Behavioural Strategy for Indoor Mobile Robot Navigation in Dynamic Environments

by

Ahmad Alsaab

Thesis submitted in fulfillment of the requirements for the Degree of Doctor of Philosophy

School of Mechanical and Systems Engineering

Newcastle University, United Kingdom

2015

This is dedicated to my parents, wife and children

Abstract

Development of behavioural strategies for indoor mobile navigation has become a challenging and practical issue in a cluttered indoor environment, such as a hospital or factory, where there are many static and moving objects, including humans and other robots, all of which trying to complete their own specific tasks; some objects may be moving in a similar direction to the robot, whereas others may be moving in the opposite direction. The key requirement for any mobile robot is to avoid colliding with any object which may prevent it from reaching its goal, or as a consequence bring harm to any individual within its workspace. This challenge is further complicated by unobserved objects suddenly appearing in the robots path, particularly when the robot crosses a corridor or an open doorway. Therefore the mobile robot must be able to anticipate such scenarios and manoeuvre quickly to avoid collisions.

In this project, a hybrid control architecture has been designed to navigate within dynamic environments. The control system includes three levels namely: deliberative, intermediate and reactive, which work together to achieve short, fast and safe navigation. The deliberative level creates a short and safe path from the current position of the mobile robot to its goal using the wavefront algorithm, estimates the current location of the mobile robot, and extracts the region from which unobserved objects may appear. The intermediate level links the deliberative level and the reactive level, that includes several behaviours for implementing the global path in such a way to avoid any collision.

In avoiding dynamic obstacles, the controller has to identify and extract obstacles from the sensor data, estimate their speeds, and then regular its speed and direction to minimize the collision risk and maximize the speed to the goal. The velocity obstacle approach (VO) is considered an easy and simple method for avoiding dynamic obstacles, whilst the collision cone principle is used to detect the collision situation between two circular-shaped objects. However the VO approach has two challenges when applied in indoor environments. The first challenge is extraction of collision cones of non-circular objects from sensor data, in which applying fitting circle methods generally produces large and inaccurate collision cones especially for line-shaped obstacle such as walls. The second challenge is that the mobile robot cannot sometimes move to its goal because all its velocities to the goal are located within collision cones. In this project, a method has been demonstrated to extract the collision cones of circular and non-circular objects using a laser sensor, where the obstacle size and the collision time are considered to weigh the robot velocities. In addition the principle of the virtual obstacle was proposed to minimize the collision risk with unobserved moving obstacles. The simulation and experiments using the proposed control system on a Pioneer mobile robot showed that the mobile robot can successfully avoid static and dynamic obstacles. Furthermore the mobile robot was able to reach its target within an indoor environment without causing any collision or missing the target.

Acknowledgments

I must first express my gratitude towards my supervisor, Dr Robert Bicker, who supported and encouraged me during my PhD study. I appreciate his excellent comments to perfect my research. I have been lucky to have a supervisor who helped me to improve my research, analysis and writing skills. Also I would like to thank my second supervisor, Dr John Hedley, for his support.

I sincerely thank my colleagues Imam, Simon, Alaa, Paramin, Jaya and Erwin who helped in making my time in the PhD study more fun and interesting

I will forever be thankful to my parents, brothers and sisters for their support and encouragement to finish my PhD study. I am extremely grateful to Dr Abdulkader Joukhadar who helped me to continue my postgraduate studies in UK.

I would like to thank my wife for her patience and encouragement during the journey of the PhD study. I will never forget her support.

Above all, I owe it all to Almighty God for granting me the health and strength to undertake this research and enabling me to its completion.

Publications

1. A. Alsaab and R. Bicker “Moving Obstacle Avoidance in Indoor Environments,” International Journal of Scientific and Engineering Research, Volume 5, Issue 2, February-2014, pp.855-862.

2. A. Alsaab and R. Bicker “Improving the Velocity Obstacle Approach for Obstacle Avoidance in Indoor Environments” , 2014 UKACC 10th International Conference on Control (accepted).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation for the Research

The ability of mobile robots to move autonomously provides a significant scope for numerous applications including routine tasks such as office cleaning and delivery, and particularly access to dangerous areas that are unreachable by humans such as cleaning up hazardous waste sites in nuclear power stations and planetary exploration. The development of intelligent behavioural strategies for indoor mobile navigation in cluttered indoor environments such as hospitals or factories, has recently become a challenging practical issue, where many static and moving objects, including humans and other robots, all of which are trying to complete their own specific missions; some objects may be moving in a similar direction to the robot, whereas others may be moving in the opposite direction. The key requirement for any mobile robot is to avoid colliding with any object which may prevent it from reaching its goal, and to avoid causing harm to any individual or another object within its workspace. This challenge is further complicated by some objects suddenly appearing in the robot's path, particularly when it crosses a corridor or an open doorway. The motivation of this research is to develop a behavioural strategy for indoor mobile navigation which allows the robot to reach its goal and manoeuvre among humans and moving obstacles. In this project, the robot will have the ability to react to intelligent (human) and unintelligent (dumb) obstacles, and thus has the potential to be used in number of service applications.

1.2 Aim and Objectives

The aim of the project is to design and develop intelligent behavioural strategies for indoor mobile robot navigation in dynamic environments. To achieve this aim, the following objectives have been defined:

- To conduct an extensive literature review of existing behaviour-based mobile robot architectures.
- To develop an algorithm which provides the ability to recognize and discriminate static and dynamic obstacles.
- To develop a robust algorithm to predict the actions of dynamic obstacles.
- To develop a control system by which a robot can manoeuvre among dynamic obstacles while considering areas where unexpected objects may appear.
- To validate the indoor mobile robot navigation algorithm using Matlab and Player/Stage simulation.
- To implement and test the navigation algorithm using an existing robot in a cluttered environment such that it can demonstrate its ability to avoid dynamic obstacles.

1.3 Research Hypothesis

The hypothesis of this research is that it is feasible to use a behavioural strategy to produce an accurate and safe robot manoeuvre, and to minimize the collision risk with unobserved objects suddenly appearing in the robot's path in a workspace cluttered by humans and moving obstacles.

1.4 Contributions

A behavioural strategy algorithm has been developed for a mobile robot to navigate in indoor environments. The proposed behavioural strategy system includes deliberative and reactive models. The velocity obstacle approach (VO) is considered to be an easy and simple method to detect the probability of collision between two circular-shaped objects, using the collision cone principle. Many studies have implemented the VO algorithm using simulation, where they considered that obstacles have circular shapes and their velocities and positions are known [Yamamoto et al., 2001;Kluge, 2003; Kluge and Prassler, 2004;Zhiqiang et al., 2008; Fulgenzi et al., 2007;Shiller et al., 2010]. However, the VO approach has two limitations when applied in indoor environments. The first challenge is that in the real world, not all obstacles are circular, while the second is that the mobile robot sometimes cannot move towards its goal because all its velocities to the goal are located within the collision cones. Hence, in this project, a method is proposed to extract the collision cones of non-circular objects, where the obstacle size and collision time are considered in weighting velocities of the robot, and a virtual obstacle principle is proposed to avoid unobserved moving objects. Many simulations and experiments have been posted on the following website: https://www.facebook.com/pages/Robotics-Group/173949736063997

1.5 Thesis Outline

Chapter 1 introduces the project's contributions and its aim and objectives. An extensive literature review and the background of mobile robots is provided in Chapter 2, in which the advantages and disadvantages of many methods used in mobile robot applications are explained. Chapter 3 focuses on the deliberative model of the proposed control architecture system, where simulation tests are implemented to verify the global path planning and localization system. The reactive model is explained in the Chapter 4, in which the capability of navigation algorithm has been tested in avoiding both static and dynamic obstacles. Chapter 5 illustrates the human tracking algorithm, where human motion is modelled using Player/Stage, and various simulation tests are implemented. Chapter 6 focuses on the overall implementation and evaluation of the proposed navigation system. Finally the achievements of this study and recommendations for future work are provided in Chapter 7.

Chapter 2

Literature Review

A robust avoidance algorithm can produce safe navigation but it is not enough to achieve sufficient navigation. The locomotion method of the robot must be chosen depending on the nature of the robot workspace. A control architecture, localization and global path planning algorithms are required to organize the robot behaviours and to produce an optimal long-term path and robust estimation of the robot position. Furthermore, to interact with dynamic obstacles (including humans) the sensor data are clustered, and then a velocity estimation algorithm is applied to estimate the obstacle's actions. This chapter includes a shortline of the history of mobile robots and their locomotion methods, control architectures, localization, global path planning, velocity estimation, clustering sensor data and robot software.

2.1 History of Mobile Robotics

In 1948 and 1949, William Grey Walter of the Burden Neurological Institute at Bristol created the first electronic autonomous robots with complex behaviours named *Elmer* and *Elsie* (Figure 2.1). Walter's robots had simple mechanical structure,; they are front wheel drive tricycle-like robots covered by a clear plastic shell. However, these robots were capable of phototaxis which is ,“ *the ability of organisms to move directionally in response to a light source*“, therefore they could plan their path to find a recharging station [LeBouthillier, 2014].

Twenty years later saw the development of the first computer controlled mobile robot (Shakey) built at the Stanford Research Institute in 1969 (see Figure 2.2). Shakey used a camera, laser range finder and binary tactile sensors to recognize its surrounding environment, where radio and video links were utilized for transferring sensor data to a DEC PDP 10 Computer which controlled actions such as travelling from one location to another, turning light switches on and off, opening and closing doors, and pushing movable objects [Nilsson, 1984].

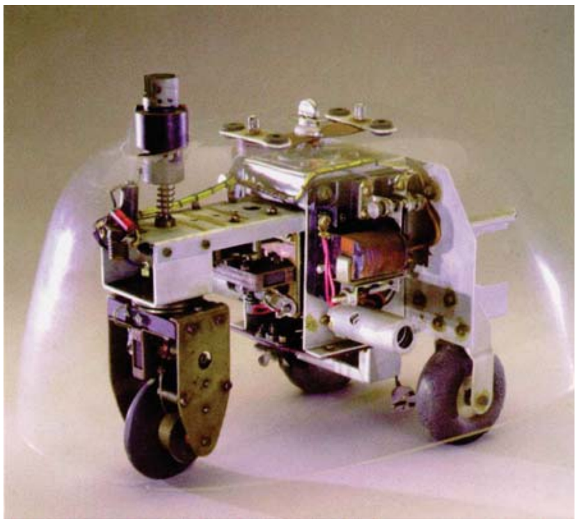


Figure 2.1: William Grey Walter robot[Online, 2105]

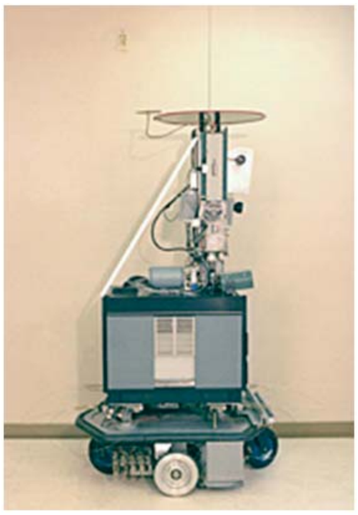


Figure 2.2: Shakey [Nilsson, 1984]

At Stanford in the late 1970s, Moravec developed the CART mobile robot, which used a camera to recognize the surrounding workspace and avoid statics obstacles [Moravec, 1983]. It was very slow because of the limited performance of the microcontrollers available in 1970s and its image processing algorithm took a long time to model a two-dimensional world map, where nine pictures of the same place are captured, and a series of steps are implemented such as digitizing the pictures, performing low level reduction, and planning a safe path. Subsequently, Moravec developed Rover, which comprised of TV camera, sonar and infrared sensors [Moravec, 1982].

HILARE is considered to be the first mobile project developed in Europe, at LAAS in Toulouse in the 1970s. It used sonar sensors, a camera and laser range finder [Laas, 2003]. However, HILARE was heavy (400kg), and also slow due to the slow speed of the processors. Although all early research focused on the software, the robot's motion was slow. However, rapid improvements in mobile hardware (sensors and actuators) and microprocessors have nowadays allowed the development of robust hardware/software architectures for mobile robots which can be used in various applications.

Nowadays, the Roomba mobile robot can be found in the market as a vacuum cleaning robot (see Figure 2.3). The task of this robot is to automatically clean floors [Layton, 2005], for which it uses four behaviours namely: obstacle avoidance, cliff avoidance, wall following and self-charging. A bumper sensor is used to avoid obstacles, where as soon as the robot knocks into something, it repeats a series of actions of backing up, rotating and moving forward until its path becomes clear. Cliff sensors are used to avoid steps; when the robot detects a cliff it turns and changes its direction. One infrared sensor is used for the wall following behaviour, by which the robot can move along walls without touching them. When the power of the batteries becomes low, the robot reaches the charging station by following the IR signal emitted by the charger.

However, the proposed navigation system for this project is suitable the Roomba robot, where its simple task does not require such a complex control system which requires adding more sensors(laser sensors) and a more powerful CPU. As a consequence, the price of the robot increases, while it has been designed to be affordable by most people.

TUG autonomous delivery robot is another commercial robot which was built by the Aethon Company to work in hospitals (see Figure 2.4). It has a hybrid control architecture, in which, the localization system depends on the AMCL localization system to estimate the actual position of the robot using a predefined map, odometry data and external sensor data [Aethon, 2007]. This robot does not have any long path planning algorithm, but instead users or the *implementation team* of the Aethon company create a map with all available routes including auto-opening doors, delivery points and charging stations. In the reactive level, the TUG uses external sensors (a scanning laser and 27 infrared and ultrasonic sensors) to detect and avoid obstacles.

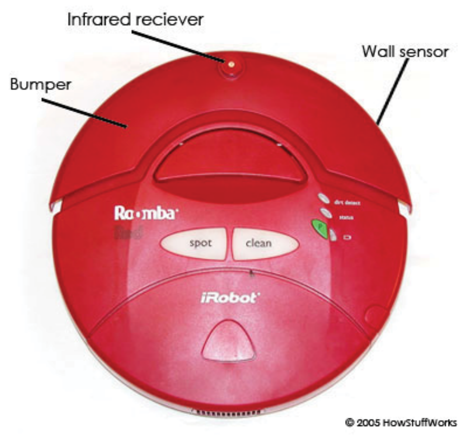


Figure 2.3: Roomba robot [Layton, 2005]

The proposed navigation system can improve the performance of the TUG robot. The robot will be able to automatically plan its long path and re-plan a new path if the optimal path is blocked. Furthermore, the proposed reactive model will enable the robot to track and avoid obstacles and humans in the hospital environment.

Figure 2.4: Tug robot [Aethon, 2007]

2.2 Mobile Robot Locomotion

Mobile robots can be classified into three general types with respect to their environments as follows (see Figure 2.5): Aerial, underwater, and ground robots. Aerial and underwater robots operate in a three-dimensional workspace. For a ground robot, wheeled locomotion is a simple mechanism which moves and turns by changing the velocities or directions of its wheels. There are four types of wheeled robots; namely, differential , synchronic, tricycle and car-type drives (see Figure 2.6).

The wheeled robot differential drive is suitable for navigation in flat ground indoor environments such as hotels, hospital and factories, and also it is considered easy to build and control, and can turn in the spot. However, it has nonholonomic constraints because there is a coupling between the robot speed and its direction [Salem, 2013]. The wheeled robot with synchronic drive is able to turn in the spot, move easily in rough terrain, but it has a complex mechanical structure because eight (or six) motors are required to power the wheels and control their steering. Consequently the control system is considered very complex [Wadai, 2000]. The wheeled tricycle robot has a simple mechanical structure and simple control system. However, it cannot turn on the spot and has a nonholonomic structure. The wheeled car-type robot has a simple control system, and moves easily in rough terrain. However, it has nonholonomic and complex mechanical structure.

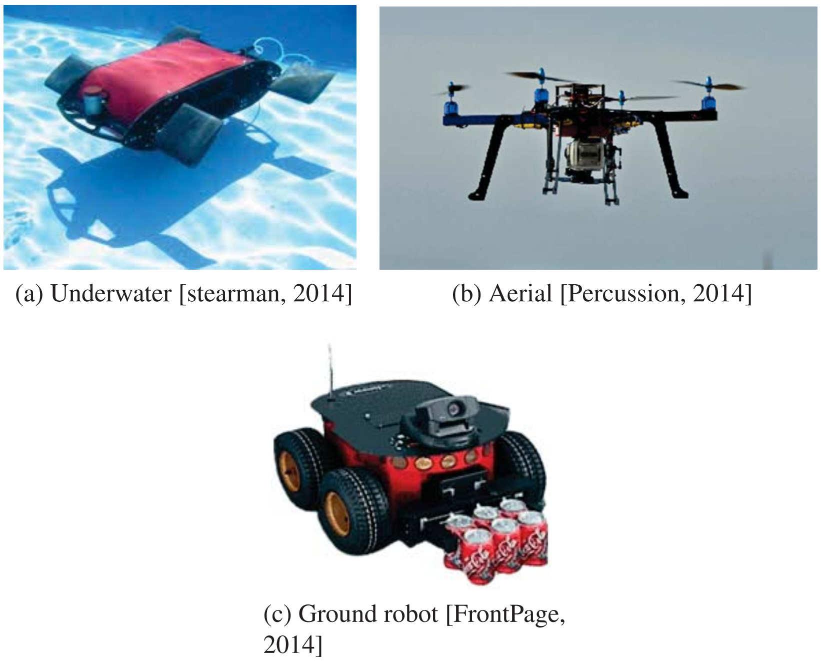

(a) Underwater [stearman, 2014]
(b) Aerial [Percussion, 2014]
(c) Ground robot [FrontPage, 2014]

Figure 2.5: Mobile robot locomotion

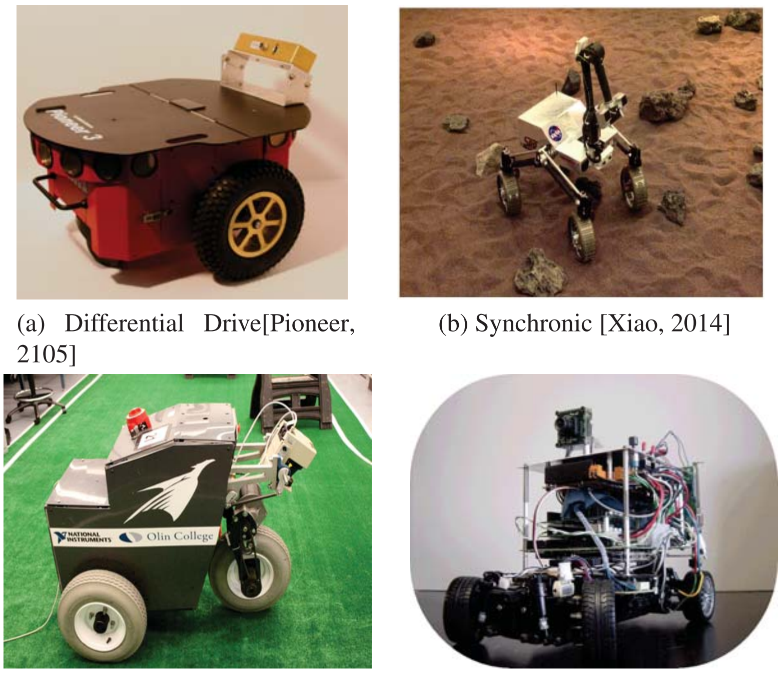

(a) Differential Drive[Pioneer, 2105]
(b) Synchronic [Xiao, 2014]
(c) Tricycle[Tricycle, 2105]
(d) Car[Car, 2105]

Figure 2.6: Wheeled robot

In this project, the mobile robots are required to navigate in flat ground environment, hence, a simple differential drive wheeled robot is chosen to implement the developed control system.

2.3 Mobile Robot Control Architectures

An autonomous mobile robot is an intelligent system which can navigate within its workspace without human interference. In this context, the robot must be able to perceive the surrounding environment through data collected from sensors, plan a trajectory which guarantees safe and accurate goal attainment, and then track this trajectory by generating suitable motion commands. The control architecture is defined as the ability of a mobile robot to develop and integrate its tasks in order to achieve successful autonomous navigation. Many studies have been conducted to develop autonomous mobile robot navigation, aiming to achieve a robust, flexible and reliable control architecture [Nakhaeinia et al., 2011; Droge et al., 2012].

The control architectures used can be classified in three main categories; namely, deliberative, reactive and hybrid architectures. The deliberative architecture plans an optimal trajectory depending on a global world model which is built from sensor data. The reactive architecture does not require the building of a global world model, but it depends on current sensor data to implement its actions, which makes it much quicker than the deliberative architecture. The hybrid architecture aims to combine advantages of the powerful planning of the deliberative architecture and the rapid response of the reactive architecture.

2.3.1 Deliberative Architecture

The deliberative architecture or top-down architecture repeats a series of steps: sense, plan and act (SPA) [Chatila and Laumond, 1985; Pruski et al., 2008]. The robot's tasks are decomposed into five serial modules; namely, perception, modelling, planning, execution and action (Figure 2.7). The perception model processes the collected sensor data, and proceeds to build or update the world model. The planning model plans an optimal trajectory to the goal, which is carried out by the execution and action models.

The SPA model has a powerful ability to plan and act, but fusing sensor data to build a world model is sometime difficult, and planning a serial sequence of actions may produce a slow response [Mataric, 2005]. Furthermore, if any task does not work properly, the whole system may fail. Hence, this method is suitable for a static environment, while it does not work well in complex environments which make the modelling of the robot map and planning a safe path more difficult.

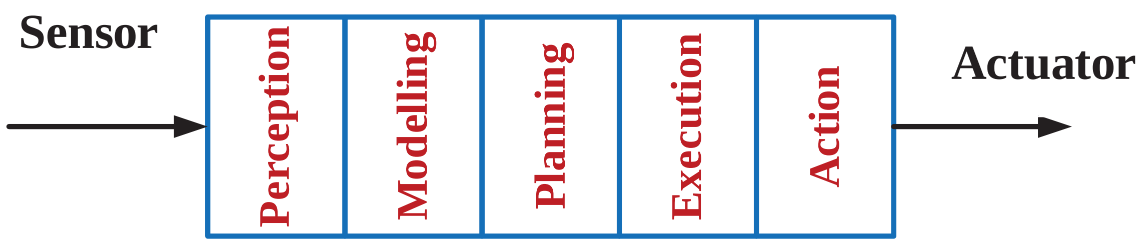


Figure 2.7: Deliberative architecture

2.3.2 Reactive Architecture

The reactive architecture decomposes the control system into multiple parallel tasks or behaviours which can access the sensor data and actuators directly, as shown in Figure 2.8. The reactive architecture can be classified into two basic types, namely: subsumption architecture and motor schema.

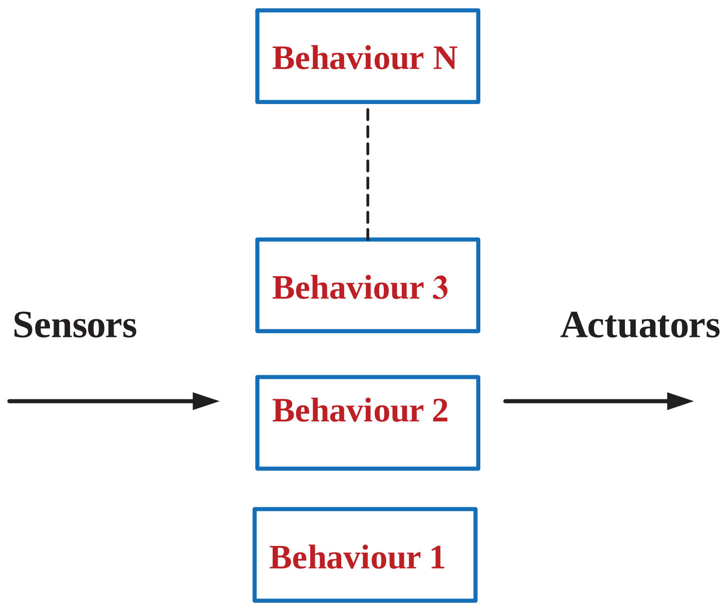


Figure 2.8: Reactive architecture

The subsumption architecture was first developed by Brooks [Brooks, 1986], where the control architecture includes multiple levels of competences which are not totally independent because the upper layers can control the lower layers by stopping their outputs or suppressing their inputs. Hence, the higher level must be able to implement the lower layer functions (see Figure 2.9). In the subsumption architecture, the complexity of a layer function increases as its level increases. Brooks defined eight levels for his control architecture, namely: avoid objects, wander, explore, build maps, monitor changes, identify objects, plan changes to world, and reason about behaviour of object [Toal et al., 1996]. Guzel and Bicker [2012] used a subsumption architecture with five behaviours, namely: goal seeking, approach, collision avoidance, completed and wander behaviours. The goal seeking behaviour is responsible for finding the robot's goal using a monocular camera and the SIFT algorithm. If the goal is founded, the approach behaviour is activated to generate optimal control commands to reach the goal; otherwise the robot moves randomly using the wander behaviour. If the distance between the robot and an obstacle is less than a minimum threshold, the obstacle avoidance behaviour is activated to avoid collision. When the robot gets very close to the target, the completed behaviour is activated.

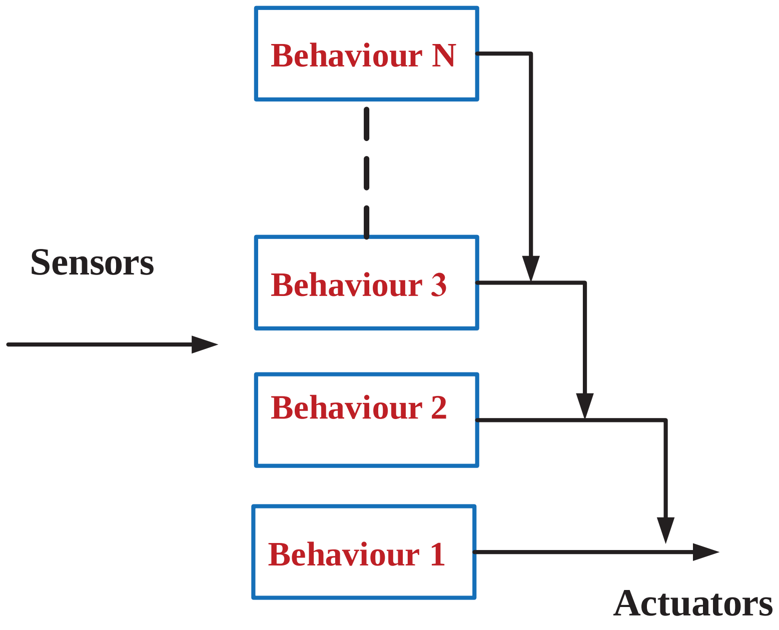


Figure 2.9: Brooks' Subsumption Architecture

The motor schema architecture was developed by Arkin [Arkin, 1987]. It consists of independent tasks or behaviours, each of which produces its output according to sensor data, and then those outputs are fused together to produce optimal motion commands. The behaviour outputs can be combined using arbitration or fusion methods [Yan et al., 2006]. In the arbitration method (Figure 2.10), each behaviour has a priority, and the output of the control system is assigned to the active output of the highest priority. Beom and Cho designed a selector unit to switch between goal-seeking and avoidance behaviours [Beom and Cho, 1995]. In the fusion method, the behaviours' outputs are weighted according to special values (w), and then summed to produce one output vector (see Figure 2.11). The potential field approach (PFA), which is explained later on in this chapter, uses the fusion method to combine between goal-seeking and obstacle avoidance behaviours [Khatib, 1985].

The reactive architecture has several advantages compared to the deliberative architecture:

- Its response is quicker in dynamic environments.
- There is no need for world modelling.
- It is more robust because the behaviour units work independently.

On the other hand it has the following disadvantages:

- Fusing the behaviour outputs becomes complex as the number of tasks or behaviours increases.
- It is not suitable if some tasks require planning.

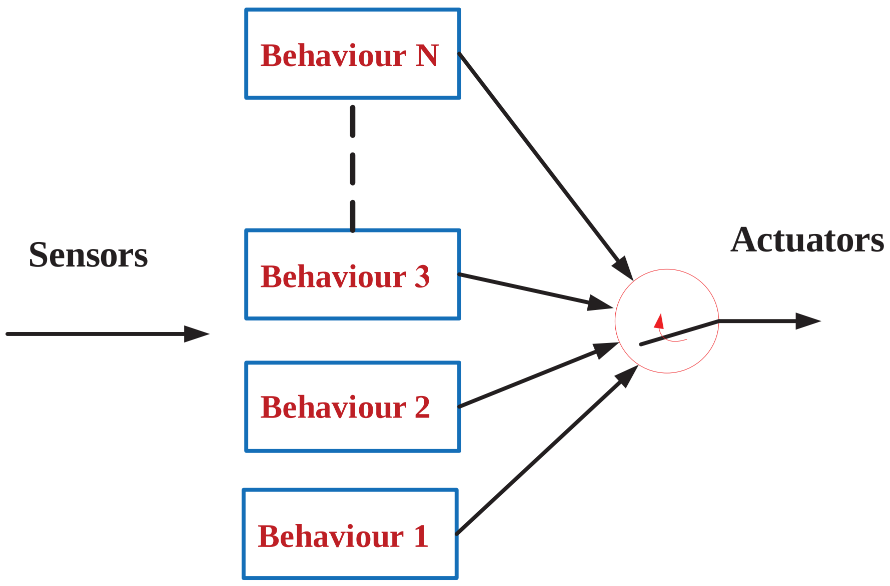


Figure 2.10: Arbitration method of the motor schemas

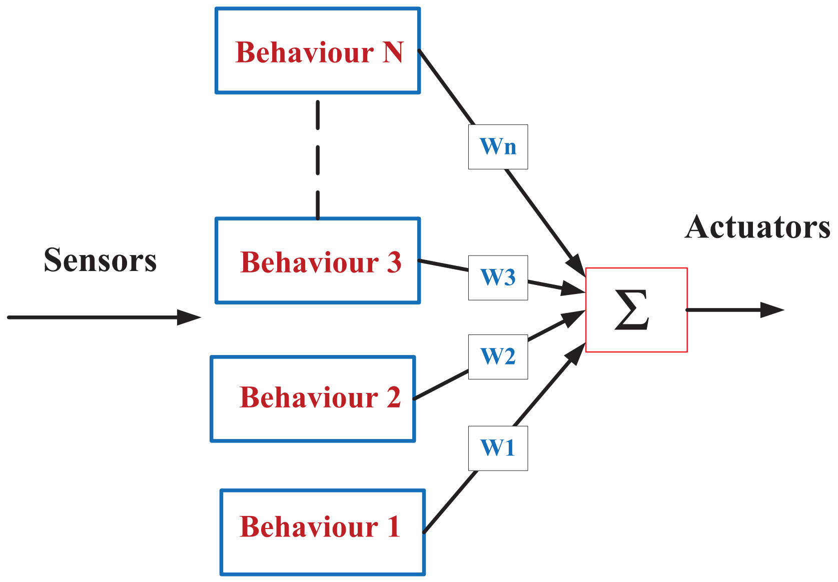


Figure 2.11: motor schema

2.3.3 Hybrid Architecture

The hybrid architecture overcomes the problem of the lack of planning in reactive architecture by combining the two previous architectures (Figure 2.12). The deliberative architecture forms the top level which is responsible for the long-term planning or global path, localization and human interaction. The intermediate layer (sequencer layer) links the top and lower levels, and transfers information from the top layer such as passing waypoints and observing the reactive layer to provide feedback to the top level. The lower level of the hybrid architecture is the reactive layer which works to implement the global path through fusing its behaviours. In other words it implements local path planning [Li and Jiang, 2007; Peng et al., 2009; Milford and Wyeth, 2010]. At Newcastle University, an intelligent hybrid architecture was developed for navigation in indoor environments, where the deliberative level includes the global path planning and localization system, while the reactive level comprises five behaviours, namely: go to goal, laser avoid, sonar avoid, wall follow and wander [Nattharith, 2010].

2.4 Localization

Many solutions have been proposed to solve the problem of estimating the mobile robot's location. Localization systems can be classified into three categories, namely: relative, absolute and hybrid localization systems. In the relative localization system, the mobile robot's location is calculated with respect to a known starting location by integrating the wheel speed measured by encoders. This technique is considered to be self-contained, simple, cheap and easy to implement in real time applications. However, its major disadvantage is the accumulation of error produced by wheel slippage. Several solutions have been proposed to minimize the integration error. De et al [2002] suggested to fix two additional encoders on the castor wheel; one is used to indicate the robot direction, and another measures its speed. The measurements collected from the encoders fixed on the driving wheels and two additional encoders were fused using a fuzzy logic algorithm. Using additional sensors such as the accelerometer, gyroscope and compass was proposed in [von der Hardt et al., 1996; Zhou and Loulin, 2011]. However, these solutions increase hardware cost and also are unable to define an accurate absolute position of the mobile robot because the robot's location is calculated with respect to the starting location.

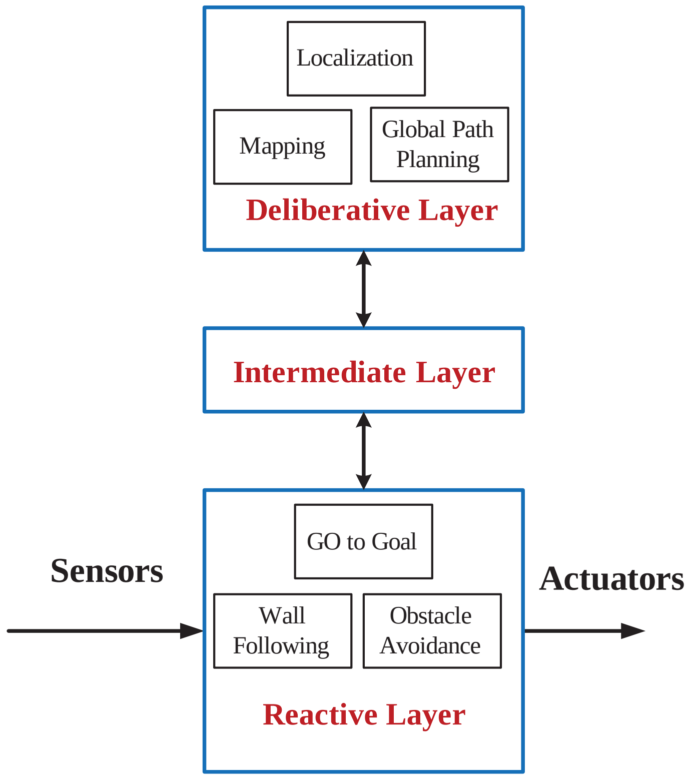


Figure 2.12: Hybrid architecture

Absolute localization systems are more complex and accurate than relative localization systems. The robot's location is estimated with respect to known reference points in its workspace, where different sensors can be employed such as GPS, vision and ultrasonic sensors. The GPS localization system is simple and does not need landmarks. However, it is only suitable for outdoor applications [Sakamoto et al., 2012]. Several researchers have developed an indoor global system (IGS) using RFID and sonar sensors for an indoor environment [Schmitt et al., 2010; Byoung-Suk and Ju-Jang, 2009; SeungKeun et al., 2008]. The IGS system utilizes many active beacons fixed in known positions, where the robot estimates its current position by measuring the arrival time (TOF) of ultrasonic signals from three beacons, but needs more beacons as the robot workspace area increases, which makes the localization system expensive and more complex. Vision based localization is currently an active research area, in which natural or artificial landmarks are extracted from the workspace to estimate the current position of a mobile robot [Heesung and Kyuseo, 2005; Chi et al., 2013]. However, there are many weaknesses in this type of localization system and it is unsuitable for fast moving navigation because the slow image processing algorithm, and illumination also effects the performance of the vision system.

Many systems have been proposed to combine relative and absolute localization systems to increase the accuracy of the estimation of the position of the mobile robot [Haoxiang et al., 2008; Bischoff et al., 2012]. The Adaptive Monte Carlo localization system (AMCL) is a good example of a hybrid localization system which uses odometry data acquired by dead-reckoning sensors and measurements collected by laser range finder, ultra sonic sensors or vision to estimate the robot location [Xudong et al., 2005]. However, AMCL requires the map of the robot workspace.

2.5 Global Path Planning

Global path planning is a key task of the deliberative layer, involving finding a safe and short trajectory from the robot's current location to its destination. Path planning methods can be classified as either topological or metric. In the topological method, the route of the robot is determined by a set of landmarks, while in the metric method the path is divided into many subgoals which are given as Cartesian coordinates. Global path planning requires two stages: world representation and path search.

2.5.1 World Representation

World representation is a procedure in which the robot's map is made suitable for applying a path search algorithm. The meadow maps method is an example of the world representation algorithm, which divides the robot's map into many convex polygons to establish safe paths for the mobile robot. Each polygon consists of lines which connect between interesting features in the map such as corners and doors [MURPHY, 2000]. However, in some cases, there is no intersection between some convex polygons as shown in Figure 2.13; and therefore, the robot may not find a continuous path between its current location and its target. To solve this problem, lines are used to connect the midpoints of the convex polygons as shown in Figure 2.14 [MURPHY, 2000].

There are two main disadvantages of the meadow method:

- It involves complex computation.
- It uses artificial landmarks.

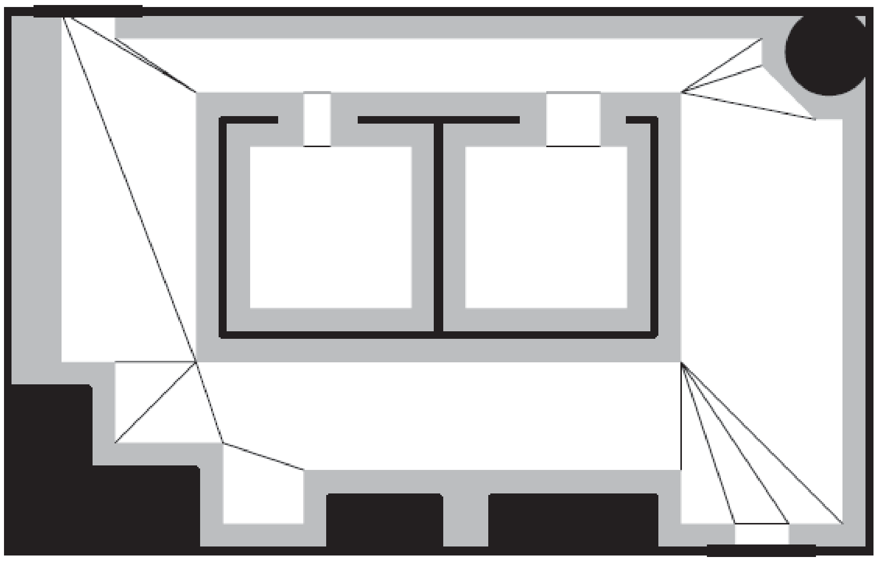


Figure 2.13: Convex polygons in the meadow map [MURPHY, 2000]

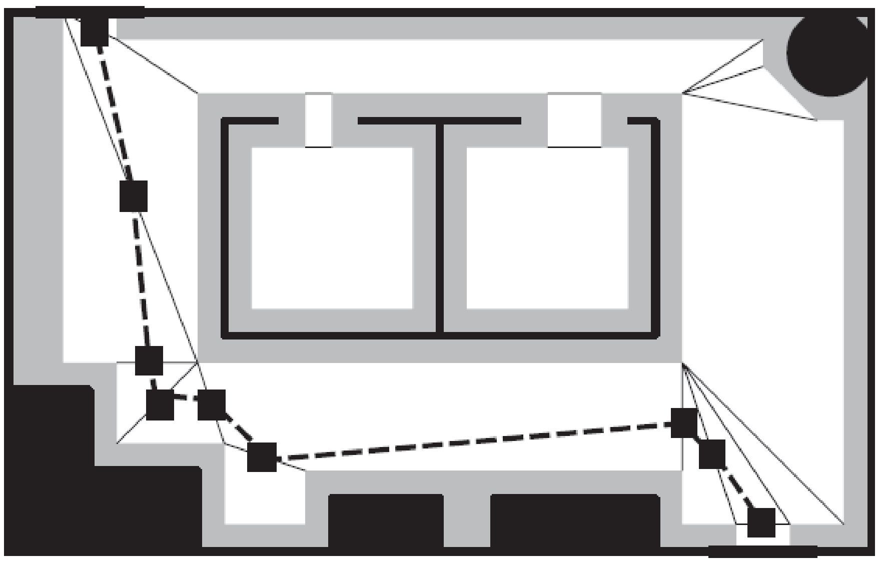


Figure 2.14: Convex polygons connections [MURPHY, 2000]

The GVDs method draws lines called Voronoi edges, in which each point is equidistant from the two nearest points on two different lines as shown in Figure 2.15. The Voronoi edges represent safe trajectories for the mobile robot which will always be in the middle area between the nearest objects [Huiying et al., 2010; Gomez et al., 2013].

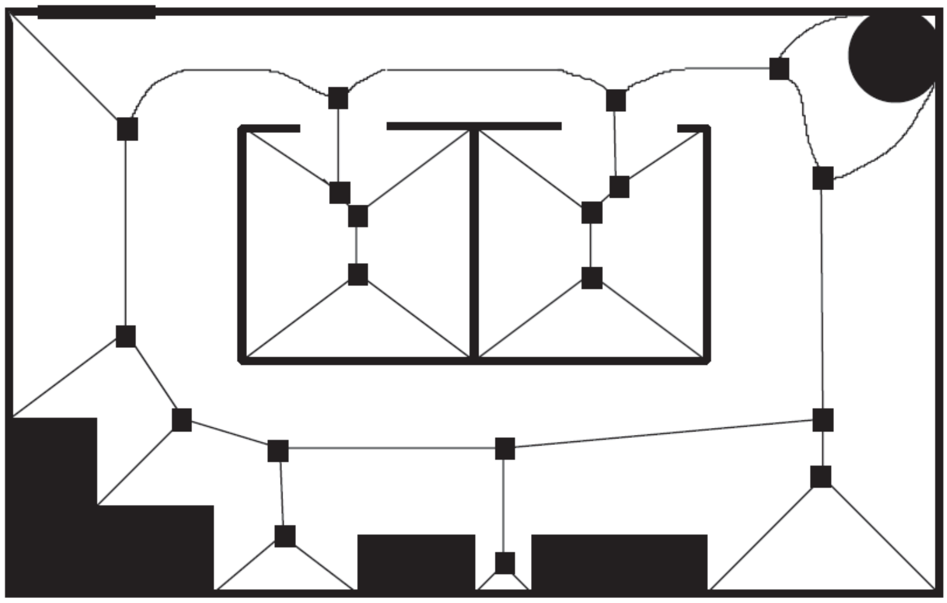


Figure 2.15: GVDs method[MURPHY, 2000]

The Occupancy Grids method describes the world map using a grid; and the centre of each square in this grid is considered as a node [Moravec and Elfes, 1985]. However, if any small part of a square is occupied then the whole square will be marked as occupied which wastes much free spaces for the path search algorithm. This problem can be resolved by reducing square size, but this requires a large memory for storing the large number of cells, and furthermore the path planner will require more time to find an optimal path. An alternative solution is the Guardtrees algorithm which divides the map into many large squares, and if any square is occupied by an obstacle it is divided into four squares [MURPHY, 2000].

2.5.2 Path Planning Algorithm

Path planning algorithm is applied to the graphic representation of the robot's workspace to find an optimal path between the current and target locations. The Dijkstra algorithm is one of the most adopted algorithms for global path planning [Jasika et al., 2012]. Here each node is assigned as unvisited, and the current location of the robot is given a value of 0. Then all its neighbouring nodes are weighted with the distance to the starting point. Next, the node of the lowest weight is selected as the current node, and all neighboring nodes are weighted with the distance to the current node plus the weight of the current node. This procedure is repeated until the goal is reached.

The A* search algorithm is an extension of the Dijkstra algorithm, and is used frequently for path planning for holonomic robots [Zeng and Church, 2009]. A cost function $f(n)$ defined by (2.1) is applied to find a sequence of nodes which achieve the shortest path to the goal.

$$f(n) = g(n) + h(n) \tag{2.1}$$

where $g(n)$ and $h(n)$ represent the distance between node n and the goal, and the distance between the starting point and node n respectively.

The A* algorithm can be applied to any graphic representation. However, the path planning becomes more difficult if additional factors such as the topology of the ground is considered.

The wavefront algorithm is another example of path planners. It is only suitable for applying to a grid representation of the workspace map [Hughes et al., 1992]. It uses the principle of heating radiation in a conductive material, where the initial position of the mobile robot is considered as a heating source. Thus, if there is any path to the goal, the heat will radiate to it; otherwise obstacles which are considered as having zero conductivity will isolate the heat from reaching the goal. In this method, the free areas are assigned large conductivities, while undesired areas are considered to be low conductivity. To find the global path, all obstacle cells are assigned to a fixed value of 1, while the goal is given a value of 2 which is increased to cover all free cells in the world map, and then the path starts from the current position of the robot and follows a sequence of nodes which decreases the numeric value.

2.6 Avoiding Obstacles

The obstacle avoidance behaviour is a key issue for executing successful navigation and protecting the mobile robot, other objects and humans from collisions. The global path planning considers just the fixed obstacles such as walls in the workspace map. However, many objects may change their locations under their own power or due to the actions of others, which is not considered during the global path planning procedure. Hence, the robot must detect and avoid such obstacles, which can be classified into two types, namely: static and dynamic.

2.6.1 Avoidance of Static Obstacles

Mobile robot navigation among static obstacles is simple because the robot has to calculate only the distances to the obstacles, can spend an uncertain amount of time in executing an optimal action, and can stop in an emergency. It can use the previously mentioned global path planning algorithms to avoid static obstacles using a deliberative control architecture, where the workspace map is updated using sensor data, and then the path is re-planned. Al-Jumaily and Leung [2005] utilized the wavefront algorithm for global path planning depending on a predefined map which is updated using a laser sensor, and then the planning algorithm is applied to define a safe path.

Several algorithms have been proposed to avoid static obstacles using a reactive control architecture. The Bug algorithm uses contact sensors to detect obstacles (see Figure 2.16). When the robot detects an object O1 it detours around the object until arriving at the intersection point with the straight line between the starting point S and the goal g, after which it follows the straight line. The Bug algorithm is simple, however, it has some weaknesses, e.g. it does not consider the robot's kinematic constraints, and the path is not optimized [Lumelsky and Skewis, 1990].

The Potential Field Algorithm (PFA) proposed by Khatib [Khatib, 1985] depends on the principle of the electric force produced between two charges, where an attractive force F_{att} is generated between two particles if they have different charges and a repulsive potential force F_r is generated if they have the same charge. The magnitude of the force is inversely proportional to the square of the distance between particles. The PFA treats a robot as a charge (for example, negative), obstacles have the same charge as the robot, and the goal has the opposite charge as mobile robot (positive). When the robot navigates within a workspace it will experience both repulsive forces generated by obstacles and an attractive force generated by the goal. The resultant force of the repulsive and attractive forces allows the robot to minimize the collision risk with obstacles and seek its goal. However, the disadvantages of the PFA algorithm concern the local minima and oscillatory behaviour in a narrow space.

The latter occurs when the repulsive and attractive forces cancel each other because of the symmetry of the environment, while local minima occurs in concave obstacles (see Figure 2.17). Chatila and Khatib [1995] extended the PFA algorithm by introducing two potentials: the *rotation* potential which considers the angle between the robot and obstacle to weight the repulsive force, and the *task* potential which filters out all obstacles which do not affect the robot's motion.

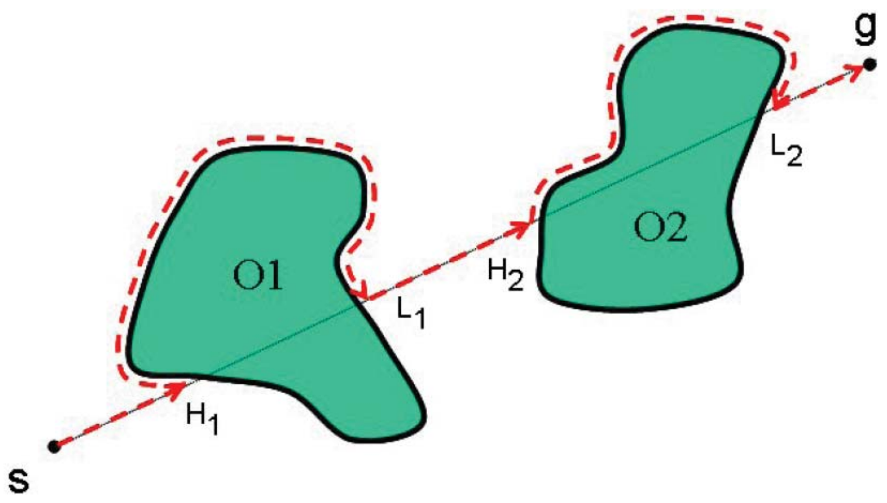
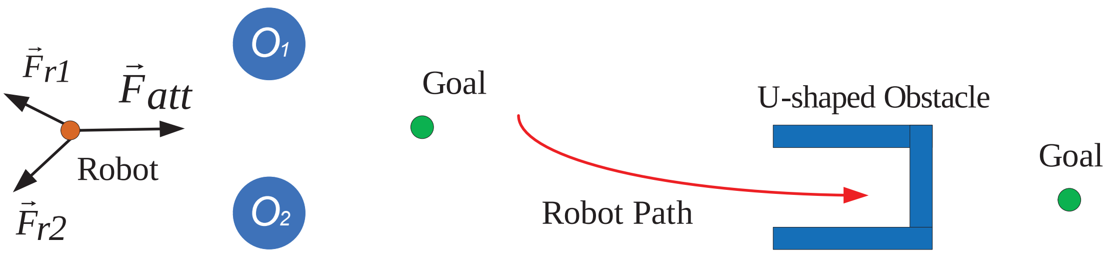


Figure 2.16: Bug method



(a) Oscillatory behaviour (b) local minima

Figure 2.17: Problems of the PFA approach

The Vector Field Histogram method (VFH) implements two-stage data-reduction processes to generate optimal control commands [Borenstein and Koren, 1991]. In the first stage, a 2D Cartesian histogram grid of the representation of the obstacle is reduced in a 1D polar histogram; the 2D Cartesian histogram is built using the certainty grid for obstacle representation algorithm, where the world map is divided into a grid, and each cell in this grid is assigned a certainty value (CV) which represents the probability of an obstacle existing in the cell [Elfes, 1987]. In the second stage, all sufficiently large open areas are found, and then a cost function is applied which considers the alignment of the robot's path to its goal, wheel direction, and previous direction. The sector with the lowest value is selected as an optimal direction for the mobile robot to achieve its goal successfully. However, the VFH method has the following limitations:

- The local minima might not be overcome
- Dynamic constraints are not considered
- Passing narrow areas might be problematic.

The VFH algorithm was subsequently extended to the VFH+ algorithm, in which two additional stages were introduced to take into consideration the dynamic constraints and to smooth the robot's path [Ulrich and Borenstein, 1998]. The VFH* algorithm combines the VFH+ and A* algorithms to avoid the robot becoming trapped in the local minima [Ulrich and Borenstein, 2000].

The Dynamic Window Approach (DWA) plans a short term path taking into consideration the kinematic constraints of the mobile robot, such as maximum velocity and acceleration. In the DWA method, a robot avoids obstacles by following circular trajectories whose radii are determined by pairs of linear and rotation velocities. The optimal path for the mobile robot is selected using the maximum value returning from a cost function, which includes three factors, namely: the alignment of the robot path with the goal, forward velocity, and the distance to obstacles. The DWA was extended in the Model Based Dynamic Window approach [Fox et al., 1997].

2.6.2 Avoidance of Dynamic Obstacles

Dynamic obstacle is any object which changes location with the time during the robot navigation. Kanet and Zucker [1986] proposed the path-velocity decomposition principle, in which a long-term path is planned to avoid static obstacles, and then the robot velocity is modified along this path to avoid dynamic obstacles. The main drawback of this method is that modifying the velocity is sometimes not enough to avoid dynamic obstacles especially those obstacles moving towards mobile robot. Fraichard and Laugier [1993] proposed the principle of adjacent paths to overcome the previously defined problem, where several paths are planned to avoid static obstacles, and for dynamic obstacles a collision-free path is chosen, however, this method is time consuming.

In [Zhuang et al., 2006], an autoregressive model proposed in [Sen and Srivastava, 1990] is used to estimate the path of a dynamic obstacle, where a path planner plans a straight path for free space, creates a subgoal to avoid a static obstacle, while the estimated future position of a dynamic obstacle is considered before creating the subgoal. The main drawback of this method is that the optimal path cannot always be found.

The velocity obstacle (VO) approach is an easy and simple method to avoid moving obstacles, in which the collision cone principle is used to predict a collision situation between two circular-shaped objects moving with linear motions [Fiorini and Shiller, 1998]. In the VO approach, collision between a robot R and obstacle O is predicted if the resulting vector v_{ro} of the robot speed v_r and obstacle speed v_o is located in the collision cone as illustrated in Figure 2.18. Hence, any speed v_r which leads the relative speed v_{ro} to be located in the collision cone will cause collision. Consequently, the collision risk is minimized if the relative speed vector is controlled and located outside of the collision cone. The VO approach has two challenges when applied in indoor environments. The first challenge is to extract collision cones of non-circular objects from sensor data, where applying fitting circle methods generally produce large and inaccurate collision cones specially for line shaped obstacle such as walls. The second challenge is that the mobile robot cannot sometimes move to its goal because all its velocities to the goal are located within collision cones.

The Non-linear V-Obstacle (NLVO) method was proposed as an extension of the VO method for avoiding obstacles moving in arbitrary trajectories [Frederic Large and Shiller, 2005]. Consider the case shown in Figure 2.19, in which an obstacle O at time t_o moves in an arbitrary trajectory. To avoid the obstacle, the intersection point $C(t)$ between the current straight path of the robot R and the obstacle's path is calculated, the time t at which the obstacle reaches the intersection point is also calculated, and then an approximation is made by considering the obstacle $O^{'}$ moves at a tangent to the obstacle path at point $C(t)$ and at a constant speed in which the obstacle $O^{'}$ reaches the intersection point at time t. Now the VO approach can be applied to define the collision situation between the robot and obstacle $O^{'}$.

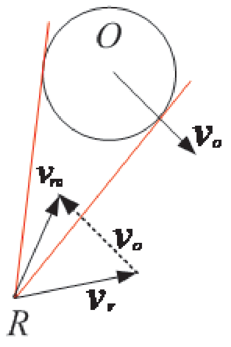


Figure 2.18: VO approach

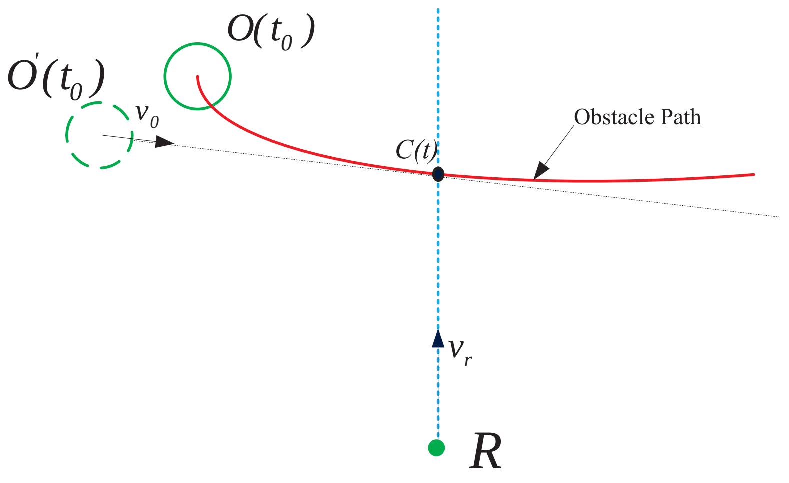


Figure 2.19: NLVO approach

Many studies have adopted this method to avoid moving obstacles using simulation [Zhiqiang et al., 2008; Fulgenzi et al., 2007]. In [Yamamoto et al., 2001], all obstacles were assumed to be circular, and the principle of the collision distance index was introduced to quantify collision risk, where obstacles are enlarged according to their distances from the robot. In [Kluge, 2003; Kluge and Prassler, 2004], the probabilistic velocity obstacle (PVO) principle and a recursive model were proposed. According to the PVO principle, the grown obstacle is extended by uncertainty to the exact radii of the obstacle and robot, where the collision probability in the exact collision cone is given a value of 1, while in the uncertain area, the collision probability is given a value between 0 and 1. The recursive model estimates the level of intelligence of the obstacles to predict their behaviour. However, only circular obstacles were used and the time to collision was not considered. The Optimal Time Horizon Principle has been proposed by [Shiller et al., 2010], where the smallest time horizon is used to define the collision course velocities. In their simulation, circular obstacles of known speeds were used. However, in the real world the robot has to extract the shapes of the obstacles from sensor data, enlarge them using the robot radius, and then extract the collision cones.

2.7 Estimation of Obstacle Speed

A key requirement in avoiding dynamic obstacles is to estimate their directions and speeds using an estimation algorithm such as the Kalman filter or Particle filter. Any tracked object can be described by a N-dimensional state vector, which is unknown precisely because of measurement noise; and also some states cannot be measured directly. The tracked obstacle can be defined by two types of models: a dynamic model (motion model or state model) and a measurement model (observation model) as shown in Figure 2.20.

In the dynamic model, the state vector changes with time, where its probability $P(x_t)$ at sample time t is calculated using the probability $P(x_{t-1})$ of the state vector at $t-1$.

$$P(x_t) = \int P(x_t/x_{t-1}).P(x_{t-1}) \tag{2.2}$$

where $P(x_t|x_{t-1})$ is the process density.

In the measurement model, z_t is defined as the measurement vector at sample time t, while the historical measurement vectors are described by $Z_t = \{z_1, z_2, z_3, ..., z_t\}$. The measurement model is described as follows:

$$z_t = h(x_t, v_t) \tag{2.3}$$

where h is the measurement function and v_t is the measurement noise. To track an object, the probability of the initial state $P(x_0)$ must be known, and also the measurement z_t depends only on the state x_t to produce the probability $P(z_t|x_t)$.

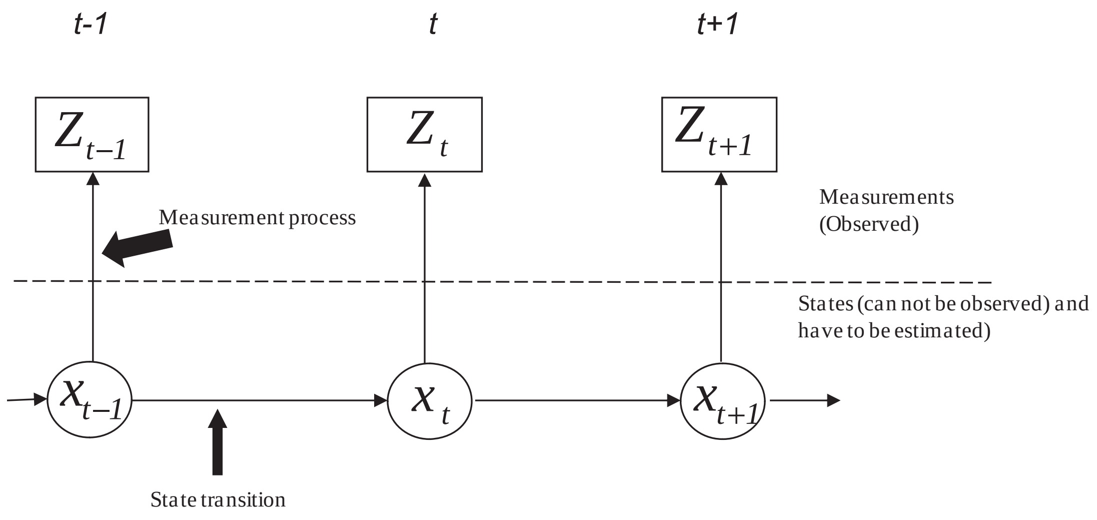


Figure 2.20: State transition and measurement process

A Kalman filter (KF) is a set of mathematical equations which implement a recursive function to estimate hidden state variables by minimizing the squared error between the observed and estimated outputs. However, the KF algorithm was not originally used for estimation purposes due to the large computation involved, but developments in microcontrollers and digital computing have made it suitable for many applications such as senseless control and autonomous navigation [Vatavu and Nedevschi, 2013; Sivaraj et al., 2012]. The estimated state x_k and measurement state z_k are given as follows:

$$x_k = Ax_{k-1} + Bu_k + w_k \tag{2.4}$$

$$z_k = Hx_k + v_k \tag{2.5}$$

where A, H, and B represent the state, measurement and input matrices respectively, and w_k and v_k represent process and measurement noise respectively .

The observed system in the KF algorithm is linear, and therefore its matrices are assumed to be constant, and the process and measurement noise are both treated as white noise with a normal probability.

The KF algorithm includes two steps, the prediction and measurement stages (see Figure

2.21). The matrices K_k and P_k are defined as the Kalman gain and posterior estimate error respectively, while Q and R represent the process noise covariance and measurement noise covariance respectively. In the predication stage, the estimated state $\hat{x}_k^-$ and estimate error P_k^- are calculated and returned to the measurement stage, where the Kalman gain is calculated and then the estimated state and estimate error are updated.

The Kalman filter algorithm consumes less time for tracking a signal obstacle than that consumed by the Particle filter but it becomes more complex as the number of tracked obstacle increases.

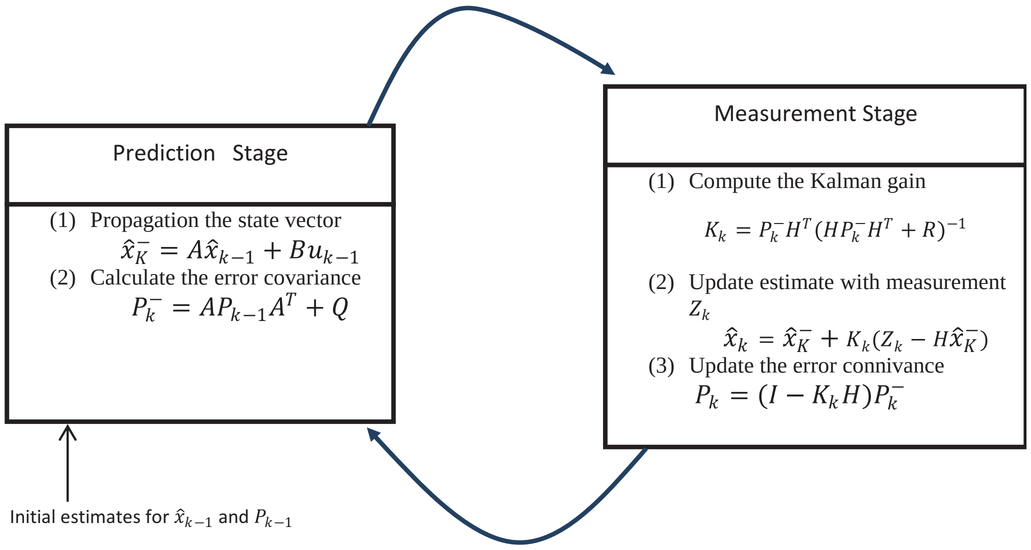

Prediction Stage
(1) Propagation the state vector
$\hat{x}_K^- = A\hat{x}_{k-1} + Bu_{k-1}$
(2) Calculate the error covariance
$P_k^- = AP_{k-1}A^T + Q$
Measurement Stage
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement Z_k
$\hat{x}_k = \hat{x}_K^- + K_k(Z_k - H\hat{x}_K^-)$
(3) Update the error connivance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure 2.21: Kalman filter algorithm

The Particle filter (PF) is sometime referred to as sequential importance sampling (SIS) and sequential Monte Carlo methods. PF was not used extensively until Isard and Blake inserted the resampling stage [Isard and Blake, 1998]. The mathematical derivations of the PF algorithm are not analyzed in depth here; however further details can be found in [Doucet et al., 2000]. The standard PF consists of five steps, namely: initialization, propagation, importance, resampling, and generating output, as follows:

1. Initialization - implemented only at $t = 0$, where the sample set S of N particles and their weight W are initialized.

2. Propagation stage - the particle set S is updated by the dynamic model described in Equation 2.4, and this produces a new particles set $S^{'}$.
3. Importance stage - the weights of the particles are calculated from the likelihood $P(z_t/x)$ depending on the error between the outputs of the observation model and the particles.
4. Resampling stage - the particle set $S^{'}$ is reselected depending on their weights, producing a new particle set $S^{''}$, where high weight particles are multiplied, while low weight particles are rejected.
5. Output stage - the estimated state is calculated by the mean of $S^{''}$.
6. Go to step 2 for the next sample time the particle set $S^{''}$ at sample time t becomes the particle set S at sample time $t+1$.

The PF method has been used to track multiple objects [Orton and Fitzgerald, 2002], but this technique has the following disadvantages; i.e a newly appearing object is not tracked unless it is very close to one of the tracked objects, and some objects can be rejected in the resampling stage.

To avoid the disadvantages of the standard particle filter, the extended particle filter (EXPF) or extended condensation algorithm was proposed for tracking multiple obstacles using only signal distribution [Marron et al., 2004; Koller-Meier and Ade, 2001]. The basic idea of the EXPF algorithm is to add a re-initialization step in each loop of the PF algorithm, where a new group of M samples is redistributed for obstacles clustered from the sensor data.

In the EXPF algorithm, any new object appearing in the sensor data will be tracked in the next cycle, and also no object will be rejected because the inserted samples in the re-initialization step have high weights since they are very close to the objects. However, choosing a small value of M does not affect the overall probability distribution of particles, which overcomes the risk of objects disappearing during the Resampling stage. For the Propagation stage, there is no modification on this step, and the whole particle set S is updated through the state model. In the Importance stage, only a single distribution is used, where the likelihood $P(z_t \Big| x_t)$ of each particle is calculated according to the nearest object, as follows:

$$P(z_t|x_t) = e^{-d_t^2/a} \tag{2.6}$$

where d_t is the minimum value of the distances of the particle from the all objects.

In the Resampling stage, $N-M$ particles are reselected. The estimated state cannot be calculated directly from the mean values of the samples, hence various solutions have been proposed to solve this problem such as the Nearest Neighbour technique as used by [Koller-Meier and Ade, 2001], whereas related studies have used K-means clustering [Cox (1993); Marron et al. (2004)].

2.8 Clustering Laser Data

The robot has to define the number, locations and sizes of obstacles to achieve successful dynamic obstacle avoidance. Hence, it is crucial to segment the sensor data using clustering methods. This section focuses on the clustering of 2D laser range finder (LRF) data, which consists of N points and each point can be described by pairs of polar coordinates (r,a) or in Cartesian coordinates (x,y), as shown in Figure 2.22.

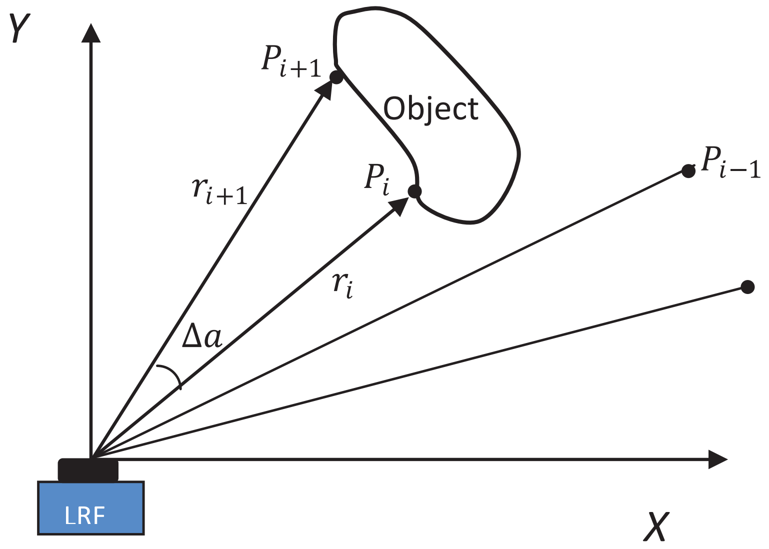


Figure 2.22: Schematic representation of scan laser data

The clustering process involves dividing the sensor data into separate groups which represent the existing objects in the robot's workspace. Clustering methods can be divided into two types; point-distance-based methods (PDBS) and Kalman-filter-based methods (KFBS) [Nunes and U., 2005; Rebai et al., 2009]. For the PDBS methods, the distance between two consequent points is given as follows:

$$d(r_i, r_{i+1}) = \sqrt{r_i^2 + r_{i+1}^2 - 2r_i r_{i+1} cos \triangle a} \tag{2.7}$$

where r and Δa define the calculated distance and the angular resolution.

The distance $d(r_i, r_{i+1})$ is compared with a threshold value D_{th}, and if the condition $d(r_i, r_{i+1}) < D_{th}$ is met, the two points belong to the same object; otherwise they belong to two different objects. However, different methods can be used to determine the threshold distance, which can be held constant, but then the clustering process does not segment the sensor data properly because the distance between the laser points changes depending on distance from the laser sensor. In [Lee, 2001], the threshold is calculated as:

$$D_{th} = \frac{|r_i - r_{i+1}|}{|r_i + r_{i+1}|} \tag{2.8}$$

It should be noted that the threshold distance is calculated using the distance between the detected object and the laser sensor, which can make separating the objects more difficult as they are further away. In [Dietmayer et al., 2001], the threshold distance is given according to the following equation:

$$D_{th} = C_0 + C_1 min(r_i, r_{i+1}) \tag{2.9}$$

where C_0 is a constant value used to reduce noise, and C_1 is given as follows:

$$C_1 = \frac{d(r_i, r_{i+1})}{r_i} \tag{2.10}$$

In this method the threshold distance depends on the distance to the sensor. In [Santos et al., 2003], a new parameter β is introduced to reduce the effect of object distances, where the threshold distance is defined as follows:

$$D_{th} = C_0 + \frac{C_1 min(r_i, r_{i+1})}{cos(\beta)(cos(\frac{\triangle a}{2}) - sin(\frac{\triangle a}{2}))} \tag{2.11}$$

It is very important to tune β, since a large value can segment an object into two, while two objects can be interpreted as one object for a small value.

The K-means clustering algorithm is another PDBS method, but it does not depend on the distance between the object and sensor [Napoleon and Lakshmi, 2010]. The K-means algorithm consists of five steps:

1. Define the number of clusters K.
2. Initialize the cluster centres randomly.
3. Define the class membership of points by assigning them to the nearest cluster centre.
4. Recalculate the cluster centre using the mean value of memberships.
5. Repeat steps 3 and 4 until no point changes its class membership.

The K-means clustering is easy, simple, and efficient, however, it is not suitable for clustering non-convex shapes, and the number of clusters K needs to be defined in advance. The K-means algorithm was improved to tune the clusters number automatically using the following steps [Marron et al., 2004]:

1. Select a random number of clusters K.
2. Initialize the cluster centres randomly.
3. Define the class memberships of the N points by assigning them to the nearest cluster centre.
4. If the distance between a point to its cluster centre is bigger than a threshold, a new cluster is added.
5. Recalculate the centres of clusters using the mean value of memberships.
6. Remove cluster centres which have very few points.
7. Repeat steps 3, 4, 5 and 6 until there are no changes in class membership.

KFBS methods are used to segment laser data by applying the following algorithm for all laser points p_i [Kmiotek, 2009]:

1. Filter initialization: in this stage, the system state vector x is assigned to the first laser point p_0, and the covariance matrix is initialized $P = P_0$.
2. Implement the prediction stage of the KF algorithm.
3. Test the gating situation for point p_i: if it is located inside the gate, the update stage is implemented; otherwise p_i is selected as a break point.

The laser point p_i is considered to be inside the gate if the condition $p_i^T S^{-1} p_i < D_{th}$ is met, where S^{-1} is the measurement covariance matrix. For 2D laser data, the linear dynamic equations can be written as follows:

$$r_{t+1} = r_t + \triangle a.\frac{dr_t}{da} \tag{2.12}$$

$$\frac{dr_{t+1}}{da} = \frac{dr_t}{da} \tag{2.13}$$

$$x_t = [r_t \quad \frac{dr_t}{da}]^T \tag{2.14}$$

where r and a represent the length and the angle of a laser line respectively, while x_t is the state vector. For the Extended Kalman filter the dynamic equations are given as follows:

$$\phi_{t+1} = \phi_t + \triangle a \tag{2.15}$$

$$r_{t+1} = \frac{sin(\phi_t)}{sin(\phi_t + \triangle a)} r_t \tag{2.16}$$

$$x_t = [r_t \quad \phi_t] \tag{2.17}$$

where ϕ represents the angle between the laser line and the object surface, as shown in Figure 2.23. The extended Kalman filter gives more precise estimations than the standard Kalman filter. However, these methods are more complex than distance-based methods and require large computation.

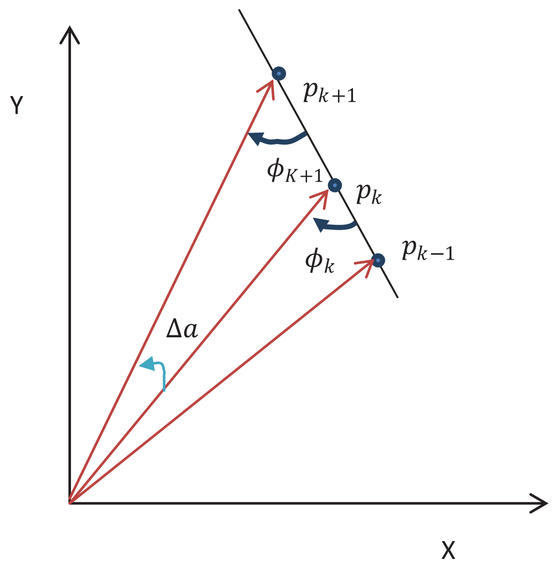


Figure 2.23: KF-based methods (KFBS)

2.9 Mobile Robot Software

Mobile robots can be programmed to execute the navigation algorithm using programming languages such as C,C++, Python and JAVA. However, the user code will be very complex and is not portable. Also it is very difficult to simulate the robot's performance. Several open source and commercial robot development environments have been proposed to solve these problems, where they provide several libraries to interface between the user code and the robot hardware. Consequently the user code can be employed on different mobile robot platforms, and is easy to debug and develop [Kramer and Scheutz, 2007]. In this section, three robot development environments namely: Player/Stage, Robot Operating System (ROS) and Microsoft Robotics Developer Studio are described.

Player/Stage is an open source robot environment developed at the University of Southern California, and utilizes the Linux operating system [Gerkey et al. 2001]. Player is a device server using the TCP/IP protocol to interface between the user code and the robot devices. It is easy to use, supports 13 robot platforms such as Pioneer , Roomba, and Segway robots, and offers several packages to simplify the user code such VHF, AMCL and Wavefront Planner [Kerr and Nickels, 2012]. Stage is a 2D simulation tool which models the robot hardware to execute the user code written in the Player environment. The workspace and the robot devices are simulated in Stage using a *configuration file* and a *world file,* where the user defines the shapes of objects and the robot, sensors, and workspace map. However Player can use the Gazebo simulation platform which is a 3D simulator. Like Stage, it is capable of simulating robot hardware, sensors and objects, furthermore, the dynamic of the robot can be simulated.

Robot operating System (ROS) is another open source software suite used to control mobile robots under Linux operating systems. It supports approximately 60 robots, can use Stage and Gazebo as simulation environments, and supports UDP and TCP/IP for information passing. However, it can take a long time to become proficient in using the ROS structure effectively.

Microsoft Robotics Developer Studio (RDS) is a commercial software suite working under the Windows operating system, and uses the C# programming language. Virtual Simulation Environment (VSE) has been developed to simulate and verify the navigation algorithm. RDS is easy to learn, but it supports few robots.

2.10 Summary

This literature review has provided the necessary background to start this project, discussing the control architectures; deliberative, reactive and hybrid by which a mobile robot integrates its tasks to achieve successful autonomous navigation. To produce a robust navigation system, the hybrid architecture can be used, where it overcomes the problems of the slow response of the deliberative architecture and the lack of planning in the reactive architecture by combining the two architectures. This requires a robust localization system to estimate the current position of the mobile robot, where the AMCL system is simple and easy to implement for map-based navigation. A global path planning system is also required to find a safe and optimal path from the current location to the destination, where many methods have been proposed to represent the world map and various other methods to plan an optimal path. For local path planning, several algorithms have been used to avoid static and dynamic obstacles. Avoiding static obstacles does not require any estimator to measure the obstacle speed, but for dynamic obstacle avoidance, sensor data must be segmented to define locations of the obstacles and then their speeds and directions are estimated using an estimator such as Kalman or particle filters.

Chapter 3

Proposed Control Architecture

For indoor environments, a mobile robot must have the ability to reach its goal from any point within its workspace and avoid any collision which can cause harm to the robot or other objects. To meet these requirements, a hybrid control architecture has been chosen to control the mobile robot. As mentioned previously, this architecture combines the powerful planning capability of the deliberative architecture with the fast response of the reactive architecture. The next section discusses tasks that should be included in of the control architecture.

3.1 Mobile Robot Architecture

The proposed control architecture includes three models: deliberative, intermediate and reactive. Each model must contribute some function or behaviour to produce an optimal navigation. The required tasks from each model will be discussed and then suitable functions will be designed to fulfil the requirements.

3.1.1 Deliberative Model

The deliberative model builds the world map of the robot workspace based on the sensor data returned from the surrounding environment. Next the world map is processed to compute a path planning algorithm. However, this will take a long time resulting in a slow response. The hybrid control architecture used in this project is designed to work in indoor environments, where the workspace maps is usually known. Therefore there is no need for building the map.

Walls and other obstacles may prevent the robot implement a straight line path between its current location and its goal, and also in most cases, the robot has to select an optimal route among several available routes to its final target. In this project, the deliberative model covers this task by including a long path planning algorithm which produces an optimal path defined by several subgoals.

The reactive level has to deal with any change in the robot's workspace, and sometimes the changes in the robot's workspace may block the robot's path such as when a corridor is closed for cleaning. In this case, the robot must wait until the planned route becomes available which may take a long time. The user can resolve this problem by modifying the map, however, in this project an alternative solution is proposed to avoid this problem by using an updating function which automatically updates the map when ever the mobile robot is trapped in a local minima.

The long path planning algorithm cannot work without defining the absolute position of the start point, and also the robot may lose its path when the accumulation of error produced by wheel slippage increases too much. As a consequence, the deliberative model must use a localization system to produce a robust estimation of the robot position. The robot needs be aware of the moving obstacles suddenly appearing from open doorways. To this, the wall extraction function is added, where it defines surrounding walls and then locations of open doors. The proposed control architecture needs a intermediate model (sequencer model) which links the deliberative and reactive levels.

3.1.2 Reactive Model

The robot can increase its response speed using the reactive model, when there is no need to continuously apply the long path planning algorithm . Here, the reactive level comprises the *go to goal* behavior which calculates the optimal angle and speed of the robot to implement a short path between its current location and its subgoals.

To avoid static and dynamic obstacles that were not considered during the global path planning, the reactive model uses the *obstacle avoidance* behaviour which extracts obstacles from the sensor data, estimates their speed and then defines free collision velocities to minimize the collision risk. The robot may work in a workspace cluttered by humans, therefore in this project, the reactive model uses the *human avoidance* behaviour to detect human and estimate their speeds from the laser data returned from their legs.

In a dynamic environment, there could be many open doorways and corridor crossing which are considered as blind spots from which unexpected obstacles may appear. As a consequence, the reactive level uses the *unobserved obstacle avoidance* behaviour to reduce the collision risk with objects appearing from those locations. Furthermore, the reactive model eases the robot task using *the follow the leader* behaviour, in which the robot follows other obstacles moving in the same direction and a similar speed. Sometimes the robot does not have enough time or space to avoid collisions, therefore the *emergency stop* behaviour is added to stop the robot when a collision becomes unavoidable.

3.1.3 Proposed Control Architecture

Based on the previous discussions in this chapter, the block diagram of the control architecture is proposed as shown Figure 3.1, in which the deliberative model includes four functions namely: global path planning, localization, map updating and wall extraction. The intermediate model is responsible for the transfer of the information between the reactive and deliberative models, such as the subgoals and locations of the walls, and also to monitor the reactive model to define when a lcoal minma occurs. When the robot is trapped in a local minima, the map updating function is activated to update the robot map, and then the long path planning algorithm is applied to reselect a new optimal path.

The reactive level includes six behaviours namely *go to goal*, *obstacle avoidance*, *unobserved obstacle avoidance*, *human avoidance*, *follow the leader* and *emergency stop*. The outputs of behaviours are integrated using the coordination model to produce optimal control commands.

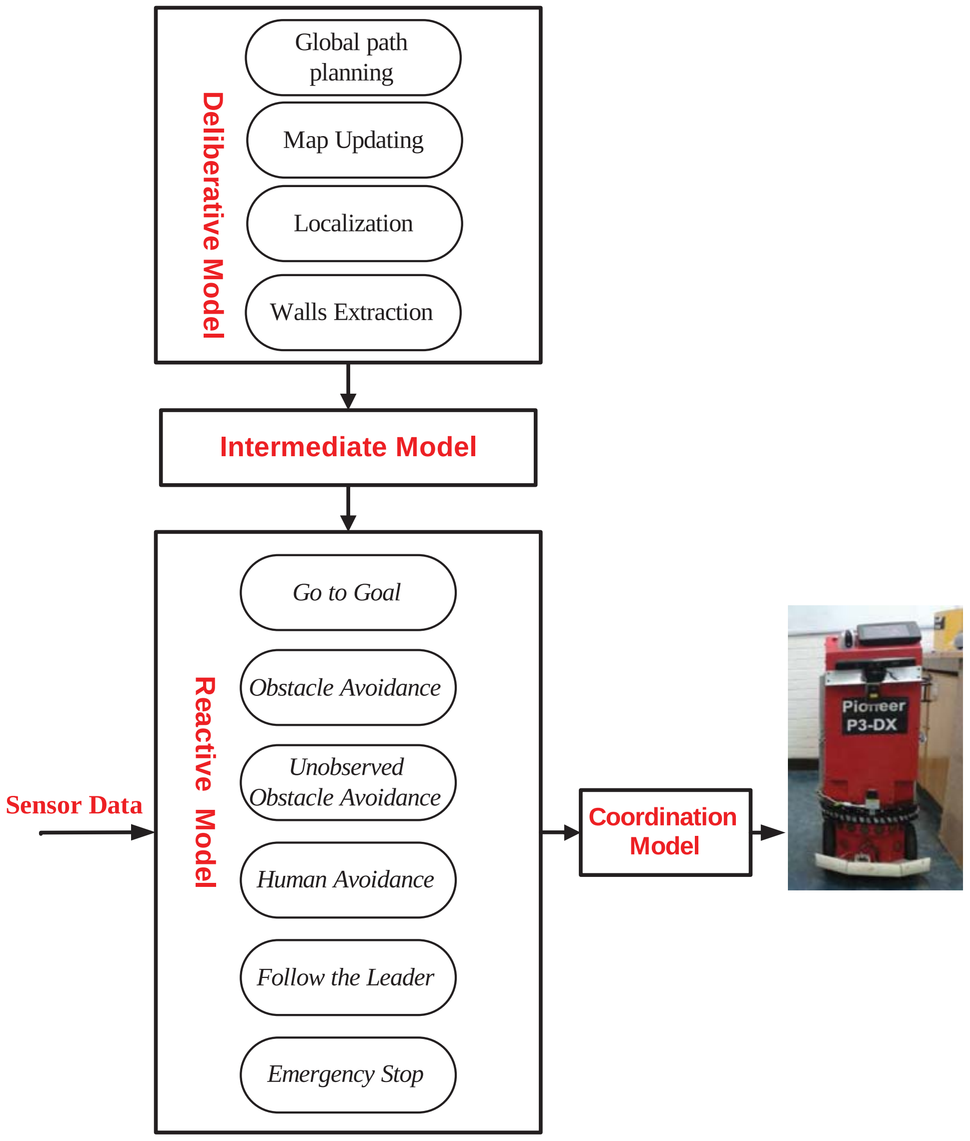


Figure 3.1: Proposed control system

3.2 Hardware Limitations

Experiments will be implemented using two mobile robots; a modified Pioneer robot (the intelligent robot) and the Nubot robot (whose role is a dynamic obstacle) (see Figure 3.2). The Pioneer robot was fitted with two Hokuyo laser sensors, a Kinect camera and sonar sensors.

Sonar Sensors: the standard Pioneer robot comes with an Onboard sensor array of eight sonar sensors which are triggered sequentially to avoid cross talking. However, an additional array of 12 Devantech SRF05 ultrasonic sensors have been attached to the robot to cover 180º and a range of 1cm up to 4m. However,the sonar sensor has a low firing rate of 3Hz [Nattharith, 2010] makes the these sensors are unsuitable to detect and estimate the speeds of dynamic obstacles.

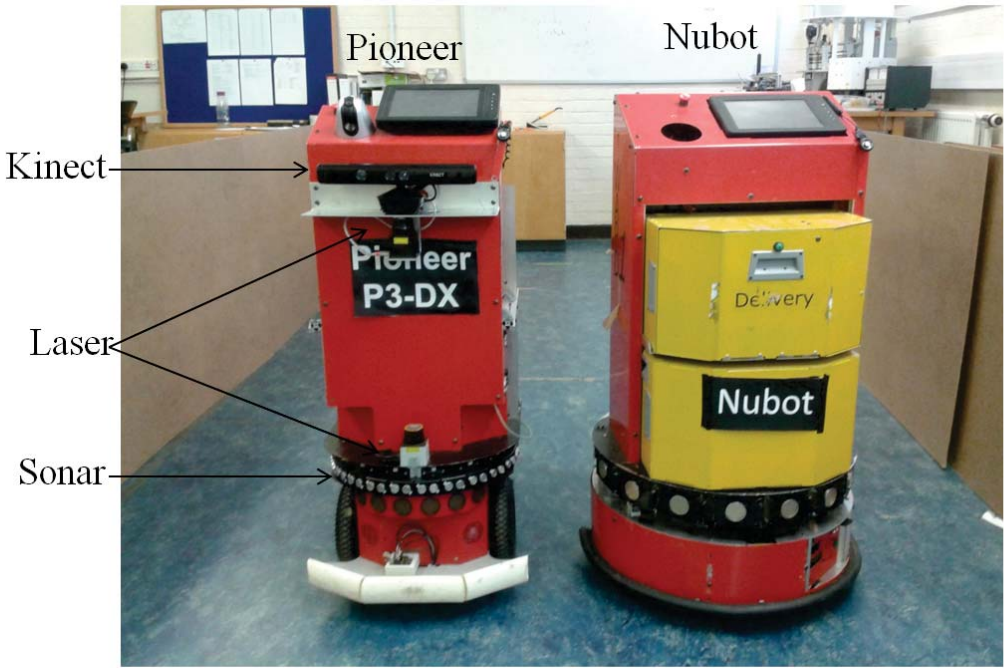


Figure 3.2: Mobile robots used in this project

Kinect sensor: Kinect sensor is a depth camera developed by Microsoft for tracking players and people motions for playing games. Recently it has been used in robotic navigation [El-laithy et al. 2012]. Although the update rate of the Kinect sensor is high, it has a narrow field of view (57^oH and 43^oV)

Laser Finder range: Two Hokuyo URG-04LX scanning laser sensors were mounted on the pioneer robot at heights 0.40m and 0.70m. The main specifications of the Hokuyo laser sensor are given:

- Scanning range 240º
- Angular resolution 0.36º
- Update rate 10Hz
- Maximum distance 5.6m

In this project, only the two laser sensors will be used to recognize the surrounding workspace. The maximum range and update rate of the sensor restrict the maximum speed of the robot to 0.5m/s and allow it to avoid only slow motion obstacles. If the robot or dynamic obstacles move at a higher speed and towards each other, the relative speed of the robot will be high and the robot will not have enough time and distance to avoid the collision. Furthermore, the high speed of the robot and the time delay (100ms) of updating the control commands will make the robot's movement quite erratic especially in narrow spaces. However, the same proposed control system can be used to avoid faster obstacles if the update rate and maximum range of the sensor are increased by using a different version of the laser sensor such as the Hokuyo UTM-30LX which has a maximum distance of 30m, angular resolution of 0.25º, scanning range of 270º and update rate 40Hz.

3.3 Simulation Environment

It is crucial to validate the proposed navigation algorithm in a simulation environment before using existing robots (Pioneer and Nubot robot). Using simulation tests helps in saving time, where the experiment's workspace can be reinitialized very quickly just by restarting the simulation program. In contrast, in the real world, each time the robot must be returned to its initial point and connected to its charger when the power of its batteries becomes low. Furthermore, any fault in the robot's hardware may lake a long time to fix. On other hand during the simulation tests, the navigation algorithm code can be safely tested without any concern about damaging the robot hardware or causing harm to any individual or another object within its workspace.

For this project, Player/Stage has been chosen to validate the navigation algorithm for the following main reasons :

1. Player/Stage was already installed in the Pioneer and Nubot On-board PCs. Using different software will consume a lot of time to install, especially for the Nubot robot which is not supported by other software.
2. It is free and very easy to learn, as simple commands are used to read the sensor data and control the motor speed. Furthermore, a lot of documents about the programming

in Player/Sage can be found on the internet.

3. It is very simple to prepare the simulation environment and to configure the sensor drivers inside the algorithm code.

In this project, simulation experiments are designed to show that each function block in the proposed control system is able to successfully implement its task. On other hand, the laser range finder will be configured as the real Hokuyo scanning laser, in this case the same code can be implemented on the physical robot without any modification.

The long path planning algorithm is tested using a predefined map with a known starting point; for successful long path planning, the robot must easily reach its goal without using the obstacle avoidance behaviour. Next, the ability of the localization system to estimate the real position of the robot is validated by choosing an unknown starting point of the long path planning, where the robot can only reach its goal if the localization system is activated. The wall extraction algorithm is tested to define the surrounding wall using a predefine map and laser data.

The reactive model is first tested to avoid static obstacles and then to avoid dynamic obstacles in both free space and indoor environments.The importance of the *unobserved obstacle avoidance*, and *follow the leader* behaviours is explained by comparing the collision results when these behaviours are inactive and active. A simulation experiment will be designed to show that the stop emergence behaviour will be automatically activated to save the robot and other objects when there is not enough space and time to avoid collision. Another simulation test will be designed to show that the map updating and long path planning algorithms can solve the problem of trapping in local minima. For the *human avoidance* behaviour, the human gait will be modelled in Player/Stage and then human tracking and avoidance algorithms will be tested before applying the navigation algorithm within a physical robot.

3.4 Summary

This chapter introduced a general explanation about the proposed control architecture and the hardware used in this project. The functions of the deliberative model were chosen to produce a long term path, estimate the absolute location of the robot, define the blind spots and update the map when the robot is trapped in a local minima. On other hand, the behaviours of the reactive model were chosen to minimize the collision risks with static and dynamic obstacles including humans, also to ease the robot task in cluttered environments and to stop the robot during emergency situations. For this project, Player/stage has been chosen to validate the navigation algorithm because it is free, easy , simple and no need for writing a new communication protocol for the Nubot robot.

Chapter 4

Deliberative Model

This chapter explains more the deliberative level. The deliberative level includes four models namely: global path planning, localization, map updating and wall extraction. Simulation tests using the Player/Stage platform showed that the localization system provides a robust estimation of the robot position, likewise the path planning algorithm can establish an optimal and safe path between the goal and the current location of the robot, and the wall extraction algorithm is able to determine the positions of walls around the mobile robot. However, an simulation test in the next chapter explains how the map updating function helps to avoid a local minima.

4.1 Importance of the Deliberative Model

Producing safe and successful navigation is considered to be a key challenge for a mobile robot. Imagine that some people want to go to a train station, although they know the location of the train station they cannot go there if they do not know their current location. Similarly for a mobile robot, the first challenge to find its goal is to estimate its current position, which requires a robust localization system.

The second challenge for the mobile robot is to plan a safe and optimal path from its current location to its goal. Car drivers, for example, may navigate using a map-based GPS as well as landmarks. Likewise, the mobile robot must have a long-term path planning system to create subgoals between its current position and its goal.

The third challenge to produce safe navigation is that during navigation in an indoor environment, the robot will have to navigate among many walls which represent long static obstacles, and at the end of each wall there could be an open door from which an unobserved moving obstacle may suddenly appear in its path. Hence, it is crucial for the robot to find walls in the surrounding space in order to detect open doors and corridor crossings, and then to execute suitable actions to minimize the collision risk with unobserved dynamic obstacles.

The fourth challenge to produce successful navigation is that when any change in the robot workspace blocks the planned path the robot will be trapped, therefore the map must be updated and then the long path planning algorithm is applied to find new path to the target.

4.2 Mobile Robot Localization System

Localization is the process of estimating a position with respect to a general reference frame. To achieve successful localization, it is very important to understand the mobile robot surrounding environment, which can normally be classified into two types; unknown and known environments.

Unknown environment - here the robot does not have sufficient information about its workspace, and therefore it must build up its own map and estimate its current position depending on the information collected from sensors such as laser or sonar. Many studies have been carried out in this area, which is referred to as simultaneous localization and mapping (SLAM) [Chakravorty and Saha, 2009; Min et al., 2009; Jinseok and Kubota, 2011]. Here, the navigation algorithm of the mobile robot is relatively complex and requires huge computations to build the workspace map, and then to determine its goal through extracting landmarks. Furthermore, the complexity of the navigation system increases considerably in dynamic environments.

Known environment - in this type of workspace, the robot has complete information about its environment, and so the navigation algorithm is easy to implement, particularly for localization and path planning. In this project, mobile robots have been designed to work in indoor environments which are considered to be known environments.

For a wheeled mobile robot, the localization system can be implemented using odometry sensors. Although this method is cheap and simple to implement, its major disadvantage is integration error caused by wheel slippage. However in a known environment, the integration error can be cancelled using the adaptive Montecarlo filter localization (AMCL) algorithm, which estimates the robot's position by combining information coming from both external (laser and sonar) and odometry sensors. The core of the AMCL algorithm is the particle filter, which uses a group of samples (particles), each of which is considered as a virtual robot having a 2D location, direction and sensors. The sample sensor data are computed depending on the current location of the sample in a predefined workspace map. The term "adaptive" is added to the Montecarlo filter because the number of samples is changed according to uncertainty in the estimated location of the mobile robot. In other words, the sample number increases as uncertainty in the robot position increases, while it decreases as the estimated position becomes closer to the real location. The AMCL algorithm has the following advantages:

- It is easy and simple to implement.
- It can deal with non-Gaussian noise.
- It estimates the global position of the mobile robot even if the initial position is unknown.

However, it also has the following disadvantages:

- It requires a large number of samples at the beginning of the robot navigation localization algorithm, which makes computation size huge.
- It is not suitable for use in unmapped areas.

4.2.1 AMCL Implementation

Figure 4.1 shows the block diagram of the AMCL localization system, which has three inputs: the workspace map, external sensor data and odometry data, while its output is the estimated position of the mobile robot. If the initial position of the mobile robot is unknown the particle samples are distributed to cover the whole map, where all particles are initially given the same weight. When the mobile robot moves, the samples update their locations depending on the position update coming from the real robot, and then random noise is added. In the Importance step of the particle filter, the weights of the samples are modified depending on the likelihood produced by the comparison between the sensor data from the real robot and that coming from samples. In the Resample step, the samples are redistributed depending on their weights, where the high weight particles are duplicated, while very low weight particles are removed.

An example of the AMCL localization system was implemented using Matlab. In Figure 4.2.a, a set of 4000 samples was initially chosen for estimating the position of the mobile robot, which moved with a constant speed of 0.5m/s. Each sample was given the same initial likelihood weight $1/4000$. Note that the sum of the sample weights must be 1. As shown in Figure 4.2.b, the number of the particle was reduced to 2793 at $t = 9$s, and also the sample distribution was reduced. At $t = 13$s, the samples number was reduced to 2029 (see Figure 4.2.c), and at $t = 24s$ almost all samples converged to the real position, as illustrated in Figure 4.2.d.

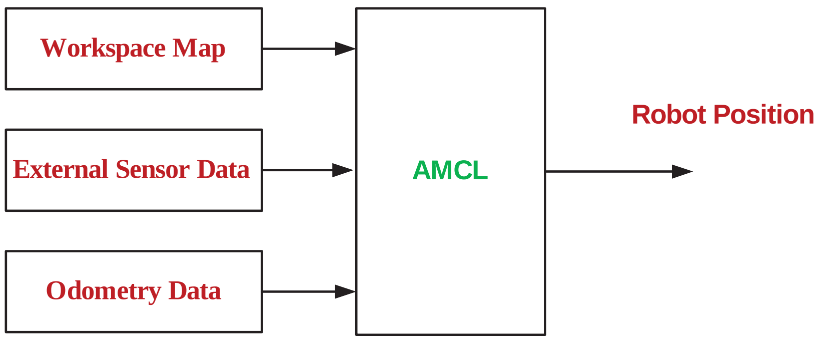
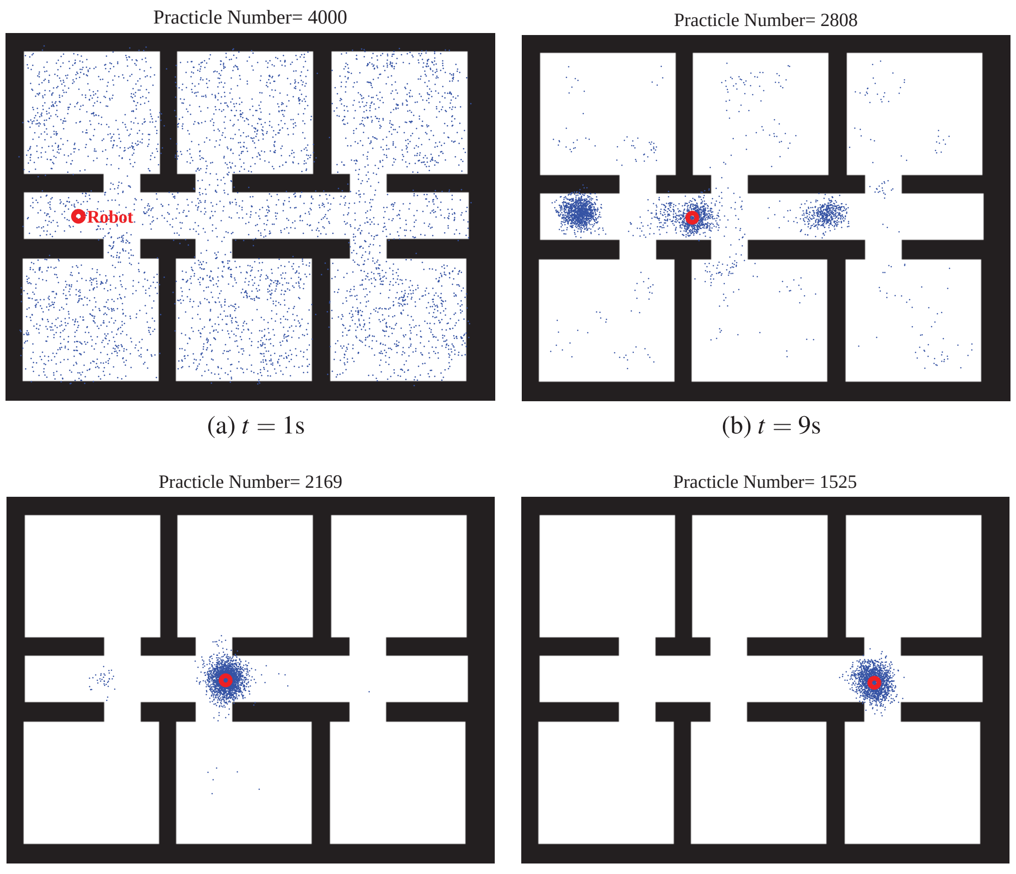


Figure 4.1: Block diagram of the AMCL localization system

4.2.2 Limitations of the AMCL

A major problem with AMCL is the relatively large amount of computation required in the early stages due to the large number of samples. However, choosing a smaller number of samples may result in failure of the localization system. Also, if the sensor data is updated faster than the AMCL algorithm can handle, a large delay is produced in position estimation. However, this problem can be avoided by decreasing the speed of the robot at the beginning. Another limitation is that failure in the localization system may sometimes occur because of the symmetry of the workspace, where different samples produce the same sensor data in different locations.



(a) $t = 1$s (b) $t = 9$s

(c) $t = 13$s (d) $t = 24s$

Figure 4.2: AMCL implementation in Matlab

4.3 Global Path Planning

Global path planning or long-term path planning is a process of finding a safe and optimal path from the current location of the mobile robot to its goal with respect to a general world frame. The inputs of this algorithm are the workspace map, current location and goal, while the output is a group of subgoals or waypoints which represent odometry landmarks. As mentioned in Chapter 2, global path planning requires two steps: a world representation stage and path search. In this project, the wavefront path planning algorithm was chosen to establish the global path because it has several advantages, namely:

- Ease of implementation.

- Low computation.
- Low sensitivity to map resolution.
- It can deal with obstacles of any shape.
- Player/Stage already has a wavefront planner implemented.

The wavefront algorithm can only be suitably applied using the grid representation of the robot workspace, which divides the workspace map into a grid of cells. A cell is considered as occupied if any part of it is occupied by an obstacle; otherwise, it is considered to be a free cell. The size of the grid square is very important, as a large cell can waste a lot of space, whereas small cell size increases the computation time. To make the global path planning easier, the mobile robot is treated as a singular point by enlarging all obstacles by the robot radius plus a safety distance.

To establish the global path, all occupied cells are given a numerical value of 1, while free cells are given numerical value of 0 as illustrated in Figure 4.3.a. The white and gray colours represent free and occupied cells respectively, and *R* and *G* refer to the robot current position and its goal. It can be observed from Figure 4.3.b that the target cell is given a numerical value of 2 , and then all zero-weight neighbouring cells are given a numerical value of 3. In the next step, all neighbouring cells of a cell having the numerical weight 3 will be given a value of 4, if they are free; otherwise they are given the lowest of its current values and the value of 4. The same procedure is repeated until the cell occupied by the robot is reached. The global path search algorithm begins from the cell occupied by the robot, where the path follows a decrease in the numerical values until the goal is reached. The cells of the global path are called waypoints or subgoals, which are represented by the green cells as shown in Figure 4.3.c. The path can be optimized by eliminating some waypoints; the *i*th waypoint is eliminated if there is no obstacle between the subgoals *i-1* and *i* as shown in Figure 4.3.d.

4.4 Wall Extraction

A wall can be extracted offline from a known map of the robot workspace or online depending on the sensor data. In this section, offline wall extraction is implemented using the Hough transform (HT), where the start and end points and the slope for each wall are determined.

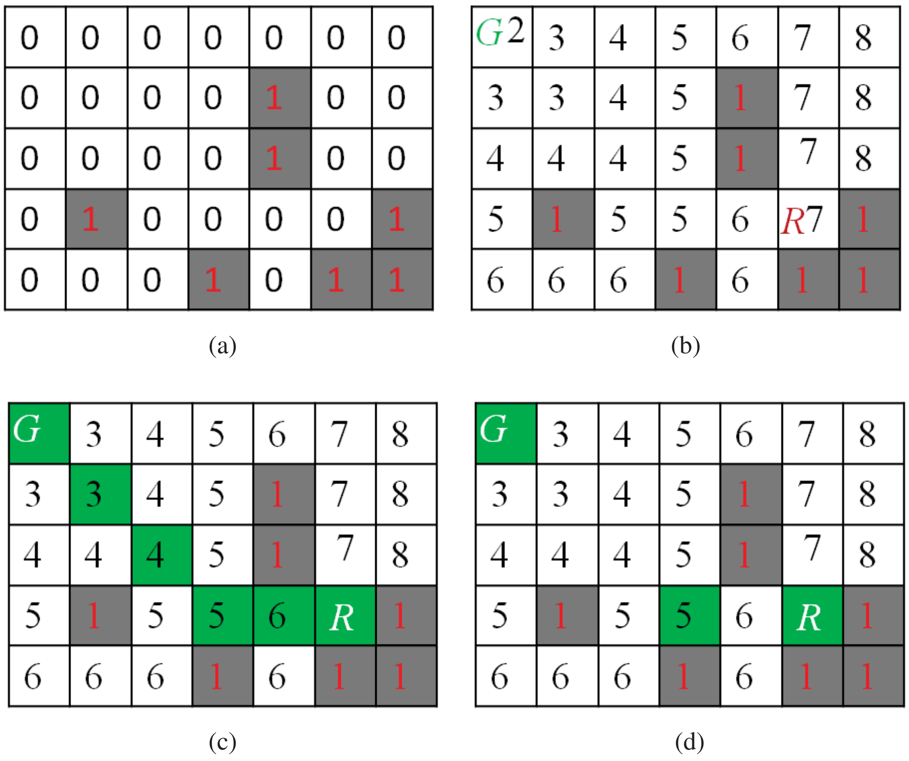
The HT algorithm aims to define the direction and location of certain features such as lines and circles from a digital image [Hough, 1962]. Hence, walls in the mobile robot workspace can be defined by extracting lines from the robot map, where the line can be defined by the following equation:

$$y = ax + b \tag{4.1}$$

where a and b represent the slope and intercept of the line respectively.

Lines passing through a point (x,y) can be found by rearranging (4.1) as follows:

$$b = y - ax \tag{4.2}$$



(a)

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

(b)

*G*2	3	4	5	6	7	8
3	3	4	5	1	7	8
4	4	4	5	1	7	8
5	1	5	5	6	*R*7	1
6	6	6	1	6	1	1

(c)

G	3	4	5	6	7	8
3	3	4	5	1	7	8
4	4	4	5	1	7	8
5	1	5	5	6	*R*	1
6	6	6	1	6	1	1

(d)

G	3	4	5	6	7	8
3	3	4	5	1	7	8
4	4	4	5	1	7	8
5	1	5	5	6	*R*	1
6	6	6	1	6	1	1

Figure 4.3: Wavefront algorithm implementation

The possible lines that pass through point (x, y) are defined by changing a and calculating b, where x and y are considered constant parameters. However, for a vertical line, the parameter

a becomes infinite for all points. To solve this problem, the line equation can be written as follows:

$$x\cos\theta + y\sin\theta = \rho \tag{4.3}$$

where ρ refers to the distance between the line and the original point (0,0), and θ represents the angle between the x-axis and the line, as shown in Figure 4.4.

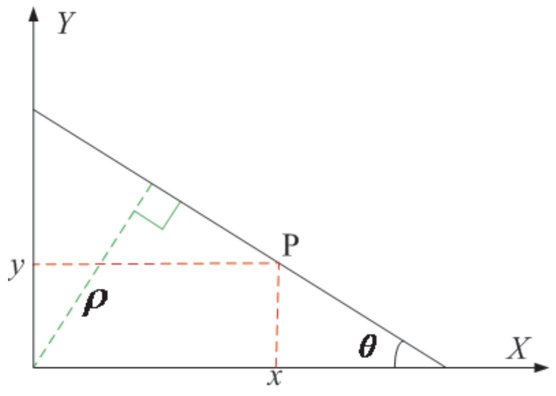


Figure 4.4: Line parameters

4.4.1 Hough Transform Implementation

The HT algorithm uses an accumulator to extract features from a digital image, where the angle θ and distance ρ are discretized using a suitable step size. Note that a small step increases the computation size. On the other hand a large step may cause the loss of some feature lines. Figure 4.5 shows the HT accumulator, where the original point is chosen as the pixel $(0,0)$, the range of the angle θ is chosen to be $(-90,90)$, and the range of the distance ρ is (0,image diameter (d)). To extract lines, each cell $(\theta\,,\,\rho\,)$ in the accumulator is initialized with a numerical value of 0, and then firstly the line angle θ is changed, while the distance ρ is calculated for each occupied point in the robot map (x_i, y_i). Next the numerical value of the cell $(\theta\,,\,\rho\,)$ is increased by 1.

Consider the image shown in Figure 4.6.a which includes two lines. Applying the HT algorithm produces the $(\theta\,,\,\rho\,)$ grid shown in Figure 4.6.b, in which the two brightest points form parameters of lines passing from most of the points in the digital image. The lines are found by applying a threshold to the numerical values of the accumulator. However, the HT algorithm does not provide the start and end points of the extracted lines, and does not consider the distance between points belonging to the same line. To solve this problem, the distance between two sequential points located on a line is calculated, and if this distance is greater than the threshold the first point is assigned as the end of the current line and the second point is assigned as the start point for a new line.

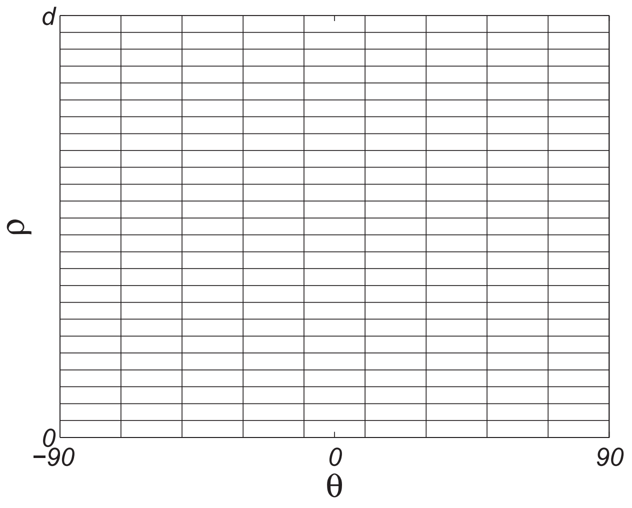


Figure 4.5: HT accumulator

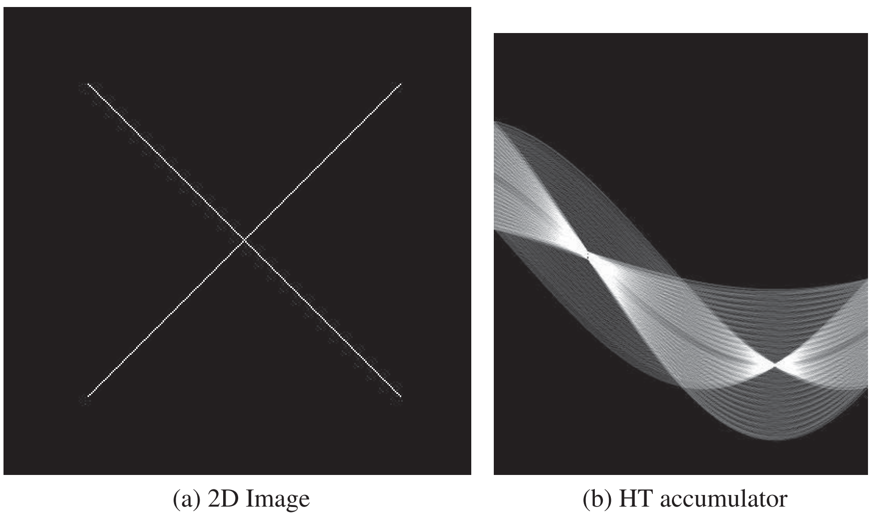


Figure 4.6: HT implementation

4.4.2 Wall Extraction from Laser Data

As mentioned previously, in indoor navigation, walls can be extracted offline from the known map, and then walls surrounding the robot are defined using its current position, where a precise localization system is required. However, the incremental error of the odometry localization system may lead to mistakes in detection of the surrounding wall locations. Note that off-line wall extraction cannot be applied in mapless-based navigation, since the mobile robot does not have enough information about the surrounding environment. It can avoid this problem by using 2D laser sensors which have become popular in indoor mobile robot navigation.

Several methods have been proposed for extracting lines from laser data such as hough transform, line regression method, line tracking algorithm and iterative-end-point-fit algorithm [Nguyen et al., 2005]. The HT algorithm has two drawbacks for extracting lines from 2D laser data, namely:

- It is difficult to define the size of the HT grid because the laser reading depends on the distance to the obstacles.
- It is considered complex and computation takes a relatively long time.

The line regression algorithm is used for map-based localization systems, in which the line parameters are transformed into the model space in a similar way to the HT algorithm, and then the adjacent line points are defined by applying the agglomerative hierarchical clustering algorithm [Siegwart., 1997]. A disadvantage of this method is that it is quite slow and complex to implement.

The line tracking algorithm is simple, and uses the total least square method to fit the laser point into lines. It finds a line from a laser cluster by constructing a line between the first two points, the next point is added to the current line, and then the line parameters are recomputed. If the new parameters of the line achieve the line condition; angle and distance thresholds, the point is added to the line, otherwise the last point of the current line becomes a start point [(Vandorpe et al., 1996b)].

This project utilizes the iterative-end-point-fit algorithm which is a simple and robust method for extracting lines from 2D laser data [HUSSON., 1997]. Before applying the wall extraction algorithm, the laser data is clustered to separate obstacles, which are classified as dynamic and static clusters. Thus, there is no need to calculate the distance between laser points during wall extraction, and also all dynamic clusters are removed which reduces the computation time of the wall extraction algorithm.

The following steps are implemented to define lines from a laser cluster of n points:

1. The first point p_0 is considered as a start point S_i (i is the wall index) and the last point p_n as the end point E_i of the line.
2. The distance between the current lines and point $p_{(n-i)}$ ($i = 1, 2..E - S$) is calculated and if it is greater than the threshold distance, the point $p_{(n-i)}$ is chosen as a new end point, otherwise the index i is increased.
3. If all points between the start point and end point have distances less than the threshold distance, the current line is considered between these two terminal points.
4. The end point of the current line is considered as the start point of the new line.

Figure 4.7 gives an example of a group of 7 points. The line extraction algorithm selects the first point as the start point $S_0 = p_1$, while the end point is selected as the last point in this group $E_0 = p_7$ as shown in Figure 4.8.a. The distance between the current line and the point p_6 is calculated, where it is > 0.2m which is the threshold distance. Therefore the point p_6 becomes the new end (see Figure 4.8.b). The procedure is repeated until point p_4 , where all the points between the start and end points of the line have distances smaller than the threshold (Figure 4.8.c). The last end point p_4 is chosen as the new start point $S_1 = p_4$, while the last point of the group p_7 becomes the last point of the new line $E_1 = p_7$. It can be observed from Figure 4.8.d that the line extraction method produces two lines.

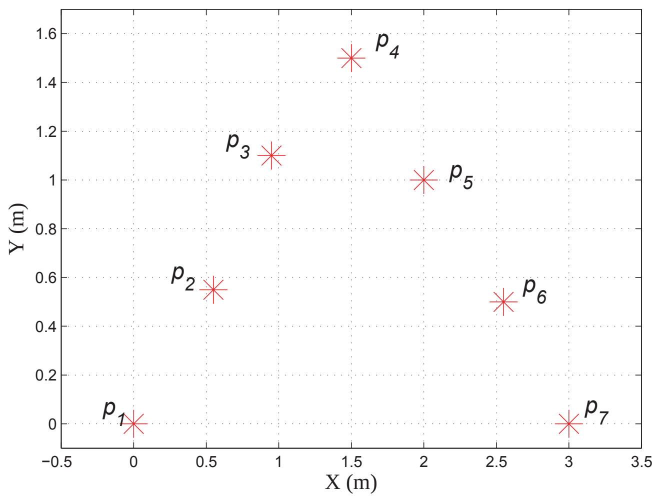


Figure 4.7: Group of points

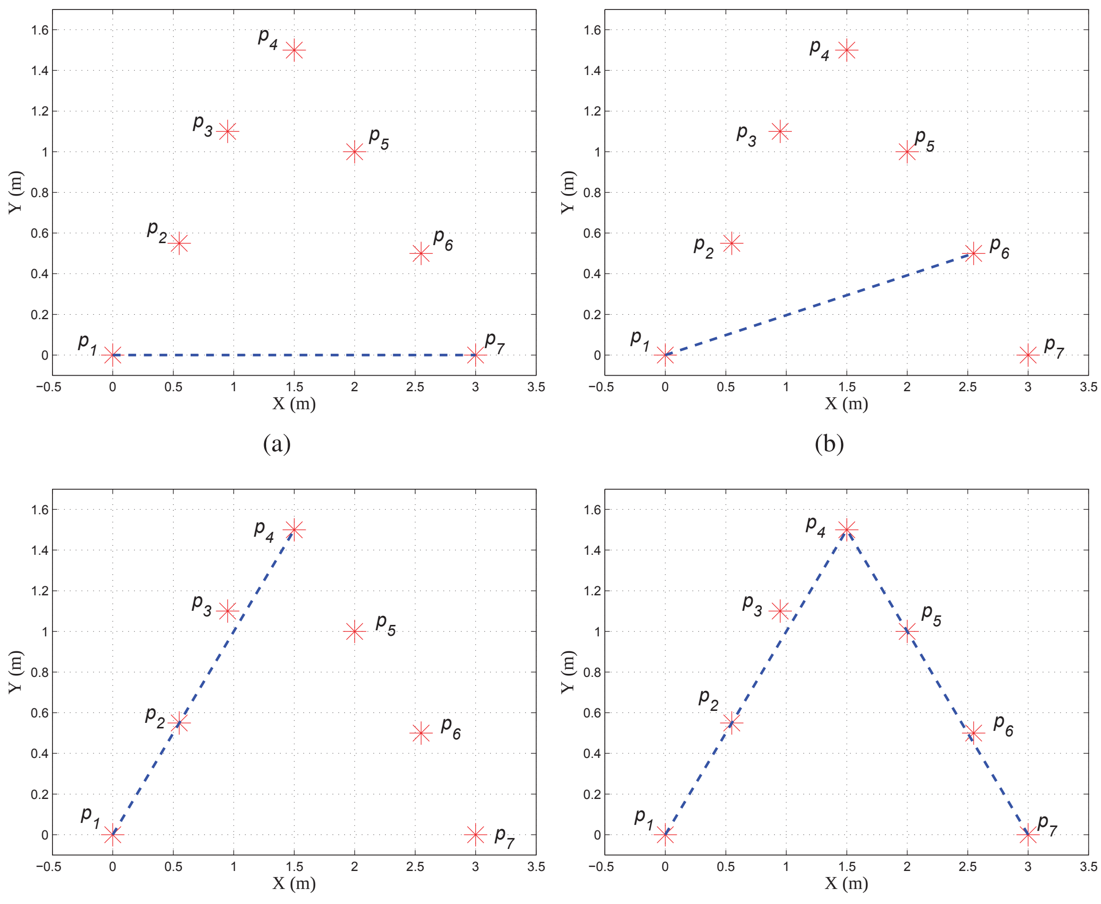


Figure 4.8: Line extraction implementation

4.5 Simulation Results

Simulation tests were undertaken using the Player/Stage platform and C++ programming language to verify the following issues:

- The global path planning algorithm is able to produce a safe path between the current known position of the mobile robot and its goal.
- Fatal collision may occur if the initial position is unknown and the robot is not programed to avoid obstacles.
- The localization system can estimate the current position of the mobile robot even if the initial position is unknown.
- The wall extraction algorithm can find the walls surrounding the robot in both situations; offline using robot map and online using laser data.

4.5.1 Global Path Planning

Consider the case in Figure 4.9, which shows a virtual robot map of several rooms, and the robot position and the goal position were chosen to be $(0,0)$ and $(6,5)$ respectively with respect to a global reference frame. The wavefront algorithm produces a global path including waypoints as shown in Figure 4.10, where the robot can move from a subgoal to the next subgoal without using sensor data to avoid obstacles.

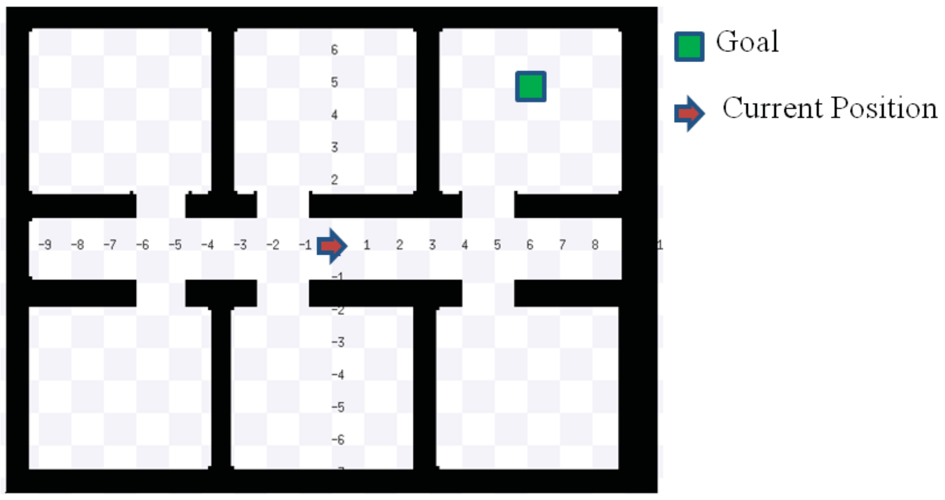


Figure 4.9: Workspace map for the wavefront algorithm

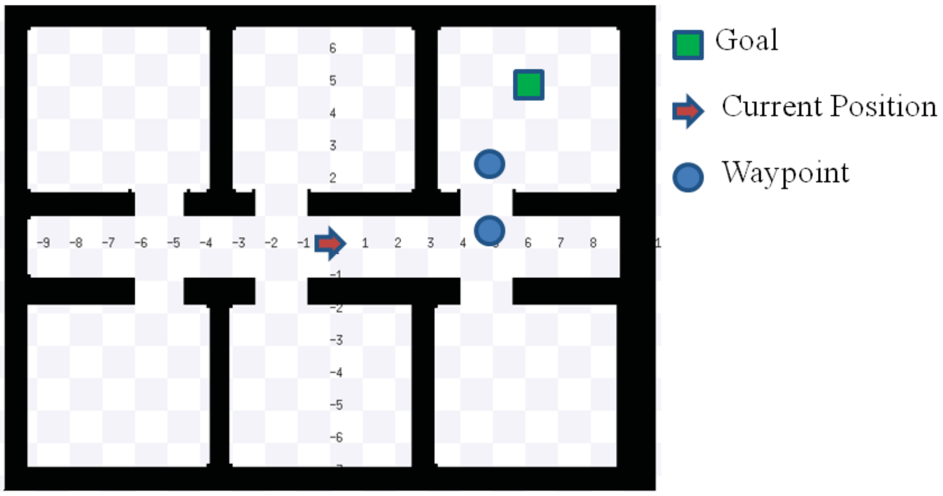


Figure 4.10: Waypoints produced by the wavefront algorithm

In this stage, just one behaviour, which is *go to goal* in the reactive layer, is used to carry out the global path planning, where the robot uses its odometry system to estimate its current position, while the angle θ_t and distance d_t between the current location of robot and the subgoal are calculated as follows:

$$\theta_t = \arctan(\frac{y_t - y_r}{x_t - x_r}) \tag{4.4}$$

$$d_t = \sqrt{(x_t - x_r)^2 + (y_t - y_r)^2} \tag{4.5}$$

where (x_t, y_t) and (x_t, y_t) represent the target and robot positions with respect to the global reference frame. The robot angular velocity can be calculated as follows:

$$\omega = K_\omega \triangle\theta \tag{4.6}$$

where K_ω is constant for steering, and $\triangle\theta$ is the difference between the robot angle θ_r and target-robot angle θ_t, which is written as:

$$\triangle\theta = \begin{cases} \theta_t - \theta_r & if\ \theta_t - \theta_r < 180^0 \\ 360 - (\theta_t - \theta_r) & else \end{cases} \tag{4.7}$$

The linear speed v is given as follows:

$$v = \begin{cases} 0 & if\ |\triangle\theta| > \theta_{thr} \\ v_{max} & if\ \ d_t > d_{thr} \\ v_{max}\frac{d_t}{d_{thr}} & else \end{cases} \tag{4.8}$$

where θ_{thr} and d_{thr} represent the angle and distance threshold respectively, and v_{max} is the maximum speed of the robot.

According to this scheme, the mobile robot stops at each subgoal and turns until $\triangle\theta$ becomes less than θ_{thr}, and then accelerates to its maximum speed. Figure 4.11 shows the implemented path from the initial position of the robot to its goal. It can be observed that the global path consists of three straight line segments, and the mobile robot was able to reach its goal without using any external sensor to avoid obstacles.

4.5.2 Localization Systems

Previously the initial position of the mobile robot was known, but if the initial position is changed to $(-3,-0.3)$ as shown in Figure 4.12, it can be observed that the new waypoint locations with respect to the world reference frame prevent the mobile robot from reaching its goal due to the fatal error in the estimated position of the robot. Consequently the robot will collide with an obstacle if it is not programmed to avoid obstacles as shown in Figure 4.13.

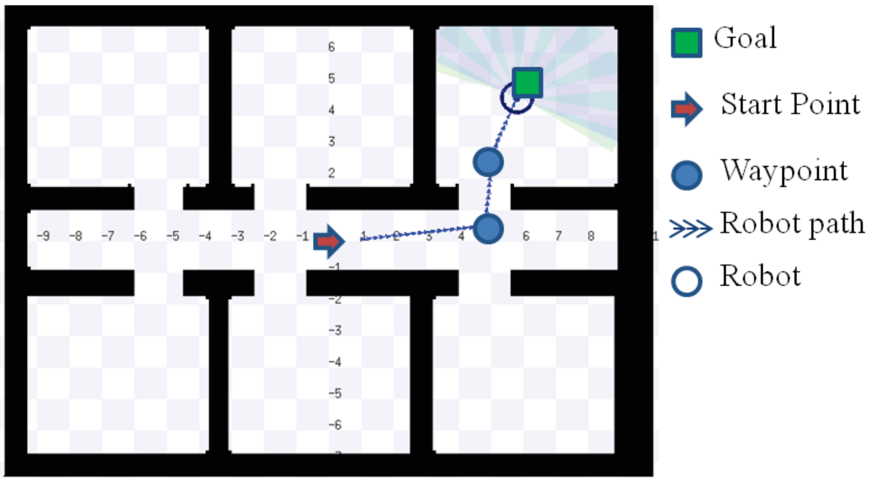


Figure 4.11: Global path planning implementation

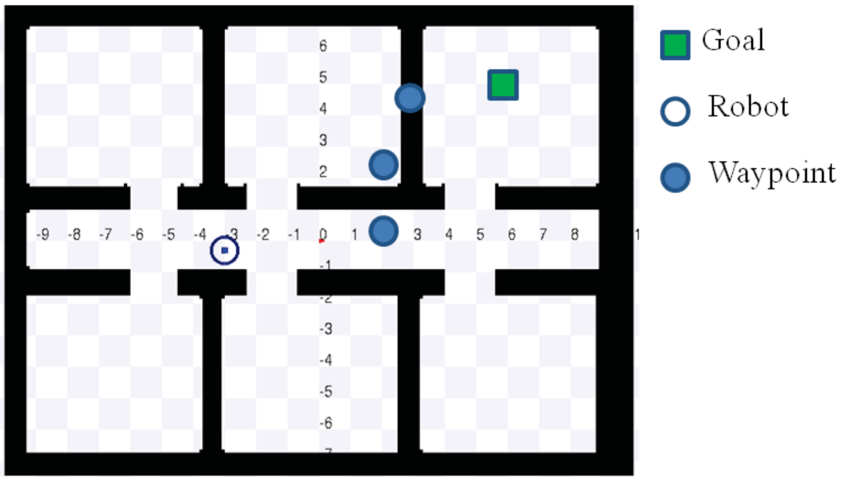


Figure 4.12: New waypoint locations

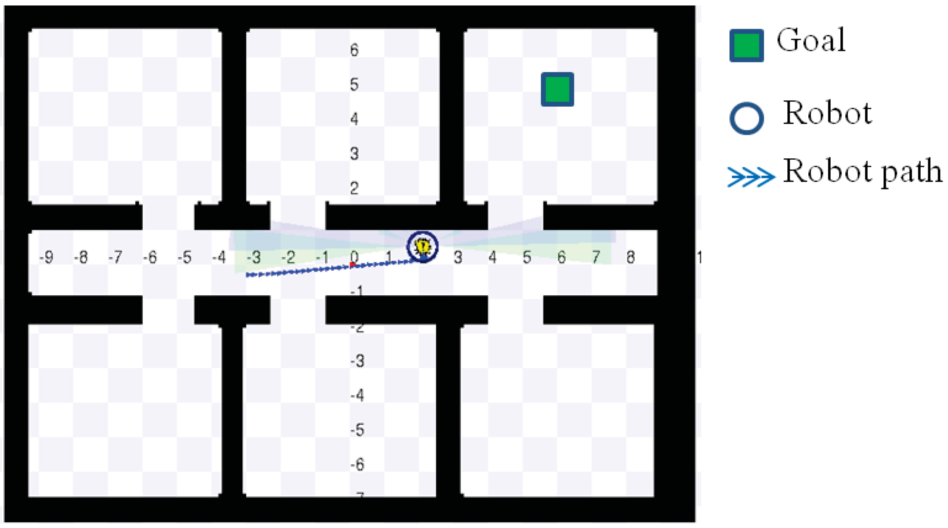


Figure 4.13: Collision caused by the localization error

To test the localization system, the AMCL localization system is incorporated to estimate the mobile robot position using the robot map and laser sensor, and the initial position of the robot was chosen to be as $(-3, -0.3)$ with an initial orientation of 30^0. The robot was able to safely reach its goal using AMLC as shown in Figure 4.14 and 4.15.

Figure 4.16 shows that the actual path started at the location (-3,0.3), while the estimated path started at the location (0,0). It can be observed that the estimated position of the mobile robot converged to its actual position, more details of which are shown in Figure 4.17, in which the initial position error was 3.1m at $t = 0$s, this increased gradually to reach a maximum error of 4.4m at $t = 4$s, and then decreased sharply to less than 0.15m after $t = 5$s. Figure 4.18

shows that the error in the estimated direction was 30^0 at the time $t = 0$s , which increased to a maximum error of 180^0 when $t = 1$s, then decreased sharply to -144^0 at the time $t = 4$s, and finally became less than 5^0 after $t = 6$s.

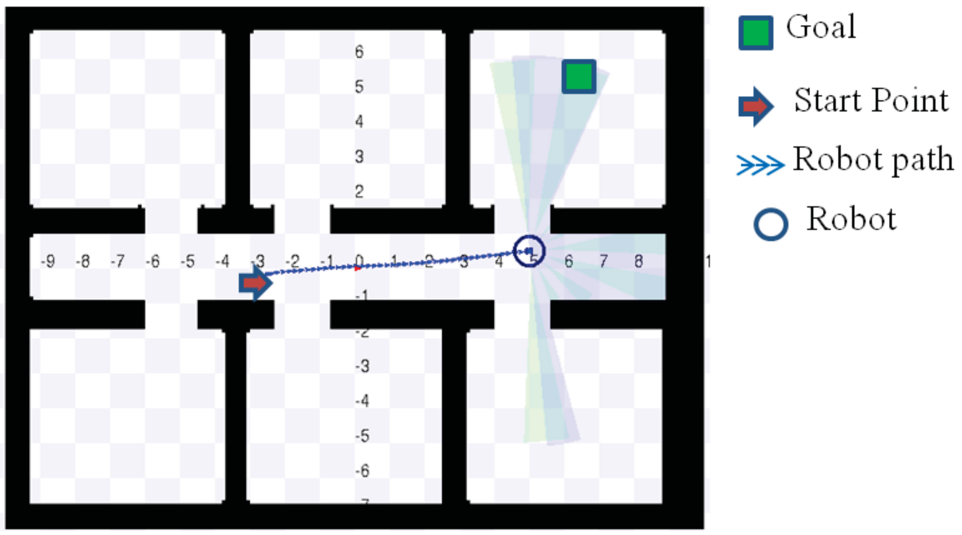


Figure 4.14: First part of the robot path using the AMCL algorithm

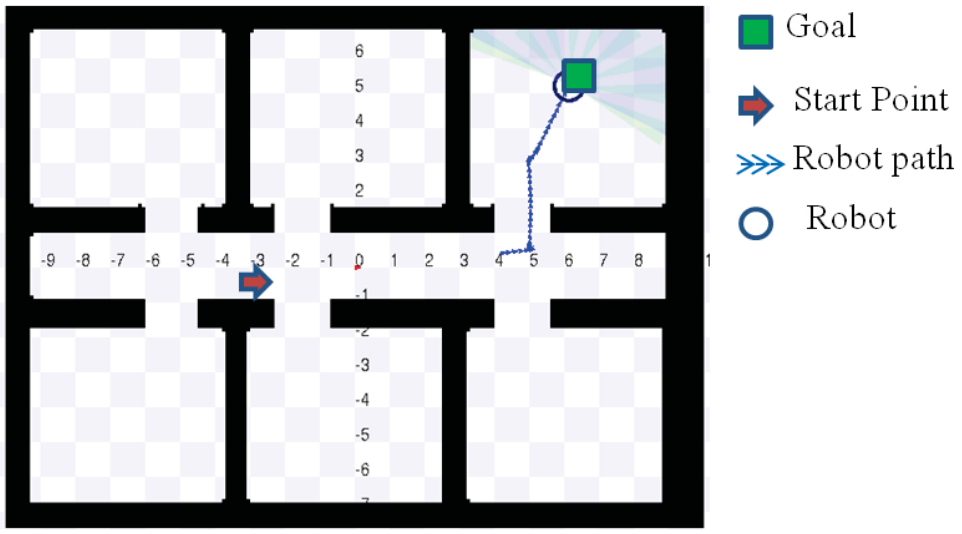


Figure 4.15: Second part of the robot path using the AMCL algorithm

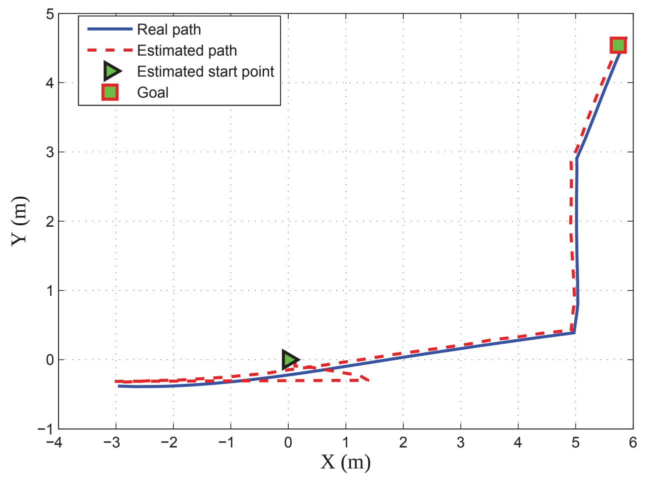


Figure 4.16: Actual and the estimated robot path

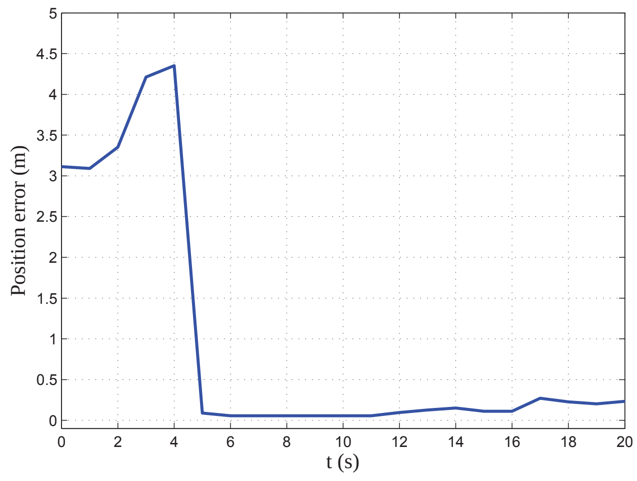


Figure 4.17: Error in the estimated position

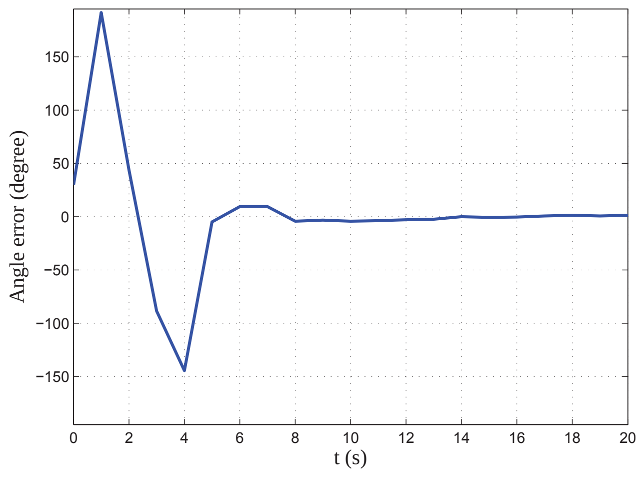


Figure 4.18: Error in the estimated direction

4.5.3 Wall Extraction

Figure 4.19 shows the offline extracted walls of a workspace map using the HT algorithm, in which each wall is assigned a number, start point and end point in the world reference frame. If the mobile robot is located at position $(2,0)$, the wall extraction algorithm defines the surrounding walls shown as blue lines in Figure 4.20. However, any error in the estimated position of the mobile robot will cause errors in the locations of the surrounding wall, and to avoid this problem the laser data can be used.

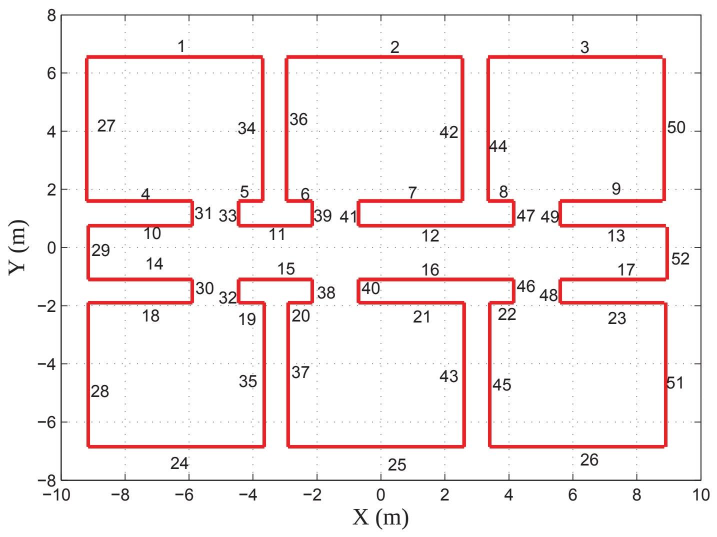


Figure 4.19: Wall extraction offline

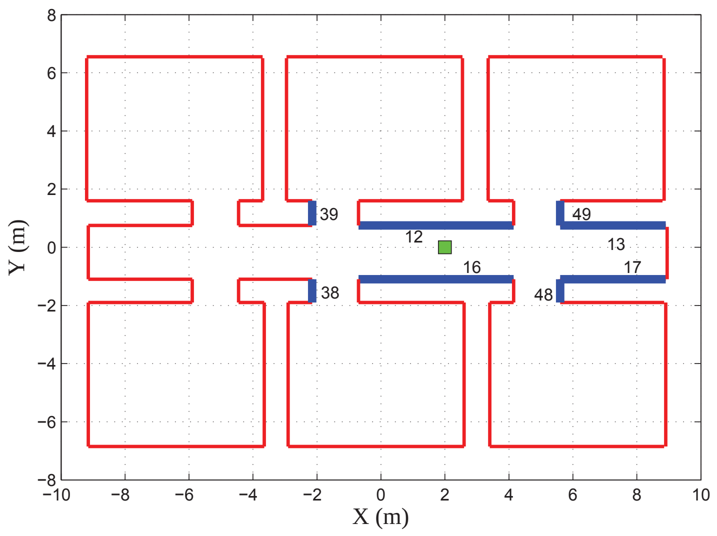


Figure 4.20: surrounding walls

Figure 4.21 shows the laser data returning from surrounding walls at the position $(2,0)$, where the cluster algorithm produced four clusters. The iterative-end-point-fit algorithm was applied and extracted six straight lines as shown in Figure 4.22. It can be observed that each of the first and fourth clusters included one wall, while each of the second and third clusters included two walls. However, the offline wall extraction gives more information about the surrounding walls, but it requires an accurate workspace map and robust localization system, and also the computation size increases as the number of walls increases.

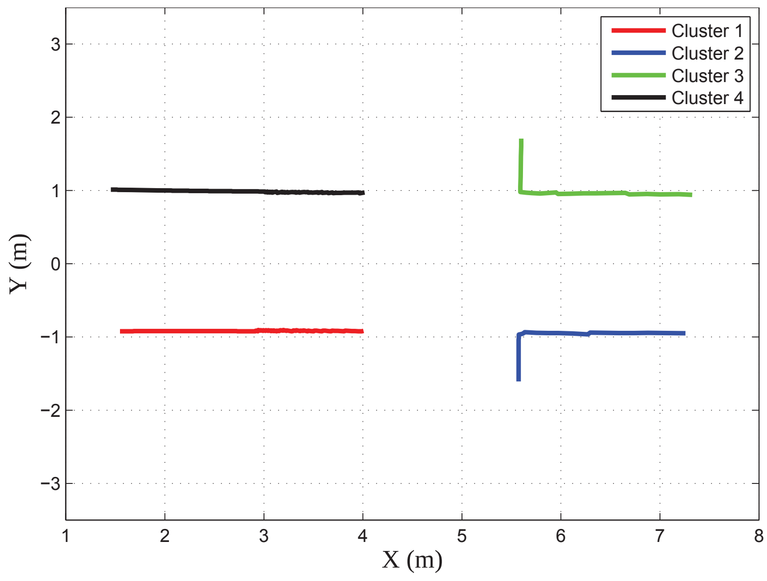


Figure 4.21: Collected laser data

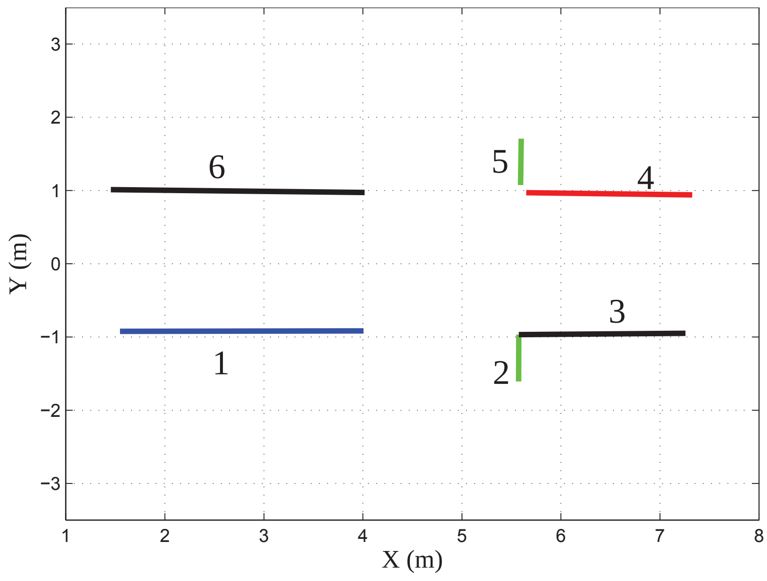


Figure 4.22: surrounding walls

4.6 Summary

The simulation results show that the wavefront algorithm produces a safe path between the current position of the mobile robot and its goal, where the robot was able to reach its target using only the *go to goal* behaviour. However, an error in the estimated position caused a collision and the robot was unable to find the right path to its goal. To solve this problem, the AMCL algorithm was used to produce a robust localization system, where the estimated position of the mobile robot converged to the actual position. In this chapter, two methods were used to extract the walls around the mobile robot. The HT algorithm was used to extract all walls in the robot map offline, and then the walls around the robot were detected depending on the location of the robot. Although this method gives more information about the surrounding walls, it requires a known workspace map and accurate localization system, while the iterative-end-point-fit algorithm can be used to extract walls form 2D laser data online without using the workspace map.

Theoretically, the robot can reach its goal by following the long path without using sensor data to avoid obstacles, but the global path planning does not considered all obstacles in the workspace such as tables or other mobile robots. Furthermore, any error in the localization system may cause a fatal collision, therefore the robot must use its sensors to avoid static obstacles such as wall and table and moving obstacles such as humans and other robots.

Chapter 5

Reactive Level

The global path using the wavefront algorithm produces a safe path for successful navigation, which requires an accurate map and a robust localization system. However, errors in the localization system can lead to collisions, and also if there is any small change in the workspace the robot has to first update its map, and then execute the global path planning algorithm again, which makes the robot navigation slow and more complex. This chapter focuses on the reactive level, which attempts to overcome the above deficiencies in the deliberative level, and tries to maximize the robot speed to the goal and to minimize the collision risk with obstacles. It includes six behaviours, namely *go to goal*, *obstacle avoidance*, *unobserved obstacle avoidance*, *human avoidance*, *follow the leader* and *emergency stop* (Figure 3.1). The *human avoidance* behaviour will be discussed in the next chapter.

5.1 *Go To Goal* Behaviour

This behaviour utilizes the current position of the robot obtained from the localization system and the goal position to define optimal speed and direction to the target. In the global reference frame, the angle θ_t to the target and distance d_t are given as follows:

$$\theta_t = \arctan(\frac{y_t - y_r}{x_t - x_r}) \tag{5.1}$$

$$d_t = \sqrt{(x_t - x_r)^2 + (y_t - y_r)^2} \tag{5.2}$$

where (x_t, y_t) and (x_r, y_r) represent the target and robot positions with respect to the global reference frame.

The desired speed to the target is given as:

$$V_t = \begin{cases} v_{max} & if\ d_t > d_{th} \\ v_{max}\frac{d_t}{d_{th}} & otherwise \end{cases} \tag{5.3}$$

where v_{max} and d_{th} denote the robot's maximum speed and threshold distance respectively.

5.2 *Obstacle Avoidance* Behaviour

The velocity obstacle (VO) algorithm is used to avoid static and dynamic obstacles [Fiorini and Shiller, 1998]. The VO approach is considered an easy and simple method to avoid dynamic obstacles, where the collision cone principle is used to detect a situation of collision between two circular-shaped objects. The VO approach has two challenges, the first of which is to extract collision cones for non-circular objects from sensor data, where simply applying circle fitting methods produce large and inaccurate collision cones, especially for long obstacles such as walls. The second challenge is that the mobile robot cannot sometimes move to its goal if all of its velocity vectors to the goal are located within collision cones.

As mentioned in Chapter 2, many studies have implemented the VO algorithm using simulation with circular obstacles and known velocities. However, in the real world, the robot has to extract the shapes of the obstacles from sensor data, grow them using the robot radius and then extract the collision cones. In the VO algorithm, the robot R is treated as a point by enlarging the circular obstacle O with the robot radius as shown Figure 5.1, in which the grown obstacle $\hat{O}$ is considered to be static, while the robot moves with the relative speed. If the relative speed v_{ro} passes through the grown obstacle or within the collision cone, this implies that there will be a collision, and therefore the robot should control its relative speed outside the collision cone.

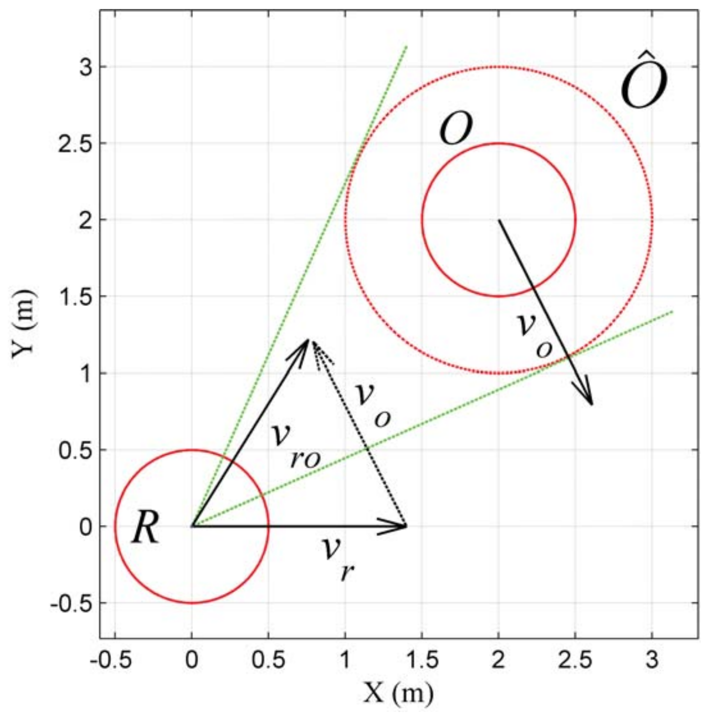


Figure 5.1: VO method

5.2.1 Collision Cone

The VO method supposes that the robot and obstacle have circular shapes, where the boundary of the collision cone is defined by the two tangents of the grown obstacle circle $\hat{O}$ to the centre of the robot R (Figure 5.2). Angles of the left and right tangents (θ_L and θ_R) of the tangents with respect to the robot reference frame are given as follows:

$$\theta_R = -\arctan(\frac{r}{L}) + \phi \tag{5.4}$$

$$\theta_L = \arctan(\frac{r}{L}) + \phi \tag{5.5}$$

where r , L and ϕ refer to the grown obstacle radius, tangent length, and obstacle angle with respect to the robot reference frame.

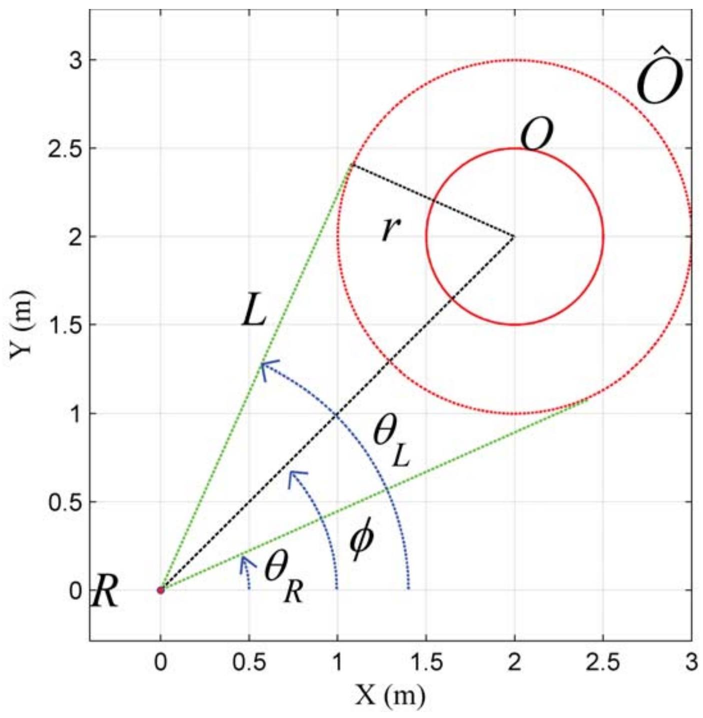


Figure 5.2: Collision cone

If each point p_i in the obstacle's circumference is grown by the robot radius r_r as shown in Figure 5.3, the outer boundaries of the grown points produce the grown obstacle circle $\hat{O}$. If the left and right tangent angles θ_{ri}, θ_{li} of the grown point $\hat{p}_i$ are calculated using Equation 5.4 and 5.5, the collision cone boundaries of the obstacle can be determined from the tangent of the minimum right angle and the tangent of the maximum left angle with respect to the robot reference frame. Hence, a method is proposed to find the actual collision cone of an obstacle from laser data using the three following steps:

- Each laser point is enlarged by the robot radius.
- Determine the left tangent angles $[\theta_{l1},,,\theta_{ln}]$ and right tangent angles $[\theta_{r1},,,\theta_{rn}]$ of the enlarged laser points.
- The tangent of the maximum left angle is selected as the left boundary of the collision cone, while the tangent of the minimum right angle is considered as the right boundary.

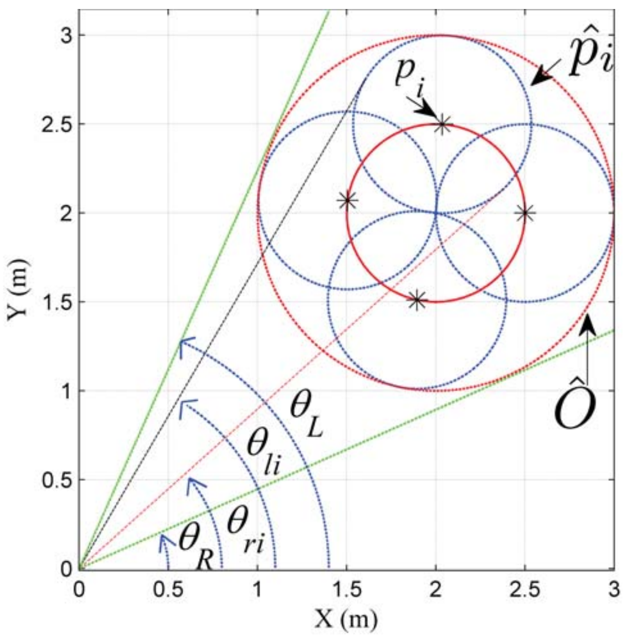


Figure 5.3: Grown obstacle

5.2.2 Influence of Collision Time

The output of the VO approach has two situations for the robot velocities: 0 for the collision velocity and 1 for collision-free velocity. Consider the case in Figure 5.4, where the robot must navigate within an indoor environment in which there are two walls and two static obstacles. The robot cannot move to its goal because all of its velocities to the goal are located within collision cones. Whilst there are many possible paths between the robot and its target, one of them is shown as a red path in Figure 5.4. However, the robot will be able to reach its target if both collision time and obstacle size are considered. To meet this requirement, a cost function is proposed where the collision-free velocity are given a weight of 1, and collision velocity is given as follows:

$$W_{obs} = \begin{cases} 0 & if\ t_c < t_{ct} \\ exp(-\frac{a}{t_c})(1 - b\delta t_c) & otherwise \end{cases} \tag{5.6}$$

$$\delta = \min(\theta_L - \theta, \theta - \theta_R) \tag{5.7}$$

where t_c refers to the collision time, a and b are constants, t_{ct} is the critical collision time, θ

is the robot direction, and δ represents the minimum angle between the relative speed and the collision boundaries, therefore, the collision risk within the collision cone increases as the relative speed increases and becomes close to the obstacle centre.

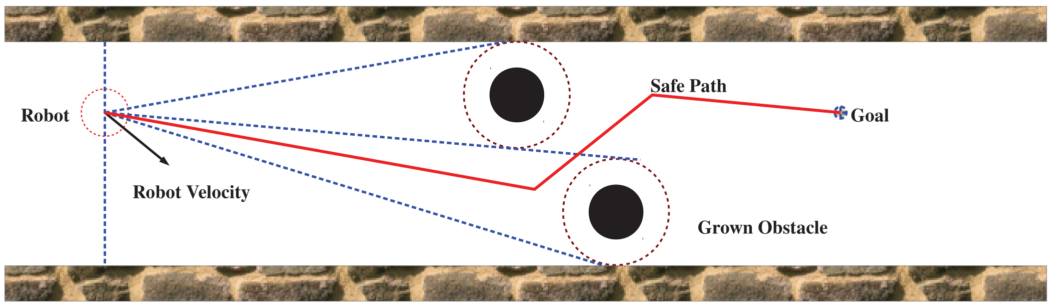


Figure 5.4: The robot cannot reach its target according to the VO method

5.2.3 Static and Dynamic Obstacle Classification

The PDBS clustering method developed in [Santos et al., 2003] is used to segment laser data, and then the extended particle filter algorithm is used to estimate obstacle speeds, and therefore as the number of obstacles increases, the number of particle samples assigned to each obstacle decreases. However, the tracking algorithm must be used to estimate the speed and direction of moving obstacles, where there is no need to apply the particle filter algorithm for static obstacles. Also sometimes the estimator defines fuse speeds of static obstacles because the laser range finder does not provide sufficient information about the whole workspace, due to the following reasons:

- Some areas are occluded behind obstacles.
- Some areas are unknown because of the limited range of the sensor.
- Some areas disappear behind the robot as it moves due to the limited angular range.

To explain this more fully, consider the situation in Figure 5.5, where a robot R and obstacle O are moving at the same speed of 0.24m/s in the same direction. Figure 5.6 shows two laser scans, where the black colour represents the laser data at the sample time $n-1$, while the red colour represents the laser data at sample time n. The clustering algorithm produced four laser clusters, and the tracking algorithm shows that the four clusters are dynamic, although only one of them is a dynamic obstacle whilst the rest are static. Therefore the particle filter samples were distributed to track four obstacles.

Hence, it is crucial to define static and dynamic obstacles before applying the tracking algorithm. The occupancy grid can be used to classify the laser cluster as static or dynamic, in which the grid cell has three situations, namely *occupied* (1), *unknown* (0.5), and *empty* (0), and also the eight childbearing cells are used to address the uncertainty in the measurement using a simple averaging filter.

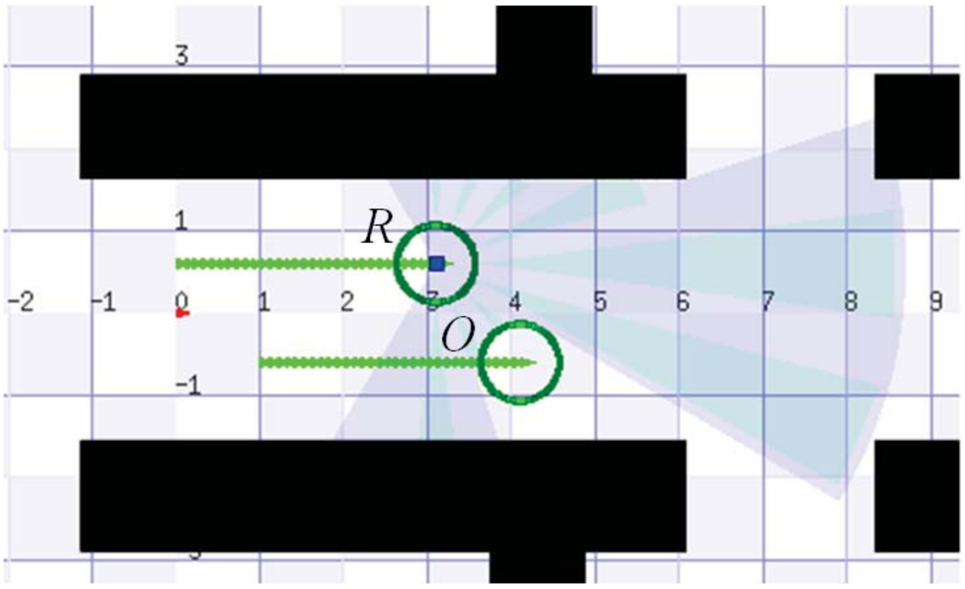


Figure 5.5: Robot and obstacle moving in the same direction and speed

To classify a laser cluster as static or dynamic, two consecutive laser scans are considered and the following steps are implemented:

1. Cluster the laser data.
2. The data association is implemented depending on the nearest neighbour method.
3. Each occupied cell in the first map is compared with that from the second map.
4. A threshold value is applied to find dynamic cells in each cluster.
5. For each cluster, if the ratio of the dynamic point number to the total point number is greater than a minimum certain value, then the obstacle is classified as dynamic.

6. Each occupied cell in the second map is compared with that in the first map, and steps 4 and 5 are repeated.

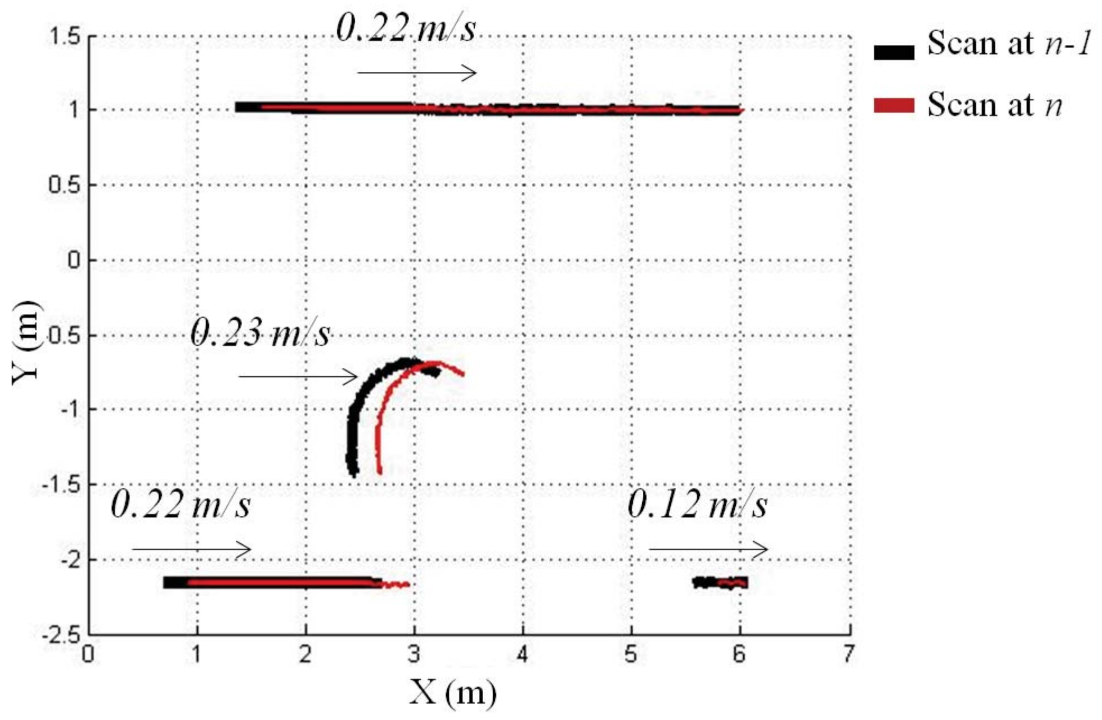
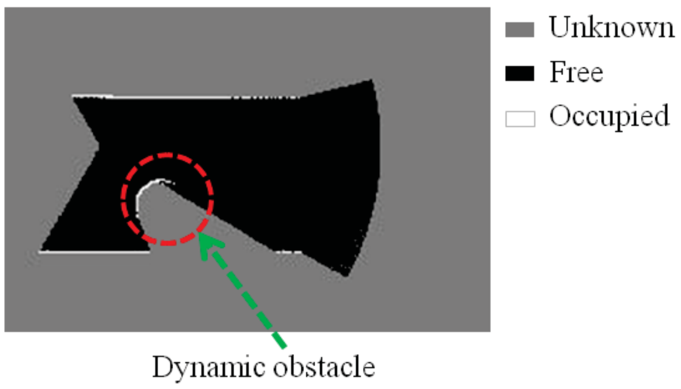


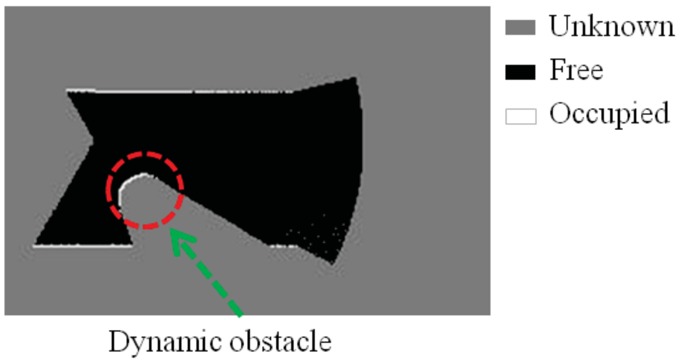
Figure 5.6: Estimated speeds of laser clusters

Figure 5.7 shows two grid maps of consecutive laser scans, in which the black, white and grey areas represent the free space, laser data and unknown area respectively. A comparison of the two images produces an image shown in Figure 5.8.a, where it can be observed that several cells of the static obstacle were classified as dynamic, and these can be removed by applying the averaging filter (see Figure 5.8.b).

The maximum reading of the laser sensor used in this project is 5.6m, and therefore the grid map must cover an area of 5.6m *x* 5.6m. Suppose that the cell size is chosen as 4mm *x* 4mm, which generates an image of 140 *x* 140 pixels. Each cell must be firstly initialized and checked to define its situation, and at every step the two grid maps are subtracted and filtered. Consequently the classification procedure is relatively time consuming. This problem is solved by considering only the cells occupied by the laser data.

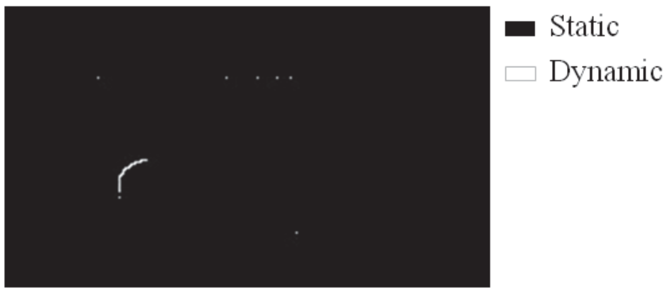


(a) Laser scan at $n-1$

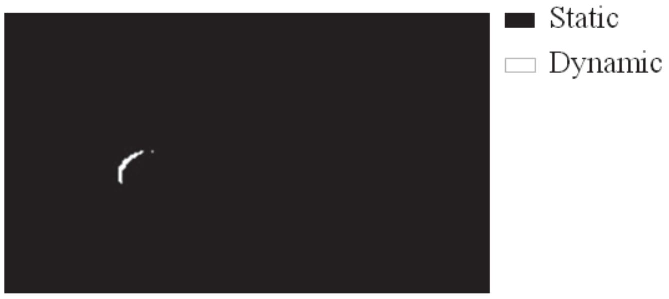


(b) Laser scan at n

Figure 5.7: Consecutive grid maps



(a) Dynamic cells



(b) Applying the averaging filter

Figure 5.8: Extracting dynamic clusters

5.3 *Unobserved Obstacle Avoidance* Behaviour

Unobserved moving objects may appear suddenly in the robot path, particularly when the robot crosses a corridor or passes an open doorway. Therefore the robot has to consider these obstacles and take suitable action to minimize the collision risk with them. The *virtual obstacle principle* is proposed to avoid potentially unobserved moving objects, where a virtual circular obstacle is created at the starting point of each open door threshold and corridor crossing as shown in Figure 5.9, in which the robot has to move away from the door threshold to avoid the virtual obstacle, but sometimes there is no enough space for the robot to move away from the door threshold. In the latter case, the robot must reduce its speed to minimize the collision risk with unobserved obstacles. To meet this requirement, the virtual obstacle radius is modified depending on the robot speed using the following equation:

$$r_0 = R_0 v_r \tag{5.8}$$

where R_o is constant, and v_r represents the robot speed

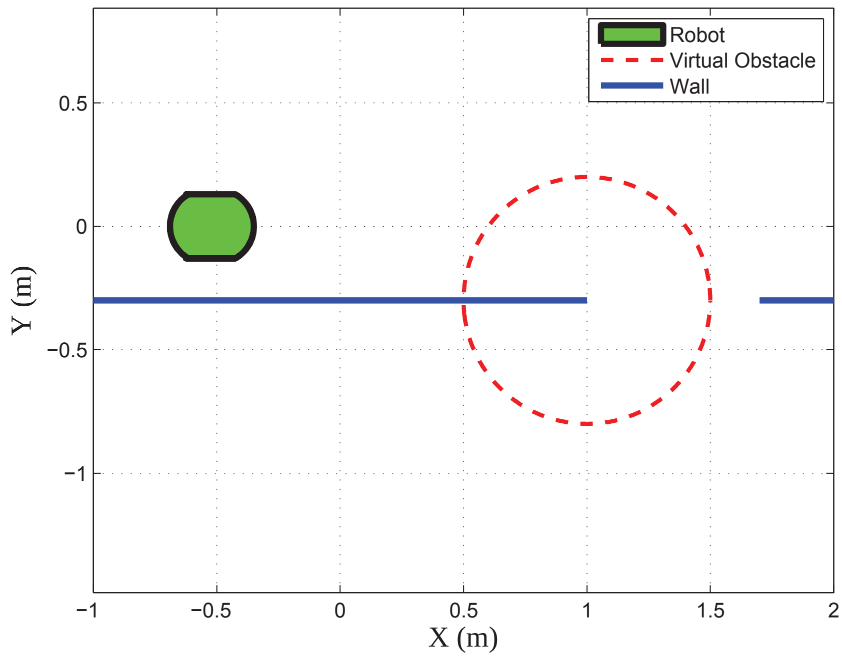


Figure 5.9: Virtual obstacle principle

In this behaviour, the robot velocity is weighted according to the following rule:

$$W_{unobs} = \begin{cases} 0 & if\ t_c < t_{th} \\ 1 & d_{ro} > D_{th} \\ \exp(-\frac{a_{unobs}}{t_c}) & (1 - b_{unobs}.\delta.t_c)\ else \end{cases} \tag{5.9}$$

where a_{unobs}, b_{unobs} , t_{th} and D_{th} are constants, d_{ro} and t_c represent the distance between the robot and the virtual obstacle, and collision time respectively, and δ is calculated using Equation (5.7).

5.4 *Follow the Leader* Behaviour

In congested areas such as shopping malls, humans sometimes prefer to walk behind someone going in the same direction to ease their passage. This behaviour can be considered to a useful aid to mobile robot navigation in a cluttered environment. In this study, the robot classifies a moving obstacle as a suitable leader to follow if the obstacle meets the following criteria:

- The distance between the robot and moving target is below a minimum threshold value (1m).
- The difference between the angles of the moving obstacle and target is below a threshold value (10^0).
- The velocity of the moving obstacle (v_o) is such that $v_o > max(v_r.W_v)$, where W_v is a velocity weight.

In selecting this behaviour, the *go to goal* behaviour is deactivated, where the orientation and distance to the moving target are calculated using Equation (5.1) and (5.2), while the desired robot speed is calculated as follows:

$$v_t = \begin{cases} v_{max} & if\ d_t > d_{max} \\ v_{max}\frac{d_t - d_{min}}{d_{max} - d_{min}} & else \end{cases} \tag{5.10}$$

where v_{max} , d_{max} and d_{min} denote the robot's maximum velocity, maximum and minimum distance thresholds respectively.

5.5 *Emergency Stop* Behaviour

In some situations, the distance between the robot and obstacle can become very small. However, sometimes all weights of the robot speed produced by the o*bstacle avoidance*, *human avoidance* and *unobserved obstacle avoidance* behaviours become zero which means the robot has no collision-free velocity. Consequently, the *emergency stop* behaviour should be activated and the robot must immediately be brought to a stop, and then it reverses until the emergency conditions disappear. The inputs of the *emergency stop* behaviour include the sensor data and the maximum weight of the robot speed, where this behaviour becomes active if the minimum distance to the obstacle is below a minimum specific threshold or the maximum weight of the robot velocity is zero.

5.6 Coordination Model

With reference to Figure 3.1, the main challenge for the coordination model block is to fuse the outputs of the behaviours and then generate control commands to minimize the collision risk and maximize the robot speed to its goal. To achieve this requirement, the robot velocity is weighted against the output of the *Go to Goal* behaviour and those computed from the *obstacle avoidance*, *human avoidance* and *unobserved obstacle avoidance* behaviours. The weight of the robot velocity is calculated according to the following function:

$$W_v = min(W_{unobs}, W_{hum}, W_{obs}) \tag{5.11}$$

where W_{unob}, W_{obs} and W_{hum} represent the robot velocity weights coming from the *unobserved obstacle avoidance*, *obstacle avoidance* and *human avoidance* behaviours. To maximize the speed to the target and minimize the collision risk the following weighting function is applied :

$$W = W_v * (a1 + cos(\theta_t - \theta))(a2 - |V_t - v|) \tag{5.12}$$

where a_1 and a_2 are constant.

The robot speed v_r and direction θ_r which achieve the maximum weight are chosen to produce control commands for angular velocity ω_c and linear velocity V_c as follows:

$$\omega_c = k_w * \theta_r \tag{5.13}$$

$$V_c = \begin{cases} v_r & if\ v_r < V_{cmax} \\ 0 & if\ 0 > V_{cmax} \\ V_{cmax} & otherwise \end{cases} \tag{5.14}$$

$$V_{cmax} = v_{max} - k_v * \frac{\omega_c}{\omega_{max}} \tag{5.15}$$

where k_w and k_v are constants, and v_{max} and ω_{max} represent the maximum linear and rotational velocities of the robot.

5.7 Simulation Results

The maximum speeds of the robot is limited to 0.5m/s and 100degree/s. The parameters in Equation 5.6 and 5.9 are selected as $a = a_{unobs} = 3s$, $b = b_{unobs} = 0.5rad^{-1}s^{-1}$ and $t_{ct} = t_{th} =$ 1s, while a1 and a2 of the weighting function described by Equation 5.12 are chosen to be 0.5 and 0.6 respectively.

5.7.1 Obstacle Collision Cone

The proposed collision cone finding method in this study is compared with two circle fitting algorithms proposed in [Gander et al., 1994; Vandorpe et al., 1996b]. Figure 5.10 shows a virtual irregular shaped obstacle and projected laser lines, with an angular resolution of 5°. The returned laser points are given in Table 5.1. Figure 5.11 shows the fitted circles; circle C1 is produced by applying the method proposed in [Vandorpe et al. 1996b], while circle C2 is produced using the method proposed by [Vandorpe et al., 1996a]. It can be observed that circle C1 is relatively large (diameter of 1.34m) and will consequently produce a very large collision cone. Hence, a comparison is made between the collision cones produced by circle C2 and the algorithm developed in this study. The calculated radius of circle C2 is 0.36m, which is enlarged by the robot radius (0.5m), and then Equation 5.4 and 5.5 are applied to find the collision cone boundary; the resulting angle of the collision cone of circle C2 is 70º. However, it can be noted from Figure 5.12 that the cone does not touch the grown laser points. For the proposed collision cone finding method in this study, Table 5.2 lists the calculated tangent angles of each enlarged laser point, of which the second point has the maximum left angle and minimum right angle, where these angles define the collision cone. Thus, it is not always correct to define the collision cone based on the first and last points of the laser cluster. The resulting angle of the collision cone defined by the proposed method is 60º (see Figure 5.13), and the collision cone touches the grown obstacle at two points. Clearly the proposed method produces a collision cone which is more accurate and smaller than that produced by either of the circle fitting methods.

Table 5.1: Collected laser points

laser point	1	2	3	4
Angle(degree)	40	45	50	55
Distance (m)	1.4	1	1.14	1.8

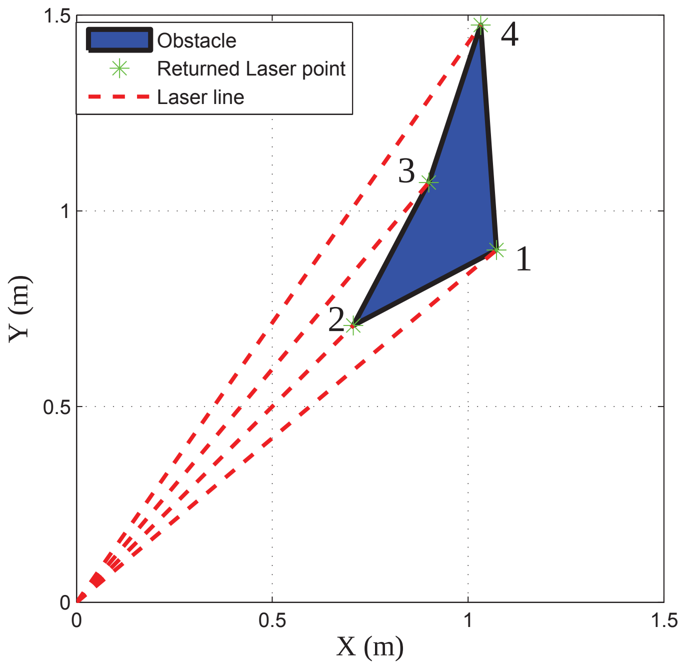


Figure 5.10: Laser data returned from a non-circular obstacle

Table 5.2: Collision cones subtending angles

laser point	1	2	3	4
left angle(degree)	60.1	75	70.1	71.1
Right angle(degree)	19	15	29	38.8

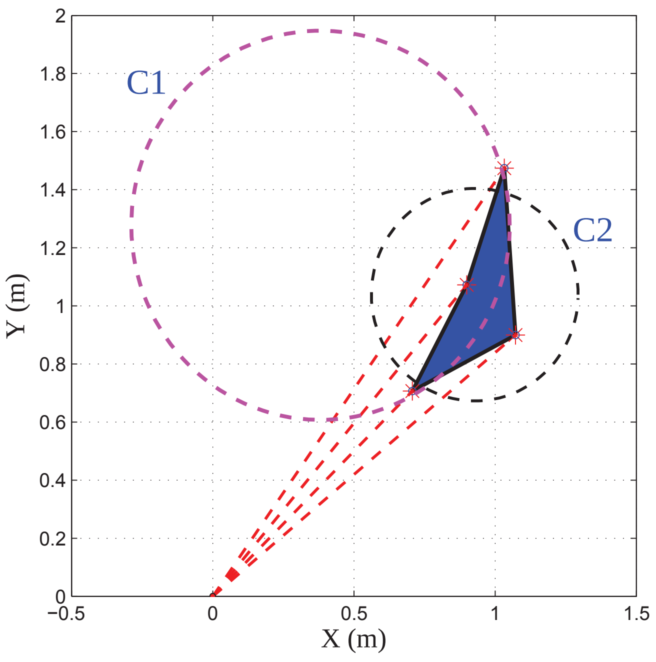


Figure 5.11: Fitted obstacle circles

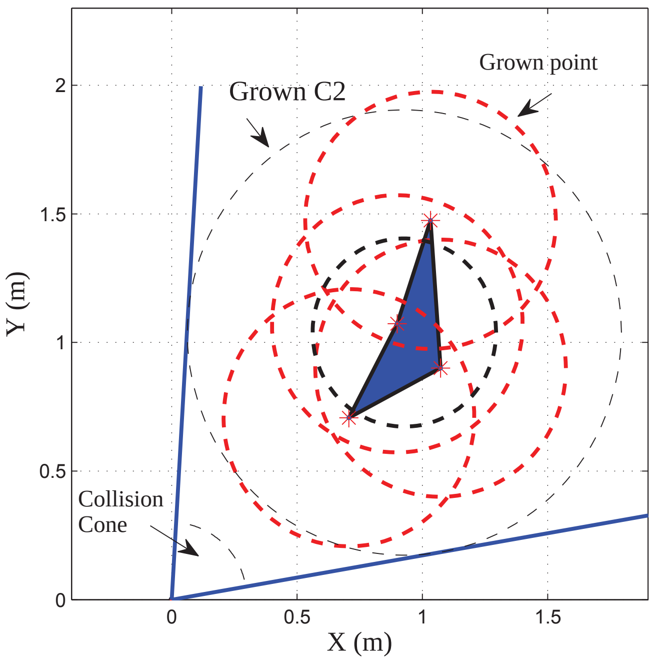


Figure 5.12: Collision cone of circle C2

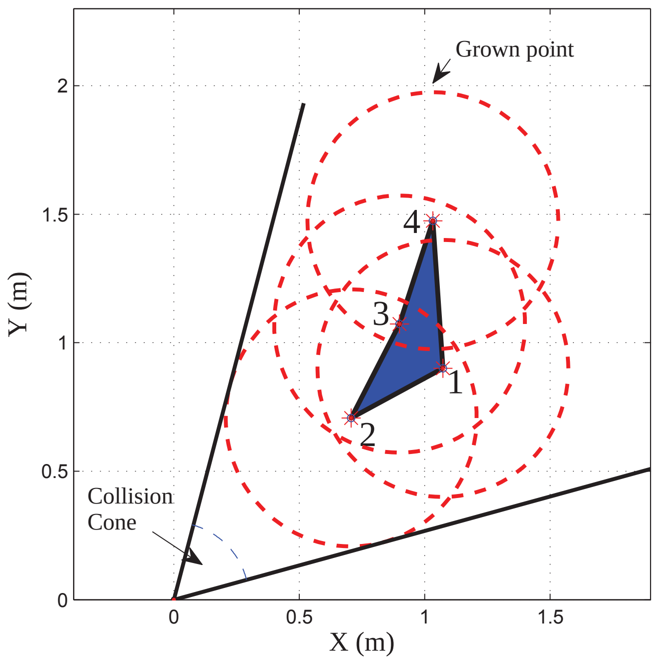


Figure 5.13: fitted obstacle circles

5.7.2 Obstacle Avoidance

5.7.2.1 Avoiding Static Obstacle

The Player/Stage simulator is used to model this behaviour, with the robot's task is to navigate from its current point $(0, 0)$ to its target $(7, 0)$ and to avoid collision with a static obstacle which prevents the robot from executing a straight line to its target. Figure 5.14 shows the resulting path to avoid static obstacles using the reactive model. It can be observed that the robot almost immediately begins to take avoiding action and moves away from the obstacle, negotiates the gap between the wall and obstacle, and then proceeds towards its goal taking 18s to complete its task.

5.7.2.2 Re-plan The Global Path

This simulation test aims to show that although the reactive level has fast response to implement a global path which is initially planned by the deliberative level, but sometimes it is trapped in local minima caused by a change in the robot workspace. Hence, the workspace map must be updated, and then the global path planning algorithm is required to regenerate the global path based on the new map and current robot position. Assume the case shown in Figure 5.15 where the robot has to navigate in an indoor environment to reach its goal(2,-3). The wavefront algorithm initially generates waypoints listed in Table 5.3. It can be observed Figure 5.16 that the robot is able to implement successfully the global path. When the workspace is changed as shown in Figure 5.17, in which the robot stops to avoid collision and the reactive level is enable produce successful navigation. In such case, the workspace map is updated based on the laser date (see Figure 5.18) and the global planning model in the deliberative level re-plans a global path which is defined in Table 5.4. Figure 5.19 shows that the robot implements the new global path and successfully researches its goal.

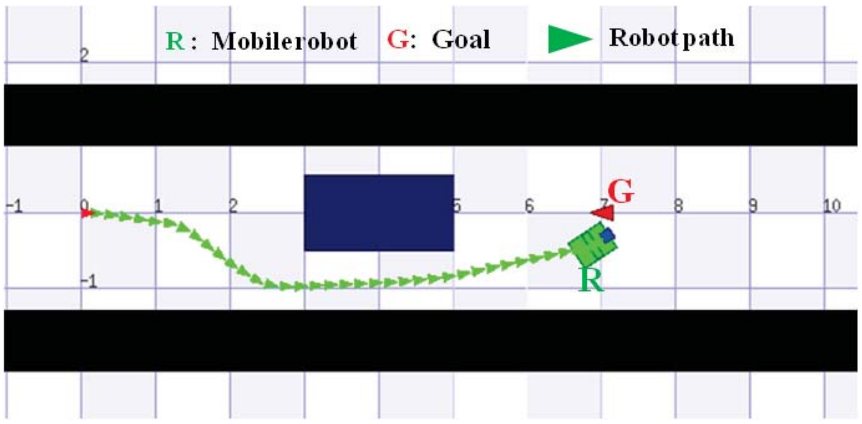


Figure 5.14: Static obstacle avoidance using the reactive model

Table 5.3: Waypoints of the global Path

Waypoint index	(x(m),y(m))
1	(-4.50,+0.00)
2	(+1.75,-0.25)
3	(+2.00,-3.00)

Table 5.4: Waypoints of the new global Path

Waypoint index	(x(m),y(m))
1	(1.92, -1.42)
2	(2.25, -0.17)
3	(4.15, -0.25)
4	(4.07, -4.15)
5	(2.25, -4.15)
6	(2,-3)

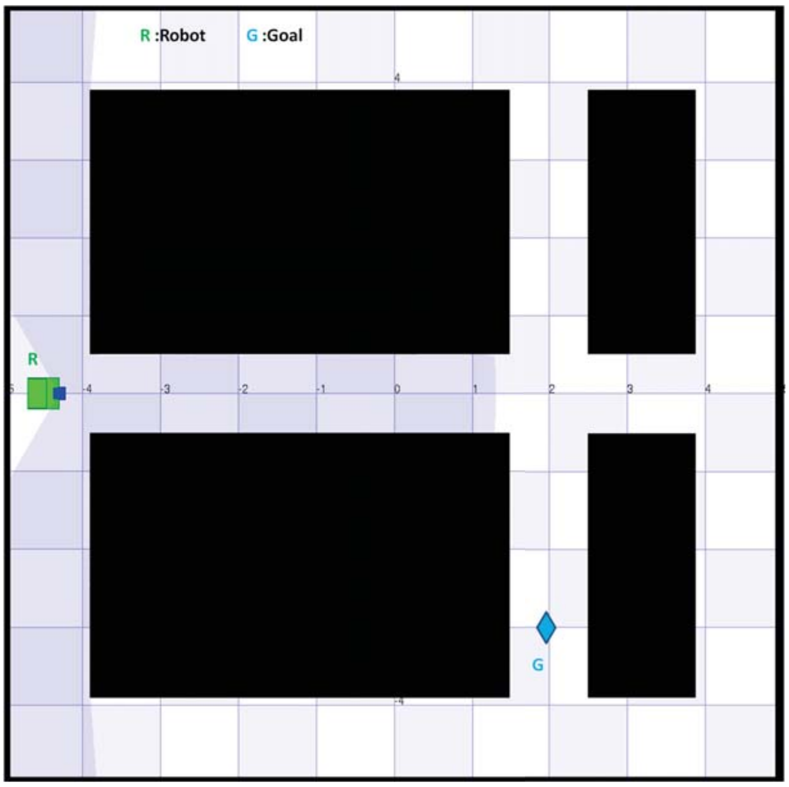


Figure 5.15: Robot successfully reaches its goal

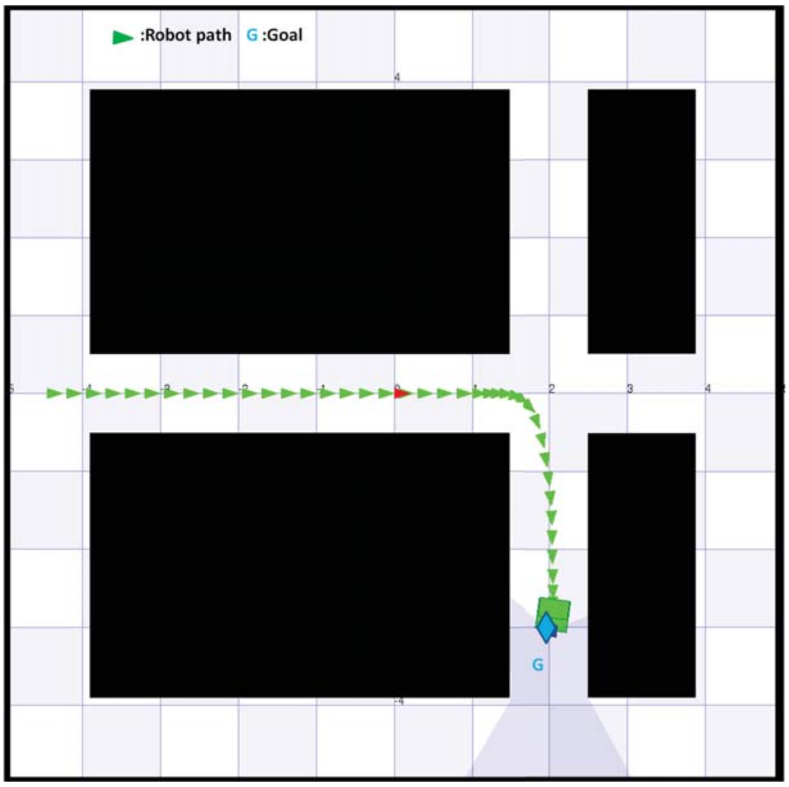


Figure 5.16: Robot successfully reaches its goal

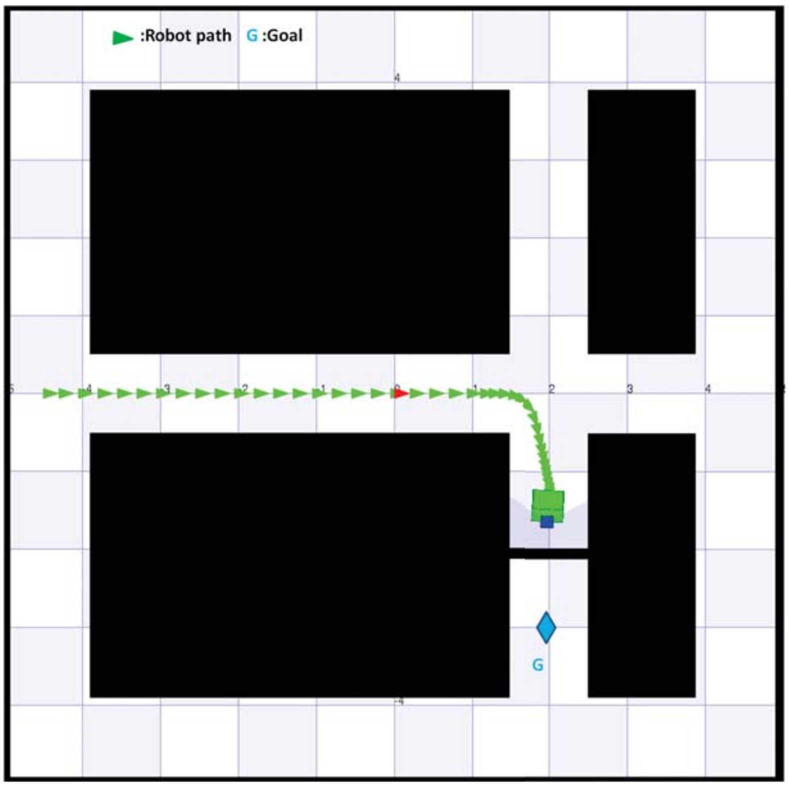


Figure 5.17: Robot is trapped in local minima

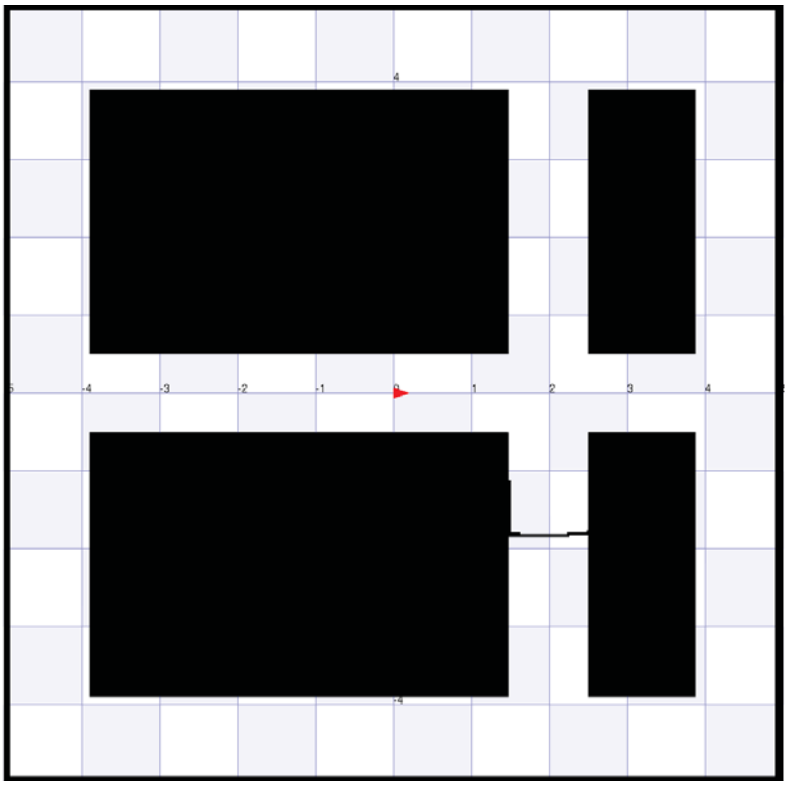
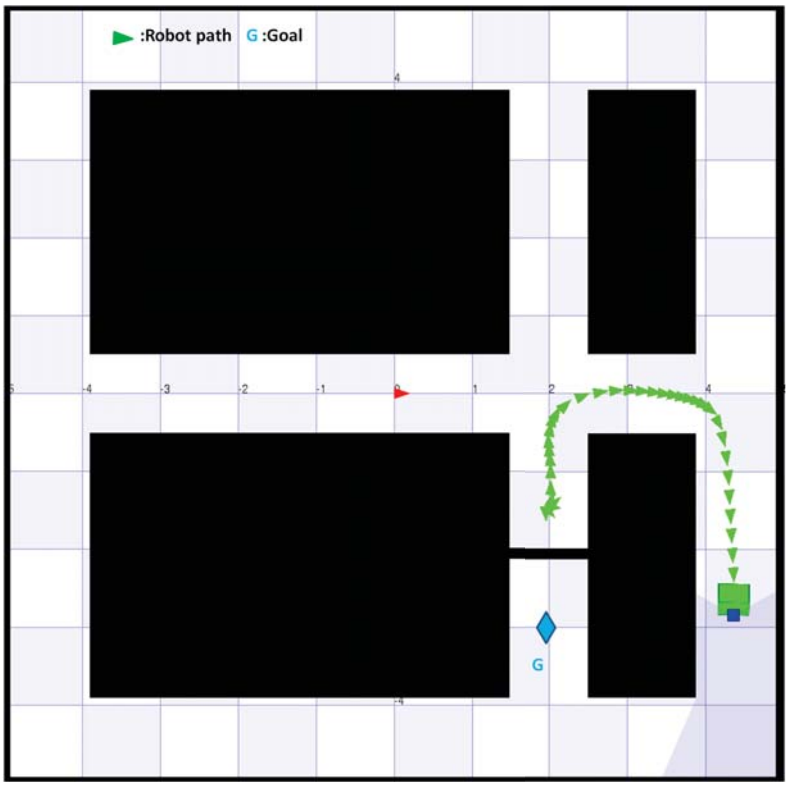
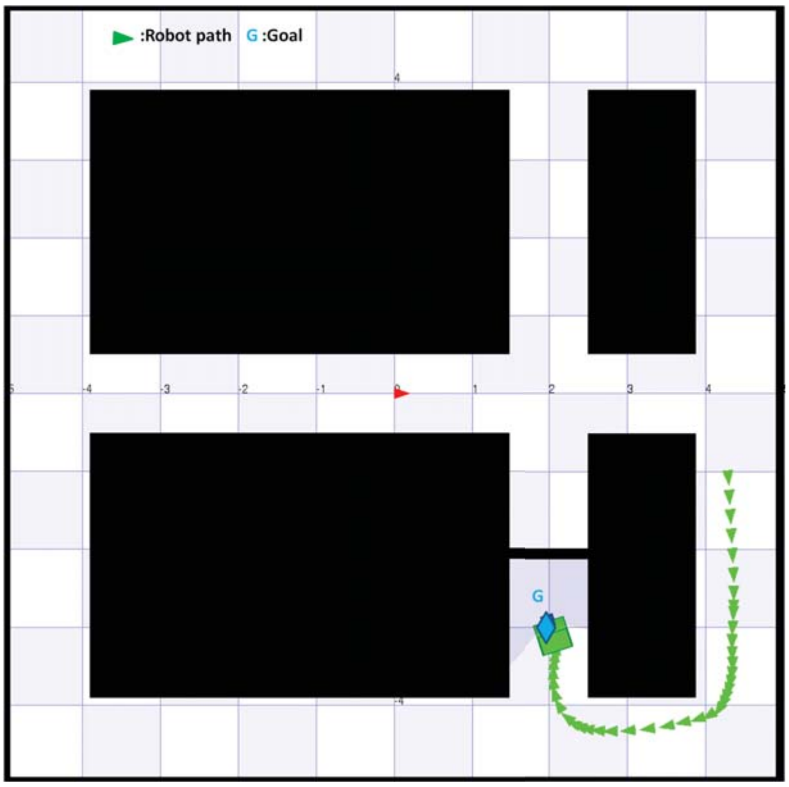


Figure 5.18: Mobile robot's Workspace



(a)



(b)

Figure 5.19: Robot carries out the new global path

5.7.2.3 Avoiding Dynamic Obstacles

In the previous simulation, the robot successfully managed to avoid static obstacles; however, moving obstacles are more problematic given the influence of their speeds, and hence the robot must estimate the speed of the dynamic obstacle and control its relative speed to avoid collision. Assume the case shown in Figure 5.20, which the initial positions of a robot, goal and dynamic obstacle in a relatively large free space, where the robot and obstacle are moving towards each others. When the robot does not consider the speed of the obstacle, the manoeuvre results in a collision as shown in Figure 5.21, and whilst the robot controls its speed outside the collision cone to avoid collision, the relative speed is within the collision cone (see Figure 5.22).

However, when the robot considers both the speed and direction of the moving obstacle, it is able to avoid it as shown in Figure 5.23, since the relative speed remains outside of the collision cone (see Figure 5.24).

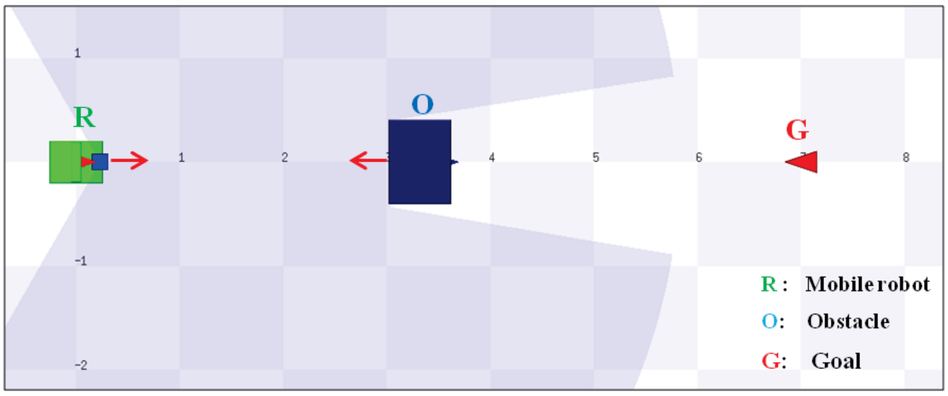


Figure 5.20: Initial positions and velocities of the robot and obstacle

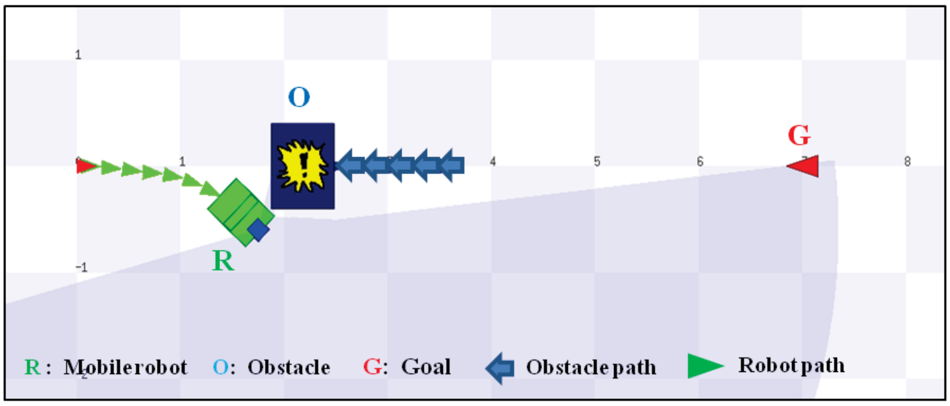


Figure 5.21: Robot collides with obstacle

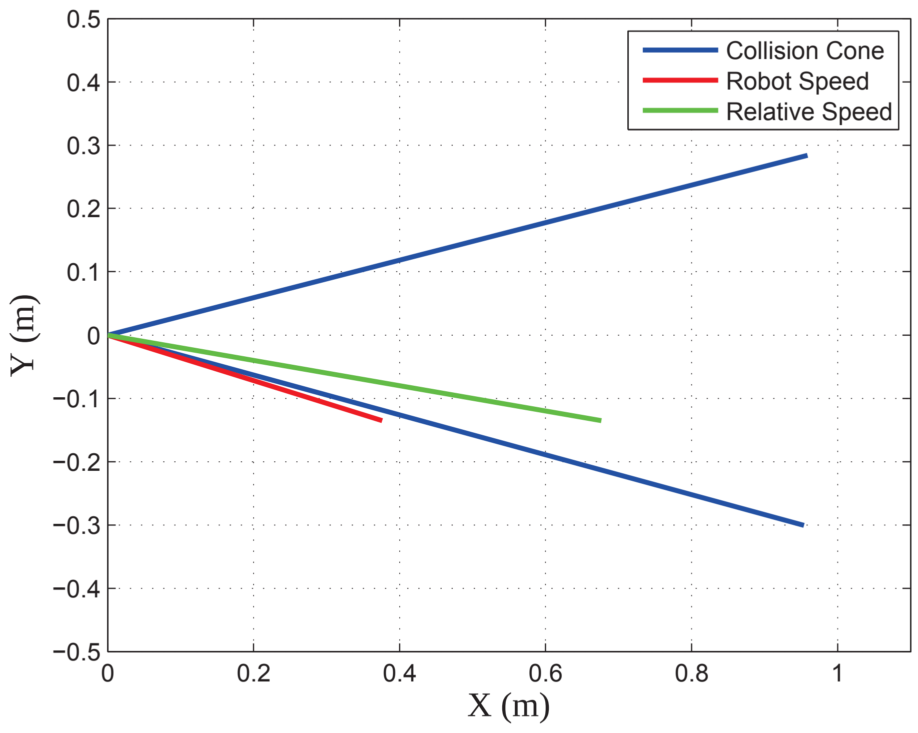


Figure 5.22: Relative speed is inside the collision cone

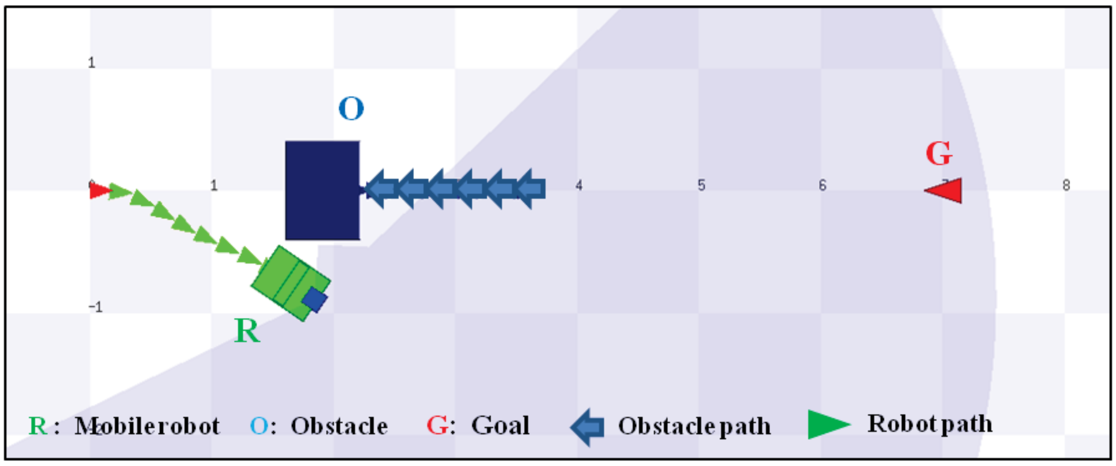


Figure 5.23: Robot avoids the dynamic obstacle

If the obstacle and robot move in the same direction, and when the robot does not consider the obstacle speed, the robot attempts to avoid the obstacle by taking a relatively long path as shown in Figure 5.25. Here, the robot controls its speed so to remain outside the collision cone, resulting in a relatively large angle between the collision cone and the relative speed (see Figure 5.26).

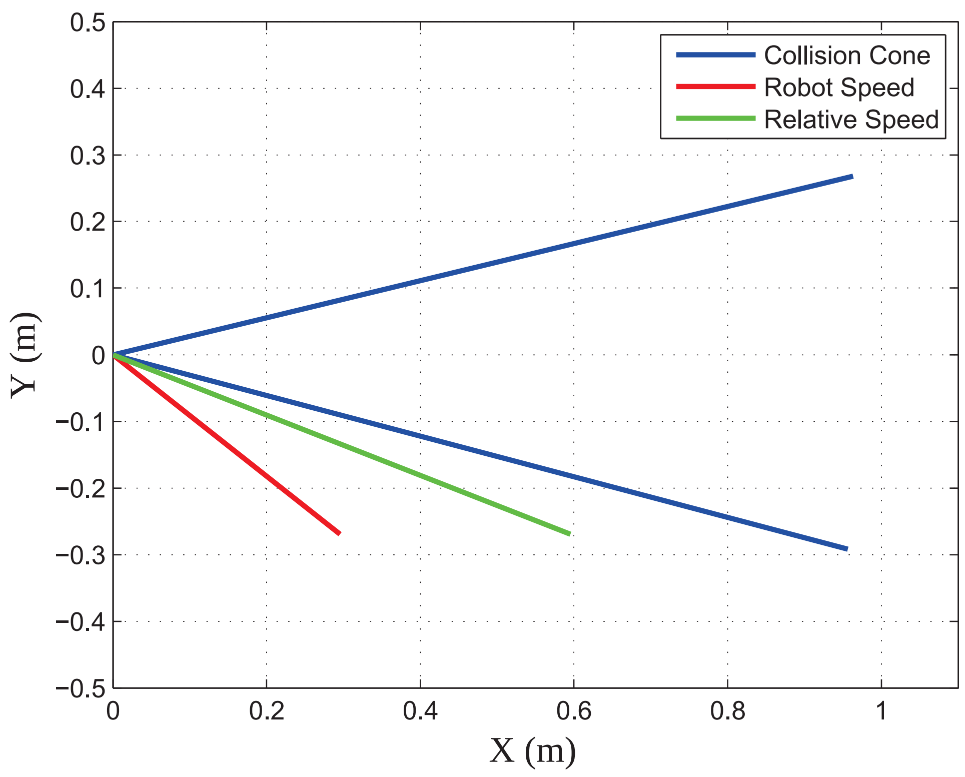


Figure 5.24: Relative speed is outside the collision cone

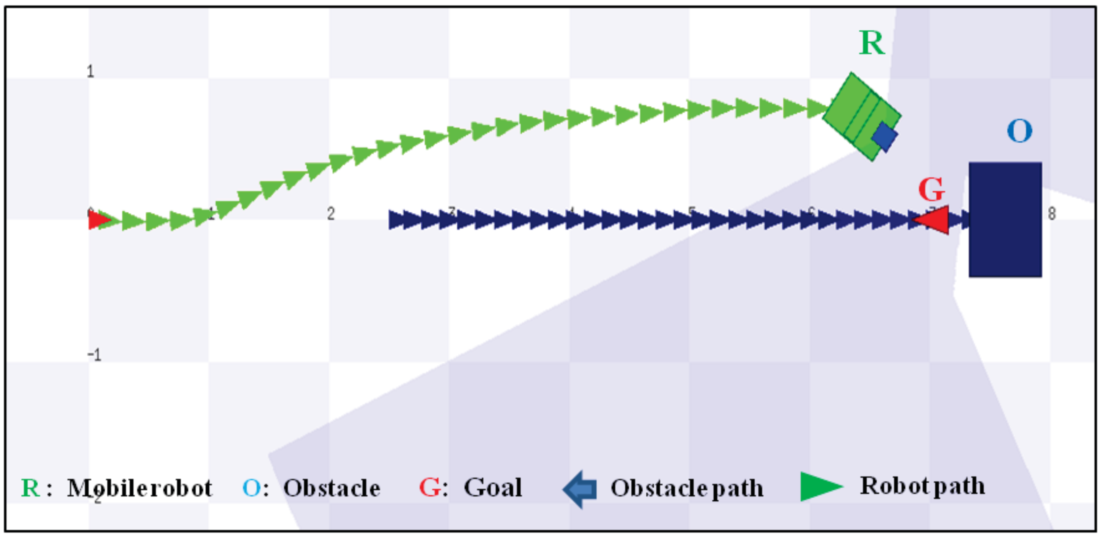


Figure 5.25: Robot is taking a long path

However, when the robot considers both the speed and direction of the obstacle, its path is much more direct than that in the previous case, as shown in Figure 5.27. It can be observed from Figure 5.27 that the robot speed is within the collision cone, while the relative speed is outside the collision cone.

These simulation cases demonstrate that if moving obstacles are considered to be static there is a very high risk of collision or else the path traveled to the goal is longer. Consequently it is very important that both the speed and direction of moving obstacles are accommodated in order to achieve a short path to the goal and minimize the collision risk.

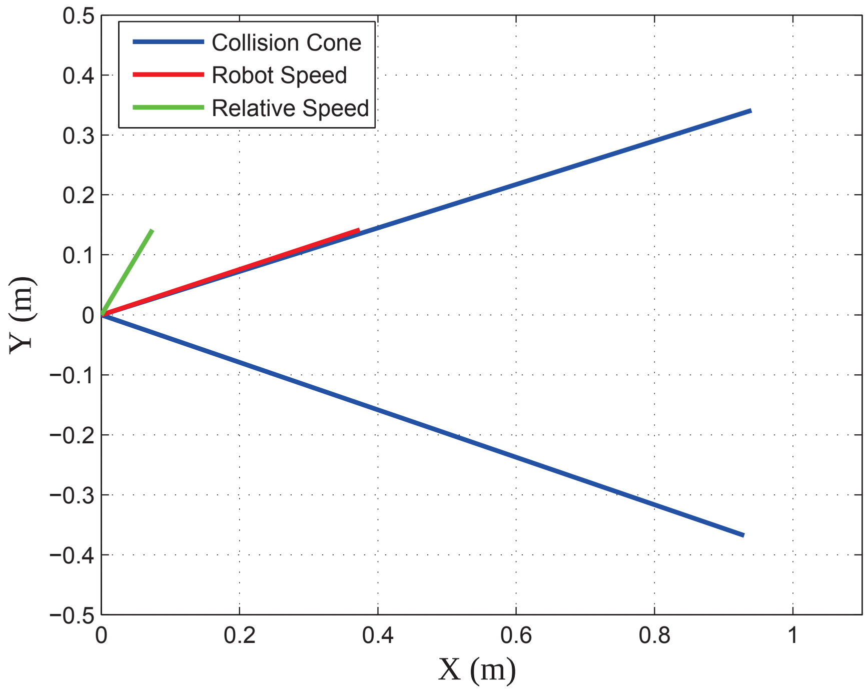


Figure 5.26: Relative speed and robot speed are outside of the collision cone

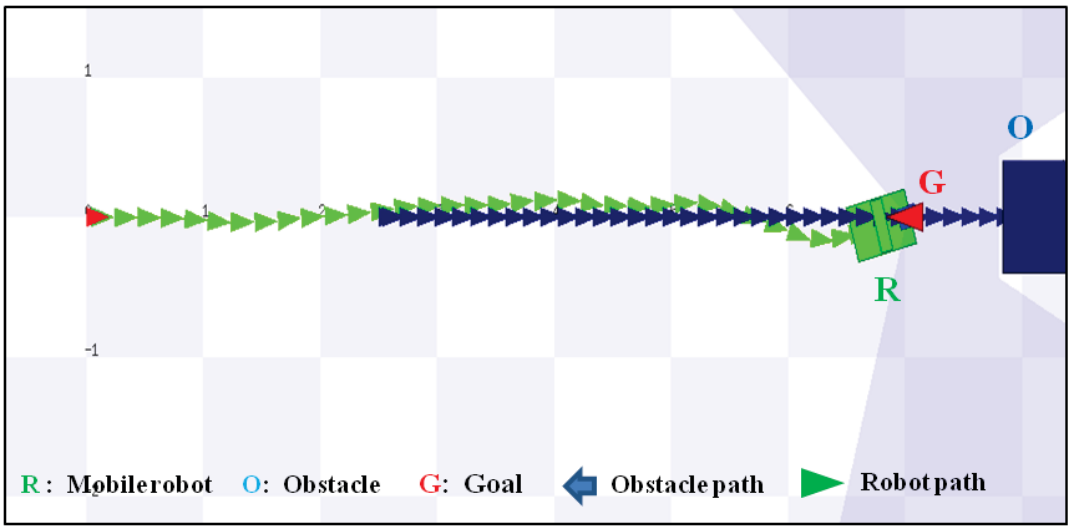


Figure 5.27: Robot is taking a short path

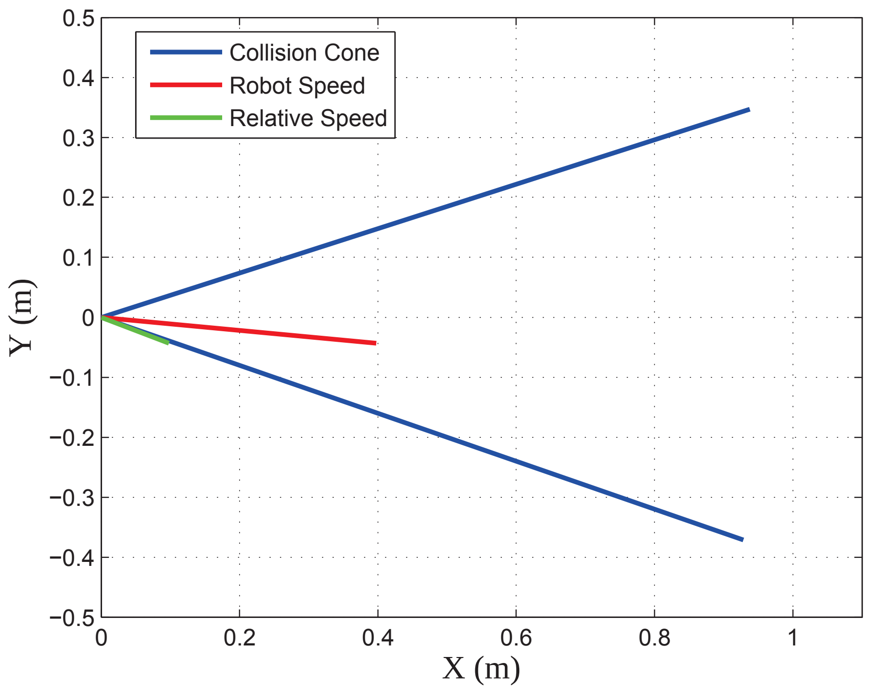


Figure 5.28: Robot speed is inside the collision cone

In previous cases, the robot moves in a large free space. Here assume the robot has to navigate within a relatively narrow corridor and avoid a dumb (non-intelligent) moving obstacle which has a fixed speed of 0.5m/s. The start points of the robot and moving obstacle were chosen as (0,0) and (7,0) respectively. In Figure 5.29, the red markers represent the obstacle path, while the green markers represent the robot path. Note that the robot easily manages to avoid the moving obstacle without any collision.

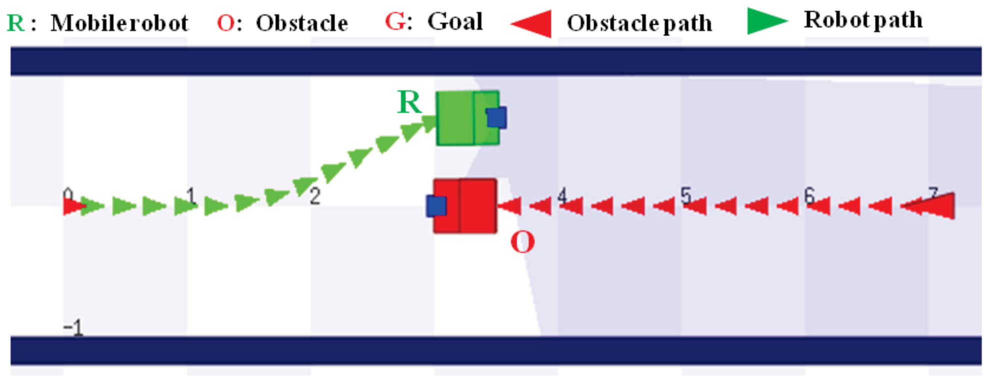


Figure 5.29: Avoiding non-intelligent moving obstacle

When the obstacle is provided with limited intelligence and is programmed to avoid static obstacles using the PFA algorithm and laser sensor, the robot successfully avoids the obstacle, while the obstacle takes limited avoiding actions to pass (see Figure 5.30). However, when the obstacle has the same navigation algorithm as that of the mobile robot, the robot and the obstacle successfully avoid each other as shown in Figure 5.31, and here the two paths are nearly symmetrical.

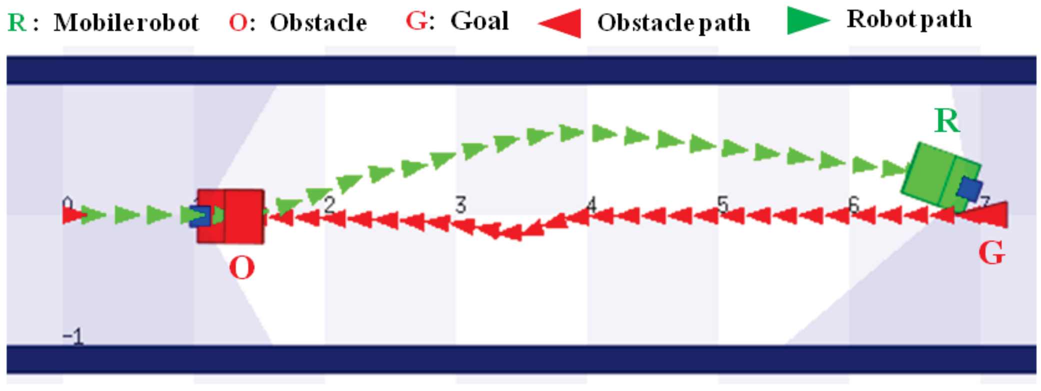


Figure 5.30: Avoiding low intelligent obstacle

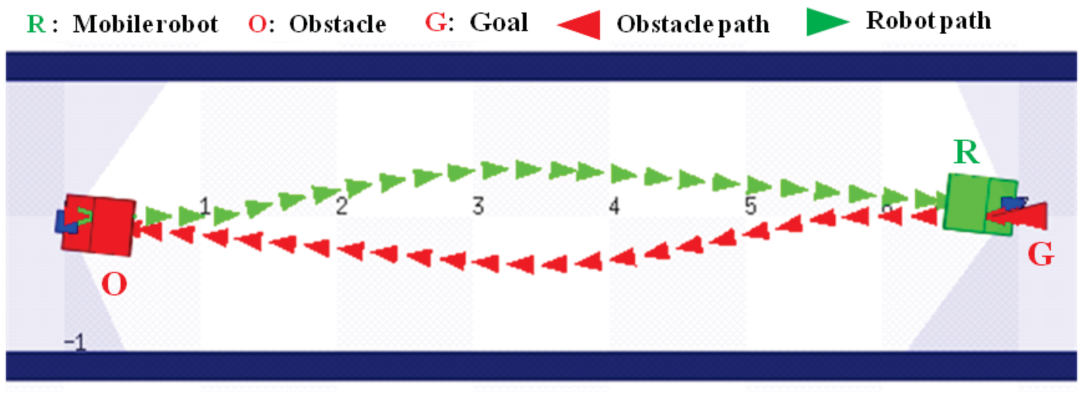


Figure 5.31: Avoiding intelligent moving obstacle

5.7.3 *Unobserved Obstacle Avoidance* Behaviour

This scenario aims to evaluate how effectively the robot is able to avoid dynamic obstacles that may suddenly appear at a corner. When the *unobserved obstacle avoidance* behaviour is deactivated (see Figure 5.32), the robot executes a straight line path and collides with a moving obstacle which appeared suddenly in the robot path. However, when the *unobserved obstacle avoidance* behaviour is activated, the robot moves away from the corner and then reacts to the obstacle by stopping, turning to allow it to pass (Figure 5.33), and resuming to its goal after the obstacle safely passes as shown in Figure 5.34.

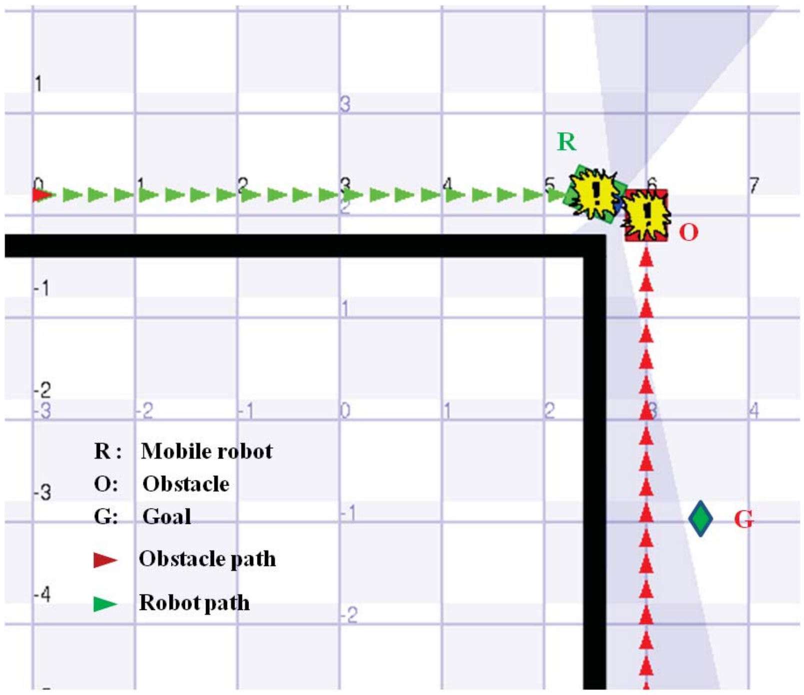


Figure 5.32: Robot collides with the dynamic obstacle

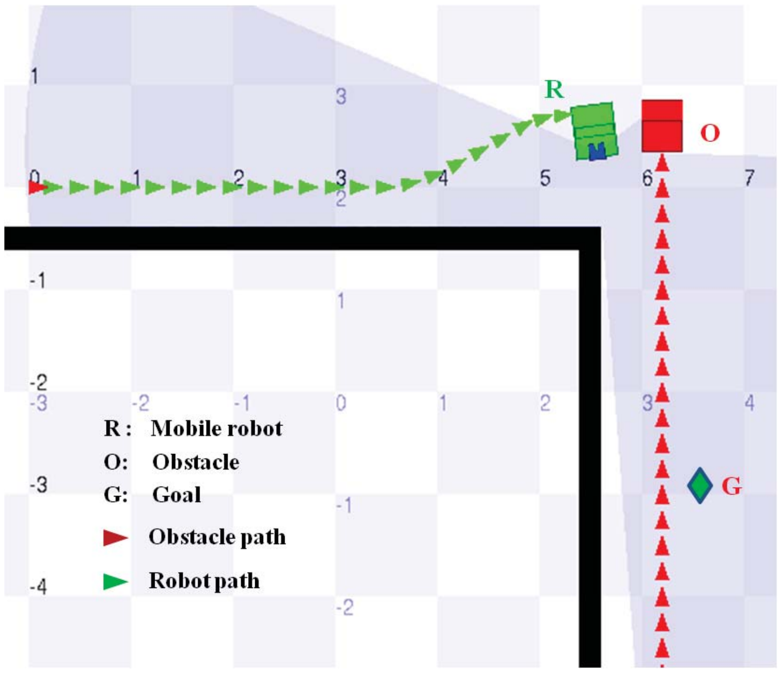


Figure 5.33: Robot Stopped and turned to avoid collision with obstacle

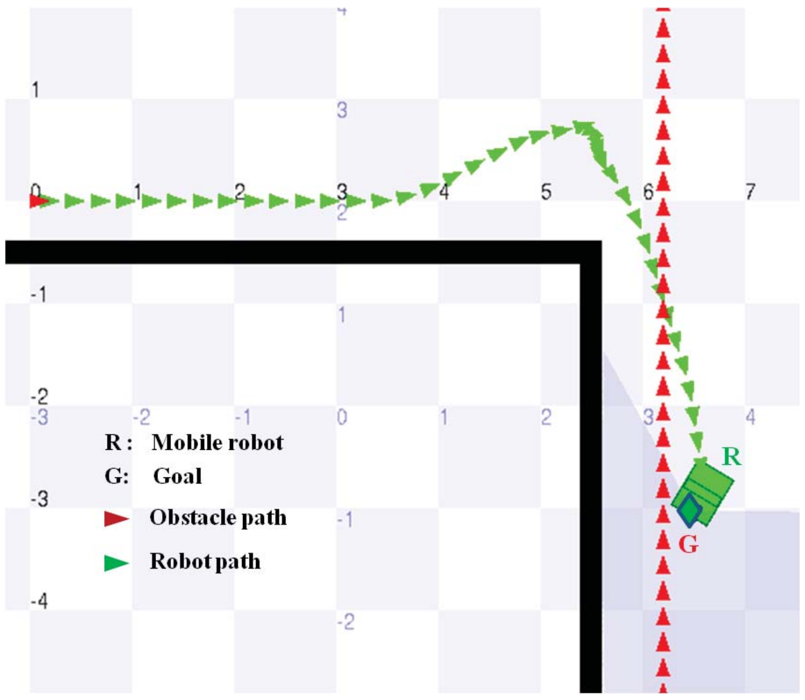


Figure 5.34: Overall path of the robot in avoiding unobserved obstacle

5.7.4 *Follow the Leader* Behaviour

Figure 5.35 shows a heavily populated corridor, where the robot has to avoid many moving obstacles, most of which are moving in the opposite direction. One obstacle *T* is moving ahead of the robot in the same direction as the robot at a speed of 0.3m/s. Two simulation tests were conducted. The first simulation demonstrates the robot reaction when the *follow the leader* behaviour is inactive and the robot continually attempts to pass the obstacle *T* (see Figure 5.36), where its movement becomes quite erratic, and it succeeds in overtaking once the crowd of approaching obstacles have passed by the obstacle *T*.

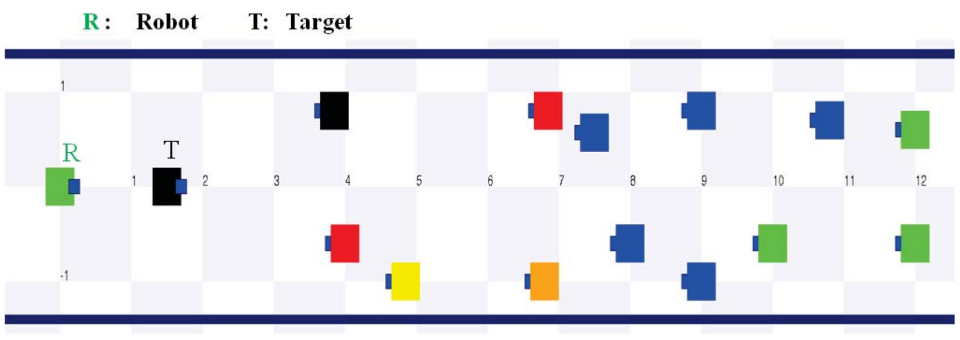


Figure 5.35: Cluttered environment

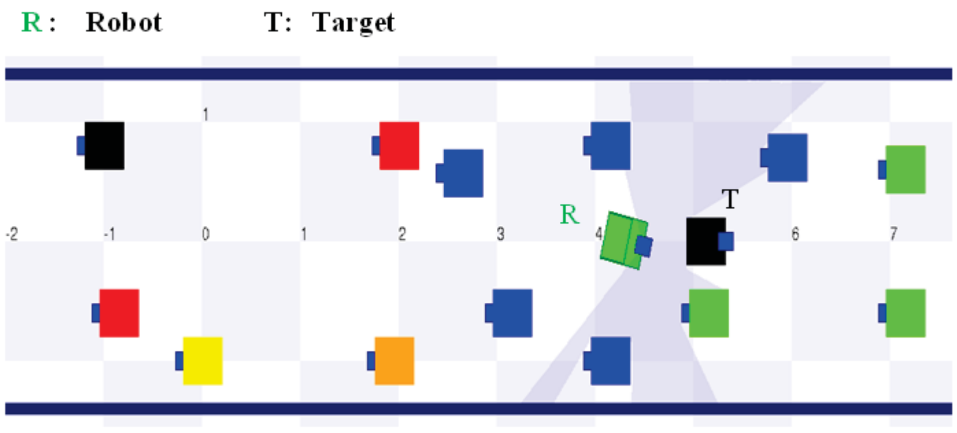


Figure 5.36: Robot is trying to avoid Obstacle-a

When the *follow the leader* behaviour is activated, it can be observed that the robot motion is much less erratic. Figure 5.37 illustrates the robot behaviour, in terms of its change in direction during the manoeuvre for both states; with the *follow the leader* behaviour active, there is less than $+/-2^0$ variation, whereas when it relies on the *obstacle avoidance* behaviour, this variation increases to $+/-6^0$ until such a time as it is safe for the robot to pass by the obstacle *T*.

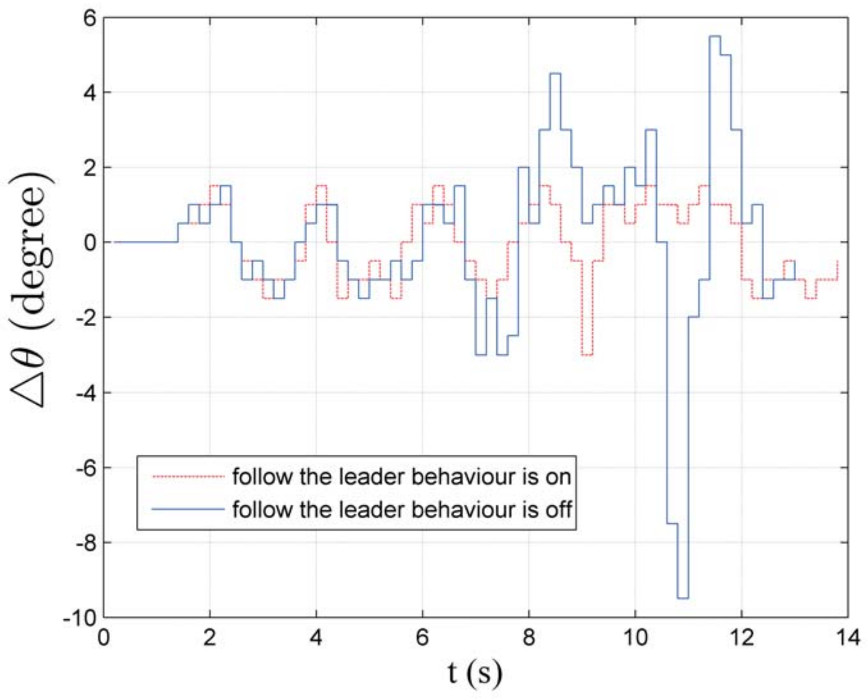


Figure 5.37: Improved performance with the *follow the leader* behaviour

5.7.5 *Emergency Stop* Behaviour

Consider the following scenario, where the robot is required to navigate from its current location (4,0) to its goal (2,0) through a narrow gap, as shown in Figure 5.38. A moving obstacle will simultaneously appear at the gap from the opposite direction. Figures 5.39-5.41 illustrate how the mobile robot is able to safely detect the obstacle, stop, and then reverse its direction to allow it to pass before resuming its journey to its goal.

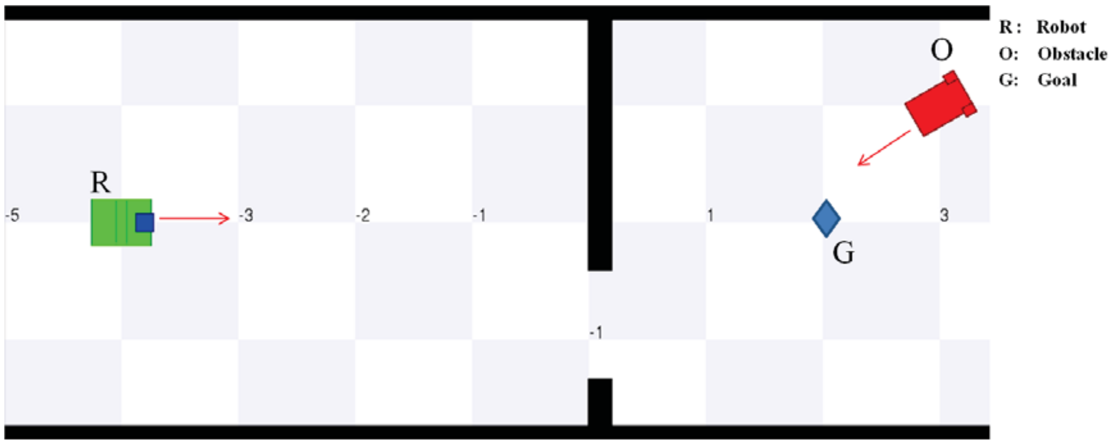


Figure 5.38: Simulation of the *Emergency Stop* behaviour

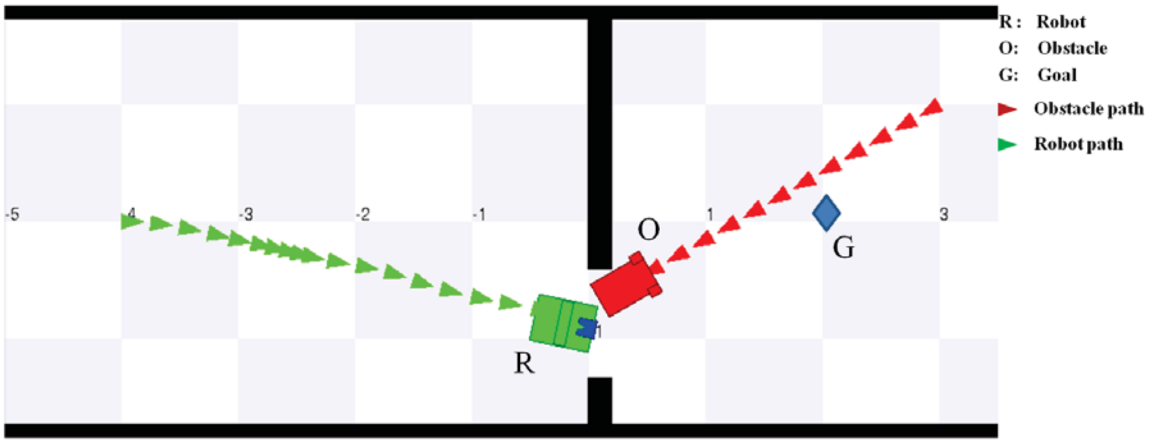


Figure 5.39: Robot stopped to avoid collision

5.8 Summary

A method has been proposed to estimate the collision cone for circular and non-circular obstacles, which is more accurate than that produced using the circle fitting algorithms. Also, the results of simulation showed that the mobile robot is able to avoid obstacles with different levels of intelligence in an indoor environment. Several novel behaviours have also been developed which provide the robot with improved avoidance capability to include multiple dynamic obstacles as well as anticipating potential collisions resulting from restricted unobserved obstacles. In particular, the *follow the leader* behaviour provides a much smoother motion when passing among multiple moving obstacles, and the *emergency stop* behaviour ensures that the robot stops and reverses to prevent collision with a moving obstacle when passing a door threshold. In the next chapter, the *human avoidance* behaviour will be developed to allow it to recognize and avoid human walking using laser data within the virtual Player/Stage environment.

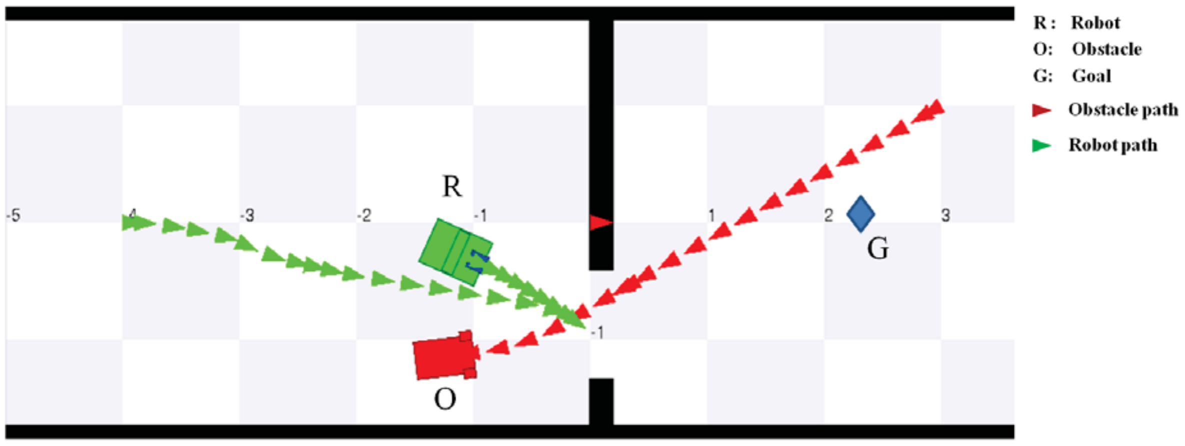


Figure 5.40: Robot reverses

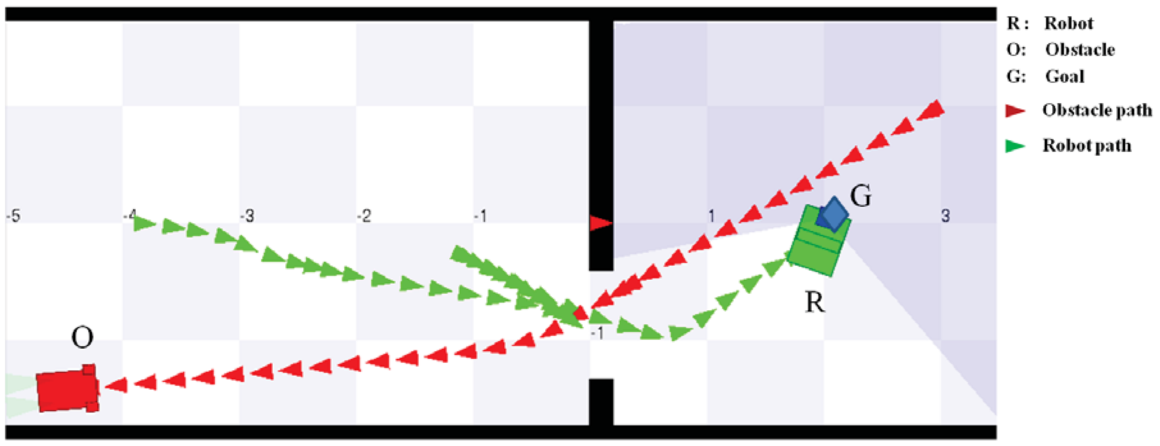


Figure 5.41: Robot passed the narrow gap

Chapter 6

Human Detection, Following and Avoidance

The previous chapter focused on five behaviours at the reactive level, namely *go to goal*, *obstacle avoidance*, *unobserved obstacle avoidance*, *follow the leader* and *emergency stop*. The simulation results show that the robot is able to avoid and minimize the collision risk with static and dynamic obstacles within crowded indoor environments. This chapter focuses on reactive behaviour associated with human detection, tracking and avoidance.

To move around a workspace occupied by humans, the mobile robot has to interact with them in an appropriate social and technological manner, and in doing so it has to first detect them and then define their direction and speed. Numerous studies have been carried out on tracking humans; and for example in [Kim et al., 2007], radio-frequency identification (RFI) was used on a mobile robot to follow another mobile robot. The same technology can be utilized to follow a human who carries a RFID tag, while the robot is fitted with a direction-finding antenna receiver. In [Belotto and Hu, 2009], a 2D laser range finder (LRF) was used to detect human legs, and a camera to detect the human face. The major problem with this technique is that when the human moves at an acute angle with respect to the camera axis, the face detection algorithm may fail. In [Carballo et al., 2009], two LRF sensors were mounted at different heights on a mobile robot to detect human legs and chests. Stereo vision technology has also been utilized for tracking humans in [Petrovic et al., 2013], with some limitations such as changes in illumination, the vibration of the camera during robot motion, and the large computation required. Furthermore, many people do not feel comfortable when they are monitored by a camera. The 3D Kinect sensor is another option to detect humans, however, this has a relatively narrow field of view [Machida et al., 2012].

Many studies used a single LRF sensor for detecting and tracking humans using the laser data returned from legs [Jinshi et al., 2006; Sung and Chung, 2013; Chang and Lian, 2012]. However, the human detection algorithm is challenged by the relative distance between the two legs during the bipedal walk, and variability due to trouser flexibility. In [Woojin et al., 2012], three parameters were proposed for detecting leg clusters namely girth, width and depth. In their study, 3258 laser images of legs in different positions were studied to define the relationship between the three parameters. [Lutz et al., 2010] used the diameter and curvature of the cluster, although they did not experience a problem of hiding one leg behind another leg because when the leg detection algorithm finds just one leg it considers this leg as a human, consequently there will be an error in the estimated position which leads to problems in estimating the speed and direction. In [Jinshi et al., 2006], the principle of the accumulated distribution of laser frames was used to extract the legs from laser data, where there is intensive difference between the standing leg and swinging leg, which is produced by accumulating the count of laser data at the same pixel of the grid map. They used several laser sensors with a high scan rate (30 fps). However for avoiding humans, their algorithm is not effective when the walking speed is relatively fast, also when the robot moves, a different part of the leg may be observed by the laser sensor.

In this project, the three parameters proposed in [Woojin et al., 2012] have been used to detect humans walking from laser data, taking in the consideration situations when a leg (or a part of it) is hidden, and when the two legs are interpreted as one cluster.

6.1 Human Tracking

The human tracking algorithm using laser data includes two stages; namely, leg detection and human detection, where leg detection stage classifies the laser clusters into 4 classes, while the human detection stage determines the classified laser clusters which could return from humans.

6.1.1 Leg Detection

The human leg can be considered as a circle or eclipse, but laser points returned from a leg may form different shapes according to the cut of the trousers. To explain this issue, a real experiment was carried out in which a person 180cm tall walked in front of a laser sensor mounted at a height of 40cm. Figure 6.1 presents laser images of a leg captured at different distances from the laser sensor, and Table 6.1 gives the distances between the leg and the laser sensor, the number of returned laser points and the leg width. As expected, it can be observed that the number of returned laser points increases as the distance to laser sensor decreases; for example, 57 points at a distance of 0.40m compared to 4 points at a distance of 4.0m. On other hand, the width of the leg cluster decreases as the distance increases.

Table 6.1: Distance, laser point number and leg width

Distance to the laser sensor (m)	Laser point number	Leg width (mm)
0.4	57	158
2.4	8	122
2.9	7	131
4.0	4	84

Assuming that the returned laser points from a leg are as shown in Figure 6.2, then the width of a cluster is given as follows:

$$W = |P_1 P_m| \tag{6.1}$$

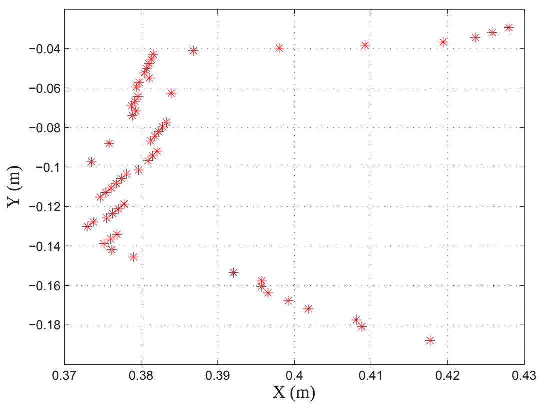
where P_1 and P_m are the first and last laser points of the leg cluster.

The girth is defined as the sum of the distances between two sequential points D_j which is given by:

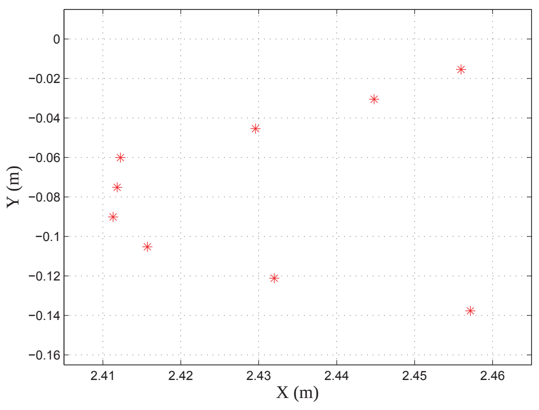
$$D_j = \sqrt{P_j^2 + P_{j+1}^2 - 2P_j P_{j+1} cos(\sigma)} \tag{6.2}$$

where σ represents the angular resolution of the laser sensor. The girth can be written using the following equation:

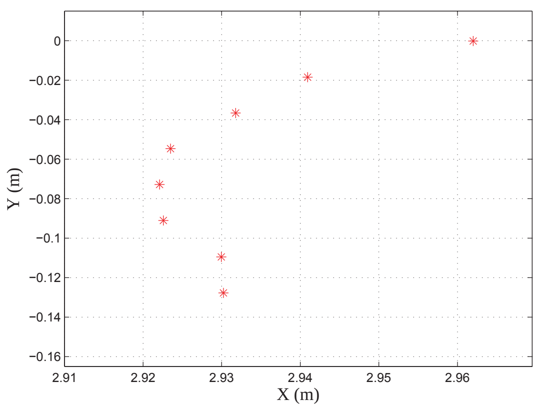
$$G = \sum_{i=1}^{i=m-1} D_i \tag{6.3}$$



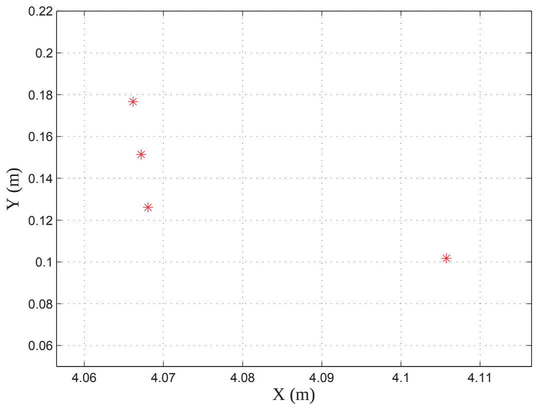
(a) Distance of 0.40 m



(b) Distance of 2.4 m



(c) Distance of 2.9 m



(d) Distance of 4.0 m

Figure 6.1: Scanned Leg images at different distances

The cluster depth is defined as the maximum distances between the cluster points and the line passing from the first to last points of the cluster. However, the distance between the points and the line between the first and last points in the leg cluster is given by the following equation:

$$d_j = \frac{(L_2^T L_2)(L_1^T L_1) - (L_1^T L_2)^2}{(L_1^T L_1)} \tag{6.4}$$

where j is the index of the cluster points, $L_1$and L_2 are given as:

$$L_1 = P_m - P_1 \;\;\&\;\; L_2 = P_j - P_1 \tag{6.5}$$

The cluster depth is given as:

$$d_{depth} = max(d) \tag{6.6}$$

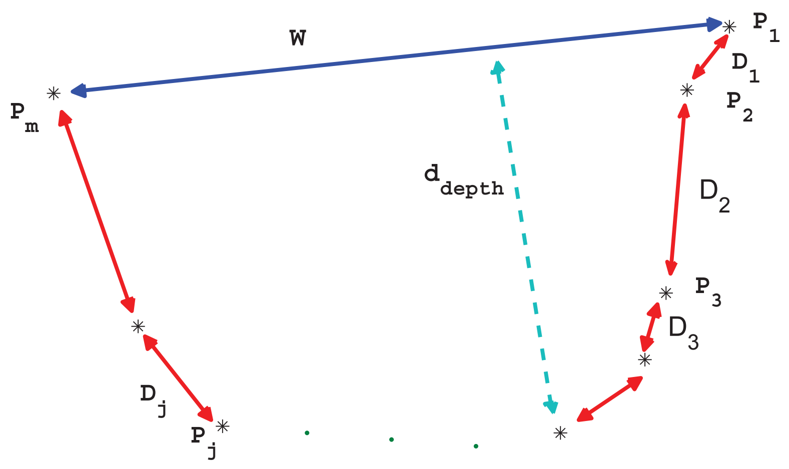


Figure 6.2: Leg detection using three parameters

In [Woojin et al. (2012)], any cluster is classified as a leg if it meets the three following conditions:

- The cluster width $W_{min} < W < W_{max}$.
- The cluster girth satisfies $G_{min} < G < G_{max}$.
- The cluster depth $d_{depth} < d_{max}$

In their study, the estimated values of the leg parameters were defined as follows: $d_{max} = 0.25\,\text{m}$; $[W_{min}, W_{max}]$=[0.09 m, 0.18 m]; $[G_{min}, G_{max}]$ =[0.091 m, 0.34 m].

6.1.2 Human Detection

Being able to successfully avoid collision with a human requires a fast and robust detection algorithm using 2D laser data to distinguish two leg clusters. The human detection algorithm proposed in [Woojin et al. (2012)] is shown in Figure 6.3, where it attempts to determine a pair of legs. However, it may fail if some parts of one leg are occluded by another leg, or if the distance between the two legs is too short. To overcome this problem, an algorithm has been developed in this project in which the laser clusters are classified into four classes:

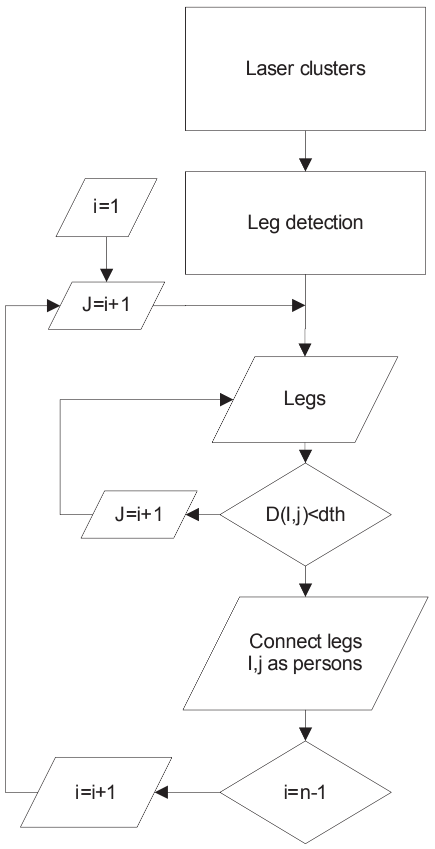


Figure 6.3: Finding the leg pairs

- Clusters having a width bigger than $2W_{max}$ are classified as L_0.
- Any cluster which meets the condition $W_{max} < W < 2W_{max}$ is classified as L_1.
- A cluster which satisfies the condition $W_{min} < W < W_{max}$ and does not meet the conditions of leg detection is classified as L_2.
- A cluster which meets the conditions of leg detection is classified as L_3.

To detect humans in time n, two steps are required. First the detected humans in time $n-1$ are associated with the laser clusters at n. The second step is to detect new people present on the laser cluster at n.

The detected humans in time $n-1$ are associated with the nearest laser clusters at n if one of the following criteria is met:

- Two clusters of class $L3$.
- Two clusters of class $L2$.
- One cluster of class $L3$ AND another of class $L2$.
- One cluster of class $L1$.
- One cluster of class $L3$ If there is no cluster of $L3$ or $L2$ within the step length.

However, a new human present in the laser data is defined by one of the following situations:

- Two clusters of the class L3.
- One cluster of the class L3 and an other of the class L2.

To verify the performance of the human tracking algorithm, a person was asked to walk in front of the mobile robot. The tracking algorithm successfully tracked the person (see Figure 6.4). Figure 6.5 shows different situations of legs captured using a Hokuyo laser sensor; where the human is detected by two clusters of class L_3 (Figure 6.5.a), one cluster of class L_1 (Figure 6.5.b), two clusters of classes L_3 and L_2 (Figure 6.5.c), and one cluster of class L_3 (Figure 6.5.d)

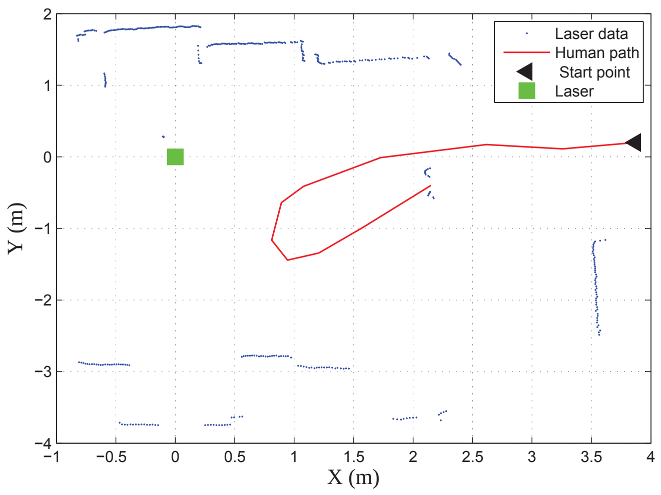


Figure 6.4: Human path

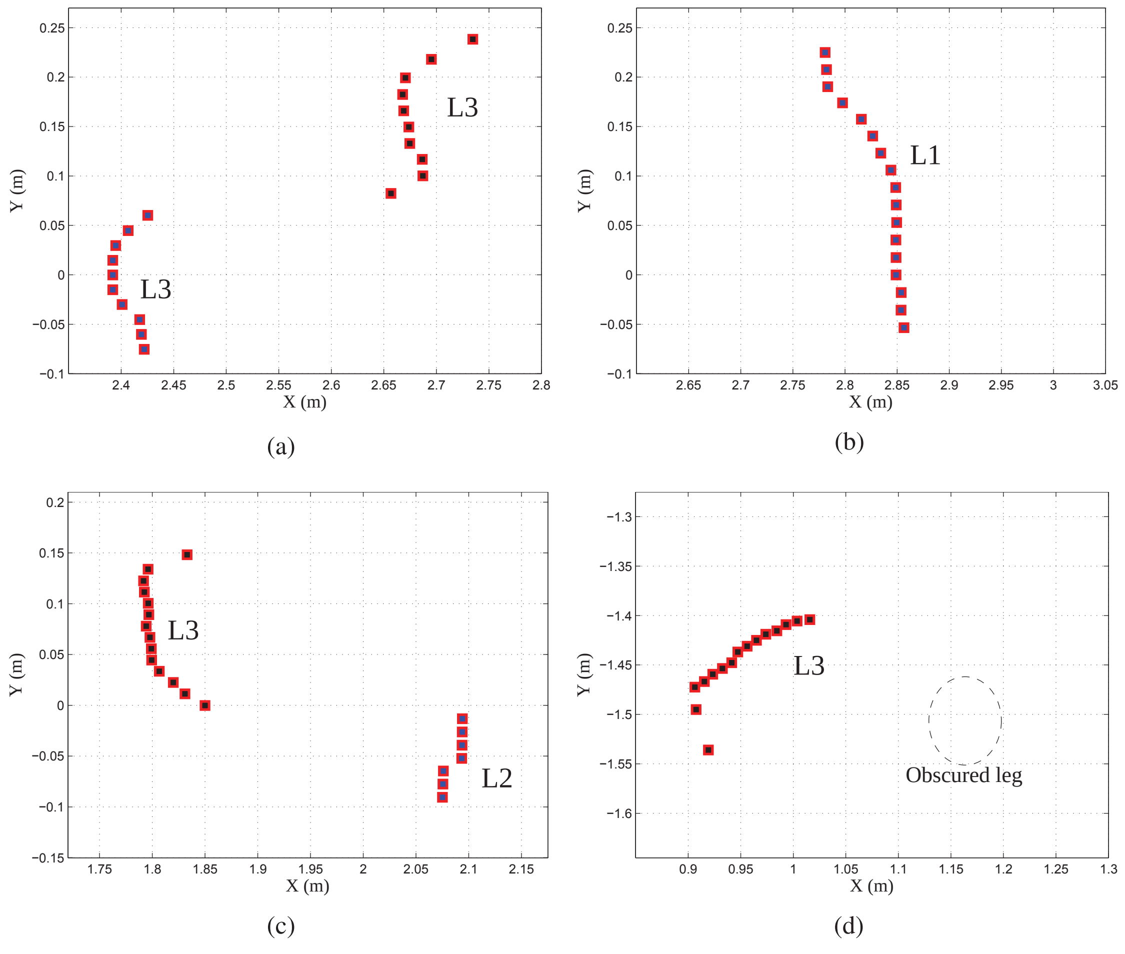


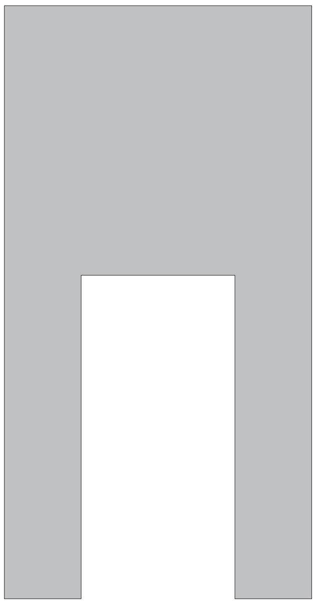
Figure 6.5: Different situations of the two human legs

6.2 Human Model in Player/Stage

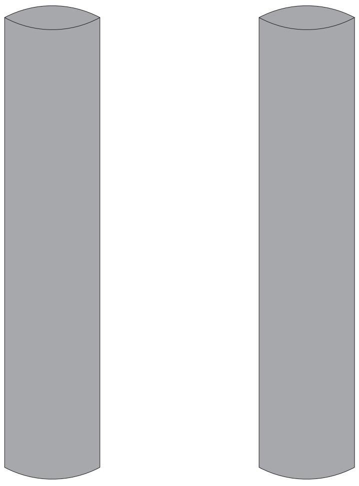
It is necessary to model the human gait in Player/Stage and implement human tracking and avoidance before applying the navigation algorithm within a physical robot. Two methods can be used to model the human in Player/Stage.

First method: the human model is built as one object, where the body and legs are attached as one piece, as shown in Figure 6.6.a. In this case, the relative distance between the legs is fixed. Kinematic control of the human model motion is easy because one object in Player/Stage is controlled.

Second method: the human is represented by two legs, as shown in Figure 6.6.b, where the upper part of the human body is not modelled because the laser sensor is only able to detect the legs. The distance between the legs changes with time according to the gait model (Figure 6.7). The equation describing the distance traveled $\triangle d_t$ by a leg is given as follows:



(a) First human model



(b) Second human model

Figure 6.6: Human modelling in Player/Stage

$$\triangle d_t = \begin{cases} \frac{A}{2} & cos(\frac{2\pi}{t_{swing}} t_1) + \frac{A}{2} + \triangle d_{stance} \qquad t_1 < t_{swing} \\ \triangle d_{stance} & if\ t_1 > t_{swing} \end{cases} \tag{6.7}$$

$$A = \triangle d_{swing} - \triangle d_{stance} \tag{6.8}$$

$$t_1 = rem(t/t_{total}) \tag{6.9}$$

$$t_{total} = t_{swing} + t_{stance} \tag{6.10}$$

where t_{max} and t_{stance} are provided by the swing and standing phases respectively, d_{swing} and d_{stance} represent the maximum movement during the swing and standing phases, respectively, and *rem* is a function which returns the remainder of a division.

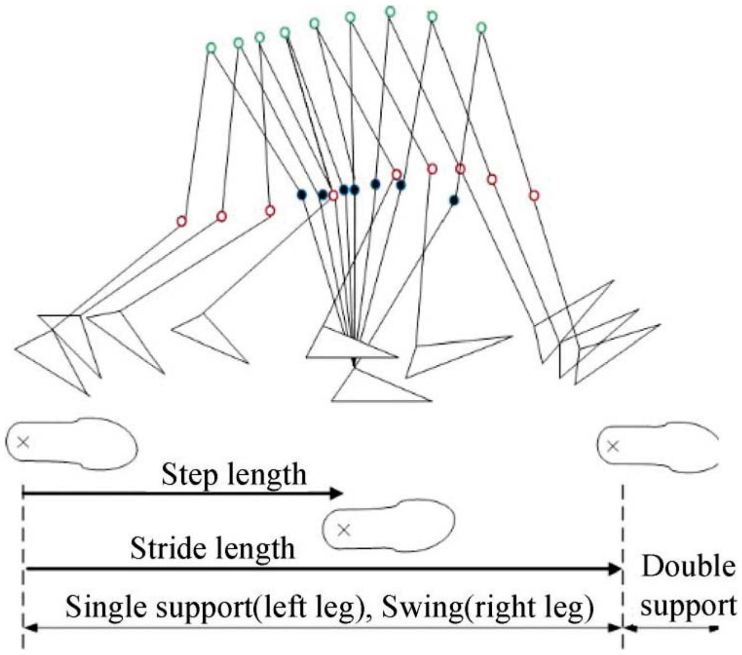


Figure 6.7: One cycle of the human walking [Woojin et al. (2012)]

6.3 *Human Avoidance* Behaviour

The *Human avoidance* behavior uses the extended particle filter algorithm to estimate the directions and speeds of humans, where the weighting function of this behaviour is given as:

$$W_{Hum} = \begin{cases} 0 & if\ t_c < t_{cth} \\ exp(-\frac{a}{tc}\quad) & else \end{cases} \tag{6.11}$$

where t_{cth} and t_c represent a threshold and the collision time respectively, and a is constant.

6.4 Human Following

The mobile robot uses the wavefront algorithm to generate long path planning, but in some applications the mobile robot may have to follow a human carrying some items or it may be travelling within a unknown environment. The *follow the leader* behaviour can easily be modified to follow a moving human target, if the location of the moving target at time n is (x_{Ht}, y_{Ht}), and the new location of the target is updated by calculating the distance between the target and all humans detected at $n+1$ as follows:

$$D(i) = \sqrt{(x_{Ht} - H_x(i))^2 + (y_{Ht} - H_y(i))^2} \tag{6.12}$$

where $H_x(i)$ and $H_y(i)$ are the coordinates of the i_{th} human.

The target is associated with the nearest human using the algorithm shown in Figure 6.8, in which the minimum distance d_{min} is initialized with a numerical value of 100, while the index $Indx$ is initialized with a numerical value of 0. The minimum distance d_{min} is compared with the distance $D(i)$, and if the condition $d_{min} > D(i)$ is met, the minimum value d_{min} and index $Indx$ are updated by the values of $D(i)$ and i respectively. If the final value of d_{min} is less than a threshold, the target location is updated using the human position having the index $Indx$, whereas otherwise the target location is updated depending on its estimated speed. However, if the robot does not find the target in a specific time, it will stop.

The control algorithm has to compute a suitable speed and direction to minimize the collision risk with the target by maintaining a safe distance and speed to the target. To meet this requirement, the linear speed command is given as follows:

$$V_t = \begin{cases} v_{max} & if\ d_t > D_{max} \\ 0 & if\ d_t < D_{min} \\ v_{max}\frac{d_t - D_{min}}{D_{max} - D_{min}} & else \end{cases} \tag{6.13}$$

where D_{max} and D_{min} define the tracking area, and d_t refers to the distance between the robot and target.

The robot always maintains a safe distance to the target as a function of the target speed using equation 6.13, and the distance between the robot and target is computed using:

$$d_{HR} = \frac{v_{Ht}}{v_{max}}.(D_{max} - D_{min}) + D_{min} \tag{6.14}$$

where v_{Ht} is the target speed.

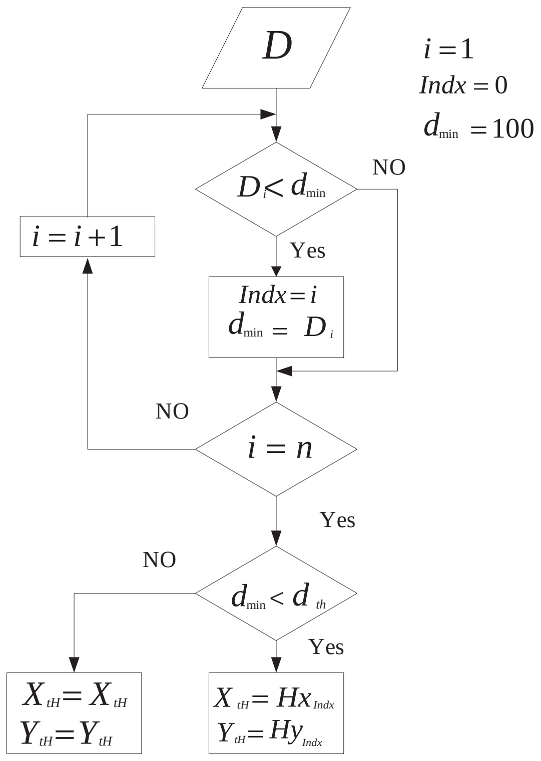


Figure 6.8: Finding the target

From equation 6.14, the distance between the robot and its target is D_{min} if the target stops, whereas it is D_{max} if the target moves with the maximum speed of the robot. The *human avoidance* behaviour should not consider the target as an obstacle. However, if the target moves towards the robot, it has to take avoidance action in order to allow him/her to pass safely. To overcome this problem, the *follow the leader* behaviour transmits the signal S (written in equation 6.15) to inform the *human avoidance* behaviour when it considers the target to be an obstacle.

$$S = \begin{cases} Indx & if\ |\theta_r - \theta_{Ht}| < \frac{\pi}{2} \\ -1 & else \end{cases} \tag{6.15}$$

where θ_{Ht} is the direction to the target human.

If the difference in the angle between the robot and target is greater than 90^o, the *follow the leader* behaviour sets the signal $S = -1$ to consider the target as an obstacle, otherwise it sets $S = Indx$ to ignore the target as an obstacle. Note that the *follow the leader* behaviour can be used for global path planning, where a human can guide the robot through its workspace, and then the robot can be programmed to reproduce the same path.

6.5 Simulation Results

6.5.1 Human Tracking in an Open Environment

Consider the case shown in Figure 6.9, in which the mobile robot has to follow a guide (G) passing between two stationary humans (S1, S2). The initial positions of the robot, target, and two persons are (0,0), (2.5 , 0.5), (5.6, 0.7), and (5.6, 1.7) respectively.

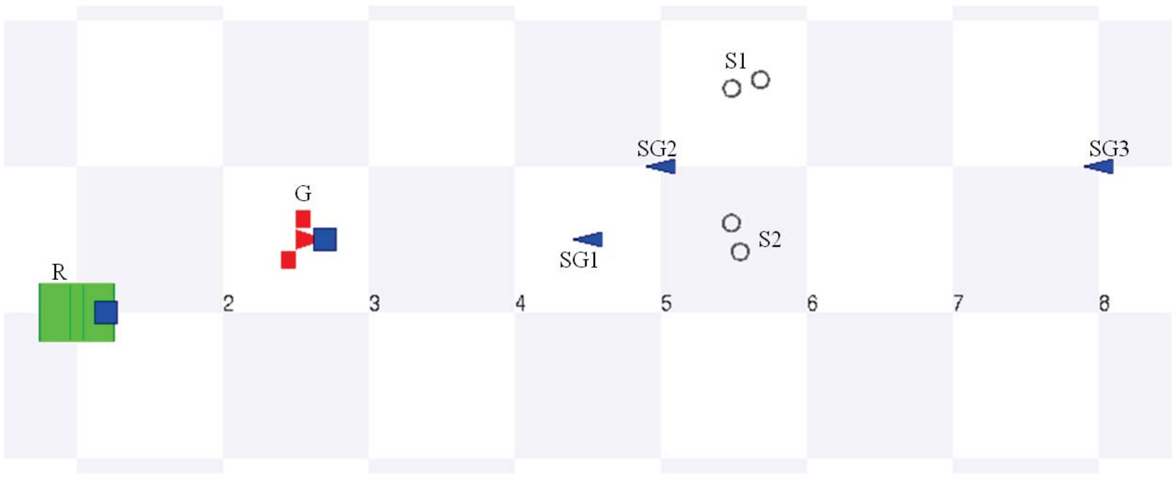


Figure 6.9: Mobile robot is Following a human

To simplify the algorithm, the target used only the *go to goal* behaviour to implement a predefined path determined by several subgoals SGi listed in Table 6.2. The linear v_{tH} and rotation w_{tH} velocities of the target are given as:

$$v_{tH} = 0.3\, m/s \tag{6.16}$$

$$w_{tH} = k.\triangle\theta \tag{6.17}$$

where k=0.5, and $\triangle\theta$ is obtained using the following equation :

$$\triangle\theta = \arctan\left(\frac{y_{Ht} - y_t(i)}{x_{Ht} - x_t(i)}\right) \tag{6.18}$$

where $x_t(i)$ and $y_t(i)$ are the coordinates of the subgoal ith.

In each sample period, the algorithm checks the distance between the human target and the subgoal, and if the distance is < 0.3m the subgoal index i is incremented to move to the next subgoal.

Table 6.2: Subgoals for the target

subgoals	x	y
1	4.5	0.5
2	5	1
3	8	1

Figure 6.10 illustrates that the robot is able to successfully follow the target guide and pass between the two persons without collision or losing the target. It can be observed that the robot path does not coincide with the target path because the robot attempts to navigate the shortest path to the target.

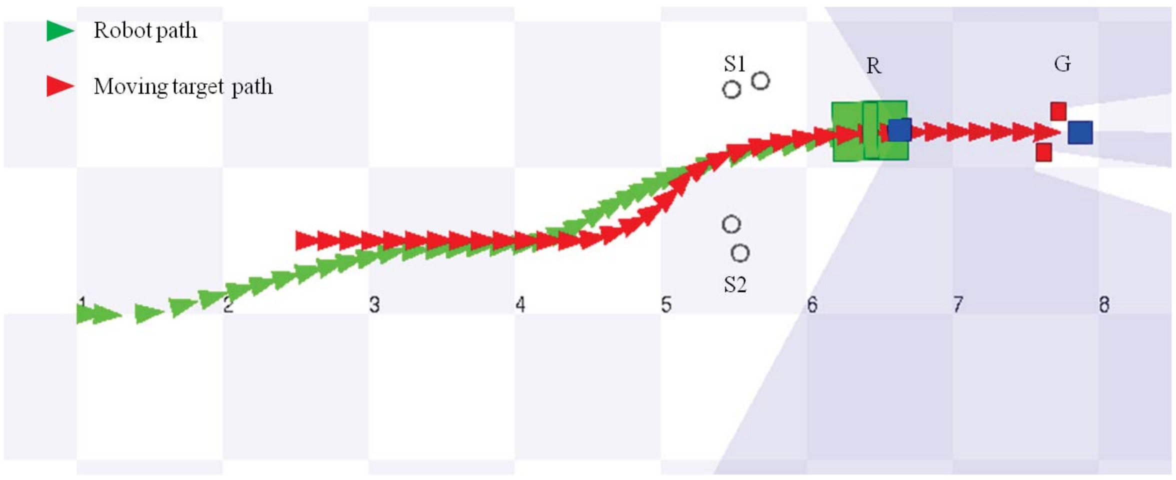


Figure 6.10: Human tracking in an open workspace

Figure 6.11 presents a laser scan, where the robot is able to detect the target G, S1 and S2 even though the distance between the target and S2 is < 0.11m . Person S1 and the target were defined by two leg clusters, while one of legs of person S2 was occluded. The maximum distance d_{max} and minimum distance d_{min} in Equation 6.13 were chosen to be 1.75m and 0.75m respectively, and the robot has to maintain a safe distance of 1.3m to the target according to Equation 6.14. Figure 6.12 shows that the proximity between the robot and its target was initially 1.72m, which was decreased and maintained at approximately 1.3m for several seconds before increasing to a distance of 1.6m, as the robot had to navigate the gap between persons S1 and S2, after which it was able to recover the tracking distance of 1.3m for the remainder of the mission.

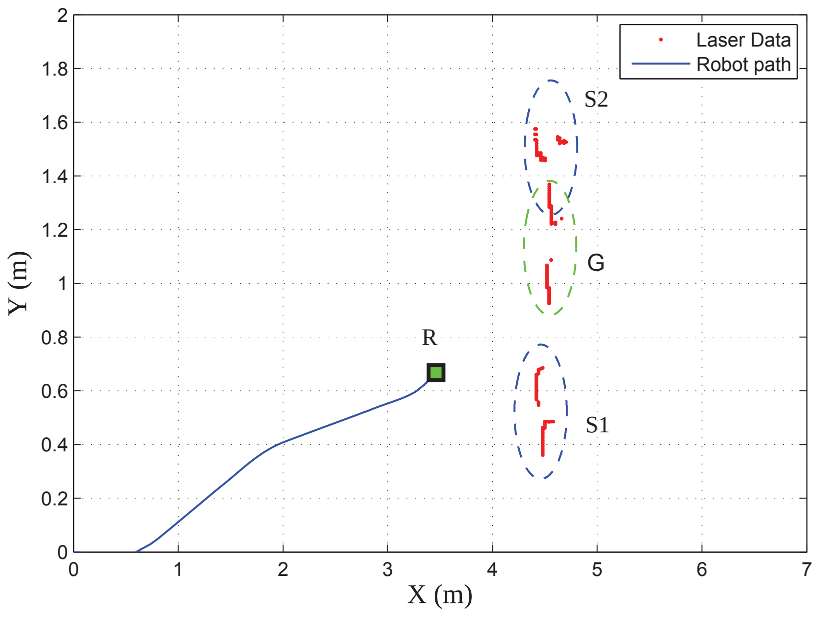


Figure 6.11: Laser image at moment 11s

To increase the complexity of the task, another person S3 was added to the workspace at an initial position of (3.5, 3.5). The velocities of the moving person S3 were chosen to be $v = 0.3$m/s and $w =$0.1rad/s to navigate a circular path to cross between the robot and its target. Figure 6.13.a shows that the robot turned right to avoid S3, and then turned left to follow the target which turned right to the next subgoal. However, the robot was able to follow the target without causing any collision with the moving human or with the two static human obstacles (Figure 6.13.b).

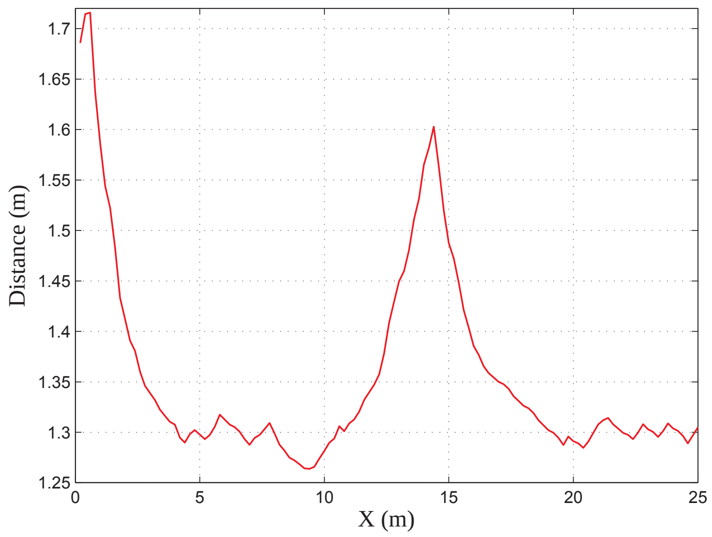
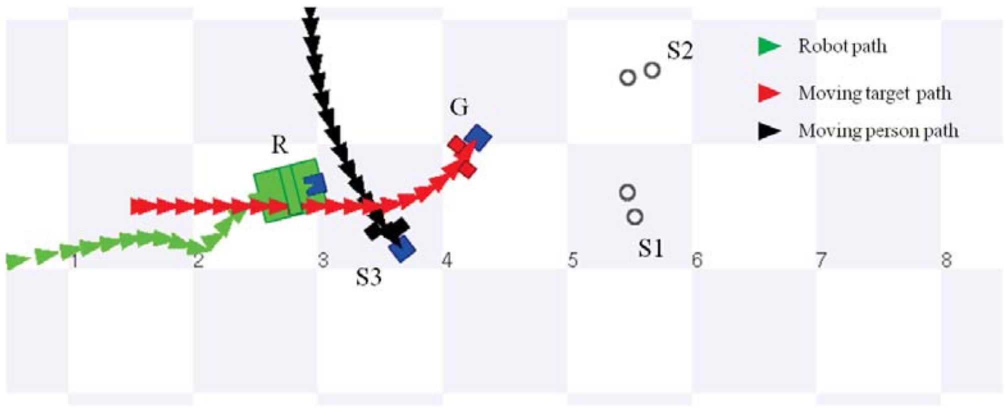
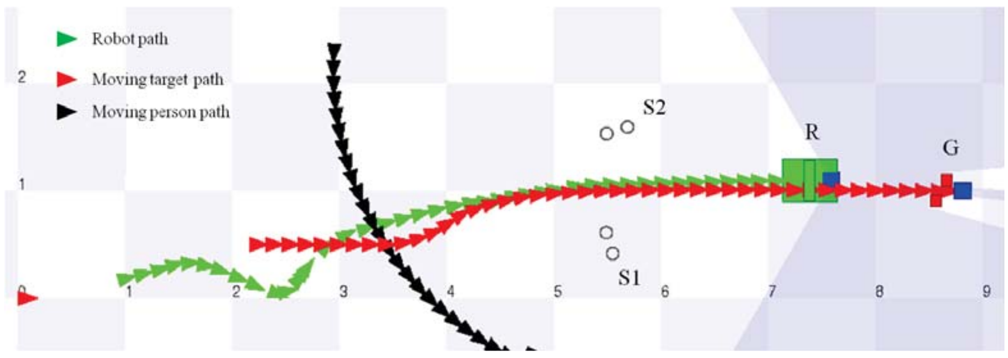


Figure 6.12: Variation in the distance between the robot and target



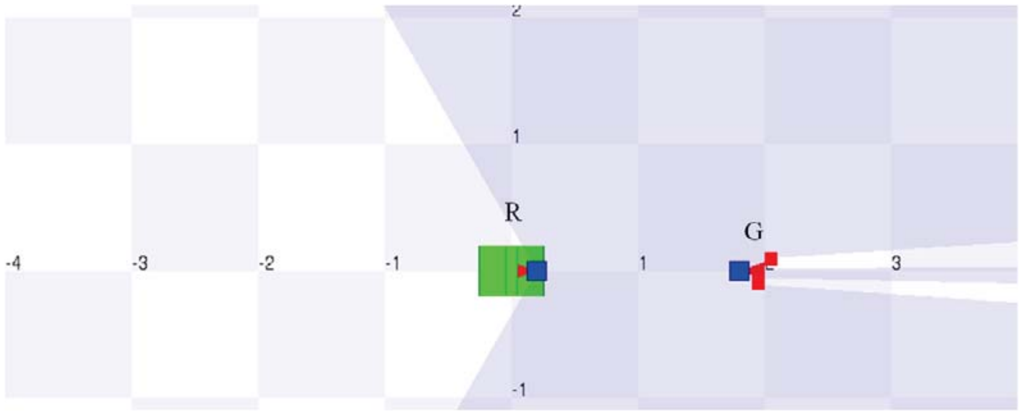
(a)



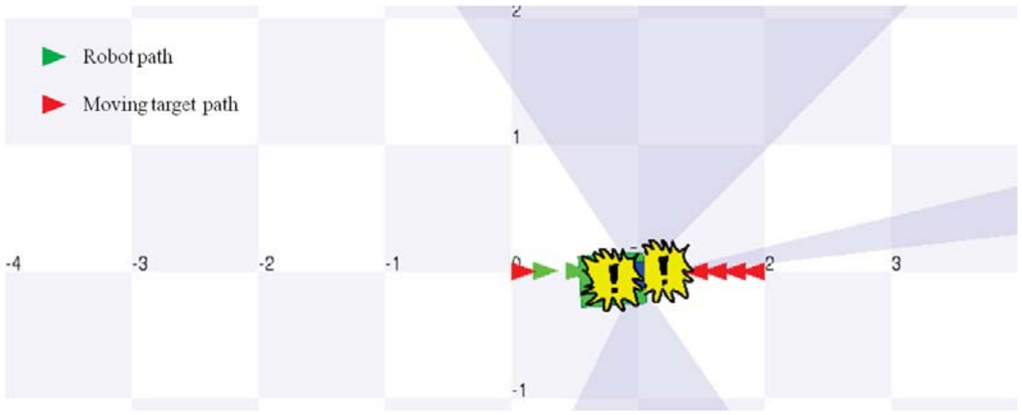
(b)

Figure 6.13: A person is passing between the robot and target

Consider the case in Figure 6.14.a, in which the initial positions of the robot and target were chosen to be $(0, 0)$ and $(2 , 0)$ respectively, and the target moves with a fixed speed of 0.3m/s towards the mobile robot. When the *human avoidance* behaviour does not consider the target to be an obstacle, they collide as shown in Figure 6.15.b, but when the robot considers the target to be an obstacle it avoids the target (Figure 6.15.a) and follows the target after it had passed (Figure 6.15.b).

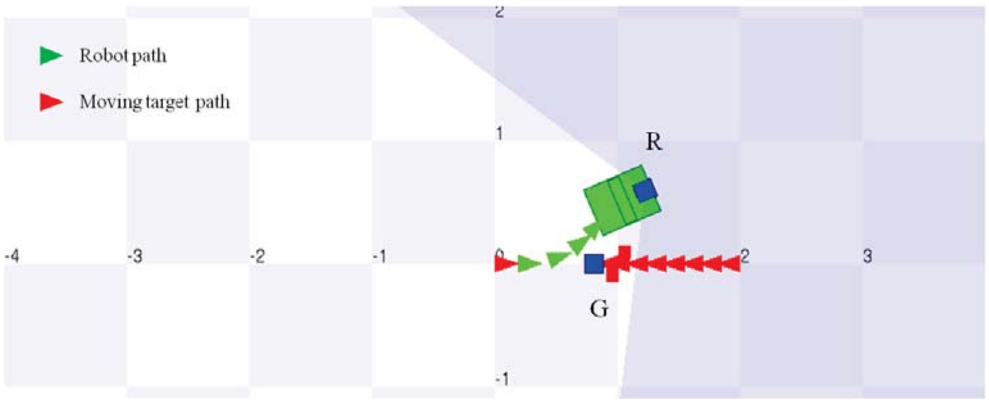


(a)

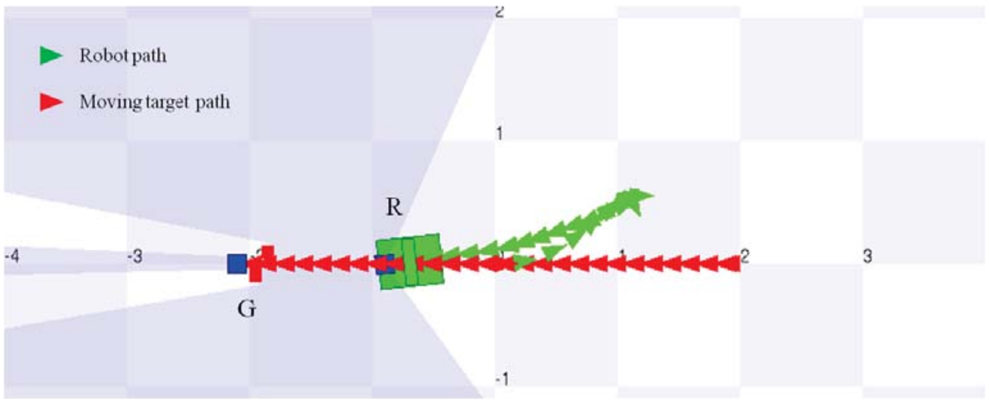


(b)

Figure 6.14: Robot collides with the target moving toward the robot



(a)

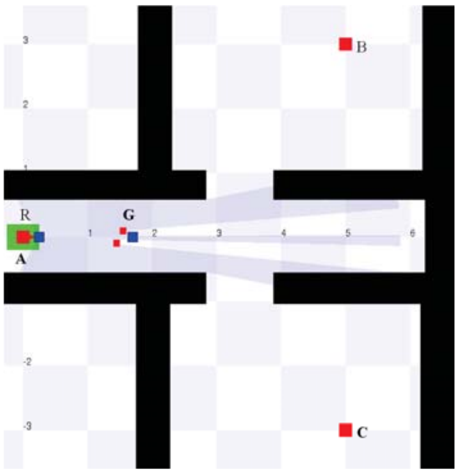


(b)

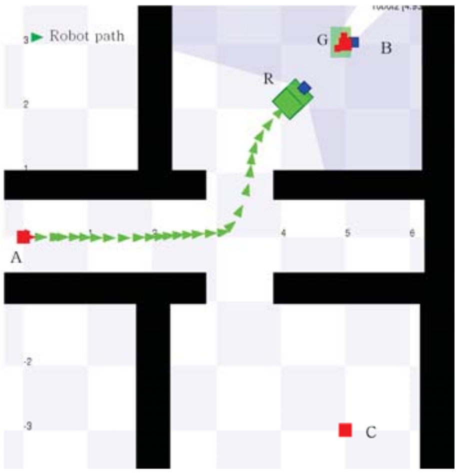
Figure 6.15: Robot avoids with the target moving toward the robot

6.5.2 Global Path Planning Using Human Tracking

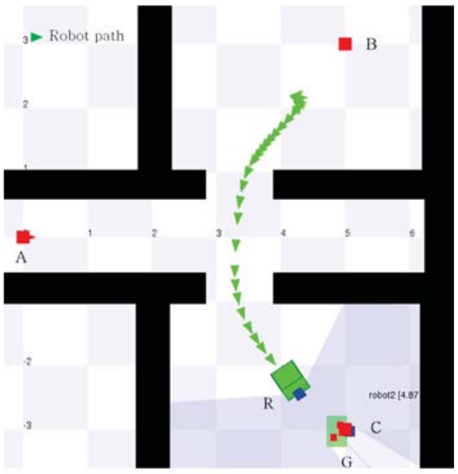
The user can assist the robot to plan a global path by guiding it between points, after which the robot can easily navigate those points without any further help from the user. Figure 6.16.a provides an example where the robot R follows a guide G to navigate between points A, B, and C. The target was moved using a mouse to guide the robot from A to B (Figure 6.16.b), from B to C (Figure 6.16.c), and from C to A (Figure 6.16.d). It can be observed that the robot stops short of the goal points during training because it is programmed to keep a safe distance from the human. The path planner generated three paths listed in Table 6.3, along with a number of waypoints. In the implementation stage, the user requests the robot to execute a counter-clockwise path from A to C to B (see Figure 6.17). It can be observed that the robot was able to navigate the path successfully.



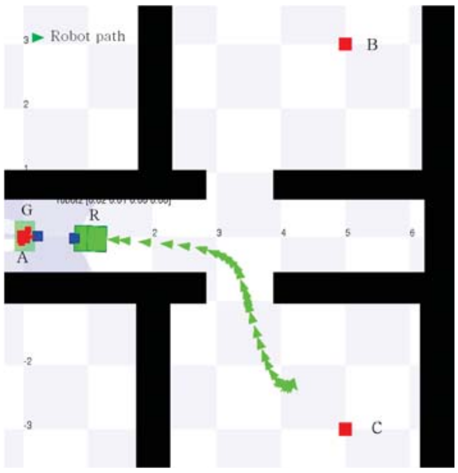
(a)



(b)



(c)

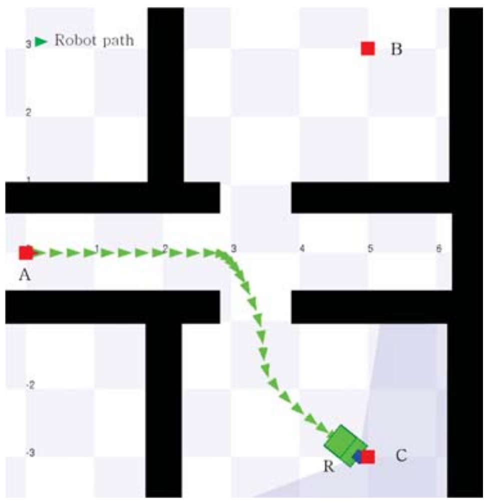


(d)

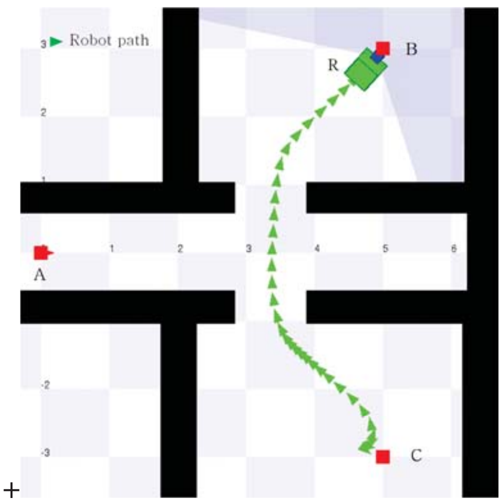
Figure 6.16: Path planning stage by following a human

Table 6.3: Long path planning outputs of the planning stage

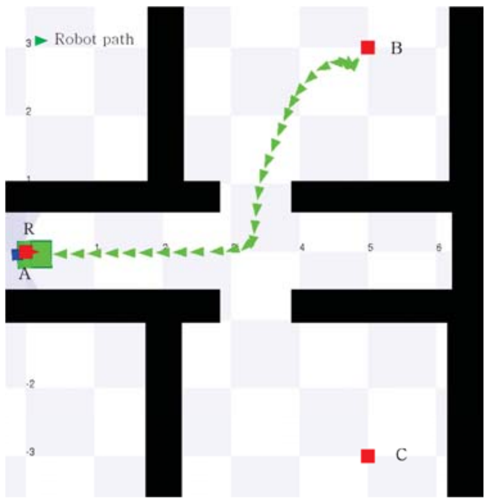
path	Start	Goal	Number of waypoints	Locations of Waypoints
1	A (0.0,0.0)	B (5.0,3.0)	4	(0.00,0.00) (1.44 ,-0.03) (3.04,0.54) (4.85,2.90)
2	B (5.0,3.0)	C (5.0,-3.0)	3	(4.82, 2.87) (3.13, -0.55) (4.73, -2.98)
3	C (5.0,-3.0)	A (0.0,0.0)	4	(4.80, -2.98) (3.48, -2.08) (2.74, -0.1) (0.13,0.00)



(a)



(b)



(c)

Figure 6.17: Implementation stage of global path planning

6.5.3 Human Avoidance

Here the ability of the navigation system to successfully avoid humans and dynamic obstacles within cluttered environments is tested. The robot task is to navigate from its initial positions (0,0) to its goal (5,0) and to avoid human (3,0) moving in the opposite direction as shown in Figure 6.18.a. Figure 6.18.b shows that the robot turns left to avoid the human moving towards it and then continues to its goal after the successful avoidance action. More details

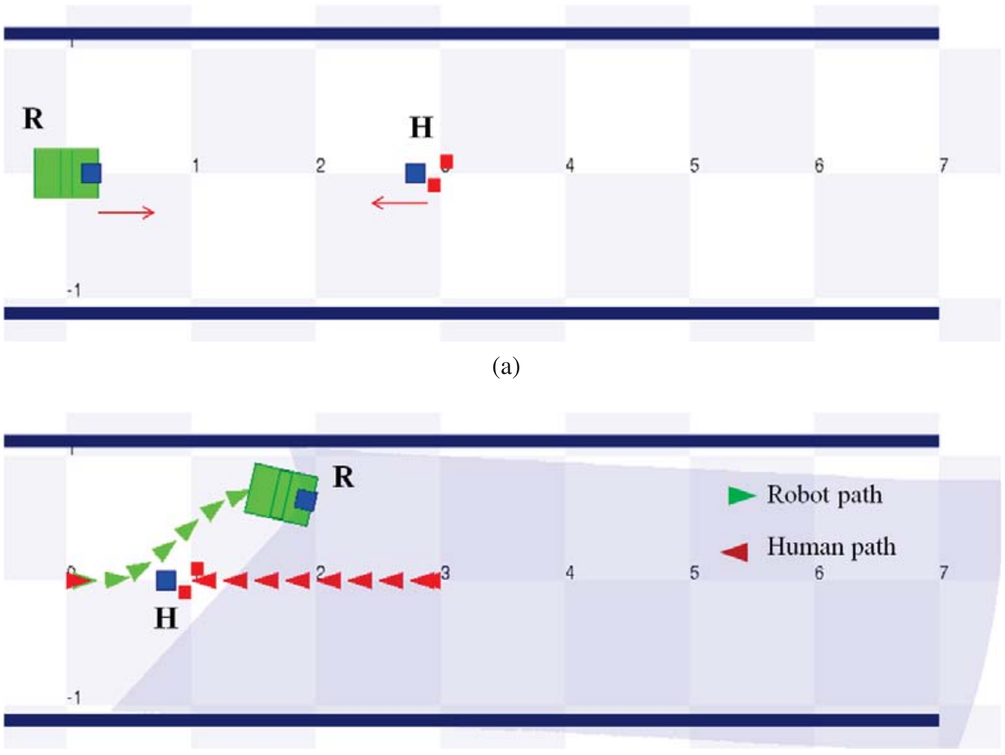


Figure 6.18: Passing a human in a corridor

of the robot path are shown in Figure 6.19, where the robot begins to turn left at $t = 1$s and resumes its path to its goal at $t = 4$s after the human passes by the robot .

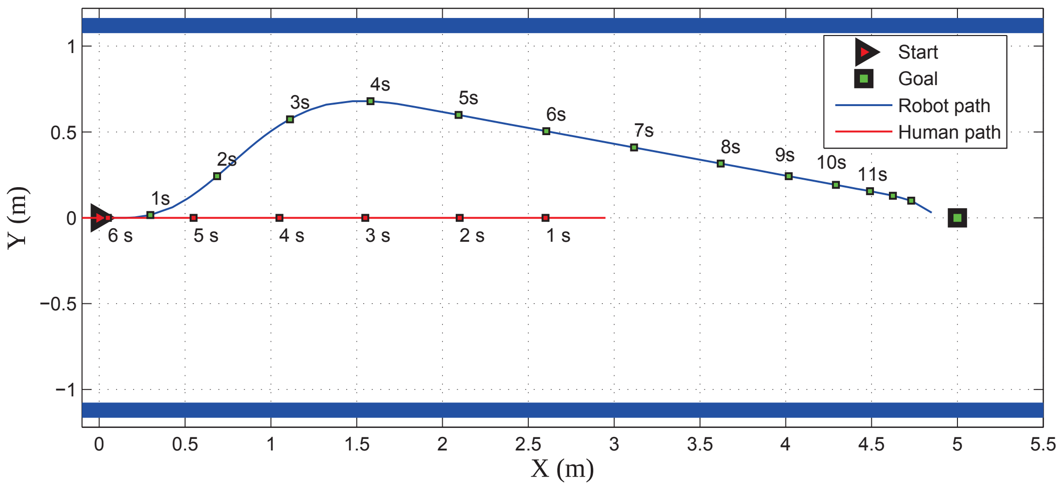


Figure 6.19: Robot avoided a human moving towards the robot

In the following test, the robot (3.5, 1.5) is tasked to cross a corridor with a human (5.5, -0.5) walking along the corridor (see Figure 6.20). It can be observed from Figure 6.21 that the robot reduces its speed at $t = 3$s to allow the human to pass, and then increases its speed after the human passes at $t = 5$s.

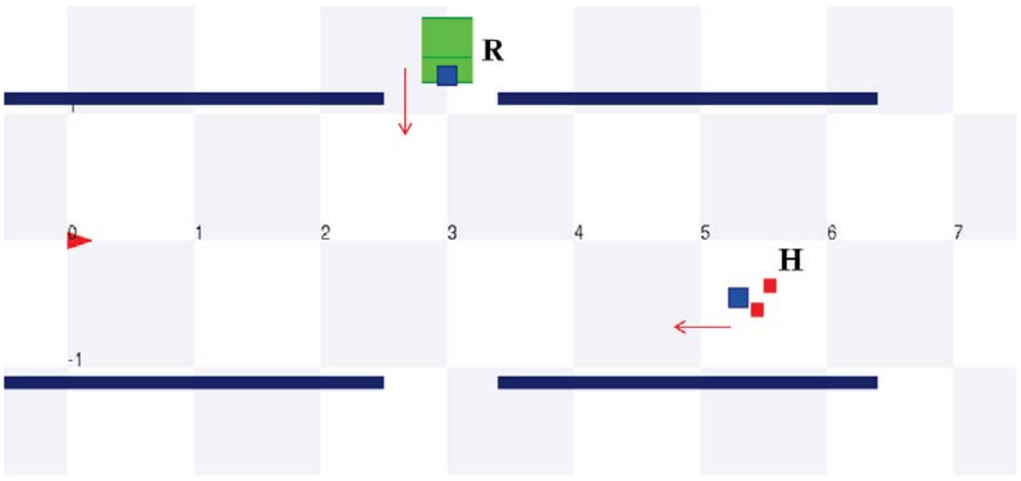


Figure 6.20: Avoiding a human during passing in a corridor

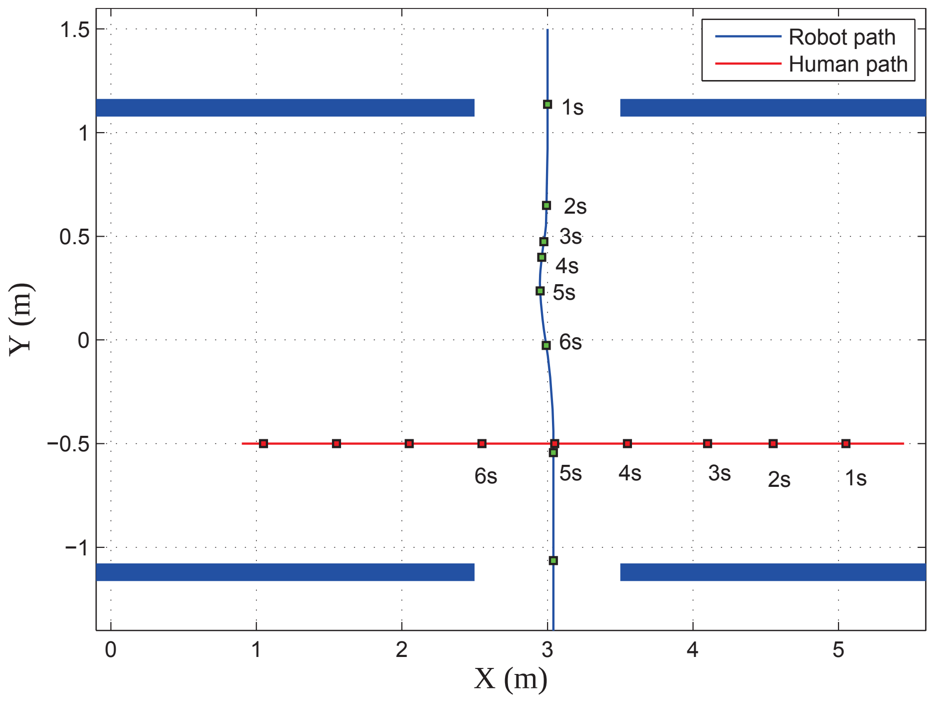
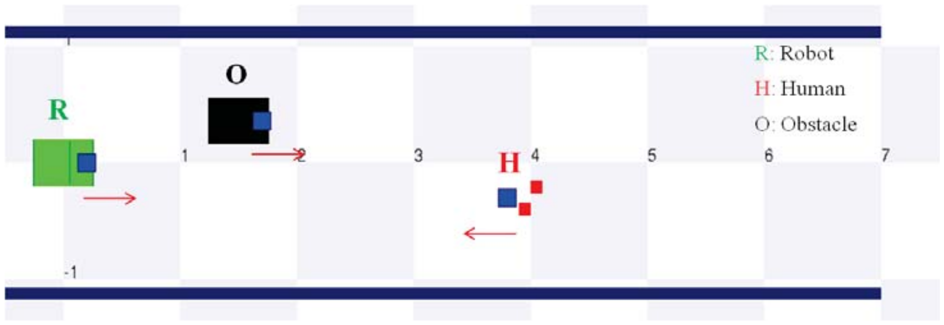
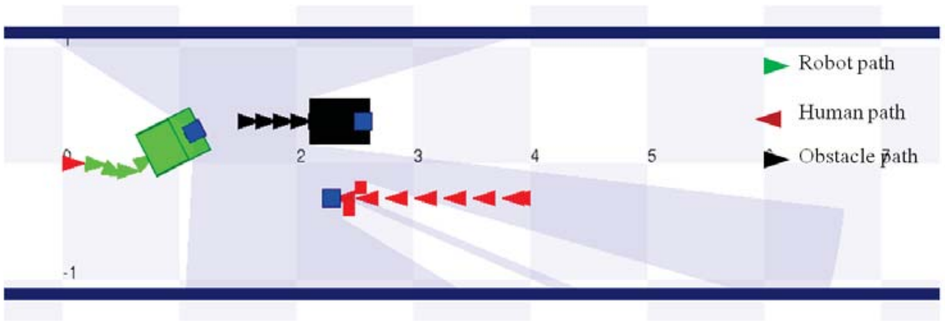


Figure 6.21: Robot path for avoiding a human during passing a in corridor

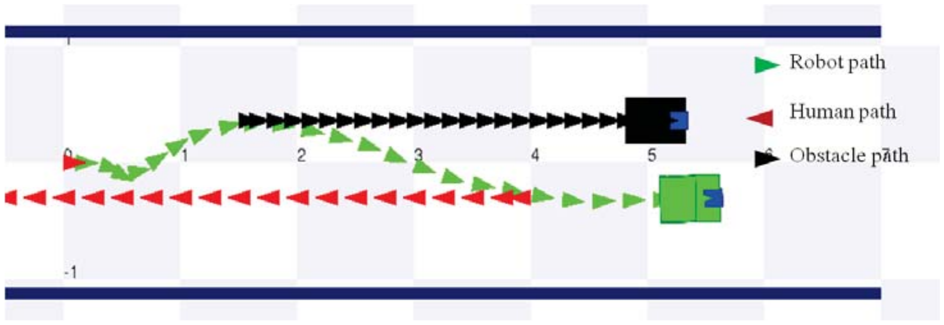
In the third simulation, a dynamic obstacle is located at (1.5, 0.3) with a speed of 0.3 m/s and moving in the same direction as the mobile robot, and simultaneously a human (4, -0.3) moves in the opposite direction with a speed of 0.5 m/s as shown in Figure 6.22.a. The robot having observed the human coming towards it, turns left to follow the dynamic obstacle (Figure 6.22.b) before resuming its path to avoid the dynamic obstacle (see Figure 6.22.c).



(a)



(b)

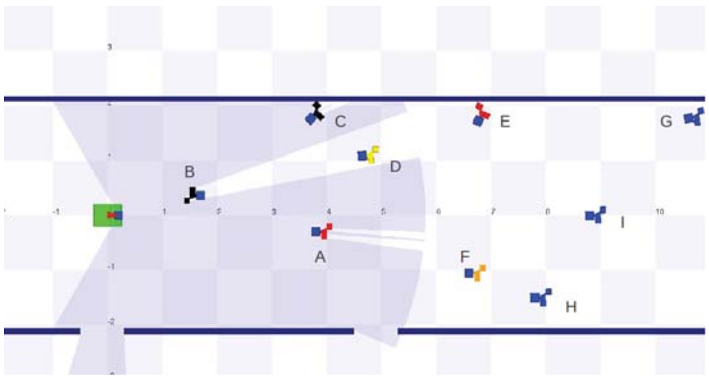


(c)

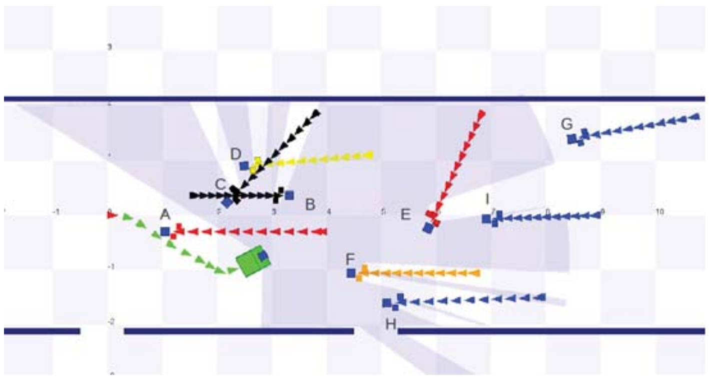
Figure 6.22: Avoiding a human and dynamic obstacle

A simulation test which was carried out to avoid two humans can be seen in appendix C.

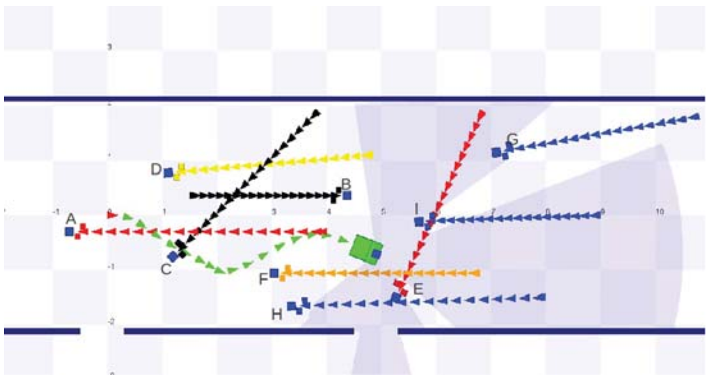
In the fourth simulation, the navigation system was tested to avoid numerous persons moving with different speeds and directions. The initial positions of the robot and humans are shown in Figure 6.23.a and listed in Table 6.4. Figure 6.23.b shows that the robot turns right to avoid A and B, after which it turns left to avoid colliding with F and H, turning right again to avoid humans B, I, and G (Figure 6.23.c) . Figure 6.23.d shows the full path of the robot which is able to successfully complete its mission.



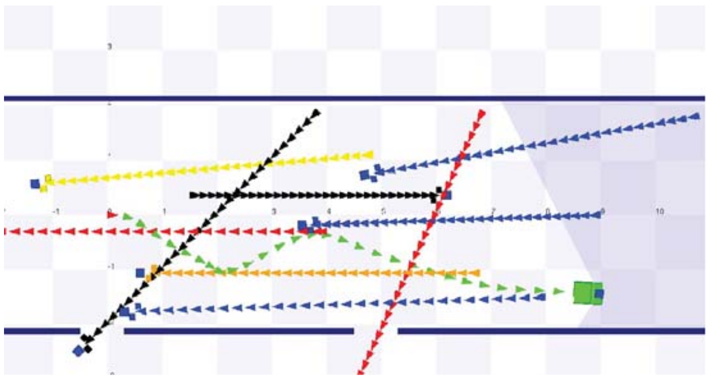
(a)



(b)



(c)



(d)

Figure 6.23: Avoiding multiple humans

Table 6.4: initial position and speeds of the humans

Actors	Position (x (m), y(m))	direction degree	Speed m/s
A	(+4.00,+0.30)	180	0.50
B	(+1.5,+0.36)	0.00	0.30
C	(+3.85,+1.90)	-135	0.40
D	(+4.85,+1.10)	185	0.40
E	(+6.85,+190)	-115	0.40
F	(+6.80,-1.06)	180	0.40
G	(+10.8,+1.80)	190	0.40
H	(+8.00,-1.50)	182	0.50
I	(+9.00,0.000)	182	0.350

6.6 Summary

In this chapter, the human motion was modelled using the Player/Stage simulation. A method was developed for tracking humans using a single laser sensor, where laser clusters are classified into four classes. A tracking algorithm was developed to allow the robot to follow a moving human and maintain a safe distance depending on the target speed. The simulation tests showed that the mobile robot using the proposed navigation system and a single laser sensor is successfully able to track and avoid humans and both static and dynamic obstacles within indoor environments.

Chapter 7

Implementation and Evaluation

The navigation system proposed in this project has been extensively tested using the Player/Stage simulation platform as mentioned in earlier chapters. The simulation results showed that the mobile robot is able to detect and avoid both static and dynamic obstacles (including humans), and minimize the risk of collision with objects which may suddenly appear in the robot's path. This chapter focuses on implementing and evaluating the proposed navigation system using physical robots in cluttered indoor environments.

7.1 Experimental Procedures

As mentioned in the third chapter, the experiments were implemented using two mobile robots; a modified Pioneer robot (intelligent robot) and the Nubot robot (dynamic obstacle) developed in Newcastle University (see Figure 3.2). In this project, only the two laser sensors were used to recognize the surrounding workspace. During avoiding dynamic obstacles, the speed and direction of the robot are measured depending on the data collected from the robot encoders.

The navigation algorithm has been written using C++, where its *classes* and *structures* help in reducing the code's complexity. To evaluate the robot performance during navigation, the collected sensor data are saved as text files, which are processed off-line using Matlab. The sampling time of the navigation algorithm is chosen to be 0.1s which coincides with the update rate of the 10Hz Hokuyo laser sensors. The maximum speed of the robot is chosen as 0.5m/s for achieving safe navigation. An HP laptop was used as a client to execute the navigation algorithm, and to collect data from the sensors through USB ports.

All experiments were conducted within two indoor environments. The first comprised an open and uncluttered laboratory space 8.0m *x* 3m in which hardboard panels were used to simulate walls. The second space comprised a more complex network of corridors and doorways, namely the ground floor of Stephenson building at Newcastle University.

The experiments were designed to establish the following:

- A physical mobile robot using the proposed navigation system is able to successfully avoid static obstacles such as walls, and to pass safely through narrow gaps without an oscillatory behaviour as in the potential field algorithm.
- The robot is able to avoid dynamic obstacles. As mentioned previously, obstacles are classified into three types: non-intelligent obstacles (not programmed to avoid obstacles), semi-intelligent obstacles (able to avoid static obstacles) and intelligent obstacles which are able to detect and estimate the speeds of dynamic obstacles and take optimal avoiding actions to minimize the risk of collision.
- The robot can identify locations form which moving obstacles may appear suddenly, and then is able to take appropriate avoidance manoeuvre to minimize risk of collision.
- The robot can successfully navigate within a workspace cluttered with humans,where the low-level (0.4m) Hokuyo laser sensor can detect human legs at only relatively short distance, thus, another laser sensor was fitted at a height of 0.70m (as illustrated in Figure 3.2) to detect the human upper torso, to increase the distance and time to detect humans.

7.2 Performance Metrics

To quantify the performance of the robot during its navigation, and to allow comparison, three sets of performance metrics have been adopted namely: to consider the potential for collision, the travel distance and time to the goal, and the quality of the robot motion (smoothness).

The following six parameters, derived from [Ceballos et al., 2010], are utilized to quantify the performance of the robot navigation system:

- Minimum time to collision (t_{min}) in avoiding obstacles, the collision risk not only depends on distance, but also on the relative speed of the robot, and hence, it is necessary to consider the time to collision t_c and minimum collision risk t_{min} which are defined as follows (see Figure 7.1):

$$t_c(i) = \frac{d_c(i)}{\triangle V(i)} \tag{7.1}$$

$$t_{min} = min(t_c) \tag{7.2}$$

where d_c, $\triangle V$ and t_c refer to the collision distance, relative velocity and collision time, respectively.

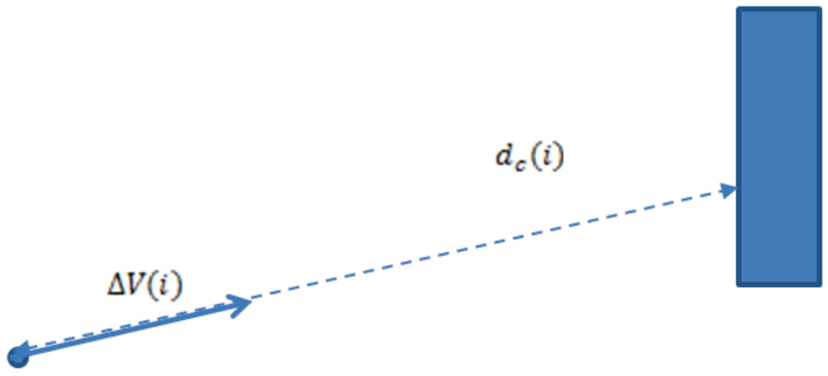


Figure 7.1: Collision Risk

- Minimum distance to collision (d_{min}) refers to the minimum perceived distance between the robot and any obstacle, and is given as:

$$d_{min} = min(d_c) \tag{7.3}$$

- Total travel time (T_t) represents the time consumed during the robot's navigation to reach its goal, and is inversely proportional to the average speed of the robot during its journey, and as T_t decreases, the average speed increases. It can be written as:

$$T_t = n.T_s \tag{7.4}$$

where T_s and n are the sampling period and the total number of cycles respectively.

- Path length (P_L) refers to the total distance travelled by the robot to achieve its goal, and is computed as follows:

$$P_L = \sum_{i=1}^{i=n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{7.5}$$

where x and y represent the robot coordinates.

- Smoothness is measured by the bending energy B_E; where less bending energy indicates a smoother path, and for a straight line motion, the bending energy should be zero. To compute B_E the curvature of the robot path must be calculated, where the curvature k is a measure of how quickly a tangent line turns on the curve (see Figure 7.2) i.e it is the rate of change of the tangent angle ψ as a function of arc length s (see Appendix E for more details).

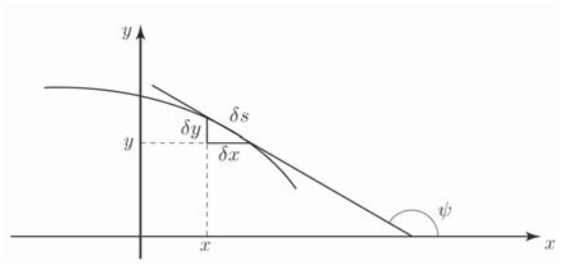


Figure 7.2: Curvature of a curve

The curvature k at a point (x_i, y_i) of the robot path and the bending energy can be defined as follows:

$$k(x_i, y_i) = \frac{y_i^{''}}{(1 + (y_i^{'})^2)^{3/2}} \ \ (\text{rad/m}) \tag{7.6}$$

$$B_E = \frac{1}{n} \sum_{i=1}^{i=n-1} k^2(x_i, y_i) \ \ (\text{rad}^2/\text{m}^2) \tag{7.7}$$

where

$$y_i^{'} = \frac{\triangle y_i}{\triangle x_i} \tag{7.8}$$

$$\triangle y_i = y_i - y_{i-1} \tag{7.9}$$

$$\triangle x_i = x_i - x_{i-1} \tag{7.10}$$

- Number of collision Nc: this refers to the number of collision events during experiments. Note each experiment in this study is repeated five times, and the previous factors are represented by their average values.

7.3 Ability to Avoid Static Obstacles

The robot may be required to work in environments which are free of dynamic obstacles. In this case the robot has to avoid only static obstacles. However, passing through a narrow space is the most challenging for the mobile robot. As mentioned in the literature review several avoidance algorithms were designed to avoid static obstacle have a problem of the oscillatory behaviour. Hence, In order to evaluate the robot performance while avoiding static obstacles only the obstacle avoidance and emergency stop behaviours are used.

7.3.1 Definition of Scenarios

The Pioneer robot navigates from its current position (0,0) to its goal (5,0) with maximum linear and angular velocities of 0.5m/s and 100 degree/s respectively.

Three different scenarios were evaluated with an increasing level of complexity as follows:

Scenario 1 (SS1) : the robot navigates in a relatively free open area, where there is no obstacle preventing the robot from implementing a straight line path to its goal.

Scenario 2 (SS2) : the robot has to navigate a narrow gap and can implement a straight line path to its goal.

Scenario 3 (SS3) : the robot navigates a narrow gap and can not implement a straight line path to its goal.

7.3.2 Implementation of Scenarios

Scenario 1 (SS1)

This test aims to define the values of the performance metrics when there is no collision risk (Figure 7.3 and 7.4). It can be observed in Figure 7.5 that the robot accelerates achieving the maximum speed (0.5m/s) at x=0.5m which is maintained till the robot becomes close to its goal, where the speed is decreased to stop the robot at its goal.

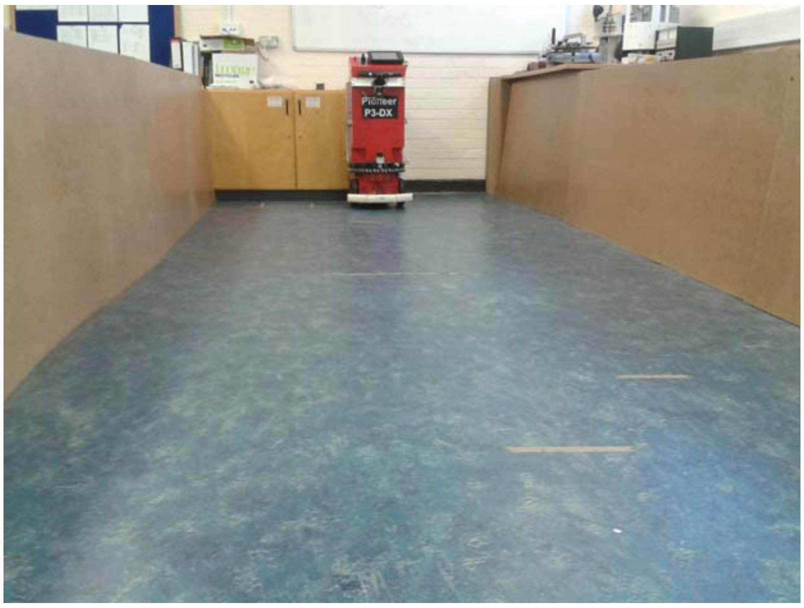


Figure 7.3: Pioneer navigates in a free open area

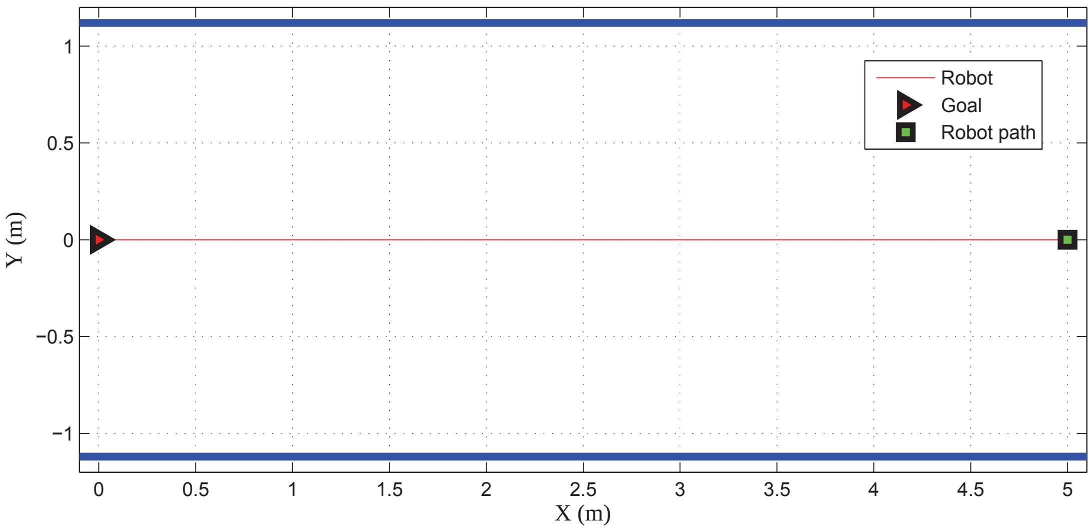


Figure 7.4: Straight line path

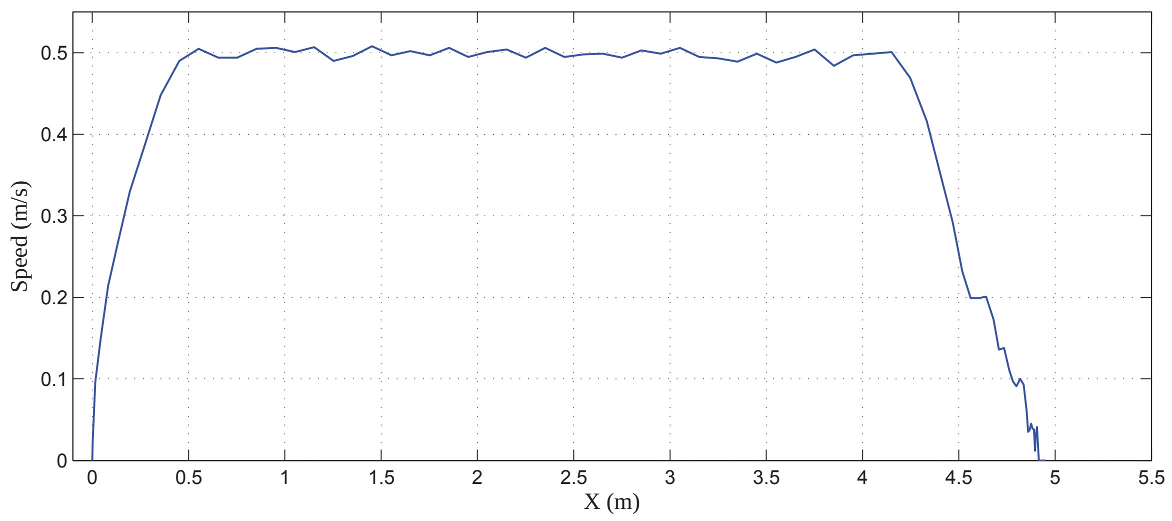


Figure 7.5: Robot speed in a relatively free open area

Scenario 2 (SS2)

Sometimes the mobile robot may be required to pass through an open door or narrow gap in order to achieve its mission (see Figure 7.6).

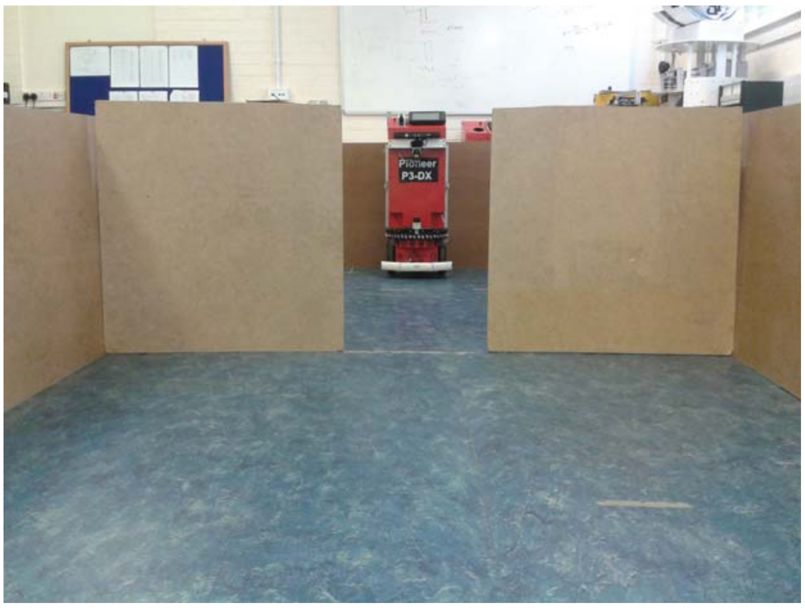


Figure 7.6: Passing a narrow gap in a straight path

Figure 7.7 shows that the mobile robot (Pioneer P3-DX) using the proposed navigation system navigates successfully from its starting point to its goal, implementing a straight line while passing a narrow gap located at $x = 2.60$m. Note that the robot has a safe gap of 75 mm on each side whilst passing through the narrow space with the widths of the robot and narrow gap are 0.45m and 0.6m respectively, where using the laser sensor enables the obstacle avoidance behaviour to detect that the narrow space in enough for the robot to pass without any collision risk, therefore, the average speed of the robot as in SS1 is maintained at 0.5m/s (Figure 7.8).

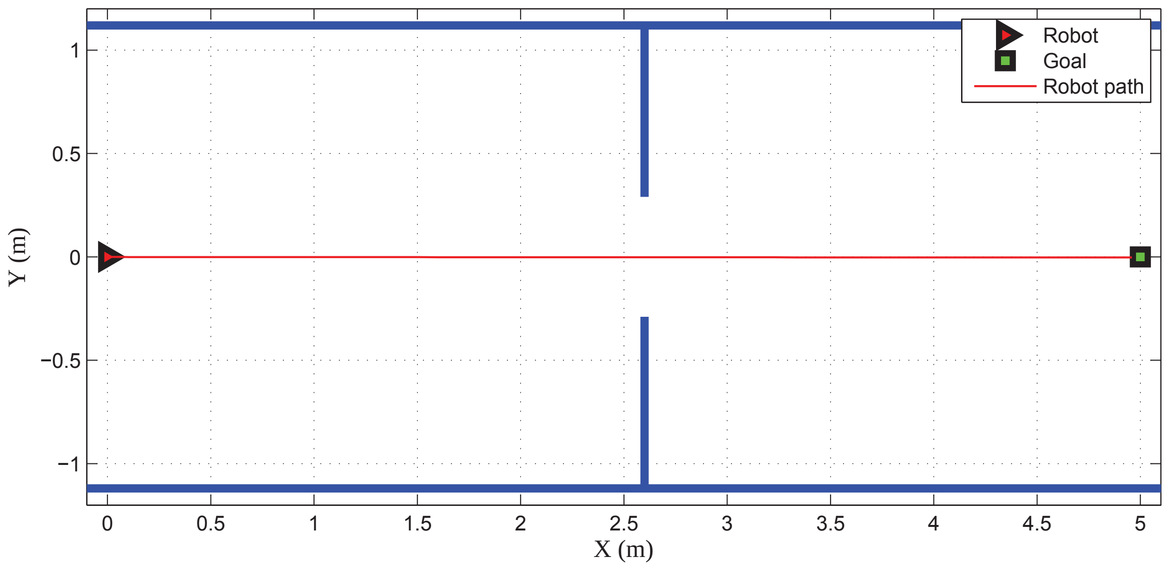


Figure 7.7: Robot passes a narrow gap in a straight path

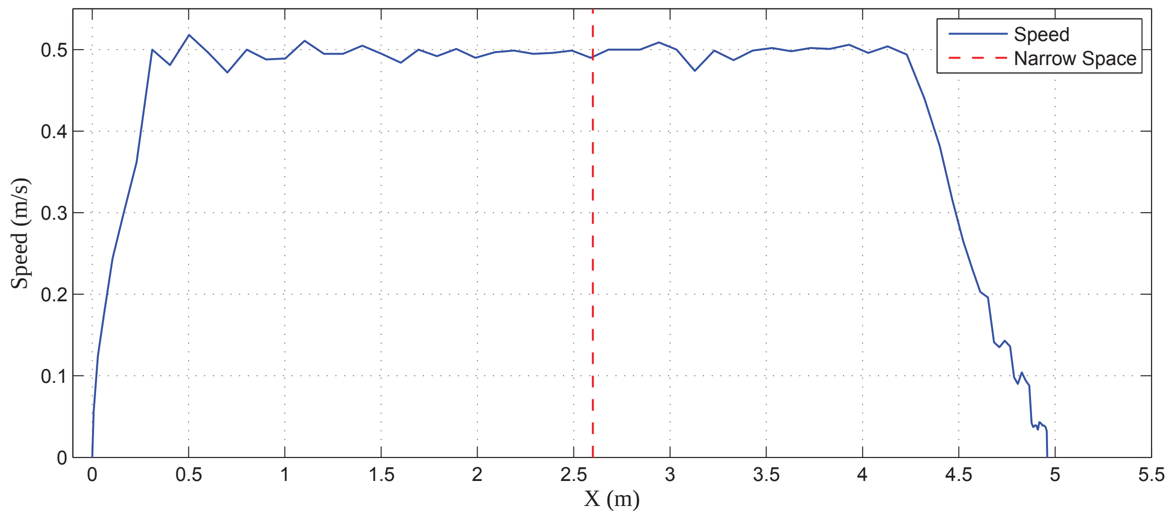


Figure 7.8: Robot speed while passing a narrow gap

Scenario 3 (SS3) :

In this scenario, the complexity of the robot's task is increased using a static obstacle which prevents the robot from implementing a straight line path to its goal (see Figure 7.9). As illustrated in Figure 7.10, the Pioneer robot implements an approximately straight line while passing the narrow gap located at $x = 2.60$m, and then turns to its goal (5,0) in a curved path. Figure 7.11 shows that the robot speed decreased to 0.35m/s while passing the narrow gap, after which it increased before decelerating to stop the robot at its goal.

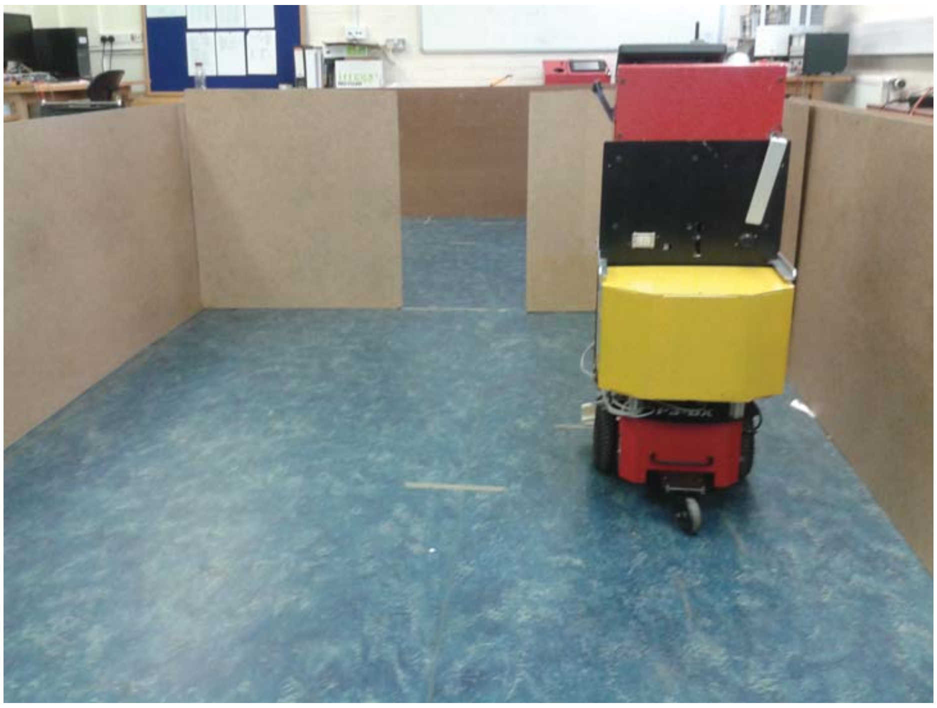


Figure 7.9: Passing a narrow gap

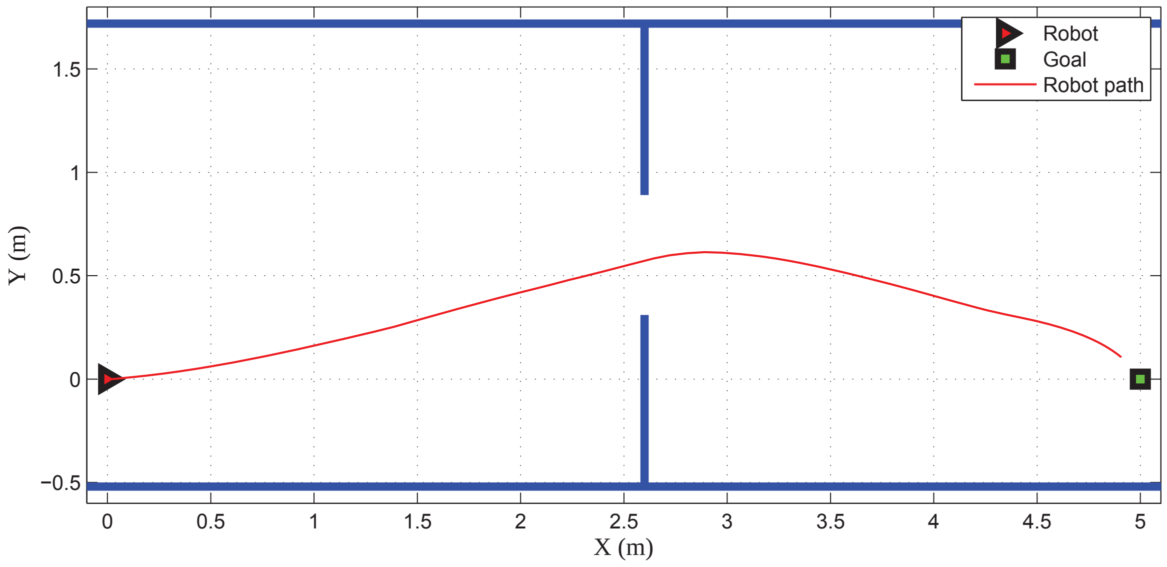


Figure 7.10: Robot passes a narrow gap

Table 7.1 illustrates the performance metrics of the mobile robot during the three scenarios, where the total time $T_t(s)$ and path length P_L in SS3 are greater than that when implement straight lines paths in both SS1 and SS2, and also it can be noted that there is no collision risk in both SS1 and SS2, while its is relatively very high in SS3 although there was no dynamic obstacles which restrict the robot motion, also the robot paths in both SS1 and SS2 is smoother that in SS3 in which the robot achieved a curved path.

Table 7.1: Performance metrics for avoiding static obstacles

avoiding static obstacles	$T_t(s)$	P_L(m)	B_E (rad^2/m^2)	t_{min}(s)	d_{min}(m)	N_c
SS1	13.5	4.91	0	-	-	0
SS2	13.7	4.92	0	-	-	0
SS3	14.6	5.05	0.95	0.46	0.22	0

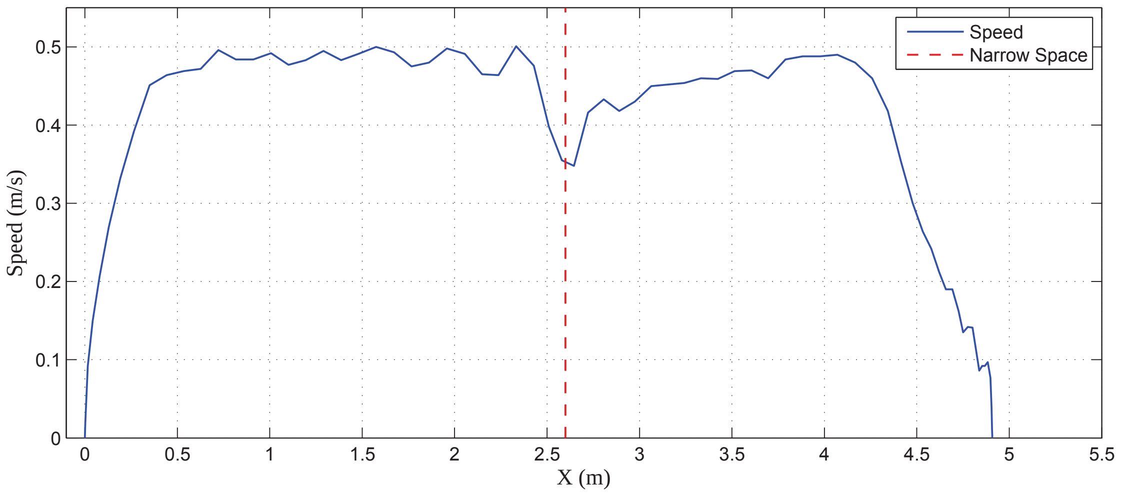


Figure 7.11: Robot speed while passing a narrow gap

To explain if the robot has a problem of the oscillatory behaviour let evaluate the changes in the robot's direction shown in Figure 7.12, in which there is no change in the robot's direction during SS1 and SS2, while in SS3, the robot starts to turn anti-clockwise, then it maintains its direction while passing the narrow gap, and then turns clockwise to reach its goal. This verify that the navigation system does not produce oscillatory behaviour in the robot motion while passing narrow spaces. Note decreasing the sampling time of the control system will make the changes in the robot's angle in SS3 much smoother.

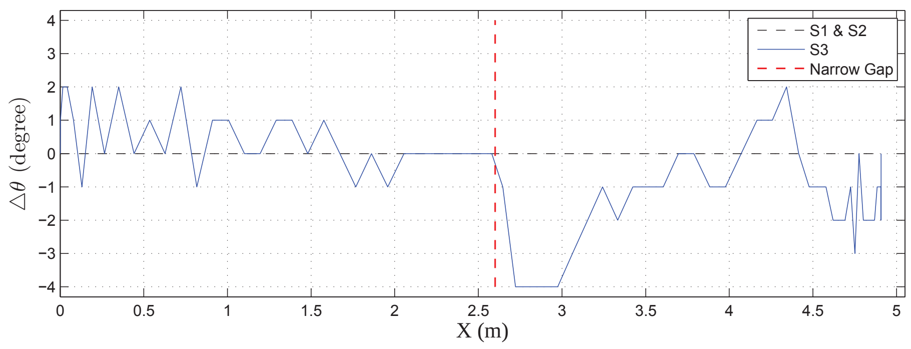


Figure 7.12: Changes in the robot's direction

7.4 Dynamic Obstacle Avoidance

The previous three experiments illustrated that the mobile robot can be employed to implement successful navigation tasks among static obstacles. However, the robot may encounter some mobile robots which are programmed in different methods to reach their goals. To evaluate the performance of the proposed navigation system during avoiding dynamic obstacles, the Pioneer robot has to navigate from its current position (0,0) to its goal (5,0), while the Nubot robot is used as a dynamic obstacle which was located at (5,0) and moves in the opposite direction to the Pioneer robot (Figure 7.13).

7.4.1 Definition of Scenarios

Three different scenarios were carried out with increasing the intelligent level of the dynamic obstacle as follows:

Scenario 1 (DS1) : the Nubot robot is used as a dumb obstacle moving at a fixed speed of 0.5m/s.

Scenario 2 (DS2) : the Nubot robot is programmed to avoid obstacles using the PFA algorithm and a laser sensor.

Scenario 3 (DS3) : the Nubot is used as an intelligent obstacle programmed with a navigation algorithm as the Pioneer robot.

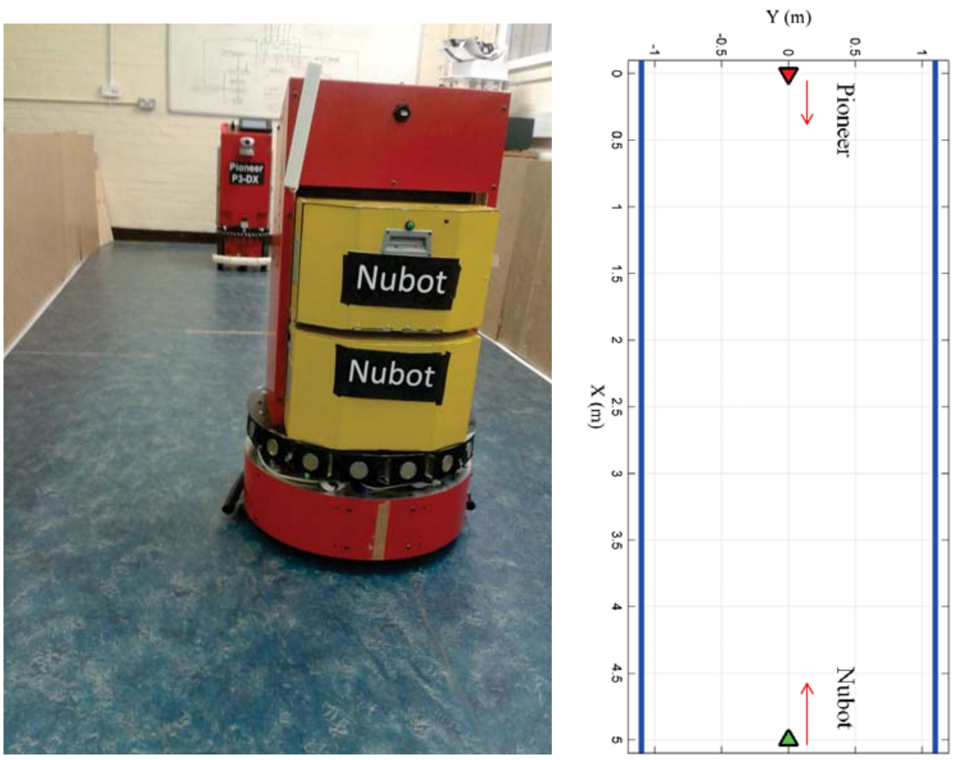


Figure 7.13: Pioneer P3-DX and Nubot

7.4.2 Implementation of Scenarios

Scenario 1 (DS1)

The mobile robot may face some dynamic obstacles which move on predefined path without using an obstacle avoidance algorithm, for example a mobile robot follows a black line [Hasan et al. (2012)]. Here the Pioneer robot successfully detects and avoids a dumb obstacle (Nubot) as shown Figure 7.14, in which the robot initially moves on a straight line towards its goal, where the relatively long distance between the robot and the obstacle produces very low collision risk. However, the robot begins to avoid the dumb obstacle at $t = 3.6$s, and then resumes its path to the goal at $t = 7.5$s after the dumb obstacle passes.

Scenario 2 (DS2)

The Nubot robot was reprogrammed to avoid obstacles using the PFA algorithm. Figure 7.15 shows how the Pioneer robot begins avoiding the Nubot robot at $t = 3$s, and then returns toward its goal after Nubot passes. It can be observed that the Nubot (low-level intelligent obstacle) carried out a small avoidance manoeuvre to avoid the Pioneer robot at $t = 5$s. Clearly the Pioneer robot started the avoidance manoeuvre 2s before the Nubot robot.

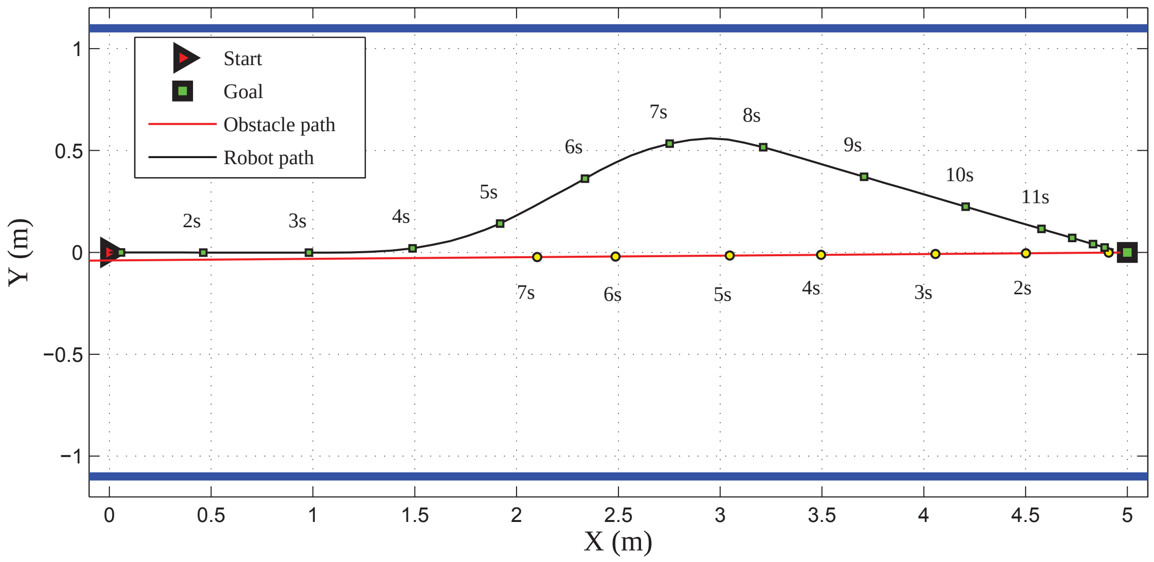


Figure 7.14: Robot avoided a non-intelligent obstacle

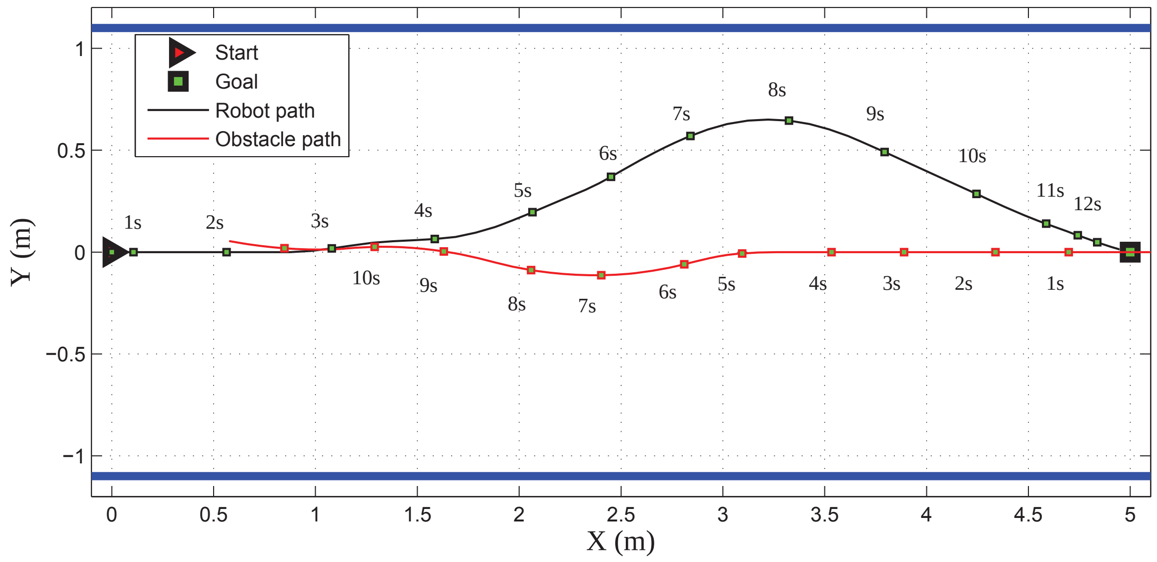


Figure 7.15: Robot passed a low-level intelligent obstacle

Scenario 3 (S3)

In this test, both the Pioneer and Nubot were programmed with the same intelligent navigation algorithm. Figure 7.16 shows that both robots initial their avoidance much sooner (Pioneer at $t = 3.4$s, Nubot at $t = 4$s), and then both resume their respective path to their targets just after $t = 6$s with near symmetrical paths.

It can be noted from the performance metrics listed in Table 7.2 that the robot executed a short and smoother path during avoiding an intelligent obstacle than that for both dumb and low intelligent obstacles, with a significantly reduced risk of collision as evidenced by increased values of t_{min} and d_{min}. The early avoidance action of the intelligent obstacle gave the robot more space and time to make a path with less curvature and relatively long distance (0.56m) to the adjacent wall. While the dumb obstacle and the late avoidance action of the low intelligent obstacle forced the robot to move a way from the straight line to the goal and making relatively short distances to the wall.

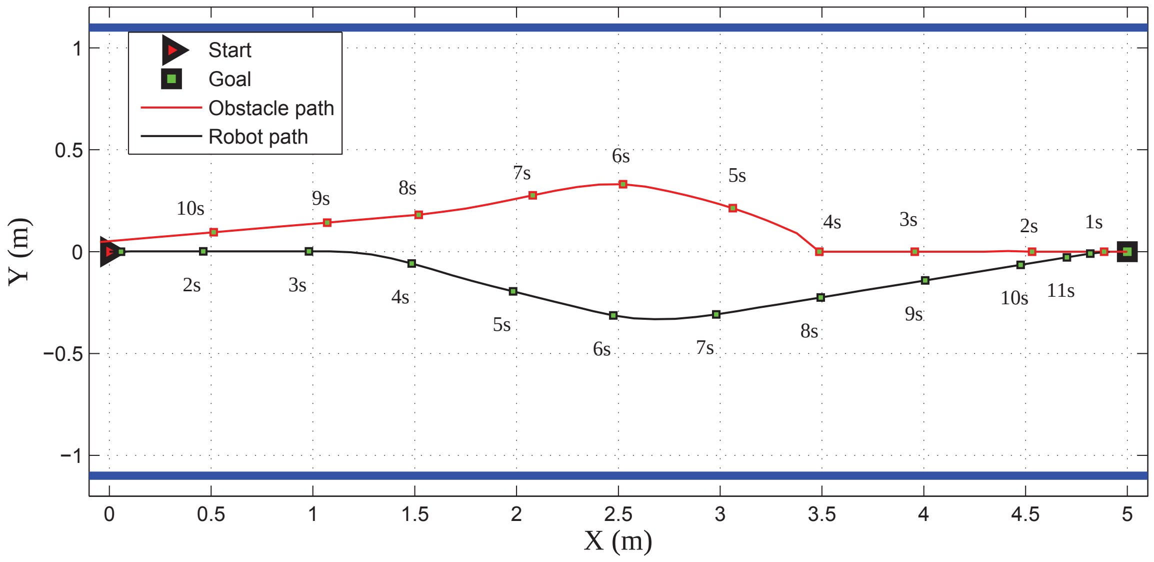


Figure 7.16: Avoiding Intelligent obstacle

Table 7.2: Performance metrics for avoiding a dynamic obstacle

Avoiding dynamic obstacle	$T_t(s)$	P_L(m)	Smoothness (rad^2/m^2)	t_{min}(s)	d_{min}(m)	N_c
DS1 (Avoiding dumb obstacle)	14.3	5.14	1.10	1.22	0.40	0
DS2 (Avoiding low intelligent obstacle)	14.2	5.16	1.22	1.20	0.38	0
DS3 (Avoiding intelligent obstacle)	13.9	5.08	0.81	2.10	1.30	0

7.5 Unobserved Obstacles

The *unobserved obstacle avoidance* (UOA) behaviour allows the mobile robot to detect locations from which dynamic obstacles may suddenly appear, and then apply the virtual obstacle principle to minimize the collision risk. Here, the robot performance using the UOA behaviour is tested in two different situations. First the robot task is to navigate from its initial location (0,0) to its target (5,0) passing in front of an open doorway (Figure 7.17). Second the robot navigates a corridor crossing to its target (7.5,-3.5) and avoids a dynamic obstacle which appears suddenly at the corner (Figure 7.18).

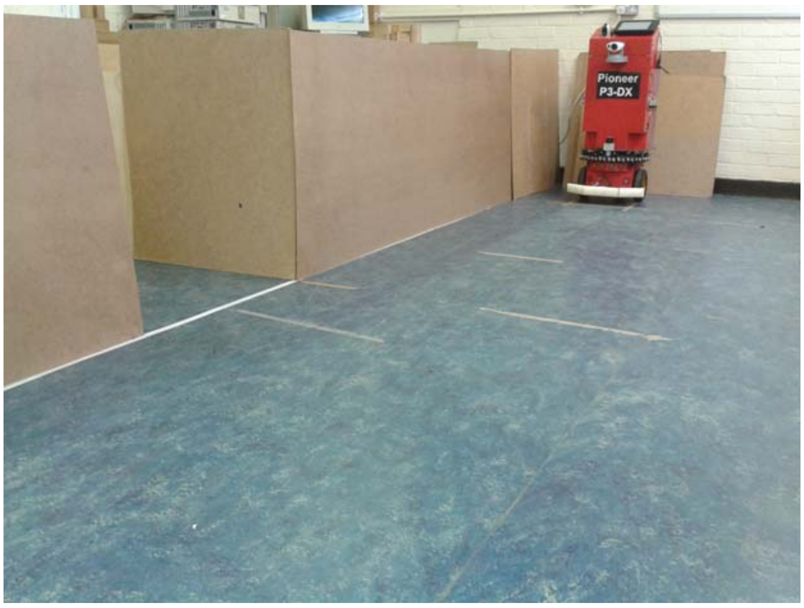


Figure 7.17: Pioneer robot and an opened door

7.5.1 Passing an Open Doorway

Four scenarios were designed for passing an open doorway as follows:

Scenario 1 (OS1) : the Pioneer robot passes the open doorway while the UOA behavior is inactive.

Scenario 2 (OS2) : the UOA behavior is activated.

Scenario 3 (OS3) : the robot avoids a static obstacle which is located opposite the door threshold (Figure 7.19).

Scenario 4 (OS4) : the robot avoids a dynamic obstacle (Nubot) while passing the door threshold.

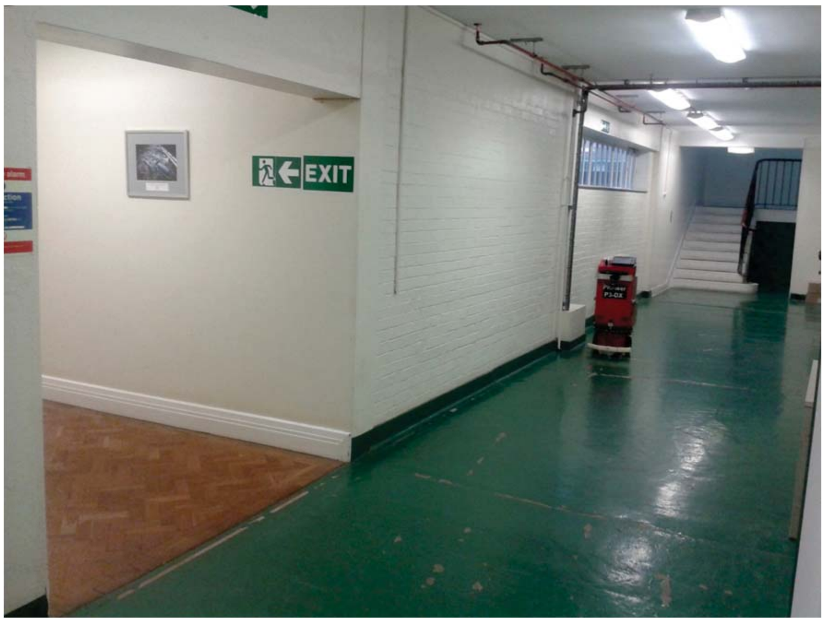


Figure 7.18: Pioneer robot navigates a corridor crossing

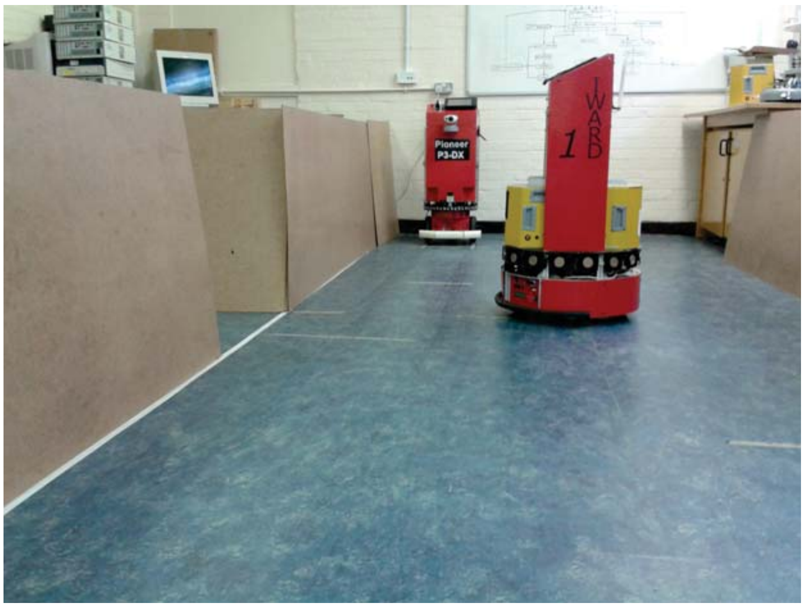


Figure 7.19: Static obstacle is located opposite the door threshold

7.5.2 Implementation of Scenarios

Scenario 1 (OS1)

OS1 aims to show the performance of the robot when the UOA behaviour is inactive. Figure 7.20 illustrates that the obstacle avoidance behaviour does not detect any collision risk preventing the robot to implement a straight line path leaving a safe distance of 0.50m from the adjacent wall and keep its speed at its maximum value during passing the open doorway.

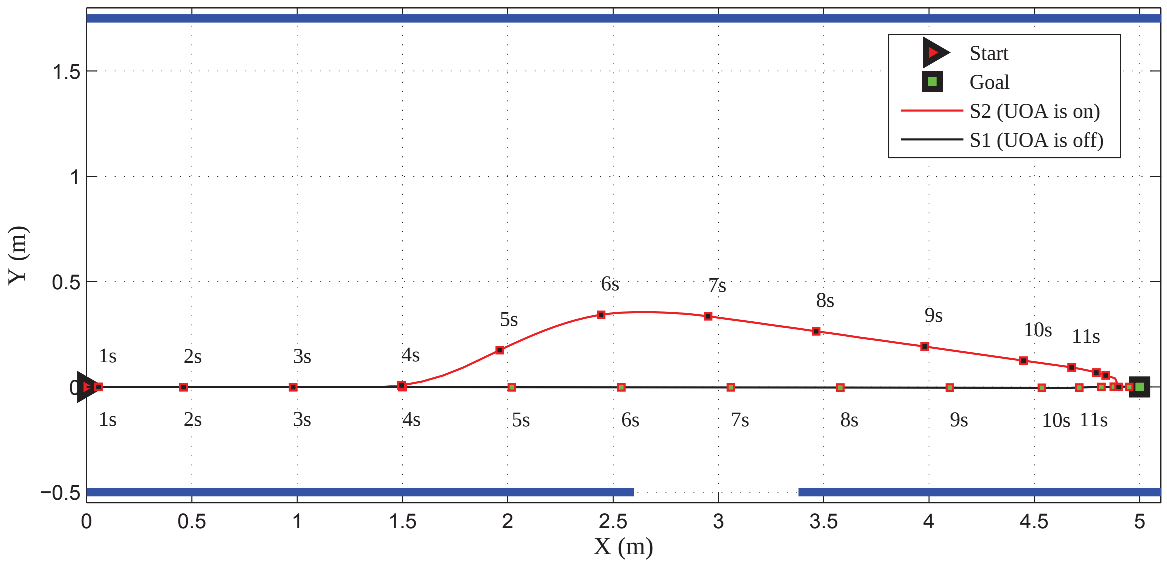


Figure 7.20: Robot path while Passing an open doorway

Scenario 2 (OS2)

When the UOA behavior is activated, the robot begins at $t = 4$s to manoeuvre further away from the wall leaving a safe distance of 0.85m as it anticipates a potential collision and then it resumes its path to the goal after the door threshold is passed (see Figure 7.20). It can be noted that the robot direction and the relatively long distance to the door threshold minimizes the collision risk with obstacles which may suddenly appear from the opened door.

Scenario 3 (OS3)

A static obstacle was located opposite the door threshold. It can be observed from Figure 7.21 and 7.22 that the robot executes a straight line path at speed of 0.5m/s until $x = 2.5$m, whereupon it decreases its speed to 0.1m/s while negotiating the door threshold and static obstacle, before accelerating back to 0.5m/s. Here, the low speed while passing the door threshold minimizes the collision risk and permits the robot to stop quickly before colliding with any obstacle which may suddenly appear.

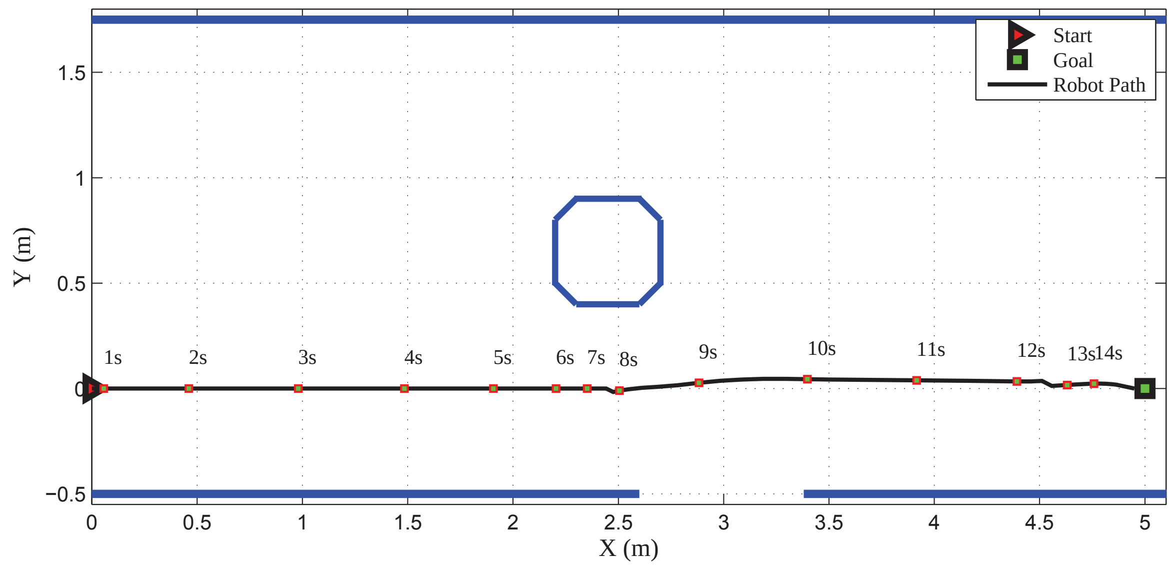


Figure 7.21: Robot avoiding a static obstacle at a door threshold

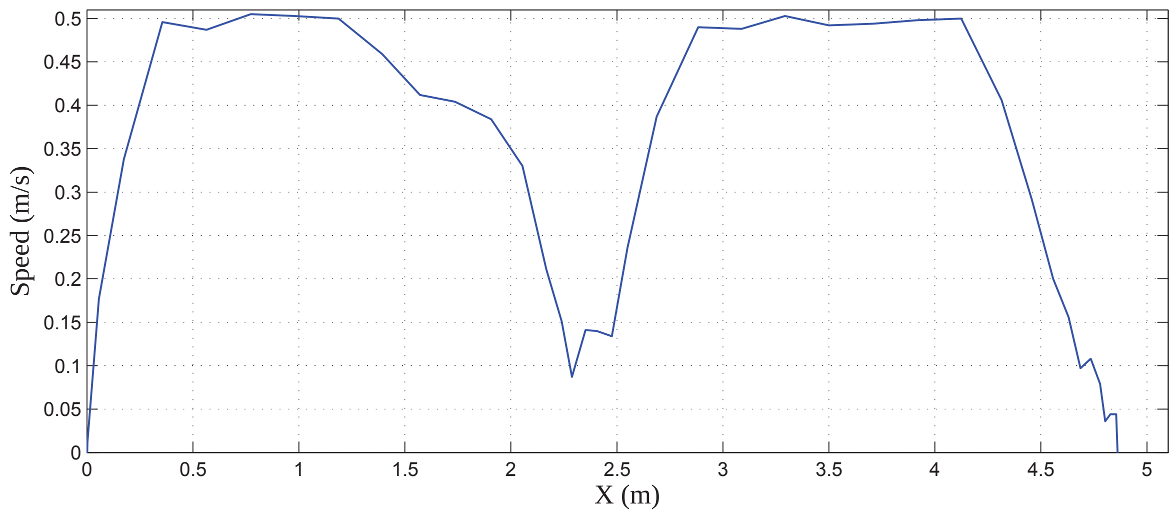


Figure 7.22: Robot speed while avoiding a static obstacle at a door threshold

Scenario 4 (OS4)

The complexity of the robot task was increased by introducing a dynamic obstacle (Nubot) moving at a speed of 0.4m/s. Figure 7.23 shows that the robot starts to move away from the door threshold at $t = 4.3$s, but then turns to avoid the oncoming obstacle whilst maintaining a safe distance of 0.64m to the door threshold. It can be observed form Figure 7.24 that the robot speed decreases to 0.24m/s while passing the door threshold, and then increases after the robot had passed by the door threshold.

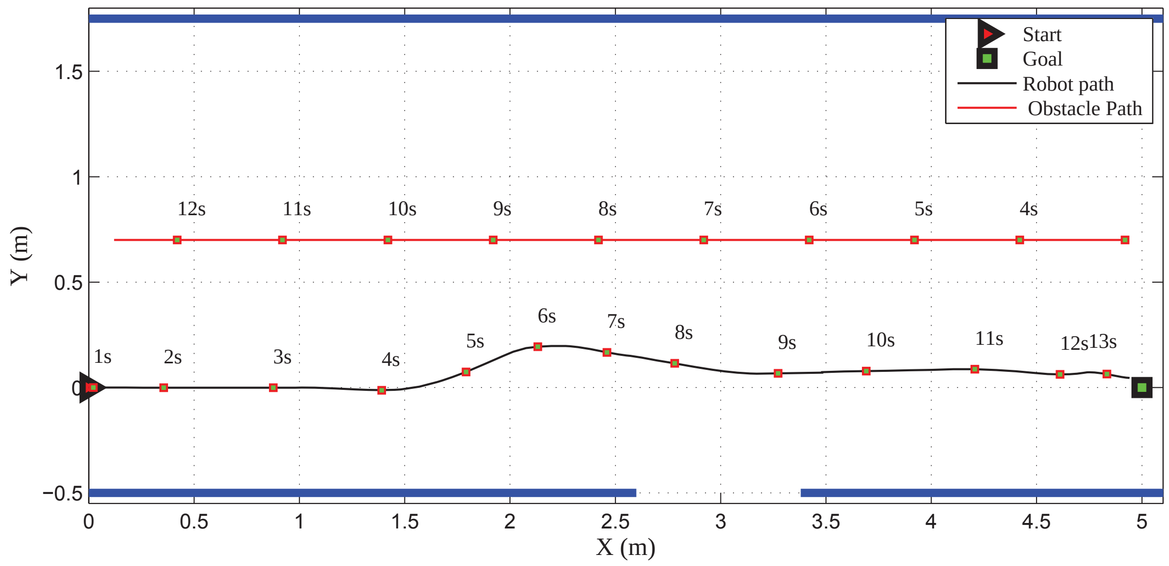


Figure 7.23: Robot avoids a dynamic obstacle at a door threshold

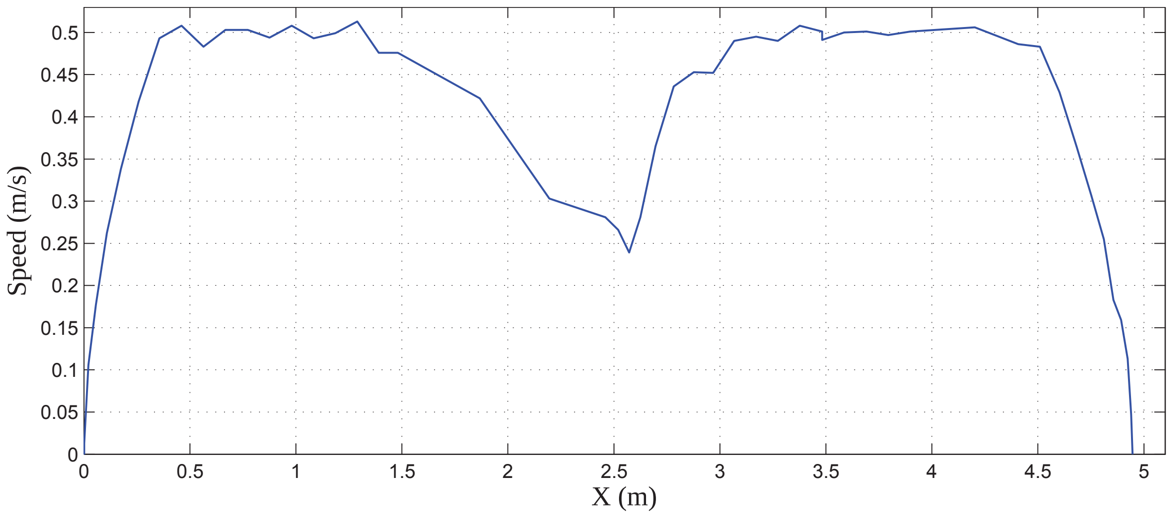


Figure 7.24: Robot speed while avoiding a dynamic obstacle at a door threshold

It can be noted from Table 7.3 that the robot had the highest collision risk in OS1 when the UOA behaviour was inactive, although the path is the smoothest. Whilst OS2 has the maximum travel distance, where the relatively free space allowed the robot to move away from the door threshold with a relatively high speed. OS3 has the maximum travel time, where the static obstacle prevented the robot from moving away from the door threshold, therefore the robot reduced its speed to minimize risk of collision. In OS4, the robot manged to balance between avoiding the dynamic obstacle and minimizing the collision risk with any obstacle which may have appeared suddenly.

Table 7.3: Performance metrics for avoiding a door threshold

Open door avoidance	$T_t(s)$	P_L(m)	Smoothness (rad^2/m^2)	t_{min}(s)	d_{min}(s)	N_C
OS1	13.60	4.91	0.002	0.14	0.27	0
OS2	14.30	4.99	0.78	4.4	2.00	0
OS3	15.10	4.93	0.25	2	0.6	0
OS4	14.80	4.95	0.88	1.6	1.02	0

7.5.3 Navigating a Corridor Crossing

Three scenarios were conducted to evaluate the robot performance during maneuvering a blind corner. In the first scenario, the robot navigates the corridor crossing in a free-dynamic area, while in the second and third scenarios, the robot avoids a dumb obstacle moving at a fixed speed of 0.5m/s and appearing suddenly from a blind corner.

7.5.4 Implementation of Scenarios

Scenario 1 (CS1):

Figure 7.25 illustrates the robot maneuvering the blind corner, where the robot implements a straight line until t =9.5s, after which it directs its path away from the corner to allow sufficient reaction time and distance to any obstacle which may navigate the corridor crossing, and then it resumes its path to its goal after passing the corner.

Scenario 2 (CS2) and Scenario 3 (CS3):

Figure 7.26 and Figure 7.27 illustrate two cases (CS2 and CS3 respectively) of how the robot takes suitable avoidance actions depending on the proximity and speed of an obstacle, relatively to its own position and speed. If the obstacle is perceived early enough, the robot can negotiate its path ahead of the obstacle without slowing down as shown in Figure 7.26, in which the obstacle appeared in the laser data at $t = 14.2$s and disappeared at $t = 18$s. While if there is no enough time and space for the robot to resume its path as in CS3, it must slow down to allow the object to pass safely with consequential longer journey time, but corresponding short path (see Table 7.4). It can be observed from Figure 7.27 the robot reduces its speed to avoid the dynamic obstacle which appears in the laser data at $t = 13.2$s, and after it safely passes, the robot begins to increase its speed and resume its path.

Table 7.4: Performance metrics for avoiding a unobserved obstacle

	$T_t(s)$	P_L(m)	Smoothness (rad^2/m^2)	t_{min}(s)	d_{min}(m)	N_C
CS1	28.40	11.40	0.64	2.03	1.67	0
CS2	31.20	11.90	0.92	0.7	0.8657	0
CS3	32.85	11.20	0.90	0.61	0.59	0

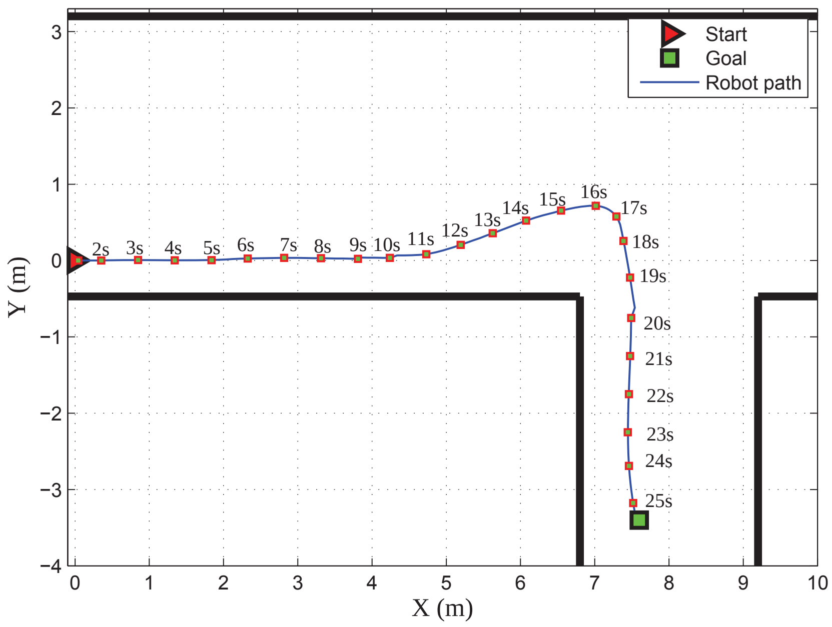


Figure 7.25: Minimizing the collision risk at a corridor corner

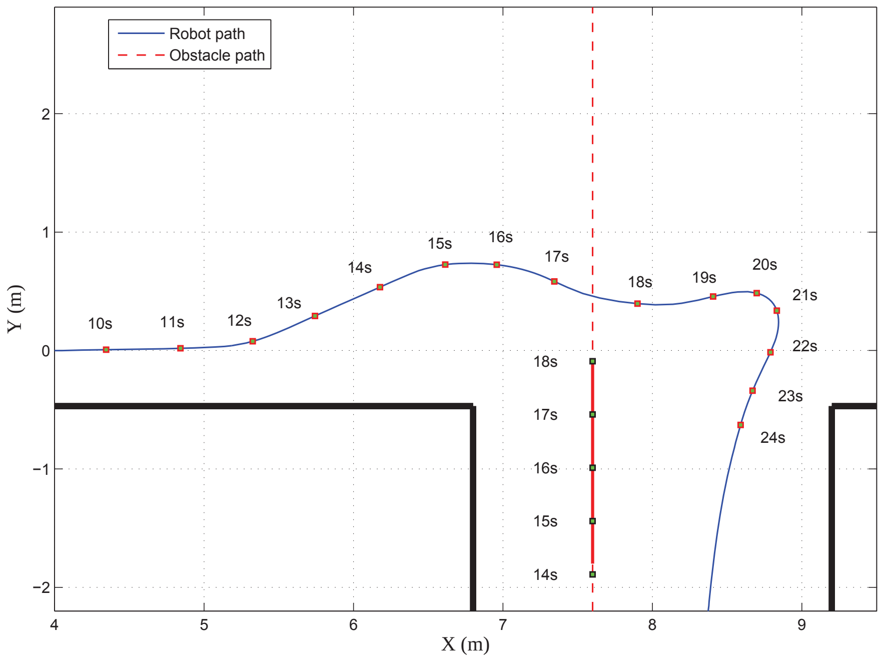


Figure 7.26: Avoiding an unobserved obstacle which appeared at the time 14.2s

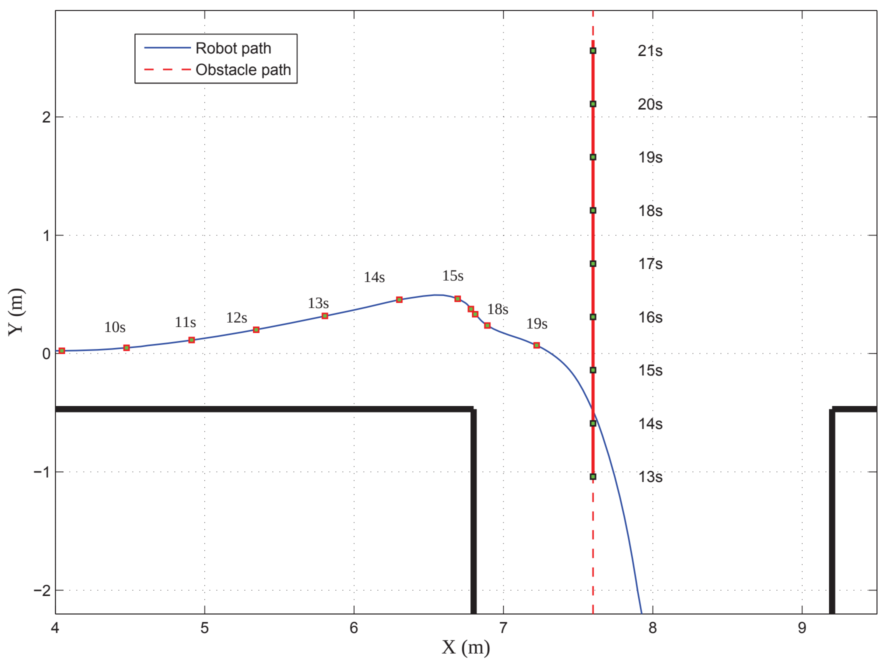


Figure 7.27: Avoiding an unobserved obstacle which appeared at the time 13s

7.6 Avoiding Humans

7.6.1 Definition of Scenarios

Six scenarios were designed to evaluate the robot performance while navigating among humans within indoor environments.

Scenario 1 (HS1) : the robot uses only the lower laser sensor to navigate from its starting location to its goal, and avoid a human (6,0) moving in the opposite direction of the robot (see Figure 7.28).

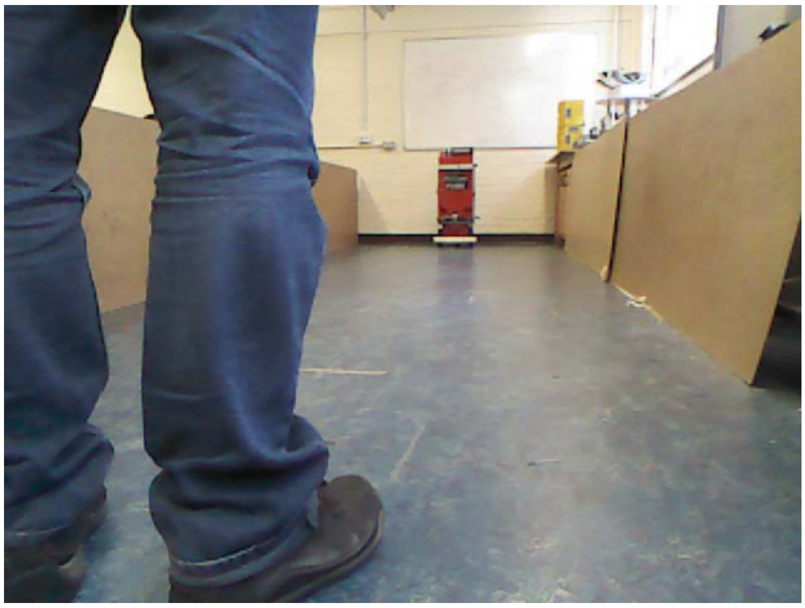
Scenario 2 (HS2) : the robot uses the both laser sensors to avoid the human. In S1 and S2, the human is required to maintain a constant speed and to walk in a straight line.

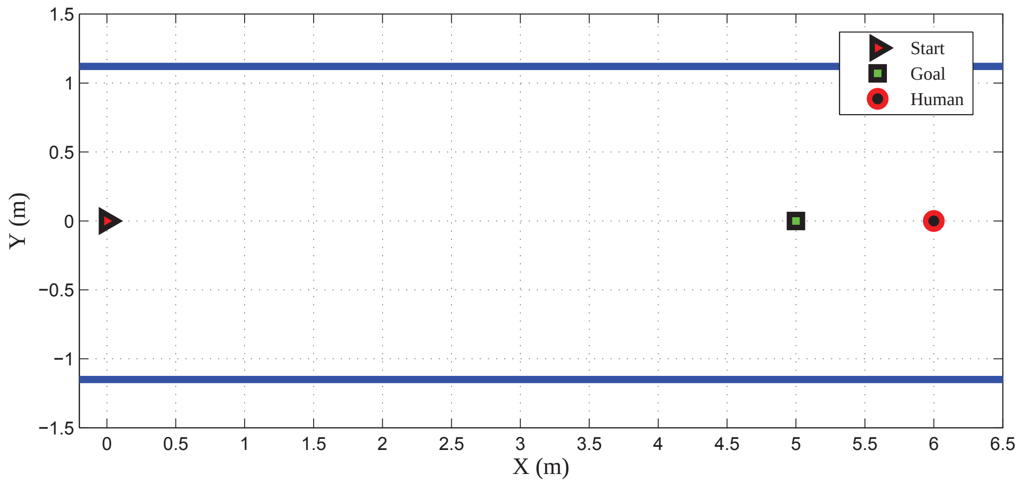
Scenario 3 (HS3) : the human is required to start avoiding the robot much sooner.

Scenario 4 (HS4) : the human is required to walk in a straight line and then to walk to the same side that is chosen by the robot to avoid collision.

Scenario 5 (HS5) : the robot avoids two humans; one of them walks in the opposite direction of the robot, while the another one walks in the same direction of the robot.

Scenario 6 (HS6) : the robot avoids an obstacle and human moving in the opposite direction of the mobile robot.


(a) Robot and a human



(b) Scheme picture

Figure 7.28: Single human avoidance

7.6.2 Implementation of Scenarios

Scenario 1 (HS1)

When only the lower laser sensor is used, the robot can reliably detect a human using the laser data coming from the two legs at a distance of 3.5m as shown Figure 7.29.a in which the right leg is leading the left leg, and then it begins taking avoiding action at a distance of 2.6m (Figure 7.29.b), having to turn through a relatively large angle of 72^o to avoid collision as shown in Figure 7.29.c in which the human is defined by one leg. The robot resumes its path to its goal after the human passed (Figure 7.29.d).

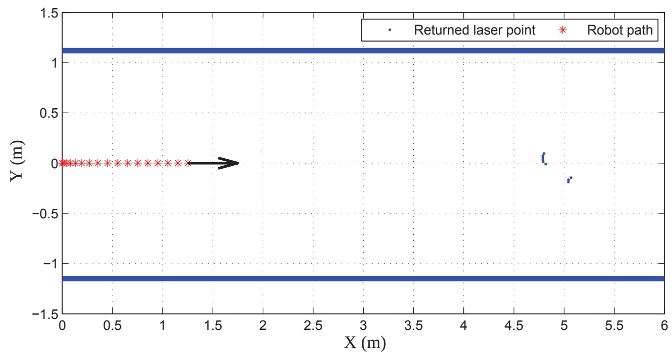
Scenario 2 (HS2)

Using both laser sensors increases the range of human detection to 4.85m (Figure 7.32.a), where the human is detected by integrating laser data returned from the two legs and human upper torso. The robot starts early the avoidance action at a distance of 3.55m (Figure 7.32.b) turning through 33^o (Figure 7.32.c and 7.31), this increases the time and distance to collision (Table 7.5), and produces a smoother path than carried out in HS1 (Figure 7.32.d).

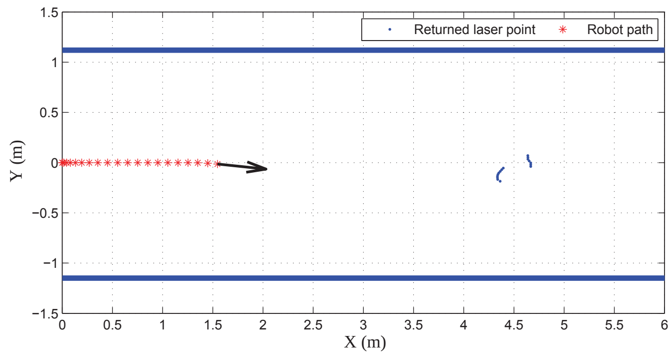
It can be observed from Figure 7.33 and 7.34 that the human walked during the previous two experiments with approximately 0.78m/s and 176^0. However, the human appears in the sensor data for 3s when only the lower sensor is used, and 4s when both sensors are used. Hence, using two laser sensors extends the human detection range and time by 2.15m and 1s respectively.

Table 7.5: Performance metrics of avoiding a human moving in a straight line

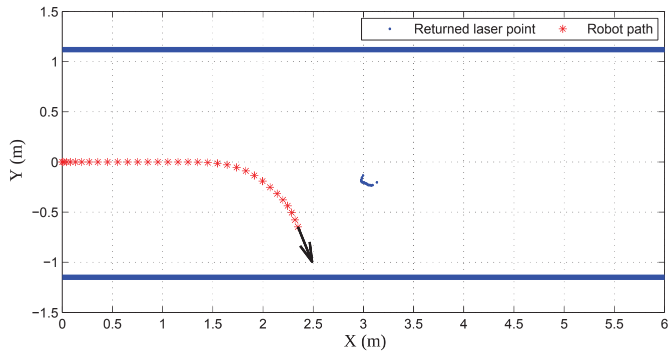
	$T_t(s)$	P_L(m)	Smoothness (rad^2/m^2)	t_{min}(s)	d_{min}(s)	Collision
HS1	14.0	5.12	1.044	0.36	0.18	0
HS2	13.7	5.08	0.728	0.71	0.33	0



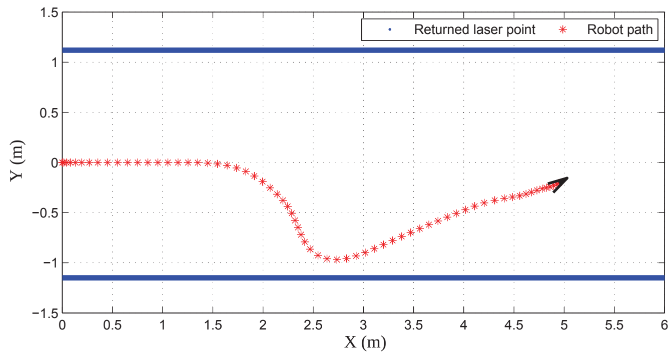
(a)



(b)



(c)



(d)

Figure 7.29: Human avoidance using a single laser sensor

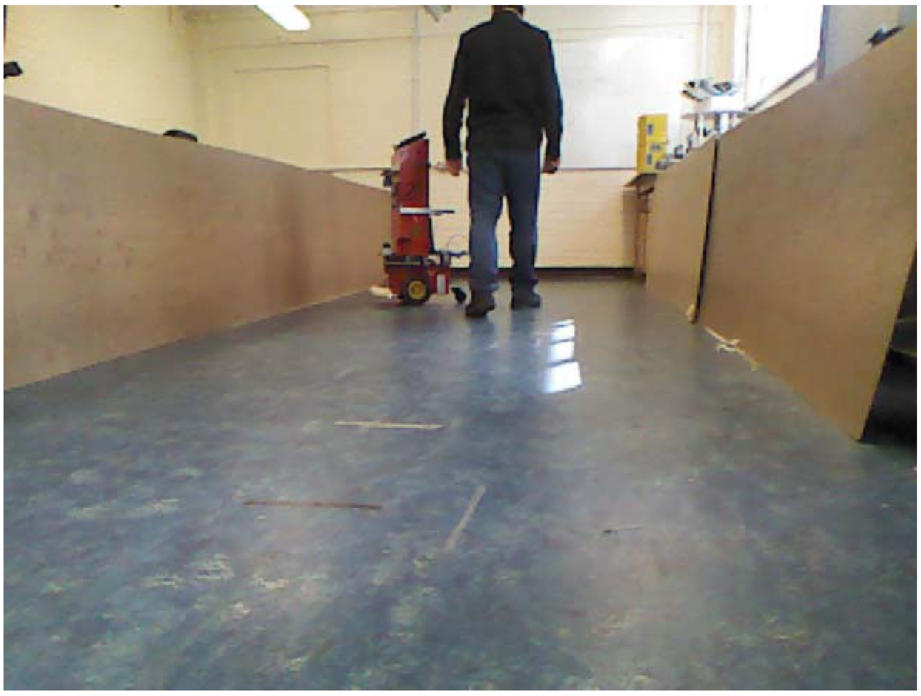


Figure 7.30: Robot is turning a relatively large angle

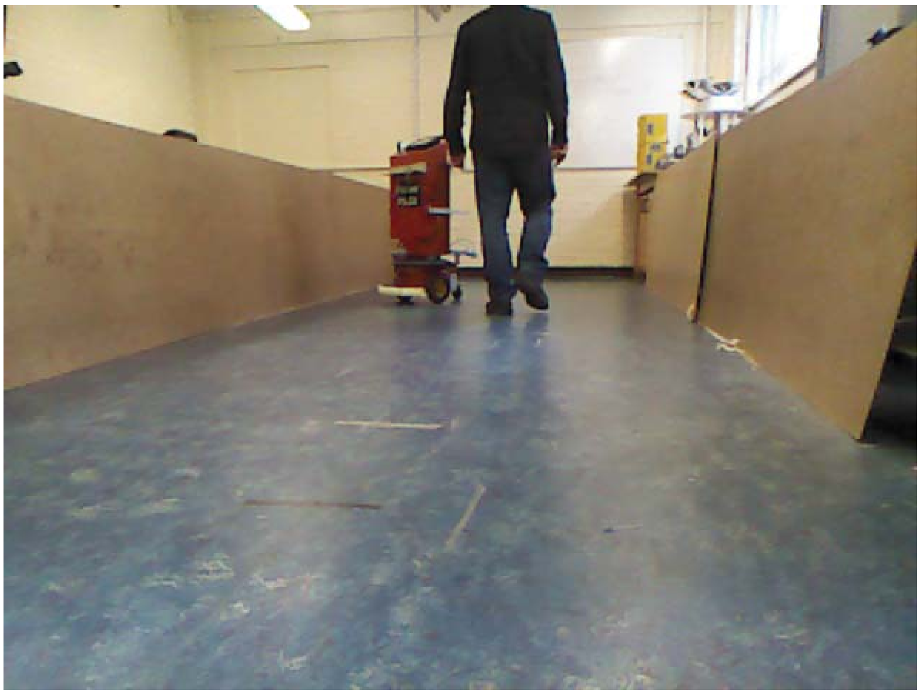
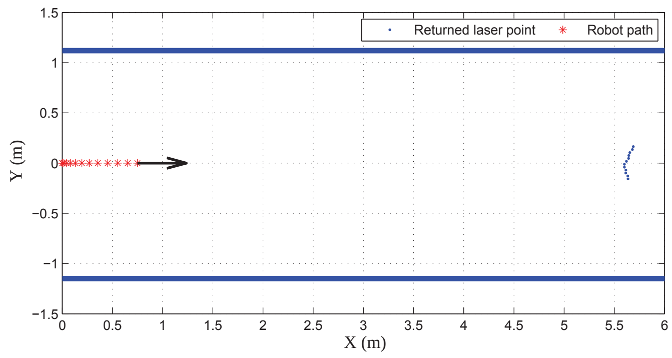
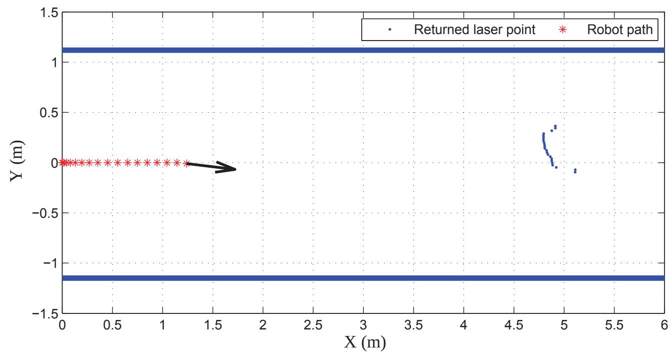


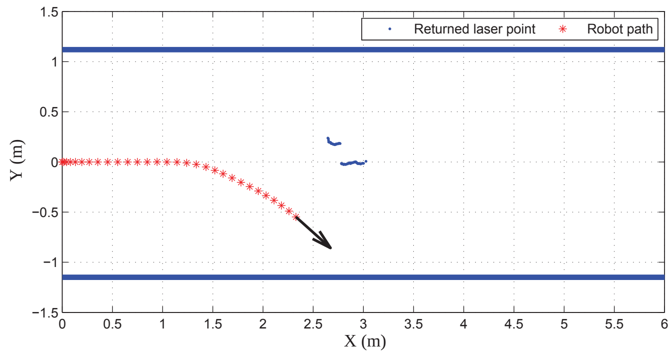
Figure 7.31: Two laser sensors to avoid a human



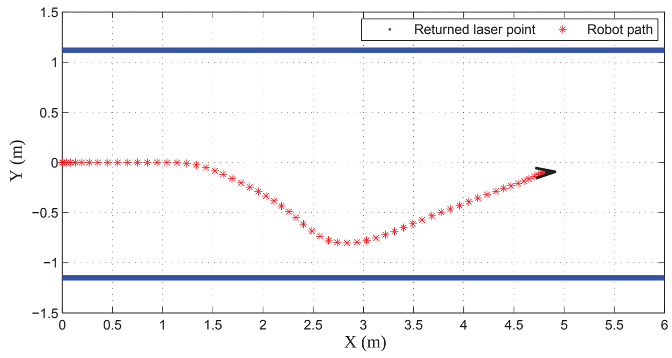
(a)



(b)



(c)



(d)

Figure 7.32: Human avoidance using two laser sensors

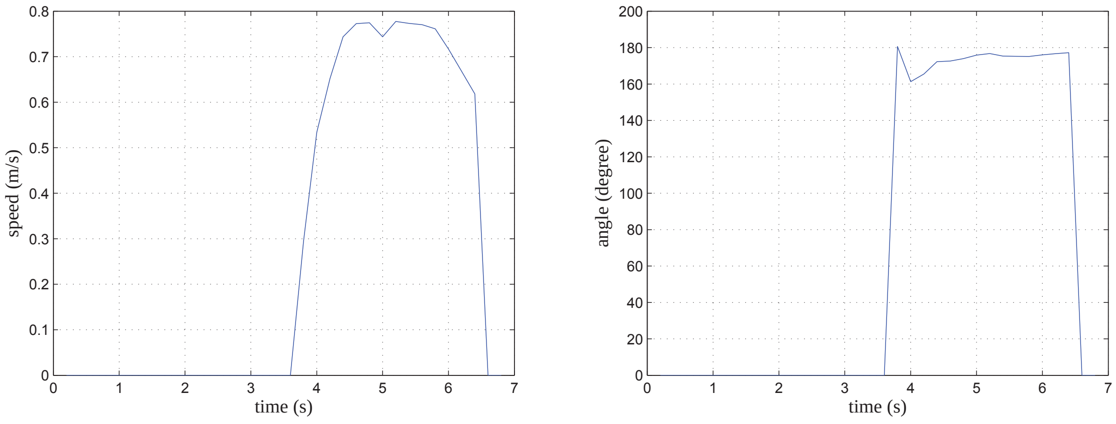


Figure 7.33: Estimated speed and direction of the human using single laser sensor

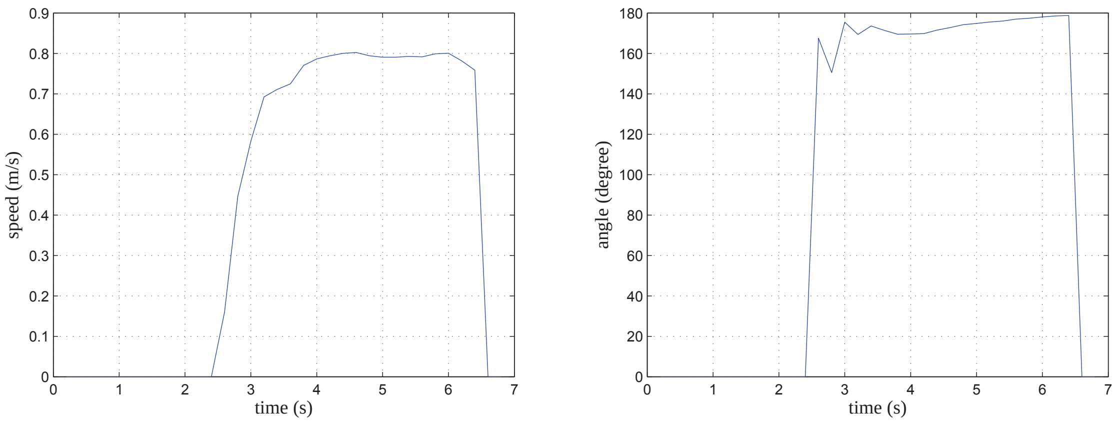


Figure 7.34: Estimated speed and direction of the human using two laser sensors

Scenario 3 (HS3)

In this experiment, the human is required to begin avoiding the robot as soon as possible. The robot initially moves in a straight line and then recognizes the avoidance action of the human who appeared in the returned laser data at $t = 3$s (Figure 7.35.a), after which the robot starts to move away through a relatively small angle of 26^o.

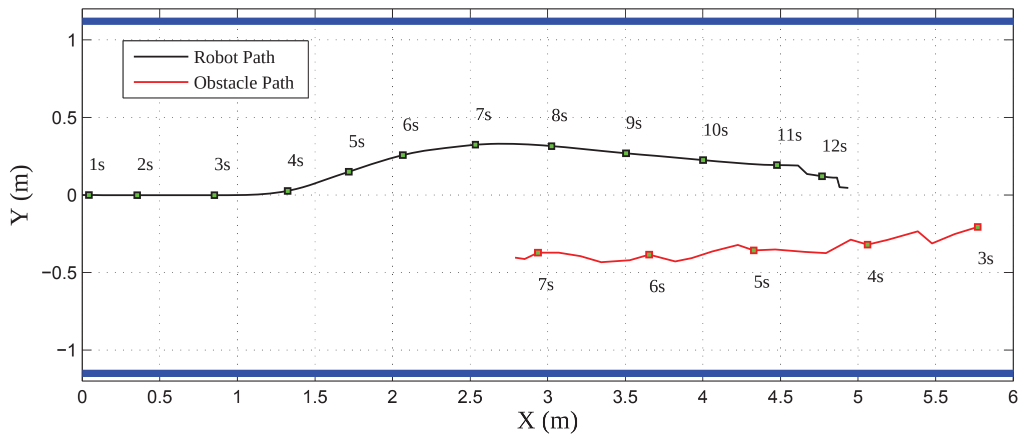
Scenario 4 (HS4)

Figure 7.35.b shows that the robot and human initially move in straight lines opposite each other, but after the robot starts the avoidance action, the human changes his orientation and walks toward the robot which quickly changes its direction to avoid collision producing less smoother path than that executed in HS3 with less time and distance to collision (see Table

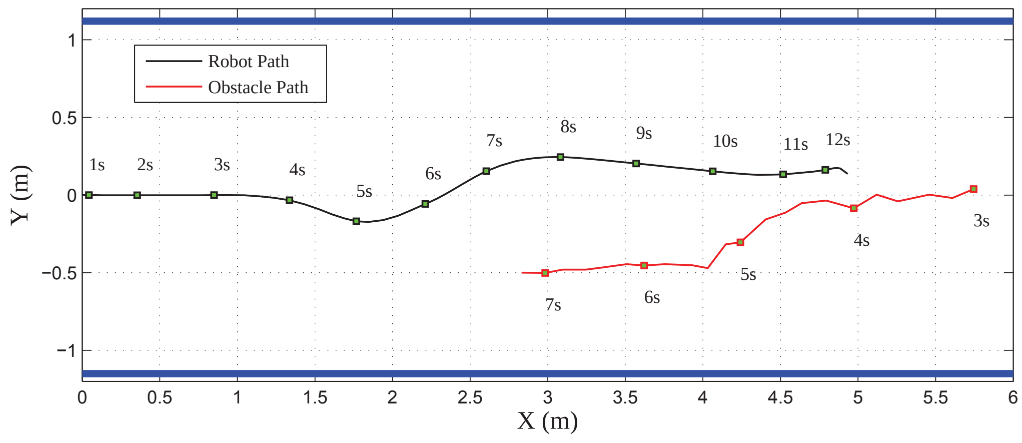
7.6).

Table 7.6: Performance metrics of avoiding a human

	$T_t(s)$	P_L(m)	Smoothness(rad^2/m^2)	t_{min}(s)	d_{min}(s)	N_C
HS3	13.7	5.08	0.242	1.93	1.13	0
HS4	13.9	5.04	1.097	1.5	1.39	0



(a)



(b)

Figure 7.35: Human tries to avoid the robot

Scenario 5 (HS5)

In this experiment, the robot navigates in the indoor environment with two humans A and B at (1.5,-0.5) and (8,0.5) respectively as shown in Figure 7.6, in which human A walks slowly at an approximate speed of 0.4m/s in the same direction as the robot, and human B walks faster than A at an approximate speed of 0.82m/s in the opposite direction. Figure 7.37 illustrates the robot executing the avoidance action for human A in $t \in [3s - 5s]$, after which it changes its direction and follows human A to avoid collision with human B who appears in the sensor data at the time $t = 2.9s$. At $t = 8.3$s the robot resumes its path to the goal moving away from human B with minimum distance and time to collision of 0.30m and 2.23s respectively (Table 7.7).

Table 7.7: Performance metrics for avoiding two humans

	$T_t(s)$	P_L(m)	Smoothness (rad^2/m^2)	t_{min}(s)	d_{min}(m)	Collision
HS5	20.0	8.20	0.95	2.23	0.30	0

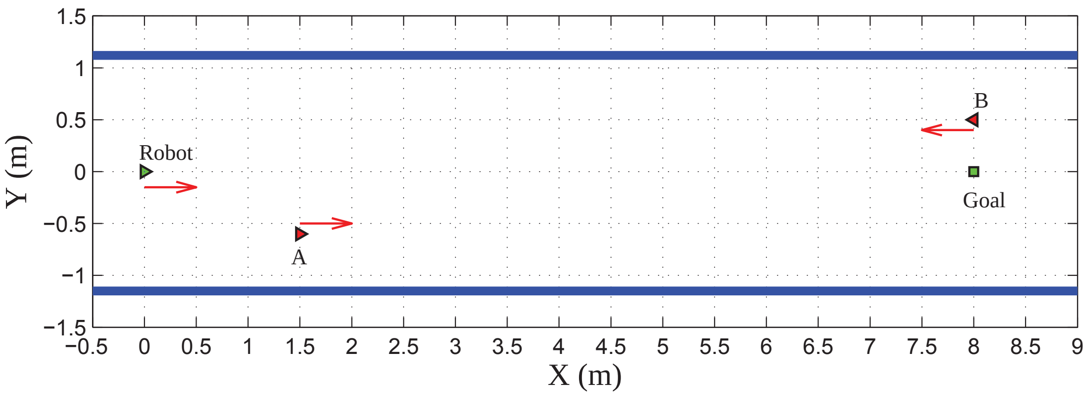


Figure 7.36: Avoid two humans

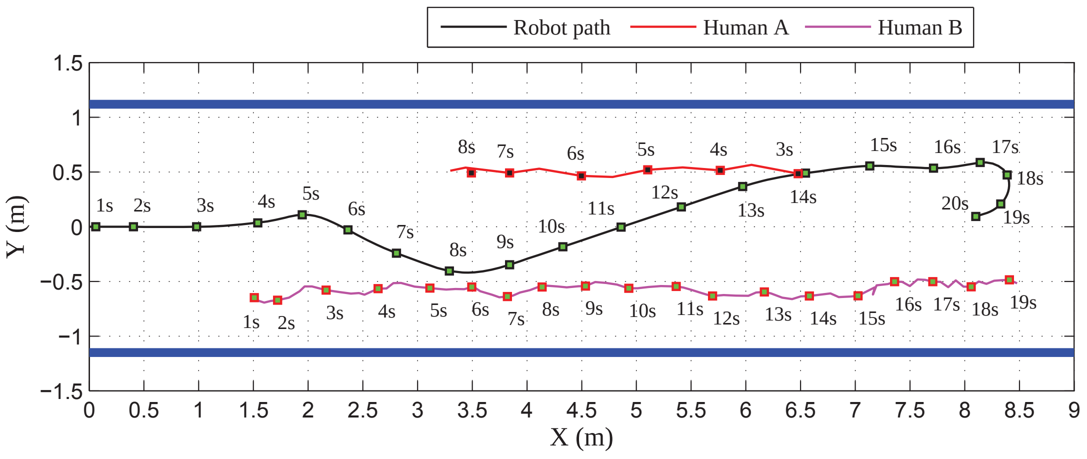
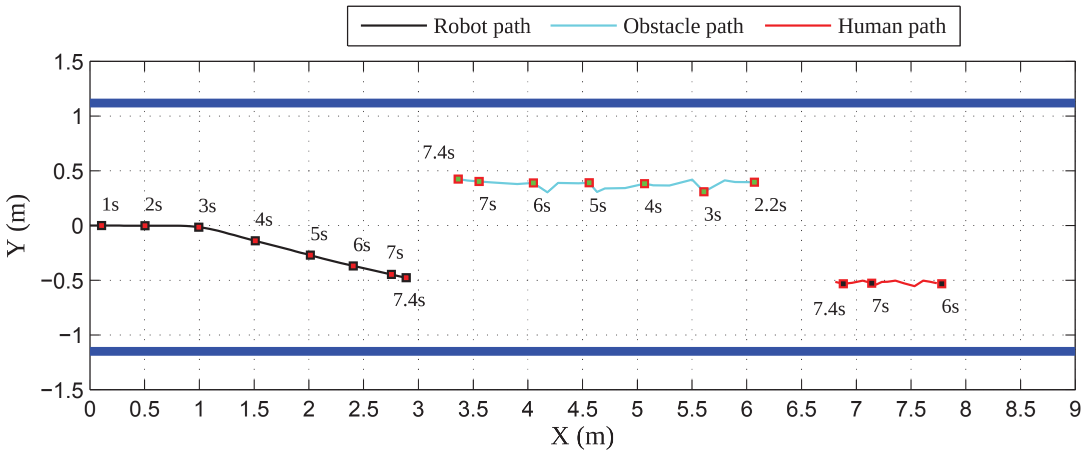


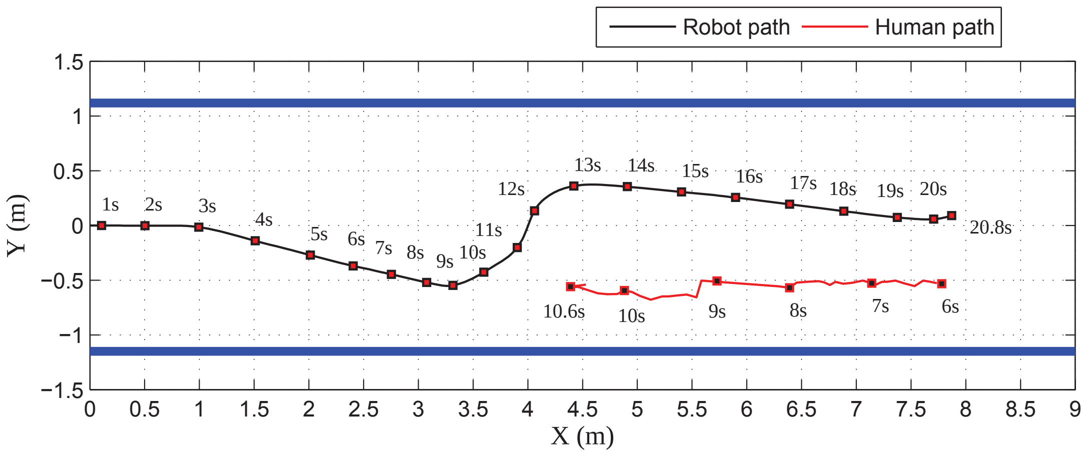
Figure 7.37: Paths of the robot and the two humans

Scenario 6 (HS6)

The robot task is to avoid an obstacle and human moving in the opposite direction of the robot. The robot initially moves in a straight line, and then at $t = 3$s begins to move away to avoid the moving obstacle which appears in the laser data in $t \in [2.2\text{s} - 7.4\text{s}]$, after which the robot starts to implement avoidance manoeuvre to avoid the high collision risk with the human (Table 7.8) who is detected at $t = 6\text{s}$ (Figure 7.38.b). Similarly, the robot is able to avoid a dynamic obstacle moving in the same direction and a human moving in the opposite direction (see appendix D).



(a)



(b)

Figure 7.38: Avoiding a human and dynamic obstacle

Table 7.8: Avoiding a human and dynamic obstacle

$T_t(s)$	P_L(m)	Smoothness (rad^2/m^2)	t_{min}(s)	d_{min}(s)	N_C
20.6	8.11	1.32	0.31	0.27	0

7.7 Navigation in a Crowded Environment

In this experiment, the proposed hybrid control architecture is tested by implementing a relatively long path (140m, see Table 7.10) in a crowded environment as shown Figure 7.39. The workspace is divided into four regions; namely A,B,C, and D (see Figure 7.40). Region A is crowded with static obstacles, while through regions B and C, many students are moving to their lecture rooms using stairs and lift. Region D is free from dynamic and static obstacles except walls.

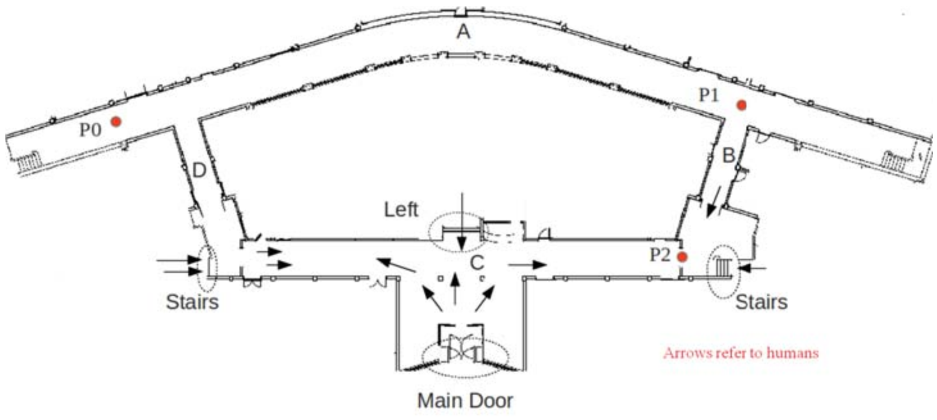


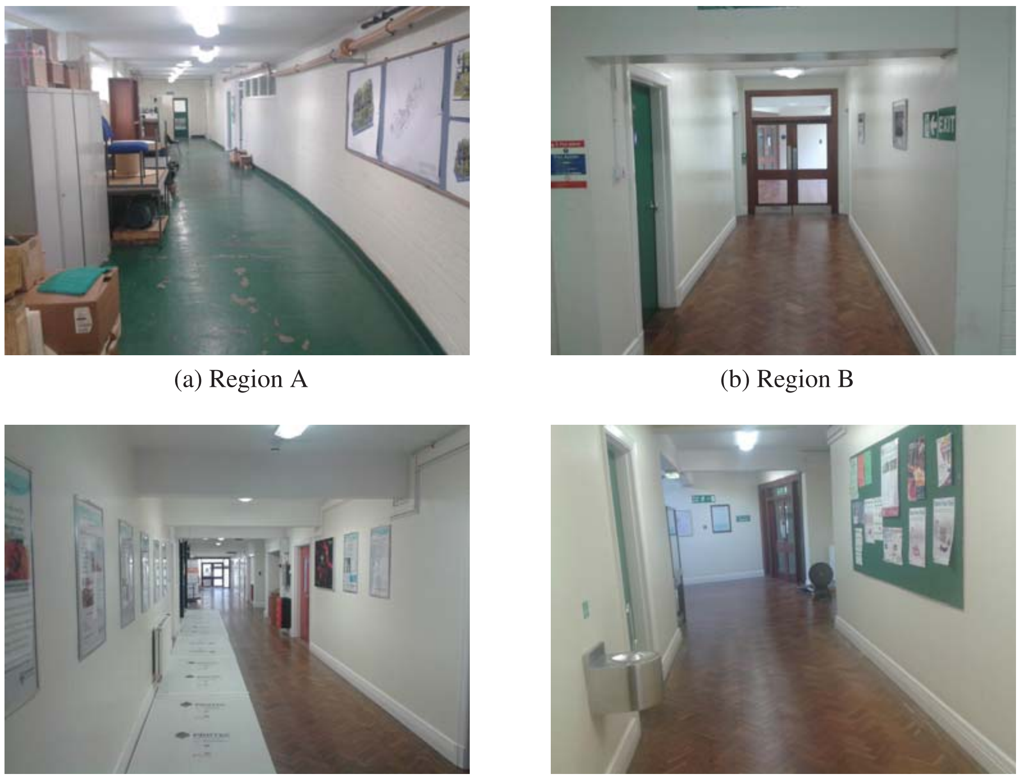
Figure 7.39: Ground floor, Stephenson Building, Newcastle University

The robot's task is to travel from P0 (robotics lab) to P1, and P2 and then return to the start point P0. The long path planning algorithm produced seven waypoints listed in Table 7.9.

Table 7.9: Waypoints produced by the wavefront algorithm

Start	Goal	Number of waypoints	Waypoints
(-35.3, 6.5)	(-35.3, 6.5)	7	(-35.3, +06.5) (+0.00, +15.5) (+26.0,+08.7) (+20.0, -06.0) (-22.7, -06.0) (-28.0, +07.0) (-35.3, +06.5)

The robot carries out successful navigation without collision or losing the target as shown in Figure 7.41, from which it can be observed that the robot motion in region D is much less erratic than that in region A, B and C which are cluttered with static and dynamic obstacles.


(a) Region A (b) Region B

(c) Region C (d) Region D

Figure 7.40: Regions of the robot workspace

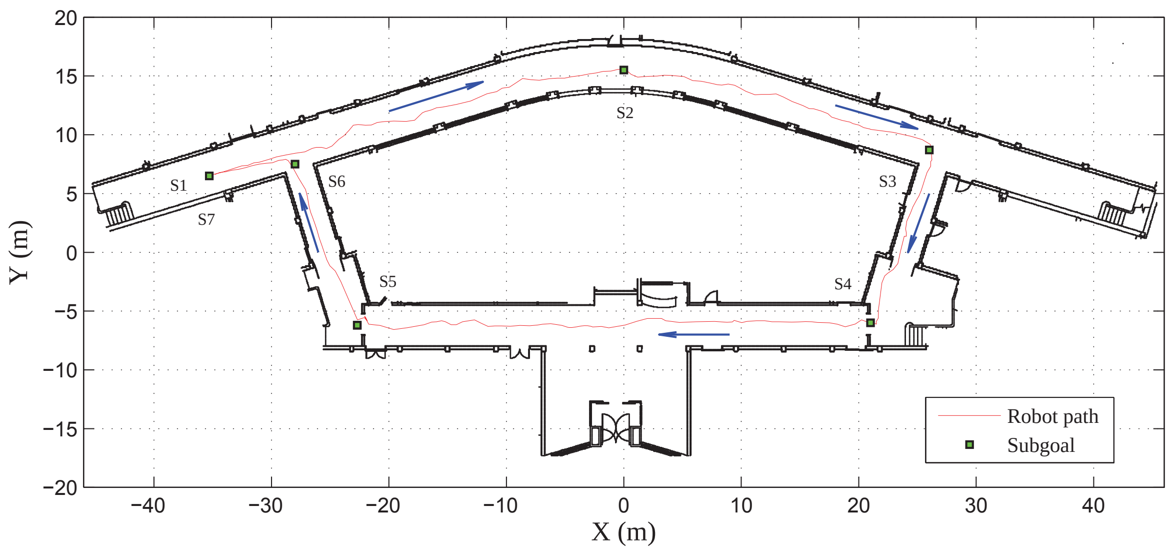
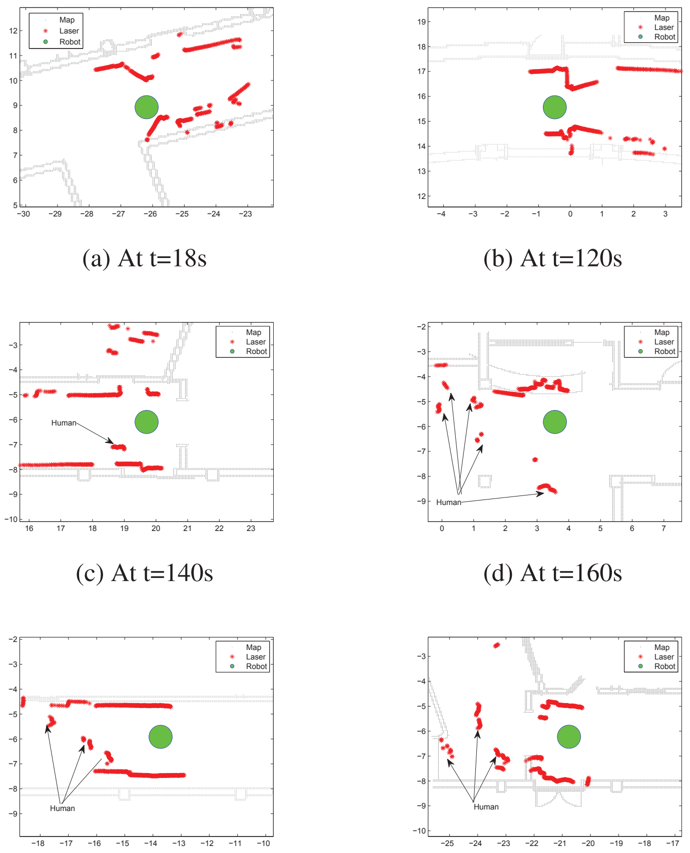


Figure 7.41: Navigation in a crowded environment

Figure 7.42 shows some laser images captured during the robot navigation; the robot passes through doors as shown in Figure 7.42.a and 7.42.b which were captured at 18s and 120s respectively. One human appears in laser data at t=140s (Figure 7.42.c). In Figure 7.42.d, four humans are waiting for the lift and another human is coming from the main entrance at t=160s. Figure 7.42.e shows three humans are walking in the same direction of the robot at t=180s, while in Figure 7.42.f three humans are trying to pass a door in the opposite direction of the robot at t=195s.



(a) At t=18s (b) At t=120s

(c) At t=140s (d) At t=160s

(e) At t=180s (f) At t=195s

Figure 7.42: Regions of the robot workspace

By zooming in on the area of the subgoal S5 (Figure 7.43), it can be observed that the robot stopped and moved back because several students were passing the narrow gap, and as a consequence there was not enough space for the robot to pass. However, when the students left enough space, the robot continued to its target. Table 7.10 lists the performance metrics of the robot which took 298.5s and 148.2m to navigate from its starting point to its goal, while the collision risk was relatively very high; the minimum time and distance to collision are 0.36s and 0.2m respectively.

Table 7.10: Performance metrics for navigation in a crowded environment

$T_t(s)$	P_L(m)	Smoothness (rad^2/m^2)	t_{min}(s)	d_{min}(m)	N_C
298.5	148.2	1.52	0.36	0.2	0

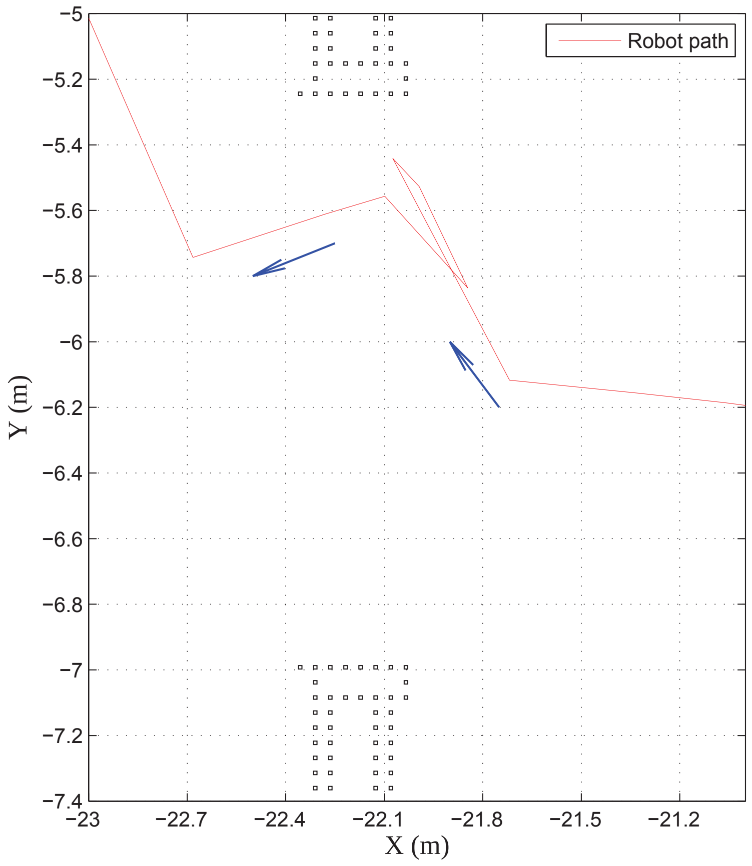


Figure 7.43: Emergency stop behaviour

7.8 Summary

Physical robot programmed by the proposed hybrid control system is able to produce safe navigation through planning an optimal and safe long-term path, producing robust position estimation, maximizing the speed to the goal, and minimizing collision risk with surrounding obstacles which have different shapes and speeds.

The experiments showed that the reactive behaviours enable the robot using the low laser sensor to successfully pass through narrow gap without an oscillatory behaviour, avoid and estimate actions of dynamic obstacles which are programmed with different algorithm, and produce optimal maneuvers to minimize the collision risk with obstacles which may appear in the robot path from the open doorways.

The robot can use the laser data returned from human legs to avoid humans, however, using another laser sensor to detect the human upper torso increases the distance and time of detecting humans. This results in reducing the collision risk and increasing the smoothness of the robot path. The experiment implemented to achieve long-path navigation in a crowded environment showed that the long path planning can plan an optimal path between the robot and its goal and and localization system is able produce a robust estimation of the robot position.

Chapter 8

Conclusions and Future Work

This chapter provides a summary of the project achievements, and gives recommendations for future work to improve the robot performance in dynamic environments.

8.1 Conclusions

The aim of this project was to develop a behavioural strategy for mobile robot navigation in cluttered indoor environments, occupied not only with static obstacles, but also dynamic ones which have different shapes, velocities and intelligence levels. To meet this aim, a hybrid control architecture was proposed to combine advantages of powerful planning using a deliberative architecture with quick response of a reactive architecture. At the deliberative level, this includes three requirements namely global path planning, localization and feature extraction. The global path planning is responsible for establishing a safe and optimal path between the current position of the robot and its target using the wavefront algorithm and workspace map. Theoretically, the robot can successfully navigate to its goal by following the global path without using an obstacle avoidance algorithm, however, any change in the workspace or any error in the estimated position of the robot may cause a collision. To solve this problem, sensor data can be used to update the robot map, and then the global path panning is applied, although the robot path may be more optimal in avoiding static obstacles, but the responsiveness of the robot will be slow. Therefore, the robot must be programmed to avoid obstacles using its sensor data to minimize the risk of collision.

In this project, mobile robots have been developed to work in indoor environments which are considered to be known environments, therefore, the Adaptive Montecarlo Localization algorithm (AMCL) was used to produce a robust estimation of the current position of the mobile robot based on sensor data and workspace map. With AMCL, the estimated position of the mobile robot converges to its actual position even if the initial position is unknown. A wall extraction model determines locations from which an obstacle may suddenly appear in the robot path. Defining surrounding walls offline using Hough Transform (HT) algorithm provides additional information about the surrounding environment, but requires a known workspace map and accurate localization system. However, more robust wall extraction can be carried out online using sensor data and the iterative-end-point-fit algorithm. The subgoals, estimated position and locations of surrounding walls are transferred through an intermediate level to the reactive level which includes six behaviours namely: *go to goal*, *obstacle avoidance*, *unobserved obstacle avoidance, human avoidance*, *follow the leader* and *emergency stop*. The *go to goal* behaviour produces an optimal speed and direction to the target using the the estimated robot position and the position of the target (subgoal), whilst The *obstacle avoidance* behaviour utilizes the velocity obstacle (VO) approach to avoid both dynamic and static obstacles. In this project, a method was developed to estimate the collision cone for circular and non-circular obstacles using laser data, where each laser point returned from the obstacles is enlarged using the robot radius and then the outer tangents of the enlarged obstacle forms the collision cone which is more accurate than that produced using the circle fitting algorithms. The *unobserved obstacle avoidance* (UOA) behaviour uses the *virtual obstacle* principle, which has been proposed to minimize the collision risk with unobserved moving objects appearing from around corners and doorways, where a virtual circular obstacle is created at the starting point of each open doorway threshold and corridor crossing. Simulation showed that if the UOA behaviour is inactive the robot may collide with a moving obstacle suddenly appearing in the robot path, whereas here the robot reduces the collision risk by moving away or reducing its speed while passing an open door threshold. The robot is also able to use the *follow the leader* behaviour to move behind someone going in the same direction to ease its passage, this behaviour provides much smoother motion when passing among multiple moving obstacles (including humans). The *emergency stop* behaviour monitors sensor data and the outputs of other behaviours to ensure that if there is no collision-free velocity the robot stops, and reverses to prevent collision. The *human avoidance* behaviour is used to recognize and avoid humans using data returned from their legs. In some instances, some parts of one leg are occluded by another leg, or the two legs are interpreted as one cluster if the distance between the two legs is too short. Hence in this project, a *human recognition* algorithm has been developed in which the laser data are put into four classes depending on their width, girth and depth.

The outputs of the reactive level behaviours are fused using different stages; in the first stage, the outputs of the *go to goal* and *follow the leader* behaviours are coordinated using the subsumption approach; when the *follow the leader* behaviour is activated it stops the output of the *go to goal* behaviour and produces the optimal direction and speed to follow a moving target. Next the outputs of the *obstacle avoid*ance, *unobserved obstacle avoid*, and *human avoidance* behaviours are integrated using an arbitration approach based on the motor schema architecture, where the minimum output is selected. The outputs of the first and second stages are coordinated using a cost function fusion method. Again, the third stage and *emergency stop* behaviours are coordinated using the arbitration method, where the *emergency stop* behaviour is selected if it is active.

For saving time and protecting the robot and others from any injure, the behavioural strategy proposed in this project was extensively tested in cluttered indoor environments using the Player/stage platform before evaluating it using two physical mobile robots; a modified Pioneer robot and Nubot robot which was developed in Newcastle University. Performance metrics of six factors were utilized to quantify the mobile robot performance; the minimum time and distance to collision were used to measure the collision risk, while the bending energy was used to measure the smoothness of the robot path. The experiments showed that using a single laser sensor mounted at a height of 0.40m enable the robot to avoid dynamic obstacles which have different levels of intelligence, and also is able to minimize the collision risk and avoid dynamic obstacles which suddenly appear from corridor crossings. For human detection, another laser sensor was mounted on the robot at a height of 0.70m, where the lower laser sensor is used to detect human legs, while the upper laser detects the human upper torso. The experiments showed that the robot performance using two laser sensors is better than when using only the lower one.

8.2 Summary of Achievements

The achievements described in this thesis are summarized as follows:

- Hybrid control architecture was developed for mobile robot navigation in dynamic environments, which includes three layer namely deliberative, intermediate and reactive layer, and was extensively tested in a cluttered environment using the Player/Stage simulation platform and existing robots. Experiments and simulation results showed that the mobile robot is able to navigate successfully among both static and dynamic obstacles (including humans).
- Algorithms were developed to cluster the sensor data, recognize and discriminate between both static and dynamic obstacles using the grid map, detecting human using four different types of the sensor clusters, and then apply the extended particle filter algorithm to estimate the speed and direction of the dynamic obstacles.
- A method has been demonstrated to estimate the collision cone for circular and non-circular obstacles using laser data, where this method produces a collision cone is more accurate than that produced using the circle fitting algorithms.
- Several novel behaviours have been developed which provide the robot with improved avoidance capability to include multiple dynamic obstacles as well as anticipating potential collisions resulting from restricted unobserved obstacles, and smoothing and easing the robot path in very cluttered workspace.

8.3 Recommendations for Future Work

The hybrid control architecture developed in this study provides a robot with the capability to autonomously navigate in a known environment from its current position to a goal defined by a user. This study can be extended by improving the hardware and software of the robot. For the hardware, the wide scan range (240^o) and relatively high angular resolution 0.36^o of 2D laser sensors provide accurate information which is used to produce robust estimation of the robot position and dynamic obstacle velocities. However, in a 3D real world, a fatal collision may occur because the robot can not detect the obstacles (for example the upper part of tables) which locate out side of the laser plane. Furthermore, the update rate (10Hz) and reading range (5.6m) restrict the anticipating of relatively fast dynamic motion of obstacles, this reduces the maximum speed of the robot to execute safe navigation. Kinect sensor can be used to provided a 3D view with an update rate of 30fps, however, its narrow field of view and short range (4.0m) will limit the robot performance. Hence, a stereo vision system can be integrated with laser sensors to improve the robot sensing. The stereo vision gives 3D information about the robot workspace, and can be build in the lab using two identical cameras and a suitable software such as OpenCv. However for saving time and effort, it is recommended to use a commercial stereo vision system such as *Bumblebee* which provides a field of view of 100^0 with an update rate of 40fps.

For the software, the current localization system (AMCL) and global path planning require an accurate map of the workspace. Hence, when the robot moves to a new workspace, a user must provide an accurate map of the new workspace, and provide the Cartesian coordinates of the goal. The navigation system can be more flexible by utilizing the simultaneously mapping and localization (SLAM) and multiple modes systems, where the SLAM system builds the robot map and updates it automatically, and the multiple modes system switches between three different modes namely manual, semi-autonomous, and autonomous, where the robot can be guided using commands generated using a joystick, smart phone, location on a map, and voice commands. In the manual mode, the robot follows exactly the commands generated by the user using joystick or smart phone, however, the emergency stop behaviour can be used to avoid collision. The semi-autonomous mode combines commands coming from the user and the outputs of the reactive behaviours. In the autonomous mode, the robot uses the proposed navigation algorithm in this project.

The cost function parameters used in the reactive level behaviours were chosen using the trial and error method. Hence, using artificial intelligence (AI) such as Fuzzy Logic and Neural Network can enhance the robot performance and optimize the local path planning by updating the cost functions depending on collected sensor data. Also, to interact with people in a social and technological manners, more behaviours can be added such as *using the lift* behaviour which makes the robot leaving enough spaces for other people to enter a lift, and *passing human* behaviour says a suitable statements (“excuses me”) when the robot passes a person moving in the same direction.

Reference

Aethon, 2007. Tug robot. `http://www.aethon.com/tug/how-it-works`, [Online; accessed 10 Apr 2015].

Al-Jumaily, A., Leung, C., 2005. Wavefront propagation and fuzzy based autonomous navigation. In: International journal of Advanced Robotic Systems. Vol. 2. pp. 1729–8806.

Arkin, R., 1987. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In: IEEE International Conference on Robotics and Automation. Proceedings.

Belotto, N., Hu, H., 2009. Multisensor-based human detection and trackingfor mobile service robots. In: IEEE Trans. Syst., Man, Cybern. B, Cybern. Vol. 39. pp. 167–181.

Beom, H. R., Cho, K. S., 1995. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. In: IEEE Transactions on Systems, Man and Cybernetics. Vol. 25. pp. 464–477.

Bischoff, B., Nguyen-Tuong, D., Streichert, F., Ewert, M., Knoll, A., Dec 2012. Fusing vision and odometry for accurate indoor robot localization. In: 12th International Conference on Control Automation Robotics Vision (ICARCV). pp. 347–352.

Borenstein, J., Koren, Y., 1991. The vector field histogram-fast obstacle avoidance for mobile robots. In: IEEE Transactions on Robotics and Automation. Vol. 7. pp. 278–288.

Brooks, R., 1986. A robust layered control system for a mobile robot. In: IEEE Journal of Robotics and Automation. Vol. 2. pp. 14–23.

Byoung-Suk, C., Ju-Jang, L., 2009. Mobile robot localization in indoor environment using rfid and sonar fusion system. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2039–2044.

Car, 2105. Like car robot. `http://odalab.spub.chitose.ac.jp/english/research.htm`, [Online; accessed 10 Apr 2015].

Carballo, A., Ohya, A., Yuta, S., 2009. Multiple people detection from a mobile robot using double layered laser range finders. In: Proceedings of the IEEE ICRA Workshop on People Detection and Tracking. pp. 1–7.

Ceballos, N. D. M., Valencia, J. A., Ospina, N. L., 2010. Quantitative performance metrics for mobile robots navigation, mobile robots navigation. In: http://www.intechopen.com/books/mobile-robots-navigation/quantitative-performance-metrics-for-mobile-robots-navigation.

Chakravorty, S., Saha, R., 2009. Simultaneous localization and mapping in dense environments. In: IEEE International Conference on Systems, Man and Cybernetics. pp. 2769–2774.

Chang, F.-M., Lian, F.-L., 2012. Polar grid based robust pedestrian tracking with indoor mobile robot using multiple hypothesis tracking algorithm. In: Proceedings of SICE Annual Conference. pp. 1558–1563.

Chatila, M., R., K., 1995. An extended potential field approach for mobile robot sensor-based motions. In: Proc of the Intelligent Autonomous Systems.

Chatila, R., Laumond, J., 1985. Position referencing and consistent world modeling for mobile robots. In: IEEE International Conference on Robotics and Automation. Proceedings. Vol. 2. pp. 138–145.

Chi, W., Zhang, W., Gu, J., Ren, H., Dec 2013. A vision-based mobile robot localization method. In: IEEE International Conference on Robotics and Biomimetics (ROBIO). pp. 2703–2708.

Cox, I. J., 1993. A review of statistical data association techniques for motion correspondence. In: International Journal of Computer Vision. Vol. 10. pp. 53–66.

De, X., Min, T., Gang, C., 2002. An improved dead reckoning method for mobile robot with redundant odometry information. In: 7th International Conference on Control, Automation, Robotics and Vision. Vol. 2. pp. 631–636 vol.2.

Dietmayer, K., Sparbert, J., Streller, D., 2001. Model based object classification and object tracking in traffic scenes from range images. In: Proceedings of intelligent Vehicles Symposium.

Doucet, A., de Freitas, J., Gordon, N., 2000. Sequential montecarlo methods in practice.

Droge, G., Kingston, P., Egerstedt, M., Oct 2012. Behavior-based switch-time mpc for mobile robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 408–413.

El-laithy, R., Huang, J., Yeh, M., 2012. Study on the use of microsoft kinect for robotics applications. In: IEEE/ION Position Location and Navigation Symposium (PLANS). pp. 1280–1288.

Elfes, A., 1987. Sonar-based real-world mapping and navigation. In: IEEE Journal of Robotics and Automation. Vol. 3. pp. 249–265.

Fiorini, P., Shiller, Z., 1998. Motion planning in dynamic environments using velocity obstacles. In: International Journal of Robotics Research. Vol. 7. pp. 760–772.

Fox, D., Burgard, W., Thrun, S., 1997. The dynamic window approach to collision avoidance. In: IEEE Robotics and Automation Magazine. Vol. 4. pp. 23–33.

Fraichard, T., Laugier, C., 1993. Dynamic trajectory planning, path-velocity decomposition and adjacent paths. In: Artifcial Intelligence , Chambery, France. pp. 1592–1597.

Frederic Large, C. L., Shiller, Z., 2005. Navigation among moving obstacles using the nlvo: Principles and applications to intelligent vehicles. In: Autonomous Robot. Vol. 19. pp. 159–171, 1073562.

FrontPage, M., 2014. wheel skid-steer robot. [Online; accessed 6 July 2014].
URL `http://groups.csail.mit.edu/drl/courses/cs542001s/skidsteer.html`

Fulgenzi, C., Spalanzani, A., Laugier, C., 2007. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In: IEEE International Conference on Robotics and Automation. pp. 1610–1616.

Gander, W., Golub, G. H., Strebel, R., 1994. Least squares fitting of circles and ellipses. In: BIT. Vol. 34. pp. 558–875.

Gerkey, B., Vaughan, R., Stoy, K., Howard, A., Sukhatme, G., Mataric, M., 2001. Most valuable player: a robot device server for distributed control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 3. pp. 1226–1231 vol.3.

Gomez, E., Martiinez Santa, F., Sarmiento, M., Oct 2013. A comparative study of geometric path planning methods for a mobile robot potential field and voronoi diagrams. In: Engineering Mechatronics and Automation (CIIMA). pp. 1–6.

Guzel, M. S., Bicker, R., 2012. A behaviour-based architecture for mapless navigation using vision. In: International Journal of Advanced Robotic Systems. Vol. 9. pp. 9–18.

Haoxiang, L., Ying, W., de Silva, C. W., 2008. Mobile robot localization and object pose estimation using optical encoder, vision and laser sensors. In: IEEE International Conference on Automation and Logistics. pp. 617–622.

Hasan, K., Al-Nahid, A., Al Mamun, A., 2012. Implementation of autonomous line follower robot. In: International Conference on Informatics, Electronics Vision (ICIEV). pp. 865–869.

Heesung, C., Kyuseo, H., 2005. Combination of rfid and vision for mobile robot localization. In: Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information. pp. 75–80.

Hough, P., 1962. Method and means for recognizing complex patterns. In: U.S. Patent 3,069,654. pp. 258–263.

Hughes, K., Tokuta, A., Ranganathan, N., 1992. trulla : An algorithm for path planning among weighted regions by localized propagations. In: Proceedings of the 1992 lEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 1. pp. 469–476.

Huiying, D., Wenguang, L., Jiayu, Z., Shuo, D., 2010. The path planning for mobile robot based on voronoi diagram. In: 3rd International Conference on Intelligent Networks and Intelligent Systems (ICINIS). pp. 446–449.

HUSSON., R., 1997. An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation. In: In Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications.

Isard, M., Blake, A., 1998. Condensation - conditional density propagation for visual tracking. In: International Journal of Computer Vision. Vol. 29. pp. 5–28.

Jasika, N., Alispahic, N., Elma, A., Ilvana, K., Elma, L., Nosovic, N., May 2012. Dijkstra's shortest path algorithm serial and parallel execution performance analysis. In: MIPRO, 2012 Proceedings of the 35th International Convention. pp. 1811–1815.

Jinseok, W., Kubota, N., 2011. Simultaneous localization and mapping using a robot partner in dynamic environment. In: Annual Conference (SICE). pp. 524–529.

Jinshi, C., Hongbin, Z., Huijing, Z., Shibasaki, R., 2006. Robust tracking of multiple people in crowds using laser range scanners. In: 18th International Conference on Pattern Recognition. pp. 857–860.

Kant, K., Zucker, S., 1986. Toward efficient trajectory planning: The path-velocity decomposition. In: International Journal of Robotics Research. pp. 72–89.

Kerr, J., Nickels, K., 2012. Robot operating systems: Bridging the gap between human and robot. In: 44th Southeastern Symposium on System Theory (SSST). pp. 99–104.

Khatib, O., 1985. Real-time obstacle avoidance for manipulators and mobile robots. In: IEEE International Conference on Robotics and Automation. Vol. 2. pp. 500–505.

Kim, M., Chong, N. Y., Ahn, H.-S., Yu, W., 2007. Rfid - enabled target tracking and following with a mobile robot using direction finding antennas. In: Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering. pp. 1014–1019.

Kluge, B., 2003. Recursive agent modeling with probabilistic velocity obstacles for mobile robot navigation among humans. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 1. pp. 376–380.

Kluge, B., Prassler, E., 2004. Reflective navigation: individual behaviors and group behaviors. In: IEEE International Conference on Robotics and Automation. Vol. 4. pp. 4172–4177.

Kmiotek, P., 2009. Multi-sensor data fusion for representing. and tracking dynamic objects. Ph.D. thesis, Universite de Technologie de Belfort-Montbeliard.

Koller-Meier, E., Ade, F., 2001. Tracking multiple objects using the condensation algorithm. In: Robotics and Autonomous Systems. Vol. 34. pp. 93–105.

Kramer, J., Scheutz, M., 2007. Development environments for autonomous mobile robots: A survey. In: Autonomous Robots. Vol. 22. Kluwer Academic Publishers-Plenum Publishers, pp. 101–132.

Laas, 2003. The mobile robots at laas. In: Available at: http://homepages.laas.fr/matthieu/robots/hilare.shtml.

Layton, J., 2005. How robotic vacuums work. `http://electronics.howstuffworks.com/gadgets/h` [Online; accessed 6 Apr 2015].

LeBouthillier, A. E., 2014. W. grey walter and his turtle robots. `http://www.rssc.org/content/w-grey-walter-and-his-turtle-robots`, [Online; accessed 6 July 2014].

Lee, K., 2001. Reactive navigation for an outdoor autonomous. Ph.D. thesis, University of Sydney.

Li, X., Jiang, l., 2007. Study on control system architecture of modular robot. In: IEEE International Conference on Robotics and Biomimetics. pp. 508–512.

Lumelsky, V. J., Skewis, T., 1990. Incorporating range sensing in the robot navigation function. In: IEEE Transactions on Systems, Man and Cybernetics. Vol. 20. pp. 1058–1069.

Lutz, M., Steck, A., Schlegel, C., 2010. Person following through cluttered environments using probabilistic methods and fast local obstacle avoidance. In: 3rd Israeli Conference on Robotics (ICR).

Machida, E., Cao, M., Murao, T., Hashimoto, H., 2012. Human motion tracking of mobile robot with kinect 3d sensor. In: SICE Annual Conference (SICE). pp. 2207–2211.

Marron, M., Sotelo, M., GarcÃa, J., 2004. Tracking multiple and dynamic objects with an extended pconference filter and an adapted k-means clustering algorithm. In: Preprints of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (IAV).

Mataric, M. J., 2005. The robotics primer. In: MIT Press. pp. 95–96.

Milford, M., Wyeth, G., 2010. Hybrid robot control and slam for persistent navigation and mapping. In: Robotics and Autonomous Systems. Vol. 58. pp. 1096–1104.

Min, W. Z., Hua, M. D., Jiang, D. Z., 2009. Simultaneous localization and mapping for mobile robot based on an improved pconference filter algorithm. In: International Conference on Mechatronics and Automation. pp. 1106–1110.

Moravec, H., 1983. The stanford cart and the cmu rover. Vol. 71. pp. 872–884.

Moravec, H., Elfes, A., Mar 1985. High resolution maps from wide angle sonar. In: IEEE International Conference on Robotics and Automation. Vol. 2. pp. 116–121.

Moravec, H. P., 1982. The cmu rover. In: National Conference of Intelligence (AAAI-82).

MURPHY, R. O. B. I. N. R., 2000. Introduction to AI Robotics, 1st Edition.

Nakhaeinia, D., Tang, S., Noor, S. M., Motlagh, O., 2011. A review of control architectures for autonomous navigation of mobile robots. In: International Journal of the Physical Sciences. Vol. 5. pp. 169–174.

Napoleon, D., Lakshmi, P. G., 2010. An efficient k-means clustering algorithm for reducing time complexity using uniform distribution data points. In: Trendz in Information Sciences and Computing (TISC). pp. 42–45.

Nattharith, p., 2010. Mobile robot navigation using a behavioural control strategy. Ph.D. thesis, Newcastle University.

Nguyen, V., Martinelli, A., Tomatis, N., Siegwart., R., 2005. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1929–1934.

Nilsson, N. J., 1984. Shakey the robot. In: Artificial Intelligence Center, Computer Science and Technology Division, Standford Research Institute.

Nunes, C. P., U., 2005. Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications. In: Robotica 2005 Scientific Meeting of 5th National Robotics vastival. pp. 17–25.

Online, 2105. Elmer elsie robots. `http://www.theoldrobots.com/ElmerElsie.html`, [Online; accessed 10 Apr 2015].

Orton, M., Fitzgerald, W., 2002. A bayesian approach to tracking multiple targets using sensor arrays and pconference filters. In: IEEE Transactions on Signal Processing. Vol. 50. pp. 216–223.

Peng, L., Xinhang, H., Min, W., 2009. A hybrid method for dynamic local path planning. In: International Conference on Networks Security, Wireless Communications and Trusted Computing. Vol. 1. pp. 317–320.

Percussion, 2014. An unmanned aerial vehicle. [Online; accessed 6 July 2014].
URL `http://prescott.erau.edu.html`

Petrovic, E., Leu, A., Ristic-Durrant, D., Nikolic, V., 2013. Stereo-vision based human tracking for robotic follower. In: International Journal of Advanced Robotic Systems.

Pioneer, 2105. Pioneer p3-dx. `http://www.i-a-i.com/products/viewplain.asp?pid=28`, [Online; accessed 10 Apr 2015].

Pruski, Alain, Rohmer, Serge, 2008. Robust path planning for non-holonomic robots. In: Journal of Intelligent and Robotic Systems. Vol. 18. pp. 329–350.

Rebai, K., Benabderrahmane, A., Azouaoui, O., Ouadah, N., 2009. Moving obstacles detection and tracking with laser range finder. In: International Conference on Advanced Robotics. pp. 1–6.

Sakamoto, Y., Ebinuma, T., Fujii, K., Sugano, S., 2012. Gps-compatible indoor-positioning methods for indoor-outdoor seamless robot navigation. In: IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO). pp. 95–100.

Salem, F. A., 2013. Dynamic and kinematic models and control for differential drive mobile robots. In: International Journal of Current Engineering and Technology. Vol. 2. pp. 253–263.

Santos, S., Faria, J., Soares, F., Araujo, R., Nunes, U., 2003. Tracking of multi-obstacles with laser range data for autonomous vehicles. In: 3rd National Festival of Robotics Scientific Meeting (ROBOTICA). Lisbon, Portugal, pp. 59–65.

Schmitt, R., Nisch, S., Scho, x, nberg, A., Demeester, F., Renders, S., 2010. Performance evaluation of igps for industrial applications. In: International Conference on Indoor Positioning and Indoor Navigation (IPIN). pp. 1–8.

Sen, A., Srivastava, M., 1990. Regression analysis: Theory methods and applications. In: Springger-Verlay, New York.

SeungKeun, C., JangMyung, L., SukChan, S., 2008. A dynamic localization algorithm for mobile robots using the igs system. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics. pp. 734–739.

Shiller, Z., Gal, O., Fraichard, T., 2010. The nonlinear velocity obstacle revisited: the optimal time horizon. In: Guaranteeing Safe Navigation in Dynamic Environments Workshop.

Siegwart., R., 1997. Feature extraction and scene interpretation for map-based navigation and map building. In: Proc. of SPIE, Mobile Robotics XII. pp. 1929–1934.

Sivaraj, D., Kandaswamy, A., Vetrivel, S., Dec 2012. Development of navigation system for autonomous vehicle. In: International Conference on Machine Vision and Image Processing (MVIP). pp. 161–164.

stearman, 2014. Underwater robot. [Online; accessed 6 July 2014].
URL `http://www.nsf.gov/news/newsletter/jun-06/index.jsp`

Sung, Y., Chung, W., 2013. Implementation of jpdafs to track humans for a mobile robot with a laser range finder. In: The 22nd IEEE International Symposium on Robot and Human Interactive Communication. pp. 358–359.

Toal, D., Flanagan, C., Jones, C., Strunz, B., Toal, D., Flanagan, C., Jones, C., Strunz, B., 1996. Subsumption architecture for the control of robots. In: University of Limerick.

Tricycle, 2105. Tricycle robot. `http://deploy.virtual-labs.ac.in/labs/cse17/forwardkinemat` [Online; accessed 10 Apr 2015].

Ulrich, I., Borenstein, J., 1998. Vfh+: reliable obstacle avoidance for fast mobile robots. In: IEEE International Conference on Robotics and Automation. Vol. 2. pp. 1572–1577 vol.2.

Ulrich, I., Borenstein, J., 2000. Vfh*: local obstacle avoidance with look-ahead verification. In: IEEE International Conference on Robotics and Automation. Vol. 3. pp. 2505–2511 vol.3.

Vandorpe, J., Brussel, H. V., Xu, H., 1996a. Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2d range finder. In: IEEE International Conference on Robotics and Automation. pp. 901–908.

Vandorpe, J., Brussel, H. V., Xul, H., 1996b. Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2d range finder. In: IEEE International Conference on Robotics and Automation, 1996. Proceedings. Vol. 1. pp. 901–908.

Vatavu, A., Nedevschi, S., Sept 2013. Vision-based tracking of multiple objects in dynamic unstructured environments using free-form obstacle delimiters. In: European Conference on Mobile Robots (ECMR). pp. 367–372.

von der Hardt, H. J., Wolf, D., Husson, R., 1996. The dead reckoning localization system of the wheeled mobile robot romane. In: IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems. pp. 603–610.

Wadai, M., 2000. A synchro-caster drive system for holonomic and omnidirectional mobile robots. In: 26th Annual Confjerence of the IEEE on Industrial Electronics Society. pp. 1937–1942.

Woojin, C., Hoyeon, K., Yoonkyu, Y., Chang-Bae, M., Jooyoung, P., 2012. The detection and following of human legs through inductive approaches for a mobile robot with a single laser range finder. In: Industrial Electronics, IEEE Transactions on. Vol. 59. pp. 3156–3166.

Xiao, J., 2014. Mobile robot locomotion. [Online; accessed 6 July 2014].
URL www-ee.ccny.cuny.edu/www/web/jxiao/Design-Locomotion.pptl

Xudong, M., Xianzhong, D., Wen, S., 2005. Vision-based extended monte carlo localization for mobile robot. In: IEEE International Conference on Mechatronics and Automation. Vol. 4.

Yamamoto, M., Shimada, M., Mohri, A., 2001. Online navigation of mobile robot under the existence of dynamically moving multiple obstacles. In: Proceedings of the IEEE International Symposium on in Assembly and Task Planning. pp. 13–18.

Yan, Y., Zhu, Q., Cai, C., 2006. Hybrid control architecture of mobile robot based on subsumption architecture. In: Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation. pp. 2168–2172.

Zeng, W., Church, R. L., 2009. Finding shortest paths on real road networks: the case for a*. In: International Journal of Geographical Information Science. Vol. 23. pp. 531–543.
URL http://dx.doi.org/10.1080/13658810801949850

Zhiqiang, Y., Gao, M., Huaping, L., Xiaoyan, D., Jianhua, L., Qiurui, W., Yuewei, L., 2008. Dynamic obstacle avoidance in polar coordinates for mobile robot based on laser radar. In: Pacific-Asia Workshop on Computational Intelligence and Industrial Application. pp. 334–338.

Zhou, J., Loulin, H., 2011. Experimental study on sensor fusion to improve real time indoor localization of a mobile robot. In: IEEE Conference on Robotics, Automation and Mechatronics (RAM),. pp. 258–263.

Zhuang, H., Li, H., S., D., 2006. On-line real-time path planning of mobile robots in dynamic uncertain environment. In: Journal of Zhejiang University SCIENCE. Vol. 4. pp. 516–524.

Appendix A

Line Extraction Using Hough Transform

```
clc
clear all
close all
%Generate an image of two lines
E=zeros(300,300)
for i=50:250
E(i,i+5)=1;
E(300-i,i+5)=1;
end
imshow(E)
imwrite(E,'Houghimage.jpg')
RGB = imread('Houghimage.jpg');
% Convert to intensity.
I = RGB;
% Extract edges.
BW = edge(I,'canny');
% Apply Hough Transform
```

```
[H,T,R] = hough(BW,’RhoResolution’,2,’Theta’,-90:0.5:89.5);

axis off

figure()

imshow(imadjust(mat2gray(H)),’XData’,T,’YData’,R,... ’InitialMagnification’,’fit’);

%title(’Hough Transform of Gantrycrane Image’);

xlabel(’\theta’), ylabel(’\rho’);

set(gca, ’ytick’, []) ;

set(gca, ’xtick’, []) ;

xlabel(’\theta’,’fontsize’,20,’FontAngle’,’ITALIC’,’color’,’k’);

ylabel(’\rho’,’fontsize’,20,’FontAngle’,’ITALIC’,’color’,’k’);

set(gca,’LooseInset’,get(gca,’TightInset’));

figure imwrite(imadjust(mat2gray(H)),’Houghimage2.jpg’)

TY = imread(’Houghimage2.jpg’);

imshow(TY)

xlabel(’\theta’,’fontsize’,20,’FontAngle’,’ITALIC’,’color’,’k’)

ylabel(’\rho’,’fontsize’,20,’FontAngle’,’ITALIC’,’color’,’k’)
```

Appendix B

Wavefront Algorithm

```
function Wavefront planner

clc

clear all

close all

COL='mrkg'

COL=[COL COL COL COL COL];

% Global Parameter global MP;

global x1 x y y1 COL F

r=0.05; % Map resulation

w=.5; % robot raduis

R=[0 0]; % initial position

G=[6 5]; % Goal location

ad=round(w/r); % number of pixles to enlarge obstacles

Me=imread('a.png'); % read robot map

M=Me';

mm=size(M);

RC=round(R/r)+round(mm/2) % location of the robot on the map
```

```
Gg=round(G/r)+round(mm/2) % location of the goal on the map
ML=M;
imshow(M)
for i=1:mm(1)
for j=1:mm(2)
if(M(i,j)==0)
ai1=i-ad;
if(ai1<1)
ai1=1;
end
ML(ai1:i,j)=0;
ai2=i+ad;
if(ai2>mm(1)) ai2=mm(1); end ML(i:ai2,j)=0; aj1=j-ad; if(aj1<1)
aj1=1;
end
ML(i,aj1:j)=0;
aj2=j+ad;
if(aj2>mm(2))
aj2=mm(2);
end
end
ML(ai1:ai2,aj1:aj2)=0;
end
end
figure imshow(ML)
```

```
s1=8; s2=8;
[x,y]=meshgrid(1:s1:mm(1),1:s2:mm(2));
plot(x,y,'k')
hold on
[y1,x1]=meshgrid(1:s2:mm(2),1:s1:mm(1));
plot(x1,y1,'k')
for i=1:length(1:s1:mm(1))-1
length(1:s1:mm(1))-i
for j=1:length(1:s2:mm(2))-1
MP(i,j)=5000;
for ik=x1(i) :x1(i+1)
for jk= y(j) :y(j+1)
if(ML(ik,jk)==0)
fill([x1(i) x1(i+1) x1(i+1) x1(i) ],[y(j) y(j) y(j+1) y(j+1)],'b')
MP(i,j)=1;
end
end
end
end
end
F=round(RC./[s1 s2])
i=F(1)
j=F(2)
fill([x1(i) x1(i+1) x1(i+1) x1(i) ],[y(j) y(j) y(j+1) y(j+1)],'g')
F1=round(Gg./[s1 s2])
```

```
i=F1(1)

j=F1(2)

MP(i,j)=2;

fill([x1(i) x1(i+1) x1(i+1) x1(i) ],[y(j) y(j) y(j+1) y(j+1)],'g')

plot(RC(1),RC(2),'*')

CK()

end

function CK()

global MP; global x1 x y y1 COL F

for ty=2:376

clear I J

[I J]=find(MP==ty)

if( isempty(I)==1)

break;

end

for kf=1:length(I)

ik=I(kf);

jk=J(kf)

for xk= ik-1:ik+1

for yk=jk-1:jk+1

MP(xk,yk)=min( MP(xk,yk),ty+1);

if( MP(xk,yk)>2)

i=xk;

j=yk;

fill([x1(i) x1(i+1) x1(i+1) x1(i) ],[y(j) y(j) y(j+1) y(j+1)],COL( MP(xk,yk)))
```

```
end
end
end
end
end
io=1;
sb(io,:)=[F(1),F(2)]
for ty=MP(F(1),F(2))-1:-1:2
tyy=ty;
ik=sb(io,1);
jk=sb(io,2)
for xk= [ik ik-1 ik+1]
for yk=[jk jk-1 jk+1]
MP(xk,yk)
if( MP(xk,yk)==tyy)
tyy=900;
sb(io+1,:)=[xk,yk]
i=xk;
j=yk;
fill([x1(i) x1(i+1) x1(i+1) x1(i) ],[y(j) y(j) y(j+1) y(j+1)],'w')
end
end
end
io=io+1;
end ; end
```

Appendix C

Avoiding Two Humans

Consider the case shown in Figure C.1, in which the robot task is to avoid two humans namely: human A and human B moving in straight lines at speeds of 0.3m/s and 0.5m/s receptively. The mobile robot initially begins to move away from human A who moves in the same direction as the robot, then avoids human B who moves in the opposite direction by following human A (Figure C.2), after human B passes the robot avoids human A and resumes its path (Figure C.3).

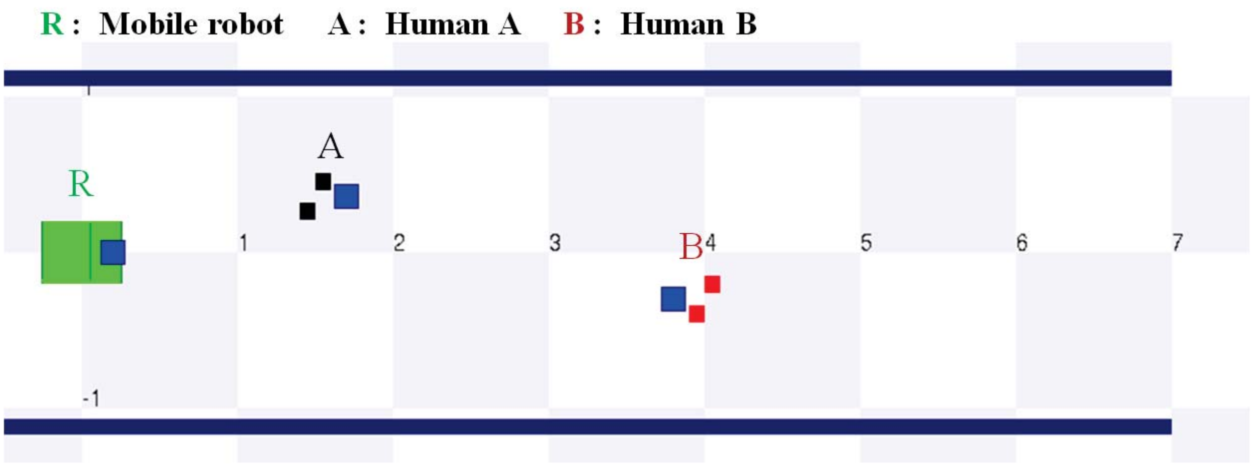


Figure C.1: Initial positions of the robot and humans

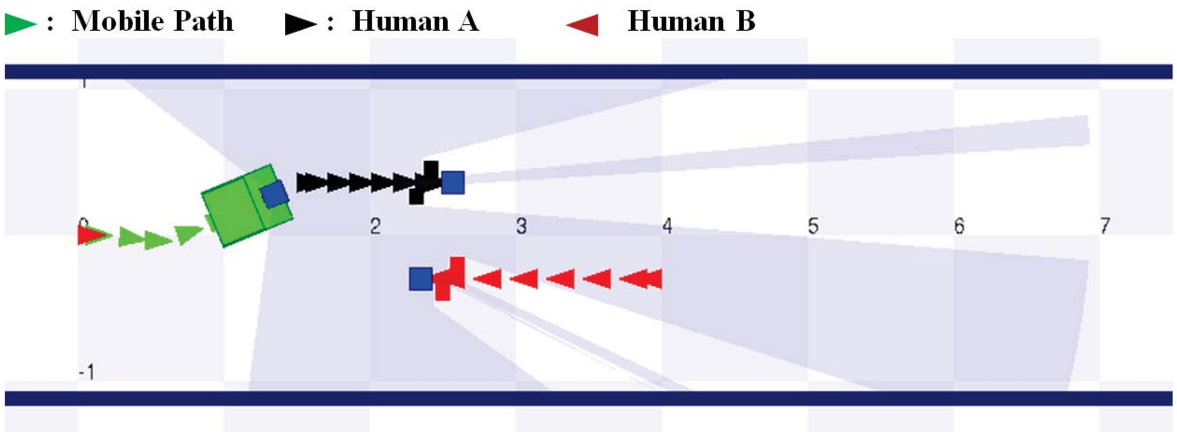


Figure C.2: Robot avoids human B by following human B

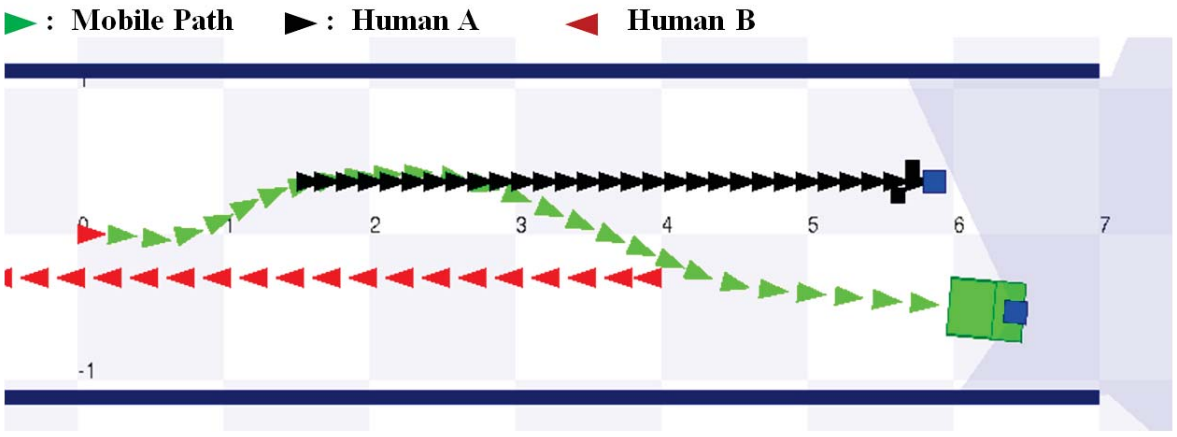


Figure C.3: Avoiding two humans

Appendix D

Avoiding a Human and Moving Obstacle

In this test, the robot has to avoid a dynamic obstacle (Nubot) moving at a speed of 0.30m/s in the same direction, and a human walking in the opposite direction. As shown in Figure D.1, the human appears at the laser data at t=2.6s, this forces the robot at t=4s to follow the dynamic obstacle for minimizing reduce the collision risk, at t=6.5 the human passes the robot and disappear from the laser data, therefore at t=8s the robot starts to avoid the dynamic obstacle.

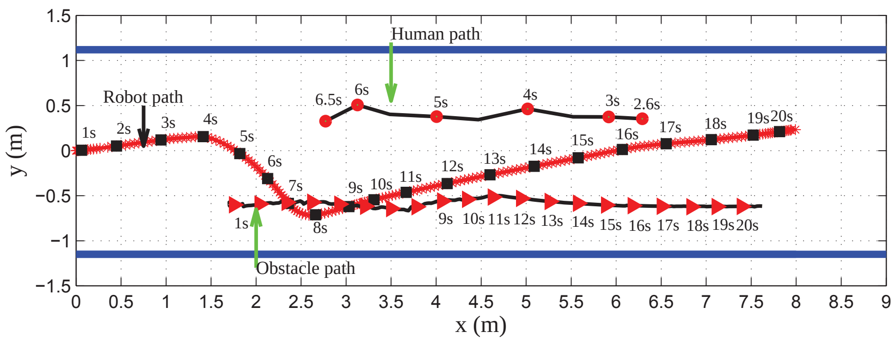


Figure D.1: Avoiding an obstacle and human

Appendix E

Curvature

Curvature of a curve at point (x,y) is given a function of tangent angle ψ and curve length s:

$$k = |\frac{\mathrm{d}\psi}{\mathrm{d}s}| \tag{E.1}$$

$\frac{\mathrm{d}\psi}{\mathrm{d}s}$ can be written as following:

$$\frac{\mathrm{d}\psi}{\mathrm{d}s} = \frac{\mathrm{d}\psi}{\mathrm{d}x}\frac{\mathrm{d}x}{\mathrm{d}s} \tag{E.2}$$

Form Figure E.1, it can write the following expressions:

$$\delta s^2 = \delta x^2 + \delta y^2 \tag{E.3}$$

$$(\frac{\delta s}{\delta x})^2 = 1 + (\frac{\delta y}{\delta x})^2 \tag{E.4}$$

It can write in this relation as a derivative form:

$$\frac{\mathrm{d}s}{\mathrm{d}x} = \sqrt{1 + (\frac{\mathrm{d}y}{\mathrm{d}x})^2} = \sqrt{1 + (y')^2} \tag{E.5}$$

$tan(\psi)$ is given as :

$$tan(\psi) = \frac{dy}{dx} = y^{'} \tag{E.6}$$

Differentiation of this relation is given as:

$$(1 + tan(\psi))\frac{\mathrm{d}\psi}{\mathrm{d}x} = \frac{\mathrm{d}2y}{\mathrm{d}x^2} \tag{E.7}$$

$$(1 + \frac{dy}{dx})\frac{\mathrm{d}\psi}{\mathrm{d}x} = \frac{\mathrm{d}2y}{\mathrm{d}x^2} \tag{E.8}$$

$$\frac{\mathrm{d}\psi}{\mathrm{d}x} = \frac{y^{''}}{(1 + y^{'})} \tag{E.9}$$

Form Equation 1, 2, 5 and 9 it can write

$$k = \frac{y_i^{''}}{(1 + (y_i^{'})^2)^{3/2}} \tag{E.10}$$

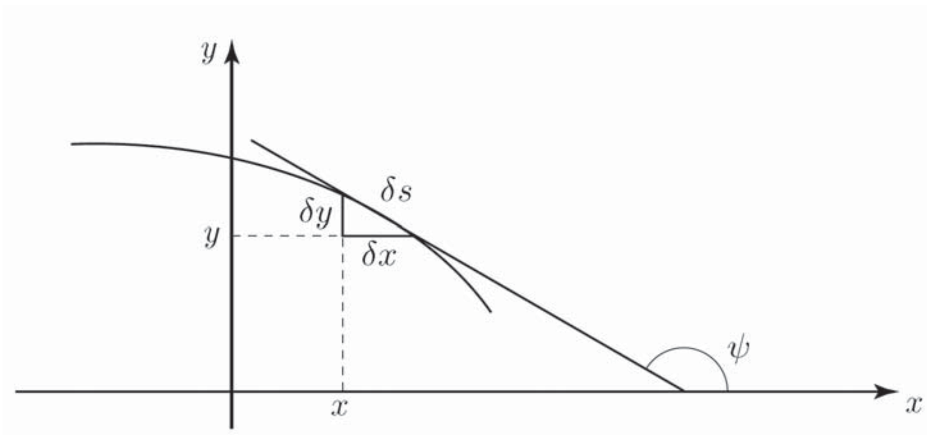


Figure E.1: Curvature of a curve