], 2));
double t = Math.Acos(((Math.Pow(lac, 2) + Math.Pow(laro, 2) -
Math.Pow(lcro, 2)) / (2 * lac * laro)));
_tofile.WriteLine("ang calc:" +t.ToString() );
 if (Math.Round(t, 2) > 0.56)
 {
  t = 0.56;
 }
 else if (Math.Round(t, 2) < -0.56)
 {
  t = -0.56;
 }
 return Math.Abs(t);
 }
/// <summary>
/// Sends spd and str to drive car
/// d is speed and r is steer sent after calculation when this is called
/// </summary>
/// <param name="d"></param>
/// <param name="r"></param>
/// <returns></returns>
 IEnumerator<ITask> drv(double d, double r)
  {
  cardrive.DriveRequest req = new cardrive.DriveRequest();
  req.Speed = d; req.Steer = r;
yield return Arbiter.Choice<DefaultSubmitResponseType, W3C.Soap.
Fault>(
 _carDrivePort.Drive(req),
delegate(DefaultSubmitResponseType response)
   {
    },
  delegate(W3C.Soap.Fault f)
    {
    LogError("Error sending drive consign");
     });
   }
 #endregion
#region StopCar
/// <summary>
/// gets stop event from form and calls stopcar
/// </summary>
/// <param name="sp"></param>
void stopHandler(stp sp)
{
 Application.DoEvents();
 while (true)
  {
   SpawnIterator(stopcar);
  }
 }
/// <summary>
/// Sends car speed and steer value=0 to stop
/// </summary>
/// <returns></returns>
IEnumerator<ITask> stopcar()
  {
  cardrive.DriveRequest r = new cardrive.DriveRequest();
  r.Speed = 0; r.Steer = 0;
 yield return Arbiter.Choice<DefaultSubmitResponseType, W3C.Soap.
Fault>(
   _carDrivePort.Drive(r),
 delegate(DefaultSubmitResponseType response)
  {
   },
delegate(W3C.Soap.Fault f)
  {
LogError("Error sending drive consign");
   });
 }
#endregion
void ResetLocHandler(ResetLoc r)
 {
   loc.Replace reset = new loc.Replace();
   reset.Body.X = 0;
   reset.Body.Y = 0;
   reset.Body.Theta = 0;
   this._locPort.Post(reset);
 }
#region Formdrop
void DropForm1Handler(DropfromForm1 d)
 {
  _gga.Close(); _tofile.Close();
  Console.WriteLine("GUI Closed");
 _mainPort.Post(new DsspDefaultDrop());
  }
 #endregion
#endregion
 }
}
```

B.2 Interface

This is a file that is a part of the main service. This file contains code for the interface that bridges the user interface inputs to the main service and receives data from main service to display.

```
/*---------------------------
 * file :form1.cs ; this is the main user interface
-----------------------------*/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
// Added references
// Because this is a standard WinForm,
//it does not automatically get CCR/DSS
using Microsoft.Ccr.Core;
using Microsoft.Dss.Core;
using Microsoft.Dss.ServiceModel.Dssp;
using gps = Microsoft.Robotics.Services.Sensors.Gps.Proxy;
using laser = Robosoft.RobuBOX.Generic.Devices.LaserSensor.Proxy;
// added to support the laser drawing
using System.Drawing.Drawing2D;
// for esri map
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geometry;
// for featureclass point
using ESRI.ArcGIS.Geodatabase;
// for reading database
```

```
using ESRI.ArcGIS.DataSourcesFile;

namespace dGpsDrive
{
/// <summary>
/// gets user input and shows the map and outher sensor datas
/// </summary>
public partial class Form1 : Form
{
 #region Members
//handle for form(this)to send events to main service
Form1port _formPort;
 StreamWriter _rpt ;
 string dt = string.Format("{0:yyyy-MM-dd_hh-mm}", DateTime.Now);
//private members for event handles esri
private IPoint savedClickPoint = null;
private string savedClickPointText = null;
private string savedCurrentPointText = null;
/// <summary>
/// port for receiving route
/// </summary>
  private RoutePort _rp = new RoutePort();
  RouteClass _rc = new RouteClass();
  string format = "####.########";
  private const String SHAPE_WORKSPACE = @"C:\*****\Balu\******\
*******\Network_data\******\NewWGS84";
  private const String INPUT_STOPS_FC = "stops";
  private string[] rout;
  string cLat;
  string cLon;
  Double _clat;
  Double _clong;
#endregion
/// <summary>
/// initiates form events when application starts
/// </summary>
/// <param name="EventPort"></param>
public Form1(Form1port EventPort)
{
 InitializeComponent();
 _formPort = EventPort;
 robotViewControl1.SetRobot(1.2, 1.8, 0);
 robotViewControl1.SetRange(18);
 _rpt = new StreamWriter(@"C:\*****\Balu\******\*******\
Network_data\******\NewWGS84" dt + "_rut.txt", false);
 new ConcurrentReceiverGroup(
 Arbiter.Receive<string>(true,_rp,messag));
  }
void messag(string st)
 {
       //add to writefile
      textBox3.Text += st + "\r\n";
 }
/// <summary>
/// Gets connection status of Partners!
/// </summary>
/// <param name="sts"></param>
public void Status(string sts)
 {
  //add to writefile
  textBox3.Text += sts + "\r\n";
 }
/// <summary>
/// reads min and max steer from car properties
/// </summary>
/// <param name="str"></param>
public void carstrmax(string str)
  {
   textBox2.Text = str+ "rad";
  }
private void Form1_FormClosed(object sender, FormClosedEventArgs
 e)
 {
  _formPort.Post(new DropfromForm1());
 }
#region from GPS Notification port
internal void gganotify(string p, double p_2, string p_3, string p_4,
string p_5, string p_6,double p_7, double p_8, double p_9,int p_10)
  {
   this._clat = p_8;
   this._clong = p_9;
   adcgga.Text = p;
   altgga.Text = p_2.ToString();
   altunitgga.Text = p_3;
   difidgga.Text = p_4;
   geoidgga.Text = p_5;
   geoidunitgga.Text = p_6;
   hdopgga.Text = p_7.ToString();
   latgga.Text = _clat.ToString();
   longigga.Text =_clong.ToString();
   label34.Text = _clat.ToString() + "\n " + _clong.ToString();
  }
internal void gllnotify(double p, double p_2, string p_3, string p_4)
  {
    glllati.Text = p.ToString(format);
    glllongi.Text = p_2.ToString(format);
    gllmoerror.Text = p_3;
    gllstat.Text = p_4;
 }
internal void gsanotify(string p, double p_2, double p_3, string p_4,
double p_5)
 {
    amgsa.Text = p;
    hdopgsa.Text = p_2.ToString();
    sdopgsa.Text = p_3.ToString();
    gsastat.Text = p_4;
    vdopgsa.Text = p_5.ToString();
  }
internal void gsvnotify(int p)
 {
   sivgsv.Text = p.ToString();
 }
internal void rmcnotify(double p, double p_2, double p_3,
double p_4, string p_5)
 {
   cdrmc.Text = p.ToString();
   rmclat.Text = p_2.ToString(format);
   rmclong.Text = p_3.ToString(format);
   rmcspeed.Text = p_4.ToString();
   rmcstat.Text = p_5;
 }
internal void vtgnotify(double p)
 {
  vtgspeed.Text = p.ToString();
 }
#endregion
#region from car odo loc port
 /// <summary>
 ///  gets upadates from loc steering angle
 /// </summary>
 /// <param name="odstr"></param>
 public void locstr(double odstr)
  {
   sangle.Text = odstr.ToString();
  }
/// <summary>
/// gets car speed from cardrive state
/// </summary>
/// <param name="cspd"></param>
public void carodo(double cspd)
  {
   odospeed.Text = cspd.ToString();
   }
 #endregion
 #region ui buttons event
 private void connect_Click(object sender, EventArgs e)
   {
     Connect _con = new Connect();
     _con.uri = nodePartner.Text;
     _formPort.Post(_con);
    }
/// <summary>
/// Path Planning ,,shows path on the form !
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void PathPlng_Click(object sender, EventArgs e)
 {
 // create  (point)class for current pos
 #region set currentgpspoint as geometry in stops
  // Open the feature workspace, input feature class, and
network dataset
  IWorkspaceFactory workspaceFactory = new ShapefileWorkspace
FactoryClass();
  IFeatureWorkspace featureWorkspace = workspaceFactory.
OpenFromFile(SHAPE_WORKSPACE, 0) as IFeatureWorkspace;
  // sets stops as input fc
  IFeatureClass inputStopsFClass = featureWorkspace.
OpenFeatureClass(INPUT_STOPS_FC);
  // for the second row update
  ITable table;
  table = inputStopsFClass as ITable;
  IRow row;
  int j = 0;
  // find the field to be updated
  j = table.FindField("Shape");
 //get the row of objectID(OID) "0"
  row = table.GetRow(0);
 //update value of the field in the present row
  IPoint crnptn = new PointClass();
  crnptn.Y =Convert.ToDouble(_clat);
  crnptn.X=Convert.ToDouble(_clong);
  row.set_Value(j,crnptn as IGeometry);
  cLat = crnptn.Y.ToString();
  cLon = crnptn.X.ToString();
  savedCurrentPointText = cLat + "," + "\n " + cLon ;
  label34.Text = savedCurrentPointText;
  //store the assigned value
  row.Store();
#endregion
 // set stopsfc with new input
 // and save the created featureclass for the solveroute()
  rout= _rc.SolveRoute();
  int r = 0;
 while (r < rout.Length)
  {
   _rpt.WriteLine(rout[r]);
   r++;
  }
  _rpt.Close();
  rt _rt = new rt();
 _rt.rou = rout;
 _formPort.Post(_rt);
 }
```

```
private void clear_Click(object sender, EventArgs e)
  {
   label33.Text = "";
   label34.Text = "";
  _formPort.Post(new ResetLoc());
  }
private void start_Click(object sender, EventArgs e)
  {
   _formPort.Post(new srt());
   }
private void stop_Click(object sender, EventArgs e)
   {
   _formPort.Post(new stp());
   }
#endregion
#region Map mouse Control Event Handlers
/// <summary>
/// on mouse down for destination
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
public void axMapControl1_OnMouseDown(object sender,
ESRI.ArcGIS.Controls.IMapControlEvents2_OnMouseDownEvent e)
 {
 // Save the mouse position on left click
  if (e.button == 1)
   {
    IPoint MapPoint = new PointClass();
    MapPoint.X = e.mapX;
    MapPoint.Y = e.mapY;
    savedClickPoint = MapPoint;
    string sLat = savedClickPoint.Y.ToString(format);
    string sLon = savedClickPoint.X.ToString(format);
    savedClickPointText =  sLat + ","+"\n " + sLon ;
 // asignes to the destin textbox
    label33.Text = savedClickPointText;
// Open the feature workspace, input feature class, and
network dataset
   IWorkspaceFactory workspaceFactory =
new ShapefileWorkspaceFactoryClass();
   IFeatureWorkspace featureWorkspace = workspaceFactory.
OpenFromFile(SHAPE_WORKSPACE, 0) as IFeatureWorkspace;
 // sets stops as input fc
   IFeatureClass inputStopsFClass = featureWorkspace.
OpenFeatureClass(INPUT_STOPS_FC);
 // for the second row update
   ITable table;
   table = inputStopsFClass as ITable;
   IRow row;
   int i = 1;
   // find the field to be updated
   i = table.FindField("Shape");
   //get the row of objectID(OID) "1"
   row = table.GetRow(1);
  //update value of the field in the present row
  row.set_Value(i, savedClickPoint as IGeometry);
  // store the assigned value
  row.Store();
  }
 else
  {
 //If right or middle mouse button zoom to user defined rectangle
 //Create an envelope and grab hold of the IEnvelope interface
 IEnvelope envelope = axMapControl1.TrackRectangle();
 //If user dragged a rectangle
 if (envelope != null)
  {
  //Set map controls extent property
  axMapControl1.Extent = envelope;
  }
 }
IMapControl2 mapcontrol = (IMapControl2)axMapControl1.Object;
 mapcontrol.Refresh(esriViewDrawPhase.esriViewGeography,
null, mapcontrol.Extent);
 }
#endregion
 }}
```