

February 2010

Master's Degree Thesis

# Obstacle Avoidance for Autonomous Navigation based on Context Awareness

Autonomous Navigation for a Mobile Robot Avoiding  
Obstacles Based on Context Awareness

Graduate School of Chosun University

Department of Information and Communication  
Engineering

Muhammad Tahir Rasheed

# Obstacle Avoidance for Autonomous Navigation based on Context Awareness

Autonomous Navigation for a Mobile Robot Avoiding  
Obstacles Based on Context Awareness

February 25, 2010

Graduate School of Chosun University

Department of Information and Communication  
Engineering

Muhammad Tahir Rasheed

# Obstacle Avoidance for Autonomous Navigation based on Context Awareness

Advisor: Prof. Lee Joon, Ph.D.

This thesis is submitted to Chosun University in partial  
fulfillment of the requirements for a Master's Degree

October 2009

Graduate School of Chosun University

Department of Information and Communication  
Engineering

Muhammad Tahir Rasheed

Muhammad Tahir Rasheed's  
Master's Degree Thesis Approval

Committee Head (인)

Committee Member (인)

Committee Member (인)

November 2009

Graduate School of Chosun University

# Contents

Abstract

1.	Introduction .....	1
1.1	Motivation .....	1
1.2	Current work.....	2
1.3	Thesis Overview .....	3
2.	Obstacle Avoidance and Context Awareness.....	4
2.1	Obstacle Avoidance .....	4
2.2	Context Awareness .....	7
2.3	Ubiquitous Computing.....	9
2.4	Activity recognition .....	10
2.5	Challenges in Context-Aware Computing .....	11
2.6	Motivation to the Proposed Obstacle Avoidance Algorithm.....	14
3.	Obstacle avoidance and autonomous navigation.....	16
3.1	Global and Local Obstacle Avoidance.....	16
3.2	Algorithm for autonomous navigation .....	17
3.3	Algorithm for panning scan by sensors for course driving .....	20
4.	Moving obstacle avoidance using LRF sensor .....	23
4.1	Obstacle Detection.....	23
4.1.1	Obstacle Identification .....	23
4.1.2	Segmentation .....	24
4.1.3	Circularization .....	24
4.1.4	Estimation of future collision .....	26
4.2	Algorithm for Collision Avoidance.....	27
4.2.1	Obstacle's direction conversion and distance measured from the LRF sensor's center to the robot's center .....	28
4.2.2	Determination of moving direction.....	29
4.2.3	Velocity and angular velocity determination for obstacle avoidance.....	32
5.	Test Results and Implementation .....	34

5.1 Panning scan System of Sensors.....	37
5.2 Implementation of Autonomous Navigation System using autonomous navigation algorithm.....	38
6. Conclusion.....	43
References .....	44

## List of Tables and Figures

### Figures

Figure 3.1 Boy Scout orienteering problem.

Figure 3.2 Coordinates of the sensors and the obstacle

Figure 3.3 Coordinates of sensor and obstacle

Figure 4.1 Segmentation

Figure 4.2 Concept of Kalman Filter algorithm.

Figure 4.3 The collision estimation with the robot and a moving obstacle.

Figure 4.4 Acquiring the avoidance point of the robot.

Figure 4.5 Translation from the sensor coordinator to the robot coordinator

Figure 4.6 case1 :  $0 < \theta_{obs,min} < \frac{\pi}{2}$

Figure 4.7 Case2 :  $\pi/2 < \theta_{obs,min} < \pi$

Figure 5.1 Autonomous navigation robot system

Figure 5.2 Panning scan of sensor

Figure 5.3 Testing map for autonomous navigation

Figure 5.4 Test result of starting at left A (75, 0)

Figure 5.5 Test result of Starting at center B (128, 0)

Figure 5.6 Test Result of starting at right C (155, 0)

### Tables

Table 5.1 Specifications of control system

Table 5.2 Test results of obstacle avoidance algorithm

# Abstract

## Obstacle Avoidance for Autonomous Navigation based on Context Awareness

Muhammad Tahir Rasheed

지도교수: 이 준

정보통신공학과

조선대학교 일반대학원

자율주행 로봇에 대한 연구는 인간이 이 세상에서 로봇을 꿈꾸던 때부터 시작 되었고 끊임없이 연구 되고 있다. 로봇이란 인공적인 인간이라고 할 수 있겠다. 이런 로봇에 자율성을 부여하고 스스로 장애물을 회피하면서 이동경로를 설정 움직이기란 쉬운 일이 아니다.

본 연구 논문은 상황인식 기반에서 자율 주행로봇을 위한 상황인식 기반 알고리즘을 보여주고 있으며 어떠한 장애물도 회피 할 수 있도록 디자인 되어 있다. 본 모델은 정지한 물체와 움직이는 물체를 다 고려해 디자인 되어 있다. 장애물을 회피하기 위해서 기본적으로 퍼지 시스템을 이용하였고 장애물 감지를 위해서는 적외선 센서를 스텝모터 위에 올려 패닝 스캔 방법을 사용하였다. 이동 물체에 대한 감지는 로봇의 중심체를 중심으로 원을 그려 반경을 구하고 패닝 스캔을 이용하여 감지 하였다. 그리고 움직이는 이동체의 거리 및 위치 반경을 예측하기 위해서 칼만 필터를 사용하였다. 실험을 위해서는 간단한 시뮬레이션과 실 로봇을 가지고 테스트 하였다. 많은 장애물들이 있었지만 거의 완벽하게 회피 하였다. 이동 물체에 대한 계산 량이 많아 위치 인식 및 로봇 이동 경로에 대한 문제가 있었지만 곧 해결될 것이다.



# **Abstract**

## **Obstacle Avoidance for Autonomous Navigation based on Context Awareness**

Muhammad Tahir Rasheed

Advisor: Prof. Lee Joon, Ph.D.

Department of Information and

Communication Engineering,

Graduate school of Chosun University

In this paper algorithm for obstacle avoidance in autonomous robot is proposed based on context awareness. The main object is to navigate the autonomous robot to the desired destination while detecting and avoiding any possible obstacles. A model theory of autonomous mechanism of robot navigation is showed. We have considered both stationary and moving obstacles. The stationary obstacles avoidance mechanism is based on fuzzy systems and infrared sensors mounted on step motors are used to detect the obstacles. The proposed algorithm detects the position and size of the obstacle and helps the robot to decide the movement action. For moving obstacles we consider the moving obstacle as a moving circle thus finding the center and radius of the circle becomes the primary concern. This is carried out by circularization. The velocity and position of the moving obstacle is estimated by using Kalman filter for future collision estimation. The collision is avoided by determining the angular velocity and direction of the moving obstacle. The test results show the navigation procedure and the action taken by the robot in certain situations.

Many other obstacle avoidance architectures are proposed consisting of resource, behavior, and controller with object-oriented approach structures showing good performance but their heavy calculation method makes it difficult to perform in real-time control. But in this paper, algorithms using geometrical methods have been introduced to ensure simple avoids with less computation aimed to improve real-time abilities. Moreover, the application of Kalman Filter led to the minimization of sensor and system errors.

# **1. Introduction**

## **1.1 Motivation**

The present day requirement for ever-increasing reliability is now more important than ever before and continues to grow constantly. Advances are continually being made in engineering. This means that the detection, location and analysis of faults play a vital role. The need to have efficiency and safety in the design and development of Automated Guided Vehicle (AGV) leads to the development of diagnostic strategies that could cover the major potential faults of the automated vehicle guidance. Following several decades of intense research in automated vehicle guidance, we are now witnessing market introduction of driver assistance functions into standard passenger cars. Most of these functions are based on inertial sensors, i.e. sensors measuring the status of the vehicle itself. A hi-tech product like a robot needs a sophisticated system for analyzing the failures, storing the related information in an integrated repository and retrieving the same via standard user-friendly interface.

The main focus of this thesis is on the sensor concept developed for the autonomous system of our robot and on the realization thereof. Reliable detection and tracking of obstacles is a crucial issue for automated vehicle guidance functions.

With the advancement of technologies, the human desire pursuing convenience and wealth brought up the development of industrial robots resulting in productivity improvement and entertainment robots for providing human pleasure in their everyday life. Furthermore, with the appearance of service robots for handicapped people and

welfare robots for elderly people, humans are now foreseeing the coexistence of humans and robots in daily environment.

As the human's expectation for robot grows, the hardware of robots becomes complicated and more diverse sensors are used. Most of all, the system integration of various software as well as hardware becomes more important. For this purpose, an architecture that allows an easy integration of diverse software is demanded.

It is difficult for existing control architectures to satisfy the current performance requirements, and therefore it is necessary to introduce new concept control architecture to satisfy the various functions required.

## **1.2 Current work**

Lindstrom [6] introduced a reactive architecture named "BERRA" that consists of resource, behavior, and controller. Their structure uses an object-oriented approach and therefore shows good performance in reusability and flexibility. However, it underperforms in real-time control. The robot control mode of the hybrid architecture suggested by Hans and Baum [3] through Care-O-bot consists of environment perception, skill, and trajectory components of interpreter. Though it uses a real-time operation system like real-time framework Vxworks, the data flow has unclearness and lacks of guaranteeing real-time characteristics when programming the software. Low and Ang [7] locate the components of target reaching, obstacle avoidance, homeostatic control and command fusion in reactive module. This module is similar with the subsumption architecture of Brooks [2].

Though it shows good performance in obstacle avoidance in a dynamic environment, the implementation of the behavior to achieve various tasks is difficult and calculation burden is heavy. As stated, the recent researches on control architecture are neither to

realize the character of reactive layer nor enough to consider for real-time implementation of robot [1, 4].

This paper presents an overall description of the Real-Time Control Architecture for autonomous mobile robots and an obstacle avoidance algorithm using geometric calculations.

### **1.3 Thesis Overview**

This thesis is organized into 6 chapters. Chapter 1 is the introduction of the motivation and objective of the thesis. Chapter 2 describes the overview of obstacle avoidance and context awareness and its relation to our research. Chapter 3 presents the idea and algorithm for stationary obstacle avoidance and autonomous navigation. Chapter 4 shows our future work on moving obstacle avoidance along with the algorithm proposed for it. Chapter 5 presents the test and experiments with the simulation results of the stationary obstacle avoidance algorithm. And finally Chapter 6 shows the conclusion of the thesis with recommendations for future work and areas for further development.

## 2. Obstacle Avoidance and Context Awareness

### 2.1 Obstacle Avoidance

Recently, many researches turned their attention to obstacle avoidance problem developing interesting real-time approaches for narrow and cluttered spaces. However, there are some classic obstacle avoidance methods that must be cited [18]. The first one, edge detection, is a very popular method that extracts the obstacle vertical edges and drives the robot around either one of the visible edges. This approach was early commonly combined with ultrasonic sensors. Due to the limited accuracy of the sensor, the approach presented some shortcomings: poor directionality, frequent misreading, and specular reflections. On the other hand, Moravec and Elfes [19] pioneered the concept of certainty grid, a map representation that is well suited for sensor data accumulation and fusion. Certainty grid is an obstacle probabilistic representation method that uses a grid-type world model. The robot's work area is modeled as a 2-D array of square elements, called cells. Each ABCM Symposium Series has a certainty value ( $CV$ ) that indicates the measure of confidence that an obstacle is within the cell area [20]. The  $CV$  is a probability function that depends on the sensor characteristics. As each cell has its  $CV$  updated constantly by the sensor readings, after a period moving across an area, the robot has a fairly accurate map of that area. The method accuracy is a function of the cell size and may be considered as its drawback as well. The third method, potential field method is based on the idea that obstacles exert imaginary repulsive forces, while the goal position applies an imaginary attractive force to the robot. The resultant robot behavior is obtained summing all attractive and repulsive forces.

The potential field method was later improved by Koren and Borenstein integrating its concept with the certainly grid concept. Based on the certainly grid data, a 2-D Cartesian histogram grid (bar graph in which the area of each bar is proportional to the frequency or relative frequency represented) is used to represent the probability of each cell contained in an obstacle. After that, the potential field idea is applied to the histogram grid in order to obtain a fast reflexive obstacle avoidance behavior. This new method was named Virtual Force Field method (*VFF*). Nevertheless, after some experiments, it was abandoned due to the method instability and inability to pass through narrow passages like doors (local minima problem). Repulsive forces from both sides of the doorway results on a force that pushes the robot away.

In the 1990s Koren and Borenstein developed the Vector Field Histogram (*VFH*) approach and afterward, Ulrich and Borenstein made some incremental improvements called *VFH+* and *VFH\** approaches [21]. The *VFH* methods create a local certainly grid map of the robot surround environment using sensor readings. Instead of a 2-D Cartesian, a Polar histogram ( $\alpha$ -P) is built based on the certainly grid map. One should observe that  $\alpha$  is the sensor angle and P is the probability that there is an obstacle in that direction. A probability threshold value is used in order to determine which directions may be considered as obstacle-free ones. Taking into account the robot's size and shape (configuration space), all obstacle-free directions are checked to verify if they are large enough for the robot to pass through. A masked polar histogram where the obstacles are enlarged is calculated. After that, the steering direction for the robot is chosen. In the *VFH+* improvement, the basic robot kinematics limitations were used to compute the robot possible trajectories using arcs or straight lines. Finally, in 2000 the *VFH\** improvement proposed the look-ahead verification. The method analyses each possible

direction provided by the *VFH+* approach, checking their consequences concerning the robot future positions. It projects the robot trajectory several steps ahead, building a search tree where the end nodes correspond to a total projected distance.

Simultaneously, another method based on the admissible robot velocities was proposed.

These methods are named *Steer Angle Field Approaches*. In 1997, the Dynamic Window Approach (*DWA*) was developed by Fox, Burgard and Thrun. This approach takes into account robot kinematics constraints in order to calculate all possible sets of velocity vectors  $(v, \omega)$  in the velocity space. One should observe that  $v$  and  $\omega$  are the robot translational and rotational velocities, respectively. Considering the robot possible accelerations, the overall search velocity space is reduced to the dynamic window, which contains only the velocities that can be reached within the next time interval. The dynamic window is a rectangle centered on the robot present velocity and its vertex positions depend on the accelerations that can be applied. The dynamic window has a rectangular shape because it was assumed that the robot dynamic capabilities for translation and rotation are independent. All velocity vectors outside the dynamic window cannot be reached within the next time interval and thus should not be considered for the obstacle avoidance. The motion direction is chosen by applying an objective function to all admissible velocity vectors in the dynamic window. This objective function depends on the robot velocity, the distance between the robot and the closest obstacle, and the robot progress toward the goal position.

Brock and Khatib proposed a significant improvement to the dynamic window approach. They added a global thinking to the *DWA* by using the grassfire technique for finding routes in the certainly grid cells. Each cell is labeled with the distance to the robot's goal position (like a wave front expansion from the goal position outward). The desired



trajectory is obtained by linking adjacent cells that are closer to the robot's goal position. This procedure allows the robot to improve their performance by using some of the advantages of global path planning without complete priori knowledge. This procedure was named Global Dynamic Window Approach (*GDWA*).

## **2.2 Context Awareness**

Information systems are about to enter a new era: the era of ubiquitous computing, or the age of calm technology, which will result in an increasing use of personal wireless devices and devices embedded in the environment.

Context awareness is the key feature to obstacle detection and avoidance. In order for a robot to be smart, it has to be fully aware of the context.

Context awareness originated as a term from ubiquitous computing or as so-called pervasive computing which sought to deal with linking changes in the environment with computer systems, which are otherwise static. Although it originated as a computer science term, it has also been applied to business theory in relation to business process management issues [9].

In computer science it refers to the idea that computers can both sense, and react based on their environment. Devices may have information about the circumstances under which they are able to operate and based on rules, or an intelligent stimulus, react accordingly. The term context-awareness in ubiquitous computing was introduced by Schilit [10, 11]. Context aware devices may also try to make assumptions about the user's current situation. Dey define context as "any information that can be used to characterize the situation of an entity" [12].

While the computer science community has initially perceived the context as a matter of user location, as Dey discussed in "Understanding and Using Context", in the last few years this notion has been considered not simply as a state, but part of a process in which users are involved; thus, sophisticated and general context models have been proposed to support context-aware applications which use them to (a) adapt interfaces, (b) tailor the set of application-relevant data, (c) increase the precision of information

retrieval, (d) discover services, (e) make the user interaction implicit, or (f) build smart environments. For example, a context aware mobile phone may know that it is currently in the meeting room, and that the user has sat down. The phone may conclude that the user is currently in a meeting and reject any unimportant calls [13].

Context aware systems are concerned with the acquisition of context (e.g. using sensors to perceive a situation), the abstraction and understanding of context (e.g. matching a perceived sensory stimulus to a context), and application behavior based on the recognized context [14]. As the user's activity and location are crucial for many applications, context awareness has been focused more deeply in the research fields of location awareness and activity recognition.

Context awareness is regarded as an enabling technology for ubiquitous computing systems. Context awareness is used to design innovative user interfaces, and is often used as a part of ubiquitous and wearable computing. It is also beginning to be felt in the internet with the advent of hybrid search engines. Schmidt, Beigl & Gellersen [15] define *human factors* and *physical* environment as two important aspects relating to computer science.

Human factors related context is structured into three categories: information on the user (knowledge of habits, emotional state, biophysiological conditions, and so on), the user's social environment (co-location of others, social interaction, group dynamics, and so on), and the user's tasks (spontaneous activity, engaged tasks, general goals, and so on). Likewise, context related to physical environment is structured into three categories: location (absolute position, relative position, co-location, and so on), infrastructure (surrounding resources for computation, communication, task performance, and so on), and physical conditions (noise, light, pressure, and so on).

The motivation behind ubiquitous computing is the need for such applications to understand the physical and social environment or context, in which they reside. This typically includes the location, identity, activity and state of people, groups and objects. Places such as buildings and rooms can be fitted with sensors that provide measurements of physical variables such as temperature or lighting. The ability of computing devices to detect, sense, interpret and respond to aspects of a user's local environment and the computing devices themselves are known as context-aware

computing. This has led to the development of the concept of Context-Awareness, which is now an emerging area of computing that involves computers' decisions based on users' environment.

Recently there has been an explosive expansion in mobile computing and an increasing availability of low-cost sensors to detect elements of the user's current context. As a result there has been an ever-increasing interest in context-aware applications. Like many emerging technologies, context-awareness has attracted a wide spectrum of claims as to its likely future impact.

Recent advances in computer technology are making the development of context aware applications possible. In the near future, these are expected to replace traditional monolithic computing applications which are often static and inflexible with contextual dependencies embedded in them. The increasing complexity requirements of context awareness make such traditional applications difficult to adapt.

Context-awareness becomes a fundamental enabling technology for Ubiquitous Computing and is a key issue when creating computers that are invisible and disappear in terms of the user's perception. In these terms context-awareness goes beyond providing context information, it also requires understanding context and ultimately understanding situations.

## **2.3 Ubiquitous Computing**

Ubiquitous computing (ubicom) is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. In the course of ordinary activities, someone "using" ubiquitous computing engages many computational devices and systems simultaneously, and may not necessarily even be aware that they are doing so. This model is usually considered advancement from the desktop paradigm.

This paradigm is also described as *pervasive computing*, ambient intelligence, or, more recently, everywhere [16]. When primarily concerning the objects involved, it is also *physical computing*, the *Internet of Things*, *haptic computing*, and *things that think*.

Rather than proposing a single definition for ubiquitous computing and for these related terms, taxonomy of properties for ubiquitous computing has been proposed, from which different kinds or flavors of ubiquitous systems and applications can be described [17].

At their core, all models of ubiquitous computing share a vision of small, inexpensive, robust networked processing devices, distributed at all scales throughout everyday life and generally turned to distinctly common-place ends. For example, a domestic ubiquitous computing environment might interconnect lighting and environmental controls with personal biometric monitors woven into clothing so that illumination and heating conditions in a room might be modulated, continuously and imperceptibly. Another common scenario posits refrigerators "aware" of their suitably-tagged contents, able to both plan a variety of menus from the food actually on hand, and warn users of stale or spoiled food.

Ubiquitous computing presents challenges across computer science: in systems design and engineering, in systems modeling, and in user interface design. Contemporary human-computer interaction models, whether command-line, menu-driven, or GUI-based, are inappropriate and inadequate to the ubiquitous case. This suggests that the "natural" interaction paradigm appropriate to a fully robust ubiquitous computing has yet to emerge - although there is also recognition in the field that in many ways we are already living in an ubicomp world. Contemporary devices that lend some support to this latter idea include mobile phones, digital audio players, radio-frequency identification tags, GPS, and interactive whiteboards.

## **2.4 Activity recognition**

Activity recognition aims to recognize the actions and goals of one or more agents from a series of observations on the agents' actions and the environmental conditions. Since the 1980s, this research field has captured the attention of several computer science communities due to its strength in providing personalized support for many different applications and its connection to many different fields of study such as medicine.

To understand activity recognition better, consider the following scenario. An elderly man wakes up at dawn in his small studio apartment, where he stays alone. He lights the stove to make a pot of tea, switches on the toaster oven, and takes some bread and jelly from the cupboard.

After taking his morning medication, a computer-generated voice gently reminds him to turn off the toaster. Later that day, his son accesses a secure website where he scans a check-list, which was created by a sensor network in his father's apartment. He finds that his father is eating normally, taking his medicine on schedule, and continuing to manage his daily life on his own. That information puts the son's mind at ease.

Many different applications have been studied by researchers in activity recognition; examples include assisting the sick and disabled. For example, by automatically monitoring human activities, home-based rehabilitation can be provided for people suffering from traumatic brain injuries. One can find applications ranging from security-related applications and logistics support to location-based services. Due to its many-faceted nature, different fields may refer to activity recognition as plan recognition, goal recognition, intent recognition, behavior recognition, location estimation and location based services.

## **2.5 Challenges in Context-Aware Computing**

Examples, demonstrators, and prototypes have been used to demonstrate that context awareness can enhance applications and systems. Typically location is sensed and then based on the location further assumptions about the more general context are made. As the concept of position and location is well understood, it also provides a powerful and easy to apply model for context-aware applications. In many cases however awareness based solely on location lacks information that can be of interest to a system for making

it context-aware. If information beyond location information is required, further complexity is introduced. The following issues are central research challenges in context-awareness:

- **Understanding the concept of context.**

What does context mean and how is it connected to situations in the real world? There is still a fundamental lack of understanding in terms *how contexts relate to situations* and how general context information can be used to help enhance applications. This is also associated with the question of how to represent context in a universal way.

- **How to make use of context?**

Assuming that context is available in a system the question *what is context useful for* becomes imminent: especially if contexts beyond location and available resources are considered. In this instance a central question is what type of applications can be enhanced? When considering context as additional input, issues of reliability and ambiguity are important. Furthermore, the relation between context and other inputs into the system and how they influence each other, have to be addressed. Ultimately this requires the smartness of the system to *understand* the context it is dealing with.

- **How to acquire context information?**

*Acquiring context is a prerequisite for any context-aware system.* Generally context acquisition can be seen as the process where the real situation in the world is captured, the significant features are assessed, and an abstract representation is created, which is then provided to components in the system for further use. Approaches to acquire context are manifold and include computer vision, location tracking, sensor systems, and also more predictive approaches such as modeling users and their behavior.

- **Connecting context acquisition to context use.**

In a location-aware system there is a close relationship between context acquisition and context use, most often the location sensor is attached to the device using position as context. In this case the context representation is also agreed between these components. In more general environments context use and context acquisition is distributed. It can

be assumed that context is provided for various applications, potentially in dynamic configurations. This makes it obvious that mechanisms to connect context acquisition and context use become essential. Here the challenge is twofold: *overcoming the distribution issue* by networking components and *agreeing on representations* that are useful for a multitude of components.

- **Understanding the influence on human computer interaction.**

When systems are context-aware their behavior is dependent on the context of use or the general situation of use. The ultimate goal is to make systems in such a way that they react as anticipated by the user. In real life however this creates complex problems, in particular if the system reacts differently from the users expectations. Two critical issues are *how can the user understand the system and its behavior?* And *how to give the user control over the system?*

- **Support for building context-aware Ubiquitous Computing Systems.**

Context-awareness is an enabling technology for Ubiquitous Computing Systems and therefore commonly required when realizing such systems. To build Ubiquitous Computing environments efficiently, it is inevitable that we need to provide support for building context-aware applications. Up to now, there are many cases where the wheel is re-invented; where all the problems have to be solved over and over again in each system. Providing *support for context acquisition, context provision, and context use* will make the process of implementing context-aware applications much simpler.

- **Evaluation of context-aware system.**

*As context-aware systems are used in context, evaluation* itself is also required *to be done in context*. In cases where functionality is only available and is useful in a certain context, it is required to create or simulate a particular situation that results in the wanted context in order to assess the system. Inducing a particular situation and context however may have also a significant effect on measures in the evaluation. Many of these research issues are highly interconnected. Nevertheless some of the issues can be tackled fairly independently of some others. In the approach pursued in the course of

research underlying this thesis, context-awareness is approached from a bottom-up perspective.

## **2.6 Motivation to the Proposed Obstacle Avoidance Algorithm**

Obstacle detection algorithms for autonomous navigation have been carried out for various purposes. In this thesis the main purpose of developing an autonomous navigation system was considering two ideas. One is to create an autonomous navigation system in a power wheelchair for the elderly in order to give them comfort while driving the power wheelchair. Nowadays with advancement in technology and aging society, the number of disabled citizens is increasing. The disabled citizens always need a caretaker for daily life routines especially for mobility. In future, the need is expected to increase more. To reduce the burden from the disabled, various devices for healthcare are introduced using computer technology. In the same flow the power wheelchair is an important and convenient mobility device.

Generally, a joystick is used at the arm of the unaffected side to mobilize the wheelchair, however, if both arms are affected, the leg of unaffected side is used. The use of power wheelchair is convenient. However, the driving of conventional wheelchair may cause a burden to the disabled due to continuous use of joystick and always paying deep attention to the surroundings and to the obstacles on the drive-way. Then the capacity to recognize path condition and to avoid obstacles is required. Moreover, the user has to be proficient in driving the wheelchair.

The proposed obstacle detection algorithm is useful as a mobility aid to reduce the burden from the disabled. For the obstacle detection, we have used two infrared sensors mounted on step motors. The infrared sensors scan the area for obstacles, detect the



obstacles and inform the control system. The infrared sensors are connected to the control system by zigbee suite. The step motors and sensors are controlled by ATMEGA 128 Processor. The joystick is connected to the system by the zigbee protocol.

The second motto of obstacle detection was to create an algorithm which helps an autonomous navigation sensor to flow on water surface avoiding collision with any obstacle. With the advancement of technology the purity of flowing water is decreasing, especially water flowing through cities like canals or small rivers need attention. Generally, the waters are tested for acidity and toxicity by taking samples manually from different areas. It may however be inconvenient to take samples frequently from different areas. The autonomous robotic sensors can flow on the water surface and test the water taking samples from different places. The main issue for the sensors is to detect and avoid the obstacles and navigate to the desired destination.

### **3. Obstacle avoidance and autonomous navigation**

In this chapter we presented an effective approach to obstacle avoidance. We then examined the desired attributes of the path planner/obstacle avoidance algorithm used for the autonomous robot.

Robots require a wide range of sensors to obtain information about the world around them. These sensors detect position, velocity, acceleration, and range to objects in the robots workspace. There are many different sensors used to detect the range to an object. One of the most common rangefinders is the ultrasonic transducer. Vision systems are also used to greatly improve the robot's versatility, speed, and accuracy for its complex tasks. For many experimental automated guided vehicles (AGV), ultrasonic transducers, or sonar, are frequently used as a primary means of detecting the boundaries within which the vehicle must operate. But in our testing we used infrared sensors mounted with step motors to rotate the sensors which are effective and inexpensive.

#### **3.1 Global and Local Obstacle Avoidance**

In autonomous vehicle research two levels of planning occur. One level of planning, called global planning, examines the whole world an autonomous vehicle can travel and plans paths from one point to the next dependent upon this world. Obstacle avoidance on a global level is accomplished by routing all paths away from potential obstacles. If the vehicle encounters an unexpected obstacle during a mission then the global planner reexamines the whole map with the added obstacle and adjusts either a portion of the affected path or the rest of the path based on the newer map data. Local or reactive planning plans a short path for the vehicle to traverse based only on the environment surrounding the vehicle, typically using only the detection sensor output. Local or

reactive avoidance is aware of the desired path of travel, but due to path obstructions, plots a path around the obstruction until the vehicle can safely reattach itself to the desired path of travel. An example of global and local obstacle avoidance can be illustrated in my drive to work. Each morning I drive to work my route is the same. If one day I drive to work and a child jumps in front of my car, I do not drive a different route to work, instead I swerve around the child and continue on my way in the usual fashion. This is analogous to reactive obstacle avoidance. However, one day while driving to work I come across a roadblock. Now I choose a different road, or set of roads, that will get me to work. This is analogous to global obstacle avoidance.

### **3.2 Algorithm for autonomous navigation**

This is a simple yet very effective technique for stationary obstacle avoidance and navigating the autonomous robot through the best possible route to the desired goal.

The setting up of the autonomous navigation mechanism of the robot for controller design is shown.

Two distance sensors are used to detect the obstacle which are placed on the right and left side at the front of the robot. A step motor is used to panning scan at the front side for obstacle identification with a prefixed pan degree.

Following is the process by which the autonomous navigation robot will avoid obstacles.

Case 1: On detection of the obstacle:

1. The robot is moving in straight direction.
2. The right and left sensors scan for obstacle.
3. If the right sensor detects an obstacle, the robot stops, else it keeps moving forward.

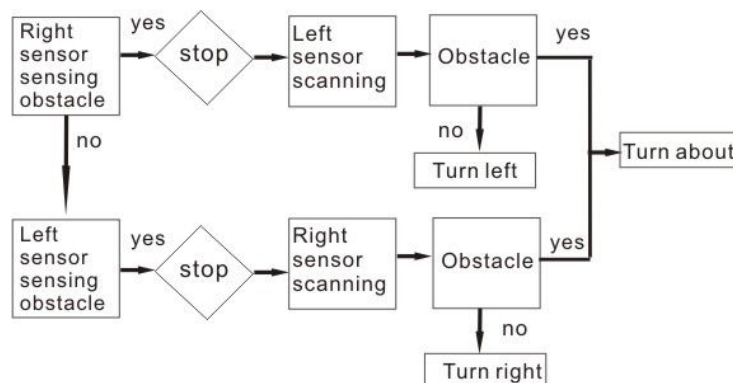
4. If the left sensor detects an obstacle, the robot stops, else it keeps moving forward.
5. If condition3 is satisfied (the right sensor detects obstacle), the robot turns left.
6. If condition4 is satisfied (the left sensor detects obstacle), the robot turns right.
7. If both condition3 and condition4 are satisfied (both right and left sensors detect obstacle), the robot turns back.

Case 2: After turning right or left on detection of obstacle

8. After completing condition 3 or 4 (turning right or left), the robot moves forward.
9. If condition5 was satisfied (the robot turned left), the robot will turn right again (opposite direction) after crossing the obstacle.
10. If condition6 was satisfied (the robot turned right), the robot will turn left again (opposite direction) after crossing the obstacle.

The robot will try to find the best suitable course of movement by learning the route with the shortest distance covered.

The following flowchart expresses the process.



Once the robot detects the obstacle, it turns either left or right in  $90^\circ$  and after avoiding the obstacle it turns back to the opposite direction to go back to the former route. The approach was kept as simple as possible using an idea from a scout orienteering problem. If a scout starts out on a hike and wants to get to the base of a large mountain he will travel a straight path at a constant heading. If a lake blocks his path and forces him to change his desired heading he will choose a new heading  $\pm 90^\circ$  of the desired heading. Suppose for this example he chooses to turn to the right  $90^\circ$ . As he travels the new heading he will always check his desired heading, or to his left, to see if he has passed the lake. Once he has passed the lake he will again travel the desired heading. Now the lake will be on his left and he will travel the desired heading until the lake is no longer on his left. At that point he will adjust his heading to  $90^\circ$  to the right so he can walk back to the original path he would have taken if the lake was not present. When he reaches the desired path he will turn right  $90^\circ$  and walk at his desired heading until he reaches the camp. Fig. 3.1 shows this concept.

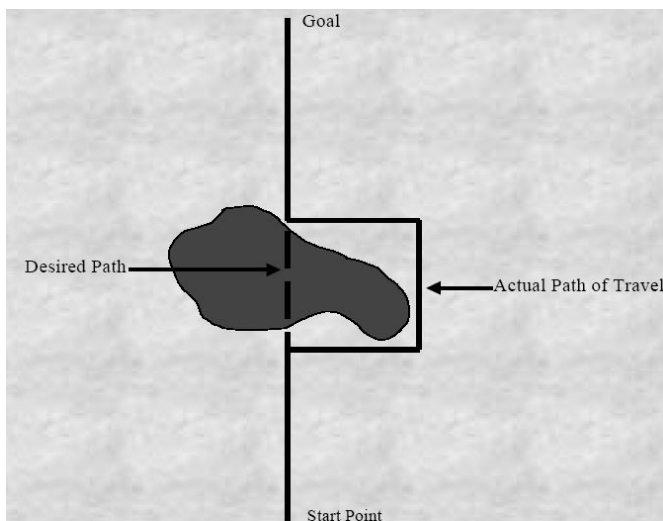


Figure 3.1 Boy Scout orienteering problem.

### 3.3 Algorithm for panning scan by sensors for course driving

The primary purpose of this algorithm is to decide for position and posture of the autonomous robot by defining the course form.

Figure 3.2 shows the relation between sensors coordinates and the obstacle.

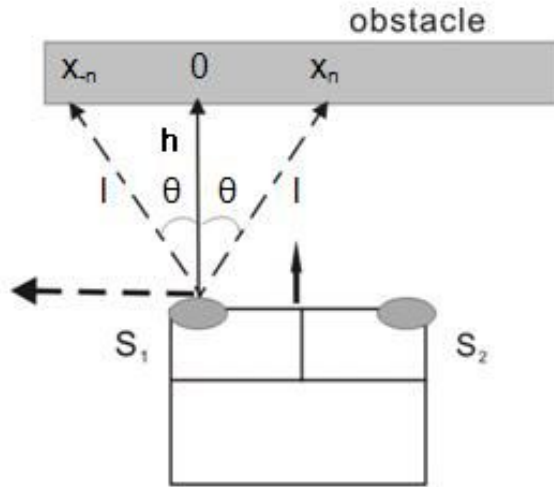


Figure 3.2 Coordinates of the sensors and the obstacle

For the movement towards the obstacle, the new coordinates can show 4x4 determinate of standard coordinates.

That is, to express the direction of the axis of obstacle coordinate, and to express the vector translation of moving obstacle coordinates origin, a matrix can be created by using the composition of these three vectors.

The point  $p1 = [x,y]^T$  is used to create a 2x2 matrix, that the random point  $p$  shows  $[x, y]^T$  on 2 dimension system.

$$\begin{pmatrix} x1 \\ y1 \end{pmatrix} = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + cy \\ bx + dy \end{pmatrix}$$

Obstacle recognition is done by panning scan of x coordinates of course form.

The next equation shows their geometrical relation.

$$\overline{\theta} = \theta(\alpha + \beta) - \theta(\alpha)$$

$$\overline{l} = l(\alpha + \beta) - l(\alpha)$$

The distance measurement is carried out by panning scan by the following process:

1. Range finder sensor detects the obstacle in a range of 50cm, and stops.
2. The sensor starts scanning on right hand side in a range of 50cm
3. The motor scans for the obstacle with 9° panning.
4. After completing the scan, the motor comes back to its center position.
5. Then the sensor starts scanning at the left side, with the same distance of 50 cm.
6. The motor scans for the obstacle with 9° panning.
7. After completing the scan, the motor comes back to its center position.

This algorithm detects obstacles on the route of the autonomous robot. It is done by panning scan by both sensors of the robot. Figure 3.3 shows the coordinates of the sensors and the obstacle.

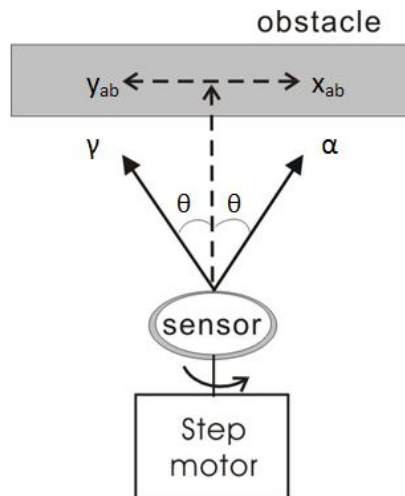


Figure 3.3 Coordinates of sensor and obstacle

The following equation shows the detection of the size of the obstacle.

$$O_{(s)} = \sqrt{(x_{ab}(\alpha + \beta)b - x_{ab}\alpha b)^2 + (\bar{x}_{az}(\alpha + \beta)b - \bar{x}_{az}\alpha b)^2}$$

The algorithm is a process to navigate the autonomous robot through the best route avoiding obstacles.

For the autonomous navigation robot, we have declared the following seven assumptions.

Assumption 1: While detecting left, detect right of front obstacle and left obstacle

Assumption 2: While detecting left, detect right of front obstacle and right obstacle

Assumption 3: Only detect left of front obstacle.

Assumption 4: Only detect right of front obstacle.

Assumption 5: Right sensor doesn't detect obstacle while left-side forward movement

Assumption 6: Left sensor doesn't detect obstacle while right-side forward movement

Assumption 7: Detect obstacle at back side while driving back.



## 4. Moving obstacle avoidance using LRF sensor

### 4.1 Obstacle Detection

A robot needs to visibly spot the dynamic condition of the obstacle before taking any action to avoid it. This is because a stationary obstacle doesn't become an object of concern because it is recognized as a part of the environment, but a moving obstacle can be avoided precisely only when its moving direction and velocity are predicted precisely.

Therefore, in this segment, while guaranteeing real-time performance, the information on the surrounding environment will be acquired using the LRF mounted on the robot; moreover we will deal with the methods that can expel the obstacle within the information.

#### 4.1.1 Obstacle Identification

The data acquired from the LRF can be expressed as:

$$LRF_k = \left\{ P_{k,i} = \begin{pmatrix} a_{k,i} \\ d_{k,i} \end{pmatrix}, i \in [0, 360], \Delta \alpha = 0.5^\circ \right\}$$

Of the 361 data obtained, objects existing within a certain range in the robot progressing direction will be detected.

Here,  $k$  represents the  $k$ -th discrete time,  $a_i$  is the  $i$ -th angle, and  $d_i$  is the distance to the object obtained from that angle.

### 4.1.2 Segmentation

It is necessary to extract the obstacle information from the LRF data in order to recognize an obstacle. The segmentation process begins with calculating the distance between the two points acquired consecutively. As shown in Figure 4.1, if the distance between the two succeeding points is the same as or smaller than the variable critical value  $C_0 + C_1$ , they will be classified as the same piece, and if the distance is larger than the critical value, they will be classified as different pieces.

### 4.1.3 Circularization

Of the data on an obstacle that are acquired by segmentation, the nearest point to the robot and the two end points are used to find the center of a circle supposing that the obstacle is circular. Circularization is a process necessary to consider the relativity of movements because the surface of the obstacle nearest to the robot at the  $k$ -th point of time is not the nearest part at the  $k+1$ th point of time.

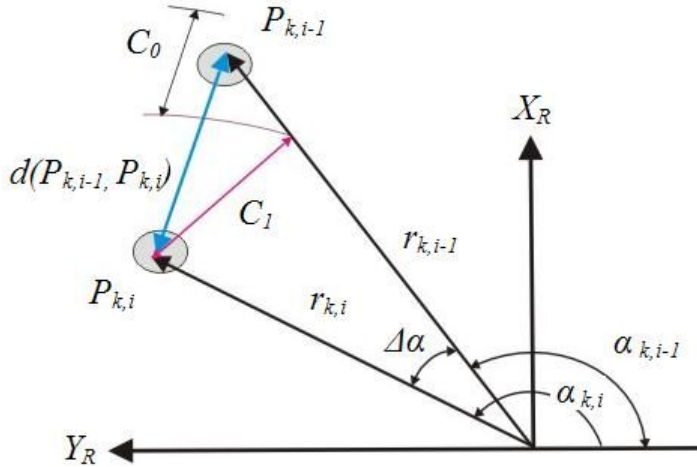


Figure 4.1 Segmentation

When the robot exists at an optional point of time  $k$ ,  $P_{k,ini}$ ,  $P_{k,near}$ , and  $P_{k,end}$  can be converted into the rectangular coordinates for circulation.

If two straight lines that are calculated are same, we conclude the form is like a wall. If different, we judge that its form is like an archetype obstacle and we estimate the subsequent position of obstacle after progressing the below process.

The middle point position of the obstacle exists at the intersection point of the two straight lines and therefore the coordinates of x axis can be obtained by calculating

$$Y'_a = Y'_b$$

The value of y axis can be obtained from

$$y'_a \mid x_k^{obs} = y_k^{obs} \text{ (or } y'_b \mid x_k^{obs} = y_k^{obs} \text{)}$$

When  $P_k^{obs}$  (the position of the calculated obstacle) is expressed as

(1)

(2)

Obstacle characteristics (dynamic/static) can be obtained through segmentation and the speed of measured obstacle is compared with the transfer speed of the robot to judge the circularization. (The static obstacle also has velocity because estimated value is relative).

If dynamic/static characteristic is distinguished, in the case of moving obstacle, the robot operates using forecasted trajectory of obstacle; otherwise it avoids the obstacle that is observed only within avoidance range.

#### 4.1.4 Estimation of future collision

When dynamic obstacle is distinguished, transfer direction of obstacle is estimated using the speed of calculated obstacle. Because the calculated cycle is very short by 20ms; even if we suppose that the momentary obstacle has a straight movement, it will be justifiable. The equation of straight line of obstacle motion is created using estimated two points of  $k$  and  $k+1$  time. The straight line that passes the origin (center of robot) which is perpendicular straight line with the above straight line, can be calculated easily. The nearest position between obstacle center and robot center can be calculated by the distance between the intersection point of the two straight lines that are calculated above and the origin. If this distance is smaller than the critical value ( $D_{radius} = r_{obs} + r_{robot}$ ), as in Figure 4.3, the robot will have a collision with the moving obstacle in future. If we get the result showing the robot and obstacle will collide, we have to calculate the position in which the robot operates the avoidance action. Hence, the position of the obstacle is the distance between point over trajectory of obstacle and the origin which is  $D_{radius}$ . Coordinate of avoidance position is calculated by looking for a point of contact of equation of the circle of obstacle and that of robot. Figure 4.3 shows, using calculated coordinate, the avoidance radius of robot is shifted along the straight line that passed the center of robot and center of obstacle. In order to apply the obstacle avoidance algorithm, we must acquire three points at the position using the calculated coordinate. Because the distance to its center and the radius of obstacle are known, the coordinate is calculated using geometrical relation as in Figure 4.3.

The values of angle of coordinate are

$$\theta_r - \theta_{obs} \text{ ' } \theta_r + \theta_{obs} \theta_r$$

And distance of the coordinate is

$$\sqrt{D_{tot}^2 - r_{obs}^2}, \sqrt{D_{tot}^2 - r_{obs}^2}, d_{safe}$$

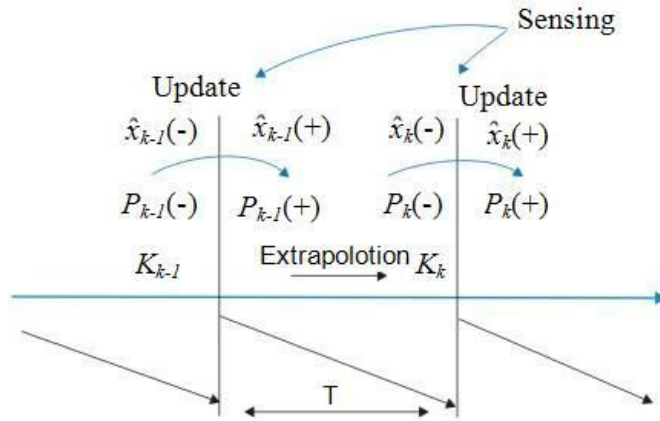


Figure 4.2 Concept of Kalman Filter algorithm.

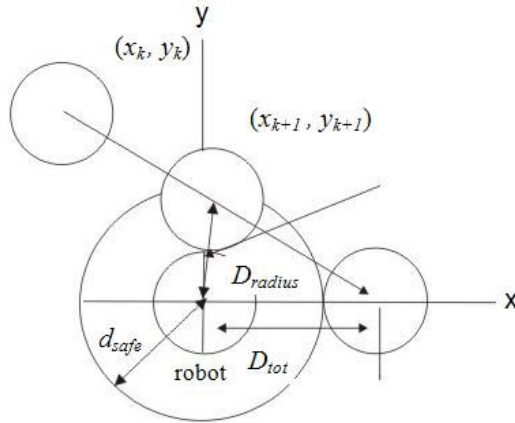


Figure 4.3 The collision estimation with the robot and a moving obstacle.

## 4.2 Algorithm for Collision Avoidance

One of the most important and basic factors in evaluating the safety and traveling abilities of an autonomous mobile robot is obstacle avoidance. This study deals with

reactive control among the various sensor-based traveling technologies. The core of reactive control is to avoid an obstacle in real time. To this end, simple algorithms, rather than complex ones, are more efficient for faster computations. In this section, the point nearest to the robot is located using the obstacle's coordinate, radius and origin obtained in the previous section. In addition, this section explains the process in which velocities and angular velocities are created by using the algorithm that makes use of the geometric relations between the robot and the nearest point.

#### 4.2.1 Obstacle's direction conversion and distance measured from the LRF sensor's center to the robot's center

Using the radius and distance between the center of the robot and the center of the circle obtained through circularization, the distance to the obstacle nearest to the robot can be obtained. But the relative coordinates of the obstacle obtained in this way are the data measured not from  $O_R$ , the center of the robot, but from the position of the sensor. As the robot is driven around  $O_R$  located on the same line as the driving wheel, it is necessary to move the relative coordinates of the obstacle to the center of the robot. As Figure 4.4 shows, using the information obtained through segmentation and circularization, the angle ( $\alpha_i$ ) and the distance ( $d_i$ ) to the obstacle nearest to the center of the robot can be calculated. As in Figure 4.5(a, b),  $\alpha'$  can be obtained in the following way by considering the case of  $\alpha_i < \pi/2$  and the case of  $\alpha_i > \pi/2$ .

$$\alpha' = \begin{cases} \frac{\pi}{2} + \alpha_i, & 0 < \alpha_i \leq \frac{\pi}{2} \\ 0, & \alpha_i = \frac{\pi}{2} \\ \frac{3\pi}{2} - \alpha_i, & \frac{\pi}{2} < \alpha_i < \pi, \end{cases} \quad (3)$$

$$d_{obs,i} = \begin{cases} d_i + d_r, & \alpha_i = \frac{\pi}{2} \\ \sqrt{d_i^2 + d_r^2 - 2 \cos \alpha' d_i d_r}, & \text{otherwise,} \end{cases} \quad (4)$$

$$\theta_{obs,i} = \begin{cases} \alpha_i, & \alpha_i = \frac{\pi}{2} \\ \cos^{-1} \left( \frac{d_i^2 - d_{obs,i}^2 - d_r^2}{2 d_i d_r} \right), & \text{otherwise.} \end{cases}$$

(5)

The distance ( $d_{obs,i}$ ) between the center of the robot ( $O_R$ ) and the obstacle can be obtained as in (4) by applying trigonometric function cosine law number 2. The angle between the center of the robot and the obstacle ( $\phi_{obs,i}$ ) can be obtained by applying to (5) the distance to the obstacle obtained in (4), the distance between the center of the robot ( $O_R$ ) and the center of the sensor ( $O_S$ ), and the distance between the center of the sensor and the obstacle.

#### 4.2.2 Determination of moving direction

The avoidance becomes activated and the robot's avoidance direction and velocity are determined when the distance between the robot and the measured obstacle becomes less than the boundary value ( $d_{safe}$ ) of the safety area, which actually carries out the avoidance of obstacle collision. When an obstacle exists in the safety zone, in that case, there are two possibilities; which can be referred as two cases depending on the positions of the obstacle and robot. It becomes the beginning point ( $\theta_{obs,r}$ ). When the entered value is larger than the boundary value, the previous point becomes the ending point ( $\theta_{obs,l}$ ).

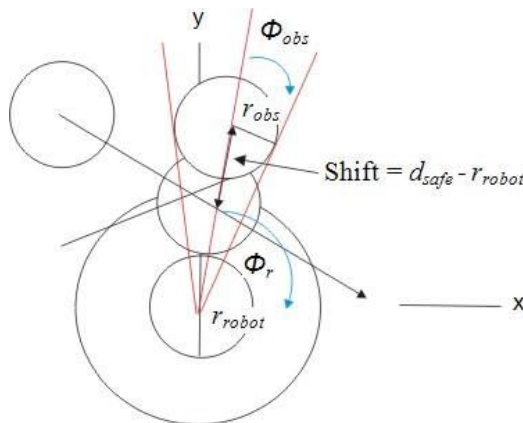


Figure 4.4 Acquiring the avoidance point of the robot.

When the distance between the robot and the obstacle obtained through circularization becomes the shortest, that distance ( $d_{obs,l}$ ) is called  $d_{obs, min}$  and  $\theta_{obs,l}$  at that time is now called  $\theta_{obs,min}$ .  $\Delta\theta$  can be obtained as in (6) using the point where the boundary value meets with the point ( $\theta_{obs,min}$ ) nearest to the obstacle from the center of the robot.

$$\Delta\theta = \begin{cases} \theta_{obs,l} - \theta_{obs,min} , & 0 < \theta_{obs,min} < \frac{\pi}{2} \\ \theta_{obs,min} - \theta_{obs,r} , & \frac{\pi}{2} < \theta_{obs,min} < \pi \end{cases} \quad (6)$$

Case 1:  $0 < \theta_{obs,min} < \pi/2$

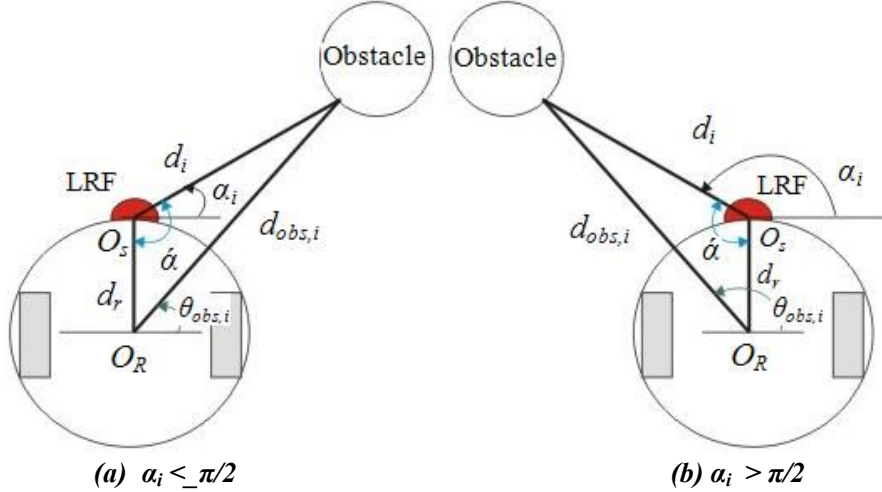


Figure 4.5 Translation from the sensor coordinator to the robot coordinator

Figure 4.6 shows that the point where the distance to the obstacle becomes the shortest ( $\theta_{obs,min}$ ) and the difference between the boundary value and the end point of the



obstacle ( $\theta_{obs,l}$ ) ( $\Delta\theta$ ) can be obtained. Then, vertical line  $D(= 2(d_r + d_{emg}))$  that includes the robot radius and the Emergency-Stop area is drawn in the normal line direction of the extension line linking the minimum point of the robot center and the obstacle. Using  $D$ ,  $d_{obs,min}$ , angle  $\beta$  is obtained as in (7), and  $\gamma$ , the direction in which the autonomous mobile robot will move, is obtained as in (8). As the direction in which the autonomous mobile robot will move is the direction of  $\gamma$  from the robot's direction of movement, the direction in which the autonomous mobile robot actually moves ( $\phi$ ) can be obtained as in (7).

$$\gamma = \beta + \Delta\theta \quad (7)$$

$$\gamma = \beta + \Delta\theta \quad (8)$$

Case 2:  $\frac{\pi}{2} < \theta_{obs,min} < \pi$

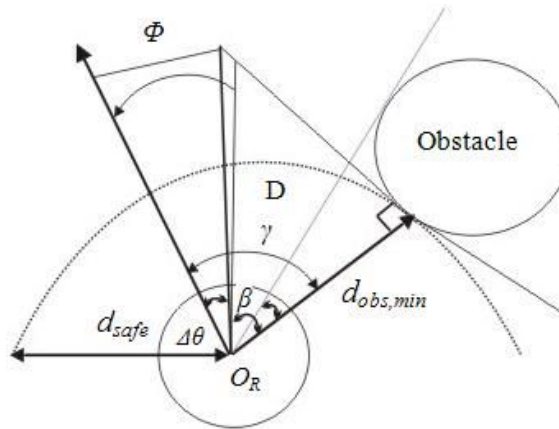


Figure 4.6 case1 :  $0 < \theta_{obs,min} < \frac{\pi}{2}$

Figure 4.7 shows that the collision avoidance direction can be obtained in the same way as Case 1 with use of (6) - (8) when an obstacle is on the left side of the robot's moving

direction. The difference from Case 1 lies in the fact that, as shown in Figure 4.7, the point where the distance to the obstacle becomes the shortest ( $\theta_{obs,min}$ ) and the difference between the boundary value and the beginning point of the obstacle ( $\theta_{obs,r}$ ) ( $\Delta\theta$ ) are calculated for application with respect to the information of the two points for the obstacle. As the direction in which the autonomous mobile robot should move is the direction of  $\gamma$  from the robot's direction of movement, the direction in which the autonomous mobile robot actually moves ( $\phi$ ) can be obtained as in (9).

$$\phi = \begin{cases} \theta_{obs,min} + \gamma - \frac{\pi}{2}, & 0 < \theta_{obs,min} < \frac{\pi}{2} \\ \theta_{obs,min} - \gamma - \frac{\pi}{2}, & \frac{\pi}{2} < \theta_{obs,min} < \pi \end{cases} \quad (9)$$

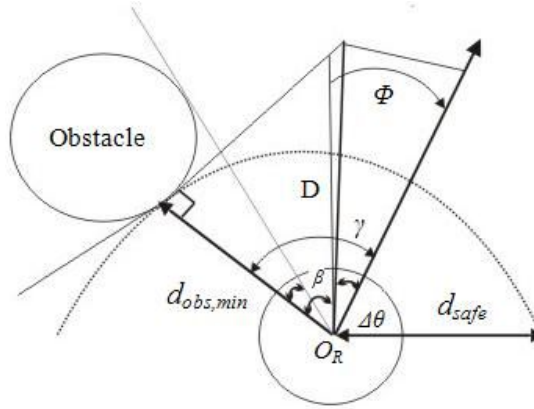


Figure 4.7 Case2 :  $\pi/2 < \theta_{obs,min} < \pi$

#### 4.2.3 Velocity and angular velocity determination for obstacle avoidance

For the robot to avoid an obstacle, to determine the velocity and angular velocity, the boundary value, the distance to the obstacle, and the robot's maximum velocity and angular velocity were applied using the following expression in the nearness diagram algorithm of Minguez and Montano [8].

$$v_o = \frac{d_{obs, \min}}{d_{safe}} \times \left( \frac{\frac{\pi}{2} - |\phi|}{\frac{\pi}{2}} \right) \times v_{\max}, \quad (10)$$

$$\omega_o = \frac{\phi}{\pi} \times \omega_{\max}. \quad (11)$$

Here,  $v_o$  represents the robot's velocity depending on the approach of the obstacle,  $d_{obs}$  the distance between the robot and obstacle,  $d_{safe}$  the boundary value to avoid the obstacle,  $\omega_o$  the robot's angular velocity depending on the obstacle position, and  $\phi$  the direction in which the robot must move.

The velocity of the autonomous mobile robot is determined by the direction and distance to the obstacle as shown in equation (10) and (11). The velocity decreases as the obstacle comes close to the robot, and the angle of the direction of movement becomes larger. The angular velocity is determined by the angle of the direction of movement.

## 5. Test Results and Implementation

For testing the obstacle avoidance algorithm in stationary obstacles, we set up ARM System for the test and implementation of autonomous navigation robot system, sensor platforms, and Control systems. This is based on Auto Control without any change in the basic system. Table 5.1 shows the specifications of the embedded system.

[Table 5.1] Specifications of control system

Category	Article	Specification	etc
Hardware	CPU	PXA270	
	Memory	SDRAM-64M	
		NAND 64M	
	Ethernet	10/100Mbps 1port	
	Audio	AC'97	
	RTC	RTC4531	
	Interface	USB Host, Client, Serial, Jtag, external pin	
	CPLD	Xilinx	
Software	OS	Linux kernel 2.6	

	Device Driver	Ethernet, Frame buffer, Touch screen, Audio, USB, VGA, Serial,	
--	---------------	--	--

The following figure shows the architecture of autonomous navigation robot system.

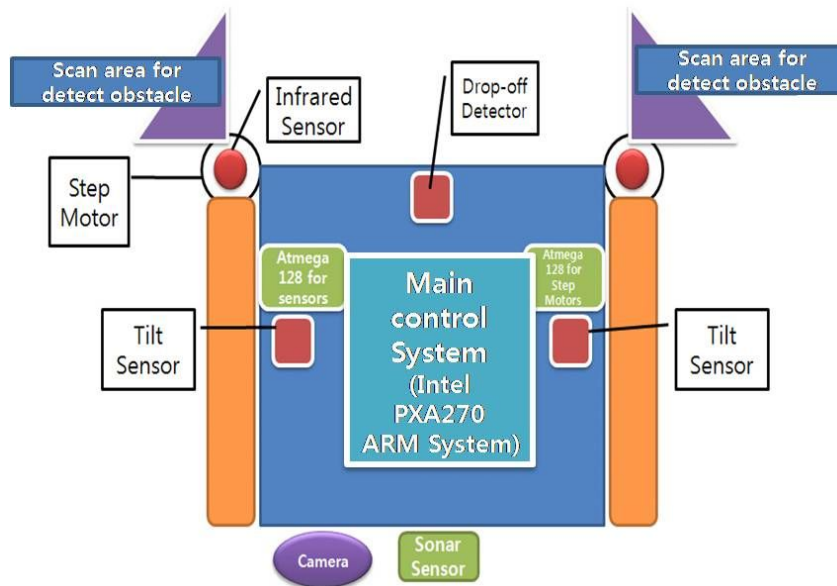


Figure 5.1: autonomous navigation robot system

The testing was performed considering 5 different aspects. The simulation is shown in table 5.2 according to each situation.

#### *Situation 1*

In this case, the obstacle is placed on the right side of the mobile robot. The robot tries to avoid it by moving to the left direction.

#### *Situation 2*

Opposite to case 1, the obstacle is placed on the left side of the mobile robot. The robot tries to avoid it by moving to the right direction.

#### *Situation 3*

The mobile robot is placed in a deadlock situation (obstacles both on right and left side).

The mobile robot tries to escape from this situation by moving in reverse direction.

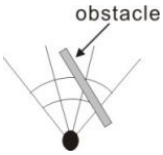
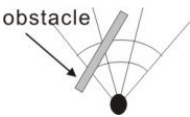
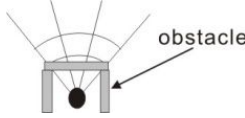
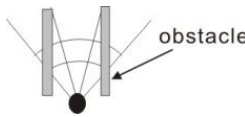
#### *Situation 4*

Obstacles are placed at left and right direction of the mobile robot. There is no obstacle in the middle path, so the robot continues to move in forward direction.

#### *Situation 5*

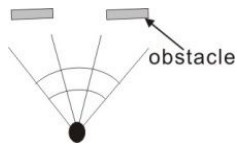
The obstacles are placed very far from the mobile robot, so the mobile robot is still moving in forward direction.

[Table 5.2] Test results of obstacle avoidance algorithm

Number	Situation	Robot Movement
1		Direction = Forward Angle = Left
2		Direction = Forward Angle = Right
3		Direction = Reverse Angle = Mid
4		Direction = Forward Angle = Mid

---

5



Direction = Forward  
Angle = Mid

---

## 5.1 Panning scan System of Sensors

We created a sensor system for test, using sensor panning scan. The sensor used for distance measurement is Sharp GP2Y0A2YK0F and the distance measurement is from 10 to 150cm. the step motor is NK243-01AT 1.8 degree.

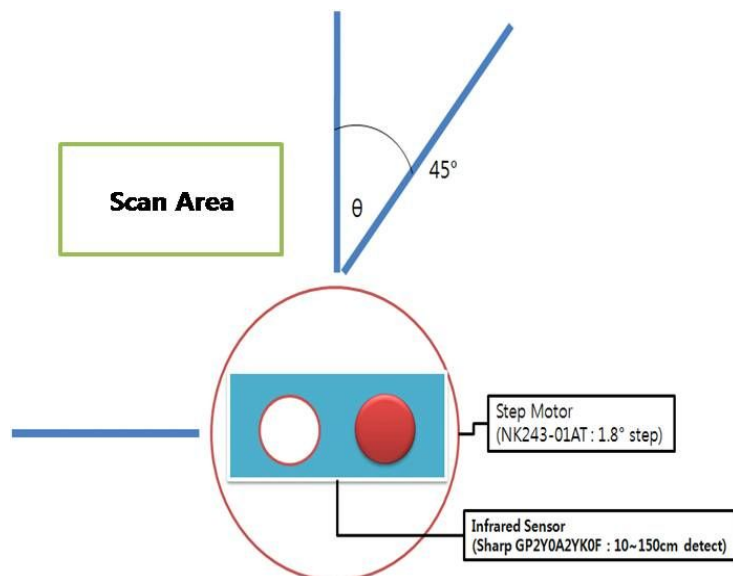


Figure 5.2: Panning scan of sensor

Data measurement gets the distance data to move the step motor at steps of 1.8 degrees from distance measurement sensor. The sensor system scans left and right by panning scan using the step motor. The obtained data is transmitted to ATmega128 system of sensor system by ADC Convertor to calculate 10bit sampling signal data. This data use learning data for autonomous navigation system. Figure 5.2 shows system architecture of Obstacle detection using distance measurement sensor panning.

## 5.2 Implementation of Autonomous Navigation System using autonomous navigation algorithm

The map environment has 255cm width and 700cm height for test. We set up obstacles of 100cm width and 30cm height along the y-axis; at 200cm, 600cm, and 400cm on right of y-axis. Driving course goal figures out the best driving courses from 0 to 650 coordinate of y-axis. The size of autonomous navigation robot is 50cm in width and 100cm in height. Figure 5.3 is the environment map for the test.

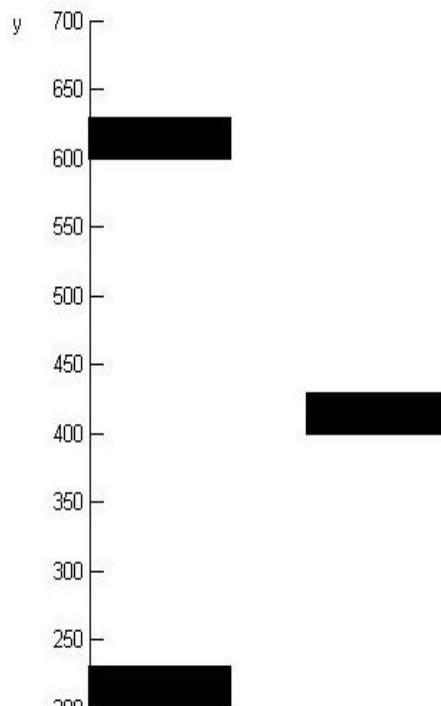




Figure 5.3 Testing map for autonomous navigation

For the testing of the autonomous navigation robot, it drives on course after setting up the start point at A(75, 0), B(128, 0) and C(175, 0) of center-axis of the robot. Figure 5.4, 5.5, and 5.6 show result of the simulation test of autonomous navigation driving.

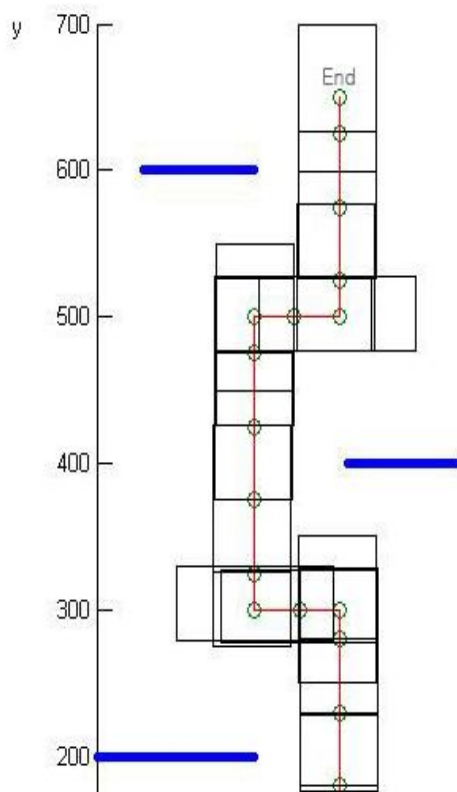


Figure 5.4 Test result of starting at left A(75, 0)

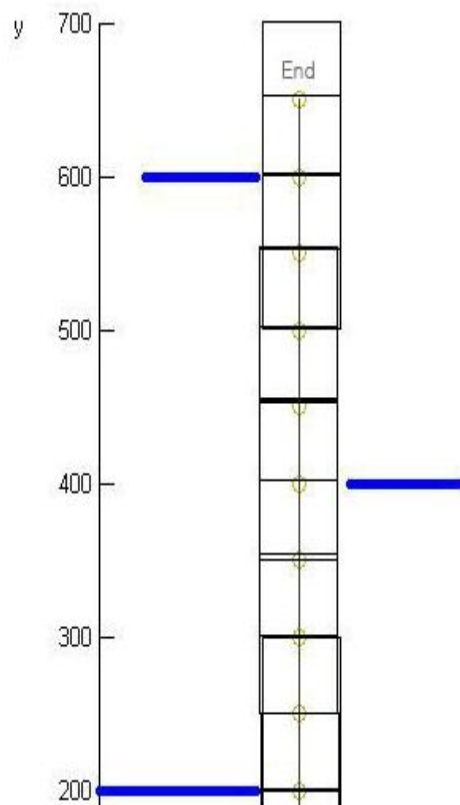


Figure 5.5 Test result of Starting at center B (128, 0)

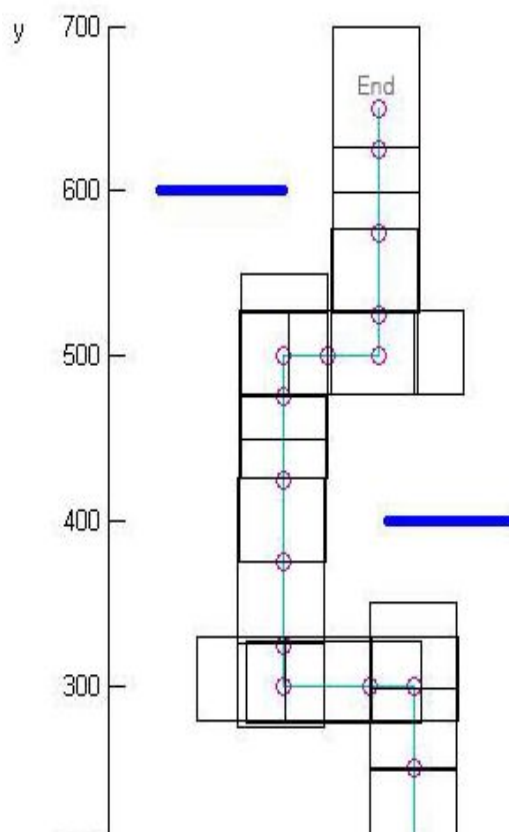


Figure 5.6 Test result of starting at right C (155, 0)

This increases the moving distance. The autonomous navigation robot detects obstacle in a range of 50cm ahead, assuming that the speed of autonomous navigation robot is regular. The autonomous navigation robot keeps up 30cm of each side to get the point values of Obstacle while driving on a side of obstacle. At first start of autonomous navigation robot at A(75, 0) point, it detected obstacles three times. Total distance moved by the autonomous navigation robot was 740cm. At the second start, while driving straight line at B(128, 0) point, autonomous navigation robot didn't detect any obstacle. Total distance moved by the autonomous navigation robot was 650cm. At the third start at C(175, 0) point, autonomous navigation robot detected obstacle twice and turned 4 times. Total distance moved by the autonomous navigation robot was 680cm. Finally, the autonomous navigation robot decides that the best course is starting point of B because the moving distance was the shortest.

## **6. Conclusion**

In this paper, we have shown the setting up of the model theory of the autonomous navigation mechanism of power wheelchair for controller design. Though the control method we selected in this paper is simple and classic, it is effective. Because we found that classic control method is based on accurate system model, while control under micro scale has the property of complex nonlinear, time-varying, uncertainty and incompleteness. According to the mechanism human collision avoidance, a unique technique of real-time obstacle avoidance for redundant robot is proposed in the paper.

By the use of fuzzy control system we were able to develop a simple yet effective mechanism to achieve the desired goal.

Finally, simulation and experiment results validate the proposed method. In addition, the presented algorithm can be applied to the obstacle avoidance for redundant robot and mobile robot in time-varying environment.

Furthermore, the future work related to moving obstacle detection has introduced control architecture of a reactive layer for the autonomous traveling of a robot. In addition, the application of Kalman Filter led to the minimization of sensor and system errors, and algorithms using geometrical methods have been introduced to ensure simple avoids with less computation aimed to improve real-time abilities. Studies on new algorithms will be necessary so that voidance of complex obstacles may be possible through the sensor fusion of sonar, IR and others.

## References

- [1] J. S. Albus, "4D/RCS: A reference model architecture for intelligent unmanned ground vehicles," Proc. of the SPIE Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, April 1-5, 2002.
- [2] R. A. Brooks, "A robust layered control system for a mobile robot," IEEE Journal of Robotics and Automation, Vol. 2, No. 1, pp. 14-23, 1986.
- [3] M. Hans and W. Baum, "Concept of a hybrid architecture for Care-O-bot," Proceedings of the IEEE International Conference on Robot and Human Interaction, RO-MAN, Bordeaux-Paris, France, pp. 407-411, 2001, 2001.

- [4] G. H. Kim, W. J. Chung, M. S. Kim and C. W. Lee, "Control architecture design and integration of the autonomous service robot PSR," Proceedings 2002 International Conference on Control, Automation, and Systems, Muju, Korea, 2002.
- [5] T. B. Kwon, J. B. Song and S. Y. Lee, "Improved exploration algorithm using reliability index of thinning based topological nodes," Proceeding of 2005 International Conference on Control, pp., Automation and Systems, 2005.
- [6] M. Lindstrom, A. Oreback and H. I. Christensen, "BERRA: A research architecture for service robots," Proceedings of the IEEE Conference on Robotics and Automation, San Francisco, CA, USA, pp. 3278-3283, 2000.
- [7] K. H. Low, W. K. Leow and M. H. Ang, Jr., "A hybrid mobile robot architecture with integrated planning and control," Proceedings of 1<sup>st</sup> AAMAS'02, Bologna, Italy, pp. 219-226.
- [8] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios," IEEE Trans. on Robotics and Automation, Vol. 20, No. 1, pp. 45-59, 2004.
- [9] Rosemann, M., & Recker, J. "Context-aware process design: Exploring the extrinsic drivers for process flexibility". T. Latour & M. Petit. 18th international conference on advanced information systems engineering. Proceedings of workshops and doctoral consortium. Luxembourg: Namur University Press. pp. 149-158, 2006.
- [10] B. Schilit, N. Adams, and R. Want. "Context-aware computing applications" (PDF). IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94), Santa Cruz, CA, US. pp. 89-101, 1994.
- [11] Schilit, B.N. and Theimer, M.M. "Disseminating Active Map Information to Mobile Hosts". *IEEE Network* 8 (5): 22–32, 1994.
- [12] Dey, Anind K. "Understanding and Using Context". *Personal Ubiquitous Computing* 5 (1): 4–7, 2001.

- [13] Schmidt, A.; Aidoo, K.A.; Takaluoma, A.; Tuomela, U.; Van Laerhoven, K; Van de Velde W. "Advanced Interaction in Context". 1st International Symposium on Handheld and Ubiquitous Computing (HUC99), Springer LNCS, Vol. 1707. pp. 89-101, 1999.
- [14] Schmidt, Albrecht. "Ubiquitous Computing - Computing in Context". PhD dissertation, Lancaster University, 2003.
- [15] Albrecht Schmidt, Michael Beigl and Hans-W. Gellersen. "There is more to Context than Location". *Computers & Graphics Journal, Elsevier* **23**, December 1999.
- [16] International Conference on Ubiquitous Computing (Ubicomp).
- [17] IEEE International Conference on Pervasive Services (ICPS).
- [18] Borenstein and Koren: *edge-detection, certainty grids, and potential field methods*, 1991.
- [19] Moravec and Elfes, 1985.
- [20] ABCM Symposium Series in Mechatronics - Vol. 2 - pp.250-257 Copyright © 2006 by ABCM.
- [21] Koren and Borenstein, 1991.