OBSTACLE AVOIDANCE FOR UNMANNED SURFACE VEHICLES: SIMULATIONS AND EXPERIMENTS

A thesis submitted to the
Department of Electrical Engineering, University of Moratuwa
in partial fulfillment of the requirements for the
Degree of Master of Philosophy

by
RANDOBAGEGEETHJAYENDRA

Supervised by: Dr. Sisil Kumarawadu

Department of Electrical Engineering
University of Moratuwa, Sri Lanka

2009

93025

Abstract

Sri Lanka ports authority and many other organizations are increasingly interested in the use of Unmanned Surface Vehicles (USV) for harbor security and surveillance applications. USVs can be used to collect information, samples and perform experiments inside a harbor or outside by. Navigating through ships and other objects.

This research study is focused on finding algorithms for obstacle avoidance (OA) of USVs. The initial paradigm that is used to establish the solution was the OA of Unmanned Ground Vehicles (UGV). The algorithms developed for UGV were implemented practically with the limitations of hardware. Then, effort is taken to apply those algorithms to the surface vehicles with some modifications.

In this study, a novel OA algorithm is proposed for static obstacles based on the Morphin algorithm. This proposed algorithm and the previous algorithm which is developed based on ground vehicles are compared with the potential field method.

Static OA without dynamic OA is not helpful for unmanned vehicles on sea. A lot of researches have been carried out to avoid dynamic objects, but have failed to find an optimum solution although comparatively good approaches have been presented. Intelligent techniques have been rarely applied for dynamic obstacle avoidance. In this research, the effectiveness of applying intelligent or mathematical techniques for path prediction of dynamic obstacles is discussed with simulations to pick the best for a given situation. Then a noval projected dynamic obstacle area method is presented to avoid dynamic obstacles effectively. Comparative results are presented at the end to prove the strength "of the noval dynamic obstacle area method.

DECLARATION

The work submitted in this thesis is the result of my own investigation, except where otherwise stated.

It has not already been accepted for any degree, and is also not being concurrently submitted for any other degree.

UOM Verified Signature

R.G. Jayendra
(Candidate)
23rd March 2009

I endorse the declaration by the candidate.

UOM Verified Signature

Dr. Sisil Kumarawadu
(Supervisor)

CONTENTS

Acknowledgement

Many thanks are due first to my supervisor, Dr. Sisil Kumarawadu, for his great insights, perspectives and guidance throughout the entire duration of the study.

Author extends his sincere gratitude to Dr. J.P. Karunadasa, Head of the Department of Electrical Engineering, for providing him the Research Assistantship and excellent assistance during the study period. Many thanks and appreciations are due to Professor H.Y.R. Perera, Commissioner General of Public Utilities Commission of Sri Lanka as well.

Sincere thanks are also due to the officers in Post Graduate Office of the Faculty of Engineering, University of Moratuwa for helping in various ways to clarify the things related to academic works in time with excellent cooperation and guidance. Thanks are also due to the staff of the Department of Electrical Engincering for the support extended during the study period. Also, I wish to gratefully acknowledge the assistance extended by Mr. Ravipriya Ranatunga, Mr. Samitha Ransara, Mr. Sanjeewa Priyadharshana, Mr. Gamini Jayasighe, Mr. Lackshan Piyasighc, Mr. Kolitha Dharmapriya, Mr. J. Baek, Mr. Harshana Somapriya, Mr. Nadun Chamikara and my brother Chanuka Jayendra.

Many thanks are also duc to many individuals, friends and colleagues who have not been mentioned here by name in making this educational process as success.

Lastly, thc author expresses his dccp appreciation towards his family for their encouragcment and support. This work is dedicated to his beloved mother and late father.

List of Figures

List of Tables

ACRONYMS

USV	-	Unmanned Surface Vehicles
UGV	-	Unmanned Ground Vehicles
DAMN	-	Distributed Architecture for Mobile Navigation
GPS	-	Global Position System
INU	-	Inertial Navigation Unit
PFM	-	Potential field method
VOM	-	Velocity Obstacle method
DNC	-	Digital Nautical Chart
GRNN	-	Generalized Regression Neural Network

1.1 Applications of Unmanned Surface Vehicles (USV)

USV can be integrated for remotely controlled combat system ideally suited to meet force protection requirements in all maritime settings. By providing long range stand-off surveillance, identification and engagement capability, USV can be quickly deployed to defend high value assets including naval vessels, port operations, oil rigs and coastal power plants.

Protector is a name of an USV developed by Israel's Rafael Armament Development Authority in response to emerging terrorist threats against maritime assets [27]. That USV is stealthy, highly autonomous and can operate with general guidance from a commander in port, harbor and coastal waterways in a variety of roles, thanks to the plug-and-play design of its various mission modules, such as force protection, anti-terror, surveillance and reconnaissance, mine warfare and electronic warfare.

With integrated navigational sensors including GPS, navigation radar and video cameras, thc USV can conduct harbor surveillance even in busy waterways. Highly autonomous and remotely controlled, USV can successfully monitor waterways with general guidance from a commander and operator at sea or from shore - no matter how hazardous the condition [38]. The USV having an on-mount camcra allowing for day and night operation and has a forward-looking infrared laser range finder capability to detect and track targets in the near vicinity. The Boat Control unit's navigation sensors are used to obtain location, speed, heading and course data.

1.2 Obstacle Avoidance of Unmanned Ground Vehicles

An intclligent vehicular systcm which is a subtopic that comes under "Intelligent autonomous systcms" is an important research topic today, due to its importance in the field of "Autonomous surface vchicles and intelligent transportation systems". These systems can be classified according to the technology based on them and their method of implementation. This lies from basic vehicular management systems such as traffic lighting systems, container management systems and simple navigation systems to more advanced systems such as the systems getting feedback from the other compatible external systems, and external sources such as live feedback, whether information and so on. Predictive techniques have been developed to implement better inference systems.These methods allow some advanced modeling and comparison with historical baseline data and real-time data hence to provide an intelligent inference system which can deduct the best option at a time to control the system [31]. Collision avoidance techniques should be implemented within the platform or within an external system and a proper communication scheme should be maintained in order to prevent any life or material hazards, for these intelligent vehicular systems. In most of the times these inter platform communication scheme for short ranges (less than 400 meters) is accomplished by using IEEE 802.11 protocols or the Dedicated Short Range Communications standard being promoted by the Intelligent Transportation Society of America and the United States Department of Transportation [31]. In the case of long range communication schemes, this is accomplished by using infrastructure networks such as WiMAX (IEEE 802.16) or Global System for Mobile Communications (GSM).

Obstacle avoiding algorithms play major role in the overall process of navigation. Waypoint navigation without obstacle avoidance is given only limited capabilities to the USVs in a real-world mission. In order to provide more functionality and reduce the reliance on operator oversight, a robust obstacle avoidance capability must be added. More advanced behaviors can be added, such as autonomous recovery in the case of lost communications, target tracking and interception, etc., after adding a obstacle avoidance controller with algorithms.

In this research, a typical intelligent vehicle prototype was implemented and tested under laboratory conditions for its mobility, controllability and communication capabilities between the vehicle and a personal computer. Then another identical prototype was implemented to test, the communication capabilities between the two prototypes (inter vehicular communication capabilities) and communication between the vehicles and a personal computer. A dead reckoning algorithm is used as the tracking algorithm for the vehicles.

The novel interactive control paradigms in here have been testified as effective solutions for reliable collision avoidance of autonomous vehicular systems via computer simulations will be experimentally validated. The so-called interactive controller negotiates collision scenarios between two vehicular systems leading to cooperative maneuvers. The key point is that in order to avoid a probable collision situation, both the vehicular systems interactively carry out maneuvers. The hierarchical differentiation of the participatory vehicular subsystems is done by using a mater-slave concept. In the experimental validation using the prototypes, the advanced collision avoidance algorithms are implemented in the personal computer due to the limitations of memory and computational speed in the prototype. RS 232 serial communication standard is used for the wireless communication between three nodes (the computer and two prototypes) based on a suitable communication protocol that is developed specially for this scenario [32].

The developed prototypes were fully equipped with required hardware such as sensors and actuators. This has the motion control and position tracking abilities. The communication scheme was implemented via the RF broadcasting. The obstacle detection was done by ultrasonic sensors. Detection of the other prototype (as an obstacle, in collision conditions) was done by the relative distance, relative angle and relative velocity information, that are exchanged between vehicles via an inter-vehicular communication scheme that was implemented. Micro controller board was programmed in order to act as the central information processing unit [8].

1.3 Appling Ground Vehicle Technologies for Surface Vehicles

Current unmanned vehicles adhere to different levels of autonomy as defined by existing technology limitations and used sensors. Important operational characteristics related to unmanned vehicle functionality (aerial, surface and ground), include perception, intelligence and action. Here, the acquiring and use knowledge about the environment and itself is called the perception [17]. This is done by taking measurements using various sensing devices and then extracting meaningful information that should be used in all later tasks such as localization, planning, collision free motion control. The meaning of the Intelligence relevant to unmanned vehicles, is operating for a considerable time period without human intervention. This is associated with the learning and inference capabilities, which of the vehicle should be able to adapt to the environment. The action is the way that unmanned vehicle should travel from one point to another. The vehicle should utilize predefined and acquired knowledge to move in dynamic environments without involving humans in the navigation loop. So, the algorithms and technologies developed for unmanned ground vehicles can apply for the surface vehicles as well due to those similarities.

1.4 Potential Field Method for Obstacle Avoidance

During the past few years, potential field methods (PFM) for obstacle avoidance have gained increased popularity among researchers in the field of robots and mobile robots [34]. The idea of imaginary forces acting on a robot has been suggested by Khatib in 1985 [18]. In these approaches obstacles exert repulsive forces onto the robot, while the target applies an attractive force to the robot [41]. The sum of all forces, the resultant force, determines the subsequent direction and speed of travel. One of the reasons for the popularity of this method is its simplicity. Simple PFMs can be implemented quickly and initially provide acceptable results without requiring many refinements.

PFM cannot be applied for dynamic obstacles directly. An ongoing unpublished research work is there to apply potential field method for dynamic obstacle avoidance. The velocity dipole is used for that. The velocity dipole field is presented for real-time collision avoidance of mobile robots. The direction of motion of the obstacle is used as the axis of the dipole field, and the speed of the obstacle is used to proportionally strengthen the dipole field. The elliptical field lines of the dipole field are useful to skillfully guide the robot around obstacles, quite similar to the way humans avoid moving obstacles. That system seems to have the capability of a new real-time collision avoidance strategy and it will overcome the weaknesses in the conventional potential field method.

The Software developed by Lee Feng [20] for potential field applications was utilized in this research to compare the proposed methods with the potential field method .

1.5 Morphin algorithm for path planning

Morphin is an area-based algorithm and it analyzes obstacles in the area which can disturb its navigation. It projects all the possible paths to the front initially. To select from among multiple paths, path evaluations are assigned to all possible candidate paths according to how effectively each path would drive the rover toward its goal point. The path that would lead directly toward the goal with less obstacles is given the highest evaluation; other paths are assigned lesser values according a predefine function [21]. These evaluations are then combined with the user's preferences to determine the overall best command, which is then sent to the rover to be executed. The cycle time for this process is about 1-2 seconds, with the stereo computations taking up about 75% of the total time.

This algorithm is novel but the approach has a long history of applications in real-world systems (including the Mars Rovers) and has its lineage back to the Carnegie Mellon University Morphin algorithm and Distributed Architecture for Mobile Navigation (DAMN) [29].

In the practical applications, the problems which are mainly attributable to an abundance of noise, particularly in the stereo-produced obstacle maps and Global Position Systems (GPS) are jammed or the Inertial Navigation Unit (INU) drifts can create problems to the previous algorithms since they are heavily depend on Obstacle positions. So another approach had to be chosen to develop new algorithms to overcome above particulars for smooth and safe navigation of the USV.

An applying ground vehicles algorithm for surface vehicles and PFM were done previously. So Morphin approach was chosen due to it's proven capabilities for ground robots like famous Mars Rovers and Lunar Rovers [21]. The algorithm is novel, but the approach has a long history of applications in real-world systems and has its lineage back to the DAMN of Carnegie Mellon University .

1.6 Defining safety distance for path planning

Safety distances need to be defined for different obstacles considering their geometrical shapes. They might change around obstacles center of gravity. That was not considered in this study. Therefore circular safety distance is defined for the safety of the USV. The ways of defining safety distance is shown in Figure 1.1.

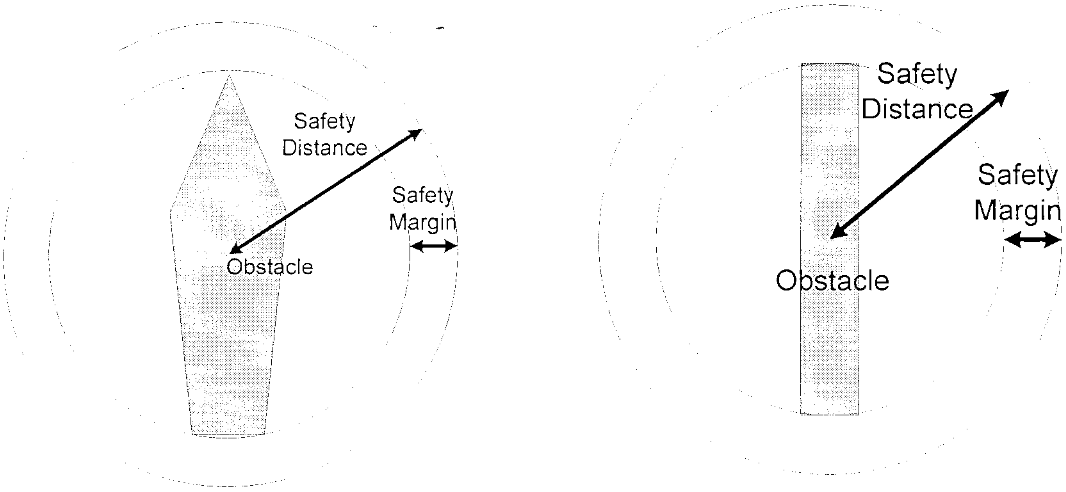


Figure 1.1 – Safety Distance for Obstacles

For an example the length of the USV named Protector used by IIsraeli Sea Corps is 9 meters.

1.7 Dynamic obstacle avoidance

A lot of researches has been carrying out to avoid dynamic objects and unable to find a best solution for that although comparatively good approaches has been presented. Canny and Reif [13] showed that motion planning for a point in a plane with bounded velocity in the presence of moving obstacles is Non-deterministic Polynomial-time hard. Aggarwal and Fujimura [22] show that a more optimal solution can be found by adding a third dimension of time and plotting the location of the moving obstacles along that three-dimensional (3D) structure. Fujimura and Samet [15] provide yet another solution, but even they admit the solution is best with few moving obstacles.

A solution for dynamic obstacle avoidance is presented by Space and Naval Warfare Systems Center, San Diego for the requirement of robust USV operation in a real world environment, primarily focusing on autonomous navigation, obstacle avoidance, and path planning. Velocity Obstacle method (VOM) is one of the good methods for dynamic obstacle avoidance is utilized mainly for those developments. To avoid moving obstacles and maintain the desired path set by the user, the safe velocity ranges using the Velocity Obstacle method [25] have to be determined by the controller. This algorithm transforms a moving obstacle into a stationary one by considering the relative velocity and trajectory of the USV with respect to the obstacle. After producing a collision area called the Velocity Obstacle, defined using the relative velocity vector, the algorithm returns a set of USV velocity vectors which are guaranty the collision avoidance. This transformation and collision area detection reduces the complexity of the path planning among moving obstacles with respect to time. This is used as a first pass to avoid moving obstacles. In the case that changing velocity the controller has to change the path by creating projected obstacle areas for each obstacle and determining a safe alternative route.

1.8 Sensor considerations

As with any unmanned vehicle attempting to navigate in a complex environment, good sensor data is critical, and getting good data is often the most difficult part of the project of developing a USV. The oceanic environment poses many challenges including waves, spray, and a disordered obstacle setting. There are some advantages to the marine environment including well charted operating areas, absence of negative obstacles (holes or cliffs), a mostly planar surface (except for the waves), no vegetation, etc. It's important that the sensors are selected to make the most of the environmental advantages and to provide the best data possible in the challenging territories.

The sensors for the obstacle avoidance need to provide data about obstacles in the far-field (e.g., >200-300 yards) and provide state information (position, course, and speed) for the moving obstacles.

High-resolution and at a much higher rate data is needed about the obstacles in close proximity to the USV (e.g., <200-300 yards). Some of these sensors are typically not found in the commercial marine industry but many have been used extensively in UGV programs.

1.8.1 Radar Contacts

Standard marine radar (Furuno) with a third-party PC controller can be used for USVs. The controller, developed by Xenex Innovations Ltd., provides a digital networked interface for the radar. The Xenex system provides an API to access the radar data and controls as well as an Advanced Radar Plotting Aid (ARPA) Software Development Kit, which provides algorithms to automatically acquire and track up to 100 contacts [23].

One significant problem with the radar is that it tends to classify noise from the shoreline return as contacts which are often shown to be moving at a significant velocity and in the direction of the USV. These false contacts are obviously detrimental to the successful operation of the path planner. To mitigate this problem, the on-board nautical chart server can be used to calculated polygons that follow the shoreline and structures along the shoreline. The radar contacts are compared with these polygons and those that fall inside a polygon are rejected and deleted from the radar's list [28].

Laboratory (JPL) for a number of years to transition technology to its UGV programs. That work is now being extended to the USV domain [14]. The stereo vision system provides high-resolution 3D data about the near-field environment, which can be converted into a 2D obstacle map and fused with data from the other sensors.

Stereo vision is capable of providing very high quality 3D data but also has the disadvantage of requiring precise calibration every time the cameras are mounted. There is also the risk that the cameras may move relative to one another slightly which will affect the calibration and result in erroneous data. So the monocular vision with sophisticated algorithms can be utilized for that as well.

2.1 Sensor selection for prototypes

The proposed vehicular prototype has ultra-sonic range sensors to detect collision conditions encountered by the prototype, a digital compass module to obtain the heading angle, optical shaft encoder modules to get the position of the prototype, RF transceiver module for communication, servo drivers, and a micro controller to process the information and to give the control signals to the actuators. A block diagram representation of the complete system is given in Figure 2.1.

The main objective of the developed prototypes is to test, fine tune, and experimentally validate intelligent collision avoidance algorithms for ground vehicles. So, the features expected in the vehicular platforms should be identified at the initial stage of the project. Individual prototypes should include a suitable controller which is capable of implementing the collision avoidance computational intelligence algorithms. The motors selected need to be easy to control. Most importantly, the prototypes should be able to communicate each other. The heading angles of the vehicles are vital information in order to realize successful collision avoidance maneuvers.

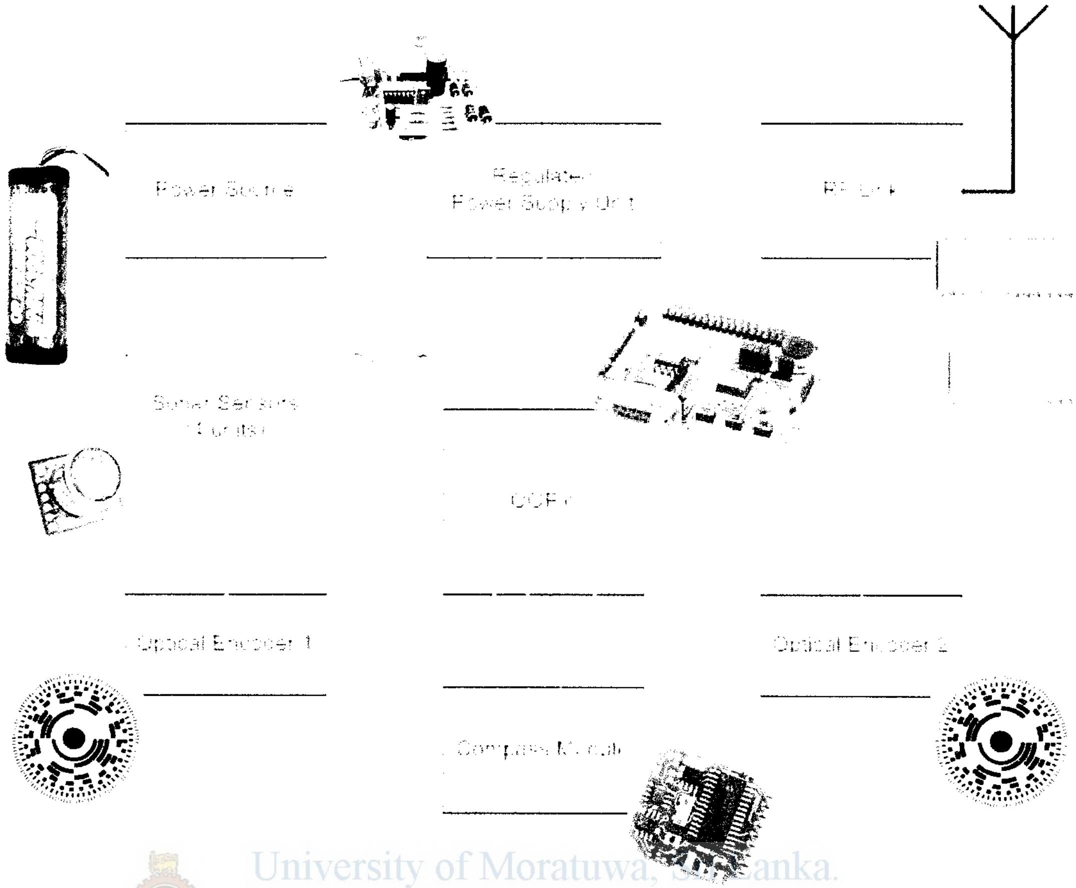


Figure 2.1 - Basic Block Diagram of the System

2.2 Development of prototypes

2.2.1 Digital Controller Selection

Digital controller is the heart of the system and should be carefully chosen. The controller should have a good memory capacity and a very high frequency. Most importantly, the programming of the controller should be straightforward. In this application, the onboard controller controls two motors, while executing the other instructions.

The OOPic-R [3] was selected for the project. It provides 16 digital I/O lines including power and ground connection. Its smaller size suits well for prototype development.

The speaker onboard OOPic-R is useful in the developmental stage. The speaker can be used to take required outputs and warning signals at the development stage. All the modules were tested separately before integrating. The speaker was deployed to check the output values of those modules successfully.

I^2C network connector of OOPic-R can be used to network two or more OOPic-R s. Multiple power lines are provided to use different power consuming modules (Figure 2.2). The jumpers onboard can be used to set the different power outputs. OOPic-R contains 3 onboard LEDs having different colors. They are internally connected to three different IO lines. Those LEDs are used in the development stage for debugging and monitoring. The RS232 serial port connector can be used to communicate with PC via MAX 232 chip. A separate three-pin connector provided direct and easy connection for the LCD displays. The green LED connected near the programming connector is lit brightly when power supply is good. It is a good indicator of supply battery level. Reset button on the board can be used to reset the controller when that becomes stuck.

OOPic has an object-oriented operating system that has been pre-programmed into a Microchip PIC. It also provides an object-oriented language model design to interact with the electrical hardware components that are attached to the PIC. Hardware interface is designed using programming software by creating objects and setting their properties to define their behavior and interaction with the hardware. These objects can also be interconnected to form a virtual circuit and then utilize this hardware interface and its associated virtual circuits by writing a program that controls and responds to hardware events that occur.

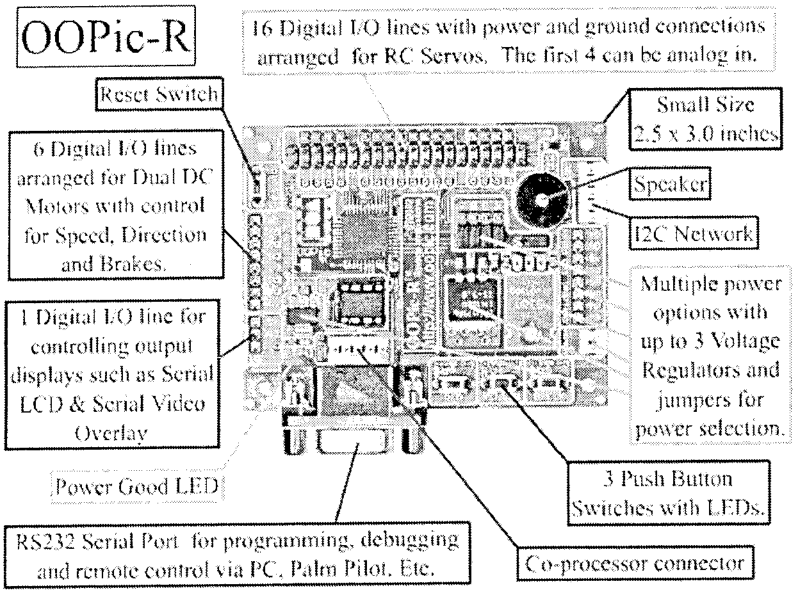

OOPic-R
16 Digital I/O lines with power and ground connections arranged for RC Servos. The first 4 can be analog in.
Reset Switch
Small Size 2.5 x 3.0 inches
6 Digital I/O lines arranged for Dual DC Motors with control for Speed, Direction and Brakes.
Speaker
I2C Network
Multiple power options with up to 3 Voltage Regulators and jumpers for power selection.
1 Digital I/O line for controlling output displays such as Serial LCD & Serial Video Overlay
Power Good LED
3 Push Button Switches with LEDs.
RS232 Serial Port for programming, debugging and remote control via PC, Palm Pilot, Etc.
Co-processor connector

Figure 2.2 - OOPic R Micro controller board

2.2.2 Digital Compass Module Selection

Path of the vehicle travels should be mapped to analyze the workability of the theories .There are two major options to map those paths. They are GPS mapping and employing a suitable dead reckoning algorithm. GPS module is needed for GPS mapping. But even differential error correcting GPS has an error around 5m [16]. But as each prototype developed is 30cm × 20cm in size, GPS mapping is not an option and a dead reckoning algorithm is adopted for path mapping.

A digital compass and an optical encoder were used in the dead reckoning algorithm. The CMP 03 digital compass module by Devantech Ltd was selected for this study (Figure 2.3). Especially, together with the optical encoders, this compass is meant to be used for dead reckoning purposes. The CMP 03 digital compass uses the Philips KMZ51 magnetic field sensor, which detects the Earths magnetic field. The output from the two of them mounted at right angles to each other is used to find the heading angle. The compass readings can be obtained both from the I^2C channels as well as a PWM signal. The resolution of the compass is 0.1 degrees.

The compass needs to be calibrated in the area it is being used.

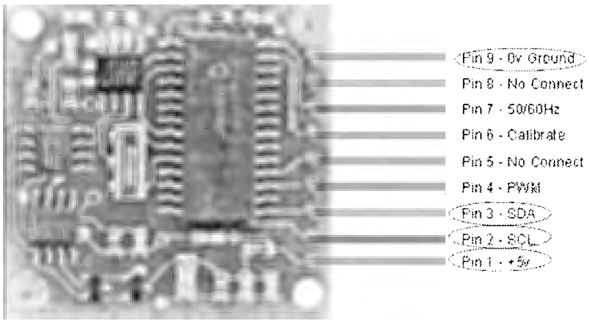


Figure 2.3 - Digital compass module

I^{2}C object in the OOPic multi-language compiler was used to access the digital compass. I^{2}C node address was set to 0×C0. Location 1 in the compass module was access to take heading angle value from 8 bit value. The value that came from the module was between 0 to 255. This was mapped to the 0-360 degree range.

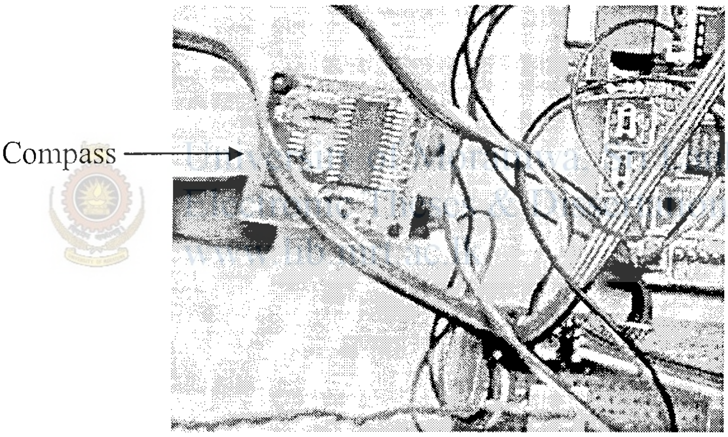


Figure 2.4 - The way of mounting the compass to the prototype

The mounting position of the compass was very important due to its high magnetic field sensitivity. Because the DC servos used to power the wheels generate magnetic fields which may interfere with the digital compass, the digital compass was mounted as far as possible from the DC servo motors as shown in Figure 2.4.

2.2.3 Digital Encoder and Encoder Wheel

The optical encoder module has three channel incremental encoders with a code wheel is choosen as shown in Figure 2.5 [3]. The speed of the two wheels of the prototype is intended to be measured using this. Optical encoder outputs together with that of the digital compass can be used for dead reckoning in navigation purposes of the prototypes.

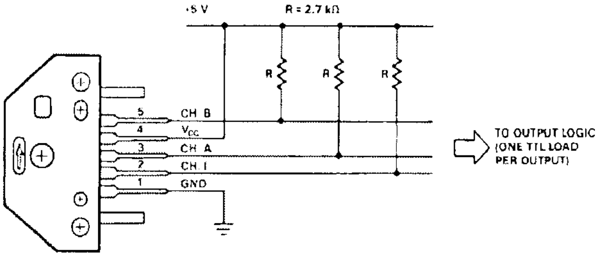


Figure 2.5 - Connection arrangement of the encoder

Qencode encoder object was used to get data from the digital encoder. Position property of the Qencode object was utilized to take the encoder position. Channel A and Channel B were connected to IO lines on the OOPic and the numbers of the IO lines were set to the Qencode object properties.

The encoder wheel and the Optical sensors should be mounted with care to have a good alignment as shown in Figure 2.6.

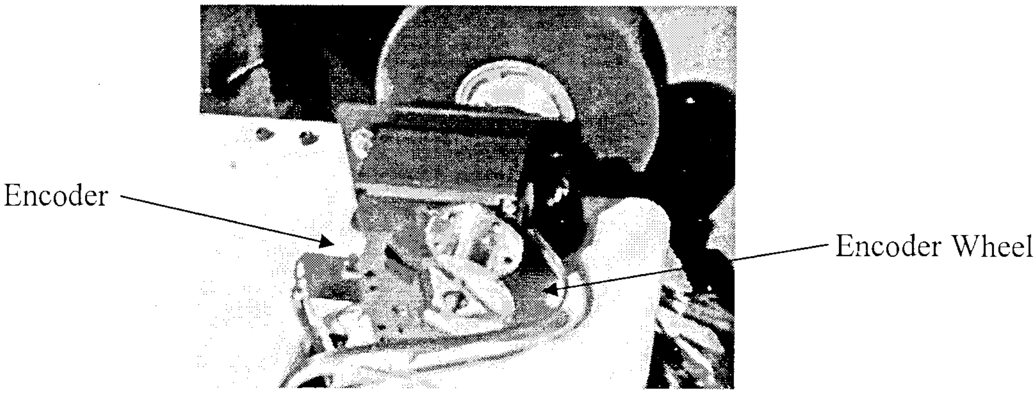


Figure 2.6 - The way of mounting the encoder to the prototype

2.2.4 Servo Motors

The HS-422 standard deluxe servomotors, by Hitec RCD Inc., are 3-pole ferrite type motor attached with an in-built potentiometer. Such a motor is shown in Figure 2.7. These motors are with the control system of pulse width 1500μs neutral type [11].

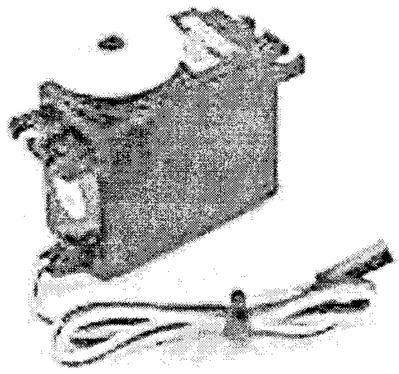


Figure 2.7 - HS-422 Servo Motor

The potentiometer was removed from the servo and a gear wheel adjustment was done for continuous run. The practical speed of the prototype was quite below the required speed. So the controlling circuit of the servo was removed and power was directly given to the motor through a motor driver (L298N) chip. A supply voltage of 7V was given to the servo when maximum speed was required.

2.2.5 Ultra-Sonic Range Sensors

Sensing the obstacles around each vehicle prototype while moving, is important. The SRF235 ultrasonic range sensors are chosen for that. Those by Devantech Ltd., are for the purpose of detecting obstacles. The ultrasonic sensors are not meant to identify the other vehicle prototypes in close proximity .The adjacent prototypes are meant to be identified, by each other, with the RF communication between them or by adding a shield to prototype (minor adjustments).

The selection of this type of ultra-sonic sensor with a narrow beam pattern, as indicated in Figure 2.8, gives the opportunity to detect the obstacles, but not the floor as an obstacle, a false alarm. Mounting arrangements of these sensors were also considered with special concern.

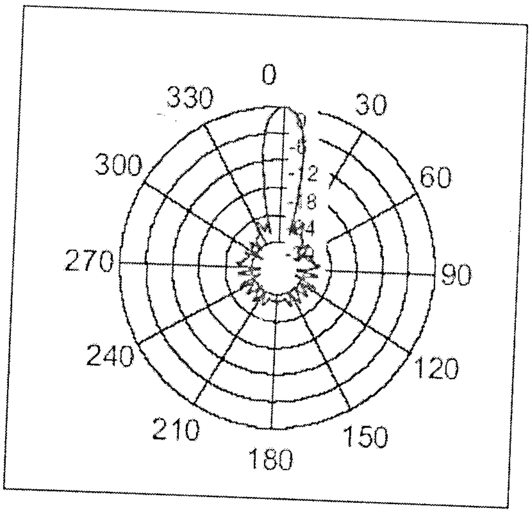


Figure 2.8 - Beam pattern of the SRF235 'Pencil beam' ultrasonic sensor

The ultra-sonic sensor is with a single transducer for both transmit and receive. Therefore, there is a blanking zone of 10cm, so the effective range is 10cm to 1.2m. Communication with the SRF235 ultrasonic rangefinder is via the I^{2}C bus. Therefore, this is easily connected to the OOPic R+ with its capability of I^{2}C. In order to connect these sensors to the OOPic R+, the address of the sensors have to be changed. Figure 2.9 is the picture of SRF235.

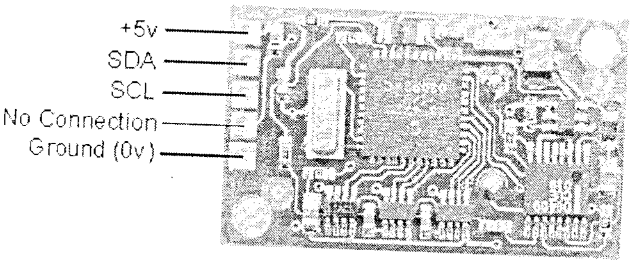


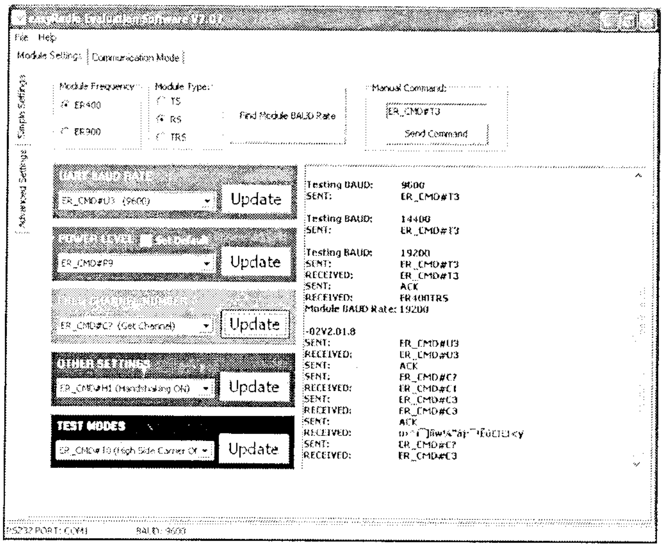
Figure 2.9 - SRF235 Pencil beam ultrasonic sonar sensor

2.2.6 Inter Vehicular Communication scheme

Inter vehicular communication was the most important part of the system. ER400 radio modules, shown in Figure 2.10, were used for that. Few important features like several channels, low power consumption, very stable operating frequency and good bandwidth for data transmission, reliable communication and good data rates were expected from the RF module. Several Channels were needed to establish a good communication. The chosen modules can have 10 different channels, which meet our requirement.

Figure 2.10 - LPRS ER400 Radio Modules

The programming software, which can download from the web, was used to give commands to the module as shown in Figure 2.11 [4].



Module Frequency: ● ER400 ○ ER900

Module Type: ○ TS ● RS ○ TRS

Manual Command: ER_CMD#T3

ER_CMD#U3 (9600)

ER_CMD#P9

ER_CMD#C? (Get Channel)

ER_CMD#H1 (Handshaking ON)

ER_CMD#T0 (High Side Carrier Of

```
Testing BAUD:     9600
SENT:             ER_CMD#T3

Testing BAUD:     14400
SENT:             ER_CMD#T3

Testing BAUD:     19200
SENT:             ER_CMD#T3
RECEIVED:         ER_CMD#T3
SENT:             ACK
RECEIVED:         ER400TRS
Module BAUD Rate: 19200

-02V2.01.8
SENT:             ER_CMD#U3
RECEIVED:         ER_CMD#U3
SENT:             ACK
SENT:             ER_CMD#C?
RECEIVED:         ER_CMD#C1
SENT:             ER_CMD#C3
RECEIVED:         ER_CMD#C3
SENT:             ACK
RECEIVED:         [illegible]
SENT:             ER_CMD#C?
RECEIVED:         ER_CMD#C3
```

RS232 PORT: COM1 BAUD: 9600

Figure 2.11 - Evaluation Software

That software was used to change the baud rate, power level and channels of the modules. The baud rate was set to 9600.The channels of RF modules were set according to the requirements and to avoid jamming as shown in Figure 2.12.

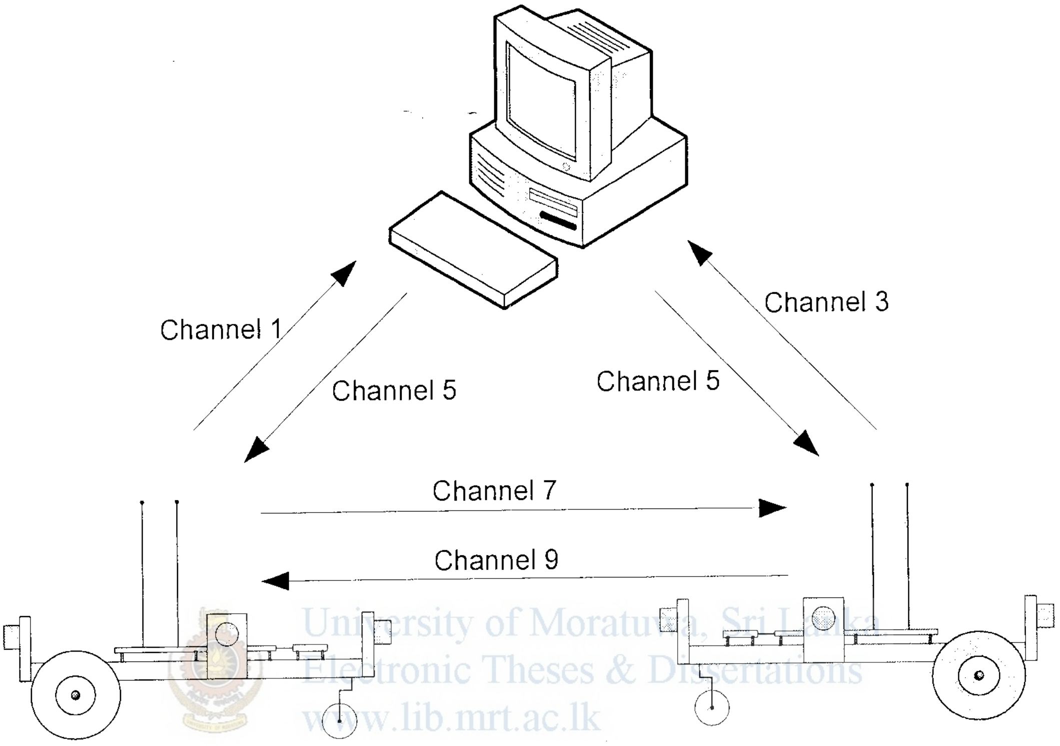


Figure 2.12 - Communication Channel Dedication

2.2.7 ER400RS Receiver

The Block diagram of the receiver module is given in Figure 2.13. Pin numbers 6 was used to give commands to micro processor to change the internal settings of the module. A programming software was used to give commands to the module [4].

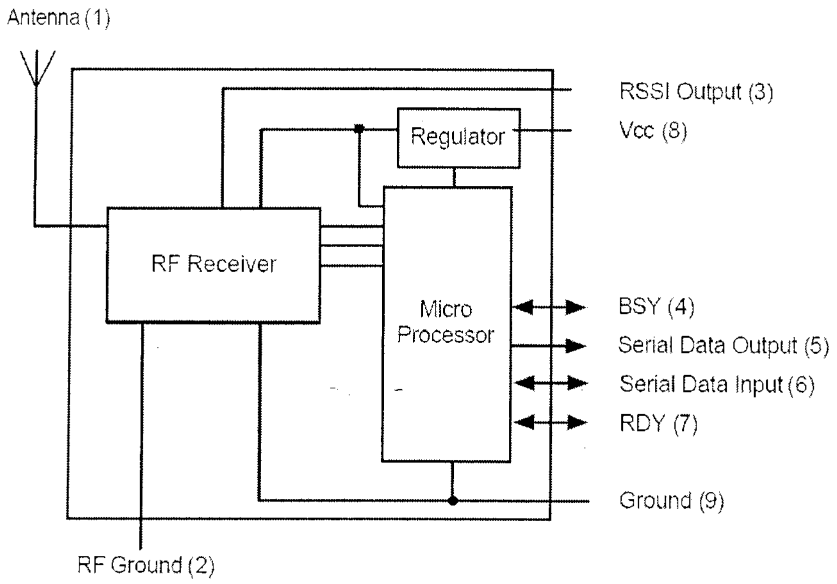


Figure 2.13-Receiver

It was quite easy to work with these modules, because feedback characters were received after every command. Thus, the verification of the changes was straightforward.

2.2.8 ER400TX Transmitter

These transmitter modules do not provide any feedback to the programming software. Only the serial data transmission line was provided as seen in Figure 2.14. So it was quite difficult to change and verify the settings of the transmitter modules. Two software windows were employed to change the settings of those transmitter modules. One window connected to com1 serial port was used to send serial data while the other window connected to com2 serial port was used to receive data from the receiver module. The RF Channel setting commarnds are presented in Table 2.1.

Initially, 19200bps baud rate was used to give command to the module. That baud rate can be change by sending baud rate setting command. An unambiguous baud rate has to be used when dealing with transmitter modules. The baud rate differences were caused data corruptions.

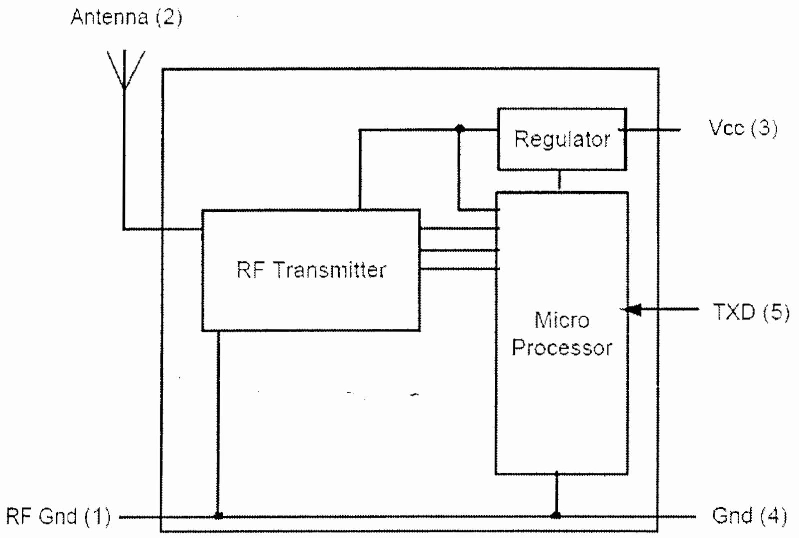


Figure 2.14 – Transmitter

Table 2.1 - RF Channel setting commands

Command	Channel number	Frequency
ER_CMD#C0	0	433.23 MHz
ER_CMD#C1	1	433.30 MHz
ER_CMD#C2	2	433.45 MHz
ER_CMD#C3	3	433.55 MHz
ER_CMD#C4	4	433.68 MHz
ER_CMD#C5	5	433.83 MHz
ER_CMD#C6	6	433.88 MHz
ER_CMD#C7	7	434.00 MHz
ER_CMD#C8	8	434.15 MHz
ER_CMD#C9	9	434.35 MHz

2.2.9 Serial Interface Circuit Design

RS232 serial work with 0 to 15V logic levels while RF modules work with 0 to 5V. The MAX232 chip was used to do the level conversion. The circuit diagram of the serial interface circuit is given in Figure 2.15.

A two-way switch was employed in the circuit to change data transmission mode to setting changing mode. Because the data transmission line from the serial port had to connect to serial data transmission pin via chip when data transmission was need and it had to be connected via chip to serial data input of the receiving module when the settings change was needed.

The 7805 regulator was used to give power to the RF transmitter and receiver modules. The RF modules were mounted to the board as shown in Figure 2.16. The regulator module was used to achieve a smooth power supply to the transceiver, hence minimized the error signals that can be caused by varying input power.

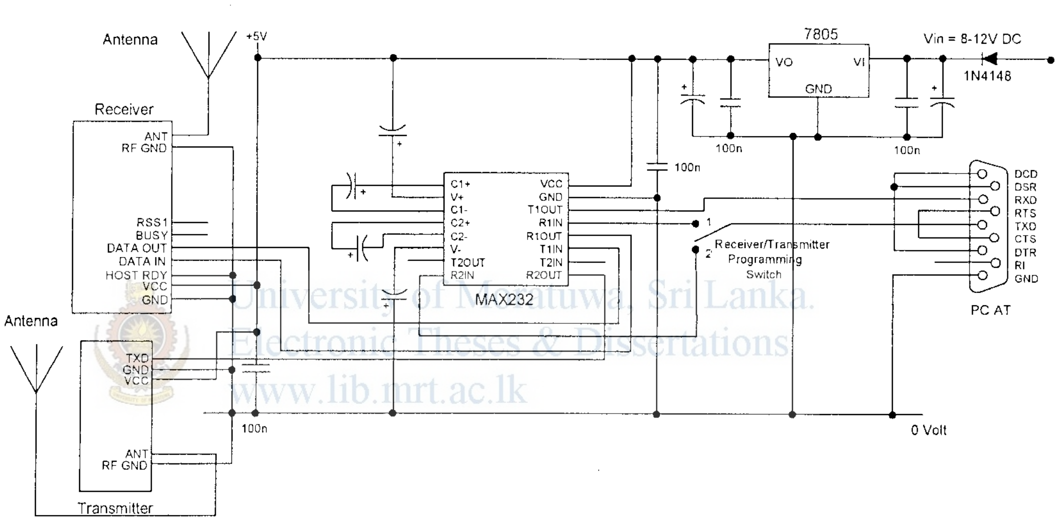


Figure 2.15 - Serial Interface Circuit

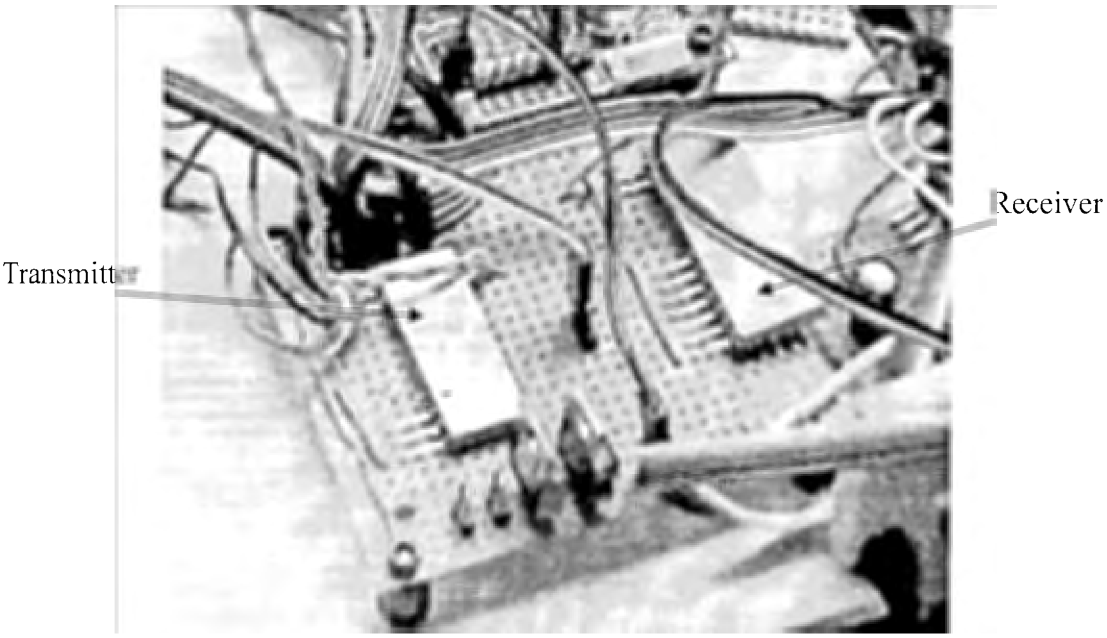


Figure 2.16 - The way of mounting Transceiver module

2.2.10 Integrating Sensors to the Controller

Integrating sensors to the controller was done using the I^{2}C bus. Different hexadecimal addresses were assigned to different components to access through the I^{2}C bus. Five modules were connected to the I^{2}C bus. Power for those modules was given directly from the 5V regulator. Side elevation and plan of the proposed prototype is presents in Figures 2.17 and 2.18. OOPic Basic Program for Vehicular Prototypes is given in appendix A.

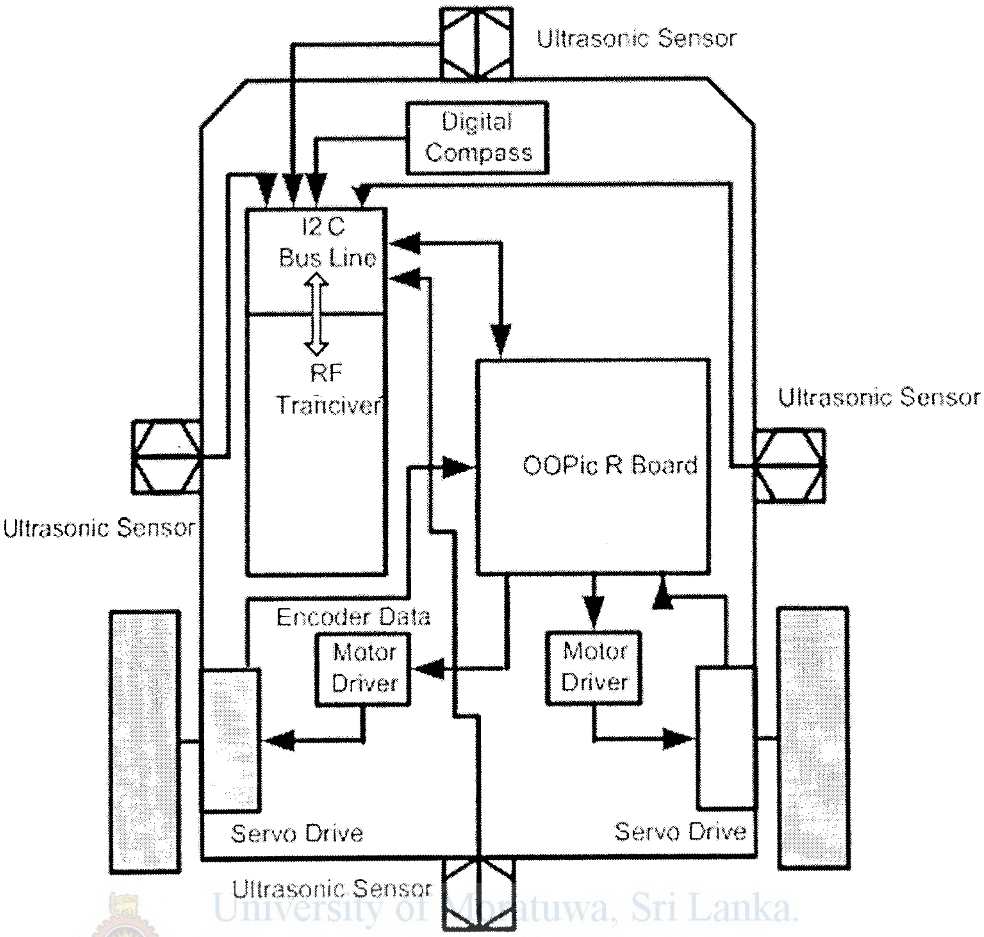


Figure 2.17 - Proposed Prototype (Plan)

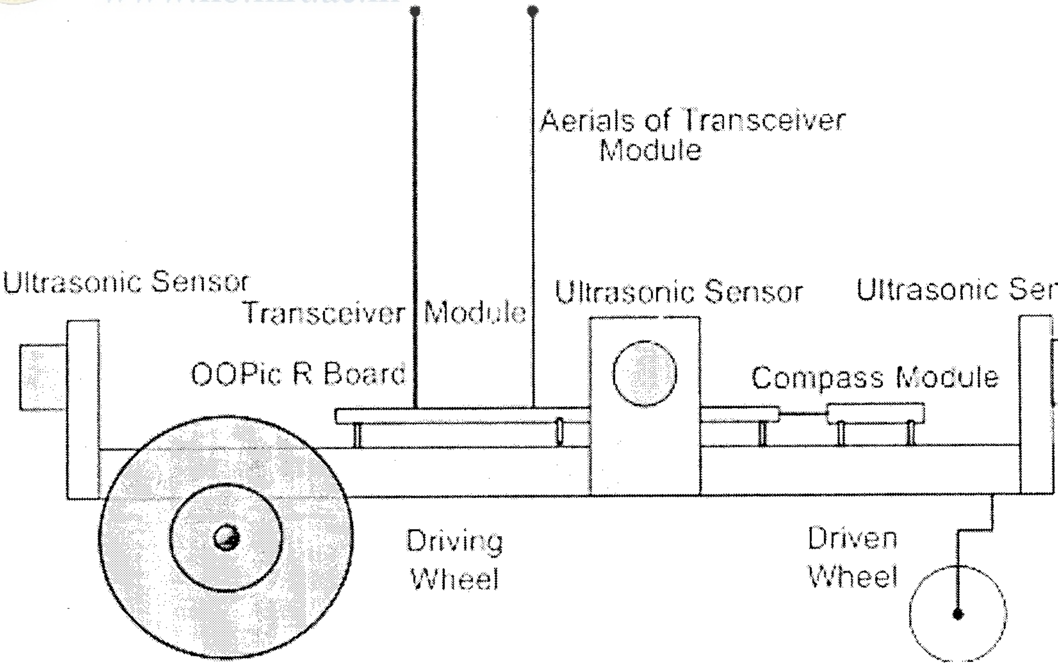


Figure 2.18 - The proposed prototype (Side elevation)

The actual implementation of the prototype can be present as follows (figure 2.19).

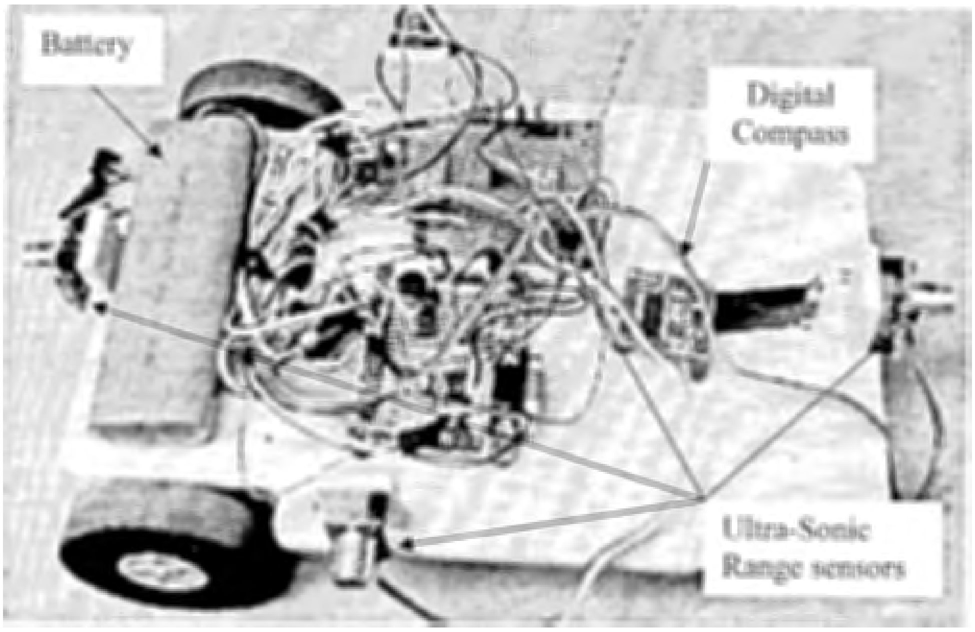


Figure 2.19 – Developed vehicle

2.2.11 Interfacing Software for Prototypes

Interfacing software was developed to give control signals to the vehicle. Visual Basic 6 was used to develop the software interface for the prototype. GMS ActiveX controllers were used as joysticks and digital compass. . It has the ability to view the current position of the prototypes in its sketch pad, the current heading angle of each prototype, current coordinates of each prototype in sketch pad, communicate with each prototype, give control signals to prototypes via the established RF link through the serial port, and draw the path followed by each prototype in the sketch pad.

2.3 Experimenting with prototypes

Two identical prototypes were developed to carryout the testing process of obstacle avoidance and inter vehicular collision avoidance.

2.3.1 Position Tracking Algorithm

As GPS is not an option because of the small distances involved in the experiments with scaled down prototypes, the following dead reckoning algorithm is adopted for position tracking. The speed of the prototype was calculated within the program by using the input values of two encoders and averaging its value similarly as in equations 2.1 and 2.2. The x(k) and y(k) positions calculated as below.

$$x(k) = x(k-1) + \dot{x} \times \sin(\theta) \times (\Delta t) \quad \text{-----------------------} \quad (2.1)$$

$$y(k) = y(k-1) + \dot{x} \times \cos(\theta) \times (\Delta t) \quad \text{-----------------------} \quad (2.2)$$

Where, θ is the heading angle of the prototype.

Figure 2.20 elaborates this further.

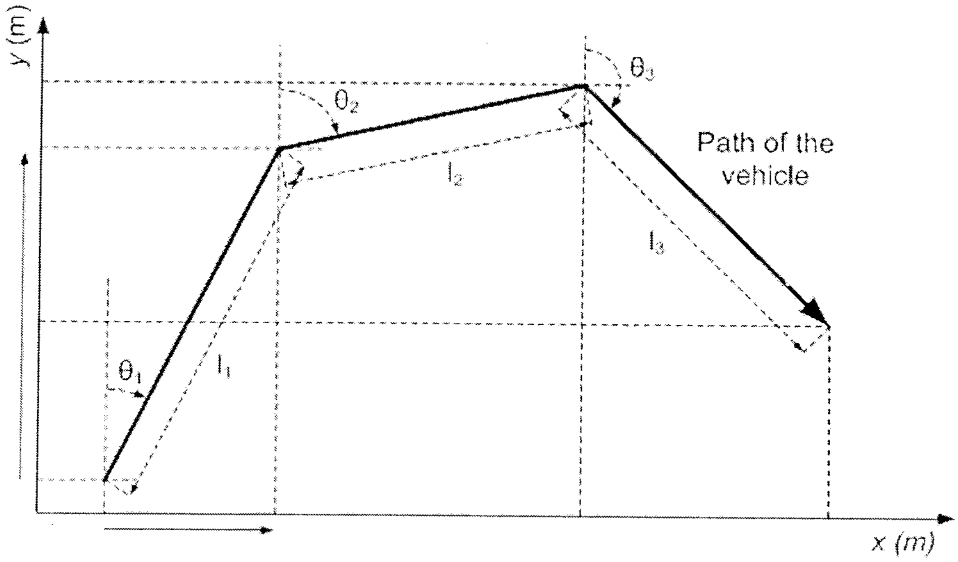


Figure 2.20 - Position Tracking with Compass

Figure 2.21 shows the program that is executed in OOPic, which used to maneuver the vehicle with the compass. The program begins with initialization of the objects that are used to represent the vehicle. Basically, the sensors and the objects are needed for communication purposes. After that, inputs from the sensory devices are fed to the execution through the input-output lines that are already set. Position manipulation takes place after the inputs are analyzed to minimize the errors that can occur due to wrong input values. Then, the program transmits the calculated current position of the vehicle to the other vehicle, or to the computer via the RF link .

2.3.2 Peripheral Obstacle Avoidance

This obstacle avoidance algorithm is for stationary obstacles such as walls and other barriers. This mode activates when the distance to the obstacle is less than 15 cm [32]. The algorithm used in this study can be presented as follows.
If Obstacle on Left and distance decreasing, then Turn Right for 3 [s]
If Obstacle on Right and distance decreasing, then Turn Left for 3 [s]
If Obstacle on Front & Left and distance decreasing, then Turn Right for 3 [s]
If Obstacle on Front & Right and distance decreasing, then Turn Left for 3 [s]
If Obstacle on Front then Stop, Reverse for 2 [s], Stop and Turn Right for 3 [s]
If Obstacle on Back then Stop (if Reversing) and Turn Right for 3 [s]

2.3.3 Collision Avoidance

This algorithm applies when the relative distance of the vehicles is less than or equal to 30 cm. This requires advance inference techniques, that processed by fuzzy based inference engine.If a prototype in Collision situation then Stop and take a “right turn” for 3 [s]. If still in collision situation Stop and take a “left turn” for 3 [s] Else “reverse” the platform for 3 [s] .

2.3.4 Position Tracking without the Digital Compass

The compass module plays a vital role in the position tracking algorithm. But some errors may occur due to the magnetic fields generated by servo motors. This can often be the case in experimenting with the physically scaled down prototypes as keeping an adequate distance between the motors and the digital compass is not practical. Here, we present a method to calculate the heading angles to be used when the compass readings are not reliable [36].

Figure 2.22 shows a typical path that the vehicle follows and the footprint of its tires. The change of angle from the previous position can be approximated using the wheel movements and simple geometry as elaborates in equation 2.3 and 2.4.

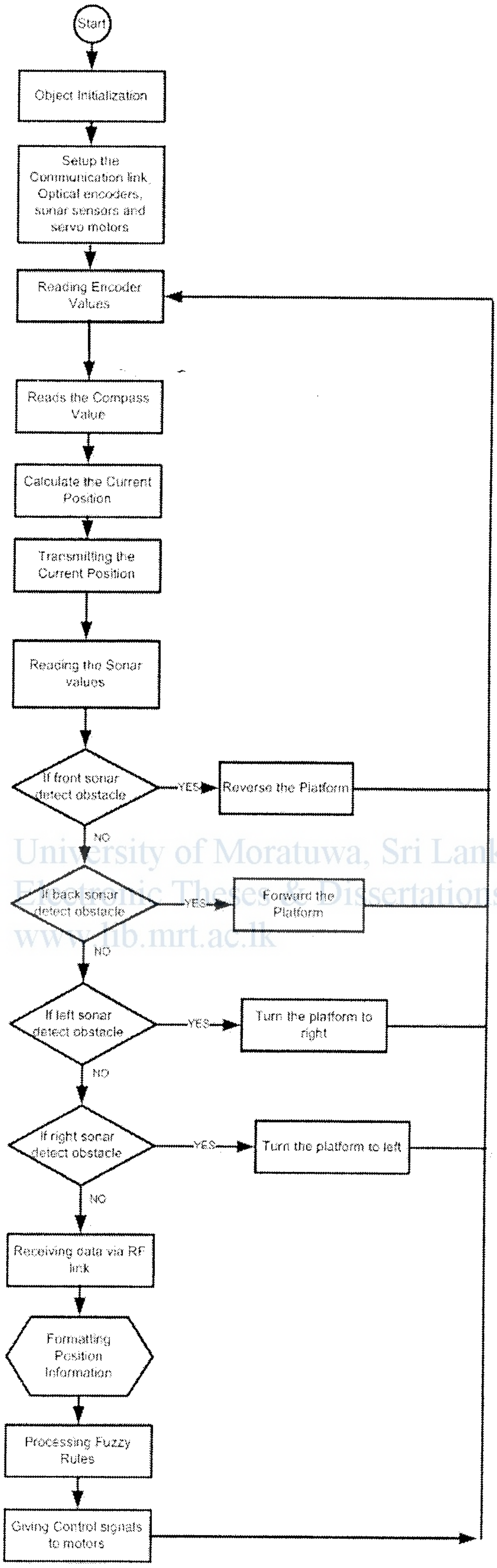

Start
Object Initialization
Setup the Communication link, Optical encoders, sonar sensors and servo motors
Reading Encoder Values
Reads the Compass Value
Calculate the Current Position
Transmitting the Current Position
Reading the Sonar values
If front sonar detect obstacle
YES
Reverse the Platform
NO
If back sonar detect obstacle
YES
Forward the Platform
NO
If left sonar detect obstacle
YES
Turn the platform to right
NO
If right sonar detect obstacle
YES
Turn the platform to left
NO
Receiving data via RF link
Formatting Position Information
Processing Fuzzy Rules
Giving Control signals to motors

Figure 2.21 - Executed program in OOPic

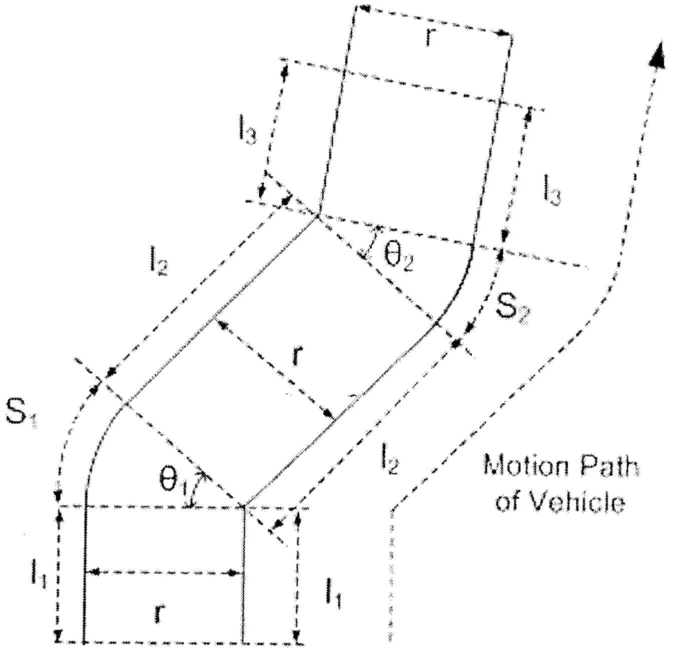


Figure 2.22 - Position Tracking without Compass

$$\theta_1 = \frac{S_1}{r} \qquad \text{--}(2.3)$$

$$\theta_2 = \frac{S_2}{r} \qquad \text{--}(2.4)$$

Where S is the traveling distance of the wheel. r is length between two wheels.

Therefore, the total angle change within the course relative to the initial direction can be obtained by considering the above instantaneous angular changes. The OOPic reads the encoder readings every 2 seconds and executes the algorithm as illustrated in Fig 5 where Enc1 and Enc2 represent the readings of encoder 1 and 2, respectively.. One limitation of this method is that floating points may occur during the execution requiring some offset actions to minimize the errors caused by this.

If the difference between two encoder values is less than 20, previously calculated θ value was forwarded to the next step assuming a straight motion of the prototype or the prototype is not moving. This program segment can be present in a flow layout as in figure 2.23.

2.3.5 Fuzzy Based Controlling

The position information obtained via the methods explained previously is sent to the central PC that implements the fuzzy controller. Position information was processed and the fuzzy base controlling signals were given to both prototypes via RF link. The block diagram of components including fuzzy inference engine is given in figure 2.24.

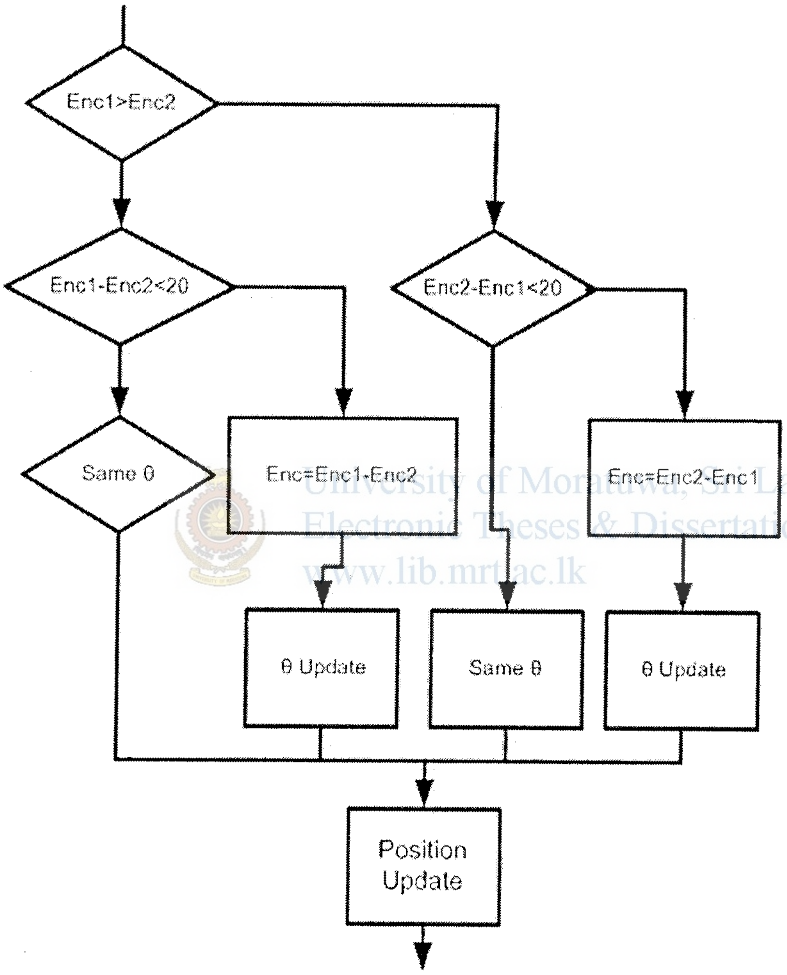


Figure 2.23 - Position Tracking without Compass (block diagram)

The encryption module therein acts as a converter of defuzzyfied output to the code that the communication module can understand.

The fuzzy based controlling functions were developed by means of MatLab simulation software simulink. ANFIS tool box was employed for that. The input membership functions were defined initially.

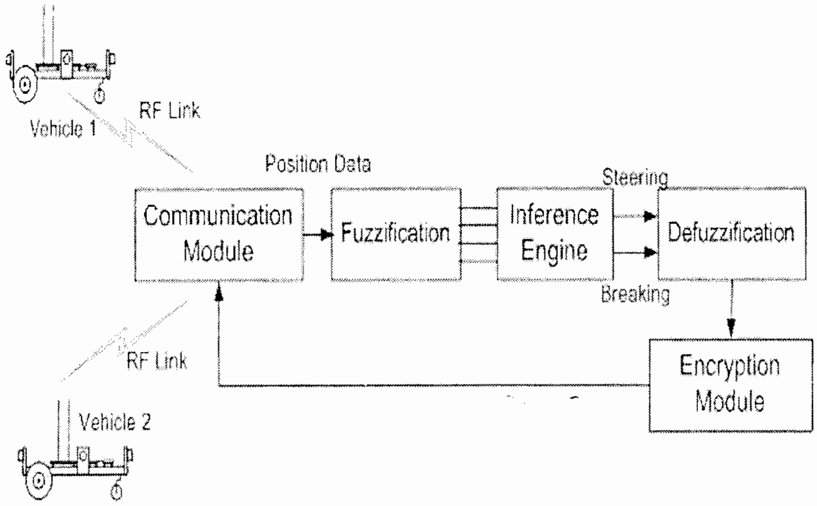


Figure 2.24 - Fuzzy Based Controlling

2.3.6 Collision Condition Function

This as the name implies, to identify the vehicles are in collision states to trigger the collision avoidance scenario. The theory of relative velocity between two vehicles was employed for this process. The program virtually creates the inertial frame with respect to one vehicle and check whether the vehicles are in collision condition by the aid of the relative distance and velocity. By means of the path of a vehicle relative to the other, some collision situations can be defined.

If the collisions between two vehicles occur in line, means that a direct collision, that state was defined as ‘in line collision’. The other two states, named as ‘in line of likely collision’ and ‘not inline collision’ following the same methodology that was previously stated. This is essential to quantify the inputs, which needs for fuzzification process. The range of values for the collision condition function was taken as -3 to 0, and decided the center of the Gaussian membership function as follows[24].

Table 2.2 - Centers of Gaussian Membership Functions

Collision Situation	Center of the Gaussian membership function
In line collision	-3
In line of likely collision	-1.5
Not inline collision	0

2.3.7 Relative Distance Function

This is the Euclidian distance between the vehicles, in an inertial frame of one vehicle. A virtual circle is drawn around the vehicles such that it covers the whole parts of the vehicle. The collision state can be stated as follows.

If the distance between the centers of the virtual circles is less than or equal to the diameter of a virtual circle, then the vehicles are in collision state (Equation 2.5).

$$RelativeDistance = \left((x_1 - x_2)^2 + (y_1 - y_2)^2 \right)^{1/2} \qquad \text{--------------------------(2.5)}$$

Therefore the breaking critical distance (d_{br}) can be present as

$$d_{br} = \begin{cases} \frac{1}{2\alpha}\left(v^2 + (v - v_{rel})^2\right), & \text{for head-on collisions} \\ \frac{1}{2\alpha}\left(v^2 - (v - v_{rel})^2\right), & \text{for rear-end collisions} \\ \frac{1}{2}\frac{v^2}{\alpha}, & \text{otherwise} \end{cases} \qquad \text{--------------------------(2.6)}$$

Where v is the velocity of the vehicle and v_{rel} is the relative velocity of the vehicle with respect to the other. α is, the maximum possible deceleration of the vehicle as shown in equation 2.6 [3].

2.3.8 Master Slave Switching

This function is mainly for hierarchical controlling of the vehicles. This assigns the labels 'Master' and 'Slave' for the vehicle and it is depends upon the current situation of the vehicles. The 'Master vehicle' has more power relative to the 'Slave vehicle'. This was used in the decision making process, inside the inference engine. A typical label assignation criterion is the speed of the vehicle. When in Inter vehicular communication mode, the master-slave status were calculated, transmitted and acknowledged by each vehicle. When the communication scheme is through the computer, it assigns the master-slave states to the vehicles by analyzing the motions of them, according to pre programmed algorithm [32].

Master-Slave switching (MSSwitch) function has three variables. They are master, slave, and driver control. In order to quantify these variables, 'master' is assigned '1' while the 'slave' is assigned with '0'. The value given to 'driver control' is '5'. As discussed before, these values are taken as the initial centers of the Gaussian membership functions of the corresponding fuzzy variables.

2.3.9 Controlling Function

These input membership functions were trained using 650 pairs. Those data pairs were generated employing spread sheet program accordance with the controller algorithm, considering the ranges of the input and output variables . The generated data set enabled to train the ANFIS so that it mimics the behavior of an expert. The trained input functions were taken from the MatLab software. The shapes of those functions were adjusted at the learning according to the training data.

The Takagi-Sugeno type 54 output functions after training, for the braking controller were taken and they are given below [15].

$f_1 = 1.85\,x_1 + 1.566e\text{-}8\,x_2 - 2.79e\text{-}21\,x_3 - 2.1894e\text{-}15\,x_4 - 1.48$

$f_2 = 0.0019x_1 - 0.039x_2 - 5.655e\text{-}23x_3 - 4.72e\text{-}17x_4 - 4.225$

$f_3 = -0.311\,x_1 - 2.892e\text{-}6\,x_2 - 5.577e\text{-}25x_3 - 5.667e\text{-}18\,x_4 + 0.0921$

. . . -----------------(2.7)

. . .

$f_{54} = -1.710e\text{-}15\,x_1 + 4.911e\text{-}18\,x_2 - 4.910e\text{-}14\,x_3 + 5.677e\text{-}15\,x_4 + 5.679e\text{-}15$

The Takagi-Sugeno type 54 output functions after training, for the steering controller was taken and they are given below [32].

$f_1 = -0.322\,x_1 + 0.0036x_2 + 5.798e\text{-}22\,x_3 + 0.00459\,x_4 + 0.233$

$f_2 = -0.0141x_1 + 0.00321x_2 + 1.069e\text{-}22x_3 + 0.00021\,x_4 + 0.621$

$f_3 = -0.151\,x_1 + 0.000659x_2 + 1.996e\text{-}23x_3 + 0.0039x_4 + 0.0652$

. . .

. . . ------------------(2.8)

$f_{54} = 2.69e\text{-}24\,x_1 + 1.755e\text{-}24\,x_2 - 3.911e\text{-}25\,x_3 + 2.691e\text{-}23x_4 - 1.217e\text{-}23$

In the prototype experiments, the above 108 (54X2) functions were implemented in the PC by means of Microsoft Visual Basic 6 IDE. The position, heading, and speed information from the prototypes was transmitted and taken via a serial port to the central PC. This information is required to assess the input membership values [32]. The controlling signals generated were transmitted to the prototypes via the RF link. Figure 2.25 to 2.27 illustrates the input membership functions after training. These illustrations were created in the MatLab environment; with the aid of the ANFIS edit tool, and Simulink.

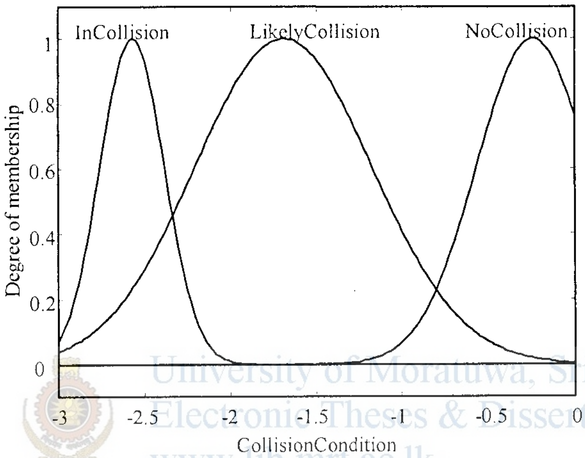


Figure 2.25 - Collision Condition Function after Training

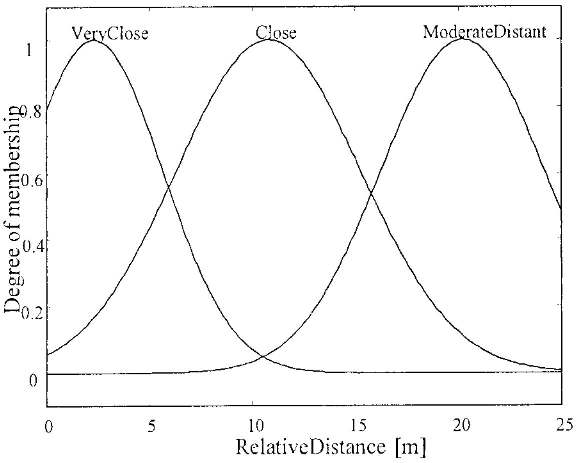


Figure 2.26 - Relative Distance Function after Training

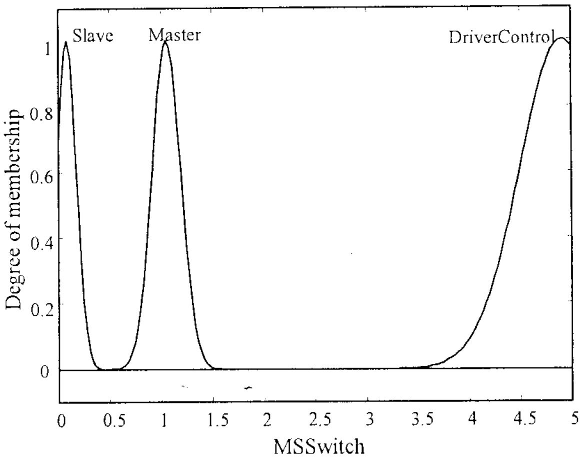
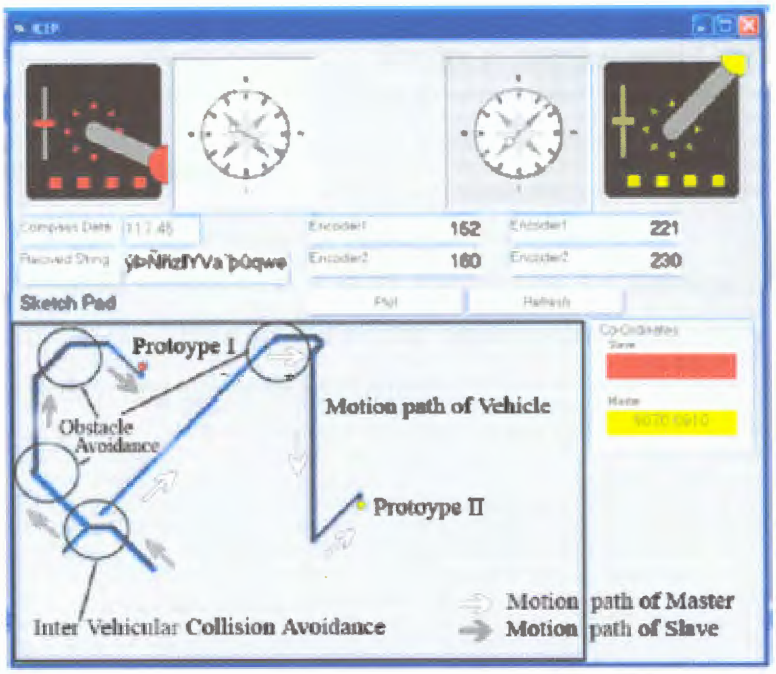


Figure 2.27 - Master-Slave Switching Function after Training

2.4 Results

Trajectories of the prototypes were taken from the graphical user interface of the developed software. It was developed using shape objects in Visual basic 6 and Joystick objects of Global Magic Software's. Predefined co-ordinate system was used to map the prototypes. Two joysticks were used at the manual driver mode. When the vehicles recognize a possible collision scenario, the joystick commands are simply ignored and the automatic collision avoidance controller turns on. It is automatically changed in to manual mode after avoiding collision scenarios.

Fig 28 represents a screenshot that was taken in a typical test case of the study that has been carried out with the two prototypes. It illustrates the paths followed by the prototypes and the encrypted data that has been rcccived.



Compass Data 117.45
Recived String ÿb-ÑñzlYVa`þOqwe
Encoder1 162
Encoder2 160
Encoder1 221
Encoder2 230
Master 5070 0910

Figure 2.28 - Screenshot of the developed GUI

The speed of the prototypes was quite low due to the performance of the dc servos. But it was healthy with the whole system. Because of the serial communication, I^2C bus and OOPic were mainly introduced delays to whole system. But those can be minimizing by using appropriate hardware when this system is going to be implemented for genuine vehicles.

This concept can be extended for multiple vehicles by considering them as pairs .

2.5 Summary

This chapter has presented and experimentally validated an interactive intelligent collision avoidance controller via testing on vehicular prototypes. A master-slave mechanism is engaged to effectively negotiate the cooperative maneuvers by the vehicle on the verge of a collision to optimally avoid the collision. The control strategy was developed based on ANFIS fuzzy and has been thoroughly validated via computer simulations in for all possible collision scenarios. The central communication PC was used to avoid memory constrains in the digital controller (OOPic) at the cost of some delays in the whole system. A better DSP can eliminate the central PC to easily overcome this problem in practical implementation.

3.1 Implementing the controller

A good navigational controller is a main requirement for traveling. So the Takagi-Sugeno type fuzzy logic base controller was implemented successfully for navigation [5]. The boat is traveling as shown in Figure 3.1.

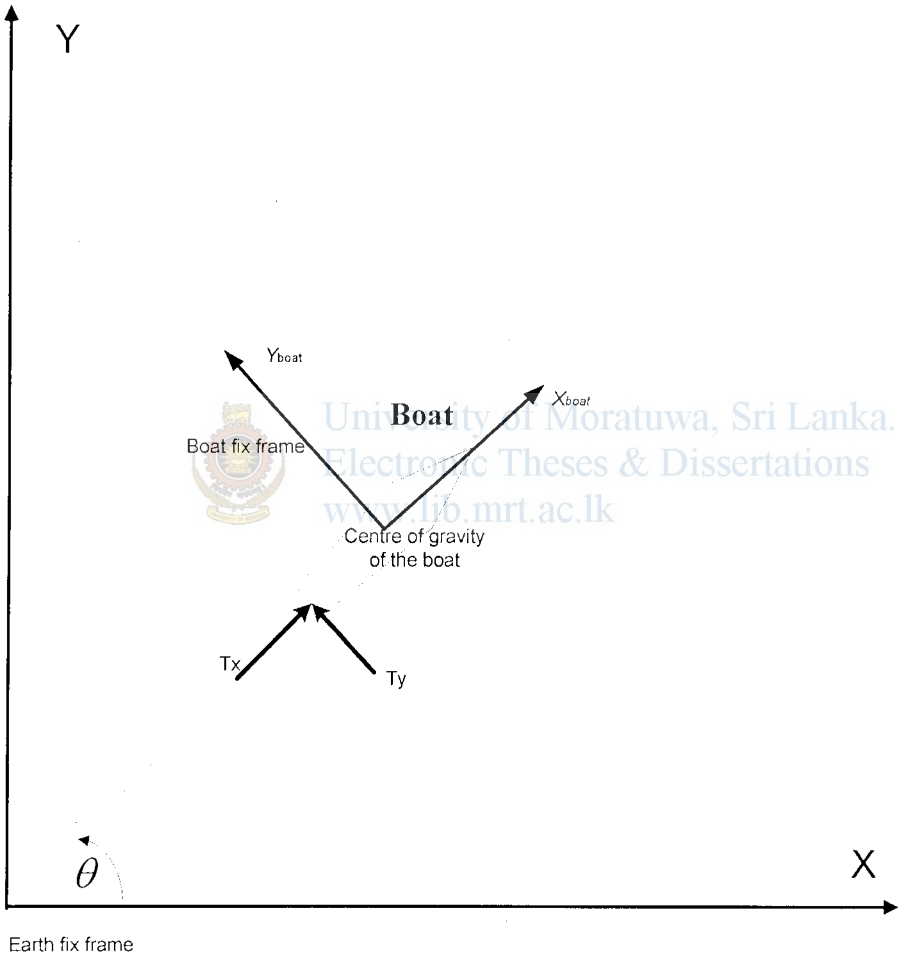


Figure 3.1 - Boat with fuzzy based navigational controller

Tx and *Ty* are the thrust forces exerted by the propellers of the boat. The heading of the boat is given by θ. The thrust force is controlled using a fuzzy controller. The overall control system (navigation) is shown in Figure 3.2. If the required position of the boat is (x_r, y_r) and the actual position is (x,y) then the error e is given by $e=x_r-x$.

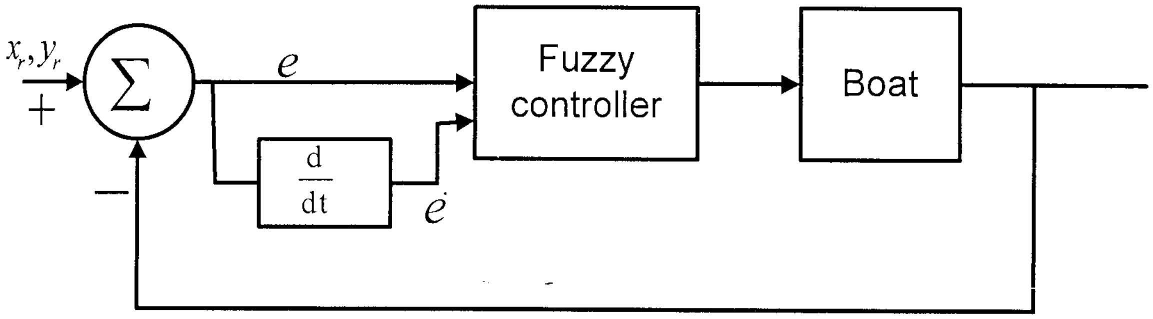


Figure 3.2 - Fuzzy based navigational controller

Five input membership functions were defined to represent the error input named negative large (NL), negative small (NS), zero (Z), positive small (PS), positive large (PL) and same names were used for the change rate of error as well. Input membership functions to fuzzy controller and output membership functions from the fuzzy controller are given in Figure 3.3, 3.4 and 3.5.

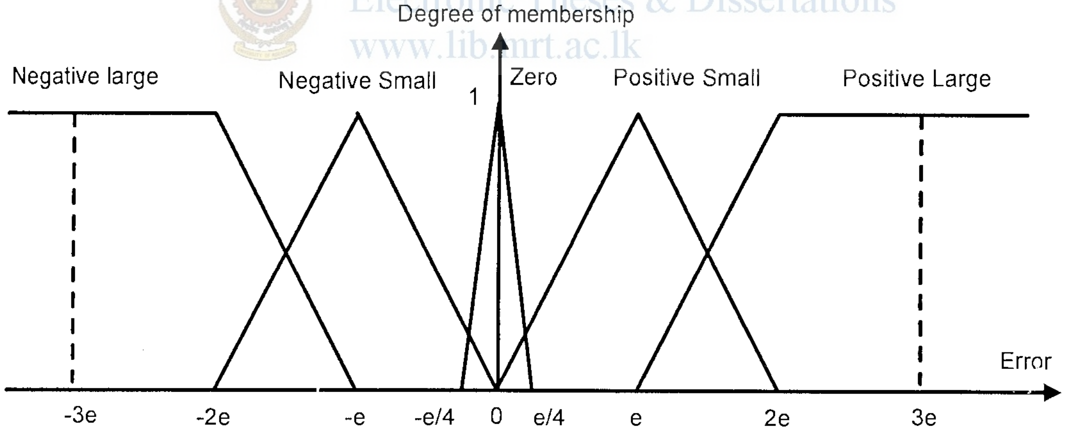


Figure 3.3 - Error input membership function of the fuzzy based navigational controller

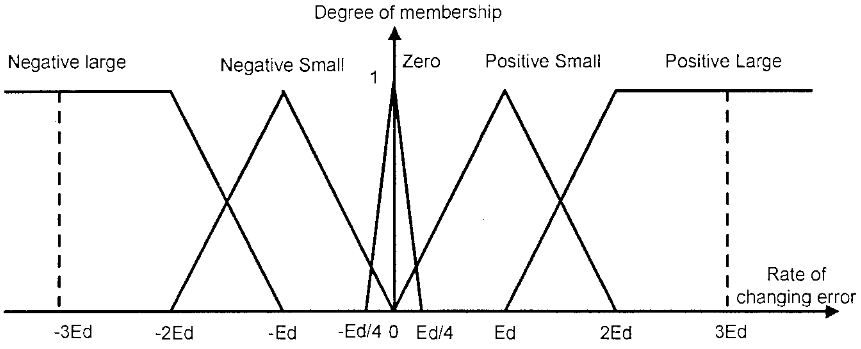


Figure 3.4 - Rate of change error input membership function of the fuzzy based navigational controller

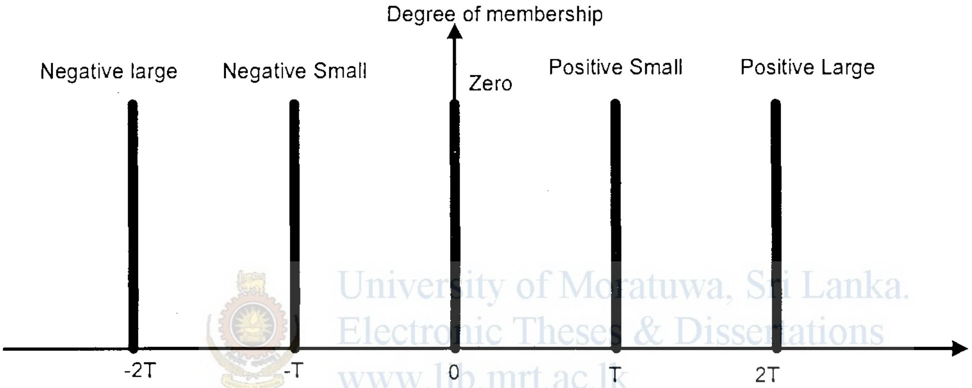


Figure 3.5 - Rate of change error input membership function of the fuzzy based navigational controller

Rule base is one of the important parts in the fuzzy controller. When Table 3.1 shows that the "thrust is positive large" means that the fuzzy controller will give controlling signals to the boat's thrust force controller to increase its thrust force by *2dT* (maximum safe thrust increase) by changing propeller angle and torque. "Thrust is negative large" is represented the maximum thrust force reduction of the boat. The Figure 3.6 presents the output surface of the fuzzy-logic navigational controller.

Table 3.1 - Rule table for fuzzy PD controller

e \ de/dt	NL	NS	Z	PS	PL
NL	NL	NL	NS	NS	NS
NS	NL	NS	NS	NS	NS
Z	NS	NS	Z	PS	PS
PS	PS	PS	PS	PS	PL
PL	PS	PS	PS	PL	PL

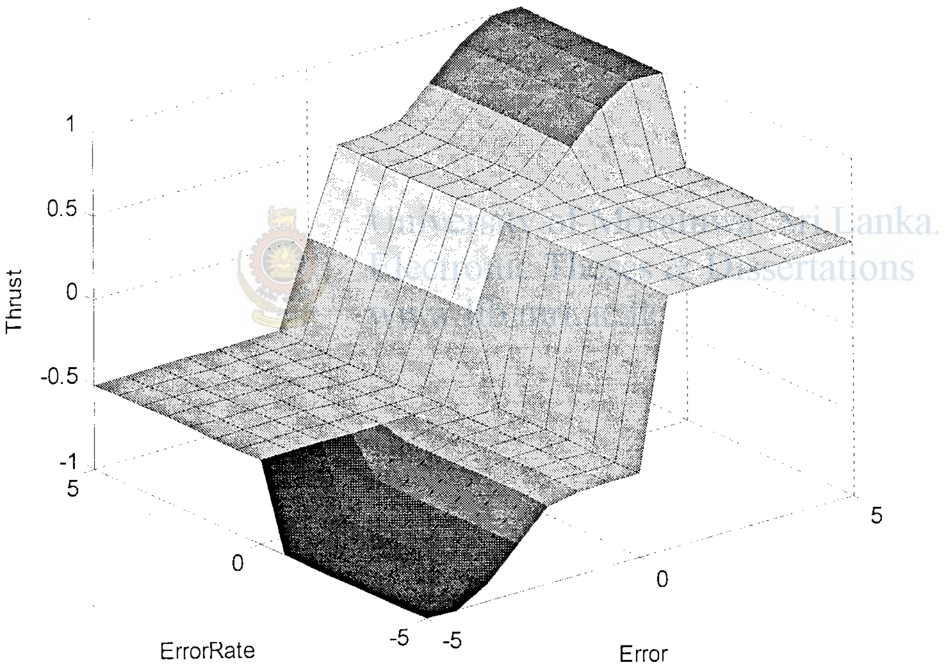


Figure 3.6 - Output surface of the fuzzy based navigational controller

3.2 Mathematical model for USV

The mathematical model of the boat called Delfim (Figure 3.7), developed by Dynamical Systems and Ocean Robotics Laboratory (Portugal) [37] is utilized for the simulation purposes. The following dynamic equations are modeled in MatLab environment [19]. This mathematical model is controlled by using the fuzzy logic navigational controller in the simulations which are performed

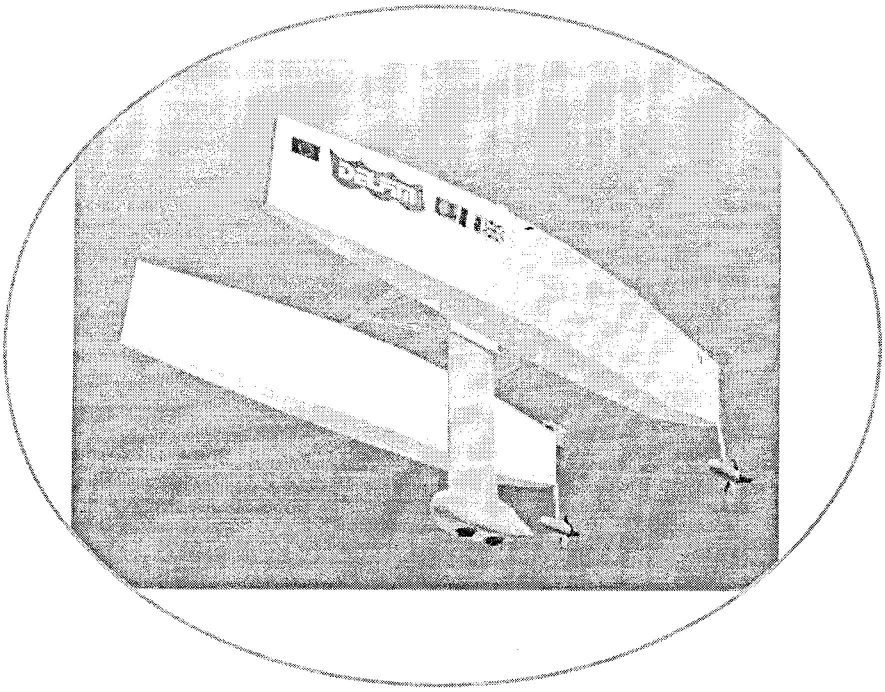


Figure 3.7 - Picture of the actual boat model

The velocities of surge (XB-direction), sway (YB-direction), and yaw (rotation about ZB-direction) are defined as $u = u(t)$, $v = v(t)$, and $r = r(t)$ .Then the dynamic equations for the model are given as below,

$$m\dot{u} = -D_X(u,v,r) + mvr + T_P \cos\alpha_p + T_S \cos\alpha_s \quad \text{---------------(3.1)}$$

$$m\dot{v} = -D_Y(u,v,r) - mur + T_P \sin\alpha_p + T_S \sin\alpha_S \quad \text{---------------(3.2)}$$

$$m\dot{r} = -D_\varepsilon(u,v,r) + T_p \cos\alpha_p \times d_{py} + T_S \cos\alpha_s \times d_{sy} + T_P \sin\alpha_p \times d_x + T_S \sin\alpha_S \times d_x. \quad \text{----(3.3)}$$

Where T is thrust delivered by port side and starboard side propellers, respectively the α_P and α_S are inclinations to the $X_{B\text{-}axis}$, d is diameter of the propeller.

Basic steps of the MatLab program are given below,

a) Define all the variables and set the dimensions of the boat
b) Calculate mass and inertia matrix of the boat
c) Calculate frictional, form and additional resistance forces

e) Calculate the damping and total moments

f) Set all the variables to its initial values, including time *t*, *t = 0*

g) Define boundary conditions for all variables of input and output

h) Initialize the reference values, such as reference trajectory of the ship in 2-D plane

i) While t < t $_{stop}$

   Calculate the *position error*

   Feed the inputs to the controller (within equal intervals)

   Process the rules according to the inputs, by means of MatLab FIS.

   Get the defuzzified output

   Solve dynamic equations to find new position

   Plot the results

3.3 Results from the navigational controller

The performance of the fuzzy based navigational controller is compared with the performance of a PD controller to prove the brilliance of the fuzzy logic based controllers. The results are presented in Figure 3.8 to 3.12 and MatLab progam is given in Appendix B. The desired path is presented in red while blue is utilized for the actual path.

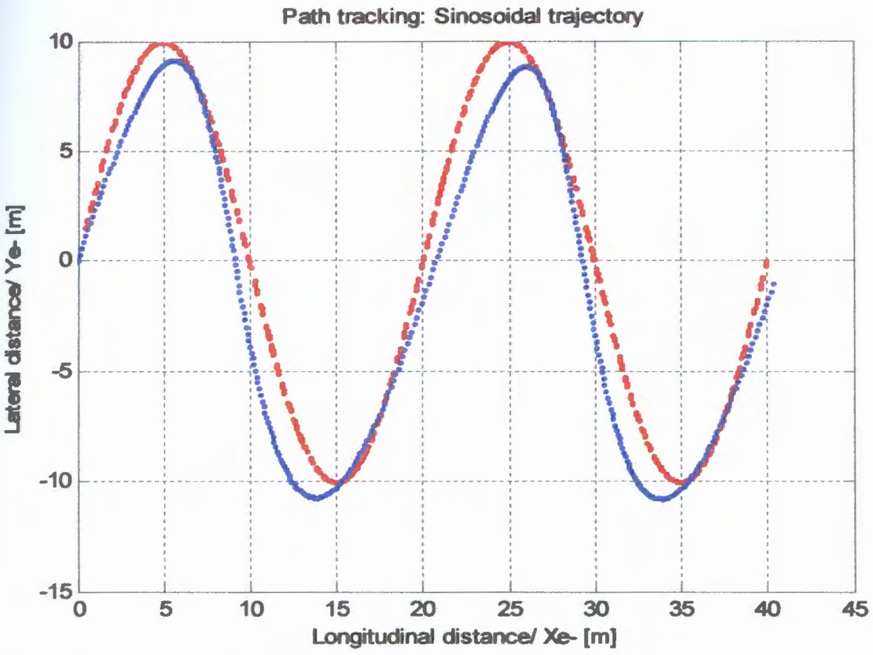


Figure 3.8 - Path tracking of a Sinusoidal trajectory with PD controller

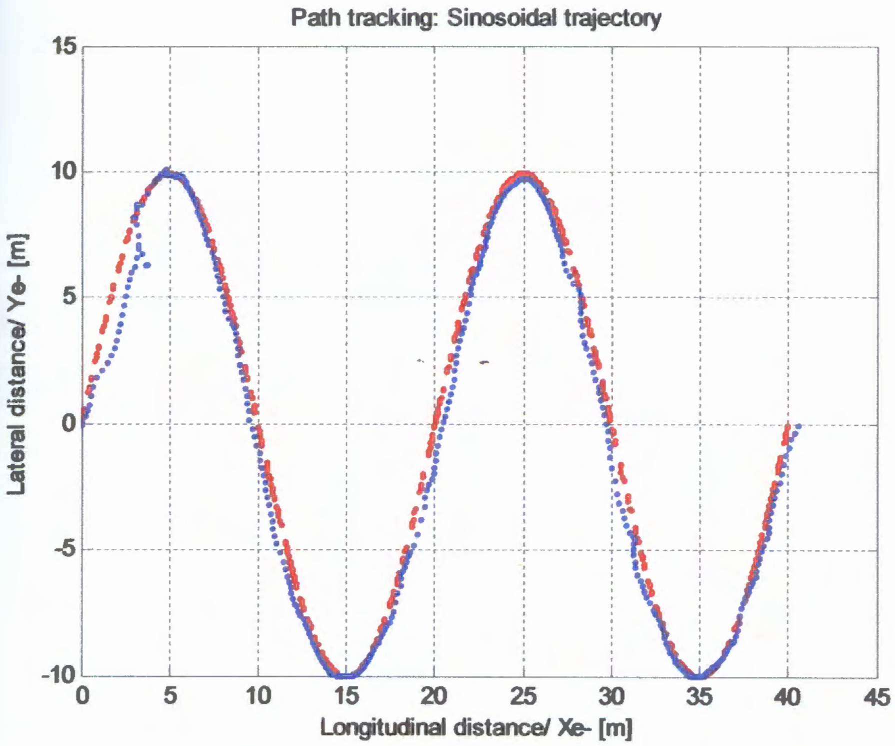


Figure 3.9 - Path tracking of a Sinusoidal trajectory with Fuzzy controller

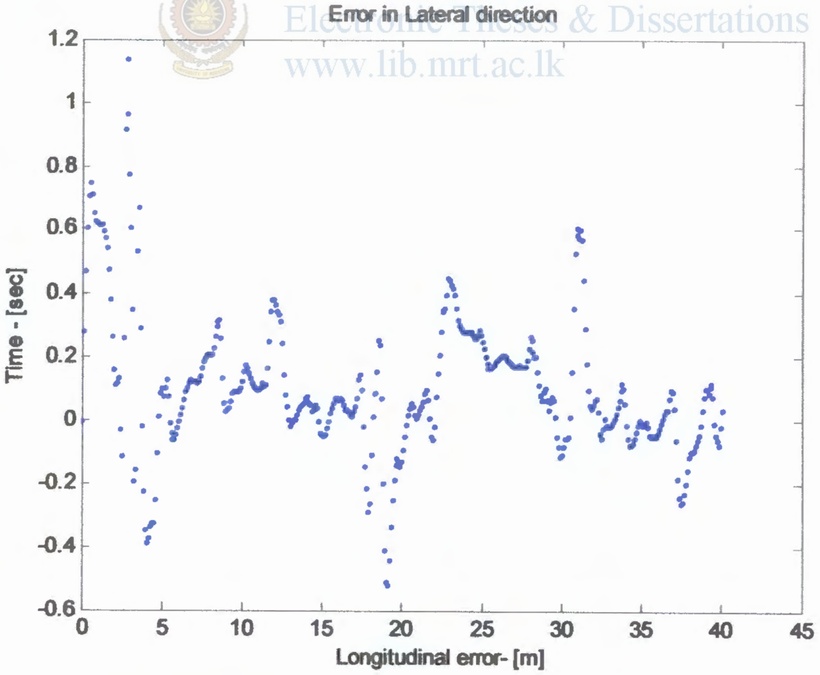


Figure 3.10 - Error in Lateral direction of a Sinusoidal trajectory with Fuzzy Controller

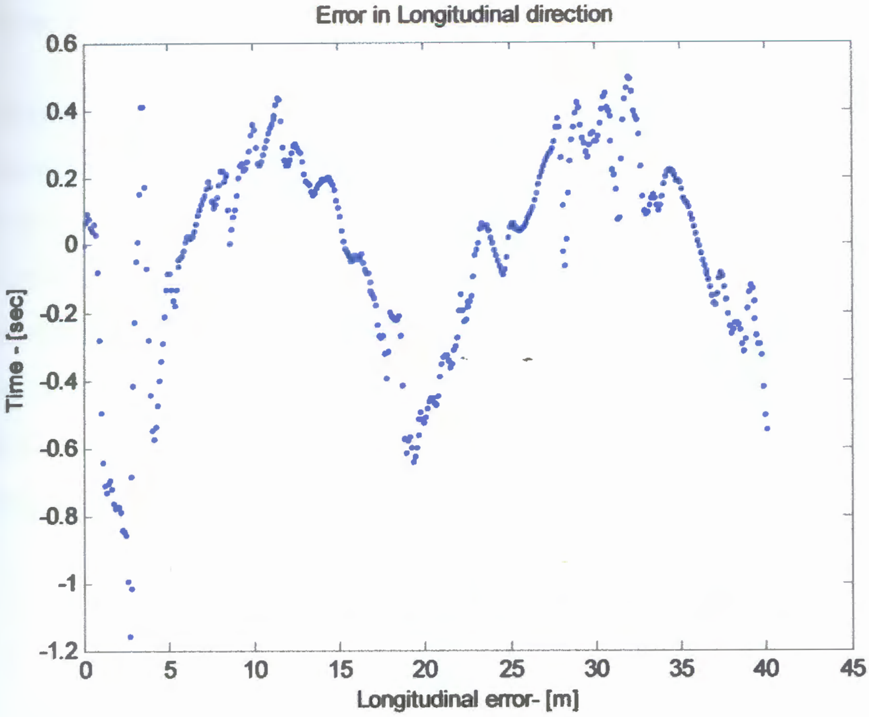


Figure 3.11 - Error in Longitudinal direction of a Sinusoidal trajectory with Fuzzy controller

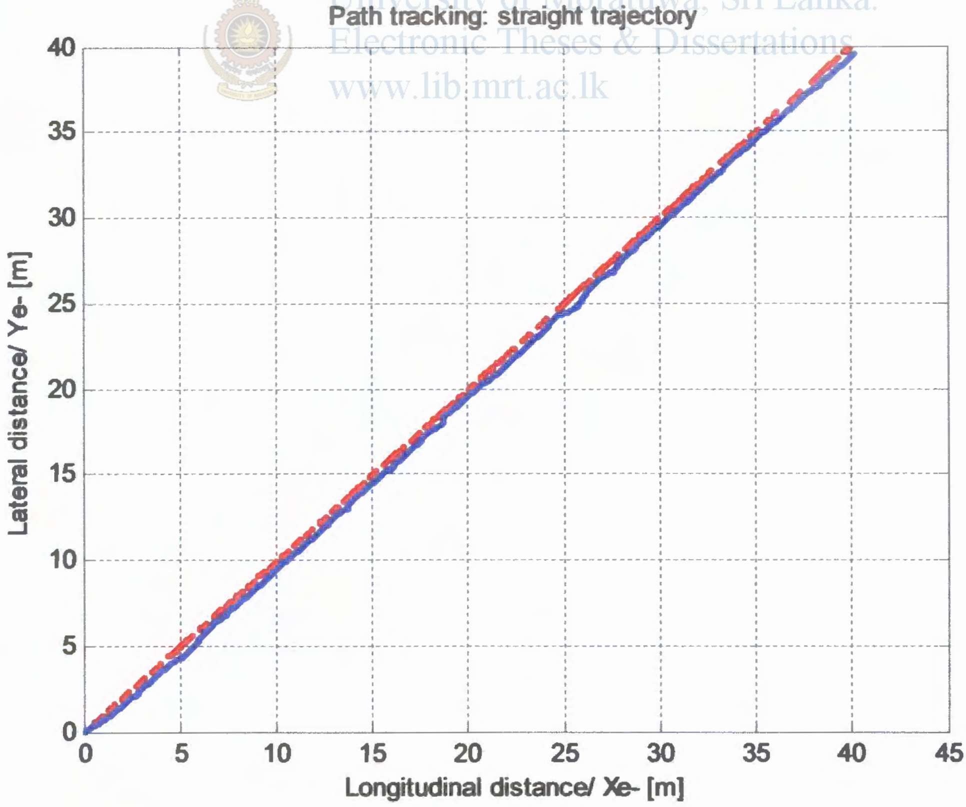


Figure 3.12 - Path tracking of a Straight trajectory with Fuzzy controller

3.4 Summary

This chapter mainly described about designing and simulating a fuzzy logic-based navigational controller for unmanned surface vehicles. An already developed dynamic model of a boat is utilized for the simulations though out the report. The controller considered in this study is a fuzzy based system. MatLab framework with Fuzzy-logic toolbox was used to design and implementing the whole system. MatLab programs were employed for the navigational controller simulations. The basic design procedure and the simulation procedure were included in detail. The results including the desired and actual paths are plotted at the end.

4.1 Utilizing Ground Vehicle Technologies for Surface Vehicles

4.1.1 Design of OA controller

Obstacle avoidance controller is essential for autonomous, semi-autonomous navigation or as a driver assistance system to encounter the collision situations for safe navigation. The navigation controller and the obstacle avoidance controller are synthesized independently but operate combined since both controllers are essential to ensure safe navigation.

This OA controller is also implemented in Matlab framework, using fuzzy logic toolbox and Fuzzy Inference systems (FIS) editor GUI. Trapezoidal membership functions are selected as the input membership functions. Sugeno type inference system is used for the controller synthesis [7].

4.1.1.1 Input Functions

Two auxiliary input functions are developed to be used as the inputs to the fuzzy controller. They are

a) Collision direction function and

b) Relative distance function between the obstacle (stationary or dynamic) and the vehicle

Collision direction function is used to take the obstacle direction with respective to the boat heading. Heading angle of the boat and obstacle angle with respective to the earth are used to find this collision direction in the MatLab simulation. Collision direction angle is measured from the Y axis of the boat frame. Figure 4.1 and 4.2 show the measurement of collision direction angle. Calculation was corrected for the whole Cartesian plan. The collision direction angle is defined as

$\beta = 90 + \theta - \alpha$ ------------------------ (4.1)

Where θ is heading angle of the boat and α is obstacle angle.

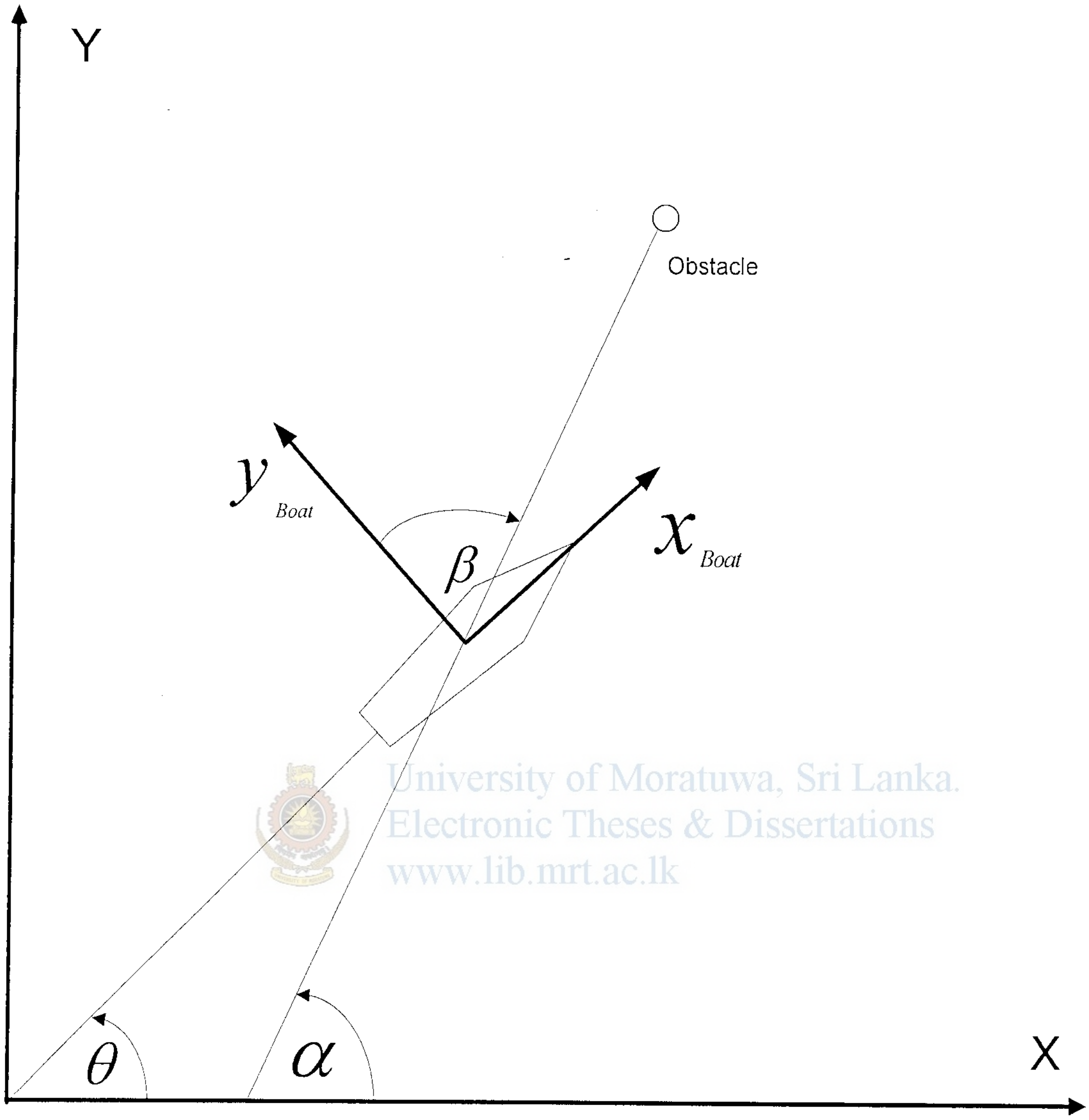


Figure 4.1 - Calculating collision direction

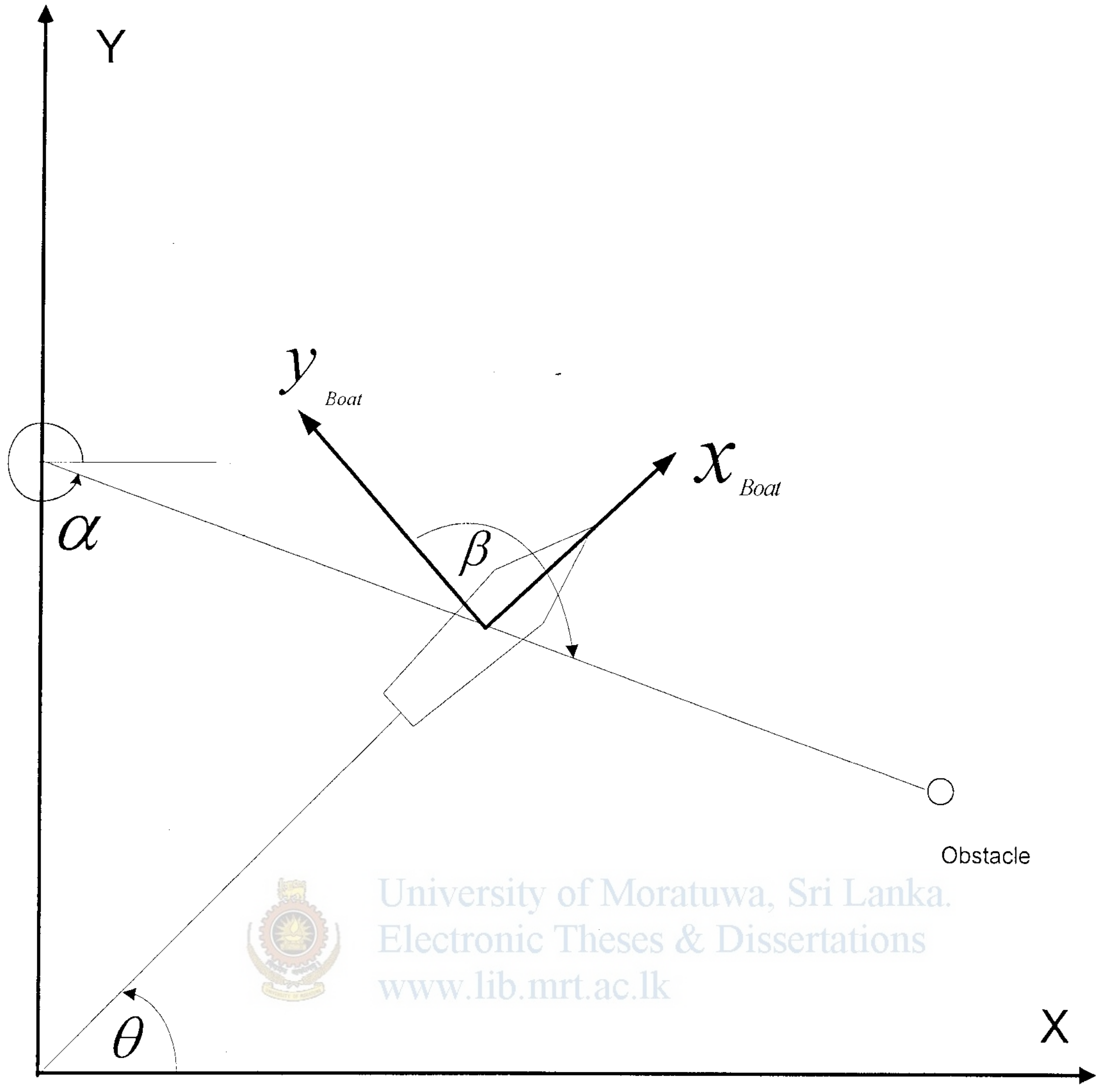


Figure 4.2 - Calculating collision direction

β changes near obstacle alone a desired path is presented in Figure 4.4. That straight path of the boat is given below (Figure 4.3).

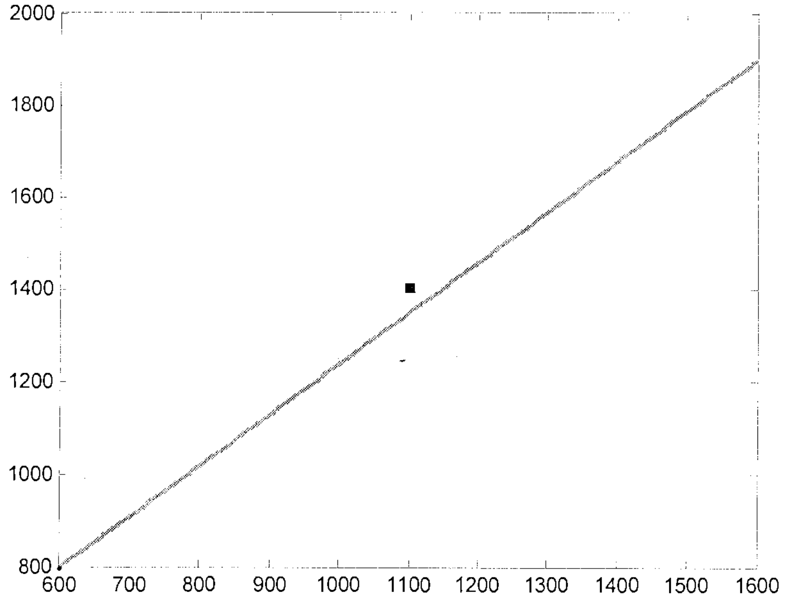


Figure 4.3 - Boats path near obstacle without Obstacle avoidance controller

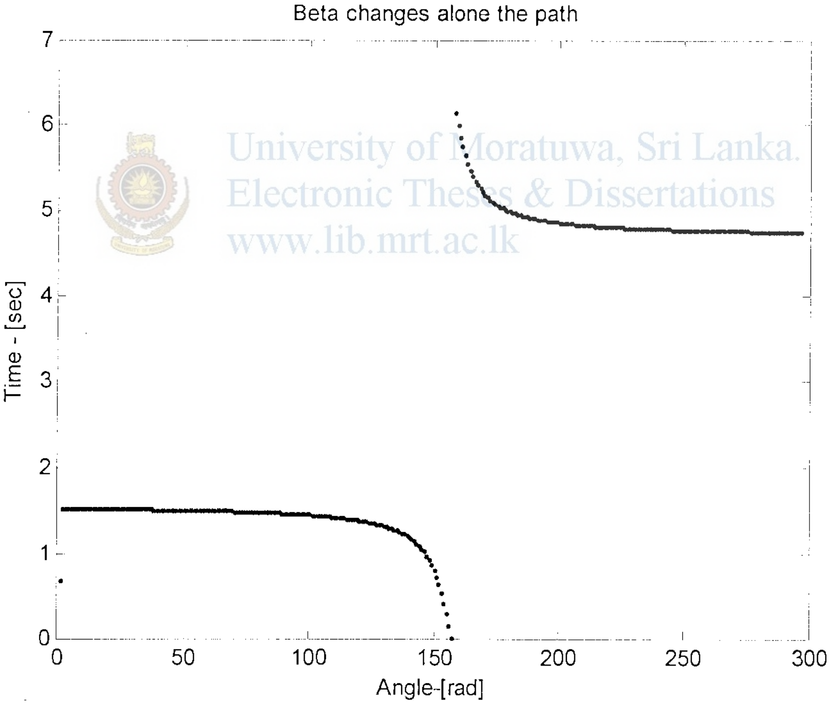


Figure 4.4 - β changes near obstacle

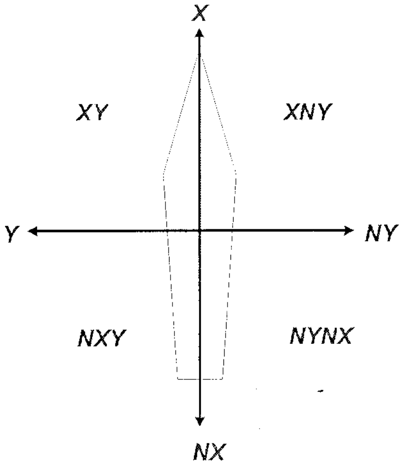


Figure 4.5 - Names of collision direction input membership function

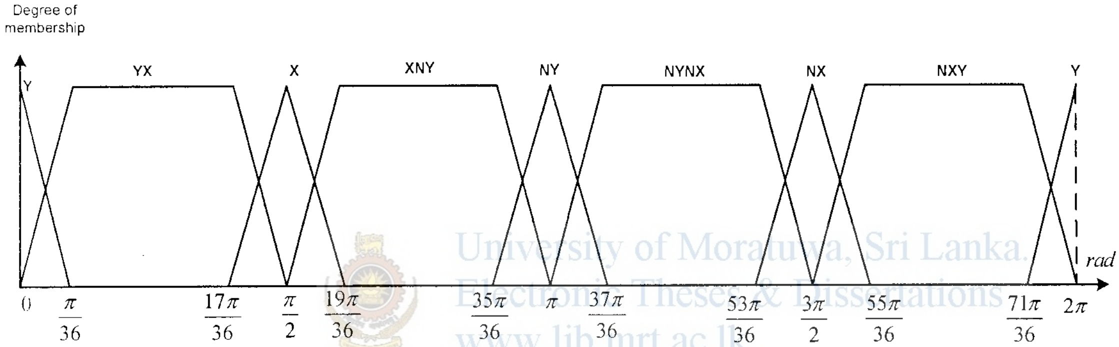


Figure 4.6 - Collision direction input membership function to the obstacle avoidance controller

Eight different membership functions are defined for the collision direction input function. Triangular functions which are having 10^{0} with were used to input collision directions on main axes of the boat fixed frame. Centres of those functions are 0, $\Pi/2$, Π, $3\Pi/2$ and 2Π respectively. Triangular and trapezoidal functions are chosen and these functions can be implements with out many complications. The approach of naming the membership function is given in Figure 4.6 while Figure 4.5 presents the membership functions of the Collision direction function. The radar maps and stereo vision can be used to find this collision direction in real world situations [14].

4.1.1.1.2 Relative Distance Function

Relative distance towards obstacle is calculated by means of coordinates in the coordinate frame system [33]. This is the Euclidian distance between the center of gravity of the boat and the obstacle. Two triangular membership functions are defined as "Low" and "High" (Figure 4.7).

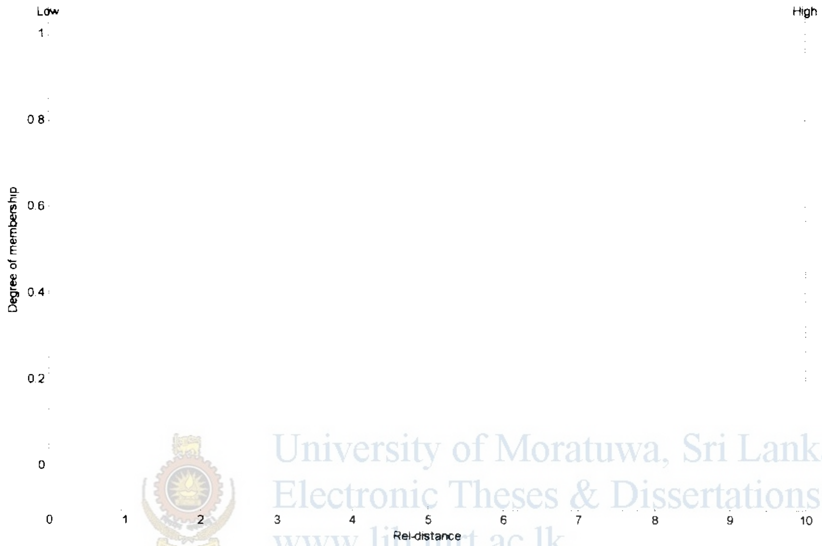


Figure 4.7 - Relative distance input membership function to the obstacle avoidance controller

The x-scale of these functions may subject to changes according to the dimensions and speed of the considered boat platform, but the design and simulation algorithm remain same. Inputs for the relative distance function are defined in between 0 to 100m for the simulation. It means that "Low" fuzzy set is define from 0 to 50m while "High" is define from 50m to 100m.

4.1.1.2 Rule base of the controller

The rules for the controller are given below. Xdot and Ydot are the velocity components of the main direction. Here "DDD" presents the maximum velocity of the boat while "DD" and "D" presents the 2/3 and 1/3 of the maximum velocity correspondingly. Negative velocities of the above velocities are appeared as "NDDD", "NDD" and "ND".

1. If (Col-direc is Y) and (Rel-distance is High) then (Xdot is Zero) and (Ydot is ND)
2. If (Col-direc is Y) and (Rel-distance is High) then (Xdot is Zero) and (Ydot is ND)
3. If (Col-direc is Y) and (Rel-distance is Low) then (Xdot is D) and (Ydot is NDD)
4. If (Col-direc is Y) and (Rel-distance is Low) then (Xdot is D) and (Ydot is NDD)
5. If (Col-direc is NY) and (Rel-distance is High) then (Xdot is Zero) and (Ydot is D)
6. If (Col-direc is NY) and (Rel-distance is Low) then (Xdot is D) and (Ydot is DD)
7. If (Col-direc is NX) and (Rel-distance is High) then (Xdot is D) and (Ydot is Zero)
8. If (Col-direc is NX) and (Rel-distance is Low) then (Xdot is DD) and (Ydot is D)
9. If (Col-direc is X) and (Rel-distance is High) then (Xdot is ND) and (Ydot is Zero)
10. If (Col-direc is X) and (Rel-distance is Low) then (Xdot is NDD) and (Ydot is D)
11. If(Col-direc is NXY)and(Rel-distance is Low)then(Xdot is D) and (Ydot is NDD)
12. If(Col-direc is NXY)and(Rel-distance is High)then(Xdot is Zero)and(Ydot is ND)
13. If (Col-direc is NXNY)and(Rel-distance is Low)then(Xdot is DD)and(Ydot is D)
14. If (Col-direc is NXNY)and(Rel-distance is High)then(Xdot is Zero)and(Ydot is D)
15. If (Col-direc is YX)and(Rel-distance is High)then(Xdot is Zero)and(Ydot is ND)
16. If (Col-direc is YX)and(Rel-distance is Low)then(Xdot is ND)and(Ydot is NDD)
17. If (Col-direc is XNY)and(Rel-distance is High)then(Xdot is Zero)and(Ydot is D)
18. If (Col-direc is XNY)and(Rel-distance is Low)then(Xdot is ND)and(Ydot is DD)

4.1.1.3 Output Functions

Only two output membership functions are used to reduce the complexity of the controller. They are

1) Velocity component along x axis ($\dot{x}$)

2) Velocity component along y axis ($\dot{y}$)

Output surface of the $\dot{x}$ is given in Figure 4.8 and output surface of the $\dot{y}$ is given in Figure 4.9 .Those velocities are given with respect to the boat frame and they are converted to the world frame before calculating the desired boat positions. The conversion matrix is given below :

$$J = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \qquad (5)$$

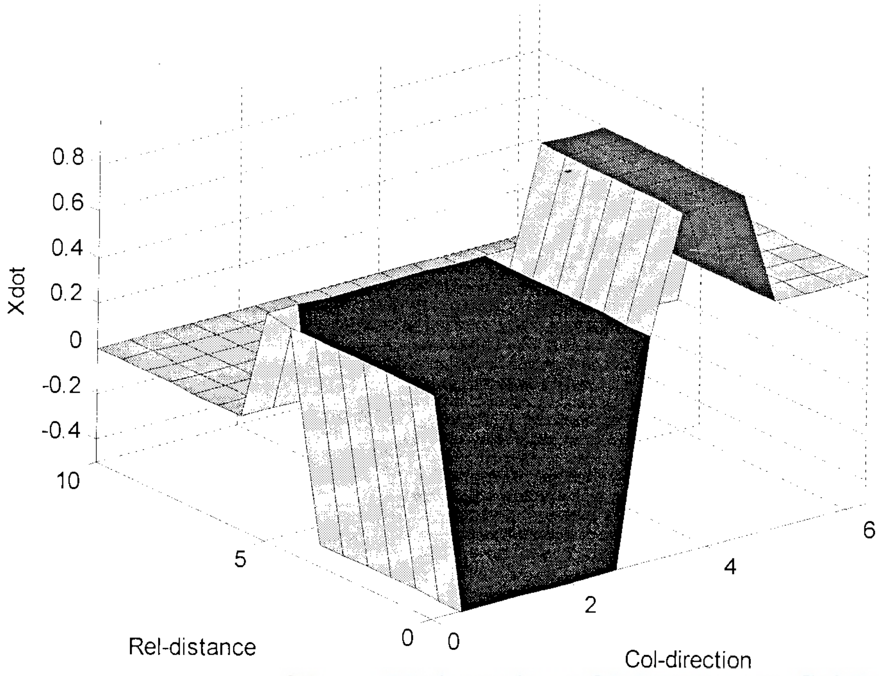


Figure 4.8 - X direction velocity output surface of the obstacle avoidance controller

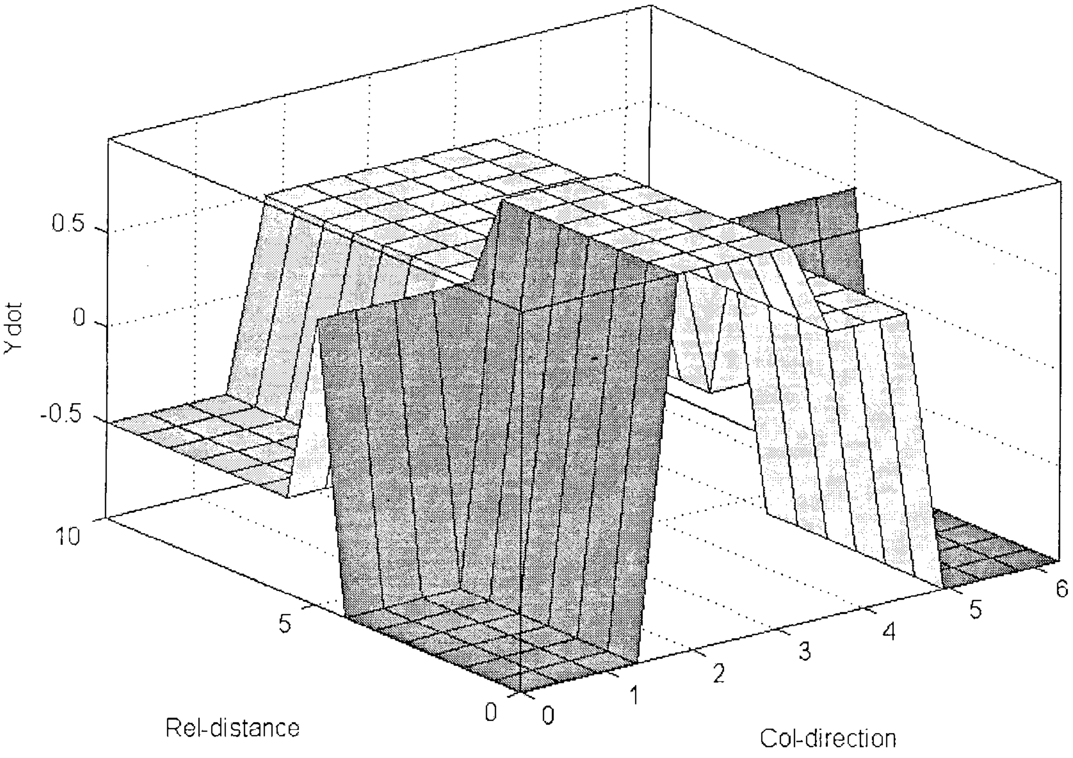
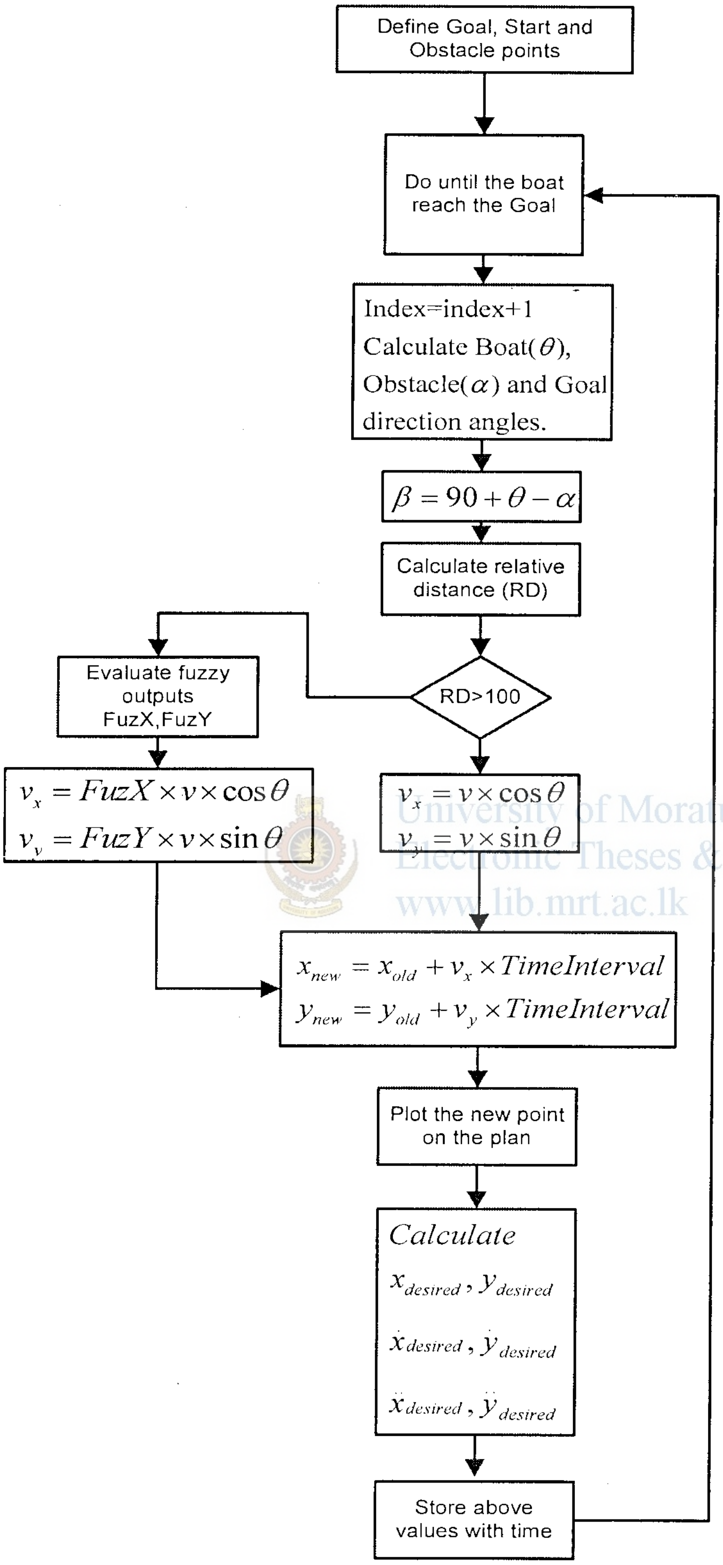


Figure 4.9 - Y direction velocity output surface of the obstacle avoidance controller

The algorithm for path planning with obstacle avoidance is given in figure 4.10. This algorithm was developed as a module in MatLab. That module can be reused for future works. Few results from the simulations are presented at the end of this chapter.

4.1.2 Algorithms for simulation of the controller



Define Goal, Start and Obstacle points
Do until the boat reach the Goal
Index=index+1
Calculate Boat(θ),
Obstacle(α) and Goal
direction angles.
$\beta = 90 + \theta - \alpha$
Calculate relative distance (RD)
RD>100
Evaluate fuzzy outputs FuzX,FuzY
$v_x = FuzX \times v \times \cos\theta$
$v_y = FuzY \times v \times \sin\theta$
$v_x = v \times \cos\theta$
$v_y = v \times \sin\theta$
$x_{new} = x_{old} + v_x \times TimeInterval$
$y_{new} = y_{old} + v_y \times TimeInterval$
Plot the new point on the plan
Calculate
$x_{desired}, y_{desired}$
$\dot{x}_{desired}, \dot{y}_{desired}$
$\ddot{x}_{desired}, \ddot{y}_{desired}$
Store above values with time

Figure 4.10 - Algorithm of the obstacle avoidance controller

Block diagram of the simulation program is shown in Figure 4.11. The algorithm of the path planning module is given in Figure 4.10. The main module is defined the total time span, initial configuration of the boat, goal point and the obstacle points. Then those data is sent to the Path Planning Module. The time span, desired path and boat's configurations are sent to the Boat's Dynamic Simulation Module. Then the actual path and configurations of the boat is taken by the Main module to plot the actual paths of the boat. Results from this module arẽ presented from Figure 4.12 to 4.16. Relavent MatLab program is given with Appendix C.

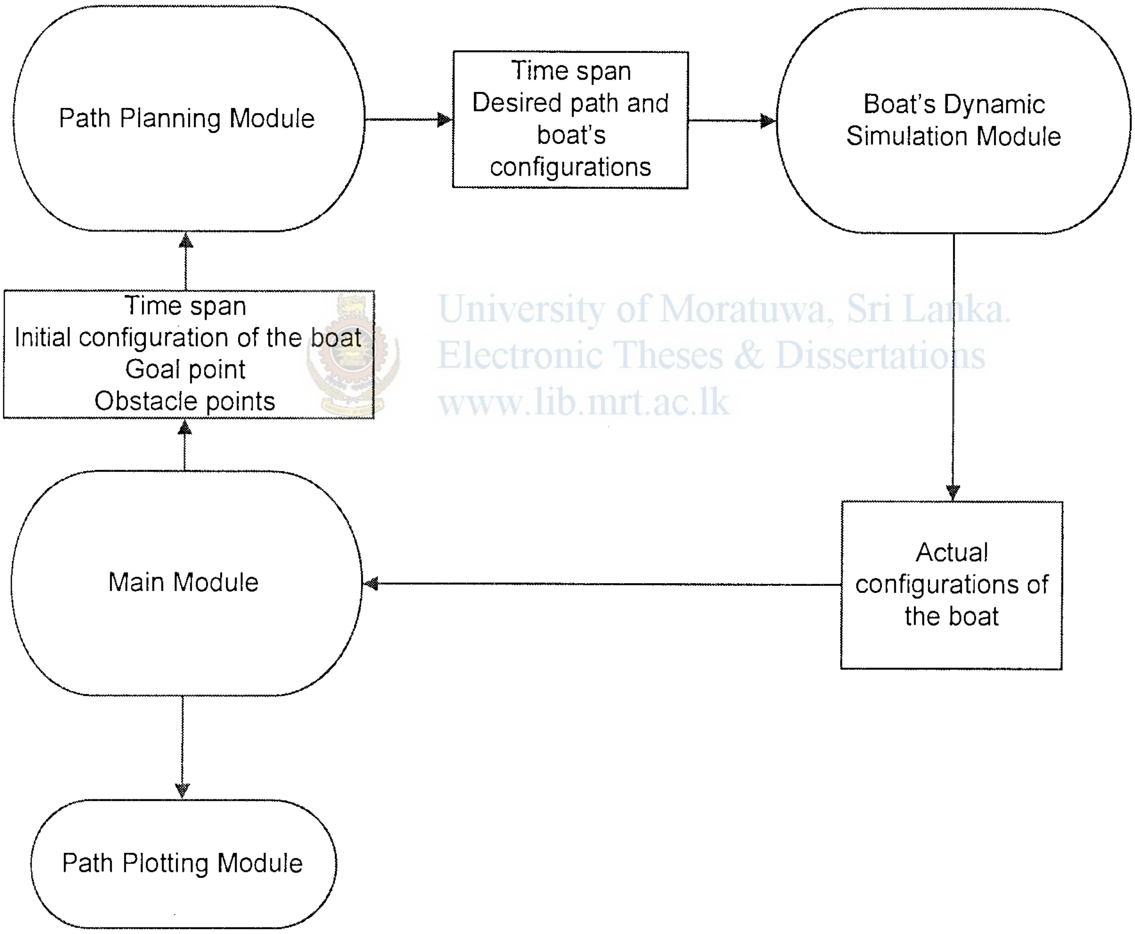


Figure 4.11 - Simulation setup of the obstacle avoidance controller

4.1.3 Simulation Results from the Controller

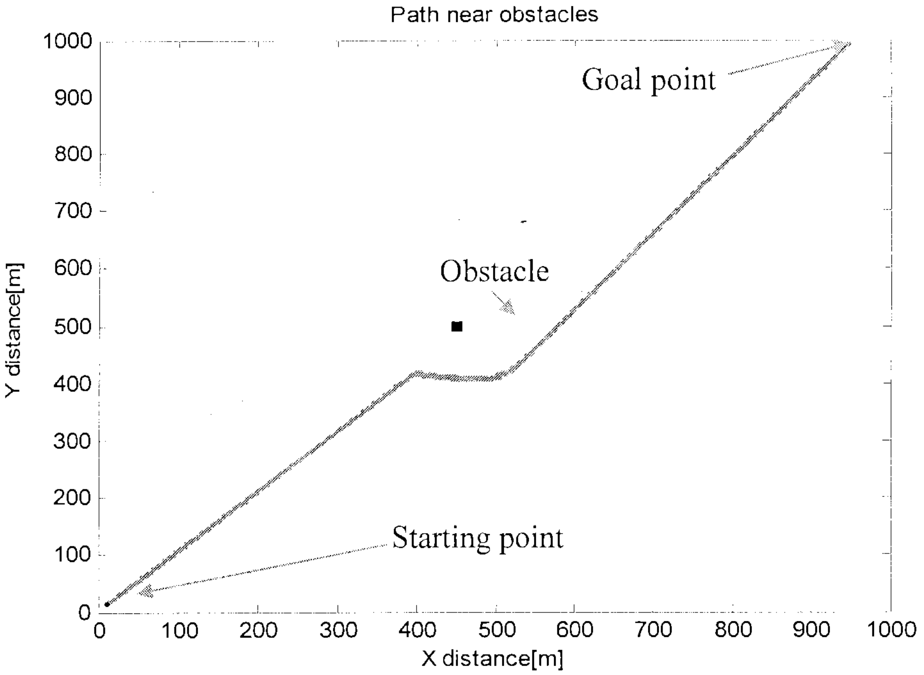


Figure 4.12 - Path starting near root of the coordinate system

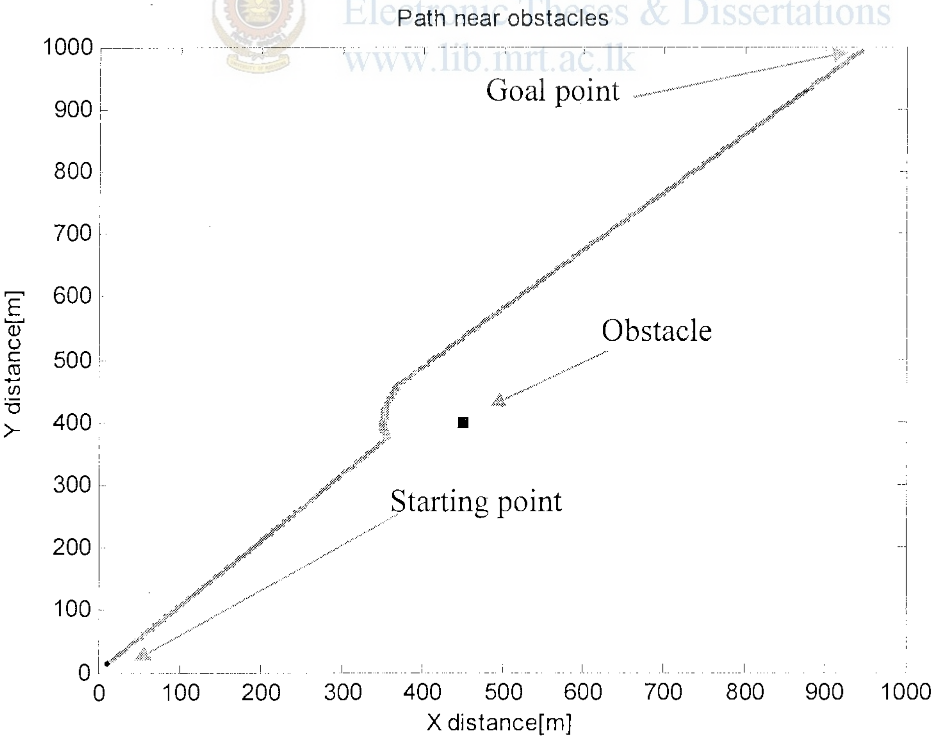


Figure 4.13 - Path starting near root of the coordinate system

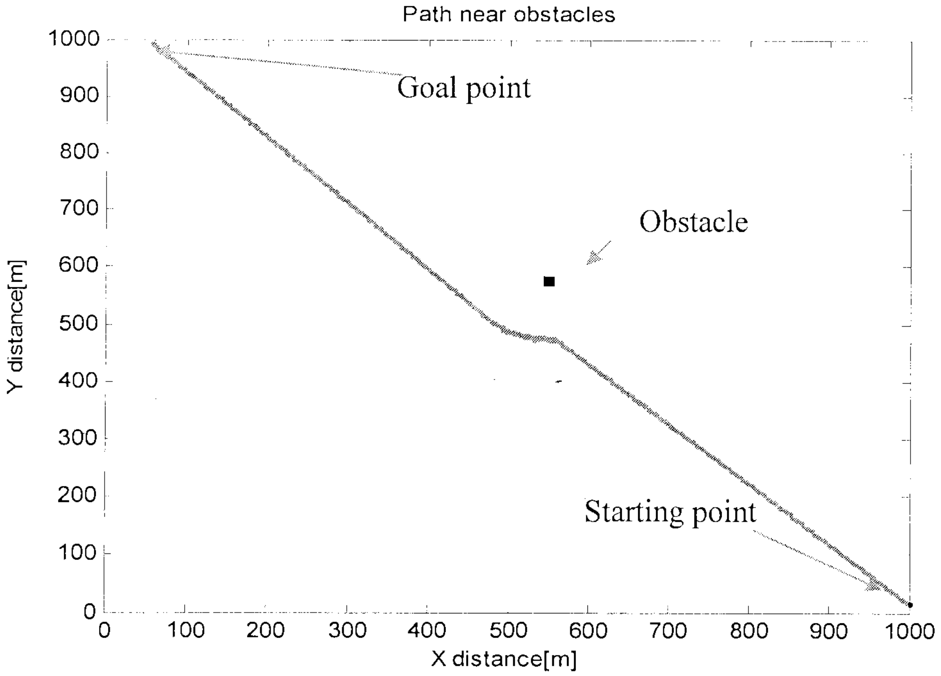


Figure 4.14 - Path starting near the left corner of the coordinate system

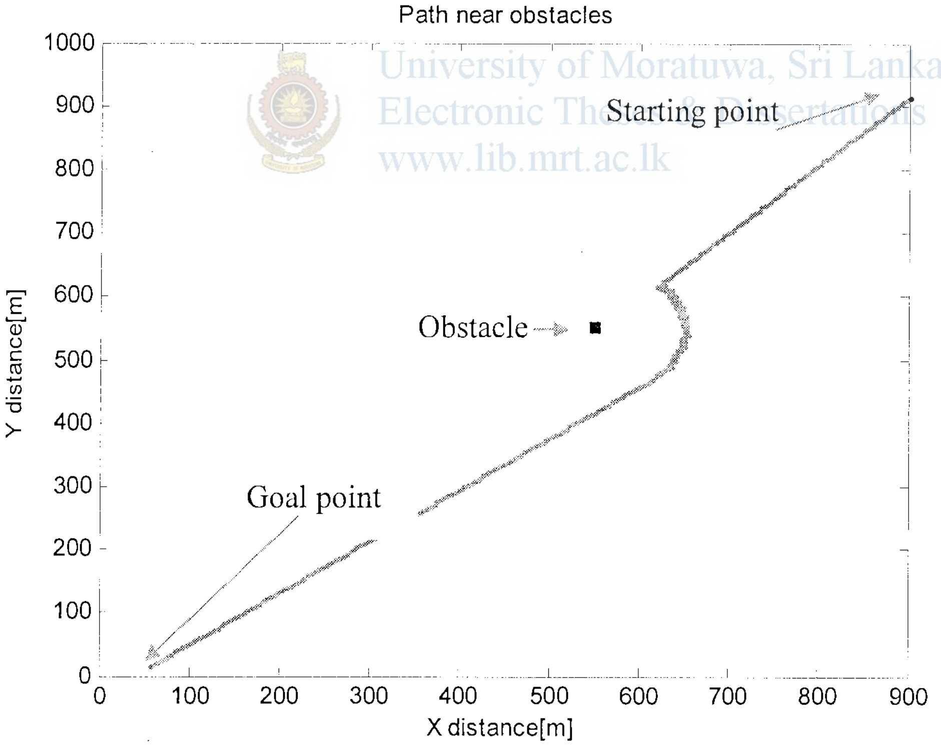


Figure 4.15 - Path finishing near root

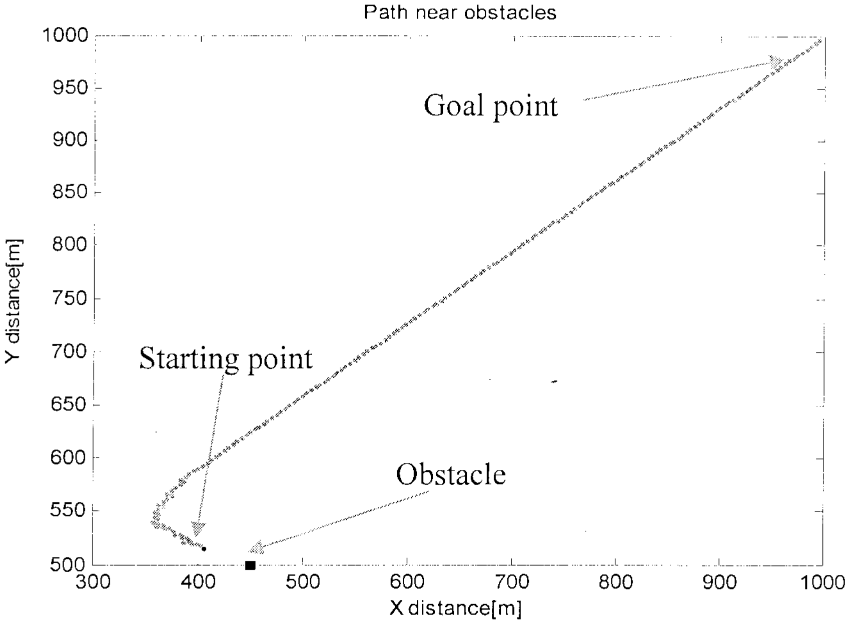


Figure 4.16 - Path starting near obstacle

4.1.4 Summary

This chapter presented obstacle avoidance algorithms for unmanned surface vehicles which are already developed for unmanned ground vehicles. A previously developed mathematical model and a fuzzy-based navigational controller is utilized with obstacle avoidance algorithms, for simulations. This was also implemented in Matlab framework, using fuzzy logic toolbox and fuzzy inference systems (FIS) editor GUI. Trapezoidal and membership functions ware selected as the input membership functions. Sugeno type inference system is used for the controller synthesis. All the programs are developed in the MatLab environment.

Obstacle avoidance control is essential for autonomous, semi-autonomous operation of vehicles or as a driver assistance system to encounter the collision situations for safe navigation. The navigation controller and the obstacle avoidance controller are synthesized independently but operate together, since both controllers are essential to ensure the task of safe navigation.

4.2 Novel Algorithm for OA

4.2.1 Methodology of the Novel Algorithm

Although the Morphin algorithm has been used for many practical applications, none of the detailed algorithms has is not published yet. So this section describes the development of novel algorithm for USVs, based on Morphin algorithm.

In this method stereo cameras and Surveillance radars can be use to capture obstacles in the field [21]. The captured data is transformed in to a grid. Then that grid is utilized as obstacle matrix to avoid that obstacle. Figure 4.17 is presents an obstacle of that grid and Figure 4.18 shows the corresponding obstacle matrix with values. Ones and zeroes are used to represents obstacles and obstacles free areas. This morphin algorithm is basically developed for unmanned ground vehicles but it is not possible to use ones and zeroes for them. Because the height and appearance of the obstacles on ground are needed to store obviously. But the case is somewhat different with the surface vehicles because they cannot overcome obstacles as ground vehicles. So, it is sufficient to use ones and zeroes in the obstacle matrix.

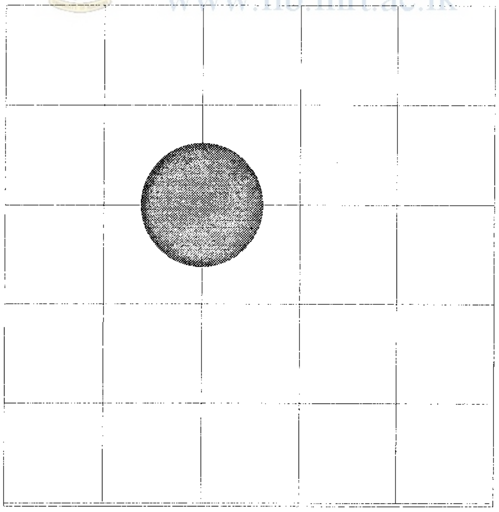


Figure 4.17 - Obstacle in the grid

0	0	0	0	0
0	1	1	0	0
0	1	1	0	0
0	0	0	0	0
0	0	0	0	0

Figure 4.18 - Obstacle matrix with obstacle

Figure 4.19 shows number of arcs projected in front of the boat over the local world-model obstacle map. The number of arcs considered is a function of the map size and grid spacing, with the arcs spaced such that one arc passes through each of the outer cells. This approach guarantees that each cell in the grid is covered by at least one arc so that all navigable paths are considered. Mathematical functions of the circles and lines are utilized to calculate the coordinates of the paths. They are put in to a path matrix with a path index. Straight line functions are employed for the middle path as their radius is tent to infinity.

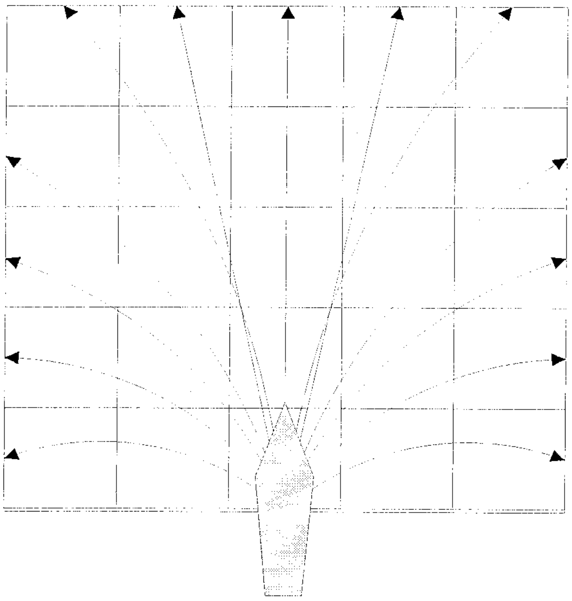


Figure 4.19 - Possible paths of the USV

The projected paths and the obstacle matrix is shown in Figure 4.20. Here the paths on the obstacles are assigned small weight while other paths are assigned using high weights. A curtain weight values are further added for the paths heading towards the goal as well. So, the best path is chosen depending on the weights assigned at that time. Then the boat moves a certain distance along that path. Again, the new obstacle matrix is processed and obstacle matrix is added to the system for other iterations.

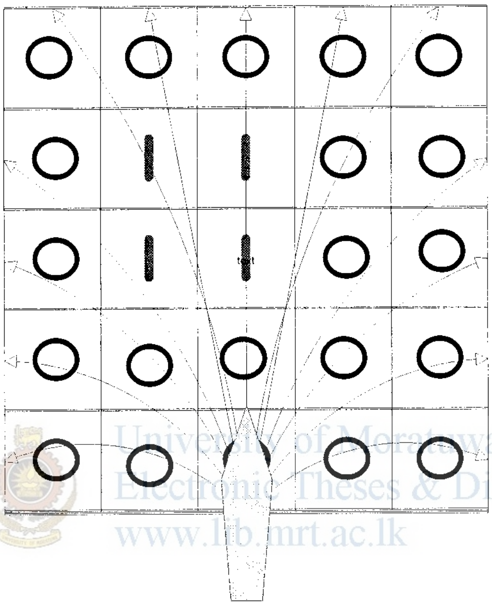


Figure 4.20 - Possible paths of the USV on the Obstacle matrix

Coordinates of the obstacles in the navigation area is necessary to be inserted in to obstacle matrix. The way of extracting obstacle coordinates in to the obstacle matrix is presented in the Figure 4.21.

The transformation is given by Equation 4.1 where obstacles on obstacle matrix are presented as OOM and obstacle on world frame is given as Oeff.

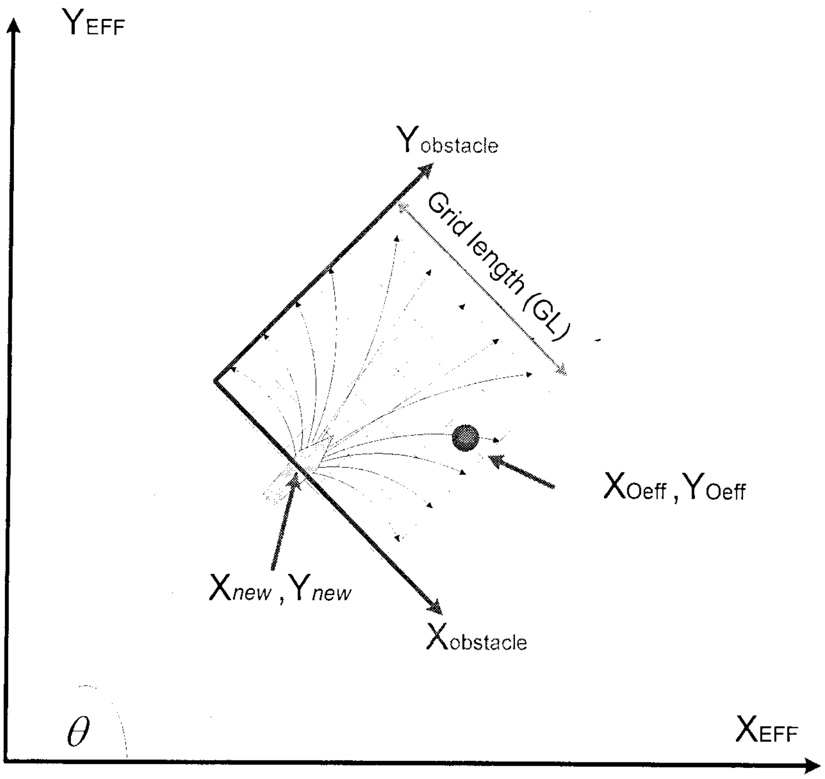


Figure 4.21 - Obstacle coordinate conversion

$$\begin{bmatrix} y_{OOM} \\ x_{OOM} \end{bmatrix} = \begin{bmatrix} sin\theta & -\cos\theta \\ \cos\theta & \sin\theta \end{bmatrix}^{-1} \begin{bmatrix} y_{Oeff} - y_{new} \\ x_{Oeff} - x_{new} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{GL}{2} \end{bmatrix} \quad \text{----------} \quad (4.1)$$

Where θ represents the heading of the boat.

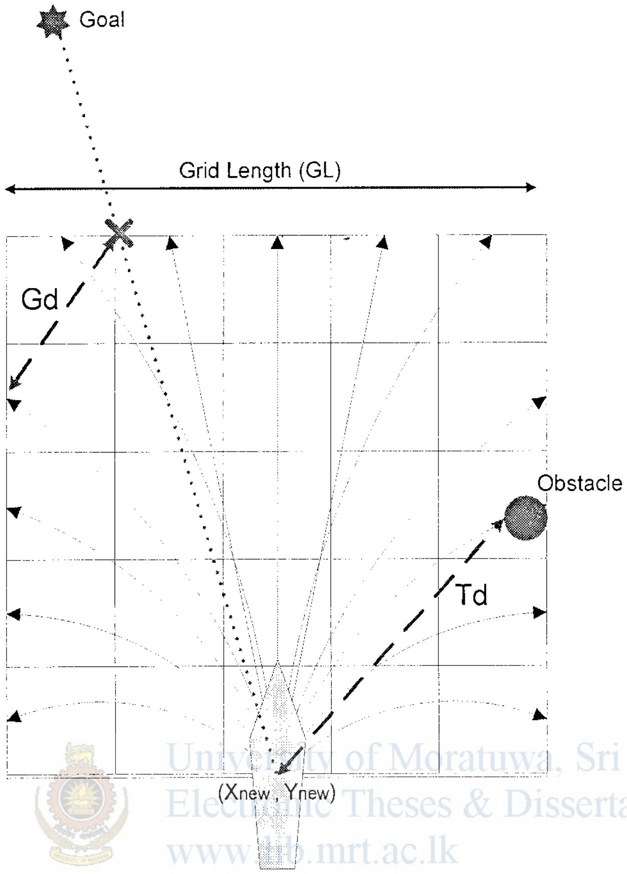


Figure 4.22 - Path value calculation

$$Path\_val(i) = \frac{GL}{(Gd_i + k)} - \frac{GL}{(Td_i + k)} \qquad \text{---------------------------} \quad (4.2)$$

$$where \ \frac{\sqrt{5}}{2} GL > Td_i > 0 \ , \ \sqrt{2} GL > Gd_i > 0$$

Equation 4.2 is utilized to calculate the path values. 10 is assigned in to the constant k for the simulations in this study. Here the paths on the obstacles are assigned small values while other paths are prearranged using high values. A curtain weight values are further added for the paths heading towards the goal as well. So the best path is chosen depending on the weights are assigned at that time [6].

Then the boat is moved certain distance on that path. Again the new obstacle matrix is processed and obstacle matrix is added to the system for other iteration of the same path.

4.2.2 Simulation of Algorithms

. Initially the Novel path planner takes the grid size, sub grid size, goal point and obstacle map. Then the coordinates of the curved paths are evaluated and assigned in to the Ypaths matrix with pathindex. It was straightforward way to find obstacles on each path. Big gaps were formed when y coordinates of the straight line functions were utilized. So that another matrix for the straight line functions is defined for the sack of inconvenience. Then x coordinates of the straight lines are stored in Xpaths. Evaluation of the goal matrix is done by comparing the obstacle matrix with the path matrixes. Minimum distance that boat can travel on each path is stored in the Distance_Matrix function. The above process is continued for the each and every path.

Then the goal direction is evaluated and that value is also employed with the obstacle liberated distance to choose the optimum path among each and every possible path. The boat is traveled ¼ of the grid size on the chosen path their after. That whole process is presented in simply for the convenience of the reader. The above process is developed as a single module. Another module is needed to call that module sequentially until the chosen boat reaches the goal. That algorithm is appeared in the Figure 4.23.

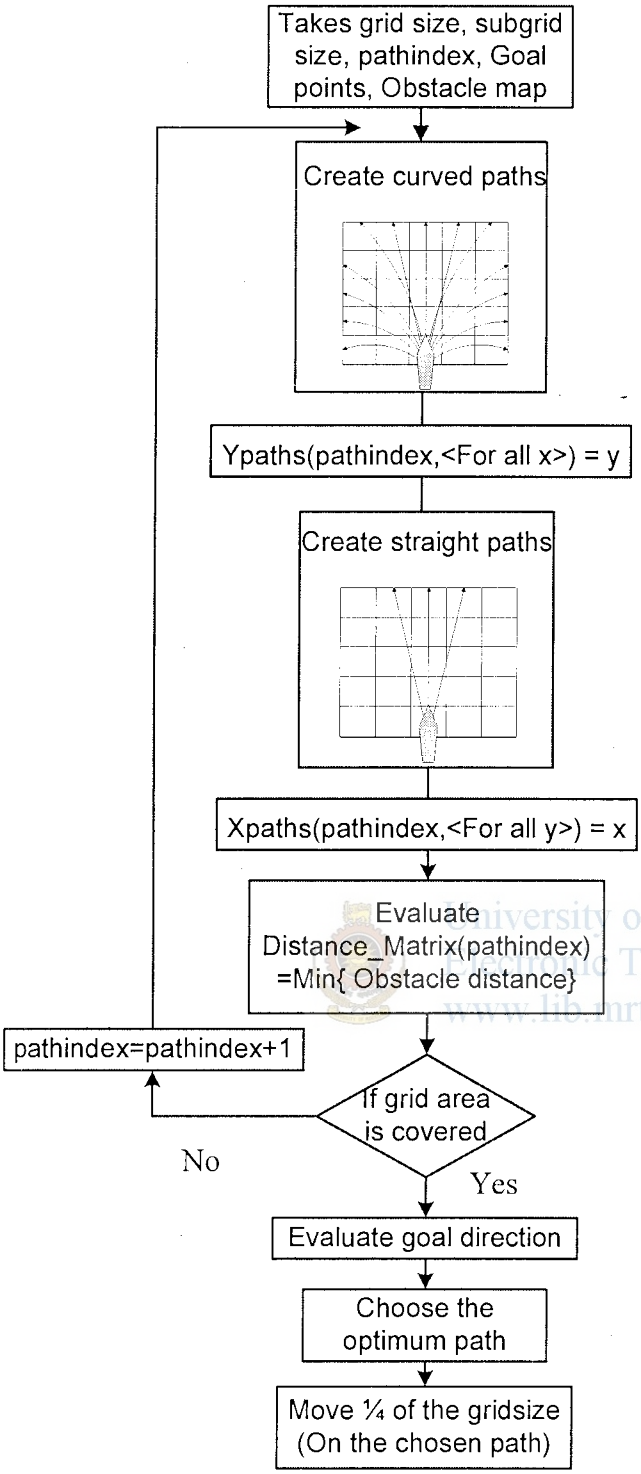

Takes grid size, subgrid size, pathindex, Goal points, Obstacle map
Create curved paths
Ypaths(pathindex,<For all x>) = y
Create straight paths
Xpaths(pathindex,<For all y>) = x
Evaluate Distance_Matrix(pathindex) =Min{ Obstacle distance}
pathindex=pathindex+1
If grid area is covered
No
Yes
Evaluate goal direction
Choose the optimum path
Move ¼ of the gridsize (On the chosen path)

Figure 4.23 - Simplified flow chart of the Novel algorithm

Chapter 5

Dynamic Obstacle Avoidance of USV

Obstacle avoidance without dynamic obstacle is not practical with obstacles on Sea. Dynamic obstacle avoidance is discussed in this chapter in detail. Simulation results relevant to dynamic obstacle avoidance is discussed in the next chapter.

5.1 Introduction to Novel Dynamic Obstacle Avoidance Method

Projected obstacle area method is a very famous method for dynamic obstacle avoidance. Dynamic obstacles can freeze time with the help of that method. Then any type of static obstacle avoidance method can utilize to avoid that. This method is employed in several places on ground and water.

Dynamic obstacle is transformed to another static obstacle which is having large dimensions in the above method. That means it utilized the effective area of the path planning plan which can be employed for path planning. That may be coursed to plan inefficient paths to avoid dynamic obstacles. Then USV may able to travel longer or it may stops suddenly due to lack of effective areas on the path planning plan. So it is very vital to utilize the traveling area effectively as well as avoiding dynamic obstacles. So the novel method is proposed to avoid Dynamic obstacles by employing the minimum areas on the effective area of the path planning plane.

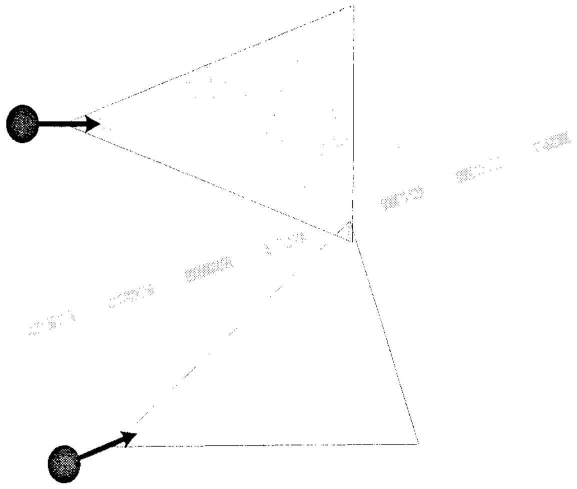


Figure 5.1- Two Dynamic obstacles with Projected Obstacle Areas

Figure 5.1 presents two dynamic obstacles which are freezed with time. Those two predicted areas have blocked the effective path from their edges. The following Figure 5.2 shows the advantage of reducing that predicted areas. So the path planner is generated the efficient path between two obstacle areas.

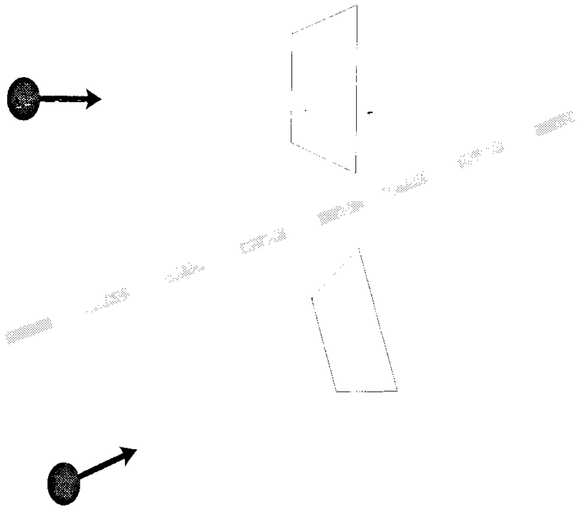
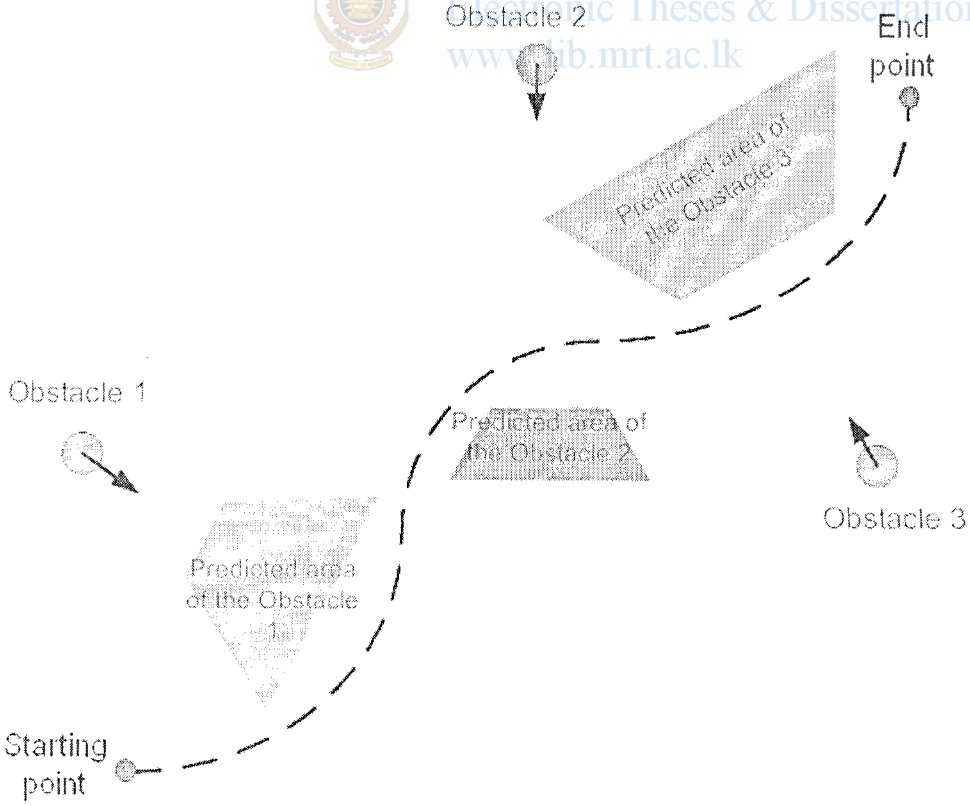


Figure 5.2-Two Dynamic obstacles with their reduced projected obstacle areas



Obstacle 2
End point
Predicted area of the Obstacle 3
Obstacle 1
Predicted area of the Obstacle 2
Obstacle 3
Predicted area of the Obstacle 1
Starting point

Figure 5.3-Path planning between three dynamic obstacles

Figure 5.3 presents the predicted path among 3 dynamic obstacles. These obstacle areas are generated by freezing 3 dynamic obstacles with time. So the path planner considers those areas as another static obstacle and then it can avoid those by utilizing any static obstacle avoidance method.

5.2 Area Prediction of Dynamic Obstacles

The effective time estimation of Dynamic obstacles is very important to avoid them. Some dynamic obstacles near to USV may not cause any effect on the USV. They may travel away from the USV. Some dynamic obstacles may travel towards the USV. So that it is very important to estimate the effective time of a Dynamic obstacle. The time estimated is presented from t_{area} in Equation 5.2 . Equation 5.1 is utilized to calculate T after reveling t_1 and t_2 . t_1 is calculated making use of the velocity of the obstacle while t_2 is calculated by employing velocity of the USV. t_2 is the time taken to the USV to cross the predicted path of a moving obstacle while t_1 is the time taken for that moving obstacle to cross the defined path of the USV. The traveling path of the USV is known before hand and has to utilize a path prediction module to predict the path of obstacles.

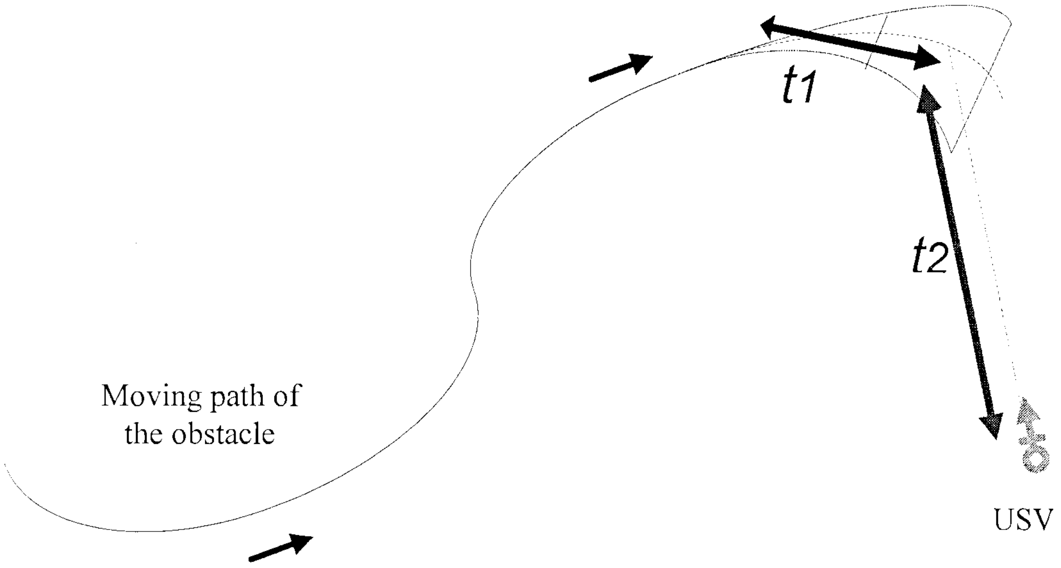


Figure 5.4-Effective time prediction of Dynamic obstacles

$$T = \frac{\Phi}{|t_1 - t_2|} \quad \text{-------------------------------(5.1)}$$

$$t_{area} \in \left[\left(t_1 - \frac{T}{2}\right), \left(t_1 + \frac{T}{2}\right)\right] \quad \text{--------------------------------(5.2)}$$

Where t_2 is the time taken to the USV to cross the predicted path of a moving obstacle, t_1 is the time taken for that moving obstacle to cross the defined path of the USV

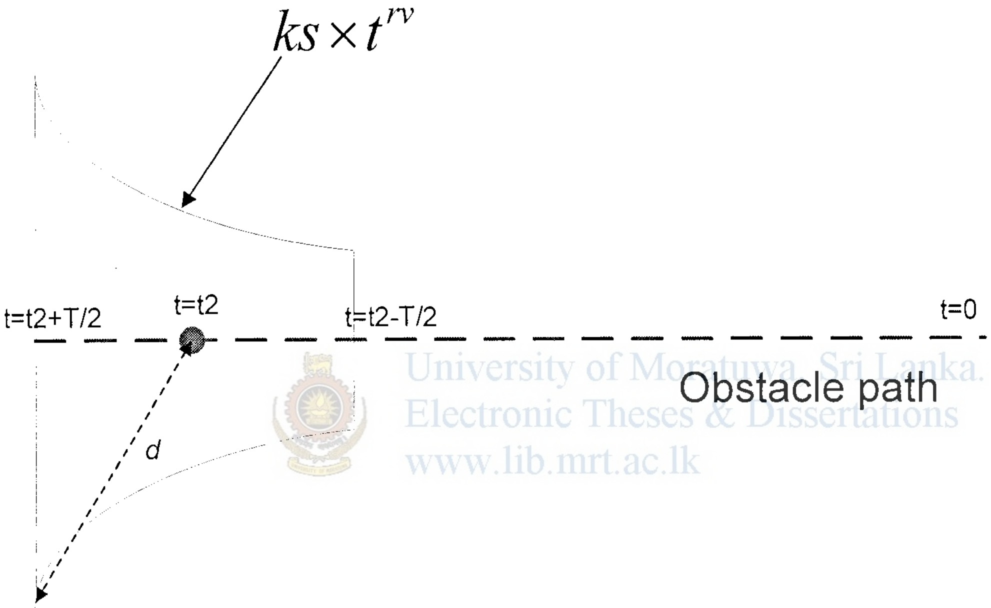


Figure 5.5- Area Calculations for Dynamic obstacle

Figure 5.5 presents the area calculation of Dynamic obstacles. That area is depending on the relative velocity of the obstacle as well. K is chosen depending on the sea. Because the behavior of the obstacle is directly depend on the sea condition as well. The obstacle movement is very high on a ruff Sea. So that predicted areas for obstacles on a ruff sea should be comparatively larger than a normal Sea. High velocities are increasing the possibility of high deviations from the predicted paths. The importance of the obstacle velocity to the predicted area is reflected through new equation well. The constant in the above equations can be change at the practical implementation phase. The dimensions of that area can be changed by changing the constant in the above equation.

5.3 Path Prediction of Dynamic Obstacles

It is very important to determine the path of dynamic obstacles to avoid them. Radars and other obstacle detection methods can utilize to take moving coordinates of the obstacles. Then those data are stored in an array. So, that array can utilized those data to predict the future movements of the obstacles in the next step. It is required to store lateral and longitudinal coordinates with time for path prediction purpose. It is quite complex thing to analyze. So two separate arrays consisting lateral coordinates with time and longitudinal coordinates with time are utilized for analyzes. Figure 5.6 presents a simulation result showing the actual and predicted path in 3-D space.

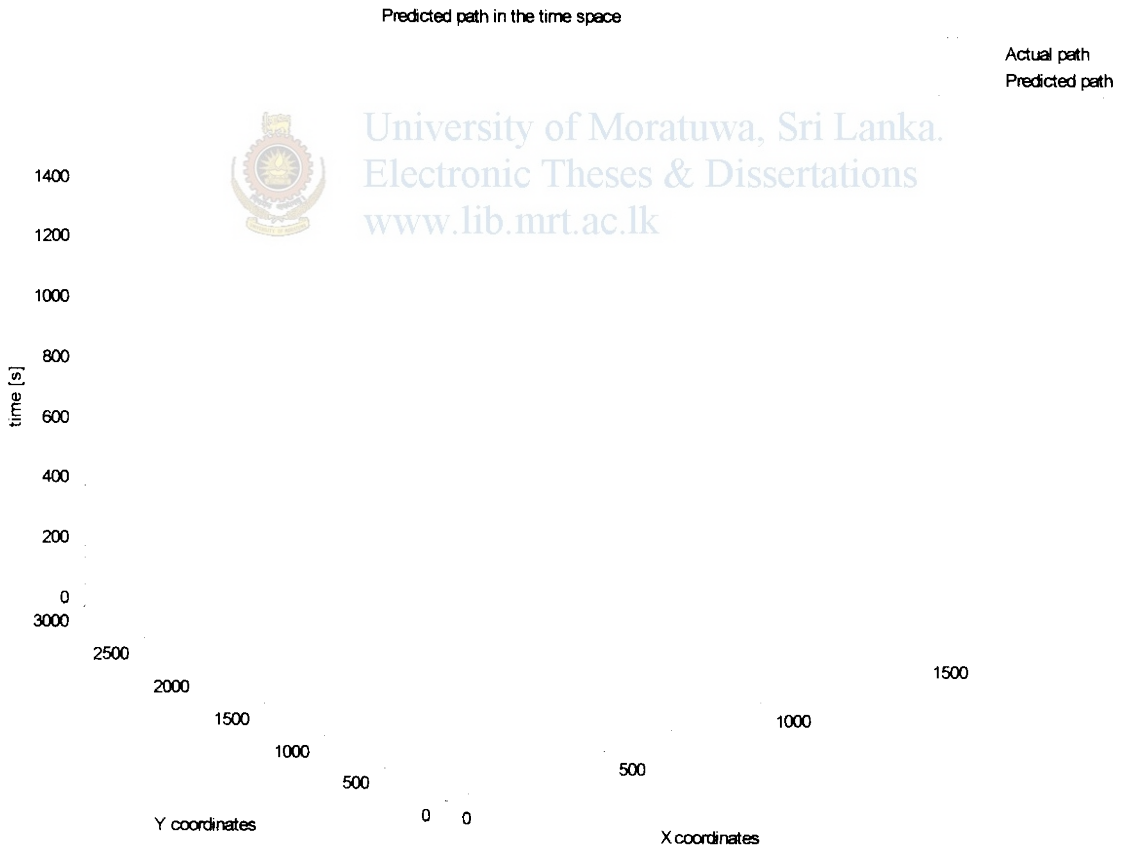


Figure 5.7-Simulation result of an Actual and Predicted path in 3D space

Two prediction methods are utilized for path prediction in this study. First conventional mathematical method is attempted and then generalized regression neural network (GRNN) method is employed [26, 9]. All the results are collected and analyzed at the end.

5.3.1 Polynomial Approximation Method for Path Prediction

Conventional mathematical method, polynomial approximation is utilized to predict the moving path of the obstacles first. The data array which was utilized to store initial positions of that obstacle is taken to approximate the moving function of the obstacle. Two functions have approximates for lateral movement and longitudinal movements. Then those two polynomial functions are employed to predict the path of the obstacle. Several arrays can be employing easily for several Dynamic obstacles as explained earlier. The difference between predicted and actual path is analyze by changing the degree of the polynomial. Obviously the sensor noise cannot be neglected as it is part of the actual data. Because the GPS and Rader data having their own deferent noise. So sensor noise is added to the data and analyzed at the end.

5.3.2 Generalized Regression Neural Network for Path Prediction

GRNN is often used for function approximation. It has a radial basis layer and a special linear layer. It consists of two-layer network. The first layer has radial basis neurons and the second layer has linear neurons [9].

MatLab Neural Network tool box is utilized for the simulations of this study. It is customized by changing the variable name ‘spread’.

A larger ‘spread′ leads to a large area around the input vector where layer 1 neurons will respond with significant outputs. Therefore if ‘spread′ is small the radial basis function is very steep, so that the neuron with the weight vector closest to the input will have a much larger output than other neurons. The network tends to respond with the target vector associated with the nearest design input vector.

As 'spread' becomes larger the radial basis function's slope becomes smoother and several neurons can respond to an input vector. The network then acts as if it is taking a weighted average between target vectors whose design input vectors are closest to the new input vector. As 'spread' becomes larger more and more neurons contribute to the average, with the result that the network function becomes smoother.

Chapter 6

Simulation Results

6.1 Simulation Results by Applying UGV theories for USV

The path planning with two obstacles by utilizing algorithms which are developed and tested for UGV is given below. Figure 6.1 shows planed paths with obstacles having 50m safety distance. The safety distance is equal to 60m of the obstacles which are utilized for the next simulations, presented in figure 6.2.

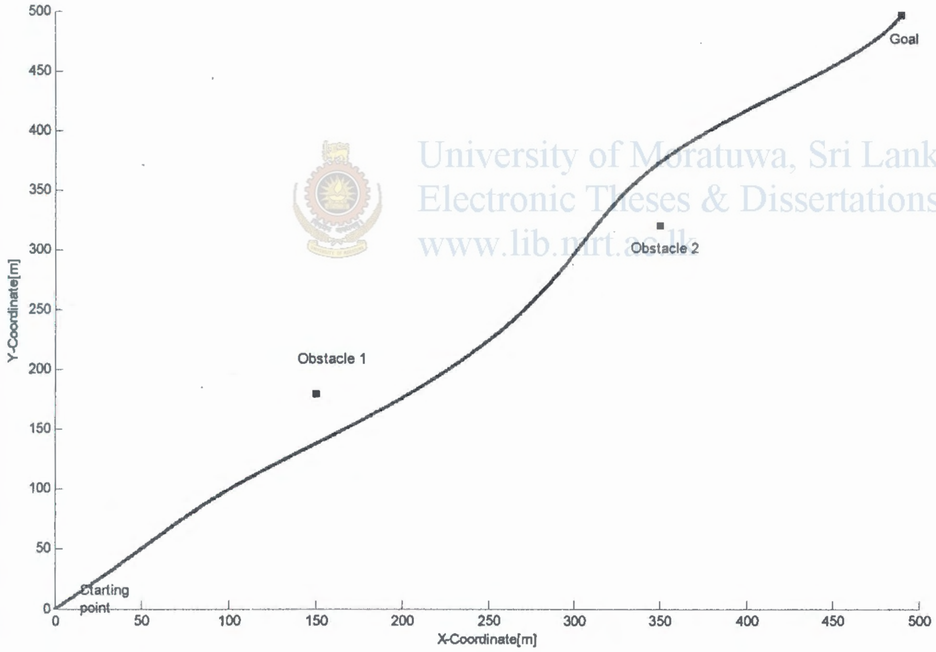


Figure 6.1- Obstacle avoidance using UGV algorithms
(Safety distance from Obstacles = 50m)

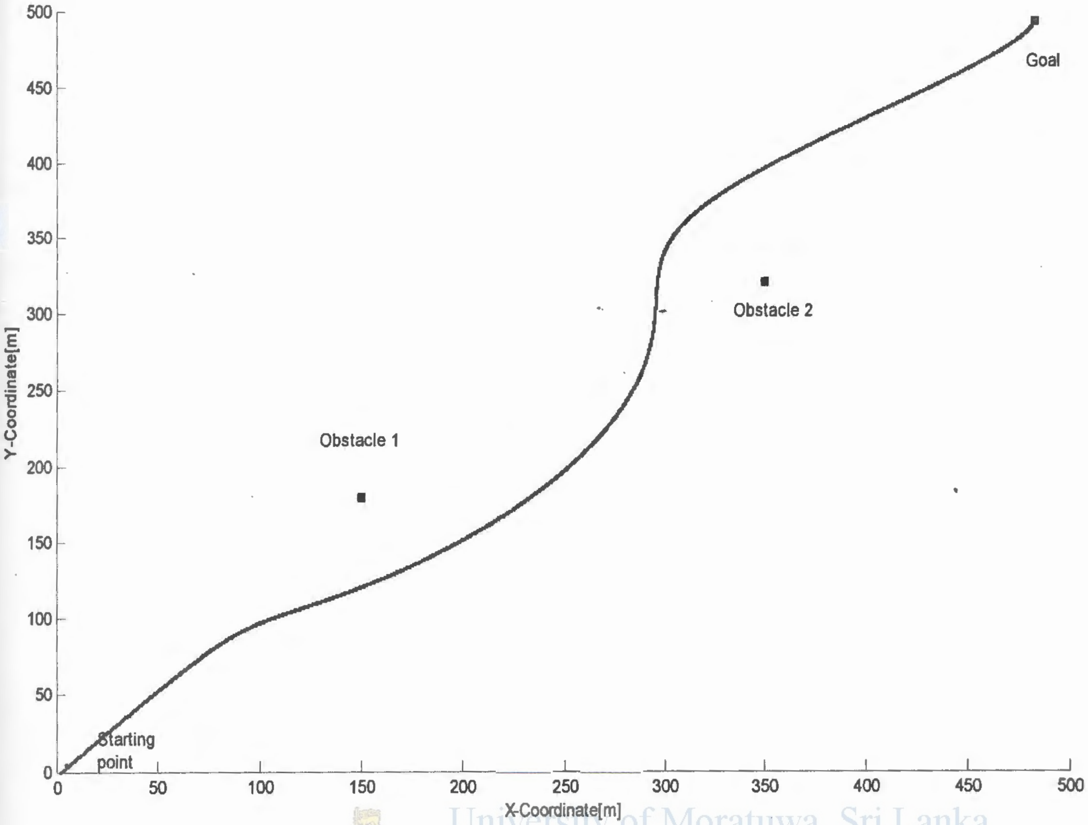


Figure 6.2- Obstacle avoidance using UGV algorithms (Safety distance from Obstacles = 60m)

6.2 Simulation Results from the Novel Algorithm

Possible paths have to generate first to choose the best among them as discussed in the Chapter 5. So that possible paths of the USV are generated as shown in Figure 6.3 . All possible paths are generated in a 20x20 matrix. That can be used to presents a 20x20m area or multiples of that similarly (40x40m) on the Sea or water.

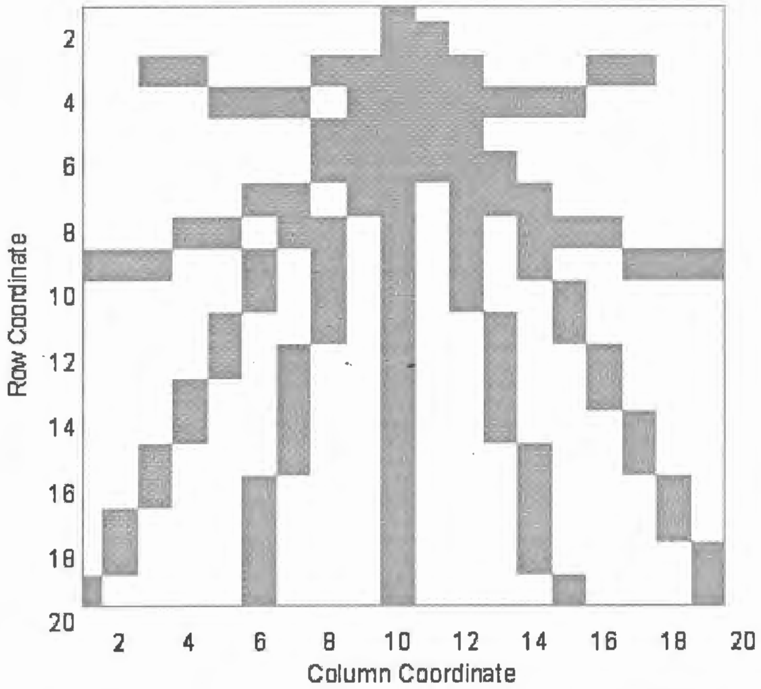


Figure 6.3 - Possible paths matrix of the USV

Obstacle in sea can identify by utilizing Radars, stereo vision or any other methods successfully. But those obstacles data have to interpret to the algorithm. Figure 6.4 to 6.7 is presented different obstacle matrix for different obstacle positions. The whole path matrix is plotted there with red color paths. Then square shape obstacles were placed on different locations of the grid. It obviously should be a 20x20 matrix since the dimensions of the possible path matrix should be equals to the dimensions of the obstacle matrix. Relavent MatLab program is given with appendix D.

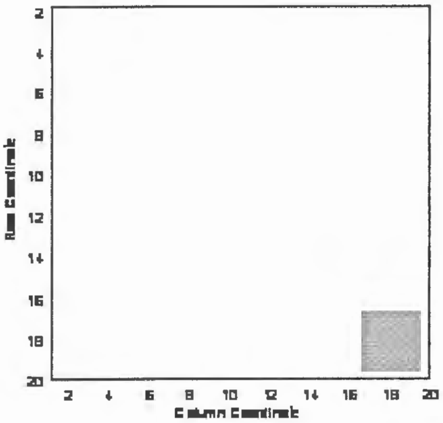


Figure 6.4 – Obstacle on the Obstacle matrix A

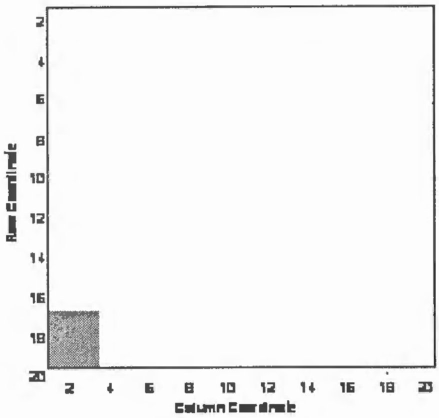


Figure 6.5 – Obstacle on the Obstacle matrix B

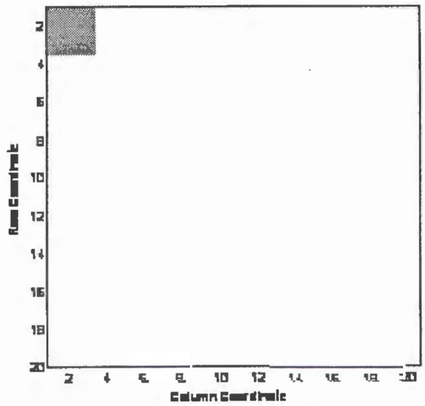


Figure 6.6 – Obstacle on the Obstacle matrix C

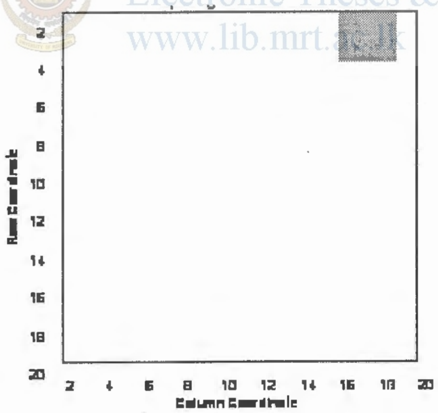


Figure 6.7 – Obstacle on the Obstacle matrix D

Table 1 presents the maximum traveling units on each and every path. Then those table data is utilized to calculate path values. After considering path values ninth path is chosen for the obstacle matrix A. 5th, 1st and 2nd paths are chosen for the obstacle matrices B, C and D respectively.

Path Number / Obstacle Matrix	1	2	3	4	5	6	7	8	9
A	40	40	40	40	40	40	40	40	17
B	40	40	40	40	18	40	40	40	40
C	3	40	40	40	40	40	40	40	40
D	40	3	40	40	40	40	40	40	40

Table 6.1- Maximum Safe traveling distance on each path

Figure 6.8 and 6.9 presents path planning with obstacles which are having safety distances 40m and 60m respectively.

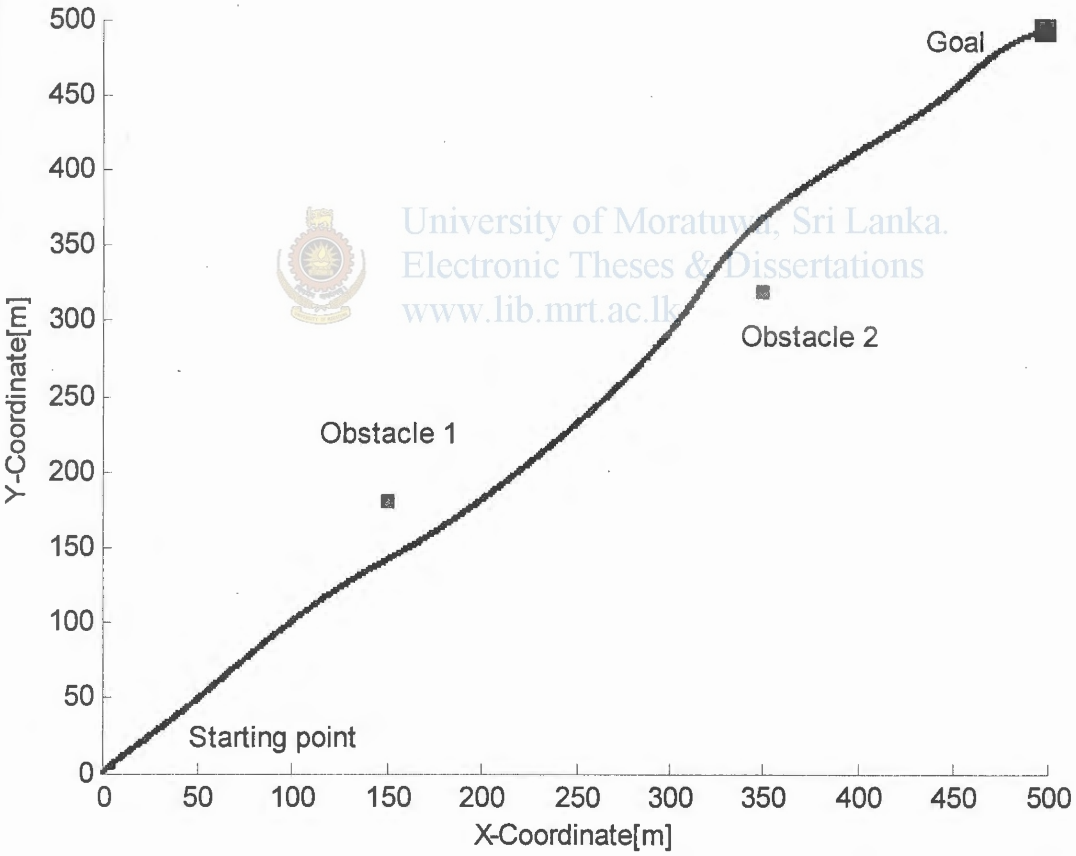


Figure 6.8 - Obstacle avoidance using the Naval algorithm (Safety distance from Obstacles = 40m)

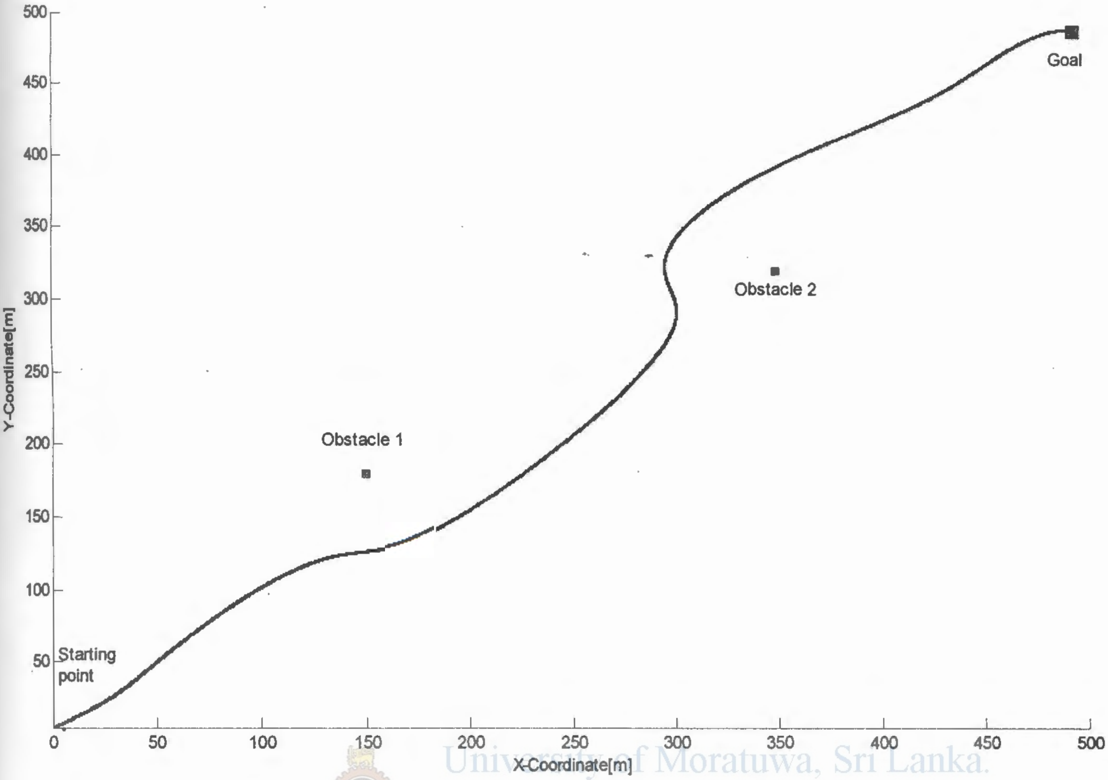


Figure 6.9 - Obstacle Avoidance using the Naval Algorithm based on Morphin Algorithm

(Safety distance frɔm 'Obstacles = 60m)

6.3 Simulation Results from Potential Field Method

Already developed software is employed for path planning with PFM. Figure 6.10 and 6.11 present the contour map and 3 dimensional surfaces of 3 obstacles. Path planning with PFM is processed by considering minimum potential from starting point.

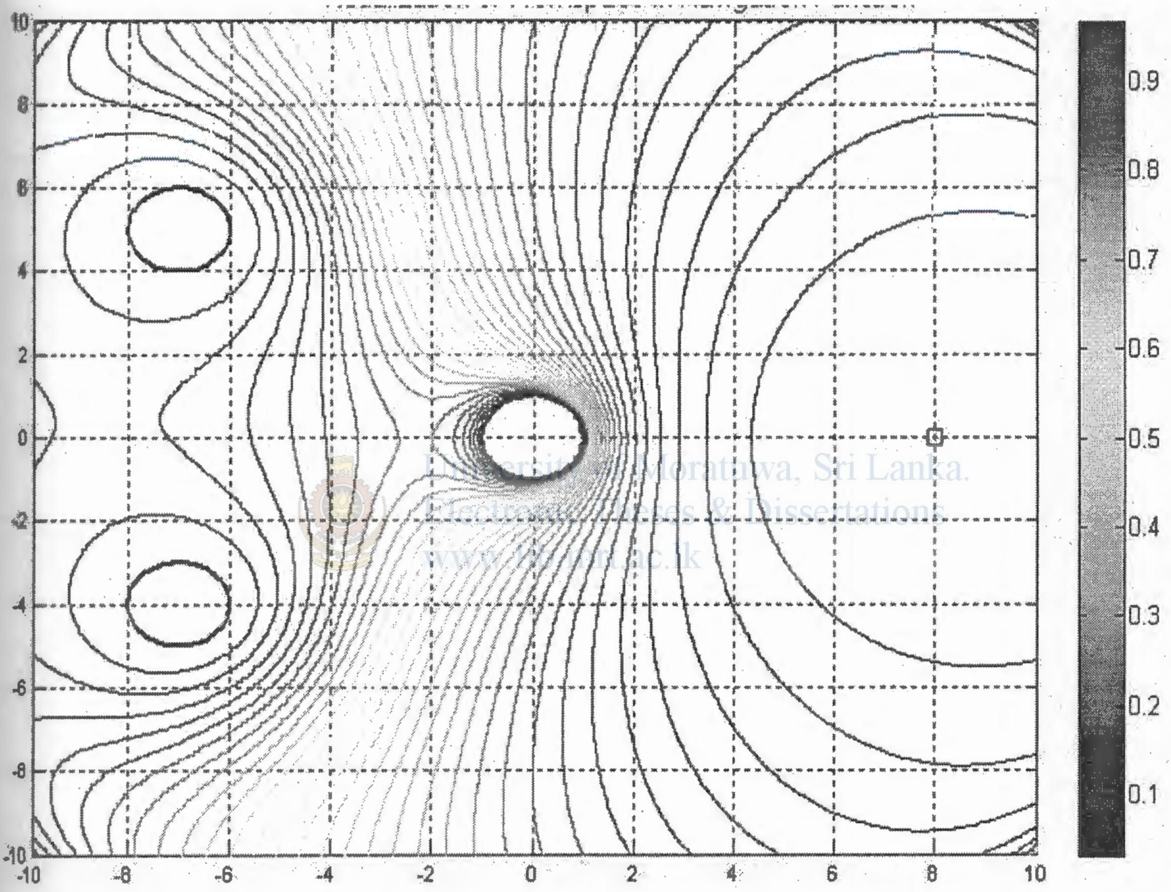


Figure 6.10 – Contour map of Potential field with 3 obstacles

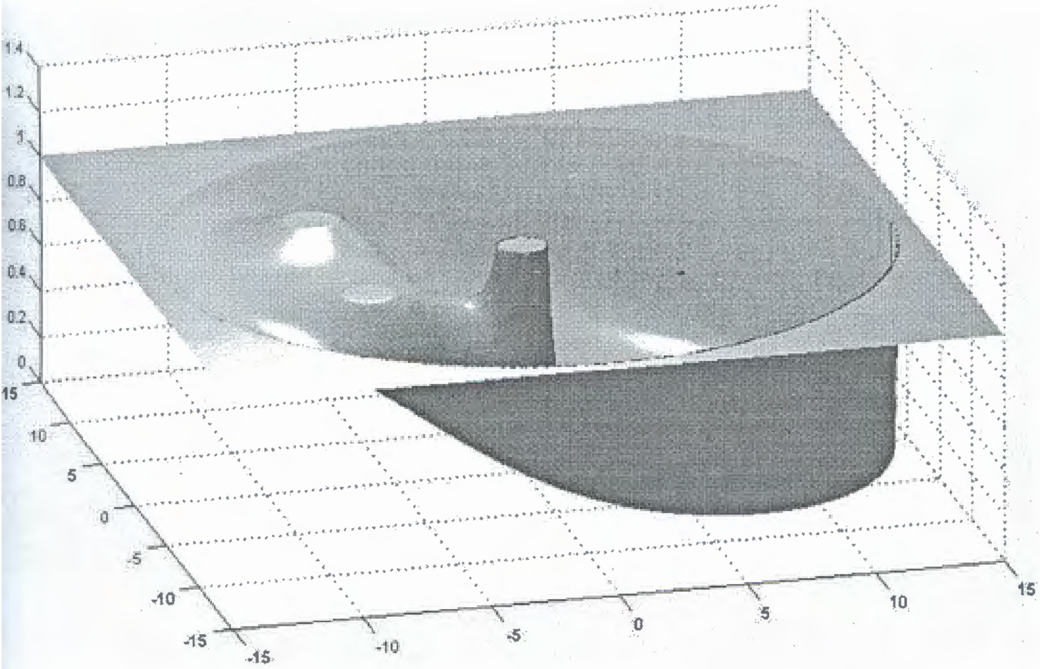


Figure 6.11 – 3D surface of Potential field with 3 obstacles

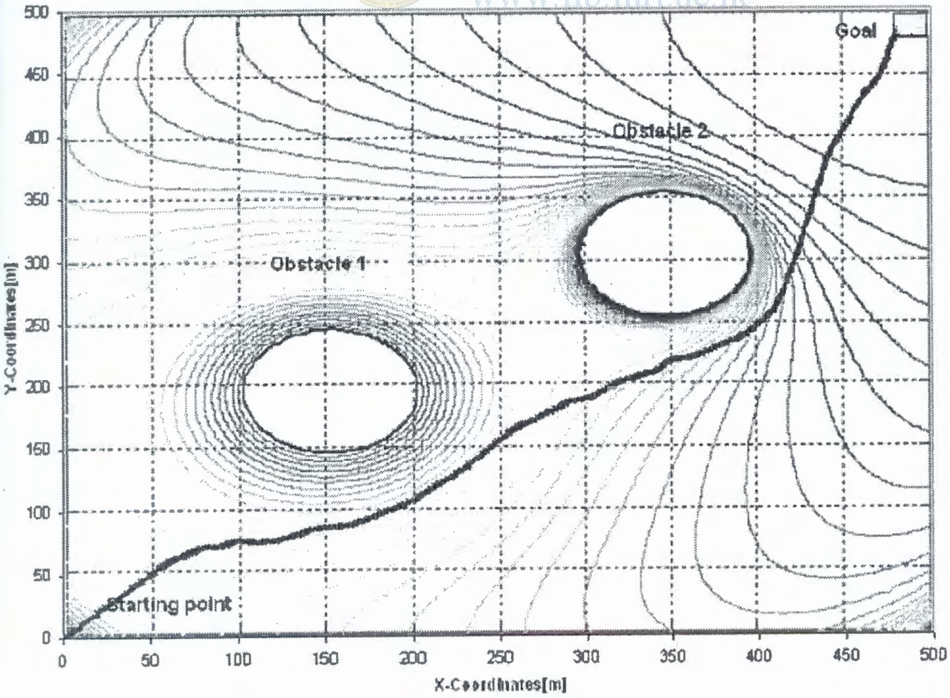


Figure 6.12 - Obstacle avoidance using PFM
(Safety distance from Obstacles = 50m)

6.4 Comparison of Obstacle avoidance methods

The other two obstacle avoidance methods were compared and results are presented in Figure 6.13 and 6.14. The travel distance verses safety distance is compared in the Figure 6.13. According to the results which are presented in the figure shows that Novel algorithm is capable of achieving the target with minimum traveling distance. Potential field method shows minimum travel distance at 50m safety distance since it was changed its traveling path between two obstacles to another path which was away from the two obstacles.

Distance towards obstacles verses time were taken for all three methods and results are given using three different colors in the figure 6.14. It shows that the generated paths utilizing potential field method is far away from the safety area of the obstacle compared with other methods. That is the main reason for traveling longer when the potential field is used.

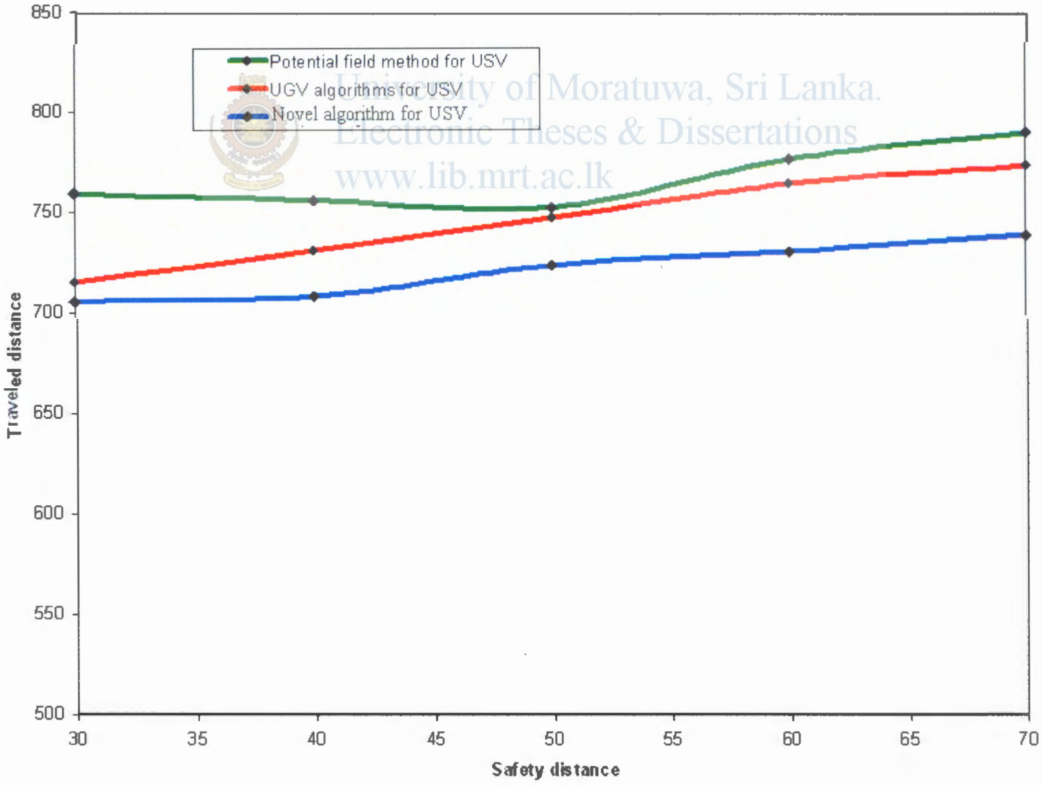


Figure 6.13 - Comparison of safety distances

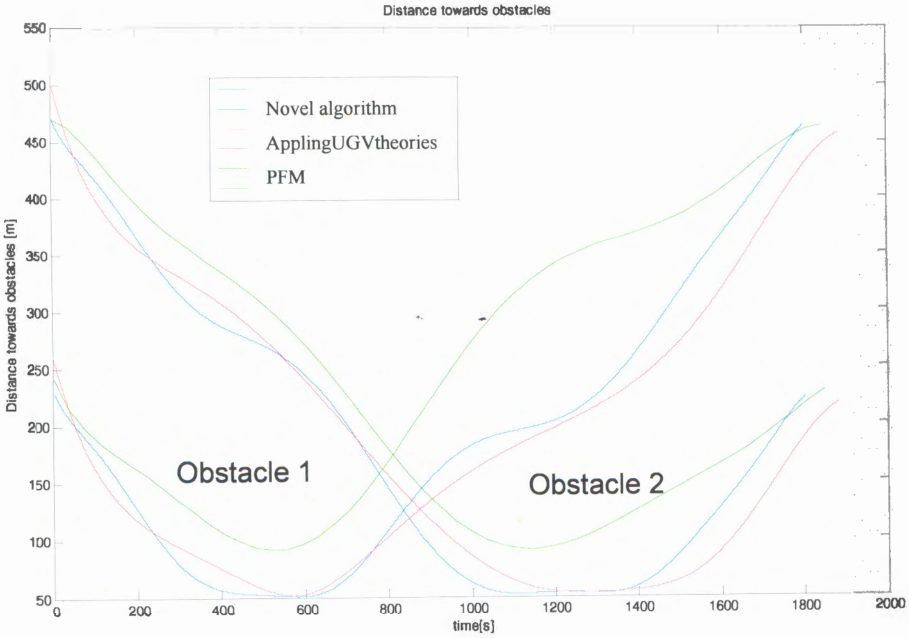


Figure 6.14 - Comparison of distances towards two obstacles

6.5 Simulation Results for Dynamic Obstacles Avoidance

Figure 1 and 2 present the make use of static obstacle avoidance for dynamic obstacles. The time information of the USV and the obstacle1 are shown in the figure 1 for the convenience. Dynamic obstacles can freeze on their moving path with time. So they become another static obstacle at the end. But it is required to predict the moving path of the obstacle before hand. Simulation results from the path prediction methods which are discussed above are presented below. Effective area prediction is also equally important as path prediction. So set of simulations have done on that and those simulation results are presented after the path prediction results as shown below.

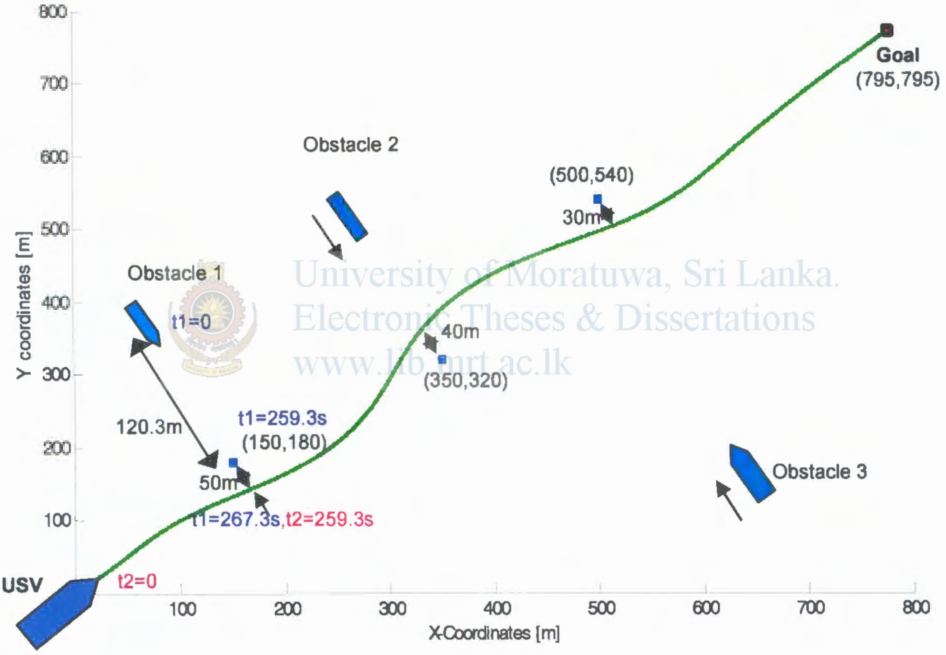


Figure 6.15 – Dynamic obstacle avoidance with time values

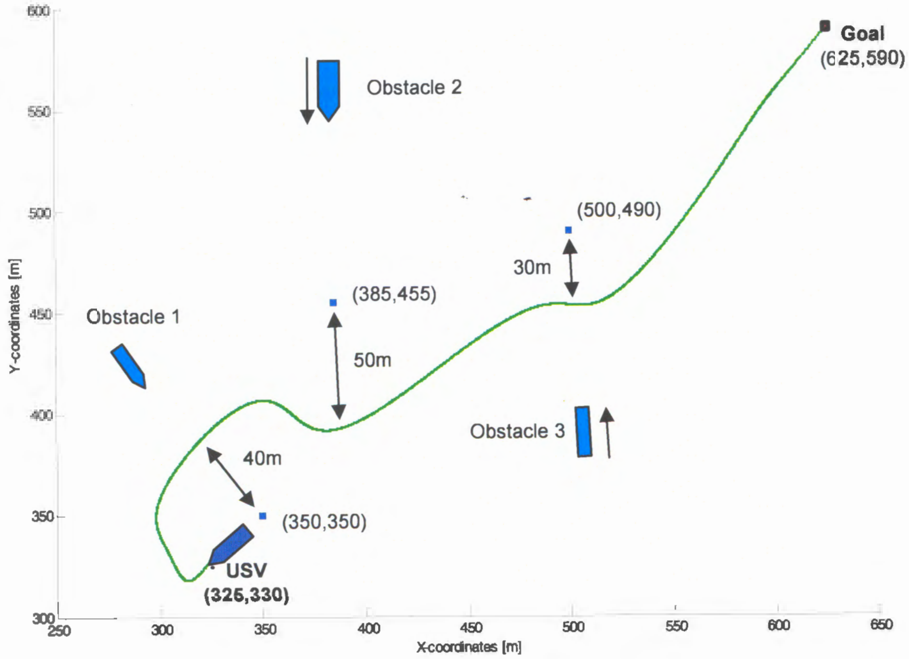


Figure 6.16 – Dynamic obstacle avoidance with 3 obstacles

6.5.1 Simulation Results for path prediction of Dynamic Obstacles

For the simulations of path prediction, Octomorphic path is generated as the actual path of the obstacle. But the path planning module is only supplied with past 100 coordinates of the obstacle movements in USV frame. Then path prediction module is predicted another 50 coordinates. Then those coordinates and the error values are plotted.

6.5.1.1 Polynomial approximation method for path prediction

Simulations done utilizing Polynomial approximation method considering sensor noise and without noise.

6.5.1.1.1 Simulations without sensor noise

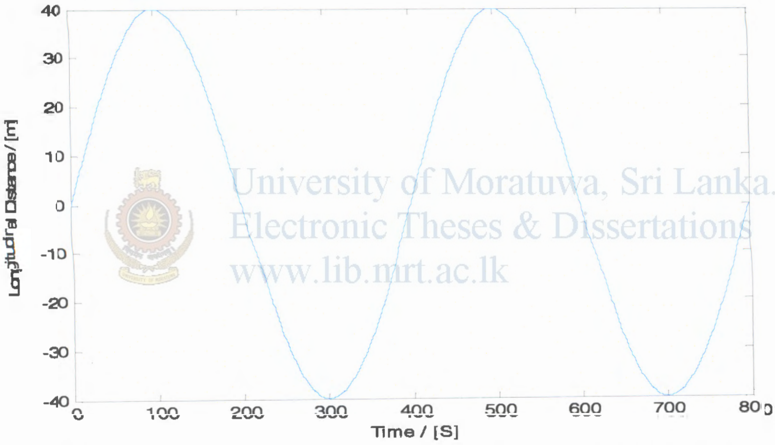


Figure 6.17 – Longitudinal coordinates of actual path

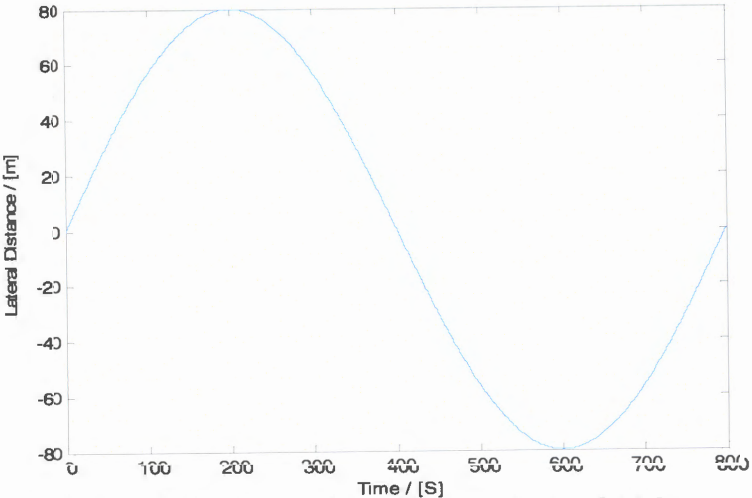


Figure 6.18 – Lateral coordinates of actual path

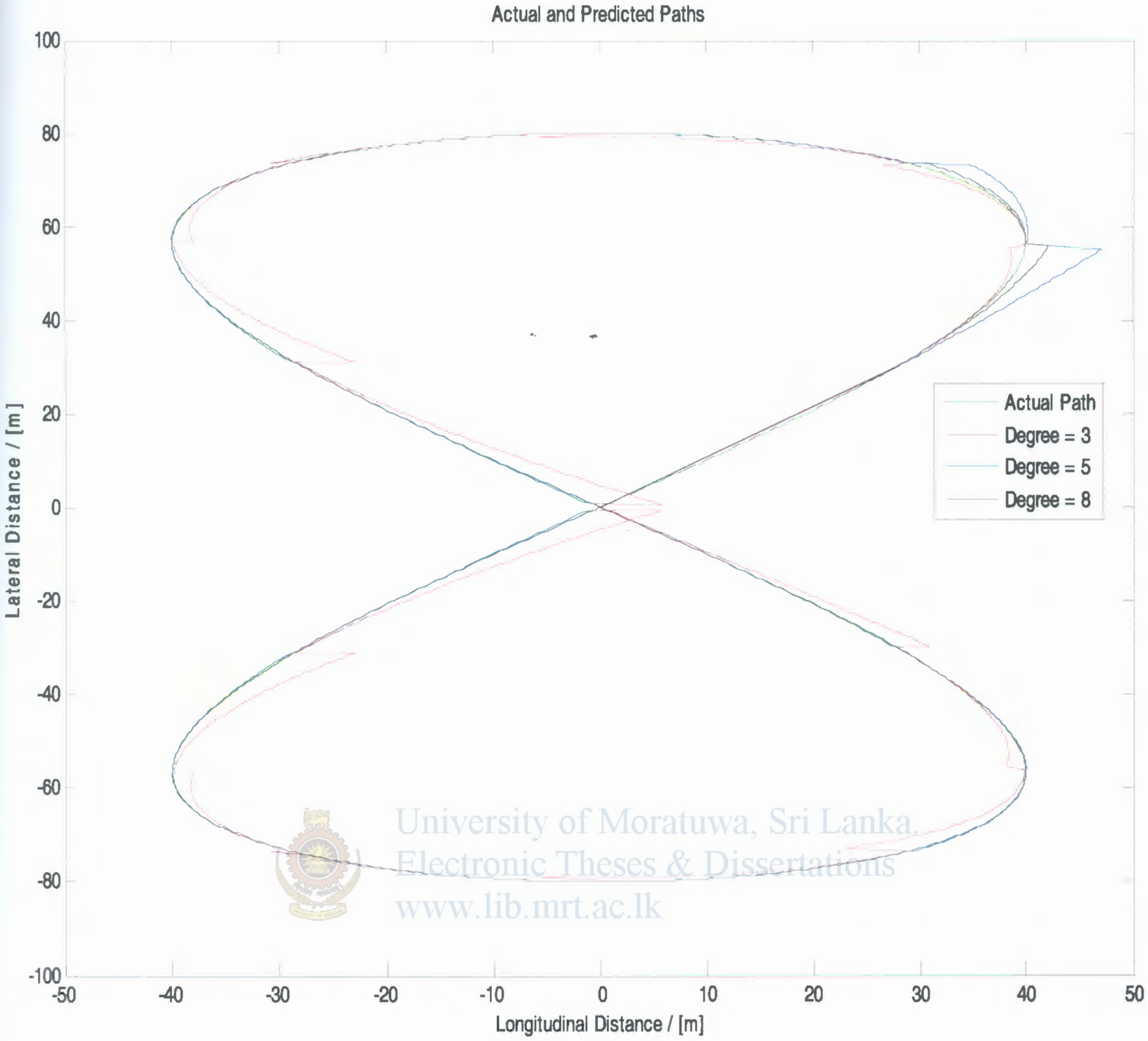


Figure 6.19 – Actual path and Predicted paths of the Obstacle for different degrees of Polynomials

Figure 6.17 and 6.18 presents the lateral and longitudinal value changes respect to time. Then Figure 6.19 shows the actual Octomorphic path of the obstacle. Predicted paths after changing the degree of the polynomials are presented in a same figure to compare the effect of the degree. Figure 6.20 and 6.21 shows the lateral and longitudinal errors after changing the degree of the polynomial. A developed MatLab program for this is given in appendix E.

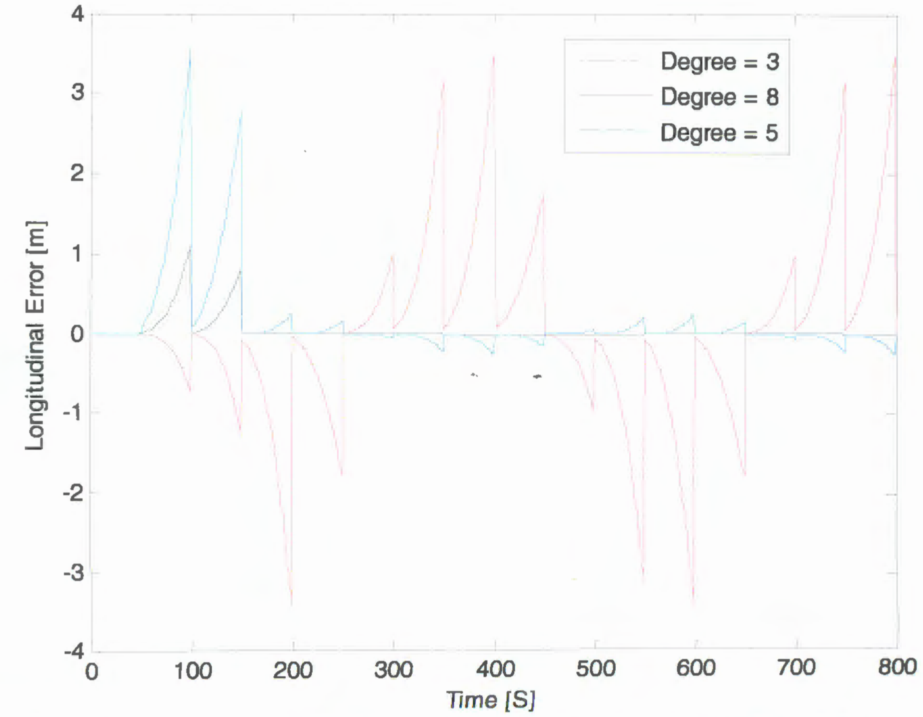


Figure 6.20 – Longitudinal error of predicted paths for different degrees of Polynomials

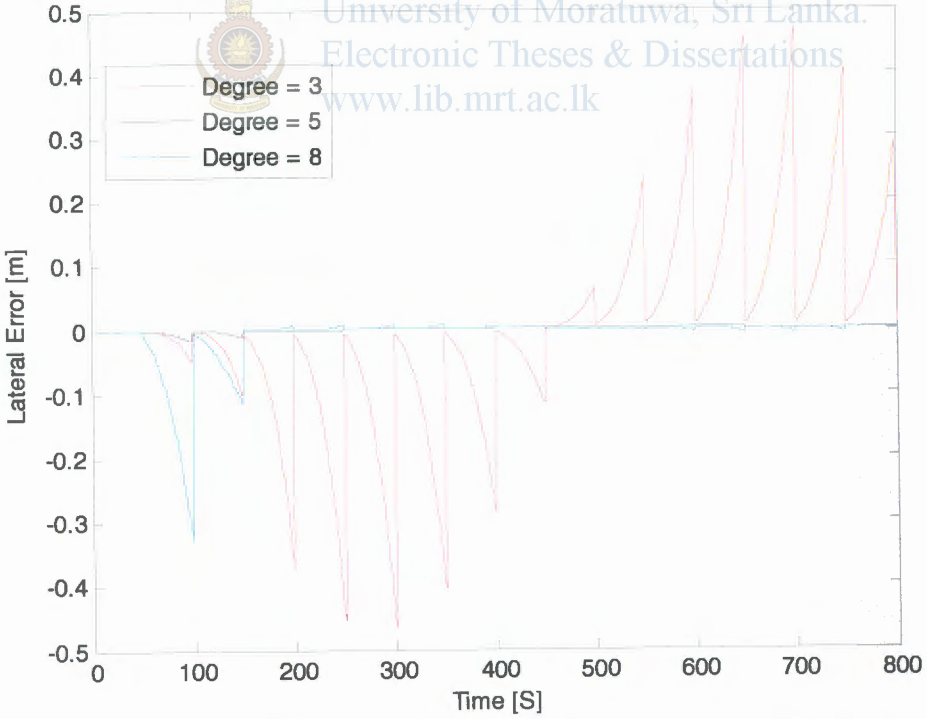


Figure 6.21 – Lateral error of predicted paths for different degrees of Polynomials

6.5.1.1.2 Simulations with sensor noise

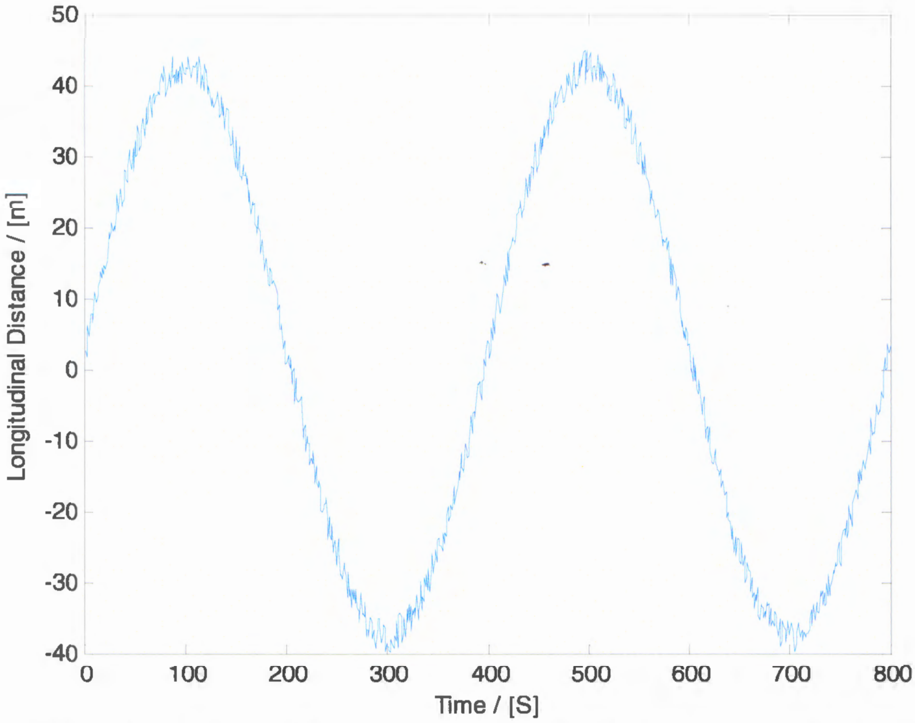


Figure 6.22 – Longitudinal coordinates of actual path with noise (noise=5m)

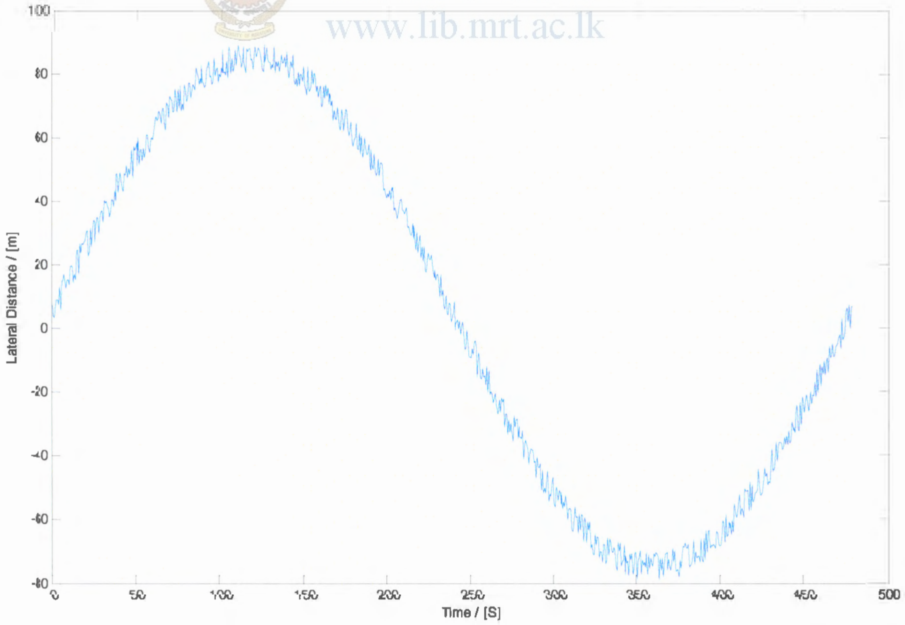


Figure 6.23 – Lateral coordinates of actual path with noise (noise=5m)

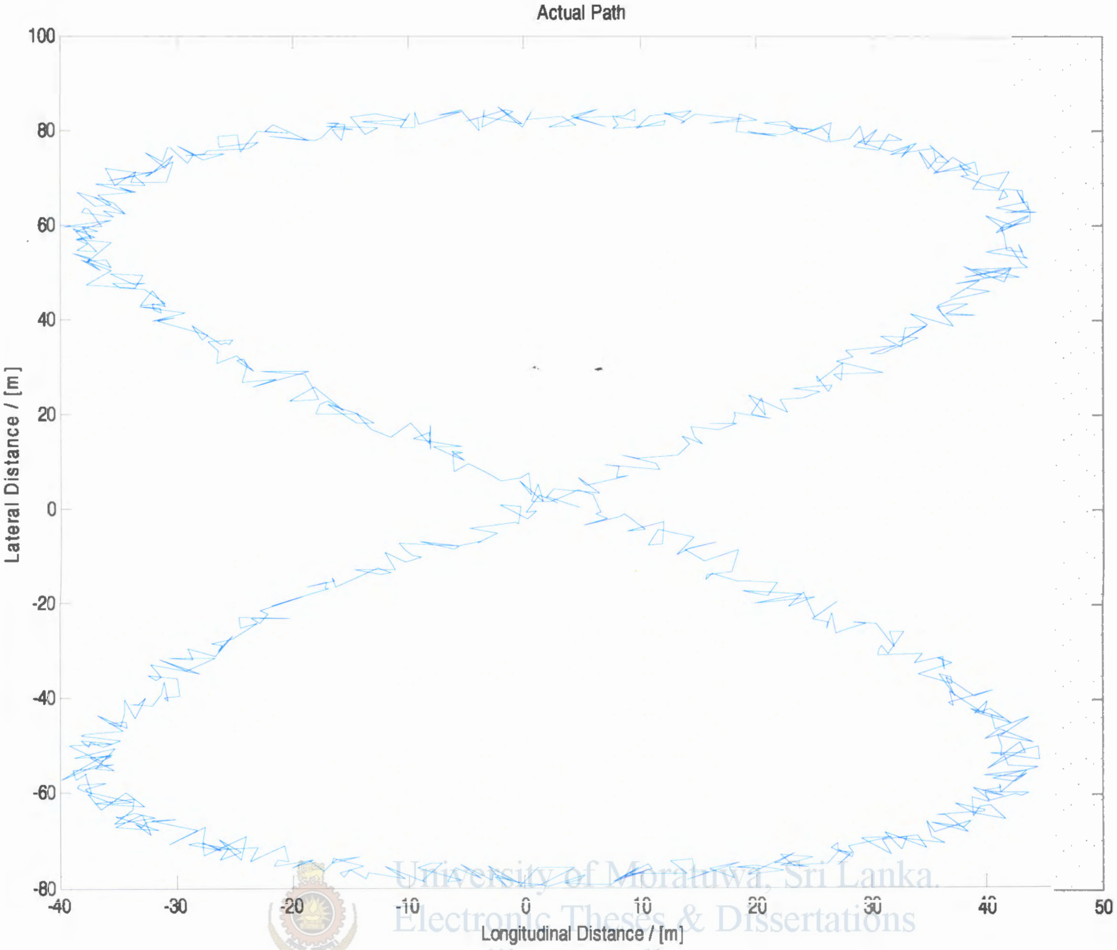


Figure 6.24 – Actual path of the Obstacle with noise (noise=5m)

Figure 6.22 and 6.23 shows the lateral and longitudinal value changes with noise respect to the time. Then Figure 6.24 presents the actual Octomorphic path of the obstacle with respect to time. Then predicted path is generated for noisy data with 6 degree polynomial. That is given in Figure 6.25. Then that degree is changed to 5 and 4 respectively as shown in the Figure 6.26 and 6.27.

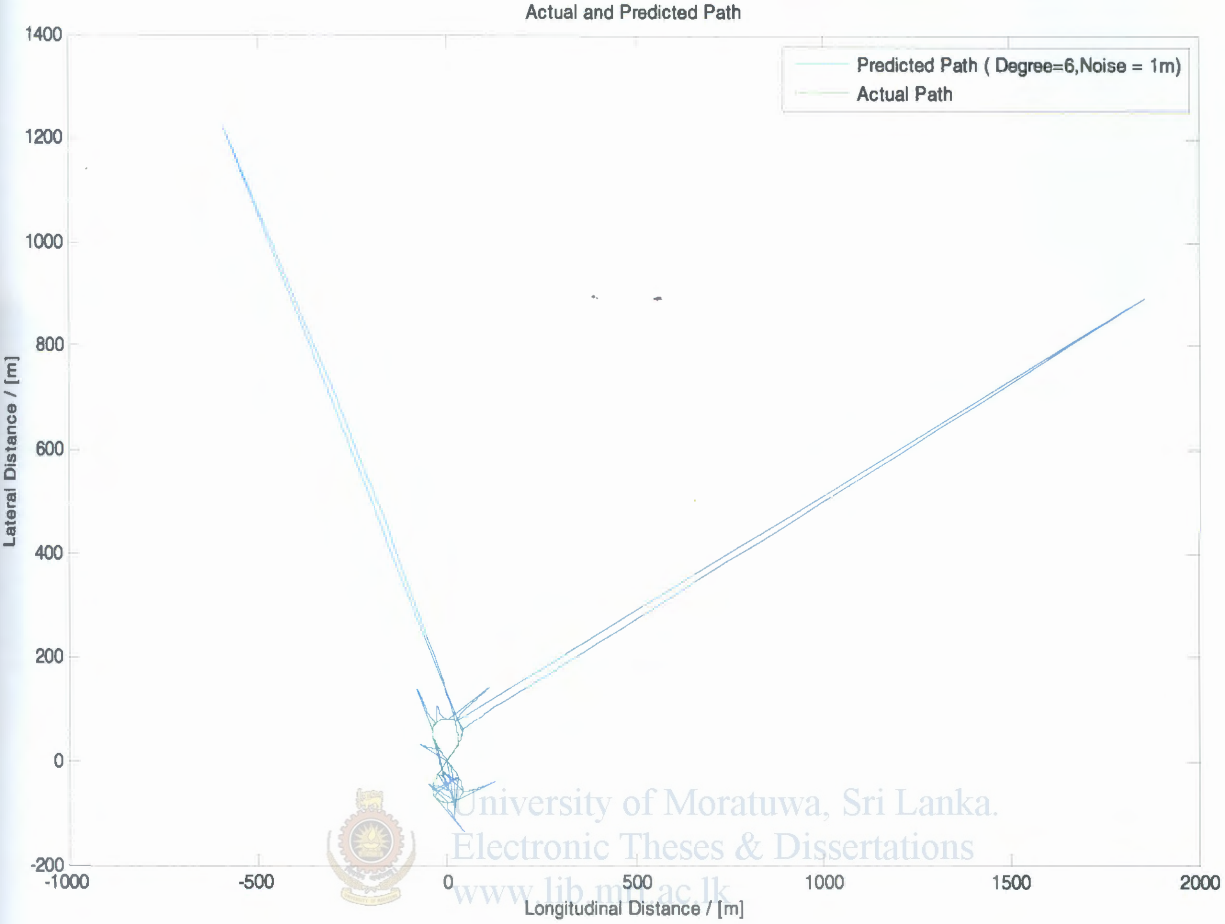


Figure 6.25 – Actual and Predicted path of the Obstacle (Degree of Polynomial = 6, noise = 1m)

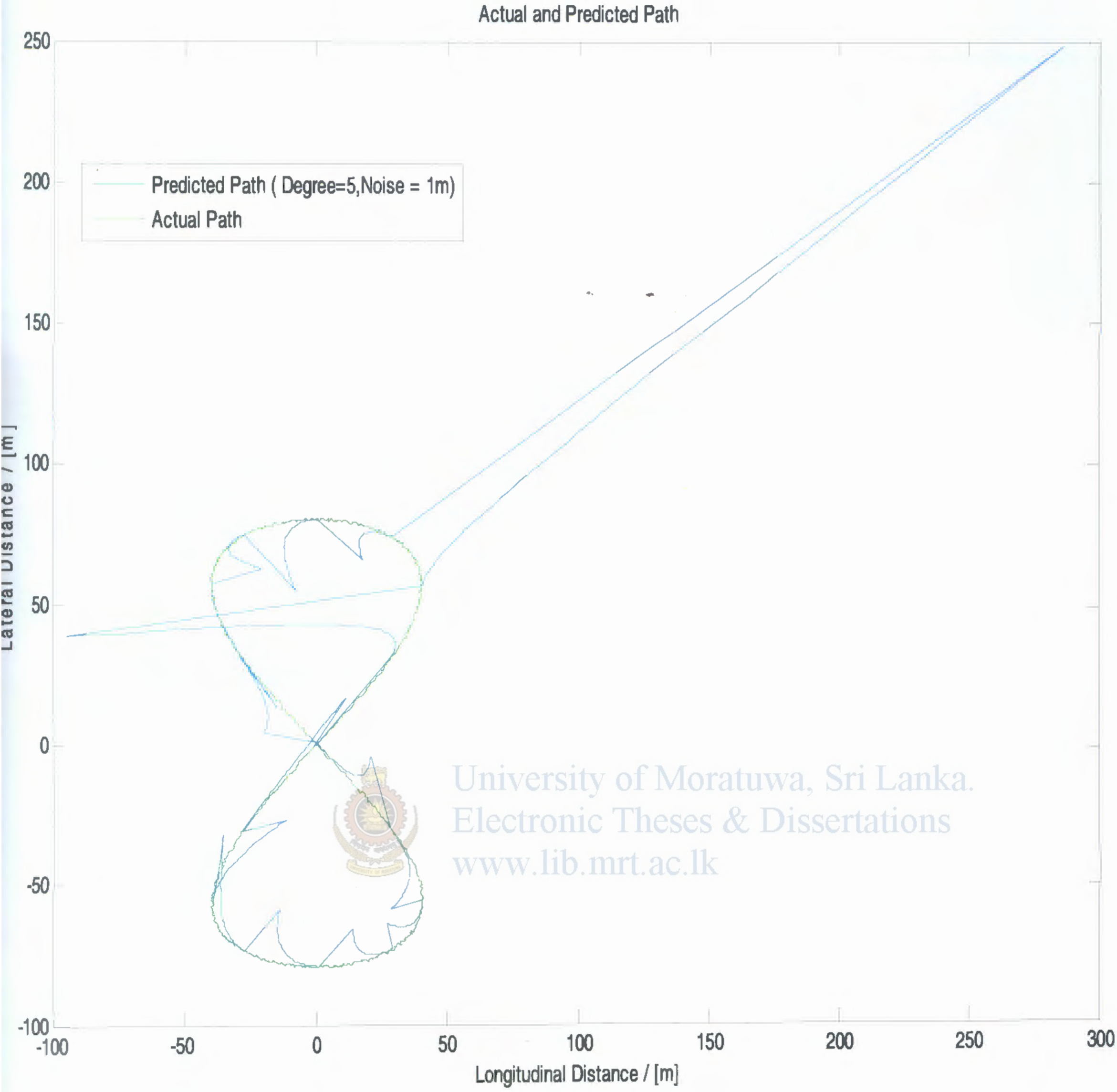


Figure 6.26 – Actual and Predicted path of the Obstacle
(Degree of Polynomial = 5, noise = 1m)

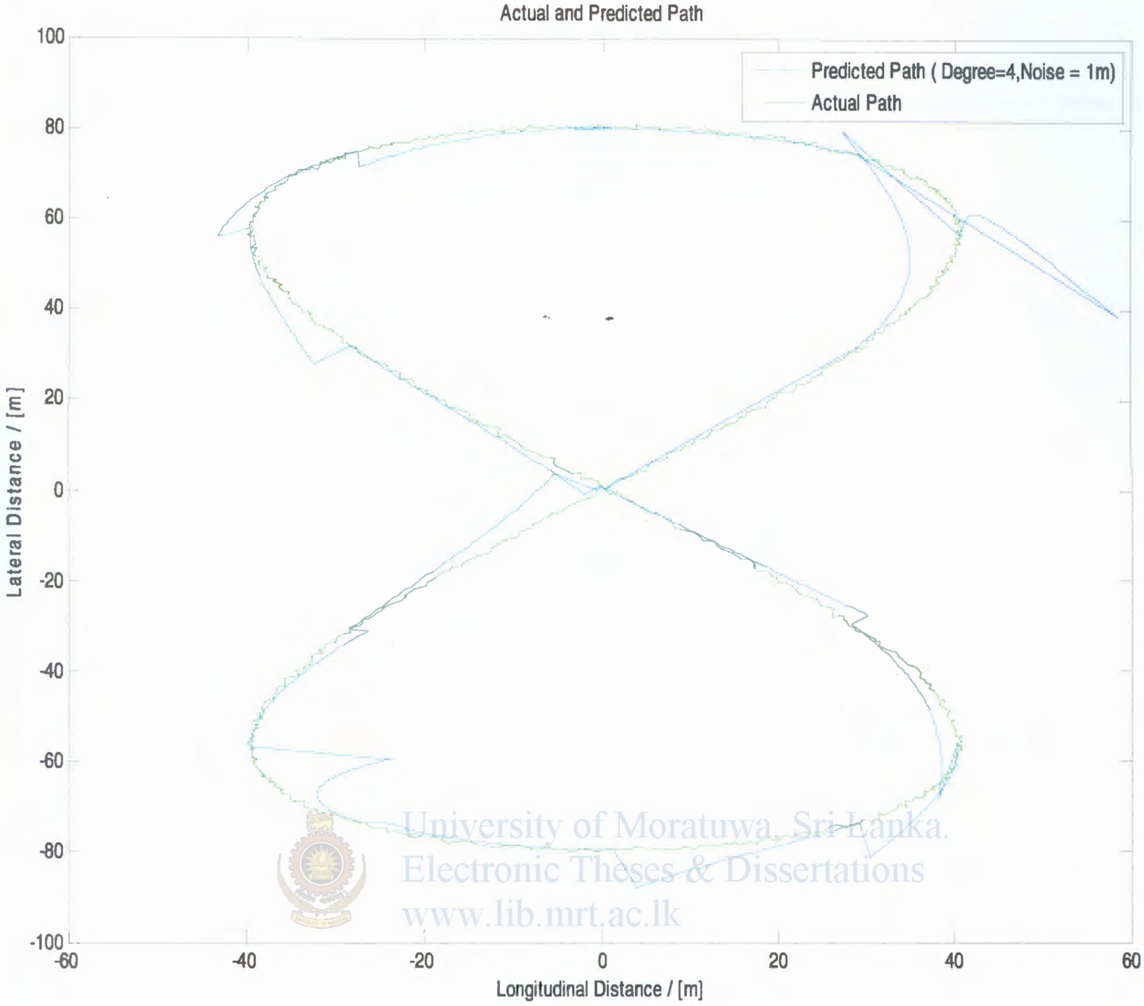


Figure 6.27 – Actual and Predicted path of the Obstacle (Degree of Polynomial = 4, noise = 1m)

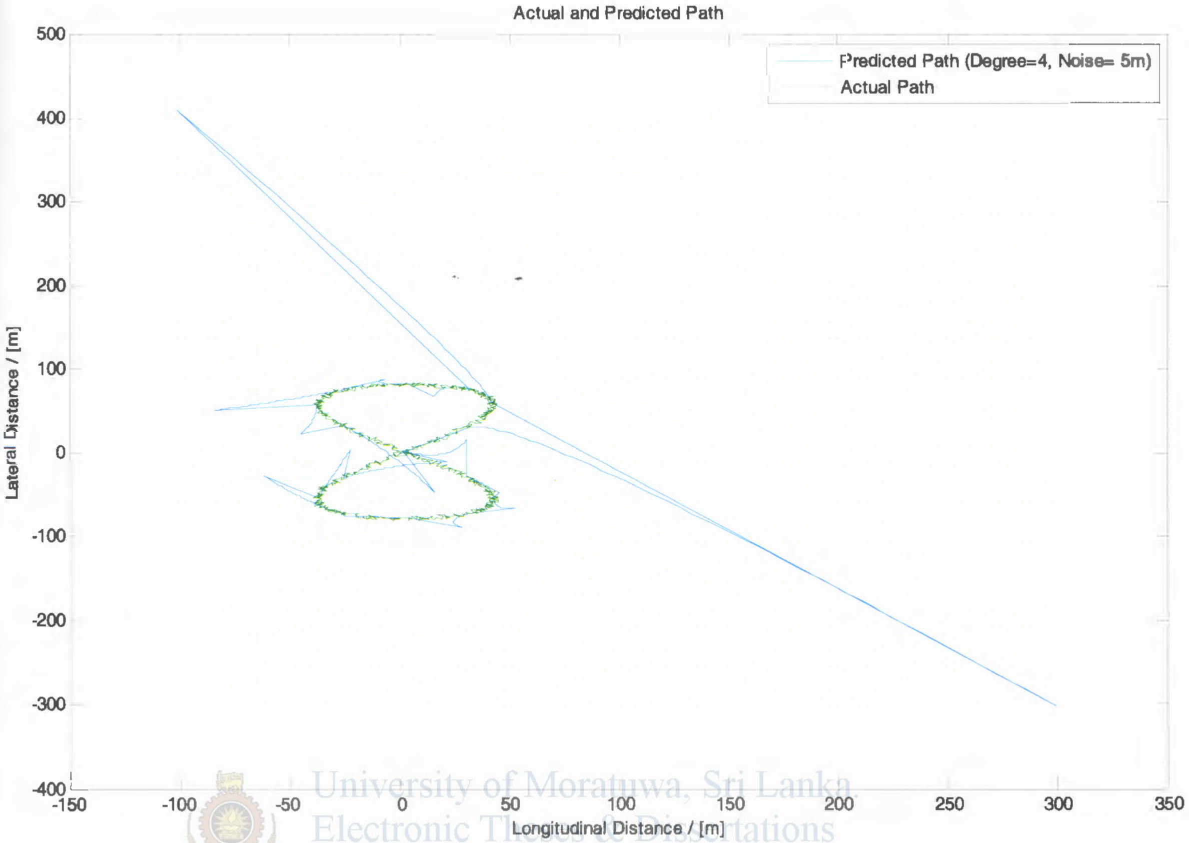


Figure 6.28 – Actual and Predicted path of the Obstacle

(Degree of Polynomial = 4, noise = 5m)

One meter noise is added to the figure 6.25,6.26 and 6.27. Quite improved predicted path is received with small (1m) noise when degree of the polynomial is equal to four. Then noise is increased by 4m to analyze the effect of noise. Normally even the improved GPS receivers are having error of two or three meters alone. So the noise is changed from 1m to 5m of 4th degree polynomial. That result is given in figure 6.28 and it proves that the polynomial approximations methods are not good with noise.

6.5.1.2 RBNN method for path prediction

Simulations are done utilizing RBNN method considering and without considering sensor noise.

6.5.1.2.1 Simulations without sensor noise

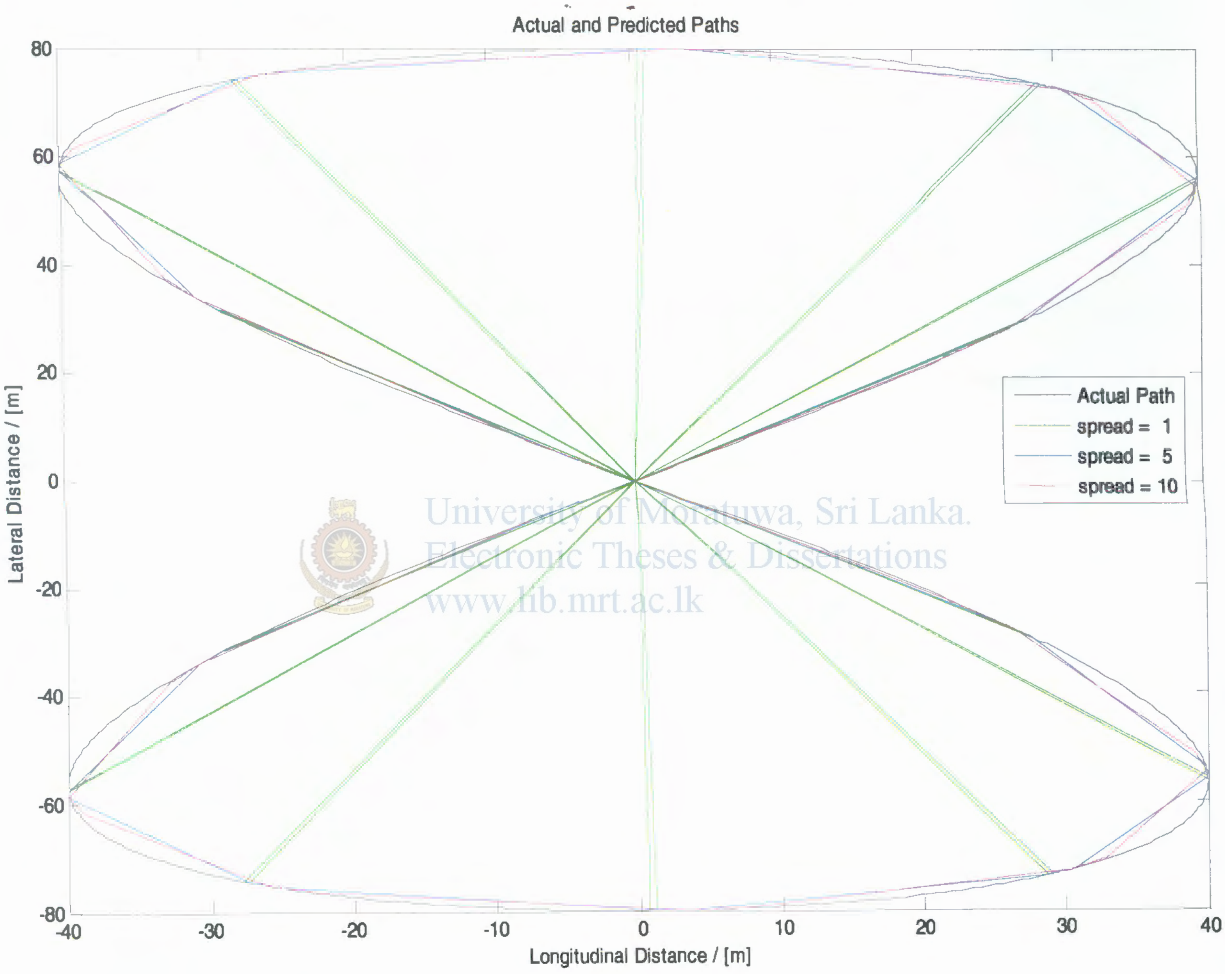


Figure 6.29 – Actual path and Predicted paths of the Obstacle for different Spreads

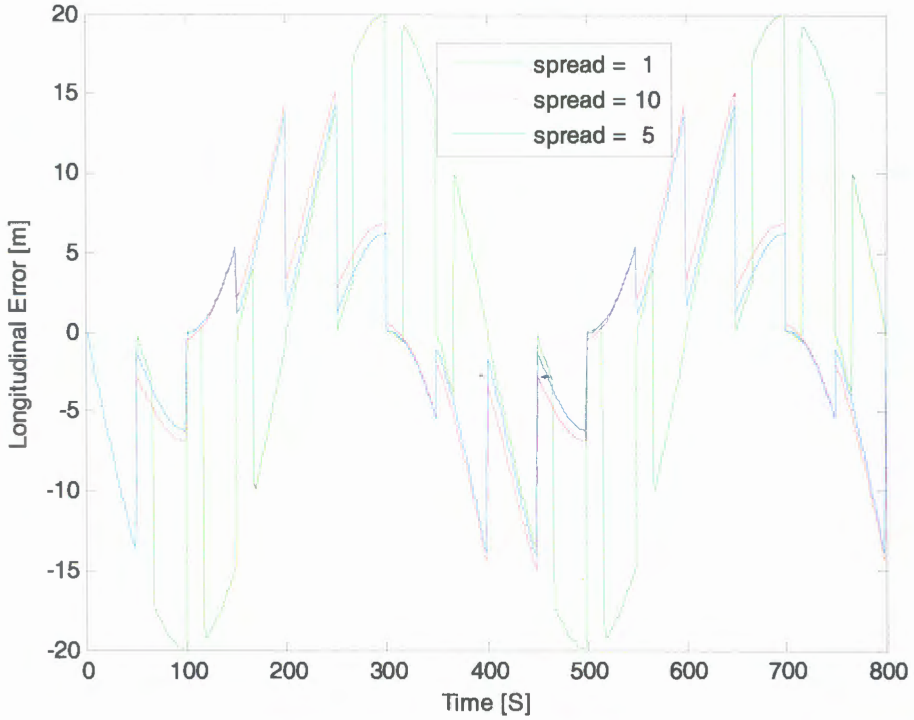


Figure 6.30 – Longitudinal error of predicted paths for different Spreads

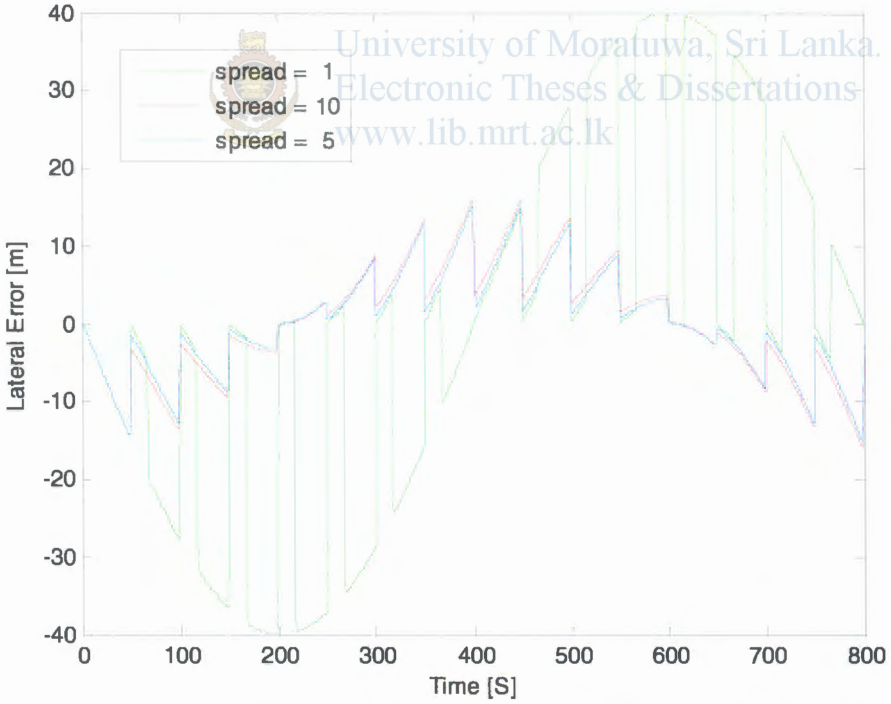


Figure 6.31 – Lateral error of predicted paths for different Spreads

Actual and predicted paths for different spreads are given in figure 6.29 to analyze the effect of spread value efficiently. The number of neuron enrollments can be increased by increasing the spread value. That predicted path can be smoothed more by increasing the spread value. Longitudinal and Lateral error changes for different spread values are presented in Figure 6.30 and 6.31 respectively.

6.5.1.2.2 Simulations with sensor noise

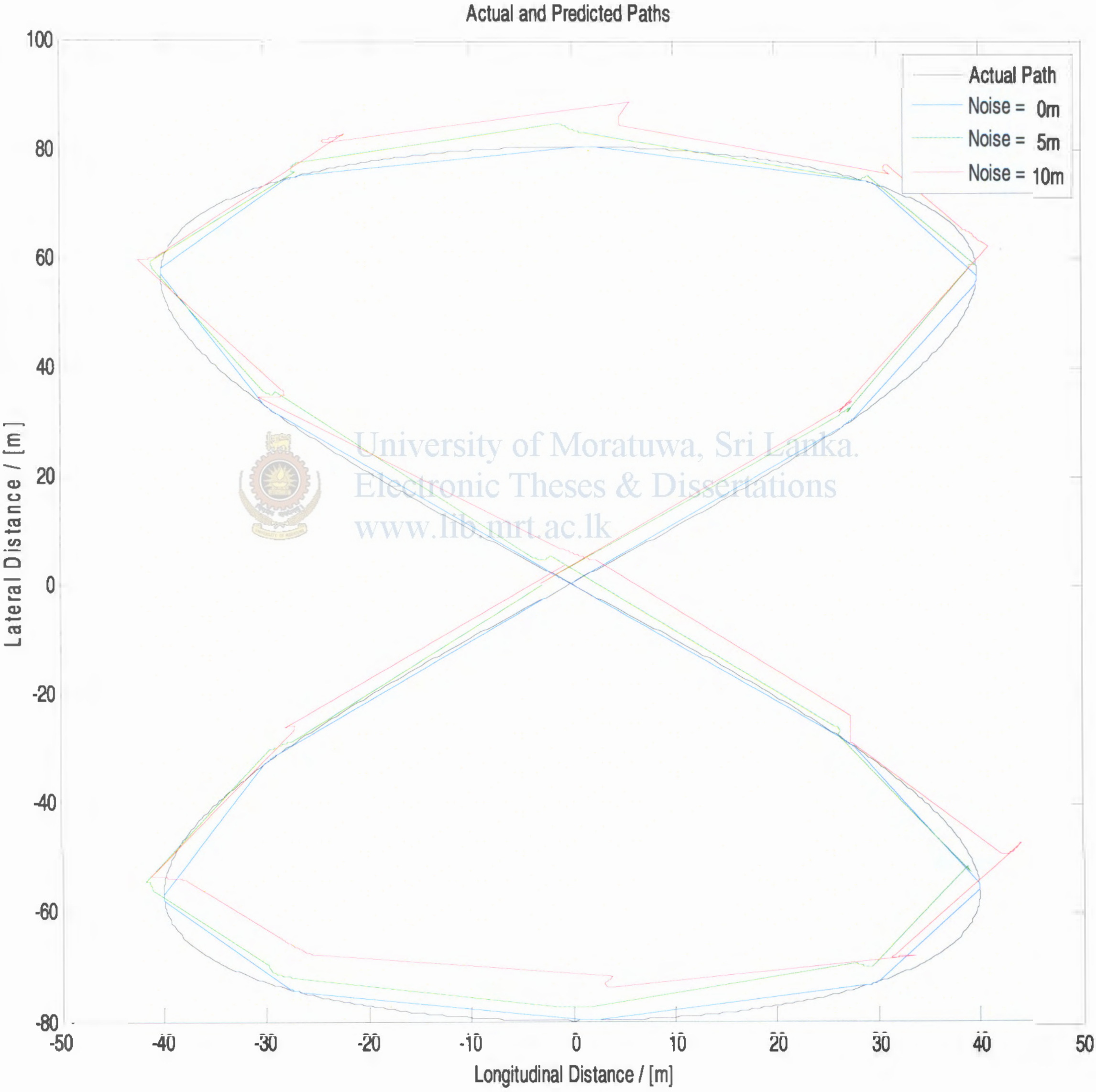


Figure 6.32 – Actual path and Predicted paths of the Obstacle for different noise values

The data without sensor noise can not be expected in the field. So that the noise is introduced to the actual path data and then predicted paths are taken for different noise values with 3m spread. Those results are given below in Figure 6.32. It proves the validity of the RBNN method for successful path prediction with noises. Longitudinal and lateral errors for different noise values are given in Figure 6.33 and 6.34 respectively. MatLab program for this is given in appendix F.

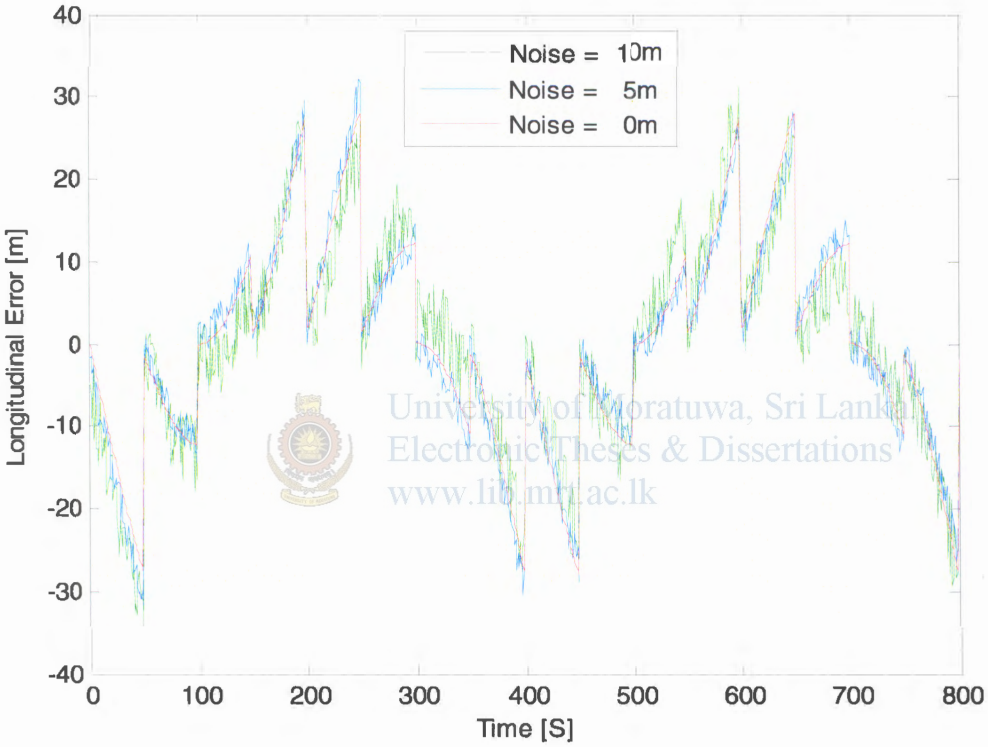


Figure 6.33 – Longitudinal error of predicted paths for different noise values

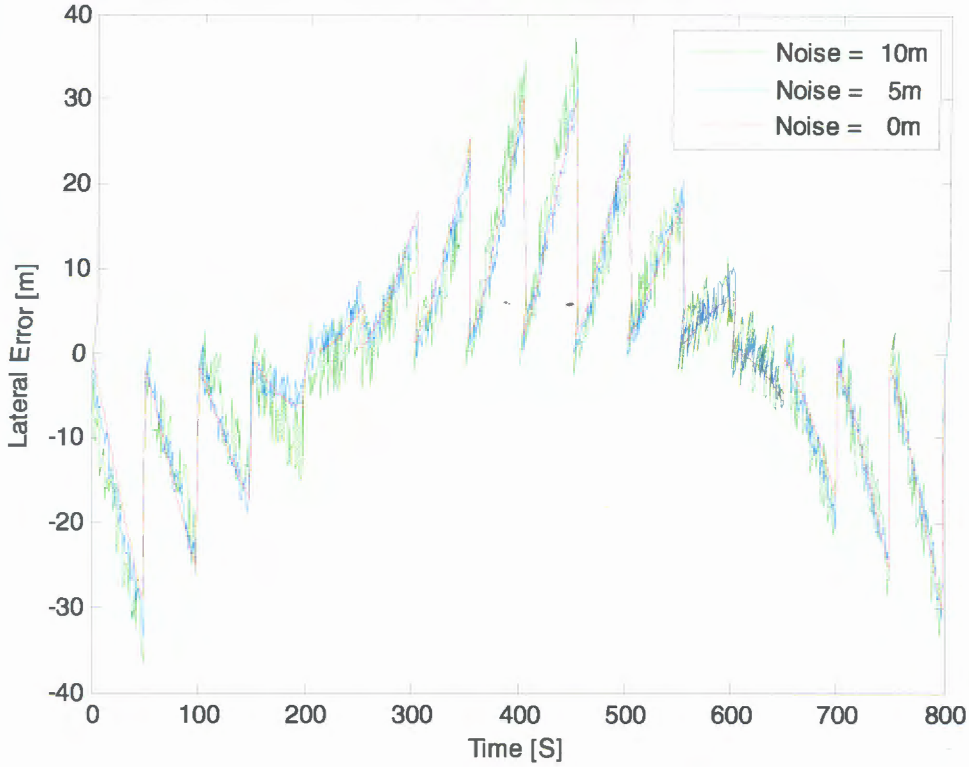


Figure 6.34 – Lateral error of predicted paths for different noise values

6.5.1.3 Summary of Simulation Results for path prediction

Path predictions of dynamic obstacles are very important for dynamic obstacle avoidance. Polynomial approximation method is tried for that initially. Its quite good method and gave very good results at the beginning, without noises. But position data can not be expected without noise from the field. So noise was introduced and polynomial approximation method is not able to give considerable results as expected. RBNN method which is famous for function approximation is tried after that. It shows very good results with noises. Those predicted paths can smoothen more by increasing the spread value.

6.5.2 Simulation Results for Obstacle area prediction of Dynamic Obstacles

6.5.2.1 Simulation Results from Velosity obstacle method

Velocity obstacle method is a conventional method which is employed for Dynamic obstacle area predictions. It predicts triangular areas for Dynamic obstacles. The size of that area is depends on a constant . Following figures present the simulation results from that conventional method. A novel obstacle area prediction method is simulated after that. The main objective of that proposed method is to predict the areas of dynamic obstacle movements effectively. So that novel method is compared with the conventional method at the end. Those simulation results are presented finally.

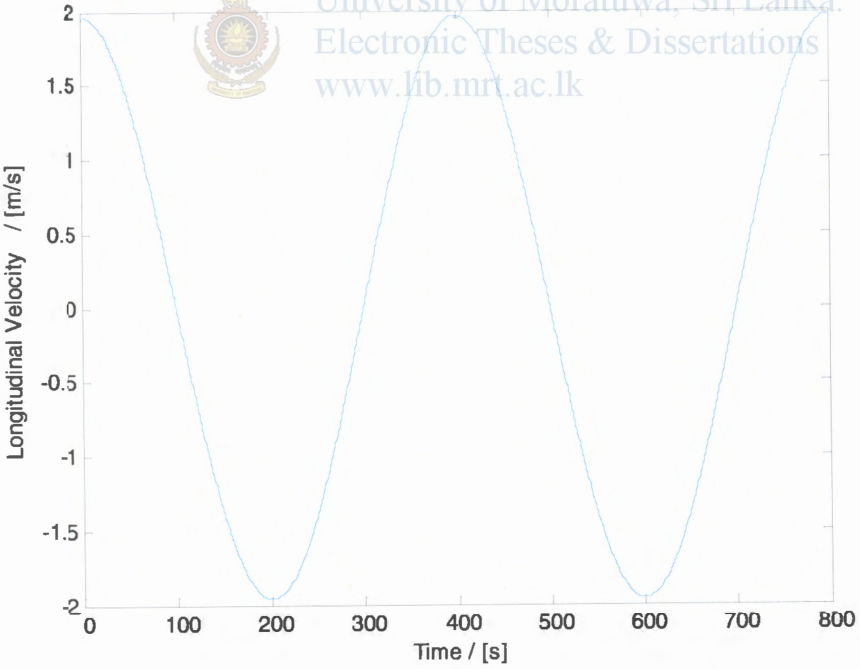


Figure 6.35 – Longitudinal velocity of the obstacle

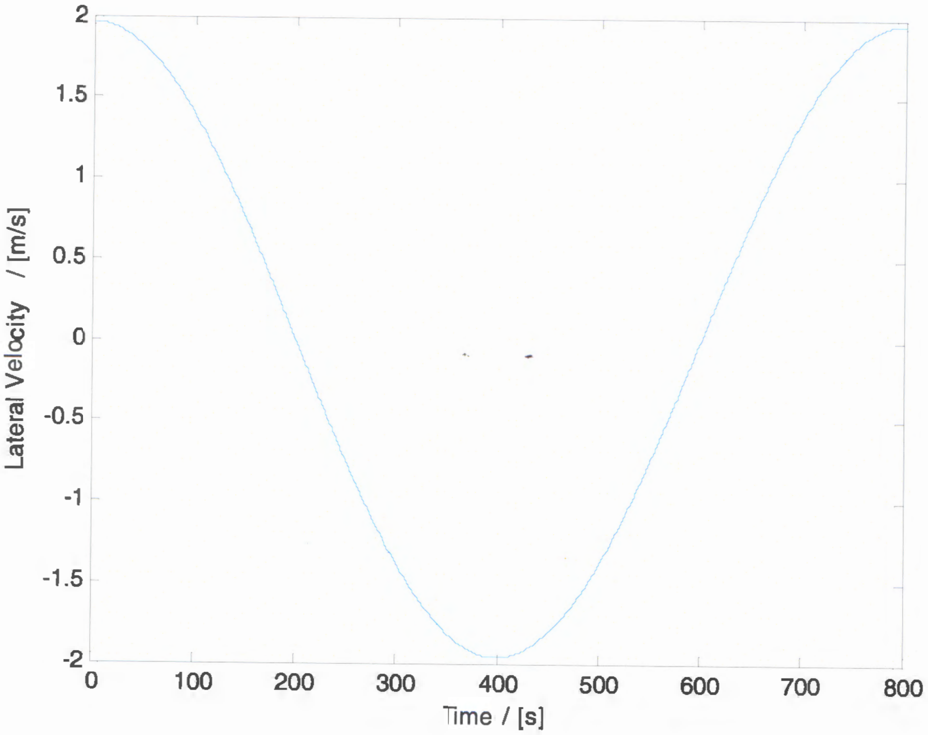


Figure 6.36 – Lateral velocity of the obstacle

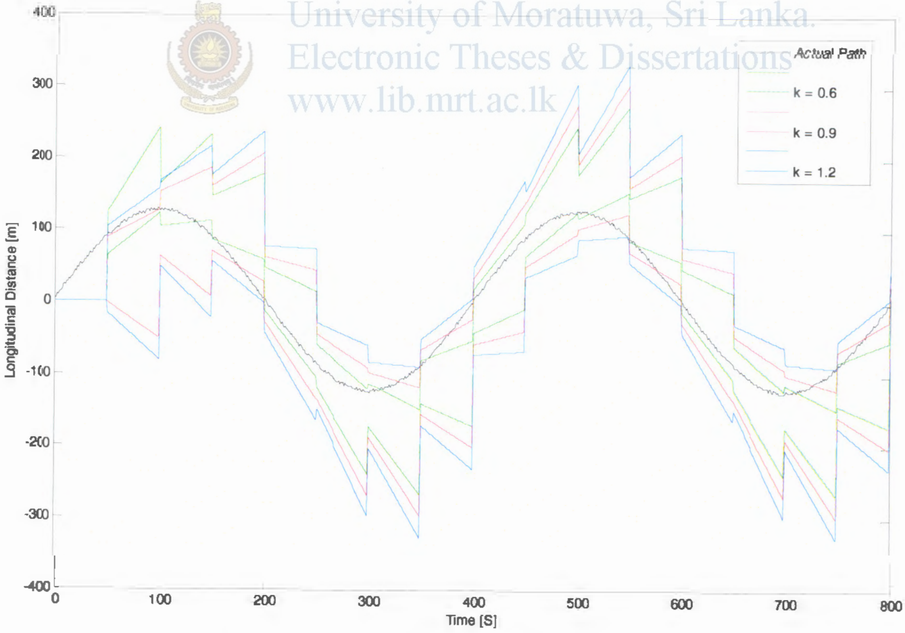


Figure 6.37 – Upper and Lower Boundaries of Predicted Obstacles Area towards Longitudinal direction

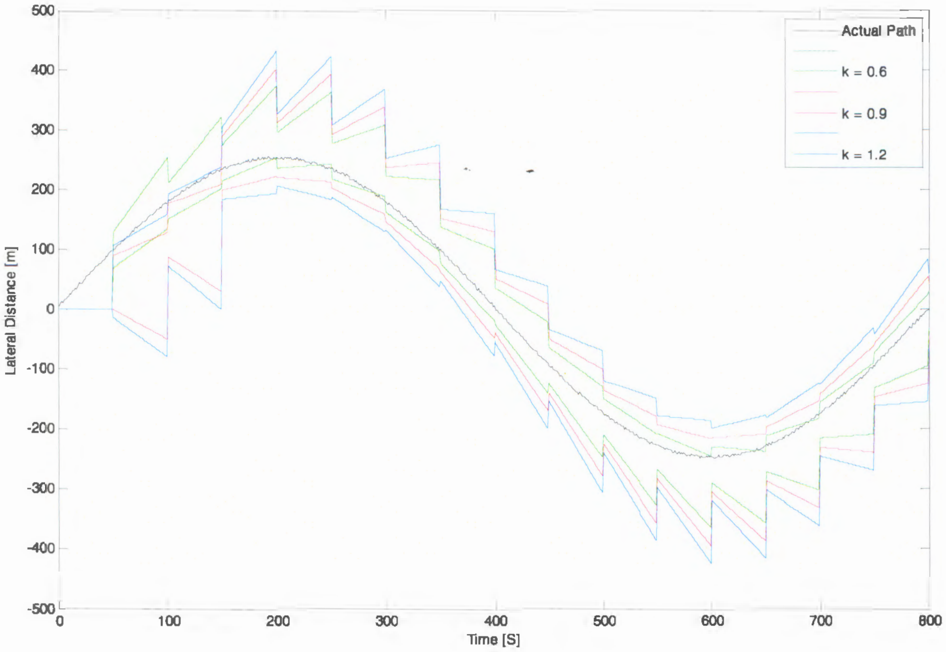


Figure 6.38 – Upper and Lower Boundaries of Predicted Obstacles Area towards Lateral direction

Figure 6.37 and 6.38 present upper and lower boundaries of predicted obstacle areas utilizing conventional method towards Longitudinal and Lateral directions. Those boundaries are generated for different constant values. This constant can be adjusted to take the efficient predicted area. The actual path may deviate out from the predicted obstacle areas. That mean those obstacles may collide with the USV. So those types of situations are calculated and plotted with time for Longitudinal and Lateral directions as shown in Figure 6.39 and 6.40.

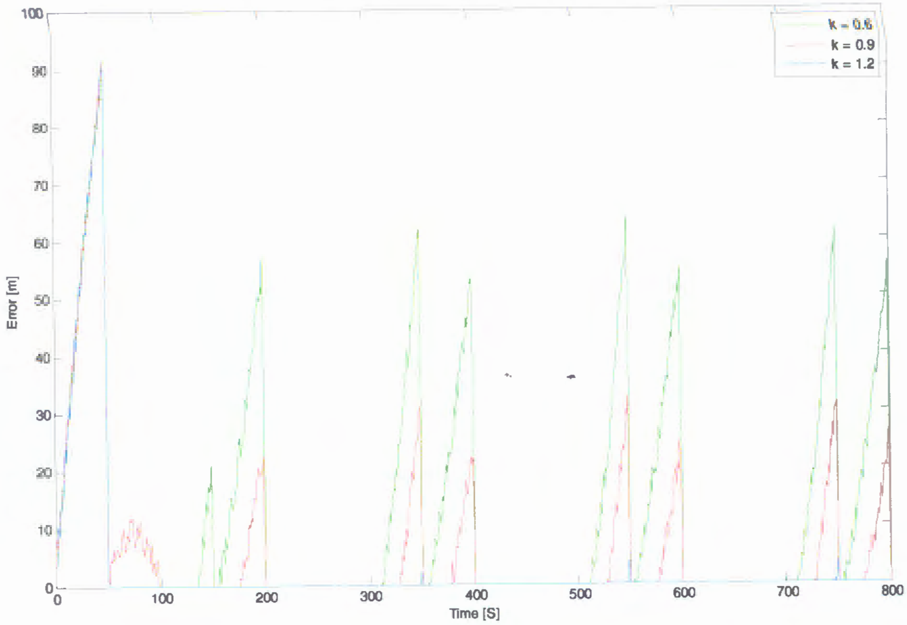


Figure 6.39 – Error towards Longitudinal direction

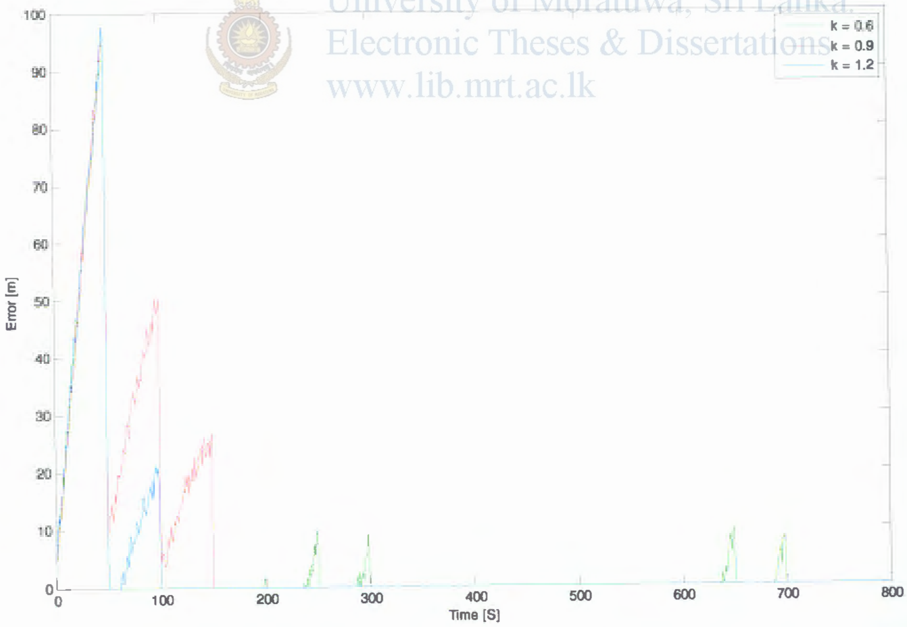


Figure 6.40 – Error towards Lateral direction

6.5.2.2 Simulation Results from the novel method

The predicted areas generated from the novel method are depending on the relative velocity of the obstacles as discussed in previous chapters. The relative velocity variation of the obstacle is presented in Figure 6.35 and 6.36. That variation is directly effecting the area prediction. So the upper and lower boundaries of predicted areas by means of novel method are presented in Figure 6.41 and 6.42. The error values of those are shown in figure 6.43 and 6.44. Then the Sea condition factor is changed and those simulation results are presented in Figure 6.45, 6.46, 6.47, 6.48 and 6.49.

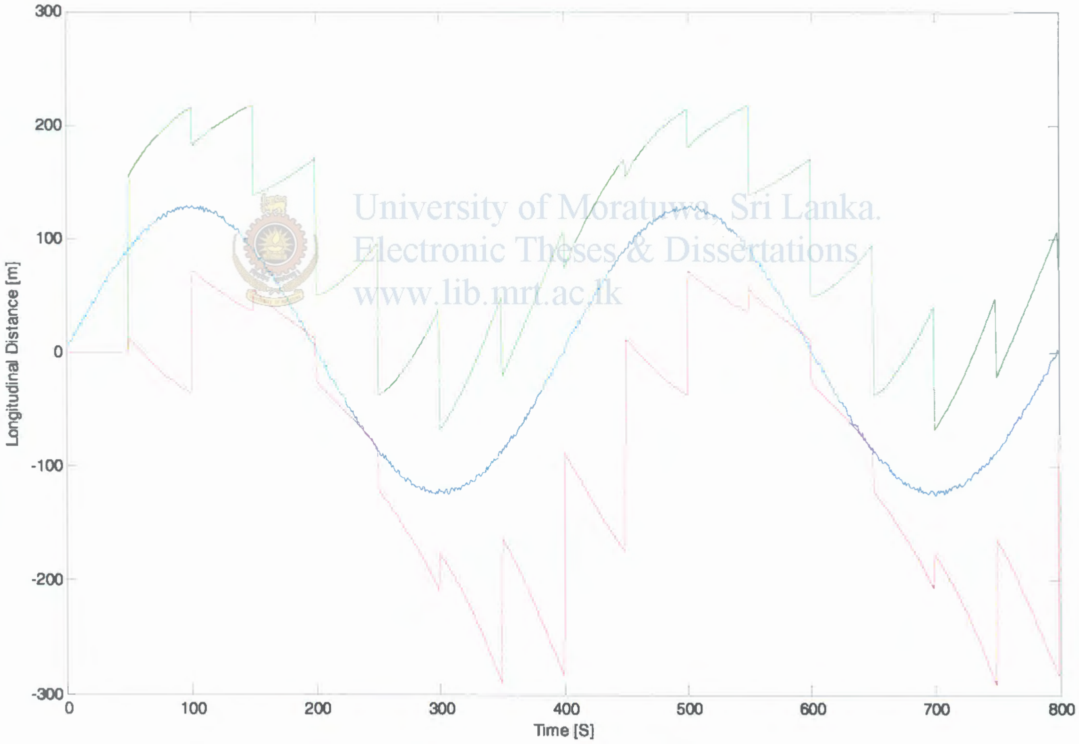


Figure 6.41 – Upper and Lower Boundaries of Predicted Obstacles Area towards Longitudinal direction (S = 0.5)

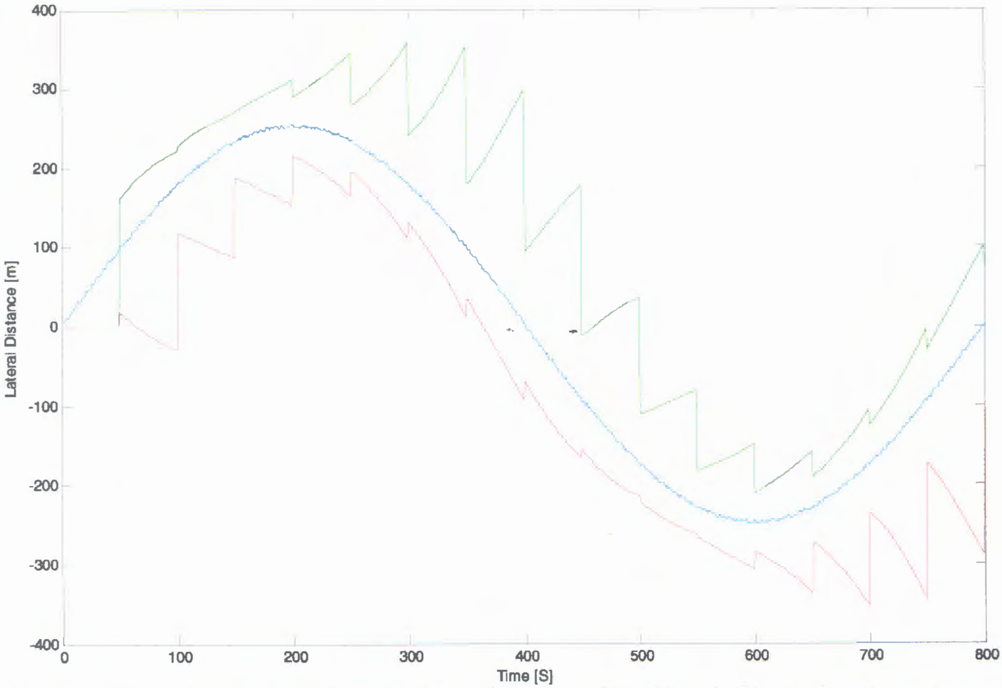


Figure 6.42 – Upper and Lower Boundaries of Predicted Obstacles Area towards Lateral direction (S = 0.5)

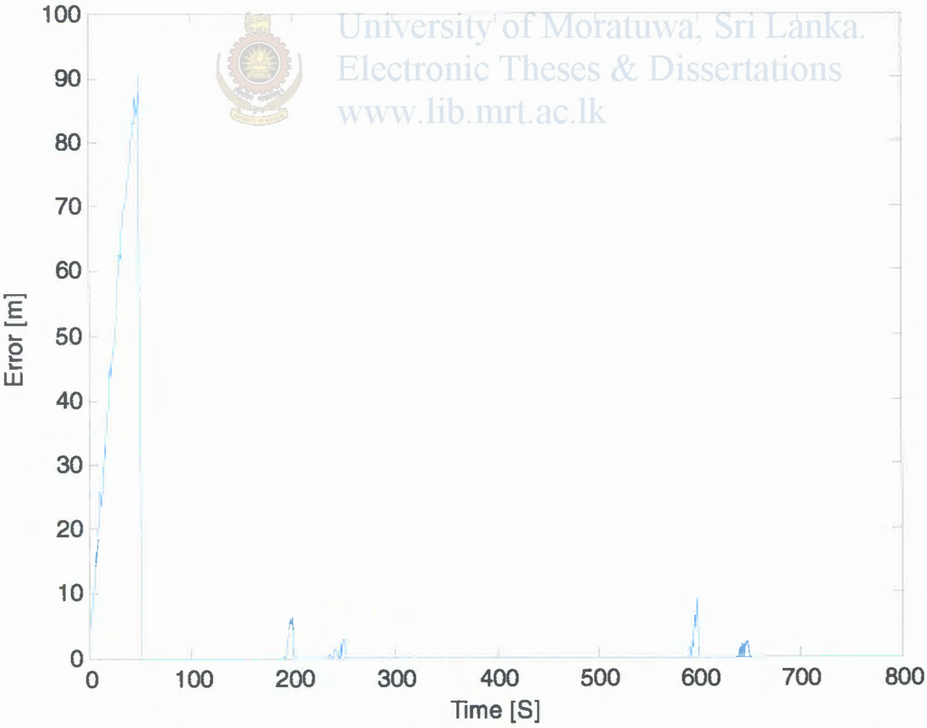


Figure 6.43 – Error towards Longitudinal direction (S = 0.5)

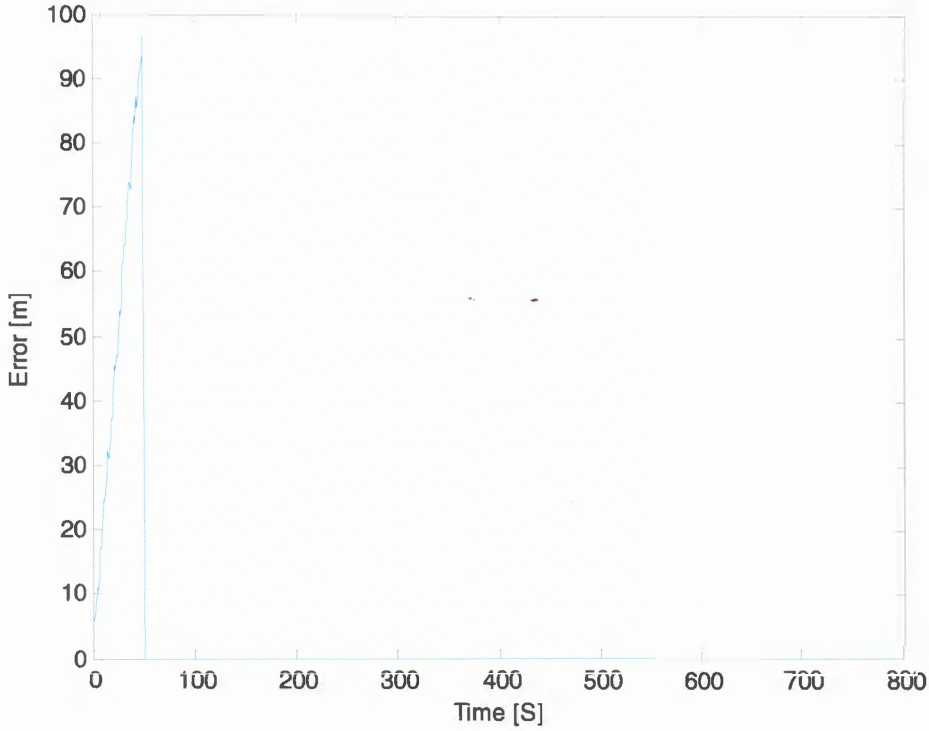


Figure 6.44 – Error towards Lateral direction (S = 0.5)

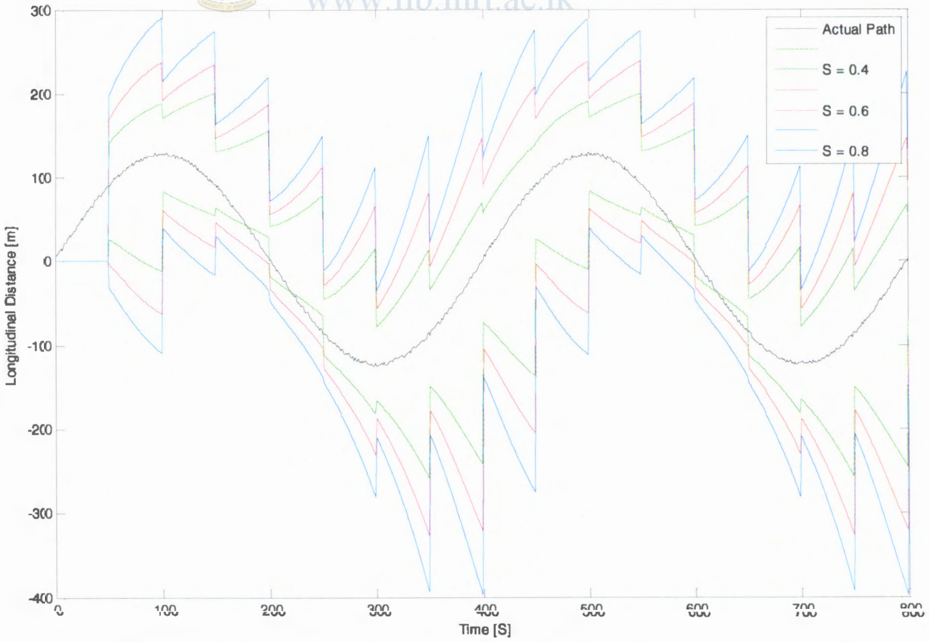


Figure 6.45 – Upper and Lower Boundaries of Predicted Obstacles Area towards Longitudinal direction for different Sea condition values

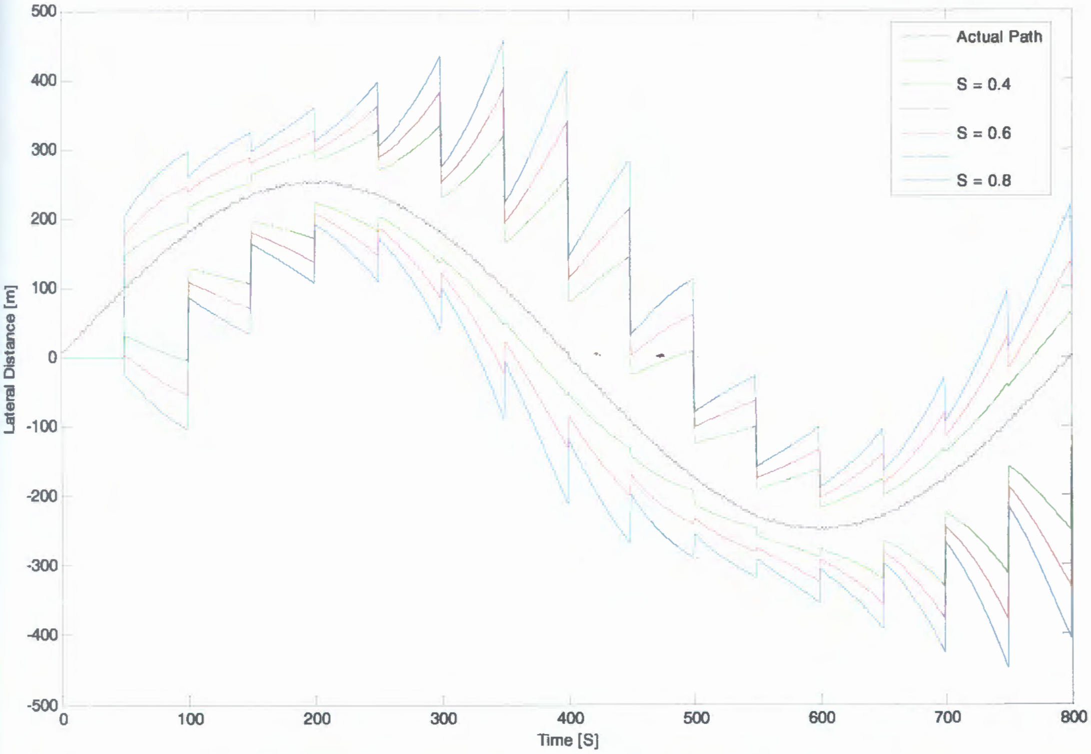


Figure 6.46 – Upper and Lower Boundaries of Predicted Obstacles Area towards Lateral direction for different Sea condition values

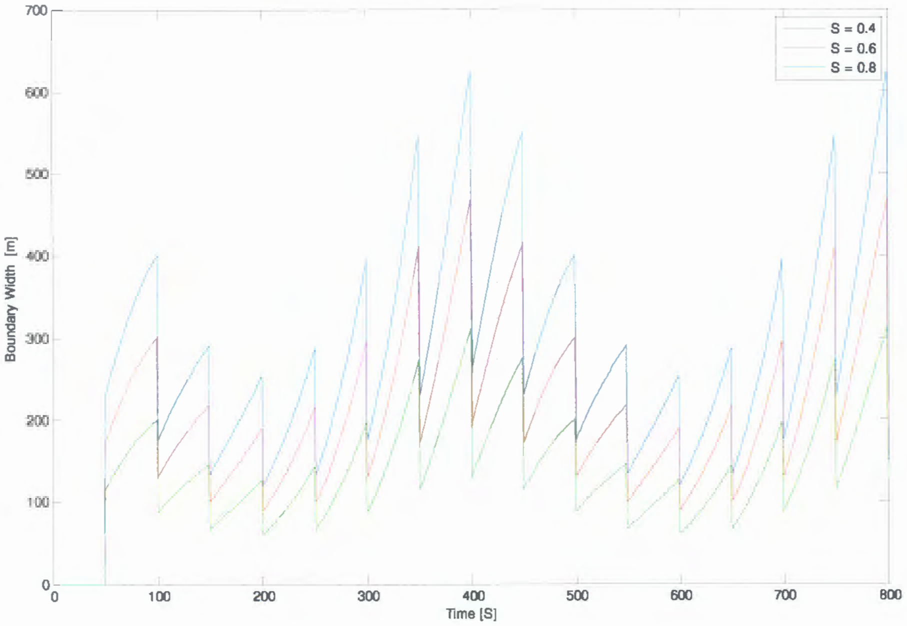


Figure 6.47 – Width of Predicted Obstacles Area towards Longitudinal direction for different Sea condition values

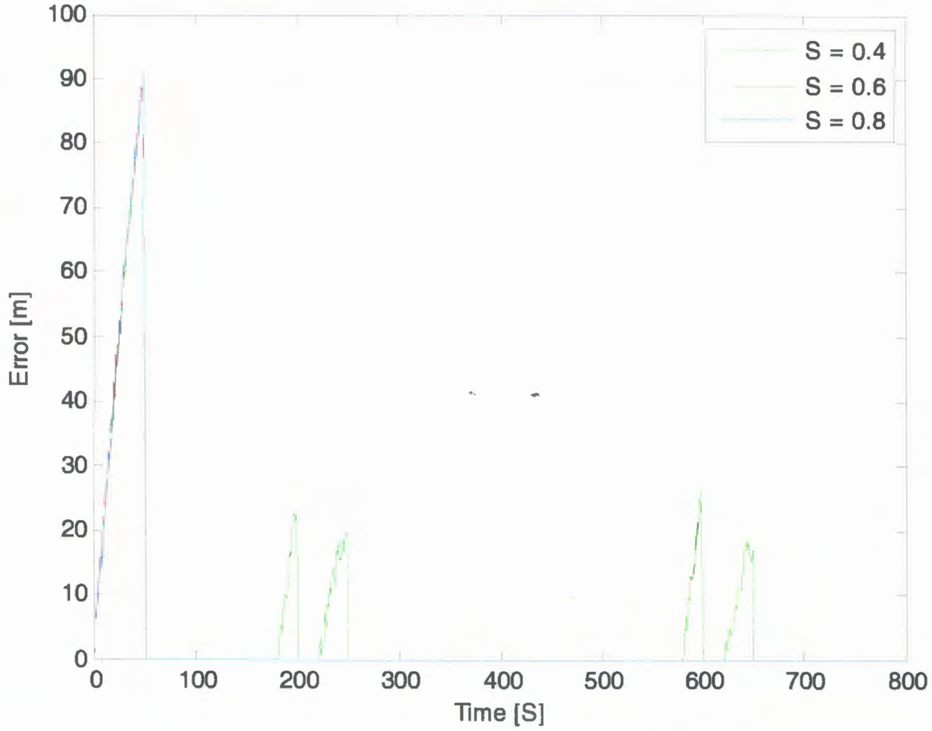


Figure 6.48 – Error towards Longitudinal direction for different Sea condition values

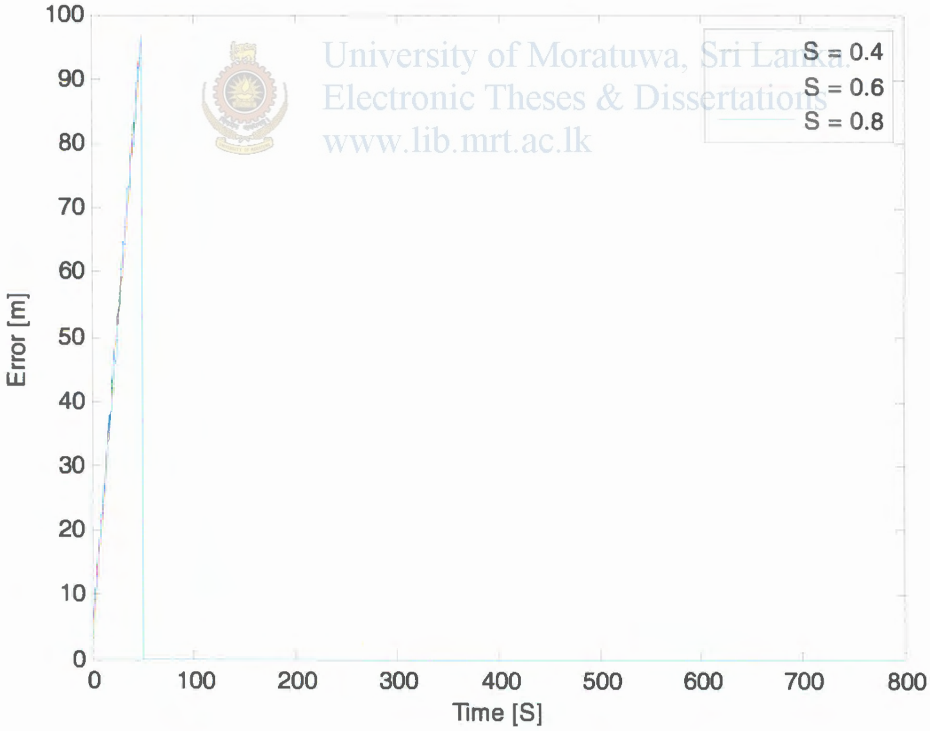


Figure 6.49 – Error towards Lateral direction for different Sea condition values

6.5.2.3 Comparison Results

The proposed novel algorithm for obstacle area prediction is compared with the conventional method to prove the effectiveness of the novel method. The upper and lower bounds of the predicted obstacle areas from both methods are shown in figure 6.50 and 6.51 and errors are presented in figure 6.52 and 6.53. Width of the predicted area from both methods towards longitudinal direction is given in figure 6.54. The area reduction from the novel method is inspired from the differences between the widths of the novel method and conventional method. That difference is given in figure 6.55 and that proves the improvements from the novel method. Matlab program which is developed for this is given with appendix G.

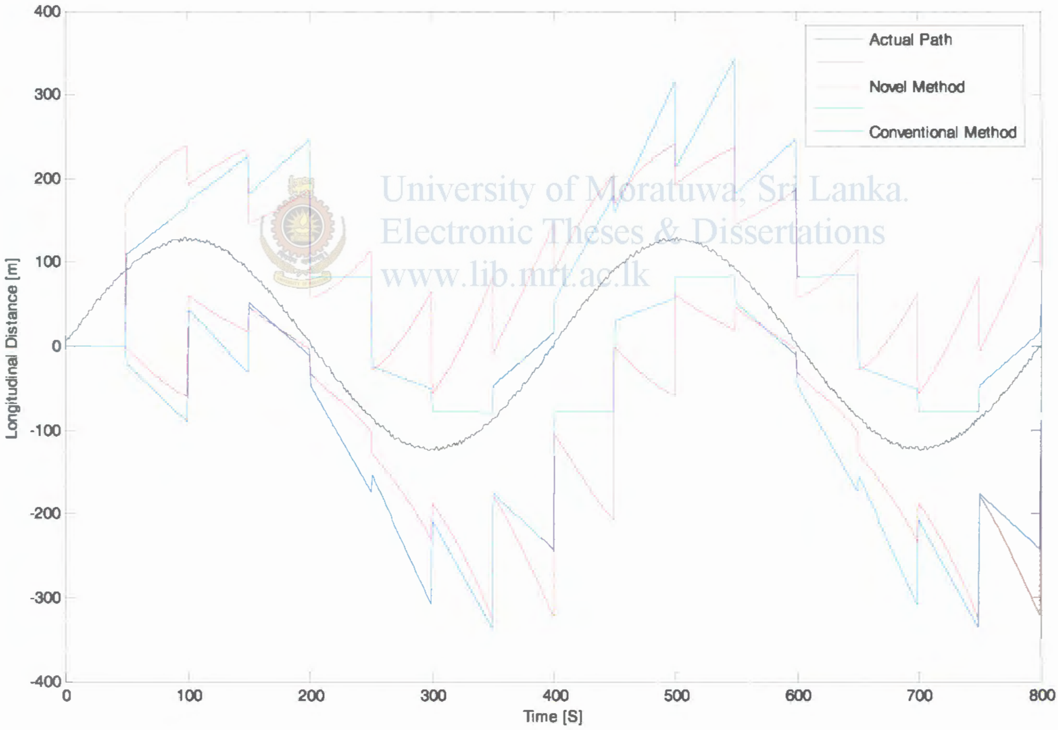


Figure 6.50 – Upper and Lower Boundaries of Predicted Obstacles Area towards Longitudinal direction from Conventional and Novel Methods

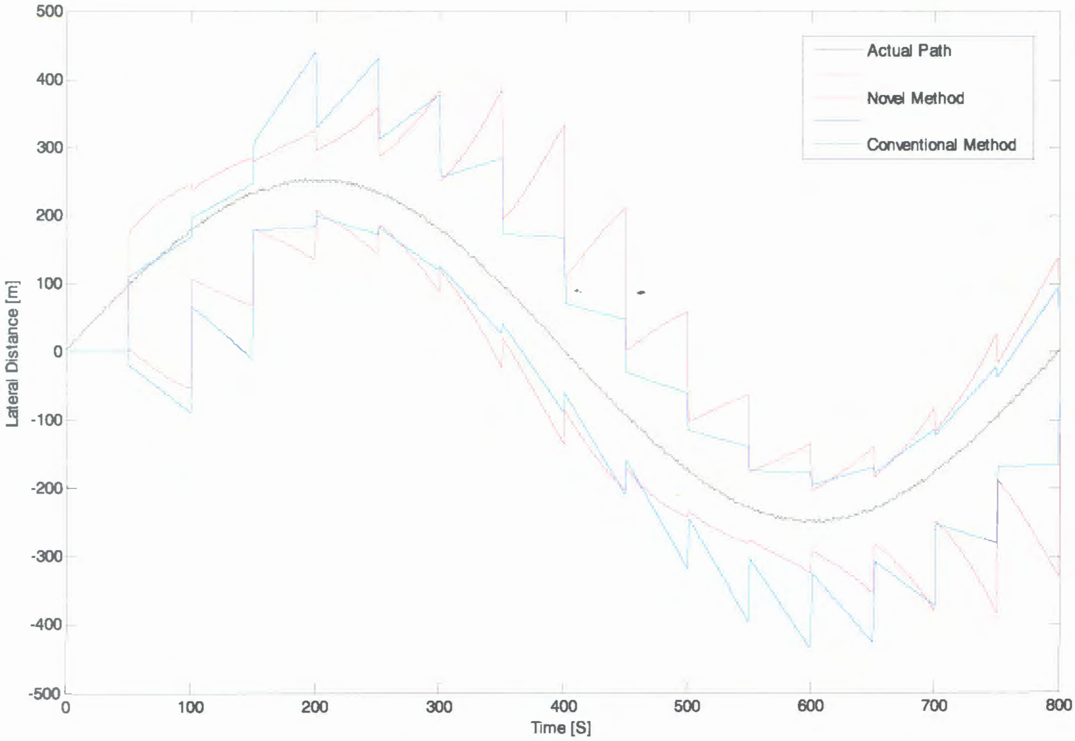


Figure 6.51 – Upper and Lower Boundaries of Predicted Obstacles Area towards Lateral direction from Conventional and Novel Methods

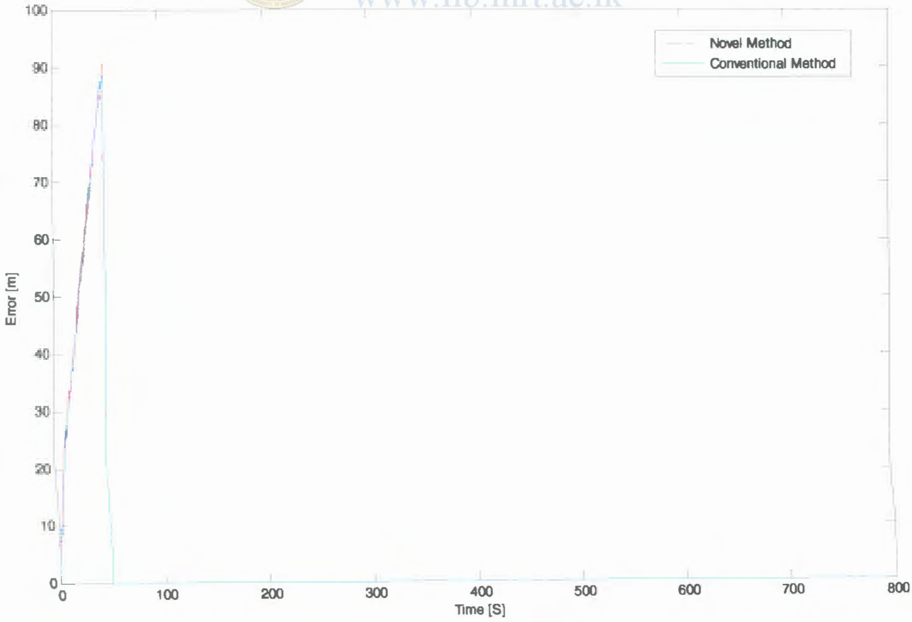


Figure 6.52 – Error towards Longitudinal direction for Conventional and Novel Methods

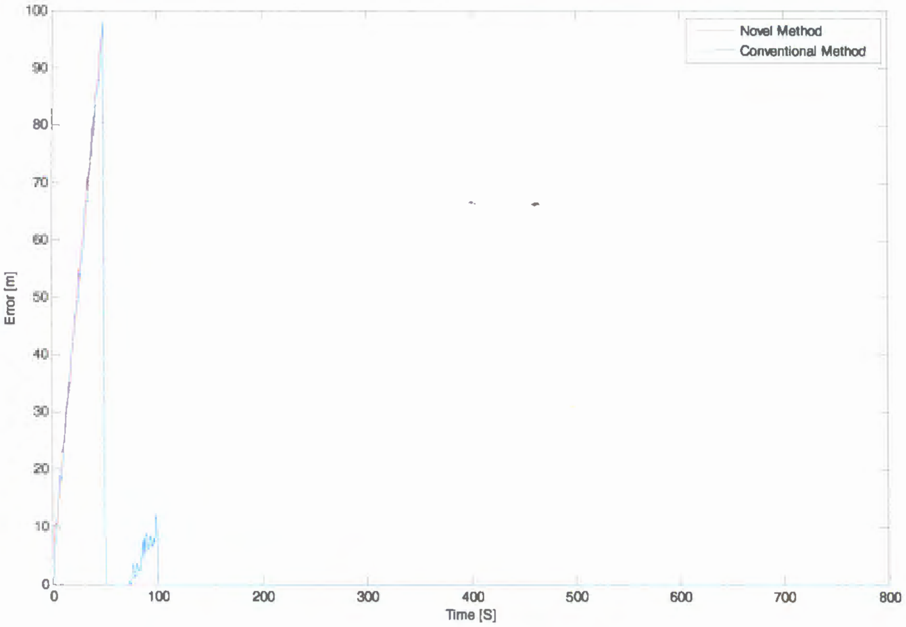


Figure 6.53 – Error towards Lateral direction for Conventional and Novel Methods

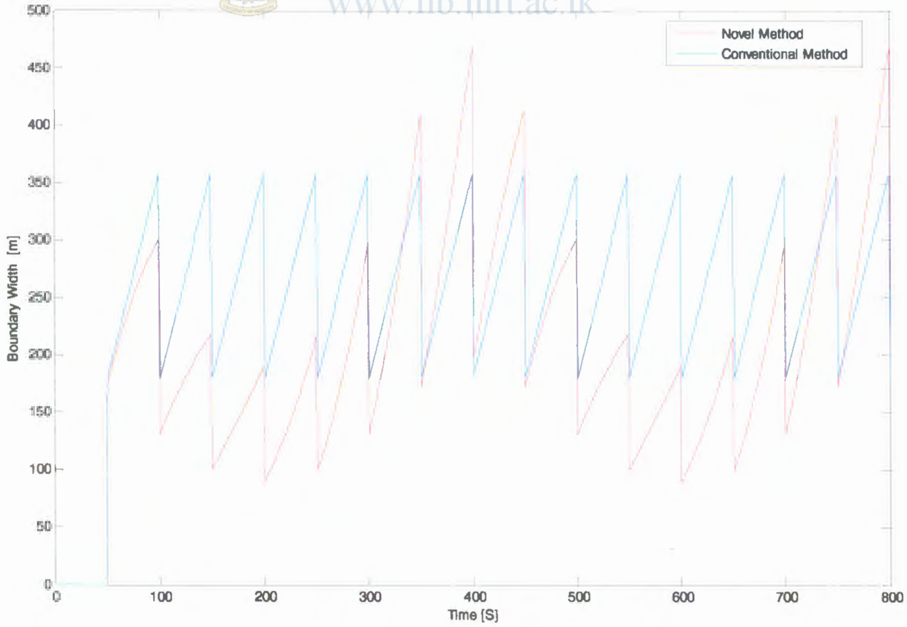


Figure 6.54 – Width of Predicted Obstacle Area towards Longitudinal direction from Conventional and Novel Methods

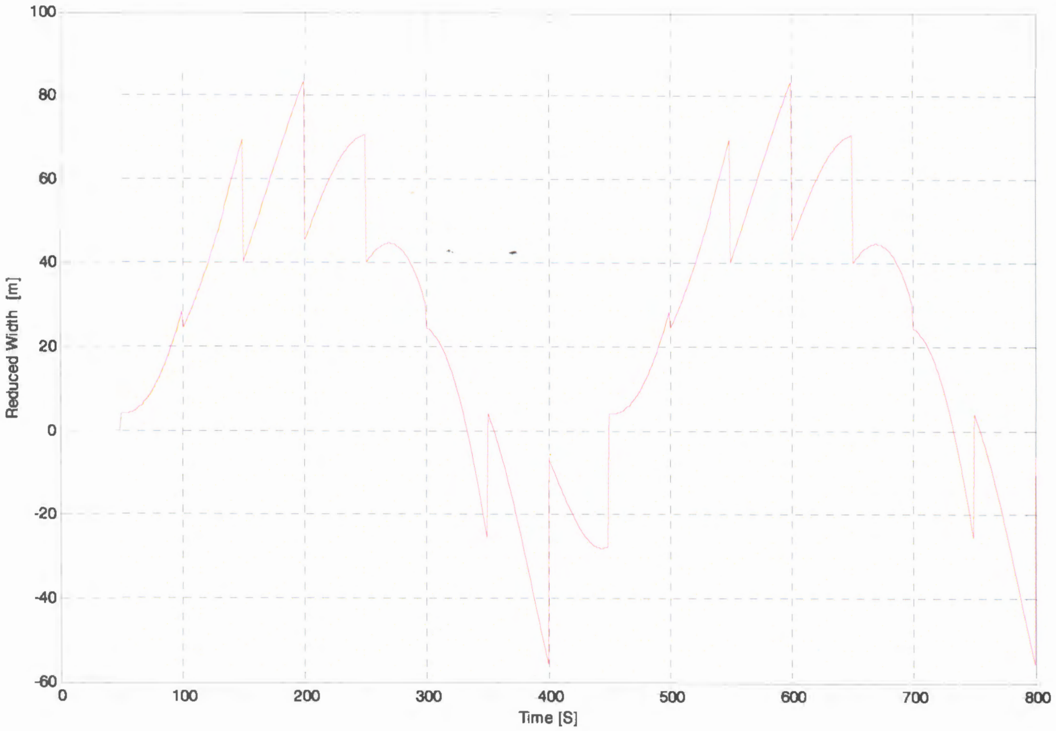


Figure 6.55 – Predicted Distance reduction from Novel Method

Above figure presents the effectiveness of employing Novel method for Dynamic obstacle avoidance well. But the Novel method is not reducing the predicted area totally. It may increase that area at sharp turns respect to the conventional method. It can be observed from the lower side of the graph given in Figure 6.55. But the overall performance of the proposed method is better than conventional one at the end.

Chapter 6

Conclusion and Recommendations

This work has presented and demonstrated some novel algorithms for static and dynamic obstacle avoidance. They have been compared with other famous obstacle avoidance methods as well.

Other research works have presented and demonstrated the capabilities of utilizing OA methods of UGVs for USVs. Implementing UGV methods practically which were developed and validated via simulations, was done as the initiative for the development. It is obviously true that practical experiment results would contradict little bit with the simulation results due to the immaturity of the available hardware. However those obstacle avoidance methods are transformed to achieve a better OA method for USVs.

A good Fuzzy based navigational controller for dynamic model of a USV was developed and it gives results as expected, needed to perform obstacle avoidance algorithms.

The fuzzy based obstacle avoidance controller is developed and legacy of that fuzzy controller is the algorithm which was developed for emergency collision avoidance of ground vehicles. That algorithm was practically implemented at the beginning of this research. That fuzzy-logic based system gives promising results in simulations. Even though it gives promising results for far away obstacles, the simulation results show that it is not much reliable for low speeds, noises and obstacles which appear suddenly.

Then another novel algorithm was introduced, which is good for low speeds as well as high speeds. That was inherited from the Morphin algorithm of Carnegie Mellon University. The exact details were not published and had to build novel algorithms for that.

The potential field method is used for obstacle avoidance as well since it dominates path planning of robots. Software is utilized for those simulations successfully.

According to the results which are presented in the figures it can be concluded that the novel algorithm is capable of achieving the target with minimum traveling distance. Further all of those algorithms work well and need to be customized more with the situation and application since each and every one is having its own pluses and minuses.

Obstacle avoidance without dynamic obstacles is not functional on in sea. Complete method for dynamic obstacle avoidance is yet to be solved in research field. But some practical dynamic obstacle avoidance methods are being employed today. Moving path prediction of a dynamic obstacle is the biggest problem to be solved by the researchers and two approaches are developed to solve that. RBNN and standard polynomial approximations are chosen as two approaches since RBNN are very famous for function prediction purposes. RBNN and Polynomial approximation methods are employed for path prediction and compared. It can be concluded that RBNN method is good for path prediction purposes because it can be utilized with high noise values as well. A smooth predicted path can be obtained by increasing the spread value of RBNN. A novel dynamic obstacle area prediction method is introduced and it is compared with the conventional velocity obstacle method. Simulation results prove the improvement of the novel method noticeably.

Three static obstacle avoidance methods and novel dynamic obstacle avoidance method which is inherited from projected obstacle area method presented and simulations done to prove the validity of those methods with sensor noise. The hardware of the USV has to be developed first. Then these algorithms can be employed with those sensors. These algorithms have to be fine tuned. The performance of the USV can improve by utilizing Cutting-edge technology for sensors.

This will lead to an eye opening for USV developers in the Sri Lanka and will be able to fill the gaps of research works on USVs in the world.

[1] Borenstein, J. and Koren, Y., "Real-time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments." The 1990 IEEE International Conference on Robotics and Automation, Cincinnati, Ohio, May 1990, pp. 572-577.

[2] Dawn J. Wright, 'The First Three-Dimensional Nautical Chart', Undersea with GIS, Chapter 7, ESRI. Inc., 2002.

[3] 'Digital Encorders', Agilent Technologies Inc., 5301 Stevens Creek Blvd, Santa Clara, CA 95051, USA, 2006.

[4] 'ER-400 Transceiver Modules', LPRS Ltd., Two Rivers Industrial Estate, Station Lane, Witney, Oxon, OX28 4BH, UK, 2007.

[5] Geeth Jayendra, Sisil Kumarawadu, Lakshan Piyasighe, "Fuzzy Logic-Based Navigator for an Unmanned Surface Vehicle" 14th ERU Symposium-2008, Colombo, Sri Lanka, October 2008.

[6] Geeth Jayendra, Sisil Kumarawadu, Lakshan Piyasighe, "Morphin based Obstacle Avoidance Controller for USV", 14th ERU Symposium-2008, Colombo, Sri Lanka, October 2008.

[7] Geeth Jayendra, Sisil Kumarawadu, Lakshan Piyasighe, "Obstacle Avoidance Controller for an Unmanned Surface Vehicle", 14th ERU Symposium-2008, Colombo, Sri Lanka, October 2008.

[8] Geeth Jayendra, Sisil Kumarawadu, Ravi Ranathunga and Samitha Ransara, "Intractive Intelligent Collision Avoidance Controller : Theory and Experiments", Proceedings of 2nd International Conference of Industrial and Information Systems, Peradeniya, Sri Lanka, August 2007.

[9] 'Generalized Regression Networks', Mathworks Inc. available : www.mathworks.com , accessed on December 2008.

[10] H. Lee and M. Tomizuka, "Coordinated longitudinal and lateral motion control of vehicles for IVHS", ASME Journal of Systems, Measurement, and Control, Vol. 123, 2001, pp.535-543.

[11] 'HS-422 Standard Deluxe Servomotor', Hitec RCD USA, Inc., 12115 Paine St, Poway, CA 92064, USA.

[12] I. Fantoni, R. Lozano, F. Mazenc, and K. Pettersen, "Stabilization of a nonlinear under-actuated hovercraft", International Journal of Robust and Nonlinear Control, vol. 10, 2000, pp. 645–654.

[13] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problem"., Proceedings of 28th Annual IEEE Symp. on Foundation of Computer Science, Los Angeles, CA 1987, pp. 49-60.

[14] J. Ebken, M. Bruch, J. Lum, "Applying unmanned ground vehicle technologies to unmanned surface vehicles", Proceedings of SPIE Unmanned Ground Vehicle Technology VII, vol. 5804, Orlando, FL, 2005, pp. 585-596.

[15] K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles", IEEE Transactions on Robotics and Automation, vol. 5, 1989, pp. 61-69.

[16] Kevin J. Walchko, Michael C. Nechyba, Eric Schwartz, Antonio Arroyo, "Embedded Low Cost Inertial Navigation System" University of Florida,Gainesville,FL,32611-6200.

[17] Kevin M. Passino, "Intelligent Control: An Overview of Techniques", Department of Electrical Engineering, The Ohio State University, 2015 Neil Avenue, Columbus, 2006.

[18] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." 1985 IEEE International Conference on Robotics and Automation, St. Louis, Missouri, March 1990, pp.500-505.

[19] KJC Kumara and Sisil Kumarawadu , "On the Speed control for Automated Surface Vessel Operation", Third International Conference on Information and Automation for Sustainability , Melbourne , Australia, December 2007.

[20] Lee F., and Krovi, V., "A Standardized Testing-Ground for Artificial Potential-Field based Motion Planning for Robot Collectives", Proceedings of the 2006 Performance Metrics for Intelligent Systems Workshop, Gaithersburg, MD, August 2006.

[21] Mark Maimone, Jeffrey Biesiadecki, Edward Tunstel, Yang Cheng, Chris Leger, "Intelligence for Space Robotics", NASA Jet Propulsion Laboratory, USA, 2006, pp. 53.

[22] N. Aggarwal and K. Fujimura, "Motion planning amidst planar moving obstacles", Proceedings of IEEE International Conference on Robotics and Automation, vol. 3, San Diego, CA, 1994, pp. 2153-2158.

[23] 'NavNet 3D Brochure', Furuno Electric Co., Ltd., Available: http://www.navnet.com/ , accessed on May 2008.

[24] 'Overview of Fuzzy Logic Toolbox- Matlab Help Files and Documentation', Mathworks Inc. available : www.mathworks.com , accessed on 05th Sep 2007.

[25] P. Fiorini and Z. Schiller, "Motion planning in dynamic environments using the relative velocity paradigm", Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, Atlanta, GA, 1993, pp. 560-565.

[26] 'Polynomial curve fitting', Mathworks Inc. available : www.mathworks.com, accessed on December 2008.

[27] 'Protector, Unmanned Surface Vehicle', Defense Update, International Online Defense Magazine, 2006. Available: http://www.defenseupdate.com/products/p/protector.htm, accessed on October 2008.

[28] 'Radar Problems', SciTech Publishing, Inc., Available: http://radarproblems.com/, accessed on May 2008.

[29] Reid Simmons, Lars Henriksen, Lonnie Chrisman and Greg Whelan, "Obstacle Avoidance and Safeguarding for a Lunar Rover", AIAA Forum on Advanced Developments in Space Robotics, Madison, WI, August 1996.

[30] Renju Thomas, Manoj Franklin, Chris Wilkerson and Jared Stark., "Improving branch prediction by dynamic data flow based identification of correlated branches from a large global history", Proceedings of the 30th International Symposium on Computer Architecture, San Diego, California, June 2003.

[31] Richard Bishhop, "Intelligent vehicle applications worldwide" , IEEE Transactions on Intelligent Transportation Systems, vol. 5, 2000, pp. 78-81.

[32] S. R. Ranatunga, "On Interactive Control For Intelligent Collision Evasive Emergency Intervention In Smart Vehicles", Master of Science Thesis Submitted to Dept. Electrical Engineering, University of Moratuwa, Sri Lanka, January 2007.

[33] S. R. Ranatunga and S.P. Kumarawadu , "Cooperatively Controlled Collision Evasive Emergency Manoeuvres", Control and Intelligent Systems, ACTA press, vol. 4, 2008 .

[34] Shingo Shimoda, Yoji Kuroda and Karl Iagnemma, "Potential Field Navigation of High Speed Unmanned Ground Vehicles on Uneven Terrain ", Proceedings of the 2005 IEEE, International Conference on Robotics and Automation, Barcelona, Spain, April 2005.

[35] Sisil Kumarawadu and Tsu-Tian Lee, 'Neuroadaptive Combined Lateral and longitudinal Control of Highway Vehicles Using RBF Networks', IEEE Transactions on Intelligent Transportation Systems, Vol. 17, No. 4, December 2006, pp. 500-512.

[36] Stephen W. Soliday, "Fuzzy Controller For Two Wheel Independent Drive Pivot Steering Vehicle", North Carolina Agricultural and Technical State University, Department of Electrical Engineering, 2006.

[37] Tannen S. VanZwieten, 'Dynamic simulation and control of an autonomous surface vehicle', Master of Science thesis submitted to the Florida Atlantic University, Florida, December 2003.

[38] 'The Protector Unmanned Surface Vehicle (USV) from on the Water', Gizmag, St Kilda South, Victoria, Australia, 2008. Available: http://www.gizmag.com/go/6023/, accessed on October 2008.

[39] Tilove, R. B., "Local Obstacle Avoidance for Mobile Robots Based on the Method of Artificial Potentials." General Motors Research Laboratories, Research Publication GMR-6650, September 1989.

[40] Tsoukalas, L. H. and Uhrig, R. E. 'Fuzzy and Neural Approaches in Engineering', John Wiley and Sons, New York, 1997.

[41] Y. Koren and J. Borenstein, 'Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation', Proceedings of the IEEE Conference on Robotics and Automation, Sacramento, California, April 1991, pp. 1398-1404.

Appendix A

OOPic Basic Program for Vehicular Prototypes

```
Dim Ver As New oByte
Dim SRF08 As New oI2C
Spk As oSpeaker
Dim Range1 As New oWord'To store sonar values
Dim Range2 As New oWord
Dim Range3 As New oWord
Dim Range4 As New oWord

Dim Compass As New oI2C'Create the compass objects
Dim Led As New oDIO1
Dim Bearing As New oByte
Dim Bearing1 As New oByte
Dim Bearing2 As New oByte

Dim A As New oSerialX 'Objects for serial communication
Dim B As New oSerialX
Dim C As New oDIO1
Dim D As New oDIO1

Dim ENC As New oQencode'Encorder objects
Dim ENCpositionN As New oWord
Dim ENCpositionO As New oWord
Dim ENCposition As New oWord
Dim EncPos1 As New oWord
Dim EncPos2 As New oWord
Dim count As New oWord

Dim E As New oDIO1'Motor control
Dim F As New oDIO1
Dim G As New oByte
Dim H As New oDIO1
Dim I As New oDIO1

Dim Y As oByte   'Position updates
Dim X As oByte

Dim TxCount As New oWord


Sub main()

  CommunicationSetup

  CompassSetup

  EncorderSetup

  SonarSetup

  MotorSetup

  count=0
```

```
   TxCount=0

 Do

     C.Invert

            EncorderValuesReading

            FormatEncPosition

            SonarValuesReading

      ObstalChecking                      -

      CompassValueReading

      If( (Bearing<5) Or (Bearing>250) ) Then
           'Spk.Beep(60757,100,200)
            End If

      FormatBearing

      C.Invert

     If G=1 Then

            MotorDrive

     EndIf

            DataTransmission

            TxCount=TxCount+1

            If TxCount=4 Then

                  TxCount=0
            EndIf

Loop
End Sub

        Sub CompassSetup
        Compass.Node=96'Decimal of Hex address 0xC0shifted right by 1
        Compass.Mode = cv10Bit' I2C mode is 10-Bit Addressing.
        Compass.NoInc = 1      ' Don't increment
        Led.IOLine = 30        ' Pin 28 on 40 way connector
        Led.Direction = cvOutput
        End Sub

        Sub CommunicationSetup
        C.IOLine = 5
        D.IOLine = 6
        C.Direction = cvOutput 'Starting communication
        D.Direction = cvOutput
        B.IOLineS = 25
        B.IOLineF = 18
        B.Baud = cv9600
        A.IOLineS = 26
        A.IOLineF = 16
        A.Baud = cv9600
```

```
End Sub

Sub EncorderSetup
ooPIC.PullUp = 1       ' For Encorder
ENC.IOLine1 = 28
ENC.IOLine2 = 29
ENC.Operate = cvTrue
End Sub

Sub SonarSetup

SRF08.Node = 112 'DecimalofHex address0xE0 shifted right by 1
SRF08.Mode = cv10Bit' I2C mode is 10-Bit Addressing.
SRF08.NoInc = 1  ' Don't increment
SRF08.Width = cv8Bit' 1 byte wide transfer
SRF08.Location = 0' Range Register
SRF08 = 81 ' Limit 1st sonar Range to 6m, 140 x 43mm = 6m

SRF08.Node = 113    'DecimalofHexaddress0xE2shiftedRight by 1
SRF08.Mode = cv10Bit' I2C mode is 10-Bit Addressing.
SRF08.NoInc = 1      ' Don't increment
SRF08.Width = cv8Bit' 1 byte wide transfer
SRF08.Location = 0  ' Range Register
SRF08 = 81          'Limit 2nd sonar Rangeto6m,140x 43mm = 6m

SRF08.Node = 114'Decimal of Hexaddress0xE4 shifted right by 1
SRF08.Mode = cv10Bit' I2C mode is 10-Bit Addressing.
SRF08.NoInc = 1      ' Don't increment
SRF08.Width = cv8Bit ' 1 byte wide transfer
SRF08.Location = 0  'Range Register
SRF08 = 81      'Limit 3rd sonar Range to 6m, 140 x 43mm = 6m

SRF08.Node=115'Decimal of Hex address 0xE6 shifted right by 1
SRF08.Mode = cv10Bit' I2C mode is 10-Bit Addressing.
SRF08.NoInc = 1      ' Don't increment
SRF08.Width = cv8Bit' 1 byte wide transfer
SRF08.Location = 0  ' Range Register
SRF08 = 81 ' Limit 4th sonar Range to 6m, 140 x 43mm = 6m

End Sub


Sub SonarValuesReading

SRF08.Node = 112  ' I2C Address of SRF08 sonar
SRF08.Location = 0' Command Register
SRF08.Width = cv8Bit' 1 byte wide transfer
SRF08 = 81   ' Ranging Command - Result in cm

Do       ' Wait for ranging to complete
  Ver = SRF08 ' This will wait forever if your sonar
Loop While Ver=255 ' becomes disconnected, so you may prefer

SRF08.Width = cv16Bit  ' 2 byte wide transfer
SRF08.Location = 2   ' 1st Range Register
Range1 = SRF08     ' Get Range to 1st object
'SRF08.Location = 3   ' 2nd Range Register
'Range2 = SRF08      ' Get Range to 2nd object

SRF08.Node = 113      ' I2C Address of SRF08 sonar
SRF08.Location = 0    ' Command Register
```

```
SRF08.Width = cv8Bit ' 1 byte wide transfer
SRF08 = 81            ' Ranging Command - Result in cm


Do          ' Wait for ranging to complete
  Ver = SRF08 ' This will wait forever if your sonar
Loop While Ver=255' becomes disconnected, so you may prefer

SRF08.Width = cv16Bit' 2 byte wide transfer
SRF08.Location = 2  ' 1st Range Register
Range2 = SRF08     ' Get Range

SRF08.Node = 114    ' I2C Address of SRF08 sonar
SRF08.Location = 0  ' Command Register
SRF08.Width = cv8Bit' 1 byte wide transfer
SRF08 = 81          ' Ranging Command - Result in cm

Do                  ' Wait for ranging to complete
  Ver = SRF08       ' This will wait forever if your sonar
Loop While Ver=255  ' becomes disconnected, so you may prefer

SRF08.Width = cv16Bit ' 2 byte wide transfer
SRF08.Location = 2    ' 1st Range Register
Range3 = SRF08        ' Get Ran

SRF08.Node = 115      ' I2C Address of SRF08 sonar
SRF08.Location = 0    ' Command Register
SRF08.Width = cv8Bit  ' 1 byte wide transfer
SRF08 = 81            ' Ranging Command - Result in cm

Do                  ' Wait for ranging to complete
  Ver = SRF08       ' This will wait forever if your sonar
Loop While Ver=255' becomes disconnected, so you may prefer

SRF08.Width = cv16Bit' 2 byte wide transfer
SRF08.Location = 2    ' 1st Range Register
Range4 = SRF08        ' Get Ran

End Sub

Sub CompassValueReading
Compass.Location = 1    ' Address of single byte bearing
Compass.Width = cv8Bit ' Compass Data is 1-byte wide.
Bearing = Compass.Value' Get it
End Sub

 Sub MotorSetup
 G=1
 E.IOLine = 14
 F.IOLine = 12
 H.IOLine = 13
 I.IOLine = 15
 E.Direction = cvOutput
 F.Direction = cvOutput
 H.Direction = cvOutput
 I.Direction = cvOutput
 End Sub

  Sub MotorDrive
  If B=100 Then'd
  D.Invert
 E.Low
 F.Low
 H.Low
 I.Low
   EndIf
   If B=103 Then'g
 E.High
 F.Low
 H.High
 I.Low
   EndIf
   If B=104 Then'h
   E.Low
 F.High
 H.Low
 I.High
 EndIf
 If B=101 Then'e
   E.Low
 F.High
 H.Low
 I.Low
 EndIf
 If B=102 Then'f
   E.Low
 F.Low
 H.Low
 I.High
 EndIf
 End Sub


   Sub ObstalChecking 'Check the Obstacles

   G=1

If Range1<17 Then
If Range1>0 Then
G=0
E.Low
F.Low
H.Low
I.Low
'Spk.Beep (60757,10, 200)
EndIf

EndIf
If Range2<17 Then
If Range2>0 Then
G=0
E.Low
F.Low
H.Low
I.Low
'Spk.Beep (60757,10, 200)
EndIf
EndIf
If Range3<17 Then
If Range3>0 Then
G=0
E.Low
```

```
F.Low
H.Low
I.Low
'Spk.Beep (60757,10, 200)
EndIf
EndIf
If Range4<17 Then
If Range4>0 Then
G=0
E.Low
F.Low
H.Low
I.Low
'Spk.Beep (60757,10, 200)
EndIf
EndIf

End Sub


Sub FormatBearing

If Bearing<200 Then
Bearing1=Bearing
Bearing2=0
End If

If Bearing>199 Then
Bearing1=199
Bearing2=Bearing-199
EndIf

End Sub


Sub EncorderValuesReading 'Reads the Enc. Value
ENCpositionN=ENC.Position
count=count+1

   If count=100 Then
ENCpositionO=ENCpositionN
ENCpositionN=ENC.Position
ENC.Position.Clear
count=0

PositionUpdate

EndIf
End Sub

Sub PositionUpdate

If (ENCpositionN>ENCpositionO) Then
ENCposition=ENCpositionN-ENCpositionO
X=100+ENCposition*Cos(Bearing)
Y=150+ENCposition*Sin(Bearing)

   EndIf

If (ENCpositionN<ENCpositionO) Then
ENCposition=ENCpositionO-ENCpositionN
```

```
     X=ENCposition*Cos(Bearing)
     Y=ENCposition*Sin(Bearing)
     EndIf

     End Sub

     Sub FormatEncPosition

       EncPos1=ENCpositionN
       EncPos2=0
       If ENCpositionN>249 Then
             EncPos1=250
             EncPos2=ENCpositionN-250
       EndIf
     End Sub

     Sub DataTransmission

     If TxCount=0 Then
            A.Value=251 'ú Sent Encorder values
      A.Value=X
    Else If TxCount=1 Then
      A.Value=252 'ú Sent Encorder values
            A.Value=Y
      Else If TxCount=2 Then
            A.Value=253'  sent Compass value
      A.Value=Bearing1
      Else If TxCount=3 Then
            A.Value=254'  sent Compass value
      A.Value=Bearing2
End If

    End Sub
```

```
'This program creates two oSerialX Objects. One is used to receive a
'serial signal and the other is used to send a serial signal.
'A oDio1 is used to show that while the oSerialX object is waiting
'for incoming serial data, the program flow is stopped.

Dim A As New oSerialX
Dim B As New oSerialX
Dim C As New oDIO1
Dim D As New oDIO1
Dim V As New Byte

Dim L As New oPWM
Dim R As New oPWM

Dim Compass As New oI2C          ' Create the compass objects
Dim Led As New oDIO1             '
Dim Bearing As New oByte

Sub Main()
  C.IOLine = 5
  D.IOLine = 6
  C.Direction = cvOutput
  D.Direction = cvOutput
  B.IOLineS = 25
  B.IOLineF = 18
  B.Baud = cv9600
  A.IOLineS = 26
  A.IOLineF = 16
  A.Baud = cv9600

 Compass.Node = 96              ' Decimal of Hex address 0xC0 shifted
right by 1
  Compass.Mode = cv10Bit         ' I2C mode is 10-Bit Addressing.
  Compass.NoInc = 1              ' Don't increment
  Led.IOLine = 30                ' Pin 28 on 40 way connector
  Led.Direction = cvOutput       '



      L.PreScale=2
      L.IOLine=17
      R.PreScale=2
      R.IOLine=18
      L.Period=79
      R.Period=79



  Do

     Compass.Location = 1         ' Address of single byte bearing
     Compass.Width = cv8Bit      ' Compass Data is 1-byte wide.
     Bearing = Compass.Value      ' Get it
    'A=B
    'A.Value=99'c
    A.Value=Bearing
    'A.String="B"
    C.Invert
    V=B
    'Cruising
    If V=100 Then'd
```

```
      L.Value=10
      R.Value=10
      L.Operate=1
      R.Operate=1
    EndIf



    'Left Turn
    If V=101 Then'e
      L.Value=14
      R.Value=10
      L.Operate=1
      R.Operate=1
    EndIf


    'Right Turn
    If V=102 Then'f
      L.Value=10
      R.Value=14
      L.Operate=1
      R.Operate=1
    EndIf



    'Reverse
    If V=103 Then'g
      L.Value=16
      R.Value=16
      L.Operate=1
      R.Operate=1
    EndIf


      ' Stop
    If V=104 Then'h
    D.Invert
      L.Value=10
      R.Value=10
      L.Operate=0
      R.Operate=0
    EndIf


  Loop
End Sub
```

```
Dim Ver As New oByte
Dim SRF08 As New oI2C
Spk As oSpeaker
Dim Range1 As New oWord
Dim Range2 As New oWord
Dim Range3 As New oWord
Dim Range4 As New oWord

Sub main()
   SRF08.Node = 112                 ' Decimal of Hex address 0xE0
shifted right by 1
        SRF08.Mode = cv10Bit             ' I2C mode is 10-Bit
Addressing.
        SRF08.NoInc = 1                  ' Don't increment
        SRF08.Width = cv8Bit             ' 1 byte wide transfer
        SRF08.Location = 0               ' Range Register
        SRF08 = 81                      ' Limit 1st sonar Range to 6m,
140 x 43mm = 6m


Do
    SRF08.Node = 112                 ' I2C Address of SRF08 sonar
        SRF08.Location = 0               ' Command Register
        SRF08.Width = cv8Bit             ' 1 byte wide transfer
        SRF08 = 81                       ' Ranging Command - Result in
cm

        Do                               ' Wait for ranging to complete
          Ver = SRF08                    ' This will wait forever if
                                                your sonar
        Loop While Ver=255               ' becomes disconnected, so you
                                                may prefer

        SRF08.Width = cv16Bit            ' 2 byte wide transfer
        SRF08.Location = 2               ' 1st Range Register
        Range1 = SRF08                   ' Get Range to 1st object
        'SRF08.Location = 3              ' 2nd Range Register
        'Range2 = SRF08

    If Range1<17 Then
    If Range1>0 Then
    Spk.Beep (60757,10, 200)
    EndIf
    EndIf

Loop


End
```

Appendix B

MatLab Program of Fuzzy PD Navigational Controller with the Dynamic model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fuzzy PD controller with Dynamic model                %%%%%%%%%%
%12/12/2007                                            %%%%%%%%%%
%control_Fuzzy_Navi.m                                  %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Xdot=control_Fuzzy_Navi(t,y)%program for speed control

n=[y(1) y(2) y(3)]';%Configuration vector=[x y ep]
V=[y(4) y(5) y(6)]';%Velocity vector=[u v r]
J=[cos(y(3)) -sin(y(3)) 0;sin(y(3)) cos(y(3)) 0;0 0 1;];
%rotation matrix
ndot=J*V;%velocities w.r.t. earth fixed frame

dhpy=-0.4;
dhsy=0.4;%distance from the CG to the Sboard & Port side hulls
Lh=2.54;%length of the hull
Bh=0.127;%beam
Th=0.1524;%draft
Ah=0.6452;%hull area

p=1025;%density of sea water
vis=1.4*10^(-6);%kinematic viscosity
g=9.81;

mp=(5*2)*2.2;        %mass of both pontoons [kg]
Ipz=(1/12)*mp*(Th^2+Bh^2);
%mass moment of inertia about mass-center z-axis [kg*m^2]
%misc. near pontoons
m1=11*2.2;
m2=(13*2)*2.2;
ms=3.6*2.2;          %mass of strut [kg]
mg=33*2.2;
%misc inside Gertler body
m3=4*2.2;
m4=(11+87)*2.2;
m5=25*2.2;
m6=19*2.2;

BAcc=1;   %Boat Max accelaration
test=405;
testarray=t;
%total
m=mp+ms+mg+m1+m2+m3+m4+m5+m6;           %vehicular mass
Iz=(3.94e5)*2.2*(0.0254^2);           %(approximate value from pro-E)
M=[m,0,0; 0,m,0; 0,0,Iz];    %mass matrix for the rigid body

C=[0 -m*y(6) 0;m*y(6) 0 0;0 0 0;];
%Coriolis & centripetal terms for rigid
                                   % body
Cr=C*V;%Coriolis terms

Re=(y(4)-dhpy*y(6))*Lh/vis; %Renolds Number
```

```
if Re>0 & Re<100    %at lamina to turbulant transistion point
    Cfhp=0.075/((log(Re)-2)^2);
%coefficient of friction using ITTC formular
elseif Re>100
   Cfhp=0.075/((log(Re)-2)^2);
else
   Cfhp=0; %coefficient of friction using ITTC formular
end

Fshpxur=Cfhp*Ah*p*(y(4)-dhpy*y(6))*abs(y(4)-dhpy*y(6));
%skin friction force for port side hull
Fshsxur=Cfhp*Ah*p*(y(4)-dhsy*y(6))*abs(y(4)-dhsy*y(6));
%skin friction force for starboard side hull

Fnpi= [.25 .3 .35 .4 .45 .5 .55 .6 .65 .7];
%to be used to find the wave drag (through interpolation)
Cwf= [1.2 1.75 2.25 2.58 3.75 4.25 4.15 3.85 3.58 3.4]*1e-3;
%The numbers were taken from fig 6 of journal of ship researech
%june2001pg 94
Fnp=(y(4)-dhpy*y(6))/(g*Lh)^0.5;

for i=1:9
    Fnp=(y(4)-dhpy*y(6))/(g*Lh)^0.5;
    if (Fnp>Fnpi(i) & Fnp<Fnpi(i+1))
        Cw=(Cwf(i)+Cwf(i+1))/2;
    else
    Cw=Cwf(i);
    end
end
Frhpxur=Cw*Ah*p*(y(4)-dhpy*y(6))*abs(y(4)-dhpy*y(6));
%wavw drag in Port hull

for i=1:9
    Fnp=(y(4)-dhsy*y(6))/(g*Lh)^0.5;
    if (Fnp>Fnpi(i) & Fnp<Fnpi(i+1))
        Cw=(Cwf(i)+Cwf(i+1))/2;
    else
    Cw=Cwf(i);
    end
end
Frhsxur=Cw*Ah*p*(y(4)-dhpy*y(6))*abs(y(4)-dhsy*y(6));
%%wavw drag in Sboard hull

XB=Lh/2;XS=Lh/2;
CDh=2;%drag coefficient
dx=-0.4;%distance from the CG to the hull centre in Xb direction
VB=y(5)+(Lh/2+dx)*y(6);%velocity @ bow end
VS=y(5)-(Lh/2-dx)*y(6);%velocity @ stern end

A1=1/3*abs(VB-VS)*(VB-VS)*(XB^2*abs(XB)-XS^2*abs(XS))/Lh^2;
A2=0.5*abs(VB-VS)*y(5)*(XB*abs(XB)-XS*abs(XS))/Lh;
A3=0.5*abs(VB-VS)*abs(y(5))*(XB^2-XS^2)/Lh;
A4=y(5)*abs(y(5))*(XB-XS);
Fhpyvr=0.5*CDh*Th*p*(A1+A2+A3+A4);%damping force
Fhsyvr=0.5*CDh*Th*p*(A1+A2+A3+A4);%damping force

mhxur=(Fshpxur+Frhpxur)*dhpy+(Fshsxur+Frhsxur)*dhsy;%damping moment

B1=1/4*abs(VB-VS)*(VB-VS)/Lh^2*(XB^3*abs(XB)-XS^3*abs(XS));
B2=1/3*abs(VB-VS)/Lh*((VB+VS)/2)*(XB^2*abs(XB)-XS^2*abs(XS));
```

```
B3=1/3*(VB-VS)/Lh*abs((VB+VS)/2)*(XB^3-XS^3);
B4=1/2*(VB+VS)/2*abs((VB+VS)/2)*(XB^2-XS^2);
mhpyvr=0.5*CDh*Th*p*(B1+B2+B3+B4);
mhsyvr=0.5*CDh*Th*p*(B1+B2+B3+B4);
mhuvr=mhxur+mhpyvr+mhsyvr;% total moment

alpa=pi/18;
CL=4.5837*alpa;
CD=0;



Sy=0.0948;%projected area
Ls=-1/2*p*abs(y(4)+y(5))*(y(4)+y(5))*Sy*CL;%lift force
Ds=1/2*p*abs(y(4)+y(5))*(y(4)+y(5))*Sy*CD;%Drag force

Fffsxur=Ls*sin(alpa)+Ds*cos(alpa);
bs=0.025;%beam
cs=0.075;
Sx=0.0395;%projected area
Vsx=y(4)-bs/2*y(6);
Fnsi= [0 0.1 .2  .3  .4  .5   .6   .7   0.8 1.1 2 3]';
Crsi=[0 0   .02 .15 .47 1.12 1.02 .73  0.55  0.23 0.21 0.2]';
for i=1:11
    Fns=(y(4)-dhsy*y(6))/(g*cs)^0.5;
    if (Fns>Fnsi(i) & Fns<Fnsi(i+1))
        Crs=(Crsi(i)+Crsi(i+1))/2;
    else
    Crs=Crsi(i);
    end
end
frsxur=1/2*p*Vsx*abs(Vsx)*Sx*Crs;

Fffsyur=Ls*cos(alpa)-Ds*sin(alpa);
Vsy=y(5)+cs/2*y(6);
Crsy=2;
frsyur=1/2*p*Vsy*abs(Vsy)*Sy*Crsy;
dsy=0;dsx=0.6;

msuvr=(Fffsxur+frsxur)*dsy+(Fffsyur+frsyur)*dsx;

Sg=2.1565;%sectional area of the gertler body
Lg=1.875;
Regi= [2 4 6 8 10 12 14 16 18 20 22 24 26 28 30]'*1e6;
%Reynolds number to match w/ Cfg vector (for interpolation)

Cgi= [3.9 3.4 3.2 3.05 2.9 2.85 2.75 2.7 2.65 2.6 2.55 2.52 2.51 2.49
2.46]'*1e-3;
%Taken for Gertler model
for i=1:14
    Reg=y(4)*Lg/vis;
    if (Reg>Regi(i) & Reg<Regi(i+1))
        Cdg=(Cgi(i)+Cgi(i+1))/2;
    else
    Cdg=Cgi(i);
    end
end

fgxu=1/2*p*y(4)*abs(y(4))*Sg*Cdg;
```

```
 for i=1:10
     dgx=[0.2006 0.2103 0.2184 0.2260 0.2332 0.2401 0.2467 ....
          0.2531 0.2592 0.2651];
     Vgy(i)=y(5)+dgx(i)*y(6);
     Cgy=[3.9 3.4 3.2 3.05 2.9 2.85 2.75 2.7 2.65 2.6 2.55 2.52 ....
         2.51 2.49 2.46]'*1e-3;
     Cg3d=[105  59   20.5 9.9 4.4 2.8 1.9 1.4 1.3 1.0 .95 1.0 ....
         1.1 1.0 0.2 .24 .54];
     Apg=[0.0373 0.0391 0.0406 0.0420 0.0434 0.0447 0.0459 ....
         0.0471 0.0482 0.0493];
     Fgyvrr(i)=0.5*p*Apg(i)*abs(Vgy(i))*Vgy(i)*Cgy(i)*Cg3d(i);
     mguvrr(i)=Fgyvrr(i)*dgx(i);

end
     Fgyvrr=[Fgyvrr(1) Fgyvrr(2) Fgyvrr(3) Fgyvrr(4) Fgyvrr(5) ....
         Fgyvrr(6) Fgyvrr(7) Fgyvrr(8) Fgyvrr(9) Fgyvrr(10)];
     mguvrr=[mguvrr(1) mguvrr(2) mguvrr(3) mguvrr(4) mguvrr(5) ....
         mguvrr(6) mguvrr(7) mguvrr(8) mguvrr(9) mguvrr(10)];
     Fgyvr=sum(Fgyvrr);
     mguvr1=sum(mguvrr);
     dgy=0;
     mguvr=fgxu*dgy+mguvr1;

DV=[(Fshpxur+Fshsxur+Frhpxur+Frhsxur+Fffsxur+Fffsxur+frsxur+fgxu);
     (Fhpyvr+Fhsyvr+Fffsyur+frsyur);
     (mhuvr+msuvr+mguvr);];

dpx=0.85;
hV1=[DV(1) DV(2) DV(3)]'+Cr;
hV=-[hV1(1) hV1(2) hV1(3)/dpx]';

%distance from CG to propellers in X-direction
M_new=[m 0 0;0 m 0;0 0 Iz/dpx;];%Mass matrix required

% Desired Profile
[x_d,xdot_d,xddot_d,y_d,ydot_d,yddot_d] = DesiredPath(t);
%..............................
%error
ex=x_d-y(1);
ey=y_d-y(2);
%error dot
exdot=xdot_d-ndot(1);
eydot=ydot_d-ndot(2);

                    a = readfis('Navi');
                    if ex>4
                        ex=4;
                    end
                    if exdot>4
                        exdot=4;
                    end
                    if ey>4
                        ey=4;
                    end
                    if eydot>4
                        eydot=4;
                    end

                    if ex<-4
                        ex=-4;
```

```
                 end
                 if exdot<-4
                     exdot=-4;
                 end
                 if ey<-4
                     ey=-4;
                 end
                 if eydot<-4
                     eydot=-4;
                 end
                 FisX=evalfis([ex exdot], a);
                 FisY=evalfis([ey eydot], a);

acc_x=xddot_d+((FisX*2-1)*10);
acc_y=yddot_d+((FisY*2-1)*10);

nddot_xy=[acc_x acc_y]'%;

J_xy=[cos(y(3)) -sin(y(3)); sin(y(3)) cos(y(3));];
Jdot_xy=[-sin(y(3)) -cos(y(3));cos(y(3)) -sin(y(3));];
V_xy=[y(4) y(5)]';
M_xy=[m 0;0 m;];
h_xy=[hV(1) hV(2)]';

Vdot_xy=inv(J_xy)*(nddot_xy-Jdot_xy*V_xy);
g_xy=1.25*M_xy*Vdot_xy;-1.5*h_xy;%with M_hat and h_hat or f_hat
gx=g_xy(1);
gye=g_xy(2);
gU=[gx gye gye]';

Vdot=inv(M_new)*(hV+gU);
%
Thrust=sqrt(gx^2+gye^2)%thrust
delta=atan(gye/gx)%steering angle

 Xdot=[ndot;Vdot;];


 %%%%%%%%%%%%%%%%%%%%%%EndOfProgramme%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Note:-

The above MatLab cording for Dynamic model is taken from the reference 26 and Fuzzy PD controller is added for simulations later.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot position error on sinosoidal path                 %%%%%%%%%%
%12/12/2007                                             %%%%%%%%%%
%plot_positions_error_pd_Sin.m                          %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

T=0:0.1:40;

X0=[0 0 0 0 0 0];
options = odeset('RelTol',1e-1,'AbsTol',[1e-1 1e-1 1e-1 1e-1 1e-1 1e-
1]);
[T Y]=ode45('control_Fuzzy_Navi',T,X0,options);


m=length(T);

[x_d,xdot_d,xddot_d,y_d,ydot_d,yddot_d] = DesiredPath(T);

for i=1:m

    %pause(0.1)

    figure(2)
    title('Error in Longitudinal direction')
    ylabel('Longitudinal error- [m]')
    xlabel('Time - [sec]')
    ex(i)=x_d(i)-Y(i,1);
    plot(i/10,ex(i),'--b');

    hold on

    figure(3)
    title('Error in Lateral direction')
    xlabel('Longitudinal error- [m]')
    ylabel('Time - [sec]')
    ey(i)=y_d(i)-Y(i,2);
    plot(i/10,ey(i),'--b');
    hold on

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%EndOfProgramme%%%%%%%%%%%%%%%%%%%%%
```

Appendix C

MatLab Program for Appling UGV algorithms to USV

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MatLab Program for Appling UGV algorithms to USV      %%%%%%%%%%
%12/12/2007                                            %%%%%%%%%%
%OAfuzzybase.m                                         %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
Ymax=2000;%Max Y of the grid
Xmax=2000;%Max X of the grid
timeInt=1;%Plot time intervals
Plottime=800;% Max ploting time

Xg=950;%Goal point cordinates
Yg=10;
Xs=30;%Starting point of the boat
Ys=915;
Xo=500;%Obstacle codinates
Yo=450;

Xact(1)=Xs;% actual position
Yact(1)=Ys;

oldX=Xs;
oldY=Ys;
BoH=0;

RH=0;%Heading towards goal
Gradiant=0;%tan(RH)
RelDis=1000;%Relative distance
index=1;
V=5;
z=0;
x(1)=Xs;
y(1)=Ys;
for t=1:timeInt:Plottime
time(index)=t;
if ((x(index)-Xg)^2+(y(index)-Yg)^2) > (V*timeInt*2+1)^2
%Objective function
index=index+1;      % Increments the indexing term so that
        % index=1 corresponds to time t=0.

%Calculating required heading
if (Xg>x(index-1))
if Yg>y(index-1)
Gradiant=(Yg-y(index-1))/(Xg-x(index-1));
RH=atan(Gradiant);
end
if y(index-1)>Yg
Gradiant=(y(index-1)-Yg)/(Xg-x(index-1));
RH=2*pi-atan(Gradiant);
end
end

if (x(index-1)>Xg)
if Yg>y(index-1)
Gradiant=(Yg-y(index-1))/(x(index-1)-Xg);
RH=pi-atan(Gradiant);
end
if y(index-1)>Yg
Gradiant=(y(index-1)-Yg)/(x(index-1)-Xg);
RH=pi+atan(Gradiant);
end
end
%Calculate obstacle heading
    if (Xo>x(index-1))
        if Yo>y(index-1)
            Gradiant=(Yo-y(index-1))/(Xo-x(index-1));
            ObH=atan(Gradiant);
        end
        if y(index-1)>Yo
            Gradiant=(y(index-1)-Yo)/(Xo-x(index-1));
            ObH=2*pi-atan(Gradiant);
        end
        if y(index-1)==Yo
            ObH=0;
        end
    end

    if (x(index-1)>Xo)
        if Yo>y(index-1)
            Gradiant=(Yo-y(index-1))/(x(index-1)-Xo);
            ObH=pi-atan(Gradiant);
        end

        if y(index-1)>Yo
            Gradiant=(y(index-1)-Yo)/(x(index-1)-Xo);
            ObH=pi+atan(Gradiant);
        end
        if Yo==y(index-1)
            ObH=pi;
        end
    end

    if Yo>y(index-1)
        if Xo==x(index-1)
            ObH=pi/2;
        end
    end
    if Yo<y(index-1)
        if Xo==x(index-1)
            ObH=pi/2*3;
        end
    end

    % Calculate boat heading
    if (x(index-1)>oldX)
        if y(index-1)>oldY
            Gradiant=(y(index-1)-oldY)/(x(index-1)-oldX);
            BoH=atan(Gradiant);
        end

        if oldY>y(index-1)
            Gradiant=(oldY-y(index-1))/(x(index-1)-oldX);
            BoH=2*pi-atan(Gradiant);
        end

        if oldY==y(index-1)
            BoH=0;
        end
    end

    if (oldX>x(index-1))
        if y(index-1)>oldY
            Gradiant=(y(index-1)-oldY)/(oldX-x(index-1));
            BoH=pi-atan(Gradiant);
        end

        if oldY>y(index-1)
            Gradiant=(oldY-y(index-1))/(oldX-x(index-1));
            BoH=pi+atan(Gradiant);
        end
        if oldY==y(index-1)
            BoH=pi;
        end
    end

    if y(index-1)>oldY
        if oldX==x(index-1)
           BoH=pi/2;
       end
   end
   if y(index-1)<oldY
       if oldX==x(index-1)
           BoH=pi/2*3;
       end
   end
   oldY=y(index-1);
   oldX=x(index-1);

        if ((pi/2)+ BoH-ObH)>=0
        Obstacle_angle=(pi/2)+ BoH-ObH;
        elseif ((pi/2)+ BoH-ObH)>2*pi
        Obstacle_angle=(pi/2)+ BoH-ObH-2*pi;
        elseif ((pi/2)+ BoH-ObH)>4*pi
        Obstacle_angle=(pi/2)+ BoH-ObH-4*pi;
        elseif ((pi/2)+ BoH-ObH)<0
         Obstacle_angle=(pi/2)+ BoH-ObH+2*pi;
         elseif ((pi/2)+ BoH-ObH)<-2*pi
         Obstacle_angle=(pi/2)+ BoH-ObH+4*pi;
         end

%Calculating the path codinates

RelDis=sqrt((Yo-y(index-1))^2 +(Xo-x(index-1))^2);
% Calculating relative distance to Obsatacle
if RelDis>100
x(index)=x(index-1)+V*timeInt*cos(RH);
y(index)=y(index-1)+V*timeInt*sin(RH);
%timeInt=10;
else
b = readfis('CAnew');
acc_CA=evalfis([Obstacle_angle (RelDis)/100], b);
acc_x=(acc_CA(1)-0.5)*2*cos(BoH)-(acc_CA(2)-0.5)*2*sin(BoH);
acc_y=(acc_CA(2)-0.5)*2*cos(BoH)+(acc_CA(1)-0.5)*2*sin(BoH);
```

```
x(index)=x(index-1)+acc_x*timeInt*cos(RH)*V;
y(index)=y(index-1)+acc_y*timeInt*sin(RH)*V; %
end
end
end
hold on
figure(2)
title('Path near obstacles')
xlabel('X distance[m]')
ylabel('Y distance[m]')
clf
plot(x,y,'rs','LineWidth',0.5,...
    'MarkerEdgeColor','g',...
    'MarkerFaceColor','g',...
    'MarkerSize',1)
Hold on

plot(Xact,Yact,'rs','LineWidth',0.5,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor','b',...
    'MarkerSize',1)
Hold on

plot(Xo,Yo,'rs','LineWidth',1,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor','b',...
    'MarkerSize',5)

title('Path near obstacles')
xlabel('X distance[m]')
ylabel('Y distance[m]')

 %%%%%%%%%%%%%%%%%%%%%EndOfProgramme%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix D

MatLab Program for Novel Algorithm to Avoid Static Obstacles

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Module for Plot Novel obstacle avoidance algorithm     %%%%%%%%%%
%12/12/2007                                             %%%%%%%%%%
%OAMainProForOA.m                                       %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xnew,ynew] = MorphinOA(xcur,ycur,Gx,Gy,O)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

 % create map grid:
 %O = zeros(20);
 % create obstacles:
 %O(1:3,1)=1;
 %O(1:3,3)=1;
 %O(1,1:3)=1;
 %O(3,1:3)=1;
 %O(2,2)=1;
 %O(10,15:90)=1;
 %for n=0:20;
 %    O(30-n,30+n)=1; % diagonal line
 %end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % create map grid:
 M = zeros(size(O));
 subgrid=5;

  ForGridSize=size(M);

 gridsize=ForGridSize(1,2);

 Ypathindex=1;
 Xpathindex=1;
pathYL=zeros(gridsize);
pathYR=zeros(gridsize);
pathX=zeros(gridsize);

 %%%Path grids

    for centre=1:subgrid:(gridsize/2);

for x=1:(gridsize/2);

if (gridsize/2-centre)^2 - (x-centre)^2 >0

y=round(sqrt((gridsize/2-centre)^2 - (x-centre)^2));

end

if (x-centre)^2-(gridsize/2-centre)^2  >0
```

```
 y=round(sqrt(x-centre)^2-(gridsize/2-centre)^2);

 end

 if (x-centre)==(gridsize/2-centre)

 y=1;

 end

 if gridsize>y
     if y>0

M(y,x)=1;
M(y,(gridsize-x))=1;
pathYL(Ypathindex,x)=y;
pathYR(Ypathindex,(gridsize-x))=y;

    end
end
end

 Ypathindex=Ypathindex+1;

    end

for y1=0:subgrid:gridsize

Xpathindex=Xpathindex+1;

for lx=1:gridsize

ly=round((lx-1)*(y1-gridsize/2)/(gridsize)+gridsize/2);

M(lx,ly)=1;

pathX(Xpathindex,lx)=ly;

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DGoalXL(1:round(gridsize/subgrid*2))=0;
DGoalXR(1:round(gridsize/subgrid*2))=0;
DGoalY(1:round(gridsize/subgrid*2))=0;

 for Ypathindex=1:gridsize

     b=gridsize*2;

     for x=1:gridsize

     if (pathYL(Ypathindex,x)>0)
```

```
          if (O(pathYL(Ypathindex,x),x)==1)

              a=pathYL(Ypathindex,x);

              if a<b
                  b=a;
              end

          end

     end

     end

    DGoalXL(Ypathindex)=b;

 end

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for Ypathindex=1:gridsize

     b=gridsize*2;

     for x=1:gridsize

     if (pathYR(Ypathindex,x)>0)

         if (O(pathYR(Ypathindex,x),x)==1)

             a=pathYR(Ypathindex,x);

             if a<b
                 b=a;
             end

         end

    end

    end

   DGoalXR(Ypathindex)=b;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for Xpathindex=1:gridsize

    a=gridsize*2;

    for y=1:gridsize

    if pathX(Xpathindex,x)>0%  Here x and y should be interchange


        if (O(y,(pathX(Xpathindex,y)))==1)
```

```
                a=y;

            end
        end
        end

      DGoalY(Xpathindex)=a;
  end

  %%%%Required heading angle
  p=0;
 % theta=atan((Gy-ycur)/(Gx-xcur));
  if Gy==ycur
  else
 p=round(((Gx-xcur)/(Gy-ycur))*(4-ycur)+xcur);
 end
 ynew=1+ycur;
 xnew=p;


%%%%%%%%%%%%%%%%%%%%%%EndOfProgramme%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix E

MatLab Program for Appling Polynomial Approximation to Path Prediction

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Polynomial Approximation for path prediction           %%%%%%%%%%
%12/10/2008                                             %%%%%%%%%%
%ForThesisPolyPlot.m                                    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
tf=20;
rf=250;
tb=50  % time block
time = [0:0.05:40]*tf;  % from 0 to 100000 seconds
T1 = [0.5*sin((time*0.1*pi)/tf)];
T1=T1*rf%+2*rand(size(T1));
T2 = [sin((time*0.1*pi/2)/tf)];
T2=T2*rf%+2*rand(size(T2));
index=tb-1;
for z=1:(index-1)
        prett(z)=time(z);
        preFy(z)=0;
        preFx(z)=0;
end
for k=0:tb:(length(time)-tb)

    if k>tb
    for i=-tb:1:tb
        if length(time)>(k+i)
        fxd(i+tb+1)=T1(k+i);
        fyd(i+tb+1)=T2(k+i);
        tt(i+tb+1)=time(k+i);
        end
    end
    else
  for i=1:tb
        if length(time)>(k+i)
        fxd(i)=T1(k+i);
        fyd(i)=T2(k+i);
        tt(i)=time(k+i);
        end
  end
    end
polydeg=10;
py = polyfit(tt,fyd,polydeg);
px = polyfit(tt,fxd,polydeg);
for i=1:tb
index=index+1;
if length(time)>index
prett(index)=time(index);
preFy(index)=polyval(py,prett(index));
preFx(index)=polyval(px,prett(index));
end
    end
end

figure(1)
plot(time,T1)
```

```
xlabel('Time / [S]');
ylabel('Longitudinal Distance / [m]');

figure(2)
plot(time,T2)
xlabel('Time / [S]');
ylabel('Lateral Distance / [m]');

figure(3)
plot(T1,T2)
xlabel('Longitudinal Distance / [m]');
ylabel('Lateral Distance / [m]');
title('Actual Path');

figure(4)
plot(T1,T2,'r',preFx,preFy,'b')
xlabel('Longitudinal Distance / [m]');
ylabel('Lateral Distance / [m]');
title('Actual and Predicted Path');

r=1:min(length(T1),length(preFx));
efx(r)=preFx(r)-T1(r);

s=1:min(length(T2),length(preFy));
efy(s)=preFy(s)-T2(s);

figure(5)
plot((efx+efy)/2,'r')
xlabel('Time [S]');
ylabel('Error [m]');

%%%%%%%%%%%%%%%%%%%%%%%EndOfProgramme%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix F

MatLab Program for Appling GRNN to Path Prediction

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%GRNN for path prediction                               %%%%%%%%%%
%12/10/2008                                             %%%%%%%%%%
%ForThesisNNErrorPlot.m                                 %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
tf=20;
rf=250;
tb=50  % time block

time = [0:0.05:40]*tf;  % from 0 to 800 seconds
T1 = [0.5*sin((time*0.1*pi)/tf)];
TA1=T1*rf;
T2 = [sin((time*0.1*pi/2)/tf)];
TA2=T2*rf;

T1 = [0.5*sin((time*0.1*pi)/tf)];
T1=T1*rf%+5*rand(size(T1));

T2 = [sin((time*0.1*pi/2)/tf)];
T2=T2*rf%+5*rand(size(T2));

index=tb-1;

for z=1:(index-1)
        prett(z)=time(z);
        preFy(z)=0;
        preFx(z)=0;
end

    for k=0:tb:(length(time)-tb)
            if k>tb
            for i=-tb:1:tb
                if length(time)>(k+i)
                fxd(i+tb+1)=T1(k+i);
                fyd(i+tb+1)=T2(k+i);
                tt(i+tb+1)=time(k+i);
                end
            end
            else
          for i=1:tb
                if length(time)>(k+i)
                fxd(i)=T1(k+i);
                fyd(i)=T2(k+i);
                tt(i)=time(k+i);
                end
          end
            end
        spread = 1;
        netY = newgrnn(tt,fyd,spread);
        netX = newgrnn(tt,fxd,spread);
            for i=1:tb
               index=index+1;
               if length(time)>index
                prett(index)=time(index);
                preFy(index)=sim(netY,prett(index));
                preFx(index)=sim(netX,prett(index));
               end
            end
        end

r=1:min(length(T1),length(preFx));
efx(r)=preFx(r)-T1(r);

s=1:min(length(T2),length(preFy));
efy(s)=preFy(s)-T2(s);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

index=tb-1;

for z=1:(index-1)
        prett(z)=time(z);
        preFy2(z)=0;
        preFx2(z)=0;
end

        for k=0:tb:(length(time)-tb)
            if k>tb
            for i=-tb:1:tb
                if length(time)>(k+i)
                fxd(i+tb+1)=T1(k+i);
                fyd(i+tb+1)=T2(k+i);
                tt(i+tb+1)=time(k+i);
                end
            end
            else
          for i=1:tb
                if length(time)>(k+i)
                fxd(i)=T1(k+i);
                fyd(i)=T2(k+i);
                tt(i)=time(k+i);
                end
          end
           end
        spread = 5;
        netY = newgrnn(tt,fyd,spread);
        netX = newgrnn(tt,fxd,spread);
            for i=1:tb
                index=index+1;
                if length(time)>index
                prett(index)=time(index);
                preFy2(index)=sim(netY,prett(index));
                preFx2(index)=sim(netX,prett(index));
                end
            end
        end

r=1:min(length(T1),length(preFx2));
efx2(r)=preFx2(r)-T1(r);

s=1:min(length(T2),length(preFy2));
efy2(s)=preFy2(s)-T2(s);
```

```
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 index=tb-1;
 for z=1:(index-1)
         prett(z)=time(z);
         preFy3(z)=0;
         preFx3(z)=0;
 end
         for k=0:tb:(length(time)-tb)
             if k>tb
             for i=-tb:1:tb
                 if length(time)>(k+i)
                 fxd(i+tb+1)=T1(k+i);
                 fyd(i+tb+1)=T2(k+i);
                 tt(i+tb+1)=time(k+i);
                 end
             end
             else
          for i=1:tb
                 if length(time)>(k+i)
                 fxd(i)=T1(k+i);
                 fyd(i)=T2(k+i);
                 tt(i)=time(k+i);
                 end
           end
             end
         spread = 10;
         netY = newgrnn(tt,fyd,spread);
         netX = newgrnn(tt,fxd,spread);
             for i=1:tb
                 index=index+1;
                 if length(time)>index
                 prett(index)=time(index);
                 preFy3(index)=sim(netY,prett(index));
                 preFx3(index)=sim(netX,prett(index));
                end
            end
        end
r=1:min(length(T1),length(preFx3));
efx3(r)=preFx3(r)-T1(r);
s=1:min(length(T2),length(preFy3));
efy3(s)=preFy3(s)-T2(s);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(1)
plot(time,T1)
xlabel('Time / [S]');
ylabel('Longitudinal Distance / [m]');

figure(2)
plot(time,T2)
xlabel('Time / [S]');
ylabel('Lateral Distance / [m]');

figure(3)
plot(T1,T2)
xlabel('Longitudinal Distance / [m]');
ylabel('Lateral Distance / [m]');
title('Actual Path');
```

```
figure(4)
plot(TA1,TA2,'k',preFx,preFy,'g',preFx2,preFy2,'b',preFx3,preFy3,'r')
xlabel('Longitudinal Distance / [m]');
ylabel('Lateral Distance / [m]');
title('Actual and Predicted Paths');

figure(6)
plot(efx/2,'g')
xlabel('Time [S]');
ylabel('Longitudinal Error [m]');
hold on

figure(7)
plot(efy/2,'g')
xlabel('Time [S]');
ylabel('Lateral Error [m]');
hold on

figure(6)
plot(efx3/2,'r')
xlabel('Time [S]');
ylabel('Longitudinal Error [m]');
hold on

figure(7)
plot(efy3/2,'r')
xlabel('Time [S]');
ylabel('Lateral Error [m]');
hold on

figure(6)
plot(efx2/2,'b')
xlabel('Time [S]');
ylabel('Longitudinal Error [m]');
hold off

figure(7)
plot(efy2/2,'b')
xlabel('Time [S]');
ylabel('Lateral Error [m]');
hold off

%%%%%%%%%%%%%%%%%%%%%EndOfProgramme%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix G

MatLab Program to Compare Velocity Obstacle Method with Novel Method for Area Prediction of Dynamic Obstacles

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Comparing VOM with Novel method                          %%%%%%%%%%
%12/10/2008                                                %%%%%%%%%%
%CompareAreaPre.m                                          %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
tf=20;
rf=250;
tb=50  % time block
N=0 % For noise
s=0.6 % Sea condition
time = [0:0.05:40]*tf;  % from 0 to 800 seconds

T1 = [0.5*sin((time*0.1*pi)/tf)];
T1=T1*rf+5*rand(size(T1));
RV1= rf*[0.5*0.1*pi*cos((time*0.1*pi)/tf)]/tf;
RVf1=1.2+(0.1*RV1/2);

T2 = [sin((time*0.1*pi/2)/tf)];
T2=T2*rf+5*rand(size(T2));
RV2 = rf*[0.1*pi/2*cos((time*0.1*pi/2)/tf)/tf];
RVf2=1.2+(0.1*RV1/2);

index=tb;

for z=1:(index-1)
        prett(z)=time(z);
        preFy(z)=0;
        ApreAFy1(z)=0;
        ApreAFy2(z)=0;
        preFx(z)=0;
        ApreAFx1(z)=0;
        ApreAFx2(z)=0;
end

    for k=0:tb:(length(time)-tb)

          if k>tb

          for i=-tb:1:tb
              if length(time)>(k+i)
              fxd(i+tb+1)=T1(k+i);
              fyd(i+tb+1)=T2(k+i);
              tt(i+tb+1)=time(k+i);
              end
          end

          else

        for i=1:tb
             if length(time)>(k+i)
```

```
                 fxd(i)=T1(k+i);
                 fyd(i)=T2(k+i);
                 tt(i)=time(k+i);
                 end

          end

            end

            spread = 7;
        netY = newgrnn(tt,fyd,spread);
        netX = newgrnn(tt,fxd,spread);

            for i=1:tb
               index=index+1;
                if (length(time)+1)>index
prett(index)=time(index);
preFy(index)=sim(netY,prett(index));
preFx(index)=sim(netX,prett(index));
ApreAFy1(index)=sim(netY,prett(index))+(N+s*(index-k-1)^RVf2(index))
;
ApreAFy2(index)=sim(netY,prett(index))-(N+s*(index-k-1)^RVf2(index))
;
ApreAFx1(index)=sim(netX,prett(index))+(N+s*(index-k-1)^RVf1(index))
;
ApreAFx2(index)=sim(netX,prett(index))-(N+s*(index-k-1)^RVf1(index))
;
                end
            end
    end

r=1:min(length(T1),length(preFx));

AwidthFx(r)=ApreAFx1(r)-ApreAFx2(r);
AwidthFx(r)=((AwidthFx(r).^2).^(0.5));

AdisFx1(r)=T1(r)-ApreAFx1(r);
AdisFx1(r)=((AdisFx1(r).^2).^(0.5));

AdisFx2(r)=T1(r)-ApreAFx2(r);
AdisFx2(r)=((AdisFx2(r).^2).^(0.5));

RMat=size(r);

for e=1:RMat(1,2)
    Aerfx(e)=0;
    if AdisFx1(e)>AdisFx2(e)
        Aerfx(e)=AdisFx1(e)-AwidthFx(e);
         if AwidthFx(e)>=AdisFx1(e)
             Aerfx(e)=0;
         end
    else
        Aerfx(e)=AdisFx2(e)-AwidthFx(e);
         if AwidthFx(e)>=AdisFx2(e)
             Aerfx(e)=0;
         end
    end
end
```

```
s=1:min(length(T2),length(preFy));

AwidthFy(s)=ApreAFy1(s)-ApreAFy2(s);
AwidthFy(s)=((AwidthFy(s).^2).^(0.5));

AdisFy1(s)=T2(s)-ApreAFy1(s);
AdisFy1(s)=((AdisFy1(s).^2).^(0.5));

AdisFy2(s)=T2(s)-ApreAFy2(s);
AdisFy2(s)=((AdisFy2(s).^2).^(0.5));

RMat=size(s);

for e=1:RMat(1,2)
    Aerfy(e)=0;
    if AdisFy1(e)>AdisFy2(e)
        Aerfy(e)=AdisFy1(e)-AwidthFy(e);
         if AwidthFy(e)>=AdisFy1(e)
             Aerfy(e)=0;
         end
    else
        Aerfy(e)=AdisFy2(e)-AwidthFy(e);
         if AwidthFy(e)>=AdisFy2(e)
             Aerfy(e)=0;
         end
    end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
index=tb;

for z=1:(index-1)
        prett(z)=time(z);
        preFy(z)=0;
        BpreAFy1(z)=0;
        BpreAFy2(z)=0;
        preFx(z)=0;
        BpreAFx1(z)=0;
        BpreAFx2(z)=0;
end

        for k=0:tb:(length(time)-tb)

            if k>tb
            for i=-tb:1:tb
                if length(time)>(k+i)
                fxd(i+tb+1)=T1(k+i);
                fyd(i+tb+1)=T2(k+i);
                tt(i+tb+1)=time(k+i);
                end
            end

            else
          for i=1:tb
                if length(time)>(k+i)
                fxd(i)=T1(k+i);
                fyd(i)=T2(k+i);
                tt(i)=time(k+i);
```

```
                  end

            end
              end
             polydeg=1;
     py = polyfit(tt,fyd,polydeg);
     px = polyfit(tt,fxd,polydeg);

              for i=1:tb

                  index=index+1;

                  if (length(time)+1)>index
 prett(index)=time(index);
 preFy(index)=polyval(py,prett(index));
 preFx(index)=polyval(px,prett(index));
 BpreAFy1(index)=polyval(py,prett(index))+(N+s*(index-k-1)^rv)  ;
 BpreAFy2(index)=polyval(py,prett(index))-(N+s*(index-k-1)^rv)  ;
 BpreAFx1(index)=polyval(px,prett(index))+(N+s*(index-k-1)^rv)  ;
 BpreAFx2(index)=polyval(px,prett(index))-(N+s*(index-k-1)^rv)  ;
                 end
             end
         end

 r=1:min(length(T1),length(preFx));

 BwidthFx(r)=BpreAFx1(r)-BpreAFx2(r);
 BwidthFx(r)=((BwidthFx(r).^2).^(0.5));

 BdisFx1(r)=T1(r)-BpreAFx1(r);
 BdisFx1(r)=((BdisFx1(r).^2).^(0.5));

 BdisFx2(r)=T1(r)-BpreAFx2(r);
 BdisFx2(r)=((BdisFx2(r).^2).^(0.5));

 RMat=size(r);

 for e=1:RMat(1,2)
     Berfx(e)=0;
     if BdisFx1(e)>BdisFx2(e)
         Berfx(e)=BdisFx1(e)-BwidthFx(e);
          if BwidthFx(e)>=BdisFx1(e)
              Berfx(e)=0;
          end
    else
        Berfx(e)=BdisFx2(e)-BwidthFx(e);
         if BwidthFx(e)>=BdisFx2(e)
             Berfx(e)=0;
         end
    end
end

s=1:min(length(T2),length(preFy));

BwidthFy(s)=BpreAFy1(s)-BpreAFy2(s);
BwidthFy(s)=((BwidthFy(s).^2).^(0.5));

BdisFy1(s)=T2(s)-BpreAFy1(s);
BdisFy1(s)=((BdisFy1(s).^2).^(0.5));
```

```
  BdisFy2(s)=T2(s)-BpreAFy2(s);
  BdisFy2(s)=((BdisFy2(s).^2).^(0.5));
  RMat=size(s);
  for e=1:RMat(1,2)
        Berfy(e)=0;
          if BdisFy1(e)>BdisFy2(e)
          Berfy(e)=BdisFy1(e)-BwidthFy(e);
           if BwidthFy(e)>=BdisFy1(e)
               Berfy(e)=0;
           end
      else
          Berfy(e)=BdisFy2(e)-BwidthFy(e);
           if BwidthFy(e)>=BdisFy2(e)
               Berfy(e)=0;
           end
      end
 end

 figure(1)
 plot(time,T1,'k',time,ApreAFx1,'r',time,ApreAFx2,'r',....
 time,BpreAFx1,'b',time,BpreAFx2,'b')
 xlabel('Time [S]');
 ylabel('Longitudinal Distance [m]');

 figure(2)
 plot(time,T2,'k',time,ApreAFy1,'r',time,ApreAFy2,'r',....
 time,BpreAFy1,'b',time,BpreAFy2,'b')
 xlabel('Time [S]');
 ylabel('Lateral Distance [m]');

 figure(3)
 plot(time,Aerfx,'r',time,Berfx,'b')
 xlabel('Time [S]');
 ylabel('Error [m]');

 figure(4)
 plot(time,Aerfy,'r',time,Berfy,'b')
 xlabel('Time [S]');
 ylabel('Error [m]');

 figure(6)
 plot(time,AwidthFx,'r',time,BwidthFx,'b')
 xlabel('Time [S]');
 ylabel('Boundary Width  [m]');

 figure(7)
 plot(time,AwidthFy,'r',time,BwidthFy,'b')
 xlabel('Time [S]');
 ylabel('Boundary Width  [m]');

 figure(8)
 plot(time,RwidthFy,'r')
 xlabel('Time [S]');
 ylabel('Reduced Width  [m]');

%%%%%%%%%%%%%%%%%%%%%%EndOfProgramme%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```