VISION-BASED TRAJECTORY TRACKING ALGORITHM WITH OBSTACLE AVOIDANCE FOR A WHEELED MOBILE ROBOT

by

Xiusong Yang

Under the Supervision of Dr. Abdelhamid Tayebi

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science
in Control Engineering

Lakehead University, Thunder Bay, Ontario, Canada

August 2005

Your file Votre référence
ISBN: 978-0-494-15650-6
Our file Notre référence
ISBN: 978-0-494-15650-6

Abstract

Wheeled mobile robots are becoming increasingly important in industry as means of transportation, inspection, and operation because of their efficiency and flexibility. The design of efficient algorithms for autonomous or quasi-autonomous mobile robots navigation in dynamic environments is a challenging problem that has been the focus of many researchers during the past few 'decades.

Computer vision, maybe, is not the most successful sensing modality used in mobile robotics until now (sonar and infra-red sensors for example being preferred), but it is the sensor which is able to give the information "what" and "where" most completely for the objects a robot is likely to encounter.

In this thesis, we deal with using vision system to navigate the mobile robot to track a reference trajectory and using a sensor-based obstacle avoidance method to pass by the objects located on the trajectory. A tracking control algorithm is also described in this thesis. Finally, The experimental results are presented to verify the tracking and control algorithms.

Acknowledgements

It is a pleasure to thank many people who made this thesis possible.
First of all I would like to express my gratitude to my advisor, Dr. Abdelhamid Tayebi. He has always found time to discuss whatever was on my mind, from mathematical details to control methods.
I wish to thank professors and graduate students in Lakehead University for their care and attention.
Lastly, and most importantly, I wish to thank my parents and my wife Fengling for being there when I needed support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For centuries, people have been interested in building machines that can think and make decisions based on the environment around them. To satisfy this goal, researchers have increasingly explored robotics which is a challenging research area and requires simultaneous application of many disciplines.

Mobile robotics is an important branch of robotics. During more than four decades of research, the mobile robot was changed from a computer on wheels with a little ability to sense some environmental properties into an intelligent machine which is able to identify features, detect patterns, learn from experience, build maps, and navigate in dynamic environments. Presently, mobile robots are becoming increasingly important as means of transportation, surveillance, inspection, cleaning and entertainment.

1.1 Development of Mobile Robots

The Artificial Intelligence Center at Stanford Research Institute developed a mobile robot with a vision system (SHAKEY) in 1969, which is the infancy of robotics (http://www.sri.com). SHAKEY was the first mobile robot to use artificial intelligence to control its actions, and the first prototype that used a mobile cart with a

TV camera, an optical rangefinder and binary tactile sensors. Endowed with a limited ability to perceive and model its environment, SHAKEY performed tasks that required planning, route-finding, and rearranging of simple objects.

In the 1970s, the Lunar rover was developed at the Jet Propulsion Laboratory (http://www.sti.nasa.gov). It was designed for planetary exploration. Using a TV camera, laser range finder and tactile sensors, the robot can recognize its environments and categorize them as traversable, not traversable and unknown.

In the 1980s, FLAKEY was developed in the Artificial Intelligence laboratory at Stanford. FLAKEY had real-time stereo vision system to distinguish and follow people, and the DECIPHER speech recognition system to respond to spoken commands. FLAKEY's computer included one of the newest personal workstations at that time, giving it far more intelligence than its predecessor.

In 1997, NASA's Mars Pathfinder delivered the Sojourner Rover to Mars. Sojourner sent back images of its travels on the distant planet. Also, in this same year, Honda showed the P3, an extraordinary prototype in humanoid robotic design (http://www.honda-robots.com).

In the future, Robots' roles can be more important in assisting and extending the reach and vision of humans, rather than trying to duplicate them.

1.2 Literature Review

1.2.1 Mobile Robot Navigation

The basic function of an autonomous mobile robot is to navigate in an at least partially unknown environment without collision with any obstacle. The mobile robot navigation problem was concluded by (Durrant-Whyte and Leonard 1991) in following three questions. First, "Where am I?", which is concerned with environmental understanding and the robot's self-localization. Second, "Where am I going?", which deals with the robot's mission planning. Finally, "How do I get there?", which deals with the robot's path planning. For solving these problems, the following fundamental elements have to be studied: locomotion mechanism, control system, sensing, environmental mapping, path planning, and localization (Spero 2004).

There are vast solutions to the robot navigation problem, which can be classified into different categories with respect to different standards.

Depending on whether and how to use the environment's map, the robot navigation problem can be classified into three groups (DeSouza and Kak 2002): Map-Based Navigation System, which depends on user-created geometric models or topological maps of the environment. Map-Building-Based Navigation System, which uses sensors to construct their own geometric or topological models of the environment and then uses these models for navigation. Finally, Mapless Navigation System, which uses no explicit representation about the environment, but rather generate motions based on visual or other observations.

Depending on the types of sensors which the robot used for positioning, robot navigation problem can be classified into relative navigation and absolute navigation. The relative navigation utilizes the internal sensors (odometry, gyroscope) to determine the robot's current location with respect to the robot's initial position, and the absolute navigation uses external sensors (active beacon, GPS) to calculate the absolute position of the robot.

Depending on the types of the maps, the robot navigation problem can be classified into metric and topological approaches. The metric maps represent the world in a global coordinate system which permits a very precise positioning at the goal point (Crowley 1989). The topological maps represent the world as a network of nodes and arcs, which guarantees robustness against getting lost (Cassandra *et al.* 1996).

Depending on the path planning area, the navigation problem can be classified into global and local navigation. The global navigation considers all environmental information and plans complete paths from initial to target (Lumelsky and Stepanov 1987). The local navigation relies on local information about nearby obstacles for avoiding the obstacles (Khatib 1986).

There still are many other navigation categories. For instance, Behavior-based approaches which rely on the interaction of robot actions with the environment to navigate (Jin and Xie 2000). Landmark-based methods which rely on the recognition of landmarks to keep the robot localized topologically (Simmons and Koenig 1995). Grid-based methods rely on metric maps of the environment to localize the robot (Moravec and Elfes 1985). And feature-based approaches which tend to view the localization problem as a matching problem between sensor observations and target model features (Arras *et al.* 2000).

The traditional approach to autonomous mobile robot navigation is to estimate the position and orientation through internal sensors (such as odometry and gyroscopes) with the help of a map of the work space. However, the errors of the internal sensors (systematic and non-systematic) as well as the unknown environment conditions make it necessary to use additional external sensors (such as tactile sensors, global positioning systems, sonar and machine vision). In all of these external sensors, vision can arguably provide the richest source of sensory data, which gives robots a great support to attain rudimentary tasks such as navigation and object manipulation (Corke and Hutchinson 2000).

1.2.2 Visual Servoing

The term of visual servoing was coined at Stanford Research Institute by (Hill and Park 1979). In 1980, Weiss made the distinction between position-based and image-based visual servoing. In Position-Based Visual Servoing systems (PBVS), or we can call it 3D visual servoing systems, features are extracted from an image, and subsequently used to compute a 3D reconstruction of the Euclidean environment. The error used in the control law is computed in the 3D Cartesian space (Corke and Hutchinson 2000). The Image-Based Visual Servo system (IBVS), or we can call it 2D visual servoing system is a model-free control approach since it does not need the knowledge of the 3D model of the target. The control error function is expressed directly in the 2D image space. In 1989, Feddema demonstrated the first IBVS system (Feddema and Mitchell 1989; Feddema 1989; Feddema *et al.* 1991). At the same time, Rives formalized the IBVS approach and investigated various features such as points, lines, and ellipses (Rives *et al.* 1989). In 1991, Wilson and colleagues have progressed the PBVS approach, and use extended Kalman filtering for pose estimation and feature prediction (Wang 1992; Westmore and Wilson 1991; Wilson 1993). In 1993 Corke introduced feature prediction and feedforward control techniques to achieve stable high performance feature tracking (Corke and Good 1993). Malis and Chaumette (Malis *et al.* 1999; Chaumette 1998) introduced the first hybrid PBVS and IBVS system, the "2.5D approach". Hybrid visual servoing (2.5D visual servoing) is based on the estimation of the partial camera displacement from the current to the desired camera poses at each iteration of the control law.

1.2.3 Obstacle Avoidance

All mobile robots need to consider collision avoidance, which is the key issue to successful applications of mobile robot systems. Edge-detection is a simple way to avoid obstacles (Crowley 1989; Kuc and Barshan 1989). In this method, the robot tries to determine the position of the vertical edges of the obstacle and considers the line which connects the two edges representing one of the obstacle's boundaries. And then, the robot attempts to steer itself around either edge (Borenstein and Koren 1991).

A method for representation of obstacles by certainty levels in a grid model has been suggested by (Elfes 1985), which used the certainty grid for off-line global path planning. This method is especially suitable for the inaccurate sensors (such as ultrasonic sensors), and it allows adding and retrieving data in real time and enables easy integration of multiple sensors. Moravec and Elfes also describe the use of certainty grids for map-building (Moravec and Elfes 1985; Moravec 1986).

The general idea of potential field methods was proposed by (Khatib 1986). Khatib computed an artificial potential field that obstacles exert repulsive forces, while the target applies an attractive force to the robot. The combination of the two forces creates a potential field with respect to information about the environment. The robot can reach the target and avoid obstacles by following the steepest gradient of the potential field from a start position. Krogh enhanced this concept further through considering the robot's velocity nearby obstacles (Krogh 1984). And in (Newman and Hogan 1987), the construction of potential functions was introduced by combining individual obstacle functions with logical operations.

The Virtual Force Field (VFF) method was proposed by (Borenstein and Koren 1989). This method uses a two dimensional Cartesian histogram grid for obstacle representation. Each cell in the grid has a certain value which represents the confidence that an obstacle exists at that location. And the potential field idea is applied to the grid that the sensor values are used efficiently to guide the mobile robot. In order to overcome the drawbacks in the VFF approach and find the optimum path to reach the target, the Vector Field Histogram (VFH) method (Borenstein and Koren 1991), VFH+ method (Ulrich and Borenstein 1998) and VFH* method (Ulrich and Borenstein 2000) were proposed.

In 1997, Nourbakhsh presented a passive vision system which provides coarse depth information efficiently and locates static and dynamic obstacles with adequate accuracy for obstacle avoidance (Nourbakhsh *et al.* 1997).

(Fox *et al.* 1997) presented the dynamic window approach which avoids the obstacles by searching in the velocities space in order to maximize an objective function. The terms of this function include the measure of the progress toward the target position, the position of the nearest obstacles and the forward velocity of the robot. This method is derived from the robot's motion dynamics and suitable for high speed motion (Koike *et al.* 2003). In (Brock and Khatib 1999), a generalization of the dynamic window approach is described, which combines methods of motion planning in an attempt to create a global reactive behavior at high speed.

1.3 Thesis Objective

The goal of this thesis is to use the B21R mobile robot's vision system to correctly identify a black trajectory on the floor, and then the mobile robot performs certain tracking algorithm to follow the trajectory. During the tracking process, if there are some obstacles located on the trajectory, the robot will detect the obstacles using sonar sensors and pass by the obstacle by using certain obstacle avoidance program. The trajectory was realized by a black tape on the floor (refer to figure 1.1), there are two half circles at both ends, with a radius of four feet. These two half circles are connected by two parallel lines which both are eight feet long.

Since the visual servoing control is only based on the regulation of the projected

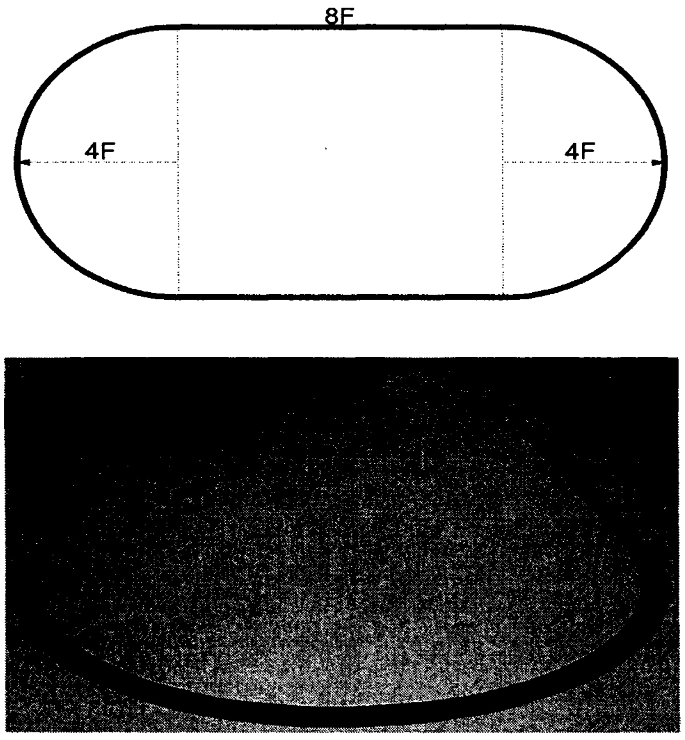


Figure 1.1: Reference Trajectory

target features into the camera image plane, it cannot be used directly to make the robot avoid the obstacles (Cadenat *et al.* 1999). In that case, the vision system needs couple with proximity sensors (laser rangefinder, ultrasonic sensor) to localize and pass by the obstacles. A popular way in robotics for dealing with obstacle avoidance problems is within the behavior based control architecture. The main methodology is to identify different controllers for a complex control task with individual robot behaviors. The main advantage of this approach is that it makes the system modular, which both simplifies the design process as well as offers the possibility to add new behaviors to the system without causing any major increase in complexity (Hu *et al.* 2003). The whole control process of this project can be decomposed into three behaviors: searching trajectory, tracking trajectory and obstacle avoidance. At the beginning, the B21R mobile robot tries to locate the trajectory by using its vision system. If the robot cannot find the trajectory in the current camera image, it will keep rotating counterclockwise until the trajectory is found. Thereafter, the tracking algorithm guides the robot to follow the trajectory. The trajectory tracking behavior should be interrupted when an obstacle is detected since the obstacle avoidance behavior is a necessary safety measure, and it should have higher priority than the trajectory tracking one. After the obstacle is bypassed using a sensor-based obstacle avoidance behavior, the searching and tracking behaviors will be resumed.

1.4 Chapters Summaries

The first chapter presents past research done in the area of mobile robotics and summarizes the research objective. The second chapter describes the components (software and hardware) of the B21R mobile robot used in this thesis, especially the sensors' functions, principles, advantages and disadvantages. The third chapter explains the vision system of the B21R mobile robot and the threshold method which was used to process the image. In the fourth chapter, the vision-based trajectory tracking algorithm is presented in detail. The fifth chapter explains the B21R's mechanical characteristics and gives the kinematic model of the B21R mobile robot. In the sixth chapter, we provide the tracking control law used in the thesis and discuss its performance. The seventh chapter describes the sensor-based real-time obstacle avoidance method. The eighth chapter summarizes the contributions of this thesis and proposes future work. The ninth chapter is the appendix which describes the important issues related to the B21R's operation, maintenance and calibration.

Chapter 2

Components of the B21R Mobile Robot

2.1 Software Environment

The "Mobility Robot Integration Software" (we call it Mobility in this thesis) has been installed in the B21R mobile robot's Linux Operating System. This software uses the "Common Object Request Broker Architecture" (CORBA) standard and the "Interface Definition Language" (IDL) to define the "Mobility Robot Object Model". The Mobility software system consists of distributed, hierarchically organized objects which represent abstractions of the whole robot's sensors, actuators, behaviors, perceptual processes and data storage. The objects provide a flexible model that can be reconfigured as new algorithms and new applications.
The Mobility development environment offers a set of common graphical interfaces that control Mobility-supplied objects. The Mobility interfaces are called the Mobility Object Manager (MOM). With MOM, one can observe, tune, configure and debug the components of the Mobility robot control programs. MOM also allows one to launch a variety of object viewers that provide visualization of the robot's actuators, sensors, algorithm outputs and debugging information.

Mobility uses CORBA to communicate among objects. The implementation of CORBA used by Mobility has a shared library called the Object Request Broker (ORB). The ORB is a communication management library that allows transparent access to objects in different address spaces on the same or other computer systems. Mobility also includes interface libraries that are compiled versions of the interfaces that define the "Mobility Core Object Model" (iRobot Rev.6).
In Mobility, a robot is a hierarchically arranged collection of elements. The top level is the "System Module Component" that contains the separate subsystems of an individual robot. The subsystems contain properties that allow client programs to discover and dynamically update "State Objects", based on the state of robot sensors and actuators.

2.2 Hardware Description

The B21R hardware system consists of three main sections (see Figure 2.1): the Base, the Enclosure and the Console. The Enclosure and the Console are physically attached. However, the Enclosure can be separated from the Base.
The Base contains the wheels, the mechanical drive and steering components, batteries, motors, motor control electronics and tactile sensors. Many low level functions are handled entirely on the Base (such as dead reckoning position integration, motor current sensing and battery voltage sensing). The Base is very low and heavy in order to improve the stability of the entire system.

The Enclosure contains the main computer, tactile sensors, much of the power distribution system, sonar, IR sensors, communication equipment and the interface console.
The Console contains robot's rFLEX robot control system knob, the emergency kill switches, joystick port and serial ports. The camera and the pan-tilt unit which are important parts of the vision system are established on the top of the Console.

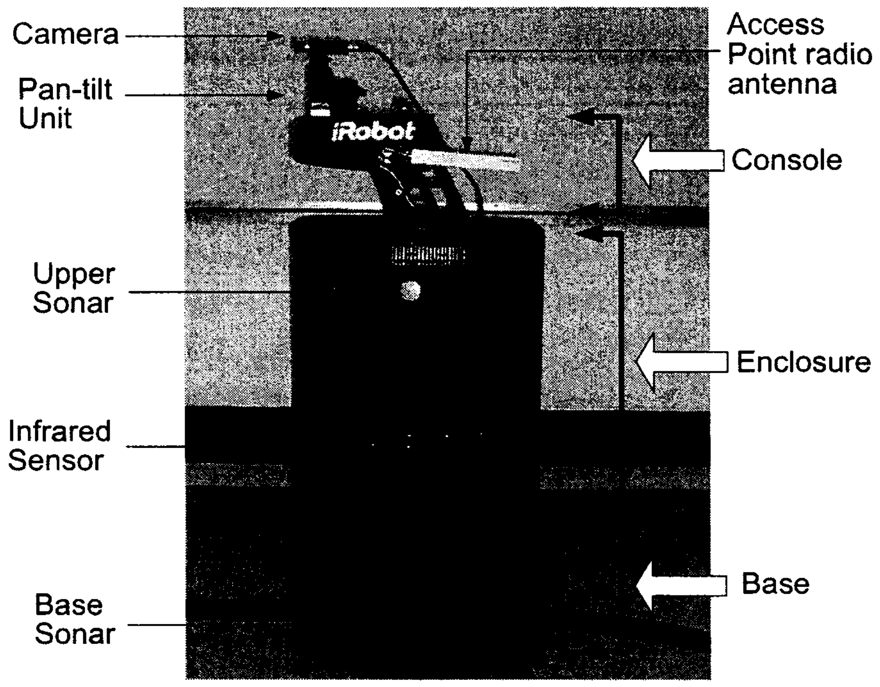


Figure 2.1: Components of the B21R Mobile Robot

The information flow of the B21R mobile robot is illustrated in figure 2.2. All sensors including monocular camera, sonar, infrared and contact sensors send their feedback to the on-board computer. The brain of the robot (on-board computer) processes the data and gives commands to control the robot and the pan-tilt unit, based on its environments.

2.3 Sensing

As automatic guided vehicles, mobile robots must have a means of sensing and should be able to anticipate uncertain changes in their environments. As the important part of the hardware of the B21R mobile robot, sensors perceive information and obtain environmental properties, such as detecting obstacles by sonar sensors, recognizing surface colors with the vision system and obtaining the robot's position with respect to the initial location through odometry. Various sensors have been used in building

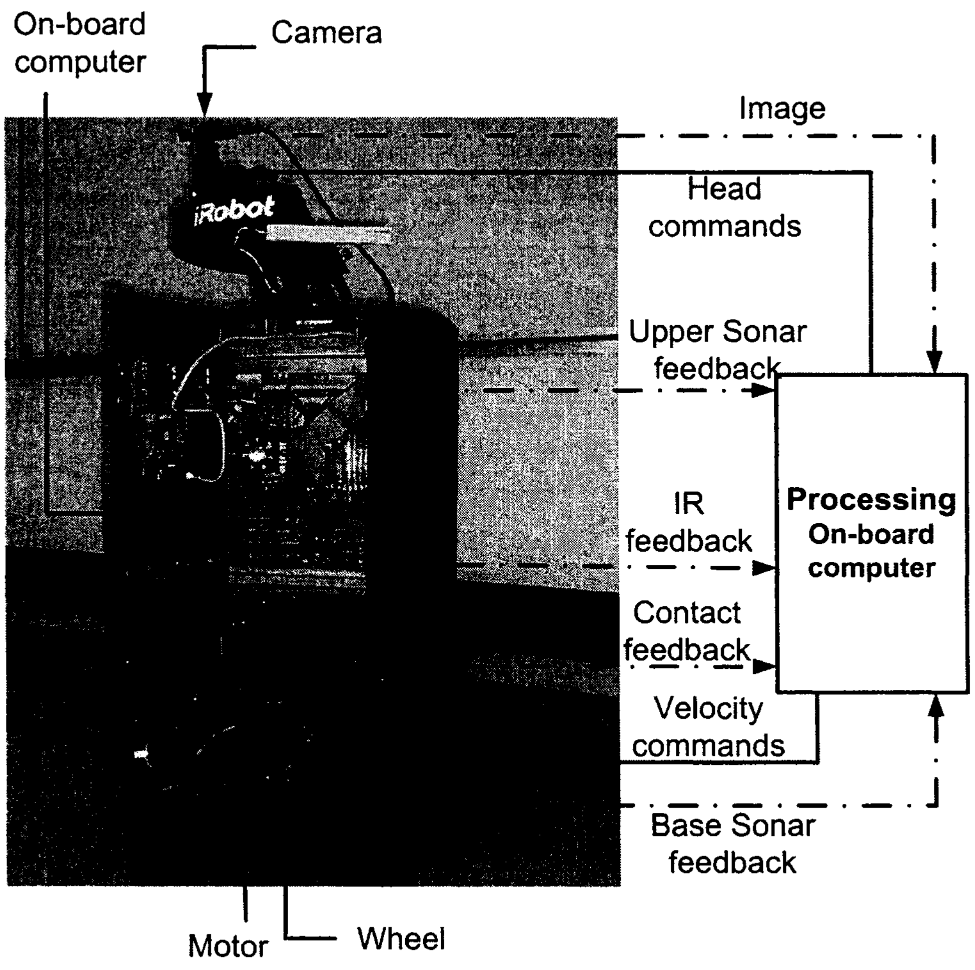


Figure 2.2: The Basic Flow of Information Around the B21R Robot System

mobile robot, which can be broadly categorized into two groups: external and internal state sensors.

2.4 External Sensors

A mobile robot needs to be able to observe the outside world to successfully navigate within its environment. Such a task is performed by using a series of external sensors including sonar sensors, infrared sensors, tactile sensors and machine vision.

2.4.1 Sonar Sensors

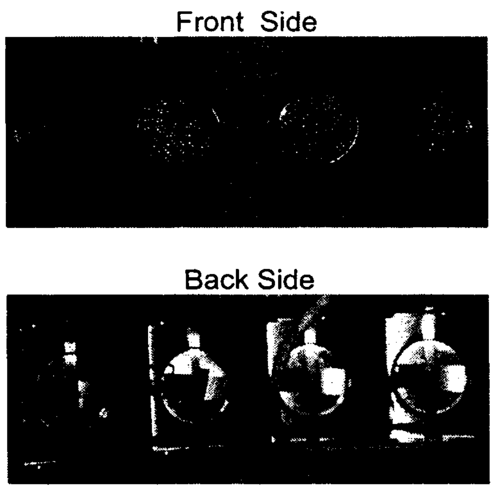


Figure 2.3: B21R Sonar Sensors

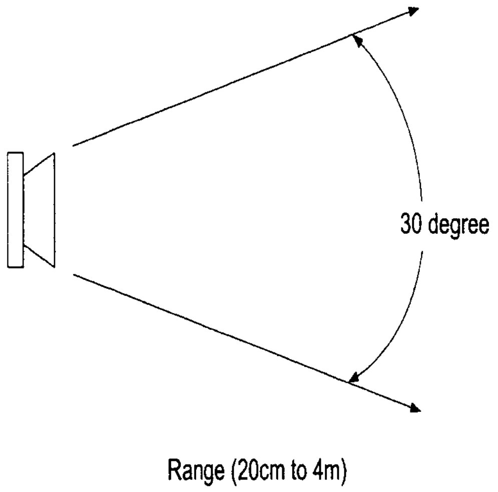


Figure 2.4: Sonar Sensor Range

SONAR (see figure 2.3 and figure 2.4), an acronym for SOund Navigation And Ranging, models the contours of an environment based on how the sonar sensor catches and throws back sound waves. iRobot's B21R mobile robot reads its sonar sensors about three times per second. For each reading, the total time between the generation of the ping and the receipt of the echo, coupled with the speed of the sound in the robot's environment, generates an estimate of the distance to the object that bounced back the echo. Each sonar sensor detects obstacles in a cone-shaped range that starts out with an angle of about 30 degrees and spreads outwards.

The B21R mobile robot has two groups of sonar sensors. One is the upper sonar group, located on each smart panel of the Enclosure part; another is the base sonar group located on each smart panel of the Base part. There are four sonar sensors on each Enclosure's panel (total 24 upper sonar sensors) and three sonar sensors on each Base's panel (total 24 base sonar sensors).

The sonar sensors of the B21R mobile robot are suitable low range sensors for applications of wall and object detection. It has a minimum range of 20 centimeters due to the sonar sensors' reacting time and with a maximum range of 4 meters since sonar sensor's signal is attenuated during transmitting through air.

How the Sonar Sensor Can be Fooled

Sonar sensing is an economical approach of detecting the proximity of objects and is accurate over several meters. However, several drawbacks have to be considered when using sonar sensors. As the speed of sound can be affected by changes in atmospheric conditions, small errors can be introduced when calculating the distance from the obstacle to the robot. And the obstacle's surface characteristics (smooth or textured) as well as the angle at which an obstacle is placed relative to the robot will significantly affect how and even whether the obstacle will be detected. The accuracy of the sonar sensors depends on the span of the transmitted signal. The wider the span, the higher the chance of detecting an obstacle; however narrower spans will locate the obstacle's position with respect to the mobile robot more accurately.

The sonar sensors can be fooled for the following reasons. First, frequent misreading that is caused by either ultrasonic noise from external sources or stray reflections from neighboring sensors (crosstalk). Misreading cannot always be filtered out and they cause the robot to detect nonexistent boundaries. Second, obstacles at steep angles might bounce their echoes off in a completely different direction. Therefore, the obstacles will be ignored by the sonar sensor, as it never received an echo (refer to figure 2.5). In addition, false readings can also be generated when the sonar's ping bounces off an obliquely angled object onto another object in the environment, which then, in turn, returns an echo to the sonar sensors (refer to figure 2.6). This effect, which is called specular reflection, can cause the sonar sensor to overestimate the distance between the robot and the obstacle (see iRobot Rev.3 in the reference list).

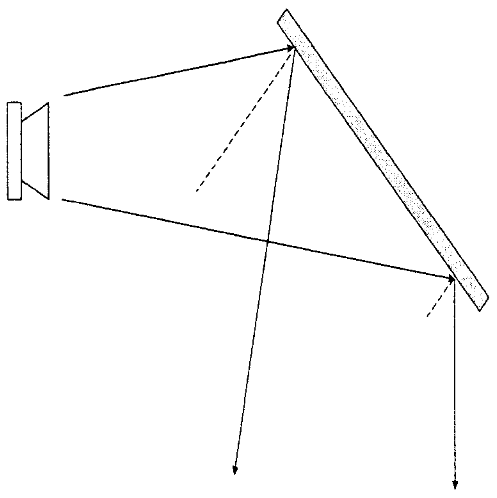


Figure 2.5: Sonar Cannot Receive the Echo

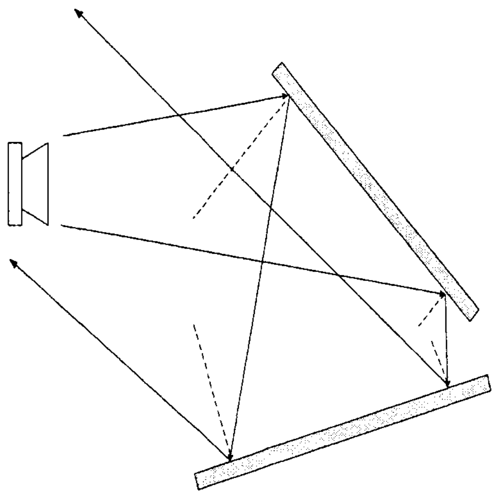


Figure 2.6: Echo is Reflected More Than Once

Avoiding the Sonar Sensor to be Fooled

The B21R mobile robot prevents its sonar from being fooled by using multiple sonar sensors which provide redundancy and enable cross checking (see figure 2.7). Since sonar sensors almost never underestimate the distance of an obstacle, it is a good approach to examine the distances returned by a group of sonar sensors and to use only the lowest values.

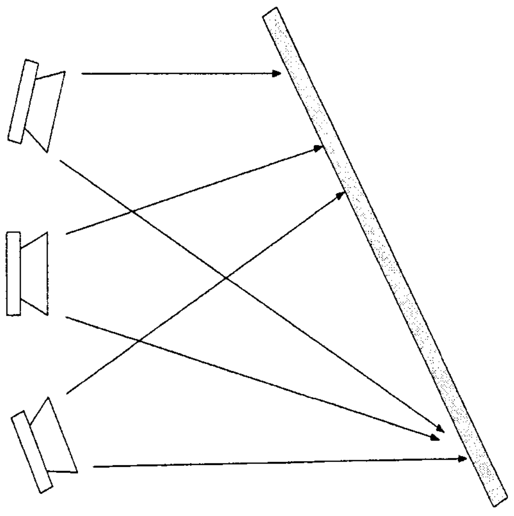


Figure 2.7: Avoiding From Being Fooled

2.4.2 Infrared Sensors

Two main techniques are used in infrared sensors to detect the distance from a particular point to an area of interest: triangulation and intensity. The B21R mobile robot is equipped with the intensity type infrared sensors (see figure 2.8). This kind of infrared sensor comprises a paired infrared emitter and receiver. The emitter sends an infrared beam and the receiver measures the amount of infrared which reflects back from nearby objects. The distance of the object is determined by the intensity of the received infrared light.

Errors also exist in the infrared sensors' feedback since the intensity of the reflection is usually not a linear relationship with the distance and is different for each sensor. Importantly, the intensity of the received light can be affected by different surface textures and colors.

The infrared sensors used by the B21R mobile robot have the minimum sensing distance of 10 centimeters and the maximum sensing range of 1 meter.

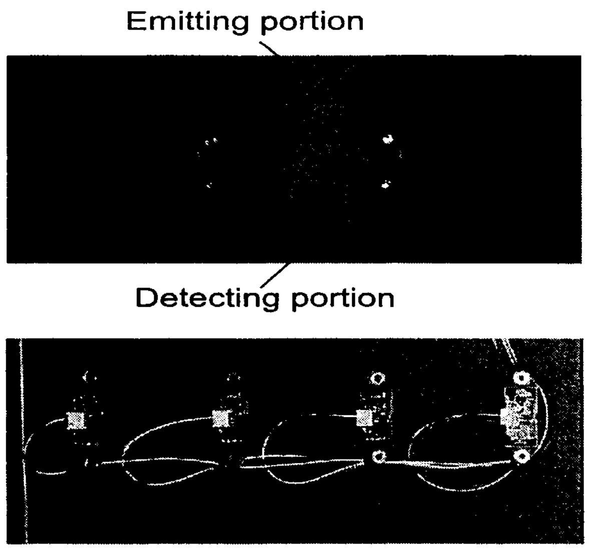


Figure 2.8: B21R Mobile Robot Infrared Sensor

2.4.3 Tactile Sensors

A tactile sensor, which is also called contact sensor or bump sensor, is located on each corner of the panels of the B21R's Base and Enclosure parts (see figure 2.9). The tactile sensor is the final safety sensor if none of the other sensors detect the obstacle. The B21R's tactile sensors just give binary information: touch or no touch. When the B21R mobile robot collides with something and one of the panels is pressed inside a little (almost 2 millimeters), the tactile sensors on this panel will give bump signals to the on-board computer to prevent further damage.

2.4.4 Visual Sensing

The sonar and infrared sensors used for mobile robots are usually limited by low resolution, specular reflections, insufficient dynamic range, and other effects. Comparing with sonar and infrared sensors, visual sensors provide richer and more complete information which is essential for recognizing and understanding the environment in real time.

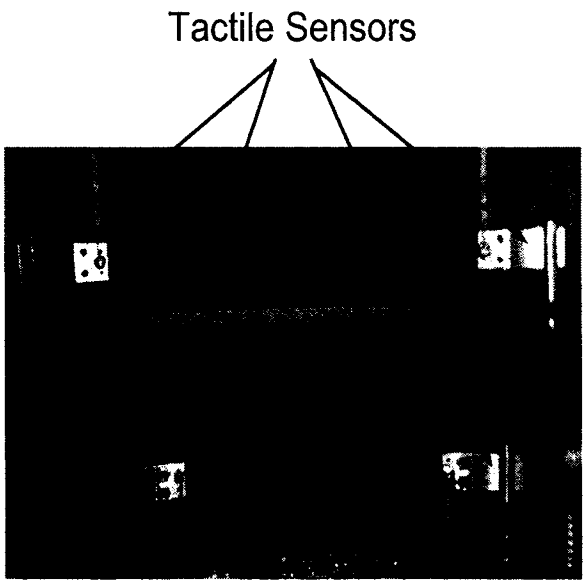


Figure 2.9: B21R Mobile Robot Tactile Sensor

Cameras are the foundation of vision systems. The simplest camera is a pinhole camera, which places the whole world in focus and provides an inverted image. More typical cameras have lenses which gather light and focus the rays from a 3D point in the environment onto a pixel on the imaging device. All cameras must have some sort of image sensors which could be silver plates, films, or CCD sensors. No image sensor is perfectly continuous; hence there is a limit to the resolution of the camera. Color cameras are preferred over black and white in some environments. For instance, in this project, the only difference between the trajectory and the floor is their colors.

The main parts of the B21R mobile robot vision system are one SONY XC-555 Charged Coupled Device (CCD) Camera, which is an ultra-small, high resolution and high sensitivity digital, color camera with a frame rate of 25 frames per second, one pan-tilt unit (Directed Perception PTU-46-17.5), one PCI frame grabber board (Hauppauge) and one pan-tilt controller (refer to figure 2.10).

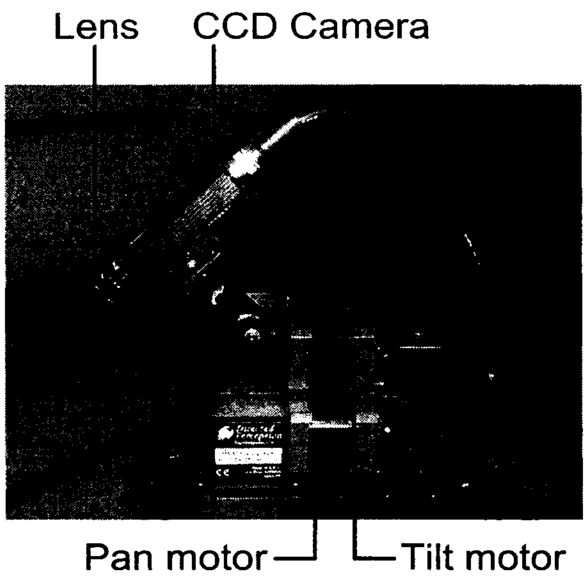


Figure 2.10: B21R Mobile Robot Vision System

2.5 Internal Sensors

Internal state sensors provide feedback on the robot's internal parameters, with no direct reference to the external world. These sensors may include encoders, gyroscopes and accelerometers. The B21R mobile robot in our laboratory is just equipped with odometric sensors: encoders.

2.5.1 Encoders

Before the end of the 15th century, sailors navigated by what was then called "dead" reckoning. This method of navigation was used by Columbus and many other sailors of his era *(http://www.nps.gov/fora/navigation.htm)*. Dead reckoning refers to pose estimation, based on the observation of internal parameters. The simplest form of dead reckoning is often termed as odometry.

Encoders are the most widely used sensors for odometry. There are different types encoders, such as brush encoders, magnetic encoders and capacitive encoders. However, the most popularly used for mobile robots is the optical encoder which enables a linear or rotary displacement to be converted directly into a digital form without intermediate analog-to-digital conversion. The basic principle behind the optical encoder is to alternately interrupt and pass a beam of light between a light emitting diode and a phototransistor as the shaft being monitored is rotated. By counting the number of light-to-dark transitions seen by the phototransistor and knowing the number of slots or holes in the encoder, the resulting rotation of the shaft can be calculated *(http://www.opticalencoder.com/applications.html)*. Optical encoders can be classified further into two categories: incremental and absolute. The incremental ones measure rotational velocity and can infer relative position. The absolute model on the other hand, is best suited for slow or infrequent rotations such as steering angle encoding and can infer velocity. If there is no special requirement, incremental encoders are usually chosen due to their lower cost and simpler interfacing compared to absolute encoders.

2.5.2 Odometry of The B21R Mobile Robot

The Base of the B21R mobile robot is equipped with four wheel encoders that keep track of the revolutions of the wheels as the robot travels within its environment. The robot's motion controller (rFlex) integrates these measurements to attempt to estimate the robot's current position at any time with respect to its original position.

The B21R's rFlex controller takes in the four raw encoders and combines the encoders feedback into two virtual axes: translation axis and rotation axis (see figure 2.11). The information of "virtual encoders" is processed by Mobility system. Mobility consists of many state objetcs and two of them represent the state of B21R's odometry: the "TransformState" object deals with the current location of the robot, and the "FVectorState" object represents the robot's current velocity. Thereafter, Mobility sends rotation and translation velocity commands to the virtual axes of the rFlex controller (a Mobility-based program can get updates from the virtual axes at

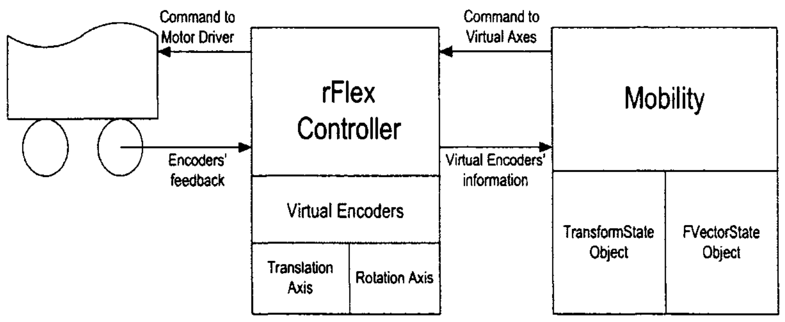


Figure 2.11: Odometry of The B21R Mobile Robot

about 10-15Hz). Finally, the controller gives commands to drive the motors.

2.5.3 Errors in Odometry

Odometry provides highly accurate measurement for short distances. It is economical and allows very high sampling rates. However, the basic principle of odometry is to estimate the robot's pose in an open loop by integrating the internal motion information over time, which leads to the accumulation of errors. Especially, the accumulation of orientation errors will cause large position errors which increase proportionally with the distance traveled by the robot (Borenstein *et al.* 1997). Odometry errors can generally be classified into two categories: systematic and non-systematic (Borenstein and Feng 1996). Systematic errors include unequal wheel diameters, actual wheelbase differs from nominal wheelbase, misalignment of wheels and finite encoder resolution. Systematic errors are usually caused by imperfections in the design and mechanical implementation. Therefore, they are vehicle specific, don't change and accumulate constantly. On most smooth indoor surfaces systematic errors contribute much more to odometry errors than non-systematic errors. Non-systematic errors include travel over uneven floors, travel over unexpected objects on the floor and wheel-slippage. Non-systematic errors are usually caused by the unpredictable environment. Therefore, on rough surfaces with significant irregularities, non-systematic errors are dominant. The problem with non-systematic errors is that they may appear unexpectedly, and they can cause large position errors.

The B21R mobile robot used in the project is a new and advanced mobile robot, and its systematic errors are very small. In our laboratory, the reference trajectory passes in some places where the floor is not flat enough. Therefore, in this project, the non-systematic error is the main problem.

In the experimental result shown in Figure 2.12, the B21R mobile robot followed exactly the same trajectory for three cycles. However, from the odometry feedback, one can see that the robot's tracking trajectories shift almost by the same distance between each cycle (around 0.04 meter). The shift distance is the accumulation of odometry's non-systematic errors. In Figure 2.12, the tracking trajectories do not exactly match the reference trajectory (dashed line) quite well, since the robot's initial position and orientation cannot be measured very precisely.

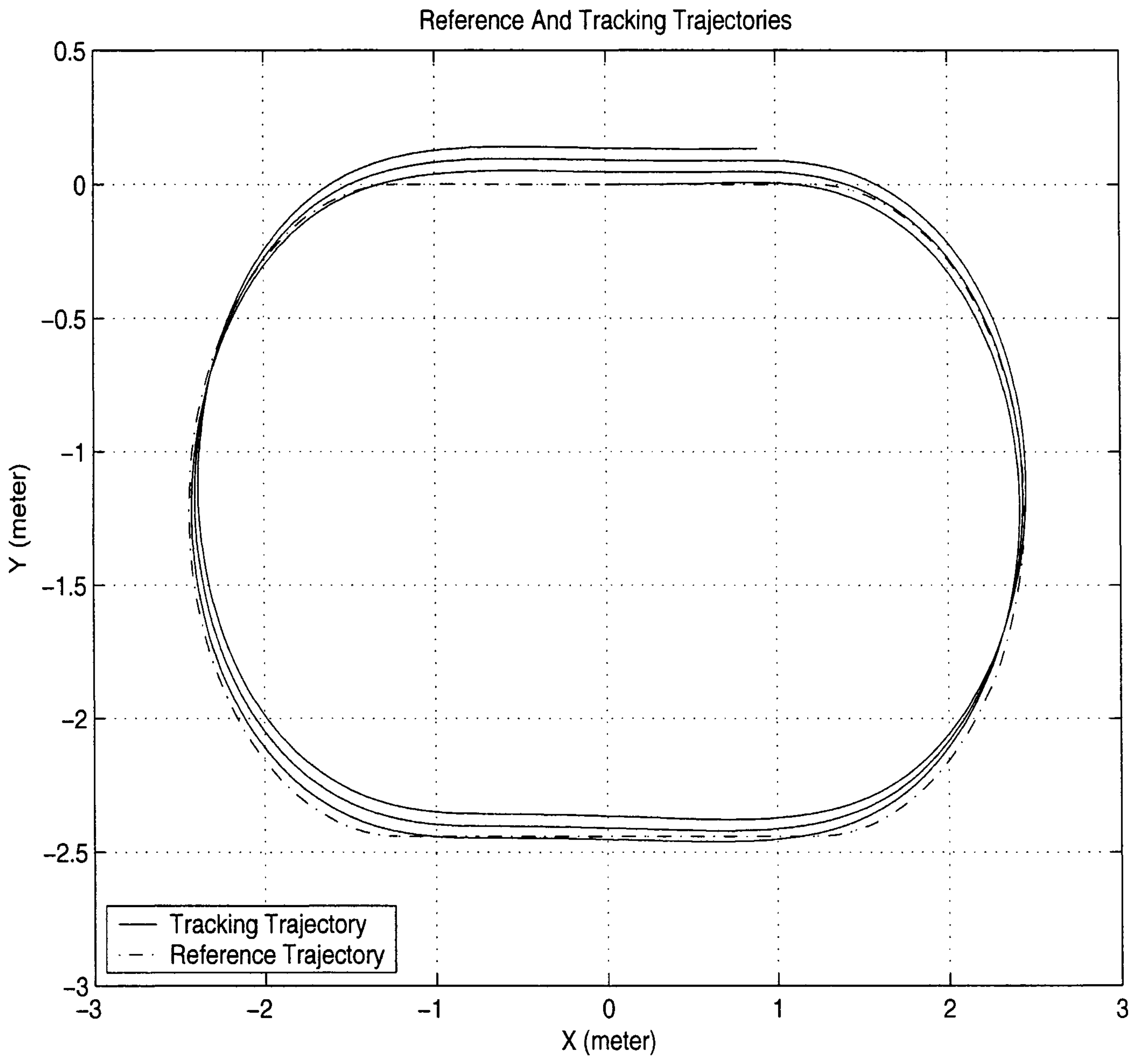


Figure 2.12: Errors During the Tracking Process

Chapter 3

Vision System and Visual Servoing

Vision is one of the most important senses which let intelligent creatures recognize objects, locate objects in space and track objects in motion. Vision is also an interdisciplinary area encompassing geometry, physics, computer science, numerical methods as well as probability and statistics.

The discipline of machine vision emerged in the early 1960s. Scientists handled physics-based vision analysis in the 1970s. Between 1980s and 1990s, researchers focused on stereo vision, real-time vision systems, appearance-based vision, motion and object recognition. In the 21st century, real-time tracking and recognition as well as interactive vision systems attract more attention.

Visual sensing generates an abundance of information from a single image, while usually the area of interest is only a fraction of the image. Therefore, vast machine vision algorithms were generated to locate and interpret certain visual features. These algorithms can be generally classified into two categories: low-level (geometric algorithms) and high-level (understanding algorithms). Geometric algorithms have the ability to take the huge space of images and extract some simpler geometry that can be analyzed. Understanding algorithms are used to recognize objects from the database, which is quite challenging for the general use (Isard and Blake 1998).

An essential problem with using machine vision is the huge amount of data generated by a video camera. Therefore, the essence of the machine vision is the "minimalism principle" which emphasizes on solving the task by using features as basic as possible. During image processing, only the information related to the vision task should be extracted from the images. This kind of information can be quantitative or qualitative, such as using color information to track certain color trajectory and using geometric information to grasp objects.

3.1 Five Frames of Reference

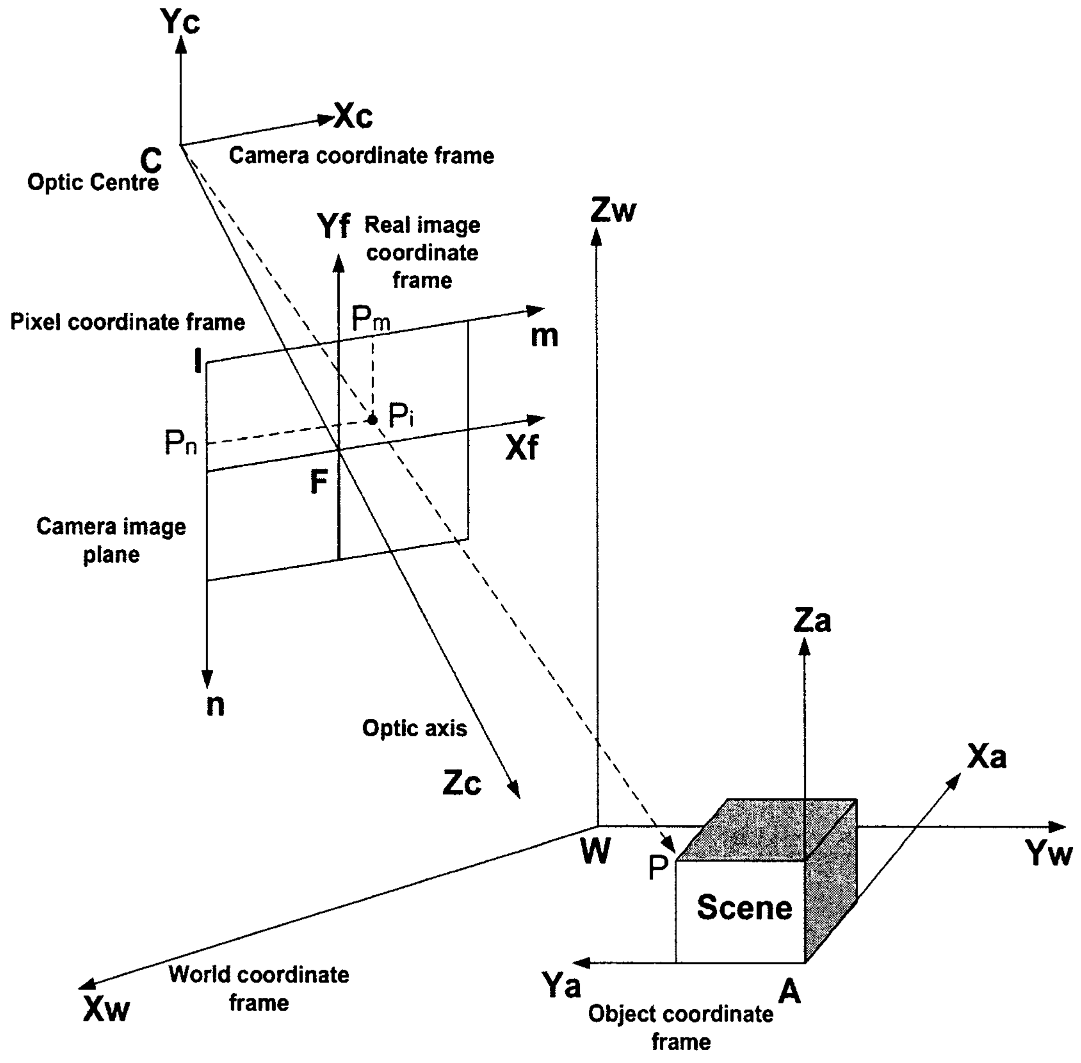


Figure 3.1: Five Frames of Reference

Reference frames are needed in order to do either qualitative or quantitative analysis of scenes. Five frames of reference are necessary for general problems in 3D scene analysis (Shapiro and Stockman 2001). The five types of frames are illustrated in figure 3.1. First is the World Coordinate Frame (W), which is needed to relate objects in 3D. Second is the Object Coordinate Frame (A), which is a frame attached to the object under consideration. The Camera Coordinate Frame (C) is a frame attached to the camera and often needed for an egocentric view. The fourth frame is the Real Image Coordinate Frame (F). 3D points project to the real image plane at coordinates (x_f, y_f, f), where (x_f, y_f) are real numbers, usually in the same units as the world coordinates, and f is the focal length (CF). In all above coordinate frames, coordinates are real numbers along continuous axes. However, coordinates of Pixel Coordinate Frame (I) are integer subscripts of the pixel array. In the pixel array, each point has integer pixel coordinates. Many things about a scene can be determined by analysis of the image in terms of only pixel rows and columns. For example in figure 3.1, the image of the point P in the scene falls within pixel $P_i(P_m, P_n)$, where P_m and P_n are integer row and column, respectively.

3.2 Color Space

Color spaces are a method using several different components to define color. Three of the most prominent color spaces are RGB, YUV, and HSI (Gonzalez and Woods 2001). Images generally come out of the cameras in RGB which is a frequency domain measurement. RGB consists of three components: red, green, and blue. RGB is useful since red, green, and blue pixels can easily be combined to create any color. However, the RGB color space is not an ideal format for manual color classification. The YUV color space is the format in which television broadcasts are transmitted. The Y component is intensity, a weighted average of the R, G, and B values. Displayed alone the Y component creates a grayscale image. The U and V components are chrominance values that can be used with the Y component to obtain R, G, and B values. The HSI color space consists of Hue, Saturation and Intensity, which set up in a cylindrical coordinate system. The HSI color space classifies colors in a way similar to the way that humans do, hence it is intuitive to the human mind.

3.3 Image Format

The CCD camera of the B21R mobile robot takes images and saves them as PPM files. A PPM file includes two parts: a header and the image data. The header consists of at least three parts. The first "line" is a PPM identifier. It can be "*P*3" or "*P*6". The next line consists of the width and height of the image as ASCII numbers, which are 160 and 120 with respect to the CCD camera used in our project. The last part of the header provides the maximum value of the color components for the pixels, which is 255 in our case.

The format of the image data depends on the PPM identifier. If it is "*P*3", the image is given as ASCII text. The numerical value of each pixel ranges from 0 to the maximum value given in the header. The image is stored in top to bottom, left to right order and should not be longer than 70 characters. If the PPM identifier is "*P*6", then the image data is stored in byte format, one byte per color component (R,G,B). "P6" image files are obviously smaller than "P3" and much faster to read. The image files taken by the B21R mobile robot express each pixel using three bytes which show the values of red, green and blue color elements. Each color has 256 gray levels from 0 to 255. The sequence of these three color elements (R,G,B) in the PPM file decides the color of this image. If the sequence of the three elements is out of order, the image will lose its real color.

3.4 Processing the Image Using Threshold

Threshold setting is an "ignore small detail" setting and is a straightforward method which suppresses noise and enhances important features. There are many kinds of thresholds, such as high and low threshold, band threshold, hierarchical threshold and multi-spectral threshold.

Color threshold is satisfied based on some simple models of appearance, such as objects are uniform in color or chromaticity and the foreground is not complex in color or illumination conditions. Unfortunately, the real environments are much different from the ideal models. Since many objects have more than one color and sometimes are easy to be confused with their environments.

The standard of selecting the threshold is to minimize the probability of a segmentation error. One can manually select thresholds or calculate a histogram of the image and acquire the threshold through analyzing the histogram shape. Therefore, threshold selection is subjective and is often determined qualitatively by the analyst.

The particular condition of our project is that a black trajectory was set on the white floor of the laboratory. The task of finding target becomes easy in this condition, because the difference of the gray levels between the black target, whose RGB is (0,0,0), and its white environment, whose RGB is (255,255,255), is very large. In this project, threshold is used to get the binary images and then the vision system can recognize the tracking target. Since our vision control system is a real-time control system, algorithm speed, reliability and robustness are all important factors. Therefore, the threshold process should be fast enough; the robot should not be too sensitive to the threshold levels and it should be robust to changes in lighting. Instead of checking the gray levels of all three color components of one pixel, our algorithm just checks the red color element, since one element is enough to present the objects in this simple condition, and this will increase the image processing speed for almost three times which means treasure in real-time system.

When the environment condition in our laboratory is constant, an image was taken and processed by four different thresholds which are 100, 150, 200 and 240 (see from

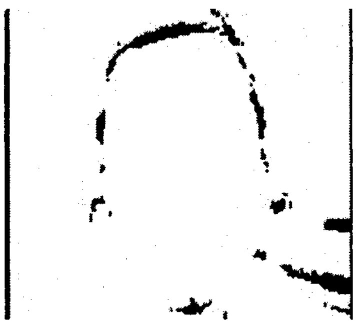


Figure 3.2: Threshold of 100

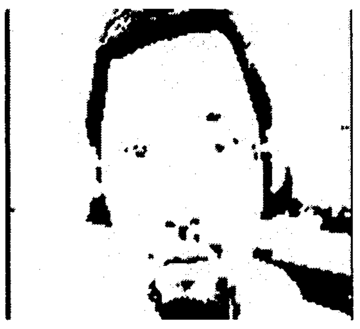


Figure 3.3: Threshold of 150

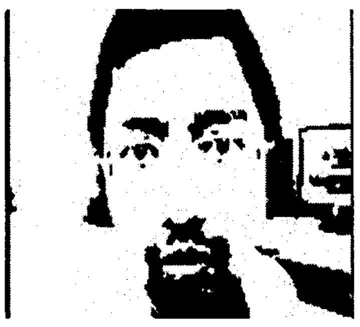


Figure 3.4: Threshold of 200

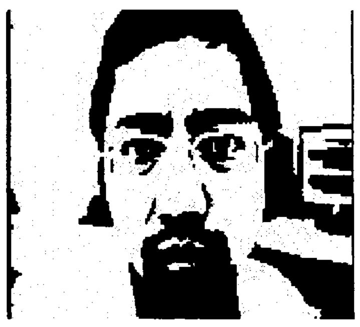


Figure 3.5: Threshold of 240

figure 3.2 to figure 3.5). Through comparing the image effect under different thresholds, we can say that 200 is a satisfied threshold for this environment which gives enough characteristic of the target without small details.

3.5 Environmental Variations Affect the Vision System

In machine vision, the key issue is how to deal with environmental variations. In this project, according to the indoor condition, the principal reason for success in machine vision is to select a lighting arrangement which is suitable for this specific application.

Figure 3.6 shows the lighting condition of our robotics laboratory. There are six ceiling lights on each side of the trajectory. The lights on one side were turned

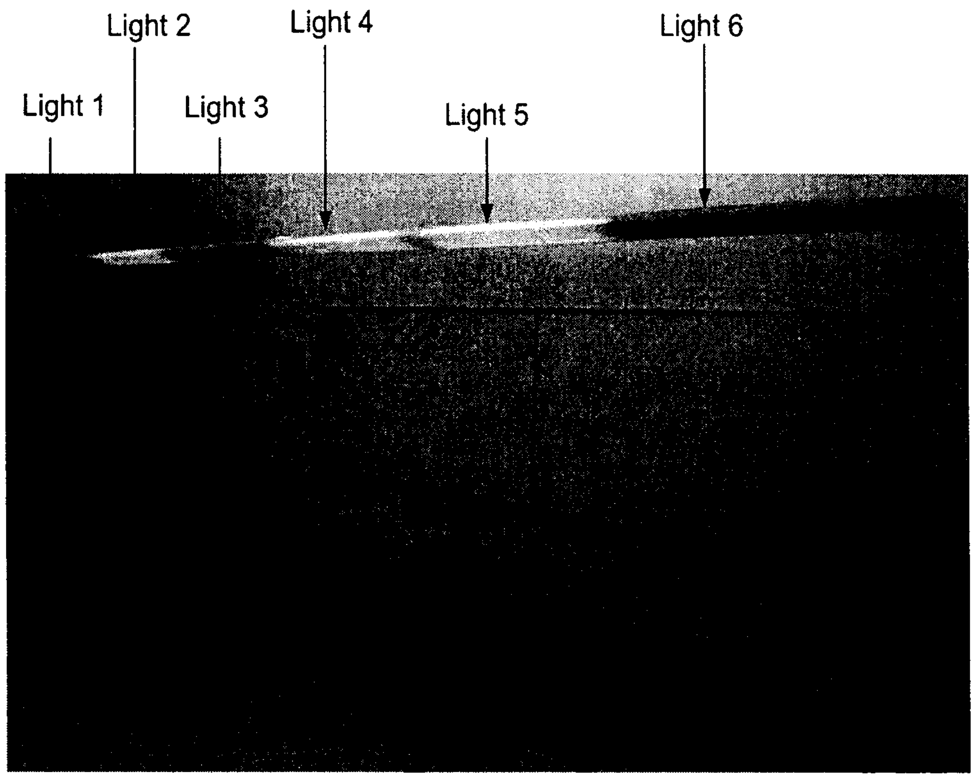


Figure 3.6: The Lighting Condition in the Lab

off in a certain sequence, and the CCD camera took a picture of the trajectory at each different lighting condition. Through comparing these images (from figure 3.7 to figure 3.11), one can conclude that the vision system is heavily affected by the lighting condition, and the middle part of the image is easier to be affected than the top and bottom parts. To improve the robustness of the vision system, the lighting condition in the laboratory should be kept constant and the system should avoid using the middle part of the image to search for the target.

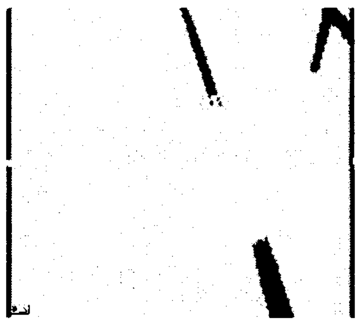


Figure 3.7: Third Light Turned Off

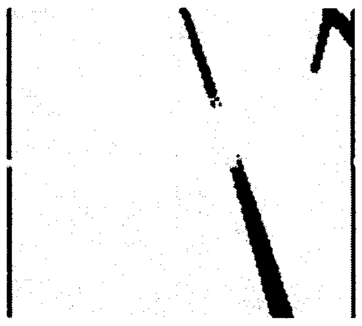


Figure 3.8: Second Light Turned Off

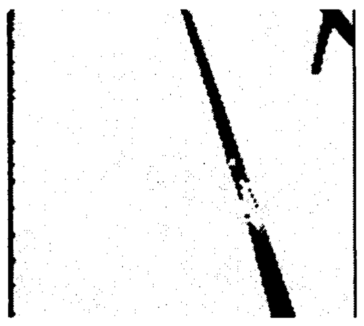


Figure 3.9: 1st and 3rd Light Turned Off

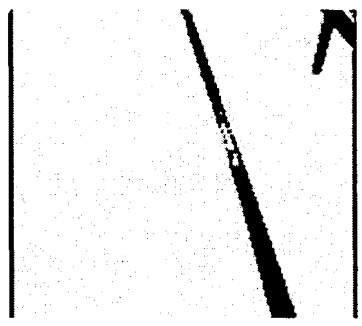


Figure 3.10: 2nd and 3rd Light Turned Off

3.6 Comparing a Sequence of Images

The simplest way to track an object is to develop a method for extracting the object from the environment and to apply the method on each frame. But this method

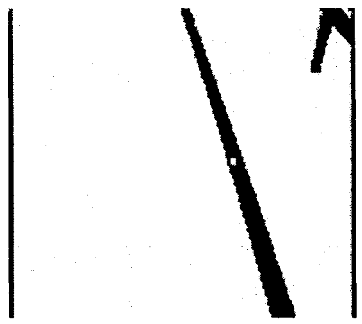


Figure 3.11: 1st, 2nd and 3rd Light Turned Off

overlooks some important aspects in the real world, such as the large amount of information shared between adjacent frames and the fact that objects normally move in predictable ways.

In our project, if the B21R mobile robot just calculates the tracking point from each isolated image, the tracking controller is easy to be disturbed by uncertainties in the dynamic environment (refer to the research video: Before Compare a Sequence of Image). Since the reference trajectory used in this project is continuous and smooth, the robot becomes more robust to disturbances and seems more intelligent by comparing the difference between adjacent images (refer to the research video: After Compare a Sequence of Image).

Chapter 4

Target Localization

Most of the available vision tracking techniques can be categorized into two main groups: feature-based and model-based. The feature-based approach focuses on tracking 2D features such as geometrical elements (points, segments and circles), object contours and regions of interest (Isard and Blake 1996; Hager and Toyama 1998). Model-based methods usually provide a more robust solution, since they are able to predict hidden movement of the object, detect partial occlusions and act to reduce the effects of outlier data introduced in the tracking process (Comport *et al.* 2004). In this project, due to the fact that the tracking target is a static black trajectory taped on the floor, the feature-based approach is used to detect the trajectory in real-time.

4.1 Representing The Robot's Position

To successfully track the trajectory on the floor, the robot needs to know the relation between the tracking object's position and its own location. For example, what is the distance from the tracking target to the robot, as well as what is the angle between the orientation of the robot and the tracking object.

First of all, it is necessary to calculate the robot's position. There are two methods of

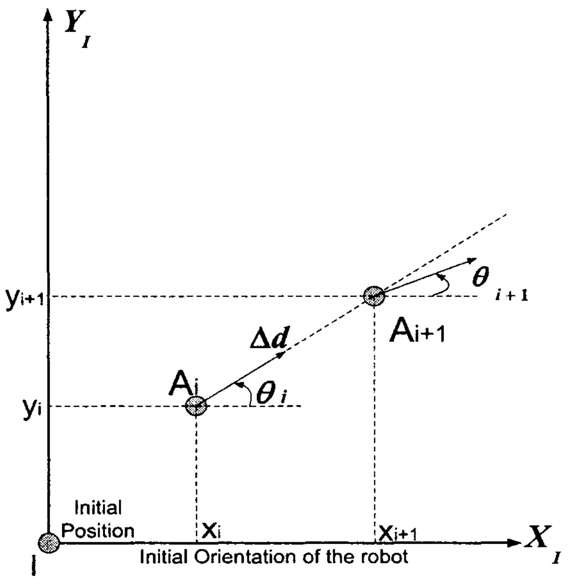


Figure 4.1: Principle of Odometry

robot positioning: absolute and relative methods. Satellite based positioning systems (GPS) can be used to provide absolute position and orientation information. However the use of GPS remains limited to outdoor environments due to satellite signal blockage inside buildings. In this project, since there is no direct way to measure the robot's position, the incremental optical encoders had to be used to relatively derive the robot's localization information. The encoders collect two parameters of the robot's motion: the total distance it travels and the total angle by which it rotates. The B21R's position can be estimated by using the distance by which the mobile robot translates between two adjacent sampling time and the robot's current orientation with respect to the initial position (refer figure 4.1).

$$\begin{aligned} x_{i+1} &= x_i + \triangle d \cos\theta_i, \\ y_{i+1} &= y_i + \triangle d \sin\theta_i, \end{aligned} \tag{4.1}$$

where, (x_i, y_i) represents the robot's coordinates at time i, (x_{i+1}, y_{i+1}) represents the robot's coordinates at time i+1, $\triangle d$ is the distance that the robot traveled between time i and time i+1, θ_i and θ_{i+1} are the orientations of the robot at time i and time i+1, respectively.

It should be known that, in practice, the distance $\triangle d$ may not be in a line and the orientation θ_i may not be close to θ_{i+1}. However, due to the very short sampling time of the odometry, we assume that $\triangle d$ is in a line and the orientation θ_i is close to θ_{i+1} .

The B21R mobile robot's odometry has a special limitation that it's orientation

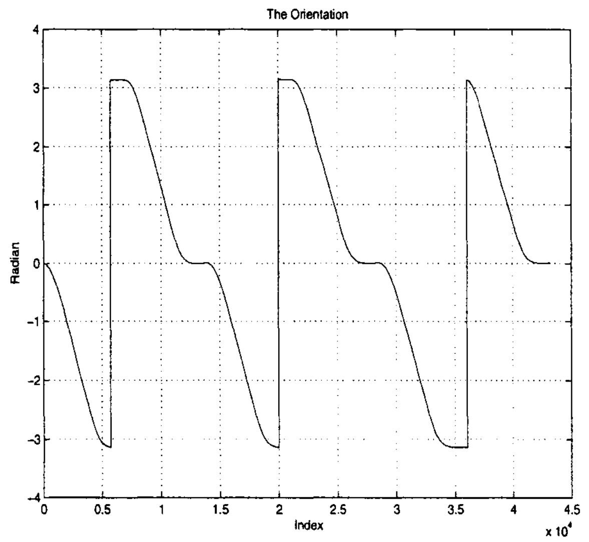


Figure 4.2: Changing of the Orientation θ (during the robot turning right)

should be presented between $-\pi$ and π (Refer to figure 4.2). For instance, when the robot stays at the initial position, all encoders are set to zero. If the robot starts to move and turn right, the robot's orientation will keep decreasing and changing gradually from zero to $-\pi$. If the robot keeps turning right, the robot's orientation will jump from $-\pi$ to π and then decrease to zero again. The situation is similar when the robot keeps moving and turning left; the robot's orientation θ will increase from 0 to π, jump from π to -π, and then, increase to zero. Although the robot's orientation jumps and is limited in $[-\pi, \pi]$, the equation which expresses the robot's position will not change and always be the equation 4.1.

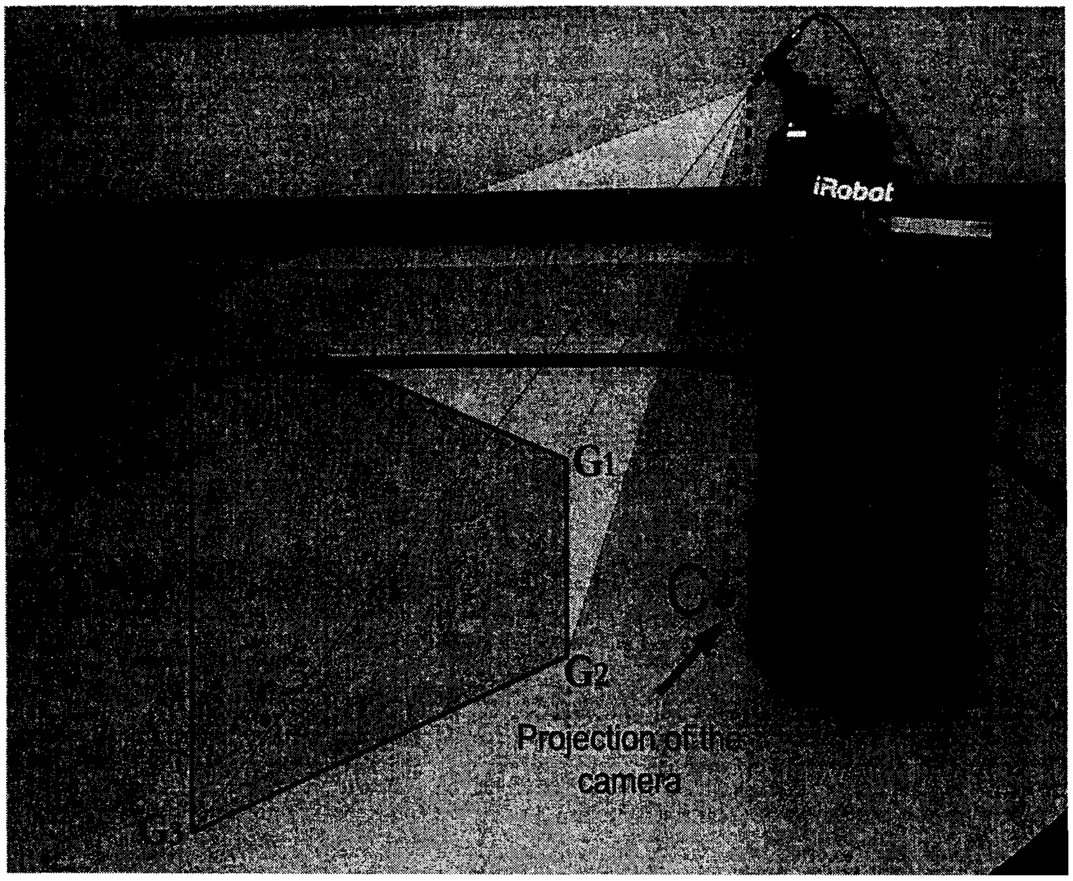


Figure 4.3: The Actual Area Taken by the Camera

4.2 Localizing An Arbitrary Point

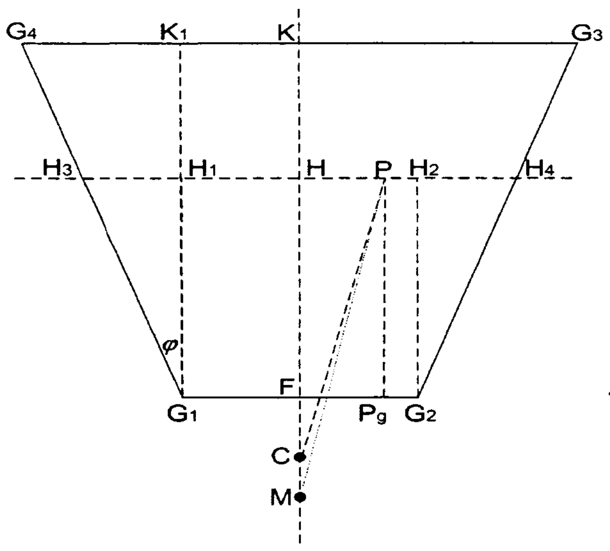


Figure 4.4: The Actual Area Taken by the CCD Camera

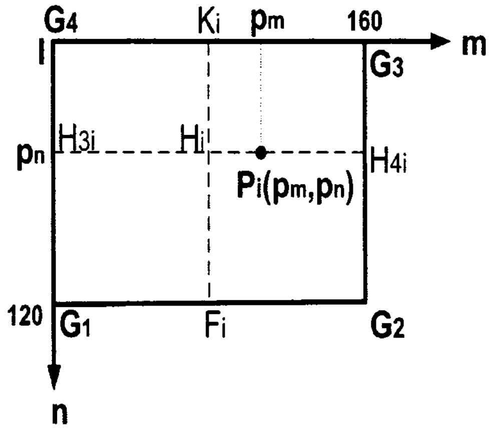


Figure 4.5: The CCD Image

Figures 4.3 and 4.4 gives the geometrical details of the actual area which is taken by the CCD camera, such as lengths of the parallel sides, the vertical height and the relation among them. In this figure, point P is the target which needs to be tracked by the robot, point C represents the CCD frame's projection on the floor, point M is the center of the mobile robot and the robot's orientation is at the same direction as $\overrightarrow{MK}$. Through measuring the actual area, we acquired the following parameters: G_1G_2 is 1.24 meter, G_3G_4 is 2.80 meter, MC is 0.25 meter, FK is 2.21 meter, and CF is 0.5 meter. Our task is to work out the length of MP which represents the distance from the center of the robot to the tracking point (P), and the angle β by which the robot needs to turn in order to face to the tracking target.

Figure 4.5 is the illustration of the CCD image. Due to the homography relation between the CCD image and the real area, for any arbitrary point P in the actual area, there exists a corresponding pixel P_i in the CCD image frame; and its coordinates in the Pixel Coordinate Frame are (P_m, P_n).

Figure 4.6 gives a detailed illustration about how to figure out the length of MP.

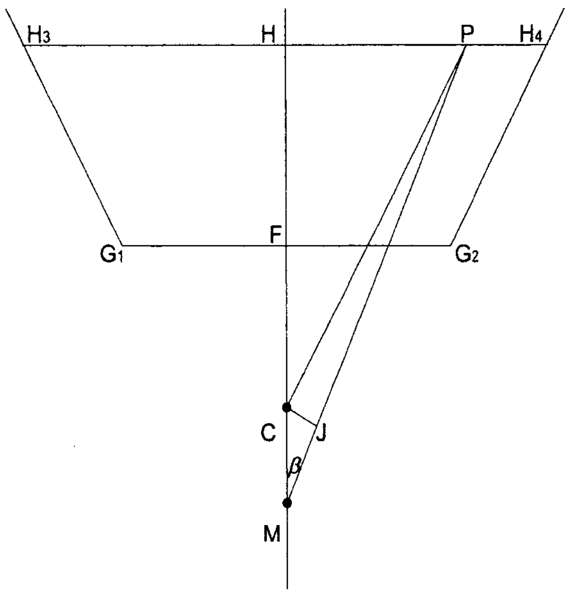


Figure 4.6: An Enlarged Part of the Actual Area

Since CJ is orthogonal to MP, we can get

$$MP = MJ + JP = \sqrt{CM^2 - CJ^2} + \sqrt{CP^2 - CJ^2}. \tag{4.2}$$

Since CM is known (0.25m), now we need to deal with the length of CP and CJ. From $CP = \sqrt{HC^2 + HP^2}$, the distance of CP will be obtained as long as the distance HC and HP is figured out. Since PP_g is orthogonal to G_1G_2 (see figure 4.4), and the distance of FK corresponds to 120 CCD pixels (line F_iK_i) in the CCD image frame, from the homography relationship, one can get the following equations:

$$\frac{PP_g}{FK} = \frac{120 - P_n}{120}, \tag{4.3}$$

$$PP_g = FK\frac{120 - P_n}{120} = \frac{2.21}{120}(120 - P_n) = 0.0184(120 - P_n), \tag{4.4}$$

$$HC = FC + FH = FC + PP_g = 0.5 + 0.0184(120 - P_n). \tag{4.5}$$

In the above equations, $(120 - P_n)$ is used instead of just P_n, because the origin of the Pixel Coordinate Frame is at the left top corner of the image frame (G_3).

It is more difficult to figure out HP than HC because G_2G_3 is not parallel to G_1G_4 (see figure 4.4). Since the constant width of the image frame (160 pixels) responds to different width of actual area (H_3H_4), the ratio between $H_{3i}H_{4i}$ and H_3H_4 cannot be determined directly. Therefore, the width of the actual area with respect to $H_{3i}H_{4i}$ needs to be calculated as follows:

since $KF = K_1G_1 = 2.21$,

and

$$G_4K_1 = \frac{G_3G_4 - G_1G_2}{2} = 0.78, \tag{4.6}$$

one can acquire

$$G_1G_4 = \sqrt{K_1G_1^2 + G_4K_1^2} = 2.3436, \tag{4.7}$$

$$\sin(\phi) = \frac{K_1G_1}{G_1G_4} = \frac{2.21}{2.3436} = 0.943, \tag{4.8}$$

and hence, $\phi = 70.56$ degrees.

Using ϕ one can get H_1H_3 and H_3H_4:

$$H_1H_3 = \frac{H_1G_1}{tan(\phi)} = \frac{0.0184}{2.83}(120 - P_n) = 0.0065(120 - P_n), \tag{4.9}$$

$$H_3H_4 = G_1G_2 + 2H_1H_3 = 1.24 + 2[0.0065(120 - P_n)] = 1.24 + 0.013(120 - P_n). \tag{4.10}$$

From the homography relation between the CCD image and the actual area, the lines HP and H_3H_4 in the actual area correspond to line H_iP_i and 160 CCD pixels (line $H_{3i}H_{4i}$) in the Pixel Coordinate Frame. Therefore, the following is determined:

$$\frac{| P_m - 80 |}{160} = \frac{HP}{H_3H_4}, \tag{4.11}$$

$$HP = H_3H_4\frac{| P_m - 80 |}{160} = (1.24 + 0.013(120 - P_n))\frac{| P_m - 80 |}{160}. \tag{4.12}$$

Since the distance of CH and HP is known, one can calculate the distance CP as follows:

$$\begin{aligned} CP &= \sqrt{CH^2 + HP^2} \\ &= \sqrt{[0.5 + 0.0184(120 - P_n)]^2 + [(1.24 + 0.013(120 - P_n))\frac{|P_m - 80|}{160}]^2} \end{aligned} \tag{4.13}$$

At the same time, we get the value of the angle β as follows:

$$\beta = arctg\frac{HP}{HC + CM} = arctg\frac{(1.24 + 0.013(120 - P_n))\frac{(80 - P_m)}{160}}{[0.5 + 0.0184(120 - P_n)] + 0.25} \tag{4.14}$$

From figure 4.6, one can see that $CJ = CM\sin(\beta)$, hence length of CJ can be acquired by

$$CJ = 0.25\sin\left[arctg\frac{(1.24 + 0.013(120 - P_n))\frac{(80 - P_m)}{160}}{[0.5 + 0.0184(120 - P_n)] + 0.25}\right] \tag{4.15}$$

Substituting equations 4.13 and 4.15 to equation 4.2, length of MP will be acquired, which is too long to show.

4.3 Localizing a Point on the Line of n=60

Since the size of the image file is very large (160 $\times$ 120 pixels), it will take the computer a long time to process each pixel of the image. If the tracking point can be determined just by checking one line of the image, the processing speed will increase more than one hundred times. In this project, since the environment is not complex and the reference trajectory is continuous and smooth, it is possible to recognize the tracking point through scanning one image line.

First, we try to determine the tracking point by checking the middle row (n=60) of the CCD image. Substituting $P_n = 60$ in equations 4.13, 4.14, and 4.15, the tracking point information will be acquired as below:

$$\begin{aligned}
CP &= \sqrt{CH^2+HP^2} \\
&= \sqrt{[0.5+0.0184(120-60)]^2+[(1.24+0.013(120-60))\tfrac{|P_m-80|}{160}]^2} \\
&= \sqrt{2.5728+[2.02\tfrac{|P_m-80|}{160}]^2} \\
\beta &= arctg\frac{HP}{HM} = arctg\frac{(1.24+0.013(120-60))\frac{(80-P_m)}{160}}{[0.5+0.0184(120-60)+0.25]} \\
&= arctg\frac{2.02\frac{(80-P_m)}{160}}{1.854} \\
CJ &= 0.25\sin[arctg\frac{2.02\frac{(80-P_m)}{160}}{1.854}]
\end{aligned} \tag{4.16}$$

Substituting CP and CJ from equation 4.16 to equation 4.2, we will get MP with respect to $P_n = 60$.

4.4 Localizing a Point on the Line of n=120

Since the middle part of the image is easy to be disturbed by uncertainties (see figure 3.7 to figure 3.11), we try to use the bottom row to detect the tracking point. Substituting $P_n = 120$ in equations 4.13, 4.14 and 4.2, the following is determined:

$$CP = \sqrt{CH^2+HP^2} = \sqrt{0.25+[(1.24)\frac{|P_m-80|}{160}]^2} \tag{4.17}$$

$$\beta = arctg\frac{HP}{HM} = arctg[0.01(80-P_m)] \tag{4.18}$$

$$CJ = 0.25\sin[arctg[0.01(80-P_m)]] \tag{4.19}$$

Substituting equations 4.17 and 4.19 to equation 4.2, we will get MP with respect to $P_n = 120$ as follow:

$$\begin{aligned}
MP &= \sqrt{(0.25)^2-[0.25\sin[arctg[0.01(80-P_m)]]]^2} \\
&+ \sqrt{[0.25+[(1.24)\tfrac{|P_m-80|}{160}]^2]-[0.25\sin[arctg[0.01(80-P_m)]]]^2}
\end{aligned} \tag{4.20}$$

The bottom row scanning approach makes the vision control system robust, and the simpler equation format increases the processing speed of the tracking point detection.

Chapter 5

Mobile Robot Kinematics

5.1 Wheel Kinematics Constraints and Assumptions

The real environment is unpredictable and full of uncertainties. At different conditions, the model of the robot should be changed correspondingly due to the interaction between the robot and the environment. Most of the research dealing with the mobile robots is based on the following assumptions: First, we consider the wheel of the robot contacting with the floor on a point. Second, the wheels of the robot will not deform, which means that the wheels are hard enough for their loads. Third, the robot needs to move on a horizontal plane. Moreover, there is just pure rolling between the wheels and the floor and no slipping, skidding or sliding exists. The mobile robots' steering axes should be orthogonal to the surface. Finally, the wheels of the mobile robot need to be connected by rigid frame (Shekhar 1997).

5.2 Mobile Robot Maneuverability

The maneuverability of a mobile robot is the combination of the mobility available based on the sliding constraints and additional freedom contributed by the steering (Siegwart and Nourbakhsh 2004).

$$\delta_M = \delta_m + \delta_s, \tag{5.1}$$

where δ_M is the robots maneuverability, δ_m is the robots degree of mobility and δ_s is the robots degree of steerability.
If n_{conf} is the dimension of the robot's configuration space, and n_{dof} is the number of degrees of freedom of the robot. Then the robot is a holonomic system when $n_{conf} = n_{dof}$, or a nonholonomic system when $n_{conf} > n_{dof}$.

5.3 Nonholonomic Systems

A nonholonomic constraint $h(q, \dot{q}) = 0$, where $q = (x, y, \theta)^T$, is a constraint on the mobile robot velocities, and is a kinematic constraint that cannot be integrated to yield a geometric constraint. For nonholonomic systems, the number of degrees of freedom is less than the dimension of the configuration space. Not all collision-free paths are feasible and the robots have to move in limited directions. However, the nonholonomic constraint is not global. For example, a car can not drive parallel, but it can get to the sideways position through the parallel parking operation.
In 1983, Brockett had proven that nonholonomic systems cannot be stabilized by using any smooth time-invariant feedback control (Brockett *et al.* 1983). Therefore, either smooth time-varying or discontinuous time-invariant controller has to be used for nonholonomic systems.

5.4 Holonomic Systems

A holonomic constraint $h(q) = 0$, where q is $(x, y)^T$, is an integrable kinematic constraint. A holonomic system is one in which the number of degrees of freedom are equal to the number of the dimension of the robot's configuration space. Any collision-free path is feasible to a holonomic system whose orientation can rotate to any given direction during a linear travel. The characteristic of the holonomic system makes the path planning easy and save the working space. (Watanabe *et al.* 1998).

5.5 Synchronous Drive Mobile Robot

The B21R mobile robot has a four-wheel synchronous drive system (Refer to figure 5.1). In this special synchronous drive mechanical structure, the four wheels are mechanically coupled by belts, drive torque is transferred down through each steering column to rubber tires. There are three DC servo motors for translation and one motor for rotation. The drive-motor output shaft is coupled to each of the steering column power shafts by a heavy-duty timing belt to ensure synchronous operation. A second timing belt transfers the rotational output of the steering motor to each steering column, allowing them to synchronously rotate in a 360 degrees range.
The four wheels of the system translate at the same linear velocity and rotate at the same angular velocity. Most importantly, the four wheels always keep the same orientation and are parallel with each other. When the mobile robot moves straight, every wheel runs at the same velocity and travels the same distance. The situation becomes complex when the mobile robot translates and rotates at the same time (Refer to figure 5.2). Each of the four wheels runs along a circle trajectory with the same diameter and certain offset among themselves. It also can be noticed from this figure that the Base of the B21R mobile robot will not change its orientation at any time, just the Enclosure of the robot turning with the wheels.

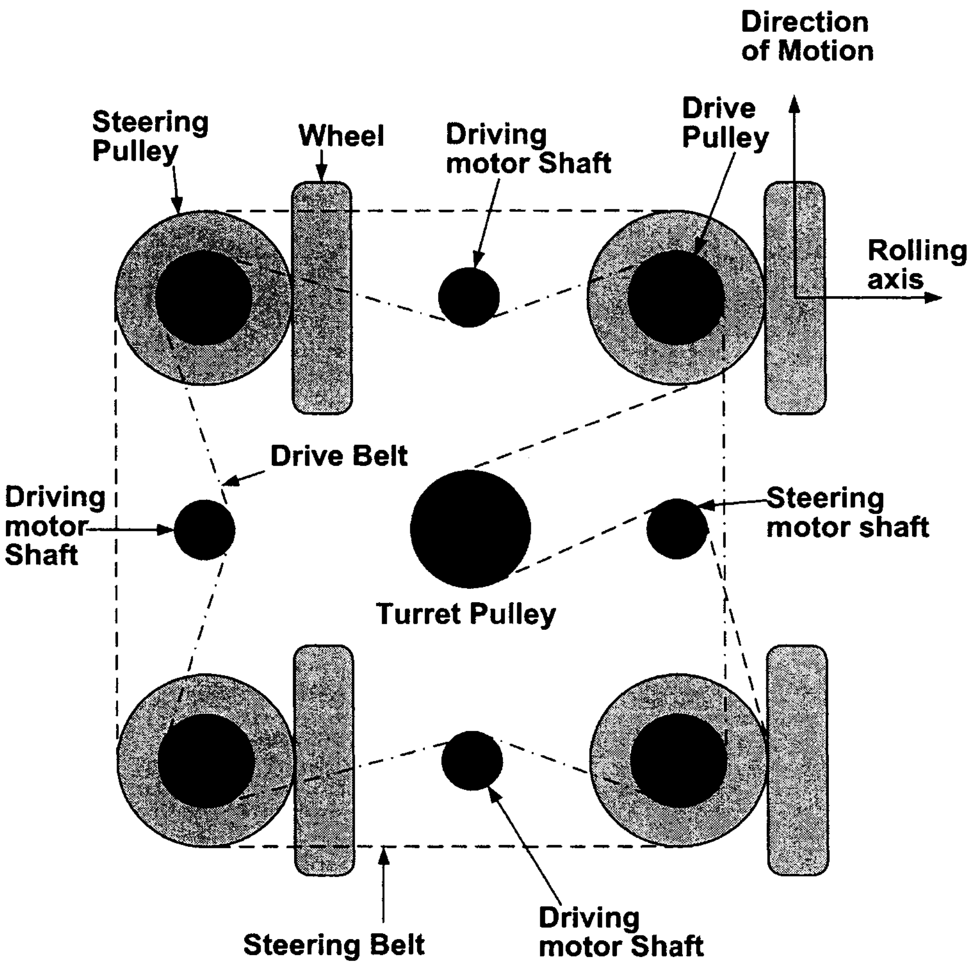


Figure 5.1: Synchronous Drive System

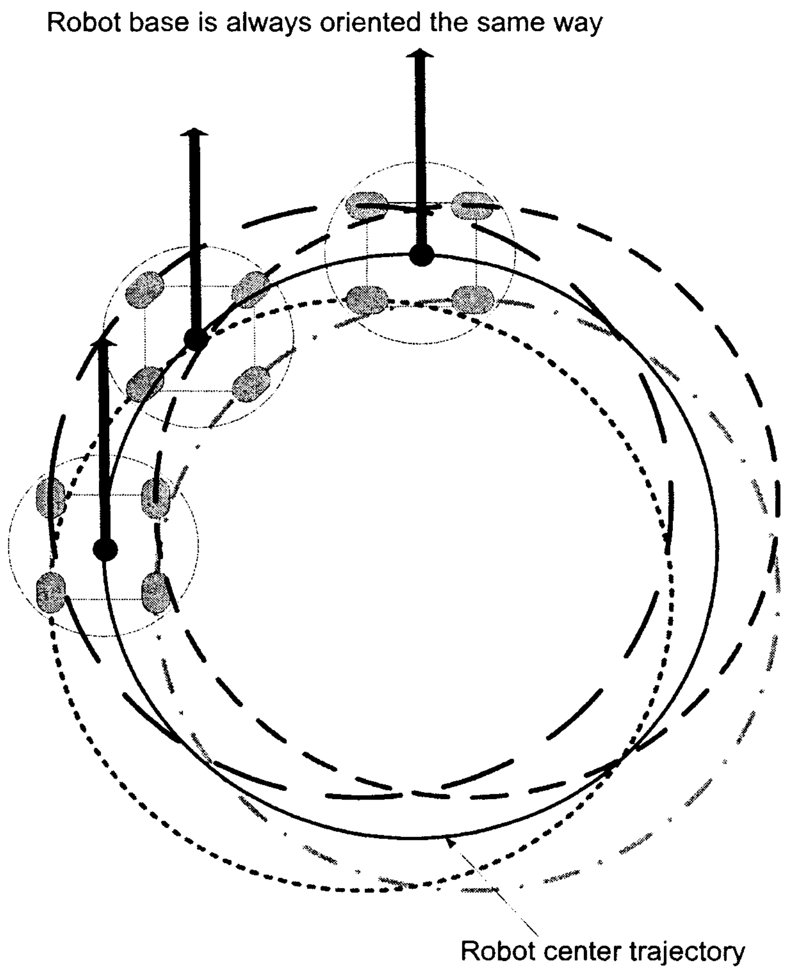


Figure 5.2: Four Wheels And Robot's Center Trajectory

5.6 Kinematic Model of the B21R Mobile Robot

Since the B21R mobile robot has a synchronous drive mechanical system, and has the same kinematics characteristics as the unicycle mobile robot, its kinematic model is given by:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{5.2}$$

where (x, y) represent the coordinates of the robot's position, θ is its orientation; v, and ω are the linear and angular velocities which are considered as the control inputs.

Chapter 6

Mobile Robot Control System

Active objects and the ability to communicate dynamic state updates between components allow the mobile robot to support the development of closed-loop robot control behaviors. These components combine input from one or more sensor systems and generate commands for one or more actuator systems within a robot.

6.1 Tracking Control Law

The kinematic model of the B21R mobile robot was obtained in section 5.6 (refer to equation 5.2). The robot can be described by the vector $(x, y, \theta)^T$, where (x, y) represent the coordinates of the point M located at the center of the robot with respect to the fixed frame $I_0 \equiv (O, \hat{\imath}_0, \hat{\jmath}_0)$ (Refer to figure 6.1).

The orientation of the mobile robot θ is measured with respect to the i_0 axis of the fixed frame. The linear velocity v and the angular velocity ω are the control inputs for the mobile robot. We assume that there is a reference mobile robot, with the same characteristics as the real one, moving on the reference trajectory (Tayebi and Rachid 1996). The reference trajectory described by the coordinates (x_r, y_r) with respect to the frame I_0, can be represented by a tangential motion of the identical "virtual"

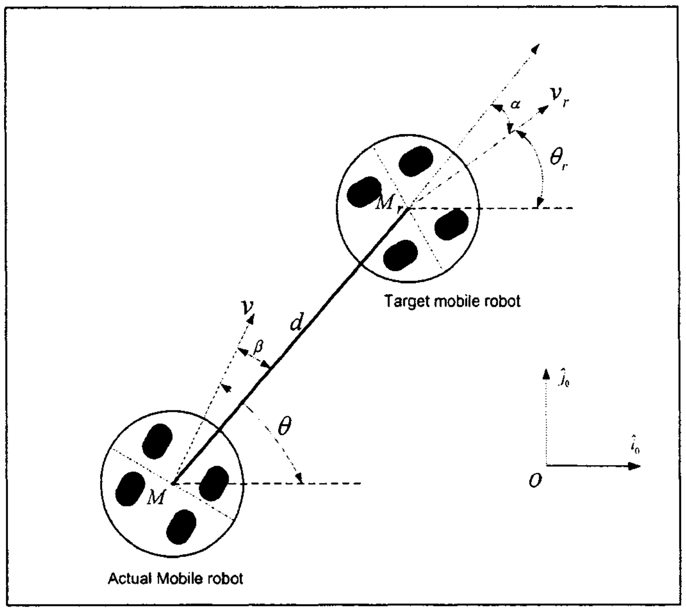


Figure 6.1: Mobile Robot Control

mobile robot, with linear velocity v_r and angular velocity ω_r, along this trajectory. The center of the virtual mobile robot M_r, which is actually the tracking target (P), moves along the reference trajectory which can be described by

$$\begin{aligned} \dot{x}_r &= v_r \cos\theta_r \\ \dot{y}_r &= v_r \sin\theta_r \\ \dot{\theta}_r &= \omega_r. \end{aligned} \tag{6.1}$$

The kinematics error model in the I_0 frame can be obtained as follows:

$$\begin{aligned} \dot{\tilde{x}} &= v_r \cos\theta_r - v\cos\theta \\ \dot{\tilde{y}} &= v_r \sin\theta_r - v\sin\theta \\ \dot{\tilde{\theta}} &= \omega_r - \omega. \end{aligned} \tag{6.2}$$

where $\tilde{x} = x_r - x$, $\tilde{y} = y_r - y$ and $\tilde{\theta} = \theta_r - \theta$. Alternatively, one can use a polar representation to describe the kinematics error model by introducing a new set of variables, namely, d, β and α. Let d denote the distance between M and M_r. β is the angle between the line MM_r and the orientation of the real mobile robot. α is the angle between the line of MM_r and the orientation of the reference mobile robot. It should be noticed that β is a negative value when the point M_r is located on the right side of the real robot's orientation (v). Since from equation 4.18, β will be negative when P_m is larger than 80. And a point with $P_m = 80$ is located on the robot's orientation.

From figure 6.2, we can figure out the angle α. Since, $|\alpha| + |\mu| = |\tau|$ and $|\tau| = \pi/2 - |\beta|$, we can acquire $|\alpha| = \pi/2 - |\beta| - |\mu|$. The value of β can be calculated from equation 4.18. Now, we focus on μ. Since M_r is following the reference trajectory, if we choose a point T_2 from the reference which is very near to M_r, $\overrightarrow{M_rT_2}$ can be realized with the same direction as V_r. In this project, since the point M_r is detected by the image line $P_n = 120$, we use image line $P_n = 117$ to scan another point T_2 on the reference trajectory. And the angle of μ can be calculated using the coordinates of T_1 and T_2 in the Pixel Coordinate Frame.

According to Figure 6.1, using the following relationships $d = \sqrt{\tilde{x}^2 + \tilde{y}^2}$, $\tilde{x} =$

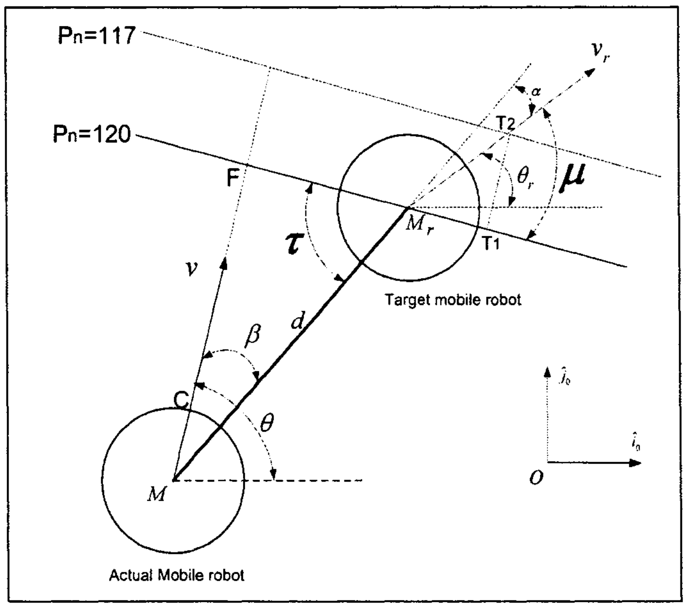


Figure 6.2: How to Calculate α

$d\sin(\theta + \beta)$, $\tilde{y} = d\cos(\theta + \beta)$, $\frac{\tilde{y}}{\tilde{x}} = \tan(\theta + \beta)$ and $\theta + \beta = \theta_r + \alpha$, together with equations 5.2, 6.1 and 6.2, one can obtain a new representation of the tracking kinematic error as follows:

$$\begin{aligned}\dot{d} &= v_r \cos\alpha - v\cos\beta \\ \dot{\beta} &= -\omega + \tfrac{v}{d}\sin\beta - \tfrac{v_r}{d}\sin\alpha \\ \dot{\alpha} &= -\omega_r + \tfrac{v}{d}\sin\beta - \tfrac{v_r}{d}\sin\alpha.\end{aligned} \tag{6.3}$$

Since the Pan-Tilt Unit has its own limit angles (the Tilt's lower limit angle is 46 degrees), the camera cannot take the image of the floor that is very close to the robot; the minimum distance which the camera can detect is 0.75 meter from the center of the robot. It means that the vision system cannot perceive the tracking target which is located at a distance less than 0.75 meter from the mobile robot's center. Due to this specific application, a new variable $\bar{d} = d - d_r$ is introduced, where d_r is a constant value (0.75 meter). The new tracking kinematic error becomes

$$\begin{aligned}\dot{\bar{d}} &= v_r \cos\alpha - v\cos\beta \\ \dot{\beta} &= -\omega + \tfrac{v}{\bar{d}+d_r}\sin\beta - \tfrac{v_r}{\bar{d}+d_r}\sin\alpha \\ \dot{\alpha} &= -\omega_r + \tfrac{v}{\bar{d}+d_r}\sin\beta - \tfrac{v_r}{\bar{d}+d_r}\sin\alpha.\end{aligned} \tag{6.4}$$

In this project, we assume that the mobile robot is initially close to the reference trajectory (*i.e.*, $v \simeq v_r$, $\omega \simeq \omega_r$, $\alpha \simeq \beta \simeq 0$), and the vision system can detect the desired trajectory at any time. In this particular situation, we obtain a local model of the form

$$\begin{aligned}\dot{\bar{d}} &= v_r - v \\ \dot{\beta} &= \dot{\alpha} = -\omega\end{aligned} \tag{6.5}$$

In this case, it is easily seen that the following linear controller

$$\begin{aligned}v &= v_r + k_1\bar{d} \\ \omega &= k_2\beta,\end{aligned} \tag{6.6}$$

guarantees local asymptotic stability of the equilibrium point ($\bar{d} = 0$,$\beta = 0$,$\alpha = 0$).

6.2 The Tracking Control Process

Figure 6.3 and 6.4 illustrate the whole process of the robot's tracking and obstacle avoidance. At the beginning, the robot sets up communication with the objects (sensors, actuators and servers). Then, the Pan-Tilt unit turns downward by 46 degrees and keeps its position during the control process. Thereafter, the sonar sensors check whether there are obstacles within 25 centimeters around the robot (the robot will not move until the environment around the robot up to 25 centimeters is clear). And the tactile sensors check whether the robot collides with something: the robot will stop immediately if any tactile sensor is activated. After that, a certain sonar will check whether there are obstacles in front of the robot up to 50 centimeters. If there are, the robot will perform the "obstacle avoidance" procedure. Otherwise the robot will acquire the sequential images from the vision system and try to extract the target point. If the robot cannot recognise the tracking point, the robot will keep rotating counterclockwise and searching. When the tracking point is acquired, it will be compared with the last sampling image. If the difference is less than 20 pixels, the robot will calculate the corresponding distance d and angle β. If the difference is larger than or equal to 20 pixels, the robot will stop and keep checking the difference until it is less than 20 pixels. Finally, using d and β, the controller will provide the linear and angular velocities v and ω to drive the robot towards the target point.

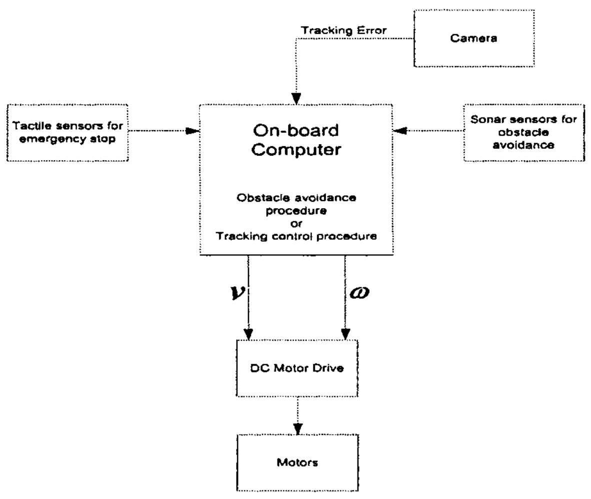

Tracking Error
Camera
Tactile sensors for emergency stop
On-board Computer
Obstacle avoidance procedure or Tracking control procedure
Sonar sensors for obstacle avoidance
v
ω
DC Motor Drive
Motors

Figure 6.3: Control Process

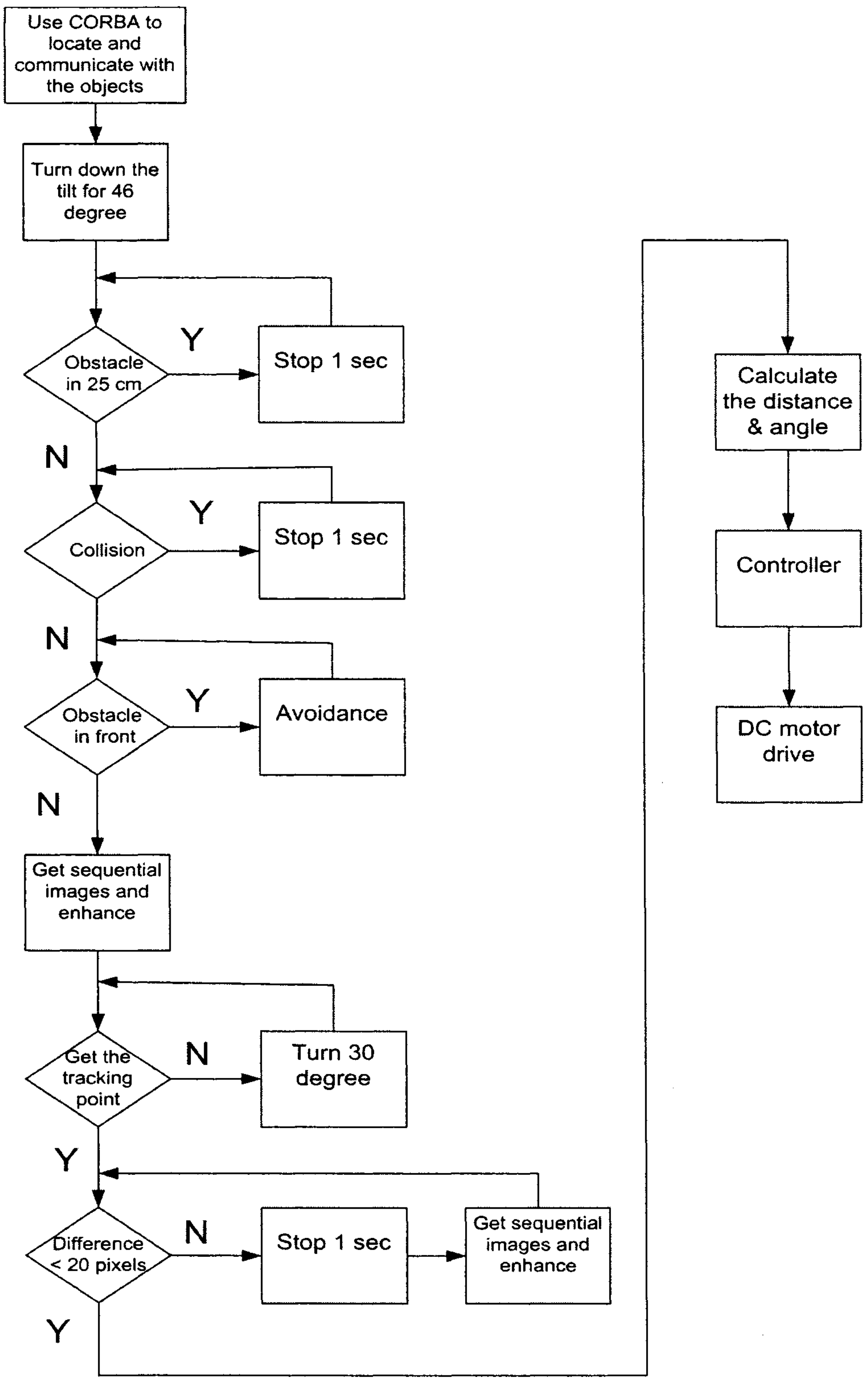

Use CORBA to locate and communicate with the objects
Turn down the tilt for 46 degree
Obstacle in 25 cm
Y
Stop 1 sec
N
Collision
Y
Stop 1 sec
N
Obstacle in front
Y
Avoidance
N
Get sequential images and enhance
Get the tracking point
N
Turn 30 degree
Y
Difference < 20 pixels
N
Stop 1 sec
Get sequential images and enhance
Y
Calculate the distance & angle
Controller
DC motor drive

Figure 6.4: Flowchart of the Control Program

6.3 Experimental Results

There are three parameters in the control law (refer to equation 6.6): v_r, k_1 and k_2. We tested these parameters using the values shown in Table 6.1. From the experimen-

Experiment	v_r	k_1	k_2
1	$v_r = 0.12$	$k_1 = 0.23$	$k_2 = 0.4$
2	$v_r = 0.12$	$k_1 = 0.05$	$k_2 = 0.4$
3	$v_r = 0.12$	$k_1 = 1$	$k_2 = 0.4$
4	$v_r = 0.12$	$k_1 = 0.23$	$k_2 = 0.26$
5	$v_r = 0.12$	$k_1 = 0.23$	$k_2 = 0.9$
6	$v_r = 0.17$	$k_1 = 0.23$	$k_2 = 0.4$
7	$v_r = 0.05$	$k_1 = 0.23$	$k_2 = 0.4$

Table 6.1: Parameters of the Control Law

tal results (see figure 6.5 to figure 6.36), one can get the following conclusions: First, the parameter k_1 affects the control performance slightly, due to the small value of $\bar{d}$ (refer to figures 6.8, 6.12, 6.16). When we chose k_1 as 0.05, 0.23 and 1 respectively, the linear velocity was increased, nevertheless we did not notice a big difference among the resulting tracking trajectories (see figure 6.5, figure 6.9, figure 6.13).

Second, the parameter k_2 can be chosen large enough, however it has a lower limit depending on v_r and k_1. If one chooses $v_r = 0.12$ and $k_1 = 0.23$, then k_2 has to be larger than 0.26 (see figure 6.17), otherwise the vision system will lose the reference trajectory when the B21R mobile robot faces the half-circle. By choosing k_2 very large, the vision system will not lose the reference trajectory, however this leads to a noticeable tracking error (refer to figure 6.21). Since the angle β is not very small during the half-circle part (refer to figures 6.7, 6.19, 6.23), the multiplication of β by a large gain k_2 will lead to a high angular velocity with respect to the linear velocity which results in a noticeable tracking error.

Third, the parameter v_r can be chosen small enough, however v_r has an upper limit depending upon the choice of k_1 and k_2. If one chooses $k_1 = 0.23$ and $k_2 = 0.4$, then v_r has to be smaller then 0.17 (Refer to figure 6.25), otherwise the vision system will lose the reference trajectory when the B21R mobile robot faces the half-circle. By choosing v_r very small (Refer to figure 6.29), the vision system will not lose the reference trajectory, however the robot cannot follow the trajectory exactly.

Fourth, the parameters should have a correct ratio among each other, especially between v_r and k_2, which are the crucial elements for the linear and angular velocity respectively. For example, $v_r = 0.12$, $k_1 = 0.23$ and $k_2 = 0.4$ is a good parameter setting according to our experimental results. If one of these parameters changes, the other two need to be changed correspondingly.

Fifth, from the figures of β and $\mid \alpha \mid$, one can see that, their values appear as a sine wave and sometimes are really large (such as -0.7 radian). This is caused by the fact that the vision system of the B21R mobile robot cannot detect the target point which is less than 0.75 meter from the robot's center. A large distance between the robot and its tracking target induces large values of β and $\mid \alpha \mid$, especially during the half-circle part. Opposite to β and $\mid \alpha \mid$, the value of $\bar{d}$ is pretty small.

Finally, figure 6.33 to figure 6.36 show our experimental results when the initial position of the robot is not located on the reference trajectory.

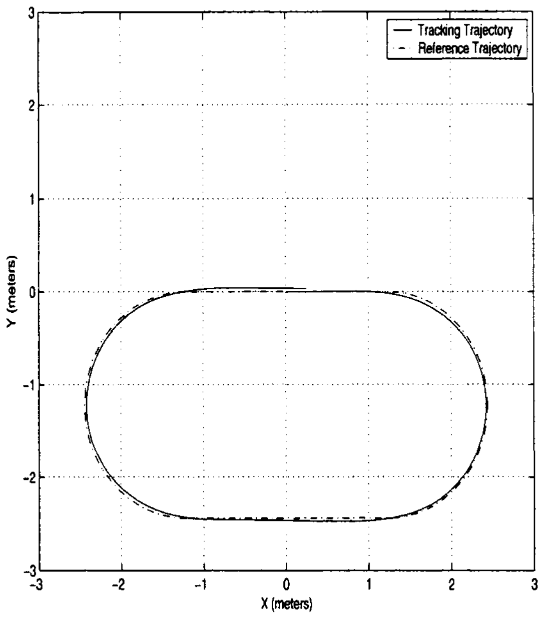


Figure 6.5: Actual and Reference Trajectories with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.4$

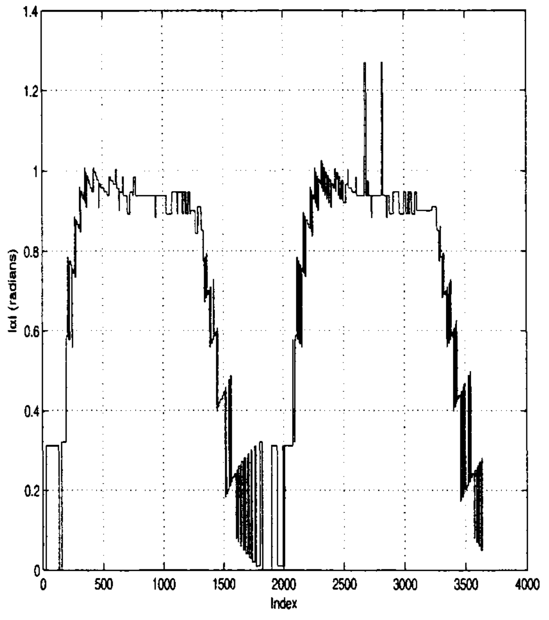


Figure 6.6: $\mid \alpha \mid$ Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.4$

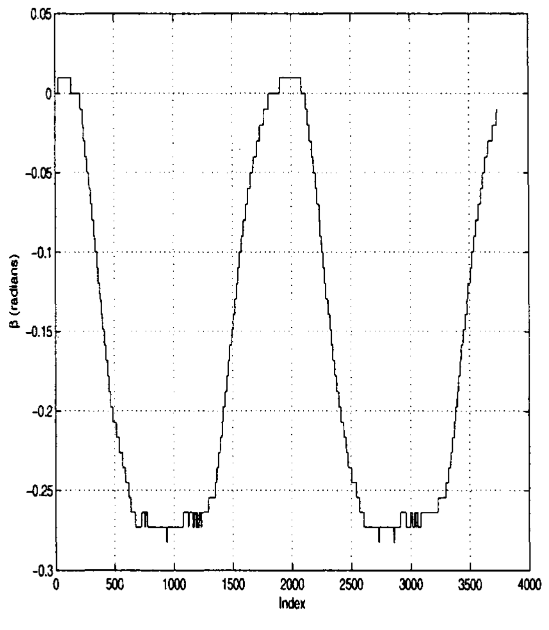


Figure 6.7: β Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.4$

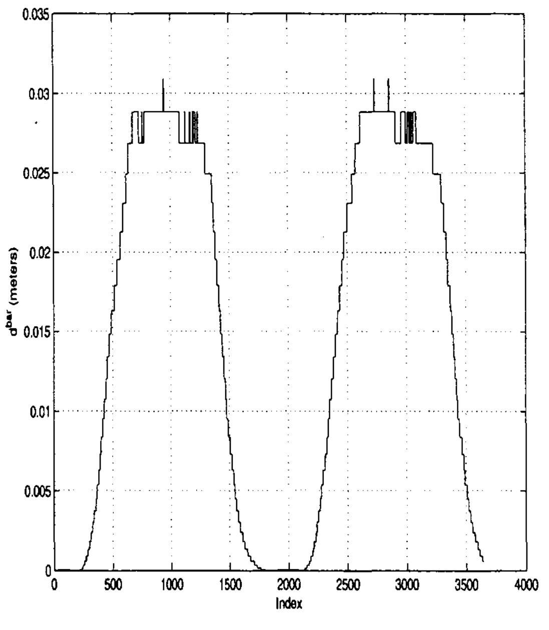


Figure 6.8: $\bar{d}$ Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.4$

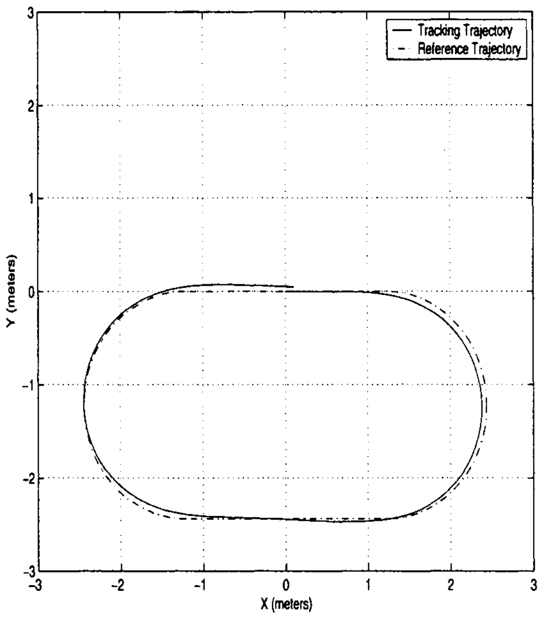


Figure 6.9: Actual and Reference Trajectories with $v_r = 0.12$ $k_1 = 0.05$ $k_2 = 0.4$

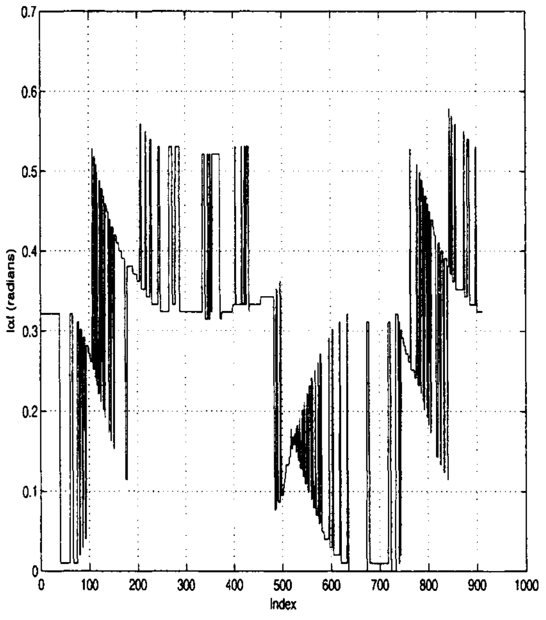


Figure 6.10: $| \alpha |$ Function of Time, with $v_r = 0.12$ $k_1 = 0.05$ $k_2 = 0.4$

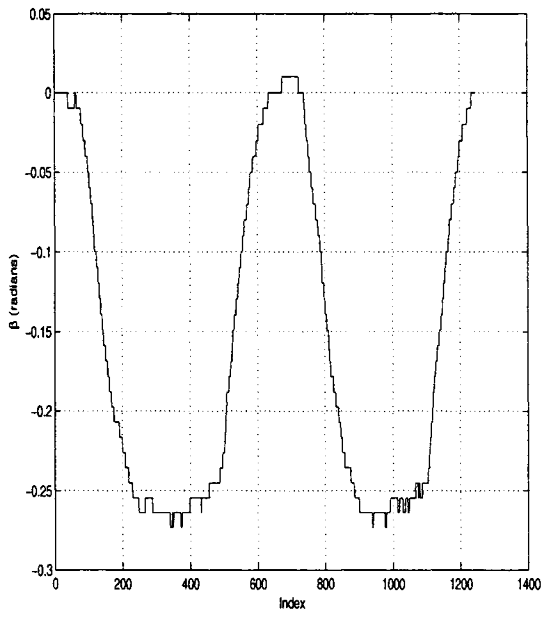


Figure 6.11: β Function of Time, with $v_r = 0.12$ $k_1 = 0.05$ $k_2 = 0.4$

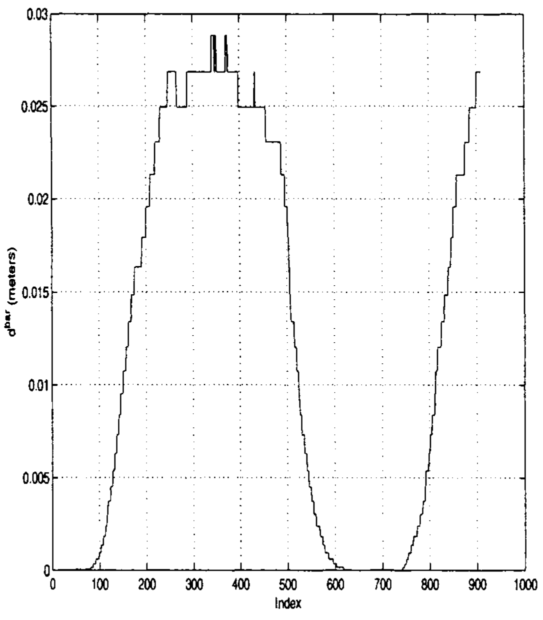


Figure 6.12: $\bar{d}$ Function of Time, with $v_r = 0.12$ $k_1 = 0.05$ $k_2 = 0.4$

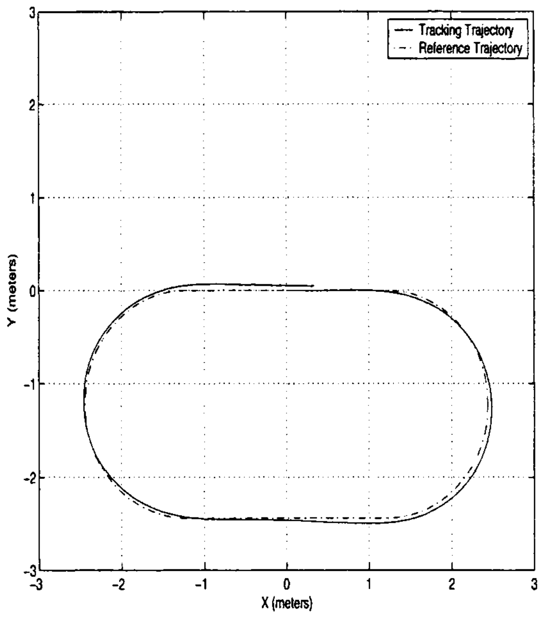


Figure 6.13: Actual and Reference Trajectories with $v_r = 0.12$ $k_1 = 1$ $k_2 = 0.4$

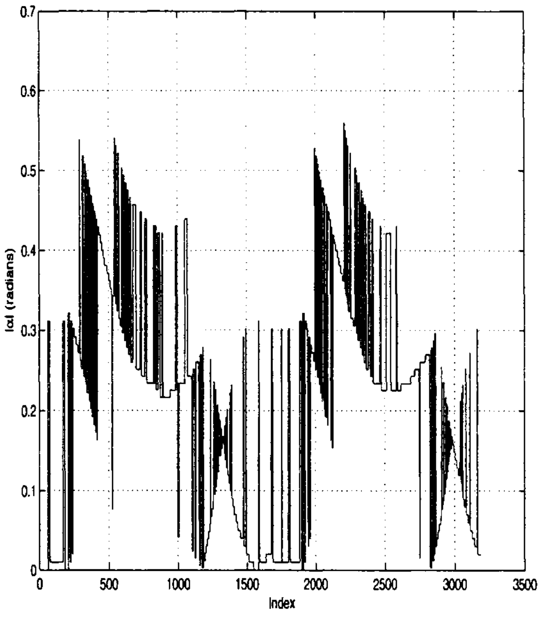


Figure 6.14: $\mid \alpha \mid$ Function of Time, with $v_r = 0.12$ $k_1 = 1$ $k_2 = 0.4$

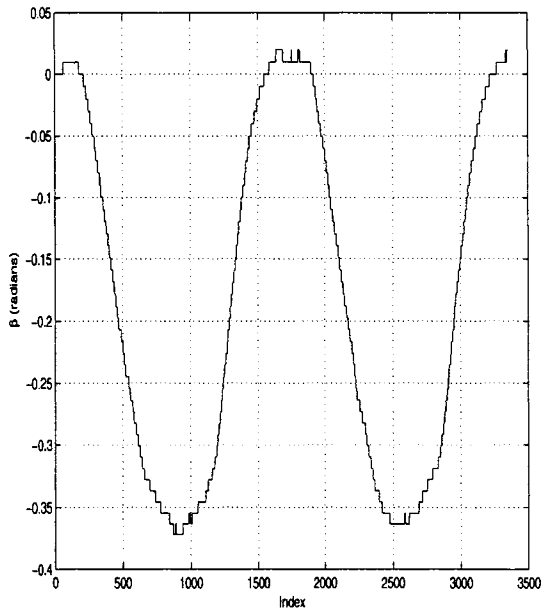


Figure 6.15: β Function of Time, with $v_r = 0.12$ $k_1 = 1$ $k_2 = 0.4$

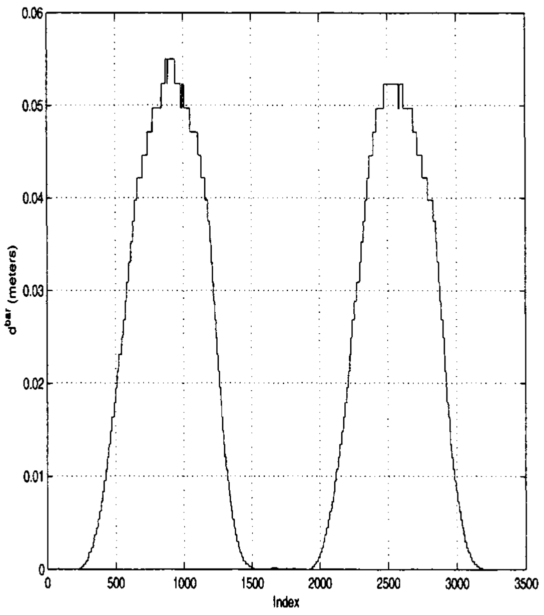
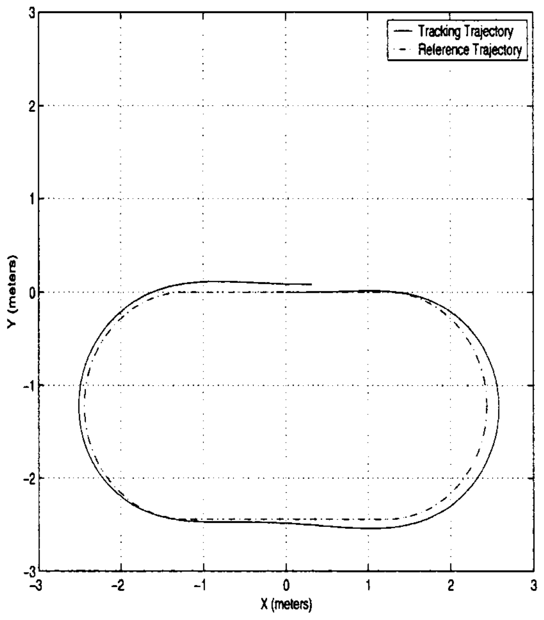


Figure 6.16: $\bar{d}$ Function of Time, with $v_r = 0.12$ $k_1 = 1$ $k_2 = 0.4$



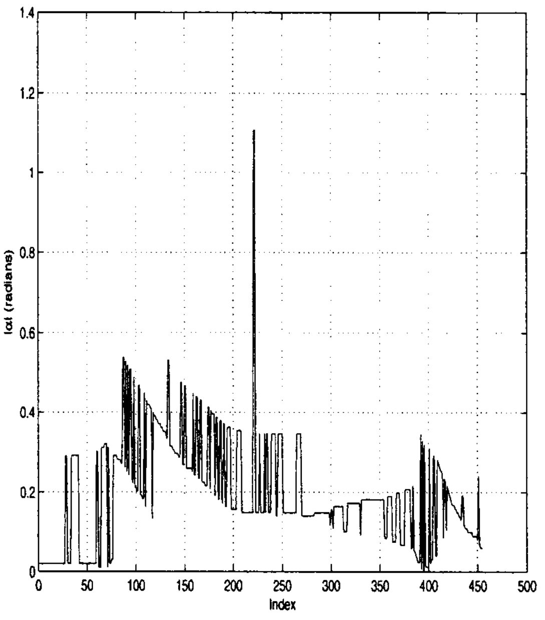
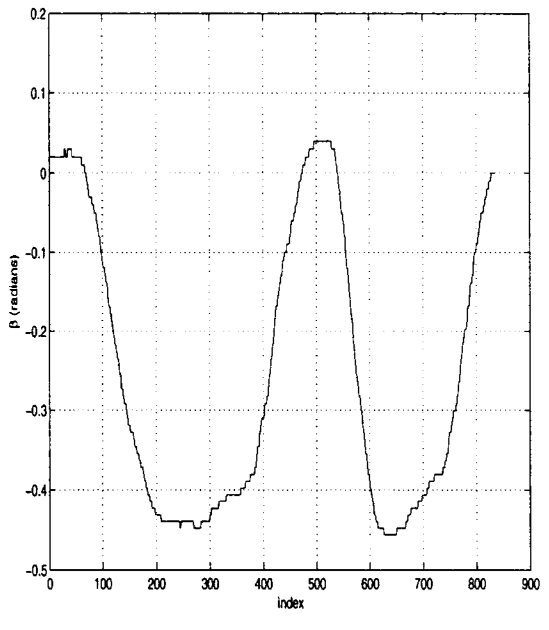


Figure 6.17: Actual and Reference Trajectories with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.26$

Figure 6.18: | α | Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.26$



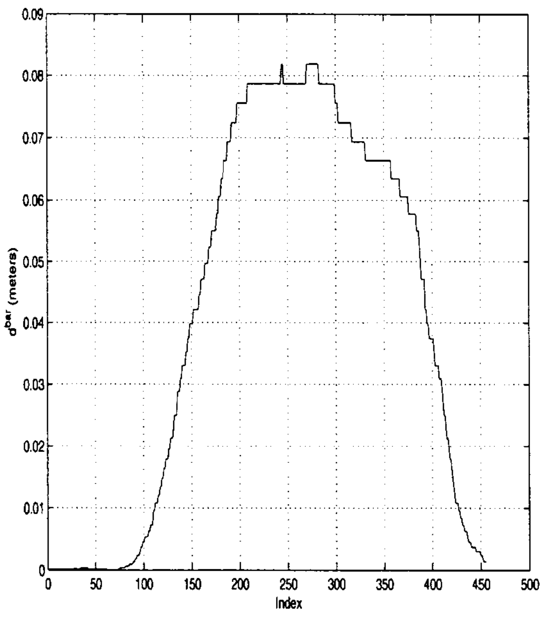


Figure 6.19: β Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.26$

Figure 6.20: $\bar{d}$ Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.26$

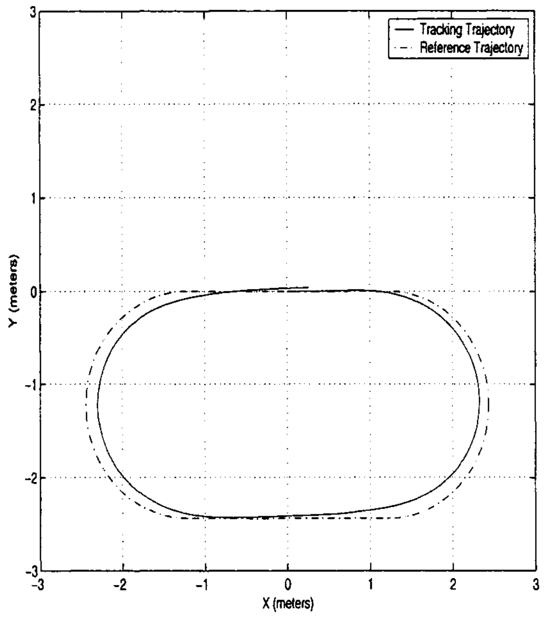


Figure 6.21: Actual and Reference Trajectories with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.9$

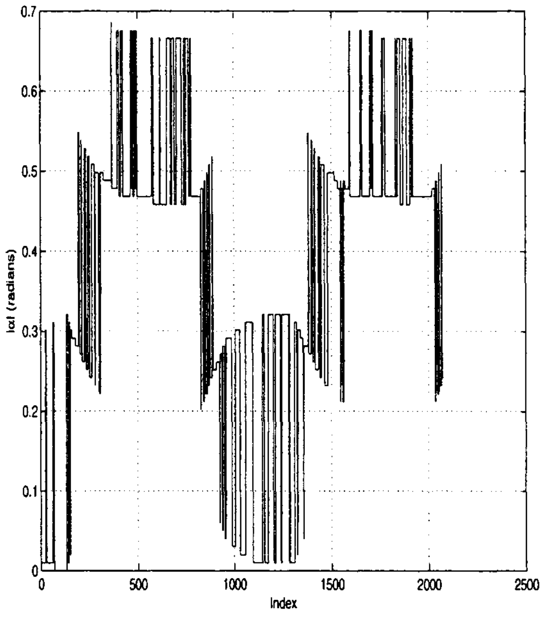


Figure 6.22: $\mid \alpha \mid$ Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.9$

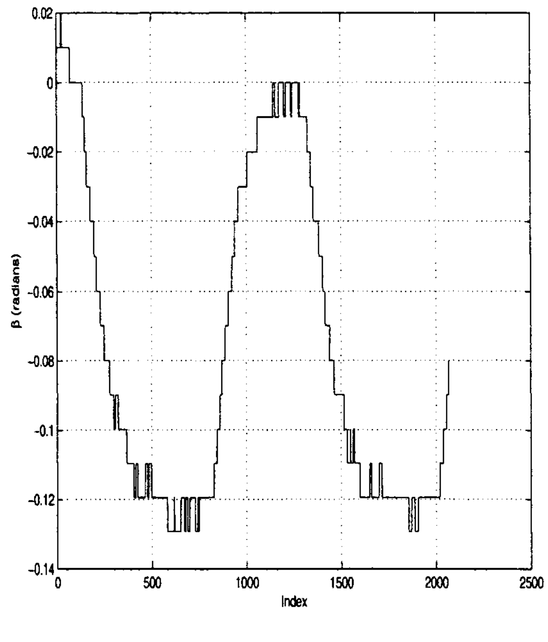


Figure 6.23: β Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.9$

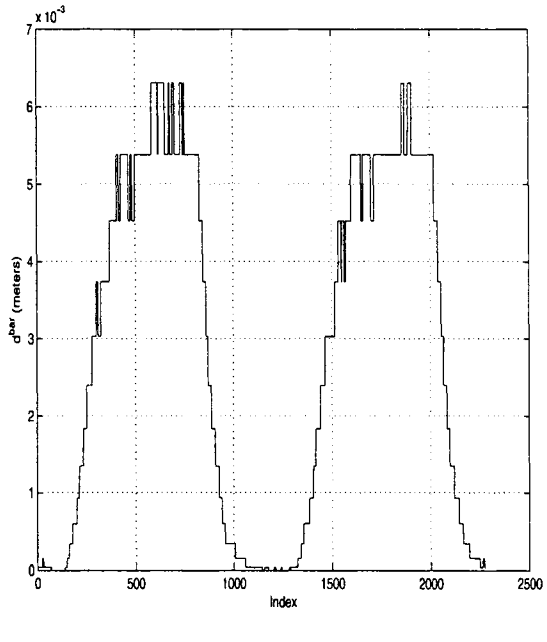


Figure 6.24: $\bar{d}$ Function of Time, with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.9$

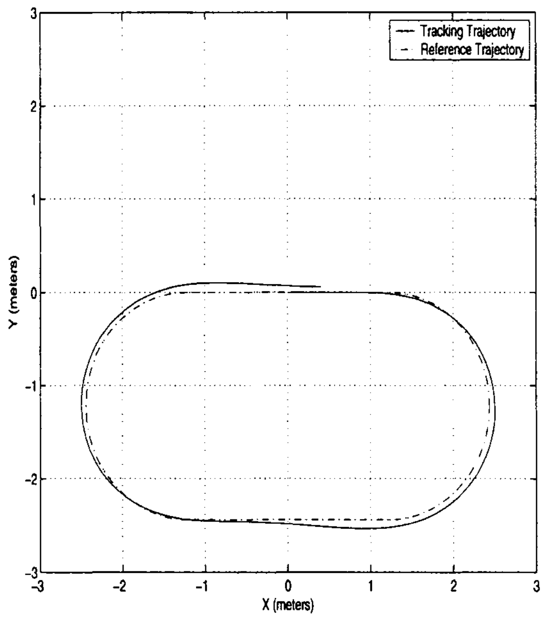


Figure 6.25: Actual and Reference Trajectories with $v_r = 0.17$ $k_1 = 0.23$ $k_2 = 0.4$

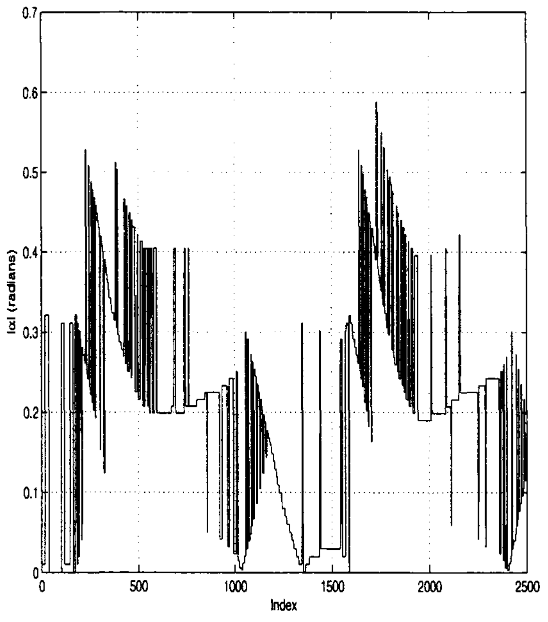


Figure 6.26: $|\ \alpha\ |$ Function of Time, with $v_r = 0.17$ $k_1 = 0.23$ $k_2 = 0.4$

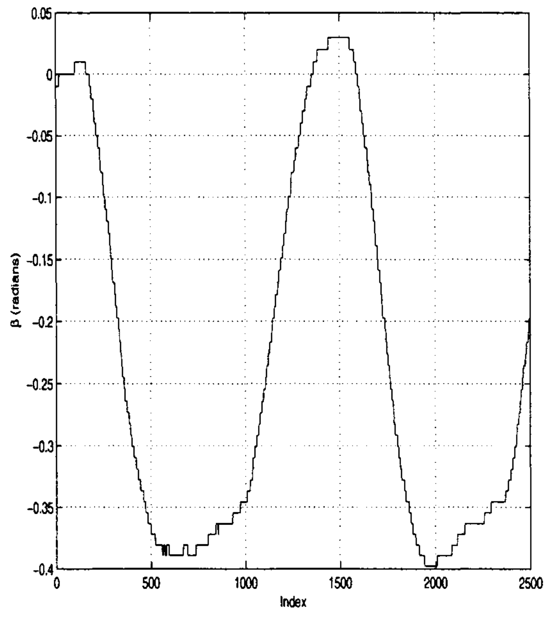


Figure 6.27: β Function of Time, with $v_r = 0.17$ $k_1 = 0.23$ $k_2 = 0.4$

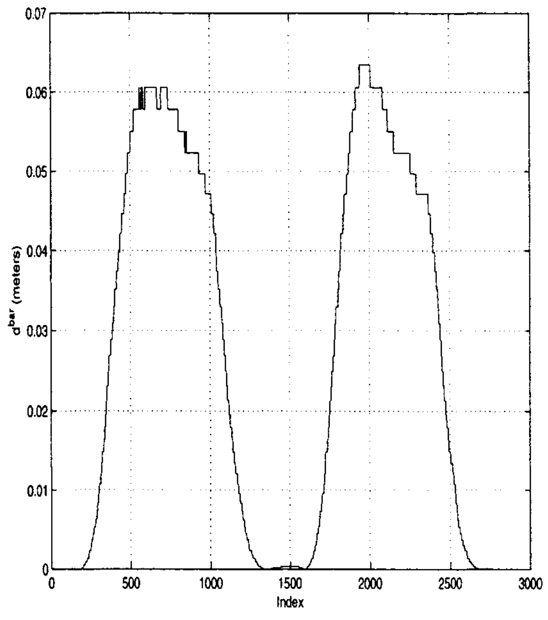


Figure 6.28: $\bar{d}$ Function of Time, with $v_r = 0.17$ $k_1 = 0.23$ $k_2 = 0.4$

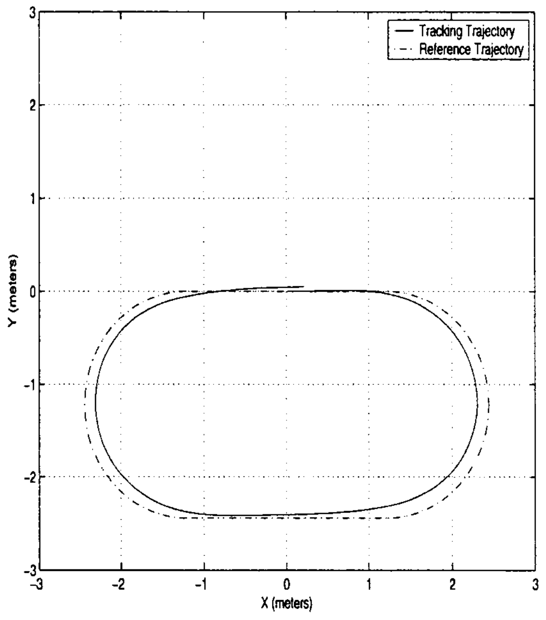


Figure 6.29: Actual and Reference Trajectories with $v_r = 0.05$ $k_1 = 0.23$ $k_2 = 0.4$

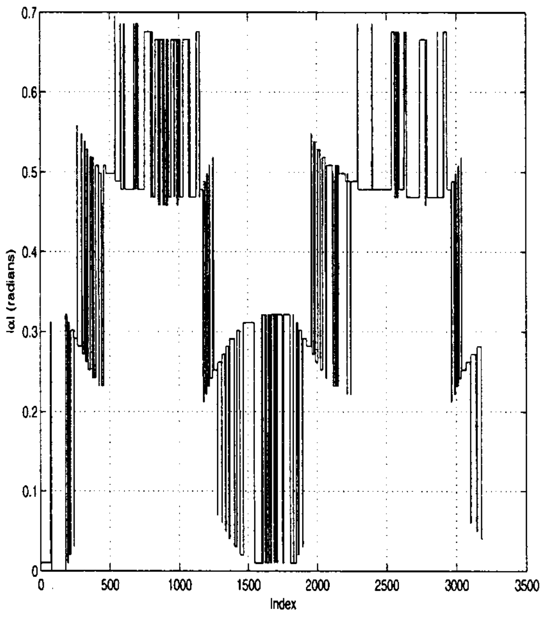


Figure 6.30: $| \alpha |$ Function of Time, with $v_r = 0.05$ $k_1 = 0.23$ $k_2 = 0.4$

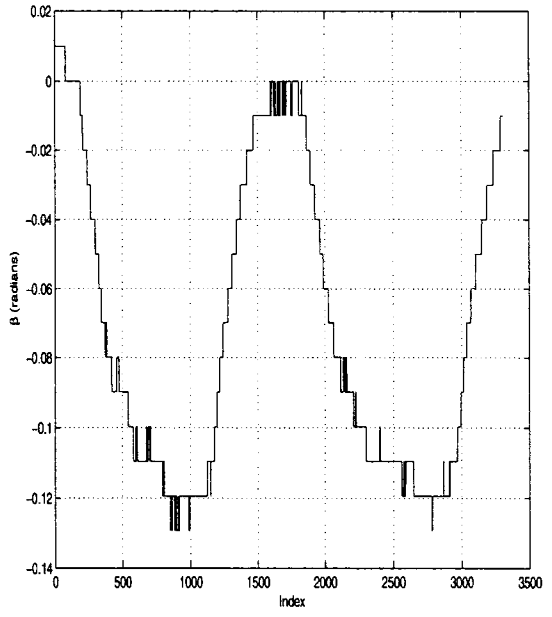


Figure 6.31: β Function of Time, with $v_r = 0.05$ $k_1 = 0.23$ $k_2 = 0.4$

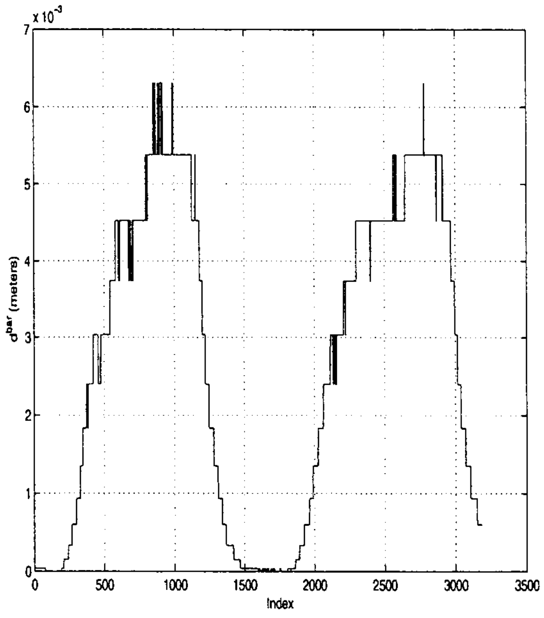
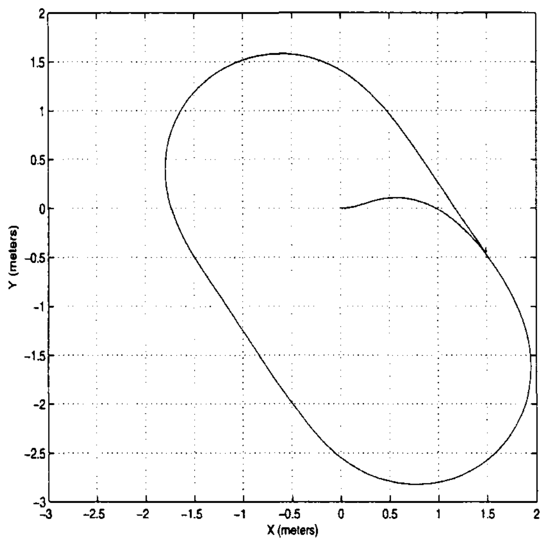


Figure 6.32: $\bar{d}$ Function of Time, with $v_r = 0.05$ $k_1 = 0.23$ $k_2 = 0.4$



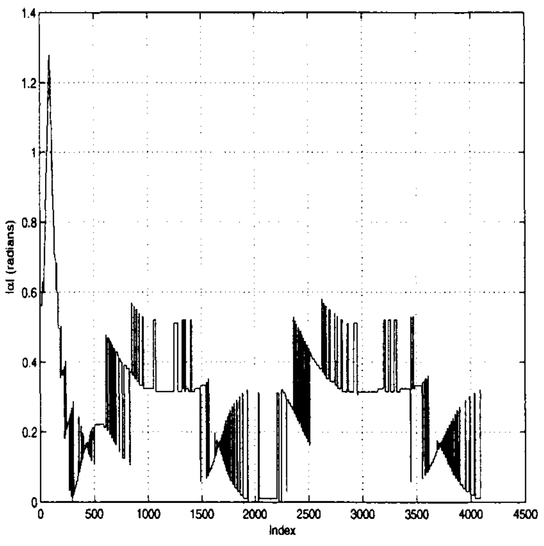
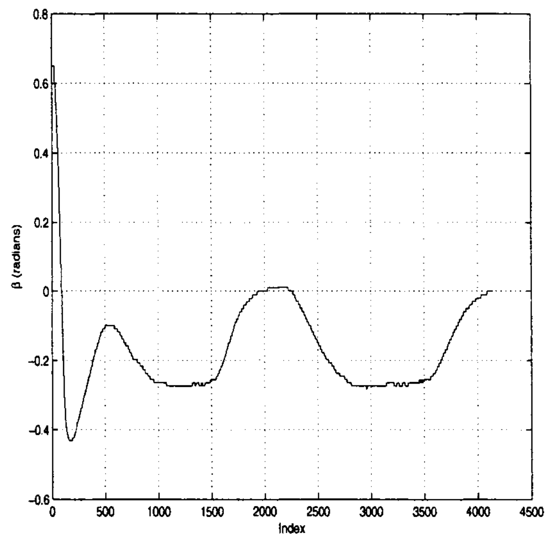


Figure 6.33: Tracking Trajectory Starting From An Arbitrary Initial Position with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.4$

Figure 6.34: $\mid \alpha \mid$ Function of Time, Robot Starting From An Arbitrary Initial Position with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.4$



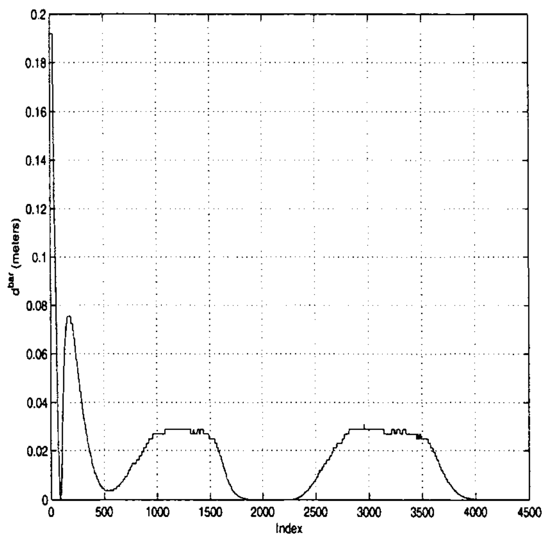


Figure 6.35: β Function of Time, Robot Starting From An Arbitrary Initial Position with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.4$

Figure 6.36: $\bar{d}$ Function of Time, Robot Starting From An Arbitrary Initial Position with $v_r = 0.12$ $k_1 = 0.23$ $k_2 = 0.4$

Chapter 7

Real-time Obstacle Avoidance

In this thesis, we consider just the case where the obstacle is located on the trajectory. Therefore, the camera cannot detect the trajectory behind the obstacle. In this special situation, a method which is similar to edge detection approach is used.

7.1 Limitations of Our Obstacle Avoidance Method

The obstacle avoidance method used in this project has some limitations, which need to be considered in our future work. First, this method can only recognize obstacles located on the trajectory using certain sonar sensors. Second, due to the limited space of our laboratory (there are protection bars on the outside of the trajectory), the robot was programmed to avoid the obstacle from inside of the trajectory. It means that the robot either turns right to avoid the obstacle when it is following the trajectory clockwise or turns left to avoid the obstacle when it is following the trajectory counterclockwise. Since no sensor is installed on our B21R mobile robot (such as compass) which can detect the direction of the robots' motion, we just let the robot follow the trajectory clockwise. Moreover, because we use a limited number of sonar sensors to guide the robot passing by the obstacle, this method is not suitable for a very large obstacle. Finally, due to the specular reflection which can cause the sonar sensor to ignore the obstacle, we assume that, the obstacle at least has a certain area which is orthogonal to the robot's orientation when the robot is facing it.

7.2 The Obstacle Avoidance Method Used in This Project

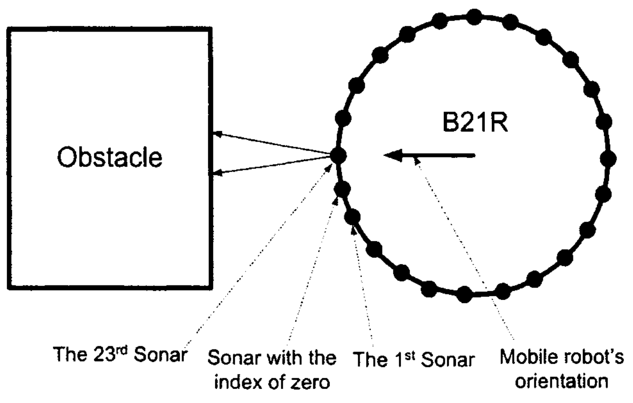


Figure 7.1: Obstacle Avoidance Position One

B21R mobile robot has 24 upper sonar sensors indexed from 0 to 23. Since the 23rd ultrasonic sensor has the same direction as the CCD camera (the orientation of the robot) and the positions of other upper sonar sensors are also constant with respect to the camera, the feedback from the upper sonar sensors is used to locate the obstacle's position relative to the robot.

Due to the specular reflection and the different shapes of the obstacles, the ultrasonic sensors cannot acquire the exact distance between the robot and an obstacle. Therefore, in this project, a comparison between the ultrasonic sensors' feedback is used to locate the obstacle's position with respect to the mobile robot.

During the tracking process, if the 23rd ultrasonic sensor detects an obstacle which is 50 centimeters far from the robot, the robot will stop for a second and detect again to see whether it is a static obstacle or just something passing by (Refer to figure 7.1). If the obstacle still exists after one second, the robot will begin its obstacle avoidance procedure.

Since the robot is restricted to follow the trajectory clockwise, the obstacle avoidance process can be separated to two steps: "right turn" and "left turn". During this procedure, the B21R mobile robot will rotate and translate at the same time. This combined movement will allow the robot to pass by the obstacle smoothly.

The flowchart of "Obstacle Avoidance Turning Right" (see figure 7.4) illustrates

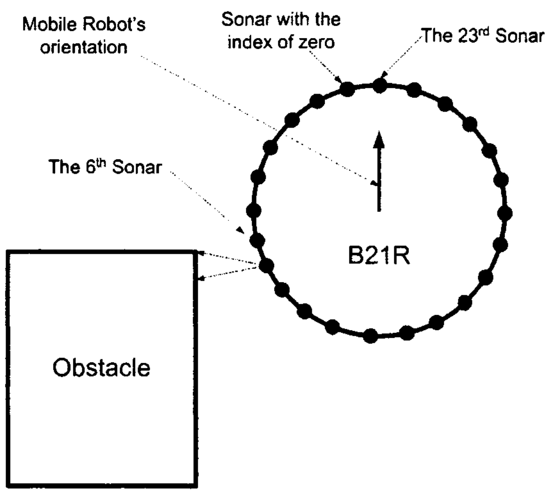


Figure 7.2: Obstacle Avoidance Position Two

the "turn right" process. At the beginning, the 23rd sonar sensor detects the shortest distance to the obstacle among all upper sonar sensors (refer to figure 7.1), and "TR" (the parameter of "turn right") is set to zero. During the robot turning right, the feedback of the 23rd sonar sensor becomes larger and larger, while the distance between the obstacle and the sonar sensor with index zero decreases gradually until it passes a certain position. Thereafter, the 1st, 2nd, 3rd, 4th sonar sensors and so on will acquire the shortest distance sequentially. This relation of the upper sonar sensors was used to lead the robot pass by the obstacle. When the feedback of the sonar sensor with index zero is larger than or equal to the feedback of the 1st sonar sensor, which means that the sonar sensor with index zero already passed the position where it has the shortest distance to the obstacle, the parameter "TR" will be increased to one. Similarly, when the feedback of the 1st sonar sensor is larger than or equal to the 2nd sonar sensor, “TR” will be increased to two. The robot will not stop to turn right until “TR” is equal to 7, which means that the feedback of the 6th sonar sensor is larger than or equal to the 7th sonar sensor. At this time, the robot has almost turned by 90 degrees to the right and half of the robot is beyond one side of the obstacle (refer to figure 7.2).

The procedure of “turn left” is similar to the “turn right”. First of all, we set “TL”

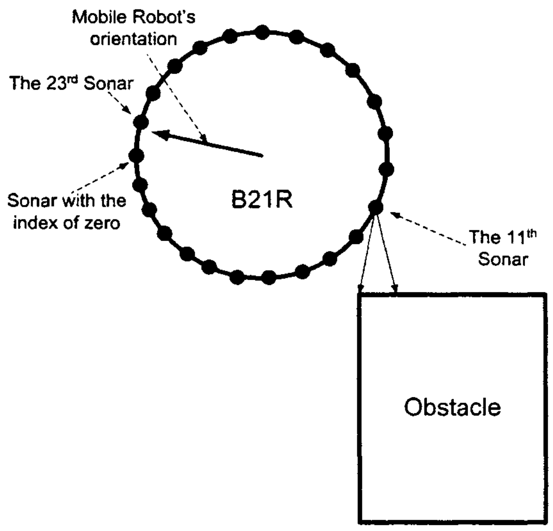


Figure 7.3: Obstacle Avoidance Position Three

(parameter of “turn left”) to zero. During the robot turning left, the feedback of the sonar sensors are compared in the following sequence: 4th, 5th, 6th, 7th, and so on (Refer to figure 7.5). For example, when the feedback of the 4th sonar sensor is larger than or equal to the feedback of the 5th sonar sensor, which means that the 4th sonar sensor has already passed the position where this sonar sensor has the shortest distance to the obstacle, the parameter “TL” will be increased to one. Similarly, when the feedback of the 5th sonar sensor is larger than or equal to the 6th sonar sensor, “TL” will be increased to two. The robot will not stop to turn left until “TL” is equal to 7which means that the feedback of the 10th sonar sensor is larger than or equal to the 11th sonar sensor's. At this time, the B21R mobile robot has already passed the obstacle (see figure 7.3), and it will start the trajectory searching procedure again.

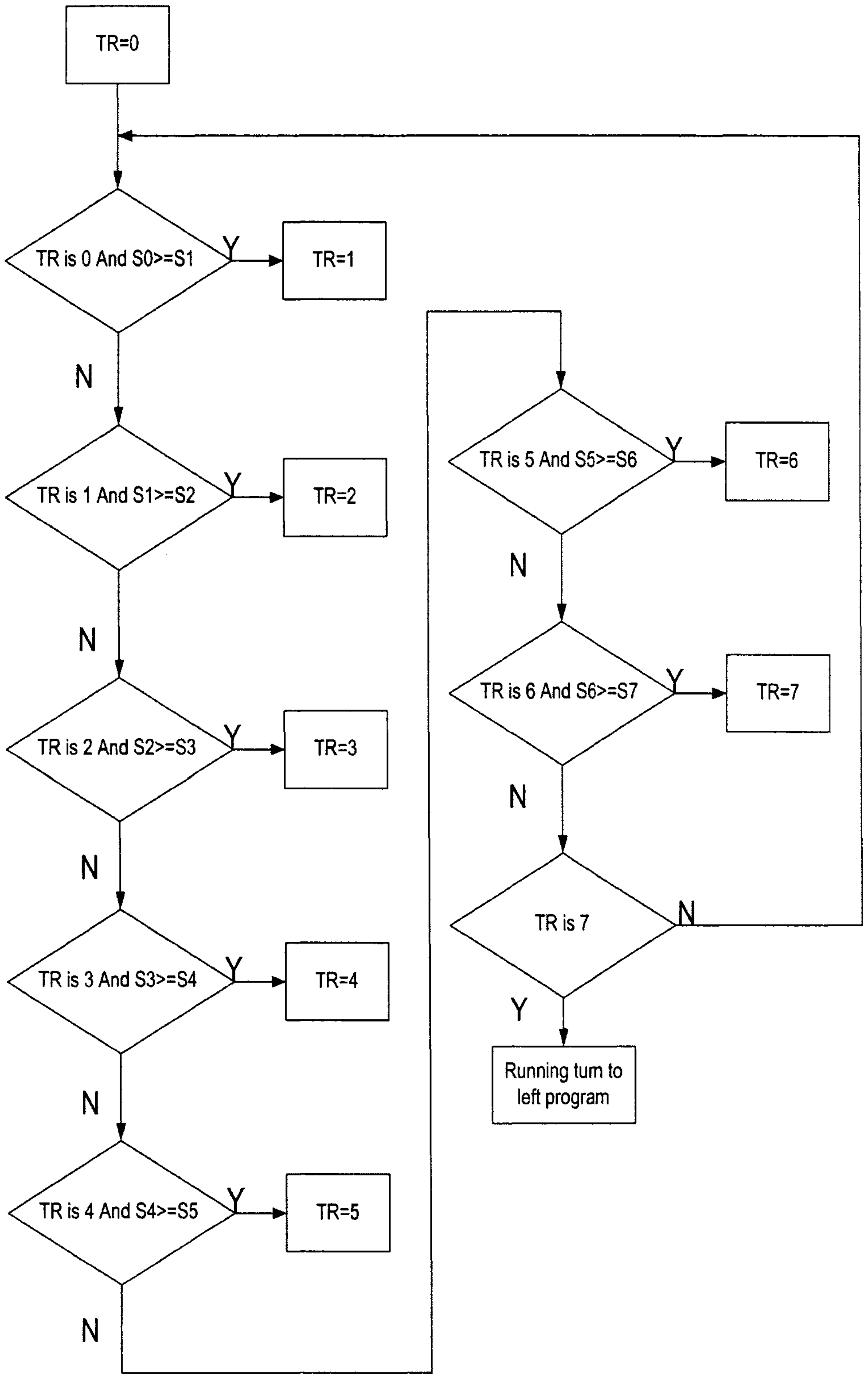


Figure 7.4: Flowchart of Obstacle Avoidance Turning Right

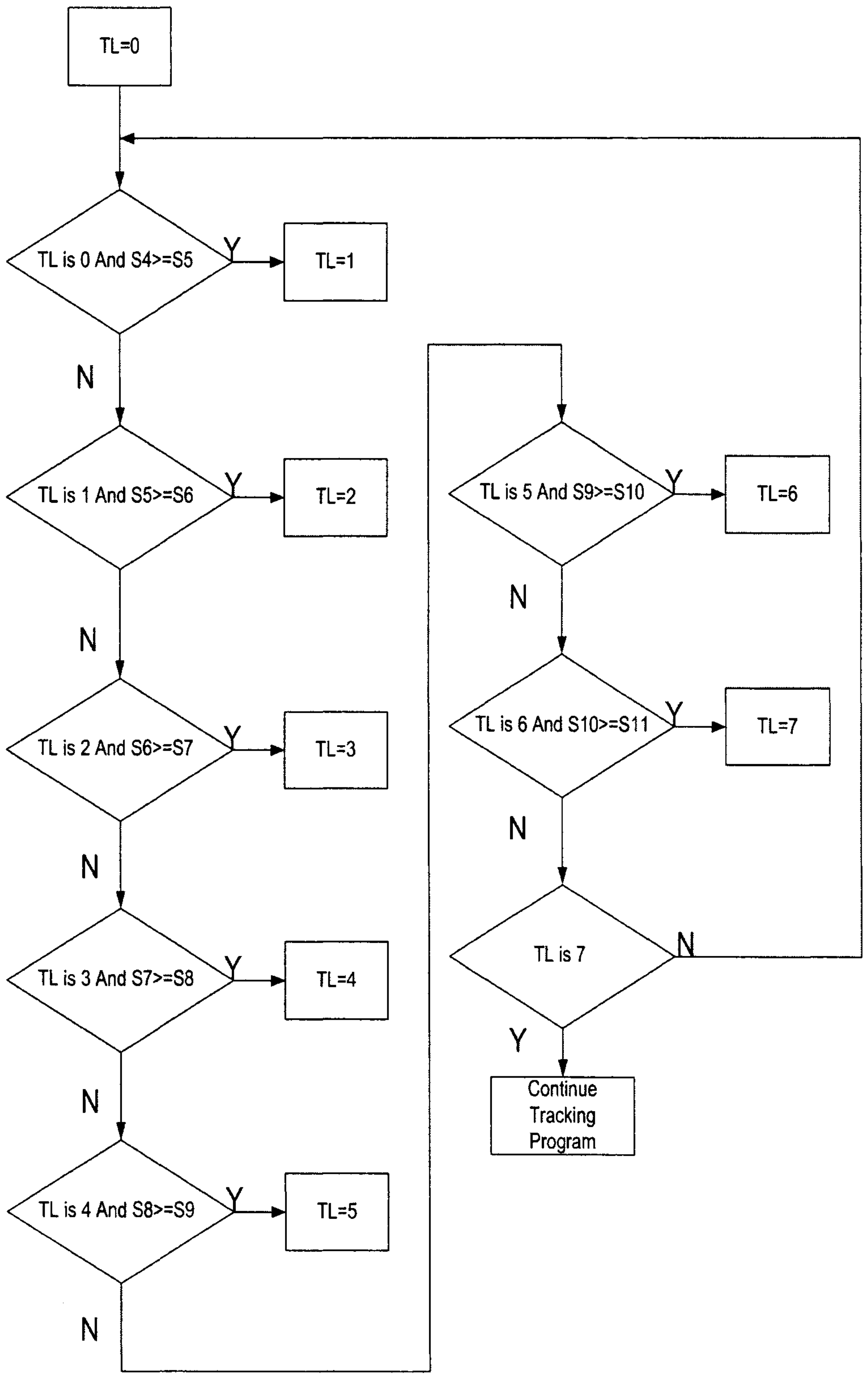


Figure 7.5: Flowchart of Obstacle Avoidance Turn Left

7.3 Experimental Results

Two different types of obstacles were used in our experiment. First, a box, whose dimension (length × width × height) is 18.5×18×39.5 inches (see figure 7.6), was located on the trajectory. The robot will detect and pass through it smoothly (see figure 7.7, and refer to the research video: Avoid a Box).

Figure 7.8 illustrates that the robot can also detect and smoothly pass by a person instead of a box (refer to the research video: Avoid a Person).

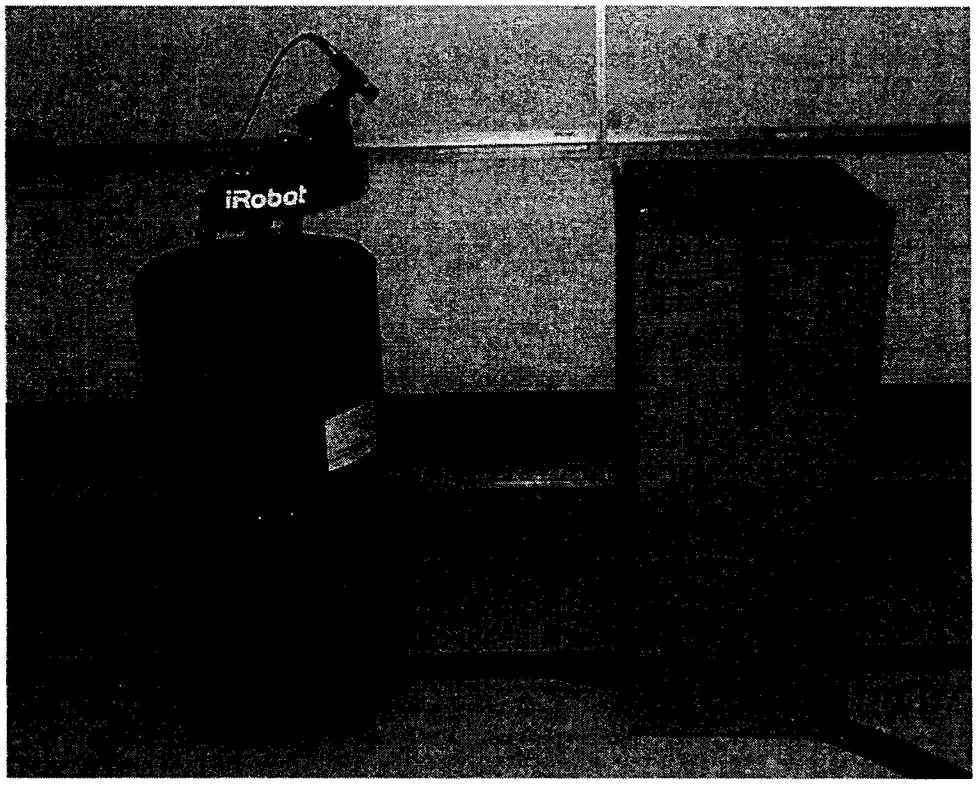


Figure 7.6: Box Obstacle

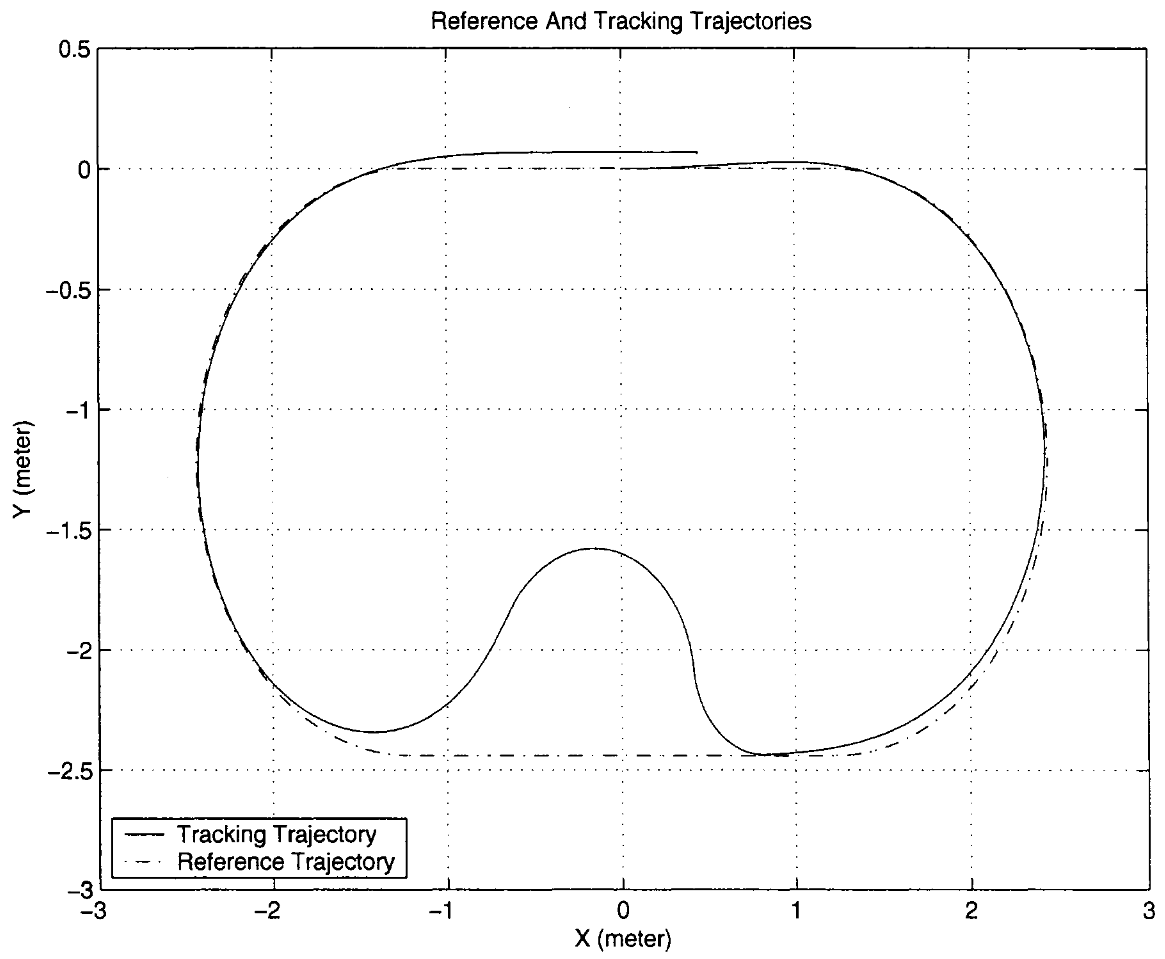


Figure 7.7: Avoiding a Box

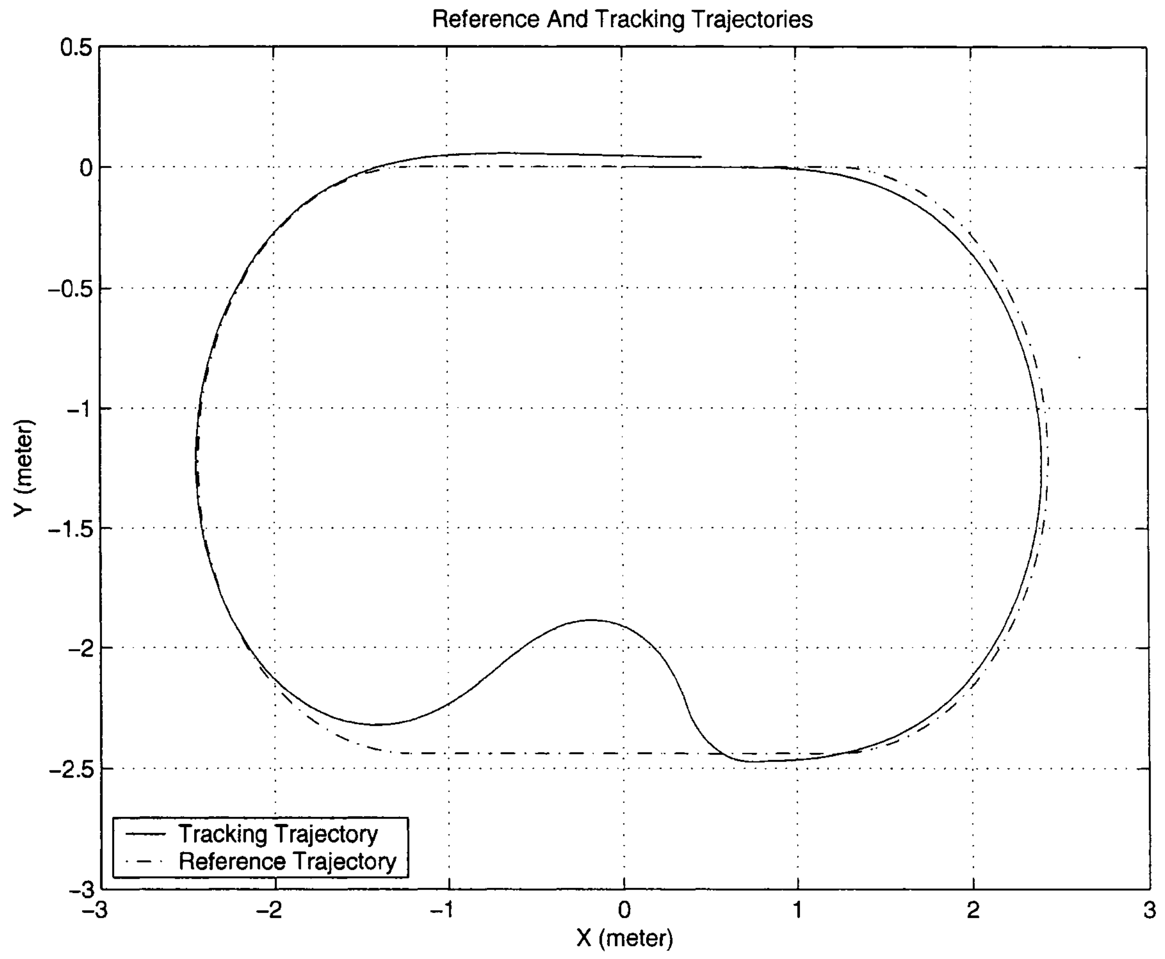


Figure 7.8: Avoiding a Person

Chapter 8

Conclusion

In this thesis, by using machine vision, an accurate, efficient, and reliable algorithm is generated for localizing a tracking object in unknown dynamic environments.
The algorithm does not try to recognize the whole or a part of the trajectory's contour, due to the limited range of the camera, and, the most important, the long processing time which cannot be tolerated by the real time control system in an unknown dynamic environment. The feature-based method was used in this algorithm to detect the tracking point. Instead of searching in the whole CCD image whose resolution is 120 times 160, the algorithm just scans one image line in one sampling image. This approach increases the image processing speed and is feasible in our case because the trajectory is continuous and smooth. And our experimental results verified that its performance is satisfied in real time condition.
Vision system is the essential component of the robot and provides information that is difficult or impossible to obtain in other ways. For instance, without the vision system, the B21R mobile robot cannot tell the trajectory's color which is the only difference between the floor and the reference trajectory.
Many approaches of obstacle avoidance were proposed in the last two decades, such as potential fields and vector field histogram. But some of them do not fit the requirement of our project. In fact, in this thesis, we assume that the environment is totally unknown, which means that the robot has no idea about the shape of the

reference trajectory and the obstacle's position. Hence, if a part of the trajectory is hidden by some obstacles and the robot can not detect the tracking point from the camera image, the robot will lose the target. Without the target, methods such as the potential field cannot be used directly.

In this project, a method which is similar to edge detection approach is used. When the robot detects an obstacle, it just simply moves around the edges of the obstacle until it finds the trajectory again. This method still has many limitations which need to be considered in the future.

A tracking control algorithm is also presented in this thesis. It has been noticed that the ratio of the control parameters v_r, k_1 and k_2 plays a crucial role in the tracking performance. The strength of this control strategy is that it is simple to implement, robust and does not require a great deal of computational resources. The algorithm has been verified experimentally on the B21R mobile robot.

8.1 Future Work

To make our method suitable under more general conditions, we need to focus on the following aspects.

First, improve the obstacle avoidance algorithm to make the robot able to decide appropriately in which direction the robot should turn instead of just assuming arbitrarily that it should turn clockwise when avoiding an obstacle.

Second, pay more attention to the robot's surrounding, not just focus on the trajectory. For instance, in this project, the robot just avoids the obstacles located on the trajectory, which means that the robot cannot pass by the obstacles which are located near but not on the trajectory. If the algorithm takes into account the sonar sensors other than the sonar with index 23, the robot will handle more complex environment.

Finally, sensors fusion needs to be investigated further to make the control more robust and efficient. In fact, the combination of the information provided by the upper and lower sonar sensors, infrared sensors and the camera will lead to much more efficient algorithm for the tracking and obstacle avoidance, but the price to pay is the computational complexity.

Bibliography

Arras K.O., N. Tomatis and R. Siegwart, 2000. Multisensor on-the-fly Localization Using Laser and Vision. Proc. *IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS)*, pp: 131-143, Takamatsu, Japan.

Augustin B., T. Lietmann and B. Lohmann, 2002. Image-Based Visual Servoing of a Non-Holonomic Mobile Platform Using a Pan-Tilt-Head. *IEEE international Conference on Methods and Models in Automation and Robotics*, Szczecin, Poland, pp. 941-946.

Borenstein J., H.R. Everett, L. Feng and D. Wehe, 1997. Mobile Robot Positioning–Sensors and Techniques. *Robotic Systems*, Special Issue on Mobile Robots. Vol.14, pp. 231-249.

Borenstein J. and L. Feng, 1996, Measurement and Correction of Systematic Odometry Errors in Mobile Robots. *IEEE Journal of Robotics and Automation*, Vol 12. pp. 869-880

Borenstein J. and Y. Koren, 1989. Real-time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 5, pp. 1179-1187

Borenstein J. and Y. Koren, 1991. The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, pp.278-288.

Brock O. and O. Khatib, 1999. High-speed Navigation Using the Global Dynamic

Window Approach. Proc. of the *IEEE Int. Conf. on Robotics and Automation*, Detroit, MI (US). pp. 341-346

Brockett R.W., R.S. Milman and H.J. Sussmann, 1983. Asymptotic Stability and Feedback Stabilization. *Differential Geometric Control Theory*, Birkhauser, Boston, pp:181-191

Cadenat V., R. Swain, P. Soueres and M. Devy, 1999. A Controller to Perform a Visually Guided Tracking Task in a Cluttered Environment. *Rapport LAAS No99012 1999. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'99)*, pp.775-780

Cassandra A. R., L. P Kaelbling and J. A. Kurien, 1996. Acting under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation. *IEEE International Conference on Robotics and Automation*, Osaka, Japan.

Chaumette F., 1998. Potential Problems of Stability and Convergence in Image-based and Position-based Visual Servoing, *Lecture Notes in Control and Information Sciences*, vol. 237 pp. 66-78, SpringeVerlag.

Comport A. I., E. Marchand and F. Chaumette, 2004. Robust Model-based Tracking for Robot Vision*IEEE/RSJ Int. Conf on Inteligent Robots and Systems, IROS04*, Volume 1, pp. 692-697, Sendai.

Corke P. and M. Good, 1993. Controller Design for High-performance Visual Servoing, Proc. *12th World Congress International Federation of Automatic Control*, (Sydney), pp.395-398.

Corke P. and S. A. Hutchinson, 2000. Real-time Vision, Tracking and Control. *Proc. IEEE Int. Control, Robot and Automation*, pp. 622-629, San Francisco.

Crowley J. L., 1989. World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging. Proc of the *IEEE International Conference on Robotics and Automation.* Scottsdale, Arizona, pp. 674-680.

DeSouza G. N. and A. C. Kak, 2002. Vision for Mobile Robot Navigation: A Survey.

IEEE Transactions On Pattern Analysis And Machine Intelligence, VOL. 24, NO. 2, pp. 237-267.

Durrant-Whyte H. and J. Leonard, 1991. Mobile robot localization by tracking geometric beacons. *IEEE Transaction on Robotics and Automation*, vol 7, pp. 376-382.

Elfes A., 1985. A Sonar-Based Mapping and Navigation System. Carnegie Mellon University, *The Robotics Institute, Technical Report*, pp. 25-30.

Espiau B., F. Chaumette and P. Rives, 1992. A New Approach to Visual Servoing in Robotics. *IEEE Transactions on Robotics And Automation*, Vol. 8, pp. 313-326.

Feddema J., 1989. Real Time Visual Feedback Control for Hand-Eye Coordinated Robotic Systems. PhD thesis, Purdue University.

Feddema J. and O. Mitchell, 1989. Vision-guided Servoing With Feature-based Trajectory Generation, *IEEE Trans. Robotics and Automation*, vol. 5, pp. 691-709.

Feddema J. T., C. S. G. Lee and O. R. Mitchell, 1991. Weighted Selection of Image Features for Resolved Rate Visual Feedback Control, *IEEE Trans. on Robotics and Automation*, vol. 7, pp. 31-47.

Feder H. S. and J. E. Slotine, 1997. Real-time Path Planning Using Harmonic Potentials in Dynamic Environments. In Proceedings of the *IEEE International Conference on Robotics and Automation (ICRA)*, Albuquerque, New Mexicopp, pp.874-881.

Fox D., W. Burgard and S. Thrun, 1997. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation Magazine*, pp.23-33.

Fox D., W. Burgard, S. Thrun and A.B. Cremers,1998. A Hybrid Collision Avoidance Method for Mobile Robots. In Proceedings of the *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, Leuven, Belgium, pp.1238-1243.

Gonzalez R. C. and R. E. Woods, 2001. Digital Image Processing. Prentice Hall.

Hager G. and K. Toyama, 1998. XVision system: A General-Purpose Substrate for

Portable Real-time Vision Applications. *Computer Vision and Image Understanding*, pp.23-37.

Hill J. and W. T. Park, 1979. Real Time Control of a Robot With a Mobile Camera, Proc. *9th International Symposium on Industrial Robots(ISIR)*, pp. 233-249.

Hu X., D. F. Alarcon and T. Gustavi, 2003, Sensor-Based Navigation Coordination for Mobile Robots. The 42nd *IEEE Conference on Decision and Control*, Maui, Hawaii.

iRobot Mobility Robot Integration Software User's Guide, Part Number: 2841, Rev. 6.

iRobot Mobile Robot User's Guide, Part Number: 2838, Rev. 3.

Isard M. and A. Blake, 1996. Contour Tracking by Stochastic Propagation of Conditional Density. Proc *European Conference on Computer Vision*, LNCS no. 1064, Springer-Verlag, pp. 343-356.

Isard M. and A. Blake, 1998. ICondensation: Unifying Low-level and High-level Tracking in a Stochastic Framework. *Proc 5th European Conf. Computer Vision*, Vol. 1, pp: 893-908.

Jin Y. and M. Xie, 2000. Vision Guided Homing for Humanoid Service Robot. In Proc. *International Conference on Pattern Recognition (ICPR)*, pp. 511-514, Barcelona, Spain.

Khatib O., 1985. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. Journal of Robotics Research*, 5, 90-98.

Kuc R. and B. Barshan, 1989. Navigating Vehicles Through an Unstructured Environment With Sonar. Proc of the 1989 *IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona, pp. 1422-1426.

Khatib M. and R. Chatila, 1995. An Extended Potential Field Approach for Mobile Robot Sensor-based Motions. In Proceedings of the *International Conference on Intelligent Autonomous Systems (IAS)*, Karlsruhe, pp.490-496.

Koike C., C. Pradalier, P. Bessierel and E. Mazeraz, 2003. Obstacle Avoidance and Proscriptive Bayesian Programming. *Eighteenth International Joint Conference on Artificial Intelligence.* Acapulco (Mexico)

Konolige K., 2000. A Gradient Method for Realtime Robot Control. Proc of the *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Krogh B. H., 1984. A Generalized Potential Field Approach to Obstacle Avoidance Control. International Robotics Research Conference, Bethlehem, Pennsylvania, pp.950-955.

Lumelsky V. and A. Stepanov, 1987. Path Planning Strategies for a Point Mobile Automation Moving Amidst Unknown Obstacle of Arbitrary Shape. *Algorithmica*, pp.403-430.

Malis E., F. Chaumette and S. Boudet, 1999. *2-1/2-d visual servoing, IEEE Trans. Robot. Autom.*, vol. 15, pp: 238-259.

Malis E., 2002. Survey of Vision-based Robot Control. *ENSIETA European Naval Ship Design Short Course*, Brest, France.

Minguez J. and L. Montano, 2000. Nearness Diagram Navigation: A new real time collision avoidance approach. Proc of the *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Moravec H. P., 1986. Certainty Grids for Mobile Robots. Preprint of Carnegie-Mellon University, The Robotics Institute, Technical Report.

Moravec H. P. and A. Elfes, 1985. High Resolution Maps From Wide Angle Sonar. In Proc. *International Conference on Robotics and Automation (ICRA)*, pp. 116-121, St. Louis, Missouri.

Newman W. S. and N. Hogan, 1987. High Speed Robot Control and Obstacle Avoidance Using Dynamic Potential Functions. Proceedings of the *IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, pp. 14-24.

Nourbakhsh I., D. Andre, C. Tomasi and M. Genesereth, 1997. Mobile Robot Obstacle

Avoidance via Depth from Focus, *Robotics and Automation Systems*, Vol. 22, pp. 151-158.

Ratering S. and M. Gini, 1995. Robot Navigation in a Known Environment with Unknown Obstacles. *Autonomous Robots*, pp.149-165.

Rives P., F. Chaumette and B. Espiau, 1989. Positioning of a Robot With Respect to an Object, Tracking it and Estimating its Velocity by Visual Servoing, in *Experimental Robotic 1* (V. Hayward and O. Khatib, eds.), vol. 139 of Lecture Notes in Control and Information Sciences, pp. 412-428, Springer-Verlag.

Samson C., B. Espiau and M. L. Borgne,1990. Robot Control: the Task Function Approach. Oxford University Press.

Schlegel C., 1998. Fast Local Obstacle Avoidance Under Kinematic and Dynamic Constraints for a Mobile Robot. In Proceedings of the *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Shapiro L. G. and G. C. Stockman, 2001. Computer Vision. *Prentice-Hall* .

Shekhar S., 1997. Wheel Rolling Constraints and Slip in Mobile Robots, in Proc. *IEEE Int. Conf. Robotics and Automation*, Vol. 3, pp. 2601-2607.

Siegwart R. and I. R. Nourbakhsh, 2004. Introduction to Autonomous Mobile Robots, *The MIT Press.*

Simmons R. and S. Koenig, 1995. Probabilistic Robot Navigation in Partially Observable Environments. In Proc. *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1080-1087, Montreal, Canada.

Spero D. J., 2004. A Review of Outdoor Robotics Research. Technical Report MECSE-17-2004

Strobel M., 1999. Navigation in Partially Unknown, Narrow, Cluttered Space. In Proceedings of the *IEEE International Conference on Robotics and Automation (ICRA)*, pp.29-34.

Tayebi A. and A. Rachid, 1996. Path Following Control Law For an Industrial Mobile

Robot. Proceeding of *IEEE International Conference on Control Applications*, Dearborn, MI, pp. 703-707.

Ulrich I. and J. Borenstein, 1998. VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots. In Proceedings of the *IEEE International Conference on Robotics and Automation (ICRA)*, pp.1572-1577.

Ulrich I. and J. Borenstein, 2000. VFH*: Local Obstacle Avoidance With Look-ahead Verification. In Proceedings of the *IEEE International Conference on Robotics and Automation (ICRA)*, pp.2505-2511.

Wang C.C., 1992. Extrinsic Calibration of a Vision Sensor Mounted on a Robot, *IEEE Transactions on Robotics and Automation*, Vol. 8, pp. 161-172.

Watanabe K., Y. Shiraishi, S. Tzafesteas, J. Tang and T. Fukuda, 1998. Feedback Control of an Omnidirectional Autonomous Platform for Mobile Service Robots, *Journal of Intelligent and Robotic Systems*, pp. 315-330.

Westmore D. B. and W. J. Wilson, 1991. Direct Dynamic Control of a Robot Using an End-point Mounted Camera and Kalman Filter Position Estimation, in Proc. *IEEE Intl Conf. on Robotics and Automatio*n, pp. 2376-2381.

Wilson W., 1993. Visual Servo Control of Robots Using Kalman filter Estimates of Relative Pose, in Proc.*IFAC 12th World Congress*, (Sydney), pp. 399-404.

Wilson L., C. Williams, J. Yance, J. Lew, R. L. Williams II, 2001. Design and Modeling of a Redundant Omni-directional RoboCup Goalie. *RoboCup AI Conference*, Seattle, WA.

Zavlangas P. G., S. G. Tzafestas and K. Althoefer, 2000. Fuzzy Obstacle Avoidance and Navigation for Omnidirectional Mobile Robots. ESIT 2000, 14-15 September 2000, Aachen, Germany, pp. 375-382

Appendix A

Operation of The B21R Mobile Robot

In early 2004, shortly after the Irobot Company bought Real Word Interface Inc. (which is the original producer of the B21R), the Irobot Company stopped to produce the research robots and no technical support was available any more. The contents in this section are obtained either from the B21R's on-board computer or the answers from the broker. So, it is valuable information for using B21R mobile robot.

A.1 Beginning a Remote Session with the Mobile Robot from a Networked Terminal (with Hummingbird Xceed installed and configured)

Step 1. Connect the power supply of the Station Adapter radio. (Note: This must be done before turning on the B21R mobile robot; otherwise the B21R mobile robot will not establish the communication link with the network.)

Step 2. Open the base door with a white dot directly to the right of the base door with a red dot. Close the two power switches of the base part and the enclosure part.
Step 3. Push the rFLEX control knob to turn on the on-board computer.
Step 4. Wait for two beeps from the mobile robot (one low, one high beep, usually takes 3 to 4 minutes). This means that the on-board computer has successfully booted.
Step 5. On the terminal computer, open the program "hummingbird connectivity V9.0/ Exceed/ Xstart ", double click "mobile robot" and input the IP address (192.168.108.30), the User name (mobility) and password (mbyrwi). This will allow to log in the on-board computer of the B21R mobile robot. (If the Xceed program is setup correctly, only the password is needed here.)
Step 6. When the terminal window opens, at the command prompt type: "xterm &", this opens a new terminal window and establishes host-terminal X communication.
Step 7. In the first terminal window that was opened, at command prompt type: "name -I" to activate the name server of the B21R mobile robot.
Step 8. Input "xterm &" to open another X_Window. In the new X_Window, type the command "base" to activate the Base Server of the B21R mobile robot. The robot itself is encapsulated in the Hardware child object of the server. It provides access to the Drive system, the sonar sensors, the infrared sensors, the tactile sensors and the battery voltage. (The B21R mobile robot should begin making the sonar clicking sounds.)
Step 9. Open another X_Window and input "dpptserver". The "dppserver" is a Mobility server for the Directed Perception Pan Tilt unit that is part of the RWI High-Performance Vision system. The input to the drive command will control the motion of the pan-tilt unit and the state and raw values of the drive component reflect the position of the pan-tilt unit to any interested client objects.
Step 10. Open another X_Window and input "v4lserver". The "v4lserver" is a Mobility server for the DBS DSG/LC1 frame grabber card that is part of the High-Performance Vision system. The server provides access to the raw image captured

from the card through the "RawImage" child object, as well as access to a compressed version of the image through the "ZImage" child.

Step 11. Open another X_Window and input "mom" to start the graphical user interface between the terminal and the host computer.

A.2 Ending a Remote Session with the Mobile Robot from a Networked Terminal (with Hummingbird Xceed installed and configured)

Step 1. Close any graphical Windows.

Step 2. In the terminal window one use the commands: Ctrl+d and then Ctrl+c to end a session and disconnect from the mobile robot.

Step 3. From the rFLEX control panel sellect "Host console" and then "shutdown PC". When the screen shows "power down" then choose "kill PWR". (Note: Do not miss the sequence, otherwise the on-board computer will lose data and restart very slowly next time.)

A.3 Establishing, editing, and running a New Program in the On-board Computer of the B21R Mobile Robot

The operating system of the B21R mobile robot is Linux. If we want to develop a new application program on the base of other programs, we can first log in the target location, and then use command:

cp -R "path of the source directory "

to load the desired programs to the target directory. Thereafter, we can use the command

emacs "name of the program"

to edit the program. We rename the program using the command:

mv "the old name of the file" "the new name of the file"

After we change the name of the program, we need to modify the content of the "Makefile" in corresponding directory. In the Makefile, there should be a line like this:

PROG = "name of the CPP file"

And we should change the name of the CPP file to the current name which we are using. Finally, we use command "make" to compile the new program and get the executable program.

A.4 Checking the Document of B21R

In the directory " /home/mobility/mobility-b-1.1.8nb/docs/ ", there are some "html" files which are the documents related to the B21R mobile robot. They can be read by using "emacs" command in Linux system. However, they also can be browsed in Windows system by using the IE explorer to visit the following web site "http://mobilerobot.lakeheadu.ca/~ mobility/docs/". Of course, the foundation of acquiring these documents is that the B21R mobile robot is successfully connecting to the internet.

A.5 Shutting down Sonar or Pan-tilt When They Lose Control

Sometimes, the sonar sensors or the pan-tilt unit can lose control. For example, the sonar sensors stay active after you close the Base server and the pan-tilt unit keeps moving randomly. When these phenomena happen, the sonar sensors or the pan-tilt unit should be shut down immediately to avoid any damage further.
In the rFLEX control panel "main menu", choose "sonar console", and then choose "off" to close the sonar sensors.
By opening the two enclosure doors at the same side of the camera, one can see a small black box labeled "Directed Perception Pan Tilt Controller". Turn off the switch for shutting down the pan-tilt unit.

A.6 Charging the Batteries

The current voltage of the batteries is indicated in the rFLEX control panel. The normal value is between 21 and 24. When the value is lower than 21, one should be careful and do not let the robot travel a long distance since the power maybe run out at any time. When the value is lower than 20.5, the batteries need to be charged. Charging time should be around four hours and be careful that never charge the batteries more than six hours.

Appendix B

Calibration of The IR sensors

The IR sensor data requires calibration data to have any accuracy. When calibrated, the sensors report accurately and repeatedly for objects within the sensor range (0.1 to 0.8 meters from the sensor surface). If there is no object for a very long distance, the sensor reports essentially random numbers. For objects closer than the range, the sensor repeatedly reports longer distances. For example, at 0.04m, most sensors report a distance of about 0.3m. The end result is that given a sensor reading, it may be an object at that distance, it may be an object very close to the sensor, or it may be no object. With a great deal of care it is possible to extend the accurate range of the sensor to approximately 0.07m to 1.0m.

The calibration procedure is to place a piece of paper facing each IR sensor at a distance of 0.6 meters, and record the hexadecimal readings from the rFlex IR Console. Board 1 controls the sensors on the front door of the robot and board 0 the rear. Sensor 0 is adjacent to the latch of the door and sensor 7 is next to the hinge. Please refer to the file *$MOBILITY_ROOT/etc/ircalibration.template* for the file format. Unfortunately, some individual IR sensors appear to need recalibration fairly frequently, perhaps every day.

Appendix C

Running Sample Programs

C.1 Running the Simple-Follow Program

There are several example programs in the mobile robot's on-board computer, such as simple-follow, simple-follow2, seeker and serial server. However, some of them are not suitable for our B21R. For example, the simple-follow2 and the seeker programs need the laser rangefinder sensor which was not installed in our B21R mobile robot. Therefore, the simple-follow program is the most useful example program which shows the simplest kind of mobility client program that closes the loop around a robot base server. The simple-follow program takes sonar readings from the B21R mobile robot and drives forward until it is about 1 meter from the closest obstacle to the center of the robot. If the obstacle gets closer to the robot, it will back up. If any sensor reading is "too close" (about 30cm) the robot will stop moving to keep from hitting something.

At the beginning, step 1 to step 8 in section 9.1 (Beginning a Remote Session with the Mobile Robot from a Networked Terminal) need to be done.

Second, Input "xterm &" to open another X_Window. In the new X_Window, use the command "cd" to reach the desired directory " /home/mobility/mobility-b-1.1.8nb/examples/mby/simple-follow-final ".

Finally, type the command “test2 -robot B21R” to run the simple-follow program. Be careful that this command is case sensitive and do not forget the spaces.
At any time, pressing the enter key will stop the program.

C.2 Running the Trajectory Tracking Program

At the beginning, step 1 to step 10 in section 9.1 (Beginning a Remote Session with the Mobile Robot from a Networked Terminal) need to be done.
Second, Input “xterm &” to open another X_Window. In the new X_Window, use the command “cd” to reach the desired directory “ /home/mobility/mobility-b-1.1.8nb/examples/mby/track-point ”.
Finally, type the command “test2” to run the trajectory tracking program.
At any time, pressing the enter key will stop the program.