DESIGN OF A MODULAR AUTONOMOUS ROBOT VEHICLE

By

EZZALDEEN EDWAN

Bachelor of Science

Electrical Engineering

Birzeit University

Birzeit, Palestine

1997

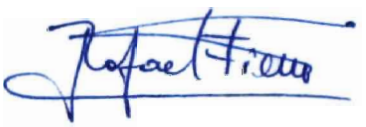
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

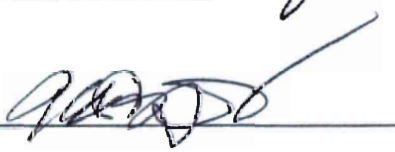
August, 2003

DESIGN OF A MODULAR AUTONOMOUS ROBOT VEHICLE

THESIS APPROVED:



Thesis Adviser



Dean of the Graduate College

Acknowledgments

I wish to express my sincere appreciation to my advisor, Dr. Rafael Fierro for his advice, guidance, encouragement and friendly supervision. My sincere appreciation extends to my other committee member Dr. Martin Hagan whose teaching and assistance are invaluable. I would like to thank Dr. Gary G. Yen for serving on the advisory committee for this thesis.

Special thanks to the Fulbright scholar program administered by the AMIDEAST for funding me during my academic program.

Thanks to the members of the MARHES laboratory team.

I would also like to give my special appreciation to my wife, Hanaa, for her patience and support during this work.

This work is dedicated to my parents.

TABLE OF CONTENTS

Chapter 1
Introduction

1.1 Motivation

In the recent years we have witnessed a revolution in information and sensors technology. These advances present unprecedented challenges and new opportunities for the years to come. We now have the necessary resources to develop an autonomous system. Developing unmanned vehicles for unknown, unstructured scenarios is an intensive area of current research. In a near future, these vehicles will navigate autonomously everywhere from space, air, ground to undersea.

The aim of this thesis is to design a modular autonomous platform for Multi-Agent, Robotics, Hybrid and Embedded Systems (MARHES) laboratory. This vehicle will be a testbed for research in cooperative multi-vehicle systems. In order to be autonomous, it must have localization techniques and navigation control strategies.

1.2 Applications

Mobile robotics is a very important field of research because of their many potential applications. Some applications reported in the literature include

1 Intelligent Vehicles

A fully autonomous vehicle is the one in which a computer performs all the tasks without human intervention. The vehicle makes the necessary maneuvers and speed changes to perform the required tasks. A completely automated vehicle is still a challenging task, however advances have been made in automating some tasks. Cruise control is very common in modern cars. Also, adaptive

cruise control in which the vehicle automatically adjust its speed to maintain a safe following distance, has become reality.

2 Search and Rescue

The mobile robots have been used in search and rescue missions [11].

3 Agriculture Automation

Robots have been utilized in agriculture automation. An example is the loan-mower robot described in [21].

4 Space Exploration

National Aeronautics and Space Adminstration (NASA) and Jet Propulsion Laboratory (JPL) programs, have been involved in unmanned robotic inter-planetary probes.

5 Military Systems

Autonomous robots are used in military applications such as reconnaissance. The objective is to develop one or more systems with substantial degree of autonomy that could be operated by persons with limited technical training [9].

6 Disabled Assistance

Mobile robots are utilized in assistance of disabled people.

7 Entertainment

The AIBO robot is a good example of entertainment robots [30].

8 Domestic Robots

Commercial robots are becoming more common today. Some companies started manufacturing and developing commercial robots for domestic purposes. A good example are the robots developed by iRobot [32]. Roomba is an automatic vacuum that uses intelligent navigation technology to automatically clean homes. Another, Evolution Robotics develops and manufactures multifunctional personal robots for the home and workplace [29].

9 Museum Robots

MINERVA is an interactive tour-guide robot, which was successfully exhibited in a Smithsonian museum. During its two weeks of operation, the robot interacted with thousands of people [26].

1.3 Why We Need a Modular Mobile Robot?

Now days there is an increasing demand on modular mobile platforms for research and the previous listed applications. The prices of these platforms are very high which limit many researchers from doing experimental work on mobile robots. Another restriction is that most of the existing platforms are limited to indoor use only. In the MARHES laboratory, we have developed a modular platform with off the shelf components that are available at a reasonable cost. The designed vehicle is low cost compared with available platforms and can be used for indoors and outdoors operation. The processing unit is a modern laptop that can be replaced or upgraded with the latest computing technologies. The laptop has a firewire port for vision and an integrated wireless communication that can be suited for cooperative sensing and

exploration.

The total weight is less than 10 kg. This light weight offers opportunity for portability. The suite of sensors includes IR sensors, quadrature encoders, GPS unit, and NI I/O multi-function card.

1.4 Organization

This thesis is organized as follows

- Introduction

  Chapter 1 explains the importance of this research. Reasons for the need of this research and advantages of the platform are detailed.

- Kinematic Modeling of the Vehicle

  The kinematic models of unicycle and car-like robot are derived in Chapter 2.

- Hardware Description

  In Chapter 3, the car hardware and operation of sensors are described in full details.

- Software Integration

  Chapter 4 describes the software architecture. Object oriented multi-threading and real-time issues are discussed.

- Tools for Vehicle Control.

  Chapter 5 describes the design and implementation of algorithms for controlling the robot.

- Navigation and Localization.

  Chapter 6 presents the nonlinear control laws implemented for target accusation using an extended Kalman filter (EKF).

- Conclusion and Future Work

  Chapter 7 concludes the thesis and presents ideas for future work.

1.5 Contribution of this Work

The purpose of this study is to develop an indoor/outdoor low cost modular mobile platform for research in cooperative multi-vehicle systems. This thesis also gives details of hardware implementation and software architecture. The software architecture was realized using object-oriented multi-threading visual C++. The operation of each sensor type is discussed and the handling of sensory data is explained.

This work can be used as a reference for researchers who want to build a cost effective mobile robot. The nonholonomic kinematic model derived in this work was validated both theoretically and experimentally. Calibration of odometry procedure is explained as well. Several controllers were tested on this model to prove its efficiency. A velocity-controlled vehicle was designed using PID controllers for regulating both linear and angular speed.

An extended Kalman filter (EKF) is implemented to fuse information from dead reckoning sensors and low cost WAAS enabled Global Positioning System (GPS) unit for localization and control purposes. Input/Output feedback linearized controller for leader following is used in target acquisition.

Chapter 2
Kinematic Modeling of the Vehicle

In this chapter, the kinematic models for the unicycle and car-like robot are derived. We will show that the unicycle model is a valid model for the car-like robot considering linear velocity has a non zero value. This will be the base model for implementing robot controllers.

2.1 Mobile Robot Modeling

In our analysis, we will consider the kinematic model which deals only with nonholonomic constraints that result from rolling without slipping between wheels and ground.

2.1.1 Kinematic Model of Disc Wheel (Unicycle)

To derive the mathematical model of the plant (i.e. car-like robot) we will, for simplicity, consider the car as a rolling disc wheel (unicycle) shown in Figure 2.1. Assuming pure rolling and no slipping, the nonholonomic constraints become

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \tag{2.1}$$

where (x, y) are the Cartesian position, and θ is the orientation with respect to the positive x axis. The kinematic model is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} \tag{2.2}$$

where

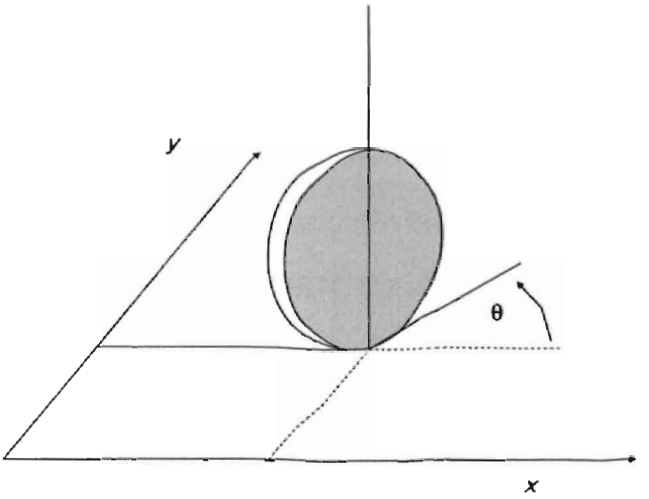


Figure 2.1: The global coordinate system for the unicycle

v : the linear velocity of the wheel.

ω : the angular velocity of the disc.

2.1.2 The Car-Like Model (Front Steering)

Consider the car-like model shown in Figure 2.2. In this model, the front wheels are steerable while the rear wheels have fixed orientation. For simplicity, we will assume both wheels on each axle collapse into a single wheel located at the midpoint of each axle [16]. The generalized coordinates of the rear wheel are $q = (x, y, \theta, \phi)$, where (x, y) are the Cartesian coordinates of the rear wheel, θ is the orientation of the car body with respect to the positive x axis, and ϕ is the steering angle. For both front and rear wheels we can write the nonholonomic constraints as

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \tag{2.3}$$

$$\dot{x_f}\sin(\theta+\phi) - \dot{y_f}\cos(\theta+\phi) = 0 \tag{2.4}$$

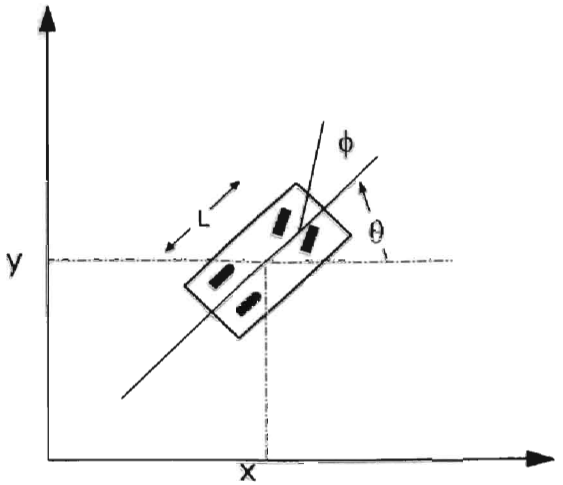


Figure 2.2: The global coordinate system for the car (front steering).

where (x_f, y_f) is location of the midpoint of front axle that is given by

$$x_f = x + l\cos\theta \tag{2.5}$$

$$y_f = y + l\sin\theta \tag{2.6}$$

where l is the distance between the two axles. Taking the derivative of equations (2.5) and (2.6) gives the velocity as

$$\dot{x_f} = \dot{x} - l\dot{\theta}\sin\theta \tag{2.7}$$

$$\dot{y_f} = \dot{y} + l\dot{\theta}\cos\theta \tag{2.8}$$

Solving equations (2.4), (2.7), and (2.8) to find $\dot{\theta}$

$$\dot{\theta} = \frac{\tan\phi}{l} v_1 \tag{2.9}$$

Now the complete kinematic model is given as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} cos\theta \\ sin\theta \\ \frac{tan\phi}{l} \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2 \tag{2.10}$$

where v_1 and v_2 are the driving and steering velocity inputs, respectively.

2.1.3 The Car-Like Model (Double Steering)

In double steering we have both the front wheels and rear wheels to be steerable. This configuration gives us a smaller radius of curvature compared to single steering. With this feature we can have faster convergence to the desired orientation of car body. Double steering does not affect the degrees of freedom for mobile robot. Figure 2.3 shows a model for double steering. In this model, we will assume both steering angles are equal and opposite in direction to simplify the computing of the kinematic model.

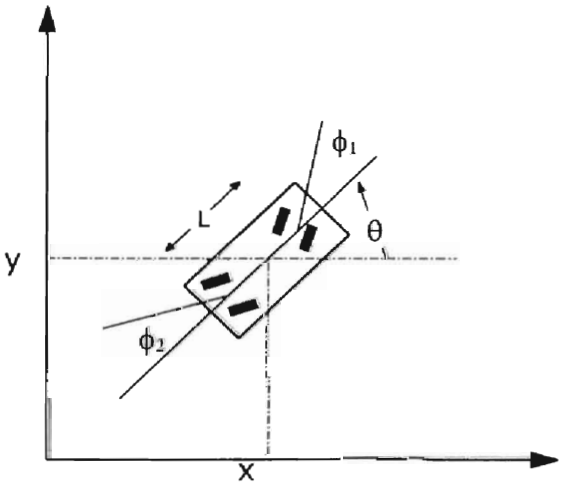


Figure 2.3: Global coordinate system for the car (double steering).

Taking (x, y) as the cartesian coordinates of the point in the center of the car. We can apply rolling without slipping to both wheels in the same way we did in previous derivation. We get the kinematic model as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} cos\theta \\ sin\theta \\ 2\frac{tan\phi}{l} \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2 \tag{2.11}$$

where v_1 and v_2 are the driving and steering velocity inputs, respectively.

2.2 Curvature and Steering Angle (Double Steering)

To get a better understanding of how the robot behaves for the commanded steering angle, we find the relationship between the steering angle (double steering model) and the curvature. This relationship is affine. If both rear and front wheels are fixed at a certain angle, the car will describe a circular path of certain radius. The curvature $c(s)$ is equal to $1/R$, where R is the radius of described circle. We will describe how to find the radius experimentally and theoretically.

2.2.1 Finding Curvature from Simulation

We used equation (2.11) to find curvature by simulating the car motion at different fixed steering angles. As expected, we got different circular paths, each with a specific radius. To find the radius of each circle, we need to know three points on the circumference of the circle. If a, b and c are the lengths of the sides a triangle as shown in Figure 2.4 and K is the area of the given triangle, then the radius R of this circle can be found as follows

$$R = \frac{abc}{4K} \tag{2.12}$$

During simulation, we let ϕ vary from $0°$ to $22.5°$ (the steering angle limits). The robot length is 33 cm.The relationship was found by fitting those points by affine equation. We got the following formula

$$c(s) = \alpha\phi + \beta \tag{2.13}$$

with $\alpha = 6.0$ and $\beta = 0.002$. We will provide the mathematical explanation of this affine relationship between the steering angle ϕ and the curvature. From equation

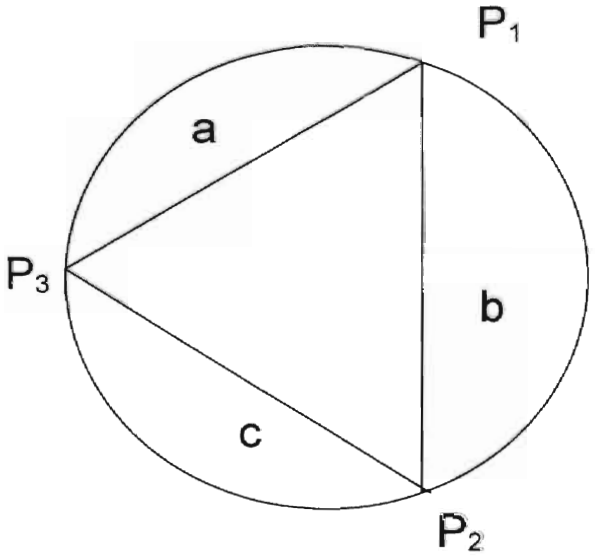


Figure 2.4: A circle circumscribed around a triangle, containing all three vertices

(2.11), we have

$\dot{\theta} = \nu_1/R, \qquad \dot{\theta} = 2\frac{tan\phi}{l}\nu_1$

Hence, we have

$$\frac{1}{R} = 2\frac{tan\phi}{l} \tag{2.14}$$

we approximate $tan\phi$ by ϕ Because the steering angle ϕ is limited by $\pm 22.5°$. As a result, equation (2.13) has $\alpha = \frac{2}{l}$ and $\beta = 0$.

2.2.2 Finding Curvature Experimentally

First, the steering servos were calibrated to have a 0° steering angle in both wheel axles at neutral position. The maximum steering angle was found to be ± 22.5°. We commanded the front wheels with a specific steering angle and the rear wheels with the same angle but in opposite direction. We picked three points on the circular described path and computed the radius of curvature from equation (2.12). We recorded this data in table and processed it using Matlab. As seen from the plot in

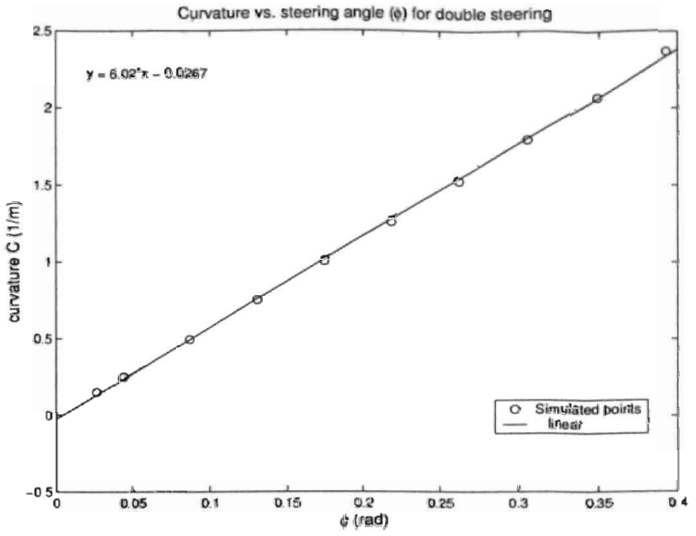


Figure 2.5: Curvature vs. steering angle (ϕ) for double steering (simulation).

Figure 2.6, the relation is affine and the values of α and β are close to the theoretical one. The reason why we did not get exact parameters is because an experimental car has two wheels in the front and two wheels in the rear. The kinematic model assumes they collapse into single wheel. Another reason because theory assumes the nonrealistic case of no slippage. If the robot follows a curved path, the slippage will occur on every single wheel. The slippage is directly proportional to the wheel width and it is impossible to have a wheel with zero width. The slippage occurs more frequently with larger steering angles. When the steering angle is small, the slippage is small. It disappears if the steering angle is set to $0°$. From the linear fitting, we found the values $\alpha = 5.2$ and $\beta = 0.0066$.

2.2.3 Maximum Reachable Angular Velocity

The maximum steering angle places a limitation on the maximum angular velocity that can be achieved by the robot. The limits on angular velocity is important

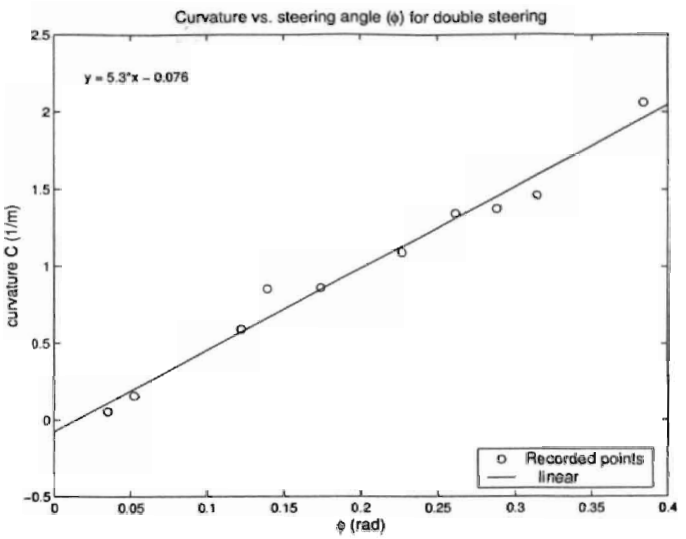


Figure 2.6: Curvature vs. steering angle (ϕ) for double steering (experimentally).

because control effort will be limited in real applications. The minimum radius of curvature occurs at maximum steering angle which is in our case $\pm 22.5°$.

$$R_{min} \approx \frac{1}{\alpha \phi_{max}} \tag{2.15}$$

The parameter α is found experimentally to be $\alpha = 5.2$, therefore, $R_{min} \approx 0.5\ m$. The angular velocity is constrained by

$$-\omega_{max} \leq \omega \leq \omega_{max} \tag{2.16}$$

where ω and $-\omega$ refer to counter clock wise and clock wise rotation, respectively. The maximum angular velocity ω_{max} for a given constant linear speed v can be computed from the following equation

$$\omega_{max} = \frac{v}{R_{min}} \tag{2.17}$$

therefore, equation (2.16) becomes

$$-\frac{v}{R_{min}} \leq \omega \leq \frac{v}{R_{min}} \tag{2.18}$$

From the above equation, we can find the limits by substituting in the minimum radius of curvature. The relationship is plotted in Figure 2.7.

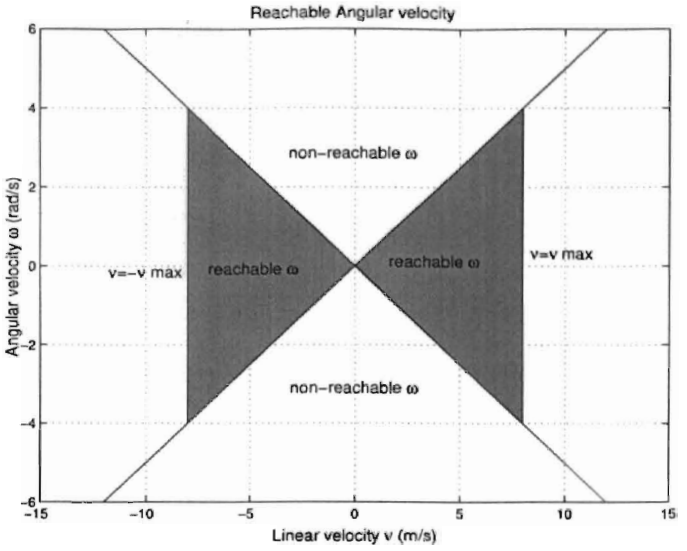


Figure 2.7: Angular speed vs. linear speed relation

2.3 The Unicycle Versus Car-Like Model

Considering the first three states (x, y, θ) in equation (2.10), the car-like model reduces to the unicycle model defined in equation (2.2) with $\omega = 2\frac{tan\phi}{l}v_1$. In practice, we are interested in controlling ω instead of controlling steering angle rate $\dot{\phi}$. Using the previous approximation of $tan\phi \approx \phi$, we can express ω as

$$\omega \approx 2\frac{\phi}{l}v_1 \tag{2.19}$$

where ϕ is in *radians*, v_1 is in m/s, and l is in m.

The above approximation is valid only when $v_1 \neq 0$. In the following chapters, we use equation (2.19) to control ω.

Chapter 3
Hardware Description

In this chapter the car hardware and sensors will be detailed.

3.1 The Experimental Testbed

The MARHES Laboratory X-treme robots (see Figure 3.1) are based upon TXT-1, a commercially available radio control truck from Tamiya Inc., with significant modifications. The TXT-1 is designed similar to a full size monster truck with an aluminum ladder frame and multi-link suspension. Solid axles with a cantilever system allows for massive suspension articulation. The kit includes the hardware and a second servo for four wheel steering. The servo (or servos) are located next to the axle for direct steering. The kit can operate on a 7.2 V or 8.4 V battery. We used a 7.2 V battery to limit the speed. The robot has a servo controller for steering and a PWM speed controller for forward/backward motion.

An on-board notebook computer provides the computational power for signal/image processing, motion control and IEEE 802.11b wireless networking. Also, a PCMCIA Multi-function I/O card from National Instruments was used for interfacing the computer with a suite of analog and digital sensors. The suite of sensors includes IR distance sensors odometer, GPS receiver, compass and stereo vision camera as we will detail them in this chapter.

The included 3 Step mechanical speed controller is replaced by a Novak Super Rooster reversible electronic speed controller. The new speed controller is used to limit the current drawn by the drive motor, provide a safe transition from forward

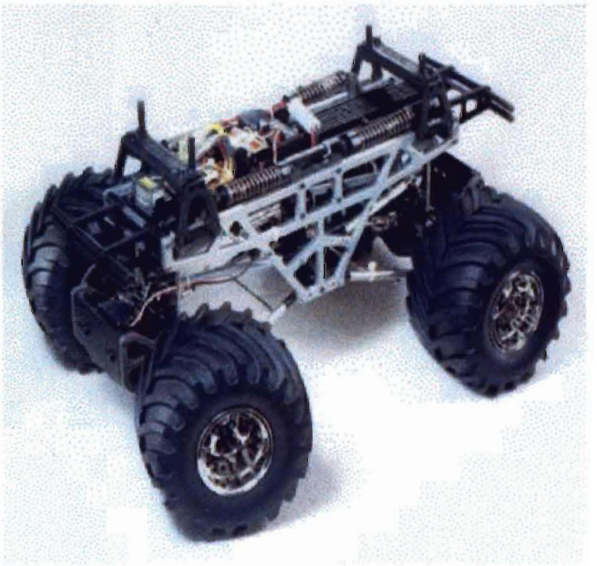


Figure 3.1: Radio control truck without any modifications.

to reverse using smart braking circuitry, and provide a power source for high-torque steering servo motors. The standard steering servo was replaced by a high-torque servo to supply an adequate amount of torque capable of steering the wheels under loads.

A pulse width modulated signal (PWM) is used to control the servos. The mini SSC is an electronic interface that allows a computer to control up to eight servos. The computer sends simple commands to the mini SSC at 2400 or 9600 baud rate, and the mini SSC generates eight channels of precise, stable servo-control pulses. The Mini Serial Servo Controllers (SSC) is a fully assembled module that includes a convenient phone-style jack for serial hookup, Futaba-J servo output headers, a sync LED to indicate when valid data is received, and a switchable servo range/resolution (90° range with 0.36° resolution, or up to 180° motion with 0.72° resolution).

Figure 3.2 shows a block diagram of robot hardware.

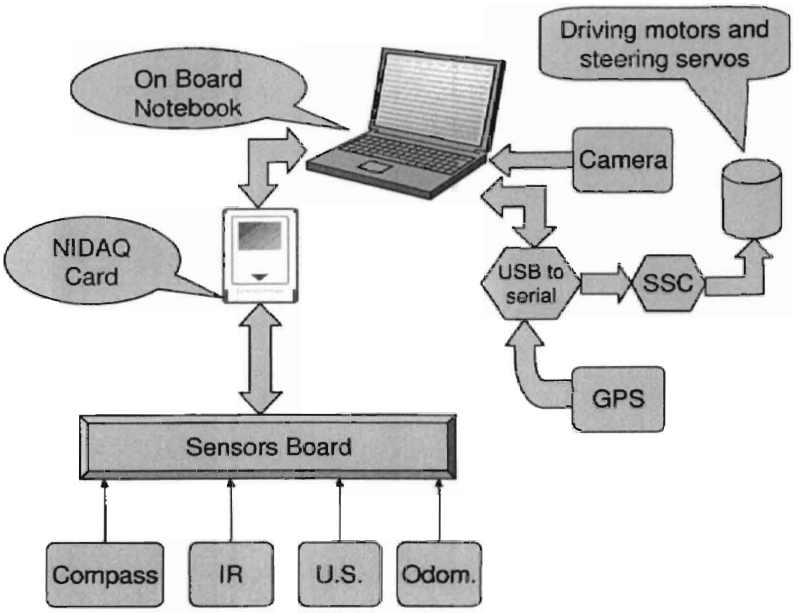


Figure 3.2: Block diagram of robot hardware

The final shape of the robot after implementing all hardawre modifications is shown in Figure 3.3. In the following sections we will describe the operation of each hardware component.

3.2 Short Distance Sensors

This sensor was chosen because of the low price and simplicity of connecting to the multi-function I/O card. This sensor takes a continuous distance reading and reports the distance as an analog voltage with a distance range of 20 cm ($\simeq$ 8 $inch$) to 150 cm ($\simeq$ 60 $inch$). The interface is 3-wire with power, ground and the output voltage.

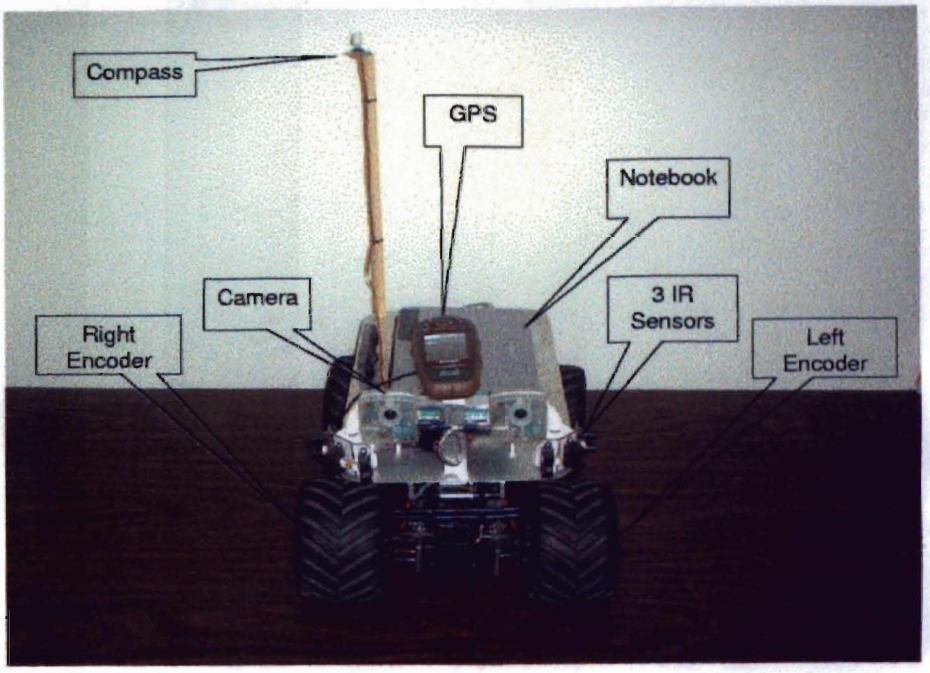


Figure 3.3: Robot after modifications

3.2.1 Calibration of the IR Sensors

To accurately read the IR sensors, they need to be calibrated to find the exact voltage corresponding to a specific distance. The reading is done using the multi-function I/O card. Voltages measurement corresponding to specific distances were recorded. An xy plot was created using Matlab and a fourth degree polynomial was used to fit this data as shown in Figure 3.5. Since this fitting polynomial will be used frequently in our routines during robot navigation, the computing time will be relatively large to find the corresponding distance for a specific voltage. A simpler form is to use piecewise linearization by dividing data into two regions as shown in Figure 3.6 and Figure 3.7.

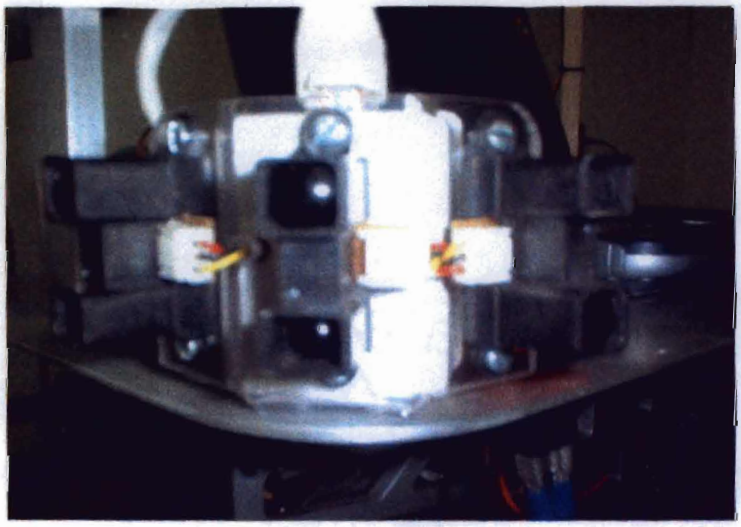


Figure 3.4: Three Sharp GP2Y0A02YK Infrared rangers configured in one unit.

3.3 Data Acquisition System

In order to interface data from the sensors with the notebook, we used the NI DAQCard-6024E for PCMCIA from National Instruments. This is considered low-cost compared to other NI cards. The analog output of short distance sensors is connected to the analog inputs of the card. The outputs of quadrature encoders are connected to the inputs of the counters of the accusation card. Technical specifications of this digital data acquisition card are listed in Table 3.1.

Table 3.1: NI DAQ card-6024E specifications

Analog Inputs	16 channels 200 kS/s, 12-Bit resolution
Analog Outputs	two channels 12-bit resolution
counters	two 24-bit counters/timers
Digital I/O	8 lines (5V/TTL/CMOS)

The I/O Connector Board

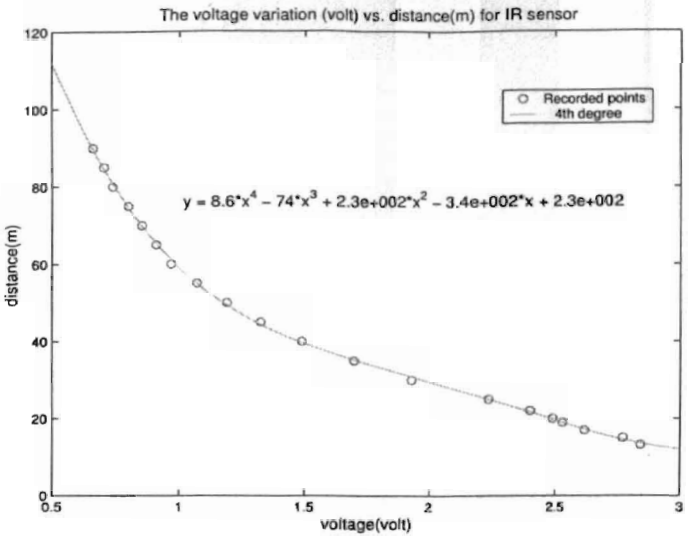


Figure 3.5: The voltage variation (volt) vs. distance(m) for IR sensor.

In order to connect the sensors and devices to the I/O acquisition card, we used a CB-68LPR I/O connector board as shown in Figure 3.8. It provides 68 screw terminals for easy connection of field I/O signals.

3.4 Dead Reckoning Sensors

Dead Reckoning (derived from "deduced reckoning" of sailing days) is a simple mathematical procedure for determining the present location of a vessel by advancing some previous position through known course and velocity information over a given length of time [10]. Dead Reckoning is defined as the process of calculating the location of a mobile robot from measurements of the angular rotation of odometer wheels. These wheels can be driven or free wheeling. The most simplistic implementation of dead reckoning is sometimes termed *odometry*. The term implies vehicle displacement along the path of travel and is directly derived from an onboard "odometer" [5]. Since the *odometry* is the most important sensor in our platform, a

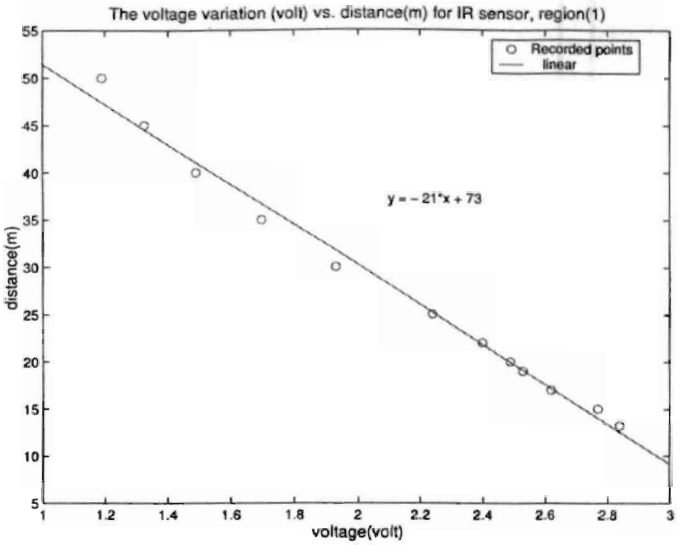


Figure 3.6: The voltage variation (volt) vs. distance(m) for IR sensor, region(1).

clear understanding of of how it works is necessary. Angular rotation of the wheel is measured by a rotary optical encoders attached to the wheel shaft. In the next section we will describe the operation of the optical encoder.

3.4.1 Optical Encoder

An optical encoder consists of a rotating disk, a light source, and a photodetector (light sensor). The disk, which is mounted on the rotating shaft, has coded patterns of opaque and transparent sectors. As the disk rotates, these patterns interrupt the light emitted onto the photodetector, generating a digital or pulse signal output. The encoding disk is made from: glass, for high-resolution applications (11 to 16 bits), plastic (mylar) or metal, for applications requiring more rugged construction (resolution of 8 to 10 bits). Quadrature encoders can be used to determine direction of rotation. This is done by adding a second channel, offset from the first, by 90°. A possible setup is shown in Figure 3.9. Channel A can be used to provide the number

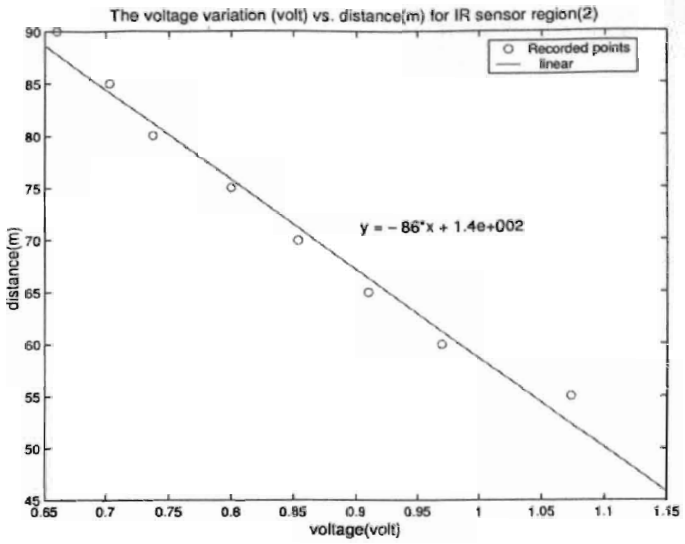


Figure 3.7: The voltage variation (volt) vs. distance(m) for IR sensor, region(2)

of counts and channel B for determining direction. We used two optical encoders placed inside each of the front wheels of the robot. Using two quadrature encoders provide us with both linear and angular displacements. The data output of each quadrature encoder is sent to one of the two counters in the data acquisition card. One direction signal from either of the quadrature encoders is adequate to tell about the direction of rotation in the 4-wheel robot. This is because it is impossible to have opposite direction of rotation in each of the front or rear wheels. In a differential drive robot, two direction signals are required because it is possible to have one wheel rotating in opposite direction to the other wheel, and hence it is possible to spin in its location.

The counter counts the rising edges for a specified period of time. The counter from the NI-DAQ card can be reset at any instant and is automatically reset once it reaches the maximum number counts. In our implementation we reset the counter to

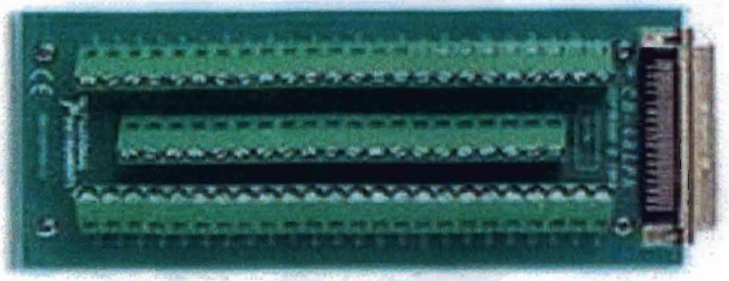
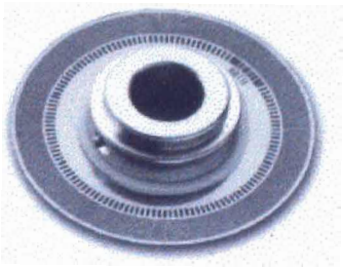


Figure 3.8: The I/O Connector board

some initial count value after reading the counted pulses to avoid overflow of accumulated pulses count. In each set period of time, the counter will start accumulating from count value of 10000. If the robot moves in the forward direction, the count will be incremented. On contrary, the count will be decremented if the robot moves backward.



State	chA	chB
1	High	Low
2	High	High
3	Low	High
4	Low	High

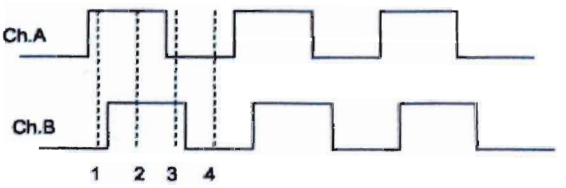


Figure 3.9: The observed phase relationship between Channel A and B pulse trains.

3.4.2 Installing Optical Encoder for the Car

We have used optical encoders from Agilent Technologies. The odometry system consists of the HEDR-8000/8100 Series encoders. It uses reflective technology to sense rotary or linear position. This sensor consists of an LED light source and a photodetector IC in a single SO-8 surface mount package. We have used a reflective codewheel HEDR-5120-H-12. The number of pulses for one complete revolution is 408. The surface mount chip is mostly affected by ambient lights, therefore, the best place to install it is inside the wheel. Doing so will also give the odometry system protection against any damages from crashing.

For installing the code wheel on the shaft, we first have to provide enough space inside the unmodified upright part. This is done by milling a distance equal to the width of the codewheel. Figure 3.10 and Figure 3.11 show the wheel well before and after modifications. The codewheel is mounted on the shaft of the car wheel in the milled space as shown in Figure 3.12. From the data sheet, it can be seen that the spacing between the code wheel and the chip is very important and should be done as specified ($2.03 \pm 0.51mm$) to get good results. The electronic circuit board of the chip is placed inside the end of the upright, keeping recommended spacing distance with codewheel.

3.5 Calibrating the Mobile Robot

Since the odometry system will depend on some parameters of the robot, the robot must be calibrated. Calibrating odometry for a 4-wheel robot is based on accurate measurement of the orientation angle as well as translations and rotations.

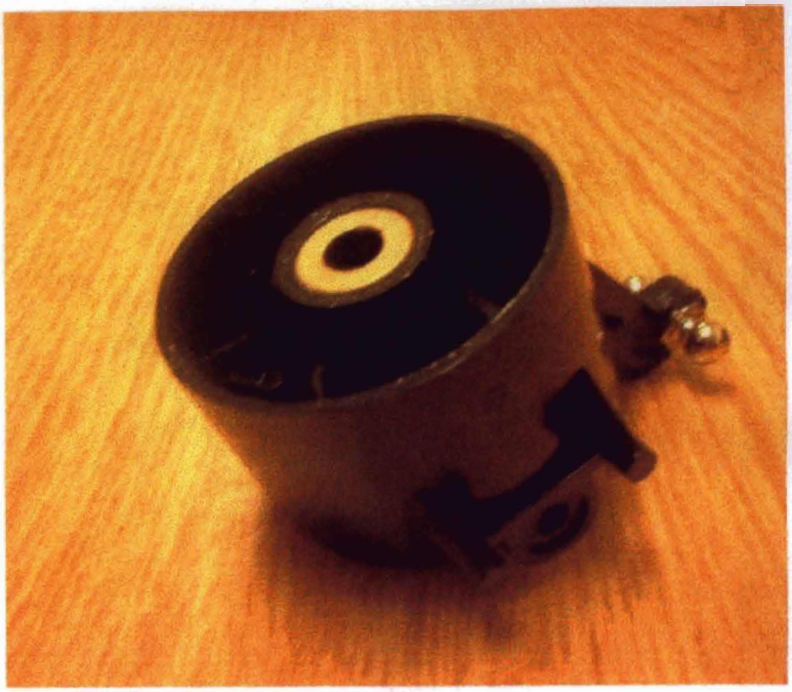


Figure 3.10: Original wheel well

We use the procedure described in [19] with some modifications. After the calibration process, we are able to use odometry effectively in practical applications.

3.5.1 Sources of Errors in Odometry

Sources of errors in odometry are classified into two main categories: systematic errors which result from construction tolerance and non-systematic errors (Dynamic errors) which result from changes in environment. In a 4-wheel robot, the systematic errors result from [5]:

- Unequal wheel diameters
- Average of actual wheel diameter differs from nominal wheel diameter.
- Actual wheelbase differs from nominal wheelbase.

Figure 3.11: Modified wheel well

- Misalignment of wheels.
- Finite encoder resolution.
- Finite encoder sampling rate.

The Nonsystematic errors result from [5]:

- Travel over uneven floors or unexpected objects over floor.
- Wheel slippage which results from
  - Slippy floors .
  - Overacceleration.
  - Fast turning (skidding).

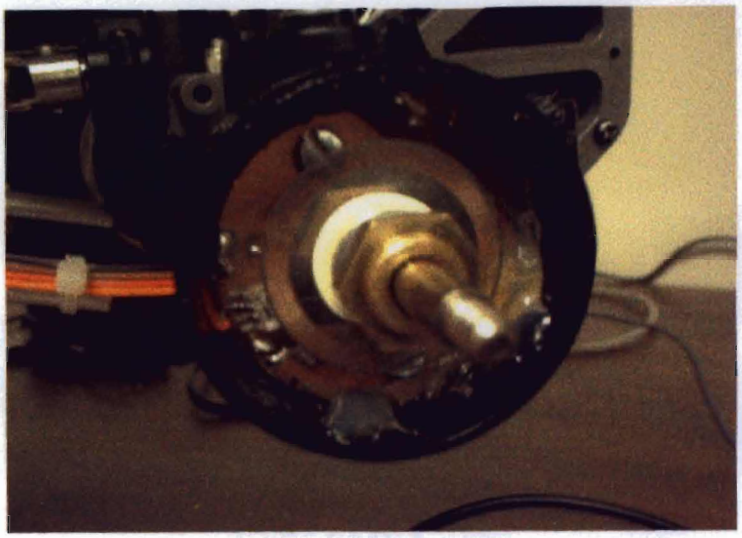


Figure 3.12: Installed codewheel and photodetector

- External forces (interaction with external bodies).
- Internal forces (castor wheels).
- Non-point wheel contact with the floor.

The calibration process is done to minimize systematic errors.

3.5.2 Odometry Equations and Calibration Parameters

When the wheels of the robot roll on the ground, the number of pulses generated from rotary encoders inside each front wheel is proportional to the travelled distance. The kinematic modeling of the robot requires a reference frame. Figure 3.13 shows the robot and its reference frame (x, y, θ) located midway between the front wheels. In our robot, two encoders are installed inside each front wheel.

The equation, which relates the change in travelled distance with the change of number of pulses in each wheel, is

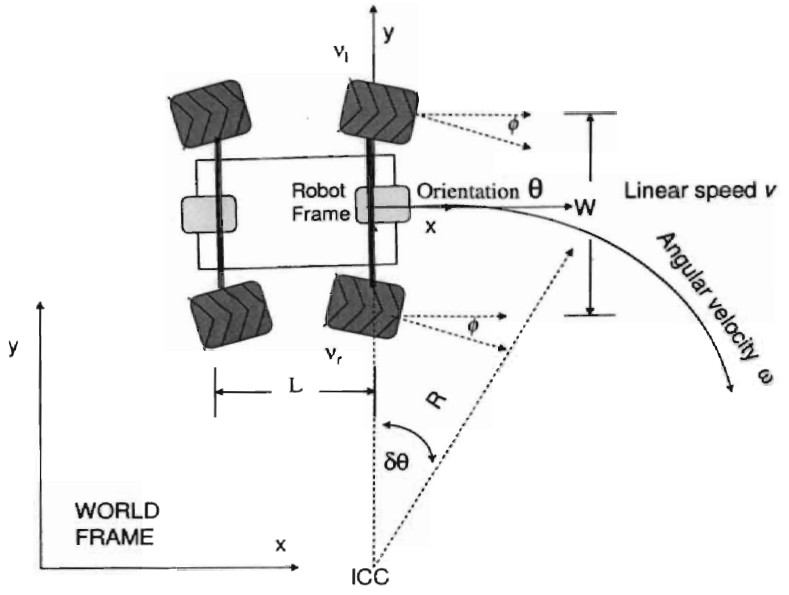


Figure 3.13: Kinematics of the robot

$$\delta d_l = k_l \delta n_l \tag{3.1}$$

$$\delta d_r = k_r \delta n_r \tag{3.2}$$

Where k_l and k_r are calibration parameters for each wheel, and their unit is (unit of length per number of pulses), δn_l and δn_r are the counted pulses in each wheel. The suffix l denotes the left wheel and r denotes the right wheel. Both of the two parameters (k_l , k_r) compensate for variation in wheel radius, tyre inflation, and gear ratio. The change in translational distance is expressed as

$$\delta d = \frac{\delta d_r + \delta d_l}{2} \tag{3.3}$$

and the travelled distance by the robot is

$$D = \frac{d_r + d_l}{2} \tag{3.4}$$

where d_r and d_l are evaluated at each time step as

$$d_r = d_r(t-1) + \delta d_r \tag{3.5}$$

$$d_l = d_l(t-1) + \delta d_l \tag{3.6}$$

The orientation angle of the robot θ after travelling a distance D is expressed in terms of each wheel travelled distance as

$$\theta = \frac{d_r - d_l}{W} \tag{3.7}$$

and the rotational displacement is computed in the same manner as

$$\delta\theta = \frac{\delta d_r - \delta d_l}{W} \tag{3.8}$$

where W is the robot base width.

The effective width in this case is the distance between the two encoders. By deriving equations (3.3) and (3.4), we can decompose the robot motion into a translation component and a rotation component. The linear velocity v and angular velocity ω of equation (2.1) are now expressed as

$$v = \frac{v_r + v_l}{2} \tag{3.9}$$

$$\omega = \frac{v_r - v_l}{W} \tag{3.10}$$

3.5.3 Calibration Procedure

First, the robot's front and rear wheels must be aligned at 0° steering angle. This is done using a straight edge that touches both side tires. To calibrate both k_l and k_r parameters, we run the robot in a straight path for short distance (5 meters) and

measure the travelled distance using measuring tape. By knowing number of pulses accumulated during this travelled distance, we can get the initial value of the two parameters. The final parameter to calibrate is the base width, which is done by running the robot one complete circle that ends at the same initial starting point. We initialize the robot heading angle to be 0° in Cartesian coordinates. The resultant orientation angle after returning to the starting point of the circular path will be 2π after one complete circle. By knowing both of the accumulated pulses from the front wheels, we can calculate the base width from equation (3.8). Now we can do fine tuning for the three parameters k_l , k_r and W by commanding the robot to move for larger distance ($20\ m$ or more) in straight and circular paths many times. We use the kinematic model to update (x, y, θ) based on odometric displacements. Each time we tune the three parameters to match real position until we get the best position results. In the case of straight line path the final heading angle should be zero as we start from zero head angle. So we modify both k_l and k_r to get 0° head angle. Table 3.2 shows the final values of calibration parameters.

Table 3.2: Calibration parameters

Parameter	**Value**
k_l	0.1178 cm/pulse count
k_r	0.1184 cm/pulse count
W	22.9 cm

3.6 Vision Cams Unit

The vision sensors are important for navigation and obstacle avoidance. We used two cameras mounted on the front of the vehicle facing forward. They give the robot its exact position at all times, allowing it to sense the location of objects and to track a predefine color. In our work, we use a simple color detection based on a stereo vision system to compute the distance from the robot to a target. The specifications of the cameras used are shown in Table 3.3.

Table 3.3: Camera specifications

Interface	IEEE-1394a (FireWire) 400 Mbps, 2 ports (6 pins)
Camera Type	IIDC-1394 Digital Camera, V1.04 Specification compliant
Sensor Type	Sony Wfine* 1/4" CCD Color, progressive, square pixels
Resolution	VGA 640 x 480
Optics	Built-in f 4.65 mm lens, anti-reflective coating
Power Supply	8 to 30 VDC, by 1394 bus or external jack input Consumption 1W max, 0.9 W typical, 0.4 W sleep mode
Dimensions ($W \times H \times D$)	$62 \times 62 \times 35$ mm

3.7 Compass Sensor

Electronic compass is an essential component of the solution to one of the long-standing robotics problems Where am I? The compass is needed because it can compensate for the foremost weakness of odometry. In an odometry based positioning method, any small momentary orientation error will cause a constantly growing lateral positioning error [5]. The advantage of using a compass rather than a gyro is that a compass gives the heading angle directly. The gyro requires integrating the angular velocity to get the heading angle. The compass will provide the heading

angle measurement update in the Kalman filter as we will explain later. We decided to use the 1655 analog compass sensor from Dinsmore Instrument Company. This sensor provides a ratiometric output on two channels. The outputs swing from approximately 3.2 V to 1.8 V in a sine/cosine fashion. It will return to the indicated direction from a 90° displacement in approximately 2.5 seconds with no overswing. Technical Specifications of 1655 analog compass sensor are listed in Table 3.4.

Table 3.4: Specifications of 1655 analog compass sensor from Dinsmore Sensors

Power	5-volts DC @ 19 mA. Since rise time is only 90 nanoseconds, input current may be pulsed to save power.
Outputs	Dual analog channels, 2.5 V ± 0.75 V swing (total voltage swing rail to rail, approximately 1.50 V), 4 mA DC signal. May feed direct to A-D front end of microprocessor.
Weight	2.25 grams
Size	12.7 mm diameter, 16 mm tall
Pins	3 pins on 2 sides on .050 centers
Temp	-40 to +85 degrees C

The compass has two sinusoidal analog outputs. One is a sine curve and the other is a cosine curve. Typical output curves look like the plot shown in Figure 3.14. Those two outputs can be decoded using a subroutine which computes the heading angle in radians relative to some directions.

3.8 GPS Unit

The GPS has become a common solution for outdoor navigation in large environments where there is no other reference available. We used the Magellan Meridian GPS unit with horizontal accuracy (RMS) <7 m 95% 2D and with WAAS <3m

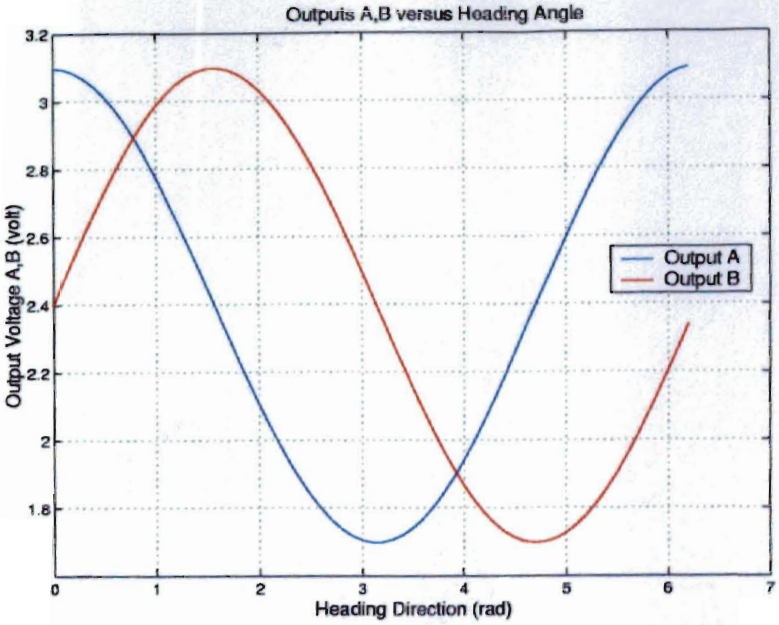


Figure 3.14: Typical analog compass outputs

95% 2D and vertical accuracy is 10 m (RMS). The Wide Area Augmentation System (WAAS) is a GPS-based navigation and landing system that provides precision guidance to aircraft at thousands of airports and airstrips where there is currently no precision landing capability. Systems such as WAAS are known as satellite-based augmentation systems (SBAS). WAAS is designed to improve the accuracy and ensure the integrity of information coming from GPS satellites [33]. A detailed technical specifications of this unit is listed in Table 3.5.

To decode GPS data, We used a commercial GPS Component for C/C++ from MarshallSoft [35]. The GPS unit is connected to a host computer through the serial port. The output of the GPS receiver is NMEA sentence. NMEA stands for National Marine Electronics Association. An NMEA sentence is a line of data containing ASCII text that defines position, velocity, time and other information computed by

Table 3.5: GPS technical specifications

Position Update Rate (per second)	1
Time to First Fix: Cold	<2
Time to First Fix: Warm	<1
Time to First Fix: Hot (seconds)	15
Maximum Velocity (mph) Maximum Velocity (km/h)	951 1530
Weight (gm)	227
Display Size Height (inches)	2.2
Display Size Height (mm)	55.9
Display Size Width (inches)	1.75
Antenna	Quadifiler Helix
Horizontal Accuracy (meters)	<7
Horizontal Accuracy (RMS)	95 % 2D
Horizontal Accuracy -RMS w/ WAAS (meters)	<3
Horizontal Accuracy (% RMS/WAAS)	95 % 2D
Vertical Accuracy (meters RMS)	10
Velocity (knots RMS)	0.1
Battery Type	AA
Battery Quantity	2
Battery Life (hours)	14
Receiver WAAS Enabled	Yes
Waterproof (IEC-529 IPX7 Standard)	Yes
Operating Temp Min (C)	-10
Operating Temp Max (C)	60

the GPS receiver. Data settings are generally 4800 baud with 8 bits of data, no parity, and no stop bit. The NMEA standard provides a large range of sentences, but many relate to non-GPS devices and some others are GPS related but rarely used. Most GPS receivers also have a binary mode but it is normally best to reserve the use of binary GPS protocols for applications that really require their use, such as those requiring position updates of greater than once per second. In our work we will be dealing with two common types of NMEA sentence which are the GPGGA and GPRMC. The GPGGA stands for Global Positioning System Fix Data. An example of the structure of the GPGGA sentence is shown below :

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

An explanation of this NMEA sentence is given in Table 3.6 [34]. The GPRMC NMEA sentence gives position, velocity, time and course data. It stands for the Recommended Minimum. An example of the structure of the GPRMC sentence is shown below :

$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A

An explanation of this GPRMC NMEA sentence is given in Table 3.7 [34].

3.9 Power System

An important problem to overcome was supplying different voltage levels to operate the additional hardware components. We have solved this problem with the addition of a Battery Booster 12 circuit. This circuit eliminates the need for a 9 V battery for the Mini SSC II serial servo controller board from Scott Edwards Elec-

Table 3.6: Structure of the GPGGA sentence

GGA	Global Positioning System Fix Data
123519	Fix taken at 12:35:19 UTC
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
1	Fix quality: 0 = invalid 1 = GPS fix (SPS) 2 = DGPS fix 3 = PPS fix 4 = Real Time Kinematic 5 = Float RTK 6 = estimated (dead reckoning) 7 = Manual input mode 8 = Simulation mode
08	Number of satellites being tracked
0.9	Horizontal dilution of position
545.4,M	Altitude, Meters, above mean sea level
46.9,M	Height of geoid (mean sea level) above WGS84 ellipsoid
(empty field)	time in seconds since last DGPS update
(empty field)	DGPS station ID number
*47	the checksum data, always begins with *

Table 3.7: Structure of the GPRMC sentence

RMC	Recommended Minimum sentence C
123519	Fix taken at 12:35:19 UTC
A	Status A=active or V=Void.
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
022.4	Speed over the ground in knots
084.4	Track angle in degrees True
230394	Date - 23rd of March 1994
003.1,W	Magnetic Variation
*6A	The checksum data, always begins with *

tronics. The IR sensors require a 5 volt DC supply. We used the regulated 5 volt from the connector board of the acquisition card that is supplied by the notebook battery. The second problem was supplying enough power to the vehicle, since it carries more weight than it was designed for and also now has high torque servos, sensors, speed controller, etc., which are extra loads on the power system. The two options to solve those problems are adding additional batteries or replacing the existing battery with a more powerful one. The problem with adding more batteries is that there is no convenient place on the chassis to store them. Also adding extra batteries will increase the load on the robot. Our possible solution is to have a single battery with large battery capacity (5000 mAh). Using a fully charged 7.2 V battery with 3000 mAh capacity, the average runtime is about 30 minutes.

3.10 On Board Computer

The onboard computer along with the multifunction acquisition card provides a computational power for sensory data processing. We decided to use a Sony VAIO SRX77P notbook. Using such a notebook saves the time of building a PC on the robot. The notebook is fast, lightweight, and has low power consumption. The processing unit used in other experimental testbeds vehicles are based on a microcontroller [20],[14], this might limit the capabilities of the robot. Using a notebook will provide a large disc space for writing code, a fast processing speed, a standard communication interfaces and a built in efficient battery. Two serial ports were needed to interface both the SSC and GPS receiver. An USB to serial port converter from Keyspan was used Because the notebook does not have a serial port. The

Keyspan USB 4-Port serial adapter allows four serial devices to be connected to a single USB port.

Table 3.8: Notebook specifications

Processor	Low Voltage Mobile Intel PentiumIII processor 800A MHz
L2 Cache Memory	512 KB (CPU Integrated)
Hard Disk Drive C/D Partition	20 GB 40% and 60% (approximation)
Standard RAM	128 MB SDRAM (Expandable to 256 MB) (PC100 unbuffered DIMM memory modules)
LCD Screen	10.4" XGA (1240 x 768)
Wireless LAN	Communication IEEE802.11b (IBSS Ad hoc mode support, DS-SS modulation) Max. 11 Mbps data transfer speed (approximation) Max. 100 meter communication distance(approximation) 2.4 GHz band frequency Wireless channels 1 to 11 64, 128 bit Network key length
Power Source	16V DC/AC 100-240V
Battery	Lithium-ion
Dimensions	10.2" (w) x 1.1" (h) x 7.7" (d) (259 mm x 27.8 mm x 194 mm)
Connection Capabilities	1 USB port Phone line (RJ-11) port Ethernet port i.LINK (IEEE 1394) port, 4-pin S400 style

In the next chapter, we will discuss the software architecture implemented in this thesis.

Chapter 4
Software Integration

This chapter presents the software development of our designed vehicle.

4.1 Software Architecture

The current software architecture is simple and flexible for any future update. The block diagram shown in Figure 4.1 gives a detailed structure of the control architecture implemented on the MARHES TXT robot. It consists of a hardware dependent software, logical sensors and controllers. This architecture allows the vehicle to exhibit intelligent autonomous behavior. We used an object-oriented C++ decomposition to provide abstractions for the components of the systems. Components are implemented using classes.

4.2 Real-Time Issues

Our vehicle control requires *real-time* performance. There are several definitions of real-time. The definition given in [31] is as follows "*A real-time system is one in which the correctness of the computations not only depends upon the logical correctness of the computation but also upon the time at which the result is produced. If the timing constraints of the system are not met, system failure is said to have occurred.*" A brief discussion of real-time issues like determinism and jitter is given in [18]. Real-time control applications perform a defined task periodically. The task is performed before the new processor period starts. Figure 4.2 shows processor activity during running time. The hard real-time performance is difficult to be achieved

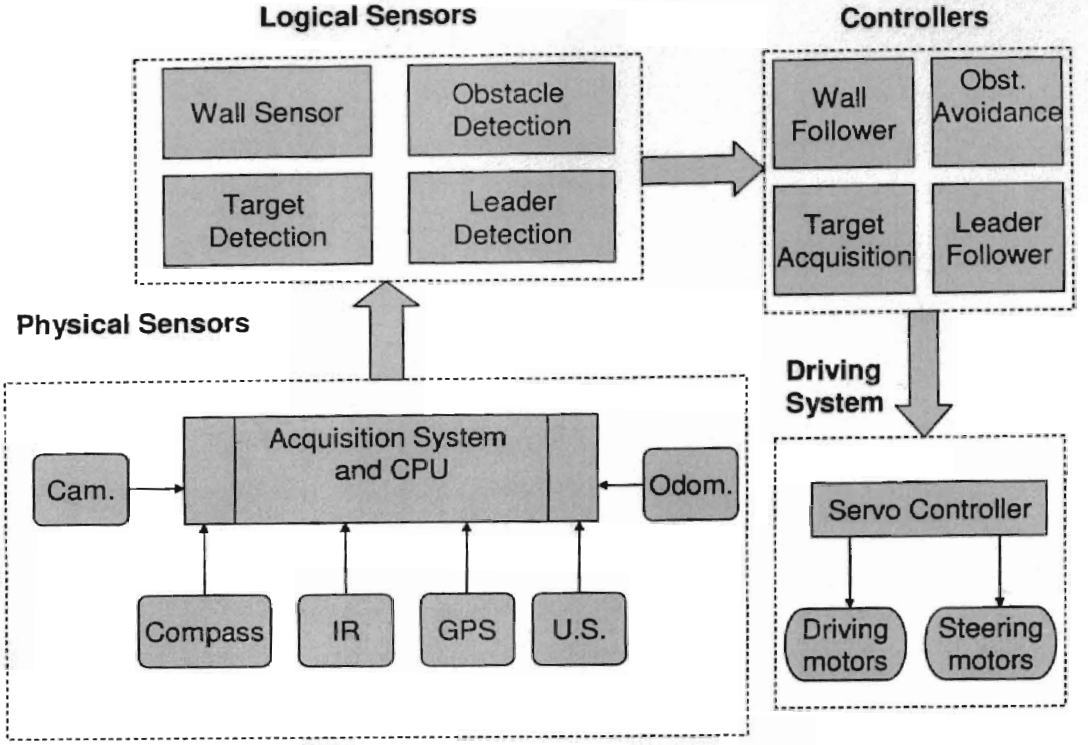


Figure 4.1: Control architecture scheme.

using gene

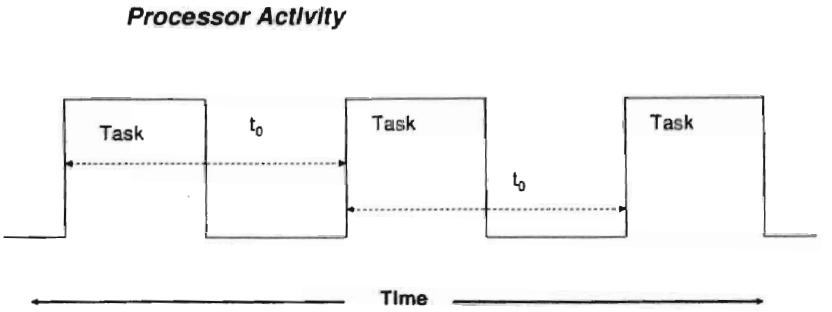


Figure 4.2: Real-time control.

4.2.1 Windows Family Operating Systems

The operating system used was Microsoft Windows XP. Some of the reasons for this choice include[36]

- The increasing power and declining price of Windows XP platforms.
- The many applications available on the platform.
- The variety of development tools available on the platform.
- The richness of the Microsoft win32 Application Programming Interface (API).
- The large number of developers, support personnel, and end users who are familiar with the system.

The Windows XP is not inherently a real-time operating system. One possible method to improve its performance is by using real time extensions. Venturcom is providing RTX extensions for real-time performance. RTX enables Windows XP, Windows 2000, or Windows NT to function as both a general-purpose operating system and a high-performance real-time operating [37]. In our operating system we did not experience with these extensions, however, they could be a solution to improve the performance.

4.3 Multi-Threading and Timing Functions

In most PC operating systems the central processing unit C.P.U is not able to run two pieces of code at exactly at the same time. The operating system solves this problem using time slicing. In time slicing, the microprocessor time is shared among pieces of code called threads. Each thread is given a portion of the microprocessor's time. Hence, the thread thinks it has the whole microprocessor time while it is actually shared among other threads. The details of how the operating system implements multi-threading are beyond the scope of this thesis.

A great advantage of using windows based operating system is the native support of multi-threading. Multi-threading is a way to let programs do more than one thing at a time. It is implemented within a single program running on a single system. It involves an operating system allowing programs to split tasks between multiple threads of execution. In our main program we need to generate timed functions calls at exact specific rates. We need to run our different controllers independently, each at a specific rate to get good results. One method to do this is to use interrupts but this method is not recommended under Windows XP. The other method is to use the multimedia library. This multimedia timer provides a high accuracy of timing schedule as we monitored the microprocessor performance. This accuracy is extremely lowered when using functions that are computationally intensive like image processing routines. The multimedia timer uses a separate thread to generate timed functions calls in the application. The handling of the thread is done internally to the multimedia function calls. The two main functions to use are:

- **timeSetEvent** : A function starts a specified timer event. The multimedia timer runs in its own thread. After the event is activated, it calls the specified callback function or sets or pulses the specified event object.

- **timeKillEvent**: A function cancels a specified timer event.

Each call to timeSetEvent for periodic timer events requires a corresponding call to the timeKillEvent function. The timer setup and shutdown must be done properly in the main program. The multimedia library, Winmm.lib, must be linked into the application and the header file, mmsystem.h, included in the source code. For more

information about multimedia library, see the MSDN library. Using multi-threading enables us to switch between behaviors through generating and terminating timed events.

Chapter 5
Tools for Vehicle Control

This chapter describes the tools we implemented for our designed robot.

5.1 Velocity Controlled Robot

Working with a kinematic model only is simple and makes the realization of the controller possible. The usual inputs for a 4 wheel mobile robot are the steering angle ϕ and speed. Dealing with these two inputs might increase the difficulty of controlling the robot since many control laws are expressed in terms of linear and angular velocity. One can think that a higher level controller (planner) generates the desired velocities and a lower level controller deals with the car dynamics (mass, inertia, etc.). As a result, we need a transformation that transforms the input commands of linear velocity (υ) and angular velocity (ω) to motor speed and steering angle ϕ. Most applications for controlling mobile robots require accurate matching between commanded velocity and the actual velocity. This property can not be achieved using an open loop controller. Hence there is a need for a closed loop controller to ensure the convergence of the actual velocity to the commanded velocity. PID controllers have been used effectively for many years in industry. Nise [23] has a good discussion of adjusting the PID gains, K_p, K_i, and K_d. Figure 5.1 shows a block diagram of the velocity controlled robot. The PID controller is designed to match the measured velocity with commanded one.

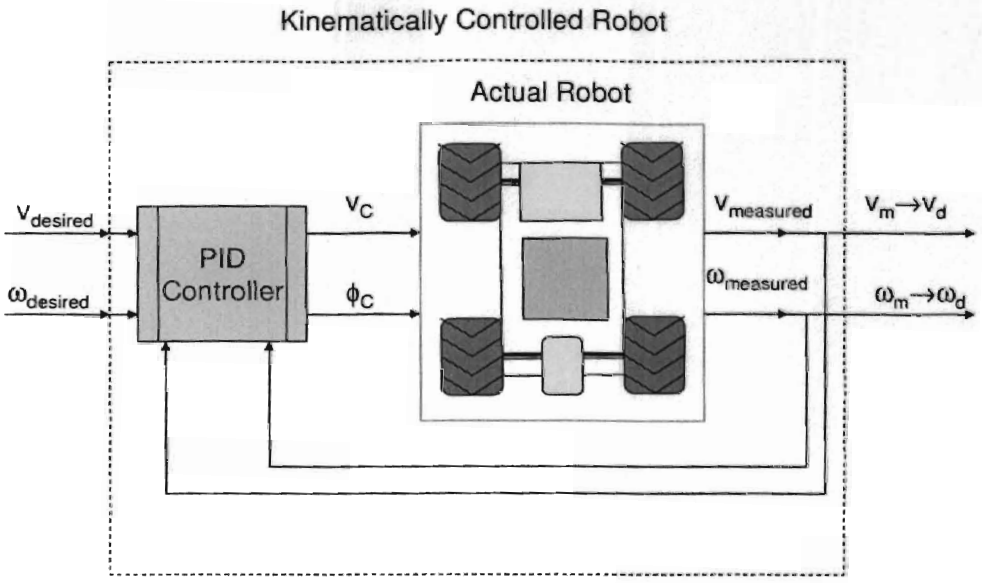


Figure 5.1: Velocity controlled robot.

5.1.1 PID Linear Speed Controller

Many control applications assumes a velocity controlled thus maintaining the actual speed as the desired one is important. For flat non-inclined surface the speed can remain constant, but if the surface becomes inclined this condition is violated. If the car encounters an incline, the power to the motor must be increased to maintain a constant speed due to the increased load. On the contrary, if the car is travelling downhill, the power to the motor must be decreased. Therefore, we implement a digital PID speed controller based on the installed optical quadrature encoder sensor. The speed is calculated by knowing the difference between two encoder reading's and the time elapsed between the two readings. In our subroutines, a sampling rate of *200 m.s* is implemented using timing function. After some experiments in measuring the speed using this sampling rate, it was found that a minimum speed of 0.10 m/s can be measured with acceptable accuracy and as the speed increases the accuracy

improves. The robot linear velocity is actuated by two DC motors. We approximate the system as a first order system. A first order system has the form

$$G(s) = \frac{K}{1 + \tau s} \tag{5.1}$$

where K is a constant and τ is the time constant of the system response which is the time it takes for the step response to rise 63% of its final value. The time constant depends upon the robot's environment. It was found experimentally to be around 1.5 seconds for even ground. A simple standard digital PID controller as shown in Figure 5.2, has the general form:

$$u = k_p e + k_i \int e dt + k_d \dot{e} \tag{5.2}$$

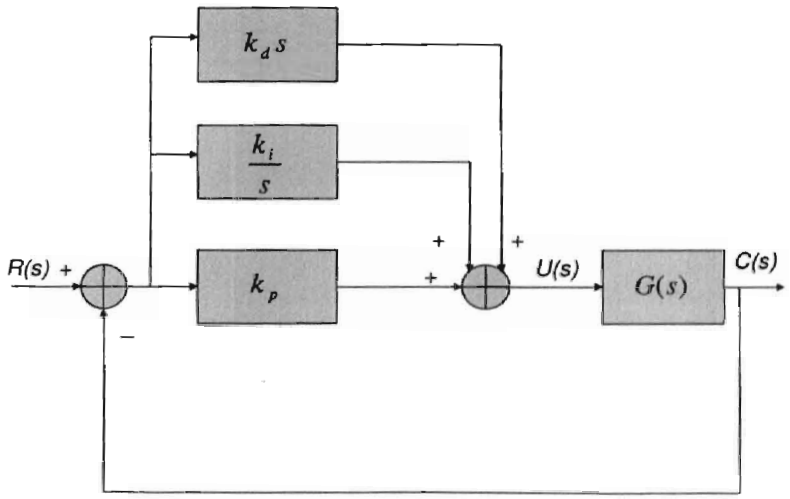


Figure 5.2: A PID controller

where u is the change in velocity, the variable e represents the tracking error (the difference between the desired input value and the actual output), k_p is the proportional gain, k_i is the integral gain and k_d is the derivative gain. The gain

parameters, k_p , k_i, and k_d are chosen to get the best system performance and stability.

The transfer function for a PID controller is

$$H(s) = k_p + \frac{k_i}{s} + k_d s = \frac{k_d s^2 + k_p s + k_i}{s} \tag{5.3}$$

The closed loop transfer function after adding the PID controller becomes

$$G_{cl}(s) = \frac{G(s)H(s)}{1 + G(s)H(s)} \tag{5.4}$$

and $G(s)H(s)$ is evaluated as

$$G(s)H(s) = \frac{K(k_d s^2 + k_p s + k_i)}{(1 + \tau s)s} \tag{5.5}$$

Now the closed loop transfer function has the form

$$G_{cl}(s) = \frac{K(k_d s^2 + k_p s + k_i)}{(\tau + K k_d)s^2 + (1 + K k_p)s + K k_i} \tag{5.6}$$

Using equation (5.6) and classical control theory, we found the initial values of the PID gains. The gains were tuned experimentally to obtain the desired overall response. In discrete time intervals $\int e$ is computed using $T_s \sum e$ and $\dot{e}$ is computed using $\frac{e_k - e_{k-1}}{T_s}$. So equation (5.2) becomes

$$u = k_p e_k + T_s k_i \Sigma e + \frac{k_d}{T_s}(e_k - e_{k-1}) \tag{5.7}$$

Where T_s is the sampling time. In our implementation, the sampling time was $200ms$. The sampling time can be lower but the multi-threading events did not occur correctly at lower sampling rates.

Smoothing Filter Using Holt Exponential Smoother

This simple and widely used recursive filter is obtained by iterating

$$y_t = \alpha x_t + (1 - \alpha) y_{t-1} \tag{5.8}$$

where $\alpha < 1$ is a tunable smoothing parameter. This filter can be used online to smooth the output commands and sensor's measurements. This allows us to smooth the output commands of digital PID speed controller. This low-pass filter gives most weight to most recent historical values and thus provides the basis for a sensible forecasting procedure when applied to trend, seasonal, and irregular components (Holt-Winters forecasting). The parameter α was tuned experimentally to get the best possible performance.

Figure 5.3 shows the performance of this controller. In this profile we have variable reference speeds in both directions. The angular velocity was set to zero without being controlled. The ground was even and has some inclination.

5.1.2 PID Angular Speed Controller

The basic idea behind this controller is, if the linear velocity is assumed to be constant then the steering angle will control the angular velocity. The relation between ω and v is

$$\omega = v/R \tag{5.9}$$

Where R is the signed distance from the ICC (instantaneous center of curvature) to the mid point between the two front wheels. We have shown experimentally in equation (2.13) that R is linearly related to steering angle ϕ. Thus we can use the same form of the previous PID controller equation (5.7) to regulate the steering

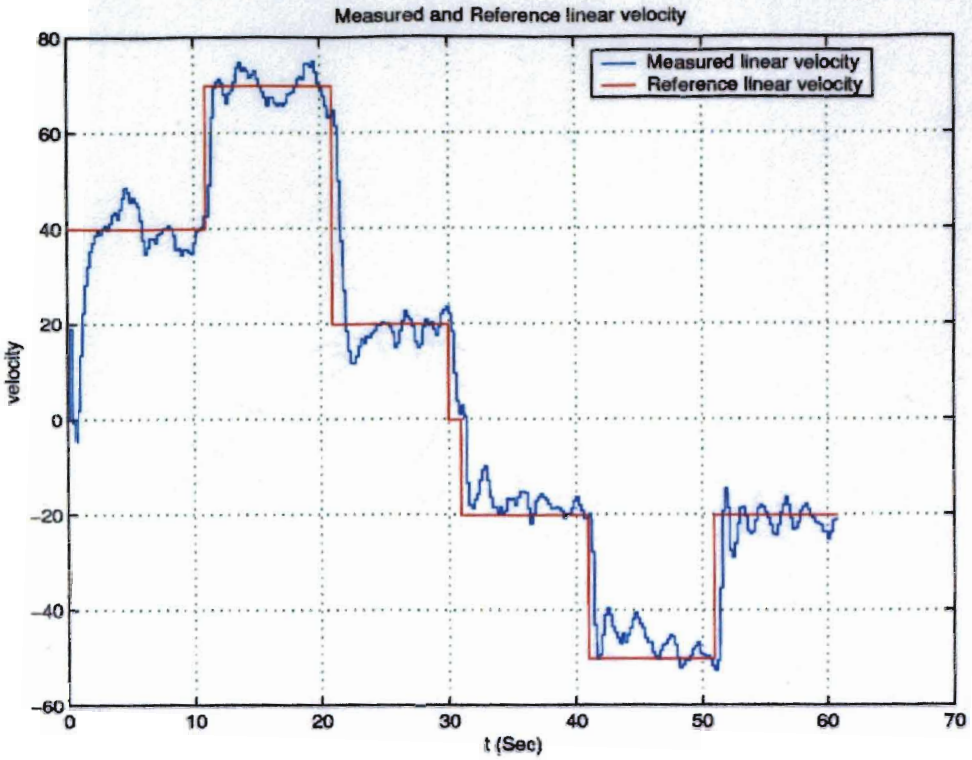


Figure 5.3: Measured and reference velocity

commands instead. One important parameter to consider is the sampling time. The sampling time is limited by the steering servo response time (i.e, the sampling time can not be smaller than steering servo response time). The response time of the servo is constant at no load but it varies according to the wheel-ground friction. The design of the PID controller will be for a nominal value of v because ω is coupled with v as in equation (2.19). The angular speed controller assumes v is constant. This problem is eliminated by having a dynamic PID gains. In other words, the PID gains will vary based on the value of desired linear velocity. We used the the same steps described in section 5.1.1 to find the gains. Figure 5.4 and 5.5 show the measured linear and angular velocities with a constant reference linear velocity of 0.5 m/s

and a constant reference angular velocity of 0.6 rad/s. The expected trajectory is a circular path with a constant radius as shown in Figure 5.6. The measured (x, y) positions are updated using odometric rotational and translational displacements. The robot follows a circular path after both linear and angular velocities has settled down. The measured radius, from Figure 5.6, was around $0.8m$, which is close to the theoretical radius computed from $R = \frac{v}{\omega} = \frac{0.5}{0.6} = 0.83\ m$. This is a good indication of correctness of odometric parameters calibration. The settling time and steady state errors were within acceptable limits.

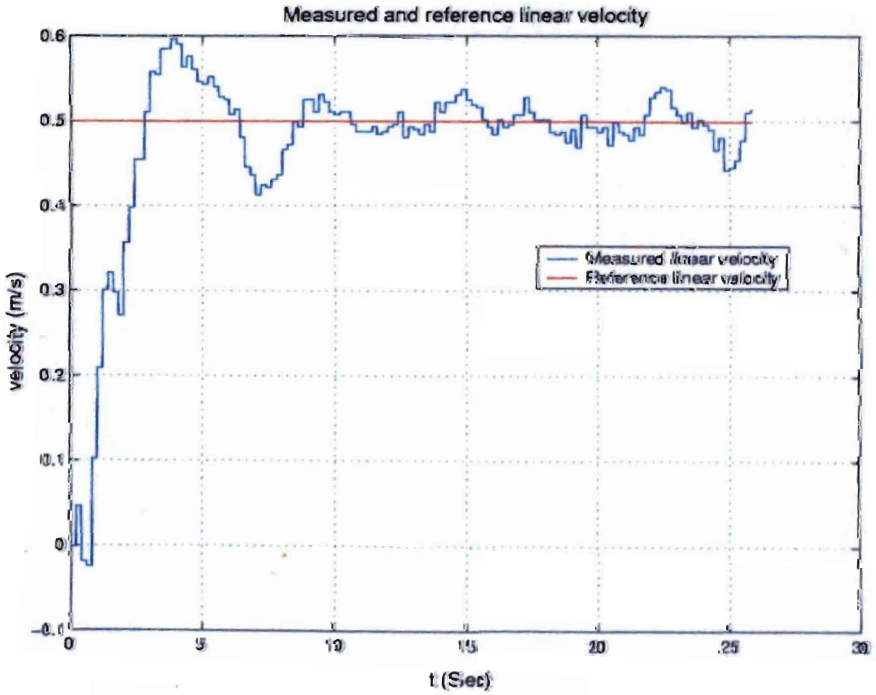


Figure 5.4: Measured and reference linear velocity

5.2 Wall Follower

Wall following is a useful and common technique for mobile robot navigation in known environments. The wall follower can be used for obstacle avoidance. The

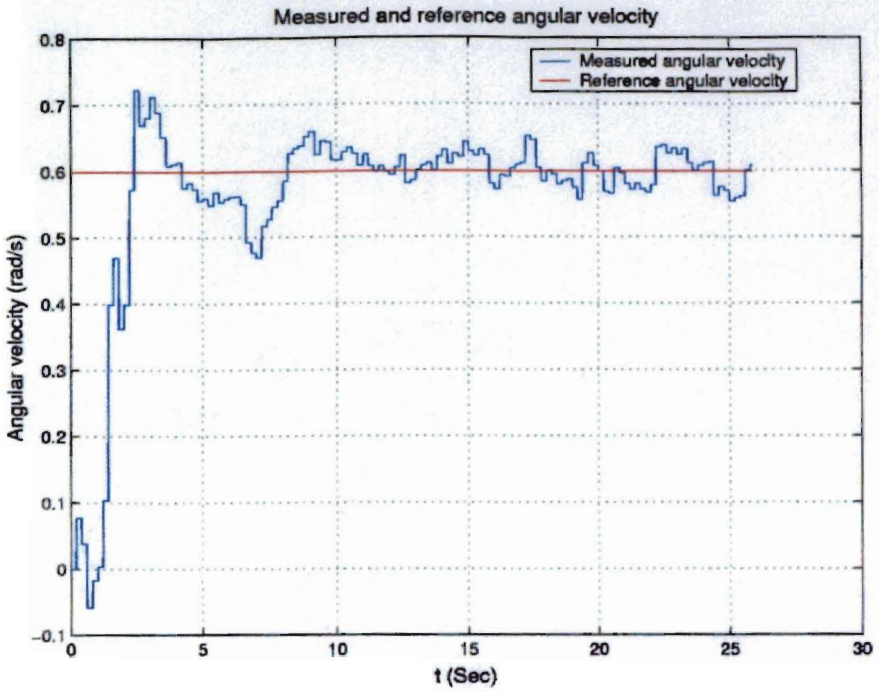


Figure 5.5: Measured and reference angular velocity

execution of a planed path can be prevented by unexpected obstacle. When the path can not be replanned, a simple strategy consists in following the contour of the obstacle by using distance sensors [17]. The common sensor used for wall following is an ultrasonic transducer [28],[3]. In our implementation , the IR sensors were employed for measuring the distance to wall. A simple PID controller was implemented to achieve wall following.

5.3 Obstacle avoidance

Obstacle avoidance is an important behavior in autonomous vehicle. There are many techniques to implements obstacle avoidance. One of the solutions is to use a potential field. In this approach, the obstacles are modeled as carrying electrical charges. The robot is modeled as a charged point having the same charge as obstacles

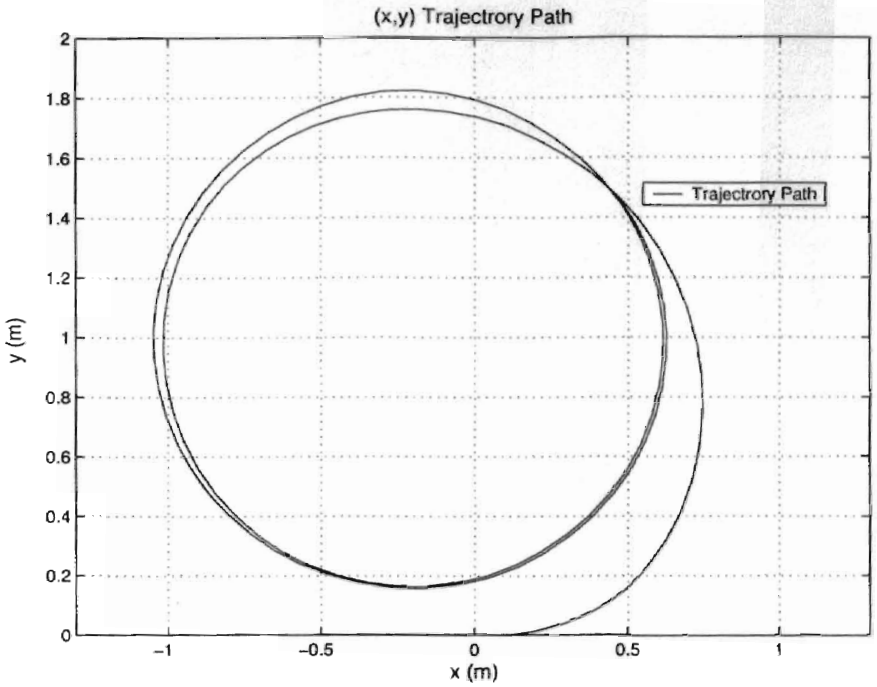


Figure 5.6: Trajectory path for controlled v and ω

in the environment. Collisions between the obstacles and the robot are avoided by the repulsive force between them. The repulsive force is the negative of the gradient of the potential field [9]. We implemented a simple obstacle avoidance by following the contour of the obstacle and commanding steering angle ϕ to move away from the obstacle using distance measurements from IR sensors.

Chapter 6
Navigation Control and Localization

This chapter describes a localization method for outdoor navigation using extended Kalman (EKF) filter which fuses odometry inputs, GPS and compass measurements.

6.1 Introduction

In this section, we discuss the effectiveness of our designed robot by testing localization algorithm using Kalman filter. Determining the robot location from physical sensors has been referred to as [6] " the most fundamental problem to providing the mobile robot with autonomous capabilities." Because of GPS position fixes are inaccurate and at times may not be available, other navigation aids are used in conjunction with GPS to enhance system performance. Dead reckoning sensors can not be used alone for indefinitely long periods, since errors grow without bound. They accurately measure changes in a vehicle's position over short time (can be used alone when GPS fixes become unavailable for short periods). The nature of the errors in GPS position fixes is somewhat different than that of the errors in dead reckoning. The errors appear in GPS position fixes and the errors in dead reckoning are complementary in nature. Proper fusion of the GPS position fixes with the dead reckoning sensor data can take advantage of the complementary errors producing positioning performance better than either type of data alone [1].

There are a few examples of autonomous outdoor navigation robots and most of them are costly research prototypes. Another example is given [24] but it uses a

Laser scanner which assumes prior knowledge about the environment, moreover, the set of sensors (Laser scanner LMS220 and inertial platform DMU-6X) used in that experiment are very expensive. In [25] and [24], the ATRV-Jr platform manufactured by iRobot used in the experiment is very expensive compared to the cost of our platform.

Outdoor environments are very difficult to work with because knowledge, and the terrain characteristics may not be available. The sonar, IR, or Laser scanner used for localization in indoors become useless in outdoors, though they might still be used as bumpers for obstacle avoidance. The next section explains how the localization problem is solved.

6.2 Odometric Kalman Filter

Kalman filters have been efficiently used for state estimation. A common filter type, is the odometric filter where readings from the odometry system on the robot are used together with the geometry of the robot movement as a model of the robot. In order to be able to use the Kalman filter the process model in equation (2.2) needs to be discretized first and then linearized. The discretized process model is explained in [27] and can be updated by the following equations

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} \delta d_k cos(\theta(k) + \frac{\delta\theta_k}{2}) \\ \delta d_k sin(\theta(k) + \frac{\delta\theta_k}{2}) \\ \delta\theta_k \end{bmatrix} \tag{6.1}$$

where

δd_k : The translational displacement at instant k

$\delta\theta_k$: The rotational displacement of the robot at instant k . The three states (x, y, θ)

constitute the process state vector. By considering δd_k and $\delta\theta_k$ as input $\mathbf{u}(k)$, $\mathbf{w}(k)$ as system noise, and $\boldsymbol{\gamma}(k)$ as input noise, we rewrite the nonlinear function in equation (6.1) as:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k); \mathbf{w}(k), \boldsymbol{\gamma}(k)) \tag{6.2}$$

The system and input noise are assumed to be Gaussian with zero mean and their covariance matrices $\mathbf{Q}(k)$ and $\boldsymbol{\Gamma}(k)$ respectively (i.e. $\mathbf{w}(k) \sim N(0, \mathbf{Q}(k))$,$\boldsymbol{\gamma}(k) \sim N(0, \boldsymbol{\Gamma}(k))$).

An extended Kalman filter can be designed using the system model in equation(6.1). Denoting $\theta(k)+\frac{\delta\theta_k}{2}$ as φ and linearizing the process model we get the following system $\mathbf{A}$ and input $\mathbf{G}$ matrices

$$A_{[i,j]}(k+1,k) = \frac{\partial f_i}{\partial x_{[j]}(k)}|_{\mathbf{x}(k)=\hat{\mathbf{x}}(k|k)} \tag{6.3}$$

$$\mathbf{A}(k+1,k) = \begin{bmatrix} 1 & 0 & -\delta d_k \sin(\varphi) \\ 0 & 1 & \delta d_k \cos(\varphi) \\ 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

$$G_{[i,j]}(k+1,k) = \frac{\partial f_i}{\partial u_{[j]}(k)}|_{\mathbf{x}(k)=\hat{\mathbf{x}}(k|k)} \tag{6.5}$$

$$\mathbf{G}(k+1,k) = \begin{bmatrix} \cos(\varphi) & -\frac{1}{2}\delta d_k \sin(\varphi) \\ \sin(\varphi) & \frac{1}{2}\delta d_k \cos(\varphi) \\ 0 & 1 \end{bmatrix} \tag{6.6}$$

The measurements $\mathbf{z}(k)$ is expressed as:

$$\mathbf{z}(k) = \mathbf{C}(k)\mathbf{x}(k) + \mathbf{v}(k) \tag{6.7}$$

Where $\mathbf{v}(k)$ is the measurement noise and it is assumed to be Gaussian with zero mean and covariance matrix $\mathbf{R}$ (i.e. $\mathbf{v}(k) \sim N(0, \mathbf{R}(k))$),

$\mathbf{C}(k+1)$ matrix is simply a 3×3 identity matrix because we can measure the three real state directly and we call it $\mathbf{C}$ for simplicity.

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.8}$$

The Kalaman filter is constructed from the general form described in [13], in the following equations

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{f}(\hat{\mathbf{x}}(k|k), \mathbf{u}(k); 0, 0) \tag{6.9}$$

$$\mathbf{P}(k+1|k) = \mathbf{A}(k+1,k)\mathbf{P}(k|k)\mathbf{A}(k+1,k)^T + \mathbf{G}(k+1,k)\mathbf{\Gamma}(k)\mathbf{G}(k+1,k)^T + \mathbf{Q}(k) \tag{6.10}$$

$$\mathbf{K}(k+1) = \mathbf{P}(k+1|k)\mathbf{C}^T[\mathbf{C}\mathbf{P}(k+1|k)\mathbf{C}^T + \mathbf{R}(k+1)]^{-1} \tag{6.11}$$

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{K}(k+1)\tilde{\mathbf{z}}(k+1|k) \tag{6.12}$$

$$\mathbf{P}(k+1|k+1) = [\mathbf{I} - \mathbf{K}(k+1)\mathbf{C}]\mathbf{P}(k+1|k) \tag{6.13}$$

where

$\hat{\mathbf{x}}(k+1|k)$ a predicted future state,

$\hat{\mathbf{x}}(k+1|k+1)$ time updated state,

$\tilde{\mathbf{z}}$ is the innovation term computed as $\tilde{\mathbf{z}}(k+1|k) = \mathbf{z}(k+1) - \mathbf{C}\hat{\mathbf{x}}(k+1|k)$,

$\mathbf{P}(k+1|k)$ is *a priori* estimate error covariance matrix,

$\mathbf{P}(k+1|k+1)$ is *a posteriori* estimate error covariance matrix.

$\mathbf{K}(k+1)$ is the Kalman gain matrix.

State error covariance matrix for the update estimate is calculated by the *Joseph form* [13] which is more computationally-stable version of equation (6.13)

$$\mathbf{P}(k+1|k+1) = [\mathbf{I}-\mathbf{K}(k+1)\mathbf{C}]\mathbf{P}(k+1|k)[\mathbf{I}-\mathbf{K}(k+1)\mathbf{C}]^T+\mathbf{K}(k+1)\mathbf{R}(k+1)\mathbf{K}(k+1)^T \tag{6.14}$$

6.2.1 Handling the System and Input Noise Covariance Matrix

The determination of the system noise covariance matrix $\mathbf{Q}(k)$ is difficult as the robot odometry may encounter nonsystematic errors which we can not handle. Having a very small system noise covariance $\mathbf{Q}(k)$ will slow the filter convergence to measured values specially when the measurements error covariance matrix is big. Thus, we inject enough uncertainty (by experimental tuning) in the system noise covariance. The system noise covariance matrix $\mathbf{Q}(k)$ is selected as

$$\mathbf{Q}(k) = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.05 \end{bmatrix} \tag{6.15}$$

The input noise covariance is selected as

$$\Gamma(k) = \begin{bmatrix} 0.05\delta d_k & 0 \\ 0 & 0.05\delta\theta_k \end{bmatrix} \tag{6.16}$$

6.2.2 Handling the Measurement Noise Covariance Matrix

The (x, y) states are measured using GPS unit. The GPGGA NMEA sentence contains the position in global coordinates. We used the MarshalSoft Toolkit package to covert from geodetic global coordinates (longitude and latitude) to local coordinates $(x(m), y(m))$ with respect to some reference point. The reference point we used is the initial position. This choice makes the robot initial position at the origin. Also we have chosen that East direction is our positive x axis and the North direction is the positive y axis. The accuracy of $\mathbf{R}$ determines the effectiveness of filtering process. Having the covariance matrix $\mathbf{R}$ greater than the real one slows up convergence of estimated state to measured values. On contrary, setting it smaller than the real value can speed up convergence of estimated state to measured value.

In reality $\mathbf{R}$ is not stationary and the errors in position are tightly related. We will assume it is stationary for simplicity. Having a constant $\mathbf{R}$ during filtering process may not result a good localization accuracy as it varies depending depending on the number of visible satellites and ionospheric conditions. To handel this we need to have a dynamic $\mathbf{R}$ that changes based on GPS data quality. The Dilution Of Precision DOP can tell about the quality of GPS data. DOP is updated based on the alignment, or geometry, of the group of satellites (constellation) from which signals are being received. We used this to modify the submatrix of $\mathbf{R}$. However, there is no guarantee that the estimated state will converge to the actual state when the measurements are not available for long time. For Heading angle measurement we used both GPS and compass to give heading angle. Heading angle from GPS is available at the GPRMC NMEA sentence. The GPS unit must be moving to get accurate reading from GPS heading angle. The accuracy depends upon the speed of the vehicle. It is very accurate at higher speeds and completely false when the robot does not move. Therefore, the heading angle measurements from GPS are disregarded at very low speeds and the variance of measured θ is modified based on the speed of the robot. Fortunately, the compass has opposite behavior. We will switch between GPS and compass readings, however, the GPS will be the dominant sensor for providing heading angle measurements because the robot is moving most of time. The heading submratix of $\mathbf{R}$ is updated by the variance of the sensor used for measuring heading angle.

6.2.3 Handling the Different Sampling Rates

The dead reckoning data is given at a higher frequency (every 200 $m.second$), than the information available from GPS (every 1 *second*), hence the time update and the measurement update part operate at different rates. We incorporate measurement when it is available. When the measurement is not available the predicted future state and the *a priori* state error covariance are used as *a posteriori* state estimate and state error covariance for the next iteration.

6.3 Navigation Controller

This section describes different controllers that use the estimated position given by EKF to reach a desired target position. This problem is solved using different controllers [2].

6.3.1 Target Acquisition Using Leader Following Approach

We use the leader following control algorithm to reach a desired destination (i.e., the leader robot is at reset). We use a velocity controlled vehicle that uses the kinematic model only. The convergence to commanded velocity can be guaranteed by our previous PID velocity controller. The formation control laws were derived using input-output feedback linearization [15]. The kinematics of both leader and follower are represented by the unicycle model given in equation (2.2). We consider two robots shown in Figure 6.1, where v_j and w_j are the linear and angular velocities of the follower at midpoint on the front axle.

We will describe the kinematic controller for leader following, presented in [8].

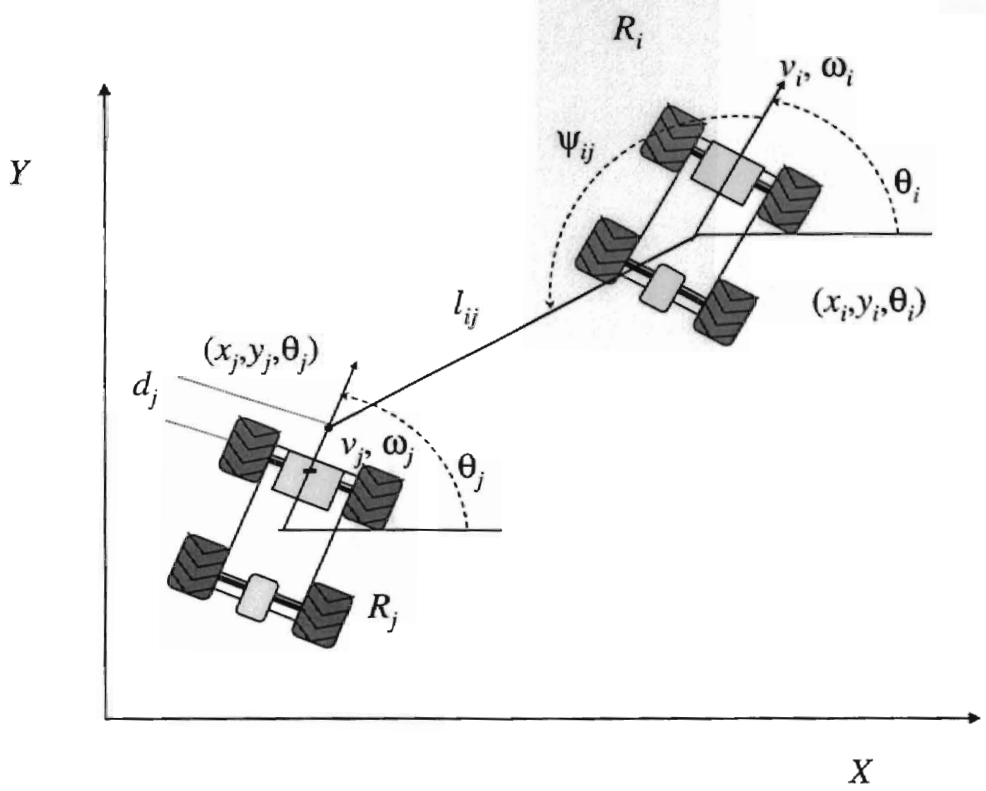


Figure 6.1: Two robots in a leader following configuration.

The leader will be fixed at the destination point with both linear and angular velocities are set to zero. By applying input/output linearization to generate a control law that gives an exponentially convergent solution in the variables l_{ij} and ψ_{ij} we get

$$v_j = k_1 \tilde{l_{ij}} \cos\gamma_{ij} - l_{ij} \sin\gamma_{ij}(k_2\tilde{\psi_{ij}} + \omega_i) + v_i \cos\beta_{ij} \tag{6.17}$$

$$\omega_j = \frac{1}{d}[k_1 \tilde{l_{ij}} \sin\gamma_{ij} + l_{ij} \cos\gamma_{ij}(k_2\tilde{\psi_{ij}} + \omega_i) + v_i \sin\beta_{ij}] \tag{6.18}$$

where

$\tilde{l_{ij}} = l_{ij}^d - l_{ij}$, $\tilde{\psi_{ij}} = \psi_{ij}^d - \psi_{ij}$, $\beta = \pi - \psi_{ij} - \theta_i$,

d is an offset distance to the midpoint of front axle P_j of the robot, and k_1 and k_2 are the user selected control gains. The closed-loop linearized system becomes

$$\dot{l_{ij}} = k_1\tilde{l_{ij}}, \qquad \dot{\psi_{ij}} = k_2\tilde{\psi_{ij}}, \qquad \dot{\beta} = \omega_i - \omega_j \tag{6.19}$$

Because we are using Cartesian coordinates, we can compute β using

$$\beta = atan2(\bar{y_j} - y_i, x_i - \bar{x_j}) \tag{6.20}$$

where

$\bar{x_j} = x_j + d\cos\theta, \qquad \bar{y_j} = y_j + d\sin\theta,$

By setting the angular and linear speed of the leader at zero, the control inputs become

$$v_j = k_1\tilde{l_{ij}}\cos\gamma_{ij} - l_{ij}\sin\gamma_{ij}k_2\tilde{\psi_{ij}} \tag{6.21}$$

$$\omega_j = \frac{1}{d}[k_1\tilde{l_{ij}}\sin\gamma_{ij} + l_{ij}\cos\gamma_{ij}k_2\tilde{\psi_{ij}}] \tag{6.22}$$

6.3.2 Experimental Results

In this experiment we initiated the robot at $(x(0), y(0), \theta(0)) = (0, 0, 0)$ and set the target position at $(x_t, y_t) = (20, 9)$. The position was estimated using the Kalman filter. We do not have a ground truth for real trajectory traversed by the robot, however, the final position reached by the robot was within $1m$ error from the position of the target. Figure 6.2 shows the trajectories computed from estimated state of Kalman filter, measured position from GPS, and odometric updated position. The odometric trajectory is updated using the kinematic model described in equation (6.1) without taking the advantage of GPS and compass. The odometric state is correct in the beginning , however, in the middle of the path the error accumulated to give completely wrong y position at the end. The GPS readings are correct and within the device accuracy. The estimated position agrees with the observation that

the robot follows a straight line toward the target.

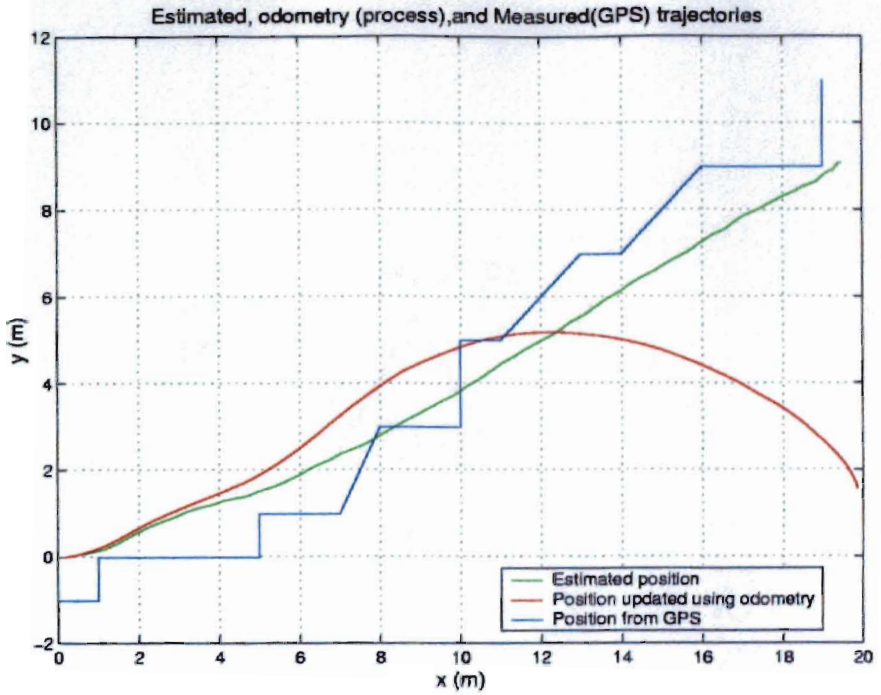


Figure 6.2: Estimated, measured and process trajectories.

To analyze this experiment in more details, we plotted the heading angle given by estimated state (Kalman filtered), measured heading angle (GPS and compass), and odometry (process only without incorporating any measurements) as shown in Figure 6.3. As we see from Figure 6.3, the orientation angle from both odometry and estimated state were almost the same in the beginning. The measured heading angle provides corrections for the filter once the linear speed is increased. The estimated heading angle takes the advantage of the correct measurement while odometric state heading angle accumulates errors to reach completely incorrect final value.

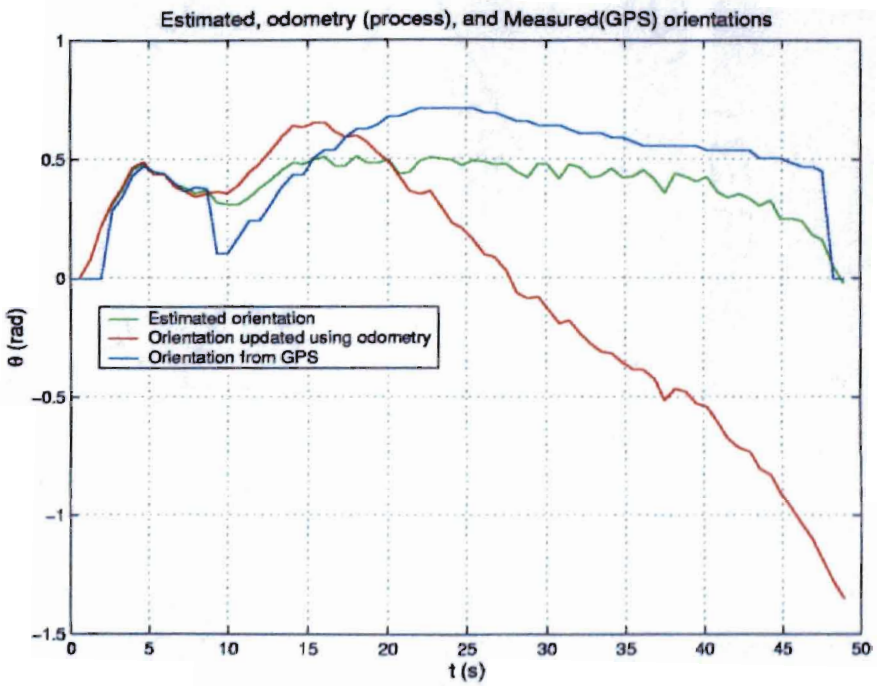


Figure 6.3: Estimated, measured and process orientations.

Chapter 7
Conclusion and Future Work

7.1 Concluding Remarks

This thesis has described the developing of a modular mobile testbed by modifying the standard chassis of a commercial inexpensive R/C truck. The vehicle is equipped with a suite of sensors to operate autonomously. The set of sensors includes IR, odometer, GPS, and vision system. The onboard notebook along with the multifunction acquisition card provided a computational power for sensory data processing.

The kinematic model for the vehicle was derived and it was shown that the unicycle model is a valid model for the vehicle under some conditions. Tools for robot control have been designed and implemented and their performance is promising. A control architecture utilizing object oriented multi-threading was used to achieve modularity.

Also we have demonstrated that good results of localization are obtainable by using only an inexpensive well calibrated dead reckoning sensors and an inexpensive commercial GPS unit. The target acquisition is achieved by applying Input/Output feedback linearized controller for leader following. Further experimental research can be carried out using the designed vehicle to verify theoretical results that have been validated using only simulation.

7.2 Future Work

There are some modifications need to be done to get a better performance of the robot.

7.2.1 Hardware Modifications

The following are some suggestions for modifications on hardware.

- Adding a gyro to measure angular velocity so we can avoid nonsystematic errors in odometry.
- Building power monitoring system to give the status of remaining power in batteries.

7.2.2 Software Modifications

Here are some suggestions for software improvement

- Devlop the communication network between vehicles. One possible solution is to use `.Net`.
- Improving real-time performance by adding real-time extensions.
- Implementing more functions that take the advantage of vision unit.

7.2.3 Localization Algorithm Improvement

Though the localization algorithm using Kalman filter works fine, the following suggestion can improve the filter performance

- Extending the state of the filter to include both angular and linear velocity.

- Improving measurement accuracy by using higher precision devices specially the GPS unit and compass precision.

- Extending the measurements vector of the filter to include angular velocity measured by a gyro.

Appendix A
Hardware Sources

Company	Tamiya America, Inc.
Product	RC car chasis
Address	2 Orion Aliso Viejo,CA 92656-4200
Telephone	1-800-TAMIYA-A
Web Site	http://www.tamiyausa.com

Company	Sony Electronics Inc.
Product	Notebook Sony VAIO SRX SRX99
Address	1 Sony Drive MD TA3-12 Park Ridge, New Jersey 07656
Telephone	(877) 865-SONY(7669)
Web Site	http://www.sonystyle.com

Company	Sharp Electronics Corporation [U.S.A.]
Product	IR Distance measuring sensor
Address	1300 Naperville Drive, Romeoville, IL 60446
Telephone	1-800-237-4277 / 1-800-BE SHARP
Web Site	http://www.sharp.co.jp

Company	National Instruments Corporation
Product	NI DAQCard-6024E for PCMCIA and CB-68LPR DAQ
Address	11500 N Mopac Expwy Austin, TX 78759-3504
Telephone	1-512-683-0100
Web Site	http://www.ni.com

Company	Agilent Technologies, Inc.
Product	Reflective optical surface mount encoders and Codewheels
Address	395 Page Mill Rd.P.O. Box 10395 Palo Alto, CA 94303
Telephone	1 650 752-5000
Web Site	http://wwwagilent.com/semiconductors

Company	Hitec RCD USA, Inc.
Product	Servos
Address	12115 Paine St. Poway CA, 92064
Telephone	1-858-748-6948
Web Site	http://www.hitecrcd.com

Company	Digi-Key
Product	Electronic components
Address	701 Brooks Avenue South Thief River Falls, MN 56701

Telephone	1-800-DIGI-KEY
Web Site	http://www.digikey.com

Company	Thales Navigation
Product	Magellan GPS
Address	471 El Camino Real Santa Clara, CA 95050-4300
Telephone	1-408-615-5100
Web Site	http://www.magellangps.com

Company	Scott Edwards Electronics Inc.
Product	Serial Servo Controllers (SSCs)
Address	1939 S. Frontage Rd. #F, Sierra Vista, AZ 85635
Telephone	1-520-459-4802
Web Site	http://www.seetron.com

Company	Tower Hobbies
Product	RC car upgrade components
Address	PO Box 9078 Champaign, IL 61826-9078
Telephone	1-800-637-6050
Web Site	www.towerhobbies.com

Company	Robson Company, Inc.
Product	1655 Analog Compass Sensor
Address	227 Hathaway St. E. Girard, PA 16417
Telephone	1-814-774-5914
Web Site	http://www.dinsmoresensors.com

Company	Novak Electronics, Inc.
Product	Electronic speed control
Address	18910 Teller Avenue Irvine, CA 92612
Telephone	1-949-833-8873
Web Site	http://www.teamnovak.com

References

1. E. Abbott, D. Powell, " Land-vehicle navigation using GPS." *Proc.IEEE* , vol. 87 Issue: 1 , Jan. 1999 pp. 145 - 162.

2. M. Aicardi, G. Casalino, A. Bicchi, A. Balestrino, "Closed loop steering of unicycle like vehicles via Lyapunov techniques", *IEEE Robotics & Automation Magazine*, vol. 2 Issue: 1 , March 1995, pp. 27 - 35.

3. A. Bemporad, M. Di Marco, A. Tesi, "Wall-following controllers for sonar-based mobile robots ", in *Proc. IEEE Conf. Decision and Control*, Dec. 1997 pp. 3063 - 3068.

4. R. Bishop, "Intelligent vehicle applications worldwide", *Intelligent Systems, IEEE*, , vol.15 Issue: 1 , Jan.-Feb. 2000, pp.78 - 81.

5. J. Borenstein, H. R. Everett, and L. Feng contributing authors: S.W. Lee and R. H. Byrne, *Where am I? sensors and methods for mobile robot positioning*, April 1996.

6. I.J. Cox, "An experiment in guidance and navigation of an autonomous robot vehicle", *IEEE Transactions on Robotics and Automation,*, vol. 7 Issue: 2 , April 1991, pp. 193 - 204.

7. A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision based formation control framework, " *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, Oct. 2002, pp. 813 - 825.

8. J. Desai, J.P. Ostrowski, and V. Kumar, " Controlling formations of multiple mobile robots", in *Proc. IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 1998, pp. 2864 - 2869.

9. G. Dudek, M. Jenkin, *Computational principles of mobile robotics*, Cambridge University press 2000.

10. G.D. Dunlap, H.H Shufeldt, "*Dutton's Navigation and piloting*", Naval institute press, pp.557 - 579.

11. I. Erkmen, A.M. Erkmen, F. Matsuno, R. Chatterjee, T. Kamegawa, "Snake robots to the rescue! ", *Robotics & Automation Magazine, IEEE*, vol. 9, Sept. 2002 pp. 17 - 25.

12. R. Fierro and E. Edwan, " The OSU multi-vehicle coordination testbed", T*he 2002 45th Midwest Symposium on Circuits and Systems* , vol. 3 , August 2002,pp. 41 - 44.

13. R. Grover, B. Patrick, Y. C. Hwang, *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions* , 3rd Ed., John Wiley Sons Inc., 1997.

14. R. D. Henry, "*Automotive ultrasonic headway control for a scaled robotic car*" , Thesis, Virginia Polytechnic Institute and State University, 2001.

15. A. Isidori, *Nonlinear Control Systems*, Springer-Verlag, London, 3rd edition, 1995.

16. J. Laumond, "Robot Motion Planning and control", Springer, 1998.

17. J.C. Latombe, "Robot motion Planning", KAP, Boston, 1991.

18. "*Measurment and Automation CATALOG 2003*", NATIONAL INSTURMENTS, pp.772 - 773.

19. P.J. McKerrow, D. Ratner , "Calibrating a 4-wheel mobile robot ", in *Proc. IEEE/RSJ International Conference on Intelligent Robots and System*, vol.1 , Oct. 2002 pp. 859 - 864.

20. P. Mellodge, "*Feedback control for a path following robotic car*", Thesis, Virginia Polytechnic Institute and State University, 2002.

21. N. Miyake, T. Aono, K. Fujii, Y. Matsuda , S. Hatsumoto, "Position estimation and path control of an autonomous land vehicle", in *Proc. IEEE/RSJ, International Conference on Intelligent Robots and Systems*, vol.2 , Sept. 1997, pp. 690 - 696.

22. S. Murata, T. Hirose, "Onboard location system using real-time image processing for self navigating vehicle", *IEEE Transactions on Industrial Eelectronics*, vol. 40 Issue: 1 , February 1993, pp. 145 - 154.

23. N. S. Nise, *Control Systems Engineering* , 2nd Ed., Addison-Wesely, NY, 1995.

24. S. Panzieri, F. Pascucci, G. Ulivi, " An outdoor navigation system using GPS and inertial platform", *IEEE/ASME Transactions on Mechatronics*, vol.7 Issue: 2 , Jun 2002, pp. 134 - 142.

25. R. Thrapp, C. Westbrook, D. Subramanian, "Robust localization algorithms for an autonomous campus tour guide", in *Proc. IEEE International Conference on Robotics and Automation*, vol. 2 , 2001 pp. 2065 - 2071.

26. S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy , J. Schulte, D. Schulz,"MINERVA: a second-generation museum tour-guide robot", in *Proc. IEEE International Conference on Robotics and Automation* , vol.3 , May 1999 pp. 1999 - 2005.

27. C.M. Wang, "Location estimation and uncertainty analysis for mobile robots", in *Proc. IEEE International Conference on Robotics and Automation*, Apr 1988 pp. 1231 - 1235.

28. T. Yata, L. Kleeman, S. Yuta, "Wall following using angle information measured by a single ultrasonic transducer", in *Proc. IEEE International Conference on Robot Automatation*, vol.2, May 1998 pp. 1590 - 1596.

29. http://www.evolution.com

30. http://www.us.aibo.com

31. http://www.faqs.org/faqs/realtime-computing/faq

32. http://www.irobot.com

33. http://gps.faa.gov/Programs/WAAS/waas.htm

34. http://www.gpsinformation.org/dale/nmea.htm

35. MarshallSoft Computing, http://www.marshallsoft.com/mgc4c.htm

36. MSDN Library, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnxpembed/html/hardrealtime.asp

37. Venturcom, Inc. http://www.vci.com/

2.

VITA

EZZALDEEN EDWAN

Candidate for the degree of

Master of Science

Thesis: DESIGN OF A MODULAR AUTONOMOUS ROBOT VEHICLE

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Rafah, Palestine, On November 17, 1973.

Education: Received Bachelor of Science degree in Electrical engineering from Birzeit University, Birzeit, Palestine in March 1997. Completed the requirements for the Master of Science degree with a major in Electrical Engineering at Oklahoma State University in (August, 2003).

Experience: Employed as lecturer engineer by Palestine Technical College, 1997 to 2001.

Professional Memberships: Institute of Electrical and Electronic Engineers, Phi Kappa Phi.