Human Friendly Robot

Control of Mobile Robots with Human Collision Avoidance using Dempster-Shafer Method and Velocity Potential Field Controller

Yu Hu

A thesis submitted for the Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of

Master of Applied Science in Mechanical Engineering

L'Université canadienne
Canada's university

University of Ottawa, Ontario, Canada

April 30, 2014

Abstract

In this project, a novel human friendly mobile robot navigation controller is investigated. By applying this controller, the mobile robot is able to work in a complicated environment with several humans and other obstacles avoiding them before a collision happens. This robot will have a preference in avoiding humans over other obstacles keeping human safety as its first consideration. To achieve this goal, three problems have to be solved. The first one is the robot should be able to "see" the environment and distinguish the human and the obstacles. The functions of human sensor and sonar sensor are presented. A new sensor fusion method for combining the information collected by these two sorts of sensors based on Dempster-Shafer evidence theory is also proposed. By using the sensor fusion method, the robot will have a better view of human. The second problem is the robot has to know how to avoid collision. A new navigation algorithm, based on an improved velocity potential field method, is then described. The way of calculating the distances of avoidance based on different kinds of obstacles is presented as well. The last problem is how to make the mobile robot put human as its first priority when avoiding collision. A summary of the methods which are used to protect human is mentioned. According to the simulation and the experimental results, the new mobile robot navigation controller successfully led the robot avoid collisions in complicated situations and always put human safety as its first consideration.

Acknowledgement

I would like to express my appreciations to my supervisor Dr. Dan Necsulescu for his patience, support, inspiration, direction and suggestions. With his help, I overcome countless difficulties in the research and finally, find the right direction of my thesis. I would also like to thank Dr. Jurek Sasiadek and Dr. Davide Spinello for their suggestions and time. I also thank my colleagues Elisha Pruner, Jin Bai, and Guangqi Nie in the research lab for their encouragement and advices. Finally, I would like to thank my parents for their love, care, patient and ongoing support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Mobile Robot in the World

There are 4 main types of mobile robots: Unmanned Ground Vehicles (UGVs), Unmanned Aerial Vehicles (UAVs), Autonomous Underwater Vehicles (AUVs) and polar robots. In the 21 century, the mobile robots become more widely used in the world, especially in military, civilian and industry applications, due to their characteristics: durable and expendable. For instance, in terms of the military reconnaissance applications, American army tends to rely more on the Unmanned Aerial Vehicles (UAVs) in intelligence, surveillance and reconnaissance while in the civilian applications, the mobile robots can also make a significant contribution in terms of distribution, cleaning and security.

Finally, in terms of industrial applications, mobile robots also started to replace humans to do dull, poorly paid or dangerous jobs in natural resources sectors such as mining, oil and gas and agriculture.

1.2 The Challenges of Mobile Robot Safety

To become more widely applied in the real world, it is necessary for the mobile robots to have much more artificial intelligence and capability to work in a more complex environments which might include humans or vehicles with unknown and unpredictable motions. Thus, the safety question whether a mobile robot has the ability to identify a potential risk and prevent it has addressed. In this thesis, the potential risk for mobile robot mainly refers to the collision between the mobile robot and the obstacles in the environment, especially with human. Actually, human safety will always be put as a top priority rule for robots. In the famous The Three Laws of Robotics (1.A robot may not injure a human being or, through inaction, allow a human being to come to harm.2.A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.) were devised by the science fiction author Isaac Asimov. It very obviously appears the writer's consideration the human safety. From the mobile robot prospective, due to the fact that motions of humans are constantly changing, the requirement of the artificial intelligence for the mobile robot is much higher than the stationary one. This issue becomes more complicate, for example, in a factory, where the mobile robot carries a part and delivers it to the workers and it should not only pay attention to the workers walking near it but also to the equipment in the plant. Sometimes, the mobile robot might face the complicated situations such as both human and equipment are in its way and it has to figure out which detour routes is

safer for human in a short time. Furthermore, as a human-friendly mobile robot, human safety must always be a top consideration and the humans should be able to conduct their own job without worrying about the mobile robots near them. The mobile robots must “watch” the environment around them and respond to different kinds of situation at all time to ensure the safety of humans who is near them. Besides, at the same time, mobile robots also need to maintain the attention on other objects in their active area to prevent the collision between them and those objects.

1.3 Research Objective

In the research before, numerous solutions have been developed for mobile robots to keep human safe such as limiting the access of the human to the mobile robot, limiting the potential and kinetic energy of the mobile robot (by limiting its mass and velocity) and developing a navigation algorithm for mobile robots to avoid the dangers. The first two kinds of methods are not flexible since they limit the mobile robots' working area and capability. Consequently, in this thesis the focus is on the navigation algorithm. The studies before tended to design a mobile robot navigation algorithm based on a relatively simple environment, i.e. only with less than two kinds of dangerous objects such as walking human and stationary obstacle while in this thesis, we will bring in all kinds of objects (moving human, stationary human, moving obstacle and stationary obstacle) to build a more complex environment. Based on this, a new navigation algorithm that can be more widely applicable will be developed.

In order to improve the human safety and the environment safety protection for the mobile robot, various kinds of sensors and a new algorithm are needed. In this research, the goal is to put two moving humans, moving obstacles and fixed obstacles in the environment, which means to design a more complex environment. Also, based on a virtual field method, we plan to change and improve the previous algorithms from the thesis of Elisha Pruner to avoid the obstacles and human with different priorities. More precisely, avoid a moving human at first and then the obstacles. In the experimental study, human sensors which are also called passive infrared sensors will be added to identify the difference between human and obstacle. Besides, we will add a new dangerous situations when the collision can happen, for example when the robot detects moving objects are coming from all the directions, it will choose the less important object to collide and once it touch the surface of the object, it stop immediately. With the priority and dangerous situation be considered, our new algorithms will be much safer for the human and cause less damage for the environment. The new algorithms will be written into the controller of the robot and first be tested in the simulation, and then it will be applied in the experimentation in our lab.

1.4 Method of Approach

After the controller has been improved, four different situations will be tested. Firstly, the robot will avoid one obstacle, one stationary human and one moving human to test

all three kinds of robot sensors. Secondly, the robot will encounter a situation that both human and obstacle are in front of it to test if the sensors of the robot can tell the difference of the human and robot. For moving obstacle such as vehicle, the robot can only detect it by the sonar sensor. Then, we will create three extreme situations that the robot will face the walking human and obstacle at the same time. Decisions will be made by the robot because in these extreme situations, it cannot avoid all the human and obstacle perfectively. By telling which kinds of decisions the robot made, we can test the human priority function of the robot. Lastly, the robot will enter a simulated city environment, we will check if the robot can only use the roadway to get to the goal while also consider about the human safety.

1.5 Thesis Layout

This thesis consists of eight chapters that go through the design and executing of the new mobile robot navigation controller with the top consideration of human safety.

Chapter 2 provides a literature review on sensor fusion and the way robot detect and avoid the human and obstacles.

In Chapter 3, we will introduce the two kinds of sensors which we will use in this thesis briefly.

Chapter 4 describes our proposed navigation controller with the consideration of human safety and a simple trajectory planning method.

In Chapter 5, the hardware and software components which will be used in our simulation and experiment are introduced.

In Chapter 6, the way how we designed the simulation is introduced. The simulation results of different situations are listed as well.

Chapter 7 presents experimental results described and compared with the simulation results.

Chapter 8 provides a conclusion of this thesis with our research contributes and the future work of our project.

Chapter 2

Literature Review

2.1 Introduction

In this thesis, both human and obstacles will appear in the test environment and we cannot predict their positions, number and state of motion. Besides, compared with obstacles, human is more important. Several additional challenges of human should be considered. First of all, human motion is unpredictable. For instance, unlike vehicles, human motion does not have its own trajectory. For example, a human can stop and move sideways without turning or directly turn left/right without a pause. So predicting human position based on the previous human motion is not reliable. Secondly, from the security perspective, human should have a higher priority than obstacles. Sometimes human may cannot see the mobile robots because the heights of robots. The robot should detour the human all the time and never ever let human worry about the robots around them. Lastly, how to identify a human is also a difficult problem, since in our simulations, human and obstacle may have the same motion states and moving trajectories. We tend to use the human sensors which can only detect human in our case. Thus, more than two kinds of sensors are necessarily needed. How to combine the information got from different kinds of sensors should be considered as well.

To navigate a mobile robot in such complicated environment, different kinds of sensors and a brand new navigate method will be needed. Next, we will introduce the relevant literatures in the view of the questions which are mentioned above.

2.2 How to Detect the Human and Obstacles

Perceiving the environment is vital in any application related to mobile robotics research. In other words, to make a mobile robot move, the first step is to give the robot knowledge of the environment around. Next, we will introduce the methods which people used during last decades for this purpose.

In [1], a new low-cost IR sensors based on the light intensity back-scattered from objects and able to measure distances of up to 1m is described. Also, the sensor model is presented and the expected errors in distance estimates are analysied and modelled. As a result, the sensor is able to be used by the mobile robot to build reasonably accurate maps within the real time constraints, given the fast response time of the IR sensor.

In [2], Tarek Mohammad described the Phong Illumination Model for determining the properties of a surface and subsequently calculating the distance to the surface. The angular position of the IR sensor is computed as normal to the surface in order to simplify the calculation. Besides, ultrasonic (US) sensor can provide the initial

information on distance to obtain the parameters for this method. Finally, the experiments indicated that even the low cost US and IR sensors are able to give reliable distance measurement.

In [3],EvsenYanmazand Hasan Guclustudied is presented the target detection and tracking problem in mobile sensor networks, where the performance metrics of interest are probability of detection and tracking coverage, when the target can be stationary or mobile and its duration is finite. They proposed a local, cooperative, coverage-based mobility model to improve stationary or mobile event coverage in mobile sensor networks. The model uses local topology information and no application specific details are considered.

Many unmanned aerial vehicle (UAV) are equipped with at least one forward-looking camera. In [4], Ryan J.Kephart and Michael S. Braasch described a See-and-avoid method, which is currently approved by FAA for pilots to avoid objects and other aircraft while flying in visual meteorological conditions. They discussed the flight-testing performed to establish air traffic detection ranges for low-time pilots, also for a low-cost UAV camera system, which was evaluated to determine if it is able to provide the equivalent see-and-avoid performance as the tested pilots. After several subject pilots had gone through the experiment, the result showed that the pilots were always identifying the traffic aircraft before the camera system and some of the traffic aircraft were not visible on the cameras during the initial subject pilot testing.

In [5], Jian-shuen Fang, Qi Hao, David J.Brade, Bob D.Guenther and Ken Y.Hsu proposed a real-time human identification system using a pyroelectric infrared (PIR) detector array and a hidden Markov models (HMMs). They used a PIR detector array with masked Fresnel lens array to generate digital sequential data which can represent a human motion feature. By evaluating a set of new feature data against the trained HMMs using the maximum-likelihood (ML) criterion, human subjects were recognized. The digital feature's advantages are in its less rigid training process, decreased sensitivity to walking speeds, effectiveness in the path-independent identification mode and high data compression ratio for wireless data transmission. A prototype system was developed to verify the proposed method. Besides, the writers also tested sensor modules with different numbers of detectors and different sampling masks to maximize the identification capability of the sensor system.

In [6], Andreas Hartmann introduced the PIR sensor's structure, function, pros and cons. PIR devices were able to detect a person moving into or through a detection zone with high reliability. Even the slightest positive or negative thermal radiation change in contrast to a background, triggers the sensor element and there will be no interference between the units nearby due to the passive nature of the detection principle. The precision optics on the sensor has the ability to define the field of view, allowing consistent and long-range coverage.

In the field of mobile robotics, infrared sensors are being widely used for many applications, such as navigation and localization. In[7], a new approach to real-time human detection through processing video captured by a thermal infrared camera mounted on the Spanish private company MoviRobotics S.L. indoor autonomous mobile platform mSecurit^{TM} has been presented. The infrared-based approach started with static analysis for the detection of human candidates. After that, a dynamic analysis by means of the optical flow algorithm based on Lucas and Kanade approach without pyramids or the standard image difference method was carried out. The robot in the experiment can detect humans feeding the algorithms with thermal infrared frames. An intelligent switching between image subtraction and optical flow which depend on the platform movement and the processing load was also introduced.

In [8],Guodong Feng, Xuemei Guo and Guoli Wang presented an infrared motion sensing system for human-following robots. The sensing system consisted of a geometric sensing layer and a cooperative sensing layer, which could directly generate the bearing measurements of the moving target and is also lightweight and robust to the environmental changes. From multiple perspectives with the least squares method, the target is localized through fusing the bearing measurements and does not involve complex computation. Thus, the infrared motion sensing system has its own advantages in the mobile sensing applications.

Nowadays, there are more and more car navigation systems using the actual video

which is captured by camera. In [9],a detection method for traffic lights that is used for estimating locating of crossroads in image is described. Actually, since it is not easy to detect the crossroad, it is a method that effectively detects traffic lights in a long distance instead of the crossroad and finally, people can estimate the location of crossroads as a result. With the new method, three main results such as traffic light detection that processes color thresholding, finding center of traffic light by Gaussian mask and verifying the candidate of traffic light using suggested existence-weight map can be achieved.

In [10], a novel approach of using autonomous mobile robots to deploy a Wireless Sensor Network (WSN) for human existence detection in case of disasters is described. The mobile robots performed cooperative Simultaneous Localization and Mapping (SLAM) and communicate over the WSN during WSN deployment. The writers also described the important advantages of the system over a human-assisted system, including autonomous deployment, aggregated intelligence and flexibility.

2.3 Sensor Fusion Method

After the robot “senses” the environment through different kinds of sensors, next step is to combine different data which is collected by all the sensors on the robot together. The papers below describe several methods for the sensor fusion.

Normally, there is more than one sensor in one system. The paper of[11], describes a data fusion algorithm. In the paper, the mobile robot system consists of multiple passive infrared sensors (PIR), two color cameras, a pair of microwave sensors and a pair of PCs for data collection, signal processing and data fusion. The area around the robot is subdivided into an occupancy grid with 0.5m by 0.5m cells. The data fusion algorithm which is based on Dempster-Shafer evidence theory is used to fuse the uncertain sensor information and estimate the probability of human occupancy for each cell. Furthermore, the result of the estimation is used to locate the human's position.

In [12], the authors reviewed techniques for sensor fusion in robot navigation and algorithms for self-location. They described how to integrate the sensor reading to help the robot seek to accomplish tasks such as constructing a map of its environment, locating itself in that map and recognizing objects that should be avoided or sought. In the paper, the integration techniques are divided into two categories. The first one is low-level fusion, which is used for direct integration of sensory data, resulting in parameter and state estimates. The other one is called high-level fusion. People use this technique for indirect integration of sensory data in hierarchical architectures, through command arbitration and integration of control signals suggested by different modules. The authors also provided several ways to solve the problem in machine intelligence in terms of Kalman filtering, rule-based techniques, behavior based algorithms and approaches that borrow from information theory, Dempster–Shafer

reasoning, fuzzy logic and neural networks. Some future-research needs are also mentioned by the writers including: robustness of decision rules; simultaneous consideration of self-location, motion planning, motion control and vehicle dynamics; the effect of sensor placement and attention focusing on sensor fusion; and adaptation of techniques from biological sensor fusion.

In the book [23], the author discussed the knowledge in terms of target and background signature-generation phenomena, sensor design, signal processing algorithms, available communications types and bandwidths, and end use of the fusion products. From chapter 3 to chapter 9, different sensor fusion methods-JDL data fusion, classical inference, Bayesian inference, Dempster-Shafer evidential theory, artificial neural networks, voting algorithm and fuzzy logic and fuzzy neural networks were described, respectively.

In [24], a method of building an accurate map was presented. The Dempster-Shafer inference rule was used to solve the ultrasonic sensors' inherent uncertainty problem at the origin of measurements by integrating sensor and modeling information. The map is built on a two-dimensional occupancy grid. Finally, by using sonar data in the real environment, the map building system was experimentally evaluated.

In [26], two less well known but useful aspects of Dampster-Shafer evidence theory for sensor fusion on autonomous mobile robots were described by Robin R. Murphy,

which are called weight of conflict metric and the enlargement of the frame of discernment. The first one is used to calculate the amount of consensus between sensors to allow the robot to react appropriately when emergency happen such as one of the sensors malfunctioned. The second one-enlargement of the frame of discernment offers the advantage of perceptual abstraction, and permits expert knowledge about the domain to be embedded in the frames of discernment, simplifying the construction and maintenance of the knowledgebase.

Bayesian inference can be used to integrate data from multiple sensors into a common description of the environment as well. In [25], the writers proposed a cellular representation named the occupancy grid. It was used to combine range information from one-dimensional stereo and sonar sensor into a two-dimensional map of the nearby environment of a robot. The probability that a grid is empty or occupied by an object is contained by each cell, which is obtained from sensor models that present the uncertainty of their range data. By using Bayesian inference, the existing information from each sensor can be combined and updated so that the robot can "see" a much clearer map.

2.4 How to Navigate the Robot

Once the mobile robot combines all the information its sensors collected, the controller inside or outside of the robot will start to navigate it to achieve the goal. For

decades, a large variety of methods are created to navigate the mobile robots according to different situations.

In [13], a problem of wheeled mobile robot (WMR) navigation towards an unknown target in a cluttered environment has been considered. A new algorithm which is called biologically inspired navigation algorithm is introduced. The algorithm combines two different modes: a goal-directed navigation which is the equiangular navigation guidance law and a local obstacle avoidance technique. In avoidance mode, by using all the active sensors in the system to get the necessary information about the obstacles near the robot, the robot can detect the en route obstacles in real time. Besides, during the avoidance of the obstacles, the angle between the instantaneous moving direction of the robot and a reference point on the surface of the obstacle is kept constant. Thus, the robot is able to avoid and detour the obstacles successfully, while preserving a predefined security margin.

In [14],a high-level approach to modeling, analyzing, and verifying complex safety-critical systems through a case study on the Traffic Alert and Collision Avoidance System(TCAS) is demonstrated. (TCAC is an avionics system that detects and resolves aircraft collision threats.) The writers advocate defining high-level hybrid system models that capture the behavior not only for the software but also for the airplanes, sensors, pilots, etc. Particularly, how the core components of TCAS can be captured by relatively simple hybrid I/O automata is described. After that, the

writers introduced a methodology for establishing conditions under which TCAS guarantees sufficient separation in altitude for aircraft involved in collision threats.

The paper [15] discusses how a behavior-based robot can construct a "symbolic process" that accounts for its deliberative thinking processes using models of the environment. The paper focuses on two essential problems; one is the symbol grounding problem and the other is how the internal symbolic processes can be situated with respect to the behavioral contexts. We investigate these problems by applying a dynamical system's approach to the robot navigation learning problem. The formulation, based on a forward modeling scheme using recurrent neural learning, shows that the robot is capable of learning grammatical structure hidden in the geometry of the workspace from the local sensory inputs through its navigational experiences. Furthermore, the robot is capable of generating diverse action plans to reach an arbitrary goal using the acquired forward model which incorporates chaotic dynamics. The essential claim is that the internal symbolic process, being embedded in the attractor, is grounded since it is self-organized solely through interaction with the physical world. It is also shown that structural stability arises in the interaction between the neural dynamics and the environmental dynamics, which accounts for the specificity of the internal symbolic process. The experimental results using a mobile robot, equipped with a local sensor consisting of a laser range finder, verify the claims.

In [16], a vision system for an autonomous mobile robot is presented. By using visual landmarks, the mobile robot can find and reach the goal position in an unknown indoor environment. In order to extract useful information for robot navigation, the robot vision processes are conducted at two levels of abstraction. More specifically, the whole system carries out two groups of parallel processes at the same time. The first one, called Low Level Vision System, performs obstacle avoidance and corridor following. The second one, called High Level Vision System, extracts landmark's contents for high level planning.

In [17], the authors proposed a novel collision cone approach to help the mobile robot detect and avoid the irregularly shape moving objects with random trajectories. The collision cone concept is derived from existing analytical results, which are available in aerospace literature that enables prediction of collision between two moving point objects. The authors then improved and extended the result step by step from predicting collision between a point and a circular to a point and an irregularly shaped object, two circular objects and finally, two irregularly shaped objects. Besides, based on the collision cone approach, several strategies which can help the mobile robot avoiding collision are described. The way to reduce computational burden by approximating the shapes of the robot and obstacles are presented as well. In the end, the experiments showed that the approach is effective and suitable for all kinds of shapes of either the robots or the obstacles.

In [18], in order to help the micro air vehicles to conquer the certain instances such as flying directly into a corner, a new MAV platform is introduced. Within the platform, the micro air vehicles are able to carry out a quick transition from cruise flight into a hovering mode to avoid the danger. The hybrid MAV combined the advantages of a fixed-wing aircraft (endurance) and a rotorcraft (hovering). The applications and design of a hybrid MAV in conjunction with sensing and control techniques to perform autonomous hovering and collision avoidance are described in the paper, which refers to hovering a fixed-wing MAV autonomously.

Sometimes, it is necessary for robots to work with humans in one environment. In [19], the authors created a new robot which is capable of navigating in an unknown, highly populated urban environment, only using the information extracted through communication with pedestrians and its own sensors. The algorithms and architecture that make up the navigation subsystem of Autonomous City Explorer (ACE), aim to combine the research fields of autonomous mobile robot navigation and human robot interaction. Furthermore, the authors also described the algorithms which are used for Simultaneous Localization and Mapping (SLAM), path planning in dynamic environments and behavior selection. After that, they presented how to integrate the algorithms with the system architecture to a complete working system.

In [20], a new virtual force field (VFF) based collision avoidance algorithm for non-holonomic mobile robot has been developed for avoiding unpredictable dynamic

obstacles, including human. Since the authors designed a novel virtual force-the detour force and a new virtual repulsive force, compared with the prior two algorithms, the new algorithm reduces the time cost for reaching the goal and the path oscillations significantly. Besides, there is a proposal, which I followed in this thesis, which created an active and a critical region as Figure 2.1 shows.

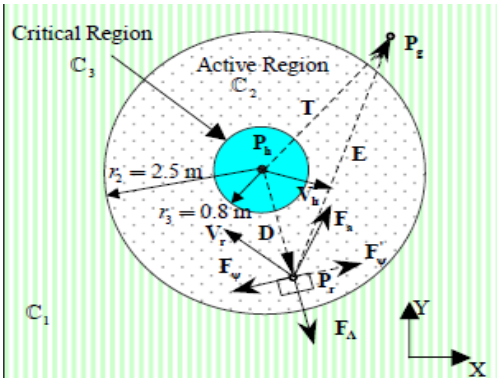


Figure 2.1 Configuration of collision avoidance for a mobile robot and a human obstacle [20]

The size of the regions is based on robot and human shapes, velocity limits and acceleration limits. The active region C2 is defined as the region near the human. If anything is in C2, the repulsive and detour forces will be activated to make the robot avoid and detour around the human. The critical region C3 is the region very close to the human and it is very dangerous if anything intrudes into this region. In this region, the robot cannot complete the avoidance and should stop to prevent or at least mitigate the collision. The author defined C as the region that excludes C2 and C3. C1 contains the goal.

In [21], the authors described several approaches to solve the local minima problem

which consists in the fact that the robot might be stuck at a local minimum position before reaching its goal-a common problem of the artificial potential field (APF) method. Furthermore, they integrated the simulated annealing approach into the APF to provide the capability of escaping from local minima to the robot.

The paper proposes [22] a new approach for identifying the robot's trapped state that seems more consistent with how a human would normally understand his trapped condition in an environment by recalling the landmarks he had seen earlier in his travel. A spatio-temporal classifier network is employed for learning and classifying temporal sequences of spatial sensory data that enables the robot to comprehend its immediate environment in terms of landmarks and remember previous experiences of a similar environment. In this way the algorithm differs from other methods that surmount the local minima problem by recollecting previous experiences to understand its trapped condition. The algorithm has been tested on cluttered, concave, and mazelike environments and its efficacy has been established.

In Elisha Pruner Master's thesis [28], a method called velocity potential method was described, which was derived differently from the artificial potential field method. The artificial field method was first introduced by Oussama Khatib and, since then, many variations on the artificial potential field method have been developed to improve robot navigation ability in more complicated situations

The artificial potential field (APF) method creates an area across the robot's map to help the robot go to the goal position. The APF approach treats a robot as a particle moving in a gradient vector field. Positively charged particles are attracted by the "negatively charged goal", while repulsed by "positively charged obstacles".

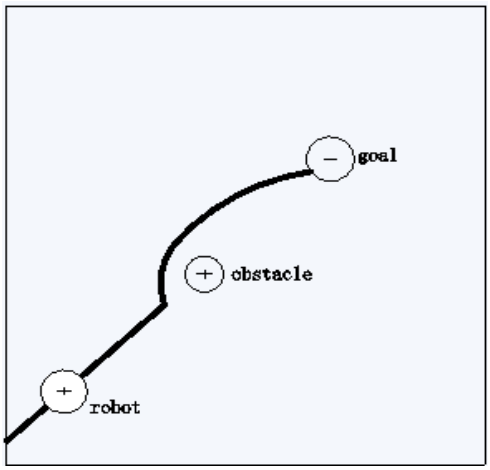


Figure 2.2 The trajectory of the robot obstacle avoidance

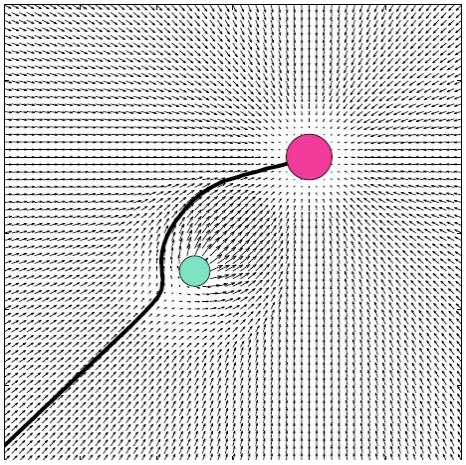


Figure 2.3 The force field created by the APF method [30]

Figure 2.2 illustrates the navigation concept and describes the attractive and repulsive force effect on the robot. The resultant force of repulsive and attractive forces direct the robot from the start position to the goal position while avoiding obstacles. Like a

positive electron, the robot is attracted by the goal which we can put as a negative "electron" as well. The obstacle here can also be treated as a positive electron which, when the robot getting close, it will push it away. Figure 2.3 shows the force the robot will face in the map.

Chapter 3

Sensors

Mobile robots need to have the ability to sense the environment and avoid dangers. For simpler tasks such as obstacle avoidance, only one kind of sensor can handle it while for complicated missions like the reconnaissance mission of the drones, maybe more sensors will be needed. We will make a brief introduction about sensors.

In this thesis, two different sorts of sensors will be used to complete our task. They are human sensor and sonar sensor.

3.1 The Human Sensor

Human sensor is also named as Passive Infrared Sensor (PIR), which is used to detect whether a human has moved in or out of the sensor's range. The basic principle of human sensor is simple. It can detect levels of infrared radiation. Everything emits some radiation based on their characteristics. The hotter the object is, the stronger radiation is emitted. Under normal circumstances, compared with other inanimate objects, humans emit the highest level of radiation. By detecting the appointed level of radiation, the human sensor is able to sense a human. The figure below shows the human sensor.

Figure 3.1 The human sensor

The human sensor detecting area is relatively large. It can detect a human from up to 6 meters away. Furthermore, the movement of human can be detected as well, and the direction of movement can be detected in the range of 1.5 meters. The human sensor operation overview is showed in the Figure 3.2[28].

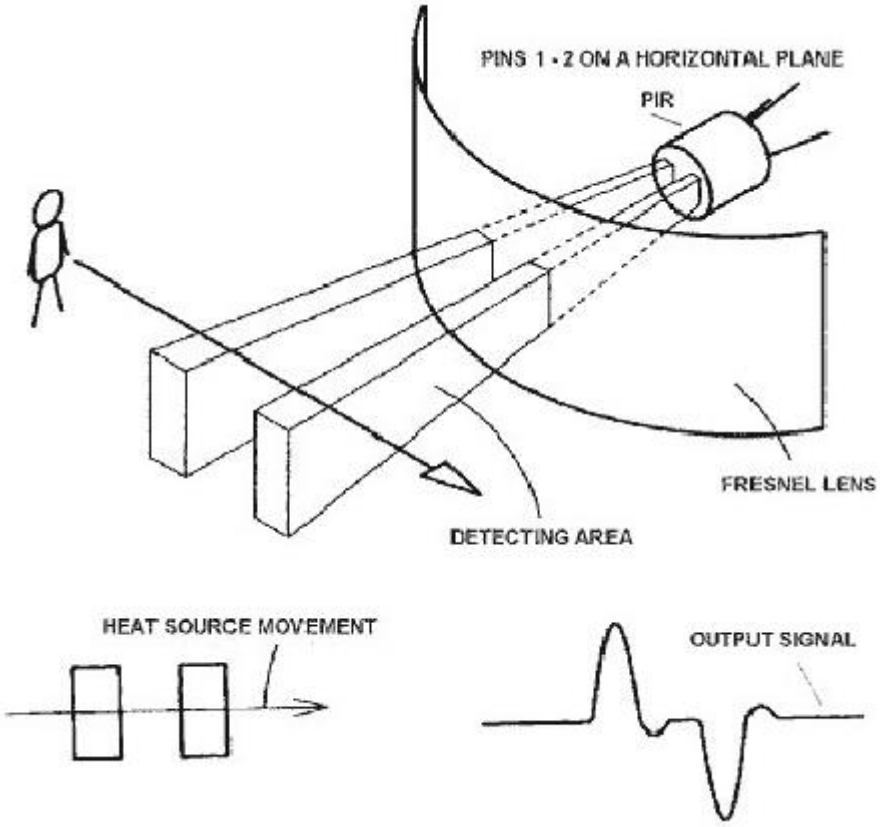


Figure 3.2 The human sensor operation overview [28]

3.2 The Ultrasonic (Range) Sensor

The ultrasonic (range) sensors work on a principle similar to radar or sonar which evaluates attributes of a target by interpreting the echoes from radio or sound waves respectively. It generates high frequency sound waves and evaluates the echo which is received back by the sensor. By calculating the time interval between sending the signal and receiving the echo, ultrasonic sensors are able to determine the distance to an object, as Figure 3.3 shows.

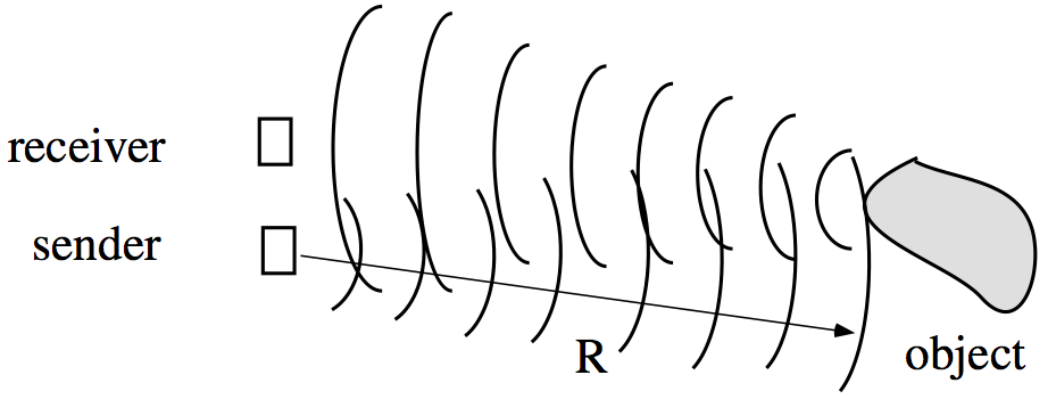


Figure 3.3 The operational principle of ultrasonic sensor [28]

The detection range of ultrasonic sensor is from 0.08 m to 2.55 m, which is shorter than human sensor and the field of view of the sensor is 10 degrees, showed in Figure 3.4. The ultrasonic sensor is shown in Figure 3.5.

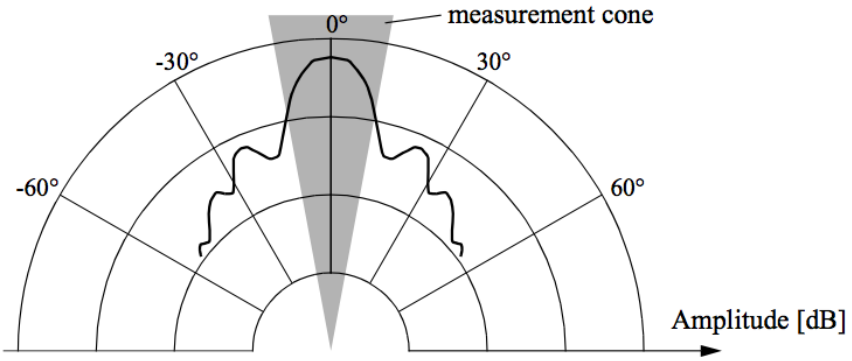


Figure 3.4 The visual angle of ultrasonic sensor [28]

Figure 3.5 The ultrasonic sensor

Chapter 4

Navigation Approach based on Sensor Fusion

4.1 Introduction

In this chapter, a new navigation controller for the non-holonomic mobile robot will be discussed. By applying this controller, the mobile robot will be able to work in a complicated environment with humans. At the beginning of this chapter, we will describe the artificial potential field method and the fluid flow principles of streamlines, in Section 4.2 and Section 4.3, respectively. In Section 4.4, the velocity potential flow theory will be brought in. After that, we will introduce our methods to calculate the sizes of the four different regions based on the characteristics of different obstacles. Then, the sensor fusion method will be described in Section 4.6. Furthermore, our new velocity controller will be presented in Section 4.7. In Section 4.8, a trajectory planning method will be briefly described.

4.2 The Artificial Potential Field Method

The artificial potential field (APF) method was first introduced by Oussama Khatib, and since then, many improvements of the artificial potential field method have been developed to help the robots accomplish more functions.

The artificial potential field (APF) method creates an attractive force and a repulsive force across the whole map that direct the robot to the goal position. The method directs a robot as if it were a charged particle moving in an electric field. Positively charged particles (our robots) are attracted by the negatively charged goal while repulsed by positively charged obstacles.

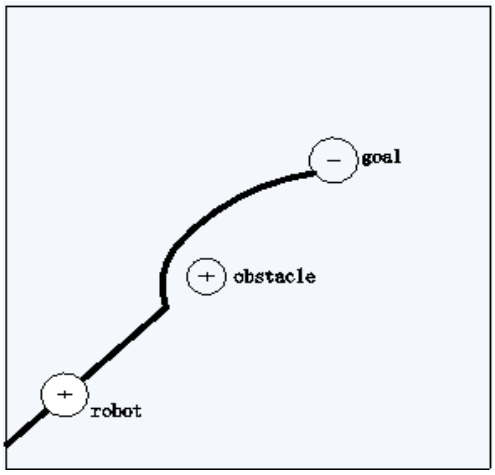


Figure 4.1 The trajectory of the robot

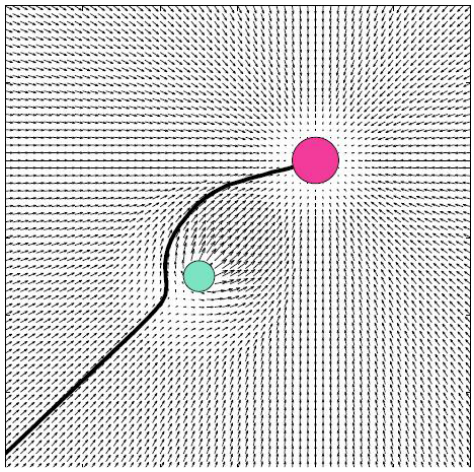


Figure 4.2 The force field created by the APF method [30]

Figure 4.1 shows the trajectory of the robot which is directed by APF navigation

method. The combination of attractive and repulsive forces directs the robot from the start location to the goal location while also avoiding obstacles. Like a positive electron, the robot is attracted by the goal which we can put as a fixed negative "electron" as well. The obstacle here can also be treat as a fixed positive electron, when the robot getting close, it will push the robot away. Figure 4.2 shows the force field created by the APF method.

4.2.1 The Attractive and Repulsive Force in Artificial Potential Field Method

The initial APF method only has two forces- the attractive force and the repulsive force, which means the potential function is the attractive/repulsive potential. Based on this concept, in our experiment, it can be seen that the goal attracts the robot while the obstacles repels it. The combination of the two forces makes the robot move to the goal while avoiding the obstacles. So the potential function can be treated as the sum of attractive and repulsive potentials [30].

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{4.1}$$

where $U_{att}(q)$ is the attractive potential and $U_{rep}(q)$ is the repulsive potential.

There are some criteria the potential field U_{att} should satisfy. First of all, with the distance from the robot to the goal getting larger, U_{att} would increase gradually.

Secondly, when the robot is far away from the goal, the robot approaches it quickly while when the robot is close enough to the goal, the robot would approach it slowly. Based on these, the common attractive field function is described below using conic and quadratic potential forms [30],

$$U_{att}(q) = \begin{cases} \frac{1}{2}\zeta d^2, \mathrm{d} \leq d_{goal}^{*}, \\ d_{goal}^{*}\zeta \mathrm{d} - \frac{1}{2}\zeta (d_{goal}^{*})^2, d > d_{goal}^{*}. \end{cases} \tag{4.2}$$

with the gradient,

$$\nabla U_{att}(q) = \begin{cases} \zeta(q - q_{goal}), \mathrm{d} \leq d_{goal}^{*}, \\ d_{goal}^{*}\zeta, d > d_{goal}^{*}. \end{cases} \tag{4.3}$$

d is the distance from the robot to the goal, ζ is the attraction gain and d_{goal}^{*} is the threshold distance from the goal where the planner switches between conic and quadratic potentials.

As we know, the robot is attracted by the attractive force from the goal position. The absolute value attractive force is the gradient of the potential [30].

$$F_{att} = - \nabla U_{att}(q) \tag{4.4}$$

Substituting into the equation,

$$F_{att} = - \nabla U_{att}(q) = \begin{cases} \zeta(q_{goal} - q), \mathrm{d} \leq d_{goal}^{*}, \\ -d_{goal}^{*}\zeta, \quad d > d_{goal}^{*}. \end{cases} \tag{4.5}$$

where the d is distance from the robot to the goal, ζ is the attraction gain and d_{goal}^{*} is the threshold distance from the goal where the planner switches between conic and quadratic potentials.

The repulsive force keeps the robot away from the obstacles. In our work, the force should only satisfy one requirement- the closer the robot is to an obstacle, the stronger repulsive force should be so that the robot can avoid and detour the obstacles efficiently. Thus, the strength of the repulsive force is dependent on the distance D(q) from the closest obstacles to the robot. Similar to the attractive force, the common equations of the repulsive potential function are [30]

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^{*}})^2, & D(q) \leq Q^{*}, \\ 0, & D(q) > Q^{*}. \end{cases} \tag{4.6}$$

with the gradient,

$$\nabla U_{rep}(q) = \begin{cases} \eta\left(\frac{1}{Q^{*}} - \frac{1}{D(q)}\right)\frac{1}{D^2(q)} \vee D(q), & D(q) \leq Q^{*}, \\ 0, & D(q) > Q^{*}. \end{cases} \tag{4.7}$$

Here, $Q^{*} \in \mathrm{R}$ is the factor that allows the robot to ignore obstacles far away enough from it while η can be treated as a gain on the repulsive gradient.

Similar to the attractive force, the repulsive force can be written as [30],

$$F_{rep} = -\nabla U_{rep}(q) = \begin{cases} \eta\left(\frac{1}{D(q)} - \frac{1}{Q^*}\right)\frac{1}{D^2(q)} \vee D(q), & D(q) \le Q^*, \\ 0, & D(q) > Q^*. \end{cases} \quad (4.8)$$

Finally, the resultant force of APF which works on the robot is

$$F_{sum} = F_{att} + F_{rep} = -\nabla U_{att}(q) - \nabla U_{rep}(q) \quad (4.9)$$

4.2.2 Drawback of the APF –Local Minima

The most common problem of artificial potential field method is local minima that occur when the robot goes into an area where there is a zero resultant force. Figure4.3 shows the situation.

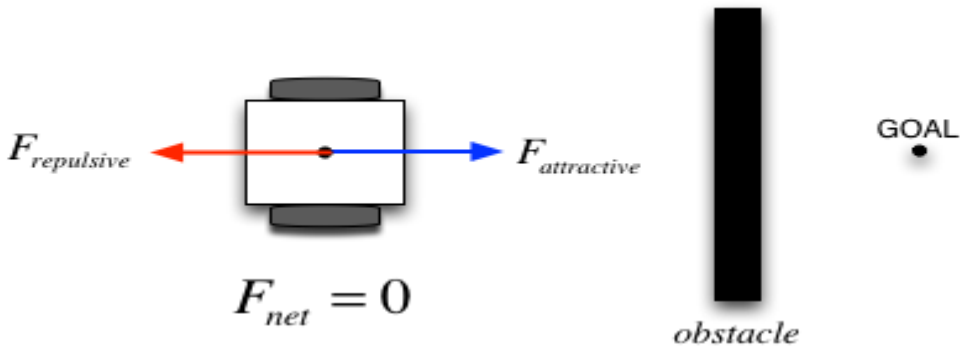


Figure 4.3 Local minima [28]

It means that the attractive and repulsive forces which focus on the robot have the

same absolute value and opposite direction.

$$F_{att} = -F_{rep} \tag{4.10}$$

$$F_{sum}=F_{att} + F_{rep} = 0. \tag{4.11}$$

The local minima problem happens in a lot of situations such as concave and narrow passage. As long as there is a chance that makes the direction of resultant repulsive force opposite to the attractive force, there is a possibility that the local minima happens. For example, in the narrow passage shown in Figure 4.4

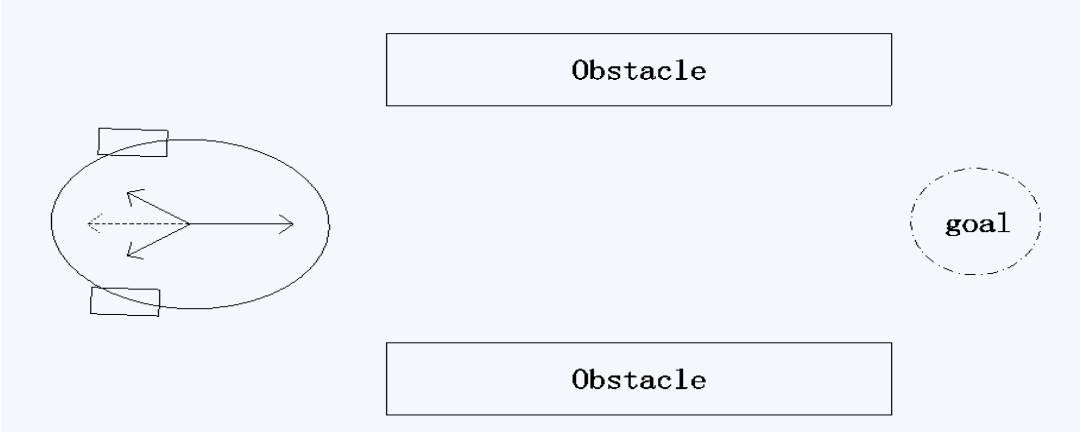


Figure 4.4 Local minima approaching a narrow passage

The robot is first attracted by the goal and when it arrives near the narrow passage, it is subject to two repulsive forces. When with the robot getting closer to the passage, the resultant force of the two repulsive forces is increasing gradually. Since the resultant force has the opposite direction of the attractive force, the robot will reach a position where the absolute value of resultant force is equal to the absolute value of

the attractive force so, as a result, the robot stops there and never reaches the goal position.

In our simulation, the local minima also happened. In the scenario as the figure shows below, by applying the artificial potential field method to the robot navigation controller, the robot was stuck there because the resultant repulsive forces equal to the attractive force. (This problem is solved by using velocity potential field method which will be presented later.)

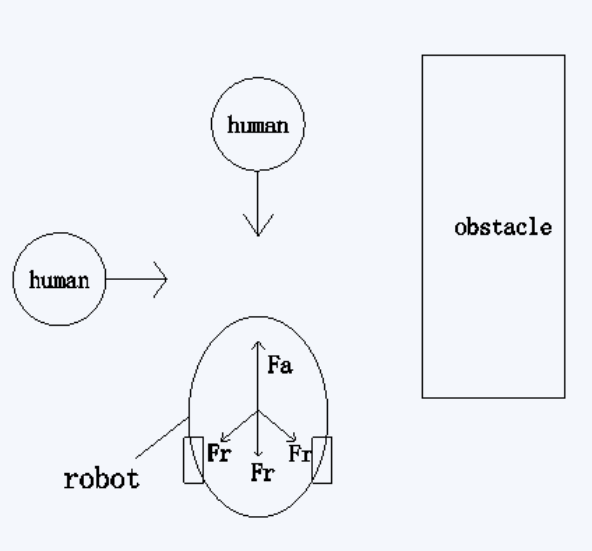


Figure 4.5 Local minima approaching the three directional dangerous situation

In our research, there is also a global minimum, which happens at the goal position shown in Figure 4.6.

Figure 4.6 Global minimum

In our research there are only two kinds of objects in the environment. One is obstacle, including human, and the other one is goal. The obstacles can only give the robot repulsive forces while the goals only attract the robot. Again, we can treat the goal as the negative charged electron while treat the human and obstacles as the positive electrons. Focused on the robot, if we set the attractive force between the goal and the robot has the negative value, the repulsive force between the obstacles and the robot should has the positive value since it has the opposite direction. In a coordinate axis we can see, no matter what kinds of the combination of the repulsive forces are, the resultant force of the repulsive forces will only has a positive value and the negative value only occurs at the goal position which is also the global minimum.

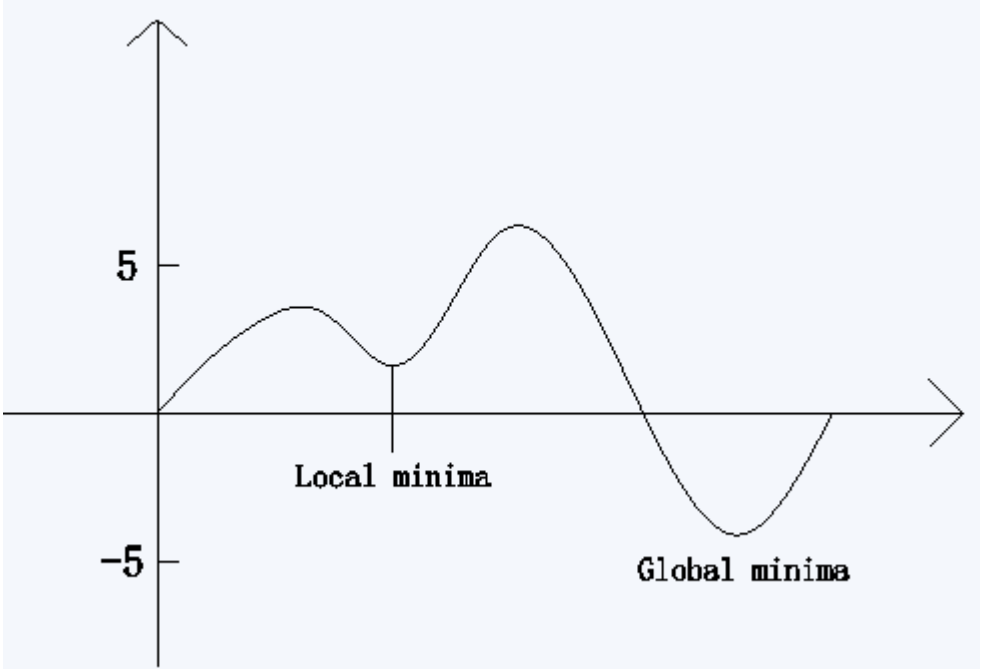


Figure 4.7 The position of the Local minima and the Global minima

Thus, we can see in our research, the global minimum occurs in the goal position.

4.2.3 The Velocity Potential Field Method based on the Artificial Potential Field

Derived differently from the artificial potential field method, the velocity potential field (VPF) method has the ability to guide the mobile robot more directly. Compared with APF, the difference of the VPF is that it uses the velocity potential while APF uses a potential whose gradient is force. Same as the APF, the potential function can be constructed as the sum of the attractive and repulsive potentials [30]. ,

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{4.1}$$

and

$$U_{att}(q) = \begin{cases} \frac{1}{2}\zeta d^2, \mathrm{d} \le d_{goal}^{*}, \\ d_{goal}^{*}\zeta \mathrm{d} - \frac{1}{2}\zeta (d_{goal}^{*})^2, d > d_{goal}^{*}. \end{cases} \quad (4.2)$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^{*}})^2, \ D(q) \le Q^{*}, \\ 0, \qquad\qquad D(q) > Q^{*}. \end{cases} \quad (4.6)$$

d is the distance from the robot to the goal, ζ is the attraction gain and d_{goal}^{*} is the threshold distance from the goal where the planner switches between conic and quadratic potentials.

Since the only difference here is that we use the velocity potential to replace the force potential in VPF, the attractive and repulsive potential gradients are no longer equal to the force but the velocity [30].

$$V_{att} = - \nabla U_{att}(q) = \begin{cases} \zeta(q_{goal} - q), \mathrm{d} \le d_{goal}^{*}, \\ -d_{goal}^{*}\zeta, \qquad d > d_{goal}^{*}. \end{cases} \quad (4.12)$$

$$V_{rep} = - \nabla U_{rep}(q)$$

$$= \begin{cases} \eta\left(\frac{1}{D(q)} - \frac{1}{Q^{*}}\right)\frac{1}{D^2(q)} \vee D(q), \ D(q) \le Q^{*}, \\ 0, \qquad\qquad\qquad D(q) > Q^{*}. \end{cases} \quad (4.13)$$

The parameters in the equations above have the same meaning of the equations 4.2 and 4.6.

Thus, the resultant velocity of VPF which works on the robot is

$$V_{sum}=V_{att}+V_{rep}=-\ \nabla U_{att}(q)-\nabla U_{rep}(q) \quad (4.14)$$

In next part, we will introduce the third velocity command, the rotation velocity, which will help the robot detour the obstacles.

4.3 Fluid Flow and the Velocity Potential Method

In our research, to overcome the problem of the local minima, we adopted a method inspired by fluid flow around the obstacles, which can create smooth paths for the mobile robot to detour around the obstacles. Figure 4.8 shows the streamlines of fluid flowing over an automobile in a wind tunnel [27][28].

Figure 4.8 The streamlines of fluid flowing over an automobile in a wind tunnel [28]

As we can see, the path of the wind flowing over the automobile is smooth and energy-saving. We want to design a navigation controller that leads the robot detour

the obstacle similar as the wind flowing. In incompressible flow, all the velocity vectors of the fluid point are tangent to the streamline. In the two-dimensional plane, the streamlines equations [27][28] are represented by the stream function ψ=ψ(x,y), which can be used to describe the velocity parameters of the fluid. Here, u and v represent the fluid velocity in x and y direction.

$$u = \frac{\delta\psi}{\delta y} \tag{4.15}$$

$$v = \frac{\delta\psi}{\delta x} \tag{4.16}$$

Furthermore, based on the stream function ψ, in cylindrical coordinates the fluid velocity components are [27][28],

$$V_r = \frac{1}{r}\frac{\delta\psi}{\delta\theta} \tag{4.17}$$

$$V_\theta = -\frac{\delta\psi}{\delta r} \tag{4.18}$$

Φ is a velocity potential function, which can be used to describe the flow of fluid with the stream function ψ. According to the equation below, we can use the velocity potential function which is a single scalar variable Φ=Φ(x,y,t) to calculate flow velocities for irrotational flow [27][28].

$$u = -\frac{\delta\Phi}{\delta x} \tag{4.19}$$

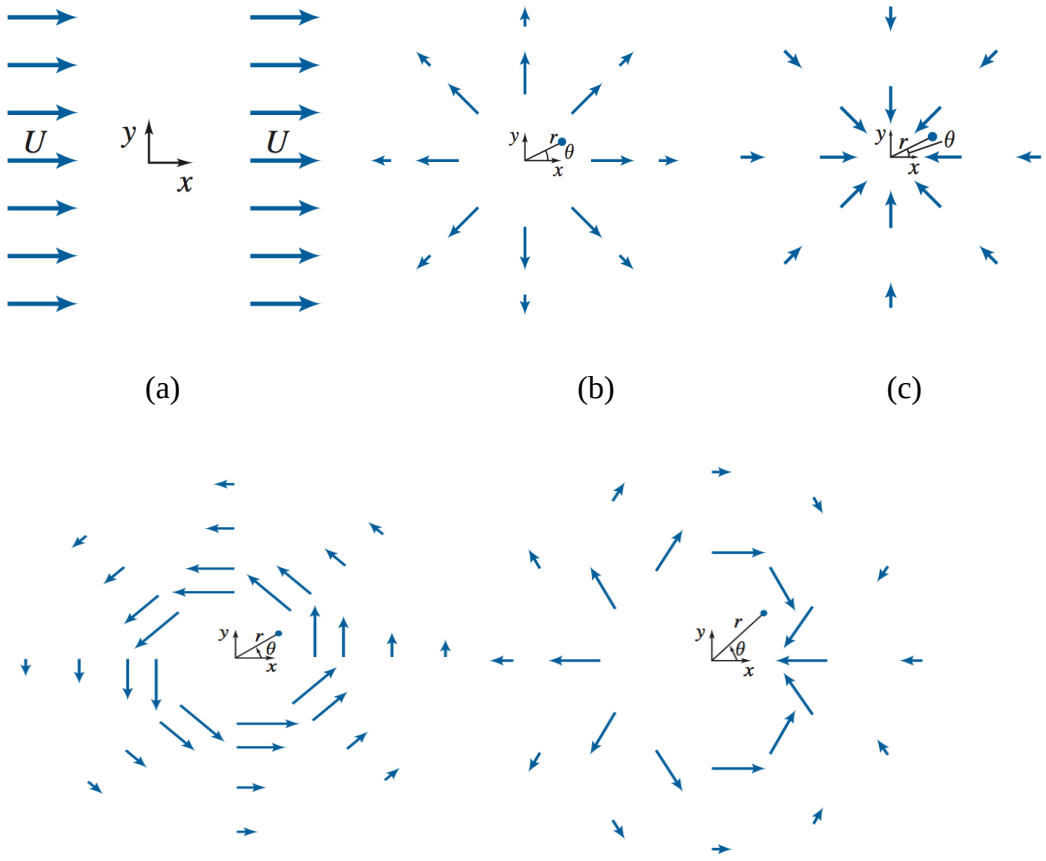
$$v = -\frac{\delta\Phi}{\delta y} \tag{4.20}$$

and based on the equations above, the cylindrical velocity values [27][28] are

$$V_r = \frac{1}{r}\frac{\delta\Phi}{\delta r} \tag{4.21}$$

$$V_\theta = -\frac{1}{r}\frac{\delta\Phi}{\delta\theta} \tag{4.22}$$

There are many types of elementary plane flows such as uniform flow, source flow, sink flow, irrotational vortex and doublet flow as the Figure 4.9 shows.



(a) (b) (c)

(d) (e)

Figure 4.9 The uniform flow (a), source flow (b), sink flow (c), irrotational vortex (d) and doublet flow (e)[27][28]

By combining the different kinds of the elementary flows, we can get the complex flows. The target that we want to achieve in this part is to help the robot detour the obstacles. There are lots of the complex flows. The suitable one in our situation is source and vortex flow. In this part, we will only show and list the useful flows' stream functions and velocity potential functions.

The source flow Figure 4.10 [27][28] shows the flow of fluid moving away from the origin.

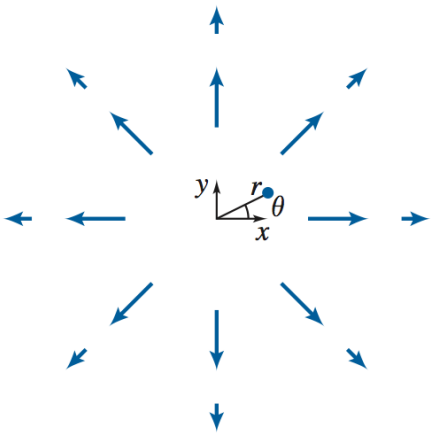


Figure 4.10 The source flow[27][28] (from origin)

$$\psi = \frac{q}{2\pi}\theta \tag{4.23}$$

$$\Phi = -\frac{q}{2\pi}\ln r \tag{4.24}$$

The irrotation vortex Figure 4.11[27][28] shows the flow flowing around the origin.

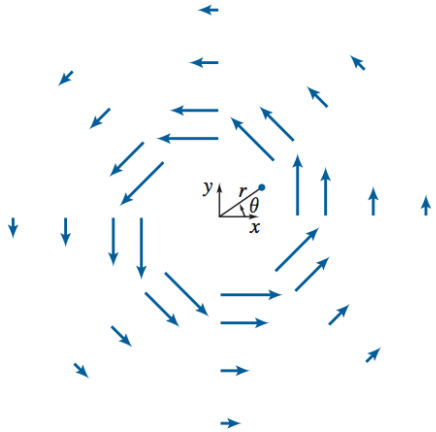


Figure 4.11 The irrotation vortex

The irrotation vortex [27][28] (counterclock wise, center at origin):

$$\psi = -\frac{K}{2\pi}\ln r \tag{4.25}$$

$$\Phi = -\frac{K}{2\pi}\theta \tag{4.26}$$

By combining the source and irritation flow together, we got the source and vortex flow which is showed in Figure 4.12 [27][28]

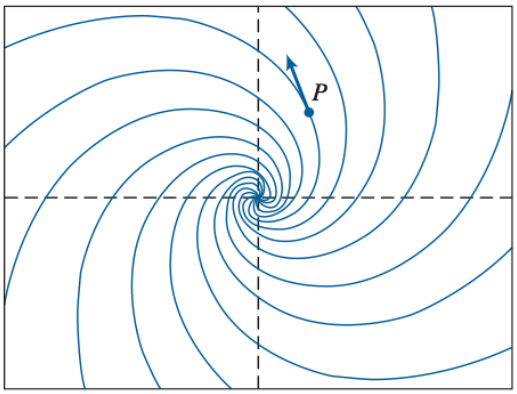


Figure 4.12 The source and vortex flow

The source and vortex flow [27][28] (spiral vortex)

$$\psi = \frac{q}{2\pi}\theta - \frac{K}{2\pi}\ln r \tag{4.27}$$

$$\Phi = -\frac{q}{2\pi}\ln r - \frac{K}{2\pi}\theta \tag{4.28}$$

4.4 Velocity Potential Flow Theory

In order to help the robot detour the obstacles and conquer the local minima, we used the spiral vortex potential flow function to direct the robot. Put simply, we added a tangent velocity commands to make the robot rotate when it meets the obstacle. The figure below shows the trajectory of the robot when it reached the obstacle. As we can see that, by applying the velocity potential flow theory, the path changed to avoid collision, along the flow lines of the spiral vortex.

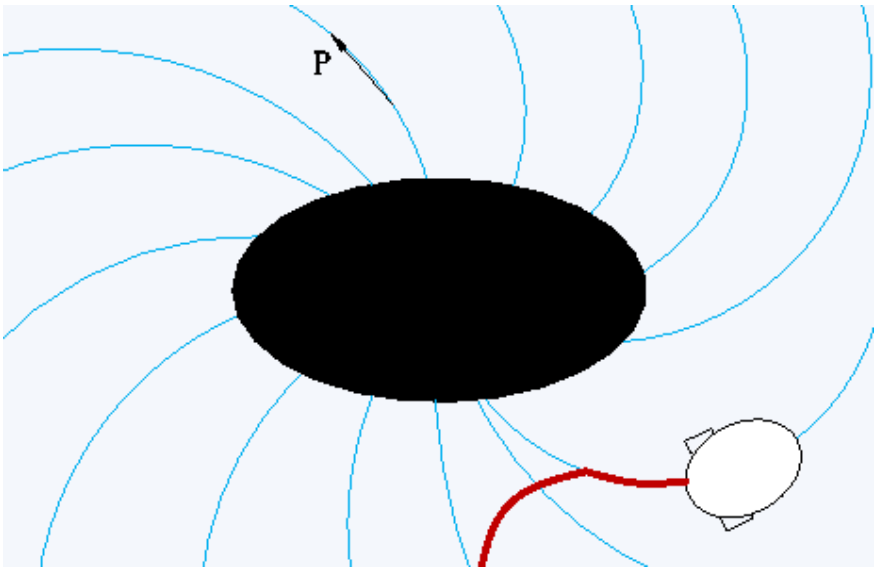


Figure 4.13 The trajectory of the robot when it reached the obstacle

From the cylindrical velocity equations of the spiral vortex, we can get the normal and

tangential velocities when the robot moving along the streamline of the spiral vortex.

The velocity potential function of the spiral vortex [28] is

$$\Phi = -\frac{q}{2\pi}\ln r - \frac{K}{2\pi}\theta \tag{4.29}$$

Substituting equation (4.27) and (4.28), we obtained the normal and tangential velocities [28]

$$V_{r1} = \frac{1}{r}\frac{\delta\Phi}{\delta r} \tag{4.30}$$

$$V_{r1} = -\frac{q}{2\pi}\left(\frac{1}{r}\right) \tag{4.31}$$

$$V_{r1} = -\frac{A}{r} \tag{4.32}$$

$$V_{\theta 1} = -\frac{1}{r}\frac{\delta\Phi}{\delta\theta} \tag{4.33}$$

$$V_{\theta 1} = -\frac{K}{2\pi}\left(\frac{1}{r}\right) \tag{4.34}$$

$$V_{\theta 1} = -\frac{B}{r} \tag{4.35}$$

where A and B are constants.

To make the robot avoid the obstacles, we gave it an opposite vortex so that it will have more chooses when it meets the obstacles. There are two directions (left and right) the robot can choose to turn based on its relative position to the obstacles. Figure below shows the opposite direction spiral vortex.

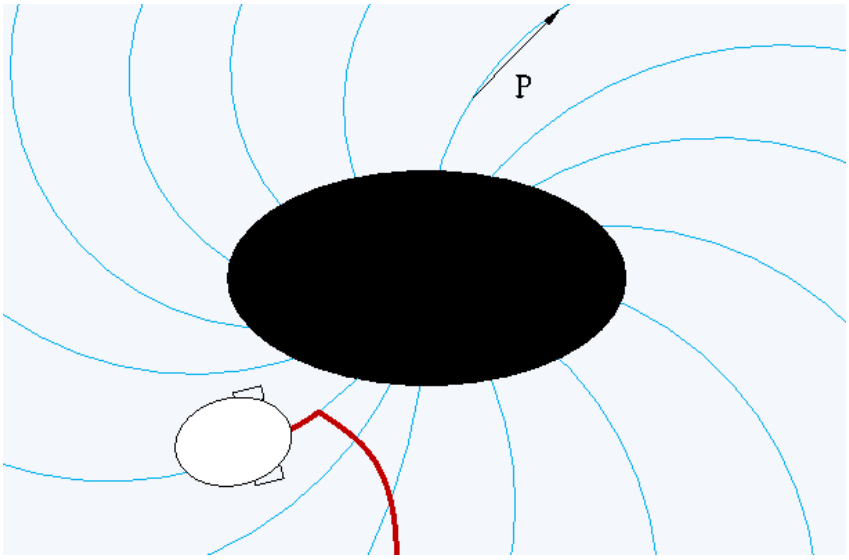


Figure 4.14 The opposite direction spiral vortex

The normal and tangential velocities of this spiral vortex [28] are

$$V_{r2} = V_{r1} = -\frac{A}{r} \tag{4.36}$$

$$V_{\theta 2} = -V_{\theta 1} = \frac{B}{r} \tag{4.37}$$

where A and B are the same constants as the first spiral vortex.

4.5 Sizes of the Four Zones based on Different Requirements

Next, we created four different areas around our robot. The smallest area is called the dangerous zone (A4).It is the area in close proximity to the obstacle. If the robot finds the obstacle in this zone, it is not able to avoid the obstacle but has to decelerate fully to eliminate or at least mitigate the damage which is caused by the collision with the obstacle. The small area is called the stationary zone (A3). The purpose to design this

zone is to protect the stationary obstacle. Once the robot detect there is stationary obstacle in this zone, it has to detour around the obstacle to prevent the collision happen. Similarly, the larger (A2) and the largest areas (A1) are designed to help the robot avoid the moving obstacle and human, respectively. The four zones are showed in the Figure 4.15.

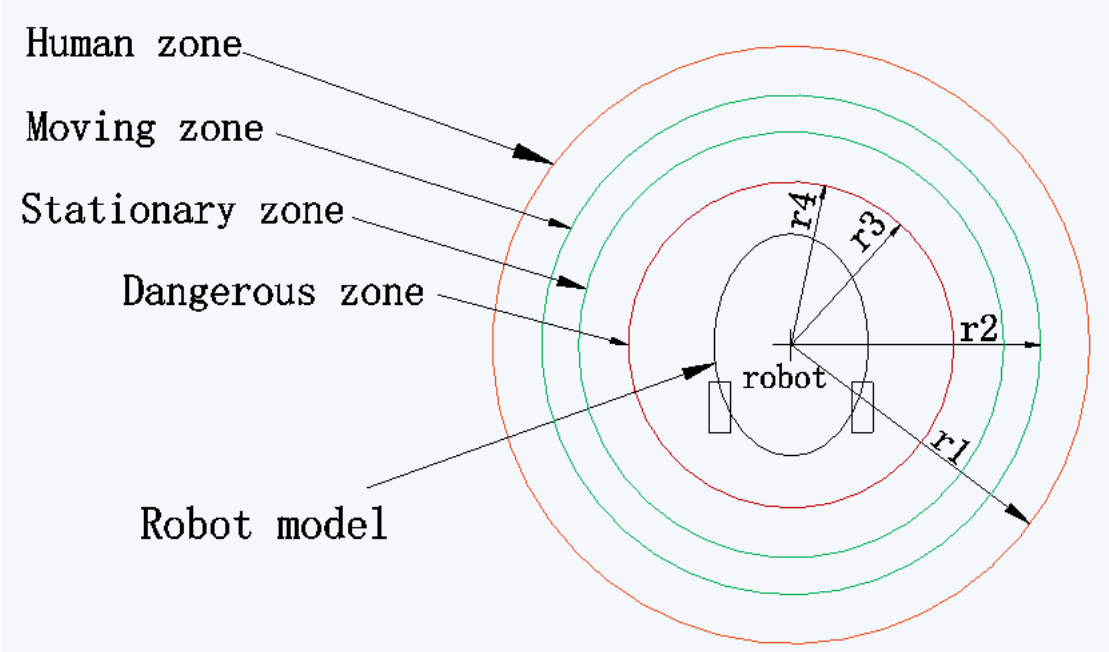


Figure 4.15 The four zones

The sizes of the four areas are related to the robot and human shapes, velocity limits and acceleration limits. In this part, we will introduce the method to calculate the sizes of the four areas and the shapes of the four areas as well.

4.5.1 The Smallest Area-Dangerous Zone

The dangerous zone is the area the closest to the obstacle. Once the obstacle enters

into this zone, our robot cannot continue moving while avoiding the obstacle and has to stop immediately. Since various sensors are installed on the robot, as the Figure 4.16 shows, the distance they measure is the distance between robot's body and the obstacle.

Figure 4.16 Sensors on the robot

Besides, the infrared sensors (IR Range) are installed around the robot, so the sensor detects the area that can be treated as a circle around the robot. As a result, to simplify the design of the dangerous zone, the shape of the zone will be created as a circle as well. The size of the dangerous zone is highly related to the stopping distance of our robot [29]. Here, we set r_4 as the stopping distance, which ensures the robot can stop before the impact happen even for the worst case; i.e. the robot moving directly to the obstacle.

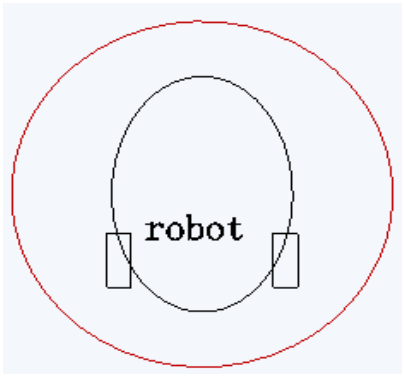


Figure 4.17 The dangerous zone

$$r_4 > \overline{\delta}_{r1} \tag{4.38}$$

where $\overline{\delta}_{r1}$ is the distance required to stop the robot [29].

$$\overline{\delta}_{r1} = \iint_{t_a} a_{mr} dt = \frac{1}{2} \bar{a}_{mr} {t_a}^2 \tag{4.39}$$

where a_{mr} is the magnitude of the robot's deceleration $\bar{a}_{mr}$ is robot's constant deceleration and t_a is the time required to stop the robot with an initial velocity $\overline{V}_{mr}$ $\overline{V}_{mr}$, the initial velocity of our mobile robot for constant deceleration to zero velocity, results

$$\overline{V}_{mr} = \bar{a}_{mr} * t_a \tag{4.40}$$

4.5.2 The Small Area-Stationary Zone

Next, we will calculate the size of the area which provides the robot enough room to prevent the collisions. The stationary zone is designed for a stationary obstacle. Once the robot detects that there is an obstacle in its stationary zone, it will detour the

obstacle without halting. That is to say, during the avoidance, the robot will not let the obstacle intrude its dangerous zone.

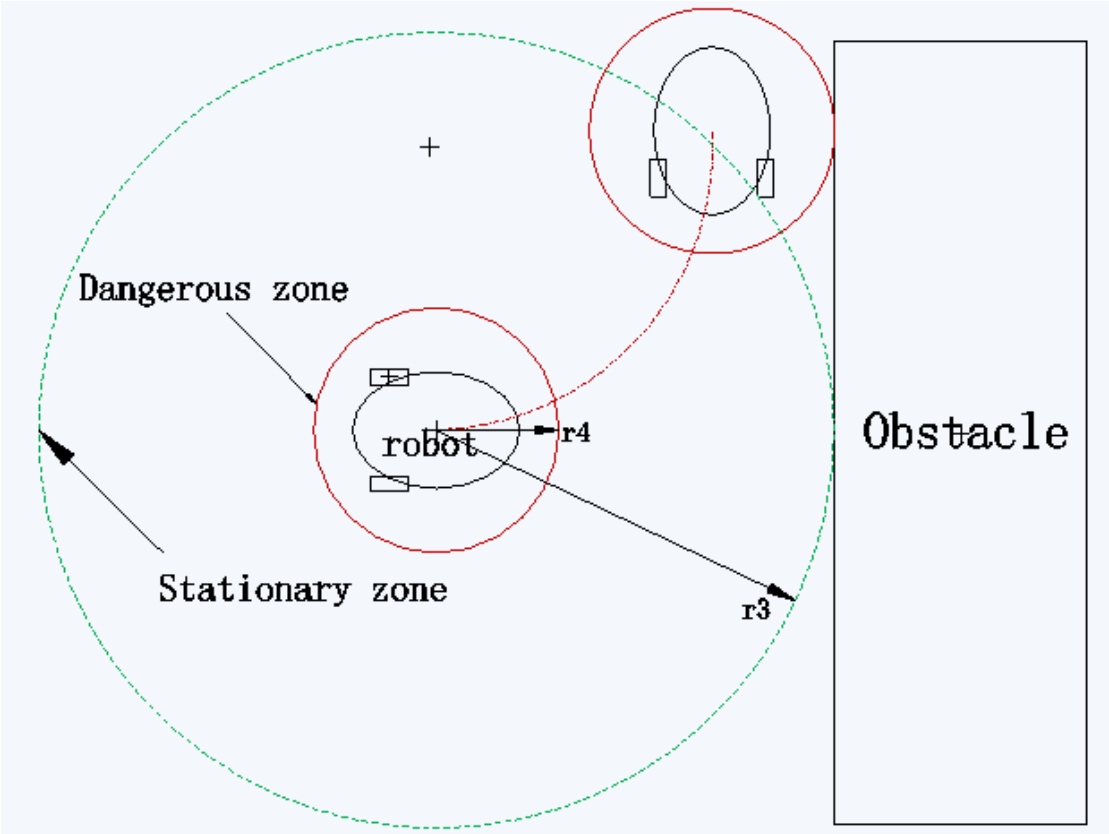


Figure 4.18 The stationary zone

Figure 4.18 shows the worst situation. In the X-direction, the robot moves directly toward the obstacle and it has available the distance$\overline{\delta}_{mrx3}$ to change its moving direction. $\overline{\delta}_{mrx3}$is the distance that the robot can use to prevent the obstacle enter into its dangerous zone. Here, with the consideration of safety, we calculated the size of the stationary zone based on the worst situation

$$\overline{\delta}_{mrx3} < r_3 - r_4 \tag{4.41}$$

where r_3 and r_4 are the radius of the dangerous zone and stationary zone, respectively.

A non-holonomic mobile robot cannot move instantaneously in Y-direction, as the figure 4.17 shows, it can only turn on a tangent to the circle of maximum curvature permitted by max deceleration from initial position and maximum turning angle to detour the obstacles. In other words, in our calculation, the robot has to accelerate in Y-direction and decelerate in X-direction at the same time. Due to this it is impossible to obtain the accurate velocity of its velocity in each moment. We first assume that the robot moves with maximum acceleration magnitude in Y-direction.

In Y-direction,

$$\bar{a}_{mry} = \bar{a}_{mr} \tag{4.42}$$

$$\Delta t_3 \times \bar{a}_{mry} = \overline{V}_{mr} \tag{4.43}$$

$$\Delta t_3 = \frac{\overline{V}_{mr}}{\bar{a}_{mr}} \tag{4.44}$$

where $\bar{a}_{mry}$ is the robot's deceleration magnitude in Y-direction, $\bar{a}_{mr}$ is the maximum acceleration magnitude of the robot, $\overline{V}_{mr}$ is the normal velocity of the robot and Δt_3 is the time that robot needs to accelerate from zero velocity to $\overline{V}_{mr}$ in Y-direction.

Actually, the acceleration magnitude will be less than the maximum acceleration magnitude since the robot has to decelerate in X-direction at the same time. Thus, the actual Δt_3 will be longer than the computed values. To make it simple and

compensate this error, we can assume the worst case that $\bar{a}_{mrx} = 0$, where $\bar{a}_{mrx}$ is the decelerating magnitude of the robot in X-direction [29]. That is to say the robot will not decelerate in X-direction during the collision avoidance. So in X-direction,

$$\overline{\delta}_{mrx3} = \overline{V}_{mr} \times \Delta t_3 \tag{4.45}$$

$$\overline{V}_{mr} \times \Delta t_3 < r_3 - r_4 \tag{4.46}$$

$$r_3 >= \overline{V}_{mr} \times \Delta t_3 + r_4 \tag{4.47}$$

where $\overline{a}_{mry}$ is the robot's deceleration magnitude in Y-direction, $\bar{a}_{mr}$ is the maximum acceleration magnitude of the robot, $\overline{V}_{mr}$ is the normal velocity of the robot and Δt_3 is the time that robot needs to accelerate from zero velocity to $\overline{V}_{mr}$ in Y-direction.$r_4$is the radius of the dangerous zone.

4.5.3 The Large Area-Moving Zone

Sometimes, the robot will face moving obstacles such as vehicles and other mobile robots. The moving zone is designed for our robot to prevent the collisions with such kind of moving obstacles. Since the movements of other moving obstacles are unknown and also unpredictable by our robot, similar to the way we calculated the size of the stationary zone, we calculated the size of the moving zone based on the worst situation. Here, the worst case is collinear condition with the obstacle and the robot approaching each other at their maximum speeds in X-direction, as shown in Figure 4.19.

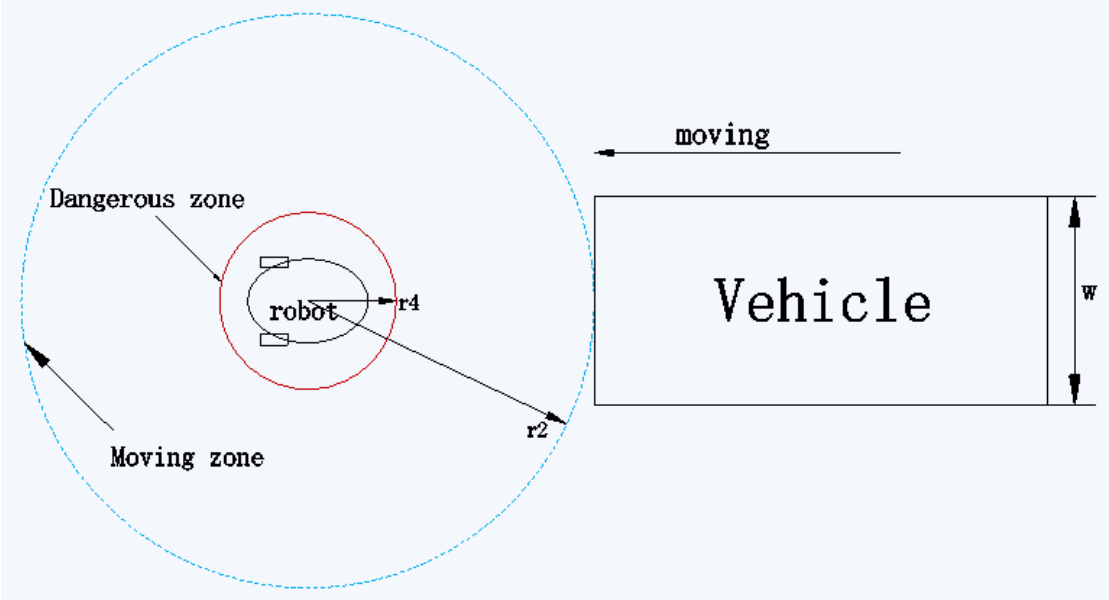


Figure 4.19 The moving zone

In this thesis, we set that the largest moving obstacle is the vehicle with the width w, shown in the figure above. The r_2 isthe radius of the moving zone and has to be set appropriately. As the figures below show, if the r_2 is large, the distance between robot and moving obstacle during the avoidance will increase, and the time to reach the goal will also increase. If r_2 is too small, the robot, depending on the situation, might not have enough time and space to complete the avoidance.

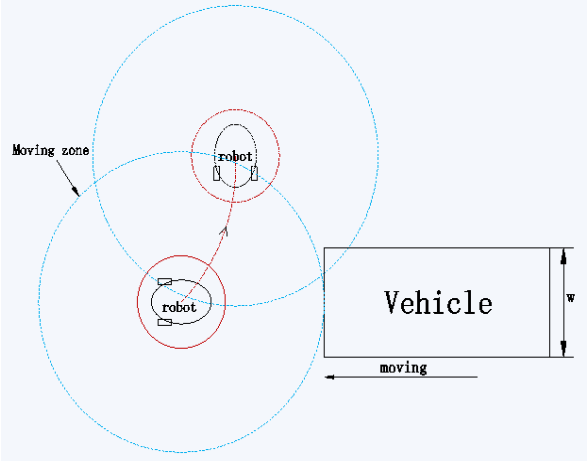


Figure 4.20 The robot avoidance path when the radius of the moving zone is large

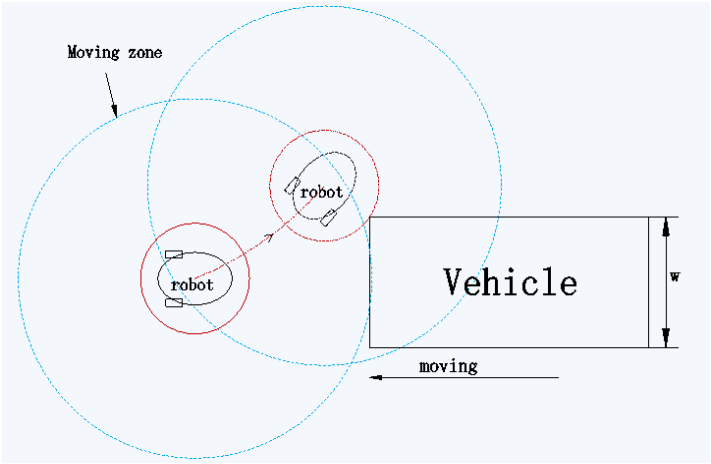


Figure 4.21 The robot avoidance path when the radius of the moving zone is small

Thus, based on the worst situation, we can find the critical value of r2. As the Figure 4.22 shows, once the vehicle touches the boundary of the dangerous zone of the robot, the robot has already traveled enough distance in Y-direction so that the vehicle will not enter the dangerous zone.

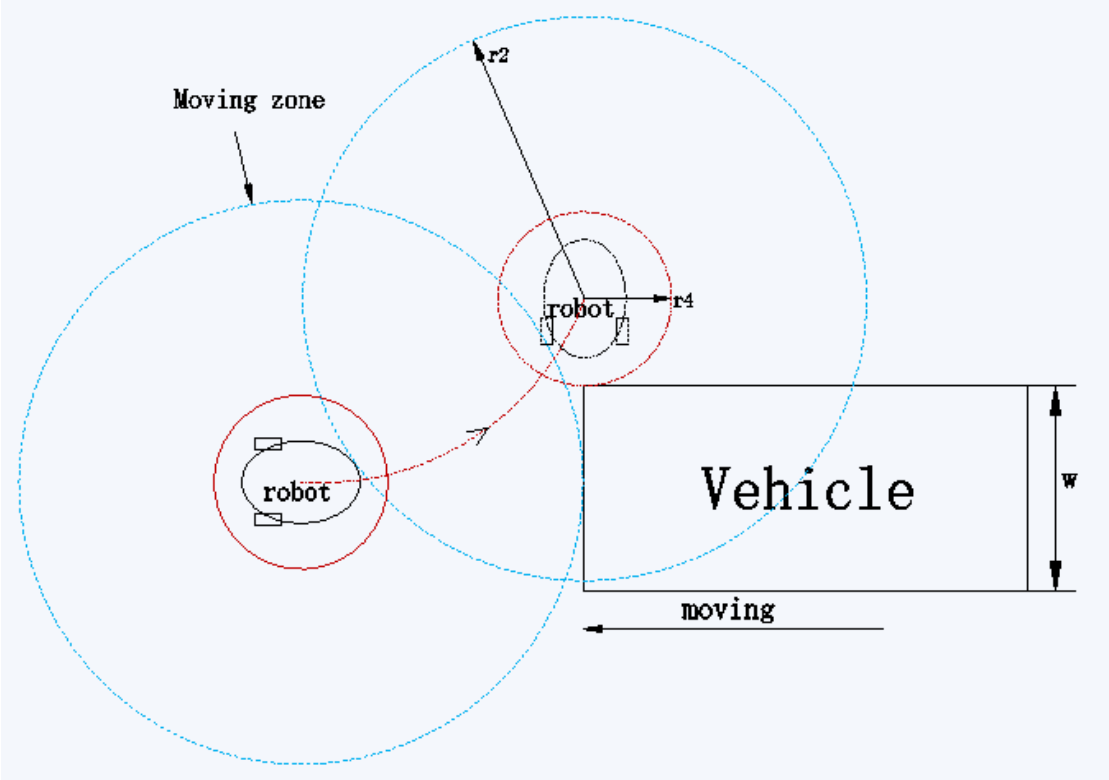


Figure 4.22 The robot avoidance path when the radius of the moving zone is appropriate

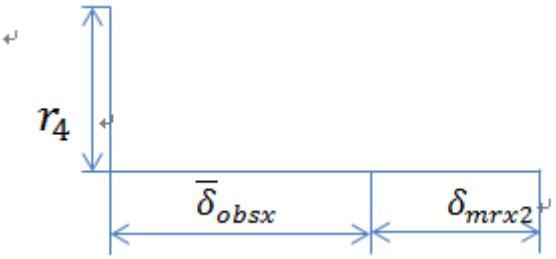


Figure 4.23 The moving zone's radius

As Figure 4.23 shows, the radius of r_2 is then constrained to be,

$$r_2 > \overline{\delta}_{obsx} + \overline{\delta}_{mrx2} \tag{4.48}$$

where $\overline{\delta}_{obsx}$ and $\overline{\delta}_{mrx2}$ are the moving distance of the moving obstacle and our robot

in X-direction, respectively. Since the speed of moving obstacle is assumed steady,

$$\overline{\delta}_{obsx} = \overline{V}_{obs} \times \Delta t_2 \tag{4.49}$$

where $\overline{V}_{obs}$ is the velocity of obstacle and Δt_2 is the total time of the whole avoidance.

Similarly to the way we calculated the size of stationary zone, we assume that the robot tries its best to accelerate in the Y-direction [29]. As we described above, the distance which the robot traveled on the Y-direction should satisfy

$$\overline{V}_{mr} \times (\Delta t_2 - t_a) + \frac{1}{2}\bar{a}_{mr}{t_a}^2 = \delta_{mry} \geq r_4 + \frac{w}{2} \tag{4.50}$$

where $\overline{V}_{mr}$ is the maximum velocity of our robot, t_a is the time that robot needs to accelerate from zero velocity to $\overline{V}_{mr}$ in Y-direction, δ_{mry} is the distance that the robot moving in Y-direction and r_4 is the radius of the dangerous zone.

We can calculate the total time of the avoidance,

$$\Delta t_2 \geq \frac{r_4 + \frac{w}{2} - \frac{1}{2}\bar{a}_{mr}{t_a}^2}{\overline{V}_{mr}} + t_a \tag{4.51}$$

Since the robot must also decelerate in the X-direction, it is not able to fully accelerate in Y-direction, the actual Δt_2 should be longer than the computed valued. As a tradeoff like before, we assume in x direction,

$$a_{mrx} \approx 0, \tag{4.52}$$

wherea_{mrx} is the acceleration magnitude of the robot in X-direction.

Thus,

$$\overline{\delta}_{mrx} = \overline{V}_{mr} \times \Delta t_2 = \overline{\delta}_{mrx2} \tag{4.53}$$

And finally,

$$r_2 \geq \overline{\delta}_{obsx} + \overline{\delta}_{mrx2} = \overline{V}_{obs} \times \Delta t_2 + \overline{V}_{mr} \times \Delta t_2 \tag{4.54}$$

where r_2 is the radius of the stationary zone.

4.5.4 The Largest Area-Human Zone

In this part, we will introduce the human zone of the robot vicinity. The human zone is created for our robot to handle the environment with human. Before the introduction, we assume the following are true:

1) The human does not chase our robot.
2) The human motion is restricted to normal walking with a velocity magnitude

$$\overline{V}_h < 1[\frac{m}{s}]. \tag{4.55}$$

3) The human future motion is random and unpredictable.

Besides, we modeled human body as a projected shape in our simulation. The projected shape is highly related to the pose of the human which is changeable. To

simplify the simulation, we modeled a human body as an enclosing cylinder which is called human active area so that the different poses of the human can be ignored while still maintaining safety [29]. As the Figure 4.24 shows, since the average step length of a human is around 0.8 m (Martin and Marshi 1992 [31]), one half of this value will be used as the radius of the human active area, i.e.$\rho_h = 0.4$.

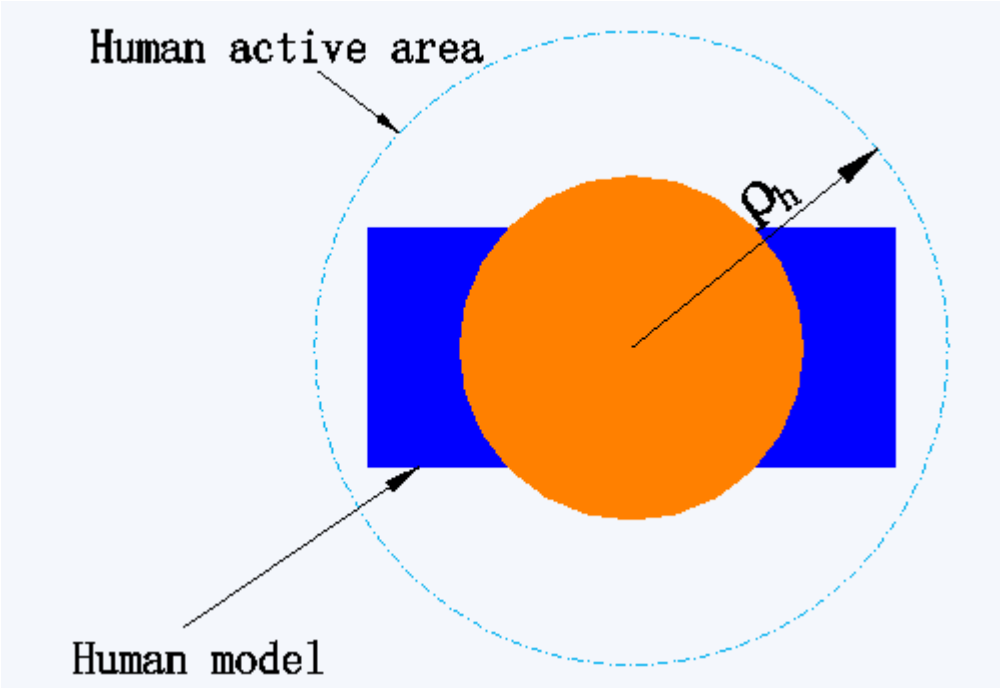


Figure 4.24 The human model

The human zone is built by the human sensors on the robot. Unlike the infrared sensors, which are installed around the robot, there are only two human sensors which are installed in front of the robot, so the human sensors detect area which is a sector as the figure shows. If the human approaches our robot, the human sensor detects the area and the human body will be treated as a moving obstacle and suitable for the moving zone. If a human approaches the robot in the human zone, the human sensor

will first detect him/her and tell the robot to avoid.

Since the human future motion is unpredictable, he/she might be stationary at this moment while starting walking at next moment; under the consideration of safety, the size of the human zone is calculated based on the worst situation which is the collinear condition with the human and the robot approaching each other at their maximum speeds in X-direction, as shown in Figure 4.25.

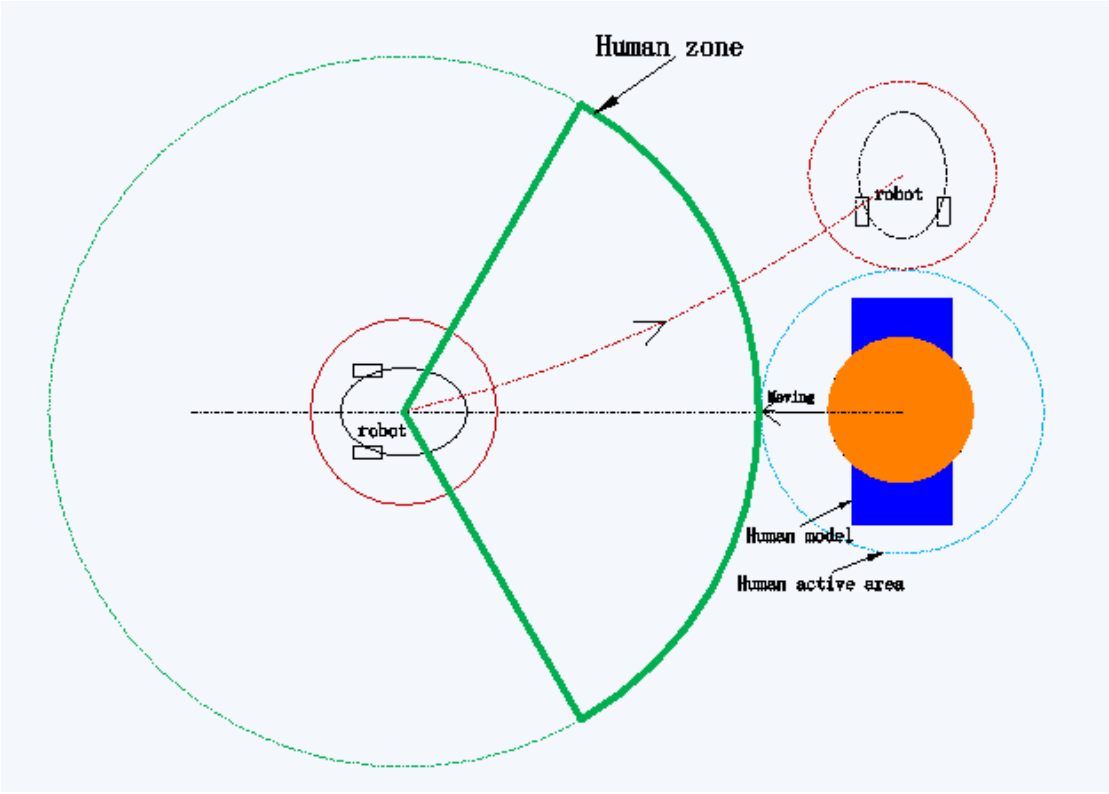


Figure 4.25 The human zone

Same situation occurs when avoiding the moving obstacle when the robot needs to change its moving direction and the displacement in the Y-direction must be larger than the radius of human active area [29]:

$$\rho_h = 0.4 \tag{4.56}$$

$$\overline{\delta}_{mry} = \overline{V}_{mr} \times (\Delta t_1 - t_a) + \iint_{t_a} a_{mry} dt > r_4 + \rho_h \tag{4.57}$$

$$\overline{V}_{mr} \times (\Delta t_1 - t_a) + \frac{1}{2} \bar{a}_{mry} {t_a}^2 = \delta_{mry} > r_4 + \rho_h \tag{4.58}$$

where $\overline{V}_{mr}$ is the maximum velocity of the robot, t_a is the time that robot needs to accelerate from zero velocity to $\overline{V}_{mr}$ in Y-direction, δ_{mry} is the distance that the robot is moving in Y-direction, r_4 is the radius of the dangerous zone, $a_{mry}$is the robot acceleration magnitude in Y-direction, Δt_1 is the total time of the avoidance and ρ_h is the radius of human active area.

Like the way we calculated the size of moving zone, we assume $a_{mry} = \bar{a}_{mr}$.
Thus, we obtain

$$\Delta t_1 > \frac{r_4 + \rho_h - \frac{1}{2}\bar{a}_{mr}{t_a}^2}{\overline{V}_{mr}} + t_a \tag{4.59}$$

Finally, the radius of the human zone r_1 should satisfy,

$$r_1 > \overline{\delta}_{hx} + \overline{\delta}_{mrx1} \tag{4.60}$$

where $\overline{\delta}_{hx}$ and $\overline{\delta}_{mrx1}$ is the human and robot moving distance in X-direction, respectively.

$$\overline{V}_{hx} = \overline{V}_h \tag{4.61}$$

$$\overline{\delta}_{hx} = P_{hx} - P_{hx0} = \overline{V}_{hx} \times \Delta t_1 \tag{4.62}$$

where $\overline{V}_h$ is the maximum human walking velocity and$\overline{V}_{hx}$ is the human walking

velocity in X-direction which is also less than 1m/s.

The robot moving distance in X-direction is

$$\overline{\delta}_{mrx1} = P_{mrx} - P_{mrx0} = \overline{V}_{mr} \times (\Delta t_1 - t_a) + \frac{1}{2}\bar{a}_{mrx}{t_a}^2 \quad (4.63)$$

The meanings of all the parameters are same as the definition of equations (4.58) and (4.60).

Similar to the way we calculated the robot moving distance in X-direction, we assume that $\overline{a}_{mrx} = 0$, so as a result,

$$\overline{\delta}_{mrx1} = \overline{V}_{mr} \times \Delta t_1 \quad (4.64)$$

where $\overline{\delta}_{mrx1}$ is the robot moving distance in X-direction. The meanings of other parameters are same as the definition of equations (4.58).

In our experiment, $\overline{V}_{mr} = \overline{V}_{obs} = 5cm/s$, $\bar{a}_{mr} = 5cm/s^2$, w = 20cm. $\overline{V}_h = 10cm/s$. So based on these data, we finally calculated the radii of the four different regions-$r_4 = 2.5cm, r_3 = 7.5cm, r_2 = 30cm, r_1 = 135cm$.

4.6 Sensor Fusion

4.6.1 Why Sensor Fusion is Necessary

Since human is the first priority, it is necessary to combine the information collected by different kinds of sensors to identify human with a higher probability.

In our research, the most important sensor which is used to detect humans is the Passive Infrared Sensor, which is also called human sensor. As we described before, the principle of PIR sensor is “sensing” the infrared and heat emitted by humans. However, the human sensor is not 100% reliable. By performing hundreds of experiments, there are at most 10 times of each 100 experiments that the human sensor does not work [11]. Besides, due to the complex environment in our laboratory, the suspicious human which is detected by human sensor might be hot wind created by heater or some other heat source. Thus, we bring in other kinds of sensors, on particular the ultrasonic range sensor. It emits high frequency sound waves to the objects and then receives them to determine how far away they are. By using the information which is get from the ultrasonic range sensor, the reliability of the robot's human detection increases to an acceptable level. The Figure 4.26 shows the flow chart of our sensor fusion process.

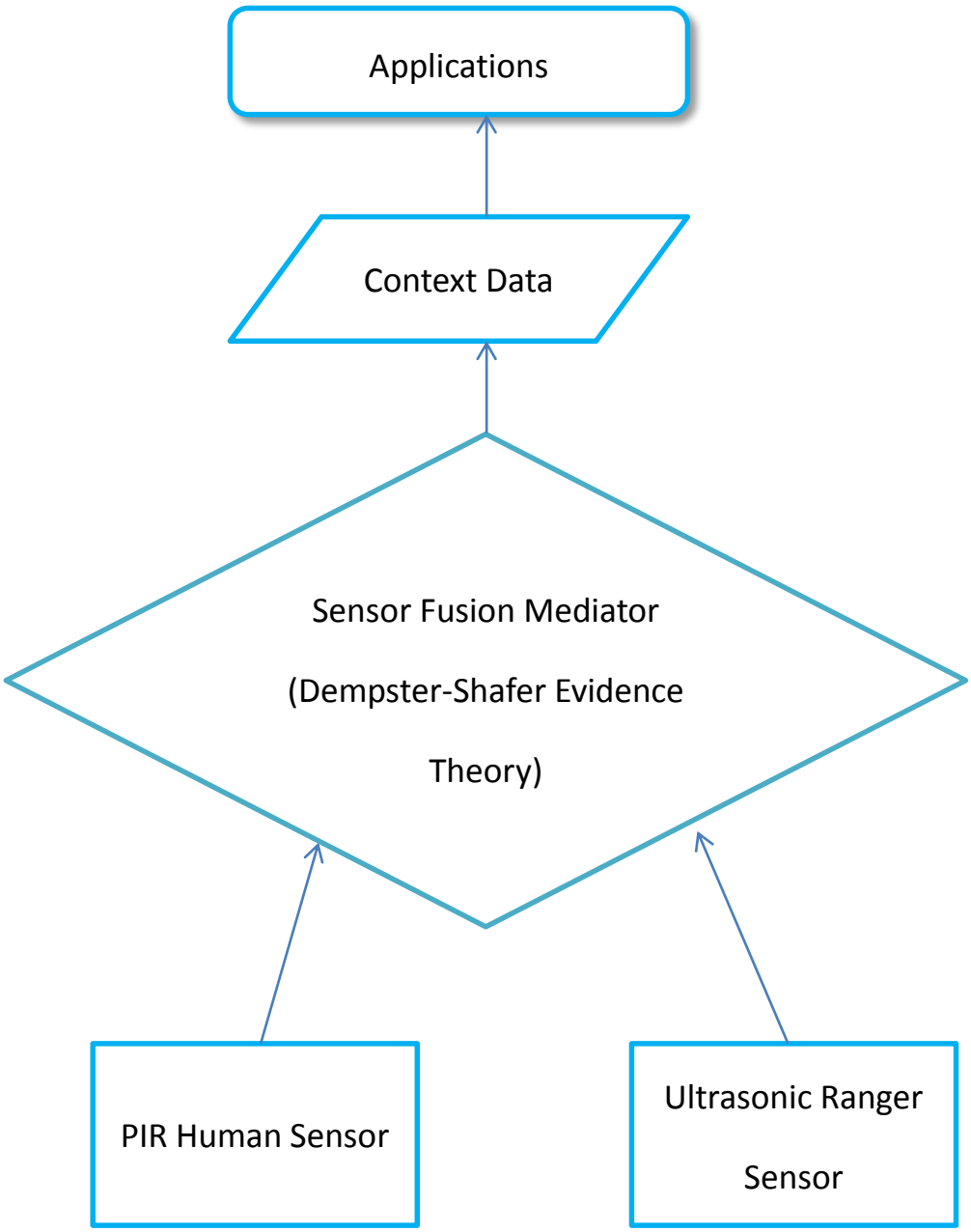


Figure 4.26 The flow chart of the sensor fusion process

4.6.2 The Method Used for Sensor Fusion

There are many methods to support the calculation of sensor data fusion such as Bayesian Inference, Dempster-Shafer Evidential Theory, Artificial Neural Networks and Voting Logic Fusion [23]. In our research, the Demster-Shafer evidential theory

(D-S theory) is chosen to support the probability calculation. As a generalization of the Bayesian Inference theory of subjective probability, D-S theory is useful when the sensors contributing information cannot associate a 100-percent probability of certainty to their output decisions. The algorithm captures and combines whatever certainty exists in the object-discrimination capability of the sensors. Knowledge from multiple sensors about events (called propositions) is combined using D-S theory to find the intersection or conjunction of the propositions and their associated probabilities [23].

In D-S theory, θ is a finite set which is called the frame of discernment. All the possible mutually exclusive context events including the empty event of the same kind are enumerated in the frame of discernment θ [11]. For example, if we know people who have two garages (Garage A and Garage B) and one car, and we want to figure out where the car is, then the "frame of discernment" about the position of this car is:

θ={Garage A, Garage B, {Garage A, Garage B}, outside}

meaning that the car is in Garage A, Garage B, Garage A or Garage B and not in the garage, which include all the possible situations. A function m is called a basic probability assignment (BPA) and it satisfies

$$m(\Phi)=0$$

and

$$\sum_{A_i \in \theta} = 1$$

where A_i is a subset of θ such as Garage A and Garage B above.The numerical value of m(A_i) is named A_i's basic probability number (BPN) and A_i is called as a focal element of m. Here, BPN is also indicated by a "confidence interval".

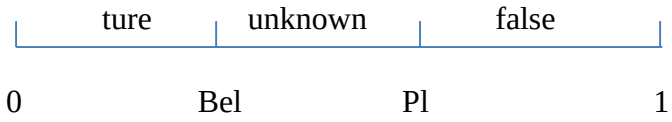
[Belief(A), Plausibility(A)]

The lower bound of the confidence interval is the belief confidence, which accounts for all evidence $E_k$that supports the given proposition "A":

$$\text{Belief(A)} = \sum_{E_k \in A} m(E_k)$$

The upper bound of the confidence interval is the plausibility confidence, which accounts for all the observations that do not rule out the given proposition:

$$\text{Plausibility(A)} = 1 - \sum_{E_k \cap A} m(E_k)$$



The D-S theory is used to combine two focal elements BPAs. Here is its equation

$$\text{m}(C_k) = \frac{\sum_{A_i \cap B_j = C_k; C_k \neq 0} m(A_i) m(B_i)}{1 - \sum_{A_i \cap B_j = 0} m(A_i) m(B_i)} C_k \in \theta \qquad (4.65)$$

where A_i and B_j are the focal elements of m_A and m_B, respectively. $m_A$and m_B are the BPAs which are combined while C_k are the focal elements of the combined BPA.

In our research, we designed the frame of discernments θ as a set of cells in the occupancy grids as Figure 4.27 shows. The eight grids are able to cover the human sensors' detect area.

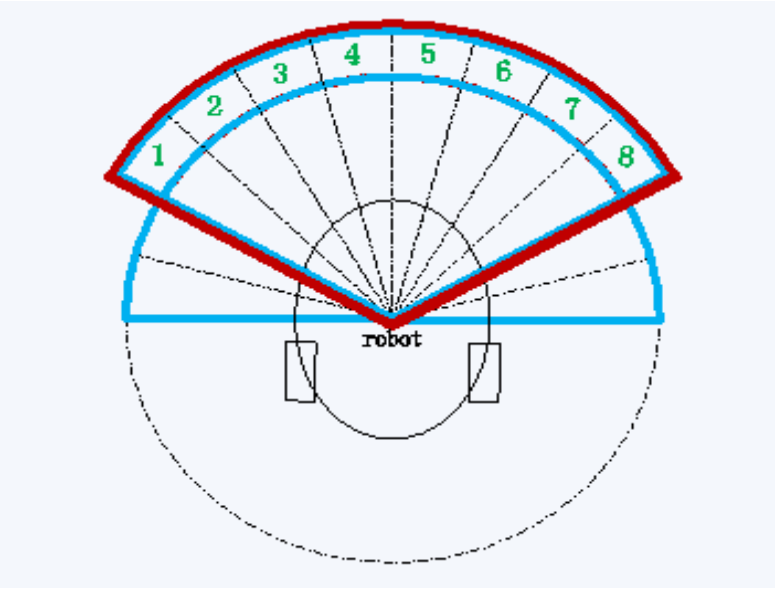


Figure 4.27 The human sensors' detect area (mixed color area)

Both ultrasonic sensors and PIR sensors keep watching the mixed color area and every sensor will contribute its observation by assigning its BPA over its own frame of discernments.

For example, in our project, the human sensor's reliability is 90%. When the output of the human sensor is high, we can draw a conclusion that there are 90% possibilities that the human is inside of the sensor's coverage area. For the rest 10%, there are two possibilities. One is that the human is outside the sensor's coverage area and the sensor is wrong so it gives a high output. The other is that the human is inside the sensor's coverage area and the sensor gives the right signal. Due to it is impossible for

the robot to identify which possibility is the real situation for the rest 10% possibility. We can only assume the possibility that human is inside/outside the sensor's coverage area is less than 10%.

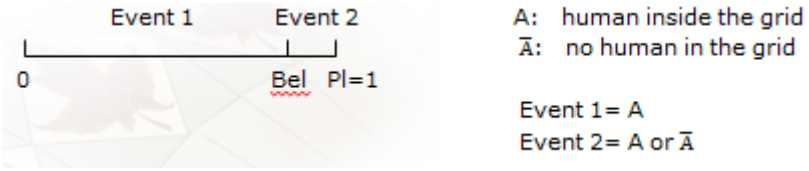


Figure 4.28 The possibility assignment of human inside the grid

In Figure 4.28, A represents human is inside the detected grid while $\bar{A}$ represents human is outside the detected grids. Event 1 represents the possibility (90%) that human is inside the grid, which is the belief confidence of the D-S theory. Event 2 represents the possibility that human is inside or outside the grid, which is the plausibility confidence of the D-S theory since the robot cannot identify the real situation (the human is inside or outside the grid) of the Event 2. The human sensor is not 100% reliable, so it cannot achieve a conclusion that human is outside the grid no matter under what kinds of situation (Even the sensor's output is low it is still possible that human is inside the grid). So in our project, the upper bound of the confidence interval (the plausibility confidence) equals to 1.Similar to the example above, when the human sensor output is low, it is 90% possibility that human is outside of the sensor's coverage area. For the rest 10%, the robot cannot judge whether the human is inside or outside the gird.

In our project, each sensor can assign two different BPAs based on its output.

Table 4.1 BPAs for sensor 1

Sensor Reading	Human Inside the Bar Area	Human Outside the Bar Area	Human Anywhere
High	$m_1(A)$	$m_1(A/\bar{A})$	$m_1(\theta)$
Low	$m_1(A/\bar{A})$	$m_1(\bar{A}) = m_1(A)$	

where *A* represents human is inside the detected grid while $\bar{A}$ represents the event that human is outside the detected grids. $A/\bar{A}$represents human is inside or outside the grid.

Since the sensor fusion method is used to help the robot calculate the possibility that human is inside its detection area, the system will only give the possibility value that human is inside the detection area no matter if the output of the sensor is high or low. As Table 1 shows, for sensor 1, when the sensor output is high, the possibility that human is inside the Bar area is $m_1(A)$ while when the sensor output is low, the possibility that human is inside the Bar area is less than $m_1(A/\bar{A})$.

Briefly, our sensor fusion process can be described into 7 steps.

1. Read the ultrasonic sensor outputs looking for the suspicious human. (assign the BPAs to each grid according to Table 3)
2. Read the human sensors outputs.(assign the BPAs to each grid according to Table 2)
3. Combine the final results of both kinds of sensors using D-S theory after one sample time.

4. After that, the system calculates a probability (the combined BPAs) that quantifies how suspicious the grid is to be occupied by a human.
5. Compare the probability value with the setting value and makes the decision whether there is a human in the grid and then makes a corresponding response based on different results.
6. Initialize the BPA of the occupied grid to zero.
7. Repeat from step one again until the mission is completed.

4.6.3 Examples of Sensor Fusion

Since the calculations for sensor fusion are independent of our final experiment, we will introduce two examples of sensor fusion in this part using assumed data to make the sensor fusion process more clear.

Table 4.2 and Table 4.3 show the BPAs of the PIR sensors and ultrasonic sensors which were calculated by performing thousands of experiments. $m_1(A)$ and $m_2(A)$ are the BPAs of the event A. It can be considered as the probability of the human in the relevant grid while $m_1(\bar{A})$ and $m_2(\bar{A})$ can be considered as the probability that the human is outside of the grid.

Table 4.2 Experimental BPAs for human sensor

Human sensor	BPA
$m_1(A)$	0.9
$m_1(\bar{A})$	Less than 0.1

Table 4.3 Experimental BPAs for ultrasonic sensor

Ultrasonic sensor	BPA
$m_2(A)$	0.95
$m_2(\bar{A})$	Less than 0.05

Next, we will introduce two examples of sensor fusion method. Each one of them shows the fusion result in a particular situation in detail. For the first situation, as figure below shows, human only intrudes the grid No.1. Figure 4.29 and Figure 4.30 show the scene and the simulation screenshot of this example, respectively.

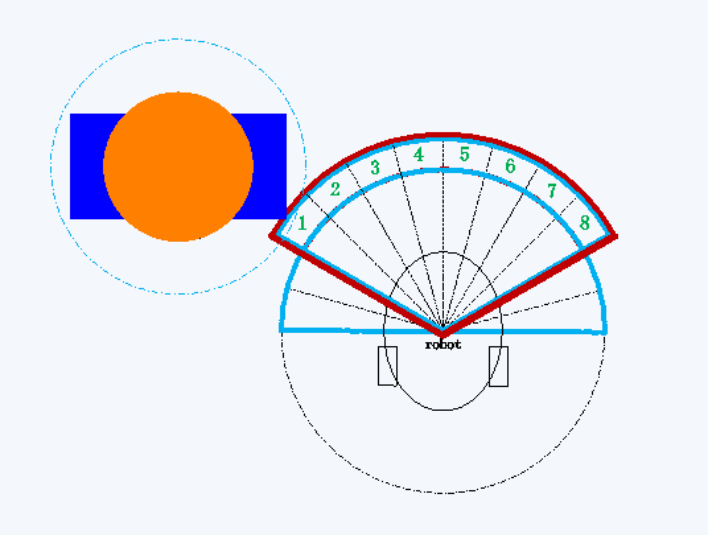


Figure 4.29 The first situation (human intrudes the grid No.1)

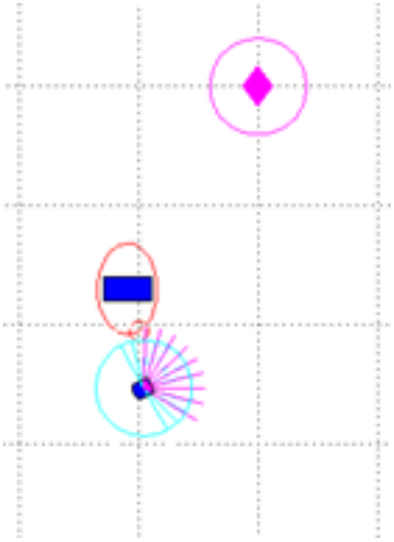


Figure 4.30 The first situation in simulation (human intrudes the grid No.1)

The Table 4.4 shows the BPAs of each grid derived by two kinds of sensors, which means the probability of human in each grid, while Table 4.5 shows the BPN calculation process for grid 1.

Table 4.4 The experimental BPAs assignments for the human and the ultrasonic sensors of the first situation

	Grid 1	Grid 2	Grid 3	Grid 4	Grid 5	Grid 6	Grid 7	Grid 8
Human sensor	0.9	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Ultrasonic sensor	0.95	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Table 4.5 Application of D-S theory for Grid 1

$m_h(A_1)$=0.9 $m_h(A_1 \cup \overline{A_1})$=0.1	m(A_1) =0.855 m($A_1 \cup \overline{A_1}$) =0.095	m($A_1 \cup \overline{A_1}$) =0.045 m($A_1 \cup \overline{A_1}$) =0.005
	$m_s(A_1)$=0.95	$m_s(A_1 \cup \overline{A_1})$=0.05

By using D-S theory, i.e. equation 4.66 which is derived from equation 4.65, we combine the data that are collected by different sensors and calculated the BPN of Grid 1 P=0.855, which means the probability of human in Grid 1 is 85.5%. Here, we set 80% as the critical value. If the probability of human is higher than 80%, our robot will treat the threat as a human.

$$\mathrm{m}(A_1) = \frac{\sum_{A_1} m_h(A_1) m_s(A_1)}{1-\sum_{\overline{A_1}} m_h(A_1) m_s(A_1)} = \frac{0.9\times0.95}{1-0} = 0.855 \qquad (4.66)$$

where A_1 is the focal elements that the human which is inside the grid 1. $\overline{A_1}$ is the focal element that the human is outside the grid1.m$(A_1)$is the combinedprobability that the human is inside the grid 1.

For the BPN value of grid 2, when the sensor output is low, the calculating process is listed in Table 4.6.

Table 4.6 Application of D-S theory for Grid 2

$m_h(\overline{A_2})$=0.9 $m_h(A_2 \cup \overline{A_2})$=0.1	$m(\overline{A_2})$ =0.855 $m(A_2 \cup \overline{A_2})$ =0.095	$m(A_2 \cup \overline{A_2})$ =0.045 $m(A_2 \cup \overline{A_2})$ =0.005
	$m_s(\overline{A_2})$=0.95	$m_s(A_2 \cup \overline{A_2})$=0.05

After we combined the data and we found that the BPN of the grid 2 is less than 0.145, this means the probability of the human inside the grid 2 is less than 14.5%.

$$m(A_2) = \frac{\sum_{A_2} m_h(A_2) m_s(A_2)}{1-\sum_{\overline{A_2}} m_h(A_2) m_s(A_2)} < \frac{0.0045+0.005+0.095}{1-0} < 0.0145 \quad (4.67)$$

where A_2 is the focal elements that human is inside the grid 2. $\overline{A_2}$ is the focal element that human is outside the grid 2.$m(A_2)$ is the combined probability that human is inside the grid 2. Since the probability of human inside the grid 2 is less than 80%, we ignore the possibility that human is inside the grid 2 in our search.

In the end, for the sample time, the system will list the BPNs of all the 8 grids as the Table 4.7 shows and compare each other to find the suspicious human position.

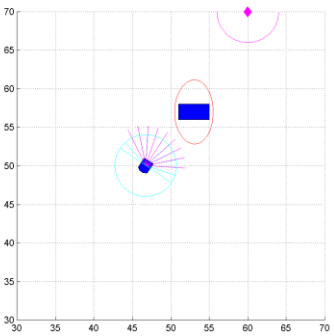
Table 4.7 The BPN values of the 8 grids of this sample time

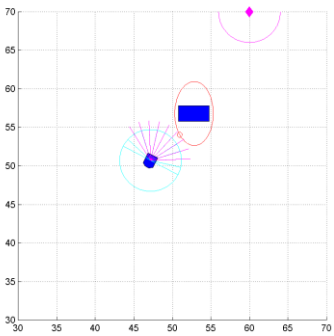
	Grid 1	Grid 2	Grid 3	Grid 4	Grid 5	Grid 6	Grid 7	Grid 8
BPN(basic probability number)	0.855	Less than 0.145	Less than 0.145	Less than 0.145	Less than 0.145	Less than 0.145	Less than 0.145	Less than 0.145

From the list, we can see that the grid 1 has the largest probability that human is inside itself. So as a result, we assumed that human is in grid 1.

The robot will scan the human zone and do such calculation every sample time and search for suspicious grid in which the human is inside. The occupancy grid is updated at each sample time. Once it finds one of the grids for which BPN is larger than 0.8, it will return angle and distance values to the system.

Finally, we will use a complete human avoiding process to show our sensor fusion method. As the figures below show, from 4 to 18 seconds, the mobile robot detected and avoided a human successfully.





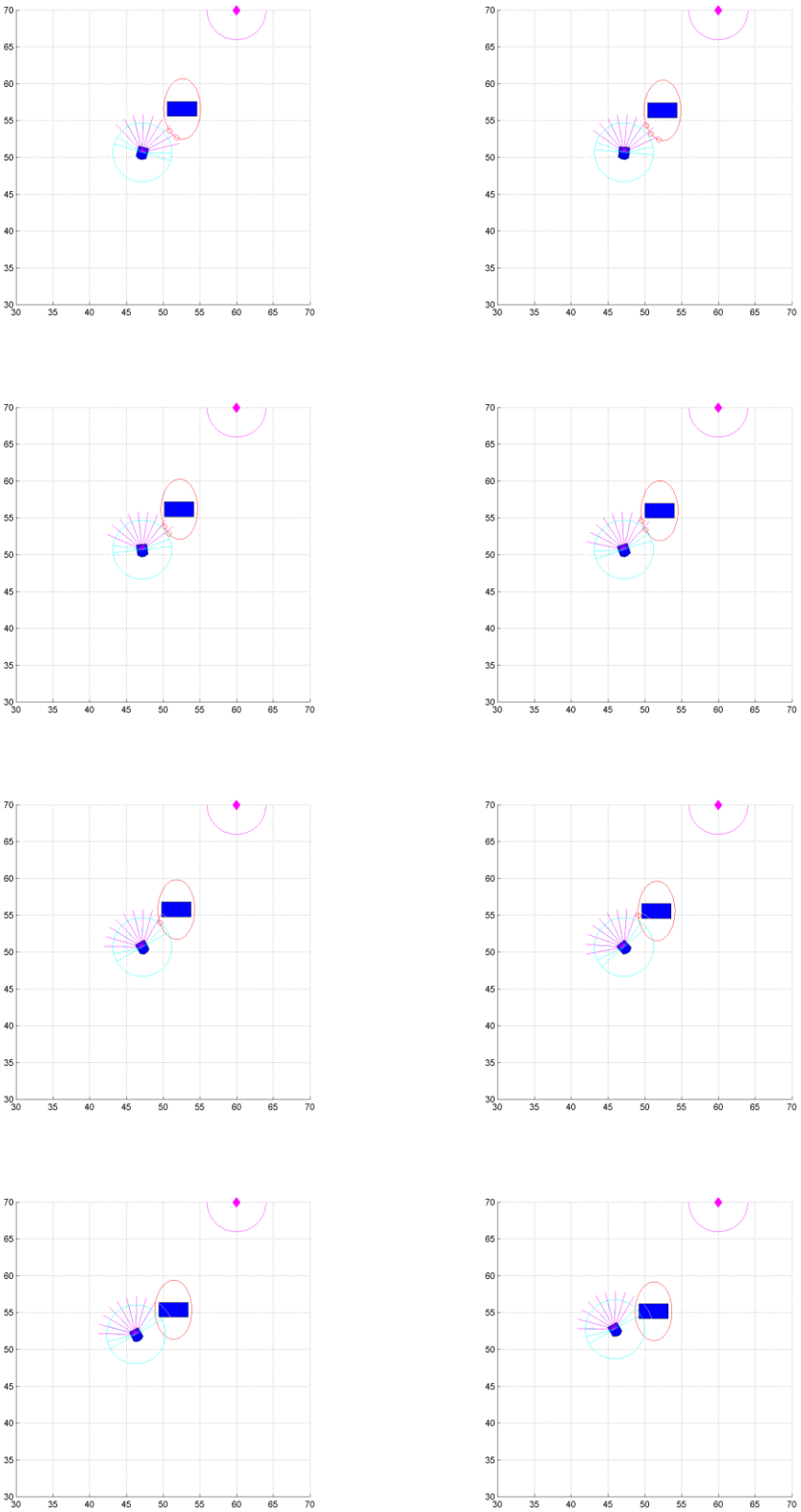


Figure 4.31 A complete human avoiding process

We recorded the BPNs values of the human during the whole process, as Table 4.8 shows.

Table 4.8 The BPN values of the human in each grid during the whole process

	Grid 1	Grid 2	Grid 3	Grid 4	Grid 5	Grid 6	Grid 7	Grid 8
BPN(0S)	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145
BPN(2S)	<0.145	<0.145	<0.145	<0.145	0.855	<0.145	<0.145	<0.145
BPN(4S)	<0.145	<0.145	<0.145	<0.145	<0.145	0.855	0.855	<0.145
BPN(6S)	<0.145	<0.145	<0.145	<0.145	<0.145	0.855	0.855	0.855
BPN(8S)	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	0.855	0.855
BPN(10S)	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	0.855	0.855
BPN(12S)	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	0.855
BPN(14S)	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	0.855
BPN(16S)	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145
BPN(18S)	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145	<0.145

From the Table 4.8, we can see that the result matches the simulation process completely. The robot first detected the robot in grid 5, then kept turning left so the human appeared at the right side of the robot and in the end, the robot avoid the moving human from his right side.

It is worth mentioning that sometimes there are more than two grids' BPN values which are larger than 0.8, which means the human is too close to the robot or there are more than one human is in the detection area as the figure below shows.

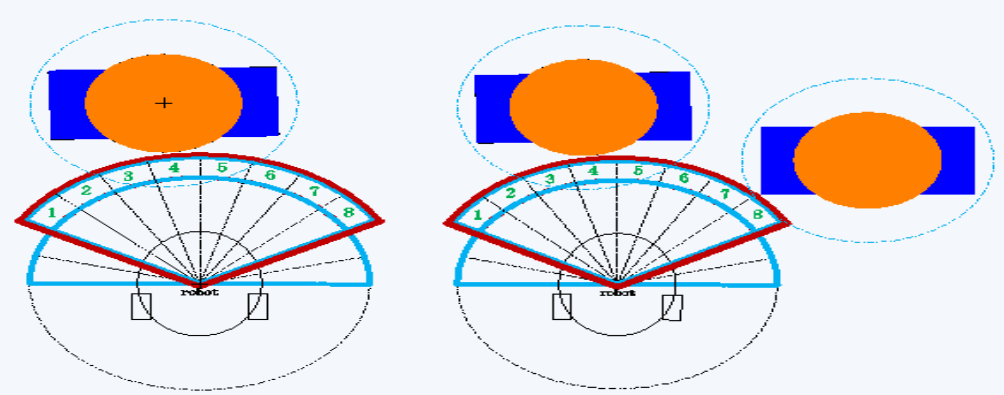


Figure 4.32 The possible situations that more than two grids' BPN values are larger than 0.8

4.7 Proposed Navigation Method with the Consideration of Human Priority

In this part, we will introduce a novel navigation method which is able to help our robot prevent different collision situations. There are three kinds of velocity commands in our navigation method-the attractive velocity V_a, the repulsive velocityV_{rep} and the rotation velocityV_{rot}.

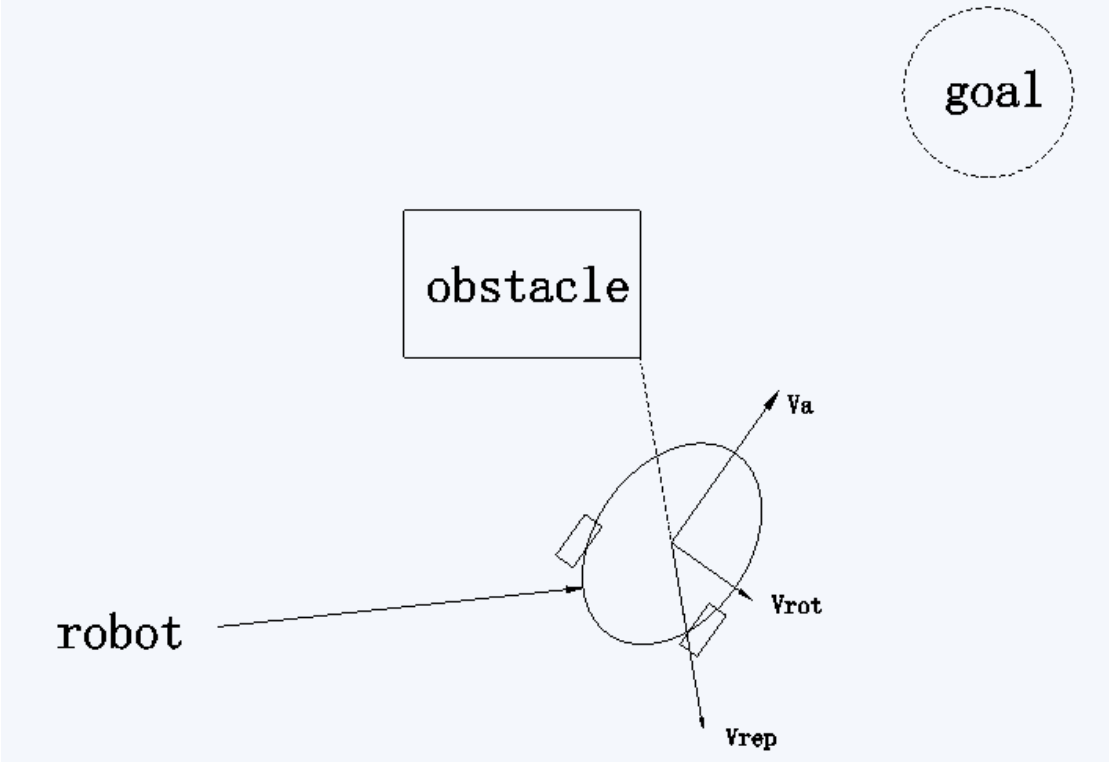


Figure 4.33 The attractive, rotation and repulsive velocity commands

4.7.1 The Attractive, Repulsive and Rotation Velocity Commands

An attractive velocity is used to guide the robot to the goal position. As the figure shows, the direction of attractive velocity is from robot to the goal so that the angle of attractive velocity command is [28]

$$\theta_a = \tan^{-1}\left[\frac{(yG-y)}{(xG-x)}\right] \quad (4.68)$$

as the Figure 4.34 shows

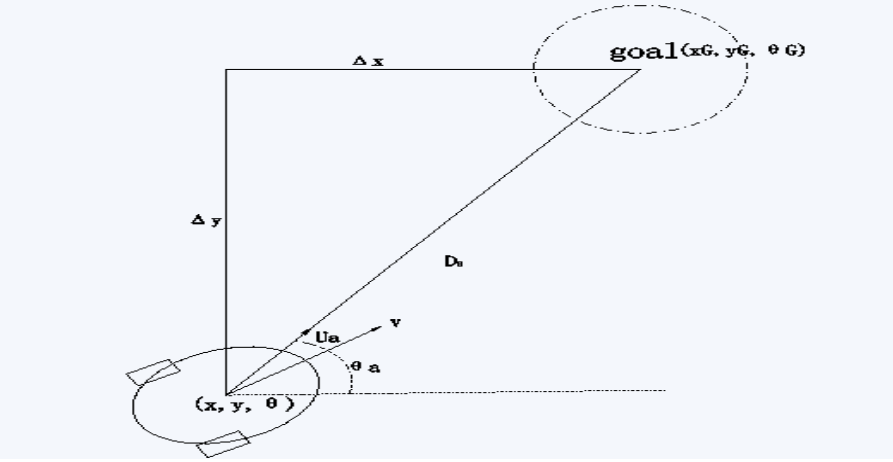


Figure 4.34 Attractive velocity variables

The distance between robot and goal d_{goal} is

$$d_{goal} = \sqrt{(yG - y)^2 + (xG - x)^2} \tag{4.69}$$

where (xG, yG) and (x, y) are the positions of the goal and robot, respectively.

The goal radius is $g_{rad} = 0.4\ m$

and the attractive velocity function is defined as [28]

$$V_a = 2 \times \overline{V}_{mr} \times (1 - e^{-\frac{d_{goal}}{g_{rad}}}) \tag{4.70}$$

where $\overline{V}_{mr}$ is the maximum velocity of robot.

A repulsive velocity is designed to help the robot to move away from the human and the obstacles whether they are moving or not. It seems like it pushes the robot away from obstacle to an opposite direction. So the angle of repulsive velocity is as the

Figure 4.35shows,

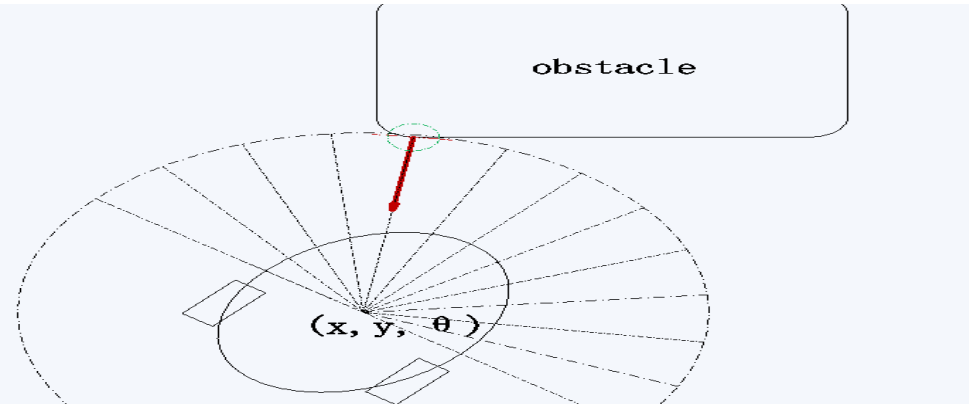


Figure 4.35 The direction of repulsive velocity command

$$\theta_{rep} = \theta_{beam} - \pi \tag{4.71}$$

where θ_{rep} is the angle of repulsive velocity and θ_{beam} is the angle form obstacle to the robot, which is collected by the sensors on the robot.

The obstacle radius is $obs_{rad} = 0.4\ m$ (4.72)

The repulsive velocity function is defined as [28]

$$V_{rep} = 0.5 \times \frac{1}{d_{obs}} \times e^{-\frac{d_{obs}}{obs_{rad}}} \tag{4.73}$$

where d_{obs} is the distance between obstacle and robot which is got by the sensors

Here, we also created a repulsive velocity function for human, which is a little bit larger than the repulsive velocity function for obstacle.

$$V_{rephuman} = 0.7 \times \frac{1}{d_{obs}} \times e^{-\frac{d_{obs}}{obs_{rad}}} \quad (4.74)$$

As we can see here, both of the $\frac{1}{d_{obs}}$ and $e^{-\frac{d_{obs}}{obs_{rad}}}$ in the equation cause a sustained increase of the repulsive velocity if the robot keeps approaching the obstacles or human, which mean that the closer the obstacle or human is, the larger is the repulsive velocity.

Besides, by changing the gain of the repulsive velocity, we can get different avoidance paths of the robot. In our project, small gains are used for the repulsive velocity commands (0.5 for obstacle and 0.7 for human) so that sometimes, the robot will avoid the obstacles or the humans compromising the distance to the less important one (the obstacles). For example, as Figure 4.36 (left) shows, with a weak repulsive velocity command (with the gain equal to 0.5 for the obstacle and the gain equal to 0.7 for the human), the robot collided with the obstacle as long as it did not has enough room to avoid all of the three directions' dangers. If we increase the gain to 3 for the repulsive velocity commands for both the humans and the obstacles, which means the obstacle and human will give the robot strong repulsive velocity commands as the equation 4.75 shows.

$$V_{rep} = V_{rephuman} = 3 \times \frac{1}{d_{obs}} \times e^{-\frac{d_{obs}}{obs_{rad}}} \quad (4.75)$$

where d_{obs} is the distance between obstacle and robot which is got by the sensors

We can achieve a different result. As the Figure 4.36 (right) shows, once the robot moved close to the obstacle and the humans, the robot were pushed back by the two humans and the obstacle and run away from the dangerous situation in the end. The result will also be showed in the simulation part.

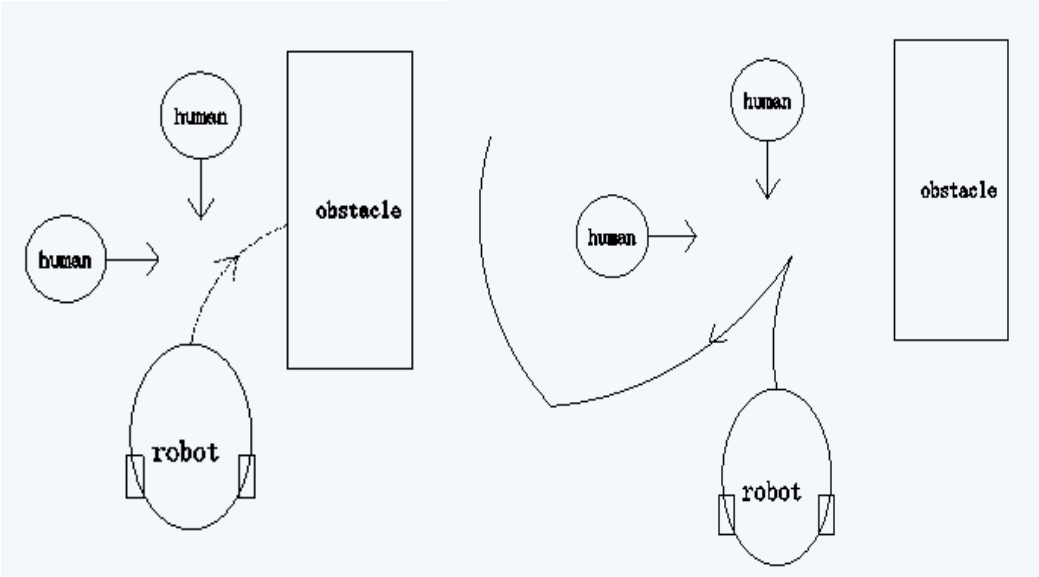


Figure 4.36 The avoidance path with small repulsive velocity gains (left), the avoidance path with large repulsive velocity gains (right)

The rotation velocity makes the robot spin when there is a human or obstacle ahead. By adding a rotation velocity in the navigation controller, our robot will be able to detour around the approaching obstacles or human while still moving to the goal. Besides, with the help of rotation velocity, the robot can resolve a collinear condition when a local minima problem happens [28]. The problem is that after the robot detected something in front of it, there are two directions for the robot to rotate.

Which direction will the robot rotate to so that it can avoid the obstacle more effectively? The Figure 4.37 illustrates the difference of the two choices.

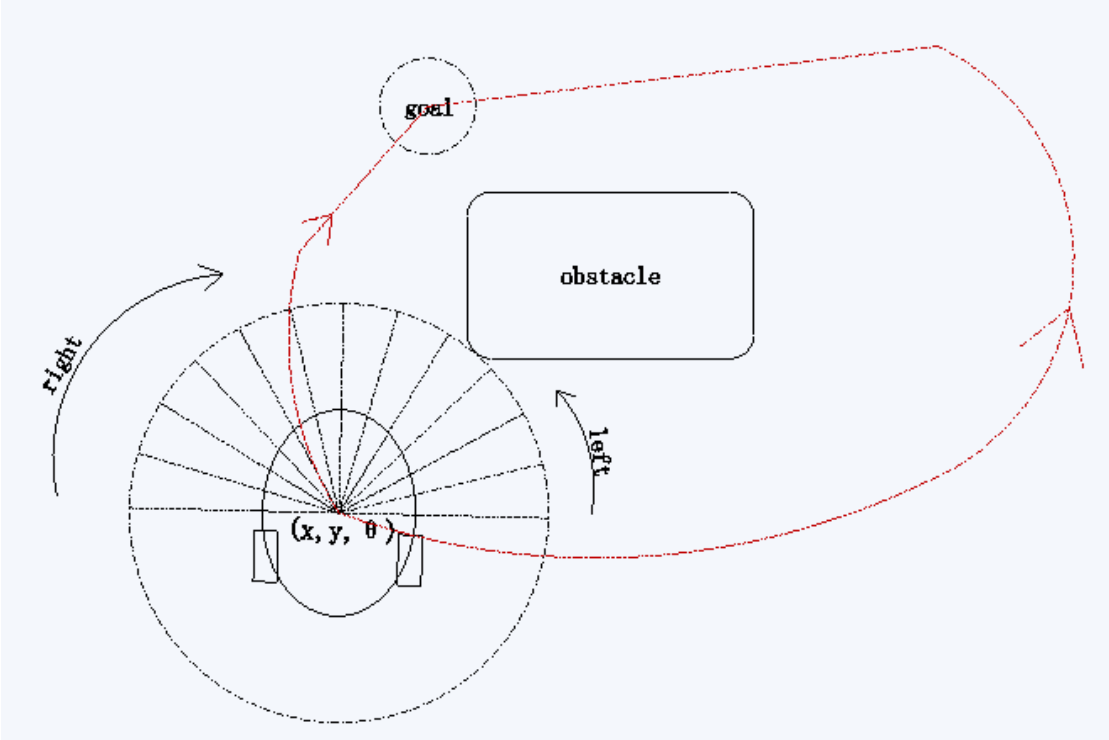


Figure 4.37 The difference of the paths for two directions of rotation

In this example, the robot has two choices for turning. Compared with the right direction, rotating to left will reduces the length of the path and the arrival time significantly. In this thesis, for different situations we created two different rotation commands:

1) If the obstacle or human is detected in the right side of the robot as the figure above shows, the robot will turn left. So the angle of rotation velocity is

$$\theta_{rotation}=\theta_{rep}-\pi/2 \tag{4.76}$$

2) If the obstacle or human be detected in the left side on in front of the robot, the

robot will turn right. So the angle of rotation velocity is

$$\theta_{rotation}=\theta_{rep}+\pi/2 \tag{4.77}$$

The rotation velocity function is defined as [28]

$$V_{rot}=0.5\times\frac{1}{d_{obs}}\times e^{-\frac{d_{obs}}{obs_{rad}}} \tag{4.78}$$

where d_{obs} is the distance between obstacle and robot which is got by the sensors the rotation velocity function for human is

$$V_{rothuman}=0.5\times\frac{1}{d_{obs}}\times e^{-\frac{d_{obs}}{obs_{rad}}} \tag{4.79}$$

4.7.2 Information Integration

As we mentioned earlier, four zones are designed for the robot. They are: dangerous zone, stationary zone, moving zone and human zone. The sizes of the four zones are based on robot and human shapes, velocity limits and acceleration limits which have been calculated in the section 4.5.

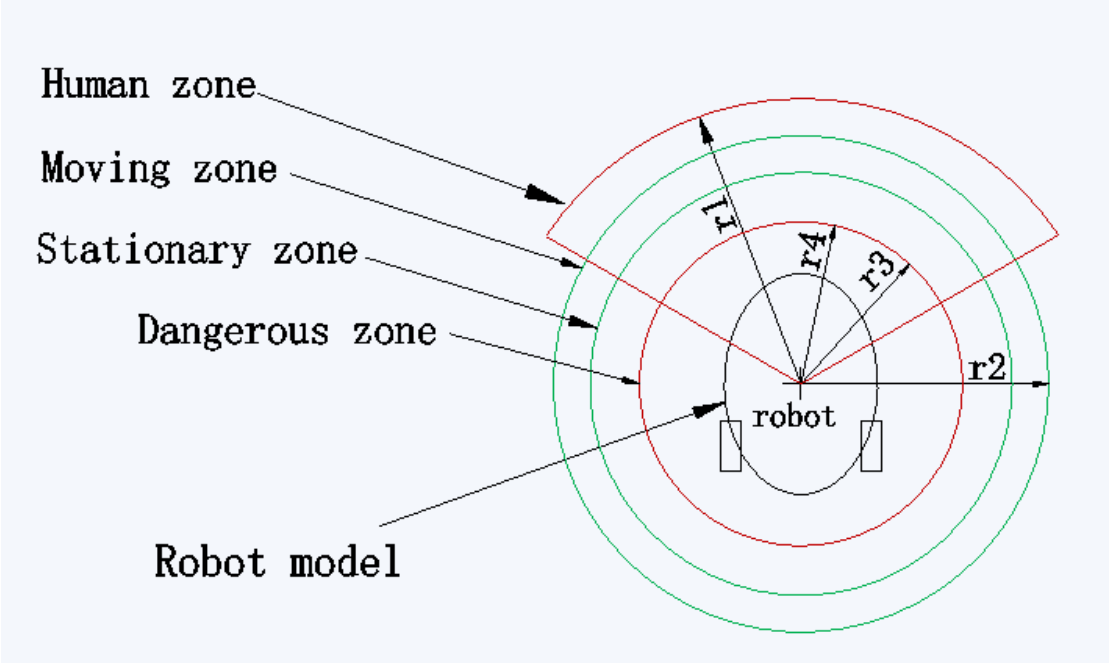


Figure 4.38 The four zones

It is very undesirable if anything intrudes into the dangerous zone because the robot cannot complete the avoidance and should stop to prevent or at least mitigate the collision. For the other three zones, once the robot finds something in them, the repulsive and detour velocities will be activated to make the robot avoid and detour around it. It is worth mentioning that the human zone can be detected by the human sensor and sonar senor while the moving and stationary zones will be kept watch on by infrared and sonar sensors. Moreover, in moving and stationary zones, the human body will be considered as an obstacle as well because the sonar sensors cannot distinguish them.

Once the obstacles or human intrude into the zones of the robot, the sensors will collect the information of the "intruder". After all the information is collected, next step is to combine them together and then give a resultant angle and velocity to the

robot's motor controller to react. In total, there are four kinds of information the sensors will collect.

1. The human direction is θ_{human} which is collected by human and sonar sensors.

2. The human distance is d_{human}, which is collected by human and sonar sensors.

3. The obstacle (human body and obstacle) angle θ_{beam}.

4. The obstacle distance d_{obs}.

As we can see from the Figure 4.35, the human zone is larger than the other zones. The human sensor, which is also named passive infrared sensor, is the sensor detects level of infrared radiation. Compared with the infrared sensor, its sensitive range is much larger (about 6 meters while infrared is less than 1 meters). Thus, it is able to "see" the human from a relative larger distance. So normally, the information which is obtained by the human sensor will be the first processed by the robot controller.

4.7.3 The Consideration of Human Priority

To make the human be the first priority when dealing with different kinds of obstacles, we designed two factors.

The first one is, as we mentioned before, that the human zone is larger than the other zones, which means the robot will start to avoid and detour the human from a relative longer distance than the obstacles. The Figure 4.39 shows the sizes of human and

moving zones.

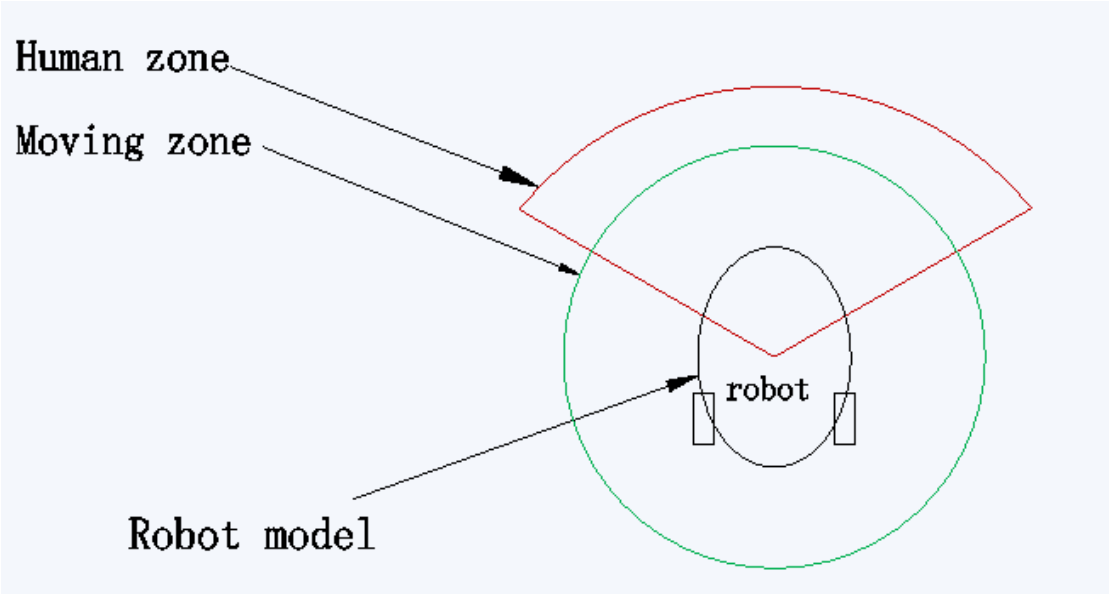


Figure 4.39 The sizes of human and moving zones

Secondly, the human direction θ_{human} and the human distance d_{human} have the priority when calculate the resultant command.

In the algorithms, based on different situations, we designed corresponding methods.

1) If the robot detects one human in its human zone, the sector area as figure below shows, the controller will use the information received from the human sensor to avoid the human in most cases.

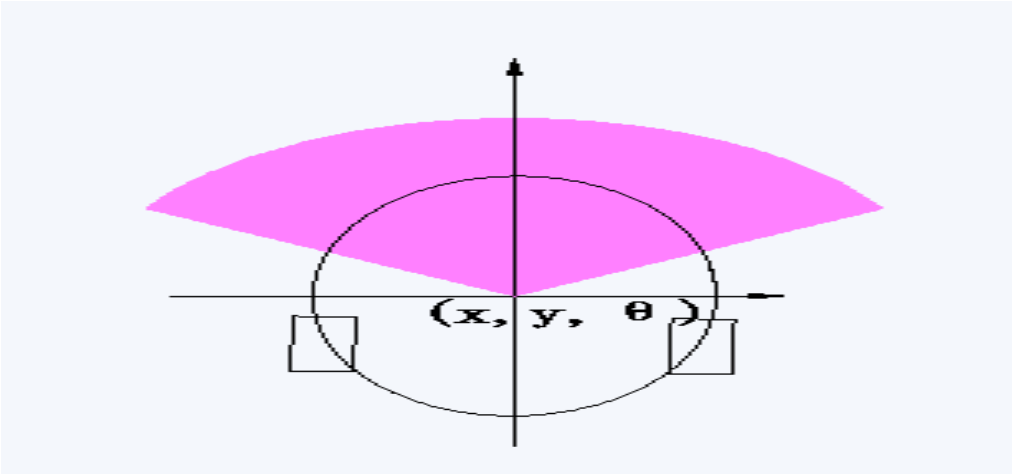


Figure 4.40 The human zone

2) If the robot detects more than one human in the human zone when the controller received more than one group of angle and distance data, the robot will ignore the information which is collected by other sensors in stationary and moving zones to make sure the human priority. That is to say, when there are more than one human in the pink area, the robot will sacrifice the avoidance of other obstacles and only avoid human. In this case, if the sensors on the robot find that something intrudes into robot's dangerous zone (because the robot pays too much attention to the human), the robot will stop immediately. That is to say, no matter under what kind of circumstance, the dangerous zone will always work as the last line of defense.

3) If the robot does not find any human in its view, the robot will then be helped by the sonar sensors. The detect area of the sonar sensors is showed by the figure below which is colored blue.

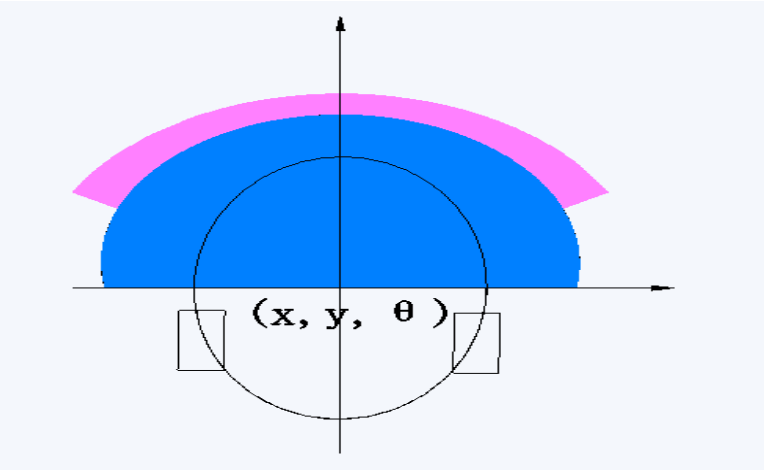


Figure 4.41 The sonar sensor detection area (the blue semi-circle)

In our simulations, we assume that human will not chase the robot, so the sensors area is only designed in front of robot.

The way that we applied these two insurances will be described in next chapter because there are program methods.

In the end, the resultant velocity command can be defined as

$$V_{sum} = \begin{cases} V_a\,, & P_r \in A \\ V_a + V_{rephuman} + V_{rothuman}, & P_r \in A,\ P_h \in A_0 \\ V_a + V_{rephuman} + V_{rothuman}, & P_r \in A,\ P_h \in A_0, P_{obs} \in A_1 \\ V_a + V_{rep} + V_{rot}, & P_r \in A, P_{obs} \in A_1 \\ undifined, & P_r \in A_2 \cup A_3 \end{cases} \quad (4.80)$$

where V_a, V_{rep} and V_{rot} are the attractive, repulsive and rotation velocity commands, respectively. The $V_{rephuman}$and$V_{rothuman}$ are velocity command for human.The A is the whole map area whileA_0,A_1,$A_2$and A_3 represent the human, moving, stationary and dangerous zones.

Chapter 5

Hardware and Software Architecture

5.1 Hardware Architecture

In our project, we chose for our experiments Mindstorms NXT platform which is a programmable robotics kit released by Lego in 2006. Until now, Lego released three generations of Mindstorms kit- the Robotics Invention System, the Mindstorms NXT 2.0 and the newest Mindsrotms EV3. In our experiment, we used the second generation Mindstorms NXT 2.0 to achieve our targets. Figure 5.1 shows the Lego mobile robot which was assembled by us.

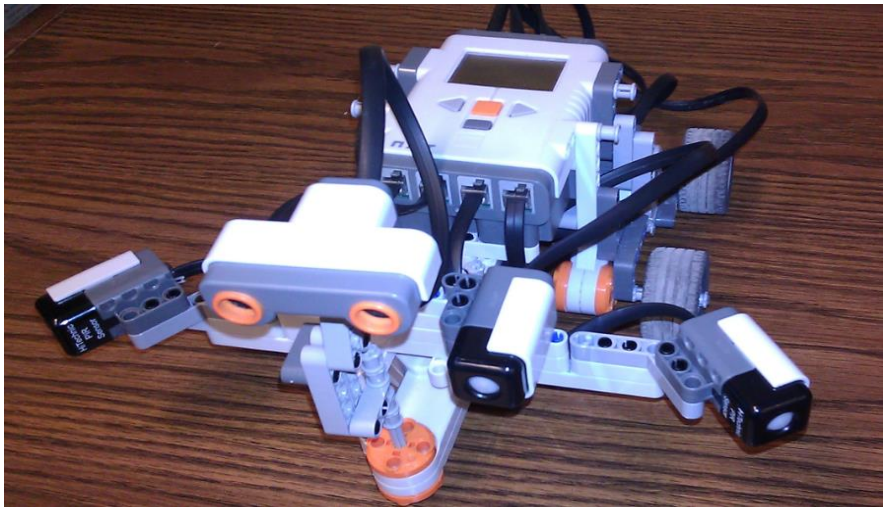


Figure 5.1 Lego mobile robot

There are more than 10 kinds of sensors that can be used on Mindstorms NXT 2.0. Under the consideration of our project and due to the limited data interfaces of the robot, we chose 3 PIR sensors (human sensor) and one ultrasonic sensor to help our

Lego robot. The Figure 5.2 shows the two different kinds of sensors.

Figure 5.2 The ultrasonic sensor (left) and the human sensor (right)

Besides of the sensors, the main component of the Mindstorms NXT 2.0 is a computer called the NXT intelligent Brick AKA (Ciara). Up to four sensors and three motors can be connected to it through a modified version of RJ12 cables. The Figure 5.3 and 5.4 show the NXT Intelligent Brick and its interfaces, respectively. The brick has a 100×60 pixel monochrome LCD display. The four buttons on the brick were designed to help the users to manipulate the interface by hierarchical menus. It has a 32-bit ARM7TDMI-core AtmelAT91SAM7S256 microcontroller with 256KB of FLASH memory and 64KB of RAM, plus an 8-bit Atmel AVRATmega48 microcontroller and bluetooth support. It is powered by 6 AA (1.5V each) batteries under the back of the unit.

Figure 5.3 The NXT Intelligent Brick

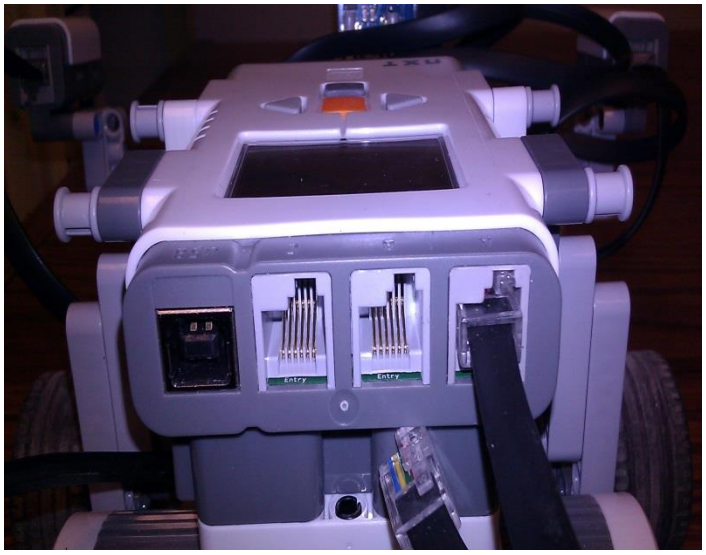


Figure 5.4 The interfaces of The NXT Intelligent Brick

Mindstorms NXT 2.0 can be controlled using various programming software packages such as NXT-G, ROBOTC, MATLAB and Simulink and so on. We chose LabVIEW Tookit to program the navigation of the robot. It is powered by LabVIEW, an industry standard in programming and we will introduce it in software architecture.

5.2 Software Architecture

In the simulation part, we first used the MATLAB™ to test the navigation controller. MATLAB is a numerical computing environment and fourth-generation programming language.

Developed by Mathworks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces and interfacing with programs written in other languages, including C, C++, Java and Fortran. Figure 5.5 shows the operation interface of MATLAB.

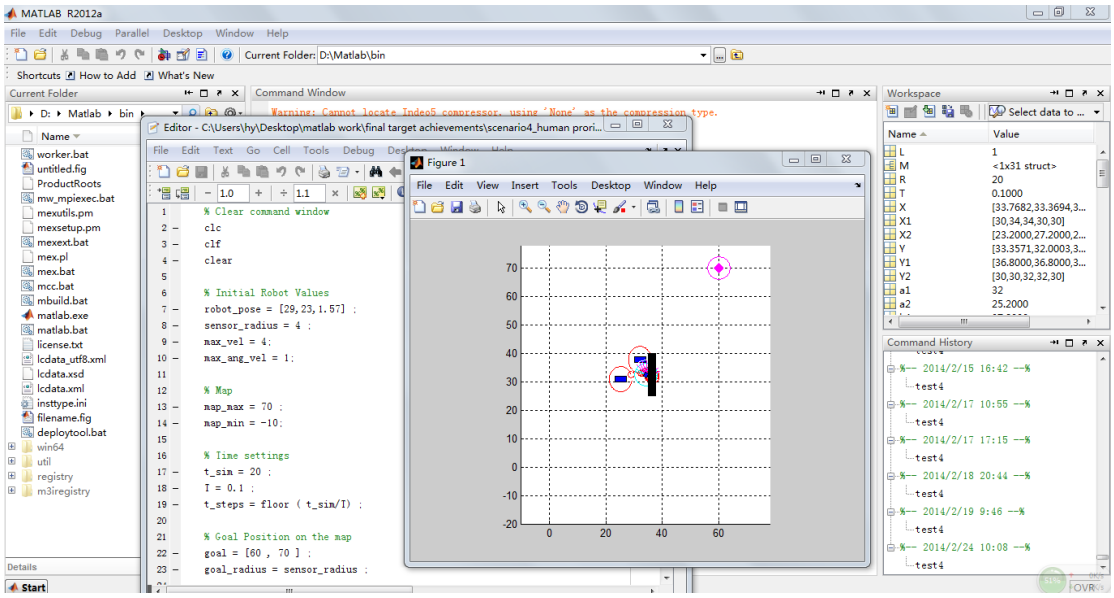


Figure 5.5 The operation interface of MATLAB

Once our navigation algorithms were tested successfully in MATLAB, the next step was transform the algorithms, which were written in MATLAB, program into G programming language which is also called G code or LabVIEW programming. After

that, we applied the G codes through LabVIEW to run our robot in the experiments.

LabVIEW is a highly productive development environment for creating custom applications that interact with real-world data or signals in fields such as science and engineering. Figure 5.6 shows the operation interface of LabVIEW.

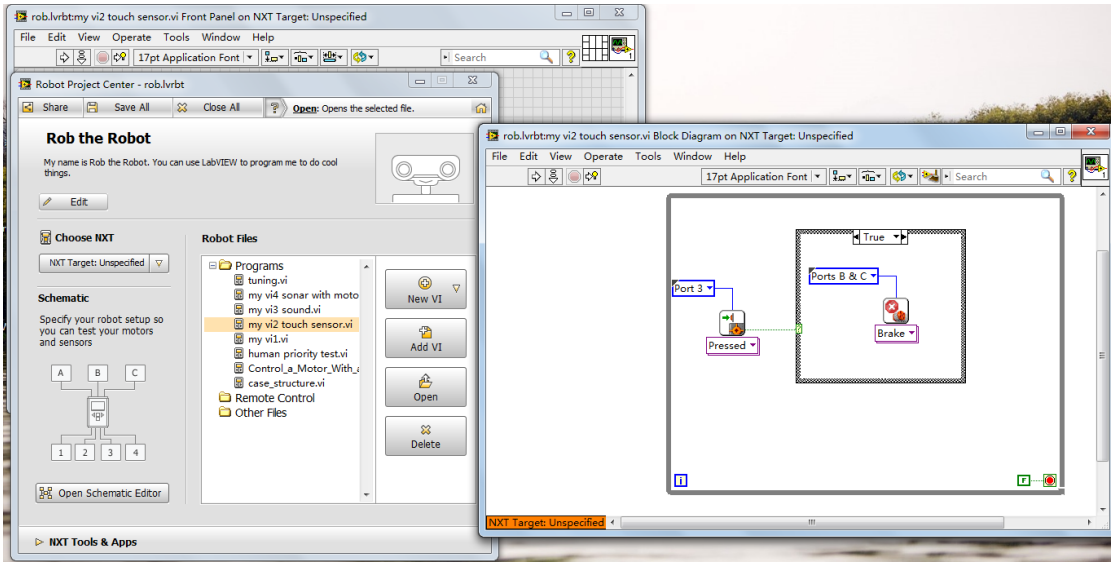


Figure 5.6 The operation interface of LabVIEW

The programming language that is used in LabVIEW, also referred as G, is a dataflow programming language. Execution is determined by the structure of a graphical block diagram (the LabVIEW-source code) on which the programmer connects different function-nodes by drawing wires. The LabVIEW programs are named virtual instruments (VIs), as showed in Figure 5.7 which corresponds to a motor's VI.

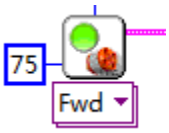


Figure 5.7 The virtual instrument of a motor

Each VI has three components: a block diagram, a front panel and a connector panel. And the VIs in LabVIEW can be also understood as the virtual representations of lab equipment. Through wires (the red string in Figure 5.7), we can combine different VIs together to build programs.

Chapter 6

Simulation Design and Results

6.1 Simulation Design

In the simulation, we test if the robot can work taking into account human priority. We will test the robot controller step by step. The following chapter describes our simulation algorithms and the results of our simulation.

Section 6.1.1 gives a quick overview of the flowchart symbols which is used to describe the methodology for our programming code. Section 6.1.2 describes our simulation algorithms while Section 6.2 presents the results of the simulation.

6.1.1 Flowchart Symbols

In this thesis, all of our simulations were carried out in the MATLAB environment. Since the programming codes are long and complicated, we will use the flowcharts to visually and briefly display the methodology of our program. The programming codes for our simulation and experiment are listed in the Appendix A and B. Figure 6.1 displays the six kinds of flowchart symbols which were used in this chapter.

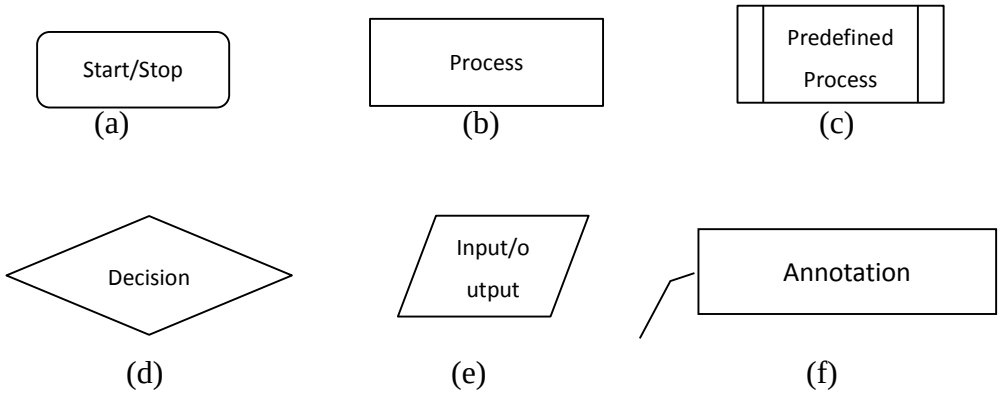
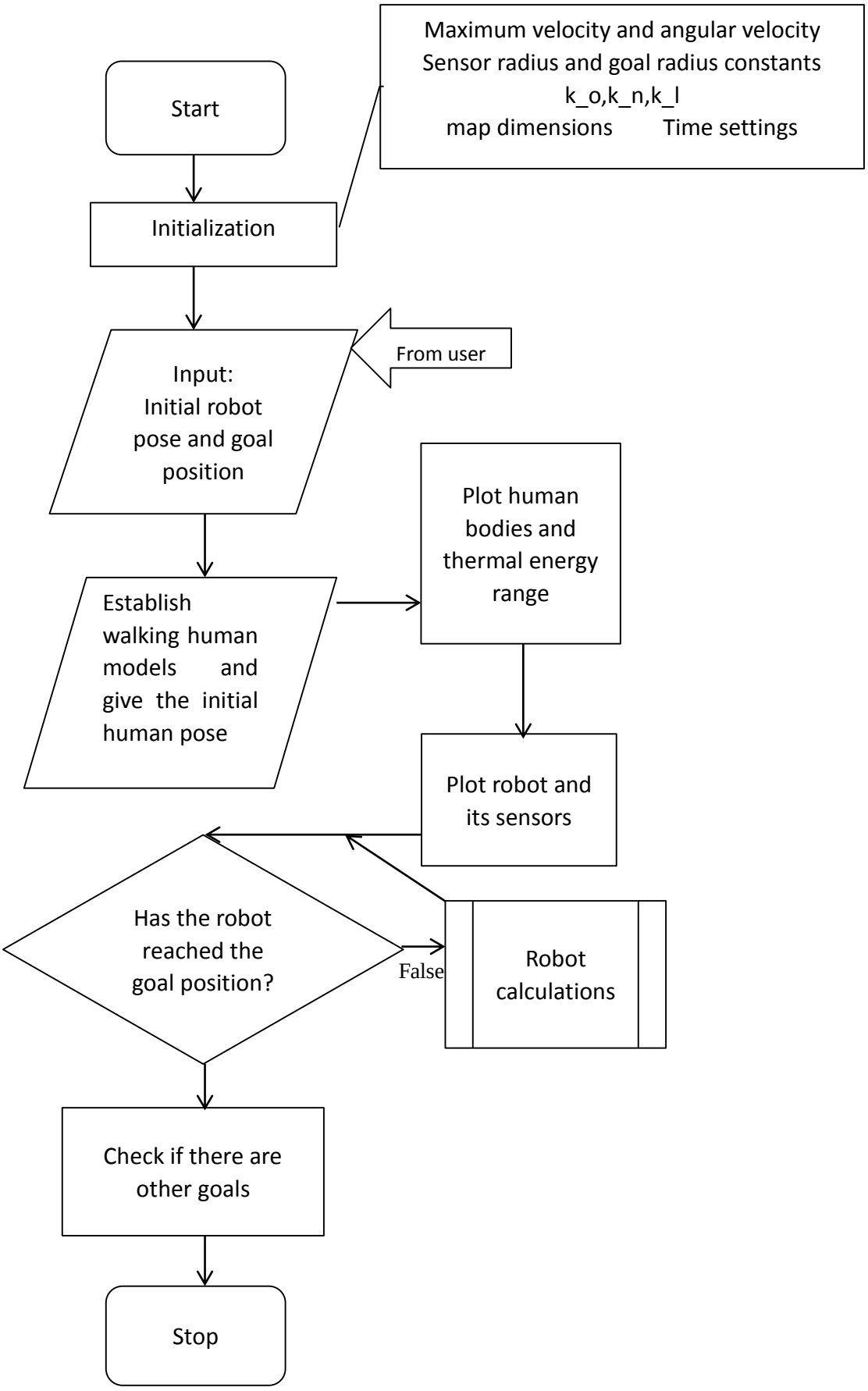


Figure 6.1 Flowchart symbols

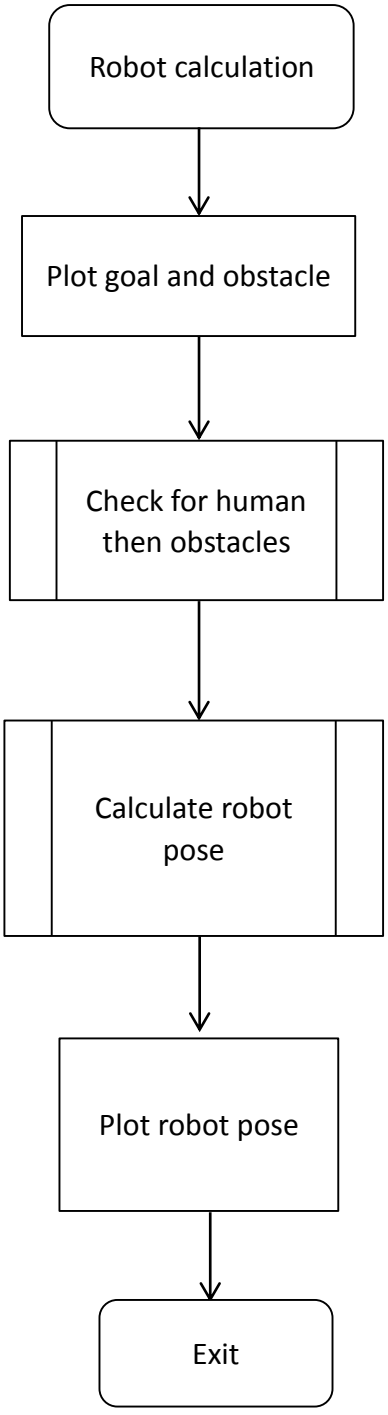
Figure 6.1(a) represents the beginning and the end of a procedure or program. Figure 6.1(b) represents a processing task within the program. All of the data manipulation, computation and movement of data in the program will be illustrated by a process symbol. Figure 6.1(c) is a symbol that represents multiple processes. That is to say it is an independent subprogram which is located parallel as the main program. The main purpose of us to use this symbol is to simplify the description. Figure 6.1(d) represents a point in the program where the logic flow will follow one of the two paths based on different given situations. Figure 6.1(e) is the Input/output process, which is used to indicate that some input or output operations being performed. The input operations include reading files and interacting with the user while output operations include writhing files and displaying information. Finally, the Figure 6.1(f) is used to make a comment on some portions of our program to help the reader has a better understanding of our program.

6.1.2 Introduction of Simulation Algorithms

In the simulation, the mobile robot navigates through the terrain towards the goal subject to an attractive velocity command. When it meets an obstacle or a moving human, a repulsive velocity command will generate and push the robot away from the obstacle or the human. At the same time, a rotation velocity command will help the robot find the best solution to detour the obstacles. The main function is showed by Figure 6.2(a). At first, it will gather all the information in terms of velocity, robot pose and goal position from the input function. Then the function will establish all the objects in the simulation such as human body, obstacles and the robots with its sensors. After that, it will do the robot calculation to help the robot avoid the human and obstacles and check if the robot reached the goal position. Finally, after all the goals have been achieved, the function will stop the robot. Figure 6.2(b) is called robot calculation, a sub-function of the main function, which is the most important part of our controller. It will first use the sensors on the robot to check if there are human and obstacles in view. If yes, it will find the distance and angle from the human or the obstacle to the robot. Then, the sub-function will calculate the robot velocity command using the method which we described in chapter 4. After that, with all the velocity command be calculated, the robot will get its velocity and angle.


Maximum velocity and angular velocity
Sensor radius and goal radius constants
k_o,k_n,k_l
map dimensions Time settings
Start
Initialization
From user
Input:
Initial robot
pose and goal
position
Plot human
bodies and
thermal energy
range
Establish
walking human
models and
give the initial
human pose
Plot robot and
its sensors
Has the robot
reached the
goal position?
False
Robot
calculations
Check if there are
other goals
Stop

(a)



(b)

Figure 6.2 (a) Main function, (b) Robot calculation function

In the robot calculation function, as we can see from the Figure 6.2 (b), there are two sub-functions named check for human then obstacles and calculate robot pose. The details of the check for human then obstacles are explained in Figure 6.3. The goal of this function is processing the information which gathered from different sensors with different priority levels. Briefly, this function gives higher priority to the information detected by human sensor when calculating the resultant angle and distance to make the human avoidance more important.

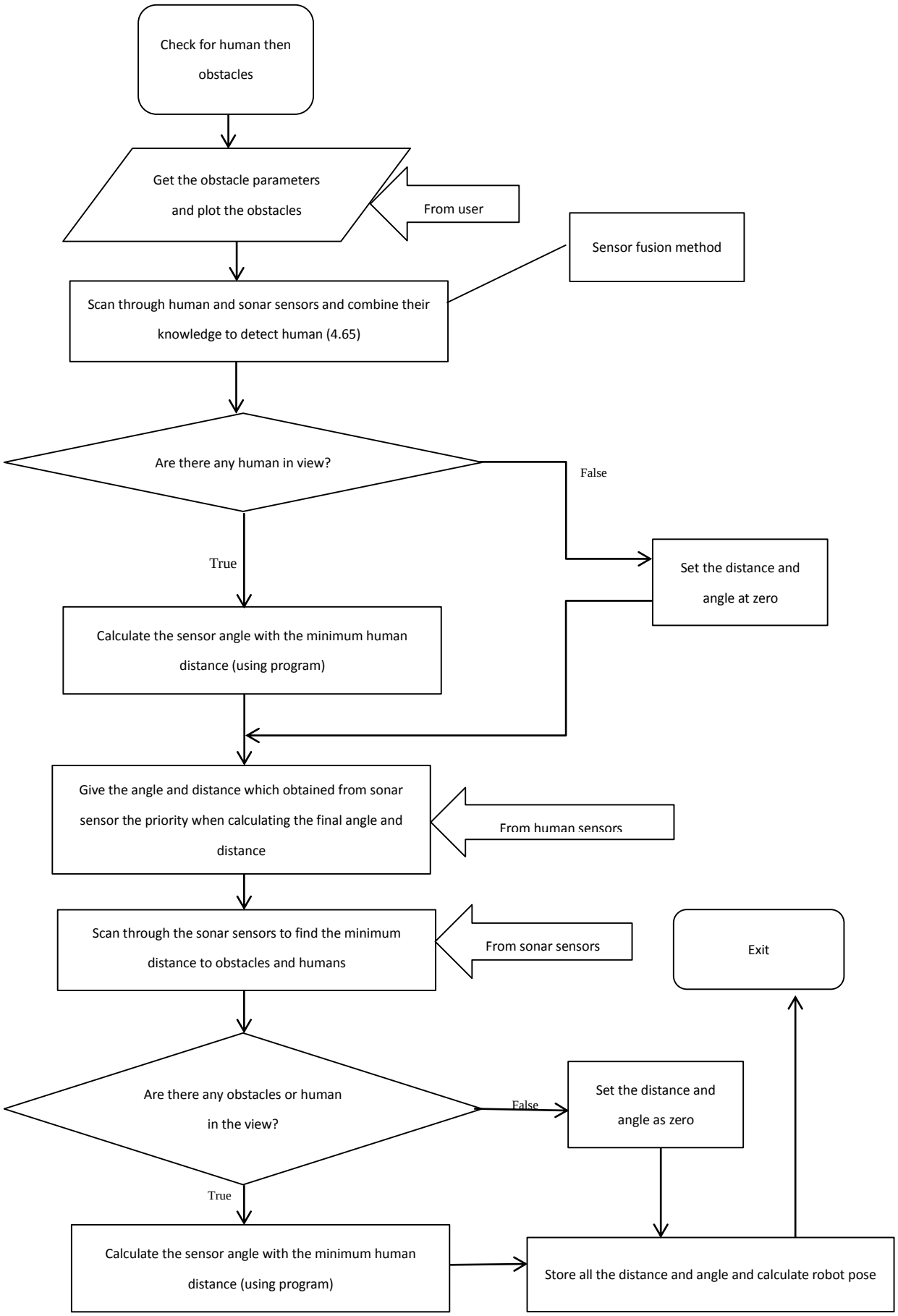

Check for human then obstacles
Get the obstacle parameters and plot the obstacles
From user
Sensor fusion method
Scan through human and sonar sensors and combine their knowledge to detect human (4.65)
Are there any human in view?
False
True
Set the distance and angle at zero
Calculate the sensor angle with the minimum human distance (using program)
Give the angle and distance which obtained from sonar sensor the priority when calculating the final angle and distance
From human sensors
Scan through the sonar sensors to find the minimum distance to obstacles and humans
From sonar sensors
Exit
Are there any obstacles or human in the view?
False
Set the distance and angle as zero
True
Calculate the sensor angle with the minimum human distance (using program)
Store all the distance and angle and calculate robot pose

Figure 6.3 Check for the human then obstacle function

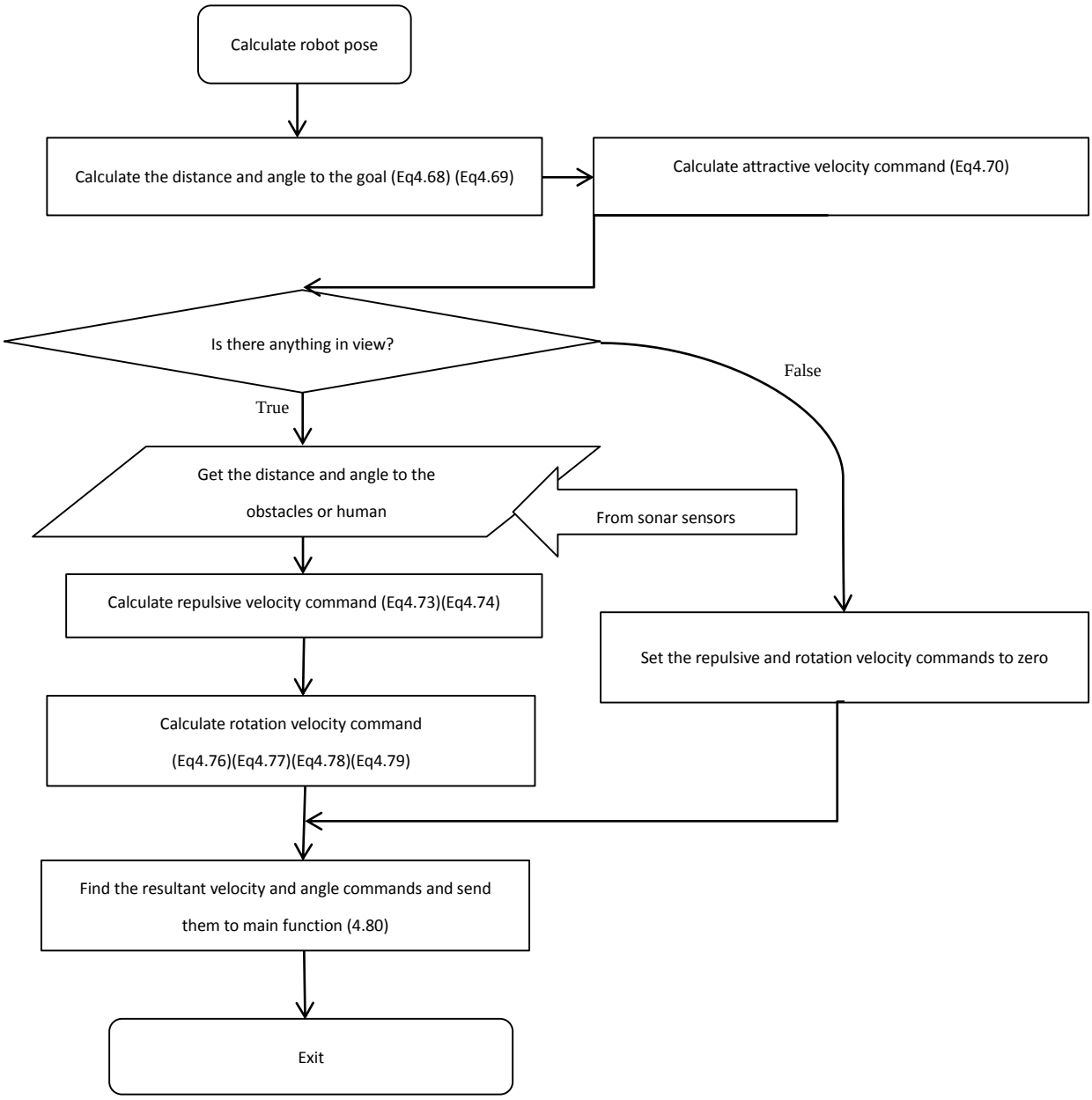


Figure 6.4 Calculate robot pose function

The function for robot pose calculation is described in Figure 6.4. It will use the final angle and distance from pre-sub-function to calculate the attractive, repulsive and rotation velocity commands. Then this function will combine all the velocity commands together and find the resultant velocity and angle as presented in Chapter 4.

6.2 Simulation Results

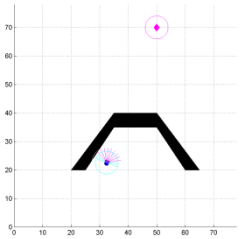
We sorted our simulations into four categories based on their different purposes. All simulation results were recorded by snapshots. In these cases, a human is represented as a blue rectangle. The red circle surrounding the human represents the human area which can be detected by the beams emitted by the robot. The red beams represent the human sensor and the sonar sensor combined detection range while the blue beams represent the sonar sensor detection area.

6.2.1 Mobile Robot Avoiding one Obstacle, one Stationary Human and one Moving Human

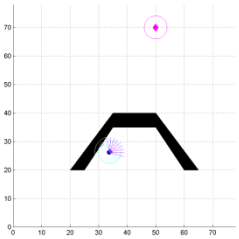
6.2.1.1 Avoiding one Obstacle

Firstly, we want to test if the robot has the ability to arrive to the goal position with only one stationary obstacle in its way. As Figure 6.5 shows, the robot was initially placed at an angle of -45 degrees, and there was a concave obstacle in between the robot and the goal. When the robot sensed the obstacle, the shortest sensor range was found and a red circle was plotted at the interface between the shortest sensor beam and the obstacle. The robot then applied the distance and angle measurement into its velocity calculation function to find the most suitable velocity and the angle to avoid the obstacle and got into the other path towards the goal position. In this case, as the obstacle is a concave, the robot moved toward the obstacle until it detour the obstacle. In the end, when the robot entered the region of interest around the goal position (the

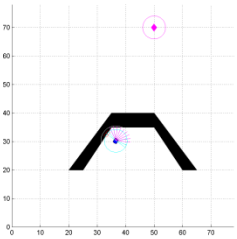
magenta circle), it slowed down gradually and stopped once it got to the goal position completely.



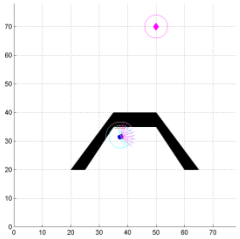
(a)



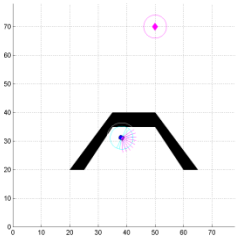
(b)



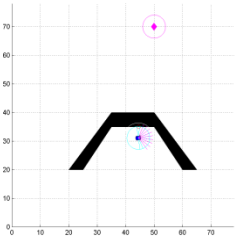
(c)



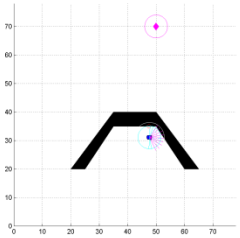
(d)



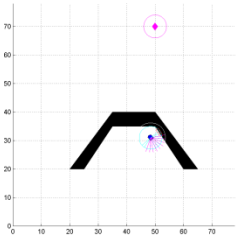
(e)



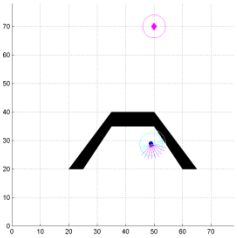
(f)



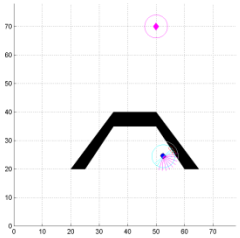
(g)



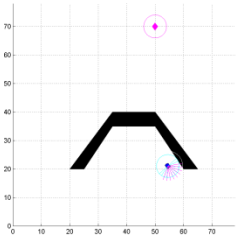
(h)



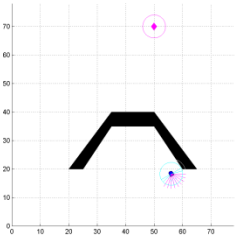
(i)



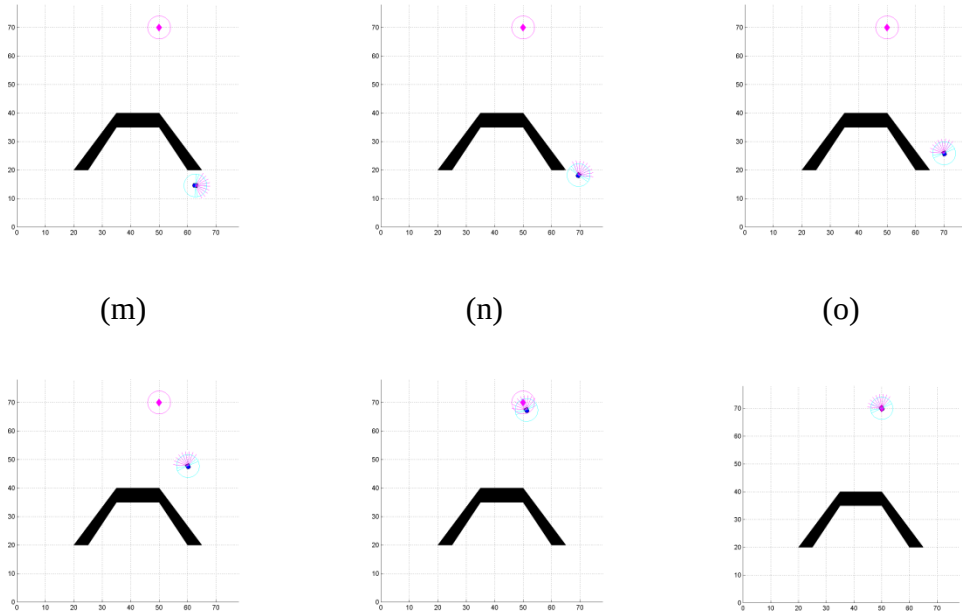
(j)



(k)



(l)



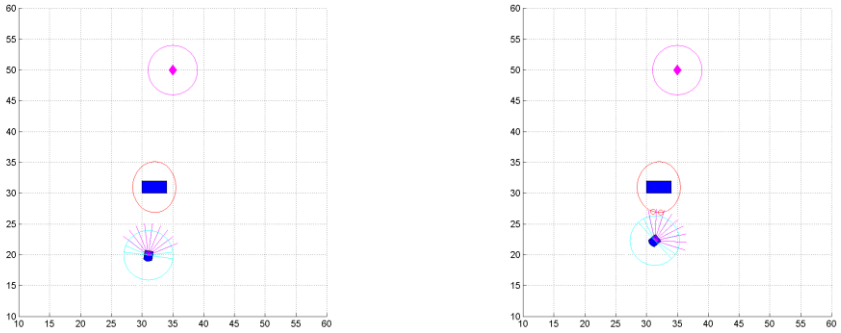
(m) (n) (o)

(p) (q) (r)

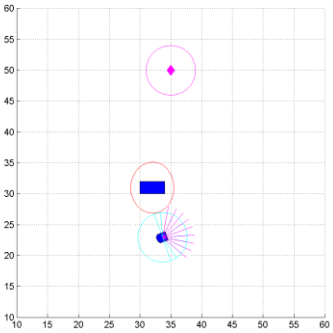
Figure 6.5 Obstacle avoidance

6.2.1.2 Avoiding one Stationary Human

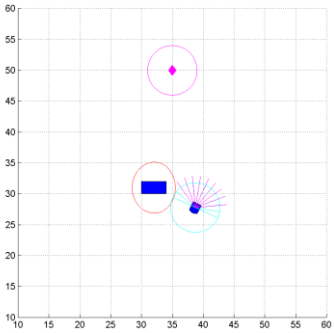
In this case, a human stood between the robot and the goal position, which means a potential collision situation. After the human was detected, the robot started to detour him/her. Finally, the robot avoided the human and walked to the goal position successfully.



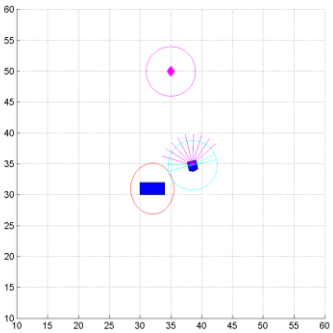
(a) (b)



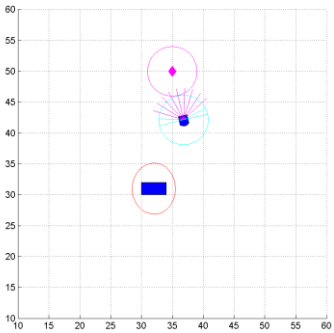
(c)



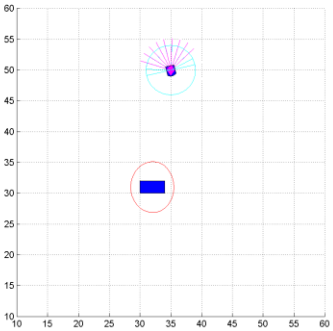
(d)



(e)



(f)



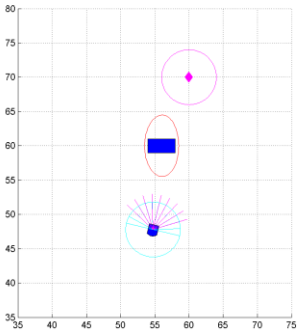
(g)

Figure 6.6 Stationary human avoidance

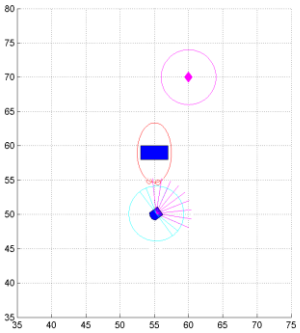
6.2.1.3 Avoiding one Moving Human

In this case, a human walked toward the robot walking direction, which means a

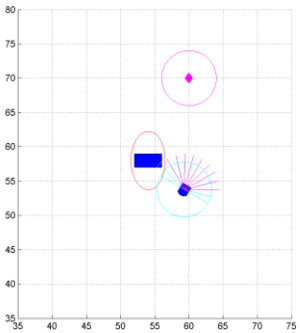
collinear situation. The purpose of this design is to test if the robot is able to avoid the moving human. Finally, the robot avoided the walking human and walked to the goal position successfully.



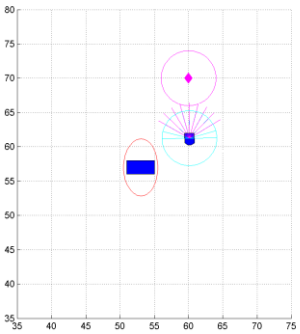
(a)



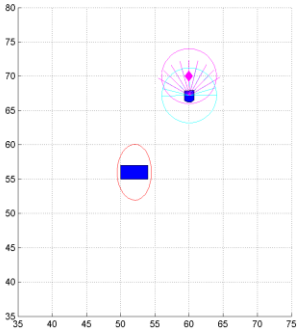
(b)



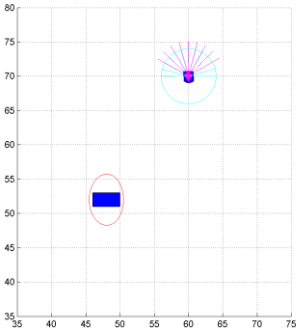
(c)



(d)



(e)

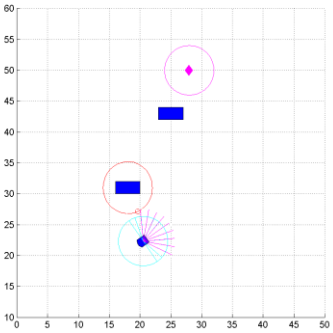


(f)

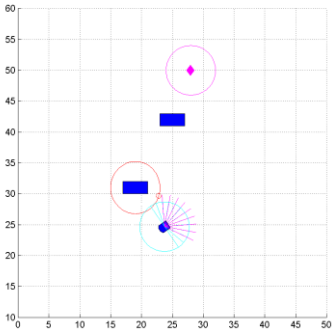
Figure 6.7 Moving human avoidance

6.2.2 Mobile Robot Avoiding one Moving Human and one Moving Obstacle at the Same Time

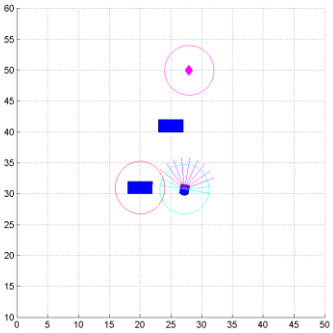
In this case, we are going to test if the robot can detect two different kinds of obstacles at the same time and make the decision to avoid the human at first then the obstacle as we expected. As the Figure 5.6 shows, the robot successfully completed the mission.



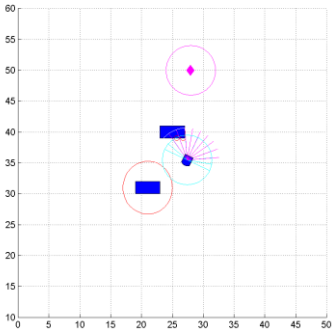
(a)



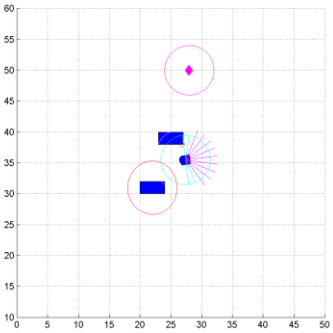
(b)



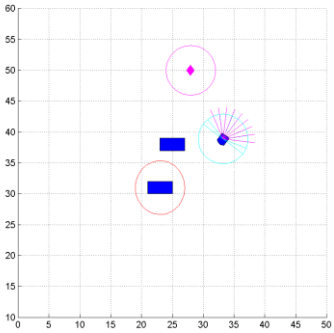
(c)



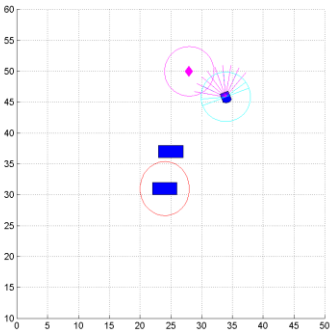
(d)



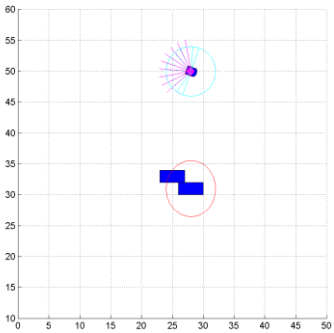
(e)



(f)



(g)



(h)

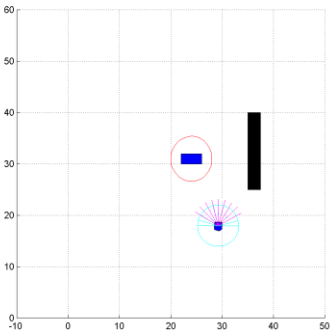
Figure 6.8 One moving human and one obstacle avoidance

6.2.3 Mobile Robot Performs Tasks with Human Avoidance Priority

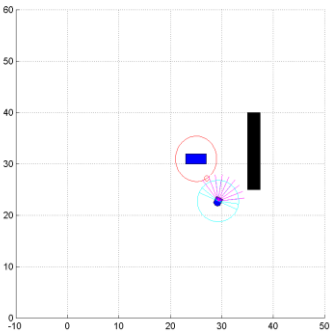
The main task of this thesis is to design a robot controller with the higher human priority for collision avoidance with humans. In Chapter 4, several functions were added to the controller to make sure the robot pay more attention to the human. In the simulation of this part, five extreme conditions were designed to test the robot. Sometimes the proposed condition was too complicated for robot controller to handle. However, the robot still considered the human safety as its first priority.

6.2.3.1 Higher Priority of Human Collision Avoiding with the Risk of Colliding with the Obstacle

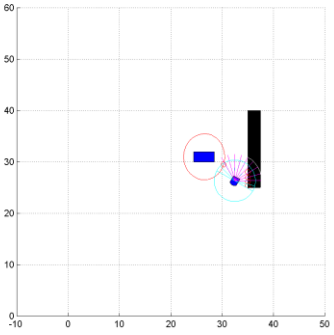
In this case, one of the humans walked in front of the robot from its left side. When the robot tried to detour the human, it detected an obstacle at its right side. At this time, the robot had no choice but go ahead. Finally, it got through the difficult situation with a steady distance to the human, which also means it took the risk to collide with the obstacle. As we can see from the Figure 6.9 the robot, compared with the human, is much closer to the obstacle.



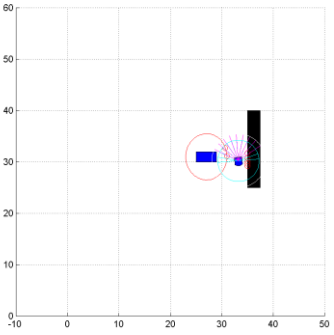
(a)



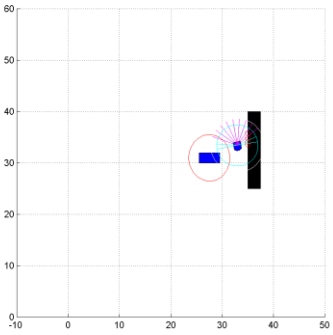
(b)



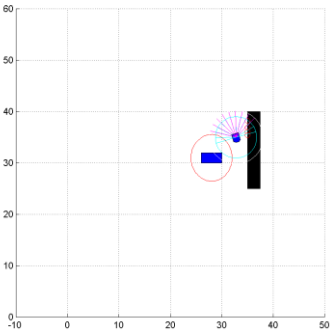
(c)



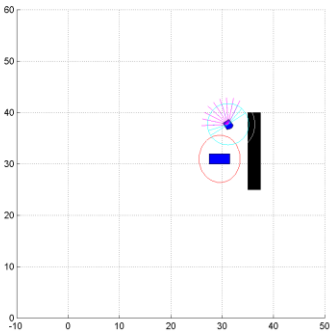
(d)



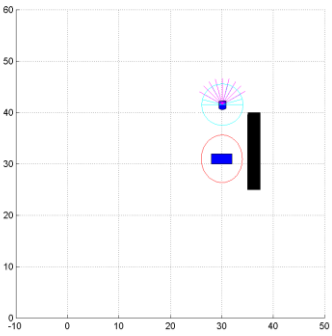
(e)



(f)



(g)

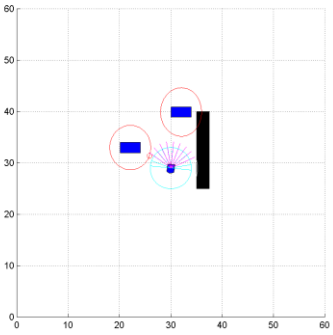


(h)

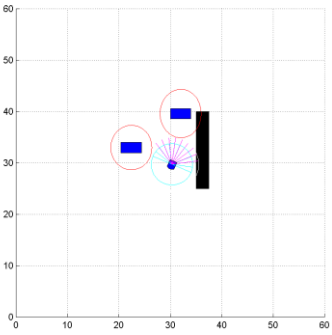
Figure 6.9 Priority with risk assignment (one human and one obstacle)

6.2.3.2 Running Away Robot

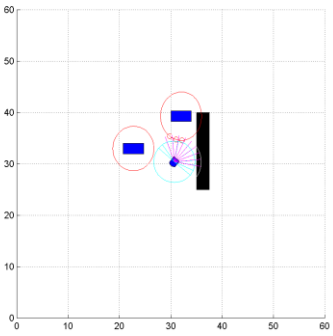
In this time, we simulated a more complicated situation that at first, the robot did not know it will face a three directions dangerous situation. The aim of creating such extreme situation is to test if the robot controller has the ability to avoid the human and the obstacle while it still has time and space. In the end, as we can see from the figure below, the robot turned around and ran away from the two human in an opposite direction which was the only direction with no danger, then turned around again to walk to the goal position.



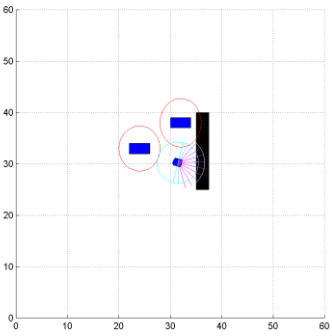
(a)



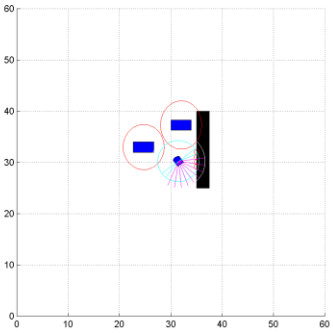
(b)



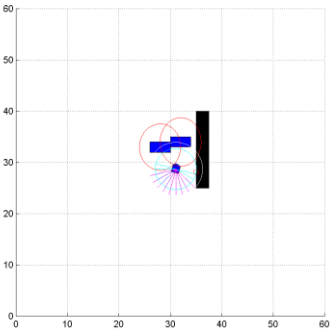
(c)



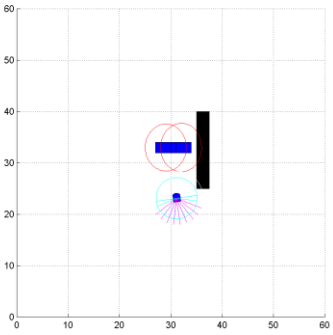
(d)



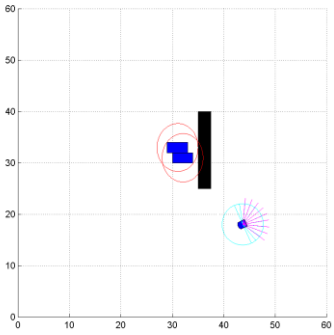
(e)



(f)



(g)

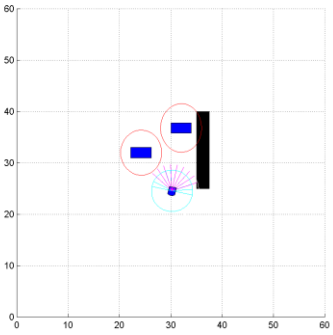


(h)

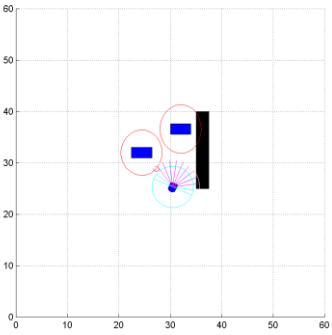
Figure 6.10 Higher human priority (two humans and one obstacle)

6.2.3.3 Stopping

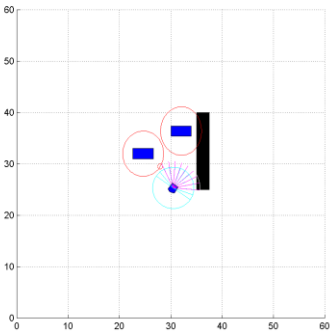
The third situation looks like a “trap”, two human walked directly toward the robot from the left and front side while there was an obstacle in the right side of the robot, which means that the robot had no time and space to run away but ended up colliding with one of them. This is the case for a non-holonomic robot. The robot turned toward the obstacle and as our controller is designed, once it touched the obstacle, it halted immediately. This result means even at the most dangerous situation, the robot will choose the less important object to collide to keep the human safe. The controller arrived to stopping the robot because the gains for the repulsive velocity commands were too low. In the next section higher gains were used.



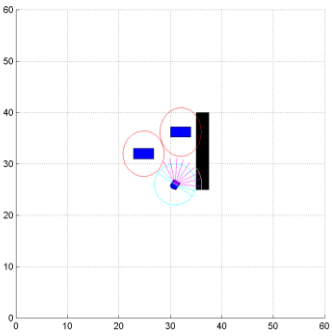
(a)



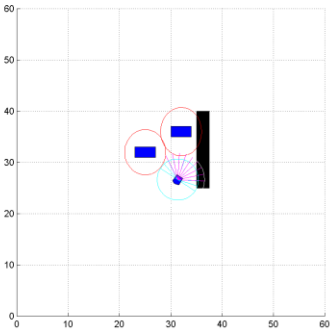
(b)



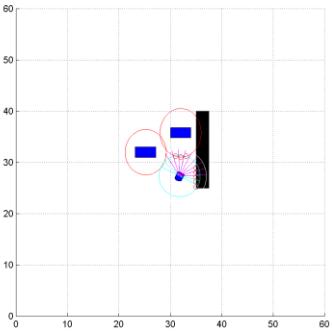
(c)



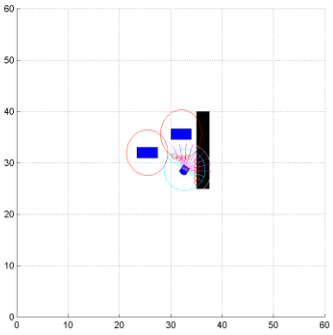
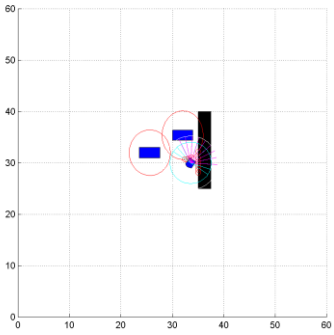
(d)



(e)



(f)

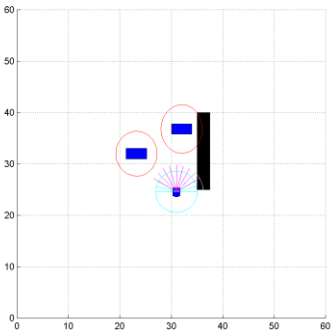
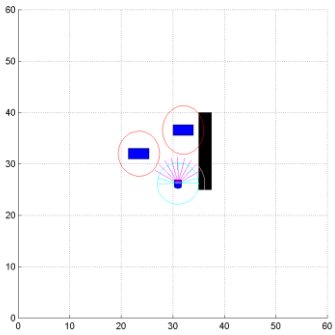



(g) (h)

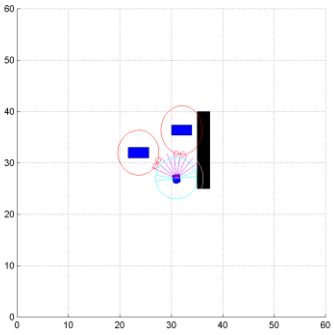
Figure 6.11 Stopping

6.2.3.4 Running Away with a Stronger Repulsive Velocity Command

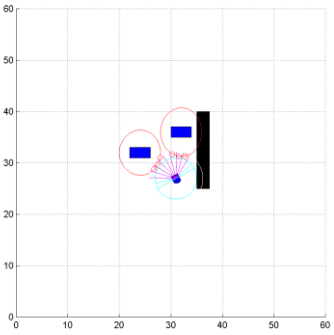
The forth situation is as same as the third situation except we applied a higher gains for the repulsive velocity commands of our robot controller. In other words, the human and the obstacles will “push” the robot away harder than before. So as a result, compared with the third situation, the robot did not collide with the obstacle but was pushed back by the humans and obstacle and run away from the dangerous situation. The result shows the appropriately chosen repulsive velocity gains will lead to successful collision avoidance while arriving eventually to the goal.

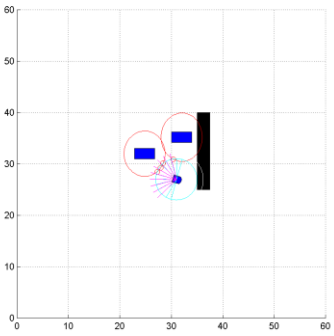
(a) (b)



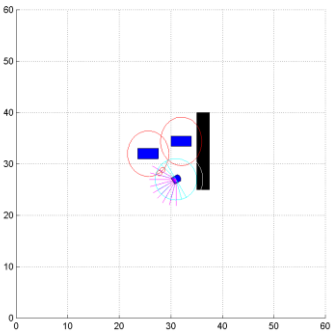
(c)



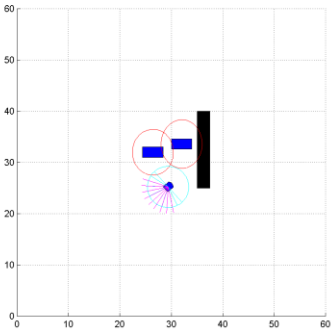
(d)



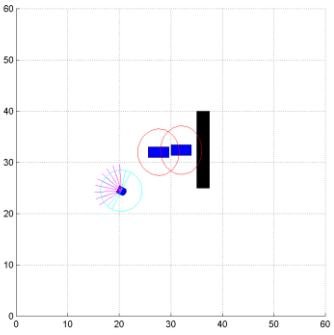
(e)



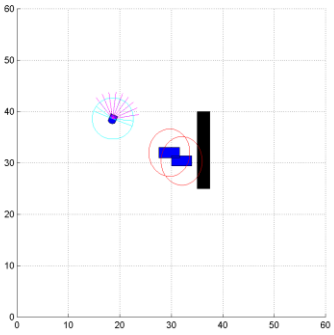
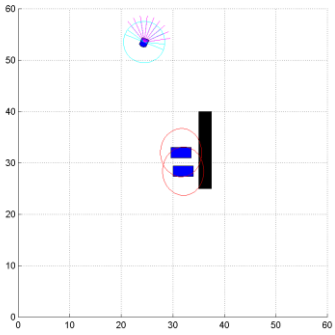
(f)



(g)



(h)

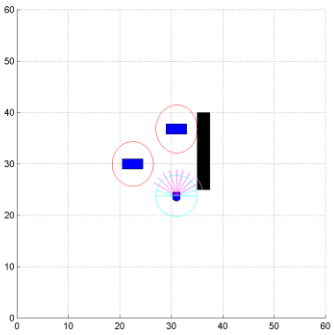
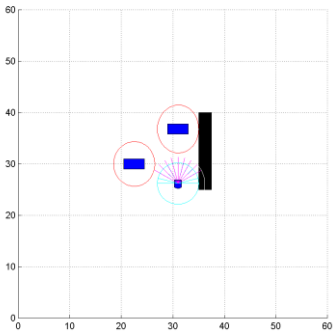



(i) (j)

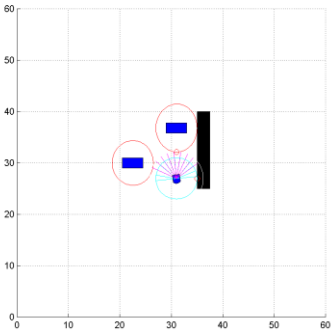
Figure 6.12 Running away with a stronger repulsive velocity commands

6.2.3.5 Stuck with a Local Minima in Case of APF

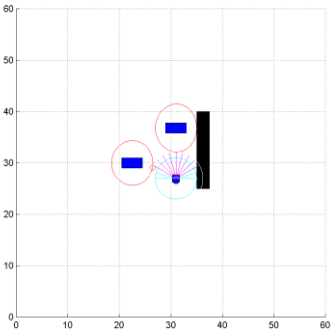
The fifth situation is as same as the third situation except the Artificial Potential Field (APF) method was applied to our robot controller as a counterexample instead of the velocity potential method. Since the APF method has no tangential force (rotation force) for the robot, the robot cannot conquer the local minima. As the Figure 6.13 shows, the robot was stuck in between the humans and the obstacle and cannot run away. This simulation illustrates the advantage of the velocity potential field method in solving the local minima problem of APF.

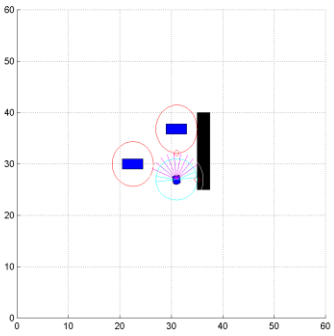
(a) (b)



(c)



(d)



(e)

Figure 6.13 Stuck in a local minima by using APF method

Chapter 7

Experimental Design and Results

7.1 Experimental Design

7.1.1 Introduction of Experimental Algorithms

In the experiments, we will apply our navigation controller to mobile robot through LabVIEW. The programming language of LabVIEW is called G, which is a dataflow programming language. Using the same flowchart as the one which was used to describe the Matlab program in Chapter 6, we can have a clear view of the structure of LabVIEW program.

Derived from equation (4.70) and also consider about the real situations of the experimental equipment (the Lego robot), we set the velocity of the robot to move ahead as

$$V_{ahead} = 2 \times \overline{V}_{mr} \times (1 - 0) = 0.10\ m/s \tag{7.1}$$

$$\overline{V}_{mr} = 0.05 m/s \tag{7.2}$$

where $\overline{V}_{mr}$ is the velocity of the Lego robot under 25% voltage.

And similar to the way we calculated the robot speed above, according to equation

(4.73), (4.74), (4.78) and (4.79), the velocity for the robot to detour the obstacle and human is

$$V_{detour} = 0.71 \times \overline{V}_{mr} \times \left(\frac{1}{d_{obs}} \times e^{-\frac{d_{obs}}{obs_{rad}}} \right) \tag{7.3}$$

where d_{obs} is the distance between obstacle and robot which is got by the sensors

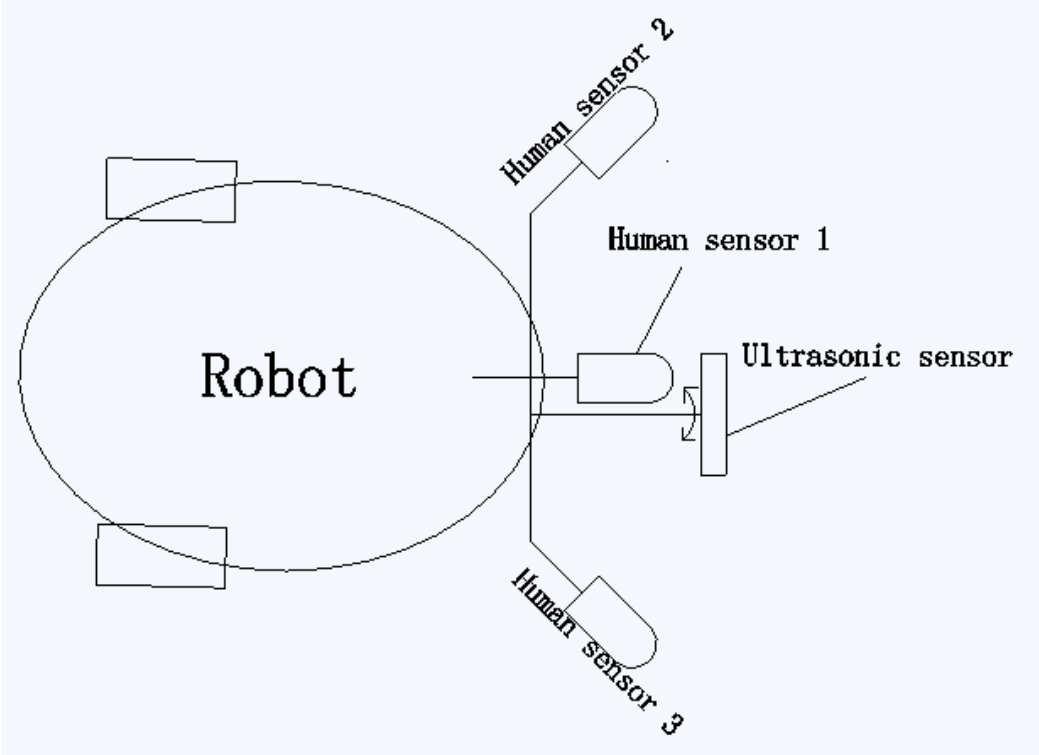


Figure 7.1 Robot layout plan

As we can see from the robot layout plan above, there are three human sensors and one ultrasonic sensor in front of the robot. Basically, when the robot is running, the four sensors keep receiving the signals about the humans and the obstacles from different directions. Once the robot receives the information form the sensors, the motors on the robot will response accordingly to avoid threatens. The flowcharts below show the meanings of the LabVIEW code.

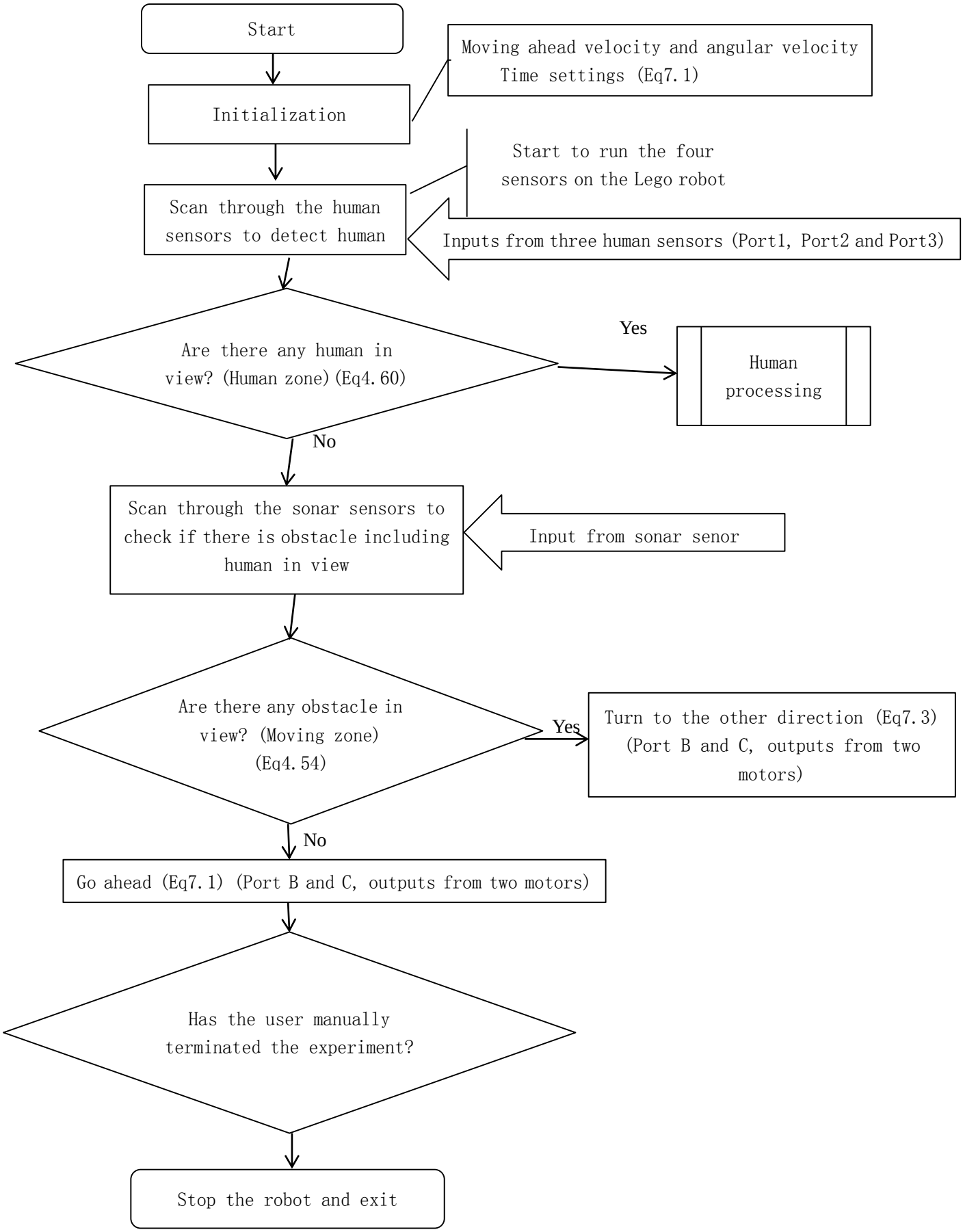


Figure 7. 2 The main function

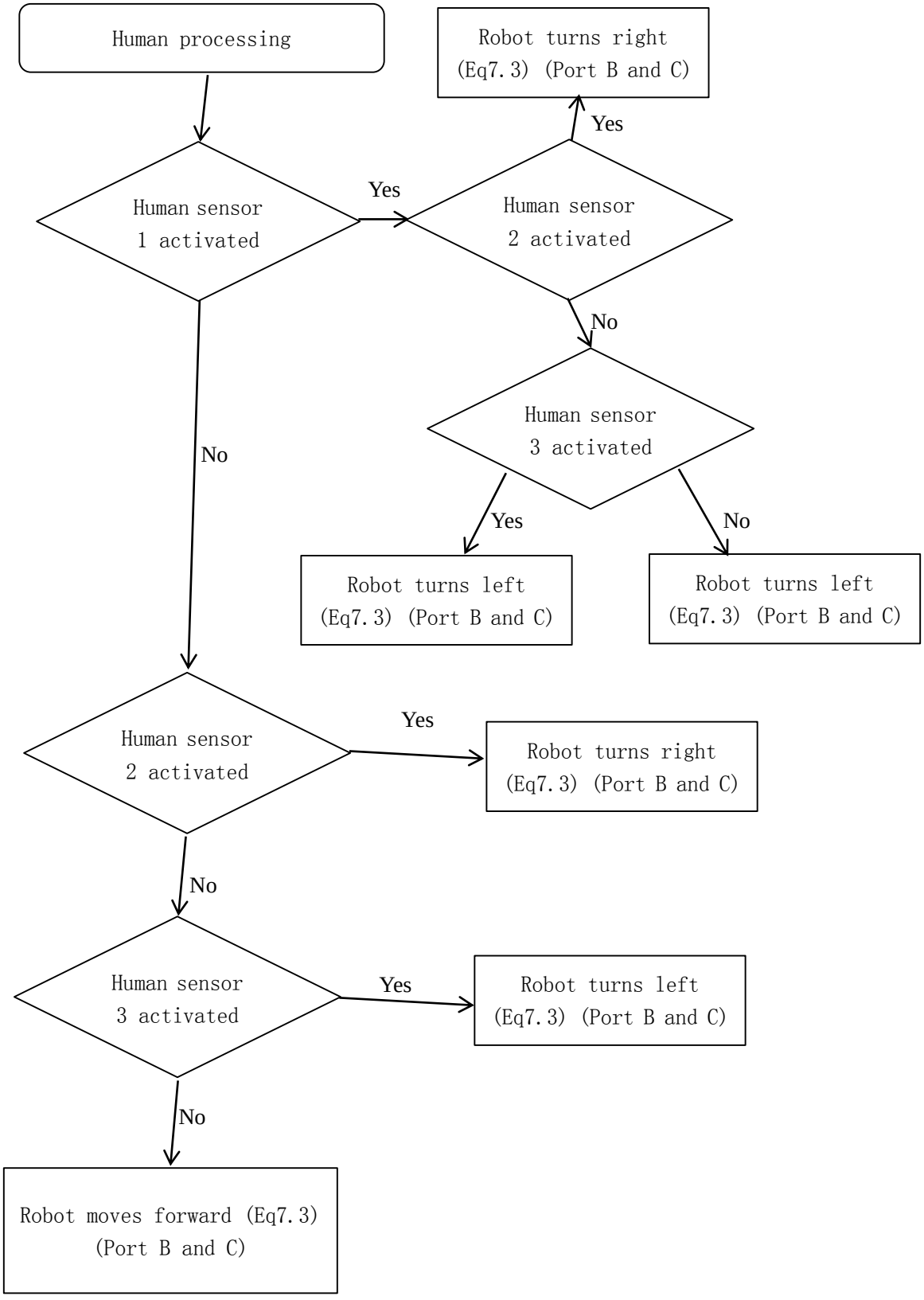


Figure 7. 3 Human processing

As the main function shows, the robot will check the human sensors first then sonar sensors. With this order, we can guarantee the Lego robot will put the human avoidance at its first priority when facing an obstacle and a human at same time.

In the human processing part, we set two conditions.

1. If two human sensors on the robot detect the human at same time, it means that the robot is facing at least two human at same time. The robot will go to the direction without human.

2. If the human sensor on the left side of our robot detect human while the human sensor on the right side does not, the robot will turn right and vice versa.

7.2 Experimental Results

The experiments were designed to be the same as for the simulation part in Chapter 6.

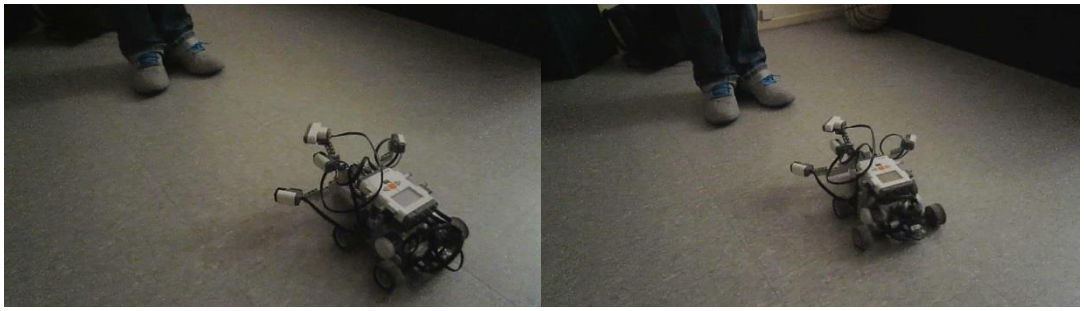
7.2.1 Mobile Robot Avoiding one Obstacle, one Stationary Human and one Moving Human

As done for the simulation, in experiments we first tested the capability of our robot in terms of avoiding the human and the obstacle. We started with some simple tasks, such as avoidance of one obstacle, one stationary human and one moving human as figures below show.

(a) (b)

(c) (d)

Figure 7.4 Obstacle avoidance

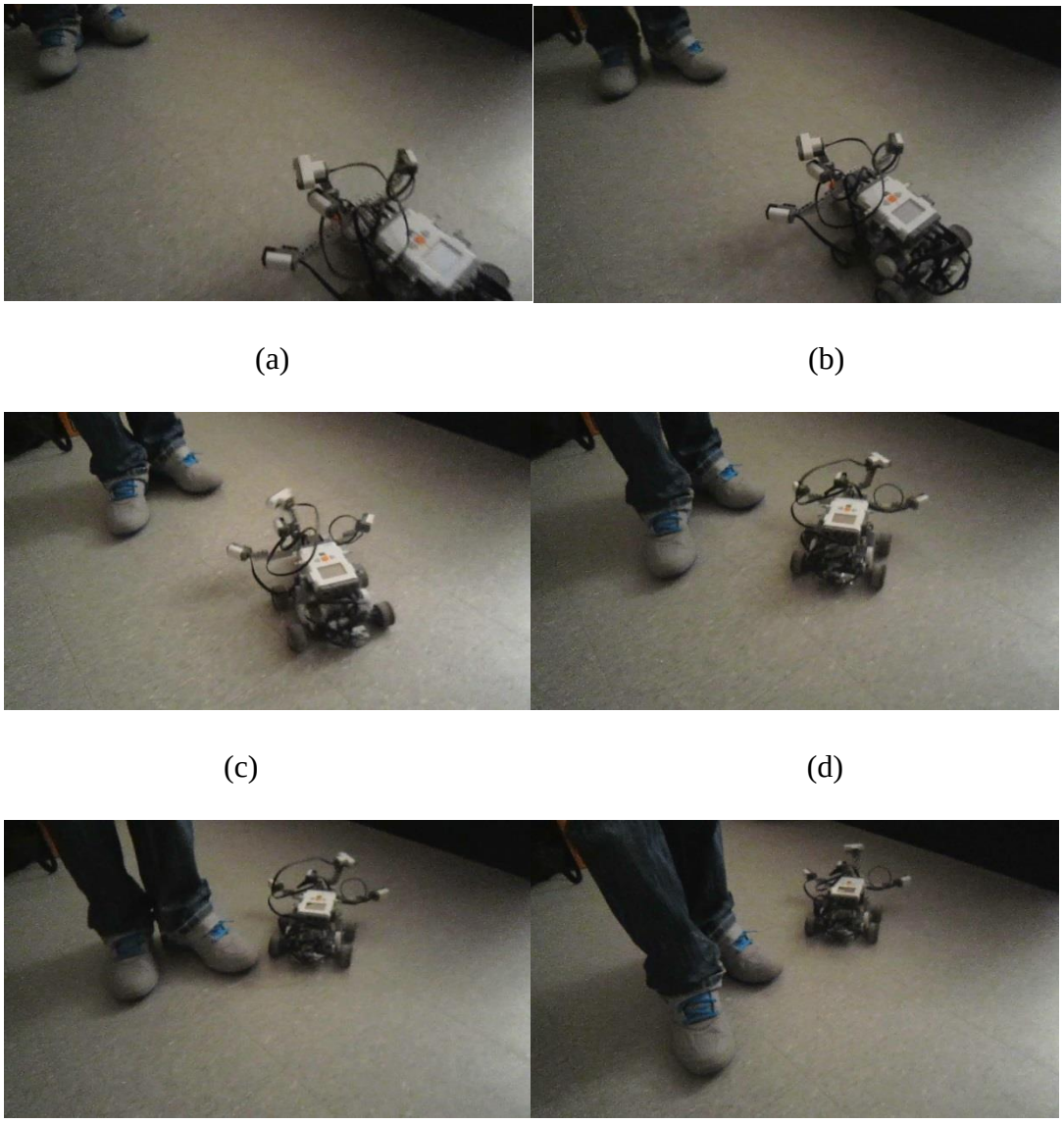


(a) (b)

(b) (d)

(e)

Figure 7.5 Stationary human avoidance



(a) (b)

(c) (d)

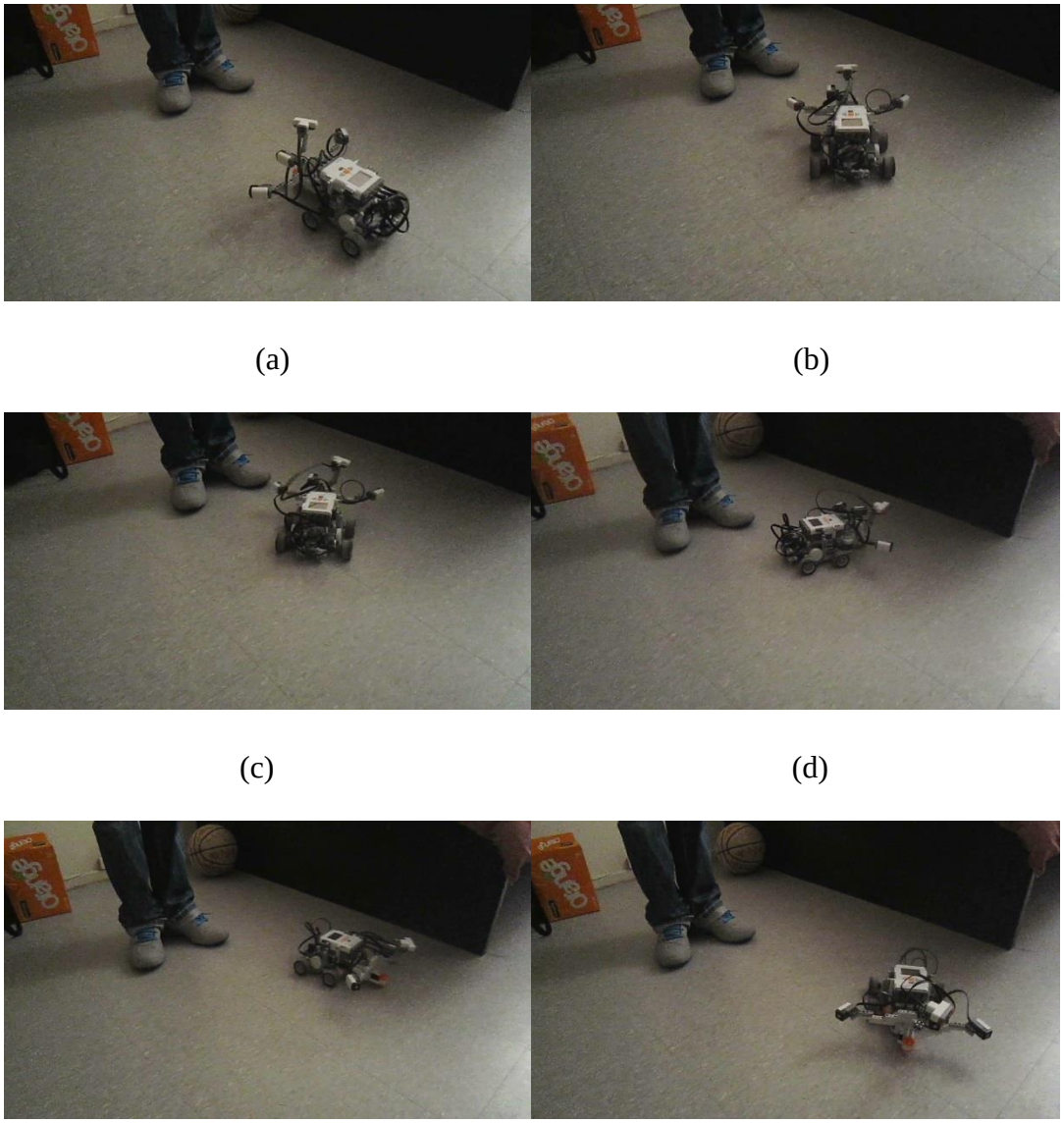
(e) (f)

Figure 7.6 Moving human avoidance

From the three above experimental results, we can see that the robot is able to detect both human and obstacle and use the information collected by human and sonar sensors to react and avoid the dangerous situation.

7.2.2 Mobile Robot Avoiding both Human and Obstacle at the same Time

This time, we brought one human and one obstacle in the environment and let our robot face them at the same time to test if our robot is able to avoid different kinds of dangers in the same time.



(a) (b)

(c) (d)

(e) (f)

Figure 7.7 Human and obstacle avoidance

As the result shows, our robot avoided the human and obstacle successfully.

7.2.3Mobile Robot Performs Task with Higher Human Priority

Three extreme situations are considered to test if the robot can place first human priority.

In these cases, the wall in the right side of the robot can be treated as an obstacle while the human appears at its right and front side.

7.2.3 Higher Priority of Human Collision Avoiding with the Risk of Colliding with the Obstacle

7.2.3.1 One Human and one Obstacle

For the first case, the robot moved along the wall (right side of the robot) while a human appeared at the robot's left side. The purpose to design this situation is to test if the robot is able to avoid human even in the case of risk to collide with the obstacle.

(a) (b)

(c) (d)

(e) (f)

(g)

Figure 7.8 One human and one obstacle avoidance (Higher human priority)

As we can see from the figures above, once the robot detected the human, it started to turn right to detour the human, which means it took risk to collide with the wall to avoid the human. Finally, the robot kept a safe distance with human and took a risk to collide with wall. Compared with the human, the robot was much closer to the obstacle and still avoided human, which means it put human as its first priority.

7.2.3.2 Two Humans and one Obstacle

In this case two humans kept approaching the robot located in front and left side and the wall was in its right side but not quite close. The purpose to design this situation is to test if the robot is able to avoid human even has a risk to collide with obstacle while it still has time and space.

(a) (b)

(c) (d)

(e) (f)

(g) (h)

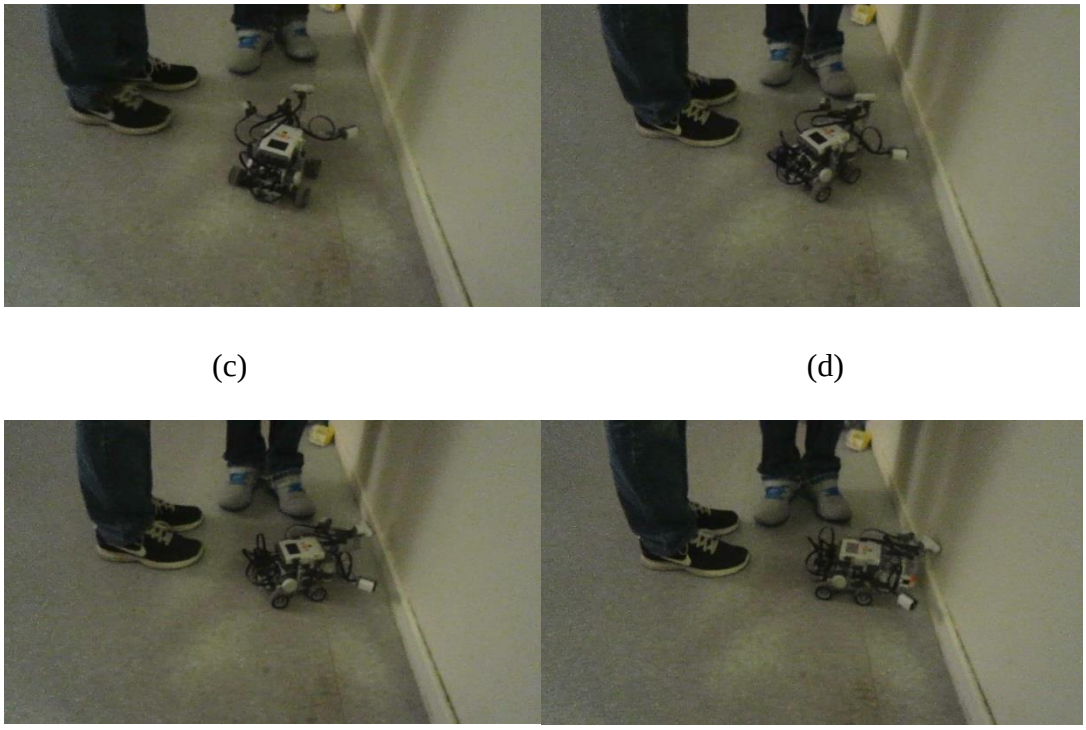
Figure 7.9 Two human and one obstacle avoidance (Higher human priority)

As the results show, the robot is able to run away from the dangerous situation as long as it has enough space. It first avoided human then turned around and ran away from all the dangers in an opposite direction, avoiding also the collision with the obstacle.

7.2.3.3 Two Humans and an Obstacle at a very Short Distance

Finally, in this experiment the human kept approaching from the front and left sides of the robot while the wall was on the left side of the robot and very close to the robot. In other words, the robot is in a situation that two humans and the wall were all very close to one another and the robot, being non-holonomic, could not avoid collision with one of them, if not stopping (given the gains chosen for the repulsive velocity commands which were low).

(a) (b)



(c) (d)

(e) (f)

Figure 7.10 Two human and one obstacle avoidance at a very short distance (Higher human priority)

As we can see from the Figure 7.9, the robot controller chose the wall to collide and avoided the human at same time, which means for the robot, human was considered more important than the obstacles.

7.3 Comparison between Simulation and Experimental Result

As we can see by comparing experimental results from this chapter with the simulation results from Chapter 6, the experimental results match the simulation

results. The robot is able to avoid different kinds of obstacles (including human) and always put human as its first priority during the avoidance.

The only difference is that in simulations we used a goal to lead the robot approach the obstacles and human while in experiment, because of the limitation of our equipment, we just designed the robot go ahead before it detected anything. However, the purpose is same, to lead it into different kinds of situations to test its controller. Moreover, given that the simulations are easier to be executed, more simulation results are presented than experimental results.

Chapter 8

Conclusion

The purpose of our project is to design a control system for mobile robot which put human safety as its first priority. In order to achieve this target, we need to solve several questions. The first problem is how to let our robot distinguish human and normal obstacles. We used the human sensor which is also called Passive Infrared Sensor to detect the human. Then, to improve the accuracy of the human sensor, the ultrasonic sensor was used to verify the information collected by human sensor to make sure the human position together. A sensor fusion method called Dempster-Shafer theory was used during this process. For other obstacles, we used the Ultrasonic Range Sensor to tell their positions and angles. After our robot sensed the environment, next problem is when and where our robot needs to start to avoid the different obstacles. We created several regions for the robot to decide the time to start the avoidance based on different obstacles. Finally, we redeveloped the velocity potential navigation method to let the robot achieve the final goal.

Simulations for testing the sensor fusion reliability and human safety of our robot were carried out in Matlab. In the simulation, we designed different situations to test the navigation algorithm of our robot step by step. Simulation results confirmed that this controller for a non-holonomic robot results in avoiding local minima, avoiding collision with fixed obstacles and humans as well as moving obstacles and humans

even for multiple case. Furthermore, the results of simulation for a crowded environment with two humans and a fixed obstacle showed that the proposed approach is limited in performance when low gains were used for repulsive velocity commands. Successful operation in this case can be achieved by proper design of the controller resulting in higher gains for the repulsive velocity commands.

After that, we rewrote our navigation algorithms into G codes and tested them in Lego robot in the real environment. The experimental results confirmed the performance of the controller and validity of the experimental results.

8.1 Research Contribution

There are 3 main contributions to the field of mobile robot. First of all, an improved velocity potential field navigation method was designed to overcome complicated situations such as unpredictable human movement and local minima of Artificial Potential Field method. The new algorithm of our robot is friendly to human, which means it will always put human safety as its first priority. When the robot faces human and obstacle at the same time, it will choose human to avoid at first. Secondly, a novel method that calculated the critical distance between robot and obstacle that robot need to start to avoid the obstacle was introduced. Based on different objects such as human, stationary obstacles and moving obstacles, we designed different regions for the robot. The robot will keep approaching its target until there are objects

intrude in their relative regions. The method was created based on the worst avoidance conditions and considered the shapes, velocity and acceleration limits of both robot and obstacles. By applying this method, our robot will achieve the target with less time and energy while still controlling the risk at a low level. Lastly, method of sensor fusion which used Dempster-Shafer theory was introduced and tested. By adding in the sensor fusion process, the robot is able to further confirm the human existence, which reduces the risk of collide with a human.

8.2 Future Work

There is still room to improve in terms of mobile robot's human safety. Strictly, in our project, our robot only has one kind of sensor-human sensor to distinguish human. The ultrasonic sensor just helps the human sensor but cannot distinguish human on its own. Although this method is good enough for human detection, it cannot ensure 100% security for human technically. The next step of our research is to add another sensor-thermal infrared camera to our robot. Human movements will be further observed and even predicted to some extent by the robot through processing pictures captured by the thermal infrared camera. Thus, the human safety will be further improved. Besides, an alarm can be added on the mobile robot to warning human in some very dangerous situation, which can avoid the collision on the other hand.

In terms of sensor fusion, next step is to fuse the information between different robots. In other words, the fused information is not only collected by the sensors in one robot

but from the sensors in all the robots. So each robot will has a better understanding of the whole environment and works more efficiently.

References

[1] G. Benet, F. Blanes, J. E. Simó, and P. Pérez, "Using infrared sensors for distance measurement in mobile robots," Robotics and Autonomous Systems, Vol.40, No.4, pp. 255–266, September 2002.

[2] Tarek Mohammad, "Using Ultrasonic and Infrared Sensors for Distance Measurement," World Academy of Science, Engineering and Technology 27, Vol.3 pp. 293–298, 2009.

[3] Evsen Yanmaz and Hasan Guclu, "Stationary and Mobile Target Detection using Mobile Wireless Sensor Networks." INFOCOM IEEE Conference on Computer Communications Workshops, pp. 1-5, March 2010

[4] Ryan J. Kephart and Michael S. Braasch, "See-and-avoid comparison of performance in manned and remotely piloted aircraft," IEEE Aerospace and Electronic Systems Magazine, Vol. 25, No. 5, pp. 36–42, May 2010.

[5] Jian-Shuen Fang, Qi Hao, David J. Brady, Bob D. Guenther and Ken Y. Hsu, "Real-time human identification using a pyroelectric infrared detector array and hidden Markov models.," Optics Express, Vol. 14, No. 15, pp. 6643–6658, July 2006.

[6] Andreas Hartmann, "PIR detector for safety" Technology and Design, pp.35-37, April 2003.

[7] Antonio Fernández-Caballero, Jose Carlos Castillo, Javier Martínez-Cantos and Rafael Martínez-Tomás, "Optical flow or image subtraction in human detection from infrared camera on mobile robot," Robotics Autonomous Systems, Vol.58, No. 12, pp.1273–1281, December 2010.

[8] Guodong Feng, Xuemei Guo and Guoli Wang, "Infrared motion sensing system for human-following robots," Sensors and Actuators, Vol.185, pp. 1–7, October 2012.

[9] Hwang Tae-hyun, Joo In-Hak and Cho Seong-Ik, "Detection of Traffic Lights for Vision-Based Car," Lecture Notes in Computer Science, Vol.4319, pp.682–691, 2006.

[10] Gurkan Tuna, V. Cagri Gungor and Kayhan Gulez, "An autonomous wireless sensor network deployment system using mobile robots for human existence detection in case of disasters," Ad Hoc Networks, Vol.13. Part A, pp.54–68, February 2014.

[11] Yucong Lu, Lingqi Zeng and Gary M. Bone, "Multisensor System for Safer Human-Robot Interaction," 2005 IEEE International Conference on Robotics and Automation, pp.1767–1772, April 2005.

[12] Moshe Kam, Xiaoxun Zhu and Paul Kalata, "Sensor Fusion for Mobile Robot Navigation," Proceedings of the IEEE 1997, Vol.85, pp108-119, No.1, 1997.

[13] Hamid Teimoori and Andrey V. Savkin, “A biologically inspired method for robot navigation in a cluttered environment,” Robotica, Vol.28, No.05, pp. 637–648, August 2009.

[14] Carolos Livadas, John Lygeros and Nancy A. Lynch, “High-level modeling and analysis of the traffic alert and collision avoidance system (TCAS),” Proceedings of the IEEE, Vol. 88, No.7, pp. 926–948, July 2000.

[15] Jun Tani, “Model-based learning for mobile robot navigation from the dynamical systems perspective,” Systems, Man, and Cybernetics, Vol. 26, No.3, pp.421–436, January 1996.

[16] S. Vitabile, G. Pilato, F. Pullara and F. Sorbello, “A Navigation System For Vision-Guided Mobile Robots.” Image Analysis and Processing 1999, pp.566-571, September 1999

[17] Animesh Chakravarthy and Debasish Ghose, “Obstacle avoidance in a dynamic environment: a collision cone approach,” IEEE Transaction on Systems, Cybernetics - Part A: Systems and Humans, Vol.28, No.5, pp.562–574, September 1998.

[18] William E. Green and Paul Y. Oh, “Optic-Flow-Based Collision Avoidance,” Robotics & Automation Magazine IEEE, Vol. 15, No.1, pp.96–103, March 2008.

[19] Georgios Lidoris, Florian Rohrmuller, Dirk Wollherr and Martin Buss, "The Autonomous City Explorer (ACE) Project – Mobile Robot Navigation in Highly Populated Urban Environments," IEEE International Conference on Robotics and Automation, pp.1416–1422, May 2009.

[20] Gary M.Bone and Lingqi Zeng, "Collision avoidance for nonholonomic mobile robots among unpredictable dynamic obstacles including humans," IEEE Conference on Automation Science and Engineering (CASE), pp. 940–947, August 2010.

[21] Qidan Zhu, Yongjie Yan and Zhuoyi Xing, "Robot Path Planning Based on Artificial Potential Field Approach with Simulated Annealing," Sixth International Conference on Intelligent Systems Design and Applications, Vol.2, pp.622–627, October 2006.

[22] K. Madhava Krishna and Prem K Kalra, "Solving the local minima problem for a mobile robot by classification of spatio-temporal sensory sequences," Journal of Robotics Systems, Vol.17, No.10, pp.549–564, October 2000.

[23] Lawrence A. Klein, "Sensor and data fusion: a tool for information assessment and decision making", Spie.org, July 2004

[24] Daniel Pagac, Eduardo M. Nebot and Hugh Durrant-Whyte, "An evidential approach to map-building for autonomous vehicles", IEEE Transaction on Robotics and Automation, Vol.14, pp.623-629, August 2002

[25] Alberto Elfes and Larry Matthies, “Sensor integration for robot navigation: Combining sonar and stereo range data in a grid-based representation”, IEEE International Conference on Robotics and Automation, pp 727-733, April 1988

[26] Robin R. Murphy, “Dempster–Shafer Theory for Sensor Fusion in Autonomous Mobile Robots”, IEEE Transactions on Robotics and Automation, Vol.14, No.2, April 1998

[27] Robert W Fox, Alan T McDonald, and Philip J Pritchard. “Introduction to fluid mechanics.” Vol. 2, John Wiley & Sons New York, 2011.

[28] Pruner, Elisha. “Control of Self-Organizing and Geometric Formations”, A thesis presented for the degree of Master of Applied Science in engineering, University of Ottawa, 2013

[29] Lingqi Zeng, “Design and control of human friendly robots”, A thesis presented for the degree of Ph.D of Applied Science in engineering, McMaster University, 2010

[30] Howie M Choset. “Principles of robot motion: theory, algorithms, and implementations.” The MIT press, 2005.

[31] Philip E. Martin and Anthony P. Marshi. “Step length and frequency effects on ground reaction forces during walking”, Journal of Biomechanics, Vol.25, No.10, pp 1237-1239, October 1992

Appendix A

MATLAB: Human Friendly Mobile Robot Navigation Controller Code

A.1 Main Function

```
% Clear command window
clc
clf
clear

% Initial Robot Values
robot_pose = [15,18,1.57] ;
sensor_radius = 4 ;
max_vel = 4;
max_ang_vel = 1;

% Map
map_max = 70 ;
map_min = -10;

% Time settings
t_sim = 10 ;
T = 0.1 ;
t_steps = floor ( t_sim/T) ;

% Goal Position on the map
goal = [32 , 70 ] ;
goal_radius = sensor_radius ;


% Leader with a circular path
```

```
R = 20 ;
gamma = 0.1 ;

%global pose1 L t a1 b1 circle_arrowxcircle_arrowy
global t



% Movie
j = 0 ;



%bring two walking human in

pose1 = zeros( t_steps , 3 ) ;
pose2 = pose1;
L=1;


for t = 1 : t_steps

% Clear figure
clf

% Axis properties
xmax = map_max ;
xmin = map_min ;
ymax = map_max ;
ymin= -20 ;
axis ( [ xmin , xmax+2*goal_radius, ymin , ymax+2*goal_radius] ) ;
axisequal ; axis manual;gridon ;hold on;

%sensor fusion

%c=[0 0 0 0 0 0 0 0];

%d=0;

% Plot goal and the circle of interest around the goal
scatter(goal(1),goal(2),100 ,'d ','m','filled' ) ;

rectangle('Position',[goal(1)-goal_radius,goal(2)-goal_radius,2*goal_radius,2*goal_radius],'Curvature',
[1,1],'EdgeColor','m');
```

```
%bring the first walking human in
pose1 ( t , 1 ) = 10;
pose1 ( t , 2 ) = -L*t*0.1+25;
pose1( t , 3 ) = gamma*t;
[X1,Y1] = human_walking( pose1 (t, : ) ) ;
%Plot a circle around the walking human
a1=pose1(t,1)+2;
b1=pose1(t,2)+1;
[circle_arrowx,circle_arrowy]=scircle1(a1,b1,4);


%bring the second walking human in
pose2 ( t , 1 ) = L*t*0.1+18;
pose2 ( t , 2 ) = 30;
pose2( t , 3 ) = gamma*t;
[X2,Y2] = human_walking( pose2 (t, : ) ) ;
%Plot a circle around the walking human
a2=pose2(t,1)+2;
b2=pose2(t,2)+1;
[circle_arrowx1,circle_arrowy1]=scircle1(a2,b2,4);

holdon ;
fill (X1,Y1, 'b ' ) ;
plot(circle_arrowx,circle_arrowy,'r');

fill (X2,Y2, 'b ' ) ;
plot(circle_arrowx1,circle_arrowy1,'r');

holdoff ;



%global a1 b1 circle_arrowxcircle_arrowy;

% Plot the concave or convex obstacle
% Check if sensor readings intersect with an obstacle
% If an obstacle is in range
% Set obst_dist and obst_angle values
% If no obstacle is in view
% Set obst_dist=0

%Avoid human at first
```

```
[concave_convex_human]
=concavehuman(robot_pose,circle_arrowx,circle_arrowy,circle_arrowx1,circle_arrowy1);
  obst_dist1 = concave_convex_human(1) ;
  obst_angle1 = concave_convex_human(2) ;
  p1=concave_convex_human(3) ;
  p2=concave_convex_human(4) ;
  p3=concave_convex_human(5) ;
  p4=concave_convex_human(6) ;
  p5=concave_convex_human(7) ;
  p6=concave_convex_human(8) ;
  p7=concave_convex_human(9) ;
  p8=concave_convex_human(10) ;
  o1=concave_convex_human(11) ;
  o2=concave_convex_human(12) ;
  o3=concave_convex_human(13) ;
  o4=concave_convex_human(14) ;
  o5=concave_convex_human(15) ;
  o6=concave_convex_human(16) ;
  o7=concave_convex_human(17) ;
  o8=concave_convex_human(18) ;
  p=[p1 p2 p3 p4 p5 p6 p7 p8]
  o=[o1 o2 o3 o4 o5 o6 o7 o8]

%Avoid obstacle then
[concave_convex]
=concave(robot_pose,map_max,circle_arrowx,circle_arrowy,circle_arrowx1,circle_arrowy1);
obst_dist = concave_convex(1) ;
obst_angle = concave_convex(2) ;
  q1=concave_convex(3) ;
  q2=concave_convex(4) ;
  q3=concave_convex(5) ;
  q4=concave_convex(6) ;
  q5=concave_convex(7) ;
  q6=concave_convex(8) ;
  q7=concave_convex(9) ;
  q8=concave_convex(10) ;
  q11=concave_convex(11) ;
  q21=concave_convex(12) ;
  q31=concave_convex(13) ;
  q41=concave_convex(14) ;
  q51=concave_convex(15) ;
  q61=concave_convex(16) ;
  q71=concave_convex(17) ;
  q81=concave_convex(18) ;
```

```
q111=concave_convex(19) ;
  q211=concave_convex(20) ;
  q311=concave_convex(21) ;
  q411=concave_convex(22) ;
  q511=concave_convex(23) ;
  q611=concave_convex(24) ;
  q711=concave_convex(25) ;
  q811=concave_convex(26) ;


    q=[q1 q2 q3 q4 q5 q6 q7 q8];
    q10=[q11 q21 q31 q41 q51 q61 q71 q81];
    q20=[q111 q211 q311 q411 q511 q611 q711 q811];
%sensor fusion
[probability]=DS_fusion(p,o,q,q10,q20);
  P1=probability(1);
  P2=probability(2);
if P1>0.9 || P2>0.9

% Calculate the new robot pose
robot=[robot_pose(1),robot_pose(2),robot_pose(3)] ;
[new_pose_human]=obstacle_controller_human(robot,goal,max_vel,max_ang_vel,goal_radius,sensor_
radius,obst_dist1,obst_angle1,T) ;
robot_pose=[new_pose_human(1),new_pose_human(2),new_pose_human(3)] ;

else

% Calculate the new robot pose
robot=[robot_pose(1),robot_pose(2),robot_pose(3)] ;
[new_pose]=obstacle_controller(robot,goal,max_vel,max_ang_vel,goal_radius,sensor_radius,obst_dist,
obst_angle,T) ;
robot_pose=[new_pose(1),new_pose(2),new_pose(3)] ;

end

% Get the arrow shape coordinates ?use fill function to draw the shape
% Plot the animation
holdon ;
% Plot the robot and its sensors
[circle] = sensor_view(robot_pose, sensor_radius) ;
[X,Y]=robot_shape(robot_pose);
fill (X,Y ,'b');
```

```
%plot(robot_pose,robot_pose,'b');
holdoff ;

% obst_distR = sqrt (xR.^2 + yR.^2 ) ;
j = j + 1 ;
M( j ) = getframe ;

% Plot labels
xlabel ( ' x [m] ' ) ;
ylabel ( ' y [m] ' ) ;

end
% Create a movie and save i t as an AVI f i l e
movie2avi (M, ' arrow_animation .avi ' ) ;
```

A.2 Plot obstacles and check sonar sensors detection area

```
function [concave_convex]
=concave(robot_pose,map_max,circle_arrowx,circle_arrowy,circle_arrowx1,circle_arrowy1)

obst_radius = 4;
% Check to see if the robot sensors intersect with an obstacle
obst_dist = 0 ;
obst_angle = 0 ;

x_dist = map_max/ 2 ;
y_dist = map_max / 2 ;
ob_width = 5 ;
ob_height = 10 ;
ob_thickness = 2.5 ;


% Convex obstacle dimensions
xlimit = [ x_dist ,
x_dist+ob_thickness ,x_dist+ob_width+ob_thickness,x_dist+ob_width+2*ob_thickness] ;

ylimit = [ y_dist , y_dist-ob_height , y_dist+ob_height/2 ] ;
concaveX2 = xlimit( [ 1 2 2 1 1 ] ) ;
```

```
concaveY2 = ylimit( [ 2 2 3 3 2 ] ) ;


% Draw obstacles in the simulation
% Obstacles are colored black -> k
hold on;
fill ( concaveX2 , concaveY2 , ' k ' ) ;

hold off;


%sensor fusion
c2=0.05*ones(1,8);
d2=0;
c3=0.05*ones(1,8);
c4=c2;
d3=d2;
d4=d2;

forbeam_angle = (robot_pose(3)-pi/3):pi/12:(robot_pose(3)+pi/3)
x_ob = [robot_pose(1),robot_pose(1)+1.3*obst_radius*cos(beam_angle)] ;
y_ob = [robot_pose(2),robot_pose(2)+1.3*obst_radius*sin(beam_angle)] ;
[xT,yT]=polyxpoly(x_ob,y_ob,concaveX2,concaveY2) ;


% Plot a red circle where the intersection points occur
mapshow(xT,yT,'DisplayType','point','Marker','o') ;
obst_distT = sqrt (xT.^2 + yT.^2 ) ;
if ( size(obst_distT,1)==2)
obst_distT=obst_distT(1);
end


%sensor fusion
d2=d2+1;
if ( isempty(obst_distT)==false )
obst_dist=obst_distT ;
obst_angle =beam_angle ;
c2(1,d2)=0.95;
end


%    Pl=c2(1,:)
```

```
if(obst_angle>= 2*pi )
obst_angle=obst_angle -2*pi ;
elseif (obst_angle<0)
obst_angle=obst_angle+2*pi ;
end
end


q1=c2(1,1);
q2=c2(1,2);
q3=c2(1,3);
q4=c2(1,4);
q5=c2(1,5);
q6=c2(1,6);
q7=c2(1,7);
q8=c2(1,8);


forbeam_angle = (robot_pose(3)-pi/3):pi/12:(robot_pose(3)+pi/3)
x_ob = [robot_pose(1),robot_pose(1)+1.3*obst_radius*cos(beam_angle)] ;
y_ob = [robot_pose(2),robot_pose(2)+1.3*obst_radius*sin(beam_angle)] ;
[xT1,yT1]=polyxpoly(x_ob,y_ob,circle_arrowx,circle_arrowy) ;


% Plot a red circle where the intersection points occur
mapshow(xT1,yT1,'DisplayType','point','Marker','o') ;


obst_distT1 = sqrt (xT1.^2 + yT1.^2 ) ;


if ( size(obst_distT1,1)==2)
    obst_distT1=obst_distT1(1);
end


%sensor fusion
d3=d3+1;

if ( isempty(obst_distT1)==false )
obst_dist=obst_distT1 ;
obst_angle =beam_angle ;
c3(1,d3)=0.95;
end
```

```
%    Pl=c2(1,:)

if(obst_angle>= 2*pi )
obst_angle=obst_angle -2*pi ;
elseif (obst_angle<0)
obst_angle=obst_angle+2*pi ;
end
end


q11=c3(1,1);
q21=c3(1,2);
q31=c3(1,3);
q41=c3(1,4);
q51=c3(1,5);
q61=c3(1,6);
q71=c3(1,7);
q81=c3(1,8);


forbeam_angle = (robot_pose(3)-pi/3):pi/12:(robot_pose(3)+pi/3)
x_ob = [robot_pose(1),robot_pose(1)+1.3*obst_radius*cos(beam_angle)] ;
y_ob = [robot_pose(2),robot_pose(2)+1.3*obst_radius*sin(beam_angle)] ;
[xR,yR]=polyxpoly(x_ob,y_ob,circle_arrowx1,circle_arrowy1) ;


% Plot a red circle where the intersection points occur
mapshow(xR,yR,'DisplayType','point','Marker','o') ;
obst_distR = sqrt (xR.^2 + yR.^2 ) ;
if ( size(obst_distR,1)==2)
obst_distR=obst_distR(1);
end


%sensor fusion
d4=d4+1;
if ( isempty(obst_distR)==false )
obst_dist=obst_distR ;
obst_angle =beam_angle ;
c4(1,d4)=0.95;
end
%    Pl=c2(1,:)

if(obst_angle>= 2*pi )
```

```
obst_angle=obst_angle -2*pi ;
elseif (obst_angle<0)
obst_angle=obst_angle+2*pi ;
end
end

q111=c4(1,1);
q211=c4(1,2);
q311=c4(1,3);
q411=c4(1,4);
q511=c4(1,5);
q611=c4(1,6);
q711=c4(1,7);
q811=c4(1,8);


% Return obst_dist and obst_angle values to the main
concave_convex= [ obst_dist,
obst_angle,q1,q2,q3,q4,q5,q6,q7,q8,q11,q21,q31,q41,q51,q61,q71,q81,q111,q211,q311,q411,q511,q61
1,q711,q811] ;
```

A.3 Check human sensors detection area (human zone)

```
function [concave_convex_human]
=concavehuman(robot_pose,circle_arrowx,circle_arrowy,circle_arrowx1,circle_arrowy1)
obst_radius = 4;


% Check to see if the robot sensors intersect with an obstacle
obst_dist = 0 ;
obst_angle = 0 ;


%sensor fusion
c1=0.1*ones(1,8);
c2=0.1*ones(1,8);
d1=0;
d2=0;

forbeam_angle = (robot_pose(3)-pi/3):pi/12:(robot_pose(3)+pi/3)
x_ob = [robot_pose(1),robot_pose(1)+1.3*obst_radius*cos(beam_angle)] ;
```

```
y_ob = [robot_pose(2),robot_pose(2)+1.3*obst_radius*sin(beam_angle)] ;
[xA,yA]=polyxpoly(x_ob,y_ob,circle_arrowx1,circle_arrowy1);
mapshow(xA,yA,'DisplayType','point','Marker','o')
obst_distA = sqrt (xA.^2 + yA.^2 ) ;

if    ( size(obst_distA,1)==2)
obst_distA=obst_distA(1);
end


%sensor fusion
d1=d1+1;
if ( isempty(obst_distA)==false );
obst_dist=obst_distA;
obst_angle =beam_angle
c1(1,d1)=0.9;
end


%pk=c1(1,:)


if(obst_angle>= 2*pi )
obst_angle=obst_angle -2*pi ;
elseif (obst_angle<0)
obst_angle=obst_angle+2*pi ;
end
end
%end

  p1=c1(1,1);
  p2=c1(1,2);
  p3=c1(1,3);
  p4=c1(1,4);
  p5=c1(1,5);
  p6=c1(1,6);
  p7=c1(1,7);
  p8=c1(1,8);


forbeam_angle = (robot_pose(3)-pi/3):pi/12:(robot_pose(3)+pi/3)
x_ob = [robot_pose(1),robot_pose(1)+1.3*obst_radius*cos(beam_angle)] ;
y_ob = [robot_pose(2),robot_pose(2)+1.3*obst_radius*sin(beam_angle)] ;
[xD,yD]=polyxpoly(x_ob,y_ob,circle_arrowx,circle_arrowy);
```

```
mapshow(xD,yD,'DisplayType','point','Marker','o')
obst_distD = sqrt (xD.^2 + yD.^2 ) ;

if    ( size(obst_distD,1)==2)
obst_distD=obst_distD(1);
end


%sensor fusion
d2=d2+1;
if ( isempty(obst_distD)==false );
obst_dist=obst_distD;
obst_angle =beam_angle ;
c2(1,d2)=0.9;
end


%pk=c1(1,:)


if(obst_angle>= 2*pi )
obst_angle=obst_angle1-2*pi ;
elseif (obst_angle<0)
        obst_angle1=obst_angle+2*pi ;
end
end
%end


  o1=c2(1,1);
  o2=c2(1,2);
  o3=c2(1,3);
  o4=c2(1,4);
  o5=c2(1,5);
  o6=c2(1,6);
  o7=c2(1,7);
  o8=c2(1,8);


% Return obst_dist and obst_angle values to the main
concave_convex_human = [ obst_dist, obst_angle,p1,p2,p3,p4,p5,p6,p7,p8,o1,o2,o3,o4,o5,o6,o7,o8] ;
```

A.4 Combine the information which is collected by the human sensor and the sonar sensor

```
function [probability]=DS_fusion(p,o,q,q10,q20)

% ¹ÄÜ£ºÈÚ ºïk,yÁ½ÐÐ ÏòÁ¿(¾ ¬µäDempster-Shafer×é°Ï¹«Ê½)
% x,yµÄ ¸ñÊ½Ð ÎÈ¢m1 m2 m3, ... , mk, m(È«¼¯), m(¿Õ¼¯)]
% Ò ªÇóm1 m2 m3 ...Ö®¼ ä»¥ÏàÎÞ½»¼¯
% m(È«¼¯) ¿É ³»Î0 £¬± íÊ¾ ³»È ·¶¨¶È
% m(¿Õ¼¯) ¿Ï¶¨ÊÇ0

%[n1,m1]=size(pk);
%[n2,m2]=size(pl);
%{
if 1~=nx
disp('xÓ ¦ÎÐÐ ÏòÁ¿');
return;
end

[ny,my]=size(pl);
if 1~=ny
disp('yÓ ¦ÎÐÐ ÏòÁ¿');
return;
end

if mx~=my
disp('x,yÁÐÊýÓ ¦ÏàµÈ');
return;
end
%}
%if m1==m2

q_combine1=q+q10+q20-2*[0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05]
%    q_combine2=q+q20-[0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05];

temp=p*q_combine1';

a=p;
a(a<0.6)=0;
b=q_combine1;
b(b<0.6)=0;

  temp1=a*b';
```

```
P1=temp1

temp2=o*q_combine1';

c=o;
c(c<0.6)=0;
d=q_combine1;
d(d<0.6)=0;

   temp3=c*d';
P2=temp3

probability=[P1,P2];
```

A.5 Plot human body

```
function [X1,Y1] = human_walking(Q)
x = Q( 1 ) ;
y = Q( 2 ) ;
theta = Q( 3 ) ;

l = 1 ;
X1 = [ x , x+4 , x+4 ,x , x ] ;
Y1 = [ y , y , y+2 ,y+2 , y ] ;
```

A.6 Plot Robot

```
function [X,Y] =robot_shape(Q)
x = Q( 1 ) ;
y = Q( 2 ) ;
theta = Q( 3 ) ;

l = 1 ;
X =
[x+l*cos(theta+pi/4),x+l*cos(theta+3*pi/4),x+l*cos(theta+0.922*pi),x+l*cos(theta+1.078*pi),x+l*cos(
theta+5*pi/4),x+l*cos(theta+7*pi/4),x+l*cos(theta+pi/4)];

Y =
[y+l*sin(theta+pi/4),y+l*sin(theta+3*pi/4),y+l*sin(theta+0.922*pi),y+l*sin(theta+1.078*pi),y+l*sin(th
```

```
eta+5*pi/4),y+l*sin(theta+7*pi/4),y+l*sin(theta+pi/4)] ;
```

A.7 Plot the human sensor and the sonar sensor detection areas

```
function [circle] = sensor_view(Q, radius )
x = Q(1);
y = Q(2);
theta = Q(3);

% Plot beam
forbeam_angle = (theta - pi /2) : pi /12 : ( theta + pi /2) ;
beamX = [ x , x + radius * cos(beam_angle) ] ;
beamY = [ y , y + radius * sin(beam_angle) ] ;
plot (beamX, beamY, 'c' ) ;
end


% Plot circle
circle = rectangle ( 'Position',....
[ x-radius, y-radius , 2 * radius , 2 * radius ], ...
'Curvature',[1,1],'EdgeColor','c') ;


%plot beam2
forbeam_angle_pir = (theta - pi/3) : pi /12 : ( theta + pi/3) ;
      beamX1 = [ x , x + 1.3*radius * cos(beam_angle_pir) ] ;
      beamY1 = [ y , y + 1.3*radius * sin(beam_angle_pir) ] ;
plot (beamX1, beamY1, 'm' ) ;
end


%plot circle2
for circle= rectangle('Position',[x-1.3*radius, y-1.3*radius, 2.6*radius ,
2.6*radius ],'Curvature',[1,1],'EdgeColor','w');
end
```

A.8 Apply collision avoidance controller for obstacle

```
function [new_position] =
obstacle_controller(robot,goal,max_vel,max_ang_vel,goal_radius,obst_radius,obst_dist,obst_angle,T)


% Constants
k_a = 2 ;
k_n = 0.5 ;
k_t = 0.5 ;


% Calculate the x and y components of the distance to goal
delta_x = goal(1)-robot(1);
delta_y = goal(2)-robot(2);


% Calculate the distance to the goal
dist_goal =sqrt(delta_x^2+delta_y^2);


% Calculate angle to goal
ifdelta_y>0 &&delta_x>0
theta =atan(delta_y /delta_x ) ;
elseifdelta_y<0 &&delta_x>0
theta =atan(delta_y /delta_x ) ;
elseifdelta_y>0 &&delta_x<0
theta =atan(delta_y /delta_x )+pi ;
elseifdelta_y<0 &&delta_x<0
theta =atan(delta_y /delta_x )-pi ;
end


% Calculate the change in the angle between the current and goal position
delta_theta = theta-robot(3);


% Calculate the distance relation
dist_relation = exp(-dist_goal/goal_radius ) ;


% Calculate attractive velocity
u_a = k_a * max_vel*(1-dist_relation ) ;
```

```
% Find new theta value
close_to_zero = 0.2 ;
if ( abs(delta_theta)<close_to_zero )
theta = theta ;
elseif ( delta_theta>= 0 && abs(delta_theta)<pi)
theta = robot(3)+max_ang_vel*T;
elseif ( delta_theta>= 0 && abs(delta_theta)>= pi )
theta = robot(3) - max_ang_vel*T;
elseif ( delta_theta< 0 && abs(delta_theta)<pi )
theta = robot(3)- max_ang_vel*T;
elseif (delta_theta<0 && abs(delta_theta)>=pi )
theta = robot(3)+max_ang_vel*T;
end


% Calculate repulsive velocity
% If no obstacle is in view
% set normal and tangent values to zero
% If an obstacle is in view
% calculate normal and tangent values
if (obst_dist==0)
normal=0 ;
tangent=0 ;
obst_relation = 0 ;
normal_angle = 0 ;
rotation_angle = 0 ;
theta_obst = 0 ;
else
normal =1/obst_dist ;
tangent =1/obst_dist ;
obst_relation = exp(-obst_dist/obst_radius ) ;


% Calculate normal angle
normal_angle = obst_angle + pi ;
if ( normal_angle> 2*pi)
normal_angle = normal_angle-2*pi ;
end

% Calculate tangent angle
%if pi/2<normal_angle<3*pi/2
rotation_angle = normal_angle-pi/15;
%elseif 0<normal_angle<=pi/2 || 3*pi/2<=normal_angle<=2*pi
```

```
% tangent_angle=normal_angle+pi/9;
%end


% tangent_angle = normal_angle-pi/2;
%if ( rotation_angle>3*pi/2)
%    rotation_angle = rotation_angle -2*pi ;
% end


% Calculate obstacle angle
theta_obst = (normal_angle+rotation_angle)/2 ;
if ( theta_obst<0)
theta_obst = theta_obst+2*pi ;
end


% Calculate change in angle between the robot and obstacle
delta_theta_obst = theta_obst -robot(3);
if ( abs(delta_theta_obst)<close_to_zero)
theta_obst=theta_obst ;
elseif(delta_theta_obst>=0 && abs(delta_theta_obst)<pi )
theta_obst = robot(3)+max_ang_vel*T;
elseif (delta_theta_obst>=0 && abs(delta_theta_obst)>=pi )
theta_obst = robot(3)-max_ang_vel*T;
elseif ( delta_theta_obst<0 && abs(delta_theta_obst)<pi )
theta_obst = robot(3)-max_ang_vel*T;
elseif (delta_theta_obst<0&& abs(delta_theta_obst)>=pi )
theta_obst = robot(3) + max_ang_vel*T;
end


% Ignore goal while obstacle is in view
u_a = 0 ;
theta = 0 ;
end


% Calculate normal and tangential velocity components
u_n = k_n*normal*obst_relation ;
u_t = k_t*tangent*obst_relation ;


% Calculate the sum of all velocity components
```

```
u_totX = u_a*cos(theta)+u_n*cos(normal_angle)+u_t*cos(rotation_angle) ;
u_totY = u_a *sin(theta)+u_n*sin(normal_angle)+ u_t*sin(rotation_angle) ;


% Calculate the change in x and y distance for the current time step
delta_d=[u_totX*T,u_totY*T] ;
delta_theta = theta+theta_obst ;


% Send the robot 's next position to the main function
new_position = [robot(1)+delta_d(1),robot(2)+delta_d(2),delta_theta,u_totX,u_totY] ;
```

A.9 Apply collision avoidance controller for human

```
function [new_position_human] =
obstacle_controller_human(robot,goal,max_vel,max_ang_vel,goal_radius,obst_radius,obst_dist1,obst_
angle1,T)


% Constants
k_a = 2 ;
k_n = 0.5 ;
k_t = 0.5 ;


% Calculate the x and y components of the distance to goal
delta_x = goal(1)-robot(1);
delta_y = goal(2)-robot(2);


% Calculate the distance to the goal
dist_goal =sqrt(delta_x^2+delta_y^2);


% Calculate angle to goal
ifdelta_y>0 &&delta_x>0
theta =atan(delta_y /delta_x ) ;
elseifdelta_y<0 &&delta_x>0
theta =atan(delta_y /delta_x ) ;
elseifdelta_y>0 &&delta_x<0
theta =atan(delta_y /delta_x )+pi ;
```

```
elseifdelta_y<0 &&delta_x<0
theta =atan(delta_y /delta_x )-pi ;
end


% Calculate the change in the angle between the current and goal position
delta_theta = theta-robot(3);


% Calculate the distance relation
dist_relation = exp(-dist_goal/goal_radius ) ;


% Calculate attractive velocity
u_a = k_a * max_vel*(1-dist_relation ) ;


% Find new theta value
close_to_zero = 0.2 ;
if ( abs(delta_theta)<close_to_zero )
theta = theta ;
elseif ( delta_theta>= 0 && abs(delta_theta)<pi)
theta = robot(3)+max_ang_vel*T;
elseif ( delta_theta>= 0 && abs(delta_theta)>= pi )
theta = robot(3) - max_ang_vel*T;
elseif ( delta_theta< 0 && abs(delta_theta)<pi )
theta = robot(3)- max_ang_vel*T;
elseif (delta_theta<0 && abs(delta_theta)>=pi )
theta = robot(3)+max_ang_vel*T;
end

% Calculate repulsive velocity
% If no obstacle is in view
% set normal and tangent values to zero
% If an obstacle is in view
% calculate normal and tangent values
if (obst_dist1==0)
normal=0 ;
tangent=0 ;
obst_relation = 0 ;
normal_angle = 0 ;
rotation_angle = 0 ;
theta_obst = 0 ;
```

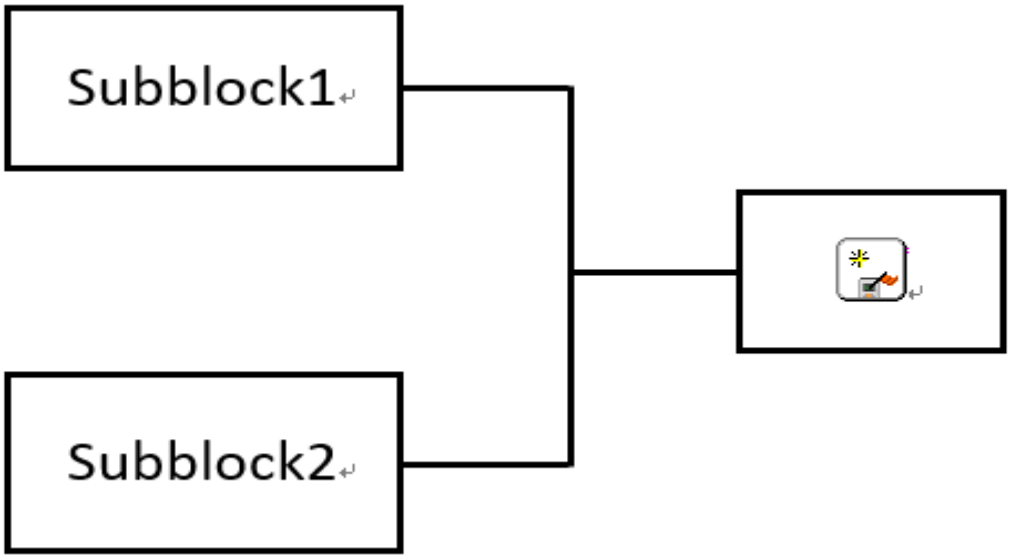
```
else
normal =1/obst_dist1 ;
tangent =1/obst_dist1 ;
obst_relation = exp(-obst_dist1/obst_radius ) ;


% Calculate normal angle
normal_angle = obst_angle1 + pi ;
if ( normal_angle> 2*pi)
normal_angle = normal_angle-2*pi ;
end


% Calculate tangent angle
%if pi/2<normal_angle<3*pi/2
rotation_angle = normal_angle-pi/15;
%elseif 0<normal_angle<=pi/2 || 3*pi/2<=normal_angle<=2*pi
% tangent_angle=normal_angle+pi/9;
%end


% tangent_angle = normal_angle-pi/2;
%if ( rotation_angle>3*pi/2)
%    rotation_angle = rotation_angle -2*pi ;
% end


% Calculate obstacle angle
theta_obst = (normal_angle+rotation_angle)/2 ;
if ( theta_obst<0)
theta_obst = theta_obst+2*pi ;
end


% Calculate change in angle between the robot and obstacle
delta_theta_obst = theta_obst -robot(3);
if ( abs(delta_theta_obst)<close_to_zero)
theta_obst=theta_obst ;
elseif(delta_theta_obst>=0 && abs(delta_theta_obst)<pi )
theta_obst = robot(3)+max_ang_vel*T;
elseif (delta_theta_obst>=0 && abs(delta_theta_obst)>=pi )
theta_obst = robot(3)-max_ang_vel*T;
elseif ( delta_theta_obst<0 && abs(delta_theta_obst)<pi )
theta_obst = robot(3)-max_ang_vel*T;
```

```
elseif (delta_theta_obst<0&& abs(delta_theta_obst)>=pi )
theta_obst = robot(3) + max_ang_vel*T;
end


% Ignore goal while obstacle is in view
u_a = 0 ;
theta = 0 ;
end


% Calculate normal and tangential velocity components
u_n = k_n*normal*obst_relation ;
u_t = k_t*tangent*obst_relation ;

% Calculate the sum of all velocity components
u_totX = u_a*cos(theta)+u_n*cos(normal_angle)+u_t*cos(rotation_angle) ;
u_totY = u_a *sin(theta)+u_n*sin(normal_angle)+ u_t*sin(rotation_angle) ;


% Calculate the change in x and y distance for the current time step
delta_d=[u_totX*T,u_totY*T] ;
delta_theta = theta+theta_obst ;


% Send the robot 's next position to the main function
new_position_human = [robot(1)+delta_d(1),robot(2)+delta_d(2),delta_theta,u_totX,u_totY] ;
```

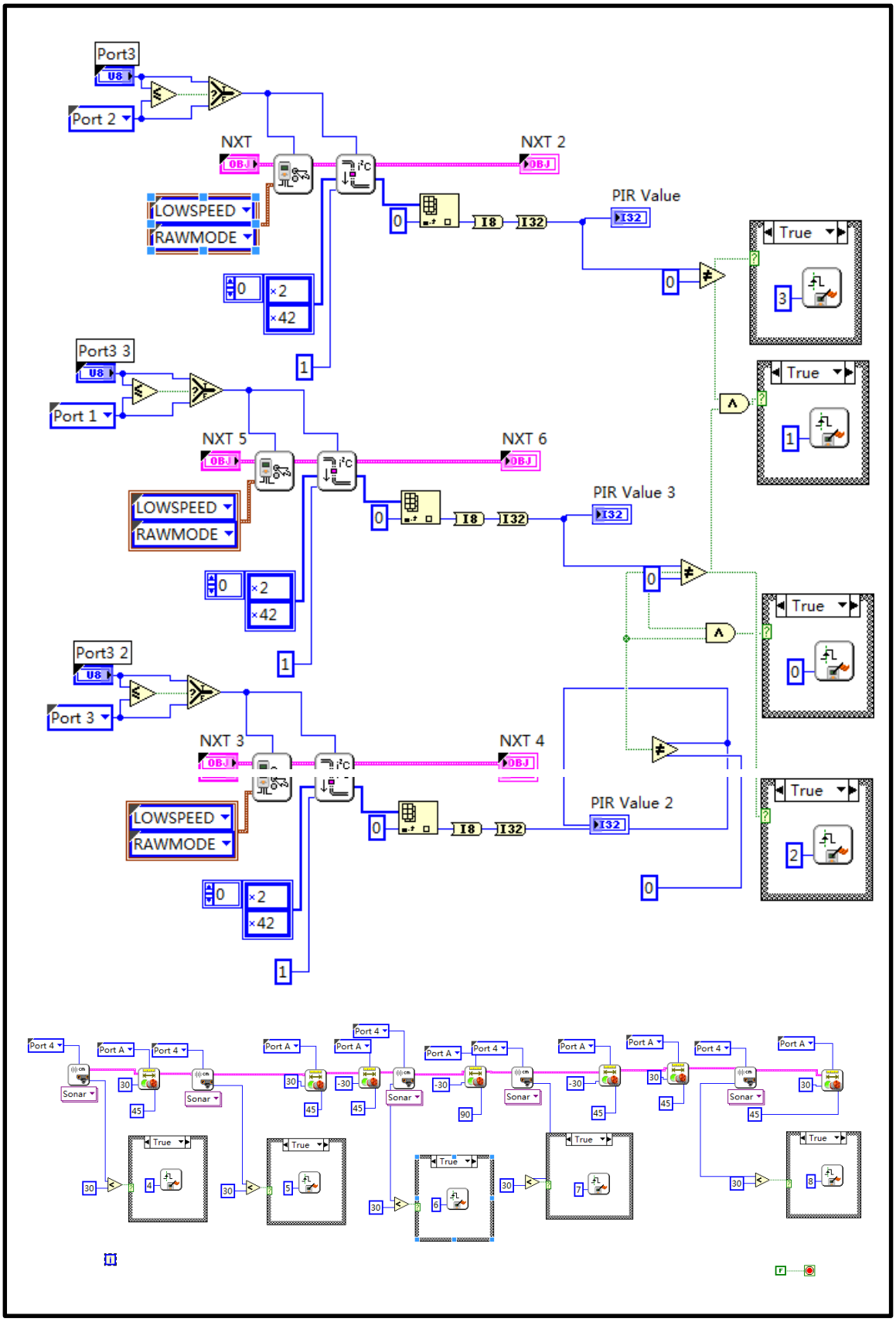
Appendix B

B.1 LabVIEW: Human Friendly Mobile Robot Navigation Controller Code

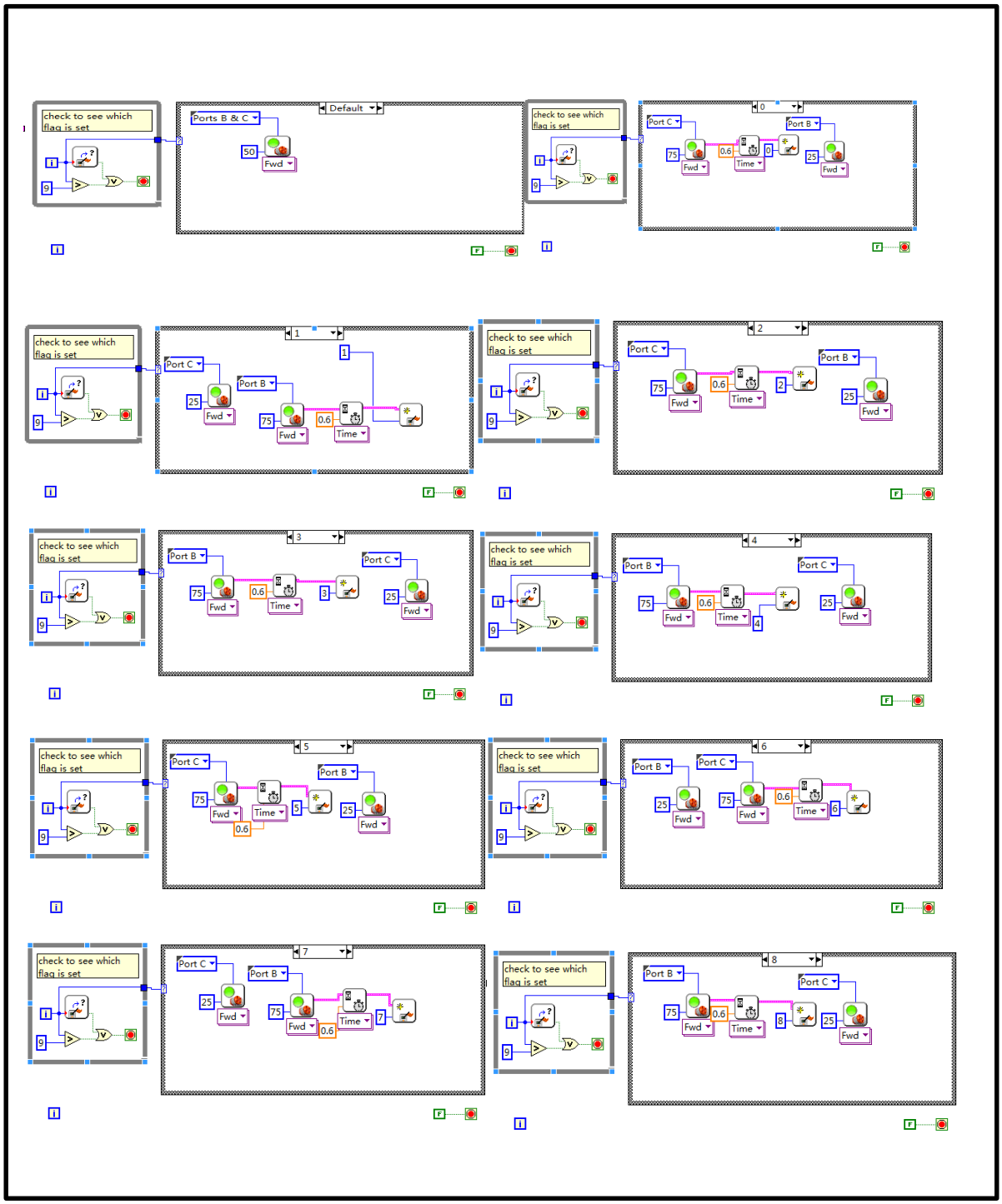
Main function



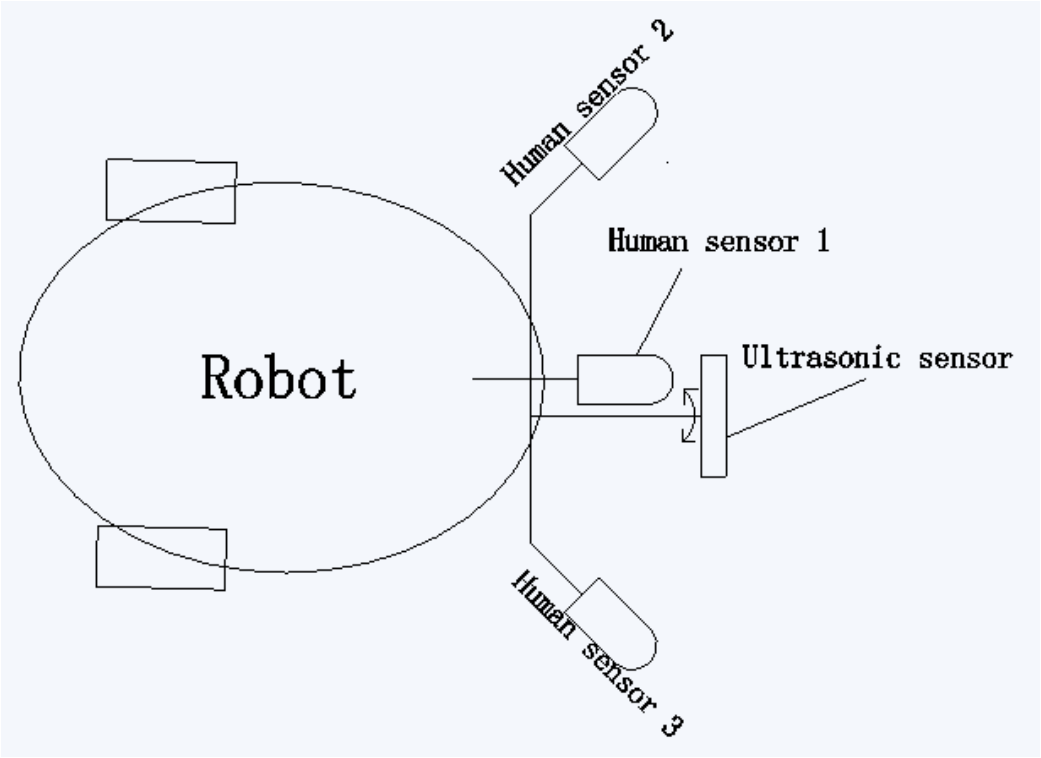
Subblock1


Port3
U8
Port 2
NXT
OBJ
NXT 2
LOWSPEED
RAWMODE
0
I8
I32
PIR Value
True
×2
×42
Port3 3
Port 1
NXT 5
NXT 6
PIR Value 3
Port3 2
Port 3
NXT 3
NXT 4
PIR Value 2
Port 4
Port A
Sonar
30
45
-30
90

Subblock2

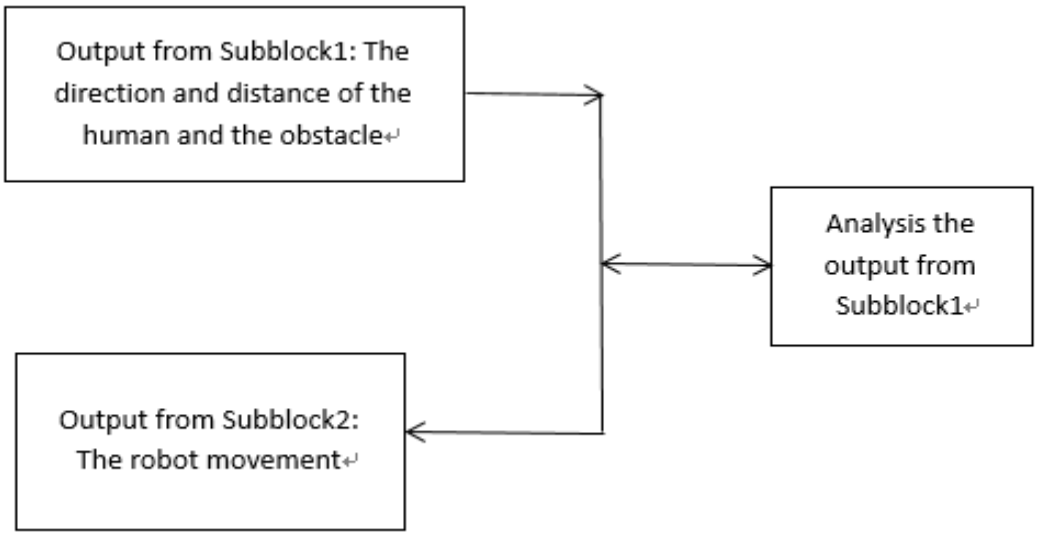

check to see which flag is set
Default
Ports B & C
50
Fwd
Port C
Port B
75
0.6
Time
25
0
1
2
3
4
5
6
7
8

B.2 The Explanation of LabVIEW Code:

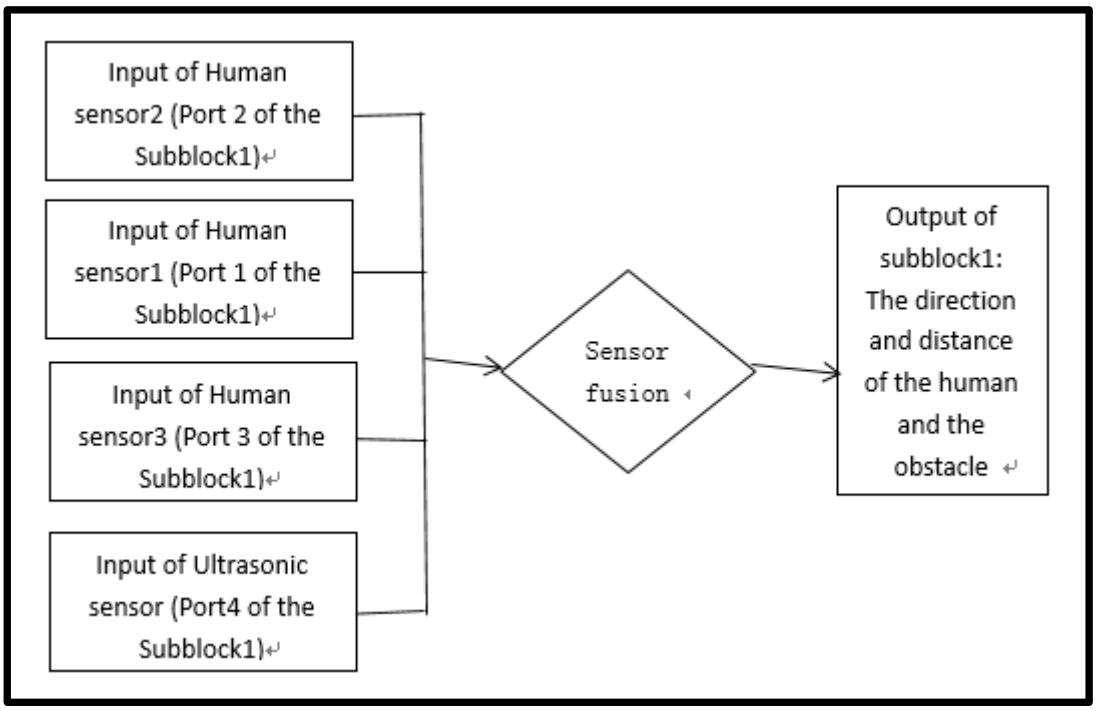


As we can see from the robot layout plan above, there are three human sensors and one ultrasonic sensor in front of the robot. Basically, when the robot is running, the four sensors keep receiving the signal about the humans and the obstacles from different directions. Once the robot receives the information form the sensors, the motors on the robot will response accordingly to avoid threatens. The flowcharts below show the meanings of the LabVIEW code.

Main function



Subblock1 (Combine the observation of different sensors to find the directions and distances of the human and the obstacles)



Subblock2 (Give the robot movement commands case by case)

For case 1, if the combined output of human sensor2 and ultrasonic sensor (at a 45 degree angle) is high, the human is on the left side of the robot. So give the robot a command to turn right through Port B and C.	For case 2, if the combined output of human sensor1 and ultrasonic sensor (at a 0 degree angle) is high, the human is in front of the robot. So give the robot a command to turn right through Port B and C.
For case 3, if the combined output of human sensor3 and ultrasonic sensor (at a -45 degree angle) is high, the human is on the right side of the robot. So give the robot a command to turn left through Port B and C.	For case 4, if the combined output of human sensor2, human sensor1 and ultrasonic sensor (at a 0 or 45 degree angle) is high, the human is on the right side of the robot. Give the robot a command to turn left through Port B and C.
For case 5, if the combined output of human sensor3, human sensor1 and ultrasonic sensor (at a 0 or -45 degree angle) is high, the human is on the left side of the robot. Give the robot a command to turn right through Port B and C.	For case 6, if the output of ultrasonic sensor (at 45 degree angle) is high, the obstacle is on the left side of the robot. Give the robot a command to turn right through Port B and C.
For case 7, if the output of ultrasonic sensor (at 0 degree angle) is high, the obstacle is in front of the robot. Give the robot a command to turn right through Port B and C.	For case 8, if the output of ultrasonic sensor (at -45 degree angle) is high, the obstacle is on right side of the robot. Give the robot a command to turn left through Port B and C.

B.3 The Information Flow of LabVIEW Code: