9-9-2007

Design and implementation of test-bed for path planning and formation control of cooperative robotic agents

Mike Majedi

Design and implementation of test-bed for path planning and formation control of cooperative robotic agents

by

Mike Majedi

B.A.S., ITT, New Mexico, 1998

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2007

Dedication

This is dedicated to my loving wife Xiomara, my three wonderful children Yasamin, Abraham, Esmael, my mother, my father, my three brothers Mohammad, Mehdi, Mohsen, and my two sisters Mahsheed, Mahnoosh, my in-laws and my dog princess.

Acknowledgments

I would first like to thank my thesis advisor Professor Chaouki T. Abdallah at University of New Mexico. I would not be here without his support, guidance and encouragement. His office was always open whenever I ran into a trouble spot or had a question about my research or writing. I am also extremely grateful to the members of my committee, Dr. Nadar Vadiee with all his support who has helped me not just with valuable suggestions and constructive comments but also with other supports like parts, equipment, guidance and encouragement. I like to thank Professor Herbert G. Tanner for his valuable research, suggestions and constructive comments. I look forward to work with him on the future robotic projects. Thanks also to my family, my loving wife for supporting me in my educational pursuits, to my in-laws for their prayer, and all my friends for their encouragement. I am particularly indebted to two guys , Pavlo, and Behshad, who assisted me in my research and set-ups; without their help this would be a lot more difficult, and also whose creative endeavors allowed me to maintain my (in)sanity throughout the writing of this thesis.

Design and implementation of test-bed for path planning and formation control of cooperative robotic agents

by

Mike Majedi

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2007

Design and implementation of test-bed for path planning and formation control of cooperative robotic agents

by

Mike Majedi

B.A.S., ITT, New Mexico, 1998

M.S., Electrical Engineering, University of New Mexico, 2007

Abstract

Robots are primary candidates to perform dangerous but controlled missions. They carry cameras, surveillance instruments, and sensors to collect information and relay their findings to their human operators. Man is continuously challenged to seek uncharted territories such as outer space and the ocean's depths. Robot data collection allows for an operator to collect data in environments where it may be unsafe for humans to do so. It also allows for humans to perform inconvenient and tedious tasks such as waiting, and collecting information over long periods of time. In this thesis, we propose to design and implement a test-bed platform, which may be used to control multiple Robots. These robotic platforms are essentially mobile wireless programmable robots with the processing power of a laptop computer. The robots consist of two mechanically motorized differential drive kinematics systems. We then discuss the capabilities of our platform with basic behaviors such as: path-to-goal, avoid-obstacle, maintain relative distance, maintain relative angle (formation), and maintain general speed to form a guidance algorithm. Furthermore we use the kinematic model of a two differential steering systems to derive the inverse kinematic equations, and generate any type of trajectory within the physical constrains of the system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Control theory is the corner stone of most industrial processes, and is widely used in vehicles [17]. The control capabilities become increasingly important as the complexity and performance of cars, planes, and other vehicles increase. The development of such systems often requires knowledge from a wide range of technical disciplines, and control theory is an important discipline.

As scientists request more data and more accurate readings from various robots, the need for more sophisticated robots with afforded autonomy and enough computation power increases. The current need for autonomous robots capable of functioning in both self and collective modes is greater than what the world of science and manufacturing can supply [30]. Robots have for many years been in use in the manufacturing industry, but are now being introduced in transportation and in various hostile environments [18].

Demand for autonomous robots [21] will surely continue to increase and their design and production are by no means easy. Many sophisticated tasks require multiples robots, working together to carry out a single mission such as the mapping of an inhospitable terrain, or acting as patient sitters, security guards, rescuers and fire fighters, helping in surgery and rehabilitation, in mining as well as in stores and museums.

Such robots must function in cooperation with each other [15,20], as they are concurrently busy and focusing on their individual tasks. As the numbers of specialized robots increase, their individual tasks may be simplified, their architecture better configured, and they may offer optimum utility factor. While human control via well rehearsed instructions and various planned exercises may sometimes be required, but such human intervention may be difficult, or impossible [8]. A robot may find itself where there is a communication breach due to (caves or large canyons).Bi-directional communication must help autonomy of individual robots as a well-coordinated subtask within the collective's mission. There is a great need for robots that are capable of autonomous or semi-autonomous behavior.

Robotic applications require more accuracy, robustness, and reliability as more autonomy is required. Working prototypes of unmanned ground vehicles (UGV) have been developed by numerous research groups ([5, 6]). Application to autonomous guided vehicles (AGV) [34] and wheeled mobile robots (WMR) [34] have focused on navifation.

Our main objective in this thesis is to evaluate mobile robot bases and to build a kinematics model of WMR suitable for mobile manipulation tasks and research.

1.2 Thesis Summary

In the second chapter we provide mathematical models of the mobile robots, some control analysis and include the results of numerical simulations, validating our algorithm, and theoretical findings.

In the third chapter, we describe the construction of a wheeled mobile robot and the general system design. We focus on the , position sensing odometry, the desired wheel performance, the choice of the discrete components, and design of a PID controller using the system identification procedure.

In the fourth chapter of this thesis, we derive the kinematic model of the robot that is based on the differential steering wheel system assuming no slipping condition. This simple model and reliable wheel-system is commonly used in smaller robots. This model is used to predict how a robot should respond to changes in its wheel speed and what path it will follow under various conditions. We develop the inverse kinematic formula that can execute an arbitrary trajectory.

In the fifth chapter we summarize the results of this thesis and highlight its contributions. We also include a discussion of the open issues and future work and improvements.

1.3 Contributions

We summarize the contribution of the work described in this thesis, as follows:

Theoretical expansion

- We implement different scenarios in our simulations and study the individual components of the control inputs. With the insight gained through this experience, we are able to prioritize our goal of flock formation by adjusting our numerical coefficients chosen to prevent the saturation in practical implementations of our flock.

- We develop modular algorithms including Matlab codes and Labview codes (Appendix A), which are directly implementable and expandable for any number of agents as needed.

Improvements on the previous design of a single agent and the Testbed:

- The previous platform was designed by the Networked Controlled System Laboratory at the University of New Mexico. In an effort to improve the existing design we develop a new generation of robots using new algorithms and techniques. The new generation provides modular, lightweight (portable), less expensive, and easily reproducible platform.
- We build a platform such that its hardware is totally independent of the processor (microcontroller, and PC), and of any programming environment. The user could easily use any programming language such as: Labview, C, C++, Java, Matlab or else, as long as the programming language is capable of establishing serial communications.
- This platform allows the researchers to interface with Labview SUBVIs or windows dynamic link libraries (DLL) and the software needed to control the robot. The control is achievable from a friendly and easy-to-use virtual instruments environments and allow the user to command the wheel movement, extract data, and establish communications between agents. Autonomous navigation capabilities of the robot include analog based obstacle avoidance, visual surveying to user-designated goals with a webcam, and modular path planning capabilities.

- Two of the robotic platforms were used in the InterMesh [4] protocols desines, which was implemented by the Networking Group at the ECE department at the University of New Mexico. The robots were able to seamlessly communicate over native Mesh networks and to implement a simple control algorithm. The demo to this experiment may be viewed hdl.handle.net/2118/tna

Chapter 2

Modeling and Simulations

2.1 Introduction

There are many approaches to the multiple agent control problem. One approach involves Internet-Like-Protocols (ILPs) to coordinate a formation of several agents [13]. Another approach uses the concept of potential function for the coordination [9]. This approach relies on specific properties of potential function, which determines the behavior of each of the agents around its fellow agents and obstacles. A descriptive discussion of the potential function approach is presented in this chapter. Different components of the control inputs and their roles in maintaining formation, obstacle avoidance, maintaining specific velocity mode, and virtual leader navigation are discussed and explained.

In this chapter we provide a detailed description of the robotic system and the simulations that were implemented on such a system.

2.2 Problem formulation

Consider n agents with position coordinates $\vec{r}_j(t)$ and velocities $\vec{v}_j(t)$. According to [10] behavior of the agents is governed by the following equations

$$\begin{cases} \dot{\vec{r}}_j = \vec{v}_j \\ \dot{\vec{v}}_j = \vec{u}_j \end{cases}, j = 1 \dots n, \tag{2.1}$$

where $\vec{u}_j$ is the control signal of agent R_j. Throughout our discussion, the standard notation for derivatives with respect to time is used:

$$\dot{\vec{r}} = \frac{d\vec{r}}{dt}. \tag{2.2}$$

The initial configuration of the flock is defined by

$$\begin{cases} \vec{r}_j(0) = \vec{r}_{0,j} \\ \vec{v}_j(0) = \vec{v}_{0,j} \end{cases}, j = 1 \dots n, \tag{2.3}$$

where $\vec{r}_{0,j}$ describes the initial location of agent R_j and $\vec{v}_{0,j}$ is its velocity at moment $t = 0$ (see Figure 2.1).

Our primary goal is to design control signals for each of the robots to incorporate the following behavioral patterns:

1. maintaining a predefined formation
2. maintaining a general velocity
3. following virtual leader
4. avoiding obstacles

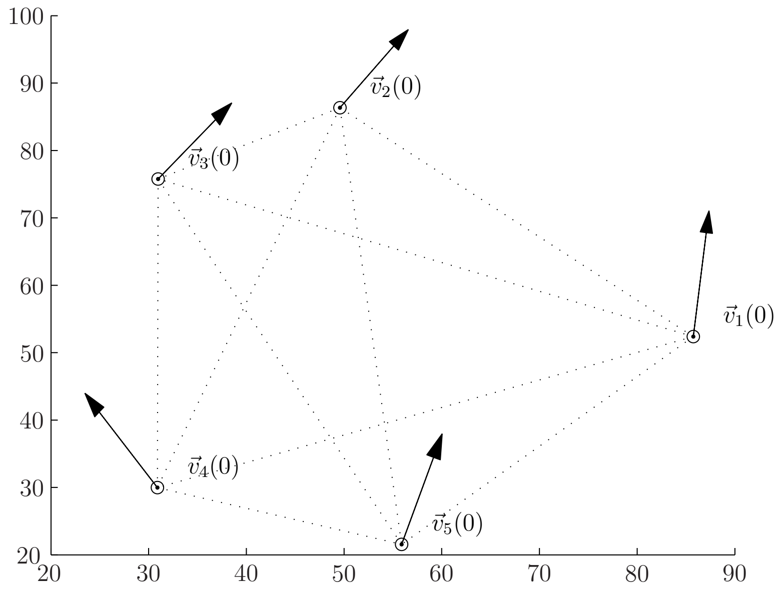


Figure 2.1: Initial Configuration of the Flock

This can be accomplished by closing the loop with the following control signal [11]

$$\vec{u}_j = k_1\vec{u}_j^1 + k_2\vec{u}_j^2 + k_3\vec{u}_j^3 + k_4\vec{u}_j^4, \tag{2.4}$$

where each of the individual input components $\vec{u}_j^i$, $1 \leq i \leq 4$ will be described in detail next.

2.3 Maintaining a Formation

The control input $\vec{u}_j^1$ is responsible for maintaining a formation. At the initial moment $t = 0$, the agents are not required to be in any specific configuration. As time passes, however, the agents must arrange in a formation described by the graph defined by the following matrix D, which we call a distance matrix:

$$D = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{n1} \\ d_{21} & d_{22} & \cdots & d_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n_1} & dn_2 & \cdots & d_{nn} \end{pmatrix} \tag{2.5}$$

where each of the components d_{ij} represents a desired distance between agents i and j of the flock:

$$d_{ij} = \|\vec{r}_i - \vec{r}_j\|, \tag{2.6}$$

where $\|\cdot\|$ is the standard two-dimensional Euclidian norm defined as follows:

$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} \right\| = \sqrt{x^2 + y^2}. \tag{2.7}$$

We note that matrix D must be symmetric since $d_{ij} = d_{ji}$. It is also obvious that the diagonal elements of D are zeros but these values are never used in the algorithms and therefore can be defined as arbitrary non-zero numbers to prevent possible division by zero.

It is important to note that any arbitrarily generated symmetric matrix D may not have a corresponding topology of the flock. For example it is clear that it is impossible to arrange four agents in the plane so that the distance between each of them is 1. Nonetheless the algorithms used in this thesis will work for any symmetric matrix D with nonnegative elements. The result will be the "closest" to D configuration of the agents.

In order to direct the agents to maintain a certain formation, we use the properties of the so-called potential function proposed in [10], and [11]

$$f(s) = \frac{1}{s} + \log s, \;\; s > 0. \tag{2.8}$$

Our algorithm will try to bring the overall system to the state where the potential function is minimized. In a certain sense the value of potential function is a penalty the system pays for remaining in a state different from the defined one. By adjusting the potential function so that its minimum occurs at the configuration of the flock that is defined by a defined D, we drive the state of the system into a desired configuration. The choice of the function is influenced by its useful properties.

Let us take a closer look at the behavior of two neighboring agents that are trying to maintain a predefined distance from each other, while maintaining a safe distance from an obstacle. We assume that after a period of time the flock has converged to a steady formation and is moving in a certain direction. If the flock encounters an obstacle in its path, it may become necessary for it to temporarily break up its formation in order to avoid a possible collision in between one agent and the obstacle. It is understood that while breaking up the formation it is safer for the agents to distance themselves from one another rather than to get closer. This is due to the fact that by getting closer, the danger of inter-collision increases. Thus, at that point, the highest priority for the agents is to not get close to each other.

Let the independent variable s represent the distance between two agents. We can then see that

$$\lim_{s \to 0} f(s) = \lim_{s \to 0} f(s) = +\infty \tag{2.9}$$

i.e as the agents get closer or too far apart, the potential function increases. However as $s \to 0$ the increase is much more rapid than the one when $s \to \infty$. This can be explained by the fact that $f(s) = O(s^{-1})$ around 0, which has a faster rate of divergence, than $f(s) = O(\log s)$ as $s \to \infty$. Therefore, by minimizing f while at the same time breaking up the formation, the system will choose to increase s (distance) to avoid the much higher penalization for decreasing s (Fig. 2.2).

It is easy to see that the minimum of the potential function is achieved at $s^* = 1$

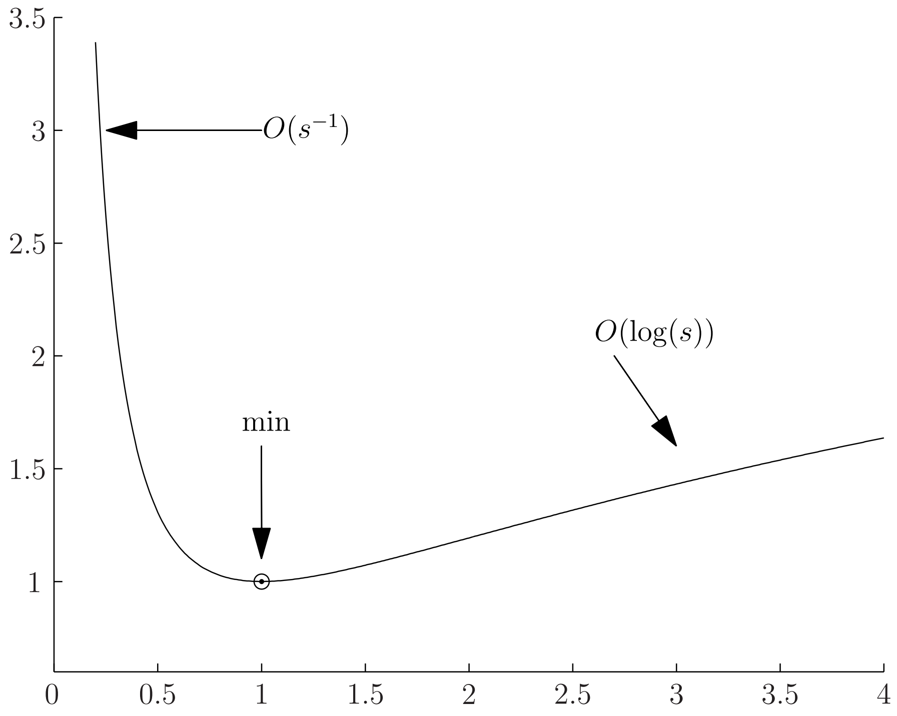


Figure 2.2: A Potential Function for a maintaining formation.

and is equal $f^* = 1$. Since we need to make sure that the minimum would be attained at $s^* = d_{ij}$ as a predefined distance between any two agents, the potential function may be easily modified to satisfy this requirement. By taking

$$F(s) = f\left(\frac{s^2}{d_{ij}^2}\right) \tag{2.10}$$

as a new potential function, we guarantee that the minimum is attained at $s^* = d_{ij}$ and, therefore, by driving the system to the minimal potential state, the control input will maintain the predefined formation.

Note that we also to modified the potential function by squaring its argument to smooth it out to provide differentiability. Later we will see that the gradient of F will have to be evaluated. Since $f(\vec{x}) = \|\vec{x}\|$ is not differentiable while $f(\vec{x}) = \|\vec{x}\|^2$ is, and since squaring does not radically change the behavior of the function we choose to work with the smooth version of potential.

In order to take into consideration the fact that there are more than two agents in the system, we introduce the total potential between agent R_i and the rest of the flock, as the sum of potentials between R_i and any other agent,

$$P_i(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_n) = \sum_{i \neq j} F\left(\|\vec{r}_i - \vec{r}_j\|\right) = \sum_{i \neq j} f\left(\frac{\|\vec{r}_i - \vec{r}_j\|^2}{d_{ij}^2}\right) = \sum_{i \neq j} f\left(\frac{\|\vec{r}_{ij}\|^2}{d_{ij}^2}\right) \tag{2.11}$$

Here for simplicity we adopt the notation

$$\vec{r}_i - \vec{r}_j = \vec{r}_{ij}. \tag{2.12}$$

Now we recall that the gradient of a function of many variables indicates the direction in which the function increases in the most rapid rate [31]. Therefore, in order to calculate the direction of the steepest descent, the antigradient must be calculated. Thus, in order to minimize the total potential and therefore for the system to reach its desired formation, the input $\vec{u}_i^1$ must be chosen to be the opposite of the gradient of the potential function, i.e

$$\vec{u}_i^1 = -\vec{\nabla}_{\vec{r}_i} P_i = -\sum_{i \neq j} \vec{\nabla}_{\vec{r}_i} f\left(\frac{\|\vec{r}_{ij}\|^2}{d_{ij}^2}\right) \tag{2.13}$$

2.4 Maintaining Velocity

The central input $\vec{u}_j^2$ in equation (2.4) is used to maintain the flock's formation. In order to achieve this goal we require that all agents move with the same predefined speed and follow the same direction. To maintain this velocity mode of the flock, the input to each of the agents must be corrected by the quantity that reflects the amount of deviation of the velocity of the agent from the rest of the flock, i.e

$$\vec{u}_j^2 = -k_2 \sum_{i \neq j} (\vec{v}_i - \vec{v}_j). \tag{2.14}$$

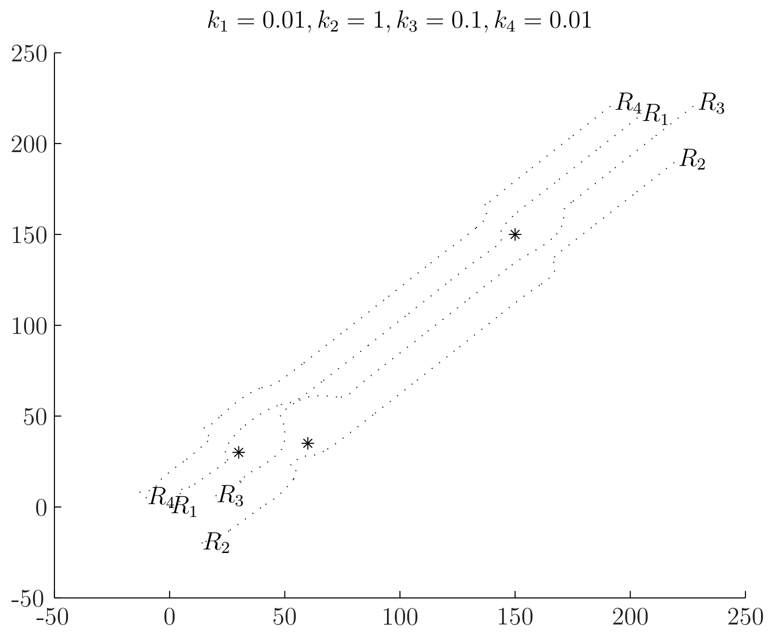


Figure 2.3: Trajectory of a flock of four agents. (Units of coordinates may be interpreted as meters.)

It is shown in [12] that such an input provides convergence of the velocity of each of the agents to the average of the velocities of the flock. This illustrated by the following simulations, that involved three obstacles and four agents (Figures 2.3, 2.4, and 2.5).

The algorithm implementing the simulations confirmed that the steady-state velocities of the four robots will be the same and equal to the average of the initial velocities of the flock

$$\vec{v}_i^{stdy} = \frac{1}{n}\sum_{j=1}^{n} \vec{v}_{0,j}. \tag{2.15}$$

<u>Remark</u>: there is no subscript i in the righthand side of equation (2.15). This should not let to a confusion. The absence of the i subscript on the righthand side indicates that the steady velocities of the agents are the same.

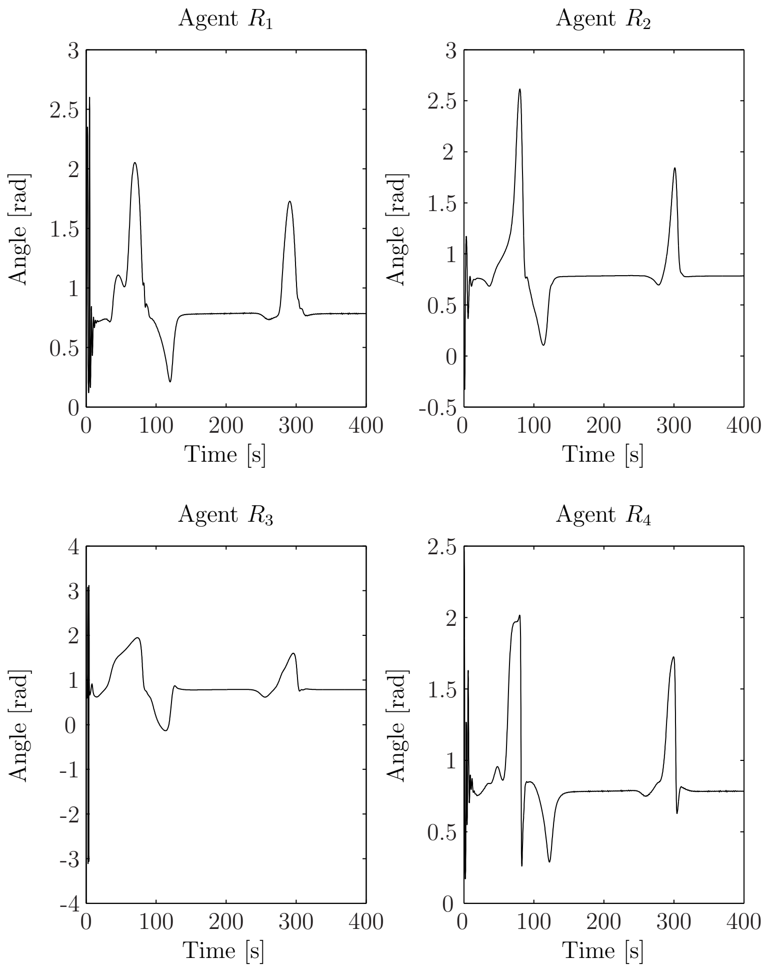


Figure 2.4: Directions of each of the agents of the flock

2.5 Following a Virtual Leader

In Section 2.4 we discussed that the input $\vec{u}_j^2$ is responsible for the convergence of the velocities of the agents to the average of their initial velocities. However, in the case when obstacles are present, the steady state velocities will be significantly influenced by these objects. This is not a desirable behavior since our primary goal is to define a direction according to which the flock should progress while avoiding obstacles.

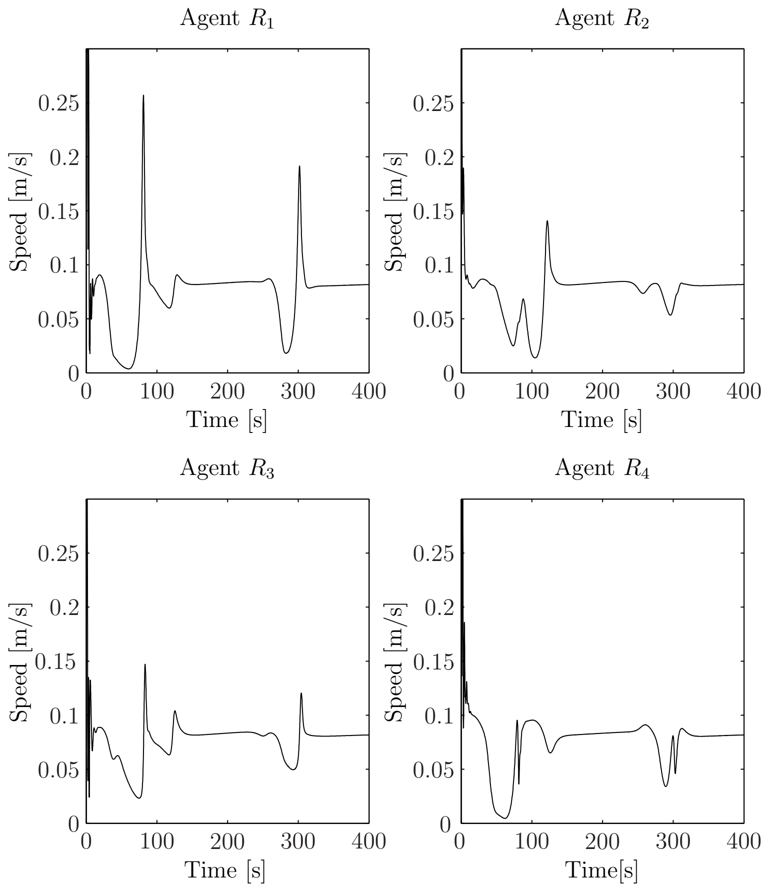


Figure 2.5: Speeds of the agents of the flock

In order to prevent the flock from diverging from its predefined course, we introduce a so-called *virtual leader*. A virtual leader is an imaginary flock member that does not sense the obstacles along its path. The dynamics of a virtual leader serves as a reference signal for the system. The control signals for the members of the rest of the flock take into account the velocity of the virtual leader. Using this approach, we insure that the flock will constantly follow the leader while complying with the rest of the requirements (i.e. maintain the formation and a constant velocity).

In general, the dynamics of a virtual leader may be described by the following open-loop linear system

$$\begin{aligned}\dot{\vec{r}}_{vl} &= \vec{v}_{vl},\\ \dot{\vec{v}}_{vl} &= \vec{u}_{vl};\end{aligned} \tag{2.16}$$

where $\vec{r}_{vl}$ is the vector-position of coordinates of a virtual leader, and $\vec{v}_{vl}$ is its velocity. Vector $\vec{u}_{vl}$ is a reference vector controlling the behavior of the virtual leader. The initial values are defined as $\vec{r}_{vt}(0)$ and $\vec{v}_{vt}(0)$. The input of the system $\vec{u}_{vl}$ determines behavior of the virtual leader.

In our case, the virtual leader must simply maintain a fixed speed and a fixed direction. We therefore set $\vec{u}_{vl} = \vec{0}$, so that the dynamics of the virtual leader is completely determined by the initial data

$$\vec{v}_{vl}(t) = \vec{v}_{vl} = const. \tag{2.17}$$

By having a complete description of the virtual leader, we factor its behavior into the flock dynamics and make the flock follow the imaginary agent regardless of any probable obstacles. The corresponding control signal component has the following form:

$$\vec{u}_i^3 = -k_3(\vec{v}_i - \vec{v}_{vl}). \tag{2.18}$$

Figures 2.6, 2.7, and 2.8 demonstrate how a virtual leader influences the behavior of the flock. In all three cases, conditions such as initial speeds and directions, initial locations of the agents and obstacles are chosen to be the same. Only the coefficient k_3 that regulates the strength of the virtual leader effect is changed.

There was no virtual leader present in the case depicted in Fig. 2.6 ($k_3 = 0$). In this case, we observe a substantial amount of uncertainty in the behavior of the flock.

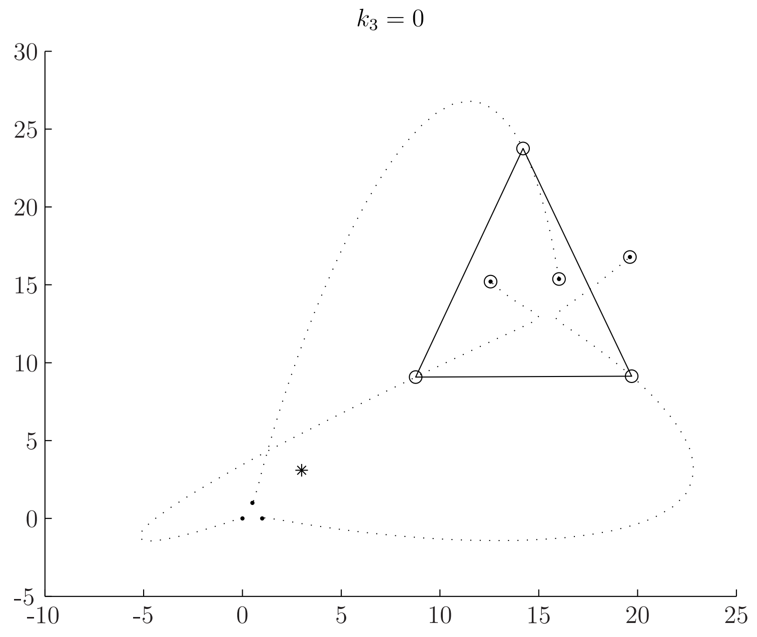


Figure 2.6: A virtual leader is not present($k_3 = 0$). Notations used: initial locations (●), terminal locations ($\odot$), obstacles ($*$).

Namely, the trajectories do not converge. In the case of a weak virtual leader, in Figure 2.7, we choose coefficient $k_3 = 0.1$, which is comparable in magnitude to the coefficients that are responsible for obstacle avoidance, while maintaining a velocity mode and formation.

In the case of a weak virtual leader the flock organized itself and followed the direction identified by the virtual leader (Fig.2.7).

Finally, in the last scenario, depicted in Figure 2.8 we set $k_3 = 1$. In this case, the trajectories fluctuate a little around the obstacle. The level of self organization in the flock is significantly higher than in the previous two cases (see Fig.2.8).

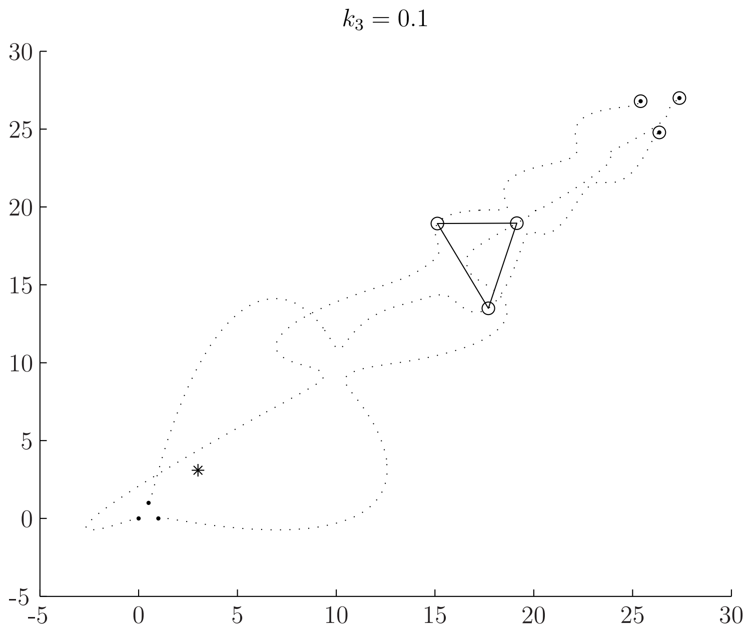


Figure 2.7: A virtual leader is present but weak ($k_3 = 0.1$). Notations used: initial locations (●), terminal locations ($\odot$), obstacles ($*$).

2.6 Obstacle Avoidance

In this section we discuss the control input $\vec{u}_j^4$ that guides a flock of agents through a field of obstacles without colliding with them.

The potential function approach described in Section 2.3 may be used to address this problem. Namely, a suitable potential function whose minimum will be achieved at the point of the most desired configuration of the flock is created. Then, a gradient control law is fed into the main dynamics to bring the system into the state with the smallest potential.

In this case however, the potential function used for maintaining a formation and described in Section 2.3 is not the best choice. Let us first take a closer look at the interaction between an agent and an obstacle versus the agent-agent interaction.

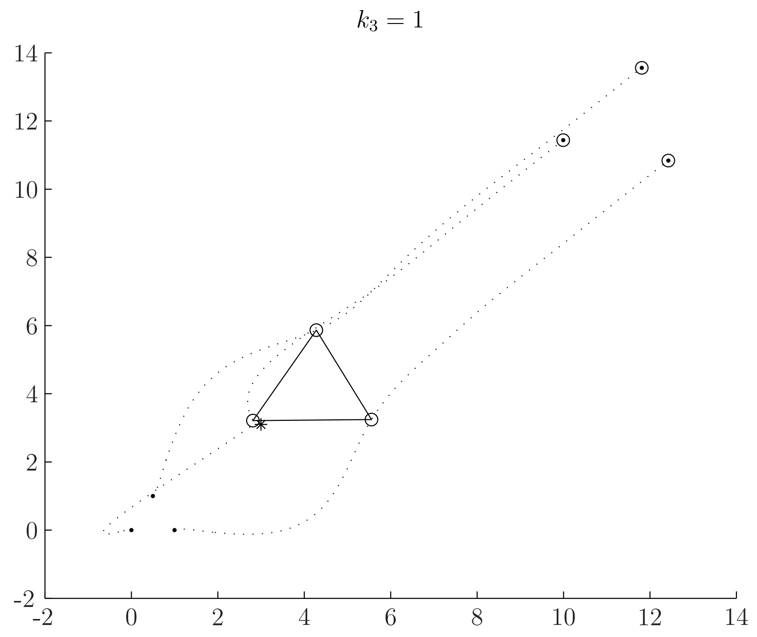


Figure 2.8: A virtual leader is present and is very strong ($k_3 = 1$). Notations used: initial locations (●), terminal locations ($\odot$), obstacles ($*$).

In the case of an agent-agent interaction, the value of potential function increases when two agents get closer or separated away from the optimum location. Both of these scenarios induce a corresponding control signal to correct the problem. In agent-obstacle interaction, the value of the potential function should remain the same if the distance between the agent and the obstacle goes above a certain critical threshold. In other words a potential function for this scenario must remain the same until a certain distance is reached. This will make the system sensitive to an obstacle only if one of the agents turns out to be at critical proximity of the obstacle. This behavior can be implemented by introducing a modified potential function (Fig. 2.9)

$$\tilde{f}(s) = \begin{cases} f(s), & x \leq 1 \\ 1, & x > 1 \end{cases} \tag{2.19}$$

where f is the standard potential function given by (2.8) in Section 2.3.

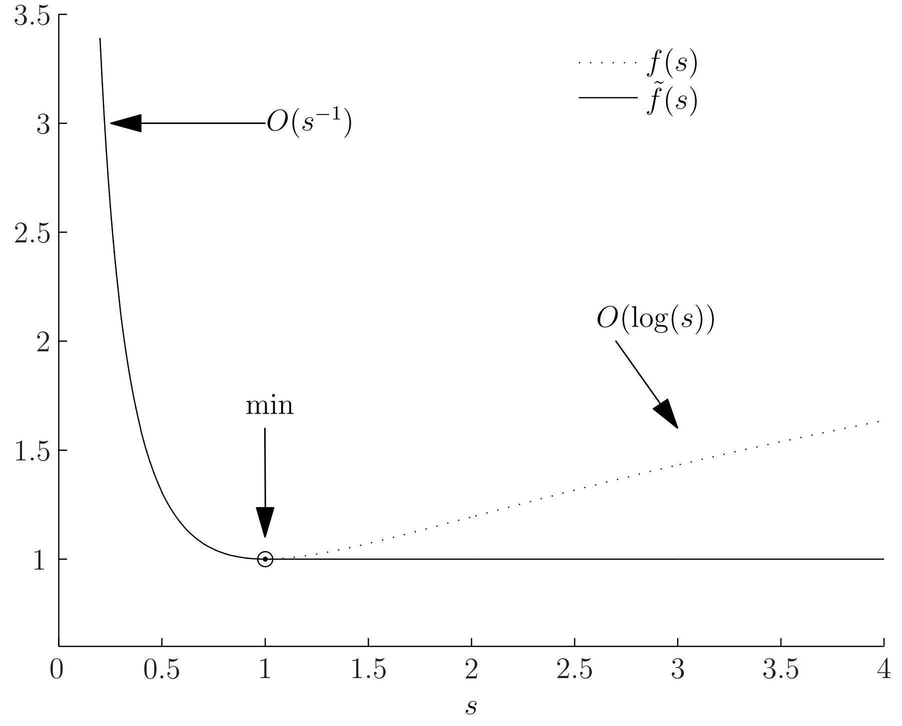


Figure 2.9: A Potential Function for Obstacle Avoidance.

The rest of the development is the same as in Section 2.3. Namely we define a combined potential with respect to all obstacles

$$\tilde{P}_j(\vec{r}_j) = \sum_{i=1}^{m} \tilde{f}\left(\frac{\|\vec{r}_j - \vec{\rho}_i\|^2}{\mathcal{R}_i^2}\right), \tag{2.20}$$

where $\vec{\rho}_i$, $i = 1, \ldots, m$ represent coordinates of the assumed circular obstacles and $\mathcal{R}_i$ their radii.

Then, the component of the control that corresponds to obstacle avoidance is given by

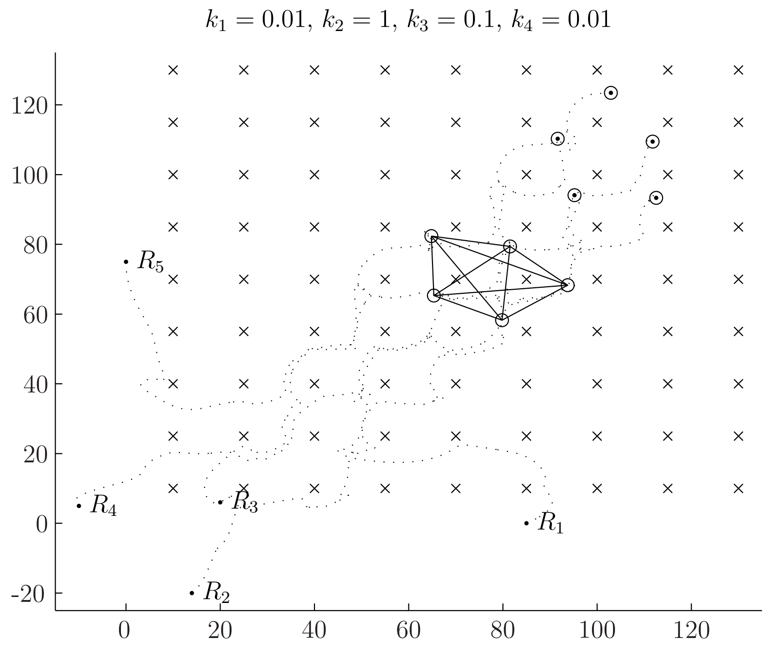


Figure 2.10: Navigation through an obstacle field. Notations used: initial locations ($\cdot$), terminal locations ($\odot$), obstacles ($\times$).

$$\vec{u}_j^4 = -\vec{\nabla}_{\vec{r}_j} \tilde{P}_j = -\sum_{i=1}^{m} \vec{\nabla}_{\vec{r}_j} \tilde{f} \left(\frac{\|\vec{r}_j - \vec{\rho}_i\|^2}{\mathcal{R}_i^2} \right). \tag{2.21}$$

This approach is demonstrated in the example of navigating a flock through a field of obstacles (Fig.2.10). In this case, we choose using trial and error, the gains $k_1 = 0.01$, $k_2 = 1$, $k_3 = 0.1$, and $k_4 = 0.01$. The obstacles were uniformly distributed throughout a rectangular area. The goal of the flock was to cross the field along its diagonal following a virtual leader. As the simulation shows the flock successfully accomplished its task (Fig. 2.10).

2.7 Conclusions

Adding all the components of the control signal that maintain a formation, avoids obstacles, provides a special velocity mode and, makes the flock follow a virtual leader, we obtain a control input that closes the loop:

$$\vec{u}_j = -k_1 \sum_{i \neq j} \vec{\nabla}_{\vec{r}_j} f\left(\frac{\|\vec{r}_{ij}\|^2}{d_{ij}^2}\right) - k_2 \sum_{i \neq j} (\vec{v}_i - \vec{v}_j) - k_3(\vec{v}_j - \vec{v}_{vl}) - k_4 \sum_{i=1}^{m} \vec{\nabla}_{\vec{r}_i} \tilde{f}\left(\frac{\|\vec{r}_j - \vec{\rho}_i\|^2}{\mathcal{R}_i^2}\right) \tag{2.22}$$

where k_1, k_2, k_3, and k_4 are numerical coefficients chosen to prevent input saturation in practical implementations of the flock. They also correspond to the rates of convergence during simulations. It was observed that for example for $k_1 = 0.1$, the flock attained a predefined structure much faster than in case when $k_1 = 0.01$

While experimenting with the simulations it was also established that by adjusting the coefficients we can prioritize which goal is more important. Thus, by setting $k_1 >> k_4$, we specify that keeping the formation is of significant importance, and that the formation will unlikely break in the event of encountering an obstacle. On the other hand if $k_4 << k_1$, the formation may be temporarily distorted around an obstacle but, will be assembled again as the formation distances itself from the obstacle.

It is important to note that the general controller given by (2.22) has no theoretical results for stability. However if we limit the control input to the first two components, namely $\vec{u}_i^1$ and $\vec{u}_i^2$, responsible for maintaining a formation and velocity convergence, theoretical results for stability are available in [10].

In the next chapter we review the general system design, its components and characteristics.

Chapter 3

Test-bed design and Construction of the cooperative agents

3.1 Introduction

In this chapter, we start by reviewing the general system design, position sensing odometry (sensing), desired of the mobile robot performance (motor control), then describe the physical structure in Section 3.2. In Section 3.2 we review the general system design of our robot. Furthermore we describe the engineering choice of our power-supply, and the DC motor in Sections 3.3 and 3.4. In Section 3.5 we explain the features of the optical encoders. In Section 3.6, we explain the concept of motor driver/controller including PWM and the Hbridge circuitry. The Section 3.7 explains the features of the National Instrument data acquisition card. The theory of operation for the IR sensors including the distance measurements formula will be presented in Section 3.8. System identification is discussed in Section 3.9 and is followed by a brief discussion about the design of the PID controller in Section 3.10.

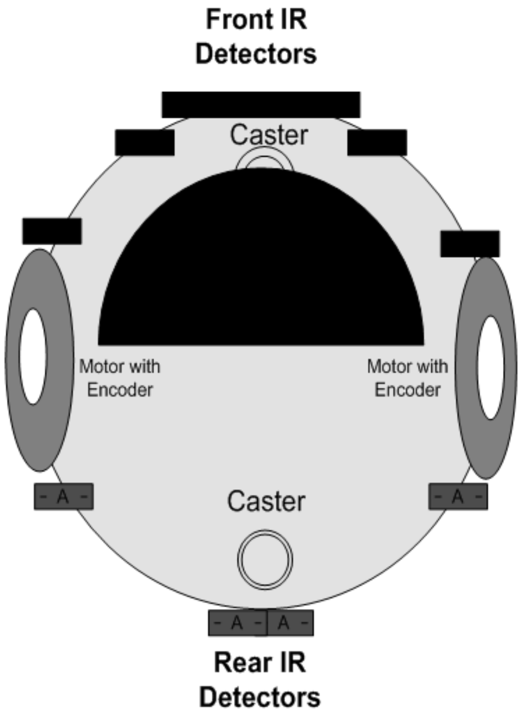


Figure 3.1: Block Diagram of the Robot.

3.2 Physical Structure

Our mobile base platform consists of a 12-inch diameter (30cm) base and half circle deck. It features dual casters and dual drive motors making it very stable. This robot will turn on its own central axis making it maneuverable. The base has dual 9.6 volt 20in-lb torque drive motors (The Max speed is 10.56 meter per minute under full load), The drive wheels are six inches (15 cm) in diameter. Each caster wheel is three inches (7.5 cm) in diameter. The base is balanced with two casters. The maximum recommended payload is 35lbs (13.6 kg). The top view of the design with casters, sensors, position of the motors and encoders is shown in Fig. 3.1.

Figure 3.2 describes the block diagram of the robot including the signal manip-

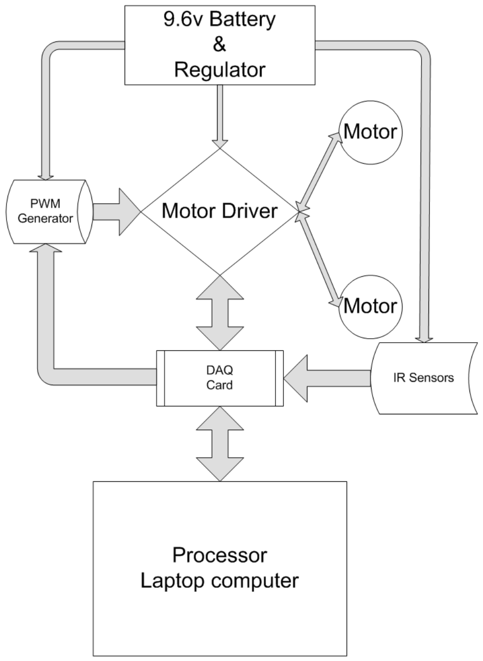


Figure 3.2: Block Diagram of our Robot

ulation through DAQ card, sensors, and the power distribution to the PWM, motor driver and IR sensor.

3.3 Power Supply and Regulation

Linear regulators provide significant advantages over switching regulators in terms of simplicity, cost, and output noise. They are, however, less efficient than their switching counterparts. When applied to battery-operated portable equipment, battery life is more important than individual circuit efficiency. Although the linear regulated power supplies have very little ripple and output noise, we chose a switching regulated power supply instead of a linear regulated power supply, due to its higher efficiency, and lower heat dissipation. In addition, the particular battery characteristics, whether alkaline, NiCd, NiMH, or Lithium (Li+), must also be considered. A

	Linear	**Switching**
Function	Only steps down; input voltage must be greater than output.	Steps up, steps down, or inverts
Efficiency	Low to medium, but actual battery life depends on load current and battery voltage over time; high if $V_{in} - V_{out}$ difference is small.	High, except at very low load currents (μA), where switch-mode quiescent current is usually higher.
Waste Heat	High, if average load and/ or input/output voltage difference are high	Low, as components usually run cool for power levels below 10W
Complexity	Low, which usually requires only the regulator and low-value bypass capacitors.	Medium to high, which usually requires inductor, diode, and filter caps in addition to the IC; for high-power circuits, external FETs are needed
Size	Small to medium in portable designs, but may be larger if heatsinking is needed.	Larger than linear at low power, but smaller at power levels for which linear requires a heat sink
Total Cost	Low	Medium to high, largely due to external components
Ripple/Noise	Low, no ripple, low noise, better noise rejection.	Medium to high, due to ripple at switching rate.

Table 3.1: Comparison Between Linear and Switch-Mode Regulators.

comparison of linear and switch mode regulators is presented in Table 3.1.

3.4 DC motor

We used a 12V Permanent Magnet Spur Gear-motors type (brushless), which has a standard torque rating of 300 oz.in, a weight of about 16.5 ounces (without the wheels), a precision high density sintered gear, and a life-lubricated motor and gearbox bearings. Figure 3.3 demonstrates the relationship between the output current, rpm, and the torque of our DC motors.

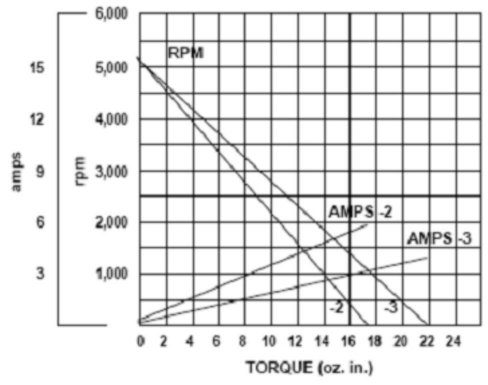


Figure 3.3: Relationships between output current, rpm, and the torque.

3.5 Encoder

Our platforms are equipped with two optical encoders featuring two channel quadrature output with optional index pulse, resolutions up to 500 counts per revolution with $-40°C$ to $100°C$ operating temperature. They are TTL Compatible, and operated with a single 5V supply. These encoders emphasize high reliability and high resolution. Each encoder contains a lensed LED source, an integrated circuit with detectors and output circuitry, and a code-wheel which rotates between the emitter and detector IC. Figure 3.4 shows the direction of the rotation when the output waveforms code-wheel rotates in the counterclockwise direction (as viewed from the encoder end of the motor), channel A will lead channel B. If the code-wheel rotates in the clockwise direction, channel B will lead channel A.

3.6 PWM and H-Bridge Motor Drive

Since the output from the data acquisition card can not directly provide enough power to drive the motors, we need to use a power amplifier. In this thesis, we used

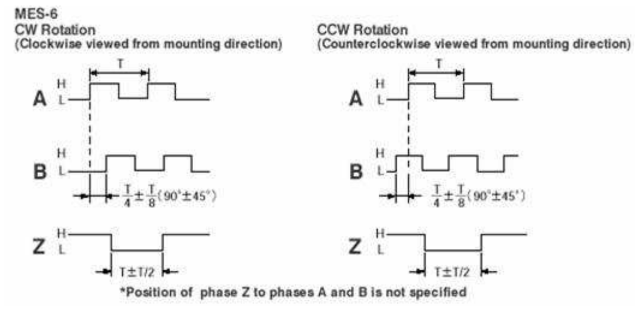
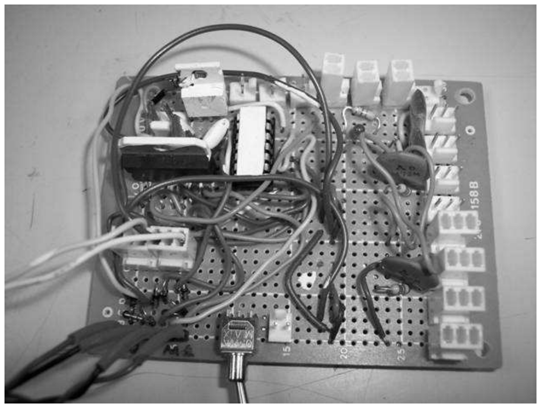
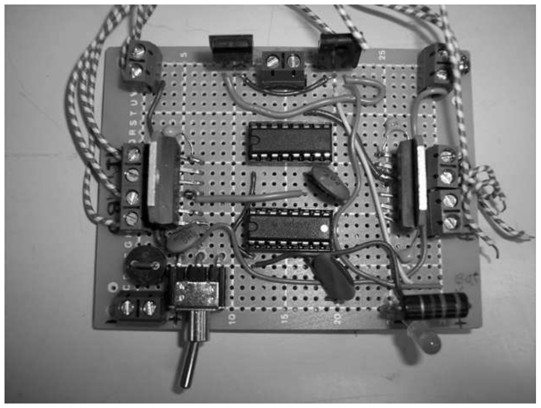

MES-6
CW Rotation
(Clockwise viewed from mounting direction)
CCW Rotation
(Counterclockwise viewed from mounting direction)
T
A
B
Z
H
L
$\frac{T}{4}\pm\frac{T}{8}(90°\pm45°)$
T±T/2
*Position of phase Z to phases A and B is not specified

Figure 3.4: Direction of Rotation Diagram.



(a) Older Design



(b) Newer Design

Figure 3.5: PWM and H.Bridge Printed Circuit Board

a common motor driver known as H-Bridge (LM298 chipset). This chip accepts a train of pulses as input, and using switching transistors, replicates the input signal with higher voltage and current levels. The chip has two separate amplifiers for each motor to provide up to 2Amp continuous current with a maximum voltage of 55V. In order to change the speed of the motor, only the pulse width of the input pulse needs to be changed. This is based on a technique called Pulse Width Modulation (PWM) that is widely known and used in industry. The PWM parameters can be set directly, or can be managed by a local motor controller. Figure 3.6 contains designed and constructed PWM and H.Bridge printed circuit board (Fig.3.5(a) -

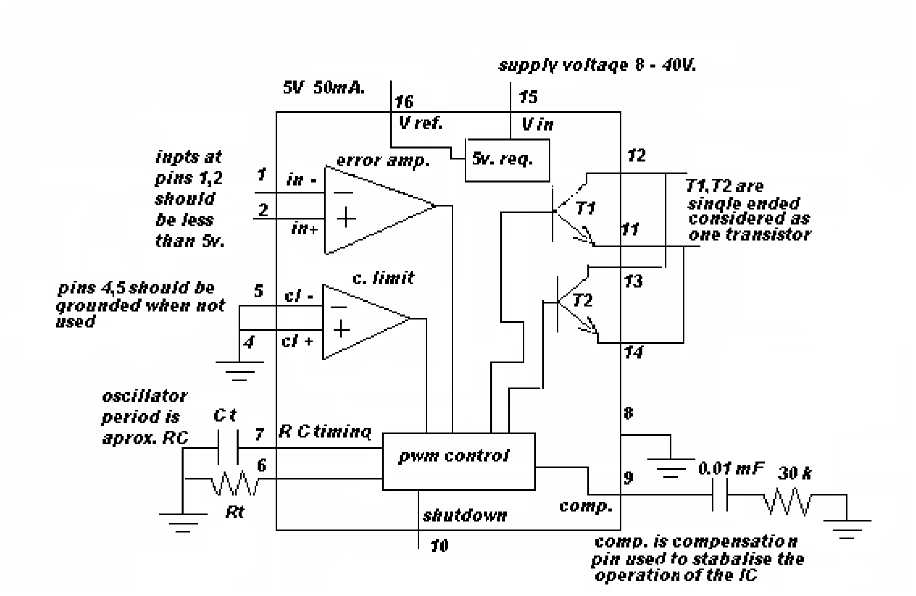

supply voltage 8 - 40V.
5V 50mA.
16
V ref.
15
V in
5v. reg.
inpts at pins 1,2 should be less than 5v.
1
2
in -
in+
error amp.
12
T1
11
T2
13
14
T1,T2 are single ended considered as one transistor
pins 4,5 should be grounded when not used
5
4
cl -
cl +
c. limit
oscillator period is aprox. RC
Ct
7
6
Rt
R C timing
pwm control
8
9
0.01 mF
30 k
shutdown
10
comp.
comp. is compensation pin used to stabalise the operation of the IC

Figure 3.6: Internal Block diagram of the 3524 PWM generator.

an older and Fig.3.5(b) - a newer design.) The motor controller can control the speed or position of the motor, setting the correct PWM value according to the real speed read on the incremental encoders (Fig. 3.4). To generate the required PWM (SG3524, Fig.3.6), a separate circuitry was developed to accept an analog input between 0 to +5 volts and maps it to 0 to 100 percent duty cycle as shown in Figure 3.7. The frequency of the pulse is fixed and was calculated to be around 20KHz using the simple formula: $f = \frac{1.8}{R_T C_T}$, where R_T and C_T are total resistance and total capacitance respectively. (Fig.3.7). We chose a 1KΩ resistor, and 50pF capacitor for our purpose. The frequency is chosen to be outside the human hearing range, yet fast enough so that the motors do not react to the single switching. The analog input to the PWM modulator is provided by the data acquisition card and is the control input that is sent out by the PID controller as described in Section 3.10.

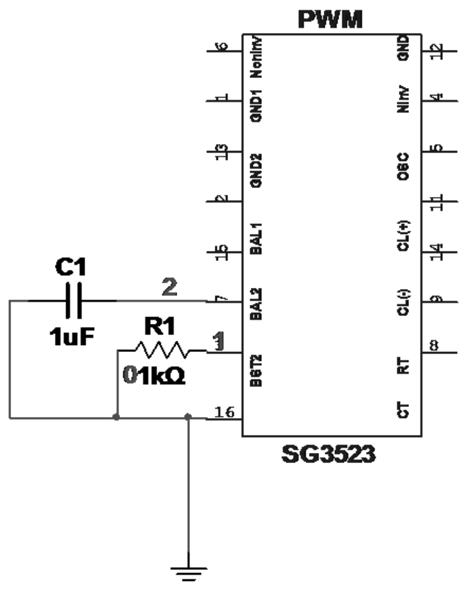


Figure 3.7: Block diagram of PWM with the 18200 (Motor Driver)

3.7 Data Acquisition

To interface with our mobile robot, we used two different types of data acquisition/ The first is the National Instruments USB-6221, which is a USB high-performance M Series multifunction data acquisition (DAQ) module optimized for superior accuracy at fast sampling rates. It is ideal for applications such as data-logging and IR sensor measurements. The other one is the 6024E PCMCI (DAQ).

The National Instruments USB-6221 is designed specifically for mobile or space-constrained applications. Plug-and-play installation minimizes configuration and setup time, while direct screw-terminal connectivity helps keep costs down and simplifies signal connections. This module also features the new NI Signal Streaming technology which allows for DMA-like bidirectional high-speed streaming of data across the USB bus. The USB-6221 and 6024E PCMCI both have sixteen analog inputs, two analog outputs, eight digital input ports and two counters.

3.8 Sensors

In the past few years, several new infrared detectors have been introduced. These detectors offer a small package, very little current consumption, and a variety of output options. This section offers an overview of the various types of detectors, and information on how we were able to interface them in our system.

These detectors are are inexpensive, use very little power, fit in small spaces, and have a unique range that is ideally suited to small robots in human spaces such as hallways, rooms, and the occasional maze.

While such detectors do not give absolute range accuracy, they offer rich information for a robot that typically deals with noisy information in the first place. Often, knowing whether a robot is close to a wall/object is enough to make choices about what to do next.

The operation of such detectors uses triangulation and a small linear CCD array to compute the distance and/or detect the presence of objects in the field of view. The basic idea is that a pulse of IR light is emitted by the emitter. This light travels out in the field of view and either hits an object or just keeps on going. In the case of no object, the light is never reflected and the reading shows no object. If the light reflects off an object, it returns to the detector and creates a triangle between the point of reflection, the emitter, and the detector.

The angles in this triangle vary based on the distance to the object (Fig.3.8). The receiver portion of these new detectors is actually a precision lens that transmits the reflected light onto various portions of the enclosed linear CCD array based on the angle of the triangle described above. The CCD array can then determine what angle

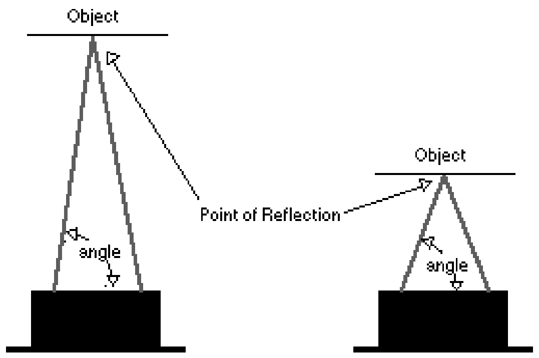


Figure 3.8: Different Angles with Different Distances

the reflected light came back at, and therefore, it can calculate the distance to the object.

This new method of ranging is almost immune to interference from ambient light and is indifference to the color of object being detected. Detecting a black wall in full sunlight is possible.

Because of some basic trigonometry within the triangle from the emitter to reflection spot to receiver, the output of these new detectors is non-linear with respect to the distance being measured.

The graph on Fig.3.9 shows the GP2D12 output from these detectors, while Fig.3.10 demonstrates how to connect the Sharp GP2D12 to the microcontroller, which uses one Analog to Digital I/O Line. There are two interesting features in the graph of Figure 3.9. First, the output of the detectors is not linear within the stated range (10 – 80cm) but rather somewhat logarithmic. This curve will vary slightly from detector to detector so it is a good idea to "normalize" the output with a lookup table or parameterized function. In this way, we calibrate each detector and end up with linear data that is consistent from detector to detector.

The second feature to notice in the graph of Figure 3.9 is that once you fall

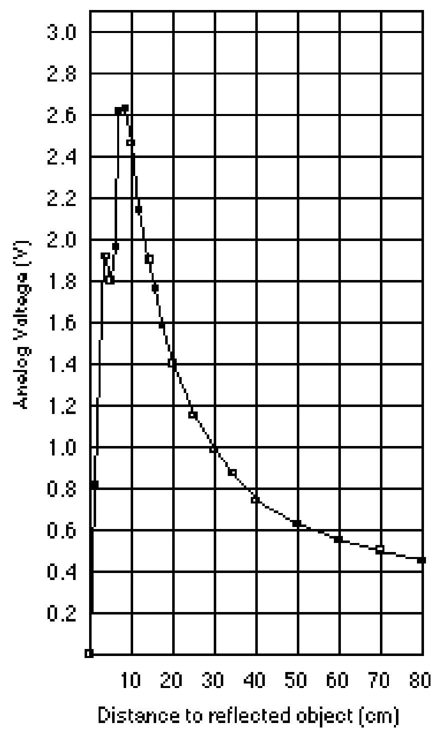


Figure 3.9: GP2D12 Output Voltage to Distance Curve

inside of the stated distance range (less than 10cm), the output drops rapidly and starts to look like a longer range reading. This can be disastrous if your robot is

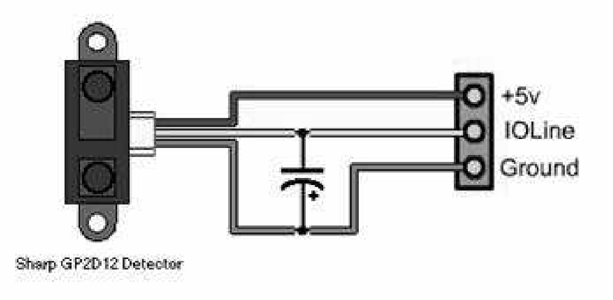


Figure 3.10: Connection to the analog input of the controller.

slowing down as it approaches a solid object, gets below the minimum range, and then misinterprets the apparently long range reading, and driving full-speed into the object. The easiest way to avoid this scenario is to cross-fire the detectors across the width or length of the robot.

A sample of created LabView software and constructed hardware are depicted in Figure 3.11. Namely Figure 3.11(a) includes the LabView code, Figures 3.11(b), 3.11(c), and 3.11(d) show one test-bed, a motion controller, and a flock of four test-beds correspondingly.

3.9 System Identification

In this section we describe two methods for identifying the parameters of a linear, autonomous, discrete time single input-single output system, namely motor. Both methods consist of finding a model that may describe our dynamic system.

The first method of finding this model is the use of the physical principles such as, electrical laws, and laws of motion, which we will need to obtain the differential equations of motion describing our system.

The second method, which we use in this section, is to find a desired model from the observed output data in the Experimental System Identification by exciting the dynamic system with a given random binary input. To simplify our system, we assume that the system we want to identify is linear and time-invariant.

Least square is one of the simplest algorithms in identification. If the system to be identified exhibits linear behavior and the input/output data is relatively noise-free, then the least-square approach will provide good estimates of the parameters of the linear model.

For our purpose we generated random binary noise using Matlab. This is a random input that fluctuates between two distinct values, and is a series of pulses with random duration. Since a step signal has infinite harmonic content, this type of input will also persistently excite the system.

To generate the desired output signal for the system identification, we exited our system with a random 5V binary input in our open loop system. We then recorded the corresponding input/output. The data is presented on Fig.3.12. Since the plant consists of a mass inertia connected to the motor, the system should be at least of second order. To account for the possible non linearity in the system and quantization noise, the model was also derived based on 3rd and 4th order models. In order to verify how close we are to the real plant we compared the step response of the simulation models with the real step response of the system as shown in Figures 3.13 and 3.14. We can clearly see in there that the 3rd order model is the best fit and that higher order models do not offer a significant advantage. Although we did not use the 2nd order model, do we assumed a second-order to calculate the natural frequency of our system. Since the rule of thumb suggests a minimum of 10 times natural frequency to be used as a sampling frequency [53], and that the natural frequency we have calculated for our system is 1.03Hz (Appendix B), the 500Hz sampling frequency is sufficient for our design. The step response of the experimental and identified system follow very closely, which is a good indication of a successful identification process. However as time passes, the experimental and derived model response start to diverge, which at first glance look unacceptable. If we examine the results carefully, it is clear that the system has a free integrator, which accumulates the error over time. For practical purpose we, neglect this effect since we closed the loop in our system. The transfer functions given by (3.1) and (3.2) represent the 2nd and 3rd order models of the system ($f_s = 500$Hz). The derivations are presented

CL Response	**Rise Time**	**Overshoot**	**Settling time**	**S-S Error**
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminated
K_d	Small Change	Decrease	Decrease	Small Change

Table 3.2: The Three-Term Controller with Proportional, Integral, and Differential gain.

in Appendix B.

$$G_2(z) = \frac{0.08751z + 0.9526z}{(z - 0.9992)(z - 0.8748)} \tag{3.1}$$

$$G_3(z) = \frac{0.08936z^2 + 0.1428z + 0.05662}{(z - 0.9982)(z - 0.9107)(z - 0.5794)} \tag{3.2}$$

As one can see, both $G_2(z)$ and $G_3(z)$ have two poles close to $z = 1$, which is expected from theoretical derivation.

3.10 PID Design

The design of a control system is concerned with the choice of a feedback system to achieve a desirable input to output response. In classical control systems, the process of output response stabilization may be achieved by the selection of either Proportional - Integral- Derivative (PID) controllers or Phase compensators (Lead, Lag and Lead-Lag networks). This section discusses the design of a PID controller or compensators for a given Linear Time Invariant Discrete Systems (LTIDS), which we have modeled in Section 3.9. The effects of each of the terms of the controller K_p, K_i, and K_d on a closed-loop system are summarized in Table 3.2.

As evident from Fig.3.14, the feedback from the plant is provided by an encoder. The angular position of the motor is then calculated by adding up the counts provided by the counter as time passes by and using a scaling factor to convert it to the angular position when needed. The desired position of the system will also be provided as counts. As discussed previously, the input to the plant is a voltage in the range of $0 - 5$V, which is converted to the $(0 - 100\%)$ duty cycle,then sent through a power amplifier to the motor.

As it is mentioned in the system Identification section, the sampling frequency of 500Hz is used to obtain a model of the plant, and hence in designing the controller. Figure 3.15 describes the block diagram of the system including the controller. Following the practical rule of thumb, we do not want to have a natural frequency of more than 50HZ when placing the closed-loop poles of the system. The 50Hz is the theoretical upper bound, but since we have approximated the plant model, it is possible that by choosing some natural frequency, some of the dormant modes would be awakened. Such was the case in the first few trials of our controller design. Although the controller showed promising results in simulation, it did not work properly when implemented. After several trials, the following parameters were chosen for controller:

$$\begin{aligned} \omega_n &= 100 \text{ rad/c} \\ \xi &= 0.5 \end{aligned} \tag{3.3}$$

which corresponds to $f_n = 15.9$Hz. These parameters will ensure an overshoot of less than 16% and a zero steady-state error. Although the overshoot of 16% may look unacceptable at first, but since we never subject the system to a direct step input, this overshoot will not be a problem. It may even be an advantage since it would result in a faster transient response.

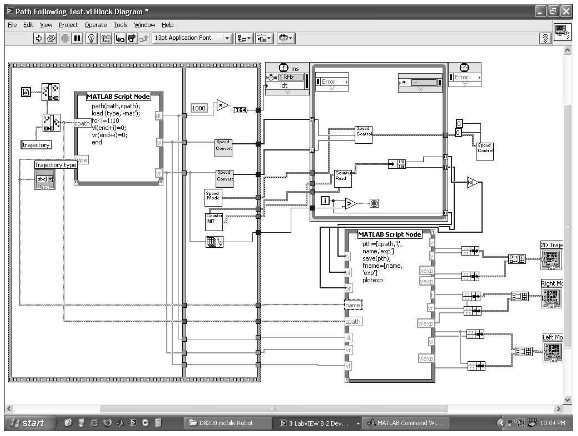
Although a PI controller may have worked, to get a better gain and phase margins, the PID controller was implemented. The PD portion of the PID controller tends to enhance the stability of the system, while the PI part will enhance the transient performance characteristics of the system.

As described in Appendix B, the resulting controller is

$$D(z) = K\frac{(z-0.9107)(z-0.8)}{z(z-1)}. \tag{3.4}$$

From the root locus plot (Fig.3.16) one can obtain that $K = 0.5$ provides a root at the desired location.

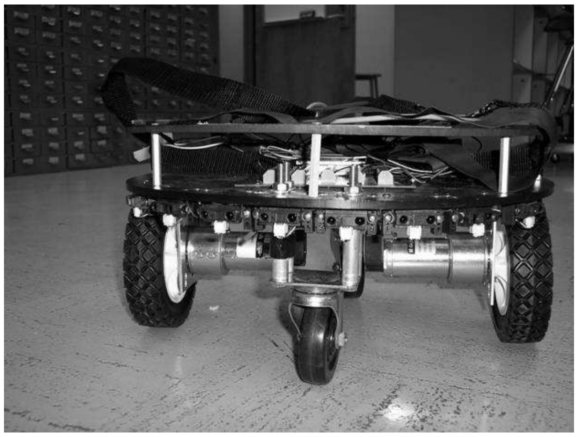
A step response comparison of the Closed Loop PID controller using experimental and theoretical data is shown on Figure 3.17.



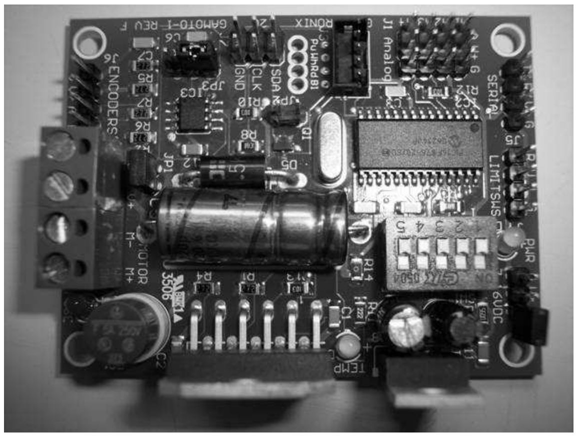
```
path(path,cpath);
load (type,'-mat');
for i=1:10
vl(end+i)=0;
vr(end+i)=0;
end
```

```
pth=[cpath,'\',
name,'exp']
save(pth);
fname=[name,
'exp']
plotexp
```

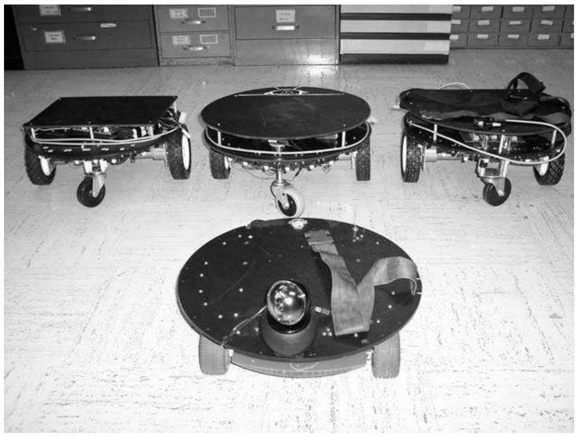
(a) LabView Code.



(b) A Test-bed.



(c) Motion Control Board.



(d) A Flock of Four Test-beds.

Figure 3.11: LabView Code/Test-bed/Motion Controller Board And a Flock of Four Test-beds

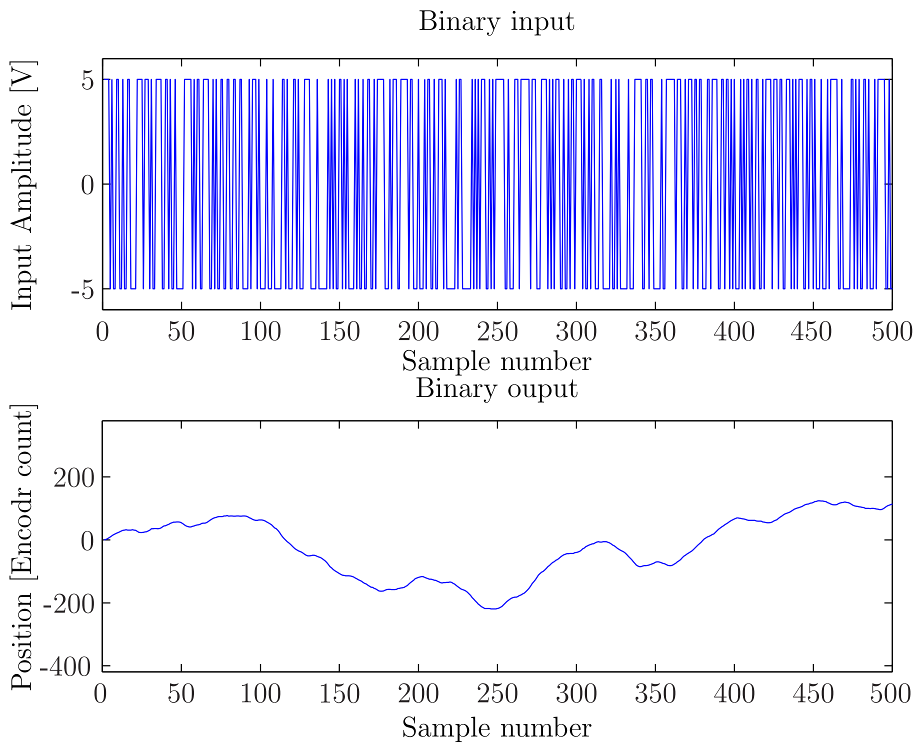


Figure 3.12: Random binary input data and corresponding output.

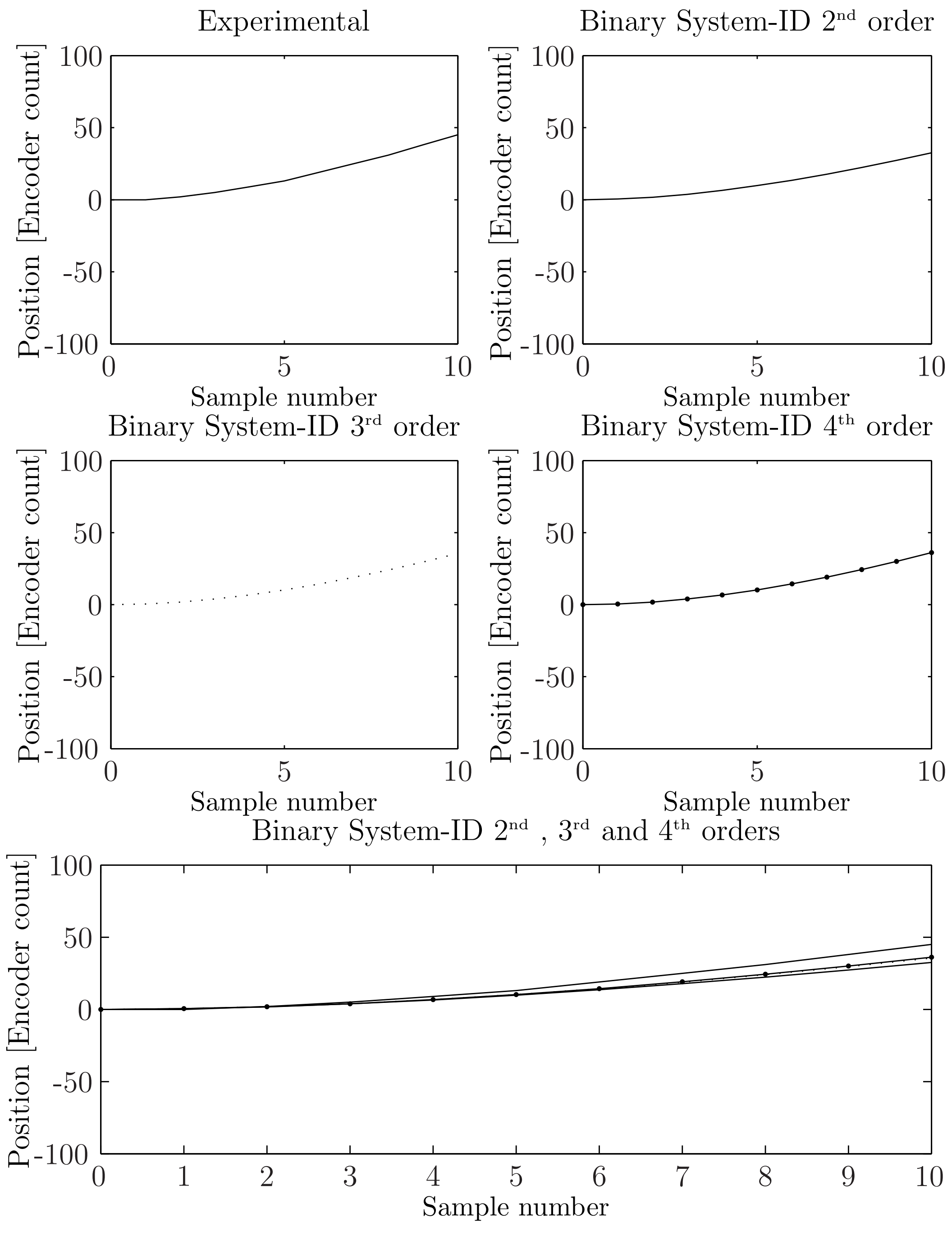


Figure 3.13: System ID for random input for orders 1, 2, 3 and 4.

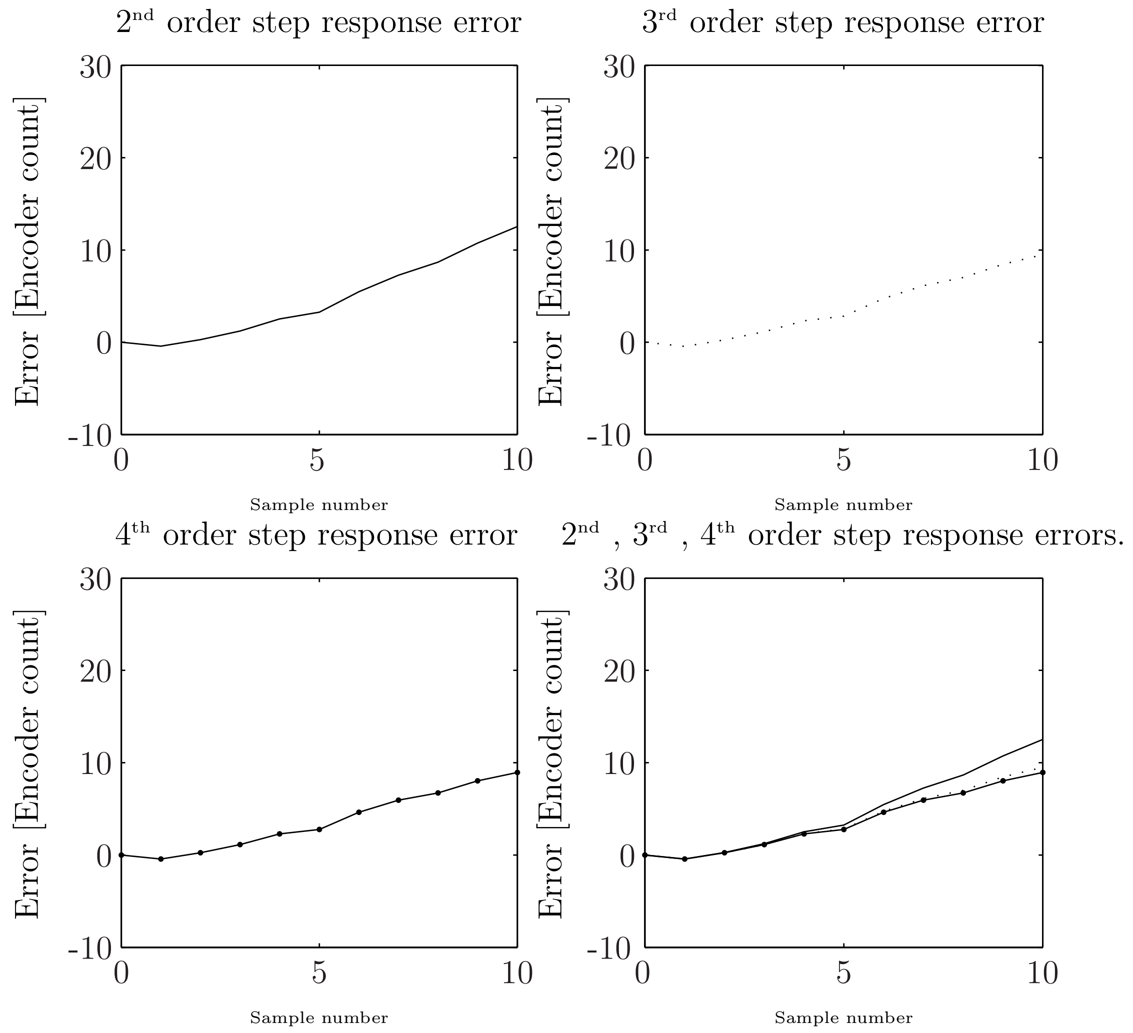


Figure 3.14: Step response errors of 2nd , 3rd , and 4th order models.

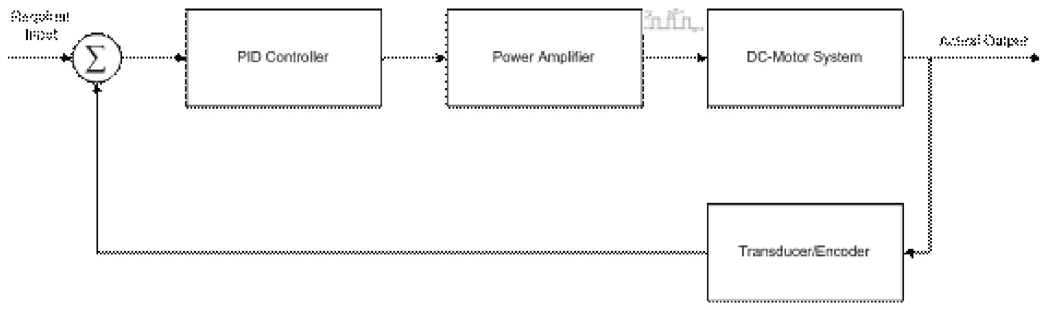


Figure 3.15: Block diagram of the system including the controller.

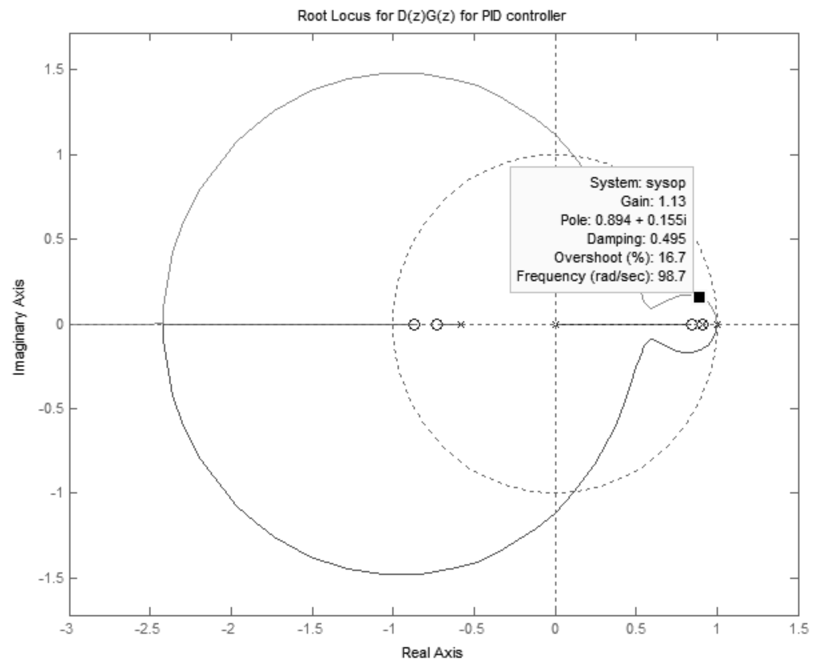


Figure 3.16: Root Locus result for our PID controller.

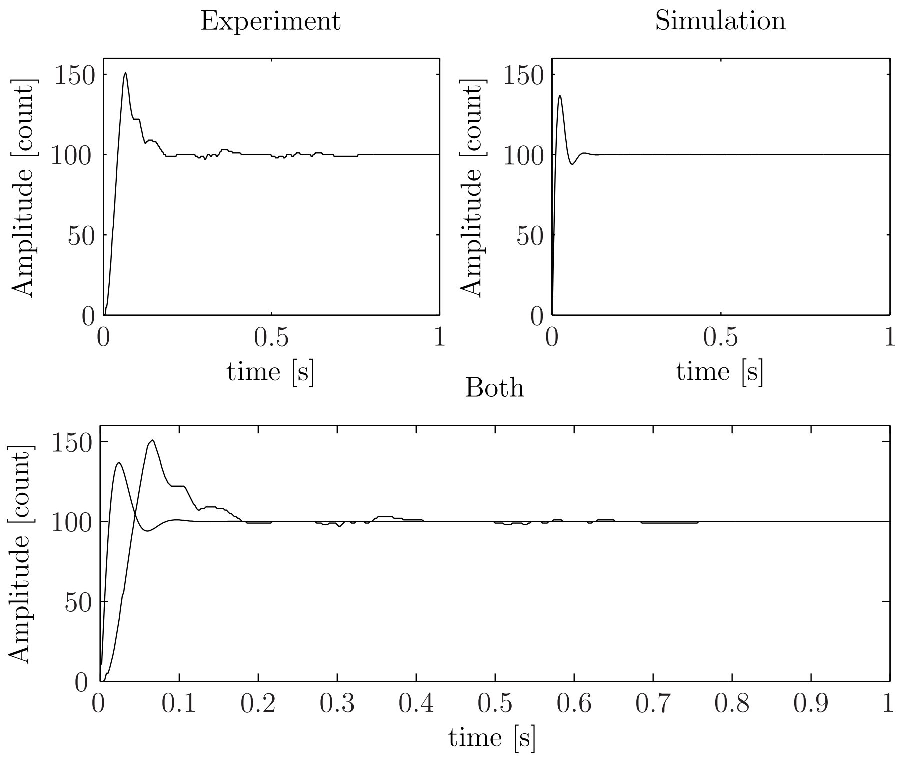


Figure 3.17: Step Response for CL PID controller: Experimental and theoretical data.

3.11 Conclusions

In this chapter we have addressed various aspects in the construction of a working, autonomous mobile robot including its electrical and mechanical modules. We then used the constructed mobile robot to introduce the principles and steps in control system design, using system identification, and PID controller design in order to control the two motors used to maneuver and steer the mobile robot.

In the next chapter we present equations describing the path of a robot equipped with a differentially steered drive system.

Chapter 4

Test-Bed Implementation and Experimental Results

4.1 Introduction

In this chapter we present equations describing the path of a robot equipped with a differentially steered drive system. This simple and reliable system is commonly used in smaller robots. This system is also used in wheelchairs, that have two wheels mounted on a single axis and are independently powered and controlled, thus providing both drive and steering functions. The equations derived in this chapter provide an elementary model for the differentially steered drive system, which is often called a differential steering system. This model may be used to predict how a robot equipped with such a system will respond to changes in its wheel speed and what path it will follow under various conditions.

We should emphasize that the equations used in this chapter represent an elementary model for the motion of a robot. They describe the robot's position and orientation as a function of the movement of its wheels, but in order to simplify our system, we ignore the physics part of this motions such as torques and forces, friction, energy and inertia. This method of describing the motion is referred to as a kinematics approach, which simply ignores the causes of motion (which would be studied in a dynamics approach) and focuses on the effects.

Wheeled mobile robots (WMR) fall into two main categories, Holonomic and Nonholonomic. Simply put when there are no restrictions on solution for motion velocities of a WMR in a plan (2D) the WMR is called holonomic. More elaborate and mathematical definitions could be found in [1]. Therefore any WMR with 3 degree of freedom in a plane is a holonomic. There are various type of mechanism, which result in holonomic platform as mentioned in [1]. One advantage of having a holonomic WMR is that it allows for easier motion planning. On the other hand, in nonholonomic systems [2] there are some links that restrict the system motion velocities. Although there are some inherent problems with this type of mechanisms such as a more difficult path planning paradigm, they are easy and economically cheap to build.

The mobile robot platform was designed based on a popular differential derive system. This platform is a nonholomic system since it is obvious that the mobile base can only move forward and backward, and the sideward motion is restricted by the nature of the wheel mechanical design. This fact will be shown mathematically when the model of the platform is derived.

In differential drive mode, we use two driving wheels (plus a roller-ball for balance). The wheels are fixed to the side of robot as shown in Figure 4.1. As is evident from Figure 4.1, point P cannot move in a direction perpendicular to the plane of the wheels.

This setup has some shortcomings. The most important one is that the robot is sensitive to the relative velocity of the two wheels (small error result in different

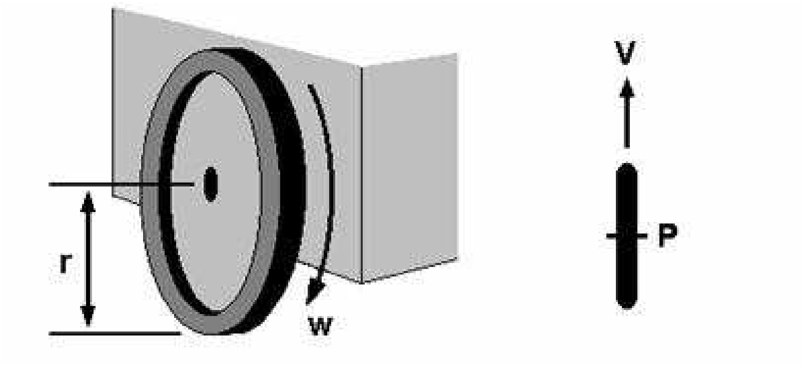
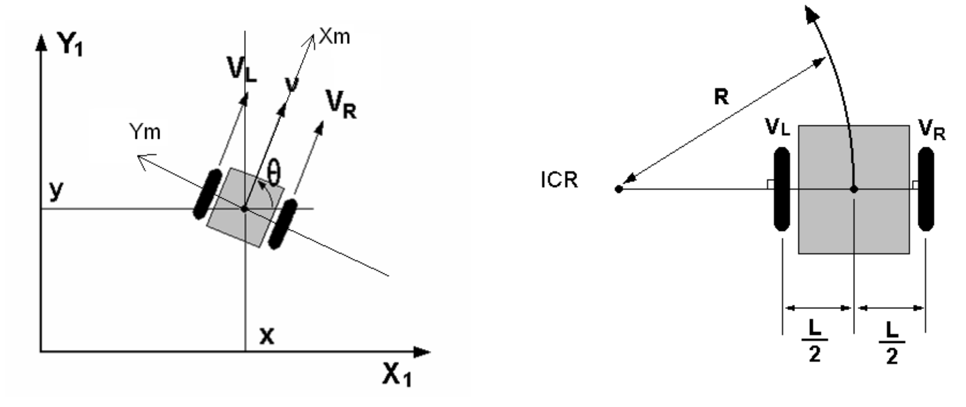


Figure 4.1: Differential Drive mode

trajectories, not just speed). A general nonholonomic differential drive robot platform is depicted in Figures 4.2. The robot consists of two drive-wheels which are symmetrically placed at the side of the platform. Any number of caster wheels may be placed in front and back to maintain the balance and stability of the robot. These caster wheels are not driving wheels, and hence are not included in the kinematics model of the system. To identify the local and global coordinate systems, the frame (X_m, Y_m) is placed at the center of the robot to act as the local coordinate system (Fig.4.2(a)). The stationary global coordinate system is identified as (X_1, Y_1). The rotation of the local coordinate system with respect to the global one is measured by angle θ, which is the angle between X_m and X_1 in a counter clockwise direction. The parameters used in developing the kinematic model are presented in Table 4.1.

4.2 Direct Kinematics

At each time instant, the left and right wheels must follow a trajectory that moves around the instantaneous center of coordinates (ICC) at the same angular rate ω. The angular velocity of the robot ω, instantaneous radius of rotation R and speeds



(a) Global orientation (b) Instantaneous Center of Rotation (ICR)

Figure 4.2: A general nonholonomic differential drive robot platform

V_L and V_R are related through the following expressions

$$\begin{cases} \omega\left(R + \frac{L}{2}\right) = V_R \\ \omega\left(R - \frac{L}{2}\right) = V_L \end{cases} \tag{4.1}$$

r	radius of each wheel ($7.3cm$)
L	distance between the driving wheels along the axis Y_m divided by 2 ($18cm$)
ω_L	angular velocity of the left motor
ω_R	angular velocity of the right motor
V_L	linear velocity of the left motor
V_R	linear velocity of the right motor
V	linear velocity of the robot
ω	angular velocity of the robot
(x, y, θ)	the current position and orientation of the robot
R	instantaneous curvature radius of the robot trajectory (distance from the ICR to the midpoint between the two wheels (see Fig. 4.2(b)))

Table 4.1: Parameters of the kinematic model.

Solving (4.1) for R and ω yields

$$R = \frac{L}{2}\frac{V_R + V_L}{V_R - V_L}; \qquad \omega = \frac{V_R - V_L}{L}. \tag{4.2}$$

The center of the robot is the midpoint of the axis commenting two motorized wheels, therefore

$$V = \frac{V_L + V_R}{2}. \tag{4.3}$$

Taking into consideration the connection between angular and linear speeds

$$V_R = \ r\omega_R \quad V_L = r\omega_L \tag{4.4}$$

from (4.2) we obtain

$$V = \ \tfrac{r}{2}(\omega_L + \omega_R); \qquad \omega = \tfrac{r}{l}(\omega_R - \omega_L)\ . \tag{4.5}$$

Let $\vec{V}$ be the velocity vector of the robot at any moment t represented in the global system of coordinates. Then

$$\frac{d}{dt}\vec{V} = \frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \end{bmatrix}. \tag{4.6}$$

By substitution (4.5) into (4.6) and recalling that $\dot{\theta} = \omega$ we obtain

$$\begin{cases} \dot{x} = \tfrac{r}{2}(\omega_R + \omega_L)\cos\theta \\ \dot{y} = \tfrac{r}{2}(\omega_R + \omega_L)\sin\theta \\ \dot{\theta} = \tfrac{r}{L}(\omega_R - \omega_L) \end{cases} \tag{4.7}$$

Equations (4.7) represent **the Direct Kinematics Equations** for the considered robot. In other words, with defined angular velocities modes $\omega_L(t)$ and $\omega_R(t)$, one can completely resolve the trajectory of the robot $(x(t), y(t))$ including its orientation $\theta(t)$ at any moment of time provided the initial data $x(0)$, $y(0)$, and $\theta(0)$ are given.

4.3 Inverse Kinematics

In this section we consider the inverse problem. Assume that the robot is to follow a predefined trajectory described by the parametric equations $x = x(t)$ and $y = y(t)$. We need to derive equations for the angular velocities of each of the wheels in order to control the currents applied to the motors.

From equations (4.1) we obtain

$$\begin{cases} V + \frac{L\omega}{2} = V_R \\ V - \frac{L\omega}{2} = V_L. \end{cases} \tag{4.8}$$

The velocity vector $\vec{V}$ defined by (4.6) in the local system of coordinates (X_M, Y_M) takes on the form

$$\vec{V}_M = \begin{bmatrix} V \\ 0 \end{bmatrix}, \tag{4.9}$$

since in the local system of coordinate the robot moves along the X_M axis only due to structural restrictions of the model.

The local system of coordinates (X_M, Y_M) and the global system of coordinates (X_1, Y_1) are related through a rotation by angle θ (Fig.4.2(a)). Therefore, the totation between $\vec{V}_M$ and $\vec{V}$ may be expressed as

$$\Phi\vec{V} = \vec{V}_M, \tag{4.10}$$

where Φ is a rotation matrix given by

$$\Phi = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{4.11}$$

By substitution (4.11) into (4.10) and expanding obtain

$$\begin{cases} \dot{x}\cos\theta + \dot{y}\sin\theta = V \\ -\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \end{cases} \tag{4.12}$$

By recalling that $V_R = r\omega_R$, $V_L = r\omega_L$, $\dot{\theta} = \omega$ and substituting (4.8) into (4.12) we obtain

$$\begin{cases} \dot{x}\cos\theta + \dot{y}\sin\theta = r\omega_R - \frac{L\dot{\theta}}{2} \\ \dot{x}\cos\theta + \dot{y}\sin\theta = r\omega_L + \frac{L\dot{\theta}}{2} \\ -\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \end{cases} \tag{4.13}$$

Solving (4.13) for ω_R and ω_L we arrive at the inverse kinematics equations:

$$\begin{cases} \omega_R = \frac{1}{r}\left[\dot{x}\cos\theta + \dot{y}\sin\theta + \frac{1}{2}L\dot{\theta}\right] \\ \omega_L = \frac{1}{r}\left[\dot{x}\cos\theta + \dot{y}\sin\theta - \frac{1}{2}L\dot{\theta}\right] \end{cases} \tag{4.14}$$

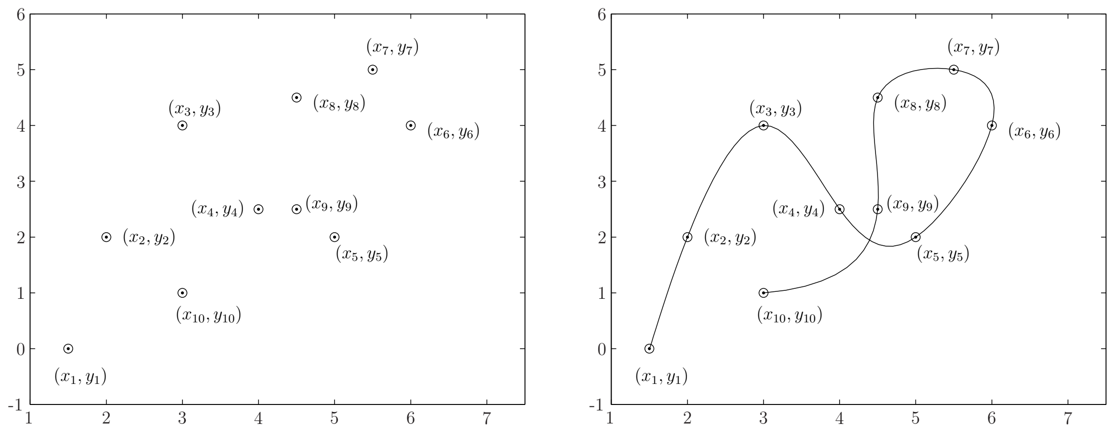
and the nonholonomic constraint

$$\frac{\dot{y}}{\dot{x}} = \tan\theta. \tag{4.15}$$

4.4 Path Planning

In this section we discuss how to implement the inverse kinematics using formulas (4.14) and (4.15). Equations (4.14) and (4.15) are applicable when the trajectory of the robot is known. By parameterizing the desired trajectory curve the angular velocity modes (ω_L,ω_R) can be calculated. In real world application, the parametric equations of motion are rarely known as the trajectories may be curves with arbitrary level of complexity. In practice, a desired trajectory is defined via an ordered set of points coordinates $\{(x_i, y_i)|i = 0 \dots n\}$ (Fig.4.3(a)). Our goal here is to produce a parametrization of the curve that passes through these points (Fig.4.3(b)). In order to accomplish this goal, we resort to a technique known as interpolation.

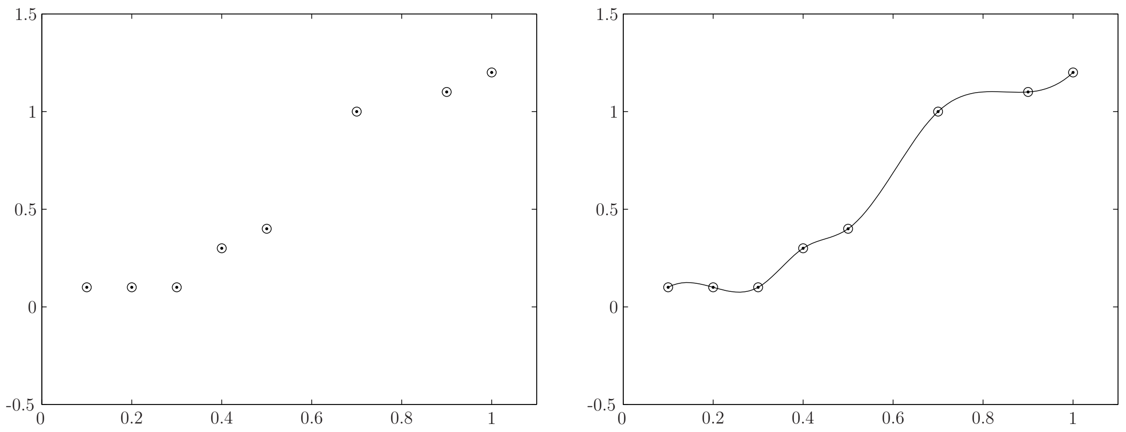
Assume the values of an unknown function $f = f(x)$ are given at points x_j, $j = 0 \dots n$ (nodes of interpolation) i.e. $f_j = f(x_j)$ (Fig. 4.4(a)). The goal is to find a mathematical expression $\hat{f}$ of function f (Fig. 4.4(b)) that allows to estimate



(a) Points of the path (b) Interpolated path

Figure 4.3: Trajectory planning

the values of f not only at the nodes of interpolation, but also at any point x in $[x_0, x_n]$. The only restriction we impose is that the original function must agree with the interpolated function at the nodes: $\hat{f}(x_j) = f_j$.



(a) Nodes of interpolation: (x_j, f_j) (b) Interpolated path: $y = \hat{f}(x)$

Figure 4.4: Interpolation Principle

There are numerous techniques of interpolation. For example, Lagrange Interpolation polynomial [54], B_3-splines [55], or piecewise linear interpolation (Fig.4.5(a)) . For the purpose of this application, cubic-B-splines are the best suited because of the following reasons:

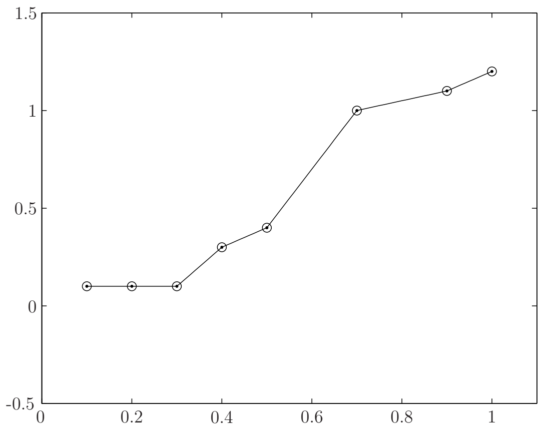
- **Smoothness.** B_3-splines are C^2-class curves meaning they are twice differentiable and their derivatives are continuous even at the nodes. This is a very important property of B_3-spline, since we would like the robot to follow its trajectory smoothly without stops and delays to adjust the heading angle θ. Smoothness is also crucial because of the kinematics model chosen in (4.15) and (4.14) involve time derivatives.
- **Predictability.** Cubic splines behave very well between the nodes of interpolation (Fig. 4.5(c)), while Lagrange polynomials are unpredictable at points that lie between neighboring nodes of interpolation (Fig. 4.5(b)). This behavior is unsuitable for the purposes of trajectory planning since the robot must follow the trajectory in a predictable way. Figure 4.5(d) shows the comparison between piecewise-linear, Lagrange, and spline interpolation methods.
- **Simplicity:** B_3-splines are simple to compute and are readily available through standard software packages such as MatLab and LabView.

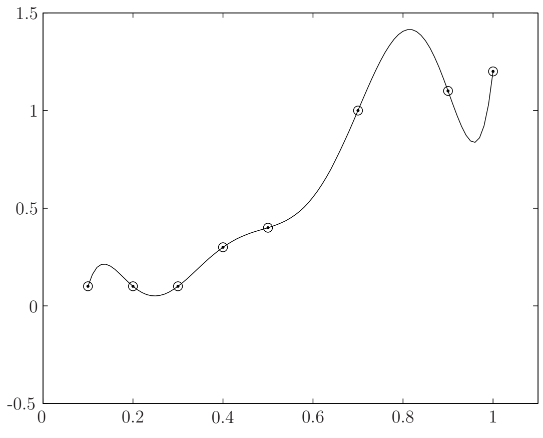
The next step required is to estimate how long it will take the robot to complete the trajectory. This step is very important due to limited hardware capabilities. For the robot at hand, the maximal linear velocity that may be achieved is

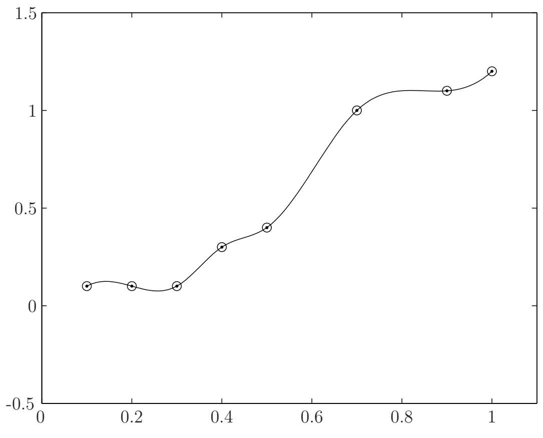
$$V_{max} = 18\frac{\text{cm}}{\text{s}} \tag{4.16}$$

due to the Gamatronix® controller limitations. The time needed to traverse the trajectory is estimated using the following formula

$$T = \frac{S}{V_{max}}, \tag{4.17}$$


(a) Piecewise Linear Interpolation


(b) Lagrange Interpolation


(c) Spline Interpolation

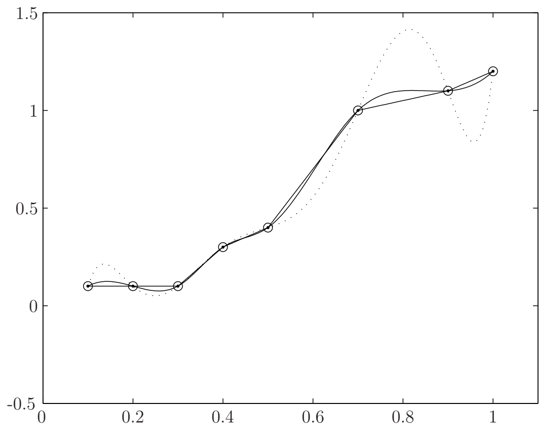

(d) Comparison

Figure 4.5: Interpolation methods

where S is estimated length of the trajectory.

$$S = k \sum_{j=1}^{n-1} \sqrt{(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2}, \tag{4.18}$$

where k is an empirical coefficient that allows us to achieve velocity scope at $[0; V_{max}]$. In practice, it was estimated that $1.0 < k < 1.1$

Since the control signal sent to the robot has a discrete nature we need to establish the time samples at which the robot receives its signals. The Gamatronix® controller allows the sampling time to be $\delta t \geq 20ms$. For this experiment we choose

$$\delta t = 100ms. \tag{4.19}$$

This is a particular choice and other discretization times were also successfully tested.

Next we calculate the number of time intervals for the data stream:

$$N = \frac{T}{\delta t}, \tag{4.20}$$

where T is calculated from (4.17).

The parametric equations of the trajectory passing through the nodes are defined via B_3-spline technique as follows. Let us consider n points of the trajectory: (x_j, y_j), $j = 1 \ldots N$. We define two time scales t_j and τ_j:

$$t_j = j\frac{T}{N} \tag{4.21}$$

and

$$\tau_j = j\frac{T}{n}. \tag{4.22}$$

τ_j is a fine time mesh used to update the robot with a control system, while t_j is a rough time scale that fixes the nodes and times at which the robot must arrive to these nodes. We define two cubic splines based on the rough mesh

$$\begin{cases} x(t) = \text{spline}(t, \{t_j, x_j\}) \\ y(t) = \text{spline}(t, \{t_j, y_j\}) \end{cases} \tag{4.23}$$

Now the trajectory is completely parameterized (Fig. 4.3(b)) and x and y can be calculated at any time t from $[0, T]$. From the constructions we have $(x_j, y_j) = (x(t_j), y(t_j))$.

Let $k = 0 \ldots n$ then at any time moment τ_j of the fine mash according to (4.15)

$$\theta_k = \theta(\tau_k) = \tan^{-1}\frac{\dot{y}(\tau_k)}{\dot{x}(\tau_k)}, \tag{4.24}$$

where $\dot{x}(\tau_k)$ and $\dot{y}(\tau_k)$ are estimated using the central finite difference operator [56]:

$$\frac{df(t_0)}{dt} \approx D(f, t_0) = \frac{f(t_0 + h) - f(t_0 - h)}{2h} \tag{4.25}$$

with h practically chosen to be $h = 10^{-6}$. In order to estimate $\dot{\theta}(\tau_k)$ we use the following approximation

$$\omega(\tau_k) = \dot{\theta}(\tau_k) \approx \frac{\theta(\tau_{k+1}) - \theta(\tau_k)}{\delta t}. \tag{4.26}$$

At this time all data is available $\theta(\tau_k)$, $\dot{\theta}(\tau_k)$, $\dot{x}(\tau_k)$, and $\dot{y}(\tau_k)$ to calculate $\omega_L(\tau_k)$ and $\omega_R(\tau_k)$ through the inverse kinematics formulas (4.14). Experimental results are discussed in the next section.

4.5 Experimental results

To test the approach described in Section 4.4 we chose two trajectories: a circle with a 1m radius and a spiral inscribed in a 2×2.6m rectangle. In order to verify how well the robot performs (follows the trajectory) we measured the outputs of encoders attached to the wheels. It is known from the documentation provided with the hardware that each revolution of the wheel results in 63800 counts of the encoder. The number of counts is dynamically obtained in LabView through the data acquisition board. By knowing the number of counts at any moment of time on each of the wheels, the angle each wheels rotates by can be estimated as follows

$$\Delta\theta = 2\pi \frac{\Delta N}{63800}. \tag{4.27}$$

By knowing $\Delta\theta_L$ and $\Delta\theta_R$, the angular velocities were estimated

$$\begin{aligned} \omega_L(t) &= \frac{\Delta\theta_L}{\delta t} \\ \omega_R(t) &= \frac{\Delta\theta_R}{\delta t} \end{aligned} \tag{4.28}$$

and using (4.4) $\hat{V}_L(t)$ and $\hat{V}_R(t)$ are estimated, where ˆ above the variables signifies that these are velocities estimated from hardware readings.

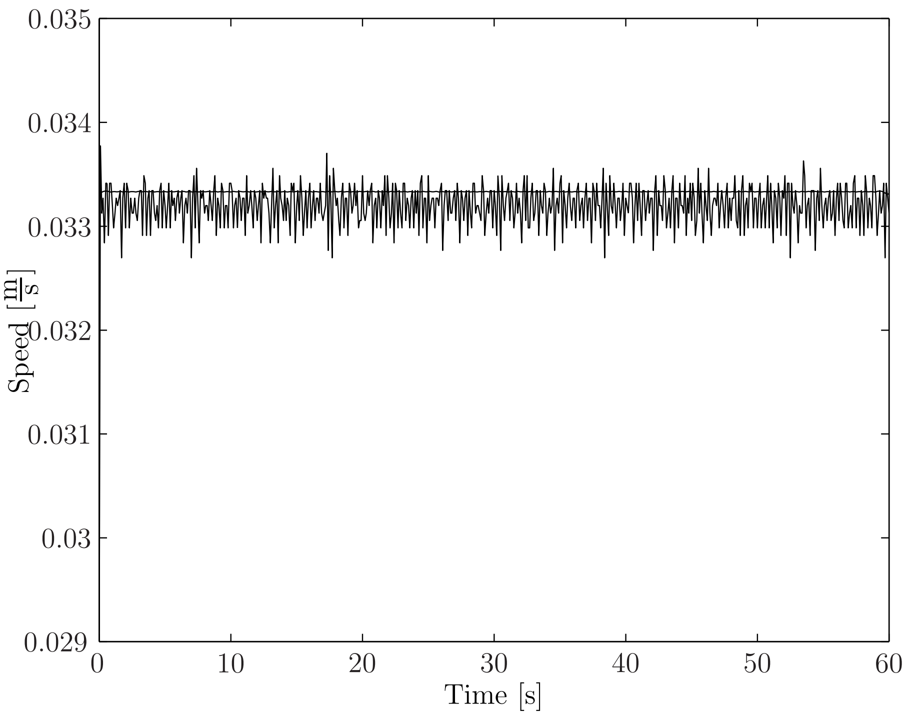
Integrating of the direct kinematics equations (4.7) with initial conditions $x(0) = x_0$, $y(0) = y_0$, and $\theta(0) = \theta_0$ estimates the coordinates of the robot $(\hat{x}(\tau_k), \hat{y}(\tau_k))$ were obtained.

Figures 4.6 and 4.7 present visual results from the experiments of following a circular trajectory. Figures 4.6(a) and 4.6(b) describes how the linear speeds of each wheel deviate from the programmed speed. Figures 4.7(a) and 4.7(b) demonstrate the configurations of the trajectory programmed and the trajectory obtained from readings from the moders in the circular case. Similar information is presented in Figures 4.9(a) and 4.9(b) for the spiral path. Figures 4.7(c) and 4.9(c) show how the obtained trajectory deviates from the preprogrammed one in the two considered cases. Specifically 4.7(c) and 4.9(c) depict graphs of the following error function

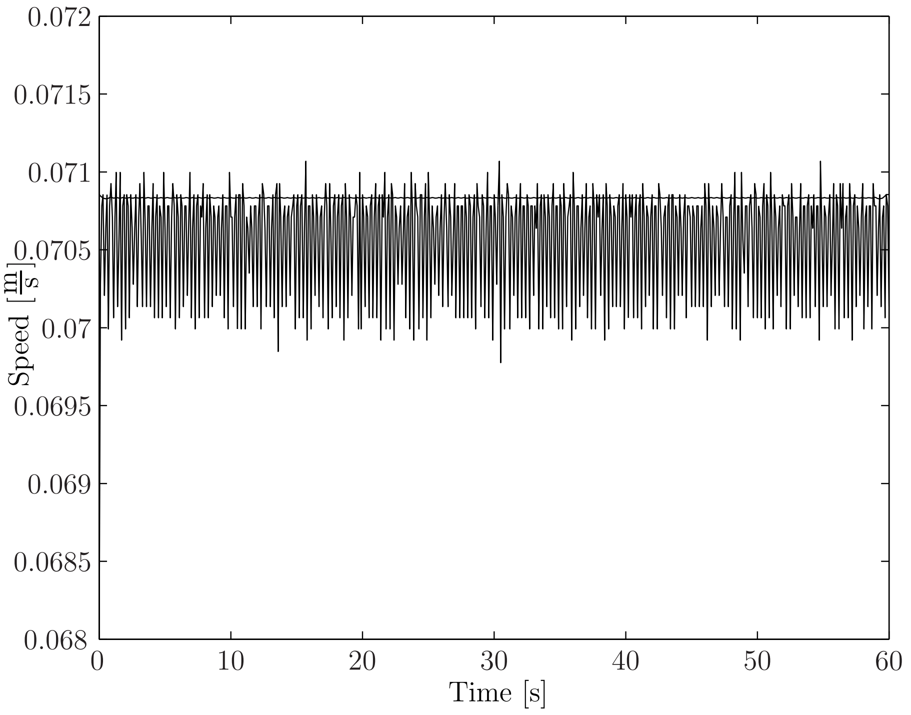
$$\epsilon(t) = \sqrt{(x(t) - \hat{x}(t))^2 + (y(t) - \hat{y}(t))^2}, \tag{4.29}$$

where $(x(t), y(t))$ are the coordinates of the ideal path and $(\hat{x}(t), \hat{y}(t))$ are the coordinates of the trajectory estimated from the hardware readings as described above.

From Figures 4.7 and 4.9, one can see that the robot has successfully completed the preprogrammed paths in the circular and spiral cases. The error may be explained by the internal signal noise due to hardware imperfection. It is also important to emphasize that numerical differentiation used in (4.28), (4.25), and (4.26) contribute significantly to the general error [56]. Figure 4.8 describes the estimated and projected linear velocities of the wheels of the robot for the spiral trajectory.

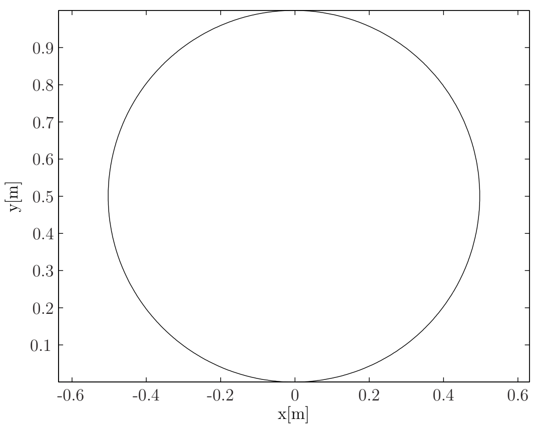


(a) Estimated $\hat{V}_L(t)$ and projected $V_L(t)$ velocities of the **left** wheel.

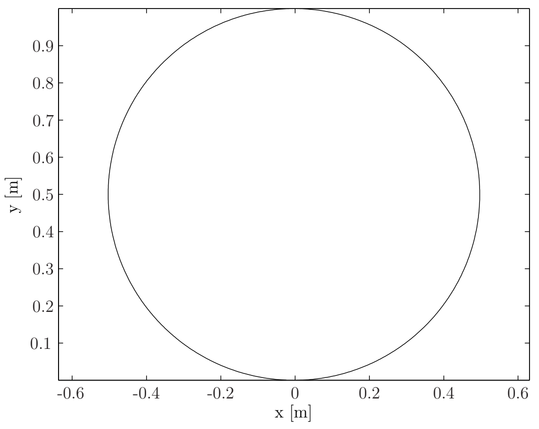


(b) Estimated $\hat{V}_R(t)$ and projected $V_R(t)$ velocities of the **right** wheel.

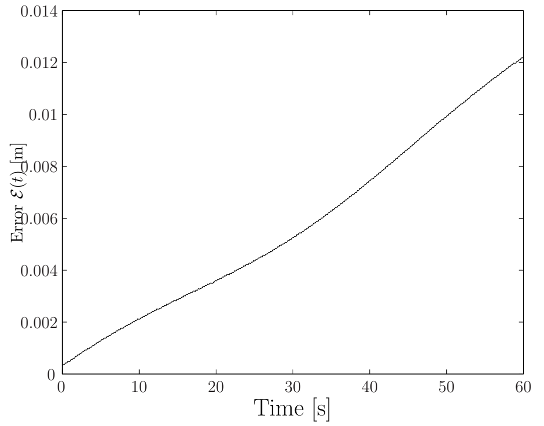
Figure 4.6: Estimated and projected linear velocities of the wheels for a **circular** trajectory.



(a) Programmed circular path.

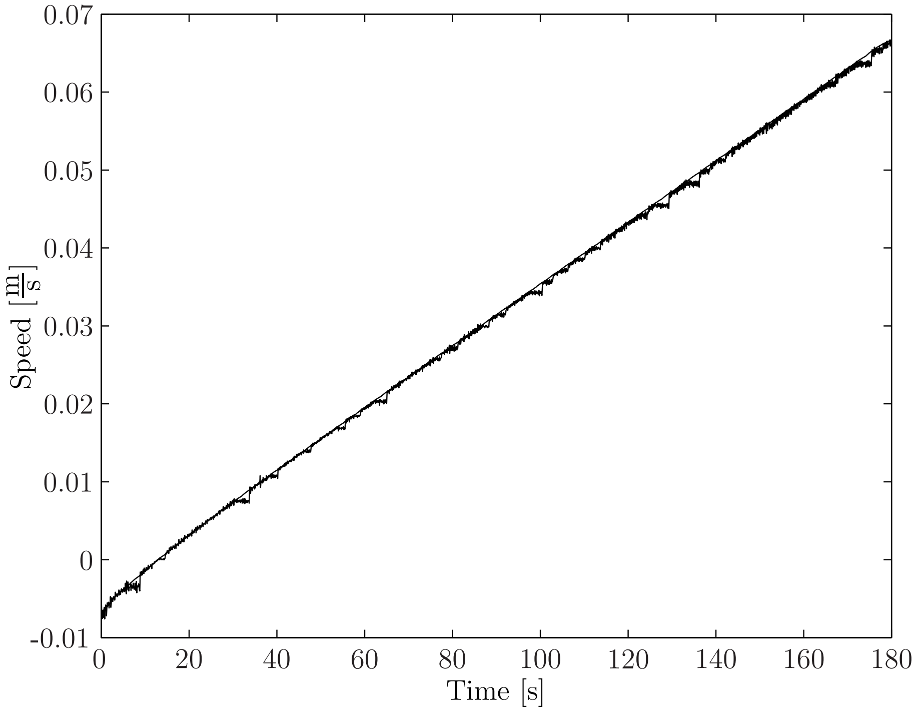


(b) Circular path estimated using moder readings

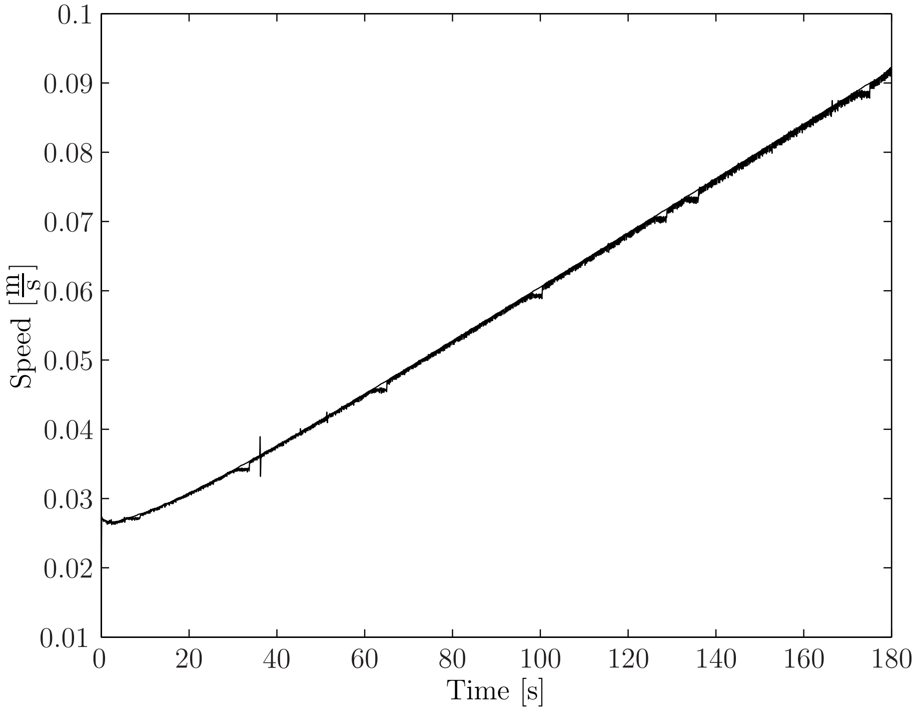


(c) Error: $\mathcal{E}(t)$ See (4.29).

Figure 4.7: Circular trajectory

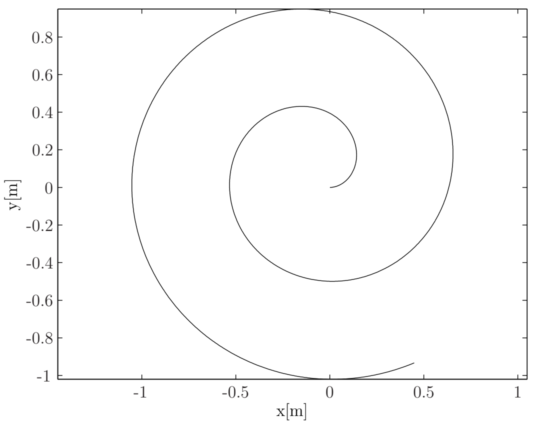


(a) Estimated $\hat{V}_L(t)$ and projected $V_L(t)$ velocities of the **left** wheel.

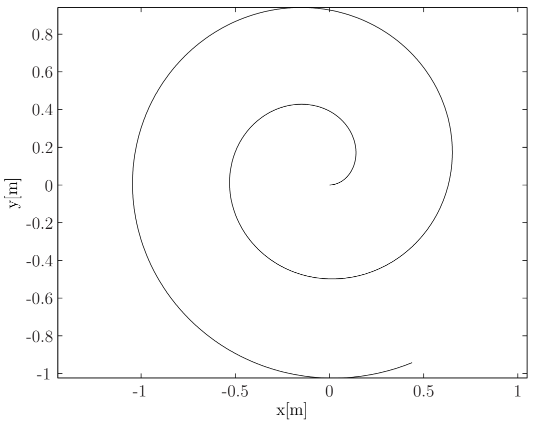


(b) Estimated $\hat{V}_R(t)$ and projected $V_R(t)$ velocities of the **right** wheel.

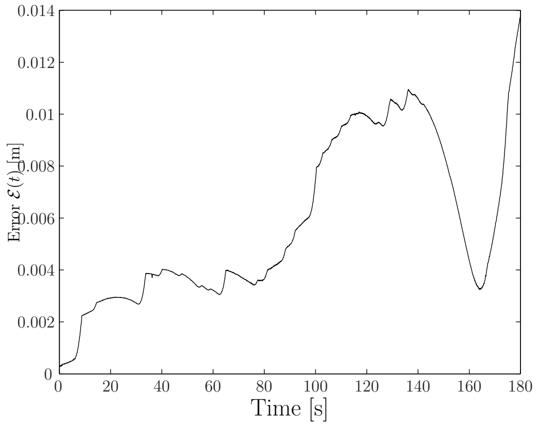
Figure 4.8: Estimated and projected linear velocities of the wheels for a **spiral** trajectory.



(a) Programmed spiral path.



(b) Spiral path estimated using moder readings.



(c) Error: $\mathcal{E}(t)$ See (4.29).

Figure 4.9: Spiral trajectory

4.6 Conclusion

In this chapter we have developed the nonholonomic kinematics model for a differentially steered wheeled mobile robot. We have applied this model to obtain the inverse kinematics formula and algorithms needed for trajectory planning. Finally we verified the developed techniques on two simple trajectories and obtained the results that agreed with the expected data.

In the next chapter we draw conclusions and outline the directions of future work.

Chapter 5

Conclusions and Future Work

The main objective of this thesis was to evaluate the possible construction of several nonholonimic testbeds for experimental use by researchers.

In this thesis, we used a potential function for agent coordination. This approach specified the behavior of each of the agents around its fellow agents, and obstacles. We then addressed various aspects in the construction of a working, autonomous mobile robot including electrical and mechanical modules. We used the constructed mobile robot to apply the principles in control system design using system identification to obtain PID controllers. In the last part of this thesis we performed different set of path-planning configurations, which involved the development of the kinematics model of the testbed, obtaining the inverse kinematics formula, and finally the algorithms needed for trajectory planning.

As described in Chapter 2, Section 2.7 there are no theoretical results available in the case of the controller input provided by (2.22). Therefore our future effort should be aimed no obtaining theoretical results in the case of presence of obstacles and a virtual leader.

Appendix A

Matlab Scripts

A.1 MatLab simulation codes.

```
%-----------------------------------------------------------------
function rr(fname);
t_max=100;  %End time
V_abs=1;    % Virtual leader speed
V_dir=pi/4; % Virtual leader direction
VL=polar(V_abs,V_dir); % Transfere to cartesians
k=[0.01 1 0.1/10*100 0.01];   % Controlling coefficiants
O  = [     3     3.1   1.5 ];          %obstacles and their radii
v0 = [1    1     1 ];
th = [pi/4 pi/4 pi/4];
r  = [0 0; 1     0;   0.5 1];
V= polar(v0,th);  % Transform velocities to cartesian
D=metrics(r)*2.5;
z0=data2Z(r,V);   % Setup  IC
rhs=@(t,Z)getRHS(Z,VL,k,D,O); % Define RHS of the system of ODEs
[T,Z]=ode45(rhs,[0 t_max],z0); % Run an SODE solver
save(fname); % Save obtained data
%-----------------------------------------------------------------
```

```
%------------------------------------------------------------------------
function Z=getRHS(z,VL,K,D,O); % Defines the Righthandside of the main SODE
[r,V]=z2data(z); % Transform Z into human readable data

M=size(O);
m=M(1);          % Get number of obstcikles
n=length(r);     % Number of agents

h=1e-4;          % Gradient step
```

```
V_tot=sum(V); %Add all velocities

for k=1:n;    %Determining control for each agent
    u1=-K(1)*(V_tot-V(k,:));   % Average direction
    u2=-K(2)*minpot(k,r,D,n);  % Keeping foramtion
    u3=-K(3)*(V(k,:)-VL);      % Following a virtual leader
    u4=-K(4)*obst(k,r,O,m);    % Keeping off obstickles

    u(k,:)=u1+u2+u3+u4;        % Total control
end;

Z(1     : n   )= V(:,1);  % Forming Z-vector (Human2Machine)
Z(n+1   : 2*n )= V(:,2);
Z(2*n+1 : 3*n )= u(:,1);
Z(3*n+1 : 4*n )= u(:,2);
Z=Z';
%---------------------------------------------------------------------------


%-----------------------------------------------------------------------------
function Z=data2z(r,V); % Human2Machine converter
%Z=[r1(1) r2(1) r3(1) r1(2) r2(2) r3(2) v1(1) v2(1) v3(1) v1(2) v2(2) v3(2)];
DIM=size(r);
n=DIM(1);
Z(1     : n)   = r(:,1);
Z(n+1   : 2*n) = r(:,2);
Z(2*n+1 : 3*n) = V(:,1);
Z(3*n+1 : 4*n) = V(:,2);
Z=Z';
%-----------------------------------------------------------------------------




%------------------------------------
% Obtains D-matric for a current configuration
function D=metrics(r);
N=size(r);
n=N(1);
for i=1:n;
    for j=1:n;
        D(i,j)=norm(r(i,:)-r(j,:));
    end;
    D(i,i)=1;
end;
%------------------------------------


%------------------------------------------------------------------
function u=minpot(i,r,D,n);
h=1e-5;
f = @(s)1/s^2+2*log(s); %potential fucntion for keeping a formation
U = @(a,d)f(norm(a)/d); %potential fucntion vector adopted

u=[0 0];
```

```
for j=1:n;
    a=r(i,:)-r(j,:);
    dx=U(a+h*[1 0],D(i,j))-U(a-h*[1 0],D(i,j));
    dy=U(a+h*[0 1],D(i,j))-U(a-h*[0 1],D(i,j));
    u=u+(i~=j)*[dx dy];
end;
u=u/h/2; %compute the contro lsignal
%-------------------------------------------------------------------


%-------------------------------------------------------------------------
function u=obst(i,r,O,m);

h=1e-5;
f = @(s)(1/s^2+2*log(s))*(1-heaviside(1-s)); % potential fnciton (obst.)
U = @(a,d)f(norm(a)/d);                       % vector adapted

u=[0 0];
for j=1:m;
    a=r(i,:)-O(j,1:2);
    R=O(j,3);
    dx=U(a+h*[1 0],R)-U(a-h*[1 0],R);
    dy=U(a+h*[0 1],R)-U(a-h*[0 1],R);
    u=u+[dx dy];
end;
u=u/h/2; %compute the control signal
%--------------------------------------------------------------------------


%------------------------------------------------------------
% This script allows to visualize the dynamica of the flock
function go(fname,dt);
close all;
load(fname);

N=size(Z);
n=N(2)/4;
TN=N(1);
M=size(O);
m=M(1);

for i=1:n;
    RX{i}=Z(:,i);
    RY{i}=Z(:,n+i);

    MINR(i)=min([RX{i}' RY{i}']);
    MAXR(i)=max([RX{i}' RY{i}']);

    VX{i}=Z(:,2*n+i);
    VX{i}=Z(:,3*n+i);
end;

minxy=min(MINR);
maxxy=min(MAXR);

figure;
hold on;
```

```
for i=1:n;
    plot(RX{i},RY{i},'k:');
end;

for j=1:m;
    plot(O(j,1),O(j,2),'r*');
end;
title(sprintf('Trajectories and obsteckels:
    k_1=%g(Dir.); k_2=%g(Dist.); k_3=%g(V.Leader); k_4=%g(Obst.)',...
    k(1),k(2),k(3),k(4)));

for i=1:n;
    text(RX{i}(1),RY{i}(1),sprintf('R_%d',i),'color','b');
    text(RX{i}(TN),RY{i}(TN),sprintf('R_%d',i),'color','b');
end;

figure;
%pause;

for t=1:dt:TN;
    for i=1:n;
        plot(RX{i},RY{i},'k:');
        hold on;
    end;
    for j=1:m;
        plot(O(j,1),O(j,2),'r*');
    end;
    title(sprintf('Trajectories and obsteckels: k_1=%g(Dir.); k_2=%g(Dist.);
        k_3=%g(V.Leader); k_4=%g(Obst.)',k(1),k(2),k(3),k(4)));

    for i=1:n;
        text(RX{i}(1),RY{i}(1),sprintf('R_%d',i),'color','g');
        text(RX{i}(TN),RY{i}(TN),sprintf('R_%d',i),'color','g');
    end;
    for i=1:n;
        X(i)=RX{i}(t);
        Y(i)=RY{i}(t);
    end;
        plot(X,Y,'b.');
        axis([minxy,1.5*maxxy,minxy,1.5*maxxy]);

    pause(0.01);
    hold off;
end;
%------------------------------------------------------------
```

Appendix B

System Identification and PID

B.1 Introduction

To design a controller for a dynamic system it is necessary to have a model that describes the dynamics of the system. There are fundamentally two approaches to finding this model:

1. Analytical Modeling uses basic principles such as the laws of motion, electrical laws, and other physical principles. The designer obtains differential equations of motion describing the response of mechanical, electrical, thermal, fluid, and other systems.

2. Experimental Identification excites the dynamic system with a given input, and records its output, then construct a model from this observed data.

While analytical modeling is almost always of some use, if for nothing more than to give insight into the expected model structure, actual physical phenomena may be too complex to permit satisfactory description using physical principles. Under these circumstances, the designer looks to experimental data and experimental system identification.

This chapter contains a brief review of system identification using the least square method. We will make the assumption that the system to be identified is linear and time-invariant.

B.2 Model and data organization

Let us use a second-order transform-based discrete system to illustrate the procedure. Consider

$$\frac{Y(z)}{U(z)} = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{B.1}$$

Note that a feedforward term b_0 was not included in above equation.

The difference equation that corresponds to equation (B.1) is

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2), \tag{B.2}$$

now consider a sequence of input samples from $k = 0 \dots N$, given by $u(0), u(1), \dots, u(N)$. Assuming zero initial conditions on output y, the output samples are given by

$$\begin{aligned}
y(2) &= a_1 y(1) + a_2 y(0) + b_1 u(1) + b_2 u(0) \\
y(3) &= a_1 y(2) + a_2 y(1) + b_1 u(2) + b_2 u(1) \\
&\vdots \\
y(k) &= a_1 y(k-1) + a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2) \\
&\vdots \\
y(N) &= a_1 y(N-1) + a_2 y(N-2) + b_1 u(N-1) + b_2 u(N-2)
\end{aligned} \tag{B.3}$$

Equations B.3 can be arranged into a matrix form

$$\begin{bmatrix} y(2) \\ y(3) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} y(1) & y(0) & u(1) & u(0) \\ y(2) & y(1) & u(2) & u(1) \\ \vdots & \vdots & \vdots & \vdots \\ y(N-1) & y(N-2) & u(N-1) & u(N-2) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} \tag{B.4}$$

or equivalently

$$b = AX, \tag{B.5}$$

where

$$A = \begin{bmatrix} y(1) & y(0) & u(1) & u(0) \\ y(2) & y(1) & u(2) & u(1) \\ \vdots & \vdots & \vdots & \vdots \\ y(N-1) & y(N-2) & u(N-1) & u(N-2) \end{bmatrix}, x = \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix}, b = \begin{bmatrix} y(2) \\ y(3) \\ \vdots \\ y(N) \end{bmatrix} \tag{B.6}$$

Just as in (B.6), in the system identification problem matrix A is known (input u and output y are known), vector b is known (output y are known), and vector x is unknown (system parameters a_i and b_i are unknown). Thus we arrive at a system of linear algebraic equations (SLAE).

B.3 Least square Approximation

A SLAE of the form $Ax = b$ may have none, infinitely many, or a unique solution. In terms of system identification, this means that the number of data samples taken is greater than the number of model parameters.

In the case of several possible solutions, one has to identify the best fit. One approach is to choose the X that minimizes the average error over the entire set of data. If we denote by $\hat{X}$ the best estimate of the actual X, one can form the error vector for the entire data set as $\mathcal{E}$, where

$$\mathcal{E} = A\hat{X} - b. \tag{B.7}$$

There are many ways to define the average error over the entire data set, but a convenient one is the sum of squares:

$$\mathcal{E}^2 = \mathcal{E}^T\mathcal{E} = (A\hat{X} - b)^T(A\hat{X} - b). \tag{B.8}$$

The sum of squares of the error $\mathcal{E}^2$ is a scalar and it represents the target function to be minimized. Minimization of $\mathcal{E}^2$ is accomplished as follows. Multiplying out equation (B.8) yields

$$\mathcal{E}^2 = \hat{X}^T A^T A\hat{X} - 2b^T A\hat{X} + b^T b. \tag{B.9}$$

To minimize (B.8) we find critical points of $\mathcal{E}^2$ by equating its gradient to 0:

$$\triangledown\mathcal{E}^2 = 2\hat{X}A^T A - 2b^T A = 0 \tag{B.10}$$

thus

$$\hat{X}^T A^T A = b^T A \tag{B.11}$$

and

$$A^T A\hat{X} = A^T b \tag{B.12}$$

and we arrive at the optimal value of $\hat{X}$:

$$\hat{X} = \left[A^T A\right]^{-1} A^T b. \tag{B.13}$$

B.4 Application to System Identification

The result of the least square approximation of the Section B.3 may be directly applied to system identification. Consider the case represented by equation (B.4), but generalized for an n^{th} order system. Let a data set consist of $N+1$ measurements $K = 0 \ldots N$. Define the following matrices, according to equation (B.4):

$$Y = \begin{bmatrix} y(n) \\ y(n+1) \\ \vdots \\ y(N) \end{bmatrix} \tag{B.14}$$

which is an $(N-n+1) \times 1$ matrix, where n is the order of the model considered.

$$\Psi = \begin{bmatrix} y(n-1) & y(n-2) & \cdots & y(0) & u(n-1) & u(n-2) & \cdots & u(0) \\ y(n) & y(n-1) & \cdots & y(1) & u(n) & u(n-1) & \cdots & u(1) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ y(N-1) & y(N-2) & \cdots & y(N-n) & u(N-1) & u(N-2) & \cdots & u(N-n) \end{bmatrix} \tag{B.15}$$

$$\Theta = \begin{bmatrix} a_1 & a_2 & \cdots & a_n & b_1 & b_2 & \cdots & b_n \end{bmatrix}^T \tag{B.16}$$

Using (B.14),(B.15), and (B.16), the equation describing the input-output data and model parameters can be written in a compact form as

$$Y = \Psi\Theta, \tag{B.17}$$

which is the standard form $Ax = b$ considered in Section B.3. According to B.3 the optimal vector of parameters Θ^* that represents the best fit in the least-square sense can be found using (B.13):

$$\Theta^* = \left[\Psi^T\Psi\right]^{-1}\Psi^T Y. \tag{B.18}$$

It is important underline that the order of the model is chosen based on the designer's understanding of the system. The selection of the order is based on physical reasoning. Often multiple fits are needed to choose the lowest acceptable order of the model. Selecting a model with an order too-high may result in extra parameter fitting.

Least square is the simplest and most straightforward identification method. If the system to be identified exhibits linear behavior and the input/output data is relatively noise-free, least square will yield acceptable results. There are other sophisticated methods that work better than least square under more difficult conditions, but they are beyond our scope.

B.5 Selection of inputs

The choice of input u used in identification is important. Consider (B.3) in a trivial case where the input $u(k)$ is a constant. In particular, consider the input term involving $u(k)$ and b_i. Then in each equation in (B.3) the b_i terms would be multiplied by the same number, and would be combined into a single quantity. The b_i terms could never be separated from the observed data. The constant u fails to excite all the dynamics of the plant. This problem has been studied extensively, and an input sequence $\{u(k)\}$ that fluctuates enough to avoid the possibility of linear combinations of the elements of model parameter vector Θ showing up in the error is called persistently exciting.

Two types of input used in the practical application considered for this theism are: (Section 3.9)

- **Random Input.** This is just a random noise, which may be easily generated using Matlab. The harmonic content in this case will be rich enough to be persistently exciting. Either a normal or uniform distributions may be used.
- **Random Binary Input.** This is a random input that fluctuates between two distinct values. It is a series of pulses with random duration. Since a step signal has infinite harmonic content, this type of input will also persistently excite the system.

B.6 PID controller design

A PID controller consists of three parts:

- **Proportional Part** - provides stability.
- **Differential Part** - enhances the stability of the system.
- **Integral Part** - improve the steady state error.

Continuous domain PID:

$$u(t) = K_p e(t) + K_1 \int_0^1 e(t)dt + K_D \frac{de(t)}{dt} \tag{B.19}$$

Discrete domain PID:

		Z-tranform
Proportional Part	$u(k) = K_P e(k)$	$D(z) = K_P$
Differential Part	$u(k) = K_D[e(k) - e(k-1)]$	$D(z) = K_D(1 - z^{-1}) = K_D \frac{z-1}{z}$
Integral Part	$u(k) = u(k-1) + K_I e(k)$	$D(z) = \dfrac{K_1}{1 - z^{-1}} = \dfrac{K_1 z}{z-1}$

$$D(z) = K_P + K_D \frac{z-1}{z} + K_1 \frac{z}{z-1} = \frac{(K_P + K_D + K_I)z^2 - (K_P + 2K_D)z + K_D}{z(z-1)} \tag{B.20}$$

or by renaming parameters

$$D(z) = K \frac{z^2 - az + b}{z(z-1)} = K \frac{(z+b_1)(z+b_2)}{z(z-1)} \tag{B.21}$$

The parameters for the model considered in this thesis are:

Natural frequency:

$$\omega_n = 100\frac{\text{rad}}{\text{s}}, \tag{B.22}$$

which corresponds to

$$f_n = 15.9\text{Hz} \tag{B.23}$$

The damping ratio:

$$\xi = 0.5 \tag{B.24}$$

The denominator of transfer function $D(z)$ in Laplace domain:

$$\Phi(s) = s^2 + 2\xi\omega_n s + \omega_n^2 = s^2 + 100s + 10000 \tag{B.25}$$

For the given ξ and ω_n zeros of $\Phi(s)$ are

$$s = -50 \pm 86.6025i \tag{B.26}$$

The discrete counterpart for the given poles is at

$$z = e^{sT}, \tag{B.27}$$

where

$$T = \frac{1}{500} = 0.002\text{s} \tag{B.28}$$

Therefore

$$z_1 = 0.9813 + 0.1559i, \quad z_2 = 0.8913 - 0.1559i \tag{B.29}$$

For the closed loop system

$$\Phi = 1 + KD(z)G(Z), \tag{B.30}$$

where

$$D(z) = \frac{(z+b_1)(z+b_2)}{z(z-1)}. \tag{B.31}$$

For z_1 to be on a root locus plot the following must be satisfied:

$$\arg\left[D(z_1)G(z1)\right] = 180. \tag{B.32}$$

$$D(z_1)G(z_1) = \frac{(z_1+b_1)(z_1+b_2)}{z_1(z_1-1)} \frac{0.08936z^2+0.1428z+0.05662}{(z-0.9982)(z-0.9107)(z-0.5794)}. \tag{B.33}$$

To get rid of the undesirable pole at $z = 0.9107$, one of the zeros is placed at the same point $b_1 = -0.9107$.

$$D(z_1)G(z_2) = (-4.0970 - 4.1252i)(z_1 + b_2)(-0.4586 - 0.7968i) \tag{B.34}$$

$$D(z_1)G(z_2) = (-1.4079 + 5.1564I)(0.8913 + 0.1559i + b_2) \tag{B.35}$$

$$D(z_1)G(z_2) = (-2.0590 - 1.4079b_2) + (4.3764 + 5.1564b_2)i \tag{B.36}$$

$$\arg\left[D(z_1)G(z1)\right] = \tan^{-1}\frac{4.3764+5.1564b_2}{-2.0590-1.4079b_2} = 180. \tag{B.37}$$

$$4.3764 + 5.1564b_2 = 0 \Longrightarrow b_2 = 0.8487 \tag{B.38}$$

References

[1] Holmberg R., O. Khatib. *Development of a Holonomic Mobile Robot for Mobile Manipulation Tasks* Proceedings of the International Conference on Field and Service Robotics - FSR'99, Pattsburg, PA, Aug.1999

[2] Katsura S, K.Ohnishi. *Human Cooperative Wheelchair for Haptic Interaction based on Dual Compliance Control* IEEE Transactions on Industrial Electronics, Vol.51, No.1, Febr., 2004.

[3] Coelho P. U. Nunes. *Path-following Control of Mobile Robots in Presence of Uncertainties* IEEE Transactions On Robotics, Vol.21, No.2, April., 2005.

[4] J. Khoury, J. Crichigno, H. Jerez, C. Abdallah, W. Shu, and G. Heileman, *The InterMesh Network Architecture*, submitted to IEEE Network Magazine, April 2007

[5] T. Stentz and P. Rander, *Integrated air/ground vehicle system for semi-autonomous off-road navigation,* in AUVSI Symposium, Orlando, Florida, July 2002

[6] D.C Mackenzie, R. C. Arkin, and J.M. Cameron, *Multiagent Mission Specification and Execution.* Boston, MA: Kluwer Academic Publisher, 2003

[7] Arkin, R. *"Behavior-Based Robotics,"* MIT Press, 1998

[8] Balch, T., and Arkin, R., *Behavior-Based Formation Control for Multi-robot Teams,* IEEE Transactions on Robotics and Automation, Volume XX, Number Y, 1999.

[9] A.Regmi, R.Sandoval, R.Byrne, H.Tanner, and C.T.Abdallah *Experimental Implementation of Flocking Algorithms in Wheeled Mobile Robots* American Control Conference. Portland, OR, USAm June 8-10-2005.

[10] H.G. Tanner, A. Jadbabaie and G.J. Pappas. *Stable flocking of mobile agents, part I: Fixed Topology,* IEEE Conference on Decision and Control. Maui Hawaii, Vol. 2, pp. 2010-2015, December 2003.

[11] H.G. Tanner, A. Jadbabaie and G.J. Pappas. *Stable flocking of mobile agents, part II: Dynamic Topology,* IEEE Conference on Decision and Control. Maui Hawaii, Vol. 2, pp. 2016-2021, December 2003.

[12] H.G. Tanner, A. Jadbabaie and G.J. Pappas. *Flocking with obstacle Avoidance in Switching Networks of Interconnected Vehicles.* IEEE International Conference on Robotics and Automation, Vol. 3, pp.3006-3011, April 1, 2004 - May 1, LA, USA

[13] R.Sandoval-Rodriguez, C.T. Abdallah, P.F. Hokayem, *Internet-like Protocols for the Control and Coordination of Multiple Agents with Time Delay.* IEEE International Symposium of Intelligent Control, Houston, TX, Oct. 2003

[14] R. Olfati. *Flocking for multi-agent dynamic systems: Algorithms and theory.* Technical Report CIT-CDS 2004-005 http://ieeexplore.ieee.org/iel5/9/33736/016055401.pdf

[15] N.E.Leonard and E.Fiorelli. *Virtual leaders, artificial potentials, and coordinated control of groups.* IEEE Conference on Decision and Contol, Orlando, FL, Vol.3, pp2968-2973, December 2001

[16] E. W. Justh and P. S. Krishnaprasad. Institute of System research Technical Report 2002-38, 2002.

[17] P. Ögren, M. Egerstedt and X.Hu, *A control Lyapunov function approach to multi-agent coordination,* IEEE Transactions on Robotics and Automation, Vol. 18, No.5, pp. 847-851, October 2002

[18] J. Cortes, S. Martinez, T.Karatas and F.Bullo. *Coverage control for mobile sensing networks.* IEEE Transactions on Robotics and Automation, Vol.20, No.2, pp.243-255, April 2004.

[19] P. Ögren, E. Fiorelli and N. E. Leonard. *Formations with a mission: stable coordination of vehicle group maneuvers.* Proc. 15th International Symposium on Mathematical Theory of Networks and Systems, August 2002.

[20] J. R. T. Lawton, B. J. Young and R. W. Beard, *Decentralized approach to elementary formation maneuvers* IEEE Transactions on Robotics and Automation, Vol. 19, No. 6, pp.933-941, December 2003.

[21] J. H. Reif and H. Wang. *Social potential fields: A distributed behavior control for autonomous robots.* Robotics and Autonomous Systems, Vol. 27, pp. 171-194, 1999.

[22] M. J. Mataric. *Behavior-based control: examples for navigation, learning and group behavior.* Journal of Experimental and Theoretical Artificial Intelligence, Vol. 9, No. 2-3, pp.323-336, 1997.

[23] C. Reynolds. *Flocks, birds and schools: a distributed behavioral model,* Computer Graphics, Vol.21, pp.25-34, 1987.

[24] T. Vicdec, A. Czirok, E. Ben Jacob, I. Cohen, and O. Schochet. *Novel type of phase transactions in a system with self-driven particles,* Physics Review Letters, Vol.75, pp. 1226-1229, 1995.

[25] H.G.Tanner, A.Jadbabaie and G.J.Pappas, *Flocks of autonomous mobile agents.* Second Annual Symposium on Autonomous Intelligent Networks and Systems, Menlo Park CA, June 2003.

[26] A. Jadbabaie, J. Lin, and A. S. Morse, *Coordination of groups of mobile autonomous agents using nearest neighbor rules,* IEEE Transactions on Automatic Control, Vol.48, No.6, pp.988-1001, July 2002.

[27] R. Vidal, O. Shakernia, and S. Sastry, *Formation control of nonholonomic mobile robots with omnidirectional visual servoing and motion segmentation,* IEEE International Conference on Robotics and Automation, Vol. 1, pp.584-589, September 2003.

[28] V. Gazi and K. M. Passino. *Stability analysis of swarms.* IEEE Transactions on Automatic Control, Vol.48, No.4, pp.692, April 2003.

[29] J. Fredlund and M. J. Mataric. *A general algorithm for robot formations using local sensing and minimal communications,* IEEE Transactions on Robotics and Automation, Vol.18, No.5, pp.837-846, October 2002.

[30] R. W. Beard, J. Lawton, and F. Y. Hadaegh. *A coordination architecture for spacecraft formation.* IEEE Transactions on Control Systems Technology, 9: pp.777-790, 2001. Available at http://www.ee.byu.edu/beard/papers/cst99.ps

[31] L. J. Corwin *Multivariate Calculus* CRC Press ISBN 0824769627 pp.128-129.

[32] R.T. Jonathan, R.W. Beard and B.J. Young. *A decentralized approach to formation maneuvers.* IEEE Transactions on Robotics and Automantions, Vol. 19, pp. 933-941, 2003

[33] T.D. Barfoot and C.M. Clark. *Motion planning for formations of mobile robots.* Robotics and Autonomous Systems, Vol. 46, pp. 65-78, 2004.

[34] D. M. Stipanovica, G. Inalhana, R. Teo and C. J. Tomlina. *Decentralized overlapping control of a formation of unmanned aerial vehicles.* Automatica, Vol. 40, pp. 1285-1296, 2004.

[35] W. Ren and R.W. Beard. *Formation feedback control for multiple spacecraft via virtual structures.* IEEE Proceedings Control Theory and Applications, Vol. 151, pp. 357-368, 2004.

[36] M. Mesbahi and F. Y. Hadaegh. *Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching.* AIAA J. Guidance, Control and Dynamics, vol. 24, no. 2, pp. 369-377, 2001.

[37] T. Balch and R. C. Arkin. *Behavior-based formation control for multirobot teams.* IEEE Transasctions is Robotics and Automation, Vol. 14, pp. 926-939, 1998.

[38] Q. Chen and J. Y. S. Luh. *Coordination and control of a group of small mobile robots.* in Proceedings IEEE International Conference in Robotics and Automation, pp. 2315-2320, 1994.

[39] N. E. Leonard and E. Fiorelli. *Virtual leaders, artificial potentials and coordinated control of groups* in Proceedings IEEE Conference in Decision and Control, Orlando, FL, pp. 2968-2973, 2001.

[40] M. A. Lewis and K.-H. Tan. *High precision formation control of mobile robots using virtual structures*, Autonomous Robots, Vol. 4, pp. 387-403, 1997.

[41] W. Kang and Yeh, H.-H. *Coordinated attitude control of multisatellite systems* International Journal on Robust Nonlinear Control, Vol.12, pp. 185-205, 2002.

[42] P. Ogren, Egerstedt, M., and Hu, X. *A control Lyapunov function approach to multiagent coordination*, IEEE Transactions on Robotics and Automation Vol.18, pp. 847-851, 2002.

[43] R.W. Beard, Lawton, J., and Hadaegh, F.Y. *A coordination architecture for formation control* IEEE Transactions on Control Systems and Technology, Vol. 9, pp. 777-790, 2001.

[44] E. Rimon and D.E. Koditschek. *Exact robot navigation using artificial potential functions.* IEEE Transactions on Robotics and Automation, Vol. 8, no. 5, pp. 501-518, 1992.

[45] E. Rimon and D.E. Koditschek. *Robot navigation functions on manifolds with boundary* Advances in Applied Mathematics, Vol. 11, pp. 412-442, 1990.

[46] H.G. Tanner, S.G. Loizou and K.J. Kyriakopoulos. *Nonholonomic navigation and control of multiple mobile robot manipulators.* IEEE Transactions on Robotics and Automation, vol. 19, pp. 53-64, 2003.

[47] H.G. Tanner and A. Kumar. *Towards decentralization of multi-robot navigation functions,* IEEE International Conference on Robotics and Automation, Barcelona, Spain, pp 4143-4148, 2005.

[48] H.G. Tanner and A. Kumar. *Formation stabilization of multiple agents using decentralized navigation functions.* Robotics: Science and Systems, in press, 2005.

[49] E.W. Jush and P.S. Krishnaprasad. *Equilibria and steering laws for planar formations.* Systems and Control Letters, Vol. 52, pp. 25-38, 2004.

[50] S.S. Ge and Y.J. Cui. *New potential functions for mobile robot path planning.* IEEE Transactions on Robotics and Automation, Vol. 16, pp. 615-620, 2000.

[51] S.S. Ge and Y.J. Cui. *Dynamics motion planning for mobile robots using potential field method.* Autonomous Robots, Vol. 13, pp. 207-222, 2002.

[52] K.D. Do and J. Pan. *Global path-tracking of underactuated ships with non-zero off-diagonal terms* Automatica, Vol. 41, pp. 87-95, 2005.

[53] H. Khalil. *Nonlinear systems.* Prentice Hall, Englewood Cliffs, NJ, 2000.

[54] Shan Sun Kuo, *Computer applications of numerical methods*, 341 pages, Addison-Wesley, 1965

[55] Carl De Boor, *A Practical Guide to Splines,* Springer, 2001, ISBN 0387953663

[56] A.Quarteroni, F.Saleri *Scientific Computing with MATLAB*, Springer-Verlag Berlin Heidelberg 2003