THE UNIVERSITY
OF ADELAIDE
AUSTRALIA

Outdoor Navigation: Time-critical Motion Planning for Nonholonomic Mobile Robots

Mohd Sani Mohamad Hashim

School of Mechanical Engineering
The University of Adelaide
South Australia 5005
Australia

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Mechanical Engineering on February 2014

TABLE OF CONTENTS

TABLE OF CONTENTS ..... **i**

LIST OF FIGURES ..... **iv**

LIST OF TABLES ..... **x**

ABSTRACT ..... **xi**

STATE OF ORIGINALITY ..... **xii**

PUBLICATIONS ..... **xiii**

ACKNOWLEDGEMENTS ..... **xv**

1. INTRODUCTION **1**

1.1 MOTIVATION 2
1.2 RESEARCH AIMS 4
1.3 LAYOUT OF THESIS 5

2. LITERATURE REVIEW **7**

2.1 MOTION PLANNING ALGORITHMS 7
2.1.1 Roadmap path planning 9
2.1.2 Cell decomposition path planning 11
2.1.3 Potential field path planning 12
2.1.4 Other path planning approaches 16
2.1.5 Geometric approach for trajectory planning 22
2.2 NAVIGATION ENVIRONMENTS 25
2.2.1 Outdoor navigation 26
2.3 OBSTACLE AVOIDANCE 27
2.4 MULTIPLE ROBOTS COORDINATION 30
2.5 SUMMARY AND GAP STATEMENT 32

3. METHODOLOGY **35**

3.1 STAGE 1: DEVELOPMENT OF ALGORITHMS FOR TIME-CRITICAL MOTION PLANNING 36
3.2 STAGE 2: OBSTACLE AVOIDANCE APPROACH 36
3.3 STAGE 3: SIMULATION WORKS 37
3.4 STAGE 4: HARDWARE PREPARATION AND EXPERIMENTAL WORKS 38
3.5 CONCLUDING REMARKS 39

4. DEVELOPMENT OF TIME-CRITICAL MOTION PLANNING ALGORITHMS **40**

4.1 KINEMATIC MODEL OF NONHOLONOMIC MOBILE ROBOT 42
4.2 BOUNDARY CONDITIONS 43

4.3 COORDINATE-X EQUATION 44
4.4 COORDINATE-Y EQUATION 45
4.5 ORIENTATION (θ) EQUATION 48
4.6 STEERING ANGLE (ϕ) EQUATION 48
4.7 ANGULAR VELOCITY (u_1) EQUATION 49
4.8 OBSTACLE AVOIDANCE APPROACH 49
4.8.1 Avoiding static obstacles 50
4.8.2 Avoiding moving obstacles 51
4.9 CONCLUDING REMARKS 54

5. SIMULATION RESULTS AND DISCUSSIONS 56

5.1 SIMULATION ARCHITECTURE 57
5.2 SIMULATED VEHICLE 59
5.3 MATLAB FRAMEWORKS 60
5.4 TRAJECTORY OPTIMIZATION 63
5.4.1 Replanning approach 69
5.5 SIMULATION RESULTS AND DISCUSSIONS 72
5.5.1 Navigation in static and open-space environments 73
5.5.2 Navigation in dynamic and open-space environments 83
5.5.3 Navigation in the city-like environments 93
5.6 CONCLUDING REMARKS 106

6. DEVELOPMENT OF A NONHOLONOMIC MOBILE ROBOT 107

6.1 ROBOT CONTROLLER 110
6.2 WHEEL ENCODER 111
6.3 DETECTION SENSORS 114
6.4 COMMUNICATION 114
6.5 CALIBRATION OF STEERING ANGLE AND VELOCITY 115
6.5.1 Steering angle 115
6.5.2 Velocity 117
6.6 OBSTACLE DETECTION 119
6.7 WIRELESS COMMUNICATION 120
6.8 CONCLUDING REMARKS 120

7. EXPERIMENTAL RESULTS AND DISCUSSIONS 122

7.1 EXPERIMENT ARCHITECTURE 123
7.2 EXPERIMENT SETUP 124
7.3 CASE 1: NAVIGATION IN AN OBSTACLE-FREE ENVIRONMENT 125
7.4 CASE 2: NAVIGATION IN A KNOWN STATIC ENVIRONMENT 130
7.5 CASE 3: NAVIGATION IN AN UNKNOWN STATIC ENVIRONMENT 135

7.5.1 Scenario 1: One unknown static obstacle 135
7.5.2 Scenario 2: Two unknown static obstacles 140
7.6 CASE 4: NAVIGATION IN AN UNKNOWN DYNAMIC ENVIRONMENTS 145
7.6.1 Scenario 1: Opposite direction of mobile robot 145
7.6.2 Scenario 2: From left-hand side of mobile robot 149
7.6.3 Scenario 3: From right-hand side of mobile robot 153
7.7 CONCLUDING REMARKS 156

8. CONCLUSIONS AND FUTURE WORKS 158

CONTRIBUTIONS 159
FUTURE WORKS 161

REFERENCE 163

APPENDIX A 167

APPENDIX B 172

LIST OF FIGURES

Figure 2.1 Path generation (a) Path constraints made of four required postures (b) Generated path (Delingette *et al.*, 1991). 8
Figure 2.2 Roadmap approach (a) Visibility Graph (Jiang *et al.*, 1997). (b) Voronoi diagram (Siegwart and Nourbakhsh, 2004). 9
Figure 2.3 Cell decomposition method (a) A fixed-resolution grid. (b) A triangulation (Ge and Lewis, 2006). 11
Figure 2.4 Simulation results by using (a) trapezoidal decomposition and (b) triangular decomposition (Ghita and Kloetzer, 2012). 12
Figure 2.5 Potential field method (Safadi, 2007). 13
Figure 2.6 Path generated by the navigation algorithm (Cosio and Castaneda, 2004). 14
Figure 2.7 Implementation of the proposed algorithm by Koh and Cho (Koh and Cho, 1999). 17
Figure 2.8 Results from the information- based method (Mihaylova *et al.*, 2003) 18
Figure 2.9 Cell mapping model (a) with 30^5 cells. (b) with 40^5 cells (Li and Wang, 2003). 18
Figure 2.10 (a) Generated trajectory (b) Velocity profile (c) Acceleration profile (Prado *et al.*, 2003). 19
Figure 2.11 Neuro-fuzzy approach (Hui *et al.*, 2006). 20
Figure 2.12 Generated trajectory with several control points (Haddad *et al.*, 2007). 21
Figure 2.13 Simulation results in (a) a complex scenario, and (b) a long corridor (Ma *et al.*, 2013). 21
Figure 2.14 Different types of curves used to connect four postures for path generation (Shin and Singh, 1990). 22
Figure 2.15 An optimal path (a) minimum energy, (b) minimum travel distance, and (c) minimum travel time (Liu and Sun, 2011). 25
Figure 2.16 Outdoor navigation (a) Pioneer3-AT with URG and SICK (Chang *et al.*, 2009) (b) The Cycab used in the experimental works (Zhang *et al.*, 2006). 26

Figure 2.17 Plan view of the observer moving in dynamic environment (a) Exocentric reference frame (b) Egocentric reference frame (Fajen and Warren, 2003). 28
Figure 2.18 Avoiding a dynamic obstacle (Jolly *et al.*, 2008). 30
Figure 2.19 A group of robots in hunting operation (Yamaguchi, 2003). 31
Figure 2.20 Subtasks of construction task (Stroupe *et al.*, 2005). 31
Figure 2.21 Overview of the system (Klancar *et al.*, 2004). 32
Figure 3.1 Stages for proposed methodology 35
Figure 3.2 Generalized steps for avoiding an obstacle 37
Figure 3.3 The modified mobile robot used in the experimental works. 38
Figure 4.1 Flowchart of the proposed algorithms 41
Figure 4.2 A car-like mobile robot 42
Figure 4.3 Avoiding a detected static obstacle which is unknown in priori 50
Figure 4.4 Avoiding a moving obstacle (a) perpendicular direction to the mobile robot and (b) in opposition to the mobile robot. 52
Figure 4.5 Collision prediction approach (a) before detection of the obstacle, (b) first detection, (c) predicted position falls inside the collision radius, and (d) obstacle avoidance approach implemented. 53
Figure 5.1 Simulation process flowchart. 58
Figure 5.2 Geometric model of a mobile robot. 59
Figure 5.3 Simulated Laser Range Finder. 61
Figure 5.4 Simulation map with static and moving obstacles. 62
Figure 5.5 The Graphical User Interface (GUI) for simulation framework (a) Input GUI, (b) Output GUI 63
Figure 5.6 Original trajectory plan. 64
Figure 5.7 Final result of the trajectory. 65
Figure 5.9 Steering angle profiles (a) Planned steering angle (red line) against adjusted steering angle (red dashed), and (b) adjusted steering angle (red dashed) against actual steering angle (blue line). 66
Figure 5.10 Velocity profiles (a) Planned velocity (red line) against adjusted velocity (red dashed), and (b) adjusted velocity (red dashed) against actual velocity (blue line). 67
Figure 5.11 Adjusted trajectory (red dashed) against actual trajectory (blue line). 68
Figure 5.12 Prior map with two waypoints connecting the initial and final point. 70

Figure 5.13 Simulation results with replanning approach. 71
Figure 5.14 A complicated obstructed environment. 73
Figure 5.15 One mobile robot navigates in the environment. 75
Figure 5.16 Robot 1: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) location. 76
Figure 5.17 Two mobile robots navigate in the environment. 78
Figure 5.18 Robot 2: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position. 79
Figure 5.19 Three mobile robots navigate in the environment. 81
Figure 5.20 Robot 3: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position. 82
Figure 5.21 Simulated environment for Case 4 83
Figure 5.22 One mobile robot navigates in a dynamic environment. 85
Figure 5.23 Robot 1: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position. 86
Figure 5.24 Simulated environment for Case 5 87
Figure 5.25 Two mobile robots navigate in a dynamic environment. 88
Figure 5.26 Robot 2: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position. 89
Figure 5.27 Simulated environment for Case 6. 90
Figure 5.28 Three mobile robots navigate in a dynamic environment 91
Figure 5.29 Robot 3: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position. 92
Figure 5.30 (a) A simplified city-like map, (b) Multiple waypoints trajectory planning. 95
Figure 5.31 Initial trajectories in a city-like map. 96
Figure 5.32 (a) Before detecting an obstacle. (b) Obstacle detected at the 9^{th} second. (c) Starts to move along new trajectory. (d) Reaches the first waypoint at the 30^{th} second. 97
Figure 5.33 (a) Before detecting an obstacle. (b) Obstacle detected at the 67^{th} sec. (c) Starts to move along new trajectory. (d) Passes through moving obstacle safely. 98

Figure 5.34 (a) Before detecting an obstacle. (b) Obstacle detected at the 68th sec. (c) Starts to move along new trajectory. (d) Passes through moving obstacle safely. 99
Figure 5.35 Final result at the 120th second. 100
Figure 5.36 Second scenario with two mobile robots and one moving obstacle. 101
Figure 5.37 Final result at the 120th second for second scenario. 102
Figure 5.38 Third scenario with three mobile robots and two moving obstacles. 104
Figure 5.39 Final result at 100th second for third scenario. 105
Figure 6.1 The modified car-like robot used in experimental works. 108
Figure 6.2 Mobile robot platform. 109
Figure 6.3 Sensor platform 109
Figure 6.4 Sensor platform attached to the mobile robot platform. 110
Figure 6.5 Robot controller 111
Figure 6.6 (a) Magnets mounting attached at the wheel (b) Hall Effect sensors attached at the rear axle 112
Figure 6.7 Hall effect sensor 112
Figure 6.8 Location of the wheel encoder 113
Figure 6.9 Magnet mounting of encoder 113
Figure 6.10 (a) Ultrasonic range sensors (b) Sensor attached to the sensor base. 114
Figure 6.11 Wireless communication (a) Router (b) Coordinator 115
Figure 6.12 Calibration work for establishment of steering angle 116
Figure 6.13 Relation between PWM values and steering angle. 117
Figure 6.14 Calibration work for establishment of velocity 117
Figure 6.15 Relation between PWM values and speed 118
Figure 6.16 Obstacle detection range for experimental works. 119
Figure 6.17 Wireless communication between the operator and the router (robot). 120
Figure 7.1 Experimental work flow 123
Figure 7.2 Testing arena 124
Figure 7.4 Mobile robot navigated in an obstacle-free environment (simulation) 126
Figure 7.5 Mobile robot navigated in an obstacle-free environment (experiment) 127
Figure 7.6 Case 1: Trajectory planning without an obstacle 128
Figure 7.7 Experimental setup for Case 2. 130
Figure 7.8 Mobile robot navigated in a known static environment (simulation) 131

Figure 7.9 Mobile robot navigated in a known static environment (experiment). 132
Figure 7.10 (a) Case 2: Trajectory planning with a known static obstacle, (b) Experimental results.....133
Figure 7.11 (a) Plan view (b) Actual experimental setup for Scenario 1 136
Figure 7.12 Initial collision-free trajectory for Case 3 136
Figure 7.13 Mobile robot navigates in the unknown static environment (simulation).....137
Figure 7.14 Mobile robot navigates in the unknown static environment (experiment).....138
Figure 7.15 Theoretical and actual trajectory for Case 3.....139
Figure 7.16 Experimental setup for Scenario 2 140
Figure 7.17 Mobile robot navigates through two unknown obstacles (simulation).....142
Figure 7.18 Mobile robot navigates through two unknown obstacles (experiment).....143
Figure 7.19 Theoretical and actual trajectory for Case 4.....144
Figure 7.20 Moving obstacle coming from the opposite direction of the mobile robot 145
Figure 7.21 Scenario 1: Moving obstacle from the opposite direction of the mobile robot (simulation) 146
Figure 7.22 Scenario 1: Moving obstacle from the opposite direction of the mobile robot (experiment) 147
Figure 7.23 Theoretical and actual trajectory for scenario 1 148
Figure 7.24 Moving obstacle coming from left-hand side of the mobile robot.....149
Figure 7.25 Scenario 2: Moving obstacle from the left-hand side of the mobile robot (simulation).....150
Figure 7.26 Scenario 2: Moving obstacle from the left-hand side of the mobile robot (experiment) 151
Figure 7.27 Theoretical and actual trajectory for scenario 2 152
Figure 7.28 Moving obstacle coming from right-hand side of the mobile robot.....153
Figure 7.29 Scenario 3: Moving obstacle from the right-hand side of the mobile robot (simulation).....154
Figure 7.30 Scenario 3: Moving obstacle from the right-hand side of the mobile robot (experiment) 155

Figure 7.31 Theoretical and actual trajectory for scenario 3 156

LIST OF TABLES

Table 2.1 Intrinsic splines' family (Delingette *et al.*, 1991) 24
Table 5.1 Input data for replanning approach scenario 69
Table 5.2 Actual collected data of simulation without replanning approach 70
Table 5.3 Actual collected data with replanning approach 72
Table 5.4 Input data for simulation Case 1. 74
Table 5.5 Actual data collected at the final point for Case 1 76
Table 5.6 Input data for simulation Case 2. 77
Table 5.7 Actual data collected at the final point for Case 2 79
Table 5.8 Input data for simulation Case 3. 80
Table 5.9 Actual data collected at the final point for Case 3 82
Table 5.10 Input data for simulation Case 4 84
Table 5.11 Actual data collected at the final point for Case 4 86
Table 5.12 Input data for simulation Case 5 87
Table 5.13 Actual data collected at the final point for Case 5 89
Table 5.14 Input data for simulation Case 6 90
Table 5.15 Actual data collected at the final point for Case 6 93
Table 5.16 Parameters for the first mobile robot (R1) 94
Table 5.17 Parameters for the second mobile robot (R2) 94
Table 5.18 Table 3 Errors for Case 1 at final point. 100
Table 5.19 Parameters for second simulation case 102
Table 5.20 Errors for Case 2 at the final point. 103
Table 5.21 Parameters for third simulation case 104
Table 5.22 Errors for Case 3 at the final point. 105
Table 6.1 Steering angles under different PWM values 116
Table 6.2 Velocities under different PWM values 118
Table 7.1 Actual initial and final positions for Case 1 129
Table 7.2 Actual initial and final positions for Case 2 134
Table 7.3 Actual initial and final positions for Case 3 140
Table 7.4 Actual initial and final positions for Case 4 144

ABSTRACT

The question of timing in mobile robot navigation still remains an area of research not thoroughly investigated. In certain situations, a mobile robot may need not only to reach a desired location safely, but to arrive at that location at a specified time. Such a situation may have significant ramifications for applications to which a robot is tasked, for example patrolling large areas, delivering goods or coordinating multiple mobile robots. Thus, it is important for a mobile robot to be able to plan its trajectories and movements in order to navigate from initial location to a final destination whilst considering timing, orientation and velocity. Furthermore, it should also be able to detect and avoid any obstacles encountered in its path during navigating through the environment.

The aim of this research is therefore to develop a time-critical motion planning algorithm, which includes planning the trajectory, position and orientation of a mobile robot, with obstacle avoidance capability for a single or multiple nonholonomic mobile robots. In addition, the mobile robot should be able to replan its original trajectories in order to 'make up' any loss of time caused by avoiding obstacles. An Ackermann car-like robot has been considered specifically during the development stage, with consideration given to the kinematic and dynamic constraints of nonholonomic mobile robot in general. The resultant algorithm is based on the geometric approach.

In achieving the research objectives, this study is conducted in four stages. The first stage deals with the development of a new algorithm for time-critical motion planning in order to navigate safely in an environment, to reach the specified location at the specified time, with the required orientation, velocity and with the consideration of the kinematic and dynamic constraints of the mobile robot. In the second stage, the algorithm should have the capability to avoid any unknown static and dynamic obstacles when the mobile robot starts to move from its initial point. The algorithm should have the ability to replan its original trajectory to compensate for time loss due to avoiding obstacles. Prior to experimental works, the simulations will be carried out to ascertain the effectiveness of the algorithm. In the final stage, experimental works will be undertaken to validate the algorithms utilising an Ackermann car-like robot.

STATE OF ORIGINALITY

To the best of my knowledge, except where otherwise referenced and cited, everything that is presented in this thesis is my own original work and has not been presented previously for the award of any other degree or diploma in any university. If accepted for the award of the degree of Doctor of Philosophy in Mechanical Engineering, I consent that this thesis be made available for loan and photocopying.

_____

Mohd Sani Mohamad Hashim

_____

Date

PUBLICATIONS

Conference papers (Main author)

1. Mohd Sani Mohamad Hashim and Tien-Fu Lu, “Time-dependent motion planning for nonholonomic mobile robot”, The 9th International IFAC Symposium on Robot Control (SYROCO’09), Gifu, Japan, 9-12 September 2009.

2. Mohd Sani Mohamad Hashim and Tien-Fu Lu, “Multiple waypoints trajectory planning with specific position, orientation, velocity and time using geometric approach for a car-like robot”, 2009 Australasian Conference on Robotics and Automation (ACRA ’09), Sydney, Australia, 2-4 December 2009.

3. Sani Hashim and Tien-Fu Lu, “A new strategy in dynamic time-dependent motion planning for nonholonomic mobile robots”, 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO’09), Guilin, China, 18-22 December 2009.

4. Mohd Sani Mohamad Hashim and Tien-Fu Lu, “Performance of a time-dependent motion planning for a car-like robot in static environments”, 2012 International Conference on Man Machine System (ICoMMS ’12), Penang, Malaysia, 27-28 February 2012.

5. Mohd Sani Mohamad Hashim, Tien-Fu Lu and Hassrizal Hassan Basri, “Dynamic Obstacle Avoidance Approach for Car-like Robots in Dynamic Environments”, 2012 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE 2012), Kota Kinabalu, Sabah, Malaysia, 3-4 December 2012.

6. Mohd Sani Mohamad Hashim and Tien-Fu Lu, “Time-critical Trajectory Planning for a Car-like Robot in Unknown Environments”, IEEE Business, Engineering and Industrial Applications Colloquium 2013 (BEIAC 2013), Langkawi, Malaysia, 8-9 April 2013.

Conference papers (Co-author)

1. Zhiyong Zhang, Dongjian He, Tien-Fu Lu and Sani Hashim, “Study on Steering Actuator Transfer Function of Picking Mobile Robot”, 2010 International Conference on Communications and Mobile Computing (CMC 2010), Shenzhen, China, 12-14 April 2010.

Journal papers

1. Mohd Sani Mohamad Hashim and Tien-Fu Lu, “Real-time Control of Time-Critical Trajectory Planning for a Car-like Robot in Unknown Environments”, International Journal of Engineering Research and Technology (IJERT), ISSN: 2278-0181, February - 2013 (Vol.2, Issue 2), 2013.

2. Mohd Sani Mohamad Hashim and Tien-Fu Lu, “Time-dependent Motion Planning for a Car-like Robot in Dynamic Environments using Geometric Approach”, International Journal of Imaging and Robotics (IJIR), Revised, 2012.

ACKNOWLEDGEMENTS

This work would not have been possible without the assistant of a number of people. I would initially like to thank my father, Mohamad Hashim and my mother, Norriah Salleh as well as my family who have continuous support and motivate me throughout my PhD study in Australia.

I wish to thank my principal supervisor Dr. Tien-Fu Lu for his help, patient guidance and encouragement throughout the period of this project. I would also like to thank my co-supervisor Dr Lei Chen for his valuable suggestions. Special thanks also go to my colleagues; ZhenZhang, Tommie, Xinrui, Guntur, Kuan and Sukri.

The help and support from people in School of Mechanical Engineering have also been invaluable. I would like to thank colleague from the electronics and mechanical workshop; Philip Schmidt, Norio Itsumi and Billy Constantine. I would also like to thank Ms. Karen Adams for her helpful language support research and to who involve directly and indirectly throughout my PhD.

1. INTRODUCTION

Mobile robotics has become a significant research field over the past few decades. This field has experienced a major evolution in design, control, application and other aspects which make the mobile robot useful for human activities. Mobile robots come in many shapes and types such as car-like robots, two- or three-wheel robots, omni-directional robots and mobile manipulators. One of the must-have basic capabilities of mobile robots is navigation. With a decent navigation system, a mobile robot is able to move and explore the environment autonomously, accurately and safely. Furthermore, the mobile robot is also able to go to any selected places without human intervention. There are two types of environments for mobile robot navigation, which are indoor environments and outdoor environments. Indoor environment mostly deals with navigation inside buildings, while outdoor environment deals with navigation outside buildings. Outdoor navigation can become more complex and sophisticated than indoor navigation due to the unknowns of the environments and dynamically-changed ambience of outdoor environments. So far, most of the researches have attempted to develop the most reliable navigation systems that meet certain criteria such as ability to choose the shortest path, minimum time path or minimum energy

usage. Thus the variation of criteria will reflect the selection of navigation strategies and approaches for the mobile robot.

1.1 Motivation

Nowadays, there are many types of mobile robots which have been developed to assist and to ease human workforce in real-world environments. These mobile robots are used in indoor or outdoor environments for varieties of tasks and applications such as factory automation, underground mining, military surveillance and even space exploration. In most cases, the mobile robots often work in unknown and dynamic environments. Thus it is needed to ensure that the mobile robots are able to navigate and execute the tasks safely and successfully. Furthermore, they should also be able to react reasonably to the environments in the presence of obstacles.

One of the fundamental issues in mobile robot navigation is motion planning. Motion planning can be understood as how a mobile robot plans and chooses its path and moves along that path. In motion planning, the main problem is to determine a collision-free and smooth path in order to reach the final location from its initial location. In general, motion planning can be divided into two steps (Delingette *et al.*, 1991; Tounsi and Corre, 1996). The first step is path planning, which is defined as a step to generate a geometric curvature to connect the initial and final position of the mobile robot. Once the path is known, the second step is to determine the motion control. Motion control is defined as a step to determine the velocity of the mobile robot by using linear velocity law, with which the mobile robot will follow the path.

As mentioned earlier, mobile robots have been used in a wide range of applications in various working environments. For outdoor environments, it is very common for the mobile robots to face unexpected conditions such as uneven terrain, unknown and dynamic obstacles as well as polluted air (dust and smoke). These conditions may cause trouble to the mobile robot's control and sensor systems as the system error may increase. Furthermore, the mobile robots may also accumulate errors from within the robot systems themselves such as friction and wheel slippage. In contrast, indoor environments are more ideal conditions with some information may be known beforehand, which serves as prior information. Prior information does not necessarily

give an accurate knowledge of the environments but sufficient knowledge will ensure the mobile robot is able to navigate effectively. For outdoor environments, prior information may also be available such as topological map which can be useful for mobile robots. Although prior information for outdoor environments is not as accurate or as extensive compared to prior information for indoor environments, the mobile robots need to take advantage of this prior information in order to navigate safely and to reduce the uncertainties and errors in outdoor environments.

Projects such as military surveillance and social security patrol are useful to monitor and maintain safety in the private areas such as cities and buildings. Such projects can prevent or reduce the rate of criminal activities, monitor social activities and traffics. Most surveillance systems are using cameras, which are installed at specific locations and these cameras are monitored by automated computer programs. For aerial surveillance system, unmanned aerial vehicles (UAVs) are usually being used. The UAVs will capture images or videos of the covered area and the captured visual will be processed and interpreted to gather information about the area. Likewise, unmanned ground vehicles (UGVs) are used for actions engaged on the ground. The capability of UAVs and UGVs usually depend on the sensors used and their ability to move around.

This study focuses only on UGVs for a ground-based surveillance. Thus a basic task for this application is to ensure the UGV is being able to navigate autonomously from one point to another point in its environment with capability of avoiding the obstacles. Furthermore, in certain situations such as large area patrol and goods delivery, timing is crucial as the mobile robot needs to arrive at the desired place with not only to the right location but also to the right orientation, exactly at the specified time. In a large area patrol, usually the mobile robot needs to arrive at every checkpoint with the correct orientation exactly at the desired time, to ensure the whole patrolling area can be covered within the specified time. In such case, the mobile robot should be able to plan its motion and complete patrolling the whole area within the specified time and also cover the angle of views for each checkpoint. For multiple mobile robot applications, especially in soccer robot competition, if robot timing can be controlled in addition to its position and orientation, the soccer robot does not need to wait for its teammate for a long time in order to receive the ball. If the robot waits at the certain

location for quite some time, perhaps it has already been detected by the opponent team and has been man-marked, which makes it difficult to the robot to receive the ball from its teammate. Furthermore in multiple mobile robot applications, two mobile robots may deliver and exchange goods at a desired meeting point at the specified time. If the journey time can be controlled for each of the robots, they do not need to wait for each other for a long time at the meeting point. Both robots can arrive at the meeting point at the specified time, exchange goods and then continue their journey to their separate final locations.

For the aforementioned examples, timing is crucial to the mobile robot to achieve its task. This situation is advantageous for a task-based mission, not only for a single mobile robot but also for multiple mobile robots which requires the mobile robot reach the final location at the specified time.

1.2 Research aims

In brief, the aim of this study is to develop a new motion planning for unmanned ground vehicles. The vehicle is a nonholonomic mobile robot navigating in a partially known and dynamic 2D environment with kinematic and dynamic constraints are taken into account during development stage. Thus, the primary objectives are:

1. To develop time-critical motion planning algorithm for nonholonomic mobile robots by associating new parameters such as position, velocity, orientation and time
2. To develop a dynamic obstacle avoidance algorithm that is able to avoid both static and moving obstacles safely. Furthermore, the dynamic obstacle avoidance algorithm needs to be able to catch up the time lost due to the mobile robot avoiding the obstacles in order to reach the final point at the specified time and orientation
3. To incorporate the newly developed time-critical motion planning algorithm for multiple robots and multiple waypoints planning and

4. To develop a real autonomous mobile robot using an Ackermann car-like robot and to conduct experimental works in order to validate the newly developed time-critical motion planning and obstacle avoidance algorithms.

1.3 Layout of thesis

The rest of this thesis is organized as follows:

Chapter 2: Literature Reviews

This chapter introduces the general background of this study. The related works on mobile robots, motion planning and obstacle avoidance approach are reviewed. At the end of the chapter, all the findings are summarized and gaps and contributions from this study are pointed out.

Chapter 3: Methodology

In order to achieve the primary objectives, this study is divided into four stages. The first stage deals with the development of time-critical motion planning algorithm for nonholonomic mobile robot. The second stage deals with the development of dynamic obstacle avoidance algorithm. In the third stage, an autonomous mobile robot will be developed. Lastly, the newly developed time-critical motion plannning and obsatcle avoidance algorithms will be validated through experimental works using the developed autonomous mobile robot.

Chapter 4: Development of Time-critical Motion Planning Algorithms

The fundamentals and the detail mathematics of the algorithms are discussed in this chapter. The development of the time-critical motion planning algorithm is based on geometric approach with cubic and quintic polynomials are adopted to generate motion trajectories. Furthermore, detail development of dynamic obstacle avoidance algorithm, multiple waypoints planning and multiple robots planning are also presented in this chapter.

Chapter 5: Simulation Results and Discussions

This chapter presents the development of a simulation framework using Matlab. The nonholonomic mobile robot and the developed algorithms are tested

using this simulation framework. A series of simulations are conducted to investigate the effectiveness and practicality of the algorithms. The algorithms are tested in the static and dynamic enviroments with a single and multiple mobile robots.

Chapter 6: Development of a Non-holonomic Mobile Robot

In this chapter, the development of an autonomous mobile robot is presented. A remote control car are modified to be used for the experimental works. Furthermore, the development of the autonomous mobile robot needs to overcome several issues such as the capability of steering wheels to turn for desired angles and the mobile robot requires to speed up and slow down at specified velocities within seconds. Hence the kinematic and dynamic constraints of the mobile robot such as steering angle and velocity limitation are also considered during development of this mobile robot. In addition, the calibration works have been conducted to establish the PWM-steering angle and PWM-speed relationships for the mobile robot.

Chapter 7: Experimental Results and Discussions

The experimental architecture and results from experimental works are presented and discussed in this chapter. The developed algorithms are tested through a series of experimental enviroments using the developed autonomous mobile robot. Then the experiment results are compared to the simulation results in order to validate the algorithms.

.

Chapter 8: Conclusions and Future Works

The findings of this study are summarized in this chapter. The recommendation for the future works are also given at the end of this chapter.

2. LITERATURE REVIEW

In this chapter, the main areas of related research have been reviewed, which are mobile robots, motion planning, obstacle avoidance and multiple robots coordination. All these reviewed areas of research will contribute to the main objectives of this study. At the end of this section, all the findings are summarised and gap statement is given.

2.1 Motion planning algorithms

For the past few decades, navigation problems have been extensively studied. One of the fundamental issues for navigation is to plan the robot's motion in the working environment without human intervention. This issue is commonly known as motion planning. Earlier works in mobile robot motion planning concentrated on how to determine the collision-free path in order to reach the final location (Salichs and Moreno, 2000). One common problem in motion planning for mobile robots is to determine the control input which the mobile robot requires to achieve a goal position (x, y), pose (x, y, θ) or posture (x, y, θ, κ) (Nagy and Kelly, 2001). Figure 2.1 shows the path constraints made of four postures, which each posture consists of position in

Cartesian coordinates (x, y), orientation (θ) and curvature (κ) (Delingette *et al.*, 1991). Generally, an autonomous mobile robot has to be able to extract information from on-board sensors in order to "know" the environment and plan its motion. Once the path has been planned, the mobile robot is expected to follow the path whilst considering velocity, position, orientation and other requirements for the mobile robot to achieve smooth motions. In addition to such considerations, it also might be able to both detect and avoid the obstacles presented during navigation. Typically, motion path is planned based on known obstacles' positions in the environment in prior (Hui *et al.*, 2006).

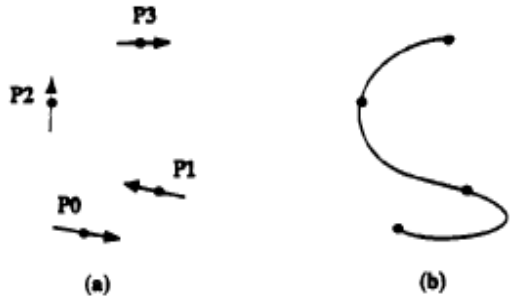


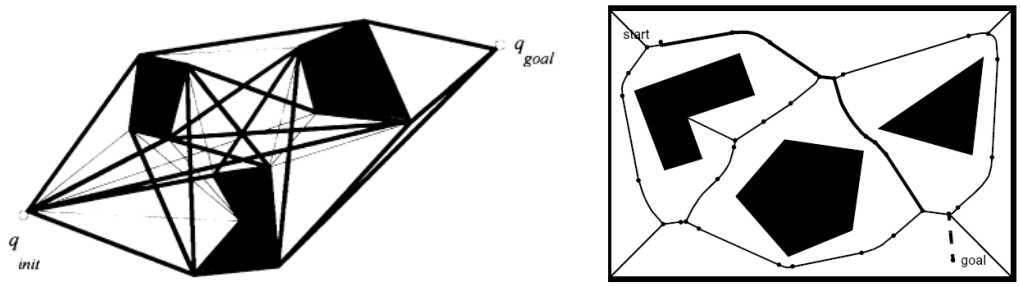
Figure 2.1 Path generation (a) Path constraints made of four required postures (b) Generated path (Delingette *et al.*, 1991).

Generally, path is planned to meet several main requirements such as shortest path, safe path and smooth path. Shortest path could be the shortest distance to arrive at the final location or the shortest travel time. While navigating in the environment, the robot also needs to consider safety issues. This means the path needs to be collision free and the robot also needs to be able to detect and avoid the obstacles. Lastly the path should be smooth in order to satisfy the kinematic constraints. The path should not have a sharp turn that is impossible for the robot to turn in smooth movement. However, the optimal path is normally a compromise among the three requirements.

In a known environment, there are well known and widely used methods for path planning such as roadmap approaches, cell decomposition methods and potential field methods.

2.1.1 Roadmap path planning

The roadmap path planning is based on connectivity in a network of robot's free space by using lines. Once the roadmap has been constructed, the path is determined by searching the series of road that are connecting the initial and final state. Visibility graph (Jiang *et al.*, 1997), Voronoi diagram and Visibility-Voronoi diagram are the well known roadmap approaches as shown in Figure 2.2. They have been used to compute the shortest collision free path. In this approach, the obstacles are represented by convex polygons. Then every two nodes between initial state and goal state in this free space are connected by line and this line does not intersect the interior of the obstacles. Visibility graph consist of straight lines that join all the polygons' edges including the initial and final points.



(a) (b)

Figure 2.2 Roadmap approach (a) Visibility Graph (Jiang *et al.*, 1997). (b) Voronoi diagram (Siegwart and Nourbakhsh, 2004).

Jiang *et al*. (1997) presented three stages to solve the time-optimal problem by using visibility graph. Firstly, the reduced visibility graph is obtained. Then it is converted into a feasible reduced visibility graph accounting the robot size and kinematic constraints. Lastly, a new algorithm is used to search the feasible reduced visibility graph in order to obtain a safe, time-optimal and smooth path. They have used an A* algorithm to search the shortest path. However, this method only considered kinematic constraints, but not dynamic constraints such as the velocity of the mobile robot. The dynamic constraints are important to be considered as the mobile robot may need to slow down during turning and accelerate as fast as possible during moving at the straight line.

Sridharan and Priya (2004) presented a parallel algorithm for constructing the reduced visibility graph in a convex polygonal environment. They aimed to reduce the computational complexity and space and implemented the algorithm in FPGA. Their algorithm consists of two steps. Firstly, binary code is assigned to the vertices of the objects to determine supporting segments between every pair of polygon. Then the next step is to eliminate the supporting segments that are hidden by the obstacles in order to obtain the final graph. From the results, the hardware-based approach is approximately 1000 times faster than using a PC. However, the major drawback of this visibility graph approach is that the path is very close to the obstacles and it is not practically safe in real applications.

On the other hand, Voronoi graph is able to overcome the problem caused by visibility graph aforementioned. Nagatani *et al.* (2001) proposed mobile robot navigation using generalized Voronoi graph (GVG). In the paper, they introduced a local smooth path planning algorithm for car-like mobile robot which is bounded by kinematic constraints. In addition, they used Bezier curve to generate a smooth path in order to satisfy the limitation of minimum turning radius. The algorithm is executed through simulations only and the computational time cost higher than the conventional approach. This means it takes more time to generate the path and it is not practical in the real-time control of the mobile robot.

Victorino *et al*. (2001) presented a new methodology for mobile robot navigation in unknown environments. Once the mobile robot started to move, it also started to construct the path using Voronoi diagram based on the information from the on-board sensor. From the results, the mobile robot was successfully constructed a map and localized itself. However, they had not discussed on the time required to construct the map and navigate to the goal point. Furthermore the map construction and localization is relevant to static environments only. Thus their method may not be appropriate to be used for time-dependent planning and in dynamic environments.

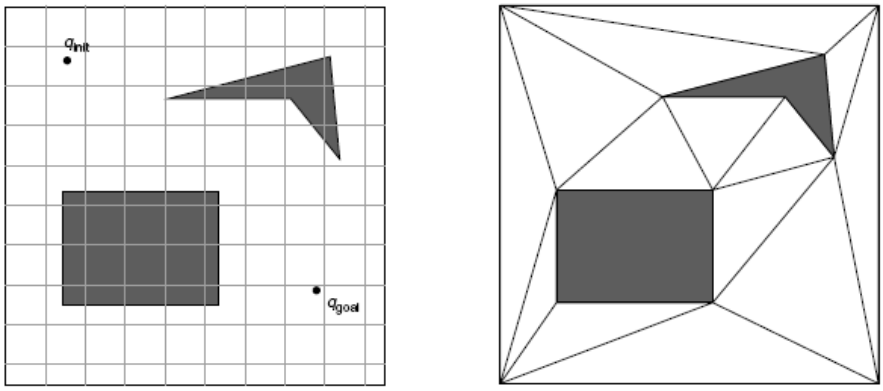
As the combination of Visibilty graph and Voronoi diagram may gives optimal path for mobile robots, Wein *et al*. (2007) introduced a new type of diagram which is a hybrid between the visibility graph and the Voronoi diagram. The aims were to find the smooth shortest path without sharp turns. This method was used for planning a

path for robots in an environment filled with polygonal obstacles. In order to keep the distance from obstacles optimum, they used predefined *clearance value, c*. In addition, they used Dijkstra search to find the shortest path. However, their method was only implemented for a robot with two degrees of motion freedom. Furthermore, Voronoi diagram tends to maximize the distance between the robot and the obstacles, in order to provide more space and safety to the robot.

Roadmap path planning approaches such as visibility graph and Voronoi diagram are effective to be used to obtain a safe path and the shortest path. The approaches used the information from map such as the shape of the obstacles to generate the path. However the mobile robot tends to make a sharp turn and move very close to the obstacles. These situations are not appropriate for a car-like robot that has a steering angle limitation.

2.1.2 Cell decomposition path planning

In cell decomposition approach, the robot's free space is divided into several simple, connected regions called "cells". There are several types of grid that normally used such as fixed-resolution grid and triangulation grid in order to construct the cells as shown in Figure 2.3. Then the cells containing the initial and goal states are located and path in the connectivity graph is searched to join the initial and goal cell.



(a) (b)

Figure 2.3 Cell decomposition method (a) A fixed-resolution grid. (b) A triangulation (Ge and Lewis, 2006).

Hazon and Kaminka (2008) presented new multi-robot coverage algorithms in their paper. Their algorithms are based on spanning-tree coverage of approximate cell decomposition of work-area and have achieved a significant improvement in coverage time by improving the efficiency of the algorithms. However, they have not mentioned the type of robot which has been used in their simulation and the algorithms were only tested by simulation works. Furthermore, the algorithms work efficiently in obtaining the optimal coverage time but not time dependent.

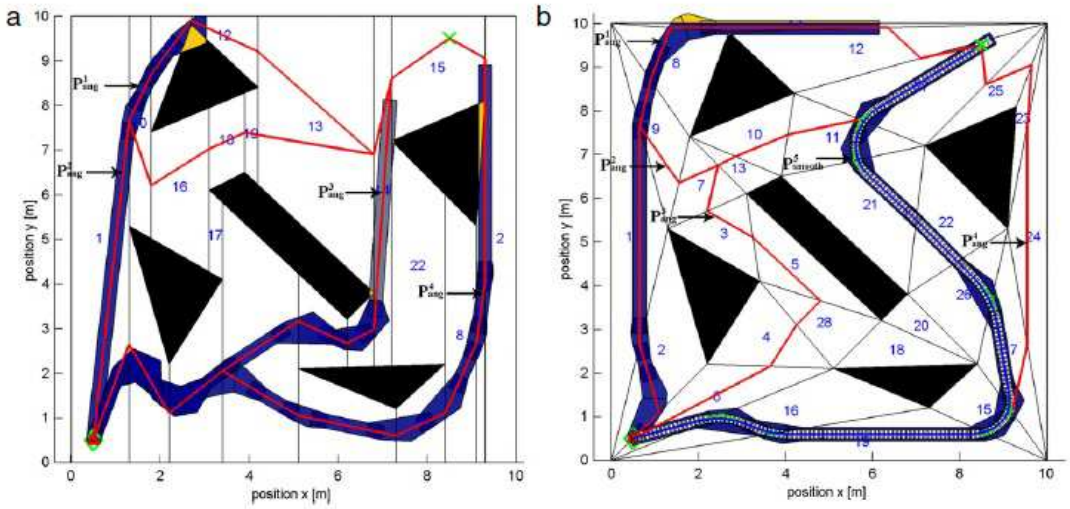


Figure 2.4 Simulation results by using (a) trapezoidal decomposition and (b) triangular decomposition (Ghita and Kloetzer, 2012).

Ghita and Kloetzer (2012) proposed a fully automatic planning and control strategy for a car-like robot based on cell compositions approach. The approach used an abstraction of the free environment and an iterative procedure to find a feasible path for the nonholonomic mobile robot. The planning and control method was developed in Matlab and the feasible and smooth path was obtained as shown in Figure 2.4. However, from the results, the generated path was closed to obstacles and collision may occur between the mobile robot and the obstacle.

2.1.3 Potential field path planning

The most widely used method for collision free path planning is the potential fields methods (Huang *et al.*, 2006; Safadi, 2007; Huang, 2009). It was initially proposed by Khatib in 1986 for mobile robot path planning. The main aspects of this method are

the mobile robot is treated as a point, the obstacle generates a repulsive force and the goal generates an attractive force. The attractive force lead the robot to the goal and the repulsive force ensures the robot is away from the obstacles as shown in Figure 2.5. The generated repulsive force also increases proportionally with the distance of the nearest obstacles. Thus the combined force should drive the mobile robot towards the goal while avoiding the obstacles.

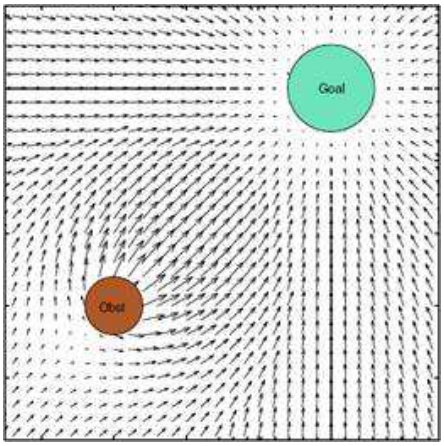


Figure 2.5 Potential field method (Safadi, 2007).

Cosio and Castaneda (2004) proposed an improved artificial potential field method for autonomous navigation of a mobile robot. In the paper, they attempted to overcome the problem that caused by using a single attraction point which lead to trap situation where the method is unable to produce the resultant force needed to avoid the large obstacles. Therefore, they introduced multiple auxiliary attraction points that allow the robot to avoid large or closely spaced obstacles. The force intensity parameters of the repulsive and attractive cells have been optimised by using a genetic algorithm. From the simulation results as shown in Figure 2.6, the generated path was not too smooth and tends to make sharp turns. Furthermore, the algorithms were tested only in Matlab and the authors have not discussed the time required for a mobile robot to reach the final point.

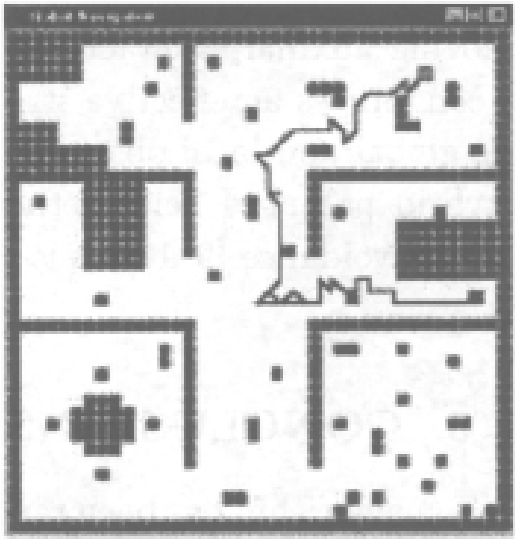


Figure 2.6 Path generated by the navigation algorithm (Cosio and Castaneda, 2004).

The earlier works on path planning using potential field method only concentrated on static environments. In the recent years, dynamic obstacles have also been included in navigation planning. Ferrara and Rubagotti (2009) proposed a dynamic obstacle avoidance strategy for a mobile robot based on harmonic potential field method. Their approach consists of two key elements which are an online generator is used to track the reference signals to reach the goal point and at the same time, a potential field method is modified online in order to avoid the moving obstacles with time-varying speed. In addition, they used a collision cone approach to avoid the moving obstacles. The key idea is to modify the radius of the 'security circle' around each obstacle on the basis of the so-called 'collision cone'. However, their proposed strategy was to control the mobile robot but not to generate the path. Furthermore, they only tested their approach by simulation works.

Jacob (2008) proposed a sensor-based navigation and obstacle avoidance algorithm for mobile robots in unknown dynamic environments. The proposed method allows a mobile robot to navigate in the environment with a large number of static and dynamic obstacles. The mobile robot will navigate through the environment via the global path which was generated based on the updated map which processed by the global planner. Then the local planner continuously tries to reach each waypoint on the path using potential field. However, their algorithm only tested by simulation works and they have not discussed the time required to reach the final point.

Furthermore, from their simulation works, they encountered several failures in the simulation such as the rear-end collision occurred due to the blind spot of the laser scanner.

Huang *et al*. (2006) proposed a method which combined a single camera and potential field method in order to navigate in real-time environment. The camera is used to estimate the "time of impact" once the obstacle is detected which then can be used to make sure the robot navigates around the obstacle. Furthermore, Huang (2009) has extended the work to deal with the dynamic obstacles. Using the same method – potential fields – Huang has applied this method for path and speed planning in order to avoid the moving obstacles. Their approach provides both the direction and the speed of the mobile robot, which guarantees that the mobile robot will able to track the moving obstacle while avoiding it. However, their algorithms only tested in the simulation and they have not discussed the time require to avoid the obstacle and reach the final point.

Beside a potential field method, a vector field method is also has been used in robot navigation. The vector field utilizes a statistical representation of the environment through the histogram grid and it consists of attractive forces, goal and repulsive forces. Both attractive and repulsive forces are usually characterised as point forces. Hong *et al*. (2007) proposed a mobile robot navigation using modified flexible vector field approach with laser range finder and infrared sensors. The laser range finder is used to generate the map and infrared sensors are used for emergency stop and obstacle avoidance. From the results, their algorithms show a smooth motion of the mobile robot navigates through the environment. However the proposed method only demonstrated in static environments and the speed of the mobile robot was set to 70cm/s only which is not optimized for the robot's motion. The mobile robot may need to speed up at the straight line and slow down at cornering. Furthermore the authors have not discussed on the time require for the mobile robot to reach the final point.

Liddy and Lu (2007) proposed waypoint navigation for an Ackermann steering autonomous vehicle. Their aim is to obtain a path with position and heading control of the mobile robot. They have introduced a complex vector field method by combining

vector field components such as point force vector field, rotational field and line force. The results successfully demonstrated the position and heading can be controlled at the goal point. However, the authors have not discussed on the time require for the mobile robot to reach the final point and the algorithms were only tested by simulation works.

Potential field method is one of the commonly used approaches to generate path for the mobile robot. The method has been utilised for various types of mobile robot such as the differential drive robot and the car-like robot with Ackermann steering limit. One of the problems in potential fields method is the robot is intended to converge in the local minima (Huang, 2009). Furthermore, most of the research in potential field approach have not addressed the time require for the mobile robot to reach the final point. This parameter is one of the important points for the time-critical motion planning.

2.1.4 Other path planning approaches

There are other approaches which have been developed by researchers in order to obtain the optimal collision free path. The approaches could be a combination of two different approaches, or sampling-based path planning. Koh and Cho (1999) presented a path tracking algorithm for a nonholonomic mobile robot in order to obtain a smooth motion of the mobile robot. This algorithm is based on time optimal bang-bang control considering dynamic constraints of the mobile robot in order to avoid the wheel slippage problem during the mobile robot navigation. Figure 2.7 shows the flow chart on implementing the proposed algorithm. In their experiment, they have used a two-wheel driven mobile robot to validate their proposed algorithm. However, their approaches only focused on obtaining a smooth motion without the consideration of avoiding obstacles.

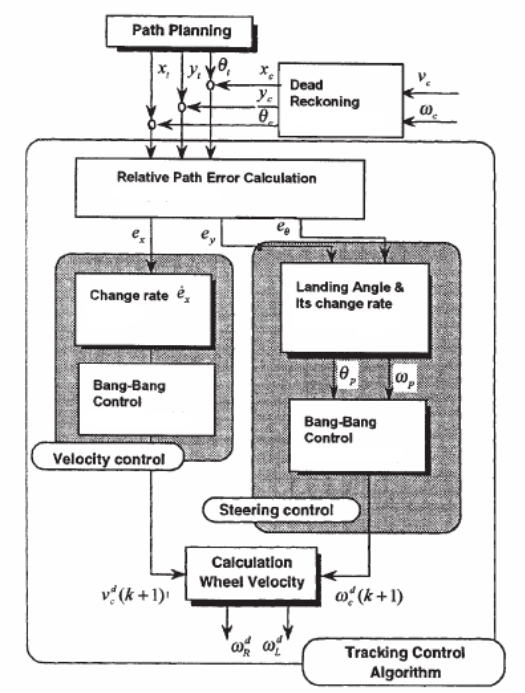


Figure 2.7 Implementation of the proposed algorithm by Koh and Cho (Koh and Cho, 1999).

Mihaylova *et al*. (2003) presented an information-based approach for trajectory optimization of a mobile robot by a linear combination of sine functions. The mobile robot was equipped with a sensor which measures the range and bearing to a beacon located at a known coordinate. The information acquired from the sensor will then be used to obtain an optimal trajectory based on a known, nominal reference trajectory. The accuracy of this approach depends on the number of beacons available in the environment. If there are more beacons at the appropriate places, the accuracy can be improved considerably. However, the effectiveness of this approach is only demonstrated by simulation results as shown in Figure 2.8. An experiment using this approach would be useful to validate the optimization effectiveness.

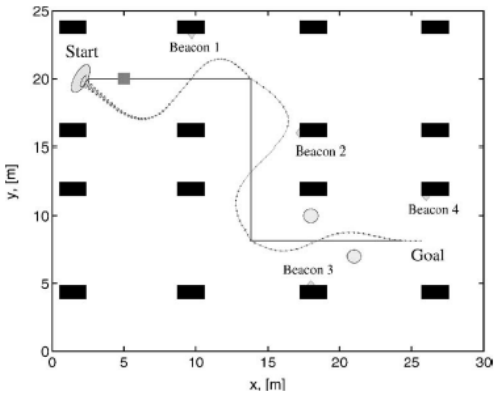
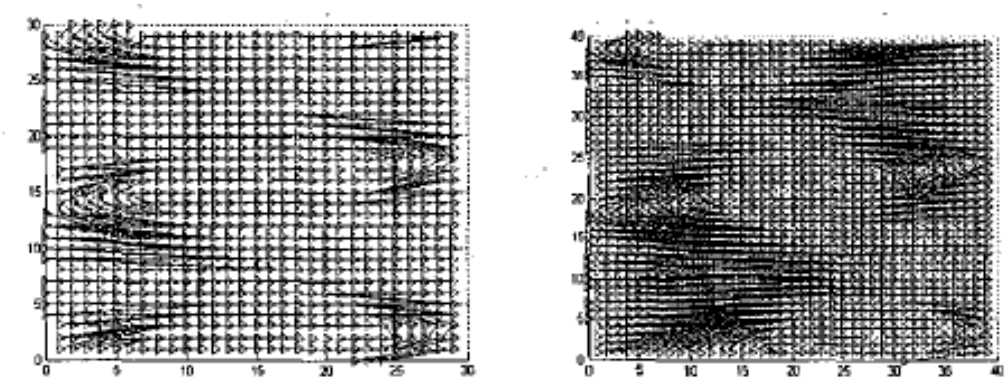


Figure 2.8 Results from the information- based method (Mihaylova *et al.*, 2003)

A new approach using the cell-mapping method was introduced by Li and Wang (2003) as shown in Figure 2.9. Their aim was to achieve the optimal trajectory in term of minimum time, energy and jerk. Firstly, this approach performs a global analysis and reconstructs the whole system into a cell space model. Then, based on this cell space model, this method finds out the stable region as a set of cells in the cellular state space after a number of integration processes to generate the optimal trajectory. In their study, they used a four-wheeled mobile robot with dynamic constraints such as velocity and acceleration limitations. However, this method was only tested in simulation works and the authors have not discussed on the obstacle avoidance approach.



(a) (b)

Figure 2.9 Cell mapping model (a) with 30^5 cells. (b) with 40^5 cells (Li and Wang, 2003).

In order to achieve the time-optimal planning for the wheeled mobile robot, Prado *et al*. (2003) proposed two tasks that can be carried out simultaneously or sequentially. The first task is spatial-planning which is to obtain the shortest feasible geometric

path. The second task is temporal-planning which is to obtain the fastest feasible velocity profile for a homogenous segment which the segment is the path length navigated over time. They also considered kinematic and dynamic constraints such as velocity and acceleration in order to get the optimal trajectory solution and to avoid the obstacles in dynamic environments. To validate their algorithm, they used a four-wheeled mobile robot which is known as RAM in their experiment and the results are shown in Figure 2.10. However, from their results, the mobile robot moved very close to the obstacles and the mobile robot tends to make a sharp turn. Furthermore, the authors have not discussed the time require for the mobile robot to reach the final point.

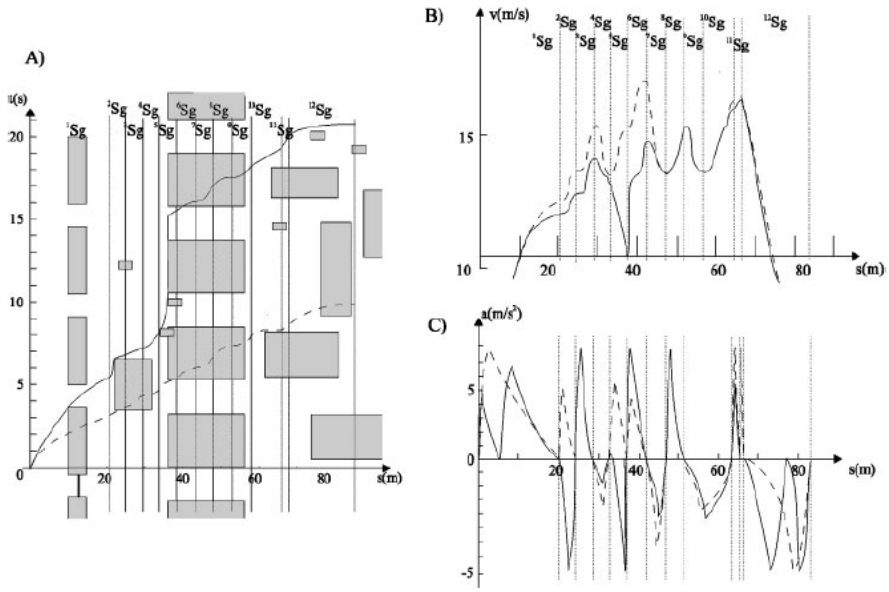


Figure 2.10 (a) Generated trajectory (b) Velocity profile (c) Acceleration profile (Prado *et al.*, 2003)

Then, Hui *et al.* (2006) presented a time-optimal, collision-free navigation of a car-like robot using neuro-fuzzy-based approaches as shown in Figure 2.11. In their paper, a fuzzy logic controller (FLC) was used to control the robot. The performance of the controller was improved by using three different neuro-fuzzy-based approaches, which are neuro-fuzzy approach, genetic-neuro-fuzzy approach and GA-tuned adaptive network-based fuzzy inference system (ANFIS), and then comparing among themselves and with other approaches such as default behaviour, manually-constructed FLC and potential field method, through computer simulation. From their results, even though the performance using neuro-fuzzy-based approaches is better

than other approaches, it is dependant on the training data. This condition caused the performance of the neuro-fuzzy-based approaches not to work well, particularly when the training scenarios are different from the real scenarios.

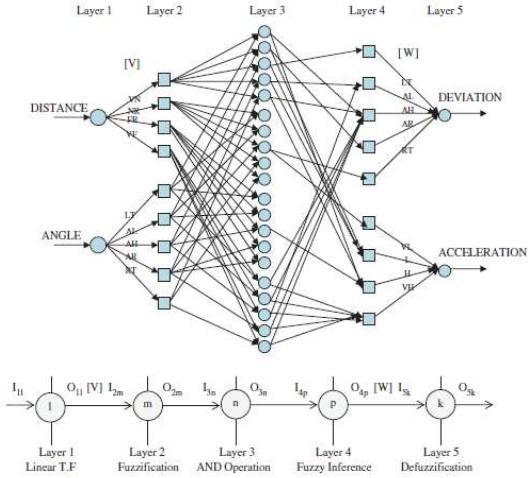
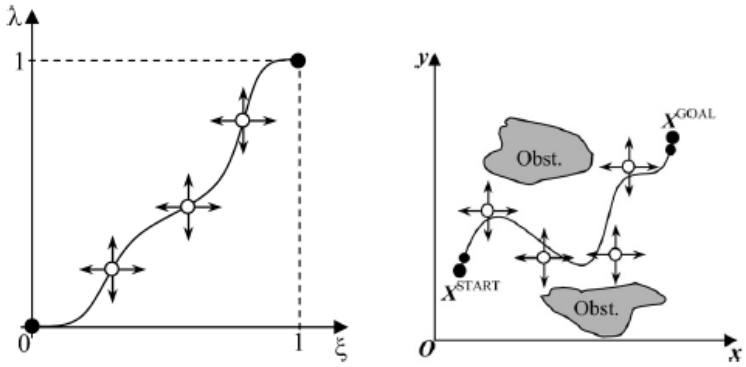


Figure 2.11 Neuro-fuzzy approach (Hui *et al.*, 2006).

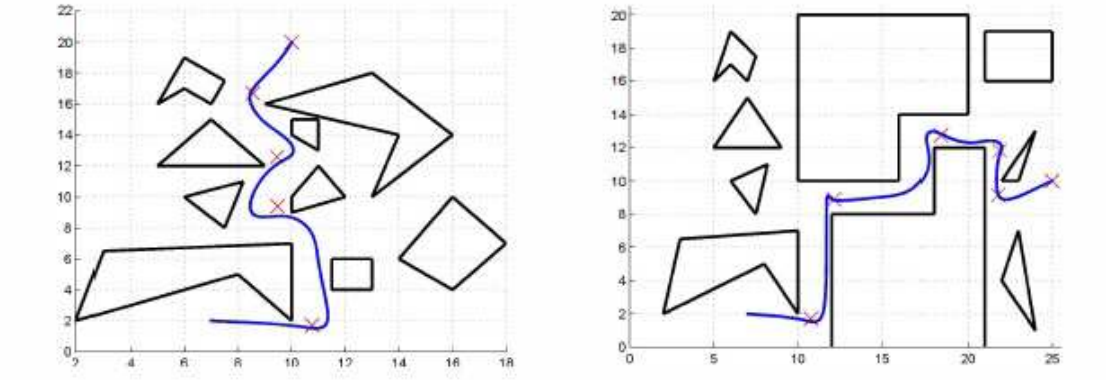
In 2007, Haddad *et al*. (2007) presented a random-profile approach in order to optimize the free-trajectory planning problem for non-holonomic wheeled mobile robots in constrained workspaces as shown in Figure 2.12. This method is based on a simultaneous search for the mobile robot path and also handles the obstacle avoidance issues during navigation. In their paper, they focused on the planning the trajectories for the mobile robot with the consideration of geometry, kinematic and dynamic constraints. However their results are presented using only two- and three-wheeled mobile robots. It remains to be seen that their works are able to be extended to the four-wheeled mobile robot. Nevertheless, the algorithm may require to be modified in order to cater the kinematic and dynamic constraints of the four-wheeled mobile robot.



(a) (b)

Figure 2.12 Generated trajectory with several control points (Haddad *et al.*, 2007).

Ma *et al*. (2013) presented a path planning algorithm for a nonholonomic mobile robot using the information of the sensors to navigate in complex environments. The robot moved toward a known target while avoiding obstacles by choosing appropriate intermediate objectives based on the local sensor information. In addition, by choosing intermediate objectives, a local minima problem can be solved. The efficiency of the approach was assessed via different simulated environments as shown in Figure 2.13. From the results, the robot was able to navigate trough the complex environments. However, the robot's path was closed to the obstacles and the robot was likely to make a sharp turn as in Figure 2.13(b).



(a) (b)

Figure 2.13 Simulation results in (a) a complex scenario, and (b) a long corridor (Ma *et al.*, 2013).

2.1.5 Geometric approach for trajectory planning

A trajectory is a path which is an explicit function of time. Initially a path can be differentiated to give a continuous velocity and acceleration profiles. One common methodology for trajectory planning in order to obtain a smooth-path and length-optimum plan is by assembling the arcs of simple curve. A mobile robot has to follow the path (curve) with specific velocity which is dependent on its position and its orientation (Tounsi and Corre, 1996). Basically, the orientation (θ) is defined as the tangent of the point ($x(s), y(s)$), which s is the length along the curve. The curvature κ is defined as the derivative of θ(s) with respect to s.

$$\theta(s) = \tan^{-1}\left[\frac{dx}{dy}\right], \qquad \kappa(s) = \frac{d\theta(s)}{ds} \tag{2.1}$$

Tounsi & Le Corre (1996) reviewed and compared several types of curves used in path generation, which are straight lines, circular arcs, polynomial functions, clothoids (cornu spiral) and cubic spirals. Generally, the path is generation by a set of robot's postures, which these postures depend on the position and orientation of the mobile robot (Shin and Singh, 1990). They also discussed the methods to generate the path as shown in Figure 2.14.

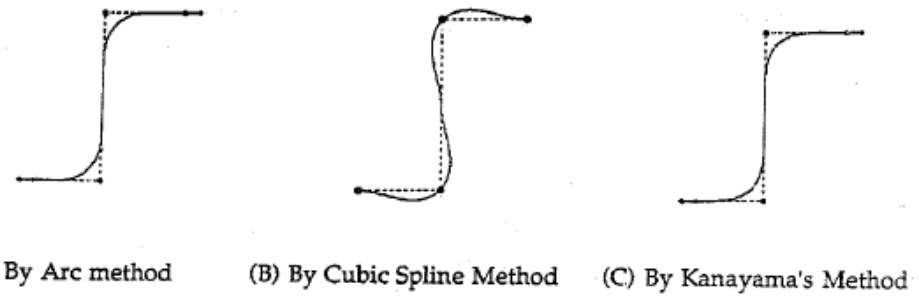


Figure 2.14 Different types of curves used to connect four postures for path generation (Shin and Singh, 1990).

The path generated by several straight lines is the simplest method in terms of calculation and requires only the choice of intermediate points. However, in most cases, the orientation is discontinuous and the mobile robot needs to stop and change its direction in order to move to the next point. Similarly in the path generation by following circular arcs of radius R, the drawback is that the path presents

discontinuous curvature at junction points, which means the speed of each wheel of the mobile robot is not continuous at these points.

In order to avoid the discontinuous curvature, polynomial curves were used. There are three different types of polynomial curves discussed by Tounsi and Le Corre (1996), which are polar polynomials, Cartesian polynomials and Bezier's polynomials. Even though the polar polynomial method gives a continuous curvature, the radius *R* must be fixed and it is only used for symmetric cases. Both Cartesian and Bezier's polynomials are used to connect non-symmetric postures. However these curves have a complex curvature profile which is not necessarily smooth and makes them difficult to follow (Delingette *et al.*, 1991).

The other type polynomial curvature is known as polynomial spiral. There are two commonly used types of spiral curves which are clothoid curves and cubic spiral curves. In general, the polynomial spirals are useful for path generation because they provide an easy-to-track polynomial curvature profile (Liang *et al.*, 2005). In a review by Delingette *et al*. (1991), the original work by Kanayama (Kelly, 2003) on clothoid curves has introduced the idea of using continuous piecewise linear curvature function that was then extended by Shin and Singh (Kanayama and Miyake, 1986) in order to eliminate discontinuity at the junction points. However, the problems with this method are difficult to choose the coefficient of the curvature (*k*) (Tounsi and Corre, 1996), difficult to compute (Delingette *et al.*, 1991) and it still results in a discontinuity in the derivative of the curvature (Nagy and Kelly, 2001). Thus, a study by Pin and Vasseur (1990) considered the problems of complexity and lengthy path using clothoid curves by generating deterministic and providing trajectories joining all the pairs of configurations of the mobile robot. Their aim was to determine the shortest path with reverse mode capabilities while the mobile robot is manoeuvring by considering non-holonomic and steering angle constraints.

Table 2.1 Intrinsic splines' family (Delingette *et al.*, 1991)

Constraints	Intrinsic Spline	Constraints	Polynomial Spline
(P_0,ϕ_0) (P_1,ϕ_1)	IS1 (Clothoids)	(P_0,P'_0) (P_1,P'_1)	Cubic Spline
(P_0,ϕ_0,k_0) (P_1,ϕ_1,k_1)	IS3	(P_0,P'_0,P''_0) (P_1,P'_1,P''_1)	Quintic Spline
(P_0,ϕ_0) (P_1) .... (P_{n-1}) (P_n,ϕ_n) (ϕ, k) continuous	Piecewise IS1	(P_0,P'_0) (P_1) .... (P_{n-1}) (P_n,P'_n) (P', P'') continuous	Piecewise Cubic Spline
(P_0,ϕ_0,k_0) (P_1) .... (P_{n-1}) (P_n,ϕ_n,k_n) (ϕ, k,k', k'') continuous	Piecewise IS3	(P_0,P'_0,P''_0) (P_1) .... (P_{n-1}) (P_n,P'_n,P''_n) $(P', ..., P^{(4)})$ continuous	Piecewise Quintic Spline

Most studies have used cubic spiral curve (Nagy and Kelly, 2001; Kelly, 2003; Liang *et al.*, 2005) in path generation because it provides a smooth path and minimizes the variation of jerk (Delingette *et al.*, 1991). In addition, it also has been used due to its simple curvature profile which is easy to follow. Later, Delingette *et al.* (1991) developed a family of trajectory called *intrinsic splines* of degree n, ISn as shown in Table 2.1. This family is based on cubic polynomials, but the end conditions of this family are defined in term of heading and curvature instead of first and second derivative for cubic polynomial. Nagy and Kelly (2001) extended the work done by Delingette *et al.* (1991). In comparison to Delingette *et al.*, the approach is gained by converting the integro-differential state equation into four nonlinear equations and solving them simultaneously in order to get the four unknown constant parameters. Subsequently, Kelly (2003) extended the work done by Nagy and Kelly (2001) by introducing an approach which produced an efficient real-time algorithm to join arbitrary points. However, most of the researchers have switched the specification of the trajectories in term of time to distance, which suits most of application but not the time-critical application targeted in this research.

Liu and Sun (2011) presented an optimal path planning of a mobile robot by utilizing Bezeir curves. The objective of their approach was to minimize energy consumption during robot navigation. The energy consumption was analysed for both in geometric path planning and smooth path planning. The effectiveness of the approach has been tested in the simulation and experimental works. The results of their works are shown in Figure 2.15 and the experiment was conducted using two-wheel mobile robot. The results show an optimal path in term of minimum energy, minimum travel distance and minimum travel time. This approach can be adopted in this study to minimize the energy consumption and at the same time to reach the final point at the specified travel time.

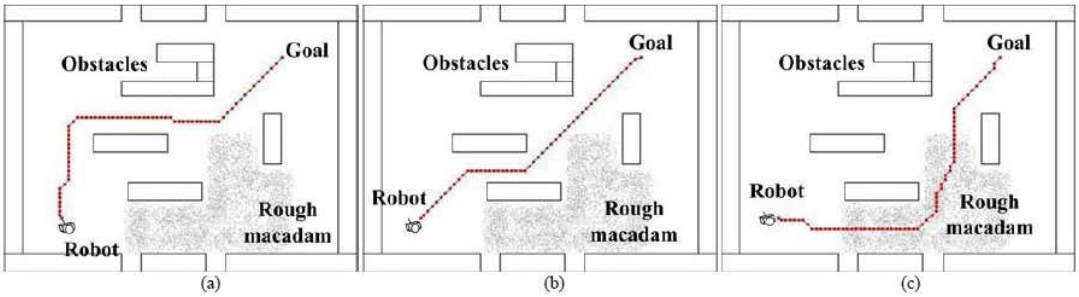


Figure 2.15 An optimal path (a) minimum energy, (b) minimum travel distance, and (c) minimum travel time (Liu and Sun, 2011).

2.2 Navigation environments

Mobile robots are being deployed in various types of environments. Some of them are tasked to navigate inside the buildings and others outside the buildings. Outdoor navigation poses a greater challenge over typical indoor navigation. Outdoor environments are usually dynamically changed over time and give uncertainty to the mobile robots. Such environment, so-called dynamic environment may consist of static and moving obstacles. Static environments normally have unmoved obstacle with various shapes and sizes. Thus, static environments are not as complicated as dynamic environments in term of planning the path.

In the previous sections, the standard path planning approaches, such as roadmap, cell decomposition and potential field methods, have been utilised whether in static or dynamic environments. However these standard approaches have not been proven to be effective in unknown environments. Due to the uncertainty of the unknown

environments, some approaches such as grid-based or roadmap-based approaches cannot generate an optimal path. Furthermore, local information is required to detect and avoid the unexpected obstacles. Thus some of the researches have developed the alternative approaches by modifying and improving the existing approaches or with combining two or more existing approaches to overcome the limitations of the existing approaches.

2.2.1 Outdoor navigation

In outdoor navigation, the robot will face a new challenge especially due to numerous uncertainties and dynamic changes in the outdoor environment such as varying terrain surface and level, and also lighting condition as shown in Figure 2.16. A robust outdoor navigation system will improve the autonomy of the robot and provide a safe and smooth navigation to reach the final location. In order to obtain a safe and smooth path, most researchers consider the moving obstacle's velocity as known to the system. With the knowledge of the moving obstacle's velocity, the system is able to predict the moving obstacle's motion and probability the collision between the mobile robot and the moving obstacle. If the mobile robot is indisputably to collide with the moving obstacle, the mobile robot is able to avoid the moving obstacle by adjusting its path. However in the real-world, it is difficult to distinguish the velocity of the moving obstacle beforehand. This circumstances may fall short the system.

(a) (b)

Figure 2.16 Outdoor navigation (a) Pioneer3-AT with URG and SICK (Chang *et al.*, 2009) (b) The Cycab used in the experimental works (Zhang *et al.*, 2006).

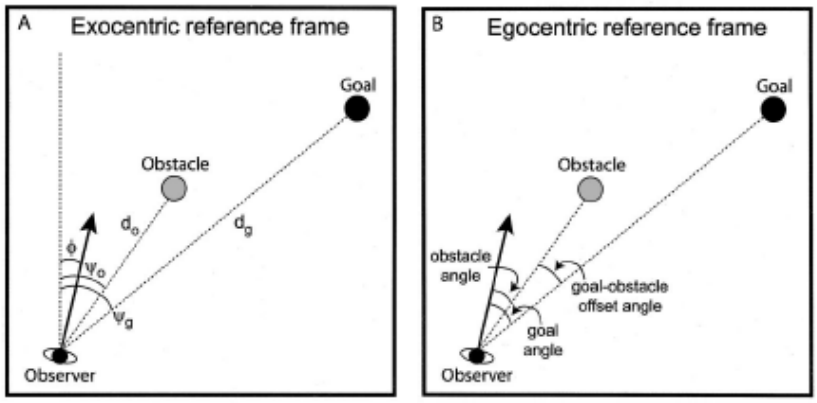
Furthermore, the unknown environment gives a further challenge to the system. The uncertainty of the information in the environment leads to the needs of a better detection and prediction approaches in order to make sure the smooth and safe path requirements are met. However in certain cases, the map of the area that the mobile robot needs to navigate is available. This map may give some information to the robot planner. By utilising this information, the planner is able to plan the path beforehand. Thus a good outdoor navigation system is still required in order to ensure the mobile robot navigates and reaches the final location safely. Therefore, many studies are attempted to develop a new and better navigation system in a dynamic and unknown environment.

2.3 Obstacle avoidance

Avoiding obstacles is one of the problems for the mobile robot to navigate in static and dynamic environments. In dynamic environment, where there are static and moving obstacles, the task becomes more complicated and difficult in comparison to static environments. Therefore, many approaches have been introduced in previous research in order to develop an effective and reliable obstacle avoidance capability for mobile robots to navigate in static and dynamic environments.

Fajen and Warren (2003) introduced a new solution for obstacle avoidance based on observing the human behaviour in dynamic environments. In their paper, the aim is to apply the dynamic model to the robot behaviour of steering towards a goal and avoiding the obstacles. Once the set of behaviour variables for steering and obstacle avoidance have been identified, the general form of the model will be introduced. The basis of their work is shown in Figure 2.17. In Figure 2.17(a), the authors considered an observer moving in a simple environment. The observer moves at a constant speed (s) and a heading direction ($ø$) with respect to fixed vertical exocentric reference axis. In Figure Figure 2.17(b), the goal and obstacle angles can be represented in egocentric reference frame with respect to the observer's point of view. In order to model their approach, they have used human as participants to observe the behaviour during walking from initial point to final point as well as during avoiding the obstacle. The collected descriptive data were then being used to develop a model of the behavioural dynamics. This work has been extended by Fajen *et al*. (2003) by using visually-

guided locomotion in a dynamic environment in order to identify a set of behavioural variables for steering and obstacle avoidance. However, the behavioural approach requires human experiments in prior in order to develop a model of behavioural dynamics. This model is directly influent by the behaviour of human at the time of the experiments is conducted that may lead to inaccuracy of the model. Nevertheless, from their experiment results, it was suggested that human route selection does not require explicit planning but may emerge on-line as a consequence of elementary behaviours for steering and obstacle avoidance.



(a) (b)

Figure 2.17 Plan view of the observer moving in dynamic environment (a) Exocentric reference frame (b) Egocentric reference frame (Fajen and Warren, 2003).

The most commonly used method for solving the obstacle avoidance problem is based on the potential field method, firstly proposed by Khatib (1986). Then Huang *et al*. (2006) proposed a vision-guided navigation approach by adapting Fajen and Warren's work on human behaviour navigation and this approach was expressed as a potential field. In their study, the potential field is used to control the angular acceleration and heading of the robot in order to steer it toward the goals and to avoid the obstacles during robot navigation. However, this approach has a limitation since they used angular width of the obstacle rather than distance, yet a large obstacle can also has the same angular width as a smaller obstacle.

Furthermore, Hamner *et al*. (2006) also proposed an extension method based on Fajen and Warren formulation. The proposed method can learn the parameters of the control

model automatically by observing behaviour of the human driver. In addition, Hamner *et al*. introduced a speed control function based on the obstacle's distance and angle in their method. This speed control function slows down the vehicle as the obstacles get closer, which gives time to the vehicle to turn and avoid the obstacles. However this method also allows a sharp turning which has a negative impact for the vehicle motion. Moreover, their results showed that the vehicle attempts to follow a far path while avoiding large obstacles and gave conservative results.

The other method to solve the problem of obstacle avoidance was proposed by Brock and Khatib (1999) using global dynamic window approach. In their paper, the global dynamic window approach used for motion planning is an extension of the dynamic window approach (Fox *et al.*, 1997) by incorporating a simple and efficient motion planning. This framework allows robust execution of high-velocity, goal-directed and reactive motion for a mobile robot in unknown and dynamic environments. However the approach was used for a holonomic mobile robot, not for non-holonomic mobile robot as targeted in this study.

Castillo *et al*. (2006) proposed an approach that using sonar detection for detecting the obstacles. Sonar was used in the research due to it provides a consistent data and it can simply detects "something" in the environment. From their results, the sonar sensor was capable to detect obstacles and ensure the wheelchair as able to navigate safely. However, they applied this approach only for an autonomous wheelchair, used in an indoor environment, which can be extended to an outdoor environment.

Recently, Jolly *et al*. (2008) proposed a method for avoiding the dynamic obstacle by modifying the initial generated Bezier curve. At the initial stage, the robot will travel along the original curve. Once an obstacle is detected, a new modified Bezier curve will be generated. This approach is shown in Figure 2.18. In their simulations, a holonomic mobile robot is used but the idea of the obstacle avoidance approach can be adopted for this study regardless the type of the curve used.

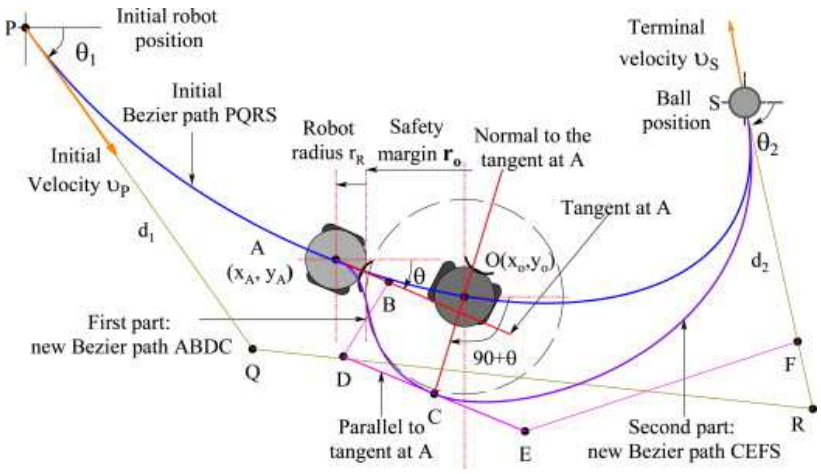


Figure 2.18 Avoiding a dynamic obstacle (Jolly *et al.*, 2008).

2.4 Multiple robots coordination

There have been many studies on using multiple robots to achieve a task given. Using a group of robots instead of single robot in task-based mission has a few advantages such ability to complete the task faster, more robust, ability to locate the goal position more accurate and ability to complete the task that by using a single robot cannot be achieved. Some of the applications using multiple robots are exploration of hazardous environment, search and rescue, autonomous construction, hunting operations and soccer robot.

Controlling a group of robots may require a significant control law of motion coordination. Yamaguchi (2003) presented a distributed motion coordination strategy for multiple robots in cooperative hunting operations as shown in Figure 2.19. Each robot in this control law has its own coordinate system and it can sense the target, other robots and obstacles. This control law is based on “formation vector” strategy as an input. The formation of each robot is controllable by the vectors.

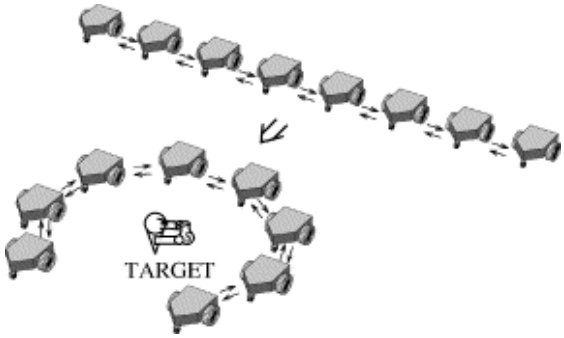


Figure 2.19 A group of robots in hunting operation (Yamaguchi, 2003).

Stroupe *et al.* (2005) presented a behaviour-based multiple robots collaboration for autonomous construction tasks. In the paper, two robots are used to form a team for the construction tasks. The construction task consists of several subtasks which are shown in Figure 2.20. Each robot will perform the subtasks at every stage in order to achieve the goal.

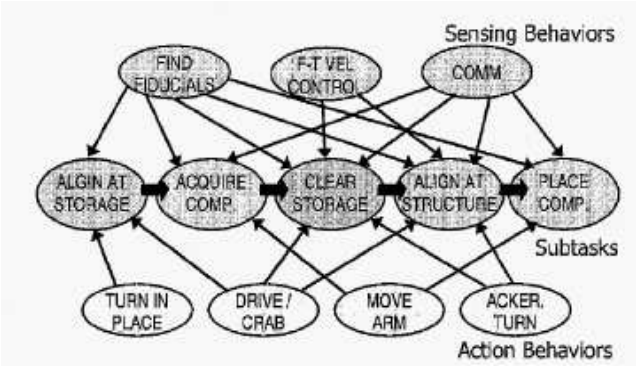


Figure 2.20 Subtasks of construction task (Stroupe *et al.*, 2005).

In soccer robot system, most of the cooperative strategy is based on vision system. The global vision system is used to track the position and orientation of the robot (Klancar *et al.*, 2004; Brezak *et al.*, 2008). Klancer *et al*. (2004) has used a robot with colour patch on the robot. In order to estimate the robot position, patches and the regions belonging to the ball, opponent team patches have to identify. Then the position of the robot can be located by using image segmentation and component labelling. Figure 2.21 shows the overview of the system. Then Brezak *et al*. (2008)

used the same approach in their paper. However they have used Bayer image format in order to interpret the position of the robots.

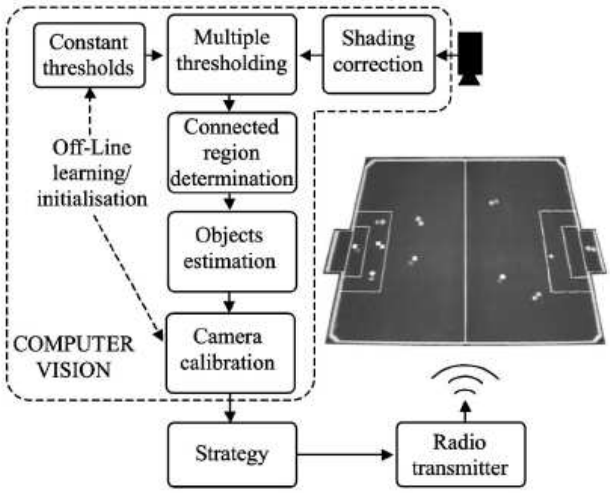


Figure 2.21 Overview of the system (Klancar *et al.*, 2004).

Other approaches for soccer robot system without using colour information are by using shape information (Treptow and Zell, 2004), artificial neutral networks (Jolly *et al.*, 2007) and reinforcement learning (Duan *et al.*, 2007).

2.5 Summary and gap statement

From the literature, most research focused on obtaining an optimal motion planning in terms of safe navigation, smoothness path, shortest path and optimal time motion plan for the mobile robots. Even though there are studies on shortest path and optimal time motion plan, the focus is only on how to reach the desired location as soon as possible. This means the mobile robot will reach the desired location in minimum or optimal time. However, there are situations that timing of reaching the desired location can be crucial especially when dealing with the multiple mobile robots coordination.

There are many methods to obtain the smooth trajectories. The common approaches such as roadmap approaches, cell decomposition and potential field method are not the best approaches to achieve the aims of this study. These approaches are usually used for holonomic robots as they tend to require sharp turns. Furthermore integration with time parameter might be difficult to be performed due to these approaches are

typically to generate a path for the mobile robot. Thus the best method is by using a geometric approach as this approach can be developed in term of time and due to its simplicity and flexibility of geometric profile. In the geometric approach, the simplest method to generate a path is by assembling the arcs of simple curves. The commonly used types of curves are clothoid curves (Kanayama and Miyake, 1986; Pin and Vasseur, 1990; Delingette *et al.*, 1991) and cubic spiral curves (Nagy and Kelly, 2001; Kelly, 2003; Liang *et al.*, 2005). However, there are drawbacks using clothoid curves such as it results in a discontinuity in the derivative of the curvature. Therefore, cubic spiral curves are adopted instead of clothoid curves in this study. This is because the cubic spiral curve provides a smooth path, minimizes the variation of jerk and is a simple curvature profile to follow. In addition, Tounsi and Le Corre (1996) introduced a variable velocity function in order to minimize the jerk problem and to obtain smooth trajectories. However, the reviewed research proposed the algorithm for cubic spiral curves in terms of distance rather than time. In contrast, timing to reach the desired location is more important rather than distance in certain situation such as for the task-based missions.

The research based on human behaviour observation in dynamic environment has been carried out in order for the robot to avoid the obstacles while navigating (Fajen and Warren, 2003; Fajen *et al.*, 2003; Hamner *et al.*, 2006; Huang *et al.*, 2006). A camera was used by Fajen *et al*. (2003) and Huang *et al*. (2006) as a navigational aid for robot to avoid the obstacles. Huang *et al*. used the potential field approach to control the angular acceleration and heading of the robot. However, this approach gives conservative results as the robot attempted to avoid the obstacles by following the far path even though to avoid smaller obstacles. In addition, Hamner *et al*. (2006) introduced a speed control function, which slows down the robot as the obstacle gets closer and gives time to the robot to turn and avoid it. However, this approach allows the mobile robot to make a sharp while avoiding the obstacles which will give a negative impact to the robot's motion. Jolly *et al*. (2008) presented a method to avoid the obstacles by using Bezier curves. The idea is to get the control point in order to generate the Bezier curves for the new path which avoid the obstacles. This idea appears to be useful for this study.

So far, there is no depth research focuses on time-critical motion planning with obstacle avoidance capability for nonholonomic car-like mobile robots and on multiple robots which each robot has a different mission or objective in time critical environments. Therefore, the purpose of this study is to develop a time-critical motion planning for Ackermann-steering-like nonholonomic mobile robots with the capability of obstacle avoidance in static and dynamic environments. In addition, the developed algorithm will capable to plan the motion for different mobile robots from the different starting point to accomplish specific missions or objectives simultaneously.

At the end of this study, it is expected that the robot should be able to move from one location and reach the next one with the specified orientation, velocity and time with consideration of the kinematic and dynamic constraints such as maximum turning radius, maximum velocity and acceleration. Moreover, the robot should have a capability of planning the trajectory with known obstacles and re-adjust its trajectory while avoiding the detected obstacles, which are unknown to the mobile robot in order to catch-up with the time delayed due to avoiding the obstacles.

3. METHODOLOGY

The methodology developed for this study is driven by the research aims of this study. Basically, the methods are divided into four stages as shows in Figure 3.1. Each stage will briefly explain in the following subsections.

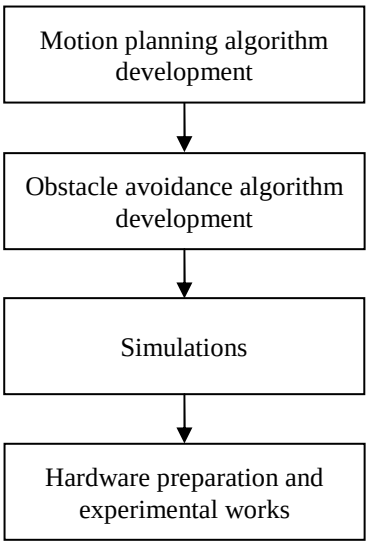


Figure 3.1 Stages for proposed methodology

3.1 Stage 1: Development of Algorithms for Time-critical Motion Planning

A new algorithm is to be developed through the use of mathematics for a time-critical motion planning with the consideration of position, orientation, velocity and timing. Geometric approach is adopted for generation of the trajectories. The types of curves that are used for the trajectory planning are cubic and quintic polynomials because they give smooth trajectories and they were derive from the kinematics and dynamic constraints, which will discuss in later chapter. In addition, the kinematic and dynamic constraints which are maximum turning radius and maximum velocity of the mobile robot are taken into the consideration during the development of this algorithm.

The development of these algorithms includes:

i. basic trajectory algorithm
ii. multiple waypoints planning
iii. multiple robots planning

3.2 Stage 2: Obstacle Avoidance Approach

Once the first stage has successfully been carried out, the second stage is to integrate the obstacle avoidance capability into the system. The steps for this algorithm are shown in Figure 3.2. At the beginning, the motion planning algorithm will generate a path despite the presence of obstacles. Then, when the mobile robot detects an obstacle, the safety margin and deviation point will be generated. This will give two options for the mobile robot, whether to turn right or left, which is depending on the current position and location of the obstacle in respect to the final point. Once the decision has been made, a new path will be generated to avoid the obstacle. Generally, during navigation, the robot will be capable of detecting and avoiding obstacles and re-adjust its original path once encounters the obstacle in order to catch-up the time delayed due to avoiding the obstacle. Finally, this algorithm will be extended to deal with both the unknown static and dynamic obstacles.

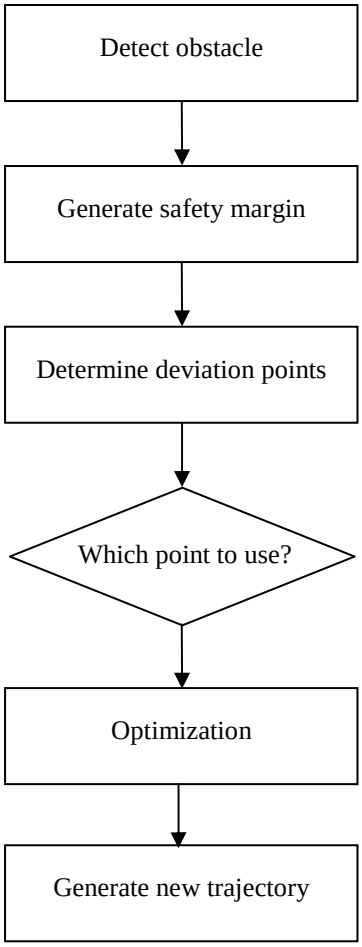


Figure 3.2 Generalized steps for avoiding an obstacle

3.3 Stage 3: Simulation Works

Simulation works will be carried out at every stage aforementioned in order to ensure the functionality and the effectiveness of the algorithms developed. The algorithms will be simulated for several types of environments. Firstly, the environment is assumed as an obstacle-free outdoor environment. Secondly, there are known obstacles in the static environment. Lastly, there are combinations of known and unknown obstacles, which make the outdoor environment more realistic for the mobile robot navigation. Physical constraints experienced by real robot will be investigated and included in the trajectory planning algorithms. The selection of the sensors also will be carried out during this stage in order to have a smooth navigation during experimental stage. Matlab software will be used for development and conduction of simulations.

3.4 Stage 4: Hardware Preparation and Experimental Works

Once the simulations have successfully been carried out at every stage, the algorithm will be validated by experiments. There will be several experiments to be carried out based on the environmental setup as in the simulation stages. An Ackermann-steering-like robot will be used in these experiments with consideration of the static and dynamic constraints of this mobile robot. The mobile robot, which is modified from the standard remote control car, will be equipped with sensors and time-critical control systems to ensure the objectives of this study are met. The modified mobile robot used in the experimental works is shown in Figure 3.3.

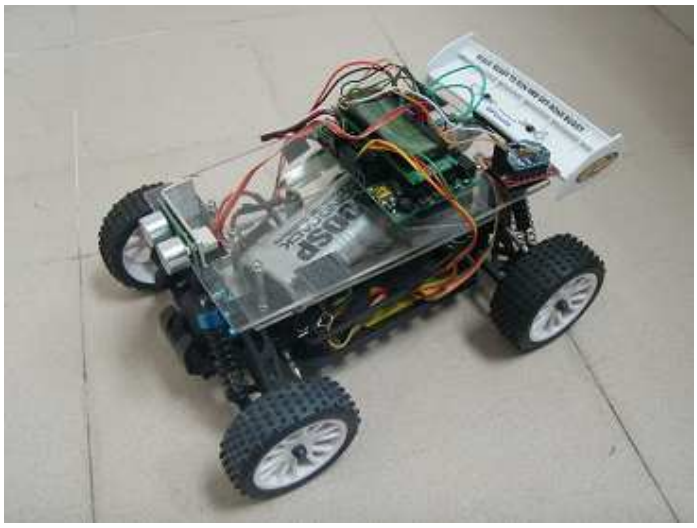


Figure 3.3 The modified mobile robot used in the experimental works.

The experimental setup has been divided into a few tasks in order to ease the experimental works. The tasks are:

1. Prepare the mobile robot, which includes upgrading, modifying and calibration works,
2. Program the microcontroller of the mobile robot,
3. Run the first test – obstacle-free environment, and
4. Run the second test – static and dynamic environments.

The main purposes of this experimental works are to validate the effectiveness of the simulation framework developed and to verify the practicality of developed algorithms in real-time applications. The experimental works will be conducted in an open-space area. The mobile robot will communicate wirelessly with the personal computer (PC), which will act as the coordinator. Then velocity and position of the mobile robot at every time step will be recorded in PC. These data will be used to plot the actual trajectory of the mobile robot. The movement of the mobile robot will also be captured using video camera to observe the behaviour of the mobile robot during navigating through the environment. These experimental results will then be used to compare and validate the algorithms against the respective simulation results.

3.5 Concluding remarks

In this chapter, the methodology of this study was discussed. The work can divided into four stages which began from the development of the algorithm for motion planning until the verification of the algorithm. The algorithm was developed for nonholonomic mobile robot by adopting geometric approach which includes obstacle avoidance. The algorithm was then tested by simulation using Matlab. The simulation works started from a simple scenario which was the obstacle-free environment in order to assess the functionality of the algorithm. It was then further tested in the more complicated environment with the combination of the static and dynamic obstacles.

Once the simulation works were successfully conducted, the algorithm was tested in the real environments using a mobile robot. The mobile robot was development by modifying a standard remote control car to become an autonomous nonholonomic mobile robot. The experimental works were conducted in a series of cases. The static and dynamic obstacles were considered in the experimental works in order to mimic the real environment. The results from the experiment were then being compared to the simulation works to validate and verify the practicality as well as the effectiveness of the algorithm for the time-critical motion planning.

4. DEVELOPMENT OF TIME-CRITICAL MOTION PLANNING ALGORITHMS

In this chapter, the development of time-critical motion planning algorithms and obstacle avoidance algorithm is discussed. The motion planning algorithms are based on the geometric approach. The development of the algorithms begins with the derivation of mathematical functions and boundary conditions until the integration of motion planning algorithm with obstacle avoidance algorithm.

The proposed algorithms for this study are shown in Figure 4.1. The algorithms are divided into several steps in order to ensure the algorithms will be executed smoothly. Firstly, the planner needs to set the input data for the mobile robot at the initial point and final point. The input data are position, orientation, steering angle, velocity and travelling time. Then an initial trajectory will be generated based on these inputs for the mobile robot to move from the initial point to the final point. Parameters such as position, velocity, orientation and steering angle will be determined at every time step. Furthermore the algorithms will check the current steering angle to ensure this output does not exceed the maximum limit. In the case of current steering angle exceeds the

maximum limit, the replanning algorithm will be initiated. A new steering angle will be used, which is the maximum steering angle and the initial trajectory will be modified in order to satisfy this limitation. On the other hand, if the mobile robot detects an obstacle, the obstacle avoidance algorithm will be initiated. If there is no obstacle and the current velocity or steering angle does not exceed the maximum limit, the mobile robot will continue its journey based on the generated trajectory until it reaches the final point.

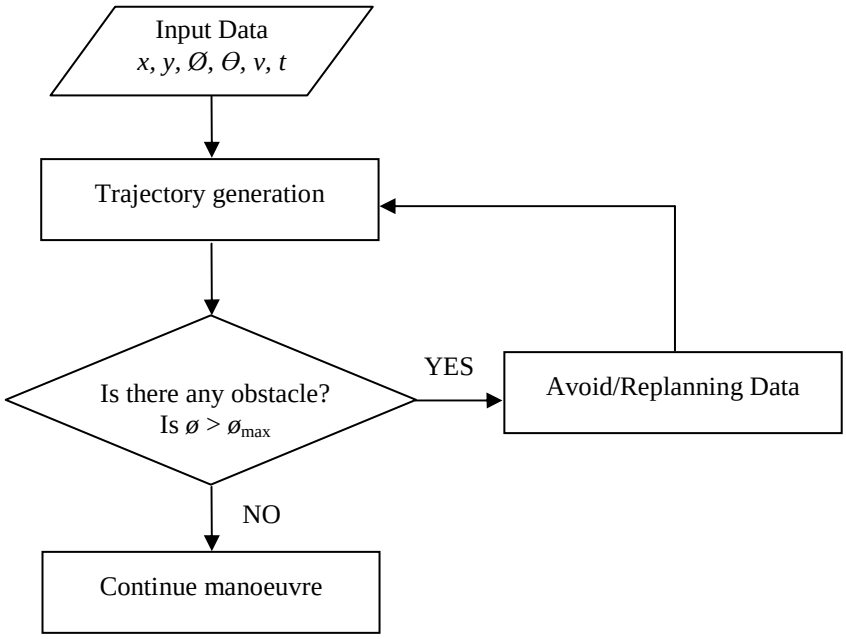


Figure 4.1 Flowchart of the proposed algorithms

Regarding the replanning algorithm, if any steering angle exceeds the maximum limit while moving along the path, the data at the current time step, which are location, velocity and orientation of the mobile robot will be obtained and will be used as the initial input data. Then the value of the steering angle will be readjusted to the specified maximum limit value for the steering angle and a new trajectory will be generated. The replanning algorithm is required for the proposed motion planning algorithms in order to obtain a smooth trajectory in which the mobile robot will be limited to kinematic and dynamic constraints such as steering angle and velocity.

4.1 Kinematic model of nonholonomic mobile robot

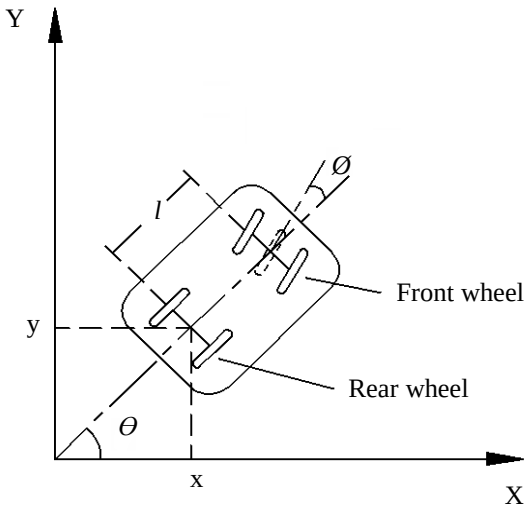


Figure 4.2 A car-like mobile robot

In this study, a car-like mobile robot is considered. The front wheels are the steering and the rear wheels are the driving wheels. For this study, it is assumed that both front wheels of the mobile robot will have similar steering angle, which is treated as a single front wheel as shown in Figure 4.2. The distance between front wheel and rear wheel axle centre is *l*. The midpoint of rear wheel axle is set to be a centre point in the space state, CP. Given the generalized coordinates is $q = [x, y, \theta, \phi, v, t]^T$, with (*x*, *y*) are the Cartesian coordinate, θ is the orientation of mobile robot with respect to the *x*-axis in Cartesian coordinate , ϕ is steering angle, *v* is the velocity and *t* is the required travel time.

Let ρ be the radius of rear wheel, u_1 be the angular velocity of the driving wheel and u_2 be the steering velocity of steering wheel (Dong and Guo, 2005). Then, the state space that represents the kinematic constraints of this mobile robot can be obtained from:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \tan\phi / l \\ 0 \end{pmatrix} \rho u_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_2 \qquad (4.1)$$

From the kinematic model (Equation 4.1), the range of θ and ϕ is limited to $(-\frac{\pi}{2}, \frac{\pi}{2})$ due to the structural and mathematical constraint of the physical mobile robot.

From Equation 4.1, we have:

$$\frac{dy}{dx} = \tan\theta, \quad \frac{d^2y}{dx^2} = \frac{\tan\phi}{l\cos^3\theta} \tag{4.2}$$

4.2 Boundary conditions

From the kinematic model (Equation 4.1), we have set the boundary conditions for the mobile robot as follow:

$$\begin{aligned} q(t_o) &= q^0 = [x_0, y_0, \theta_0, \phi_0, v_0, t_0]^T, \\ q(t_f) &= q^f = [x_f, y_f, \theta_f, \phi_f, v_f, t_f]^T, \end{aligned} \tag{4.3}$$

with v is the velocity of the mobile robot. In this study, we have set the initial velocity as v_0 and final velocity as v_T, so that we can control the velocity at both states. The generalized velocity function for the x- and y-axis is given by:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta \end{aligned} \tag{4.4}$$

The details of boundary conditions at initial and final state for x- and y-axis are as follow:

$$\begin{aligned} x(t_0) &= x_0, \\ \dot{x}(t_0) &= \dot{x}_0, \\ x(t_f) &= x_f, \\ \dot{x}(t_f) &= \dot{x}_f; \end{aligned} \tag{4.5}$$

$$\begin{aligned} y(t_0) &= y_0, \\ \frac{dy}{dx}\Big|_{t=t_0} &= \tan\theta_0, \\ \frac{d^2y}{dx^2}\Big|_{t=t_0} &= \frac{\tan\phi_0}{l\cos^3\theta_0}; \end{aligned} \tag{4.6}$$

$$
\begin{aligned}
& y(t_f) = y_f, \\
& \left.\frac{dy}{dx}\right|_{t=t_f} = \tan\theta_f, \\
& \left.\frac{d^2y}{dx^2}\right|_{t=t_f} = \frac{\tan\phi_f}{l\cos^3\theta_f};
\end{aligned}
\tag{4.7}
$$

For *x* equation, with consideration of the boundary conditions (Equation 4.5), we have chosen a cubic polynomial equation as:

$$x = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \tag{4.8}$$

with first and second derivative as follow:

$$\frac{dx}{dt} = a_1 + 2a_2 t + 3a_3 t^2, \tag{4.9}$$

$$\frac{d^2x}{dt^2} = 2a_2 + 6a_3 t \tag{4.10}$$

For *y* equation, with consideration of the boundary conditions (Equation 4.6 and Equation 4.7), we have chosen a quintic polynomial equation as:

$$y = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5, \tag{4.11}$$

with first and second derivative as follow:

$$\frac{dy}{dt} = b_1 + 2b_2 t + 3b_3 t^2 + 4b_4 t^3 + 5b_5 t^4, \tag{4.12}$$

$$\frac{d^2y}{dt^2} = 2b_2 + 6b_3 t + 12b_4 t^2 + 20b_5 t^3 \tag{4.13}$$

4.3 Coordinate-x equation

From the boundary conditions (Equation 4.5) and the cubic polynomial equations (Equation 4.8, 4.9 and 4.10), we have:

$$x(t_0) = x_0$$
$$a_0 + a_1 t_0 + a_2 {t_0}^2 + a_3 {t_0}^3 = x_0 \tag{4.14}$$

$$\frac{dx(t_0)}{dt} = \dot{x}_0$$
$$a_1 + 2a_2 t_0 + 3a_3 {t_0}^2 = \dot{x}_0 \tag{4.15}$$

$$x(t_f) = x_f$$
$$a_0 + a_1 t_f + a_2 {t_f}^2 + a_3 {t_f}^3 = x_f \tag{4.16}$$

$$\frac{dx(t_f)}{dt} = \dot{x}_f$$
$$a_1 + 2a_2 t_f + 3a_3 {t_f}^2 = \dot{x}_f \tag{4.17}$$

Let $a = [a_0, a_1, a_2, a_3]^T$ is the constant vector and rearrange Equation 4.14 to Equation 4.17 as $a = A^{-1}c$, we have:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix}^{-1} \begin{bmatrix} x_0 \\ \dot{x}_0 \\ x_f \\ \dot{x}_f \end{bmatrix} \tag{4.18}$$

4.4 Coordinate-y equation

From the boundary conditions (Equation 4.6) and the quintic polynomial equations (Equation 4.11 ─ 4.13), we have:

$$y(t_0) = y_0$$
$$b_0 + b_1 t_0 + b_2 {t_0}^2 + b_3 {t_0}^3 + b_4 {t_0}^4 + b_5 {t_0}^5 = y_0 \tag{4.19}$$

From Chain Rule, we have:

$$\frac{dy(t_0)}{dt} = \frac{dx}{dt}\frac{dy}{dx}$$

$$b_1 + 2b_2 t_0 + 3b_3 {t_0}^2 + 4b_4 {t_0}^3 + 5b_5 {t_0}^4 = \frac{dx}{dt}\tan\theta_0$$

$$b_1 + 2b_2 t_0 + 3b_3 {t_0}^2 + 4b_4 {t_0}^3 + 5b_5 {t_0}^4 = \alpha_1 \tan\theta_0 \qquad (4.20)$$

with

$$\alpha_1 = \frac{dx}{dt}|_{t=t_0}$$

and

$$\frac{d^2 y(t_0)}{dt^2} = \frac{dy}{dx}\frac{d^2 x}{dt^2} + \frac{d^2 y}{dx^2}\left(\frac{dx}{dt}\right)^2$$

$$2b_2 + 6b_3 t_0 + 12b_4 {t_0}^2 + 20b_5 {t_0}^3 = \frac{d^2 x}{dt^2}\tan\theta_0 + \left(\frac{dx}{dt}\right)^2 \frac{\tan\phi_0}{l\cos^3\theta_0}$$

$$2b_2 + 6b_3 t_0 + 12b_4 {t_0}^2 + 20b_5 {t_0}^3 = \alpha_2 \tan\theta_0 + \alpha_3 \frac{\tan\phi_0}{l\cos^3\theta_0} \qquad (4.21)$$

with

$$\alpha_2 = \frac{d^2 x}{dt^2}|_{t=t_0}$$

$$\alpha_3 = \left(\frac{dx}{dt}|_{t=t_0}\right)^2$$

From the boundary conditions (Equation 4.17) and the quintic polynomial equations (Equation 4.11 ─ 4.13), we have:

$$y(t_f) = y_f$$

$$b_0 + b_1 t_f + b_2 {t_f}^2 + b_3 {t_f}^3 + b_4 {t_f}^4 + b_5 {t_f}^5 = y_f \qquad (4.22)$$

From Chain Rule, we have:

$$\frac{dy(t_f)}{dt} = \frac{dx}{dt}\frac{dy}{dx}$$

$$b_1 + 2b_2 t_f + 3b_3 {t_f}^2 + 4b_4 {t_f}^3 + 5b_5 {t_f}^4 = \frac{dx}{dt}\tan\theta_f$$

$$b_1 + 2b_2 t_f + 3b_3 {t_f}^2 + 4b_4 {t_f}^3 + 5b_5 {t_f}^4 = \alpha_4 \tan\theta_f \qquad (4.23)$$

with

$$\alpha_4 = \frac{dx}{dt}|_{t=t_f}$$

and

$$\frac{d^2 y(t_f)}{dt^2} = \frac{dy}{dx}\frac{d^2 x}{dt^2} + \frac{d^2 y}{dx^2}\left(\frac{dx}{dt}\right)^2$$

$$2b_2 + 6b_3 t_f + 12b_4 {t_f}^2 + 20b_5 {t_f}^3 = \frac{d^2 x}{dt^2}\tan\theta_f + \left(\frac{dx}{dt}\right)^2 \frac{\tan\phi_f}{l\cos^3\theta_f}$$

$$2b_2 + 6b_3 t_f + 12b_4 {t_f}^2 + 20b_5 {t_f}^3 = \alpha_5 \tan\theta_f + \alpha_6 \frac{\tan\phi_f}{l\cos^3\theta_f} \qquad (4.24)$$

with

$$\alpha_5 = \frac{d^2 x}{dt^2}|_{t=t_f}$$

$$\alpha_6 = \left(\frac{dx}{dt}|_{t=t_f}\right)^2$$

Let $b = [b_0, b_1, b_2, b_3, b_4, b_5]^T$ is the constant vector and rearrange Equation 4.19 to Equation 4.24 as $b = A^{-1}c$, we have:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 & {t_0}^4 & {t_0}^5 \\ 0 & 1 & 2t_0 & 3{t_0}^2 & 4{t_0}^3 & 5{t_0}^4 \\ 0 & 0 & 2 & 6t_0 & 12{t_0}^2 & 20{t_0}^3 \\ 1 & T & T^2 & T^3 & T^4 & T^5 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 \\ 0 & 0 & 2 & 6T & 12T^2 & 20^3 \end{bmatrix}^{-1} \begin{bmatrix} y_0 \\ \alpha_1 \tan\theta_0 \\ \alpha_2 \tan\theta_0 + \alpha_3 \frac{\tan\phi_0}{l\cos^3\theta_0} \\ y_f \\ \alpha_4 \tan\theta_f \\ \alpha_5 \tan\theta_f + \alpha_6 \frac{\tan\phi_f}{l\cos^3\theta_f} \end{bmatrix} \qquad (4.25)$$

4.5 Orientation (θ) equation

By Equation 4.1, we have:

$$\begin{aligned}
&\frac{dy}{dx} = \tan\theta \\
&\frac{dy/dt}{dx/dt} = \tan\theta \\
&\frac{b_1 + 2b_2 t + 3b_3 t^2 + 4b_4 t^3 + 5b_5 t^4}{a_1 + 2a_2 t + 3a_3 t^2} = \tan\theta \\
&\theta = \tan^{-1}\left(\frac{b_1 + 2b_2 t + 3b_3 t^2 + 4b_4 t^3 + 5b_5 t^4}{a_1 + 2a_2 t + 3a_3 t^2} \right)
\end{aligned} \tag{4.26}$$

4.6 Steering angle (ϕ) equation

By Equation 4.2, we have:

$$\begin{aligned}
&\frac{d^2 y}{dt^2} = \frac{dy}{dx}\frac{d^2 x}{dt^2} + \frac{d^2 y}{dx^2}\left(\frac{dx}{dt}\right)^2 \\
&\frac{d^2 y}{dx^2} = \frac{\frac{d^2 y}{dt^2} - \frac{dy}{dx}\frac{d^2 x}{dt^2}}{\left(\frac{dx}{dt}\right)^2} \\
&\frac{\tan\phi}{l\cos^3\theta} = \frac{(2b_2 + 6b_3 t + 12b_4 t^2 + 20b_5 t^3) - (2a_2 + 6a_3 t)\tan\theta}{(a_1 + 2a_2 t + 3a_3 t^2)^2} \\
&\phi = \tan^{-1}\left\{ l\cos^3\theta \left[\frac{(2b_2 + 6b_3 t + 12b_4 t^2 + 20b_5 t^3) - (2a_2 + 6a_3 t)\tan\theta}{(a_1 + 2a_2 t + 3a_3 t^2)^2} \right] \right\}
\end{aligned} \tag{4.27}$$

4.7 Angular velocity (u_1) equation

Let $v_0 = \rho u_1$. From Pythagoras' Theorem, we have:

$$
\begin{aligned}
(\rho u_1)^2 &= (\rho u_1 \cos\theta)^2 + (\rho u_1 \sin\theta)^2 \\
(\rho u_1)^2 &= (\rho u_1)^2 \cos^2\theta + (\rho u_1)^2 \sin^2\theta \\
\rho u_1 &= \rho u_1 \cos^2\theta + \rho u_1 \sin^2\theta \\
\rho u_1 &= \rho u_1 \cos\theta(\cos\theta) + \rho u_1 \sin\theta(\sin\theta) \\
\rho u_1 &= \frac{dx}{dt}(\cos\theta) + \frac{dy}{dt}(\sin\theta) \\
u_1 &= \frac{[(a_1 + 2a_2 t + 3a_3 t^2)\cos\theta] + [(b_1 + 2b_2 t + 3b_3 t^2 + 4b_4 t^3 + 5b_5 t^4)\sin\theta]}{\rho}
\end{aligned}
\tag{4.28}
$$

4.8 Obstacle avoidance approach

Obstacle detection is fundamental for a mobile robot to navigate safely in a dynamic environment. In this study, the obstacle avoidance approach deals with both static and moving obstacles in a 2D workspace using a laser range finder (LRF). The approach is evolved from the dynamic trajectory planning scheme presented (Jolly *et al.*, 2008), In a dynamic trajectory planning scheme, the mobile robot will replan and modify its trajectory once it detects an obstacle and the newly generated trajectory may differ from the initially planned trajectory. However, instead of using the Bezier curves, which were used by Jolly *et al*. (2008), polynomial curves have been adopted in this study. The reason behind this is to ensure that the mobile robot will pass through all the control points to have a better control for the mobile robot's motion, compared to the Bezier curves, which only pass through the first and last control points (Jolly *et al.*, 2008). In this study, all the control points are used as inputs to generate the polynomial curves to ensure the generated curves will pass all the control point. Furthermore, the dynamic trajectory planning scheme is divided into two planning schemes, which are utilised to avoid static obstacles and moving obstacles.

4.8.1 Avoiding static obstacles

In this study, the static obstacles are divided into two categories: known and unknown. Known static obstacles are known in advance to the planner during offline planning, while unknown static obstacles are unknown to the planner and will only be detected by the sensor during navigation. For the known static obstacles, the planner will consider them in the initial stage while generating the trajectory. Thus the generated trajectory should navigate the mobile robot to be away from the potentially colliding obstacles. Meanwhile, the unknown static obstacles will only be considered when the mobile robot starts to navigate through the environment. The general view of avoiding an unknown static obstacle is illustrated in Figure 4.3.

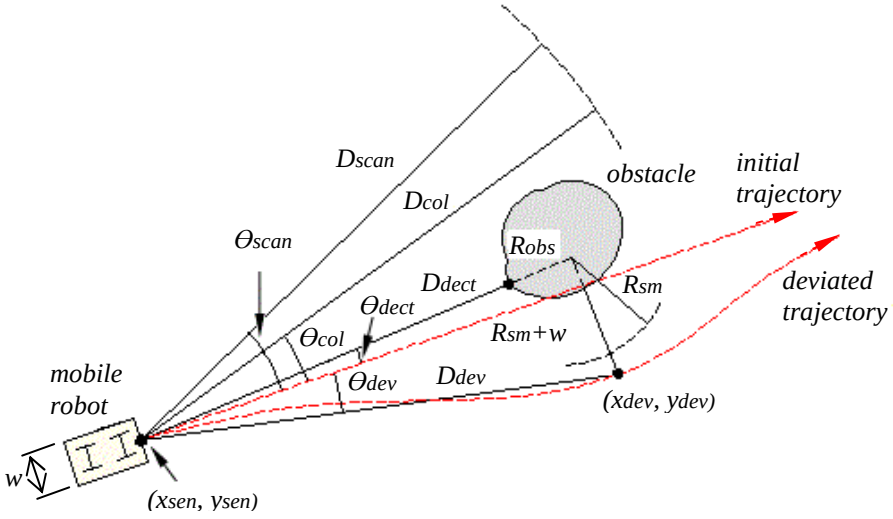


Figure 4.3 Avoiding a detected static obstacle which is unknown in priori

When the mobile robot starts to navigate along the initial trajectory, the range finder will also start to scan the environment. The maximum scanning range and resolution is set by D_{scan} and Θ_{scan}, respectively. Once the mobile robot detects an obstacle, it will check whether the obstacle is within collision region or not. The collision region is defined by collision range (D_{col}) and collision angle (Θ_{col}). If the obstacle falls into this region, a new deviated point will be calculated in order to readjust the initial trajectory and to ensure the mobile robot avoids the obstacle. The deviated point (x_{dev},y_{dev}) is determined by detection distance (D_{dect}), detection angle (Θ_{dect}), obstacle's size (R_{obs}), safety margin (R_{sm}), robot's width (w) and sensor's position (x_{sen},y_{sen}). The following equations are used to obtain the deviated point:

$$D_{dev} = \sqrt{\left(R_{sm} + w\right)^2 + \left(D_{dect} + R_{obs}\right)^2} \tag{4.29}$$

$$\theta_{dev} = \theta_{dect} + \tan^{-1}\left(\frac{R_{sm} + w}{D_{dect} + R_{obs}}\right) \tag{4.30}$$

$$x_{dev} = x_{sen} + D_{dev} \times \cos\theta_{dev} \tag{4.31}$$

$$y_{dev} = y_{sen} + D_{dev} \times \sin\theta_{dev} \tag{4.32}$$

Once the deviated point is obtained, a new trajectory (deviated trajectory) is generated from the current point to the final point, through the deviated point. The new trajectory will have to ensure that it catches up with the time lost during obstacle avoidance in order to reach the final point at the specified time. Note that the new trajectory does not necessarily follow the initial generated trajectory as the new trajectory is based on the updated information.

4.8.2 Avoiding moving obstacles

The strategy to avoid a moving obstacle is usually based on prior information of the moving obstacle's velocity (Guo *et al.*, 2003; Qu *et al.*, 2004). However, in this study the strategy is based on the direction and position of the moving obstacle. Furthermore, the moving obstacle's direction of movement will influence the selection of appropriate strategy to avoid it. For instance, if the moving obstacle is approaching perpendicularly to the mobile robot, the mobile robot will avoid the moving obstacle as illustrated in Figure 4.4(a). On the other hand, if the moving obstacle is approaching from the opposite direction of the mobile robot, the moving obstacle is treated as a static obstacle and the mobile robot will avoid the obstacle as illustrated in Figure 4.4(b).

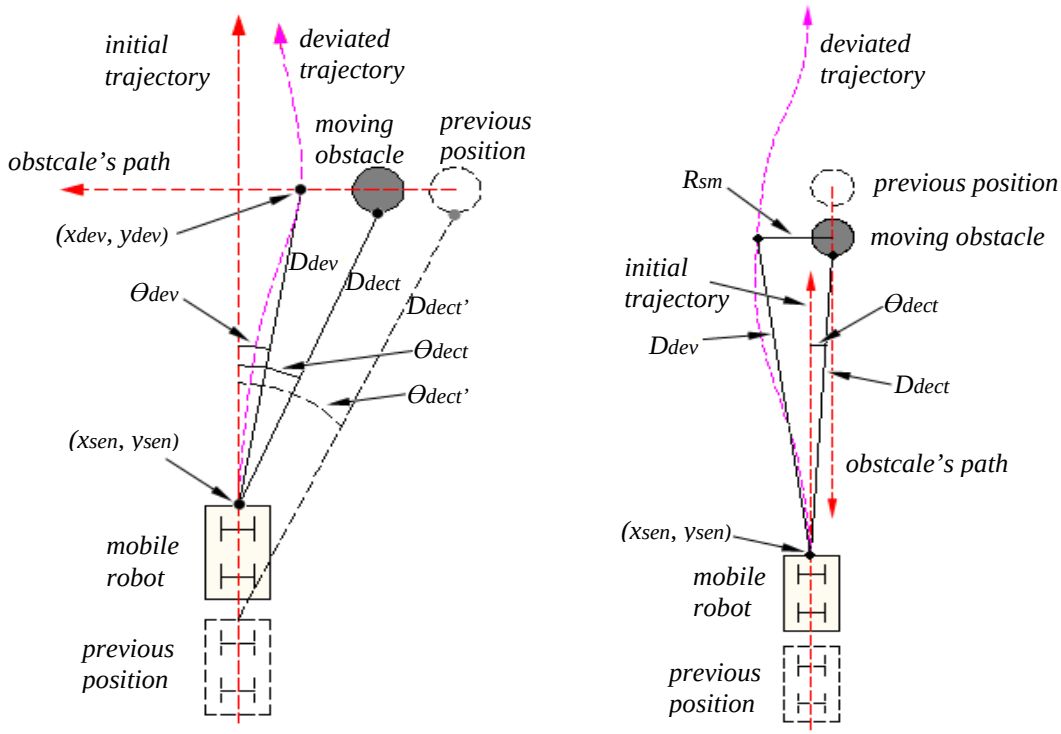


Figure 4.4 Avoiding a moving obstacle (a) perpendicular direction to the mobile robot and (b) in opposition to the mobile robot.

Despite the direction of the moving obstacle, the mobile robot will predict the possibility of collision between the mobile robot and the moving obstacle. As shown in Figure 4.5(b), when the mobile robot first detects a moving obstacle, the position for both the mobile robot and the moving obstacle will be registered into the registry. Then, when the next detection occurs, the system will compare the stored position (first detection) with the current position (second detection) to obtain direction and distance between these two locations for both the mobile robot and the moving obstacle. In addition, the planner will estimate the velocity of the moving obstacle.

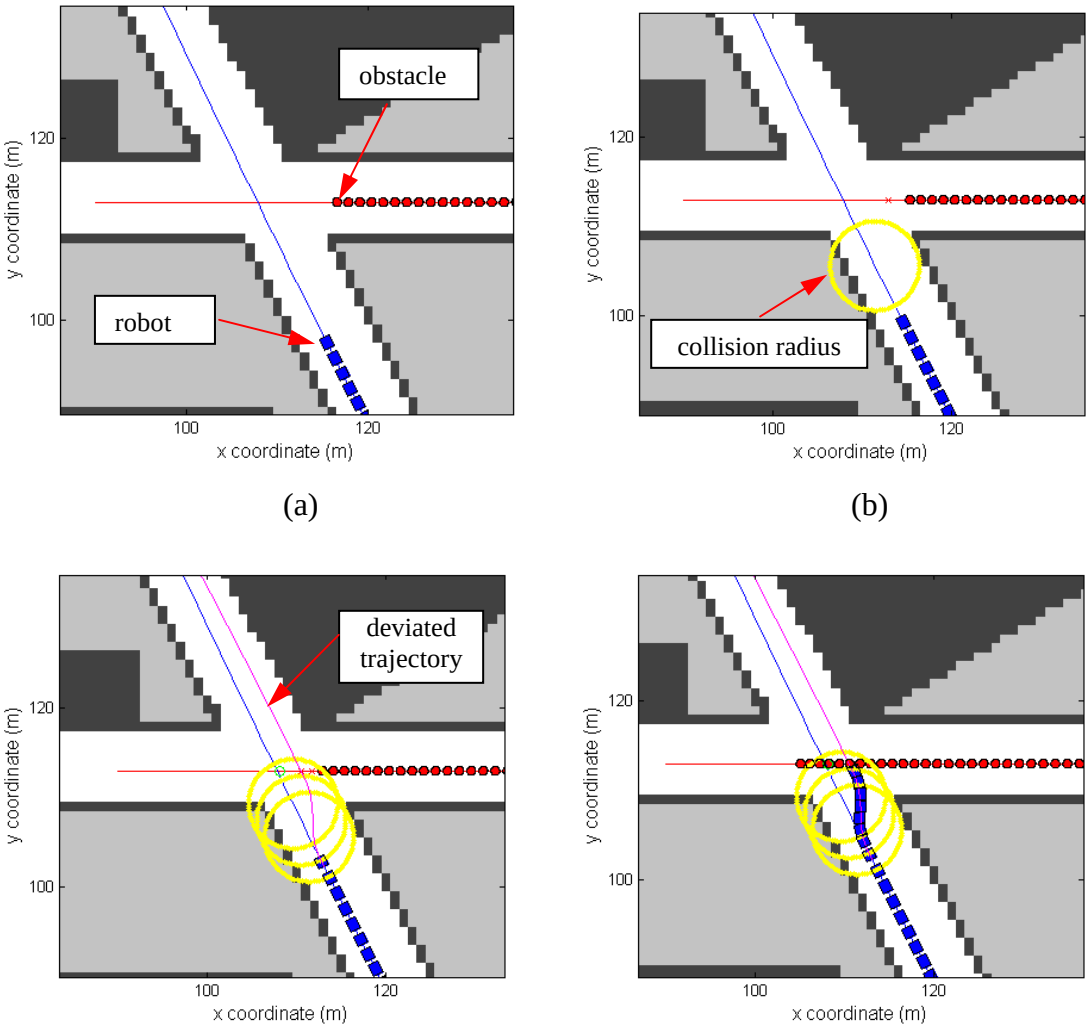


Figure 4.5 Collision prediction approach (a) before detection of the obstacle, (b) first detection, (c) predicted position falls inside the collision radius, and (d) obstacle avoidance approach implemented.

From this information, the system can predict the mobile robot's and moving obstacle's position for the next two steps. If the predicted moving obstacle's position falls inside the collision radius of the mobile robot, then the collision is likely to happen as shown in Figure 4.5(c), the collision point (x_{col}, y_{col}) and the deviation point (x_{dev}, y_{dev}) are determined by using the following equations:

$$x_{col} = x_{sen} + D_{dect} \times \cos\theta_{dect} \tag{4.33}$$

$$y_{col} = y_{sen} + D_{dect} \times \cos\theta_{dect} \tag{4.34}$$

$$D_{col} = \sqrt{(x_{movobs} - x_{col})^2 + (y_{movobs} - y_{col})^2} \quad (4.35)$$

$$x_{dev} = x_{col} - (r_{movobs} + w) \quad (4.36)$$

$$y_{dev} = y_{col} \quad (4.37)$$

where,

r_{movobs} = size of a moving obstacle

w = width of the robot.

Then a new trajectory which is a deviated trajectory will be generated from the mobile robot's current point to the final point, through the deviation point as shown in Figure 4.5(d).

4.9 Concluding remarks

The algorithm for time-critical motion planning was developed for a nonholonomic mobile robot by geometric approach. The kinematic constraints were taken into consideration during the development of the algorithm. Furthermore, the development of the algorithm was also considered the limitation of the mobile robot such as steering angle and velocity in order to obtain a smooth trajectory. In addition, the obstacle avoidance approach was incorporated in the algorithm in order for the mobile robot to avoid obstacles.

The obstacle avoidance algorithm was divided into two categories which are for static obstacles and for moving obstacles. The approach utilizes the developed trajectory planning algorithm in order to avoid obstacles. When avoiding obstacles, the algorithm replans its initial trajectory once the mobile robot detects an obstacle. The algorithm uses the current information such as location of the mobile robot and obstacle, velocity and orientation to generate new trajectory. Therefore, the mobile robot will able to avoid the obstacle and reach the final point at the specified time.

In addition, for moving obstacle, the algorithm will predict the motion of the detected moving obstacle and the possibility that the collision will occur between the mobile robot and the moving obstacle. If the collision is likely to happen, the algorithm will

replan its trajectory based on the direction of the moving obstacle. Therefore, the mobile robot will able to avoid the obstacle and reach the final point at the specified time.

5. SIMULATION RESULTS AND DISCUSSIONS

Simulation is one of the popular tools to investigate the effectiveness and capability of a system prior to the real experimental works. A few benefits can be gained by conducting simulation works such as reducing experimental material's cost and time. Robotics field also is no exception in using simulation. The algorithm can be investigated and any adjustment and modification on the algorithm can be done during simulation works. Then once the algorithm is working well, it can be downloaded into real robots. Furthermore, simulation results can be used as a guideline or comparison for the experimental works.

In this study, the platform for the simulation works is conducted in Matlab. Matlab is one of the development tools that has been widely used in engineering fields. Matlab is to develop the simulation platform because of its powerful graphics and ease of use. In addition, it is also supported by many different computer systems and it comes with an extensive built-in library of predefined functions for mathematical and technical solutions.

5.1 Simulation architecture

The algorithms introduced in this study consist of an offline and online planning. These algorithms will ensure the mobile robot is able to navigate with a smooth motion and within the limitation of the kinematic constraints such as steering angle and it also be able to avoid obstacles.

Figure 5.1 shows the scheme of the algorithm steps for the simulation, which were used in this study. At the early stage, the initial trajectory will be generated using the input data from the user. Then, the offline planning was executed to deal with the known or predefined static obstacles. The algorithm will check whether there is a known static obstacle along the way of the initially generated trajectory. If there is an obstacle, the new input data at the detection point will be used, such as position, orientation, steering angle, velocity and time, to generate a new trajectory. The process will continue until a collision-free trajectory is generated, with incorporation of steering angle limitation.

Once the offline planning has been completed and the trajectory has been generated, the new data from the offline planning will be used for the online planning. The inputs, such as time, steering angle and velocity, will be used to simulate the mobile robot at every time steps. While navigating within the environment, the robot also will check the presence of new obstacle, which is previously unknown. If there is an identified obstacle, the obstacle avoidance will be executed to avoid the obstacle and once the robot has avoided the obstacle, the trajectory from the deviated point to the final point will be replanned using the actual data in order to catch-up the time loss from avoiding the obstacle and to maintain a smooth trajectory. The process will continue until the robot reaches the final point.

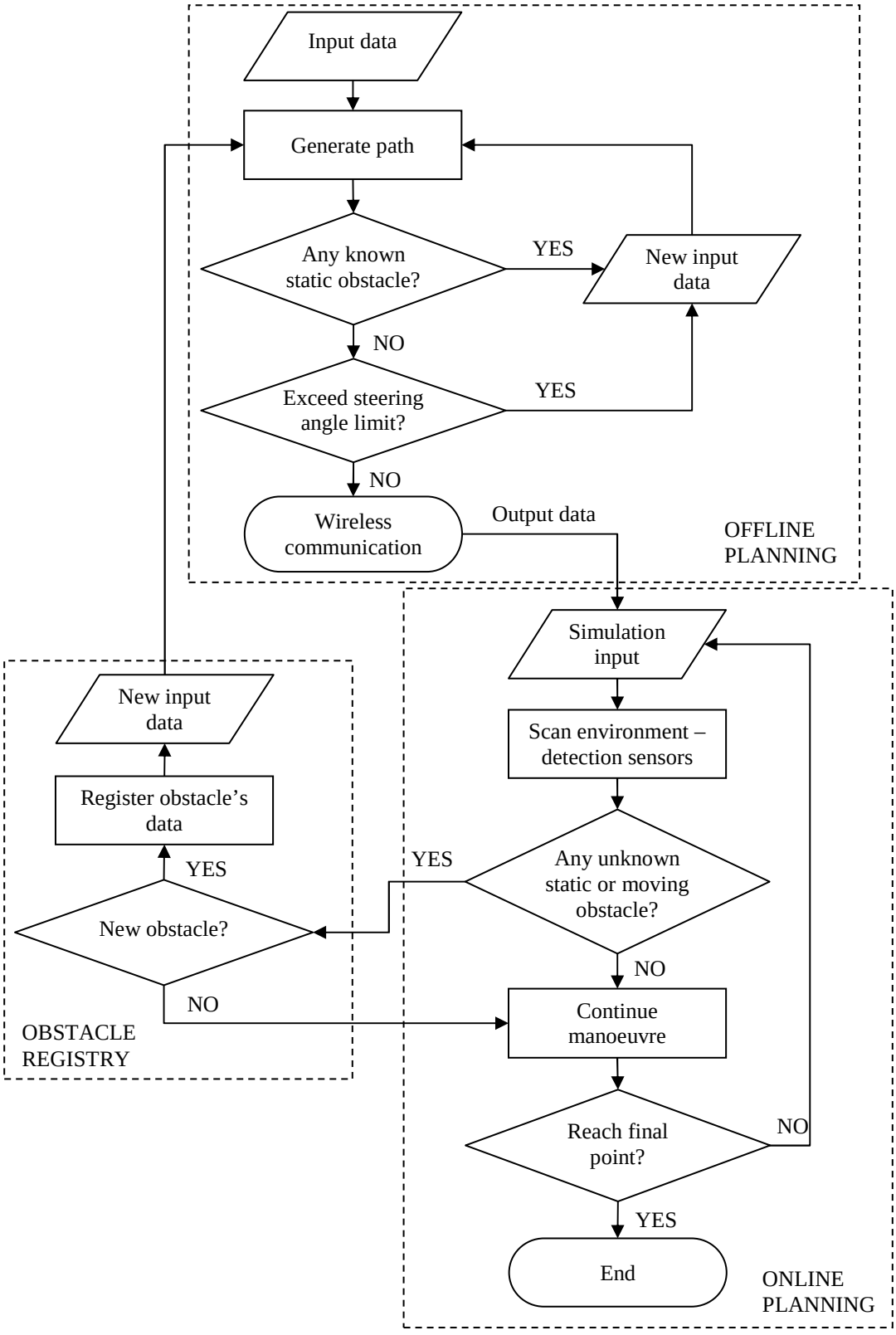


Figure 5.1 Simulation process flowchart.

5.2 Simulated vehicle

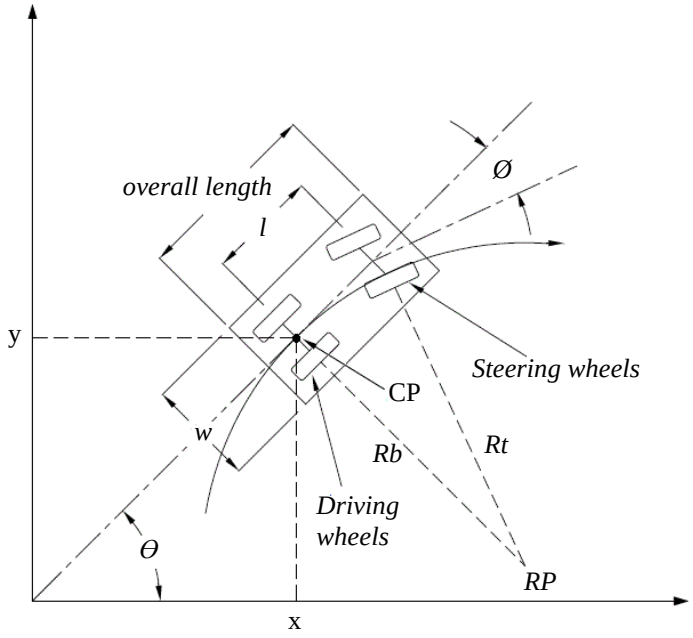


Figure 5.2 Geometric model of a mobile robot.

For simulation environment, the mobile robot used for this study is an Ackermann-steering mobile robot which based on the modified remote control car. It is assumed that the mobile robot behaves as a tricycle which the rear wheels are the driving wheel and the front wheel is a steering wheel. The geometry model of the mobile robot is modelled based on work by Liddy and Lu (2007). However a few modifications have been made to suit this study. In this project, the centre point (CP) is located at the middle of the rear axle instead of centre of the vehicle and instantaneous turning points (*RP*) is shown as in Figure 5.2. The inputs for this simulation environment are velocity (v) and orientation (θ) which calculated in Section 4.1. The following equations will demonstrate the actual position, orientation and steering angle during online planning. To find the steering angle (ϕ):

$$d = u \times \Delta t + \frac{1}{2} \Delta v \times \Delta t \tag{5.1}$$

$$Rb = \frac{d}{2\sin(\Delta\theta / 2)} \tag{5.2}$$

$$\phi = \tan^{-1}\left(\frac{l}{Rb}\right) \tag{5.3}$$

where,

d = distance between time interval
u = initial velocity
Rb = distance between CP and RP.

To find position of x and y, the equations can be reversed by using steering angle and velocity as inputs, as shown by the following equations:

$$Rb = \frac{l}{\tan\phi} \tag{5.4}$$

$$\Delta\theta = 2\sin^{-1}\left(\frac{d/2}{Rb}\right) \tag{5.5}$$

$$\theta_{new} = \theta_{old} + \Delta\theta \tag{5.6}$$

$$x_{new} = x_{old} + d\cos\theta \tag{5.7}$$

$$y_{new} = y_{old} + d\sin\theta \tag{5.8}$$

5.3 Matlab frameworks

In this section, the development of the simulation frameworks using Matlab is discussed. In order to simulate the mobile robot planner as close as possible to the actual environment, the Laser Range Finder (LRF) is simulated as shown in Figure 5.3.

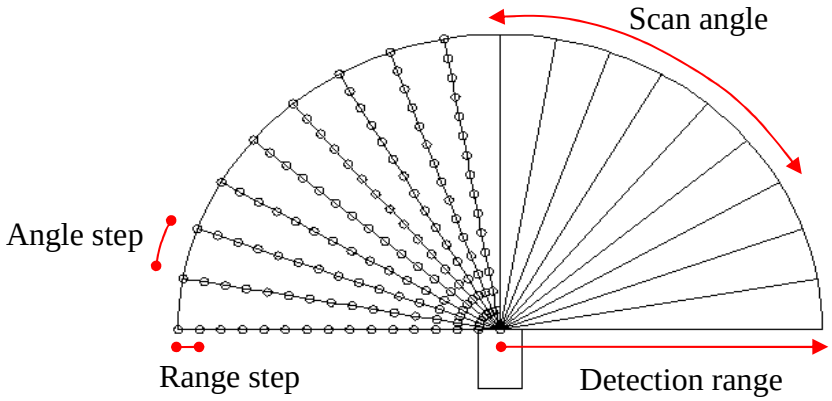


Figure 5.3 Simulated Laser Range Finder.

The LRF is placed in front of the mobile robot so that the detection coverage can be optimised. The coverage of the LRF is defined by the *scan angle* and the *detection range*. And the resolution of the LRF is defined by the *angle step* and the *range step*. In the actual LRF, only the angular resolution is counted. However in order to obtain the detection distance once the mobile robot detects an obstacle, the range step is one of the approaches in the simulation framework that can be adopted to overcome this issue.

The map can be generated in any graphic editor software such as Paint. In this study, the obstacles are represented by square and circle as shown in Figure 5.4. The known and unknown static obstacles are represented by black and green squares, respectively. And the moving obstacle is represented by red circle. The obstacles can be randomly placed in the simulation map or can be arranged properly to indicate the fixed objects in the actual environment such as lamp posts and trees.

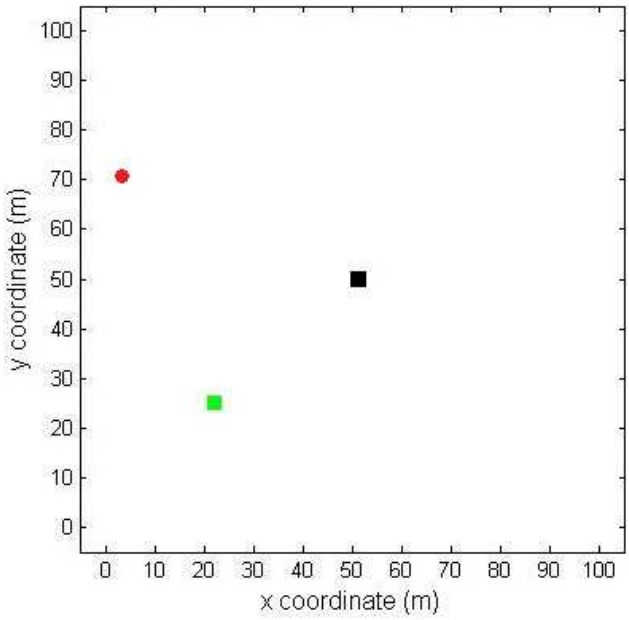
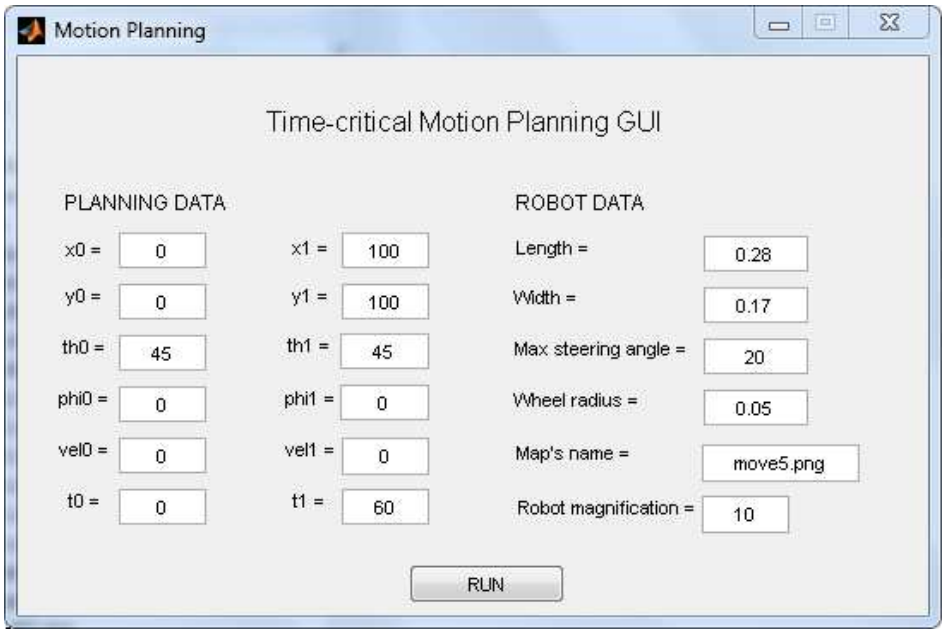
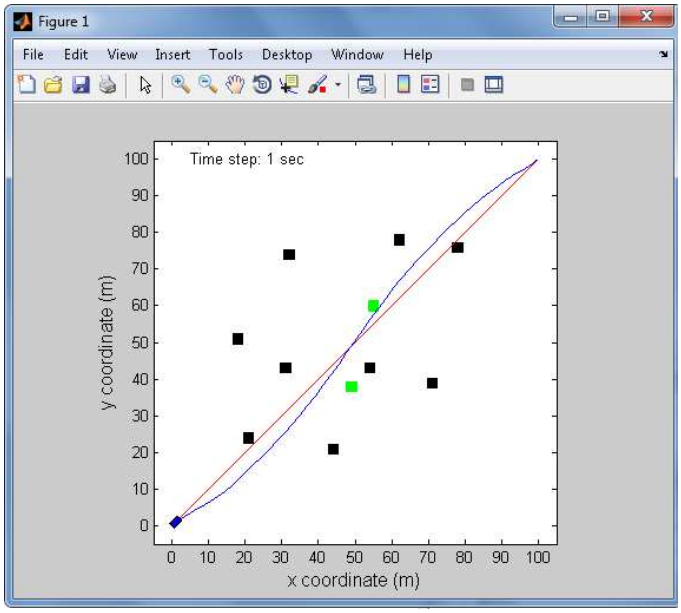


Figure 5.4 Simulation map with static and moving obstacles.

Furthermore, the Graphical User Interface (GUI) was developed for the simulation framework as shown in Figure 5.5. This interface was developed to ensure the user will be able to run the simulation with ease and will give a user-friendly simulation framework. With the GUI, the user will only need to key in the initial and final state of the mobile robot. In addition, the user will also need to key in the physical data of the mobile robot.



(a)



(b)

Figure 5.5 The Graphical User Interface (GUI) for simulation framework (a) Input GUI, (b) Output GUI

5.4 Trajectory optimization

The original trajectory planning may exceed the restriction of the physical limitations of the mobile robot such as maximum turning radius and maximum velocity or acceleration. In real-life driving, the lower speed is preferred when the driver is closed to the obstacles or when the driver is making a sharp turn. Thus the mobile robot needs to follow a reasonable velocity profile in order to mimic the actual driving behaviour. In Section 5.1, the architecture of the simulation has been discussed. As we know, the original offline trajectory planning will consider the limitation of steering angle and velocity of the mobile robot. Furthermore, these boundary conditions will also be considered in the online planning, meaning that when the mobile robot starts to navigate the environment following the original trajectory. An example of the generated trajectory is shown in Figure 5.6.

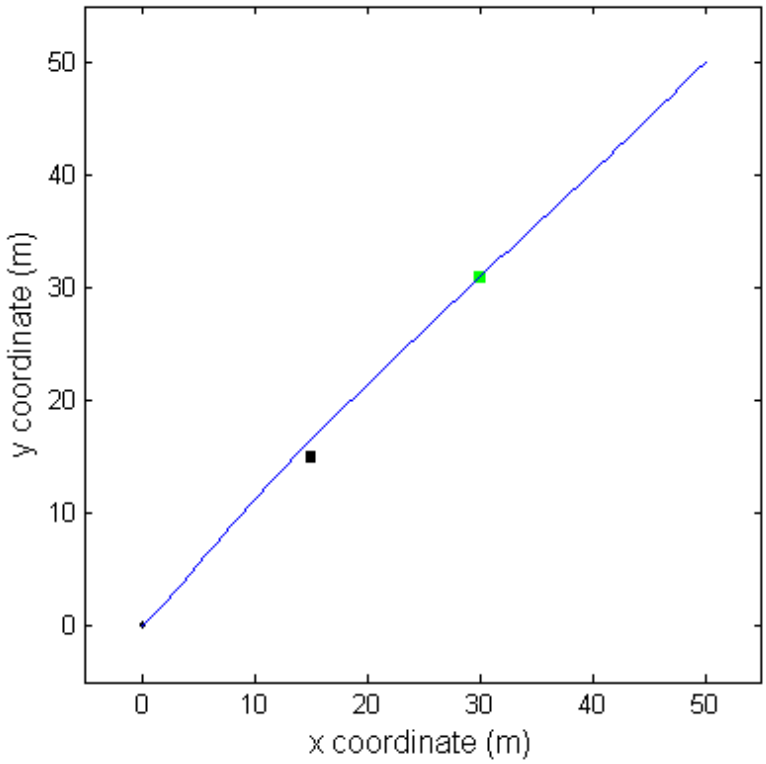


Figure 5.6 Original trajectory plan.

The original trajectory is planned with two obstacles – known and unknown static obstacles. The map dimension is 50*m* x 50*m* and the travel time is set to be 60*s*. As shown in Figure 5.6, the original trajectory was planned pass through the unknown obstacle. This is because the algorithm only considered the known obstacle at the first place. The unknown obstacle will be considered after the sensor detects the obstacle and the obstacle is potentially blocking the path. Then the algorithm will planned a new trajectory in order to avoid the obstacle. The final result of the trajectory is shown in Figure 5.7.

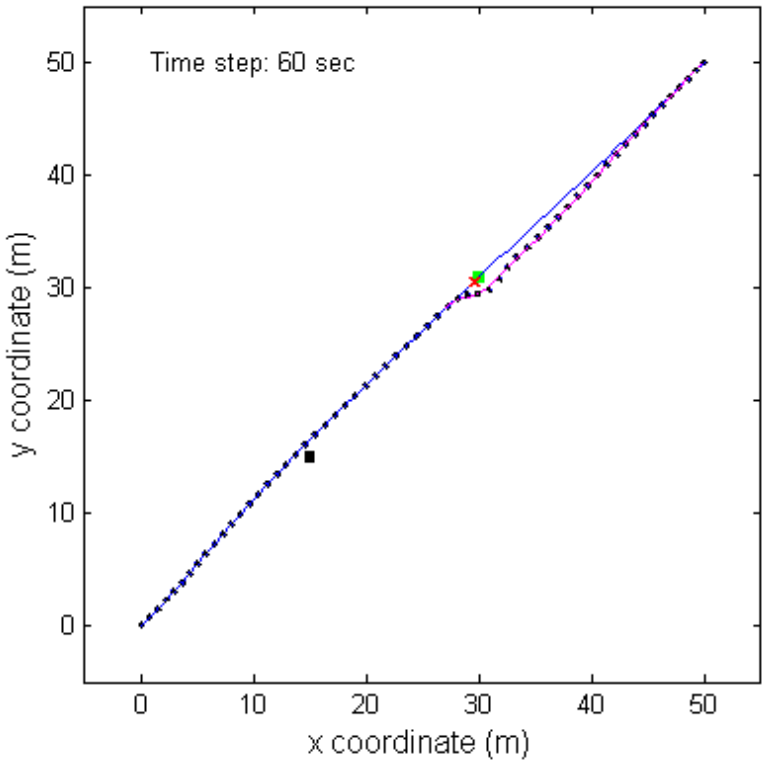
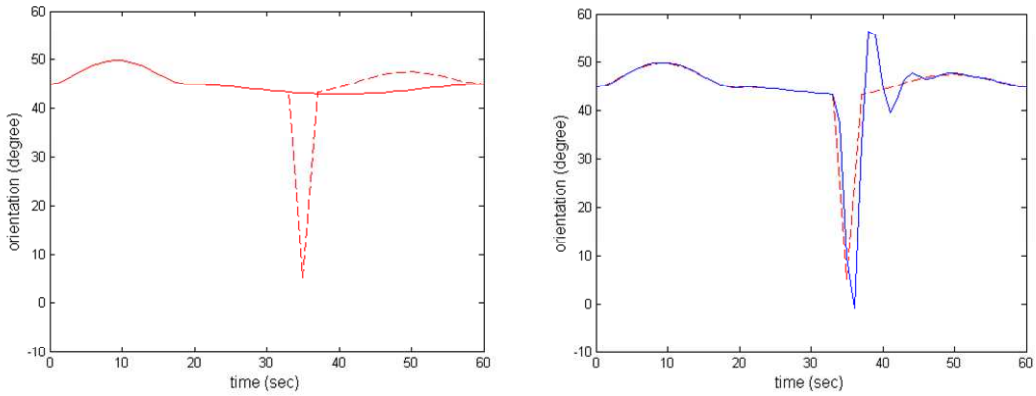


Figure 5.7 Final result of the trajectory.

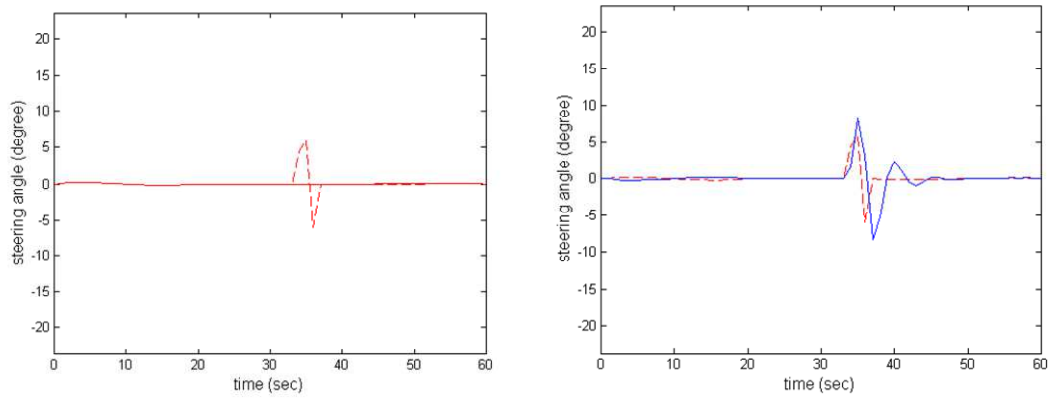
The orientation of the mobile robot was changed dramatically as the mobile robot tries to avoid the obstacle as shown in Figure 5.8(a). If there is no other obstacle detected, the mobile robot will try to follow the original trajectory as close as possible. Once it detected an unknown obstacle, for example in this case, the mobile robot detected an unknown obstacle at time step 33*s*, it has turned right in order to avoid the obstacle. The decision of turning right or left is made by observing the location or position of the obstacle. For example, if the obstacle is at the left region of the mobile robot respect to the mobile robot's orientation, it will turn right. Furthermore, the actual orientations were given by the red line. The actual orientations were slightly different from the adjusted orientation due to the adjustment made in order to satisfy the actual steering limitation of the mobile robot during navigation as shown in Figure 5.8(b).



(a) (b)

Figure 5.8 Orientation profiles (a) Planned orientation (red line) against adjusted orientation (red dashed), and (b) adjusted orientation (red dashed) against actual orientation (blue line).

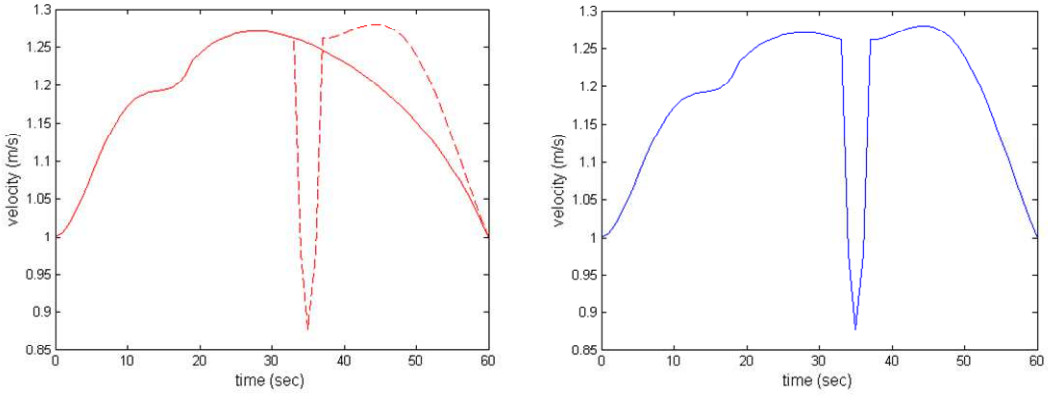
The steering angle plots are shown in Figure 5.9. The adjustment on the robot's orientation will also reflect the steering angle values of the mobile robot. Figure 5.9(a) shows the original planned steering angle values (red line) compared to the adjusted on steering angle values (red dashed) once the mobile robot detected an obstacle. The positive values indicate the mobile robot is turning right while the negative values indicate the mobile robot is turning left. The actual steering angle values are given by the blue plot as shown in Figure 5.9(b).



(a) (b)

Figure 5.9 Steering angle profiles (a) Planned steering angle (red line) against adjusted steering angle (red dashed), and (b) adjusted steering angle (red dashed) against actual steering angle (blue line).

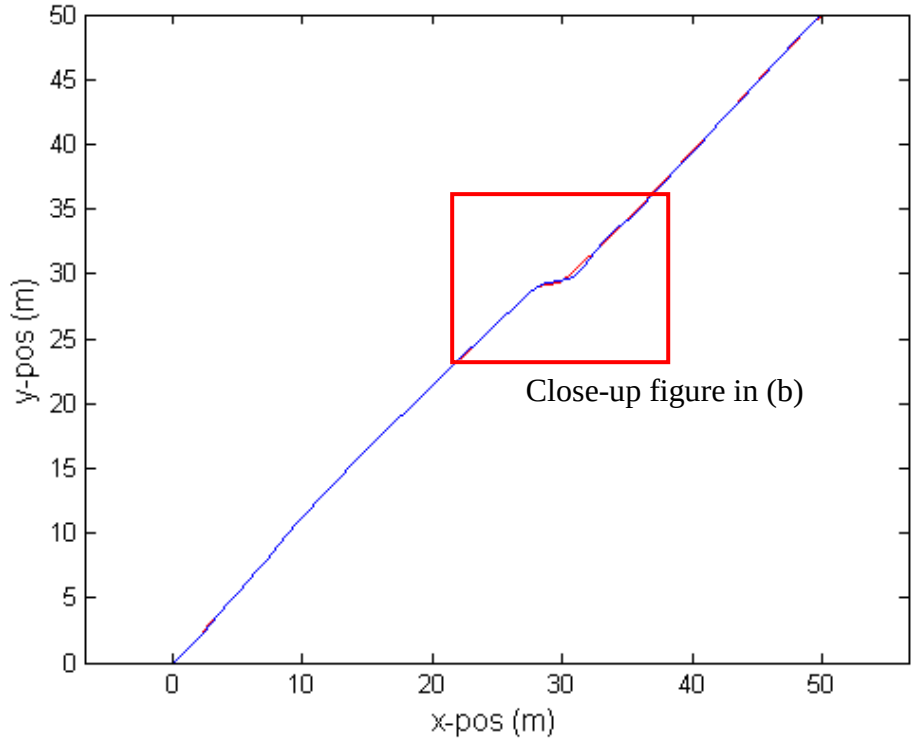
One of the input parameter is the velocity of the mobile robot. Figure 5.10(a) shows the original planned velocity values which are given by the red plot and the adjusted velocity values once the mobile robot detected an obstacle which is given by the blue plot. Once the mobile robot detected an obstacle, the algorithm tends to decrease the velocity of the mobile robot in order to avoid the obstacle smoothly. It is also mimic the actual human driver when he encounters an obstacle. The driver will try to slow down its vehicle once he detects an obstacle so that he can steer steering smoothly. Then once the mobile robot was already avoided the obstacle, the mobile robot will speed up in order to pick up the time lost due to avoiding the obstacle. This is the reason why the adjusted velocity value is higher than the original velocity after 39s. In Figure 5.10(b), the actual velocity was matched with the adjusted velocity because the algorithm used the adjusted velocity as input parameter for the mobile robot to manoeuvre.



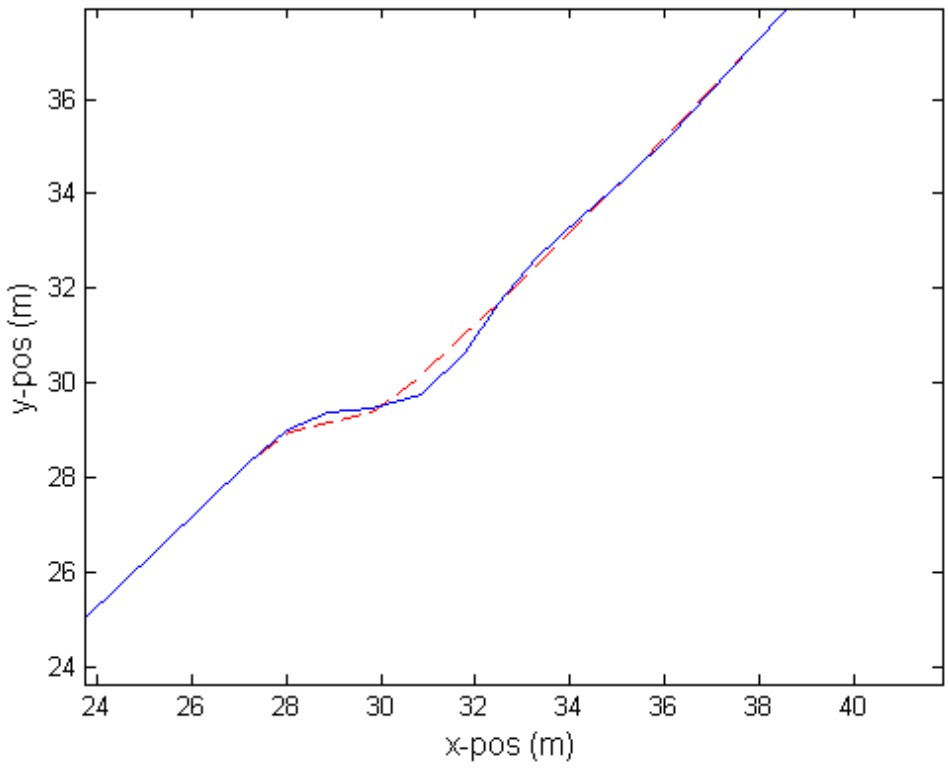
(a) (b)

Figure 5.10 Velocity profiles (a) Planned velocity (red line) against adjusted velocity (red dashed), and (b) adjusted velocity (red dashed) against actual velocity (blue line).

The final actual trajectory is shown in Figure 5.11. The planned trajectory is given by the red dashed plot while the actual trajectory is given by the blue line plot. As we can see in Figure 5.11(b), the actual trajectory is slightly deviated from the adjusted trajectory due to the adjustment made in order to cater the steering angle limitation. This proved that the algorithm is able to incorporate the kinematic constraints of the mobile robot.



(a)



(b)

Figure 5.11 Adjusted trajectory (red dashed) against actual trajectory (blue line).

5.4.1 Replanning approach

Replanning approach is introduced to reduce the errors while the mobile robot navigates through the intermediate waypoints. When the mobile robot reaches the waypoint, the replanning approach will be executed and a new trajectory will be generated from the current intermediate waypoint to the next waypoint. The current data at the waypoint such as position and velocity will be used to generate the new trajectory.

The replanning approach scenario is in Figure 5.12. The black areas represent the walls and/or known obstacles. The waypoints are represented by red circles. Point 1 and Point 4 are the initial and final points, respectively. Point 2 and Point 3 are the desired waypoints. The orientation at each point is indicated by an arrow. The input data for this simulation are summarized in Table 5.1. The control inputs for this simulation are steering angle and velocity.

Table 5.1 Input data for replanning approach scenario

Point	t (sec)	x (m)	y (m)	θ (^{o})	ϕ (^{o})	v (m/s)
1	0	15	10	90	0	0
2	20	30	50	0	0	2
3	40	70	50	0	0	2
4	60	85	90	90	0	0

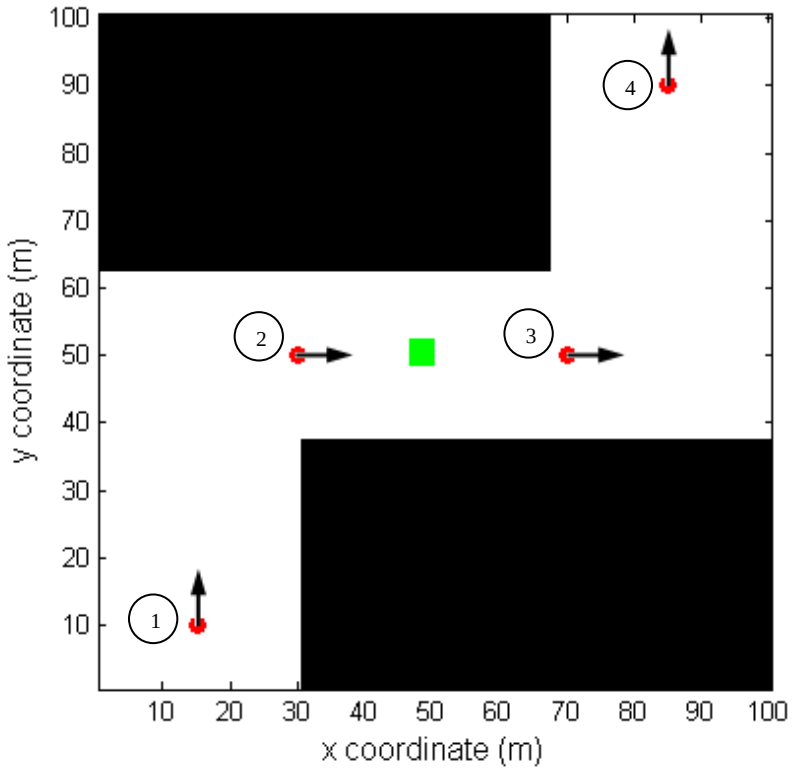


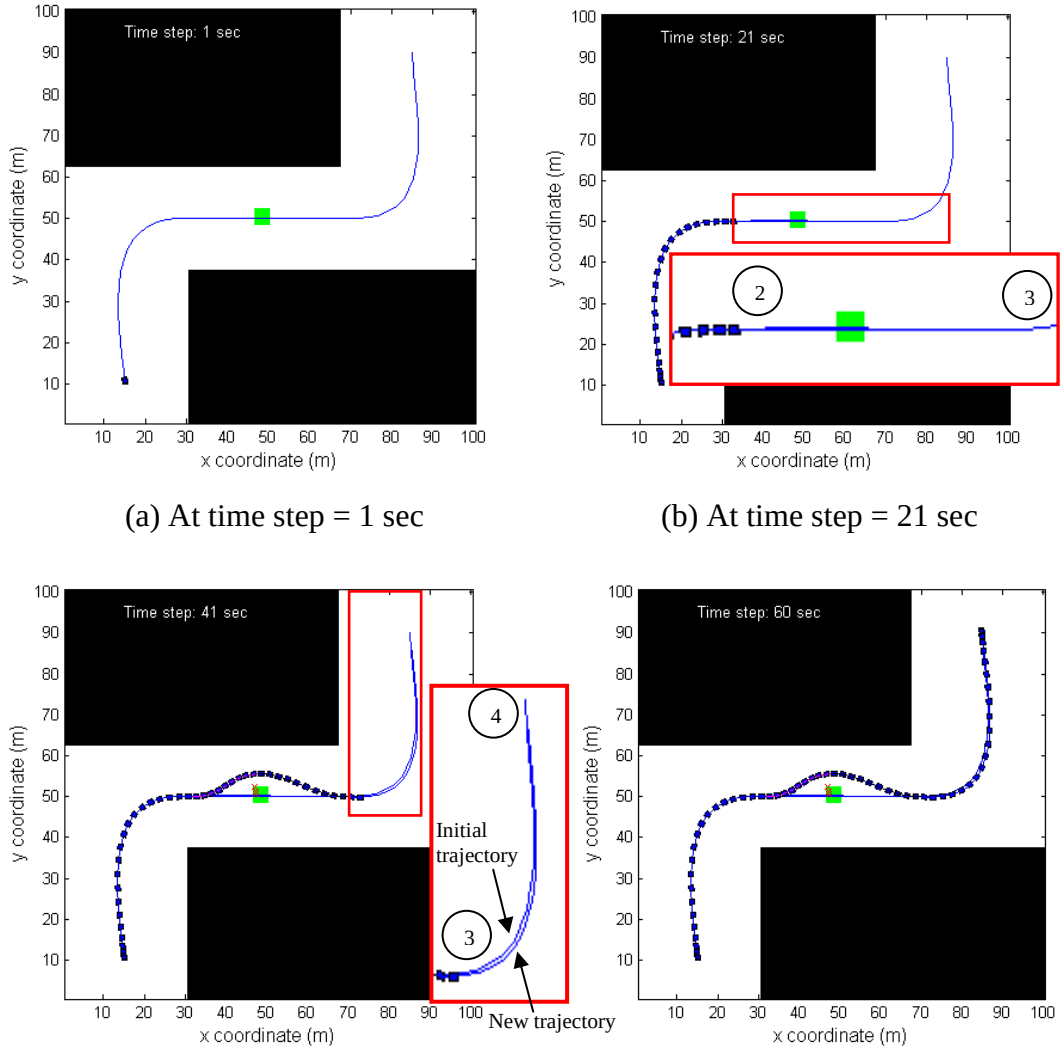
Figure 5.12 Prior map with two waypoints connecting the initial and final point.

The simulation algorithm consists of both offline and online planning components. The offline planning deals with the known obstacles and it will be executed at the initial stage. Then, the online planning will be executed once the mobile robot starts navigating in the environment. The online planning is to detect and deal with unknown obstacles Table 5.2 summarizes the actual collected data at every waypoint without replanning approach. In comparison to input data for simulation in Table 5.1, the errors in position and orientation at the final point are around 2.29 meters and 0.3 degrees, respectively. These errors are quite large, especially for position error.

Table 5.2 Actual collected data of simulation without replanning approach

Point	t (sec)	x (m)	y (m)	θ (^{o})	ϕ (^{o})	v (m/s)
1	0	15	30	90	0	0
2	20	30.08	49.99	2.1	0	2
3	40	70.00	49.97	-3.3	0	2
4	60	87.21	89.04	89.7	0	0

In the scenario without replanning approach, the mobile robot will follow the initial trajectory. While in the scenario with replanning approach, the new trajectory will be generated once the mobile robot reached the waypoint. The simulation results for replanning approach are shown in Figure 5.13 Simulation results with replanning approach.



(a) At time step = 1 sec

(b) At time step = 21 sec

(c) At time step = 41 sec

(d) At time step = 60 sec

Figure 5.13 Simulation results with replanning approach.

Figure 5.13(a) shows the initial planned trajectory from the initial point to the final point, and pass through all the waypoints. Then the mobile robot navigates along the initial planned trajectory until it reaches Point 2. Once it reaches Point 2, the replanning algorithm is executed. Using the actual data at Point 2, a new trajectory is generated from Point 2 to Point 3, as shown in Figure 5.13(b). The new trajectory is almost identical to the initial trajectory because the errors are quite small. Then the

mobile robot continues its journey along the new trajectory, avoiding the obstacle and reaches Point 3. At this point, replanning algorithm once again is executed and a new trajectory is generated from Point 3 to the final point, as shown in Figure 5.13(c). The mobile robot then continues its journey and finally reaches the final point, as shown in Figure 5.13(d). The actual collected data at every waypoint are summarized in Table 5.3.

Table 5.3 Actual collected data with replanning approach

Point	t (sec)	x (m)	y (m)	θ (^{o})	ϕ (^{o})	v (m/s)
1	0	15	30	90	0	0
2	20	30.08	49.99	2.1	0	2
3	40	70.00	49.97	-3.3	0	2
4	60	84.9	90.00	93.2	0	0

From the, the data at both waypoints - Point 2 and Point 3 - are not much different from data in Table 5.2. This is because at Point 2, the replanning approach was not yet been executed. While at Point 3, the trajectory was affected by the obstacle avoidance algorithm. Therefore the significant errors different can be perceived at the final point. With replannning approach, the errors in position and orientation are around 0.1 meters and 3.2 degrees, respectively. Although the error in orientation is greater than the previous orientation error, it is considered as still in satisfactory limits, with the consideration of both position and orientation errors.

5.5 Simulation results and discussions

In the previous section, the simulation framework used a simple example to explain how the simulation and algorithms work. The generated map was consisted of two static obstacles. In this section, more complicated examples are presented. A series of simulation cases have been setup to investigate the capability and effectiveness of the algorithms. The scenario of the simulations will include more mobile robots, obstacles and more complicated environments.

All the simulations were conducted in Matlab. Steering angle and velocity of the mobile robot have been used as the control input parameters for these simulations. A

few general assumptions have been made for the modelled Ackermann steering car-like robot in these simulations:

1. The mobile robot moves on horizontal plane – no topological effects,
2. Single point contact of the wheels,
3. The wheels are not deformable,
4. No slipping, skidding or friction, and
5. The wheels are attached at the rigid chassis.

5.5.1 Navigation in static and open-space environments

The first set of simulation cases is conducted in the static and open-space environments. In these scenarios, the there are only static obstacles and the map is set as an open space are such as a field or large area. The complicated obstructed environment is a 100m x 100m region and the obstacles are randomly placed in the map consisting known and unknown static obstacles as shown in Figure 5.14. The environment is tested by one, two and three robots with various initial and final points.

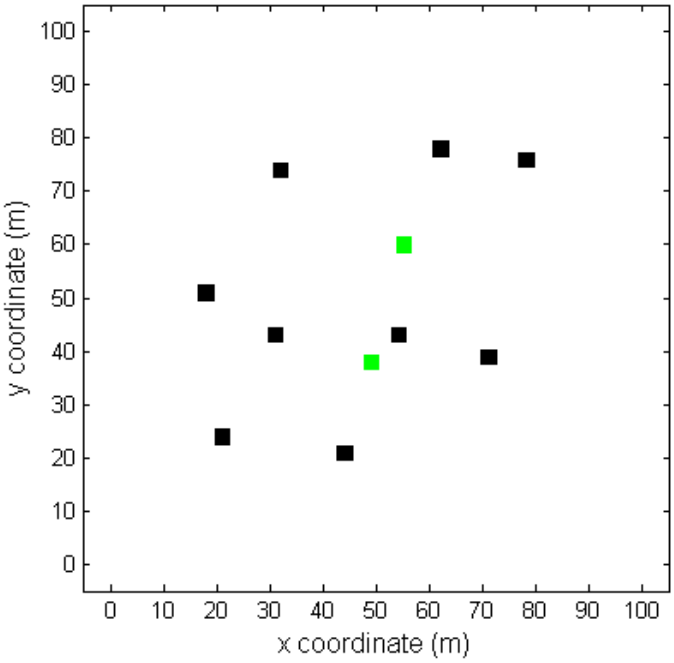


Figure 5.14 A complicated obstructed environment.

In Case 1, the input data is tabulated in Table 5.4. In this case, only one mobile robot was used to navigate in the environment. The mobile robot started from the bottom-left of the map as shown in Figure 5.15(a). The red line is the original planned trajectory without considering the known static obstacles, while the blue line is the pre-planned trajectory with the consideration of the known static obstacles. At 29s, the mobile robot detected an obstacle and the new trajectory was generated in order to avoid the obstacle as shown in Figure 5.15(c). The final result at 60s is shown in Figure 5.15(d).

Table 5.4 Input data for simulation Case 1.

Point	t (sec)	x (m)	y (m)	Θ (^{o})	$\varnothing$ (^{o})	v (m/s)
Initial	0	0	0	45	0	0
Final	60	100	100	45	0	0

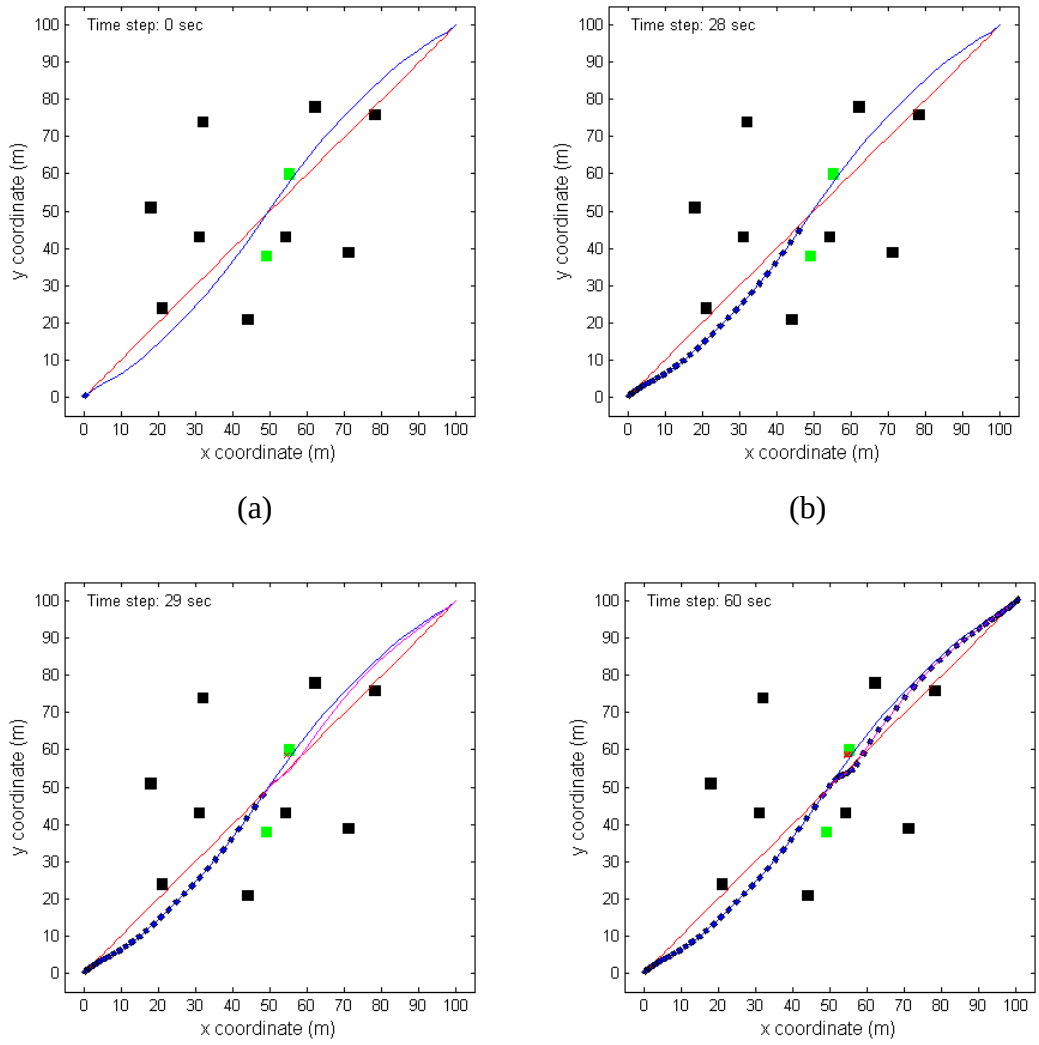


Figure 5.15 One mobile robot navigates in the environment.

The comparisons between the planned and actual orientation, steering angle, velocity and location of Robot 1 are shown in Figure 5.16. Due to the presence of the unknown static obstacles, the trajectory was adjusted in order to avoid the unknown obstacles. From Figure 5.16, the mobile robot tried to follow the planned input from the beginning until it encountered the unknown static obstacles. The actual position of the mobile robot at the final point is tabulated in Table 5.5.

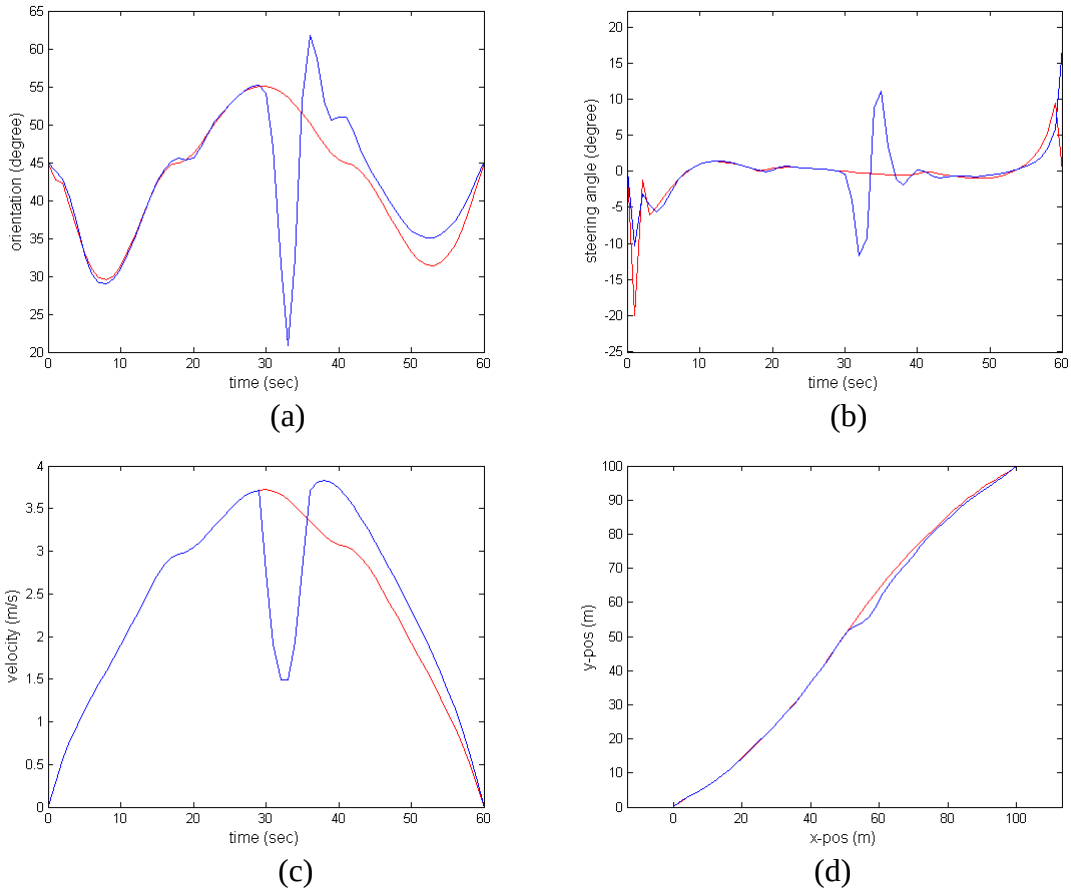


Figure 5.16 Robot 1: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) location.

Table 5.5 Actual data collected at the final point for Case 1

	t (sec)	x (m)	y (m)	Θ (^{o})	$\varnothing$ (^{o})	v (m/s)
Planned	60	100	100	45	0	0
Actual	60	99.93	99.93	45.15	17.16	0

In Case 2, two mobile robots were used and the input data for each robot are tabulated in Table 5.6. The first mobile robot started from the bottom-left of the map and the second mobile robot started from right-hand side of the map. Each robot navigates to the different final point as shown in Figure 5.17(a). The travel time for both mobile robots was assigned to 60 seconds. As we can see in Figure 5.17(b) and (c), both mobile robots were capable to detect and avoid the obstacle. The final result at 60s is shown in Figure 5.17(d).

Table 5.6 Input data for simulation Case 2.

Point		t (sec)	x (m)	y (m)	Θ (^{o})	$\emptyset$ (^{o})	v (m/s)
Robot 1	Initial	0	0	0	45	0	0
	Final	60	100	100	45	0	0
Robot 2	Initial	0	100	30	-30	0	0
	Final	60	0	100	-30	0	0

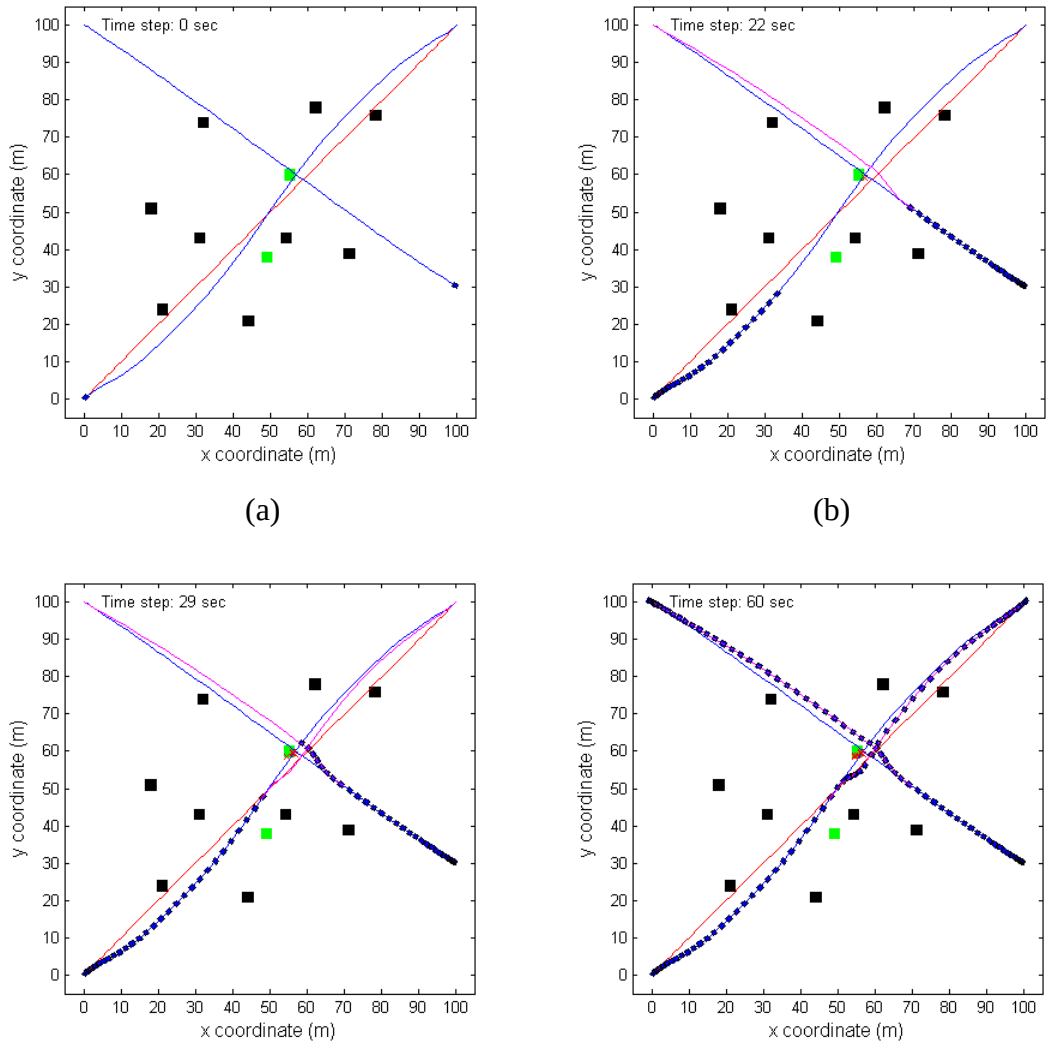


Figure 5.17 Two mobile robots navigate in the environment.

The comparisons between the planned and actual orientation, steering angle, velocity and location for Robot 1 are similar to Case 1 as shown in Figure 5.16. For Robot 2, the comparisons are shown in Figure 5.18. Due to the presence of the unknown static obstacles, the trajectory was adjusted in order to avoid the unknown obstacles. The actual position of the mobile robot at the final point is tabulated in Table 5.7.

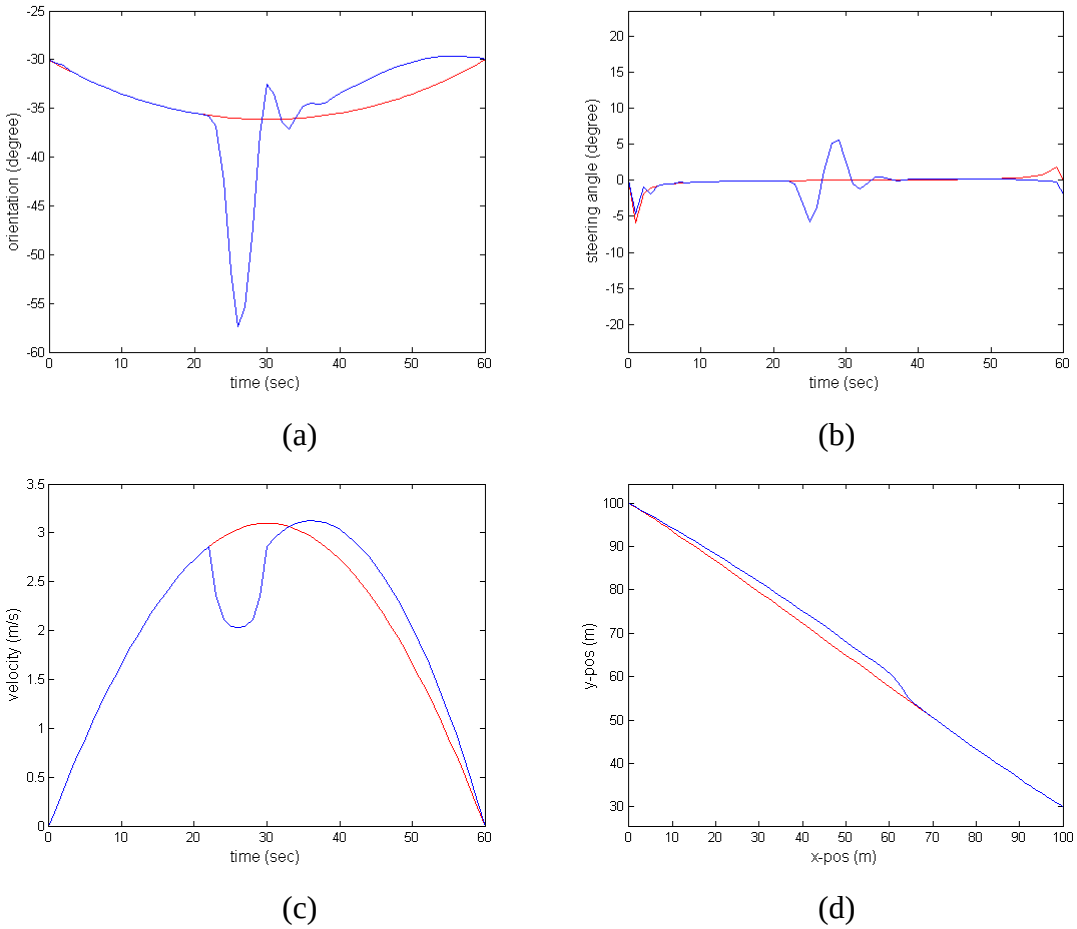


Figure 5.18 Robot 2: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position.

Table 5.7 Actual data collected at the final point for Case 2

Point		*t* (sec)	*x* (m)	*y* (m)	Θ (^{o})	$\emptyset$ (^{o})	*v* (m/s)
Robot 1	Planned	60	100	100	45	0	0
	Actual	60	99.93	99.93	45.15	17.16	0
Robot 2	Initial	60	0	100	-30	0	0
	Final	60	-0.0057	100	-29.98	-1.84	0

In the third case (Case 3), three mobile robots were used and the input data for each mobile robot are tabulated in Table 5.8. The first mobile started from the bottom-left of the map, the second mobile robot started from the right-hand side of the map and the third mobile robot started from the left-hand side of the map as shown in Figure 5.19(a). The travel time for the first and the second mobile robot was assigned as 60 seconds, while the third mobile robot was 40 seconds. At 40s, the third mobile robot was already reached the final point as shown in Figure 5.19(c). The final result at 60s is shown in Figure 5.19(d).

Table 5.8 Input data for simulation Case 3.

Point		t (sec)	x (m)	y (m)	Θ (^{o})	$\varnothing$ (^{o})	v (m/s)
Robot 1	Initial	0	0	0	45	0	0
	Final	60	100	100	45	0	0
Robot 2	Initial	0	100	30	-30	0	0
	Final	60	0	100	-30	0	0
Robot 3	Initial	0	0	50	0	0	0
	Final	40	80	100	30	0	0

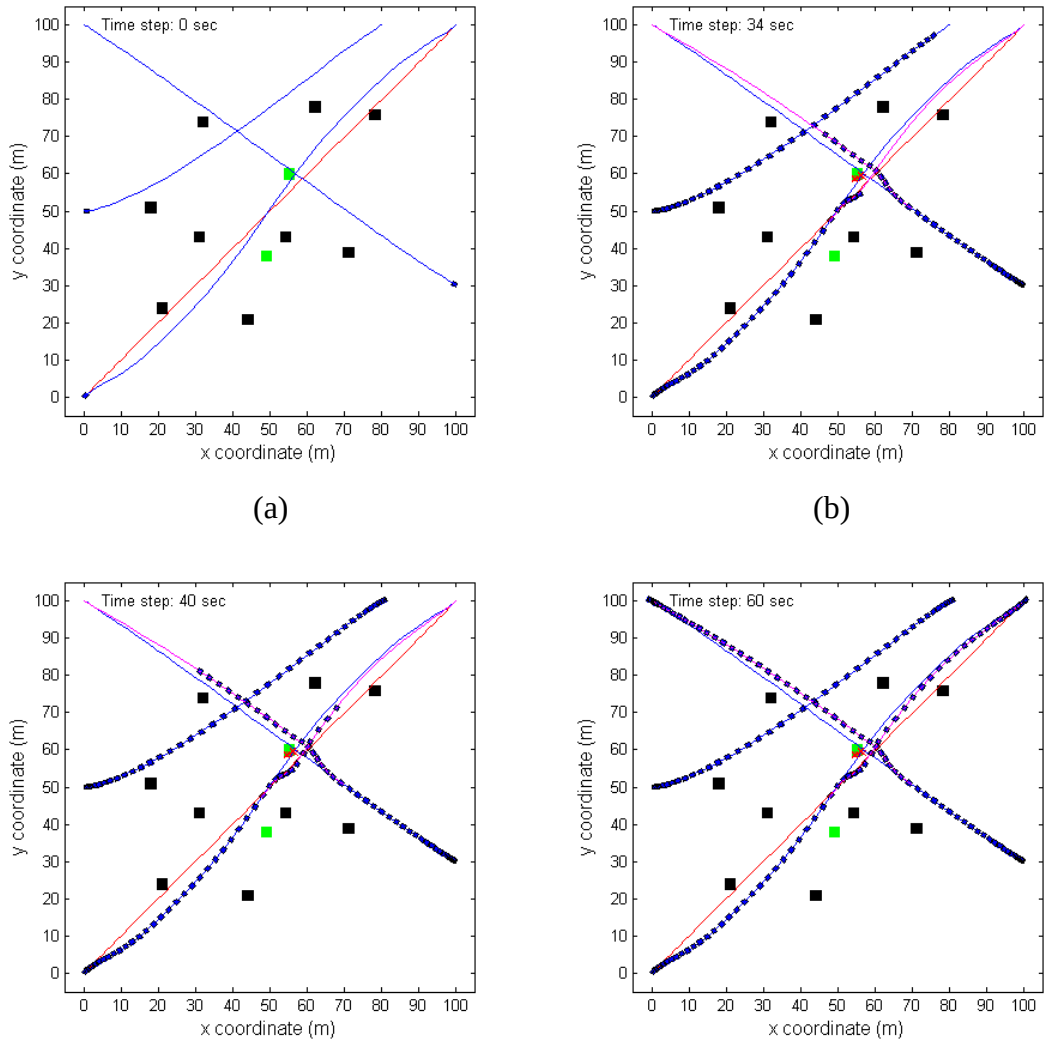


Figure 5.19 Three mobile robots navigate in the environment.

The comparisons between the planned and actual orientation, steering angle, velocity and location for Robot 1 and Robot 2 are similar to Case 1 and Case 2, respectively. For Robot 3, the comparisons are shown in Figure 5.20. The planned and actual trajectory for Robot 3 is almost identical because Robot 3 did not encounter any unknown static obstacles. The actual position of the mobile robot at the final point is tabulated in Table 5.9.

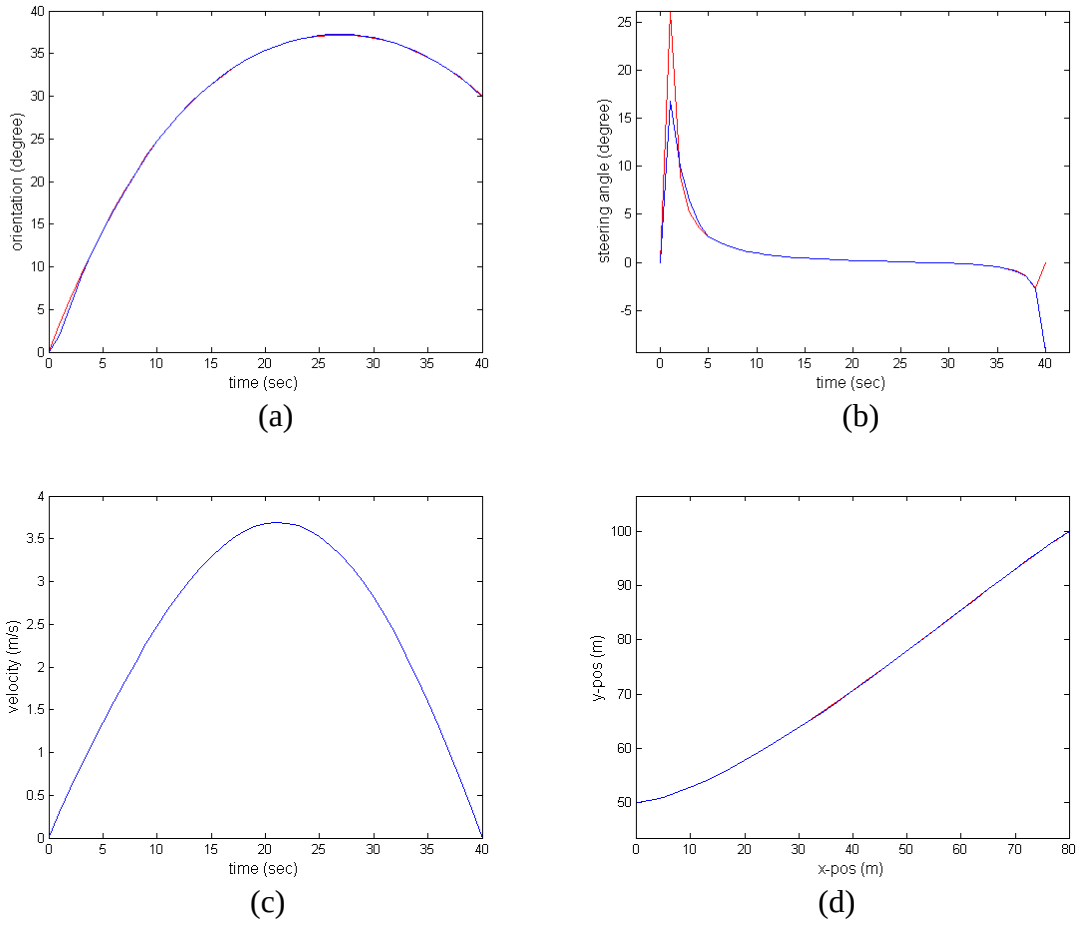


Figure 5.20 Robot 3: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position.

Table 5.9 Actual data collected at the final point for Case 3

Point		t (sec)	x (m)	y (m)	Θ (^{o})	$\varnothing$ (^{o})	v (m/s)
Robot 1	Planned	60	100	100	45	0	0
	Actual	60	99.93	99.93	45.15	17.16	0
Robot 2	Planned	60	0	100	-30	0	0
	Actual	60	-0.0057	100	-29.98	-1.84	0
Robot 3	Planned	40	80	100	30	0	0
	Actual	40	79.94	99.97	29.86	-9.33	0

From the results of Case 1, Case 2 and Case 3, the mobile robots were safely reached the final points with the capability to avoid known and unknown static obstacles. They have tried to follow the planned trajectories closely. However, there is a slight

different in orientation and steering angle of the mobile robots due to limitation of the steering angle. For example, at the beginning of the journey, steering angle for Robot 3 is 25^{o}. Thus the algorithm has adjusted the steering angle to 15^{o} so that the limitation is not exceeded and the mobile robot can turn smoothly. The actual positions of all the mobile robots at every time step are close to the planned position and the actual positions at the final point are summarized in Table 5.9. The maximum relative error for *x*-position is 0.07 m, *y*-position is 0.07 m, orientation is 0.15^{o} and steering angle is 17.16^{o}.

5.5.2 Navigation in dynamic and open-space environments

In this section, the previous examples are extended to dynamic environments. Moving obstacles are added into the environment and a series of simulation cases are presented in order to investigate the capability of the algorithms. In this study, the moving obstacle is a mobile robot that moves along the predefined path.

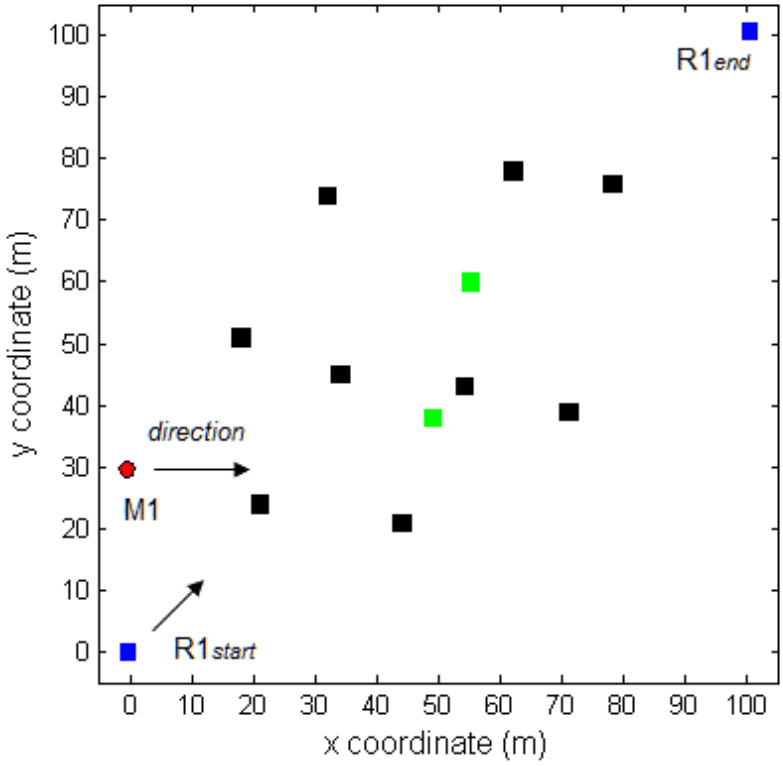


Figure 5.21 Simulated environment for Case 4

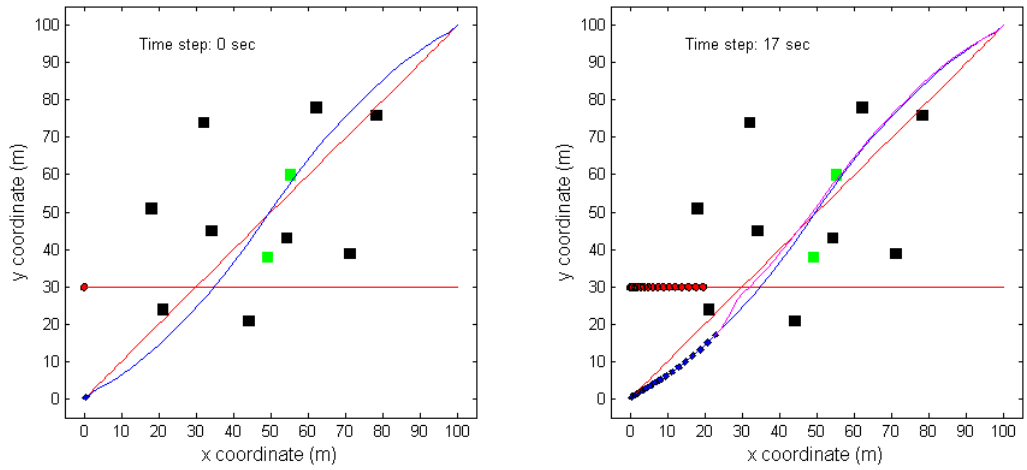
The first scenario (Case 4) discussed in this section involves one mobile robot and one moving obstacle in a complicated obstructed environment as shown in Figure 5.21.

The input data for the mobile robot is tabulated in Table 5.10. The moving obstacle started from the left-hand side of the map.

Table 5.10 Input data for simulation Case 4

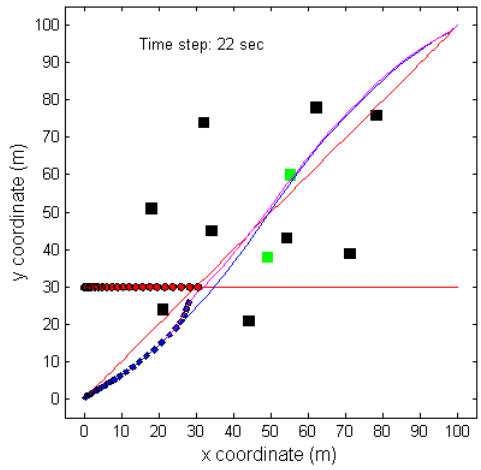
Point	t (sec)	x (m)	y (m)	Θ (^{o})	$\emptyset$ (^{o})	v (m/s)
Initial	0	0	0	45	0	0
Final	60	100	100	45	0	0

Figure 5.22 shows the simulation results for Case 4. The initial trajectory for the mobile robot and the moving obstacle is represented by red line as in Figure 5.22(a). However after consideration of known static obstacles in the environment, the new initial trajectory for the mobile robot is generated and represented by blue line. Once the trajectory generation is completed, the mobile robot starts to move along the trajectory. At 17 seconds, the mobile robot detects a moving obstacle as shown in Figure 5.22(b). The algorithm for avoiding the moving obstacle is executed and the new deviation trajectory is generated. As in Figure 5.22(c) and (d), the mobile robot starts to avoid the moving obstacle and reduce its speed to ensure the moving obstacle avoided before the mobile robot increase its speed to catch up the time lost during avoiding the obstacle. The mobile robot will continue to navigate along the new generated trajectory until it detects an unknown static obstacle at 31 seconds as shown in Figure 5.22(e). Then the obstacle avoidance algorithm for avoiding a static obstacle is executed and a new deviated trajectory is generated. The mobile robot will continue to navigate along the new trajectory and reaches the final point as shown in Figure 5.22(f).

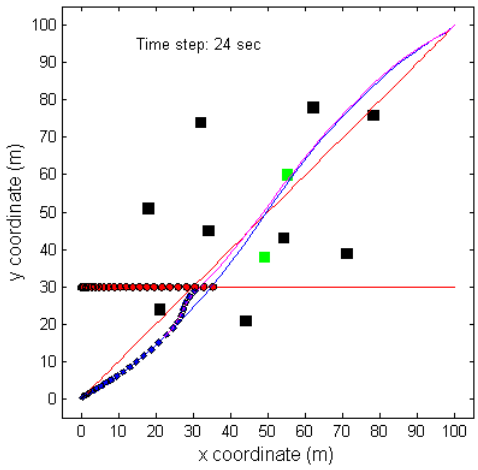


(a)

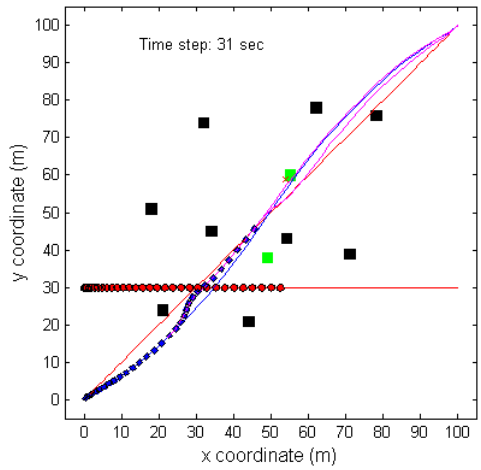
(b)



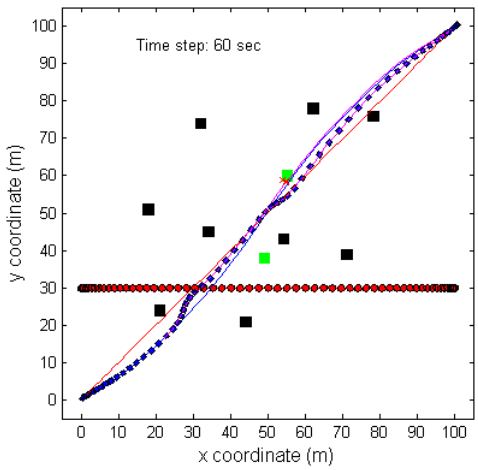
(c)



(d)



(e)



(f)

Figure 5.22 One mobile robot navigates in a dynamic environment.

The comparisons between the planned and actual orientation, steering angle, velocity and location for Robot 1 are shown in Figure 5.23. The actual position of the mobile robot at the final point is tabulated in Table 5.11.

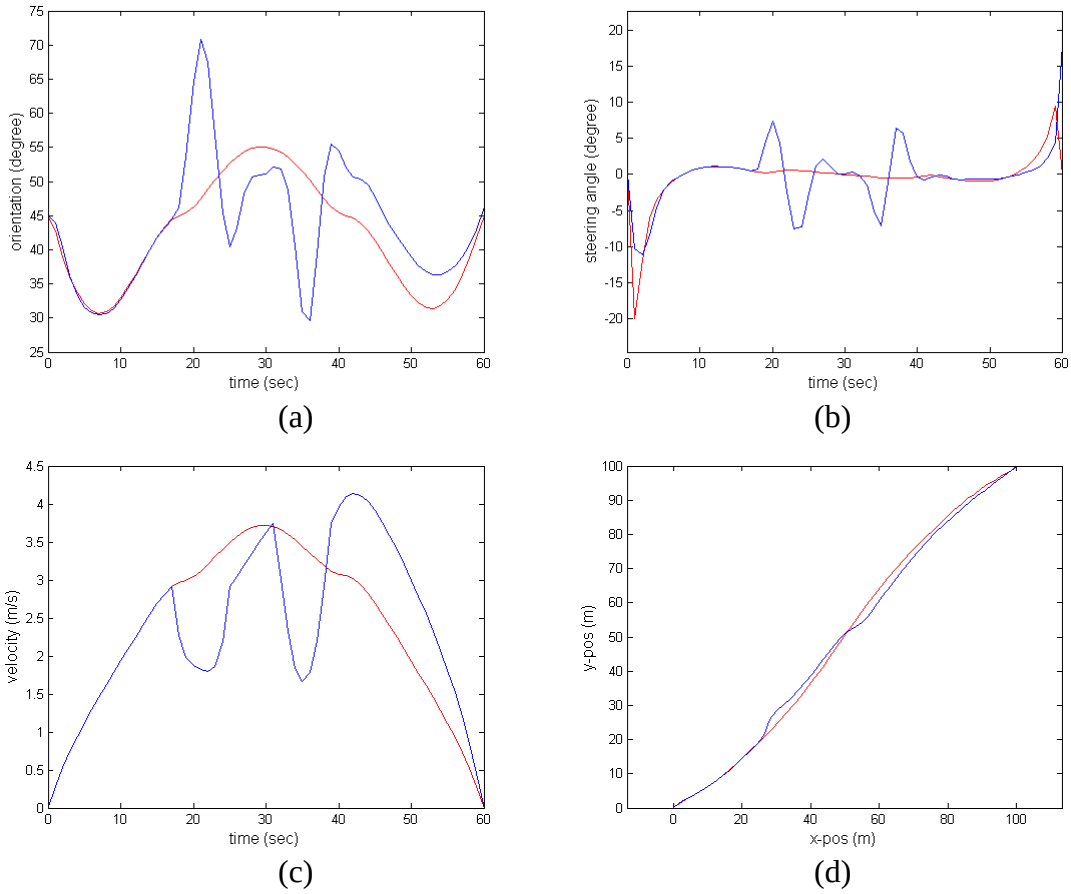


Figure 5.23 Robot 1: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position.

Table 5.11 Actual data collected at the final point for Case 4

Point	t (sec)	x (m)	y (m)	Θ (^{o})	$\varnothing$ (^{o})	v (m/s)
Planned	60	100	100	45	0	0
Actual	60	99.98	99.97	46.21	18.06	0

The next two cases involve multiple mobile robots and multiple moving obstacles in dynamic environment. In the second case, two mobile robots and two moving obstacle are used as shown in Figure 5.24. The first mobile robot, *R1* started from the bottom-left of the map and the second mobile robot, *R2* started from the right-hand side of the map. The first moving obstacle, *M1* started from the left-hand side of the map which the initial point is (0,30) and the second moving obstacle, *M2* started from the right hand side of the map which the initial point is (100,70).

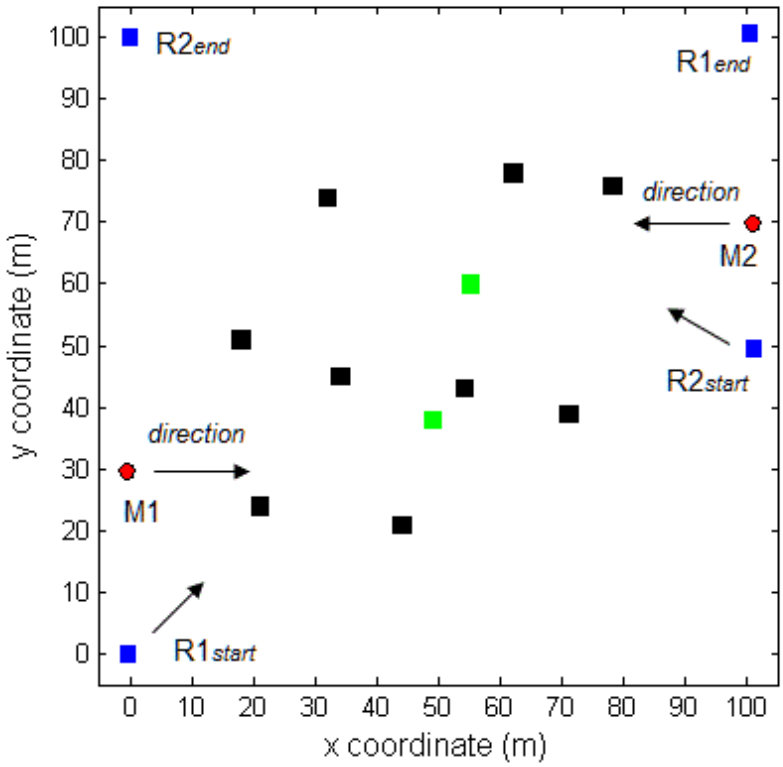


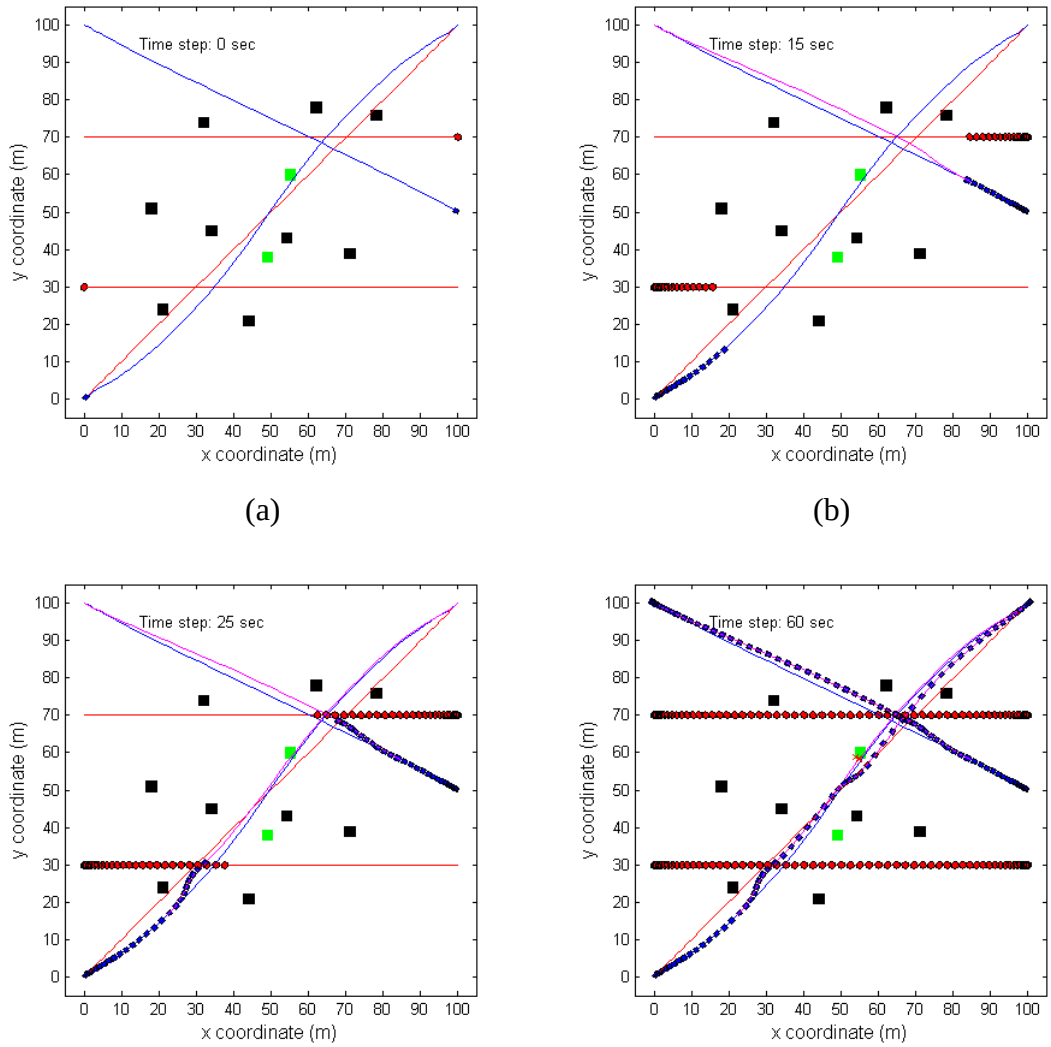
Figure 5.24 Simulated environment for Case 5

The input data for each mobile robot are tabulated in Table 5.12.

Table 5.12 Input data for simulation Case 5

Point		*t* (sec)	*x* (m)	*y* (m)	Θ (°)	$\emptyset$ (°)	*v* (m/s)
Robot 1	Initial	0	0	0	45	0	0
	Final	60	100	100	45	0	0
Robot 2	Initial	0	100	50	-30	0	0
	Final	60	0	100	-30	0	0

Figure 5.25 shows the simulation results for Case 5. As we can see, the initial trajectory for mobile robot *R2* is represented only by the blue line since there is no obstacle along the original initial trajectory as shown in Figure 5.25(a). Once the offline planning is completed, both mobile robots start to navigate along their trajectories. At 15 seconds, mobile robot *R2* detects a moving obstacle, which is moving obstacle *M2*, as shown in Figure 5.25(b). Then the algorithm for avoiding a moving obstacle is executed and the new deviated trajectory is generated. As we can see in Figure 5.25(c), mobile robot *R2* is slowing down in order to ensure the moving obstacle passes. In the case of mobile robot *R1*, the result is similar to the previous simulation result. Finally, both mobile robot *R1* and *R2*, safely reach the final point as shown in Figure 5.25(d).

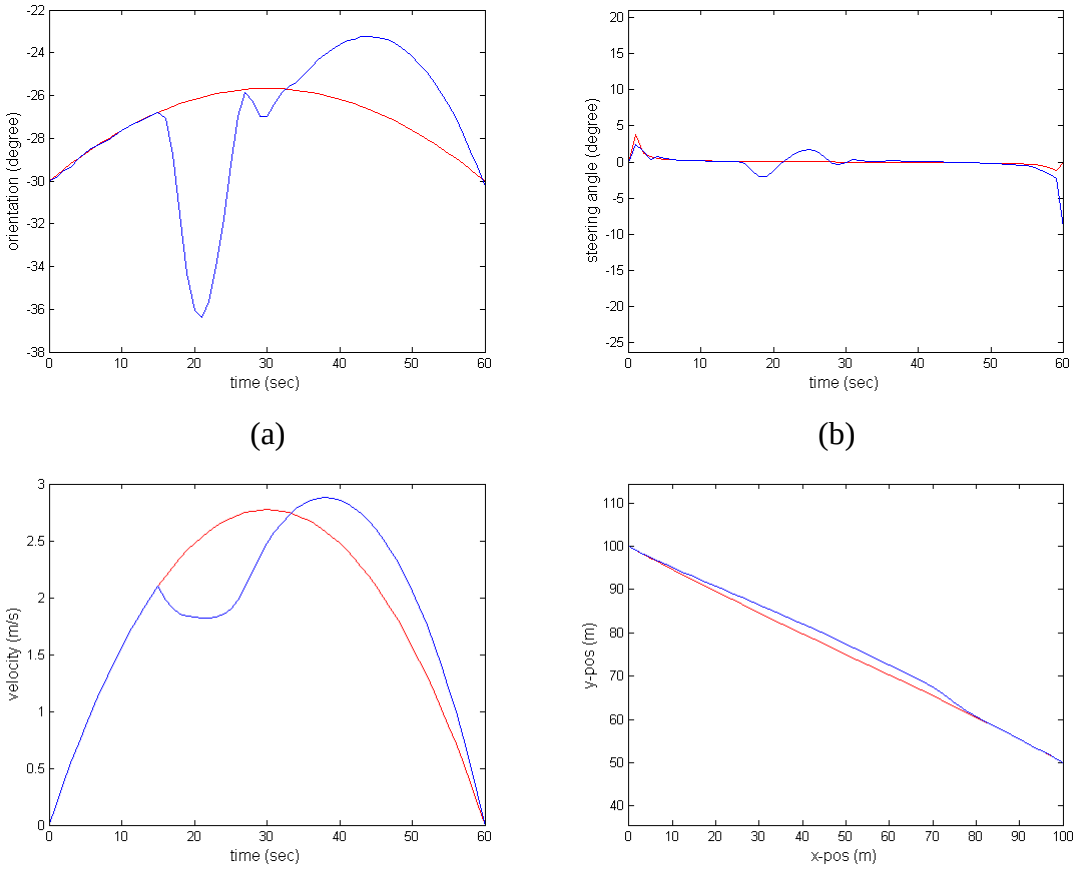


(a) (b)

(c) (d)

Figure 5.25 Two mobile robots navigate in a dynamic environment.

The comparisons between the planned and actual orientation, steering angle, velocity and location for Robot 1 are similar to Case 4. For Robot 2, the comparisons are shown in Figure 5.26. The actual position of the mobile robot at the final point is tabulated in Table 5.13.



(a) (b)

(c) (d)

Figure 5.26 Robot 2: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position.

Table 5.13 Actual data collected at the final point for Case 5

Point		t (sec)	x (m)	y (m)	Θ (^{o})	$\varnothing$ (^{o})	v (m/s)
Robot 1	Planned	60	100	100	45	0	0
	Actual	60	99.98	99.97	46.21	18.06	0
Robot 2	Planned	60	0	100	-30	0	0
	Actual	60	0.012	99.98	-30.27	-8.99	0

In the third case, there are three mobile robots and two moving obstacles in the environment as shown in Figure 5.27. The initial points for both moving obstacles are set similar to the previous case. However, in this case, two mobile robots (*R2* and *R3*) are planned to meet each other at the final points at the desired time. This type of scenario may have implication in the real world such as goods exchange and goods delivery between robots at the same location.

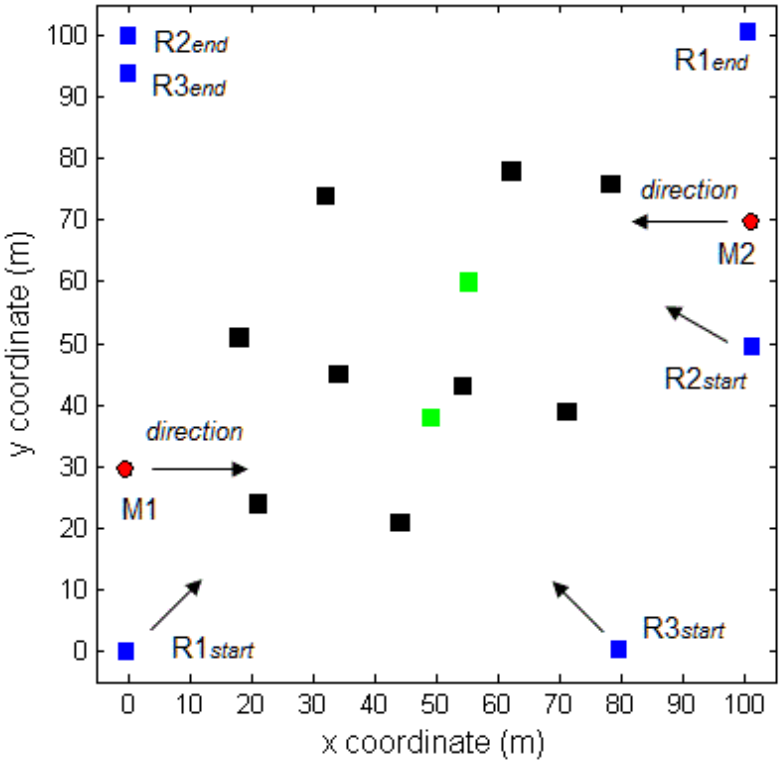


Figure 5.27 Simulated environment for Case 6.

The input data for all the mobile robots are tabulated in Table 5.14.

Table 5.14 Input data for simulation Case 6

Point		*t* (sec)	*x* (m)	*y* (m)	Θ (°)	$\emptyset$ (°)	*v* (m/s)
Robot 1	Initial	0	0	0	45	0	0
	Final	60	100	100	45	0	0
Robot 2	Initial	0	100	30	-30	0	0
	Final	60	0	100	-30	0	0
Robot 3	Initial	0	80	0	0	0	0
	Final	40	0	95	-45	0	0

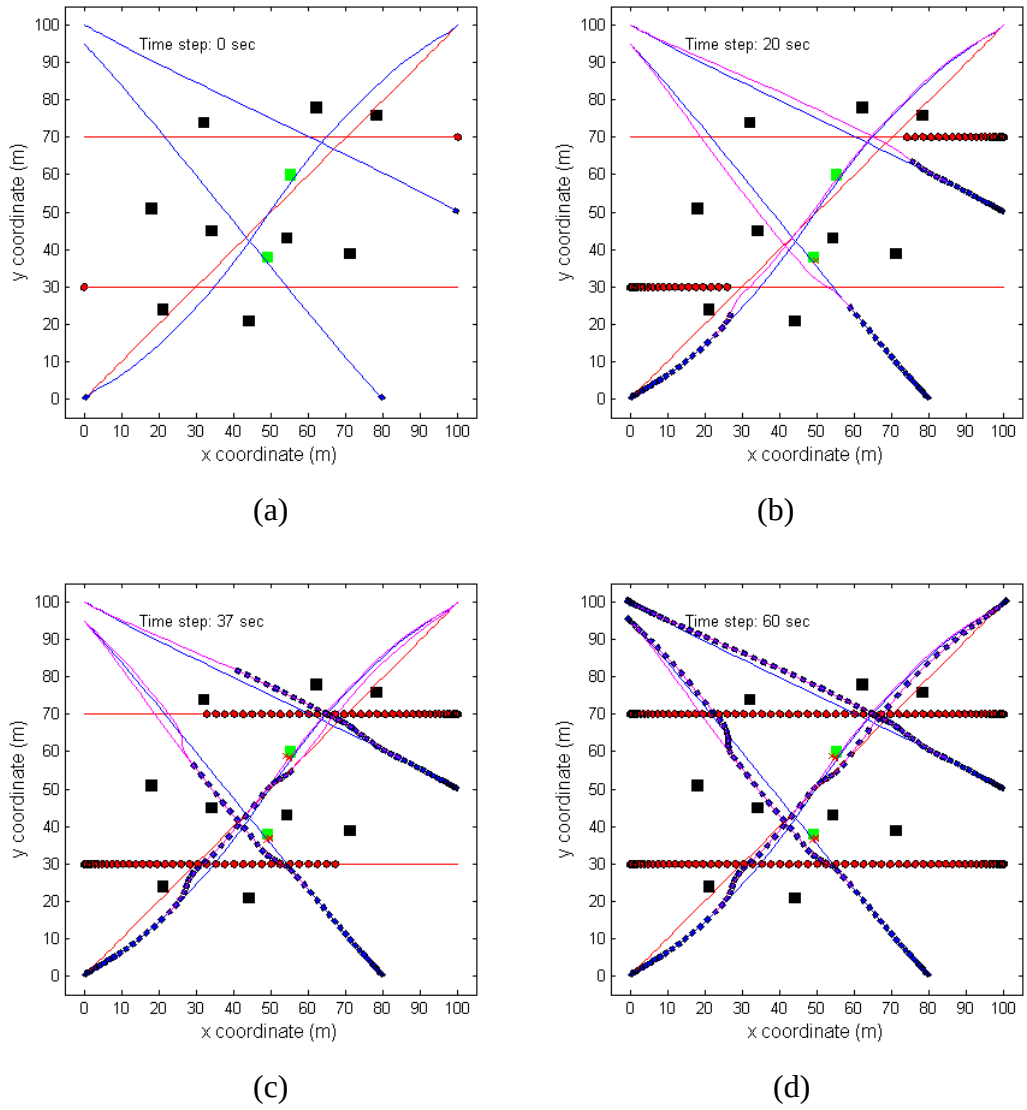
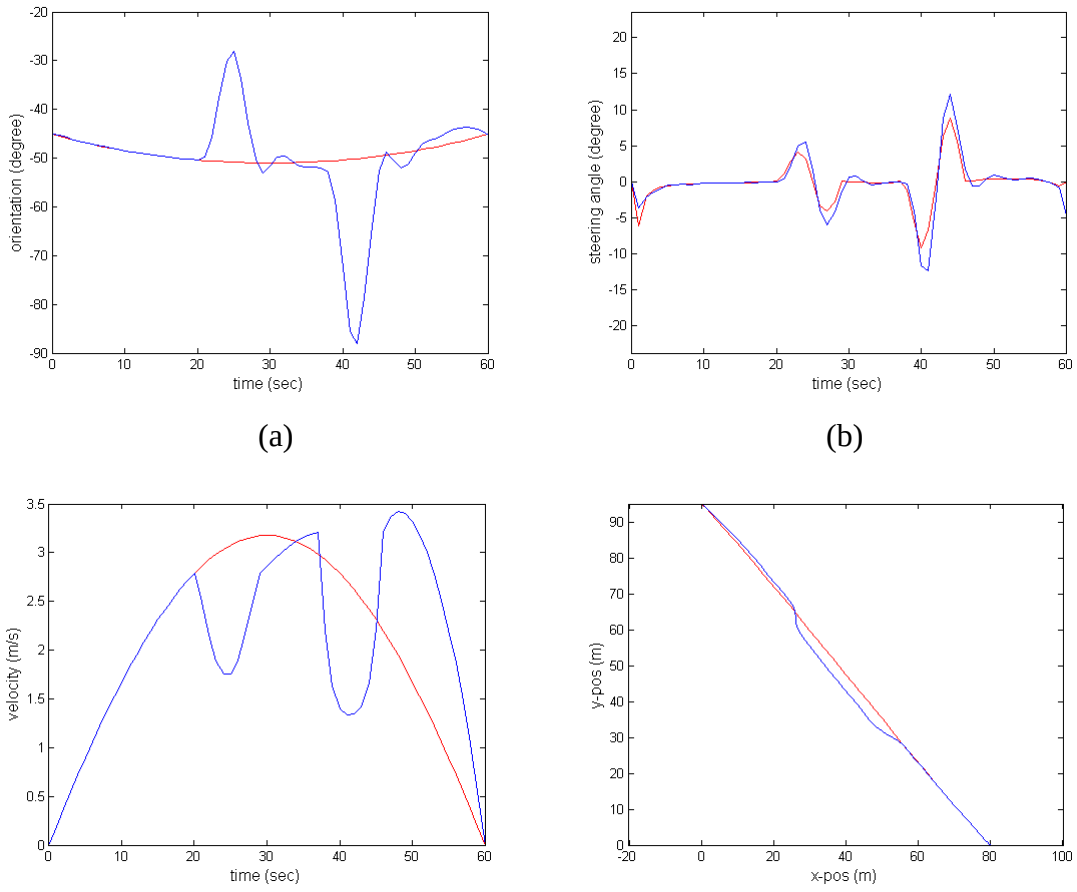


Figure 5.28 Three mobile robots navigate in a dynamic environment

Figure 5.28 shows the simulation results for Case 3. The initial generated trajectories for all the mobile robots are shown in Figure 5.28(a), which are represented by blue line. Once the offline planning is completed, all mobile robots start to move along their trajectories. In this case, we only discuss the movement of mobile robot R3 as the other two mobile robots' motions are similar to previous cases. At 20 seconds, mobile robot R3 detects an unknown static obstacle and a new deviated trajectory is generated. As we can see in Figure 5.28(b), there is a known static obstacle near to the newly generated trajectory. If the static obstacle is blocking the trajectory, the offline planning will be executed along with the obstacle avoidance algorithm. However, in this case, the known static obstacle is not blocking the way. Thus, the mobile robot R3 will continue its journey along the new trajectory. At 37 seconds, it detects a moving

obstacle (M2), which is coming from the right side, as shown in Figure 5.28(c). The algorithm for avoiding a moving obstacle is executed and a new deviated trajectory is generated. As we can see, mobile robot R3 reduce its speed in order to make sure the moving obstacle M2 passes. After that, mobile robot R3 continues its journey and reaches the final point, as shown in Figure 5.28(d).

The comparisons between the planned and actual orientation, steering angle, velocity and location for Robot 1 and Robot 2 are similar to Case 4 and Case 5, respectively. For Robot 3, the comparisons are shown Figure 5.29. The actual position of the mobile robot at the final point is tabulated in Table 5.15.



(a) (b)

(c) (d)

Figure 5.29 Robot 3: Planned (red) against actual (blue) plot for (a) orientation, (b) steering angle, (c) velocity, and (d) position.

Table 5.15 Actual data collected at the final point for Case 6

Point		*t* (sec)	*x* (m)	*y* (m)	Θ (^{o})	$\varnothing$ (^{o})	*v* (m/s)
Robot 1	Planned	60	100	100	45	0	0
	Actual	60	99.98	99.97	46.21	18.06	0
Robot 2	Planned	60	0	100	-30	0	0
	Actual	60	0.02	99.98	-30.27	-8.99	0
Robot 3	Planned	40	0	95	-45	0	0
	Actual	40	-0.0012	94.99	-45.29	-4.60	0

From the results of Case 1, Case 2 and Case 3, the mobile robots were safely reached the final points with the capability to avoid known and unknown static obstacles as well as dynamic obstacles. The results show that the algorithms are capable to detect and avoid not only static obstacles, but also dynamic obstacles. Furthermore the entire mobile robots were capable to follow the planned trajectories closely. The actual positions for the mobile robots are summarized in Table 5.15. The maximum relative error for *x*-position is 0.02 m, *y*-position is 0.03 m, orientation is 1.21^o and steering angle is 18.06^o.

5.5.3 Navigation in the city-like environments

In this section, the simulations are based on the multiple waypoints trajectory planning in a city-like environment as shown in Figure 5.30. All the parameters used for the mobile robots; R1 and R2, are listed in Table 5.16 and Table 5.17, respectively. North direction of the map is set pointing up on the map. As listed in Table 5.16, R1 starts from the bottom of the map at point (10, 20) and facing north. At the first junction, it needs to turn right. The first and second waypoints are set to ensure the mobile robot can turn at the junction smoothly. Then, it needs to move along the road until it reaches the second junction. It then needs to turn left and move until it reaches the final point (60, 170) at the 120^{th} second with 90^o orientation.

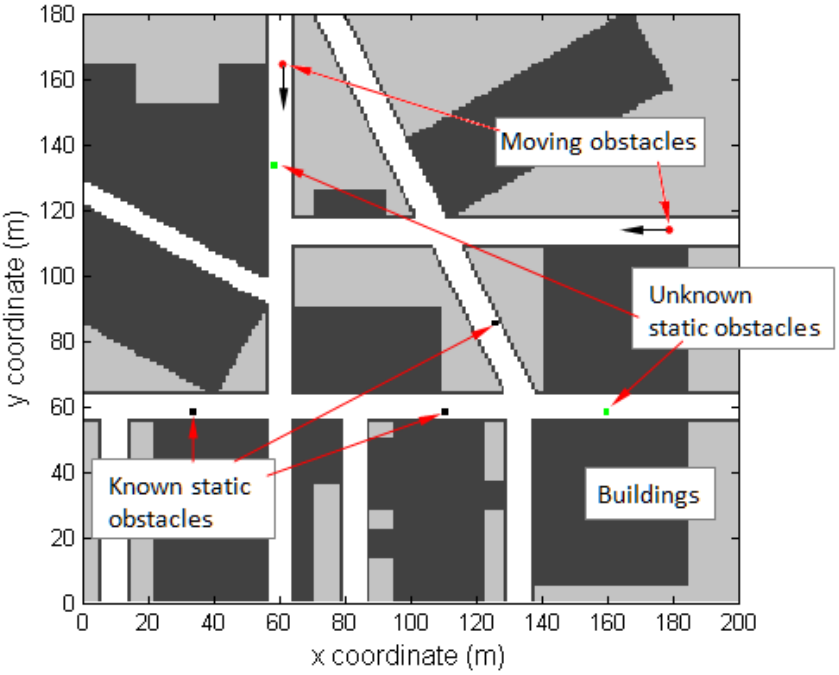
Table 5.16 Parameters for the first mobile robot (R1)

Points	*t* (sec)	*x* (m)	*y* (m)	Θ (o)	$\emptyset$ (o)	*v* (m/s)
1	0	10	20	90	0	0
2	20	10	50	90	0	1
3	30	20	60	0	0	2
4	50	50	60	0	0	1
5	60	60	67	90	0	2
6	120	60	170	90	0	0

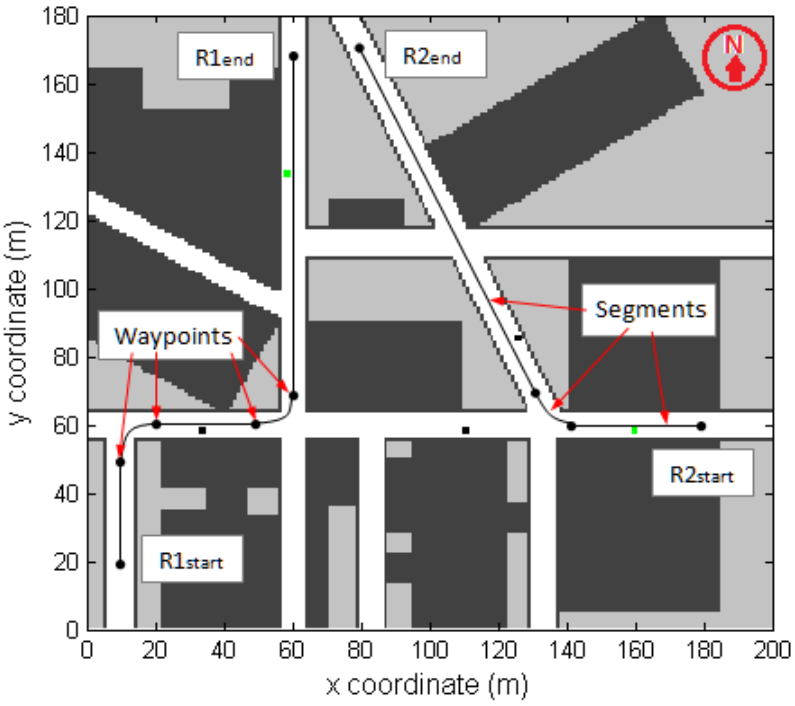
As listed in Table 5.17, R2 starts from the right side of the map at point (183,60) and facing west. Then at the junction, it needs to turn right and move along the road until it reaches the final point (80,170) at the 120^{th} second with 60^{o} orientation. Note that the road is tilted at about 60^{o} from x-axis.

Table 5.17 Parameters for the second mobile robot (R2)

Points	*t* (sec)	*x* (m)	*y* (m)	Θ (^{o})	$\emptyset$ (^{o})	*v* (m/s)
1	0	183	60	0	0	0
2	30	143	60	0	0	1
3	40	133	65	-60	0	1
4	120	80	170	-60	0	0



(a)



(b)

Figure 5.30 (a) A simplified city-like map, (b) Multiple waypoints trajectory planning.

In addition, there are two moving obstacles in the map as shown in Figure 5.31. The first moving obstacle (M1) starts from the north of the map and moves straight down to the south of the map. The initial and final point for M1 is (61, 180) and (61, 80), respectively. The second moving obstacle starts from east of the map and finishes at the middle of the map. The initial and final point for M2 is (180, 113) and (90, 113), respectively. Both moving obstacles move from their respective initial points and reach their final points at the 100th second.

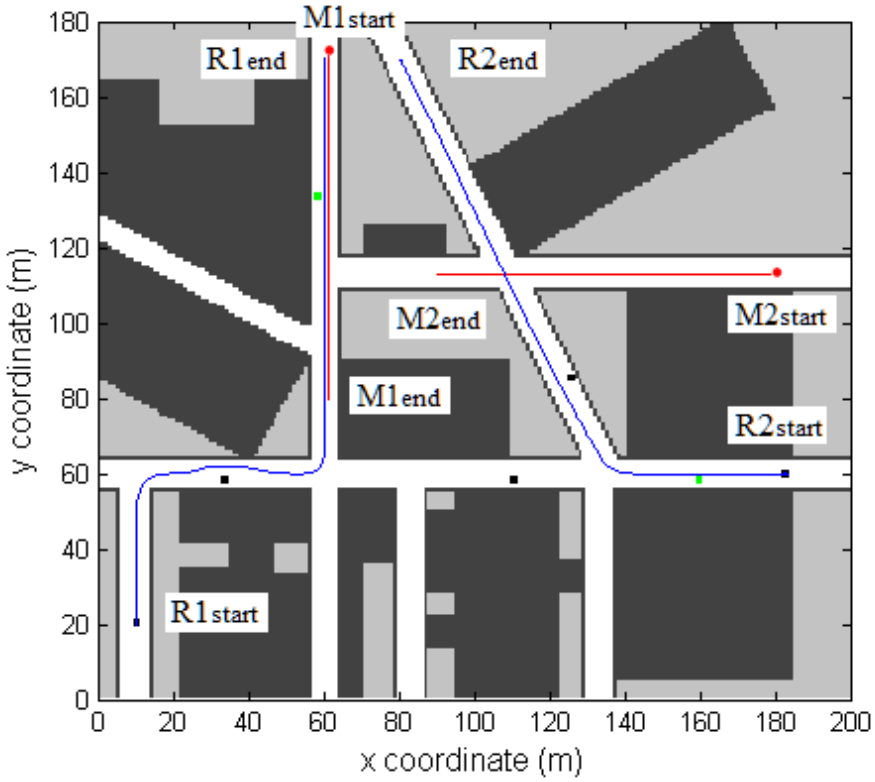


Figure 5.31 Initial trajectories in a city-like map.

The initial trajectories for mobile robots and moving obstacles are shown in Figure 5.31. Once all the trajectories were generated, the mobile robots and the moving obstacles were started to move along their respective trajectories. At the 9th second, R2 detected an unknown static obstacle as shown in Figure 5.32(b). Then a new trajectory was generated from the detection point to the closest waypoint, which was in this case the first waypoint. R2 started to move along the new trajectory as shown in Figure 5.32(c) and reached the first waypoint at the 30th second as shown in Figure 5.32(d).

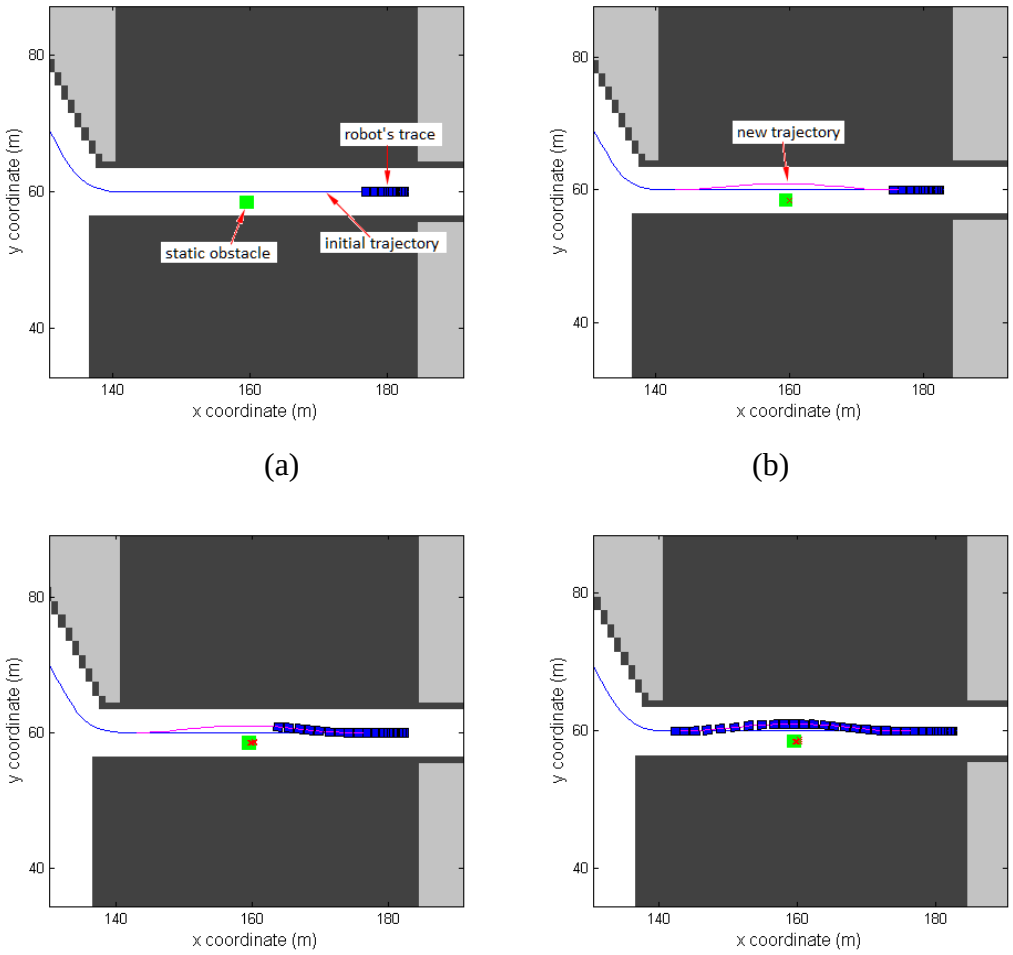


Figure 5.32 (a) Before detecting an obstacle. (b) Obstacle detected at the 9th second. (c) Starts to move along new trajectory. (d) Reaches the first waypoint at the 30th second.

Furthermore, at the 67th second, R2 detected a moving obstacle (M2) coming from the right side of it as shown in Figure 5.33(b). It then predicts whether it might collide with the moving obstacle or not. In this case, collision is expected to happen and a new trajectory is generated from detection point to the closest waypoint, which is the final point, based on the obstacle avoidance algorithm of a moving obstacle. Then R2 started to move along the new trajectory as shown in Figure 5.33(c). Also as we can see, the mobile robot actually slowed down to cautiously passing through the moving obstacle as shown in Figure 5.33(d).

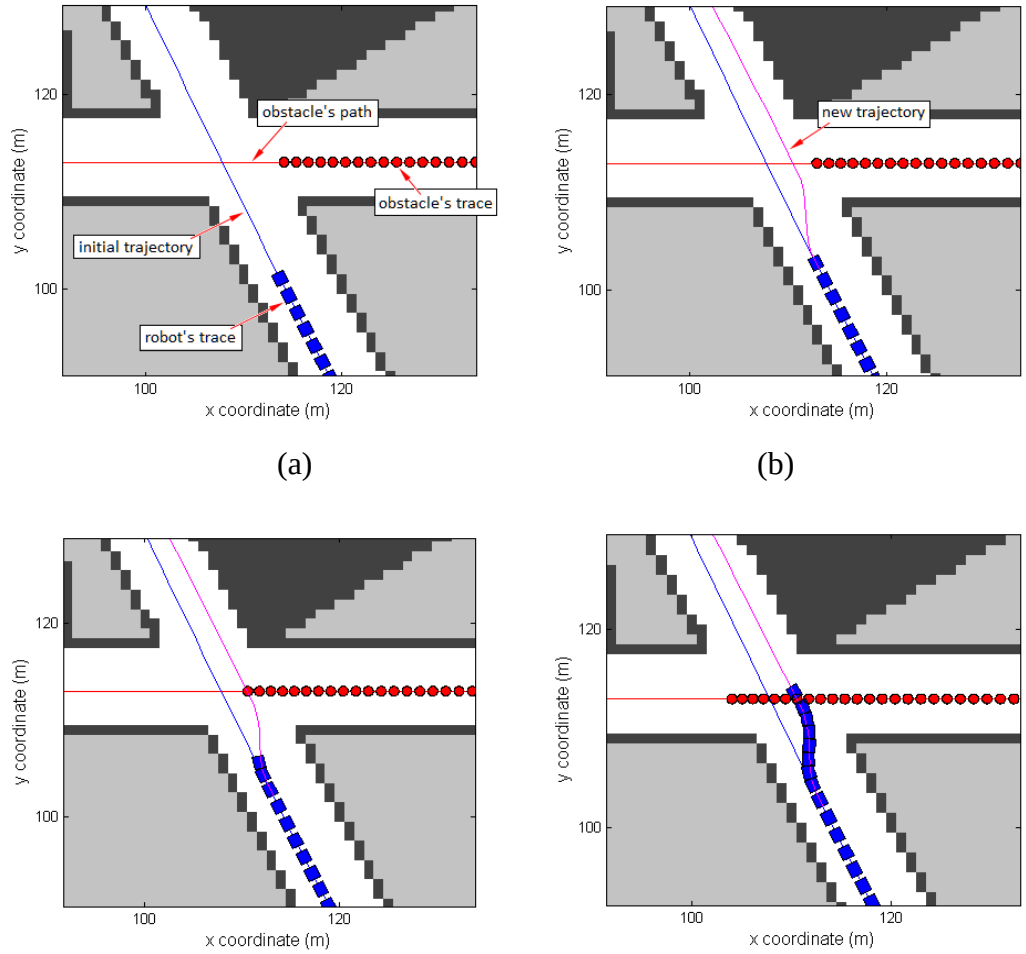


Figure 5.33 (a) Before detecting an obstacle. (b) Obstacle detected at the 67th sec. (c) Starts to move along new trajectory. (d) Passes through moving obstacle safely.

As we can see in Figure 5.31, the initial trajectory for R1 was already considered known static obstacles during offline planning. Then R1 started to move along the initial trajectory and passes through all the waypoints. However at the 68th second, R1 detected a moving obstacle (M1) as shown in Figure 5.34(b). Also R1 checked the direction of moving obstacle and in this case, M1 came from the opposite direction of R1. Therefore, M1 was treated as a static obstacle and a new trajectory was generated from the current point to the final point, through the deviation point. Then R1 started to move along the new trajectory and safely avoided M1 as shown in Figure 5.34(c) and (d). Furthermore, after avoiding the moving obstacle, R1 detected an unknown static obstacle at the 86th second and successfully avoided it. Figure 5.35 shows the final overall simulation results at the 120th second.

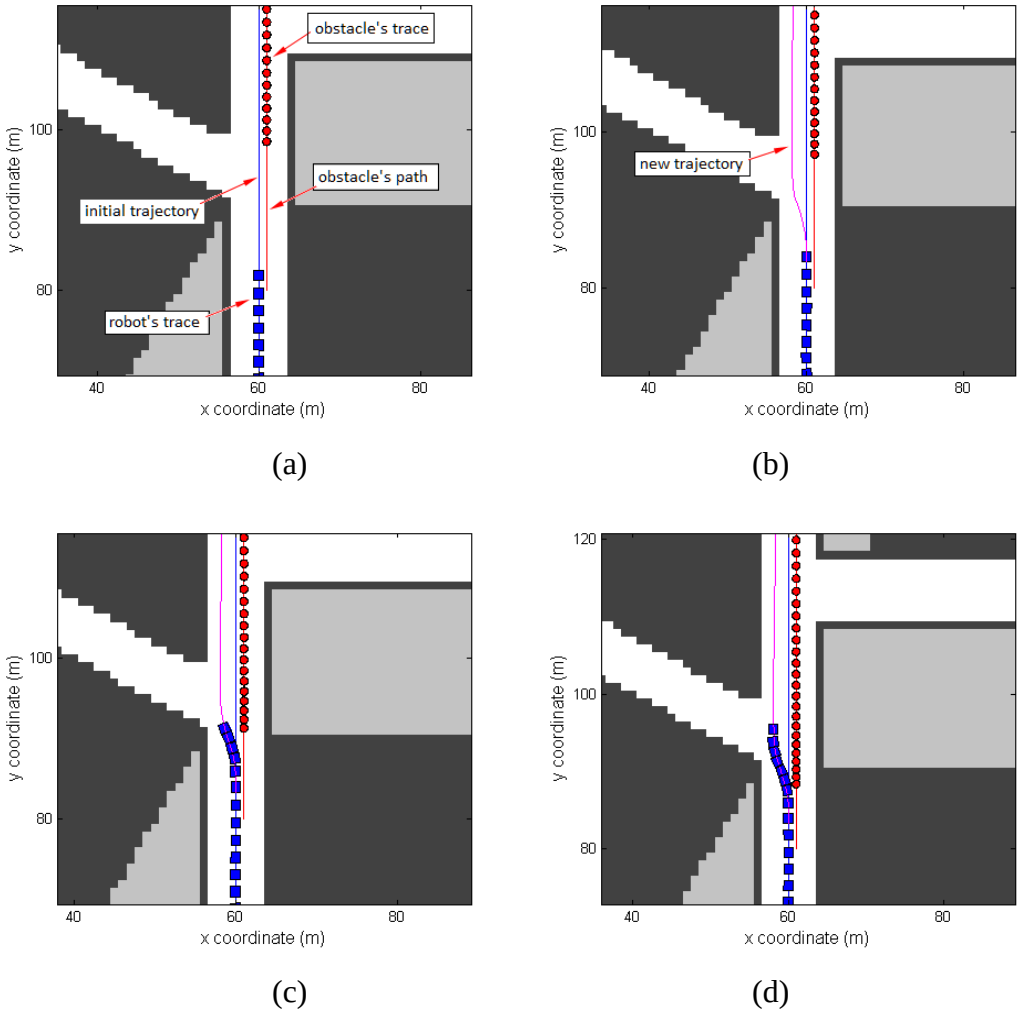


Figure 5.34 (a) Before detecting an obstacle. (b) Obstacle detected at the 68th sec. (c) Starts to move along new trajectory. (d) Passes through moving obstacle safely.

As we can see, both mobile robots reached the final point at the specified time, position and orientation with certain errors as shown in Table 5.18. The errors are reasonably small as a result of the online planning approach. At every time step, the online planner will use the actual data to get to the next pre-planned position of the mobile robot. This means the planner will need to determine a new steering angle using the actual position and orientation, and the pre-planned velocity of the mobile robot. This practice will eliminate or at least reduce the errors at every time step. Furthermore, the mobile robots successfully passed through all the waypoints and avoided all the static and moving obstacles.

Table 5.18 Table 3 Errors for Case 1 at final point.

	Actual			Relative error		
	x (m)	y (m)	Θ (^{o})	x (m)	y (m)	Θ (^{o})
Robot 1	59.999	169.999	89.427	0.001	0.001	0.573
Robot 2	79.972	170.04	-62	0.028	0.04	2

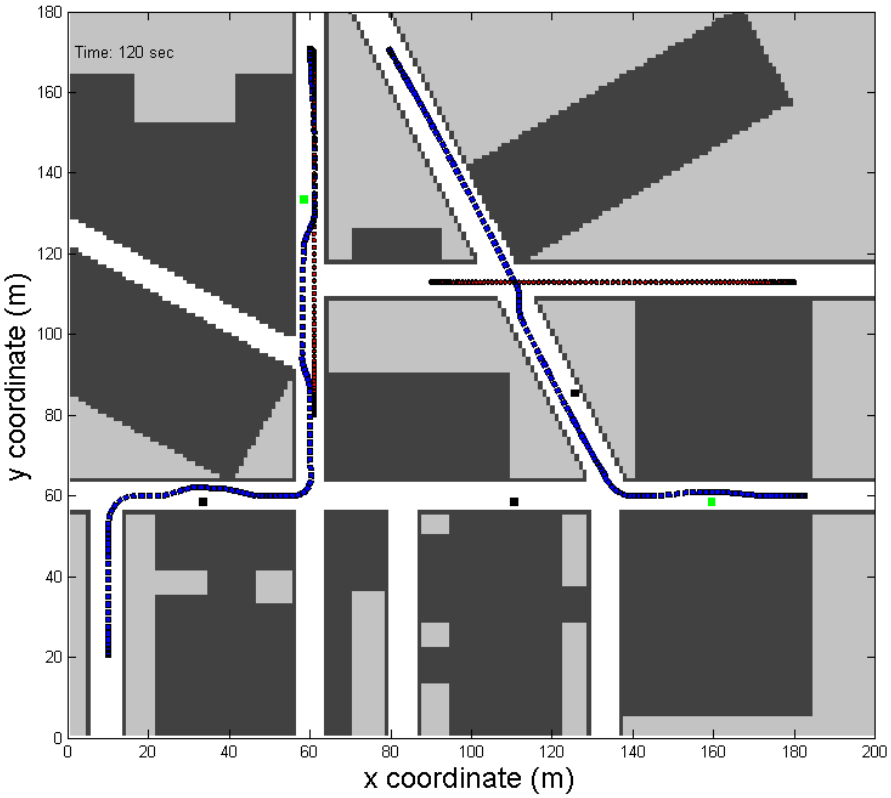


Figure 5.35 Final result at the 120th second.

In addition, two more simulation cases have been conducted to investigate the effectiveness of the algorithms. Figure 5.36 shows the second simulation scenario with two mobile robots, R1 and R2, and one moving obstacle, M1.

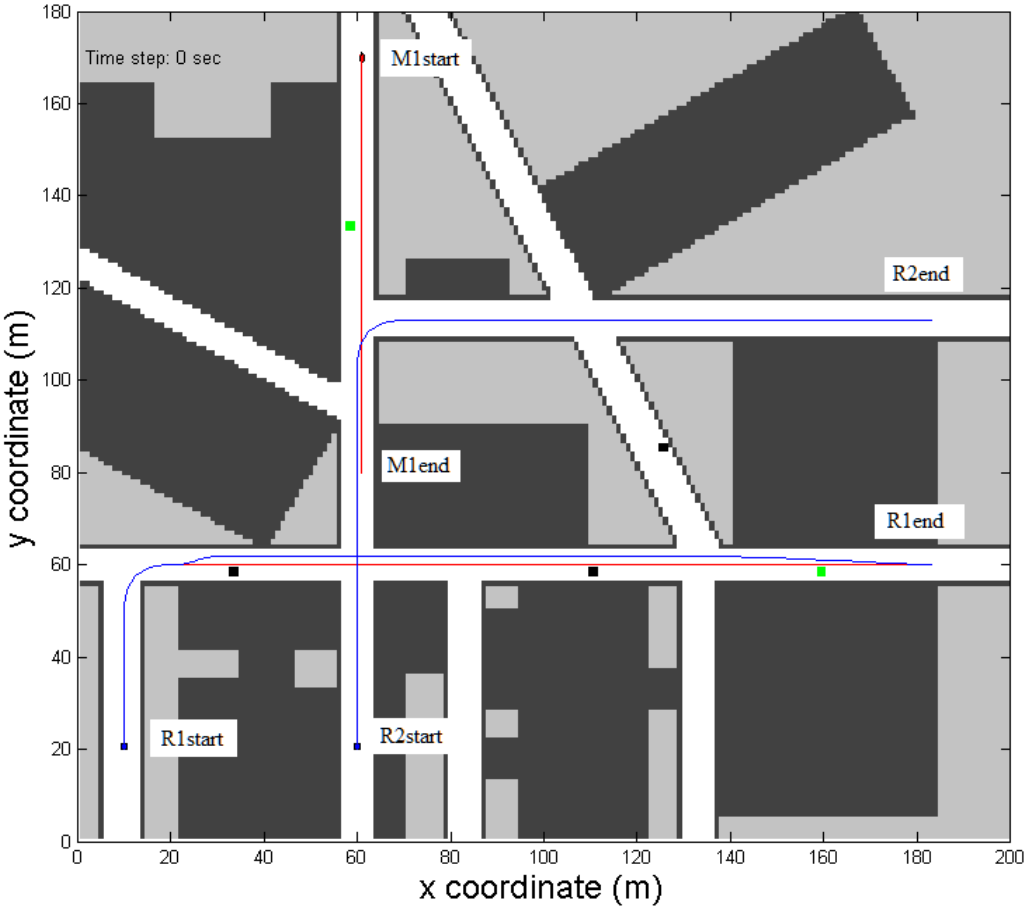


Figure 5.36 Second scenario with two mobile robots and one moving obstacle.

Using the similar map setup to the first case, the inputs for both mobile robots are tabulated in Table 5.19. As we can see in Figure 5.36, the pre-planned trajectories are presented by the blue line with the consideration of the known static obstacles. The moving obstacle is set to move along the road from initial point (61,170) to final point (61,80).

Table 5.19 Parameters for second simulation case

	Points	*t* (sec)	*x* (m)	*y* (m)	*Θ* (o)	*Ø* (o)	*v* (m/s)
Robot 1	1	0	10	20	90	0	0
	2	30	10	50	90	0	1
	3	40	20	60	0	0	1
	4	120	183	60	0	0	0
Robot 2	1	0	60	20	90	0	0
	2	50	60	103	90	0	1
	3	60	70	113	0	0	1
	4	120	183	113	0	0	0

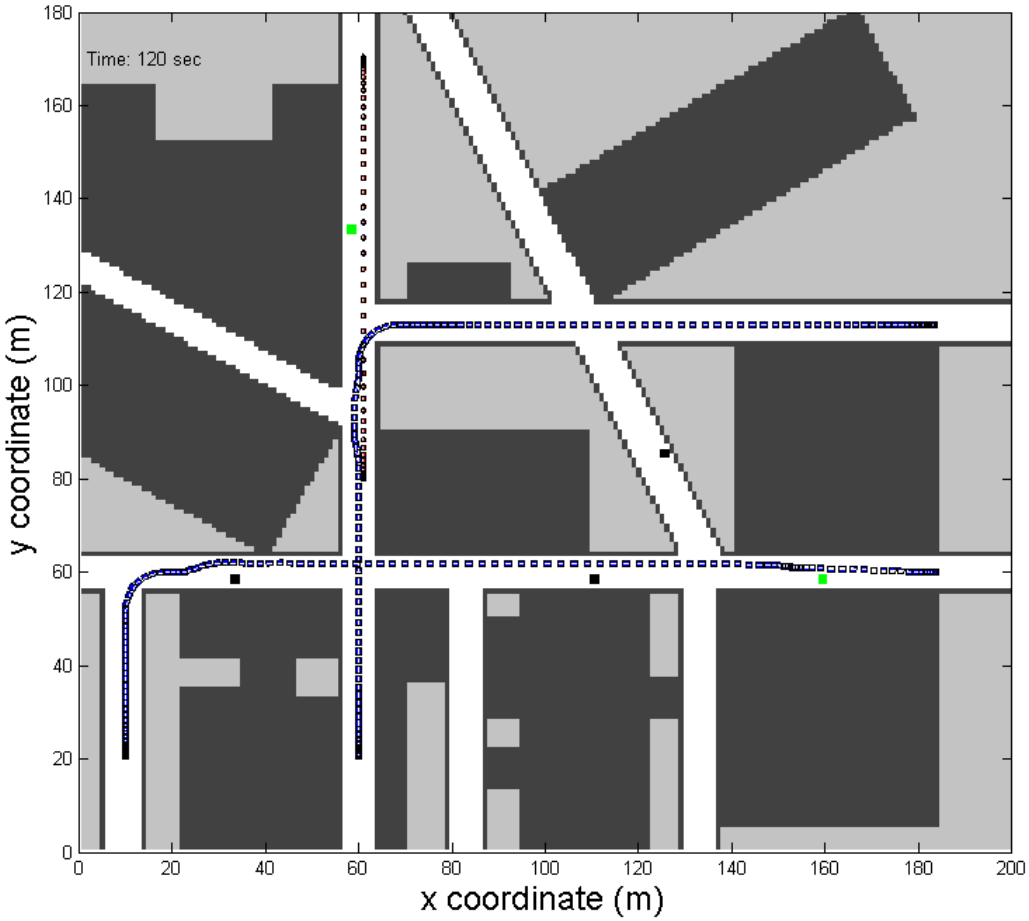


Figure 5.37 Final result at the 120th second for second scenario.

The final results are shown in Figure 5.37. As we can see in the final results, the mobile robots are capable to navigate safety and avoid the moving obstacle as well as the static obstacles and reach the final points at the specified time. The errors at the final point are tabulated in Table 5.20. The errors are reasonable and still fall within the satisfactory limits as the mobile robots do not deviate too far from the final points, considering the distance that the mobile robots have travelled. The maximum final errors for positions are 0.06 m for R1 and 0.37 m for R2, while the maximum orientation errors are 0.818^{o} for R1 and 0^{o} for R2.

Table 5.20 Errors for Case 2 at the final point.

	Actual			Relative error		
	x (m)	y (m)	Θ (^{o})	x (m)	y (m)	Θ (^{o})
Robot 1	183.06	60	0.818	0.06	0	0.818
Robot 2	182.63	113	0	0.37	0	0

In the third case, the scenario is extended with three mobile robots and two moving obstacles are used as shown in Figure 5.38. The mobile robots started at the different initial points and moved to the different final points as tabulated in Table 5.21. The travel time for each mobile robot is set to 100 second. This case is conducted to demonstrate the capability of the algorithms to handle the different travel time and to demonstrate multiple robots coordination in the unknown environment.

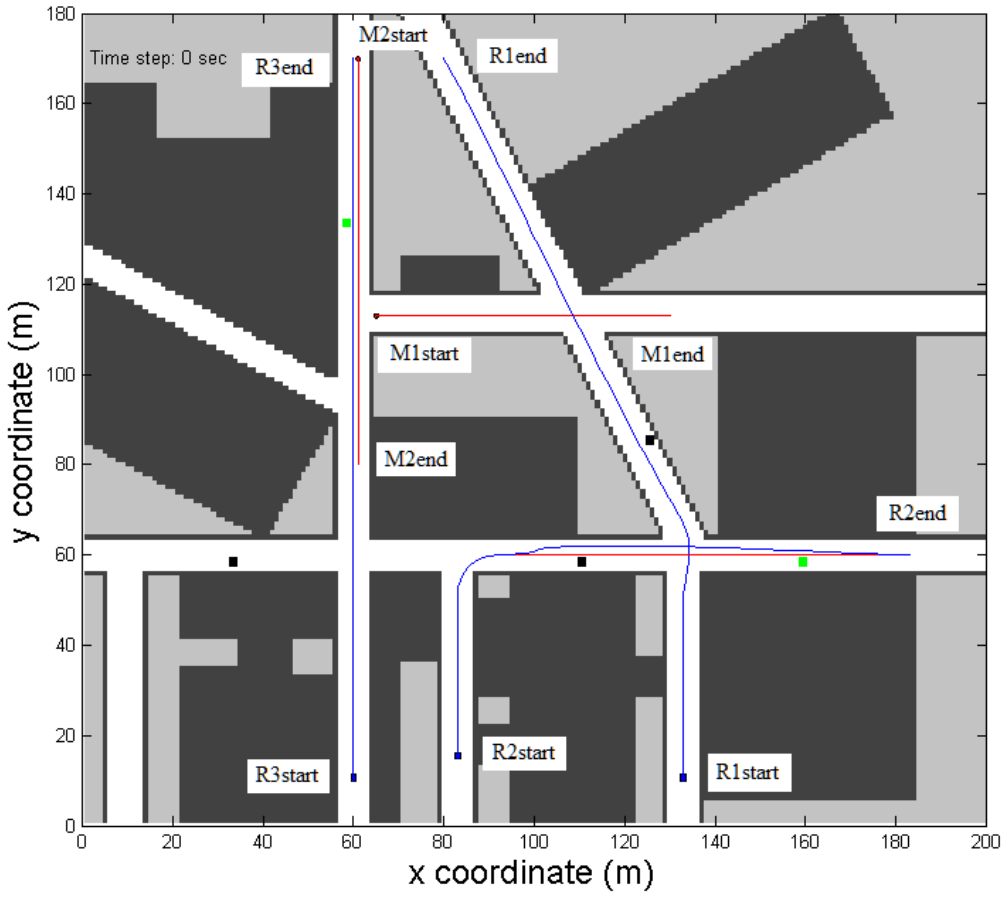


Figure 5.38 Third scenario with three mobile robots and two moving obstacles.

Table 5.21 Parameters for third simulation case

	Points	t (sec)	x (m)	y (m)	Θ (o)	$\emptyset$ (o)	v (m/s)
Robot 1	1	0	133	10	90	0	0
	2	30	133	50	90	0	1
	3	40	133	67	-60	0	1
	4	100	80	170	-60	0	0
Robot 2	1	0	83	15	90	0	0
	2	40	83	50	90	0	1
	3	50	93	60	0	0	1
	4	100	183	60	0	0	0
Robot 3	1	0	60	10	90	0	0
	2	100	60	170	90	0	0

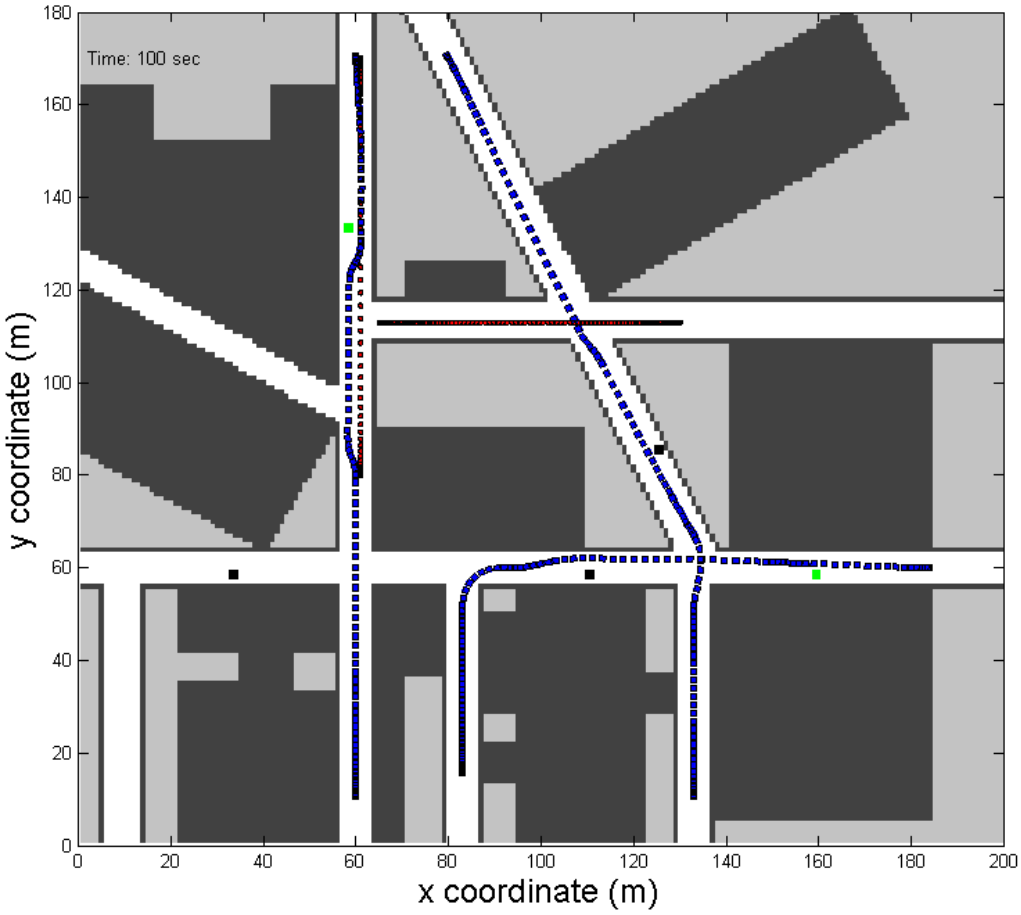


Figure 5.39 Final result at 100th second for third scenario.

From the final results are shown in Figure 5.39, all the mobile robots reached the final points at the specified time. As we can see the final errors in Table 5.22, the position errors for each robot are acceptable. Even though the orientation errors for R1 and R2 are larger than the second case, the results are still within the satisfactory limit as the maximum orientation error only 1.5°.

Table 5.22 Errors for Case 3 at the final point.

	Actual			Relative error		
	x (m)	y (m)	Θ (°)	x (m)	y (m)	Θ (°)
Robot 1	79.98	170.03	-58.53	0.02	0.03	1.47
Robot 2	183.06	59.999	1.042	0.06	0.001	1.042
Robot 3	59.999	169.96	90	0.001	0.04	0

5.6 Concluding remarks

In this chapter, the Matlab was adopted for development of simulations and to implement and test the algorithms for mobile robot navigation. The algorithm was tested through a series of the simulation setup. The maps for the simulation works were adopted from open-space environment and city-like environments which is more complicated.

The simulation results show the mobile robot was able to follow the planned trajectory as close as possible. Furthermore, the mobile robot was able to reach close to the final point at the desired time. The algorithm also was able to simulate the different environment setup for the mobile robot as well as for multiple robots with the presence of dynamic obstacles.

However, there are errors occurred between the planned trajectories and actual trajectories due to the actual calculation of the position and steering angle of mobile robots. For example, the actual calculation of the steering angle is taking into consideration of the current data at every time step to calculate the next motion of the mobile robot. Thus this cumulative error caused the slight different between the planned and actual at the final point.

Furthermore, the developed GUI framework ensures the user able to modify the settings of the mobile robot easily. The user only needs to modify the settings at the GUI framework without interfering the control functions of the algorithms.

6. DEVELOPMENT OF A NONHOLONOMIC MOBILE ROBOT

The development of a mobile robot is based on the application and requirement of the mobile robot in the environment. In this study, a nonholonomic mobile robot is used to navigate in the outdoor environment. Thus a car-like robot is preferred as it can be converted from a standard car and able to travel in the large outdoor environments.

Therefore, the mobile robot used for the experimental works was converted from a standard remote control car as shown in Figure 6.1. The wheelbase length and width of the mobile are 174 mm and 191 mm, respectively. It has a similar structure to the normal car with front steering wheels and rear driving wheels. All four wheels have the same diameter which is 69 mm. The rear wheels are conventional fixed wheels on the rear axle and the front wheels are centred turning wheels on the front axle. The steering wheels are assumed to turn at the same angle and acted as a single wheel located at the middle of the front axle as discussed in Chapter 4.

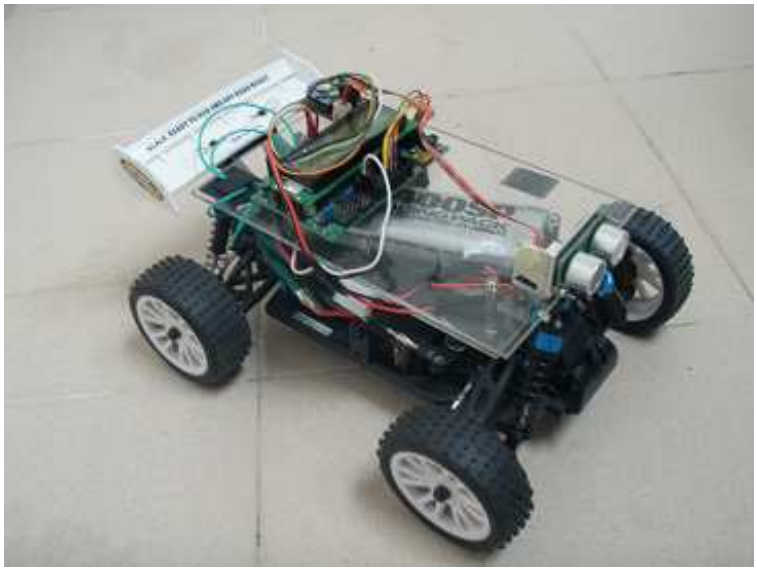


Figure 6.1 The modified car-like robot used in experimental works.

As the focused mobile robot is a nonholonomic mobile robot, an Ackermann steering robot is required as the mobile robot platform. Furthermore the selection of the mobile robot platform should fulfil a few selection criteria for this study such as:

- Steering should be driven by a digital servomotor.
- The driving wheel should be driven by a motor.
- Enough size to host all the sensors and the microcontroller.
- Easy access to all the components attached to the car.

The basis of the mobile robot platform is shown in Figure 6.2. The mobile robot has a motor that driven the rear wheels and acted as driving wheel, while the front wheels are steered by a servo motor and acted as an Ackermann steering wheels. In addition the RC car is powered by battery pack. This RC car needs to be modified in order to install all the sensors, microcontroller, battery pack and other accessories.

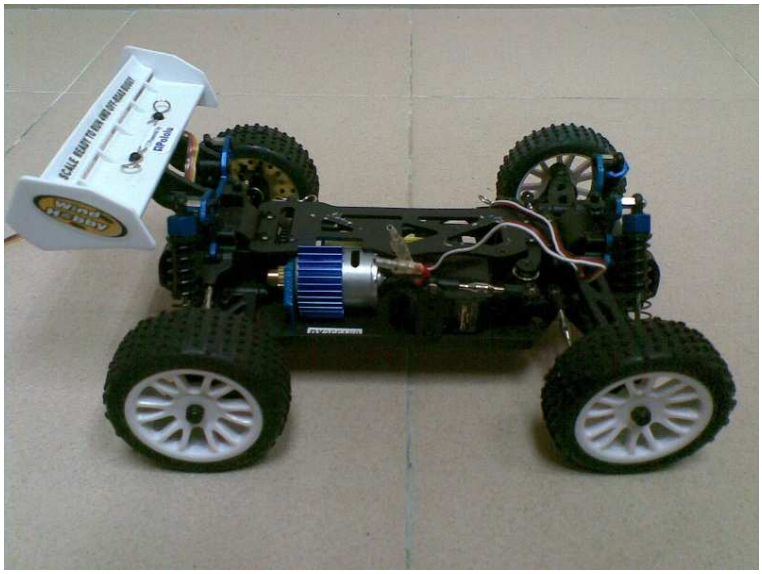


Figure 6.2 Mobile robot platform.

The sensor platform is constructed as shown in Figure 6.3. The platform is designed to be as simple as possible and the material used is acrylic. This material has a few advantages such as light weight and durable. The platform is a two tier platform which the lower tier is used to install battery pack and the upper platform is used to install sensors, microcontroller and other accessories.

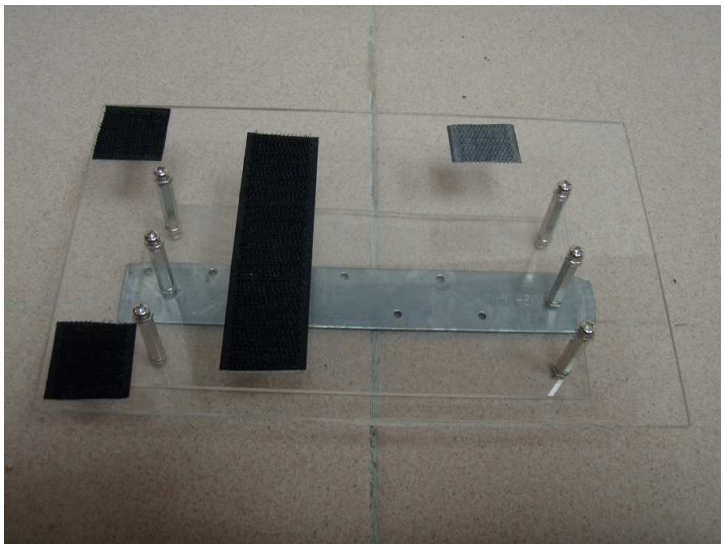
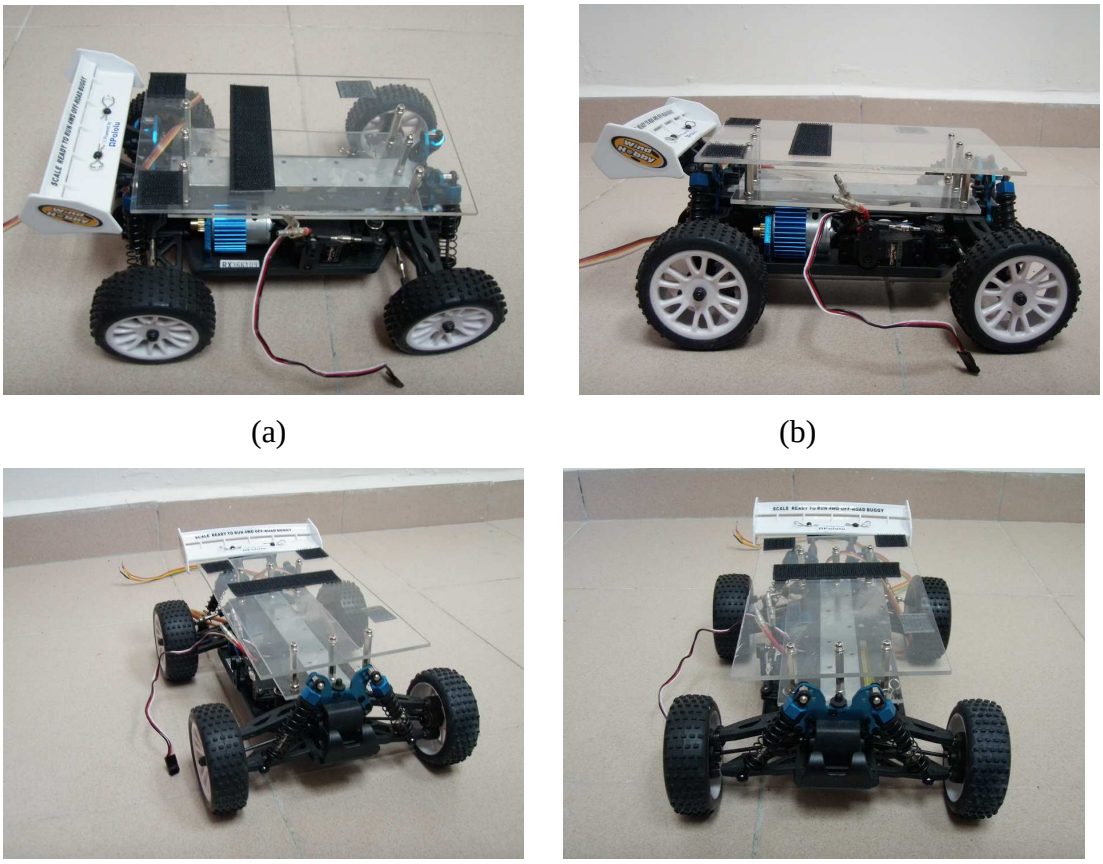


Figure 6.3 Sensor platform

The final attachment of the sensor platform to the mobile robot platform is shown in Figure 6.4.



(a) (b)

(c) (d)

Figure 6.4 Sensor platform attached to the mobile robot platform.

6.1 Robot controller

A controller is essential for an autonomous mobile robot in order to control the mobile robot. The robot controller is used to process the raw data from the sensors as well as to transmit the processed data to the PC. In this study, the Orangutan SVP robot controller is selected as the main robot controller as shown in Figure 6.5. This robot controller is simple and a complete solution for small and medium-sized robots.

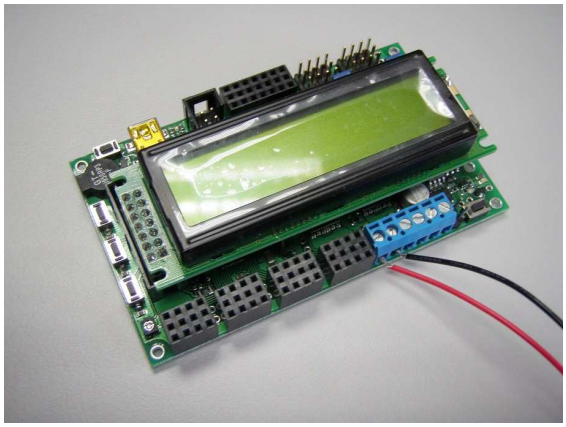
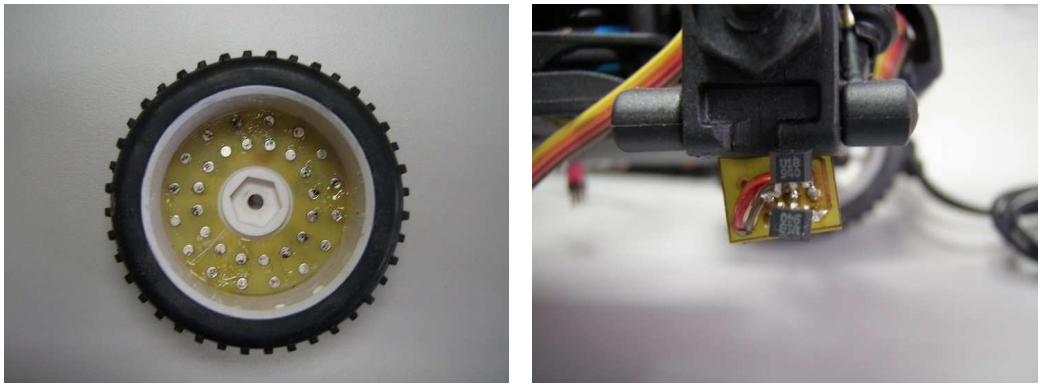


Figure 6.5 Robot controller

The features in this robot controller fulfil the requirements for a nonholonomic mobile robot developed in this study.. The module is design based on the powerful Atmel microcontroller. It has two motor drivers, a demultiplexer to control servo motors, I/O lines that can be used as analog and digital inputs and also the auxiliary processor that can read two quadrature encoders. In addition, the advantages of this microcontroller are easy to program the algorithms as it has extensive software libraries for the compiler and it is compatible with all development software for Atmel's AVR microcontroller. Details functions and specifications are given in Appendix B.

6.2 Wheel encoder

The purpose of the encoder is to provide feedback on the speed and travelling distance of the mobile robot. In this study, the magnetic encoder is used and attached to the wheel as shown in Figure 6.6(a). Magnetic encoder is chosen due to its simplicity and can provide a better accuracy for a small mobile robot. In the magnetic encoder, the Hall Effect sensors are used as transducers in which the output voltage is varied by the changes in magnetic field density. The Hall Effect sensors physical appearance is shown in Figure 6.7.



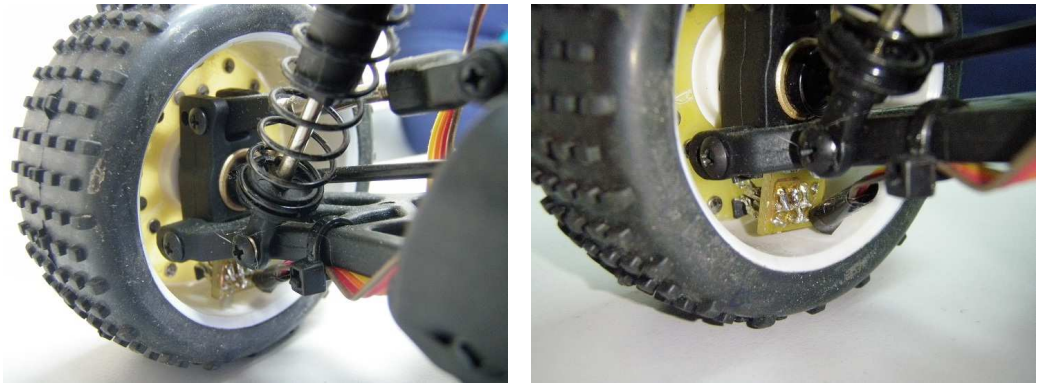
(a) (b)

Figure 6.6 (a) Magnets mounting attached at the wheel (b) Hall Effect sensors attached at the rear axle.

(a) (b)

Figure 6.7 Hall effect sensor

The magnetic encoder is designed to suit the dimension of the wheel, which the readily available encoder may not be suitable for the specific model of the mobile robot. Details of the Hall Effect sensor are given in Appendix B. The final attachment of the magnetic encoder to the mobile robot is shown in Figure 6.8.



(a) (b)

Figure 6.8 Location of the wheel encoder

The magnets mounting has 16 magnets that has been arranged to cover 360 degrees as shown in Figure 6.9. The calculation of distance is given by

$$\theta = \frac{pulse}{16} \times 360^{o}$$
$$\text{distance} = \frac{\theta}{360} \times 2\pi r \qquad (6.1)$$

where, Θ is resolution of wheel and r is the radius of wheel. The pulse is obtained once the Hall Effect sensor passing through the magnet. The total number of pulses are counted at every time step and then are used to calculate the distance.

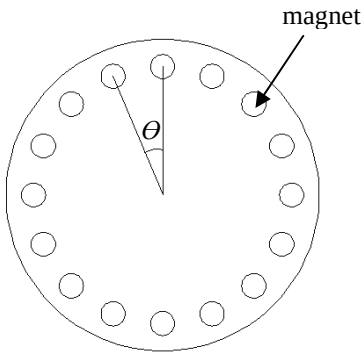
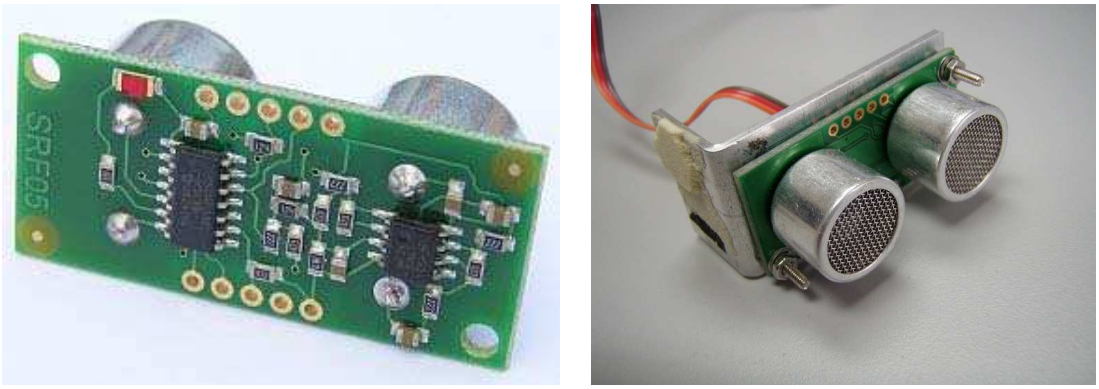


Figure 6.9 Magnet mounting of encoder

6.3 Detection sensors

Sensors for obstacle detection are essential in robot navigation. Infrared, ultrasonic, vision camera and laser range finder are among the sensors that have been used in obstacle detection. Laser range finders provide precise and stable range reading, however the drawback is the cost of the LRF is expensive. Using a vision camera is a better option to detect the obstacles; however it needs a significant computational power in order to process the data and may be a burden to a small robot. Infrared sensors only provide a single line detection which more suitable to use as a range sensor. Ultrasonic sensor is the best solution to use as a detection sensor for a small robot. It can provide the similar detection function as the LRF that adopted in the algorithm, which is to determine the region of detection relates to the position of the mobile robot. The region can be on center, left or right of the mobile robot. The ultrasonic sensor adopted in this study is the Devantech SRF05 as shown in Figure 6.10. This sensor can cover up to 45o angle and has a detection range from 3 cm to 4 m.



(a) (b)

Figure 6.10 (a) Ultrasonic range sensors (b) Sensor attached to the sensor base.

6.4 Communication

In this study, communication between the mobile robot and the PC is required as all the data need to send to the PC to process. The decision is made based on the data in the PC and then the results will be sent back to the mobile robot in order to navigate in the environment. Thus, a wireless communication is preferable as the mobile robot will move away from the PC.

The wireless communication module adopted in this study is Xbee/Zigbee RF module as shown in Figure 6.11. The transmitting range for this module can be ranged up to 30 m for indoor and 100 m for outdoor environments. The wireless communication consists of a router and a coordinator. The router is attached to the mobile robot, while the coordinator is plug into the PC. Detail specifications are given in Appendix B.

(a) (b)

Figure 6.11 Wireless communication (a) Router (b) Coordinator

6.5 Calibration of steering angle and velocity

Prior to the experimental work for the mobile robot in the real environment, the calibration works are required to establish the actual steering angle and the speed of the mobile robot. The steering angle and the speed are controlled by the PWM (Pulse-Width Modulation) values. Thus, the relation between the PWM values and steering angle as well as the relation between the PWM values and velocity of the mobile robot need to be established. The calibration work was conducted in the open-space and flat area.

6.5.1 Steering angle

The calibration work for the steering angle was conducted by setting the steering angle to the constant PWM. Constant steering angle results in a drive along a circle. The radius of the circle was then been measured and the relationship between the steering angle and PWM can be established. The data for calibration work of the steering angle are tabulated in Table 6.1.

Table 6.1 Steering angles under different PWM values

PWM (hex)	Diameter 1 (cm)	Diameter 2 (cm)	Ave diameter (cm)	Steering angle (^{o})
60	73	79	76.0	24.6
69	94	99	96.5	19.8
70	139	134	136.5	14.3
75	166	173	169.5	11.6
85	inf	inf	0.0	0.0
95	186	194	190.0	10.4
99	145	153	149.0	13.1
A0	110	113	111.5	17.3
A5	90	92	91.0	20.9

The reference point for the radius of the circle is shown as in Figure 6.12. With radius of circle and distance between front and rear wheels, the steering angle can be calculated from the following equation

$$\phi = L/R \qquad (6.2)$$

where is L = length between front and rear wheels

It was assumed that the steering angle is a virtual middle wheel between two front wheels and distance from centre point to ICC is similar to radius of the circle.

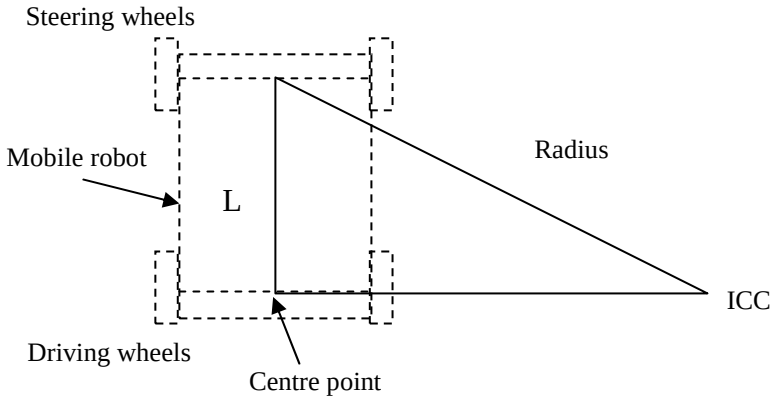


Figure 6.12 Calibration work for establishment of steering angle

From the data in Table 6.1, the relation between the PWM and the steering angle is shown in Figure 6.13. The following linear relations were obtained by the Least Square Method:

$$\phi = 0.6502 \cdot PWM + 86.477 \text{, for turning left,} \quad (6.3)$$

$$\phi = -0.6717 \cdot PWM - 89.698 \text{, for turning right.} \quad (6.4)$$

For zero steering angle, the PWM value of 132 is used.

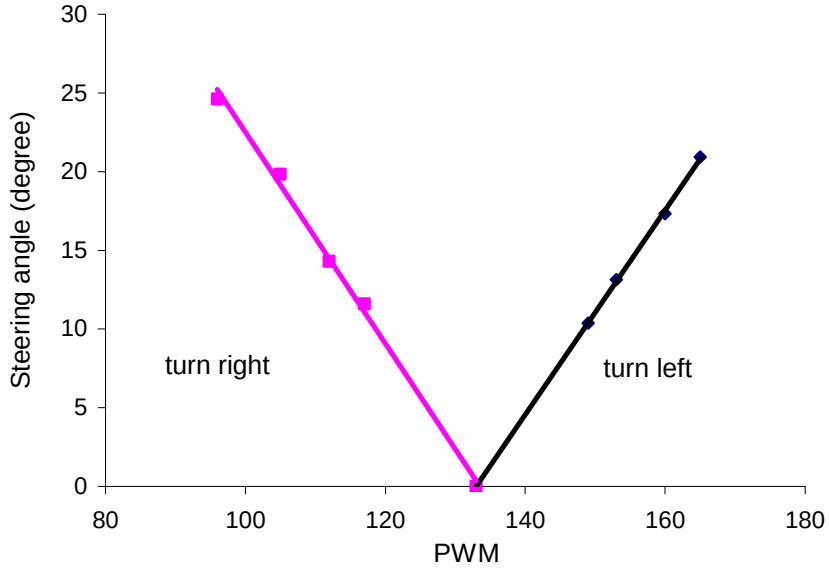


Figure 6.13 Relation between PWM values and steering angle.

6.5.2 Velocity

The speed control is crucial to the time-critical motion planning as it can reflect the motion of the mobile robot. The speed is also controlled by the PWM value. The calibration work was conducted by taking time for 100 m distance from start to final point for each PWM value as shown in Figure 6.14.

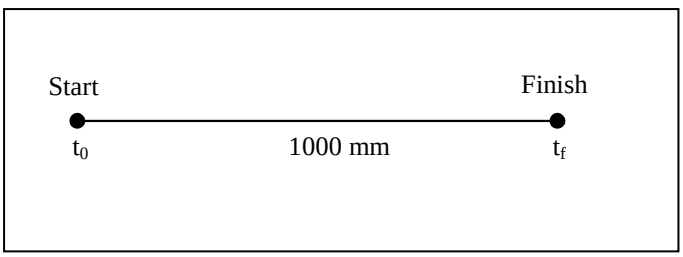


Figure 6.14 Calibration work for establishment of velocity

The data for calibration work of velocity are tabulated in Table 6.2.

Table 6.2 Velocities under different PWM values

PWM (hex)	Time 1 (s)	Time 2 (s)	Time 3 (s)	Ave time (s)	Vel (cm/s)
10	0	0	0	0.0	0
20	2.1	2.5	2.3	2.3	43.5
30	1.3	1.2	1.1	1.2	83.3
40	0.9	0.9	0.9	0.9	111.1
50	0.7	0.6	0.7	0.7	150.0
60	0.6	0.6	0.6	0.6	166.7
70	0.5	0.5	0.5	0.5	200.0

From the data in Table 6.2, the relation between the PWM and the speed is shown in Figure 6.15. With the Least Square Method, the following equation was established:

$$v = 2.038 \cdot PWM - 22.636 . \qquad (6.5)$$

Note that, the measured speeds were obtained during the battery is fully charged.

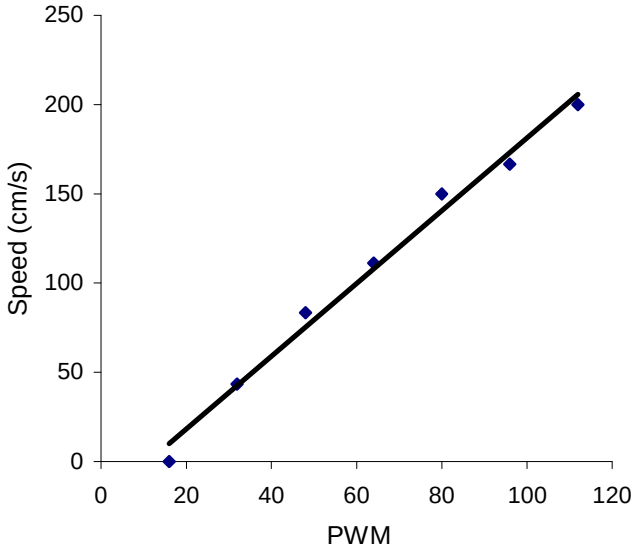


Figure 6.15 Relation between PWM values and speed.

6.6 Obstacle detection

In the experimental works, an ultrasonic sensor was used to detect obstacles during mobile robot navigation. Ultrasonic sensors have been proven to be effective to detect an obstacle in the actual environment. The ultrasonic sensor was placed at the front of the mobile robot. For the purpose of the experimental works in this study, the detection range is set to be 100 cm and the detection angle for an ultrasonic sensor is around 45º as shown in Figure 6.16. The full specifications of the ultrasonic sensor used in this study can be referred in Appendix A.

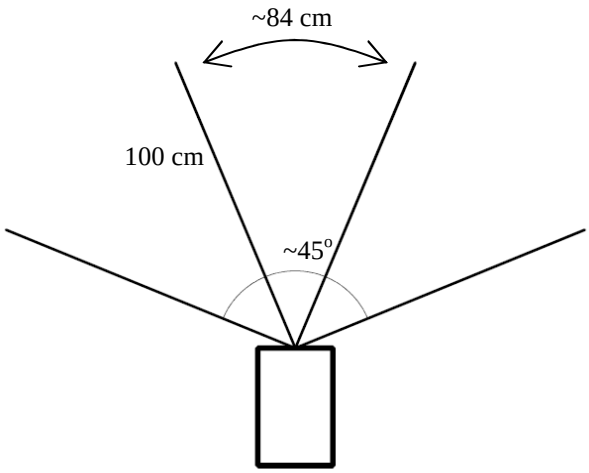


Figure 6.16 Obstacle detection range for experimental works.

6.7 Wireless communication

In order to achieve remote control over the mobile robot, the Xbee RF module is used. The wireless connection configuration is shown in Figure 6.17.

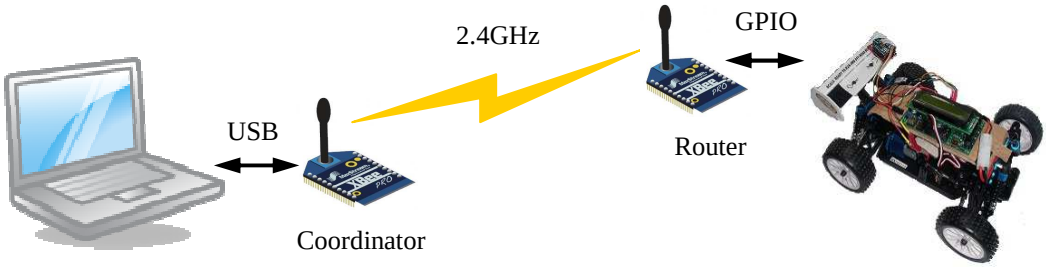


Figure 6.17 Wireless communication between the operator and the router (robot).

For this task, two Xbee RF modules are used as a Coordinator and a Router. Basically, the Coordinator is connected to the PC via USB and the Router is attached to the mobile robot via the General Purpose Input/Output (GPIO) port. Once the initial collision-free trajectory was generated, the data – output data – will be sent to the mobile robot by the Coordinator. Then the mobile robot will receive the data – as control input data – by the Router. These control inputs which are steering angle and velocity will be used to move the mobile robot. The communication can be a two-way communication with the Router will send the data and the Coordinator will receive the data. Such communication is necessary when the sensor's reading needs to be processed in the PC.

6.8 Concluding remarks

The development of a mobile robot is required to investigate the algorithms is presented in this chapter. In order to implement the algorithms appropriately, the mobile robot should has a capability similar to a car and be able to detect the obstacles in front of it. A range of options were identified in the development stage and the appropriate solutions were chosen such as the selection of the of the mobile robot's base, the arrangement of the sensors and accessories and the sensors' selection. The

base of the mobile robot was adopted from a RC car and a two-tier sensor platform was assembled and placed on top of the mobile robot's base. The sensors data will be extracted from the sensors using the microcontroller. Furthermore, the mobile robot was also fitted with a wireless communication, which allowed the data transmission between the mobile robot and the PC in real time during testing.

7. EXPERIMENTAL RESULTS AND DISCUSSIONS

In this chapter, a series of experimental works have been conducted. The aims of these experimental works are to validate the algorithms that have been used to control the mobile robot and also to verify the effectiveness of the developed simulation framework. The mobile robot model was tested in the various conditions of the environments. The algorithms tested were derived from the algorithms that were developed in the Chapter 4. In these experimental works, the sensor values were transmitted from the mobile robot to the PC in real time during the testing. The results were then collected in the PC and plotted in graphs.

For each case, the experimental results can be compared to the simulation results for the given scenarios. The experimental works were divided into four parts:

1. Mobile robot navigates in an obstacle-free environment.
2. Mobile robot navigates in a known static environment.
3. Mobile robot navigates in an unknown static environment.
4. Mobile robot navigates in an unknown dynamic environment.

7.1 Experiment architecture

The overall view of the system architecture for the mobile robots' navigation is shown in Figure 7.1. In this study, the experiment architecture was designed to cater the known and unknown static obstacles and it can be expanded to the moving obstacles in the future works.

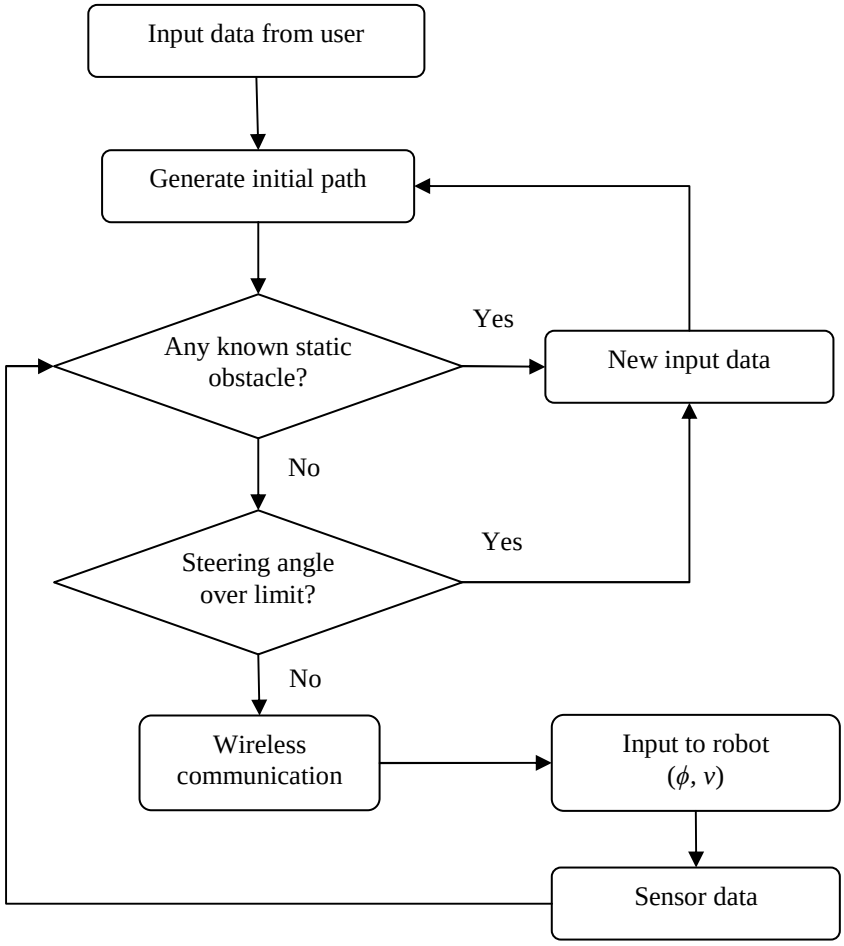


Figure 7.1 Experimental work flow

The obstacle-free trajectory will first be obtained from the offline planning after having inputs from the user. The output data, which are steering angle and velocity, were then transmitted to the mobile robot via the wireless communication and these data were used to move the mobile robot for every time step. In the same time, the data extracted from the sensors; such as data from detection sensor; will be sent to the PC to be processed. The decision making process will take place at this stage and once the decision has been made, the mobile robot will then react based on the

decision. For example, if the algorithm decided there is an obstacle in front of the mobile robot, the obstacle avoidance algorithm will be executed and the new input data will be transmitted to the mobile robot and the mobile robot will react based on these new inputs.

7.2 Experiment setup

The testing arena is an open-space and flat terrain area as shown in Figure 7.2.

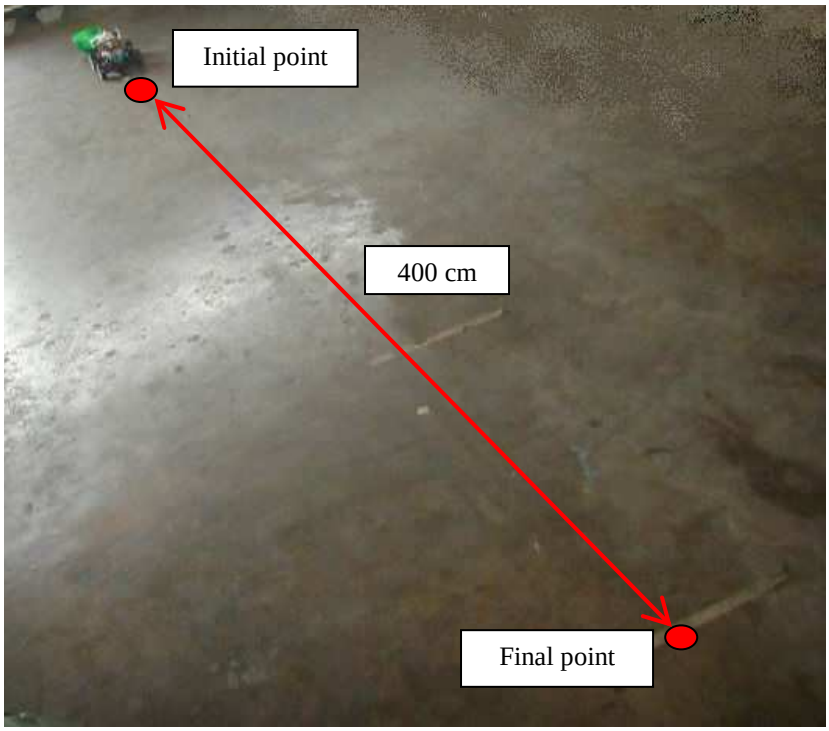


Figure 7.2 Testing arena

The first two cases have been conducted with the aim to initially validate the algorithm. A car-like robot discussed in previous section has been used. For both cases, the distance between the initial and final point was 400 cm and the travelling time was set as 20 seconds.

In the first case, the algorithm was tested without an obstacle and in the second case, the algorithm was tested with the presence of a known static obstacle. The experimental work was then further extended to the unknown static obstacle in the third case. The algorithm was executed in a PC using MATLAB and the output was sent to the mobile robot as the control input via wireless communication.

Furthermore, a marker was located at the back of the mobile robot in order to map out the actual trajectory. When the mobile robot moves from the starting point, the marker leaves a trace of the trajectory on the floor. Then the trace of the trajectory was measured manually in order to obtain the actual trajectory for each experiment. In addition, the movement of the mobile robot was also recorded using a video camera for each experiment in order to trace the actual trajectory.

7.3 Case 1: Navigation in an obstacle-free environment

The purpose of the first experiment is to verify the control strategy of a car-like robot. The steering angle and velocity are the two parameters that need to verify. In this experiment, the initial state was set as [0, 200, 0, 0, 0, 0] and the final state was [400, 200, 0, 0, 0, 20]. The mobile robot was moved in a straight line for a distance of 400 cm in the environment without an obstacle as shown in Figure 7.3.

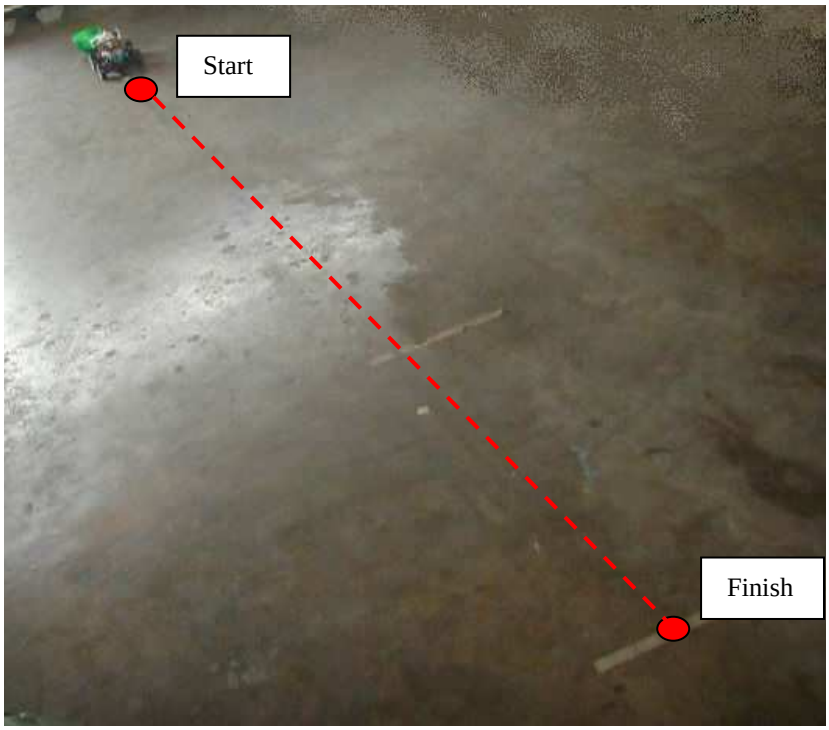
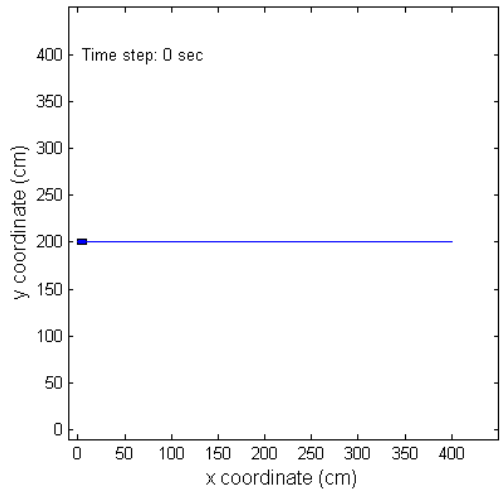
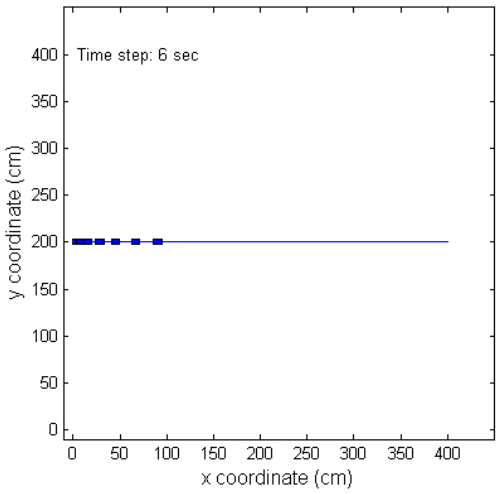


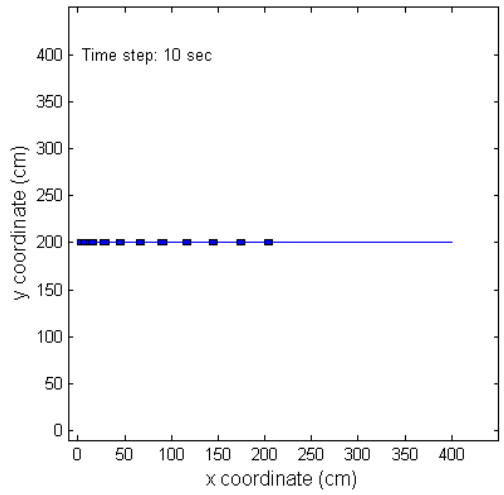
Figure 7.3 Experimental setup for Case 1



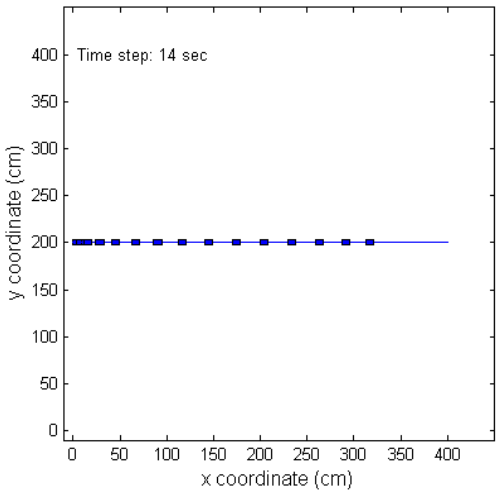
(a) At time = 0 s



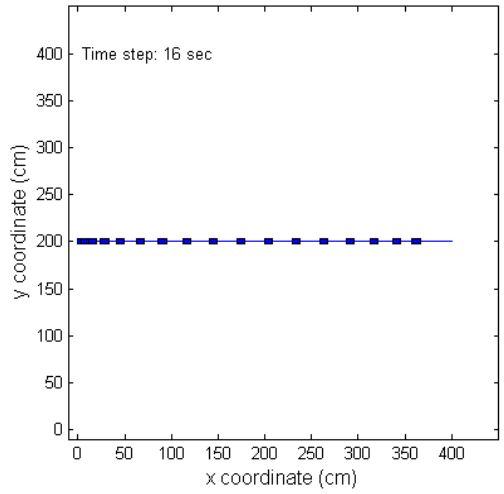
(b) At time = 6 s



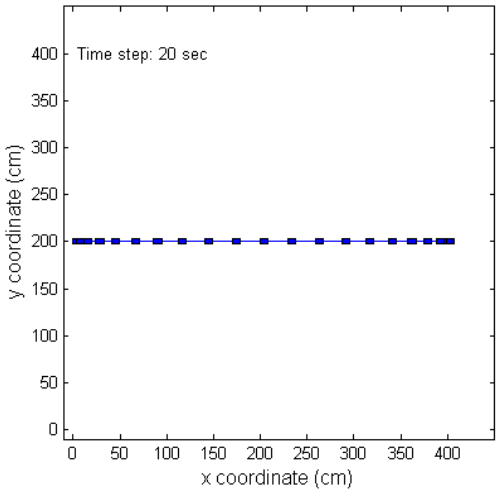
(c) At time = 10 s



(d) At time = 14 s



(e) At time = 16 s



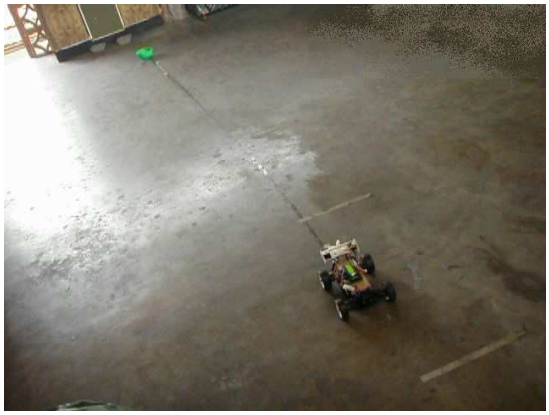
(f) At time = 20 s

Figure 7.4 Mobile robot navigated in an obstacle-free environment (simulation)

(a) At time = 0 s

(b) At time = 6 s

(c) At time = 10 s



(d) At time = 14 s

(e) At time = 16 s

(f) At time = 20 s

Figure 7.5 Mobile robot navigated in an obstacle-free environment (experiment)

The results for the simulation works and experimental works are shown in Figure 7.4 and Figure 7.5, respectively. The mobile robot was placed at the initial point as shown in Figure 7.5(a). It was then started to move from the initial point as shown in Figure 7.5(b). At the 10th second, the mobile robot was at the half of its trajectory. The mobile robot was approaching the final point as shown in Figure 7.5(e) and reached the final point at 20th second as shown in Figure 7.5(f).

The simulation and experimental results can be compared at the respective point through the respective figures as shown in Figure 7.4 and Figure 7.5. From the results, the experimental works demonstrate the mobile robot was able to match the simulation results in term of location of the mobile robot at the specific time. In addition, the experiment was conducted in three trial runs and the actual trajectory is compared to the planned trajectory as shown in Figure 7.6.

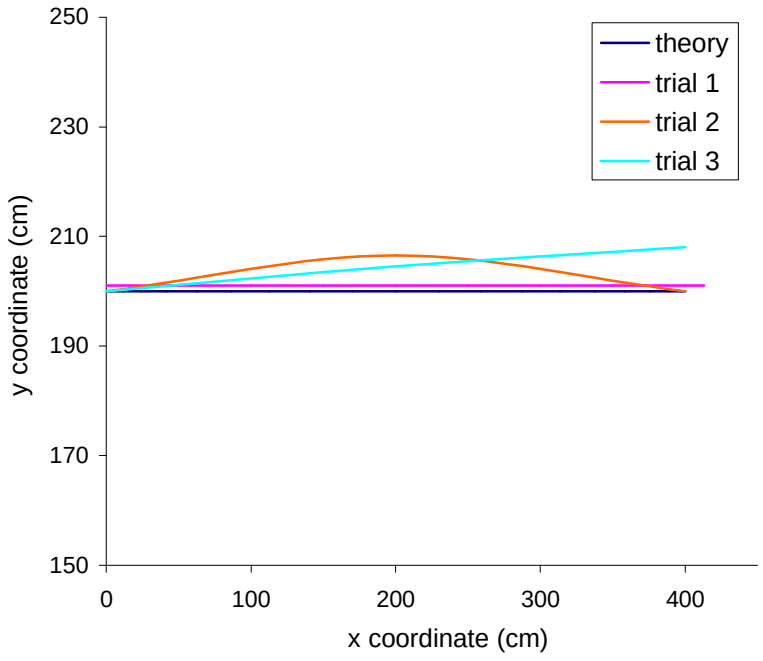


Figure 7.6 Case 1: Trajectory planning without an obstacle

From the results in Figure 7.6, the first trial is almost identical to the planned trajectory, but the mobile robot stopped at 13 cm more than it should be. In the second trial, the mobile robot basically reached the final point, but the mobile robot was not moved in a straight line as we can see from the results. The final trial, the mobile robot moved in a straight line, but its trajectory is tilted at about 1.1^{o} from the planned x-axis. The position errors for each trial are listed in Table 7.1. The time taken for the mobile robot from initial point to the final point was 20 seconds and matched the time that initially planned. The maximum percentage error at the final point is 3.3%. As a conclusion, the result for control strategy is satisfactory as the mobile robot was able to follow the desired trajectory closely.

Table 7.1 Actual initial and final positions for Case 1

	xs (cm)	**ys (cm)**	**xf (cm)**	**yf (cm)**	**x error (cm)**	**y error (cm)**
Theory	0	200	400	200	-	-
Trial 1	0	200	413	200	13	0
Trial 2	0	200	400	200	0	0
Trial 3	0	200	400	208	0	8

7.4 Case 2: Navigation in a known static environment

The purpose of this experiment is to validate the newly developed algorithms. In this experiment, a known static obstacle will be placed in the environment. The planner will generate an initial collision-free trajectory; which the obstacle is taken into account during generation of the trajectory and the mobile robot is expected to follow the trajectory, avoid the obstacle and reach the final point at the desired time.

In this experiment, the environment was set as in Figure 7.7. The distance from the starting point to the finishing point is 400 cm. A known static obstacle was placed in the middle of the x-axis with the obstacle's diameter is 20cm. The initial and final states of the mobile robot were [0, 200, 0, 0, 0, 0] and [400, 200, 0, 0, 0, 20], respectively.

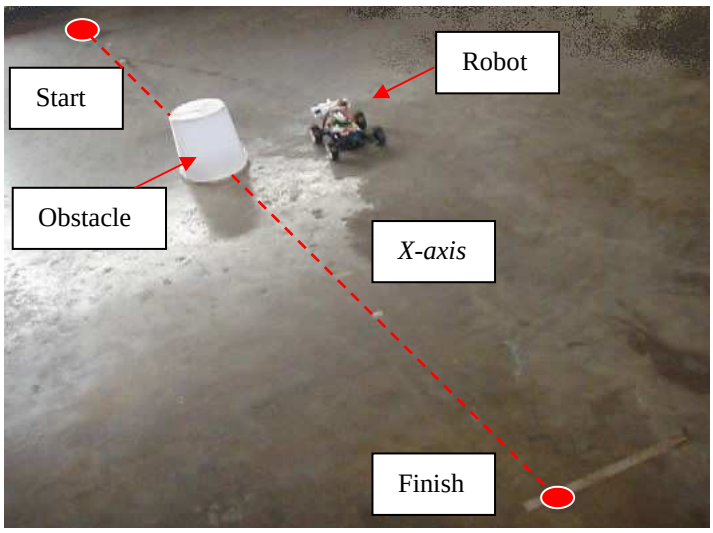
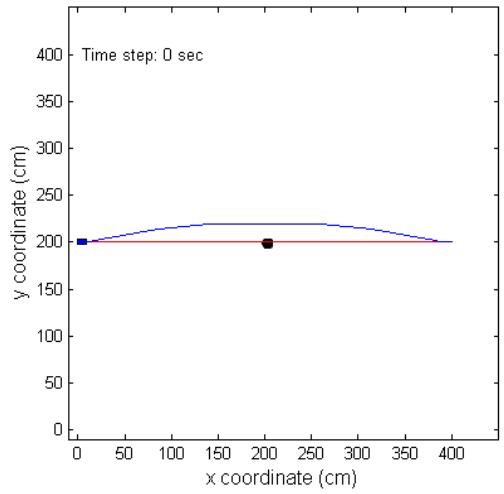
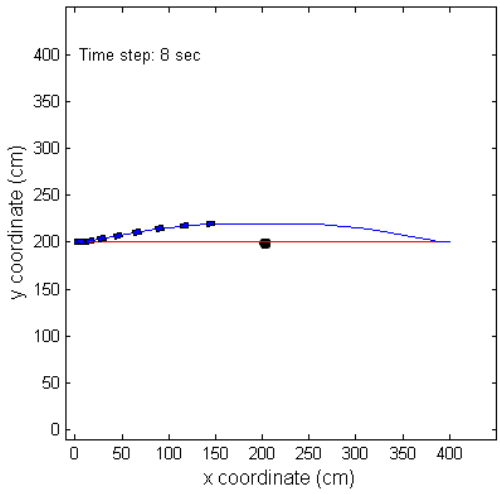


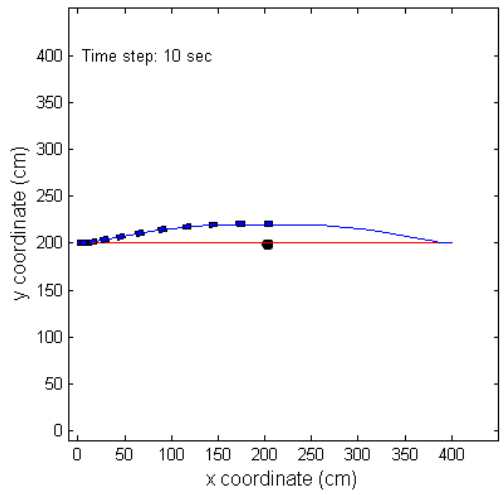
Figure 7.7 Experimental setup for Case 2.



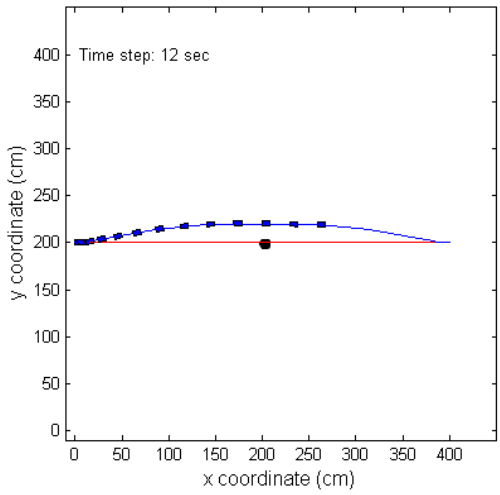
(a) At time = 0 s



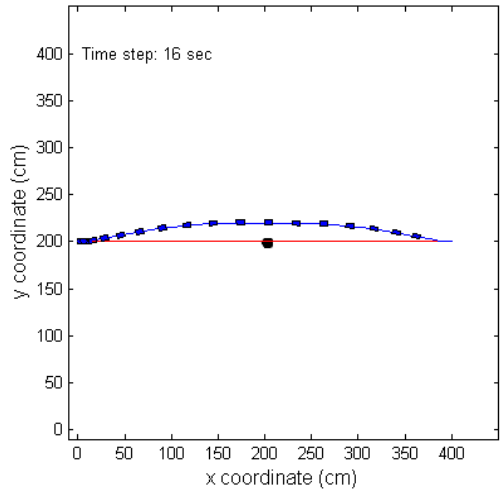
(b) At time = 8 s



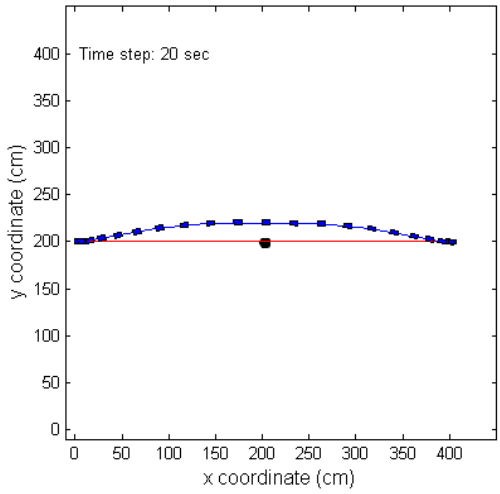
(c) At time = 10 s



(d) At time = 12 s

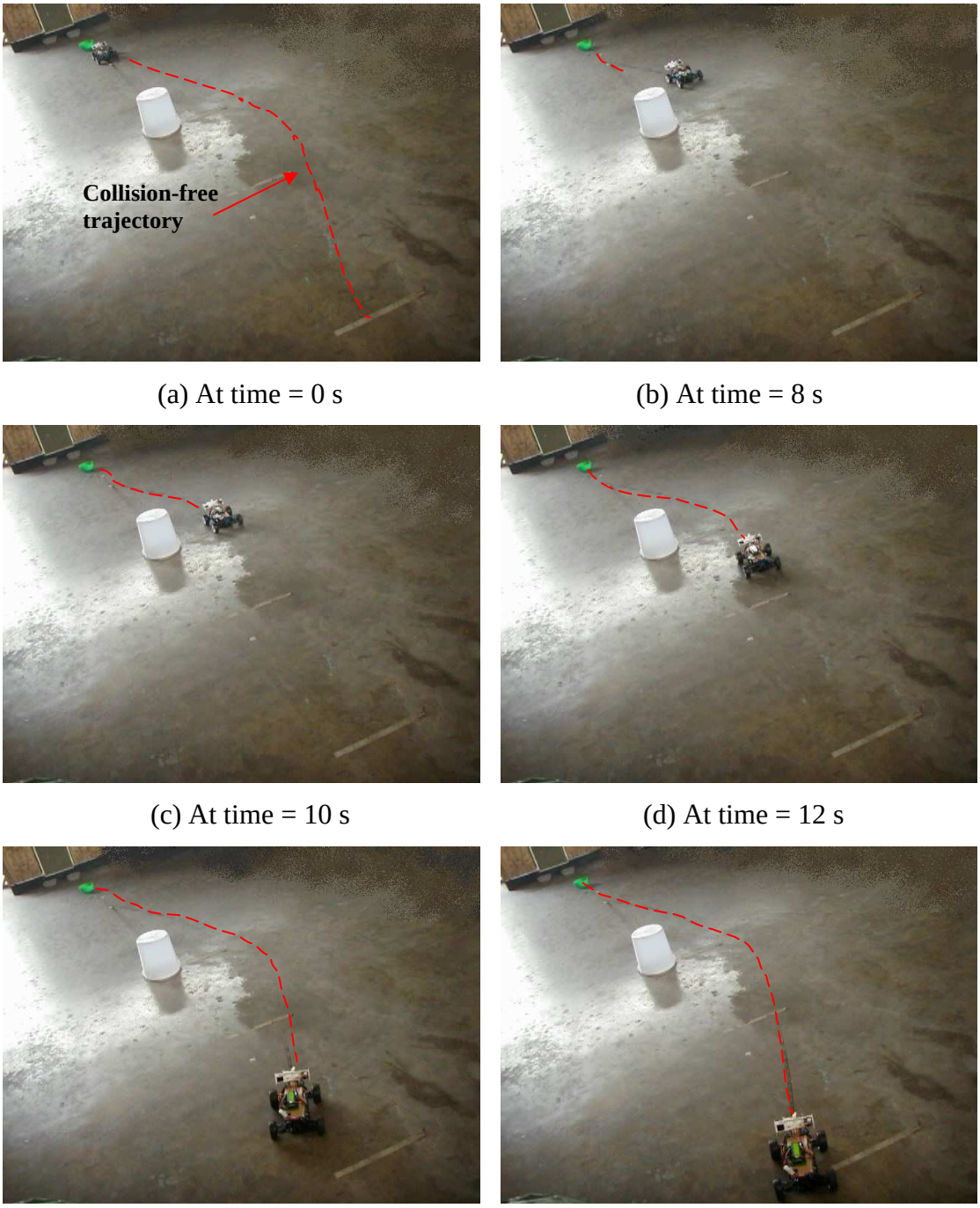


(e) At time = 16 s



(f) At time = 20 s

Figure 7.8 Mobile robot navigated in a known static environment (simulation).



(a) At time = 0 s (b) At time = 8 s

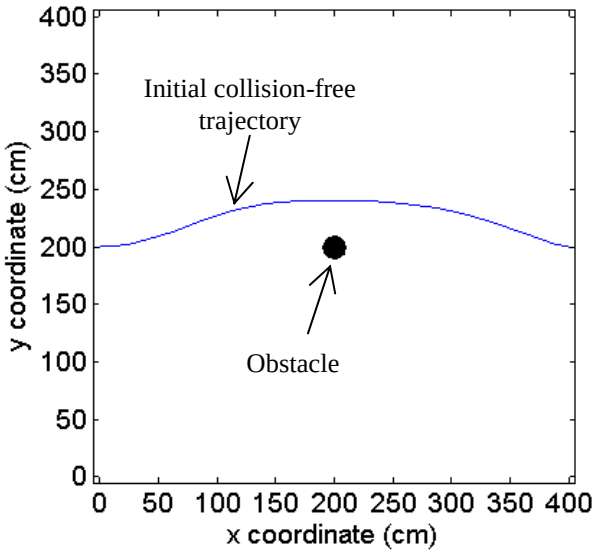
(c) At time = 10 s (d) At time = 12 s

(e) At time = 16 s (f) At time = 20 s

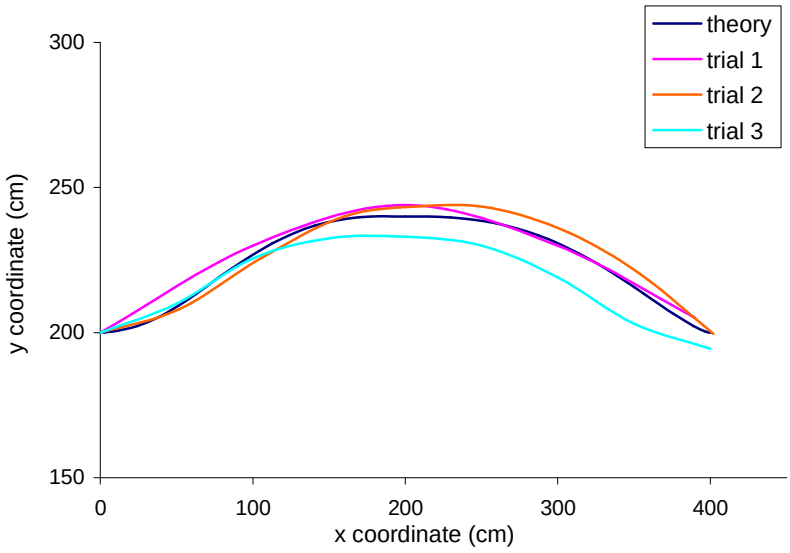
Figure 7.9 Mobile robot navigated in a known static environment (experiment).

The simulation and experimental results for Case 2 are shown in Figure 7.8 and Figure 7.9, respectively. The mobile robot and the static obstacle were placed at the initial point and the middle of the trajectory, respectively, as shown in Figure 7.9 (a). The mobile robot was then started to move from the initial point as shown in Figure 7.9(b). At the 10^{th} second, the mobile robot was at the half of its way and successfully followed the initially planned trajectory with the consideration of the static obstacle as

shown in Figure 7.9(c). The mobile robot was approaching the final point and reached the final point at 20th second as shown in Figure 7.9(e) and Figure 7.9(f), repsectively.



(a)



(b)

Figure 7.10 (a) Case 2: Trajectory planning with a known static obstacle, (b) Experimental results.

The generated initial collision-free trajectory is shown in Figure 7.10(a) with the obstacle is represented by black circle. From the results in Figure 7.8 and Figure 7.9, the location of the mobile robot at the specific time in the experiment was matched with the simulation works. In addition, this second experiment was also conducted in three trial runs and the results are compared to the planned trajectory as shown in Figure 7.10(b).

From the results, the actual trajectory for the first and second trials is almost identical to the planned trajectory. And in the final trial, the mobile robot was able to avoid the obstacle, but the mobile robot was not reached the final point accurately. The position errors for Case 2 are tabulated in Table 7.2. The maximum percentage error at the final point is 2.5%. As a conclusion, the result for control strategy is satisfactory as the mobile robot was able to avoid the obstacle and follow the desired trajectory closely. In addition the mobile robot was able to reach closed to the final point at 20 seconds.

Table 7.2 Actual initial and final positions for Case 2

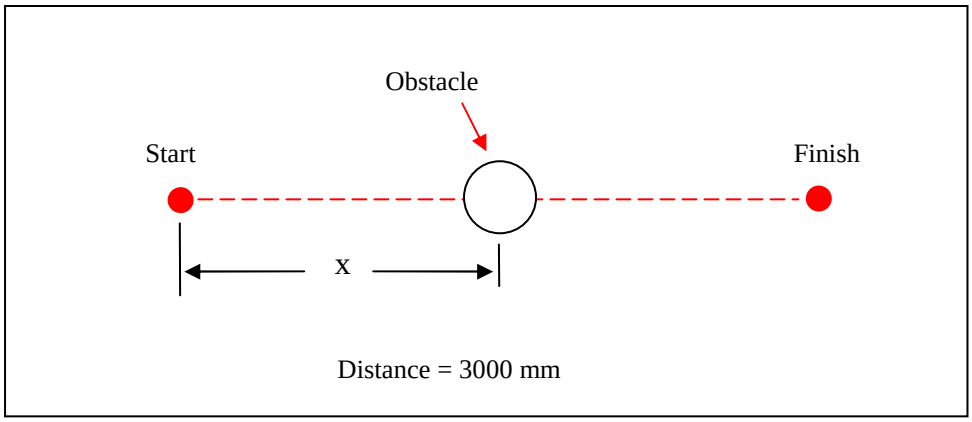
	xs (cm)	**ys (cm)**	**xf (cm)**	**yf (cm)**	**x error (cm)**	**y error (cm)**
Theory	0	200	400	200	-	-
Trial 1	0	200	390	205	-10	5
Trial 2	0	200	402	194.5	2	-5.5
Trial 3	0	200	400	194.5	0	-5.5

7.5 Case 3: Navigation in an unknown static environment

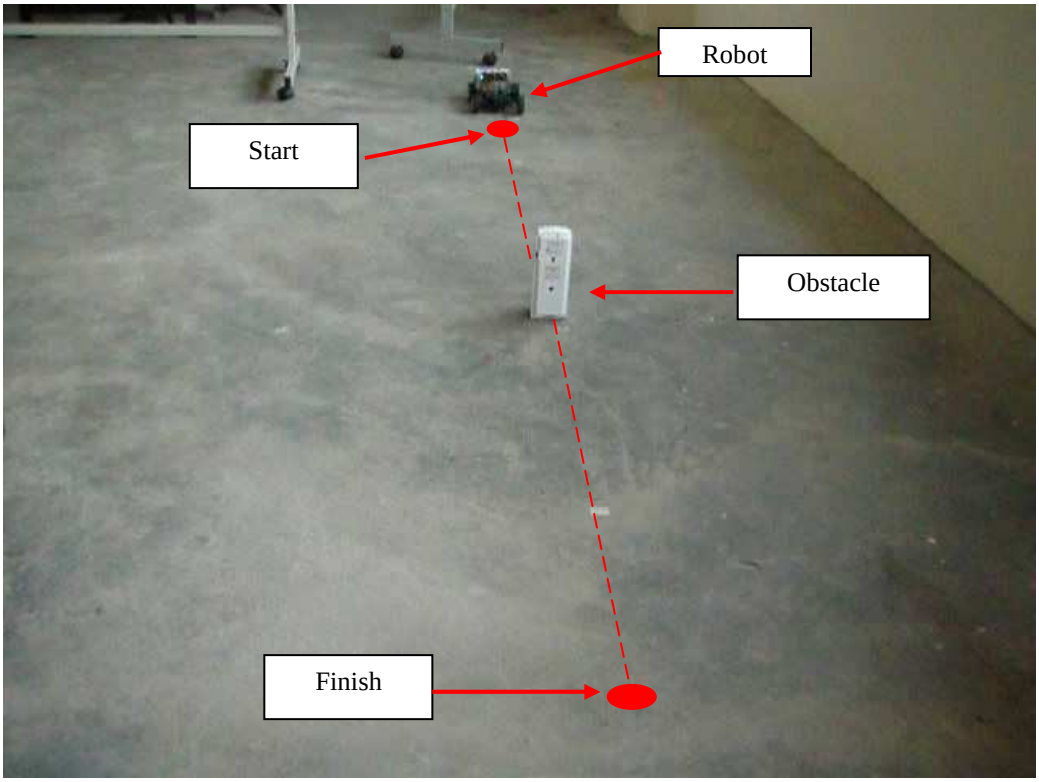
The experiment was further performed on the mobile robot in an unknown static environment. The purpose of this experiment is to validate the obstacle avoidance approach for the time-critical motion planning. In this experiment, unknown static obstacles will be placed in the environment. The planner will generate the collision-free trajectory without the knowledge of unknown static obstacle and it is expected to detect and avoid the obstacle. Furthermore, the mobile robot is also expected to reach the final point at the desired time. The algorithm is tested through a series of scenarios.

7.5.1 Scenario 1: One unknown static obstacle

In Scenario 1, the mobile robot was required to move from the initial point to the final point as shown in Figure 7.11. The distance from the initial point to the final point is 3000 mm. An unknown static obstacle was placed randomly between the initial point and final point with the obstacle's diameter is 8 cm. The initial and final states of the mobile robot were [0, 50, 0, 0, 0, 0] and [300, 50, 0, 0, 0, 20], respectively.



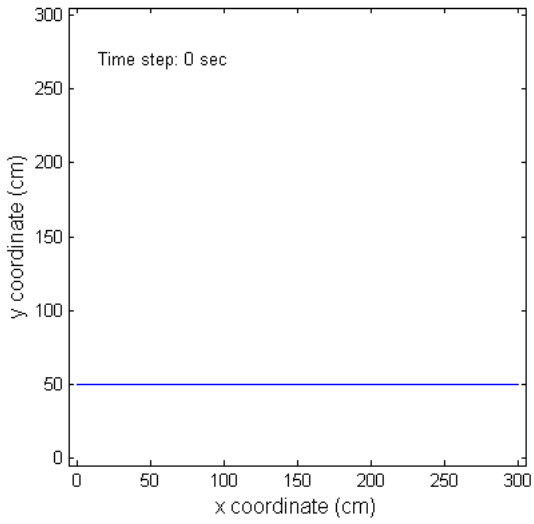
(a)



(b)

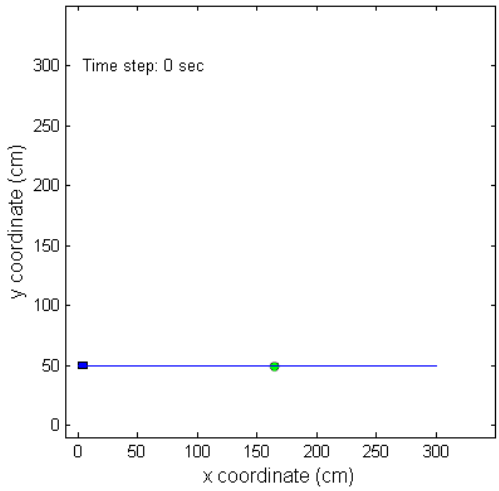
Figure 7.11 (a) Plan view (b) Actual experimental setup for Scenario 1

Using the initial and final states of the mobile robot, the planner an initial collision-free trajectory which is represented by a blue line is shown in Figure 7.12. Note that there is no obstacle in the map as the obstacle is unknown to the planner.

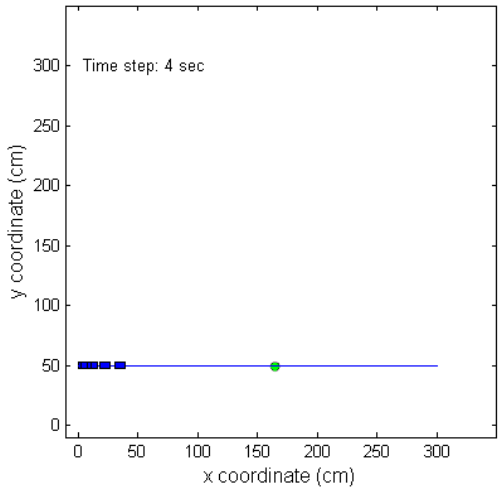


(a)

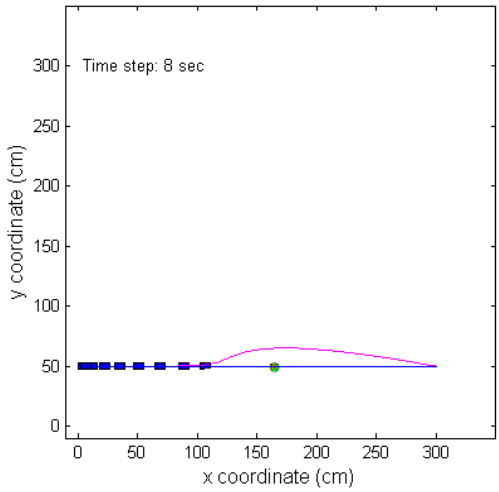
Figure 7.12 Initial collision-free trajectory for Case 3



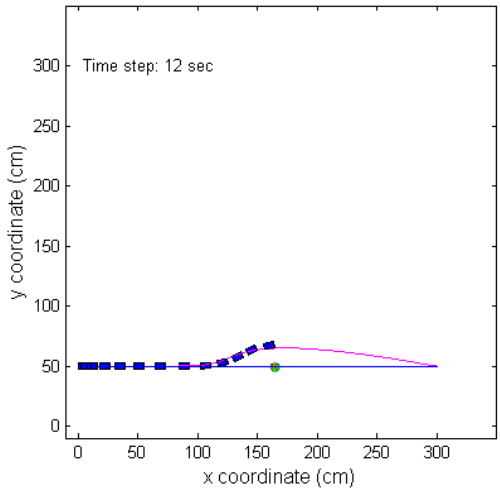
(a) At time = 0 s



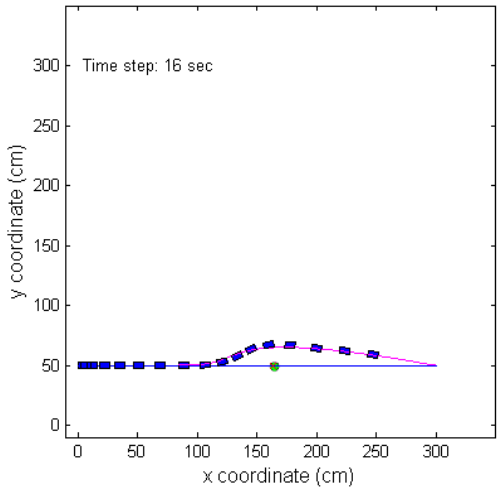
(b) At time = 4 s



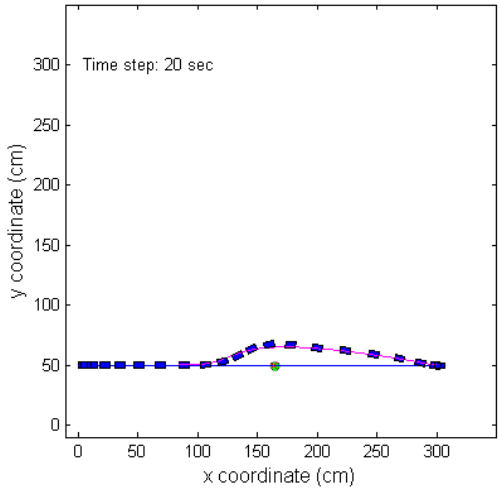
(c) At time = 8 s



(d) At time = 12 s

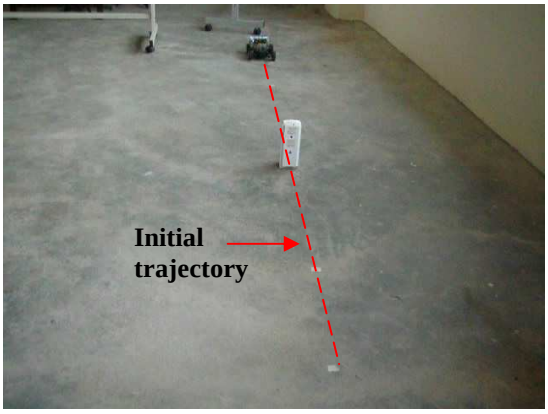


(e) At time = 16 s



(f) At time = 20 s

Figure 7.13 Mobile robot navigates in the unknown static environment (simulation)



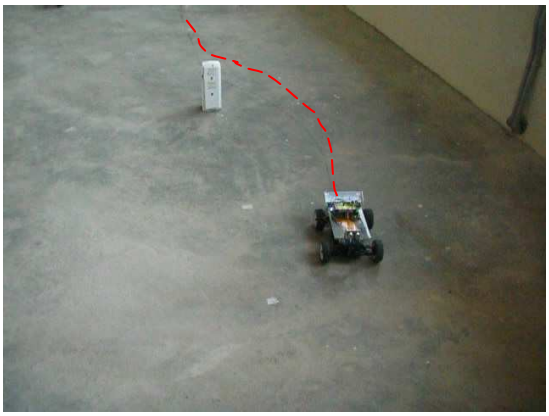
(a) At time = 0 s

(b) At time = 4 s

(c) At time = 8 s

(d) At time = 12 s

(e) At time = 16 s



(f) At time = 20 s

Figure 7.14 Mobile robot navigates in the unknown static environment (experiment)

The simulation and experimental results for Scenario 1 are shown in Figure 7.13 and Figure 7.14, respectively. Both simulation and experimental results can easily compared through the respectively figures. At the start of the experimental work, the mobile robot was followed the initial collision-free trajectory until it detected the obstacle in front of it as shown in Figure 7.14(c). Then the dynamic obstacle avoidance approach was executed and a new trajectory was generated from the point of detection to the final point. The mobile robot was then followed the new trajectory successfully until it reached the final point as shown in Figure 7.14(f).

The experiment results were then being compared with the theory as shown in Figure 7.15. During the experiment, the mobile robot detected the obstacle's location at (169, 50). The actual measured location of the unknown static obstacle was (163, 50). This was showed that the ultrasonic sensor was able to detect and locate the obstacle close to the actual location. In comparison with the theory, the mobile robot was able to follow the initial planned trajectory until it detected the obstacle and the new trajectory was generated in order to avoid the obstacle. From the point of detection, the mobile robot's movement was slightly deviated from its planned trajectory and stopped before the final point. The experiment has proven that the algorithms was practical to be used in trajectory planning as control strategy for the mobile robot was able to translate the input to the actual trajectory and the mobile robot was able to follow the planned trajectory as close as possible.

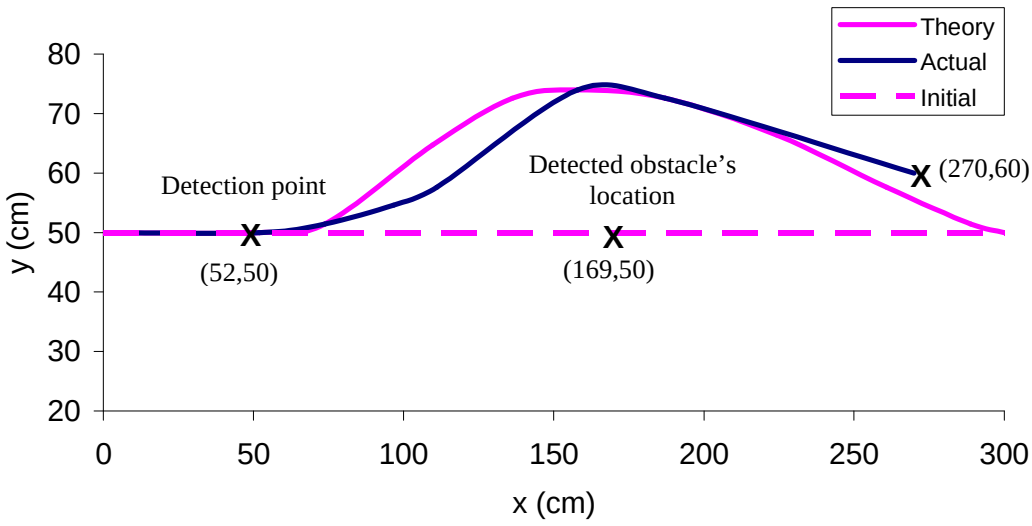


Figure 7.15 Theoretical and actual trajectory for Case 3

The experiment has been conducted in three trial runs and the errors in positions at the final point were compared with the theory as tabulated in Table 7.3. The maximum relative errors recorded in *x*-axis and *y*-axis at the final point for 20 seconds are around 18.3% and 20%, respectively.

Table 7.3 Actual initial and final positions for Case 3

	xs (cm)	**ys (cm)**	**xf (cm)**	**yf (cm)**	**x error (cm)**	**y error (cm)**
Theory	0	50	300	50	-	-
Trial 1	0	50	270	60	-30	10
Trial 2	0	50	355	43	55	-7
Trial 3	0	50	350	40	50	-10

7.5.2 Scenario 2: Two unknown static obstacles

The experimental works for the unknown static environments were further investigated by adding more unknown static obstacles. In Scenario 2, two unknown obstacles were placed in the environment at the unknown location and the distance was increased to 4000 mm. The travelling time from initial point to final point was set to 30 seconds. The experimental setup was shown in Figure 7.16.

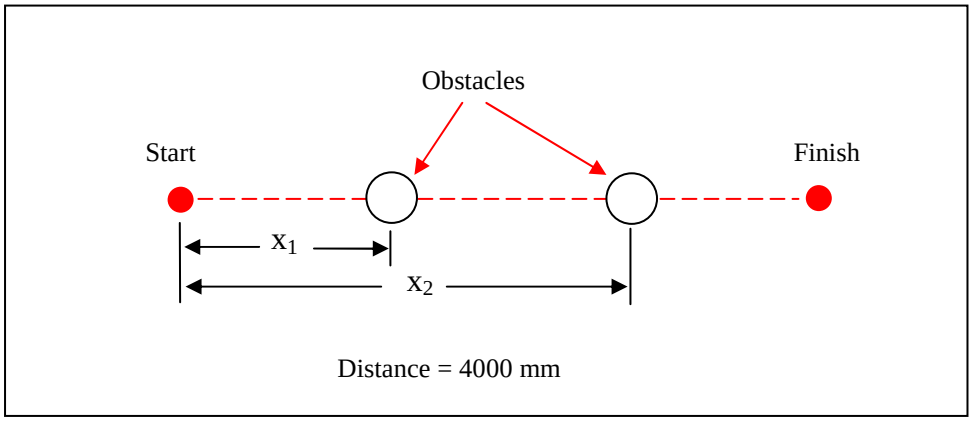
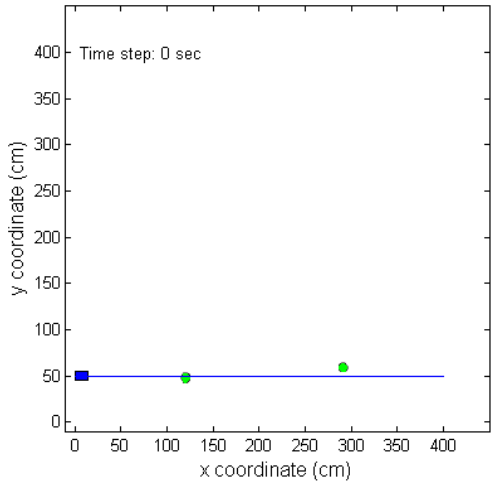
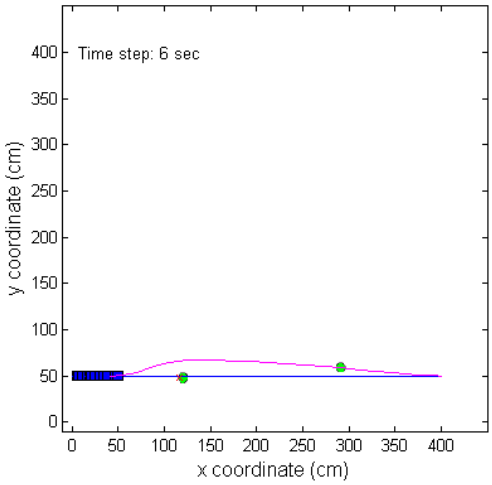


Figure 7.16 Experimental setup for Scenario 2

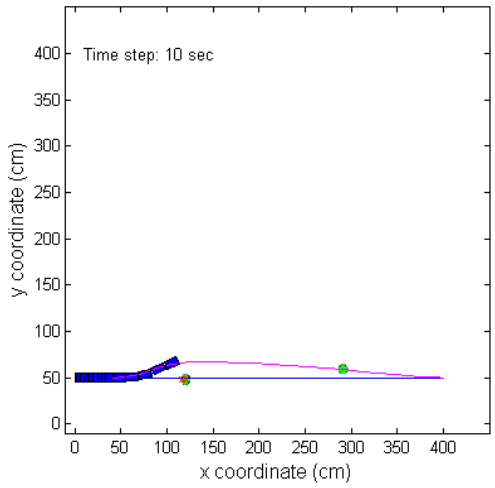
The simulation and experimental results are shown in Figure 7.17 and Figure 7.18, respectively. Both simulation and experimental results can be compared through the respectively figures. At the beginning of the experiment, the mobile robot was followed the initial collision-free trajectory until it detected the first obstacle. Then the obstacle avoidance algorithm has been executed and the mobile robot avoided the first obstacle as shown in Figure 7.18(b). Once the mobile robot has avoided the first obstacle, it then continued it journey until it detected and avoided the second obstacle as shown in Figure 7.18(d). After successfully avoiding the second obstacle, the mobile robot continued following the new trajectory until it reached the final point as shown in Figure 7.18(f).



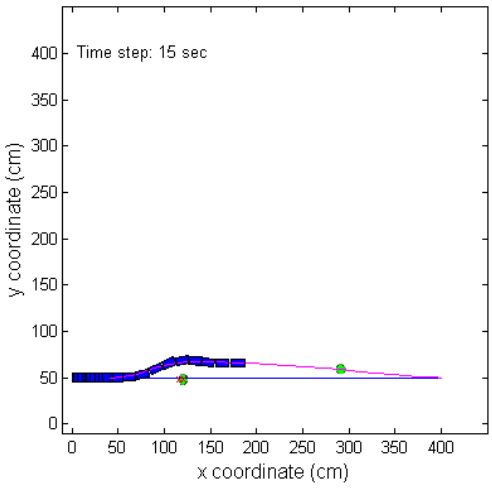
(a) At time = 0 s



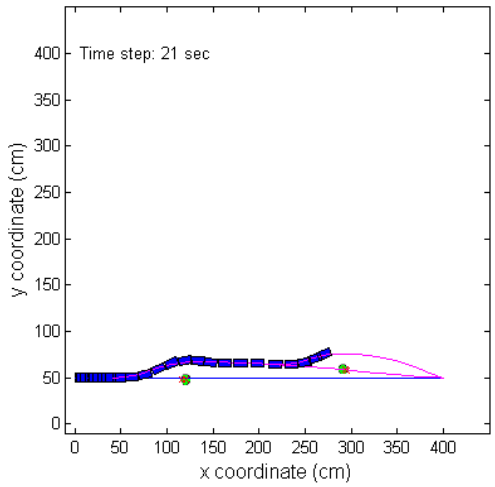
(b) At time = 6 s



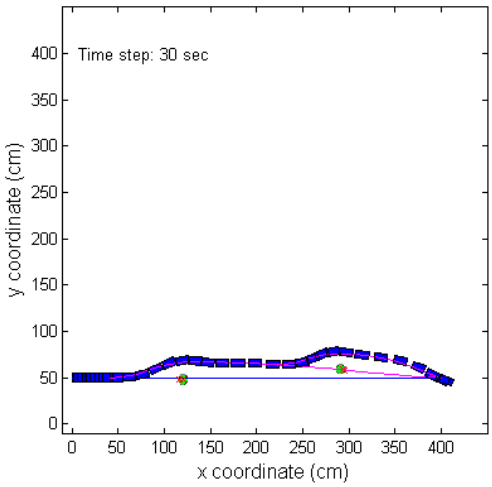
(c) At time = 10 s



(d) At time = 15 s

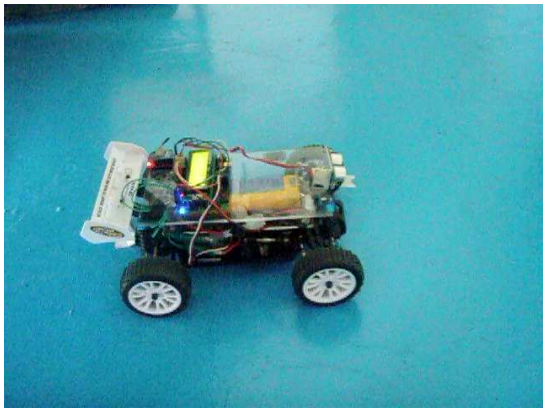


(e) At time = 21 s

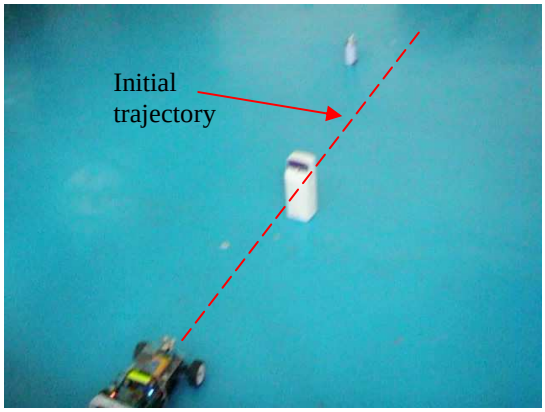


(f) At time = 30 s

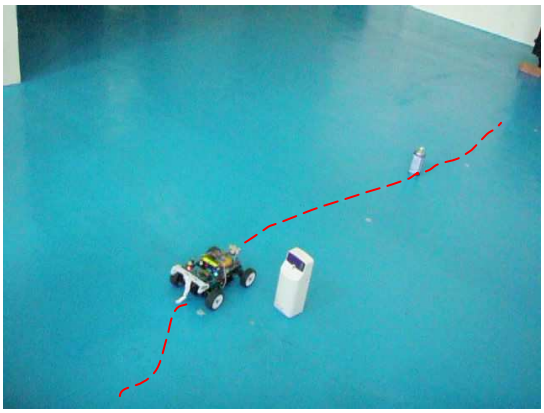
Figure 7.17 Mobile robot navigates through two unknown obstacles (simulation)



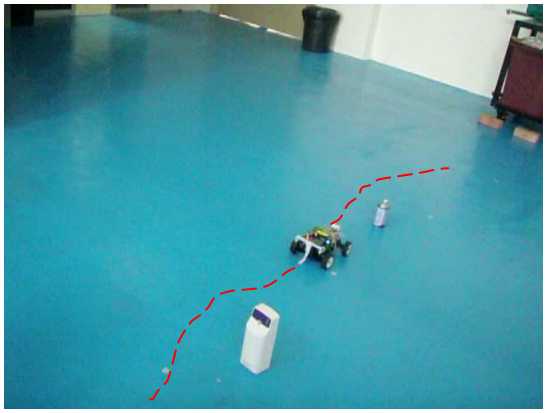
(a) At time = 0 s



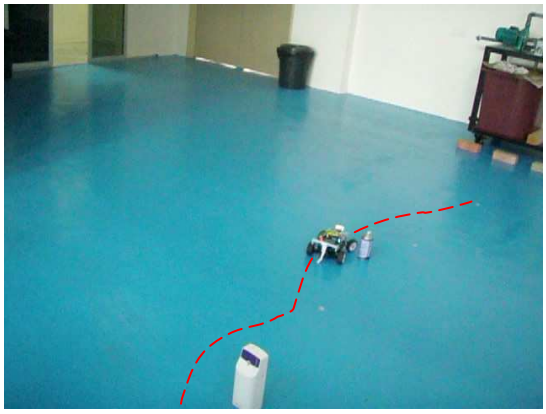
(b) At time = 6 s



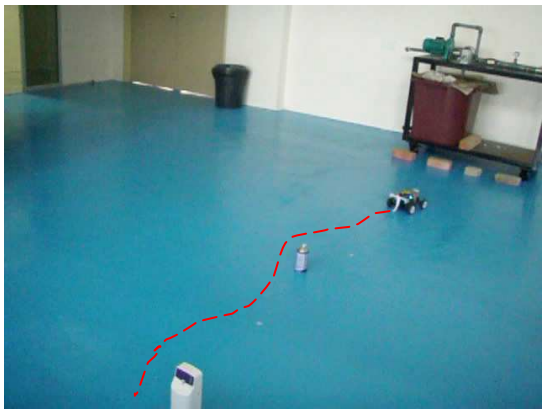
(c) At time = 10 s



(d) At time = 15 s



(e) At time = 21 s



(f) At time = 30 s

Figure 7.18 Mobile robot navigates through two unknown obstacles (experiment)

The experiment has been conducted in three trial runs and the final results were compared to the theory as shown in Figure 7.19.

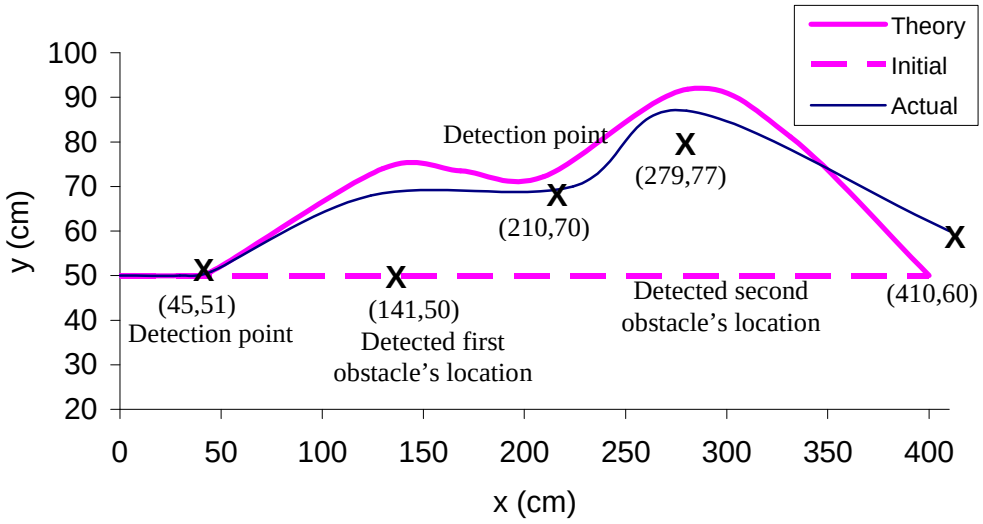


Figure 7.19 Theoretical and actual trajectory for Case 4

The errors in positions at the final point were compared with the theory and tabulated as in Table 7.4. The maximum errors recorded in *x*-axis and *y*-axis are around 6.3% and 36%, respectively.

Table 7.4 Actual initial and final positions for Case 4

	xs (cm)	**ys (cm)**	**xf (cm)**	**yf (cm)**	**x error (cm)**	**y error (cm)**
Theory	0	50	400	50	-	-
Trial 1	0	50	410	60	10	10
Trial 2	0	50	375	50	-25	0
Trial 3	0	50	377	68	-23	18

7.6 Case 4: Navigation in an unknown dynamic environments

In this section, the experimental works were conducted in order to validate the algorithm for moving obstacles. The dynamic obstacle avoidance approach was used to detect and avoid the moving obstacles as discussed in previous chapter. In this experiment, the remote control car was used as the moving obstacle. The algorithm was tested through a series of scenarios.

7.6.1 Scenario 1: Opposite direction of mobile robot

In the first scenario, the moving obstacle came from the opposite direction of the mobile robot as shown in Figure 7.20. The moving obstacle was placed randomly in front of the mobile robot. The distance from initial point to final point was set to 350 cm and the travelling time for the mobile robot was set to 30 seconds.

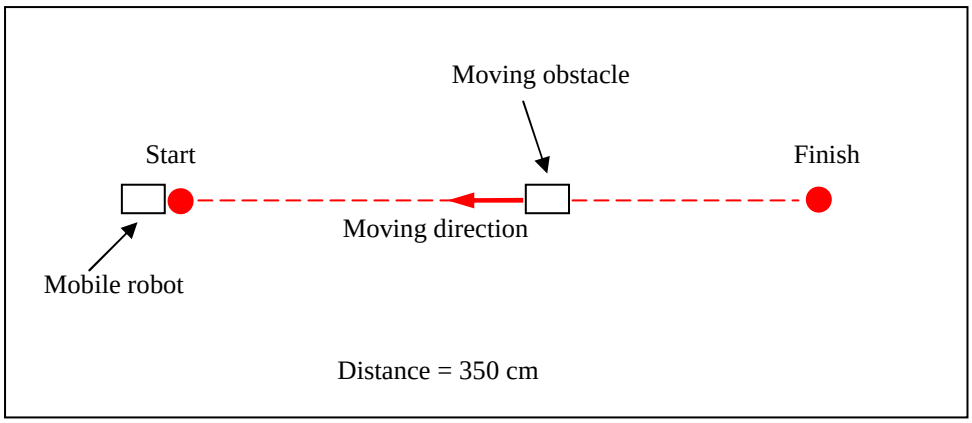
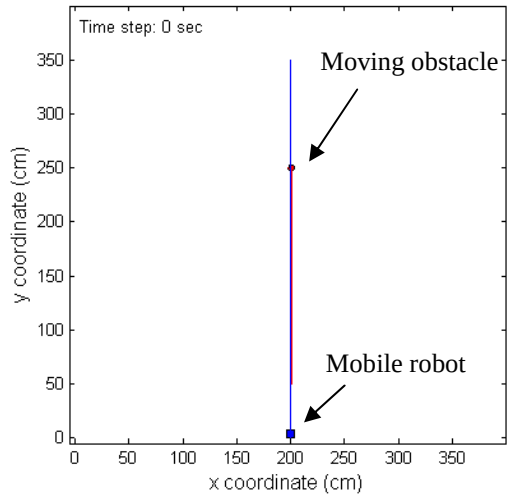
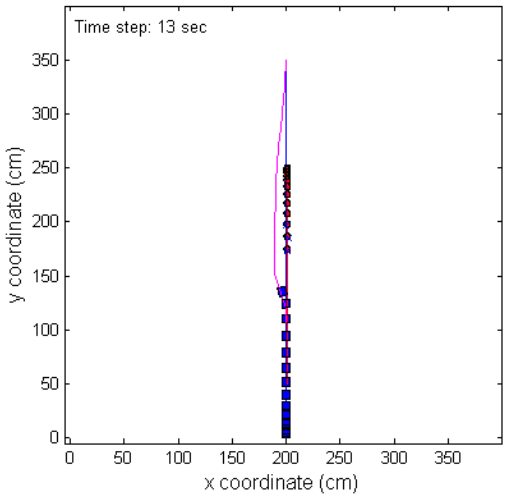


Figure 7.20 Moving obstacle coming from the opposite direction of the mobile robot

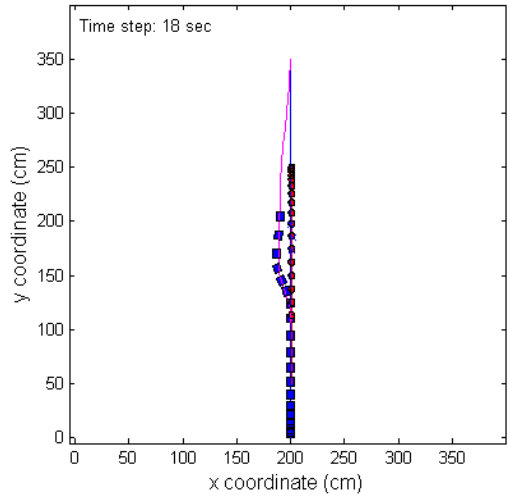
The simulation and experimental results are shown in Figure 7.21 and Figure 7.22, respectively. From the experimental results, the planned trajectory is represented by red dash line, while the actual trajectory is represented by solid red line. At the beginning of the experiment, the mobile robot was followed the initial collision-free trajectory until it detected the moving obstacle. Then the obstacle avoidance algorithm was executed and the new deviated trajectory was generated as shown in Figure 7.22(b). The mobile robot then avoided the mobile robot and followed new trajectory until it reached the final point as shown in Figure 7.22(f).



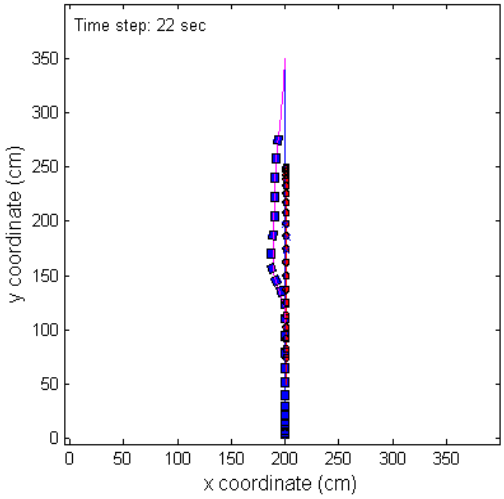
(a) At time = 0 s



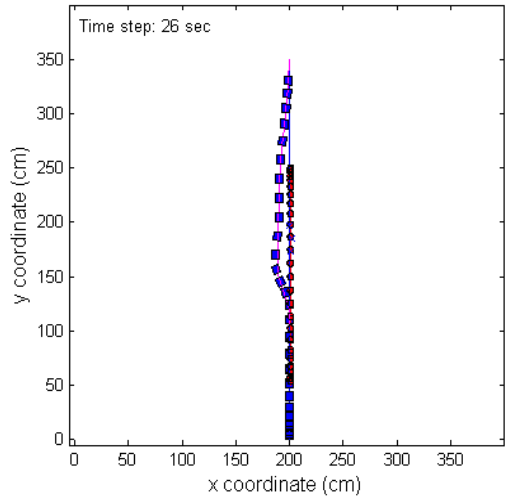
(b) At time = 13 s



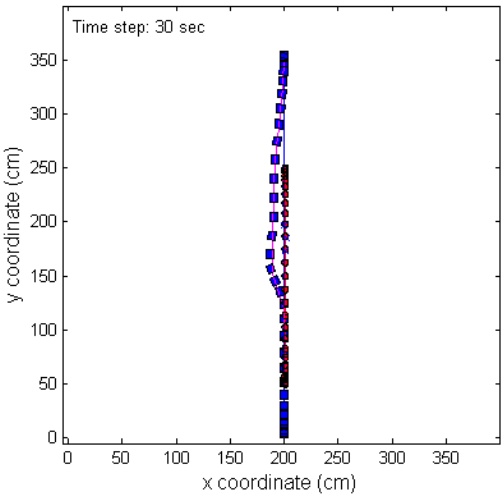
(c) At time = 18 s



(d) At time = 22 s

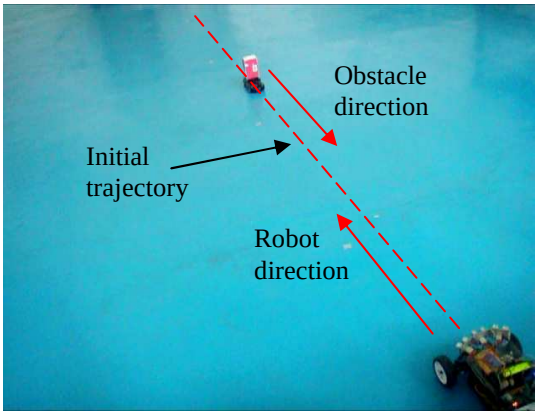


(e) At time = 26 s

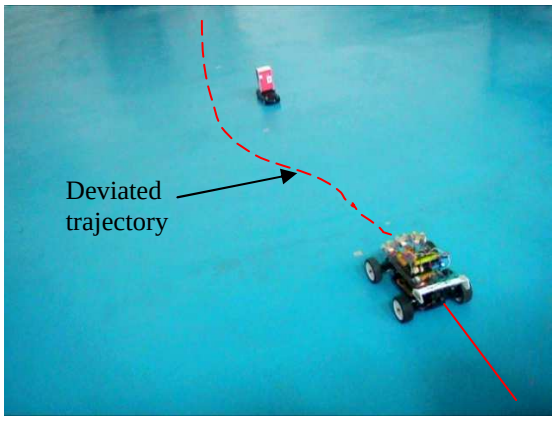


(f) At time = 30 s

Figure 7.21 Scenario 1: Moving obstacle from the opposite direction of the mobile robot (simulation)

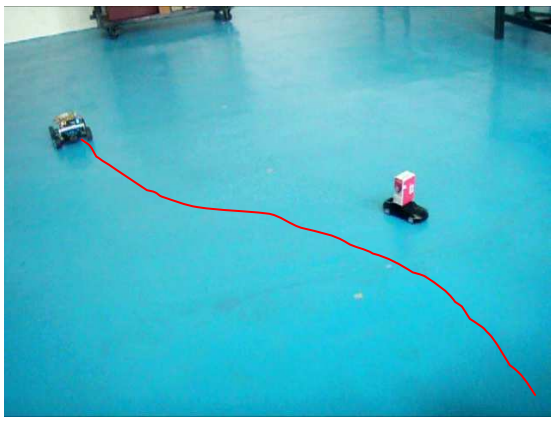


(a) At time = 0 s

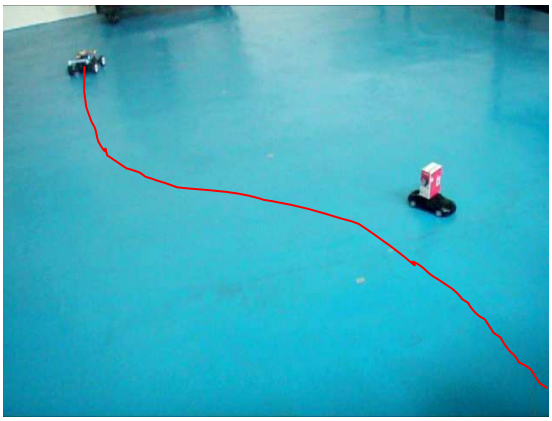


(b) At time = 13 s

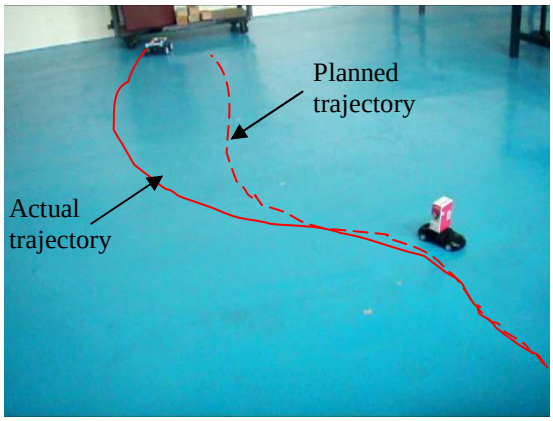
(c) At time = 18 s



(d) At time = 22 s



(e) At time = 26 s



(f) At time = 30 s

Figure 7.22 Scenario 1: Moving obstacle from the opposite direction of the mobile robot (experiment)

From the experiment results, the planned (theoretical) trajectory and the actual trajectory can be compared as shown in Figure 7.22(f) and Figure 7.23. The planned trajectory was extracted from Matlab as the real-time data were collected from the sensors in order to execute the obstacle avoidance algorithm. The mobile robot was able to follow the planned closely it detected the moving obstacle. It then started to deviate from the planned trajectory. At the final time, the mobile robot stopped at (370, 120) compared to the planned final point at (350, 100). The errors recorded for *x*-axis and *y*-axis for Scenario 1 are around 5.7% and 20%, respectively.

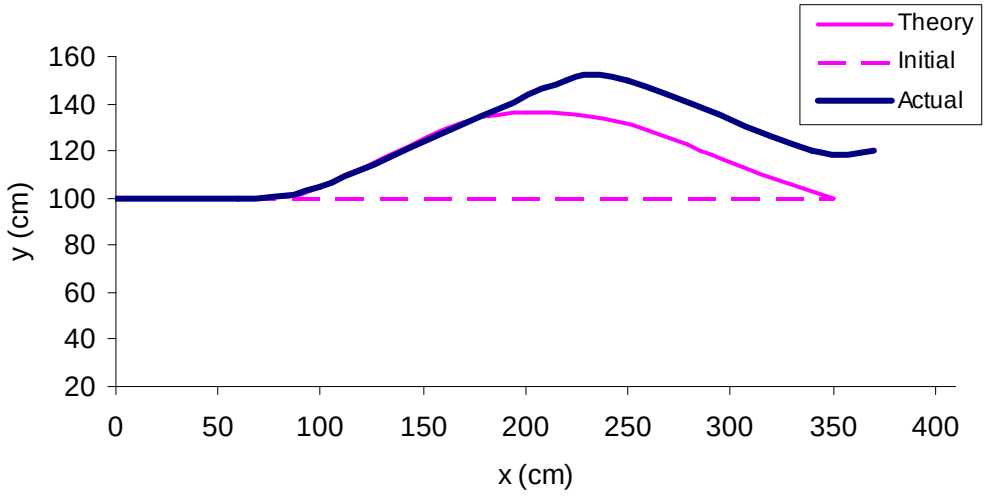


Figure 7.23 Theoretical and actual trajectory for scenario 1

7.6.2 Scenario 2: From left-hand side of mobile robot

In the second scenario, the moving obstacle came from the left-hand side direction of the mobile robot as shown in Figure 7.24. The moving obstacle was placed randomly at the left-hand side of the mobile robot and it moves on the straight line across the mobile robot from left to right. The distance from initial point to final point for the mobile robot was set to 350 cm and the travelling time was set to 30 seconds.

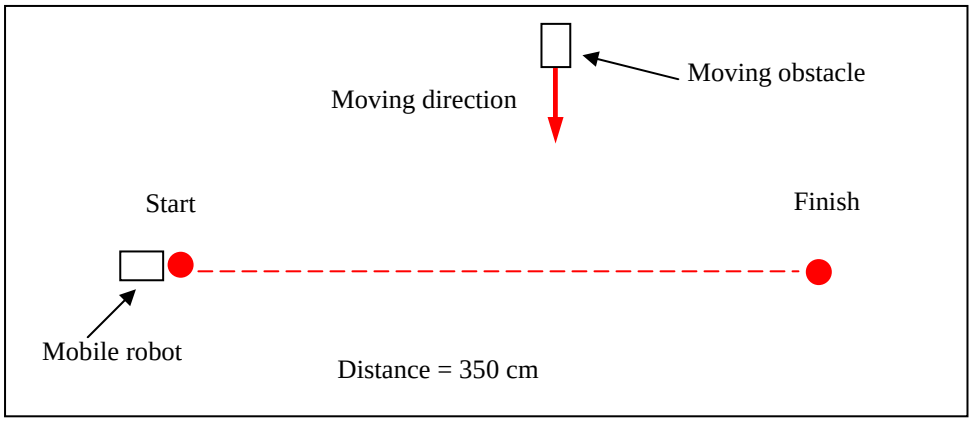
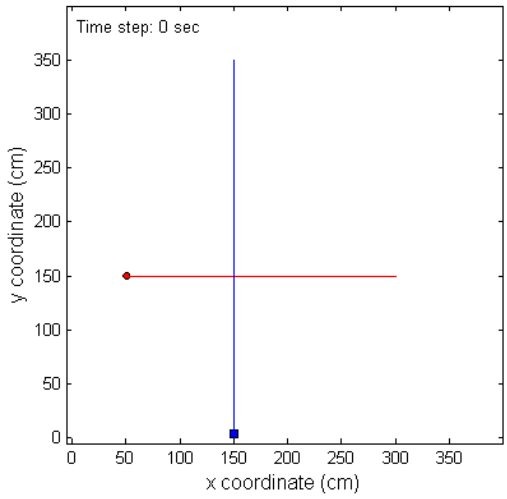
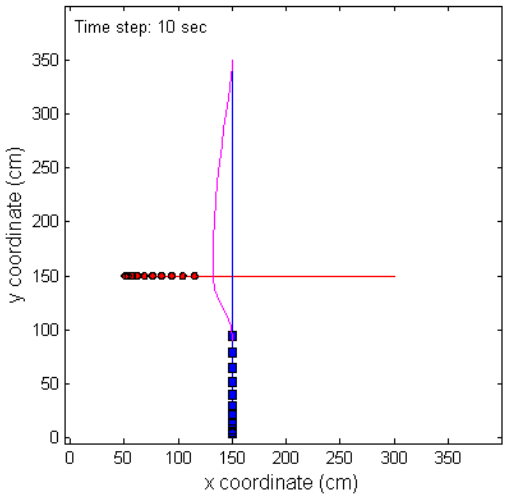


Figure 7.24 Moving obstacle coming from left-hand side of the mobile robot

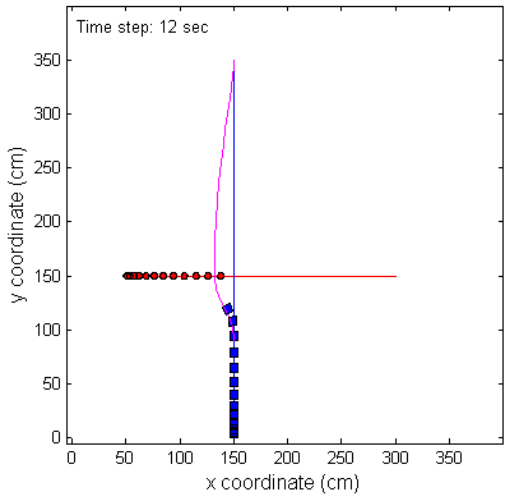
The simulation and experimental results are shown in Figure 7.25 and Figure 7.26, respectively. At the beginning of the experiment, the mobile robot was followed the initial collision-free trajectory until it detected the moving obstacle and a new deviated trajectory was generated as shown in Figure 7.26(b). The mobile robot was then followed the new deviated trajectory and it reached the final point as shown in Figure 7.26(f).



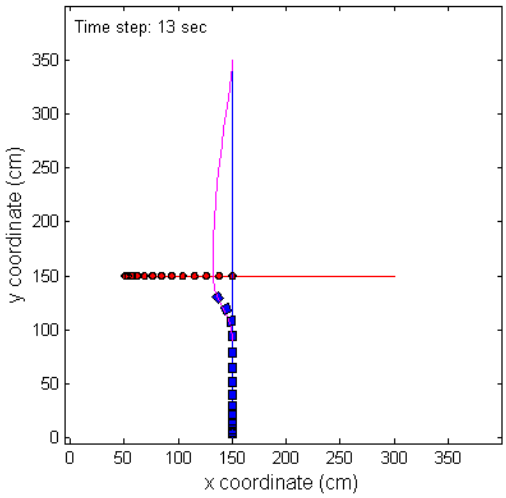
(a) At time = 0 s



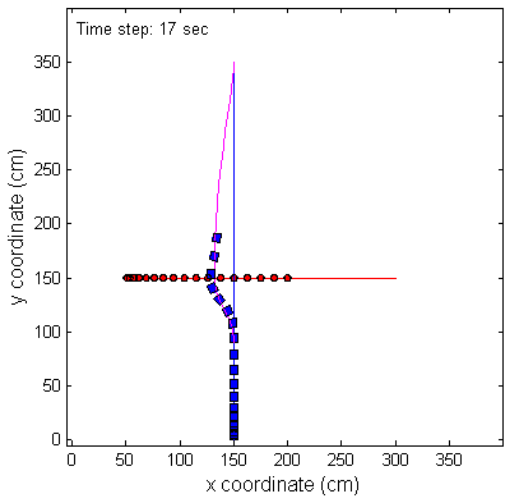
(b) At time = 10 s



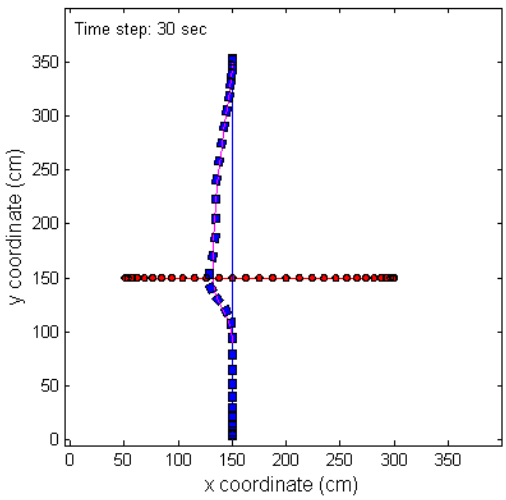
(c) At time = 12 s



(d) At time = 13 s

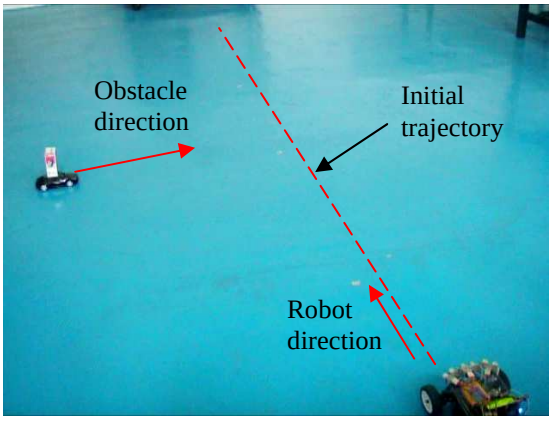


(e) At time = 17 s

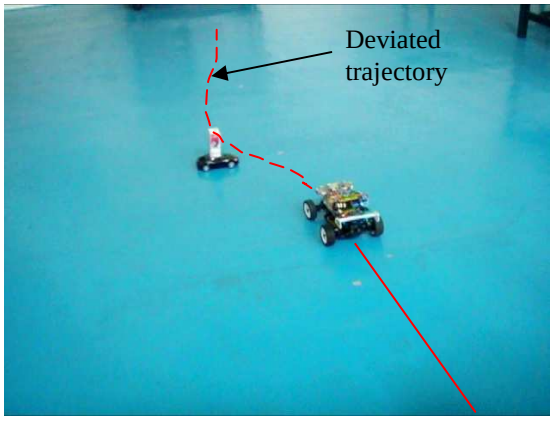


(f) At time = 30 s

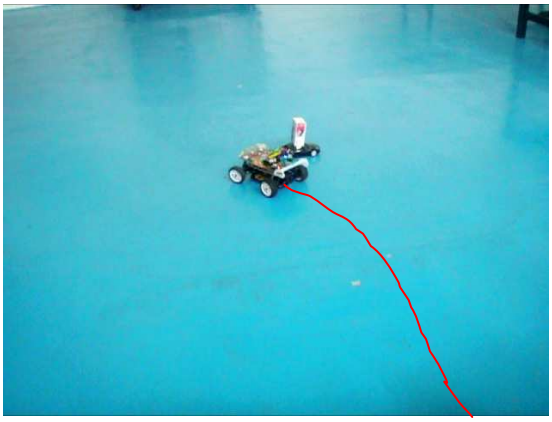
Figure 7.25 Scenario 2: Moving obstacle from the left-hand side of the mobile robot (simulation)



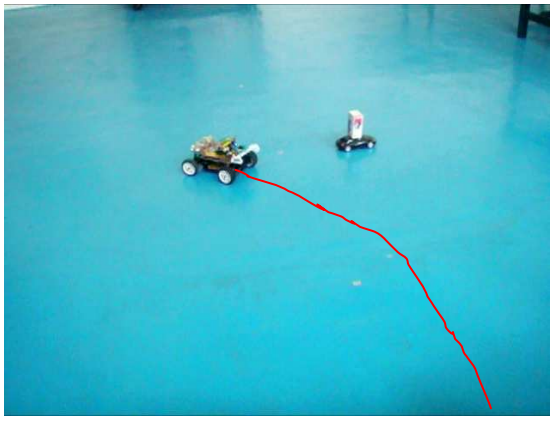
(a) At time = 0 s



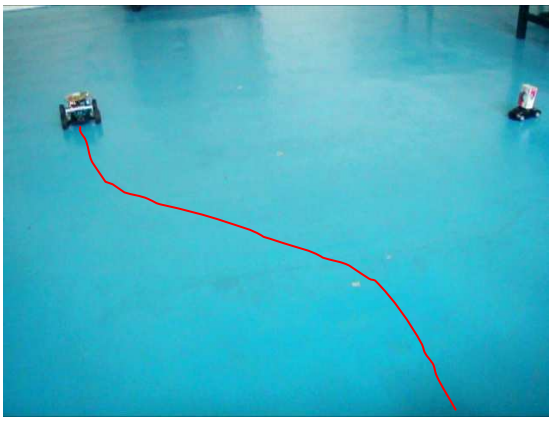
(b) At time = 10 s



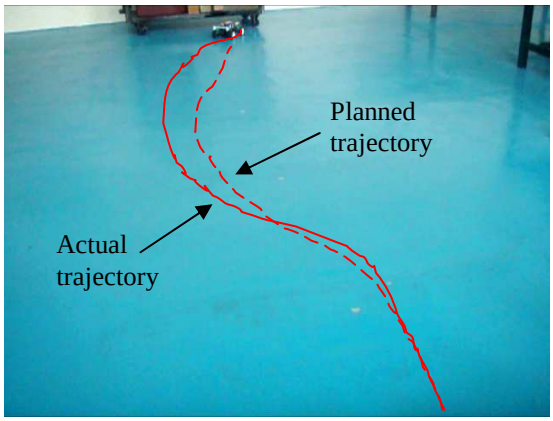
(c) At time = 12 s



(d) At time = 13 s



(e) At time = 17 s



(f) At time = 30 s

Figure 7.26 Scenario 2: Moving obstacle from the left-hand side of the mobile robot (experiment)

The comparison between the planned (theoretical) trajectory and the actual trajectory is shown in Figure 7.26(f) and Figure 7.27. From the figures, the mobile was able to follow the planned trajectory until it detected the moving obstacle. It then started to deviate from the new deviated trajectory. However the mobile robot was able to avoid the moving obstacle safely. At the final time, the mobile robot stopped at (360, 90) compared to the planned final point at (350, 100). The errors recorded for *x*-axis and *y*-axis for scenario 2 are around 2.9% and 10%, respectively.

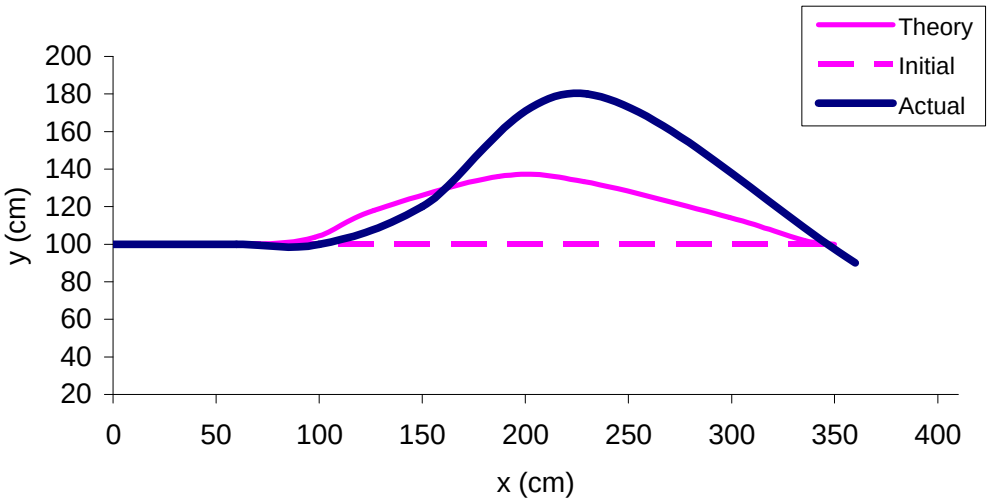


Figure 7.27 Theoretical and actual trajectory for scenario 2

7.6.3 Scenario 3: From right-hand side of mobile robot

In the third scenario, the moving obstacle came from the right-hand side direction of the mobile robot as shown in Figure 7.28. The moving obstacle was placed randomly at the right-hand side of the mobile robot and it moves on the straight line across the mobile robot from left to right. The distance from initial point to final point for the mobile robot was set to 350 cm and the travelling time was set to 30 seconds.

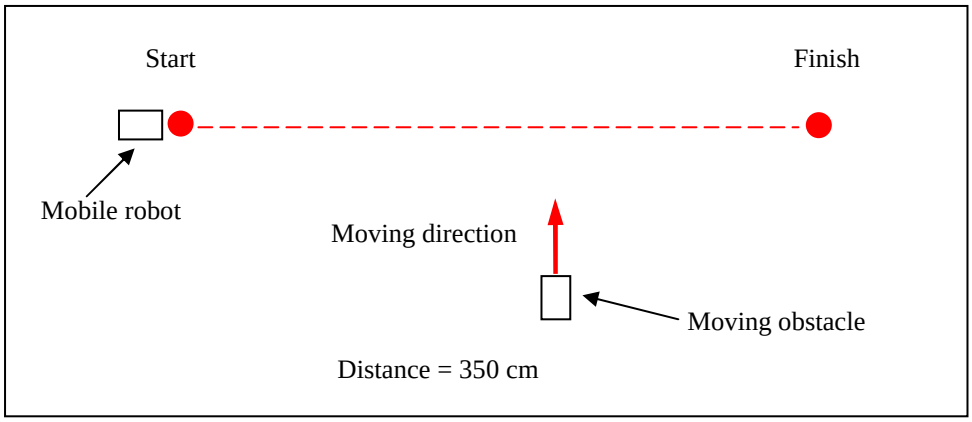
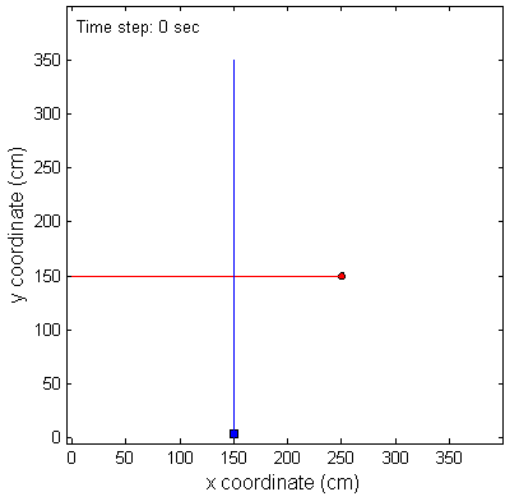
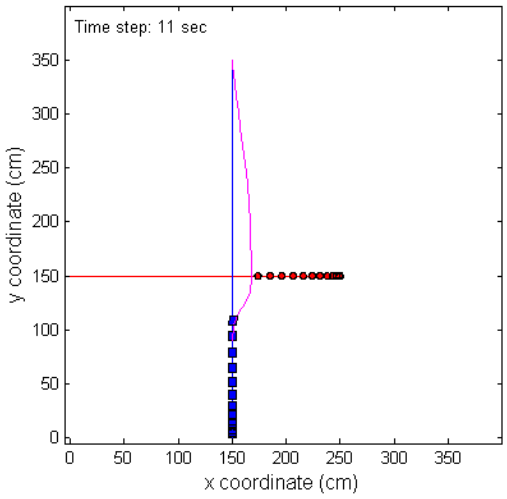


Figure 7.28 Moving obstacle coming from right-hand side of the mobile robot

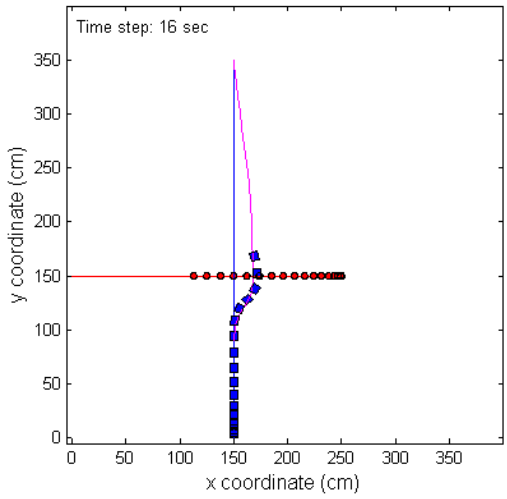
The simulation and experimental results are shown in Figure 7.29 and Figure 7.30, respectively. At the beginning of the experiment, the mobile robot was followed the initial collision-free trajectory until it detected and avoided the moving obstacle as shown in Figure 7.30(c). Then the new deviated trajectory was generated and the mobile robot was followed the new trajectory until it reached the final point as shown in Figure 7.30(f).



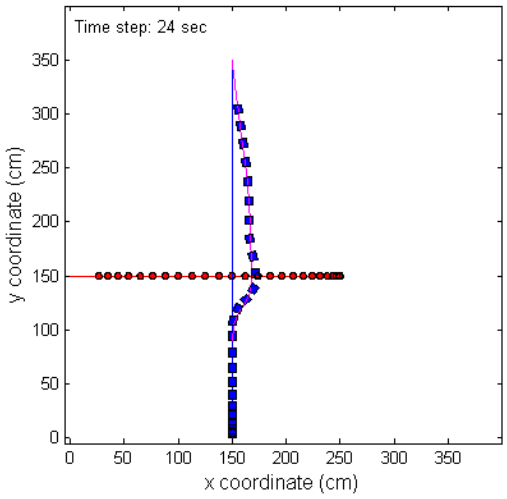
(a) At time = 0 s



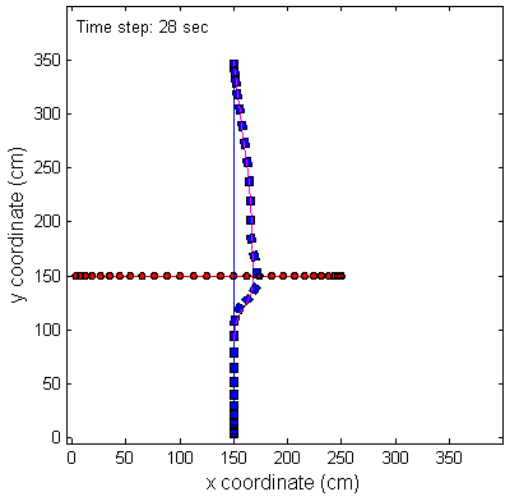
(b) At time = 11 s



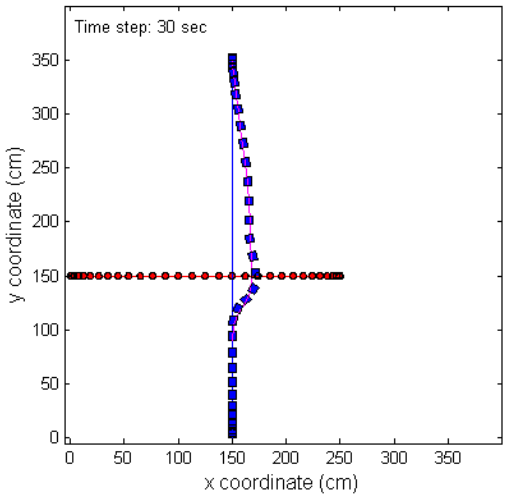
(c) At time = 16 s



(d) At time = 24

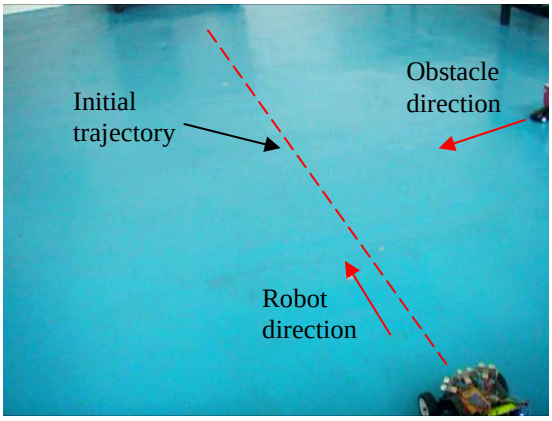


(e) At time = 28 s

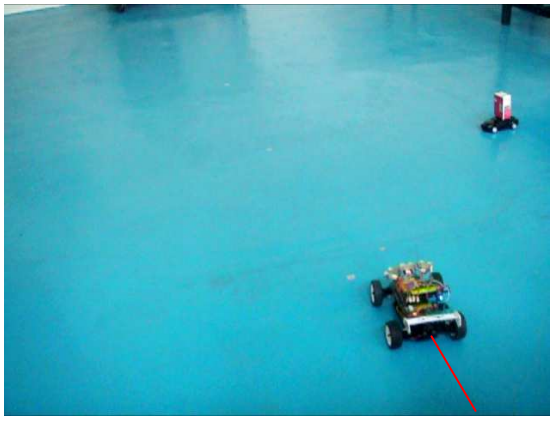


(f) At time = 30 s

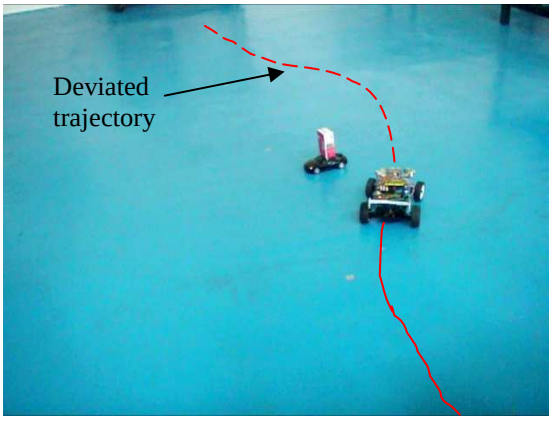
Figure 7.29 Scenario 3: Moving obstacle from the right-hand side of the mobile robot (simulation)



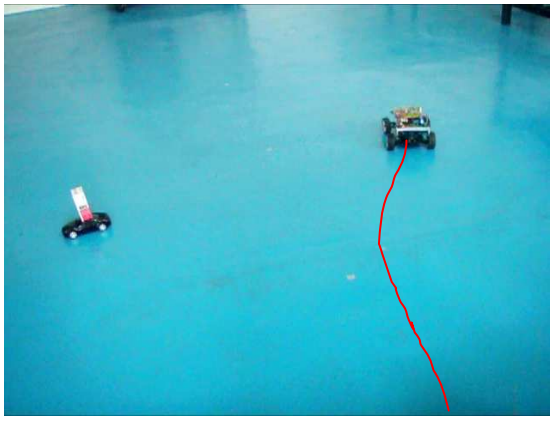
(a) At time = 0 s



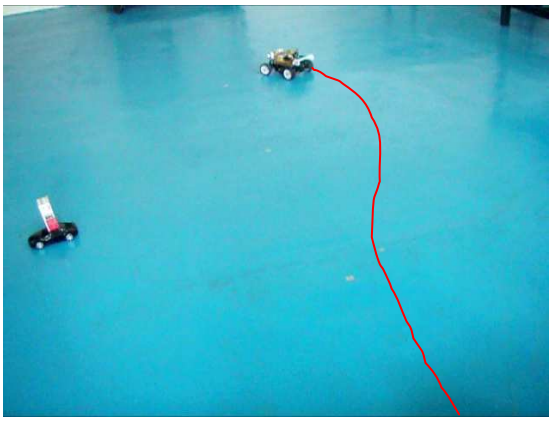
(b) At time = 11 s



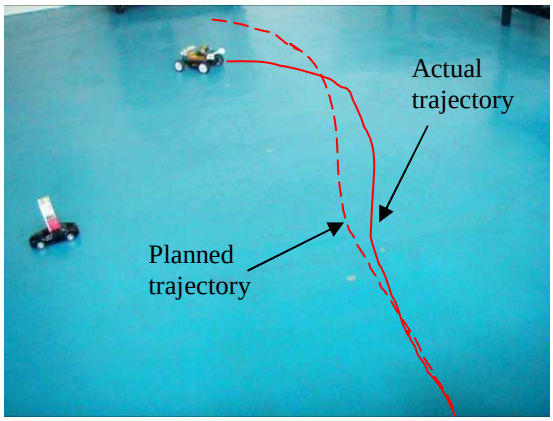
(c) At time = 16 s



(d) At time = 24



(e) At time = 28 s



(f) At time = 30 s

Figure 7.30 Scenario 3: Moving obstacle from the right-hand side of the mobile robot (experiment)

The comparison between the planned (theoretical) trajectory and the actual trajectory is shown in Figure 7.30(f) and Figure 7.31. From the figures, the mobile robot was able to follow the planned trajectory until it detected the moving obstacle. It then started to deviate from the new deviated trajectory. At the final time, the mobile robot stopped at (320, 110) which is shorter than the planned final point at (350,100). The errors recorded for *x*-axis and *y*-axis for scenario 3 are around 8.6% and 10%, respectively.

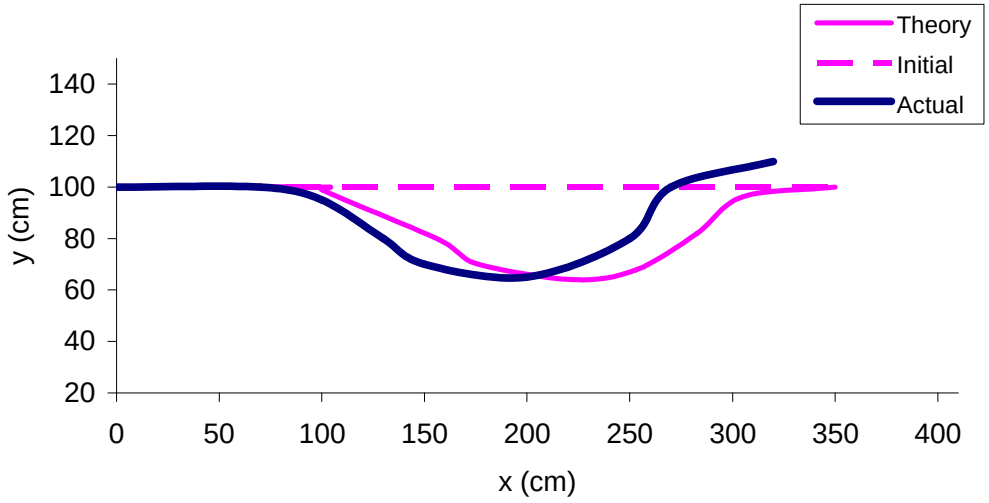


Figure 7.31 Theoretical and actual trajectory for scenario 3

7.7 Concluding remarks

From the experimental results, it was found that the mobile robot was capable to follow closely the planned trajectories. The errors for each case were compared between the planned and actual results at the final point for respective travel time. The position errors obtained from the experiments show an acceptable result as most of the trials for each case, the mobile robot has stopped close to the final point. Furthermore, the control strategy of the nonholonomic mobile robot was applicable to the real mobile robot and the control of the mobile robot was ideal with the calibration work which was conducted prior to the experimental works. The algorithm was also showed a good result in computational time which was shown by the mobile robot when it detected an obstacle and started to re-plan its trajectory in order to avoid the obstacle in the real testing arena. Furthermore, the integration of the sensors and

algorithm showed that the mobile robot was capable to detect and avoid the obstacle. This indicates the dynamic obstacle avoidance approach was practical to use to avoid the obstacle in the real experiments.

However, there is a slight deviation between the planned and n actual trajectories due to inaccuracy of actual steering angle and actual travelled distance caused by the open-loop system. The control strategy for a car-like robot can be developed further in order to obtain a better result. For example, the introduction of close-loop feedback control will ensure the speed and steering angle respond better to the calibrated speed and steering angle. In addition, the errors occurred due to:

1. The friction between the tyres and surface caused slippage and reduce the accuracy of velocity and distance recorded for the mobile robot.
2. The torque of the motor used to control the speed was quite small and the mobile robot needs a kick start to move.

Therefore, the newly developed algorithms are applicable and practical to be used for a car-like robot in real-time applications as demonstrated in the experimental works. The validation works for the algorithms and the verification of effectiveness of the simulation frameworks for the mobile robot were successfully conducted through a series of experimental setup. Furthermore the development of simulation framework will be used to further research and running more scenarios that are difficult to be conducted by experiment works.

8. CONCLUSIONS AND FUTURE WORKS

The reviewed literature indicates that there is no comprehensive research focused on the time-critical motion planning for a nonholonomic mobile robot. Several simulation works have been conducted for trajectory planning of the nonholonomic mobile robot to simulate the algorithms, but there was no experimental work was conducted to validate the algorithms (Guo *et al.*, 2003; Qu *et al.*, 2004; Guo *et al.*, 2007). This study has been carried out to further investigate the time-critical motion planning for a nonholonomic mobile robot. The geometry approach has been examined and used to generate the time-critical motion planning for mobile robots. Cubic and quintic polynomials are used to obtain a smooth and continuous trajectory for the mobile robots. The kinematic constraints of the mobile robot have been taken into consideration during the development of the algorithms. From the simulation results, all the cases proved that the algorithms are practical to be used in motion planning for single and multiple mobile robots. Furthermore the experimental works validate and verify the algorithms and the control strategies of the mobile robot.

In this chapter, all the findings will be concluded as a whole and the future works for this study will be described.

Contributions

This study had made several contributions to the current knowledge and the contributions are summarized as followings.

(1) Development of the time-critical motion planning algorithms for nonholonomic mobile robots.

In this study, a new time-critical motion planning algorithm was developed for nonholonomic mobile robots. The movement of the mobile robot can be analysed through this algorithm with given initial and final states, which are *x*-position, *y*-position, orientation, steering angle, velocity and time, of the mobile robot. The environment setup can be easily changed by adopting a required map into the algorithm. This algorithm can also be utilised for multiple mobile robots planning with each mobile robot has a different initial and final state.

The advantage of this algorithm is it can indicate the initial and final trajectory at every time step for each scenario that has been set. In addition the size of the mobile robot can be set simply by changing the parameter of the mobile robot to suit the real mobile robot. Furthermore, using MATLAB as an object-oriented programming allows a real time programming which is used in real time experiments.

On the other hand, the planned trajectory has considered obstacles that known in prior to the planner before the mobile robot starts to move from the initial point. The advantage of this approach is the planner will generated an initial free-collision trajectory for the mobile robot. This will ensure the mobile robot will not collide with any known obstacle while manoeuvring through the environment and will able to reach the final point at the desired time.

Furthermore, the developed 2D simulation framework in Matlab give a user friendly interface for the user to set the initial and final state of the mobile robot before running the simulation.

(2) Development of the dynamic obstacle avoidance approach

In this study, a new strategy in avoiding an obstacle has been is developed for a time-critical motion planning for nonholonomic mobile robots. The obstacle avoidance approach has modified the initial trajectory in order to avoid the obstacle and at the same time, the new generated trajectory will ensure the mobile robot reach the final point at the desired time.

The advantage of this dynamic obstacle avoidance approach is it can compensate the time lost during avoiding the obstacle. The obstacle avoidance algorithm ensures the mobile robot able to avoid unknown static and dynamic obstacles which the mobile robot encounters during navigating through the environments. To the current knowledge, this dynamic obstacle avoidance approach is the first obstacle avoidance approach that considering timing when encountering an obstacle.

(3) Validation and verification works through series of simulations and experiments

One of the most important parts of this study is to validate the algorithm and effectiveness of the simulation framework with the real mobile robot. A car-like robot was used to carry out the simulated trajectory in the real environment. At the preparation stage, the steering angle and speed of the mobile robot were calibrated. Then an actual mobile robot was used to validate and verify the theory.

The experiments were conducted through a series of cases. Basically, there were four cases have been carried out for this study. The first case was to verify the control strategy of the car-like robot. The second case was to validate the collision-free time-critical motion planning algorithm. In this case, a known static obstacle was placed in the environment and the obstacle was known to the planner prior to trajectory generation. The third and forth cases were carried out in the unknown static

environments to validate the dynamic obstacle avoidance algorithm. In the third case, a static obstacle was place in the environment, but it was not known to the planner prior to trajectory generation. While in the forth case, the scenario was extended by placing more unknown static obstacles. Finally the fifth case was carried out in the unknown dynamic environments to test and validate the algorithm for moving obstacles. There were three scenarios which differentiated by the moving direction of the moving obstacle from the mobile robot – moving from opposite side, left-hand side and right-hand side of the mobile robot. For each case, the experimental results were then being compared to the theory or simulation results.

From the experiment results, the mobile robot was able to navigate through the environment and reach the final point at the desired time, with capability to avoid the obstacle along its way. This shows that the mobile robot can be used for a task-based mission as the mobile robot can be set to reach a certain waypoint or the final point at the desired time. Furthermore, the setting of parameter for the algorithm is also flexible as the modification can only be made in Matlab interface without interfering the onboard control algorithm at the mobile robot.

Future works

This study can be further investigate and improve in the future to establish a robust and optimize algorithms that can be used for the real applications. The possible future works for this study are briefly described as followings.

(1) Optimization of the time-critical motion planning algorithm.

Currently, the algorithm is not considering the overall distance of the trajectory for the mobile robot from the initial to the final point. The algorithm can be optimised in order to ensure the mobile robot will travel using the shortest trajectory. In addition, the usage of battery for mobile robots is one the main concerns. Thus the energy management approach can be adopted for the algorithm so that the mobile robot can travel for long distance.

Furthermore, in order to obtain a better driving experience to the mobile robot, the velocity can be optimised. For example, at the straight line, the mobile robot can achieve a higher speed than at the curve. This may has impact on dynamic of the mobile robot during cornering. Furthermore the mobile robot might be also start at a higher speed from the initial point and will slow down when it approaching the final point.

At this moment, the steering angle and speed are set with an open loop system which means there is no feedback control. However, the experimental results show the mobile robot was able to perform the driving along the planned trajectory well, but introducing a feedback controller can increase the accuracy results in real time experiments.

(2) Experimental work for multiple mobile robots in dynamic environments.

At this moment, the experimental works only conducted in the static and dynamic environments with single mobile robot. In the future, the experiment can be conducted in environments with the presence of multiple mobile robot and moving obstacles such as other mobile robot and human. However, prior to the experiment, the mobile robot need to equip with a better detection sensor for moving obstacle such as laser range finder (LRF) or a mobile camera. These sensors will ensure the moving obstacle can be detected and tracked so that the algorithm will be able to plan the next step.

Furthermore, the limitations of the current car-like robot need to be considered. The mobile robot was modified from the small scale RC car and will not be suitable to carry a large sensor such as LRF. The battery life also needs to be taking into consideration as the battery will power all the sensors, motor, servo and microcontroller.

REFERENCE

Brezak, M., I. Petrovic and E. Ivanjko (2008). "Robust and accurate global vision system for real time tracking of multiple mobile robots", Robotics and Autonomous Systems, vol.56, pp. 213-230.

Brock, O. and O. Khatib (1999). "High-speed navigation using the global dynamic window approach", International Conference on Robotics and Automation, Michigan, USA, 10-15 May 1999.

Castillo, G. D., S. Skaar, A. Cardenas and L. Fehr (2006). "A sonar approach to obstacle detection for a vision-based autonomous wheelchair", Robotics and Autonomous Systems, vol.54, pp. 967-981.

Chang, Y.-C., Y. Y. Lwin and Y. Yamamoto (2009). Sensor-based trajectory planning strategy for non-holonomic mobile robot with laser range sensors. IEEE International Symposium on Industrial Electronics (ISIE'09). Seoul, Korea.

Cosio, F. A. and M. A. P. Castaneda (2004). "Autonomous robot navigation using adaptive potential fields", Mathematical and Computer Modelling, vol.40(2004), pp. 1141-1156.

Delingette, H., M. Hebert and K. Ikeuchi (1991). "Trajectory generation with curvature constraint based on energy minimization", International Conference on Intelligent Robotics Systems, Osaka, Japan, November 1991.

Dong, W. and Y. Guo (2005). "New trajectory generation methods for nonholonomic mobile robots", International Symposium on Collaborative Technologies and Systems, Missouri, USA, 15-20 May 2005.

Duan, Y., Q. Liu and X. Xu (2007). "Application of reinforcement learning in robot soccer", Engineering Applications of Artificial Intelligence, vol.20, pp. 936-950.

Fajen, B. R. and W. H. Warren (2003). "Behavioral dynamics of steering obstacle avoidance and route selection", Journal of Experimental Psychilogy: Human and Perception and Performance, vol.29(2), pp. 343-362.

Fajen, B. R., W. H. Warren, S. Temizer and L. P. Kaelbling (2003). "A dynamic model of visually-guided steering, obstacle avoidance and route selection", International Journal of Computer Vision, vol.54(1/2/3), pp. 13-34.

Ferrara, A. and M. Rubagotti (2009). A dynamic obstacle avoidance strategy for a mobile robot based on sliding mode control. IEEE International Conference on Control and Applications. Saint Petersburg, Russia.

Fox, D., W. Burgard and S. Thrun (1997). "The dynamic window approach to collision avoidance", IEEE Robotics and Automation Magazine, vol.4(1), pp. 23-33.

Ge, S. S. and F. L. Lewis (2006). "Autonomous mobile robot: Sensing, control, decision making and applications", Taylor and Francis Group, 2006.

Ghita, N. and M. Kloetzer (2012). "Trajectory planning for a car-like robot by environment abstraction", Robotics and Autonomous Systems, vol.60, pp. 609-619.

Guo, Y., Z. Qu and J. Wang (2003). "A New Performance-Based Motion Planner for Nonholonomic Mobile Robots", The 3rd Performance Metrics for Intelligent Systems Workshop, Gaithersburg, MD, USA, 16-18 September 2003.

Haddad, M., T. Chettibi, S. Hanchi and H. E. Lehtihet (2007). "A random-profile approach for the trajectory planning of the wheeled mobile robots", European Journal of Mechanics A/Solids, vol.26, pp. 519-540.

Hamner, B., S. Singh and S. Scherer (2006). "Learning obstacle avoidance parameters from operator behavior", Journal of Field Robotics, vol.23(11/12), pp. 1037-1058.

Hazon, N. and G. A. Kaminka (2008). "On redundancy, effieciency and robustness in coverage for multiple robots", Robotics and Autonomous Systems, vol.56, pp. 1102-1114.

Hong, J., Y. Choi and K. Park (2007). Mobile robot navigation using modified flexible Vector field approach with laser range finder and IR sensor. International Conference on Control, Automation and Systems. Seoul, Korea.

Huang, L. (2009). "Velocity planning for a mobile robot to track a moving target - a potential field approach", Robotics and Autonomous Systems, vol.57, pp. 55-63.

Huang, W. H., B. R. Fajen, J. R. Fink and W. H. Warren (2006). "Visual navigation and obstacle avoidance using a steering potential function", Robotics and Autonomous Systems, vol.54, pp. 288-299.

Hui, N. B., V. Mahendar and D. K. Pratihar (2006). "Time-optimal, collision-free navigation of a car-like mobile robot using neuro-fuzzy approach", Fuzzy Sets and Systems, vol.157, pp. 2171-2204.

Jacob, M. (2008). Path planning and obstacle avoidance in unknown dynamic environments.

Jiang, K., L. D. Senevirate and S. W. E. Earles (1997). "TIme-optimal smooth-path planning for a mobile robot with the kinematic constraints", Robotica, vol.15, pp. 547-553.

Jolly, K. G., R. S. Kumar and R. Vijayakumar (2008). "A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits", Robotics and Autonomous Systems, vol.57, pp. 23-33.

Jolly, K. G., K. P. Ravindran, R. Vijayakumar and R. S. Kumar (2007). "Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks", Robotics and Autonomous Systems, vol.55, pp. 589-596.

Kanayama, Y. and N. Miyake (1986). "Trajectory generation for mobile robots", 3rd Symposium on Robotics Research, Gourvieux, France, 1986.

Kelly, A. (2003). "Efficient parametric synthesis of optimal mobile robot trajectories", 11th International Conference on Advanced Robotics, University of Coimbra, Portugal, 30 June - 3 July 2003.

Khatib, O. (1986). "Real-time obstacle avoidance for robot manipulator and mobile robots", International Journal of Robotics Research, vol.5(1), pp. 90-98.

Klancar, G., M. Kristan, S. Kovacic and O. Orqueda (2004). "Robust and efficient vision system for group of cooperating mobile robots with application to soccer robots", ISA Transactions, vol.43, pp. 329-342.

Koh, K. C. and H. S. Cho (1999). "A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints", Journal of Intelligent and Robotic Systems, vol.24, pp. 367-385.

Li, L. and F.-Y. Wang (2003). "Trajectory generation for driving guidance of front wheel steering vehicles", IEEE Intelligent Vehicles Symposium, Ohio, USA, 9-11 June 2003.

Liang, T.-C., J.-S. Liu, G.-T. Hung and Y.-Z. Chang (2005). "Practical and flexible path planning for a car-like mobile robot using maximal-curvature cubic spiral", Robotics and Autonomous Systems, vol.52, pp. 312-335.

Liddy, T. J. and T.-F. Lu (2007). "Waypoint navigation with position and heading control using complex vector fields for an Ackermann steering autonomous vehicle", Australasian Conference on Robotics and Automation, Brisbane, Australia, 10-12 December 2007.

Liu, S. and D. Sun (2011). Optimal motion planning of a mobile robot with minimum energy consumption. 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2011). Budapest, Hungary.

Ma, Y., G. Zheng and W. Perruquetti (2013). "Real-time local path planning for mobile robots", 9th International Workshop on Robot Motion and Control, Wasowo, Poland, 3-5 July 2013.

Mihaylova, L., J. D. Schutter and H. Bruyninckx (2003). "A multisine approach for trajectory generation optimization based on the information gain", Robotics and Autonomous Systems, vol.43, pp. 231-243.

Nagatani, K., Y. Iwai and Y. Tanaka (2001). "Sensor based navigation for car-like mobile robots using generalized Voronoi Graph", IEEE/RSJ International Conference on Intelligent Robots and Systems, Hawaii, USA, 29 Oct. - 3 Nov. 2001.

Nagy, B. and A. Kelly (2001). "Trajectory generation for car-like robots using cubic curvature polynomials", International Conference on Field and Service Robots, Helsinki, Finland, June 1991.

Pin, F. G. and H. A. Vasseur (1990). "Autonomous trajectory generation for mobile robots with non-holonomic and steering angle constraints", IEEE International Workshop on Intelligent Motion Control, Istanbul, Turkey, 20-22 August 1990.

Prado, M., A. Simon, E. Carabias, A. Perez and F. Ezquerro (2003). "Optimal velocity planning of wheeled mobile robots on specific paths in static and dynamic environments", Journal of Robotic Systems, vol.20(12), pp. 737-754.

Qu, Z., J. Wang and C. E. Plaisted (2004). "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles", IEEE Transaction on Robotics and Automation, vol.20, pp. 978-993.

Safadi, H. (2007). Local path planning using virtual potential field, School of Computer Science, McGill University.

Salichs, M. A. and L. Moreno (2000). "Navigation of mobile robots: Open question", Robotica, vol.18, pp. 227-234.

Shin, D. and S. Singh (1990). Path generation for robot vehicles using composite clothoid segments. Pittsburgh, Pennsylvania, Carnegie-Mellon University.

Siegwart, R. and I. R. Nourbakhsh (2004). "Introduction to autonomous mobile robots", The MIT Press, 2004.

Sridharan, K. and T. K. Priya (2004). "A parallel algorithm for constructing reduced visibility graph and its FPGA implementation", Systems Architecture, vol.50(2004), pp. 635-644.

Stroupe, A., T. Huntsberger, A. Okon, H. Aghazarian and M. Robinson (2005). "Behavior-based multi-robot collaboration for autonomous constraction tasks", International Conference on Intelligent Robots and Systems, Alberta, Canada, 2-6 August 2005.

Tounsi, M. and J. F. L. Corre (1996). "Trajectory generation for mobile robots", Mathematics and Computers in Simulation, vol.41, pp. 367-376.

Treptow, A. and A. Zell (2004). "Real-time object tracking for soccer robots without color information", Robotics and Autonomous Systems, vol.48, pp. 41-48.

Victorino, A. C., P. Rives and J.-J. Borrelly (2001). "Mobile robot navigation using a sensor-based control strategy", IEEE International Conference on Robotics 8 Automation, Seoul, Korea, 21-26 May 2001.

Wein, R., J. P. v. d. Berg and D. Halperin (2007). "The Visibility-Voronoi complex and its applications", Computational Geometry, vol.36(2007), pp. 66-87.

Yamaguchi, H. (2003). "A distributed motion coordination strategy for multiple nonholonomic mobile robots in cooperative hunting operations", Robotics and Autonomous Systems, vol.43, pp. 257-282.

Zhang, S., L. Xie and M. D. Adams (2006). "Feature extraction for outdoor mobile robot navigation based on a modified Gauss–Newton optimization approach", Robotics and Autonomous Systems, vol.54(2006), pp. 277-287.

APPENDIX A

DATASHEETS

A1. Robot Controller (http://www.pololu.com/catalog/product/1327)

Specifications

- Overall unit dimensions: 3.70" × 2.20"
- Input voltage: 6 – 13.5 V
- Programmable 20 MHz Atmel ATmega324PA AVR microcontroller with 32 KB flash, 2 KB SRAM, and 1 KB EEPROM *
- Programmable 20 MHz Atmel ATmega1284P AVR microcontroller with 128 KB flash, 16 KB RAM, and 4 KB EEPROM *(SVP-1284 version)
- Built-in USB AVR ISP programmer (USB A to mini-B cable
- 2 bidirectional motor ports (2 A continuous per channel, 6 A maximum
- 8-output demultiplexer tied to one of the AVR's hardware PWMs for easy control of up to 8 servos
- 21 free I/O lines
    - 17 free I/O lines on the main MCU, of which 8 can be analog inputs
    - 4 input lines on the auxiliary processor, which can be either 4 analog inputs or dual quadrature encoder inputs
    - 2 hardware UARTs
- Removable 16-character × 2-line LCD with backlight
- Primary 5V switching regulator capable of supplying 3 A
- Secondary adjustable (2.5 V – 85% of VIN) buck (step-down) voltage regulator capable of supplying 3 A
- Buzzer tied to one of the AVR's hardware PWMs
- 3 user pushbutton switches
- 2 user LEDs
- Power (push-on/push-off) and reset pushbutton switches
- Power circuit makes it easy to add extra power buttons and provides a self-shutdown option
- Auxiliary processor (connected via SPI) provides:
    - Battery voltage reading
    - User trimmer potentiometer reading
    - Integrated USB connection
    - In-System-Programming of the main processor
    - Ability to read two quadrature encoders

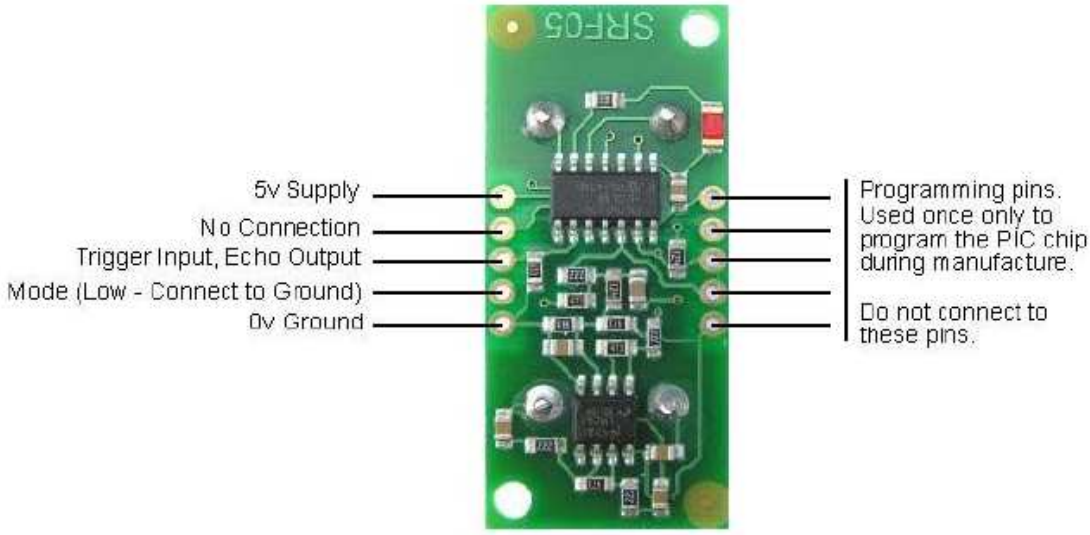
A2. Ultrasonic Sensor (http://www.robot-electronics.co.uk/htm/srf05tech.htm)

SRF05
Range - 1cm to 4m.
Power - 5v, 4mA Typ.
Frequency - 40KHz.
Size - 43mm x 20mm x 17mm height.

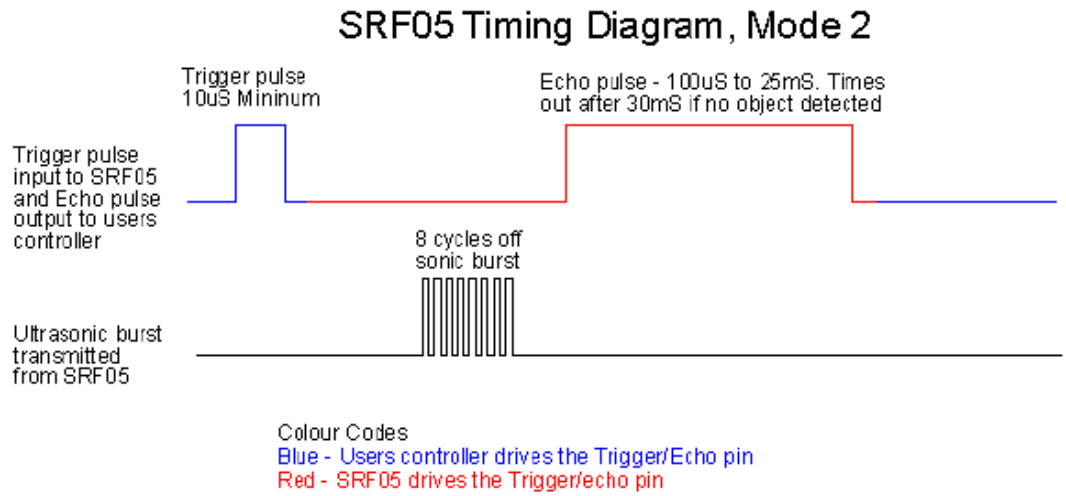
Two operational modes are available, Single pin for trig/echo or 2 Pin SRF04 compatible.

The input Trigger is a 10uS Min. TTL level pulse
Echo Pulse is Positive TTL level signal, with the width proportional to the object range.

Mode 2 – Single pin for both Trigger and Echo



Connections for single pin Trigger/Echo Mode

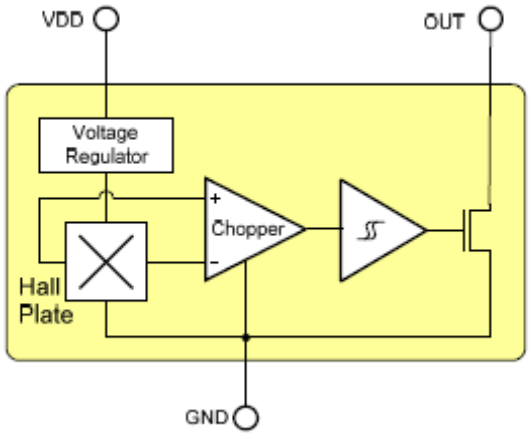


A3. Hall Effect Sensor (http://www.melexis.com/Hall-Effect-Sensor-ICs/Hall-Effect-Latches/US1881-140.aspx)

Features and Benefits

- Wide operating voltage range from 3.5V to 24V
- High magnetic sensitivity – Multi-purpose
- CMOS technology
- Chopper-stabilized amplifier stage
- Low current consumption
- Open drain output
- Thin SOT23 3L and flat TO-92 3L both RoHS Compliant packages

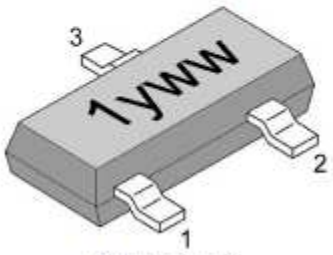
Functional diagram



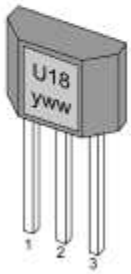
Pin definitions and descriptions

SE Pin №	UA Pin №	Name	Type	Function
1	1	VDD	Supply	Supply Voltage pin
2	3	OUT	Output	Open Drain Output pin
3	2	GND	Ground	Ground pin

Table 2: Pin definitions and descriptions



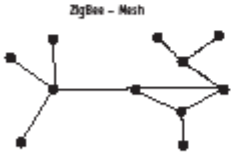
SE package

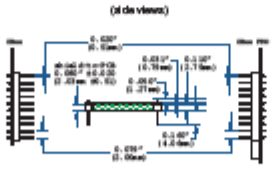


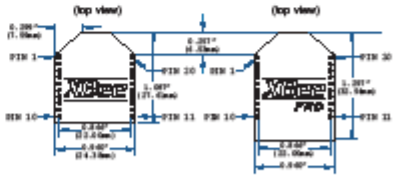
UA package

A4. Wireless Communication (http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module)

Platform	XBee® ZB	XBee-PRO® ZB	Programmable XBee-PRO® ZB
Performance			
RF Data Rate	250 Kbps		
Indoor/Urban Range	133 ft (40 m)	300 ft (90 m)	
Outdoor/RF Line-of-Sight Range	400 ft (120 m)	2 miles (3200 m) / Int'l 5000 ft (1500 m)	
Transmit Power	1.25 mW (+1 dBm) / 2 mW (+3 dBm) boost mode	63 mW (+18 dBm) / Int'l 10 mW (+10 dBm)	
Receiver Sensitivity (1% PER)	-96 dBm in boost mode	-102 dBm	
Features			
Adjustable Power	Yes		
I/O Interface	3.3V CMOS UART, ADC, DIO		3.3V CMOS UART, SPI, I2C, PWM, DIO, ADC
Configuration Method	API or AT commands, local or over-the-air		
Frequency Band	2.4 GHz		
Interference Immunity	DSSS (Direct Sequence Spread Spectrum)		
Serial Data Rate	1200 bps - 1 Mbps		
ADC Inputs	(4) 10-bit ADC inputs		
Digital I/O	10		
Antenna Options	Chip, Wire Whip, U.FL, RPSMA	PCB Embedded Antenna, Wire Whip, U.FL, RPSMA	
Operating Temperature	-40° C to +85° C, 0-95% humidity non-condensing		
Programmability			
Memory	N/A		32 KB Flash / 2 KB RAM
CPU/Clock Speed	N/A		HCS08 / Up to 50.33 MHz
Networking & Security			
Encryption	128-bit AES		
Reliable Packet Delivery	Retries/Acknowledgments		
IDs and Channels	PAN ID, 64-bit IEEE MAC, 16 channels	PAN ID, 64-bit IEEE MAC, 15 channels	
Power Requirements			
Supply Voltage	2.1 - 3.6VDC	2.7 - 3.6VDC	
Transmit Current	35 mA / 45 mA boost mode @ 3.3VDC	205 mA	220 mA
Receive Current	38 mA / 40 mA boost mode @ 3.3VDC	47 mA	62 mA
Power-Down Current	<1 uA @ 25° C	3.5 uA @ 25° C	4 uA @ 25° C
Regulatory Approvals			
FCC, IC (North America)	Yes		
ETSI (Europe)	Yes		
C-TICK (Australia)	Yes		
TELEC (Japan)	Yes	Yes (Int'l unit only)	







Visit www.digi.com for part numbers.

DIGI SERVICE AND SUPPORT - You can purchase with confidence knowing that Digi is here to support you with expert technical support and a one-year warranty. www.digi.com/support

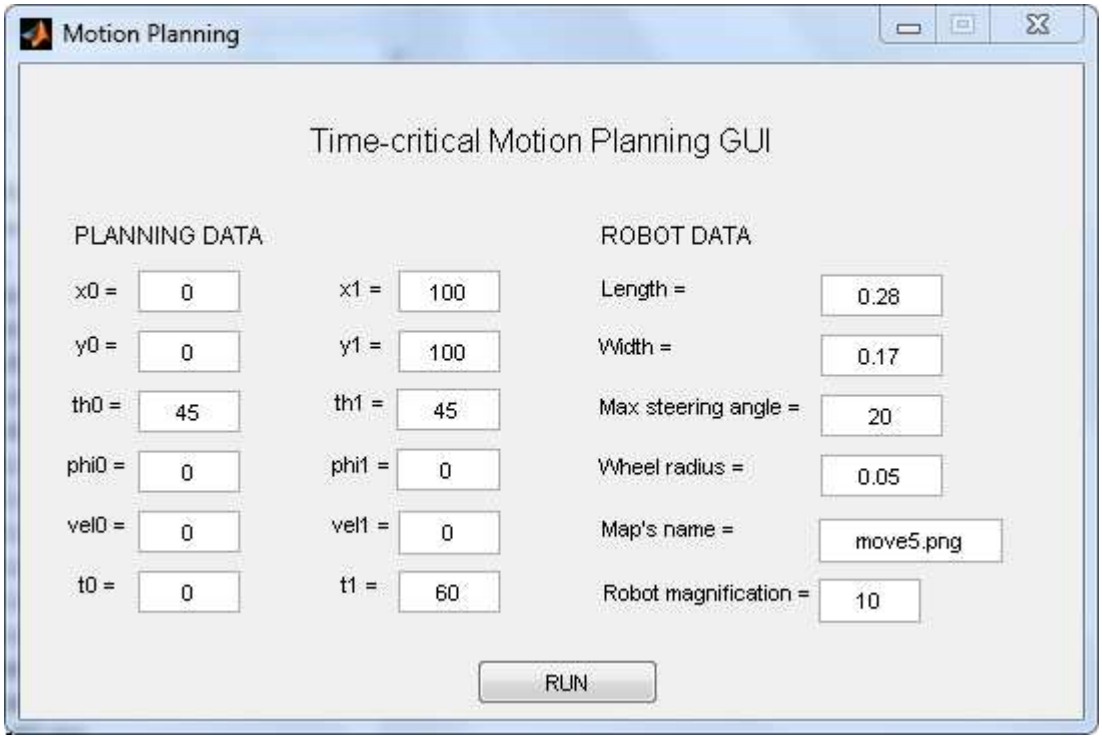
Digi International	Digi International France	Digi International KK	Digi International (HK) Limited	Digi m2m Solutions India Pvt. Ltd
877-912-3444	+33-1-55-61-98-98	+81-3-5428-0261	+852-2833-1008	+91-80-4287-9887
952-912-3444	www.digi.fr	www.digi-intl.co.jp	www.digi.cn	info@digi.com
info@digi.com				

e1aa1471
Da/511

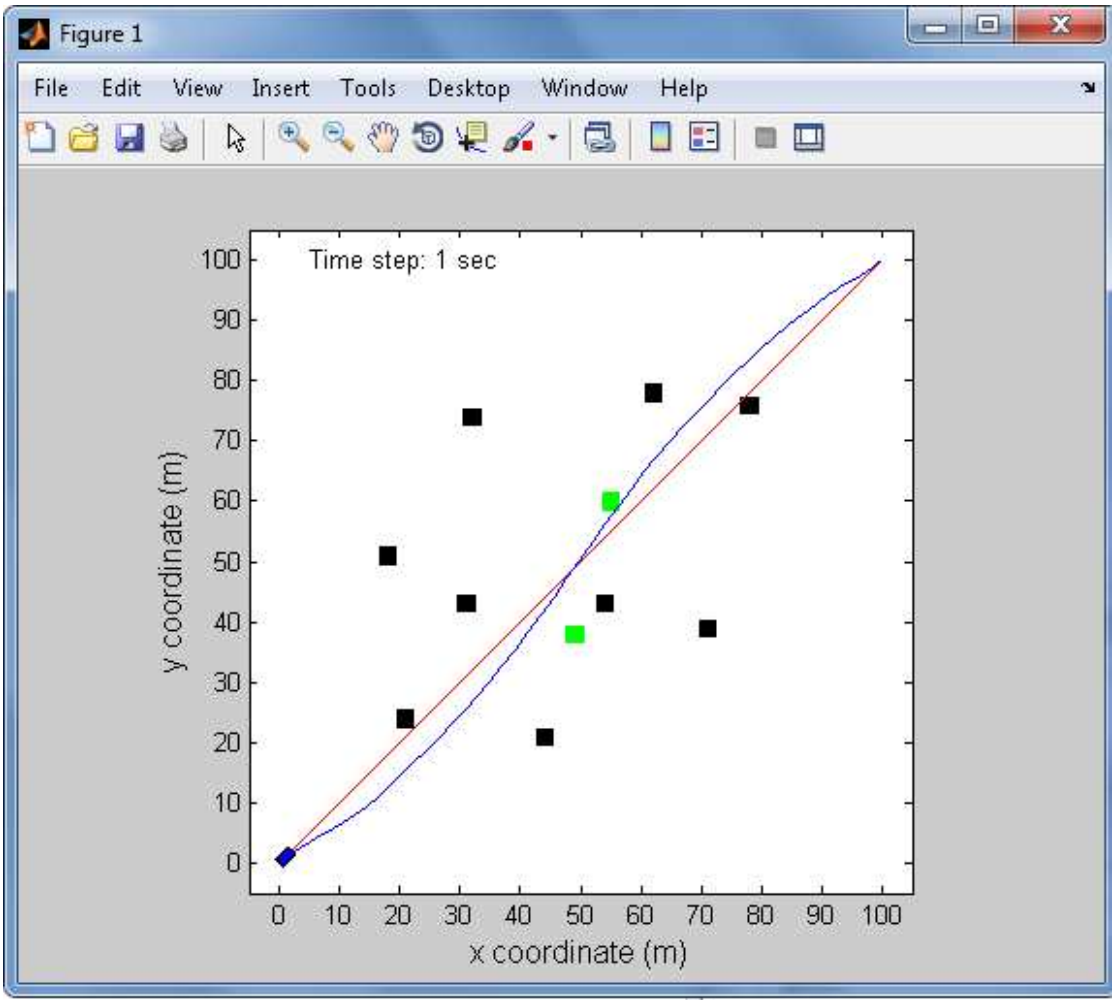
BUY ONLINE • www.digi.com

APPENDIX B

MATLAB GRAPICAL USER INTERFACE (GUI) AND PROGRAMMING



Input window's interface



Output window's interface

```
function mainstart()

clear all; clc;
addpath('Fig','Figures','function')
%%%%%%%% CONDITIONS %%%%%%%%%%
% a) time for robot 1 >= robot 2
% b) time for movobs 1 >= movobs 2
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


num_of_robs = 1; % number of robots
num_of_movobs = 0; % number of moving obstacles
mapmax = max(str2num(getappdata(0, 'xf')),str2num(getappdata(0,
'yf')));
xmapact=mapmax;
ymapact=mapmax;
% time=60; % overall time

%%%%%%%%%%%%
% SET MAP
%%%%%%%%%%%%
figure(1)
map=imread(getappdata(0, 'mapname')); %move4
image(map)
axis image on
colormap gray
axis ij
axis equal
axis tight

if mapmax<200
    mapinc = 10;
else
    mapinc = 50;
end

xlabel('x coordinate (m)','fontsize',12)
ylabel('y coordinate (m)','fontsize',12)
xlim([-10 (mapmax+50)])
ylim([-10 (mapmax+50)])
set(gca,'YDir','normal','XTick',0:mapinc:mapmax,'YTick',0:mapinc:mapm
ax)
hold on

% xlim([0 200])
% ylim([0 200])
% set(gca,'YDir','normal','XTick',0:20:200,'YTick',0:20:180)
% hold on

% set(gcf,'PaperPositionMode','auto');
% print(gcf,'-dtiff','-r0','newmap.tif')

%%%%%%%%%%%%%%%%%%%%%%%
% MOVING OBSTACLE PATH
%%%%%%%%%%%%%%%%%%%%%%%
if num_of_movobs>0
    for i=1:num_of_movobs
        run(['movobs' num2str(i)]);
```

```
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GENERATE INITIAL TRAJECTORY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:num_of_robs
    run(['robot' num2str(i)]);

    if num_of_movobs>0
        for j=1:num_of_movobs
            filename=['xmovobs' num2str(j) '.dat'];
            xmovobs_old=csvread(filename);
            filename=['xmovobs' num2str(j) 'R' num2str(i) '.dat'];
            csvwrite(filename,xmovobs_old)

            filename=['ymovobs' num2str(j) '.dat'];
            ymovobs_old=csvread(filename);
            filename=['ymovobs' num2str(j) 'R' num2str(i) '.dat'];
            csvwrite(filename,ymovobs_old)
        end
    end
end


%%%%%%%%%%%%%%%%%%%%%%
% SIMULATE ROBOTS
%%%%%%%%%%%%%%%%%%%%%%

length= str2num(getappdata(0, 'length')).*str2num(getappdata(0,
'magnify'));%1.3; %0.28; % robot length
width=str2num(getappdata(0, 'width')).*str2num(getappdata(0,
'magnify'));%0.6; %0.2; % robot width
phimax=str2num(getappdata(0, 'phimax'))*pi/180; %20*pi/180;
obs_size=10;
movobs_size=0.5;
radsm=10;
radmov=10;
% n=time; % maximum time

%%%%%%%%% CHECK ROBOT TIME %%%%%%%%%%%%%%%
%%%% 1 robot
if num_of_robs==1
    dataR1 = csvread('offlineR1.dat'); % read offline data Robot1
    [mR1 nR1] = size(dataR1);
    tRmax = nR1;
    csvwrite('tR1.dat',nR1)
end

%%%%% 2 robots
if num_of_robs==2

dataR1 = csvread('offlineR1.dat'); % read offline data Robot1
[mR1 nR1] = size(dataR1);

dataR2 = csvread('offlineR2.dat'); % read offline data Robot2
[mR2 nR2] = size(dataR2);
```

```
tRmax = max(nR1,nR2); % check max time

csvwrite('tR1.dat',nR1)
csvwrite('tR2.dat',nR2)
end

%%%% 3 robots
if num_of_robs==3

dataR1 = csvread('offlineR1.dat'); % read offline data Robot1
[mR1 nR1] = size(dataR1);

dataR2 = csvread('offlineR2.dat'); % read offline data Robot2
[mR2 nR2] = size(dataR2);

dataR3 = csvread('offlineR3.dat'); % read offline data Robot2
[mR3 nR3] = size(dataR3);

time=[nR1,nR2,nR3];
tRmax = max(time); % check max time

csvwrite('tR1.dat',nR1)
csvwrite('tR2.dat',nR2)
csvwrite('tR3.dat',nR3)
end

%%%% MOVING OBSTACLES %%%%
%%%% 1 moving obstacle
if num_of_movobs==1
    dataMov1 = csvread('movobs1.dat'); % read moving obstacle 1 data
    [mV1 nV1] = size(dataMov1);
    tVmax=nV1;
    csvwrite('tV1.dat',nV1)
end

%%%% 2 moving obstacles
if num_of_movobs==2
    dataMov1 = csvread('movobs1.dat'); % read moving obstacle 1 data
    [mV1 nV1] = size(dataMov1);
    dataMov2 = csvread('movobs2.dat'); % read moving obstacle 2 data
    [mV2 nV2] = size(dataMov2);

    tVmax = max(nV1,nV2); % check max time

    csvwrite('tV1.dat',nV1)
    csvwrite('tV2.dat',nV2)
end

ht=text(5,100,'Time step: ');
set(ht,'string','Time step: 0 sec','Color',[0 0 0])
% ht=text(147,193,'Time: ');
% set(ht,'string','Time: 0 sec','Color',[0 0 0])

% save figure
set(gcf,'PaperPositionMode','auto');
foldername='Figures';
filename='time0.tif';
```

```
print(gcf,'-dtiff','-r0',[foldername,filesep,filename])
foldername='Fig';
filename='time0.fig';
saveas(gcf,[foldername,filesep,filename])

olddata1=csvread('offlineR1.dat');
csvwrite('olddata1.dat',olddata1)

olddata2=csvread('offlineR2.dat');
csvwrite('olddata2.dat',olddata2)

olddata3=csvread('offlineR3.dat');
csvwrite('olddata3.dat',olddata3)

pause(3)
delete(ht)

for i=2:tRmax

    %%%%%% moving obstacles %%%%%%%%%%
      if num_of_movobs>0
        for k=1:num_of_movobs

            filename=['tV' num2str(k) '.dat'];
            tV=csvread(filename);

            if i>tV
                break;
            end

            filename = ['movobs' num2str(k) '.dat'];
            movobs_data = csvread(filename);

            x_movobs=movobs_data(2,i);
            y_movobs=movobs_data(3,i);
            p = linspace(0,2*pi,100);
            x_mov = x_movobs + 0.5*sin(p)';
            y_mov = y_movobs + 0.5*cos(p)';
            phmov = fill(x_mov,y_mov,'r');
            movobsno=k;
        end
      end

    %%%%%% mobile robots %%%%%%%
    for j=1:num_of_robs

        filename=['tR' num2str(j) '.dat'];
        tfR=csvread(filename);

        if i>tfR
            break;
        end

        % read data
        filename = ['offlineR' num2str(j) '.dat'];
        data1 = csvread(filename);
        filename = ['timeR' num2str(j) '.dat'];
        tdata = csvread(filename);
```

```
[tm tn]=size(tdata);

if tn==1
    data=data1;
    tR=tdata(1,1)+1;
end

if tn==2
    t1=tdata(1,1);
    t2=tdata(1,2);

    if i-1<t1
       data=data1(:,1:t1+1);
       tR=t1+1;
    end

    if i-1>t1
       data=data1(:,1:t2+1);
       tR=2+1;
    end
end

if tn==3
    t1=tdata(1,1);
    t2=tdata(1,2);
    t3=tdata(1,3);

    if i-1<=t1
       data=data1(:,1:t1+1);
       tR=t1+1;
    end
    if i-1>t1 && i-1<=t2
       data=data1(:,1:t2+1);
       tR=t2+1;
    end
    if i-1>t2
       data=data1(:,1:t3+1);
       tR=t3+1;
    end
end

  if tn==5
    t1=tdata(1,1);
    t2=tdata(1,2);
    t3=tdata(1,3);
    t4=tdata(1,4);
    t5=tdata(1,5);

    if i-1<=t1
       data=data1(:,1:t1+1);
       tR=t1+1;
    end
    if i-1>t1 && i-1<=t2
       data=data1(:,1:t2+1);
       tR=t2+1;
    end
    if i-1>t2 && i-1<=t3
       data=data1(:,1:t3+1);
       tR=t3+1;
    end
```

```
            if i-1>t3 && i-1<=t4
               data=data1(:,1:t4+1);
               tR=t4+1;
            end
            if i-1>t4
               data=data1(:,1:t5+1);
               tR=t5+1;
            end
        end


        filename = ['tempdataR' num2str(j) '.dat'];
        actdata = csvread(filename);
        robno=j;

        t=data(1,i);
        x=actdata(2,i-1);
        y=actdata(3,i-1);
        xth=data(2,i);
        yth=data(3,i);
        theta=actdata(4,i-1)*pi/180;
        thetas=data(4,1)*pi/180;
        phi=data(5,i)*pi/180;
        vel=data(6,i);
        ti=data(1,i)-data(1,i-1);

        % calculate new robot steering angle

        thetanext=atan((yth-y)./(xth-x));

        if (thetanext<=0 && x>xth) || (x>xth && y>yth)
            thetanext=pi+thetanext;
        end

        if thetanext<0 && x<xth && thetas<0
            thetanext=2*pi+thetanext;
        end

        dtheta=-(thetanext-theta);
        dvel=data(6,i)-data(6,i-1);
        d=data(6,i-1).*ti+0.5*dvel*ti;
        rb=d./(2*sin(dtheta/2));
        phi=atan(length/rb);


        if phi>phimax || phi<-phimax
            phi=phi/abs(phi)*phimax;
        end

        %%%%%%%%% calculate new robot data %%%%%%%%%

%         drb=data(6,i-1).*ti+0.5*dvel.*ti
        num_dig = 5;
        phi = round(phi*(10^num_dig))/(10^num_dig);

        if phi==0
           phi=0.0001;
        end
```

```
        rb=length/tan(phi);
        beta=2*(asin(d/(2*rb)));
        alpha=theta-(beta/2);

        xold=x;
        yold=y;

        theta=theta-beta;
        dx=d*cos(alpha);
        dy=d*sin(alpha);
        x=x+dx; % new x position
        y=y+dy; % new y position

        dxsen=length*cos(theta);
        dysen=length*sin(theta);

        xsen=x+dxsen; % initial x for sensor
        ysen=y+dysen; % initial y for sensor
%         plot(xsen,ysen,'og')

        drawrobot(xold,yold,t,x,y,theta,phi,vel,length,width,robno)

     %%%%%%%%%% static obstacles detection %%%%%%%%%%%%%%%%

     % to check whether the robot needs to avoid static obstacle or
not

        xmap=[data(2,tR), xmapact];
        ymap=[data(3,tR), ymapact];
        dist=rangefinder(xmap,ymap,xsen,ysen,theta,map);
        [o p]=size(dist);
        check_dist=min(dist(1,1:p));

    % if check_dist<=15, avoid
        tempdist=[];
        tempangle=[];
        if check_dist<=80%15
           for a=1:p
               distance=dist(1,a);
               angle=dist(2,a);
               if distance<=80%15
                  distance1=distance;
                  angle1=angle;

                  tempdist=horzcat(tempdist,distance1);
                  tempangle=horzcat(tempangle,angle1);
               end
           end

            dist=[tempdist; tempangle];
            min_dist=min(tempdist);
            angle_detect1=min(tempangle);
            angle_detect2=max(tempangle);
            angle_detect_obs=(angle_detect2+angle_detect1)/2;

            dscan_angle1=abs(theta-angle_detect1);
            dscan_angle2=abs(theta-angle_detect2);
```

```
            if dscan_angle1 <=10*pi/180 || dscan_angle2 <=10*pi/180

                distance=min_dist;
                scan_angle=angle_detect_obs;

                xobs=xsen + distance*cos(scan_angle);
                yobs=ysen + distance*sin(scan_angle);

                plot(xobs,yobs,'xr')

%                 hold on

                filename=['xobsR' num2str(robno) '.dat'];
                xobs_old=csvread(filename);
                filename=['yobsR' num2str(robno) '.dat'];
                yobs_old=csvread(filename);

                dobs=sqrt((xobs-xobs_old).^2+(yobs-yobs_old).^2);

                filename=['xobsR' num2str(robno) '.dat'];
                csvwrite(filename,xobs)
                filename=['yobsR' num2str(robno) '.dat'];
                csvwrite(filename,yobs)

            if dobs >= 2*(obs_size)

                if scan_angle<theta
                    % Turn left

                    h=sqrt((radsm+width).^2+(distance+obs_size).^2);
                    gamma=scan_angle +
atan((radsm+width)./(distance+obs_size));
                    dxc=h.*cos(gamma);
                    dyc=h.*sin(gamma);

                    xc=xsen+dxc;
                    yc=ysen+dyc;
                    sta=0;

                    td=data(1,tR);
                    xd=data(2,tR);
                    yd=data(3,tR);
                    thetad=data(4,tR)*pi/180;
                    phid=data(5,tR)*pi/180;
                    vd=data(6,tR);


avoidpath=avoid(t,x,y,theta,phi,vel,xc,yc,sta,td,xd,yd,thetad,phid,vd
,ti,obs_size,length,width,map,phimax,robno);
                    newpath=[data1(:,1:t), avoidpath,
data1(:,tR+1:tfR)];
                    csvwrite(['offlineR' num2str(robno)
'.dat'],newpath)

                else
```

```
 % Turn right

                    h=sqrt((radsm+width).^2+(distance+obs_size).^2);
                    gamma=scan_angle -
atan((radsm+width)./(distance+obs_size));
                    dxc=h.*cos(gamma);
                    dyc=h.*sin(gamma);

                    xc=xsen+dxc;
                    yc=ysen+dyc;
                    sta=0;

                    td=data(1,tR);
                    xd=data(2,tR);
                    yd=data(3,tR);
                    thetad=data(4,tR)*pi/180;
                    phid=data(5,tR)*pi/180;
                    vd=data(6,tR);


avoidpath=avoid(t,x,y,theta,phi,vel,xc,yc,sta,td,xd,yd,thetad,phid,vd
,ti,obs_size,length,width,map,phimax,robno);
                    newpath=[data1(:,1:t), avoidpath,
data1(:,tR+1:tfR)];
                    csvwrite(['offlineR' num2str(robno)
'.dat'],newpath)

                end
            end
            end
        end


    %%%%%% moving obstacles detection %%%%%%
    if num_of_movobs>0

    for k=1:num_of_movobs

       filename=['tV' num2str(k) '.dat'];
       tV=csvread(filename);

       if i>tV
           break;
       end


        movobsno=k;

        filename = ['movobs' num2str(movobsno) '.dat'];
        movobs_data = csvread(filename);

        x_movobs=movobs_data(2,i);
        y_movobs=movobs_data(3,i);
        theta_movobs=movobs_data(4,i);

        filename = ['xmovobs' num2str(movobsno) 'R' num2str(robno)
'.dat'];
        xmovobs_old=csvread(filename);
```

```
         filename = ['ymovobs' num2str(movobsno) 'R' num2str(robno)
'.dat'];
         ymovobs_old=csvread(filename);

         xmovdir=round(xmovobs_old - x_movobs);
         ymovdir=round(ymovobs_old - y_movobs);
         robdir=yold-y;

        checkmovdata =
checkmov(movobsno,x_movobs,y_movobs,xsen,ysen,theta);
        distmov = checkmovdata(1,1);
        scanmov = checkmovdata(2,1);

        dobs=sqrt((x_movobs-xmovobs_old).^2+(y_movobs-
ymovobs_old).^2);

        if distmov<15 && dobs>radmov

            if robdir<0

                if xmovdir<0 && y_movobs>y && scanmov>=theta

                    xmov_old=movobs_data(2,i-1);
                    ymov_old=movobs_data(3,i-1);
                    dobspred=2*(sqrt((x_movobs-
xmov_old).^2+(y_movobs-ymov_old).^2));
                    xmovpredict=x_movobs+dobspred*cos(theta_movobs);
                    ymovpredict=y_movobs+dobspred*sin(theta_movobs);

                    xpred_old=actdata(2,i-2);
                    ypred_old=actdata(3,i-2);
                    drobpred=2*(sqrt((x-xpred_old).^2+(y-
ypred_old).^2));
                    xpredict=x+drobpred*cos(theta);
                    ypredict=y+drobpred*sin(theta);

                    lp=linspace(0,2*pi,100);
                    xv=xpredict+5*cos(lp)';
                    yv=ypredict+5*sin(lp)';
                    xv=[xv;xv(1)];
                    yv=[yv;yv(1)];
                    [mp np]=size(xv);

                    for jp=1:mp
                        in =
inpolygon(xmovpredict,ymovpredict,xv,yv);
                    end

%                      plot(xmovpredict,ymovpredict,'xr',xv,yv,'.y')

                else if xmovdir>0 && y_movobs>y && scanmov<=theta

                    xmov_old=movobs_data(2,i-1);
                    ymov_old=movobs_data(3,i-1);
                    dobspred=2*(sqrt((x_movobs-
xmov_old).^2+(y_movobs-ymov_old).^2));
                    xmovpredict=x_movobs-dobspred*cos(theta_movobs);
                    ymovpredict=y_movobs-dobspred*sin(theta_movobs);
```

```
                    xpred_old=actdata(2,i-2);
                    ypred_old=actdata(3,i-2);
                    drobpred=2*(sqrt((x-xpred_old).^2+(y-
ypred_old).^2));
                    xpredict=x+drobpred*cos(theta);
                    ypredict=y+drobpred*sin(theta);

                    lp=linspace(0,2*pi,100);
                    xv=xpredict+5*cos(lp)';
                    yv=ypredict+5*sin(lp)';
                    xv=[xv;xv(1)];
                    yv=[yv;yv(1)];
                    [mp np]=size(xv);

                    for jp=1:mp
                        in =
inpolygon(xmovpredict,ymovpredict,xv,yv);
                    end

%                      plot(xmovpredict,ymovpredict,'xr',xv,yv,'.y')

                    else
                        in=0;
                    end
                end

          %%%%%%%%% case 1 - rob from bottom & movobs from left
%%%%%%%%%%
             if xmovdir<0 && y_movobs>y && scanmov>=theta && in==1

%                  d_mov=sqrt((x_movobs-x).^2+(y_movobs-y).^2)
                d_mov=abs(distmov/sin(theta));
                ycent=y_movobs;
                xcent=xsen+d_mov*cos(abs(theta));
%                 plot(xcent,ycent,'og')

                dm=sqrt((x_movobs-xcent).^2+(y_movobs- ycent).^2);
                xc=xcent-dm/2; % movobs from left
                yc=y_movobs;
                sta=1;

%                 radmov=abs(xcent-x_movobs);
%                 xref=xcent; % movobs from left
%                 yref=ycent;

                td=data(1,tR);
                xd=data(2,tR);
                yd=data(3,tR);
                thetad=data(4,tR)*pi/180;
                phid=data(5,tR)*pi/180;
                vd=data(6,tR);

                filename = ['xmovobs' num2str(movobsno) 'R'
num2str(robno) '.dat'];
                csvwrite(filename,x_movobs)
                filename = ['ymovobs' num2str(movobsno) 'R'
num2str(robno) '.dat'];
                csvwrite(filename,y_movobs)
```

```
avoidpath=avoid(t,x,y,theta,phi,vel,xc,yc,sta,td,xd,yd,thetad,phid,vd
,ti,obs_size,length,width,map,phimax,robno);
                newpath=[data1(:,1:t), avoidpath, data1(:,tR+1:tfR)];
                csvwrite(['offlineR' num2str(robno) '.dat'],newpath)
            end

            %%%%%%%% case 2 - rob from bottom & movobs from right
%%%%%%%
            if xmovdir>0 && y_movobs>y && scanmov<=theta && in==1

%                 d_mov=sqrt((x_movobs-x).^2+(y_movobs-y).^2)
                d_mov=abs(distmov/sin(theta));
                ycent=y_movobs;
                xcent=xsen+d_mov*cos(abs(theta));
%                 plot(xcent,ycent,'og')

                dm=sqrt((x_movobs-xcent).^2+(y_movobs- ycent).^2);
                xc=xcent+dm/2; % movobs from right
                yc=y_movobs;
                sta=1;

%                 radmov=abs(xcent-x_movobs);
%                 xref=xcent; % movobs from right
%                 yref=ycent;

                td=data(1,tR);
                xd=data(2,tR);
                yd=data(3,tR);
                thetad=data(4,tR)*pi/180;
                phid=data(5,tR)*pi/180;
                vd=data(6,tR);

                filename = ['xmovobs' num2str(movobsno) 'R'
num2str(robno) '.dat'];
                csvwrite(filename,x_movobs)
                filename = ['ymovobs' num2str(movobsno) 'R'
num2str(robno) '.dat'];
                csvwrite(filename,y_movobs)


avoidpath=avoid(t,x,y,theta,phi,vel,xc,yc,sta,td,xd,yd,thetad,phid,vd
,ti,obs_size,length,width,map,phimax,robno);
                newpath=[data1(:,1:t), avoidpath, data1(:,tR+1:tfR)];
                csvwrite(['offlineR' num2str(robno) '.dat'],newpath)
            end

            if (ymovdir<0 && y_movobs>y) || (ymovdir>0 && y_movobs>y)

          %%%% case 3 - rob from bottom & movobs from top or bottom
%%%%%
               scan_angle_mov=abs(theta-scanmov);

               if scan_angle_mov<=11*pi/180
                 if scanmov<=theta
                    % Turn left

                    h=sqrt((2*movobs_size+width).^2+(distmov).^2);
```

```
                    gamma=scanmov +
atan((2*movobs_size+2*width)./(distmov+movobs_size));
                    dxc=h.*cos(gamma);
                    dyc=h.*sin(gamma);

                    xc=xsen+dxc;
                    yc=ysen+dyc;
                    sta=2;

                    td=data(1,tR);
                    xd=data(2,tR);
                    yd=data(3,tR);
                    thetad=data(4,tR)*pi/180;
                    phid=data(5,tR)*pi/180;
                    vd=data(6,tR);

                    filename = ['xmovobs' num2str(movobsno) 'R'
num2str(robno) '.dat'];
                    csvwrite(filename,x_movobs)
                    filename = ['ymovobs' num2str(movobsno) 'R'
num2str(robno) '.dat'];
                    csvwrite(filename,y_movobs)


avoidpath=avoid(t,x,y,theta,phi,vel,xc,yc,sta,td,xd,yd,thetad,phid,vd
,ti,obs_size,length,width,map,phimax,robno);
                    newpath=[data1(:,1:t), avoidpath,
data1(:,tR+1:tfR)];
                    csvwrite(['offlineR' num2str(robno)
'.dat'],newpath)

                else

                    % Turn right

                    h=sqrt((2*movobs_size+witdh).^2+(distmov).^2);
                    gamma=scanmov -
atan((2*movobs_size+2*width)./(distmov+movobs_size));
                    dxc=h.*cos(gamma);
                    dyc=h.*sin(gamma);

                    xc=xsen+dxc;
                    yc=ysen+dyc;
                    sta=2;

                    td=data(1,tR);
                    xd=data(2,tR);
                    yd=data(3,tR);
                    thetad=data(4,tR)*pi/180;
                    phid=data(5,tR)*pi/180;
                    vd=data(6,tR);

                    filename = ['xmovobs' num2str(movobsno) 'R'
num2str(robno) '.dat'];
                    csvwrite(filename,xmovobs)
                    filename = ['ymovobs' num2str(movobsno) 'R'
num2str(robno) '.dat'];
                    csvwrite(filename,ymovobs)
```

```
avoidpath=avoid(t,x,y,theta,phi,vel,xc,yc,sta,td,xd,yd,thetad,phid,vd
,ti,obs_size,length,width,map,phimax,robno);
                    newpath=[data1(:,1:t), avoidpath,
data1(:,tR+1:tfR)];
                    csvwrite(['offlineR' num2str(robno)
'.dat'],newpath)

                  end
                end
              end
              end
          end
      end
      end
        %%%%%%%%% end of moving obstacle detection %%%%%%%%%%%%%

    end
   ht=text(5,100,'Time step: ');
   set(ht,'string',['Time step: ',num2str(t),' sec'],'Color',[0 0 0])
%    ht=text(147,193,'Time: ');
%    set(ht,'string',['Time: ',num2str(t), 'sec'],'Color',[0 0 0])

   % save figure
   set(gcf,'PaperPositionMode','auto');
   foldername='Figures';
   filename=['time' num2str(t) '.tif'];
   print(gcf,'-dtiff','-r0',[foldername,filesep,filename])
   foldername='Fig';
   filename=['time' num2str(t) '.fig'];
   saveas(gcf,[foldername,filesep,filename])

   pause(1)
   delete(ht)
end
```