VISION BASED CONTROL ALGORITHMS FOR A SMALL SIZE ROBOT SOCCER TEAM

by

Hakan Karaoğuz

B.Sc, Electrical and Electronics Enginnering, Koç University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering

Boğaziçi University

2009

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my thesis supervisor, Assoc. Prof. Mehmet Akar, for his help. Without his knowledge and guidance, this thesis would have never been accomplished.

I want to thank my best, friend Duygu Çekiç, for her incredible support during the writing phases of the thesis. Her moral motivation always kept me fresh and determined.

Finally, I want to thank my parents, without whom it would be impossible for me to realize my achievements.

ABSTRACT

VISION BASED CONTROL ALGORITHMS FOR A SMALL SIZE ROBOT SOCCER TEAM

Multi-robot coordination is one of the hot topics of today's robotics research. In our case, our main concern is to build up a small sized (SSL) robot soccer team. Robot soccer requires effective and fast robot control and coordination in order to score goals while avoiding frequent collisions with opponent robots.

In this thesis, our multi-robot platform and current control techniques we use are discussed. Our robots have 4 wheel omnidirectional drive system, running with brushless DC motors. A camera which is 3 meters above the field provides vision feedback to identify and control the robots. Robot identification is carried out by recognizing colored patches on top of the robots. Several formation strategies are proposed using the holonomic robots we have built.

ÖZET

KÜÇÜK BOYUTLU ROBOT FUTBOLU TAKIMI İÇİN GÖRÜNTÜ BAZLI DENETİM ALGORİTMALARI

Çoklu robot eşgüdümü günümüz robot araştırmalarında önemli bir yer tutar. Bizim hedefimiz de, çok sayıda robottan oluşan bir takım oluşturarak, küçük boyutlu robot futbolu liginde yarışmalara katılmaktır. Robot futbolu, gerçek futbol kuralları ile oynanan, bu nedenle birden çok robotun verimli ve hızlı bir şekilde denetlenmesini gerektiren bir oyundur.

Bu tez çalışmasında, oluşturduğumuz deneysel platform ve kontrol tekniklerimizden bahsedilmiştir. Robotlarımız 4 tekerleklidir ve tüm yönlü hareket sistemine göre tasarlanmıştır. Hareket için fırçasız doğru akım motorları kullanılmıştır. Sahanın 3 metre üzerinde yer alan kameralar ile oyun takip edilmektedir. Robotların tanınması için, üzerlerine renkli tanımlama benekleri konulmuştur. Bu robotların eşgüdümü için geliştirilen birçok düzen denetim algoritmasından da bu tezde bahsedilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\vec{a}$	Acceleration vector
a_l	a parameter of a line
$a_{l_{ht}}$	Higher threshold of the search interval for the parameter a_l
$a_{l_{lt}}$	Lower threshold of the search interval for the parameter a_l
a_{lsm}	a parameter of a line that is obtained by the least squares solution
a_{opt}	Optimal a_l parameter
a_r	Acceleration vector of the robot
B	Parameter used to find b_{lsm}
b_l	b parameter of a line
$b_{l_{ht}}$	Higher threshold of the search interval for the parameter b_l
$b_{l_{lt}}$	Lower threshold of the search interval for the parameter b_l
b_{lsm}	b parameter of a line that is obtained by the least squares solution
b_{opt}	Optimal b_l parameter
$Center_{x_{hc}}$	x coordinate of the center of the circle that is created by the proposed circle formation algorithm
$Center_{y_{hc}}$	y coordinate of the center of the circle that is created by the proposed circle formation algorithm
$C(s)$	Transfer function of the controller
$C(z)$	Overall pulse transfer function of $C_1(z)$ and $C_2(z)$
$C_1(z)$	Pulse transfer function of the first part of the digital controller
$C_2(z)$	Pulse transfer function of the second part of the digital controller
C_i	First counter value when a Hall sensor signal is captured
C_f	Second counter value when a Hall sensor signal is captured
$CS(t)$	Output of the $C_1(t)$
c_x	x coordinate of the center of a circle
$c_{x_{ht}}$	Higher threshold of the search interval for the parameter c_x
$c_{x_{lt}}$	Lower threshold of the search interval for the parameter c_x

$c_{x_{opt}}$	optimal x coordinate of the center of a circle
c_y	y coordinate of the center of a circle
$c_{y_{ht}}$	Higher threshold of the search interval for the parameter c_y
$c_{y_{lt}}$	Lower threshold of the search interval for the parameter c_y
$c_{y_{opt}}$	optimal y coordinate of the center of a circle
D	Diameter of the circle formed by Proposed Heuristic Circle Formation algorithm
d_{c_t}	The total perpendicular distance of a set of points to a circle
$d_{line_{hl}}$	The length of the line created by the proposed line formation algorithm
d_{hc}	The distance between the robots with the maximum and minimum x coordinates
d_{l_t}	The total perpendicular distance of a set of points to a line
$d^2_{lsm_t}$	The square of the perpendicular distance of a set of points to a line
$d_{t_{hl}}$	Distance threshold
$E(s)$	The difference between the desired and the actual output of the system
e	Position error vector
e_x	x component of the position error vector
e_y	y component of the position error vector
f_c	Frequency of the counter
$\Delta\, C$	The difference between C_i and C_f
$F(s)$	Frequency of the Hall sensor signal
$F_d(s)$	Desired Hall sensor signal frequency for the continuous time systems
$f_d(s)$	Desired Hall sensor signal frequency for the discrete time system
f_{ss}	Steady state value of the continuous time system
$\vec{F}_{wn}$	Force vector of wheel n
f_{wn}	Magnitude of the force vector F_{wn}
f_w	Magnitude vector
F_z	Steady state value of the discrete time system

$G(s)$	Open loop transfer function of the overall system
$G_{act}(s)$	Open loop transfer function of the actuator
$G_{mot}(s)$	Open loop transfer function of the DC motor
$G(z)$	Pulse transfer function of the plant
$H(s)$	Closed loop transfer function of the system
$H(z)$	Closed loop pulse transfer function of the system
I	Moment of inertia
I_z	Moment of inertia in the z direction
K_p	Proportional gain for continuous time system
K_{pdis}	Proportional gain for discrete time system
K_{ddis}	Derivative gain for discrete time system
K_i	Integral gain for continuous time system
K_{act}	Steady state gain of the actuator
K_m	Steady state gain of the DC motor
K_s	Steady state gain of the DC motor, load and actuator
M	Mass of the robot
m_{hl}	Slope between the robots with maximum and minimum x co-ordinates for the proposed line formation algorithm
P_e	Period of the Hall sensor signal
P_m	Period of the motor speed
R	Radius of the robot chassis
R_c	Radius of a circle
$R_{c_{ht}}$	Higher threshold of the search interval for the parameter R_c
$R_{c_{lt}}$	Lower threshold of the search interval for the parameter R_c
$R_{c_{opt}}$	Optimal radius of a circle
R_{max}	The robot with the maximum x coordinate
R_{min}	The robot with the minimum x coordinate
R_i	Position vector of the i^{th} robot
R_z	Rotation matrix in the z direction
r	Closed loop poles of the system
S_m	Speed of the motor

$V(s)$	Voltage applied to the motor
V_{thrs}	Velocity threshold
v_r	Robot's commanded linear and angular velocity vector
v_x	Velocity vector in the x direction
v_y	Velocity vector in the $y$direction
v_{xnorm}	Normalized velocity vector in the x direction
v_{ynorm}	Normalized velocity vector in the $y$direction
v_{xn}	Final velocity vector in the x direction
v_{yn}	Final velocity vector in the y direction
v_{xrobot}	Commanded velocity vector in the x direction
v_{yrobot}	Commanded velocity vector in the y direction
w_{vn}	The velocity vector of the n^{th} wheel
w_v	The vector composed of w_{vn}s
$x_{dc,j}$	x coordinate of the desired locations of the proposed circle formation algorithm
$x_{dl,j}$	x coordinate of the desired locations of the proposed line formation algorithm
$x_{R_{max}}$	x coordinate of the R_{max}
$x_{R_{min}}$	x coordinate of the R_{min}
$y_{dc,j}$	y coordinate of the desired locations of the proposed circle formation algorithm
$y_{dl,j}$	y coordinate of the desired locations of the proposed line formation algorithm
$y_{R_{max}}$	y coordinate of the R_{max}
$y_{R_{min}}$	y coordinate of the R_{min}
z_{cls}	Closed loop zeros of the system
α_{hl}	The angle between the robots with the maximum and minimum x coordinate values for the line algorithm
α_{hc}	The angle between the robots with the maximum and minimum x coordinate values for the circle algorithm
Δt	Time interval
Δx_l	The difference between the x coordinates of the robots with maximum and minimum x coordinates

Δy_l	The difference between the y coordinates of the robots with maximum and minimum x coordinates
η	Force coupling matrix
$\gamma(s)$	PWM duty cycle
κ	Wheel velocity coupling matrix
λ	Inertia constant
ν_n	The matrix for calculating second derivative of $d^2_{lsm_t}$ with respect to the parameters a_{lsm_t} and b_{lsm_t}
Ω	Angular speed of the DC motor in rad/sec
ω_n	Natural frequency
$\dot{\omega}$	Angular acceleration of the robot
Ψ_n	Angle of the n^{th} wheel with respect to the x axis
τ_{act}	Time constant of the actuator
τ_m	Time constant of the DC motor
τ_s	Time constant of the DC motor, load and actuator
$\Theta(s)$	Angular position of the DC motor in radians
Θ_e	The angle between the desired and actual location of the robot
Υ	The orientation of the robot relative to the global coordinate system
ζ	Damping ratio

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application programming interface
BBC	Behavior Based Coordination
CM	Center of Mass
CMU	Carnegie Mellon University
FPS	Frames per Second
GPS	Global Positioning System
IC	Integrated Circuit
ID	Identification
LTI	Linear Time Invariant

N–D	N–Dimensional
OpenCV	Open Source Computer Vision Library
R&D	Research and Development
SSL	Small Size League
PWM	Pulse Width Modulation
RPM	revolutions per minute
VOMAS	Virtual Operator Multi Agent System

1. INTRODUCTION

The idea of building autonomous machines goes as old as 1206 when Al-Jazari, an ancient time scientist, described autonomous machines in his manuscript *Book of Knowledge of Ingenious Mechanical Devices*. However, fully autonomous machines did not appear until the 20^{th} century. Technological achievements in the first half of the century allowed people to build first mobile robots in 1940's. Despite its short history, huge technological advancements have been made since then. Today, human can send mobile robots to other planets for exploration and use them in hazardous environments.

However, there is much more to do in this field. Current Artificial Intelligence (AI) level is not enough for robots to work autonomously, except in some controlled environments. Efficient energy usage is also another concern, since most of the robots have limited battery. Locomotion systems are mostly based on wheels as human–like walking is very complex and unstable. Sensors are far from ideal, limiting the perception of robots.

In order to address some of these problems and encourage research in the field, competitions are being organized and Robocup is one of them. The ultimate aim of Robocup is to build a fully autonomous humanoid robot team that can win against the FIFA World Cup winner soccer team by the year 2050. The main reason for choosing soccer as the competition environment is that it addresses current problems of mobile robotics. In order to be successful in soccer, one should have very strong AI that can analyze the game and develop tactics accordingly. Robots must be able to operate continuously at least for 45 minutes before they can be recharged, therefore they should have enough power and know how to use it efficiently. Moreover, their perception should be superior in order to detect the opposing team members, the ball, the goals and own team members in a dynamically changing environment. The most important of all, robots should be able to dribble the ball, walk, run and shoot like humans without falling. When all of these can be achieved, building robots as capable

as humans will be possible. That's why, Robocup draws the attention of researchers and the public, making it the best known robotic competition. More details on Robocup are provided in the following section.

1.1. History and Organization of Robocup

The concept of soccer playing robots and Robocup was first started in 1993. After initial research and feasibility studies, the first official games were held in Nagoya, Japan in 1997 [1]. Following the success of these first games, Robocup continued to be held in different countries with increasing support and public attention. Since 2003, Turkey also participates in Robocup and will host the competition in 2011. Different leagues are available for teams to participate in. These are explained below.

1.1.1. Simulation League

In this league, software agents play soccer on a virtual field. There are no real robots.

1.1.2. Small Size League (SSL)

In this league, robots with constrained size play soccer on a field of 6.5 by 4.5 meters. Because of the dimensions of robots, the game is played with an orange golf ball instead of a real soccer ball. There can be a maximum of 5 robots from each team during the match. Robots are observed by cameras that are placed above the field. Visual feedback enables the AI algorithms to control the robots and gameplay. Commands are sent to robots using wireless trans-receivers. Detailed explanation of this league is given in Section 1.2. of this chapter.

1.1.3. Middle-Size League

In this league, robots that are significantly larger than the ones in SSL, play soccer with a real soccer ball. All of the sensors, including cameras, are on the robots and they can communicate each other via wireless network.

1.1.4. Standard Platform League

This league will undergo significant change in 2009. Until now, 4 legged Sony AIBO robots were used. Beginning with the 2009 Robocup, teams will compete using 2 legged humanoid Nao robots which will bring additional difficulties such as maintaining robot's stability while walking or shooting the ball. This is the only league in which everyone uses the same robots; therefore, teams focus only on software development.

1.1.5. Humanoid League

Custom made biped humanoid robots compete against each other in this league, that is further divided into two categories according to the size of the robots: Kid-Size and Teen-Size.

Apart from soccer, there is also another competition of Robocup, called Robocup Rescue, that serves as a test and development platform for rescue robots with two different leagues. In the first league, which is called as the Rescue Robot League, semi–autonomous robots compete in a disaster site, trying to explore the area and identify the location of victims. On the other hand, in the second league, known as the Rescue Simulation League, only virtual agents and algorithms compete in a simulated environment with no real physical robots.

In addition to the soccer competitions and the Rescue League described above, seminars and tutorials are also being held during the event. With such a comprehensive structure, Robocup serves as a major $R\&D$ platform in the field of robotics.

1.2. SSL Competition

The modular architecture of the SSL is shown in Fig. 1.1. The details of these modules are given below.

1.2.1. Robots

As mentioned before, robots are custom made and their dimensions are constrained. The robots' height should not exceed 15 cm and should fit in a sphere with 18 cm radius. Teams have different approaches for the robot design, but nearly all of them use the omni–directional drive system which is the most efficient one for this league. Either 3 or 4 wheeled omni–drive configurations are used. Fig. 1.2 shows

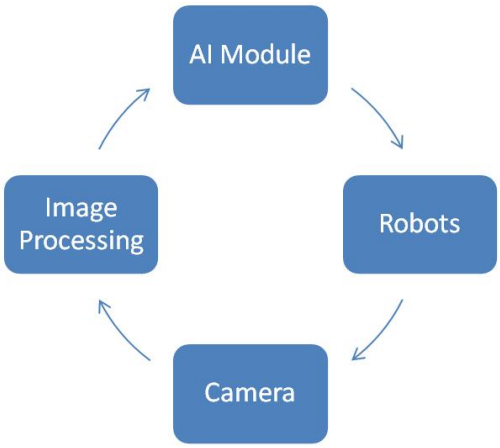

Figure 1.1. Modular architecture of SSL.

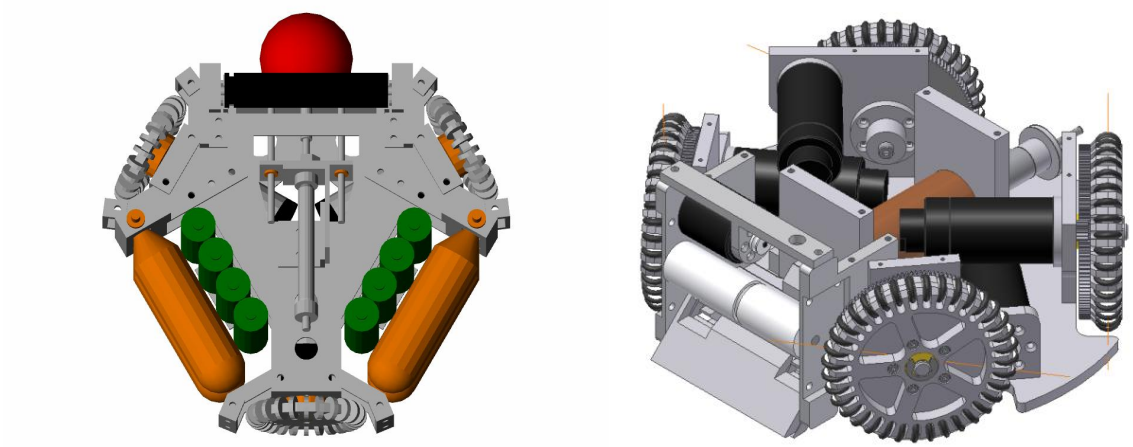

Figure 1.2. 3 and 4 wheeled omni–drive configurations [3]

the designs of 3 and 4 wheeled configurations [2], [3]. Our team has chosen the 4 wheeled configuration, as it has certain performance advantages over 3 wheeled one in exchange for increased complexity and power requirement. What differentiates between the robots are the motors, dribbling systems, electronic control units, mechanical design and kicking mechanisms. A typical SSL robot has a top speed of 2.5–3 m/s and it can accelerate at a rate of 9-10 m/s^2.

1.2.2. Camera

Cameras are used to detect the position of the ball, own and opponent robots. High frame rate ($\geq$ 60 fps) is required for fast object tracking. Because of the size of the field, 2 cameras have to be used and they should be hanged 4 meters above. Wide angle lenses are utilized with the cameras in order to cover the whole area. These lenses have certain disadvantage as they cause distortion on the image. Image processing software should handle this distortion in order to detect the position of the robots and the ball correctly.

1.2.3. Image Processing

The Image Processing module is used for detecting the ball and the individual locations of the robots. In order to achieve this task, colored patches are placed on top of the robots (see Fig. 1.3) [4]. Each robot has a unique color pattern so that all of the robots can be identified. The speed of this module is vital to detect the positions of the robots and the ball with minimum latency.

1.2.4. AI Module

This module is responsible for controlling the overall game strategy. According to the positions of own robots, the ball and the opponent robots, it sends necessary commands to the robots via wireless transmission. As the environment in this league is highly dynamic with game conditions changing rapidly, it is crucial for the AI module

Figure 1.3. SSL Robot with Color ID on top.

to compute and send commands as fast as possible.

1.3. Motivation and Scope of the Thesis

The motivation of this thesis is to develop high level formation control algorithms for our SSL team. Moreover, we want to develop a testbed for the SSL that can be also used to implement our proposed formation control algorithms.

This thesis has 6 chapters. Chapter 1 is the introduction that gives background information about the Robocup and SSL competition.

The literature review about the formation control of robots is provided in Chapter 2. The 3 main approaches to the formation control problem are analyzed in detail.

In Chapter 3, the Low Level Control Module of our SSL architecture is discussed. The most important part of this module is the motor speed control; therefore, the design procedure for the digital speed controller is mainly discussed in this chapter.

The Image Processing Module is presented in Chapter 4. The image processing and object recognition steps are explained in detail. A brief discussion about the object tracking and Kalman filter is also provided in this chapter.

In chapter 5, High Level Control (AI Module) of our SSL architecture is explained in detail. In the first section of this chapter, the derivation of velocity vectors for sending a robot to a desired location is given. In the next section, the steps taken for transforming these velocity vectors into individual velocity vectors are explained in detail. Finally, the proposed formation control algorithms for line and circle formations are discussed and their performance is compared with other formation methods.

Chapter 6 provides the conclusions part of the thesis.

2. LITERATURE REVIEW

As previously mentioned, the main aim of this thesis is to develop algorithms for the formation control of our SSL team; therefore we have focused my literature review on this topic. Several sources can be found for the other chapters such as [5], [6], [7] for low level control and [8], [9], [10] for image processing. [11] is also a very useful source for understanding the overall architecture.

2.1. Formation Control

A lot of research has been going on in the area of multi-robot formation control [12]–[24], that can be grouped into two according to the number of robots involved. In the first group, multi-robot teams have limited number of robots (i.e, no. of robots ≤ 5) while in the second one, research is focused on robot swarms which are composed of tens or hundreds of identical robots. As our team is composed of 5 robots at maximum, we have focused on literature for the first group. For the details of robot swarm control, one can refer to [20] and the references there in.

As mentioned in [18] and [24], there are mainly 3 approaches for formation control of robots: Leader follower [12, 13, 14, 15], behavior based coordination (BBC) [16, 17, 19] and virtual structure [22, 24]. There are also some hybrid structures which combine these approaches as in [21]. Detailed discussion on these topics is below.

2.1.1. Leader Follower Approach

In the Leader Follower Approach, a robot is chosen as the leader and the others follow the leader in a formation. The robots can be heterogeneous or homogenous, i.e., the leader robot can be more advanced than the followers or all of the robots can be identical.

In their work [14], Huang et. al. use a superior leader robot with complex sensing and computational abilities that leads the simpler follower robots. Their aim is to replace the standard traffic barrels with the robotic ones that can self deploy and retrieve so that risks in work zones on highways or other public places can be decreased. According to [14], decentralizing the architecture increases system costs and makes it harder to coordinate the group. By using the leader follower approach, the overall system costs can be reduced and it would be easier to coordinate the group.

In their proposed architecture, the leader tracks the follower robots' positions via its laser sensor. For estimating the followers' orientations, the leader first fits a cubic polynomial to approximate followers' paths. By using the derivative of this polynomial and the inverse tangent function, the leader can estimate the orientation of the follower. In order to decrease the tracking and orientation estimation errors, Huang et al. have set certain distance and turning radius thresholds which also limit the maneuvering ability of the group. Another fact is that, single leader architectures such as this one, can easily fail if the leader robot fails [15]. This actually conflicts with the idea of reducing risks in work zones since a group of robotic barrels with a failed leader can cause severe accidents.

In order to increase the robustness of the leader follower approach, architectures with more than one leader has been introduced like the one in [15]. In this approach, several robots can be assigned as leaders; therefore, if one fails, others can take control of the group. Only the leaders know the desired formation state, but the follower robots communicate with their neighbors in order to estimate their formation state and tracking errors. The control law for each vehicle is based on the so–called consensus algorithm to drive each vehicle's formation state to the desired one [15].

Multiple leader structure makes the leader follower approach more decentralized compared to the classical leader follower structures such as [14]. Inter-vehicle communication feature is also a plus for the overall robustness of the system. Moreover, experimental results are provided in order to confirm that the given control laws can

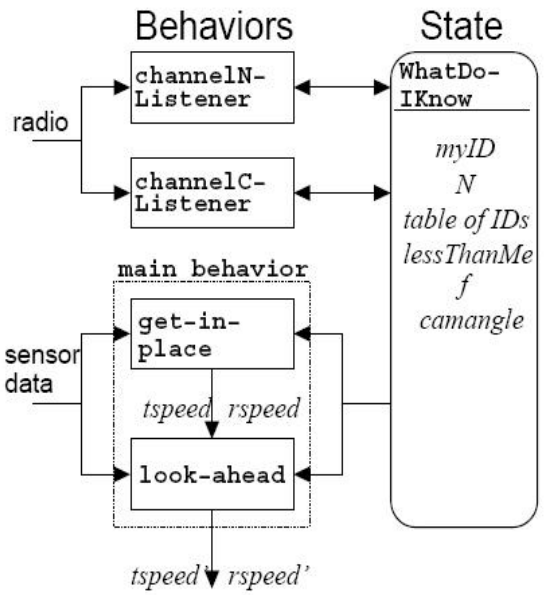

Figure 2.1. An example of a BBC structure [19]

be applied to the physical robots.

2.1.2. Behavior Based Coordination

In BBC, several pre-determined behaviors are generated and the final movement of the agent is derived from the weighted union of these behaviors. In [17] for example, two behaviors compete during the formation maneuver. These are, moving to the final destination and maintaining the formation during movement. The relative importance of these behaviors are determined by associated weight matrices. Similarly in [19], a main behavior initially sets the robot's translational and rotational velocity according to the current perception (see Fig.2.1). Sub-behaviors such as obstacle-avoidance and get-in-place modify these speeds according to the current state of the robot.

The main advantage of BBC is its decentralized architecture which increases its robustness in case of a failure of a robot [17, 19]. Furthermore, its minimal communication requirements simplify the overall system design and increases the robustness. However, because of being an algorithmic approach rather than a mathematical one, there is no guarantee that the desired formation will be established. Moreover, all the robots have to be identical and should be equipped with complex sensors in order to

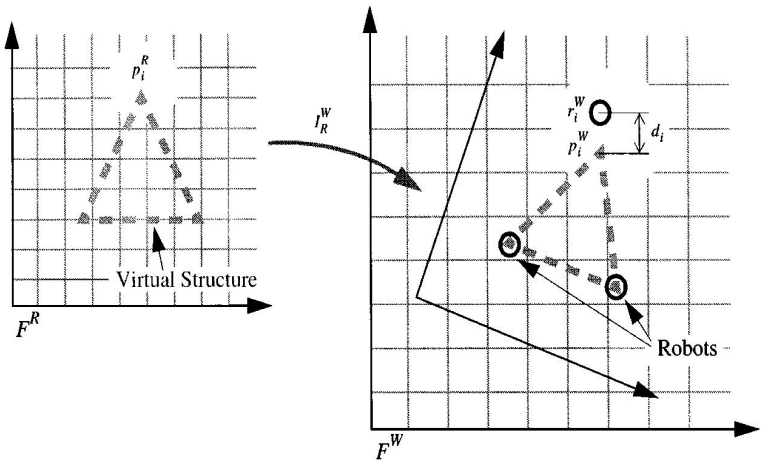

Figure 2.2. An example of a virtual structure which is shown in its own frame of reference (left) and in global frame of reference (right) [22]

successfully achieve the desired behaviors [14].

2.1.3. Virtual Structure

Virtual Structure approach treats the entire formation as a single entity [17], [24]. The robots are viewed as particles which are embedded in a structure [22]. This structure which is composed of robots that maintain certain geometric relationships is called the virtual structure (see Fig. 2.2). In [22], Lewis and Tan propose a 4–step algorithm in order to keep this virtual structure while moving it around. Basically, the algorithm fits an initial structure for the robots on the field and moves it in an arbitrary direction and orientation. New destination points are calculated according to the new position of the structure and the individual robot trajectories are computed. In the final step, angular and linear velocities are calculated and sent to the robots in order to follow these trajectories. Lewis and Tan claim that, this approach can be used in many platforms such as aircrafts or spacecrafts which operate in 3–D space instead of mobile robots that can move only in 2–D plane (see [23]). However, drawbacks discussed below can actually prevent the use of this approach in the coordination of aircrafts and spacecrafts.

In order to maintain the formation, Lewis and Tan develop a control scheme

which takes the physical and maneuvering abilities of the robots into account. In this scheme, the virtual structure is displaced according to the positions that the robots can reach at each time step. These positions are called the *reachable positions* and they are calculated according to the acceleration limits of the robots. In addition, Lewis and Tan create cost functions for measuring the "goodness" of initially *fitted structure* and *displaced structure*, because it is not always possible to fit an initial structure perfectly or displace it within the reachable positions of each robot. The control task then becomes minimizing these cost functions. In order to validate their proposed control scheme, Lewis and Tan provides experimental and simulation results.

The main advantage of this approach is, in case of a failure in one of the robots, the others adjust their positions to maintain the formation, therefore it is a more robust approach compared to the classical Leader Follower strategy [22]. Another advantage is its flexibility, i.e., it is possible to add more robots to the formation by just creating a new virtual structure.

The main disadvantage of this approach, is that, it is a centralized architecture that requires an outer central controller communicating with each robot. This can indeed decrease the robustness in case of a communication failure with one of the robots. Moreover, it is necessary to know the exact positions and orientations of all of the robots in order to apply the suggested control schemes. For planar robots, this can only be accomplished by using overhead cameras, therefore the method is only applicable in a certain area which is bounded by the cameras' field of view. In aircrafts, a GPS system can be used for determining the exact positions, but its accuracy can dramatically affect the performance of the control scheme. On the other hand, spacecrafts, it is even harder to determine their exact positions and track them accurately in deep space.

2.1.4. Hybrid Architectures

Hybrid Architectures are formed by merging the above strategies. This is usually done in order to combine the advantages of different approaches so that the success

rate of the overall architecture can be increased. In exchange, the complexity of the system increases and the synchronization of these architectures can be problematic. One such architecture, named as VOMAS, is introduced in [21] and it is a combination of BBC and the leader follower approach.

In VOMAS, there are two operators which are responsible for different levels of control. Virtual operator is the one that is responsible for the high level control of the entire group. It uses high level behaviors such as merging and splitting the group or adding a new member in order to perform necessary control actions. The low level controller, called the robot operator is responsible only for the robot it is assigned to. Its main duty is to bring its robot to the desired position in the formation. For this purpose, a leader follower control scheme is used.

First of all, numbers are assigned to the robots for identification. For a group composed of N robots, number 1 always identifies the group leader. IDs of the remaining robots are given as $ID_i = i$, $i = 2, 3.....N-1$. The formation is formed according to the leader's initial position. Starting with the followers that are neighbor to the leader, all the followers adjust their positions according to their preceding neighbors. This chain like structure allows the entire group to form the desired formation pattern. After the desired formation is established, the low level operators switch to the BBC in order to maintain formation while the group is on the move. All of these behaviors are applied to the robots by using potential field techniques.

Because it is made up of classical leader follower structure, this architecture is prone to the same weaknesses, such as single point failure. Also, the overall structure is complex and it can be computationally expensive for a real world application. On the other hand, it combines the strength of two different architectures which makes it possible to achieve more advanced tasks.

3. LOW LEVEL CONTROL

Our low level control architecture is onboard the robot and its schematic is shown below in Fig. 3.1.

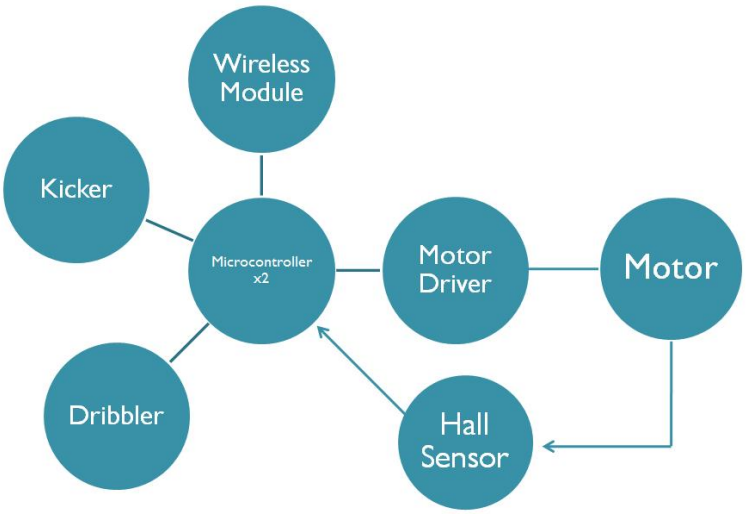

Figure 3.1. The schematic of our low level control architecture

The primary task of the low level control unit is to control the motor speeds. The components of the Low Level Control Module are given in Table 3.1.

Table 3.1. Components used in the Low Level Control Module

2 Microprocessors
1 Wireless Zigbee trans-receiver module
4 Brushless DC motors with Hall sensors
1 Brushed DC motor with gear system for the dribbler
Voltage booster and charge pump circuits for the kicker

The desired motor speeds are sent to the robot via wireless Zigbee trans-receiver module from the remote PC. Microprocessors get the motor speed data from the Zigbee trans-receiver module onboard and activate the speed control loop.

3.1. Motors

We use Maxon EC-45 Flat 30W Brushless DC Motors for the locomotion of our robots. The main reason for choosing this type of motor is that, its small size allows us to use limited space more efficiently. Fig. 3.2 shows the motors attached to the robot chassis and Fig. 3.3 depicts the internal structure of the motor. Our motors work with

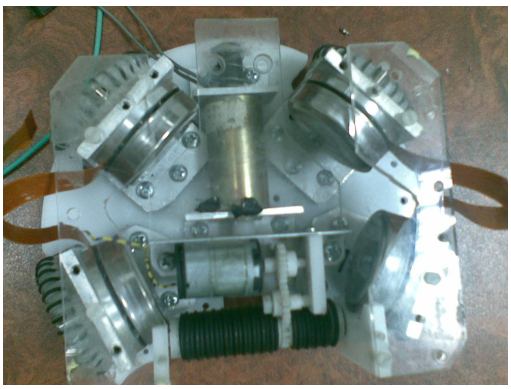

Figure 3.2. The chassis of our prototype

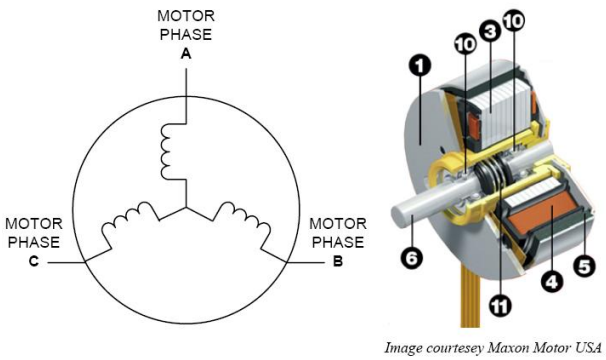

Figure 3.3. The internal structure of the motor [25]

12V, at a maximum speed of 4400 rpm and can produce 59 mNm continuous nominal torque. We have used 1:3 gear reduction ratio in order to increase the overall torque. Three Hall sensors with 120° phase difference are attached to the motors for speed measurement. These sensors output series of pulses with varying frequencies according to the motor speed. Fig. 3.4 shows a typical output of these Hall sensors. The main disadvantage of these motors is the complex driver circuit. We have researched a lot for finding a robust brushless DC motor driver and finally decided to use the one in [6]. The circuitry is complex but it is working without any problems and seems to be

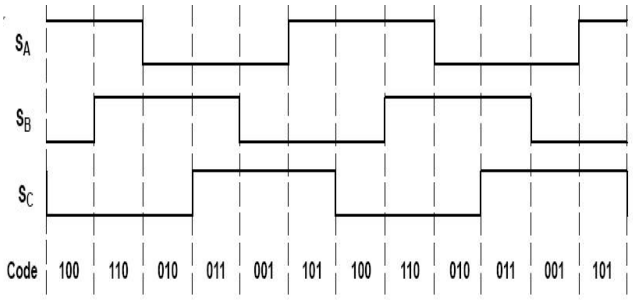

Figure 3.4. Sensor outputs of the brushless DC motor.

robust.

3.1.1. Motor Speed Control

As mentioned in the beginning, the most important part of our low level unit is the motor speed control. We need a robust controller in order to drive the wheels correctly. The omni–drive robots are very hard to control and precise wheel speeds have crucial importance for correct maneuvering. If the speeds cannot be controlled precisely, the robot oscillates around the trajectory or it might even go unstable. For motor speed estimation, we use the Hall sensor feedback signal. A counter starts to run when the microcontroller receives a rising edge from the Hall sensor. The counter stops when the next rising edge is received. If we let C_i and C_f to be the two values of the counter that is stored for the period calculation, the difference between them is calculated as (see Fig. 3.5),

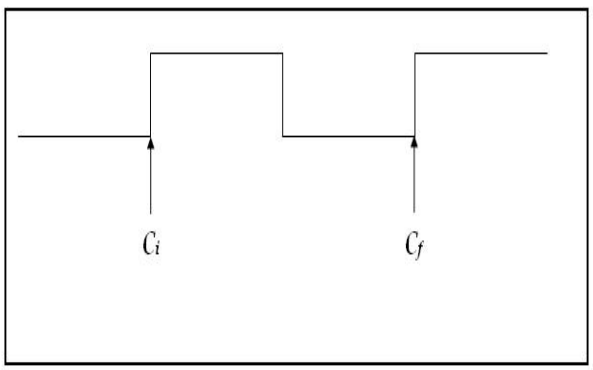

Figure 3.5. Captured Hall sensor signal

$$\Delta C = C_i - C_f. \tag{3.1}$$

The period of the sensor signal P_e is calculated by dividing this difference to the frequency of the counter (f_c):

$$P_e = \Delta C / f_c. \tag{3.2}$$

Since the number of pole pairs in the motor is 8, the mechanical period of the motor P_m is 8 times larger than the electrical period of one of the sensor signal, i.e.,

$$P_m = P_e \times \text{no. of pole pairs.} \tag{3.3}$$

From (3.3), the speed of the motor can be computed as

$$S_m = 1/(8 \times P_e) \text{ rev/sec} = 60/(8 \times P_e) \text{ rev/min} \tag{3.4}$$

When the motor runs at maximum speed, the electrical period is, $P_e = 60/(8 \times 4400) = 1.7 msec$, corresponding to a Hall sensor signal frequency of 587 Hz.

3.2. Mathematical Model of the DC Motor

There are various methods of tuning controllers such as Ziegler–Nichols method or trial and error. We have approached this problem more scientifically by deriving a mathematical model of our system first. To this end, the step response of the system has been obtained. In order to achieve this, we have input a step signal to the system and sampled the Hall sensor feedback with 1 ms intervals. The obtained response is shown in Fig. 3.6.

As seen in Fig. 3.6, the motor response seems to be overdamped and it approaches to a steady state value of approximately 540 Hz. Using(3.4), we find the maximum motor speed as Max Motor Speed = $60/8 \times (1/540) = 4050$ rpm. The maximum motor speed has decreased due to the additional load from the wheels. There is also a 25 ms delay at the beginning of Fig. 3.6, which is the response time of the system.

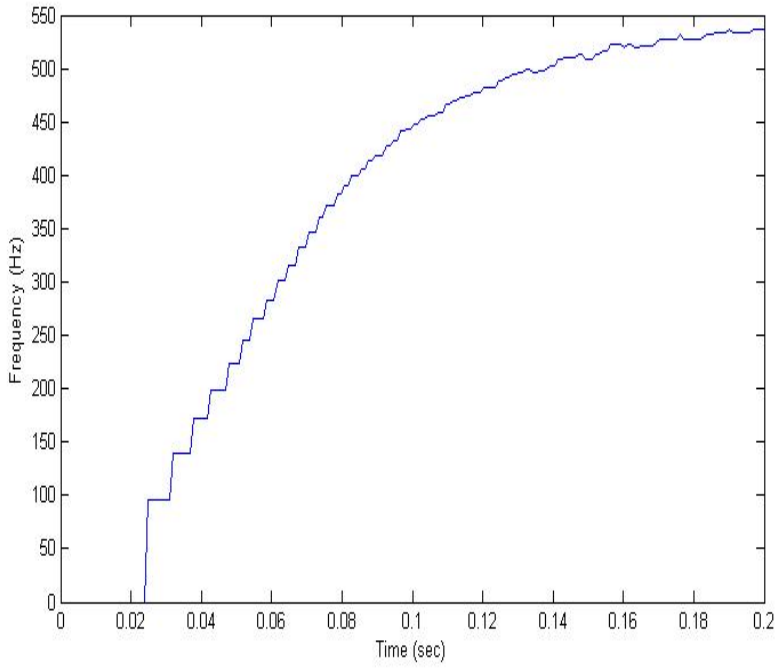

Figure 3.6. Open loop response of the motor

The second step in identification of the motor is to make a time-domain approximation for obtaining the transfer function of this system. We have used the transfer function of a DC motor in Laplace domain, ignoring the effects of the actuator circuit. The transfer function of a DC motor is given by

$$\frac{\Theta(s)}{V(s)} = \frac{K_m}{s(s\tau_m + 1)} \tag{3.5}$$

where $\Theta(s)$ and $V(s)$ represent the Laplace transforms of the the angular position in radians and the voltage applied to the motor respectively; τ_m is the mechanical time constant of the motor; K_m is the steady state gain. For the angular speed $\Omega(s)$ Eqn. (3.5) becomes

$$\frac{\Omega(s)}{V(s)} = \frac{s\Theta(s)}{V(s)} = \frac{K_m}{s\tau_m + 1} \tag{3.6}$$

Alternatively, we might write (3.6) as

$$\frac{\Omega(s)}{V(s)} = \frac{a}{s + b} \tag{3.7}$$

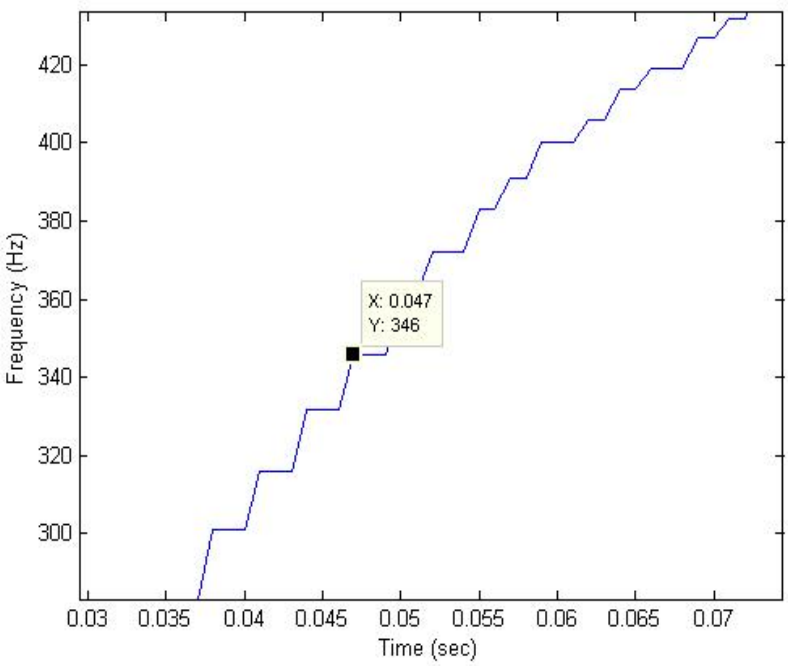

Figure 3.7. Closest sample to the 340 Hz point

where $a = \frac{K_m}{\tau_m}$ and $b = \frac{1}{\tau_m}$. As seen from Fig. 3.6, motor response with respect to the angular speed is represented by a first order transfer function.

In order to fit a transfer function to our step response given in Fig. 3.6, we have ignored the response delay at the beginning and determined the time constant (τ_s) of our system. The time constant (τ) of a system represents the time it takes the system's step response to reach %63 of its final value. In our case, the %63 of the final value corresponds to $540 \times 0.63 = 340$ Hz. The closest point to the output value of 340 Hz is marked in Fig. 3.7. The time value at that point is 47 msec, therefore we can roughly estimate τ_s to be 46.5 msec.

The steady state gain is determined by dividing the steady state value to the applied input value. In our case, we use the frequency of the Hall sensor feedback signal as the output of the system and PWM duty cycle as the input, therefore the steady state gain of the system is

$$K_s = 540/600 = 0.9 \tag{3.8}$$

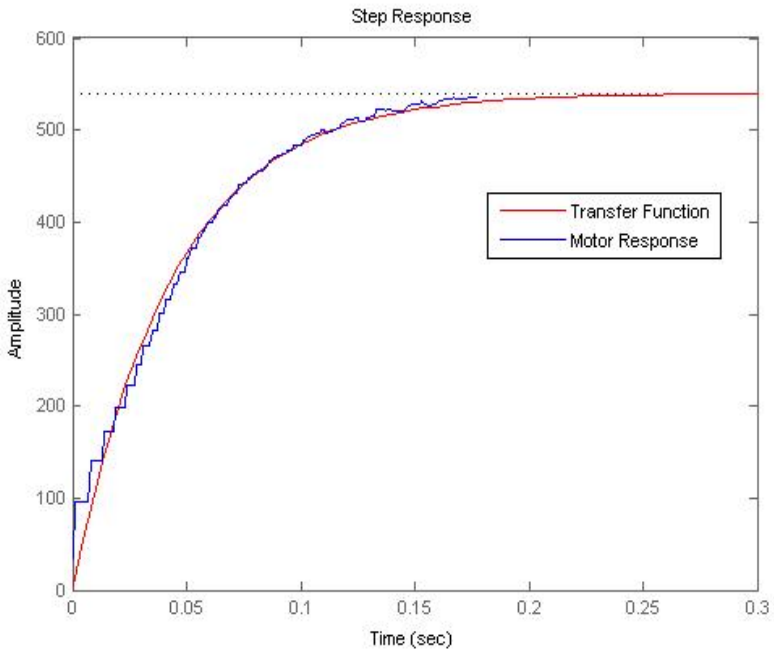

Figure 3.8. Step responses of the obtained transfer function and the motor

and the overall open loop transfer function of the system is

$$\frac{F(s)}{\gamma(s)} = \frac{0.9}{0.0465s + 1} = \frac{19.35}{s + 21.5} \tag{3.9}$$

where F represents frequency of the feedback signal and γ represents PWM duty cycle. In order to verify our approximation, we plot the step response of the obtained transfer function together with that of the motor in Fig. 3.8. As seen, our approximation is pretty close to the actual motor response.

3.2.1. PWM

PWM is a very popular method for speed control of DC motors. The basic idea is to vary the applied input to the system by varying the pulse width. The term duty cycle is the ratio between the pulse duration and the period of the signal. %100 duty cycle means we apply a continuous DC signal to the system while %50 duty cycle means we apply half of the maximum applicable input. By varying the duty cycle of the pulse, we can change the applied voltage to the motor. Fig. 3.9 shows examples of

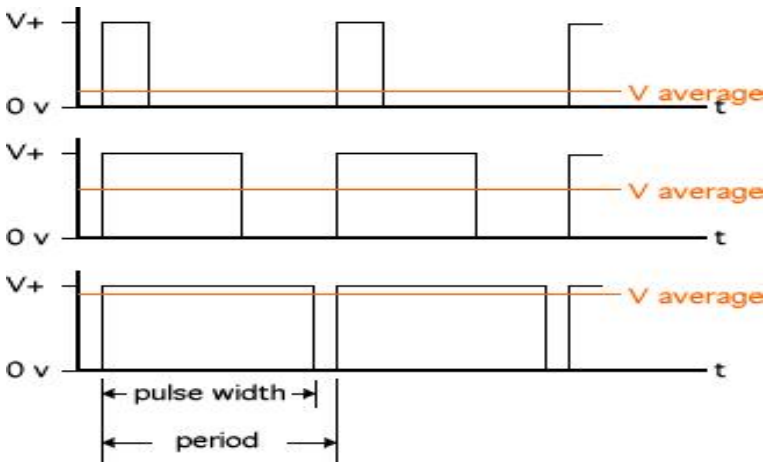

Figure 3.9. PWM signals with different duty cycles

PWM signals with different duty cycles.

As seen in the figure, when duty cycle decreases, the average input applied to the system decreases. In our case, we have divided the whole duty cycle interval into 600 steps. The input voltage range of the motor actuator circuit for the speed control is between 0–6.25V; hence a single duty cycle step corresponds to a change of $6.25/600 = 0.0105V$ at the actuator input which is sensitive enough for this application.

3.3. Controller Design

3.3.1. Ideal Case

After mathematically modeling our system, we have built a Simulink model in order to design the controller. In the ideal case, we have ignored the computational limitations of the microcontroller and delays caused by the Hall sensor. We have only included the saturation block to the PWM duty cycle because we cannot exceed the maximum PWM duty cycle limit. Basically, we want speed tracking for the system whose transfer function is given as

$$G(s) = \frac{a}{s+b} \tag{3.10}$$

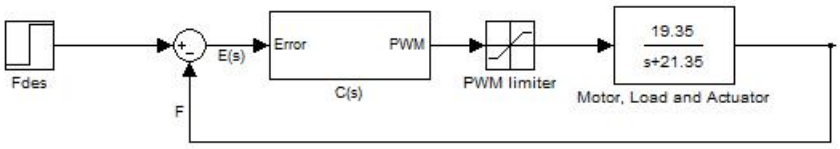

Figure 3.10. Closed loop Simulink model of our system

This system requires a pure integrator in the loop in order to achieve the given task. Hence, we have designed a PI controller whose transfer function is given by

$$C(s) = K_p + \frac{K_i}{s} \tag{3.11}$$

In Fig. 3.10, $E(s) = F_d(s) - F(s)$, while K_p and K_i are the proportional and integral gains, respectively.

The overall Simulink model is shown in Fig. 3.10 and the structure of the PI controller is given in Fig. 3.11. The closed loop transfer function of our system is

$$H(s) = \frac{19.35(K_i + K_p s)}{s^2 + (19.35K_p + 21.35)s + 19.35K_i} \tag{3.12}$$

Our system becomes second order after adding the controller. In order to understand

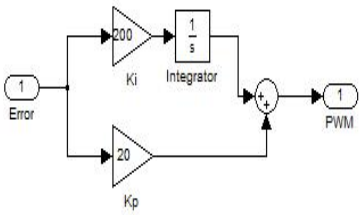

Figure 3.11. Structure of $C(s)$

our new system better, a brief analysis of second order systems is given in Section 3.3.3 of this chapter. For the stability analysis of our system, we have constructed the Routh table (see Table 3.2). For a second order system, the system is stable if and only if the coefficients of the denominator are positive, i.e,

$$K_p > -1.11, \quad K_i > 0 \tag{3.13}$$

Table 3.2. Routh table for the ideal closed loop system

s^2	1	$19.35K_i$
s^1	$19.35K_p + 21.35$	0
s^0	$19.35K_i$	0

3.3.2. Steady State Error Performance

Until now, we have discussed the transient response of our system but not its steady state performance. Basically, the steady state analysis of the system is based on whether the given system can track a given input perfectly or there is some error, called the steady state error that the system cannot eliminate. For our system, the output at any given instant is

$$F(s) = H(s)F_d(s) \tag{3.14}$$

Using (3.14), the steady state output of the system is given by

$$f_{ss} = \lim_{s \to 0} sF(s) \tag{3.15}$$

If we substitute $F_d(s) = \frac{F_d}{s}$ in (3.15) for a step input and and take the limit, we obtain

$$f_{ss} = F_d \tag{3.16}$$

We note that the output converges to the desired value. This result is expected since we have added an integrator to the system for eliminating the steady state error.

3.3.3. Second Order Systems

Second order systems exhibit a wide range of responses compared to first order systems. The general transfer function of a second order system is given by

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.17}$$

where ω_n is the natural frequency of the system and ζ is the damping ratio. There are 4 different types of response we can observe according to the damping ratio of the system and they are given in Table 3.3 [26].

The damping ratio changes according to the poles of the system, therefore it is obvious that the pole locations affect the system response.

With these in mind, we have focused on the under–damped response for our system. Our priority was to decrease the settling time without having too much overshoot (OS). For this purpose, we have varied K_i and K_d until we were satisfied with the obtained response. Our optimal gains are $K_i = 110$ and $K_p = 3$. With these gain values, the overall transfer function became,

$$H(s) = \frac{58s + 2128}{s^2 + 79.4s + 2128} \tag{3.18}$$

We have two complex conjugate roots with real parts meaning that, our closed loop response is under–damped. Fig. 3.12 shows the obtained response versus the open

Table 3.3. Table of second order system responses with respect to the damping ratio

ζ	Response Type
0	Undamped
$0 < \zeta < 1$	Underdamped
1	Critically Damped
$\zeta > 1$	Overdamped

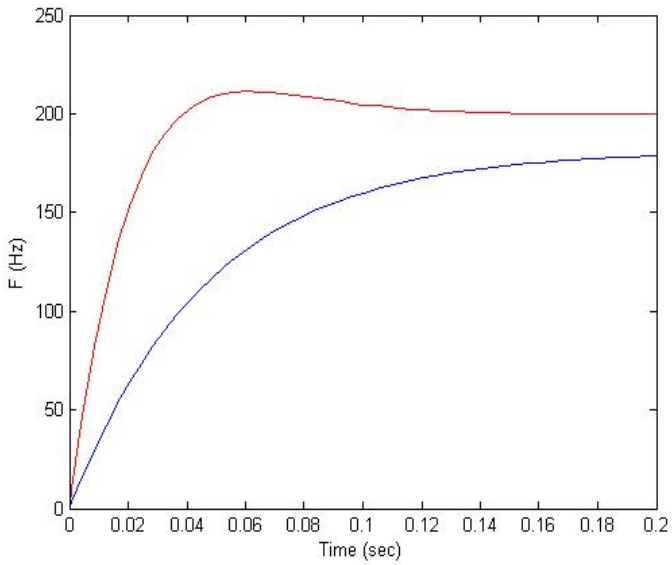

Figure 3.12. Closed loop versus open loop response of the system

loop response of the system for a 200 Hz input. We can clearly see that the transient response has improved and our system's settling time becomes approximately 100 msec while the open loop system has 180 msec settling time and a 10% steady state error.

3.3.4. Effect of the Actuator

Now we investigate the effect of the actuator dynamics on our system. The actuator dynamics can be modeled by a first order transfer function given in (3.10). The new system is shown in Fig. 3.13. The time constant of the actuator τ_{act} is obtained by the following calculations. Our motor has 8 poles and 3 Hall sensors, therefore, for each turn, $8 \times 3 = 24$ pulses are generated. The closed loop speed controller IC which reads these sensors, generates $24 \times 2 = 48$ pulses, one for each rising and falling edge. The maximum number of pulses are generated when the motor turns at maximum

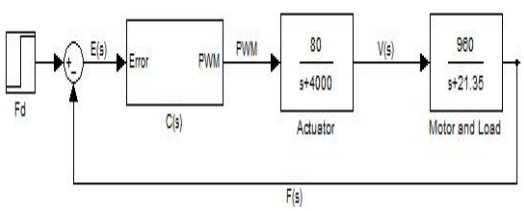

Figure 3.13. Closed loop system with actuator dynamics

speed. For our motor the maximum speed is found as 4050 rpm which is approximately 67 rps. At this speed, the IC generates $48 \times 67 = 3216$ pulses per second; therefore the pulse width of the IC must be less than $1/3216 = 0.31$ msec. In order to satisfy the desired pulse width requirement, we have chosen suitable resistor and capacitor values and set the pulse width to 0.25 msec which is also the time constant of the actuator (τ_{act}).

The input to the actuator circuit is the PWM pulse and the output is the voltage applied to the motor; therefore, the steady state gain K_{act} is calculated as

$$K_{act} = \frac{12}{600} = 0.02 \tag{3.19}$$

The overall transfer function of the actuator is

$$G_{act}(s) = \frac{V(s)}{\gamma(s)} = \frac{0.02}{0.00025s + 1} = \frac{80}{s + 4000} \tag{3.20}$$

The transfer function of the motor has also changed because of the change in the input. The K_m of the motor is recalculated as

$$K_m = \frac{540}{12} = 45 \tag{3.21}$$

and the overall transfer function becomes

$$G_{mot}(s) = \frac{F(s)}{V(s)} = \frac{45}{0.0465s + 1} = \frac{960}{s + 21.35} \tag{3.22}$$

The closed loop transfer function of this system is found to be

$$H(s) = \frac{76800(K_p s + K_i)}{s^3 + 4021.35s^2 + (85400 + 76800K_p)s + 76800K_i} \tag{3.23}$$

For the stability analysis, we again use the Routh table (see Table 3.4). The condition

Table 3.4. Routh table for the closed loop system with actuator dynamics

s^3	1	$85400 + 76800K_p$
s^2	4021.35	$76800K_i$
s^1	$85400 + 76800K_p - 19.1K_i$	0
s^0	$76800K_i$	0

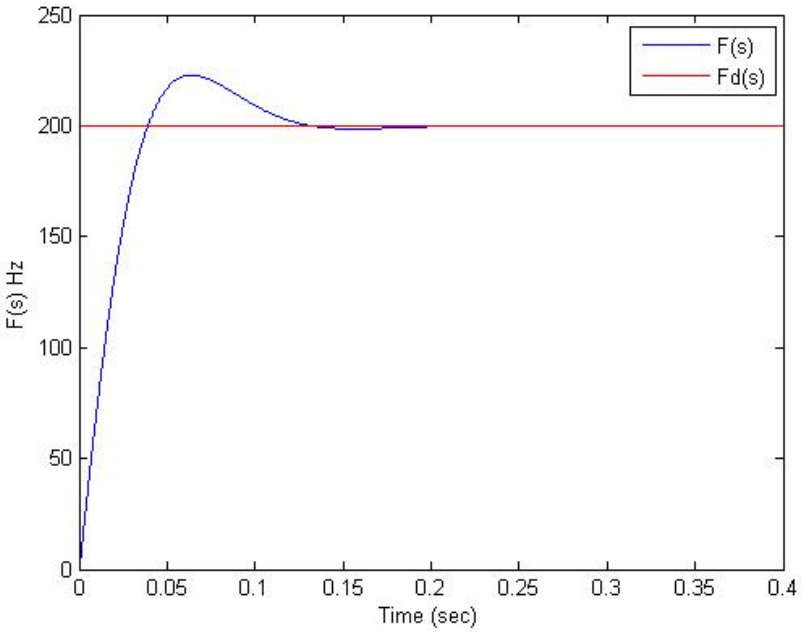

Figure 3.14. Closed loop system response

for the stability is:

$$0 < K_i < 4471 + 4020\, K_p \tag{3.24}$$

For an improved step response, we have chosen gains as $K_p = 2$ and $K_i = 105$. The step response for 200 Hz input is shown in Fig. 3.14. With this choice of the controller parameters, he closed loop poles are at

$$r_1 = -3961.54 \tag{3.25}$$

$$r_{2,3} = -29.9 \pm 33.78i \tag{3.26}$$

and the closed loop zero is at

$$z_{cls} = -52.5 \tag{3.27}$$

The closed loop transfer function of the system is

$$H(s) = \frac{153600s + 8064000}{s^3 + 4021.35s^2 + 239000s + 8064000} \tag{3.28}$$

When we compare this system with the ideal case, we see that the effect of actuator is negligible because its pole is far away from the dominant poles of the system.

3.3.5. Steady State Error Performance

The steady state performance of the system is analyzed by using the same procedure in the ideal case. In s–domain, the output of the system is given by

$$F(s) = H(s)F_d(s) \tag{3.29}$$

When we substitute $F_d = \frac{F_d}{s}$ and take the limit, we obtain

$$f_{ss} = \lim_{s \to 0} sH(s)\frac{F_d}{s} = F_d \tag{3.30}$$

i.e., the output converges to the desired value as expected.

3.3.6. Digital Effects

In the previous sections, we have adapted a continuous time controller design approach. However, in real implementation, a digital controller is used; therefore we have to employ discrete-time analysis for a better design. The digital control scheme is shown in Fig. 3.15.

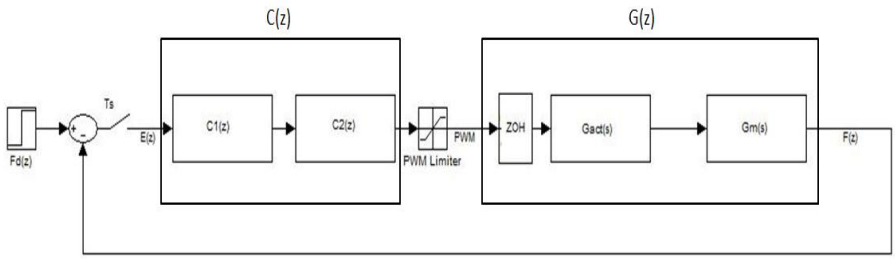

Figure 3.15. Discrete closed loop control system

The first thing that needs to be done is to determine the pulse transfer function G(z).

$$G(z) = Z^{-1}\left[ZOH(s)G_{act}(s)G_{mot}(s)\right] \tag{3.31}$$

where ZOH(s) represents the transfer function of the Zero Order Hold (ZOH). Using inverse z transform of ZOH, (3.31) becomes,

$$G(z) = (1 - z^{-1})Z^{-1}\left[\frac{G_{act}(s)G_{mot}(s)}{s}\right] \tag{3.32}$$

If we substitute the transfer functions of $G_{act}(s)$ and $G_m(s)$ in (3.32), we obtain

$$G(z) = (1 - z^{-1})Z^{-1}\left[\frac{76800}{s(s+4000)(s+21.35)}\right] \tag{3.33}$$

We apply partial fraction expansion to the term in parenthesis and get

$$G(z) = (1 - z^{-1})Z^{-1}\left[\frac{0.9}{s} + \frac{0.0048}{s+4000} - \frac{0.9}{s+21.35}\right] \tag{3.34}$$

Now we can employ the inverse z transform to the terms in parenthesis

$$G(z) = (1 - z^{-1})\left[\frac{0.9z}{z-1} + \frac{0.0048z}{z - e^{-4000T_s}} - \frac{0.9z}{z - e^{-21.35T_s}}\right] \tag{3.35}$$

where $T_s = 2msec$ is the sampling time. With this sampling time, the exponential

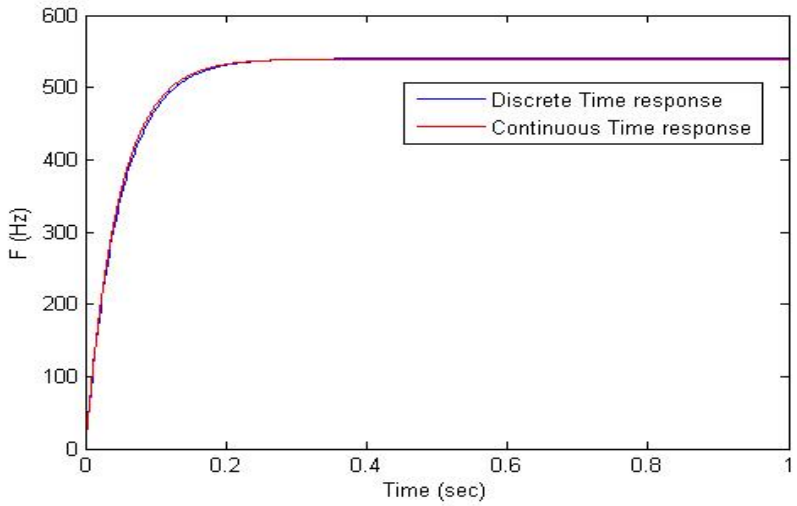

Figure 3.16. Open loop responses of discrete and continuous time systems

terms in (3.35) become

$$e^{-4000T_s} = 0.00035 \tag{3.36}$$

$$e^{-21.35T_s} = 0.96 \tag{3.37}$$

We can ignore the first exponential term because its value is so small compared to the other terms. If we put the numerical value of the second exponential term in (3.35) and leave out the ignored terms, we get

$$\frac{0.9z^2(z-0.96) - 0.9z^2(z-1)}{(z-1)z(z-0.96)} \tag{3.38}$$

The overall discrete transfer function G(z) becomes

$$G(z) = \frac{0.036}{z-0.96} \tag{3.39}$$

In order to see whether our discrete time transformation is correct or not, we have compared the open loop responses of the continuous time and the discrete time system (see Fig. 3.16). We note that, the two responses are nearly identical.

For the controller part C(z), we have simulated the same controller that we have

used in our system. We have used a PD controller in the first part $C_1(z)$. The discrete transfer function of the PD controller is given by

$$C_1(z) = K_{pdis} + K_{ddis}\frac{z-1}{T_s\, z} \tag{3.40}$$

where K_{pdis} and K_{ddis} are the proportional and derivative gains respectively. The relation between the output of the $C_1(t)$ (CS(t)) and the PWM duty cycle $\gamma(t)$ is given by

$$\gamma(t) = \gamma(t - T_s) + CS(t) \tag{3.41}$$

If we take the z–transform of (3.41), we get

$$\gamma(z) = \frac{\gamma(z)}{z} + CS(z) \tag{3.42}$$

$$= \frac{z\, CS(z)}{z-1} \tag{3.43}$$

Eqn.(3.43) gives us the transfer function of the second part of the controller.

$$C_2(z) = \frac{z}{z-1} \tag{3.44}$$

The overall transfer function of the controller C(z) is given by

$$C(z) = \frac{z(T_s\, K_{pdis} + K_{ddis}) - K_{ddis}}{T_s\,(z-1)} \tag{3.45}$$

Using (3.31–3.45), we determine the closed loop transfer function

$$H(z) = \frac{18(T_s\, K_{pdis} + K_{ddis})z - 18\, K_{ddis}}{z^2 + (-1.96 + 18(T_s K_{pdis} + K_{ddis}))\, z + 0.96 - 18K_{ddis}} \tag{3.46}$$

For the stability analysis, we apply the Jury stability test. Our system's Jury stability table is given in Table 3.5. From the Jury theorem, the stability conditions are as

Table 3.5. Jury stability table

Row	z^0	z^1	z^2
1	$0.96 - 18K_{ddis}$	$-1.96 + 18(T_sK_{pdis} + K_{ddis})$	1

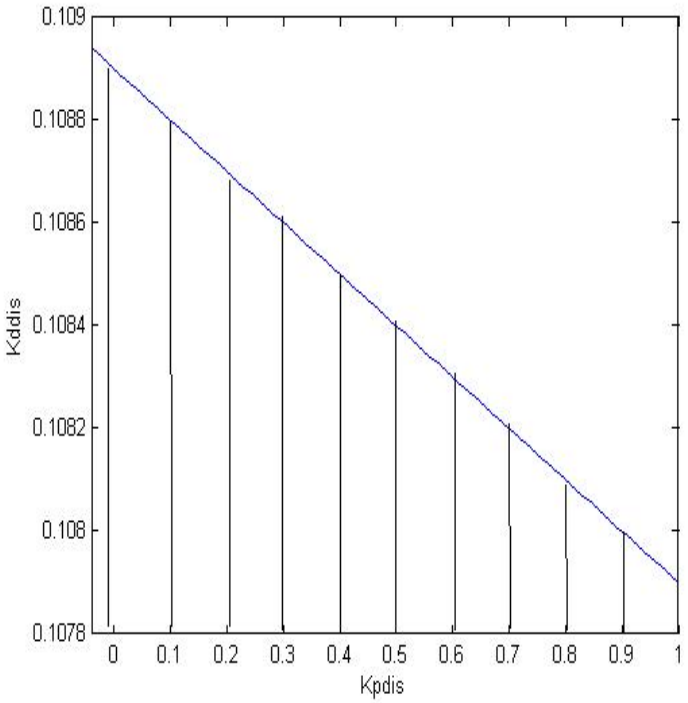

Figure 3.17. Stability region of the system

follows,

$$K_{pdis} > 0, \quad -0.04 < K_{ddis} < 0.1089 - 0.001K_{pdis} \tag{3.47}$$

The stability region of the system is the shaded part of Fig. 3.17.

3.3.7. Steady State Error Performance

For the steady state analysis we again apply a step input to the system. The transfer function of the step in z domain is given by

$$F_d(z) = \frac{f_d\, z}{z-1} \tag{3.48}$$

The output of the system at any given instant is given by

$$F(z) = H(z)F_d(z) \tag{3.49}$$

If we apply the input and take the limit, we get

$$F_z = \lim_{z\to 1} \frac{z-1}{z} H(z) \frac{f_d\, z}{z-1} = f_d \tag{3.50}$$

(3.50) shows that our system can track a step input perfectly. This ends our discussion about the Low Level Control of our overall system.

4. IMAGE PROCESSING

This chapter summarizes our image processing module. The structure of this module is shown below in Fig. 4.1. Image processing module is a very important part in our overall system. It is the only feedback for the high level control unit. We utilize Intel's OpenCV libraries in our software. It includes most of the currently used image processing algorithms and functions. It is optimized and computationally efficient, therefore it is used in several real time applications. All the parts of this module is discussed in the subsequent sections.

4.1. Image Capturing

The first step is to capture an image from our cameras. In order to cover the whole field, we use two cameras in our system. The experimental setup can be seen in Fig. 4.2.

We have used the CMU-1394 Firewire camera driver and API for accessing camera functions and acquiring image [27]. We have developed an interface code which allows us to process acquired images with OpenCV. The camera sends data in Bayer format which is first converted to RGB and then to YCrCb (a variant of YUV) format before the image is processed. The YCrCb format is less sensitive to different illumination conditions in the environment which enables us to build a more robust system.

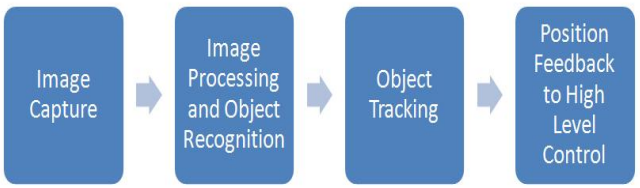

Figure 4.1. Structure of image processing module

Figure 4.2. Our multiple camera vision system

4.2. Image Processing and Object Recognition

This is the key part of the image proceesing module. If we cannot recognize objects correctly, then it will be impossible to send correct commands to the robots. When an image is received, it is processed step by step as explained below:

4.2.1. Color Thresholding and Preparing Binary Images

In this step, we apply color space thresholding to the image in order to extract the colors of interest. Since each color has different channel values, we set upper and lower thresholds for each channel Y, Cr and Cb of the image between 0 and 255 (each channel has 8-bit depth). Next, each pixel of the image is scanned. The binary values of the pixels that satisfy the threshold constraints are set to 1; and the rest are set to 0. The resulting image is a binary that filters out all the colors except the desired one. Morphological operations are also performed on these binary images in order to remove noise. In order to save up time, we determine the thresholds in a separate program and save them in a text file. Our processing code directly reads the thresholds from the text file and produces a binary image for each color of interest.

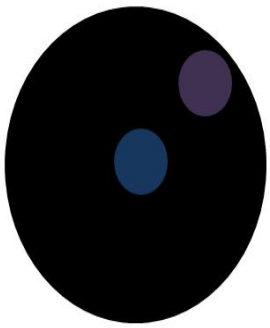

Figure 4.3. Robot identification patch

4.2.2. Finding Contours

This step is used to detect and retrieve the coordinates of the ball and colored patches on our robots (see Fig.4.3). After creating a binary image, we find the contours within, and measure its area to avoid false detection. If the area is within the range, center coordinates (x, y) of the contour are returned.

4.2.3. Identification of Robots

When central coordinates of a contour is retrieved, it is classified according to its color. For example, one of our robots has blue central patch while the other has yellow; therefore if we detect a blue and a yellow contour, we create two robot objects in the code; if only one contour is retrieved, then only one robot object is created. Center coordinates of the contours provides us the central location of each robot.

We also have purple patches in each robot to designate its front side (see Fig.4.3). In order to assign the purple patches correctly to the robots, we measure their distance relative to the central patches. The distance between the central patch and the purple patch must not exceed a certain threshold. If the identification is successful, the parameters such as the orientations and relative distances of the robots are calculated and returned. Fig. 4.4 shows all the steps that are explained above.

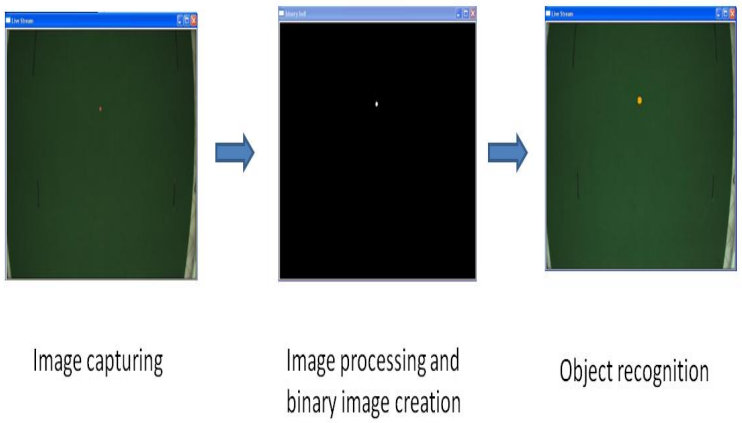

Figure 4.4. Image processing steps

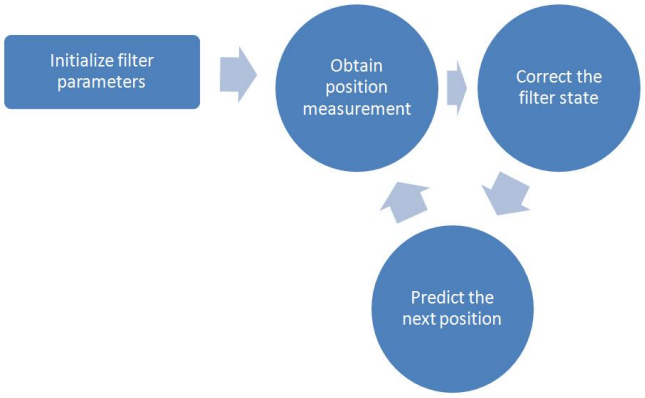

Figure 4.5. Flow chart for the Kalman Filter

4.3. Object Tracking and Position Feedback

As mentioned before, the robots and the ball move very fast during the game so it is nearly impossible to track a robot in real time, even with the most optimized algorithms. In our system, there is approximately a delay of 150 ms between calculating the robot's position and sending a movement command. With an assumption that the robot moves 2 m/s on the average, it displaces approximately 30 cm until a new motion command is received. This can cause unexpected and oscillatory behavior and would be impossible for the robot to pass and shoot precisely. Therefore, it is necessary to predict the robots' future movements.

Many different methods are used for this task such as ANN's, Smith Predictor and Kalman Filter. In our system, we have decided to use Kalman Filter as a predictor.

We utilize OpenCV's Kalman Filter functions in our system. The flow chart for the Kalman filter is given in Fig. 4.5. A detailed discussion is beyond the scope of this thesis but one can refer to [28] and [29] for further information.

5. AI MODULE

The AI module is important in controlling multiple robots. It assigns roles to each robot in order to complete a task, e.g., scoring a goal or defending its own goal. Such tasks require proper locomotion of individual robots which is carried out by taking advantage of their holonomic drive systems. Our primary task for this module is to send a robot to a point on the field by calculating necessary velocity commands.

5.1. Velocity Vector Calculation

In order to send a robot to a desired location, one can try the easiest solution, i.e., we orient the robot's front towards the goal point and then command it to go forward until the goal point is reached. Vision feedback enables us to make corrections if the robot diverges from its path. An alternative method can be developed by taking the advantage of holonomic drive system of our robot. To briefly describe our methodology,

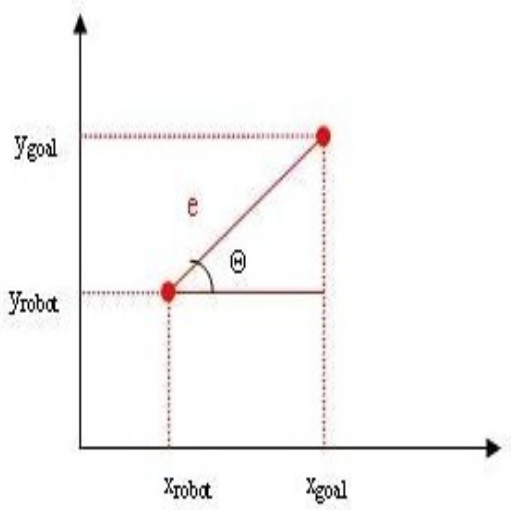

Figure 5.1. Error vector

suppose that we set a goal point in the 2-D plane as shown in Fig. 5.1. The location errors in x and y coordinates are defined as:

$$e_x = x_{goal} - x_{robot}, \tag{5.1}$$

$$e_y = y_{goal} - y_{robot}. \tag{5.2}$$

Using (5.1–5.2), we create a position error vector:

$$\Theta_e = \tan^{-1}(e_y/e_x), \tag{5.3}$$

$$|e| = \sqrt{e_x^2 + e_y^2}. \tag{5.4}$$

In order to direct the robot towards the goal point, we need proper velocity vectors in the x and y directions, which can be formulated, we as follows:

$$v_x = e_x = |e| \cos \Theta_e \tag{5.5}$$

$$v_y = e_y = |e| \sin \Theta_e \tag{5.6}$$

This type of velocity formulation is frequently used in robot control [21]. The velocities are proportional to the norm of the error vector that is the distance between the desired and current location of the robot. However, we should ensure that the norms of these velocity vectors will not exceed the physical limits of the robot. This can be achieved by limiting the velocity vectors using (5.7) as follows.

$$\text{IF } |e| > V_{thrs}, \text{ then } v_{xnorm} = v_x/|e|, \quad v_{ynorm} = v_y/|e| \tag{5.7}$$

In (5.7), v_{xnorm} and v_{ynorm} represents the normalized velocity vectors and V_{thrs} is the velocity threshold. The new velocity vectors can be calculated as

$$v_{xn} = V_{thrs}\, v_{xnorm} \tag{5.8}$$

$$v_{yn} = V_{thrs}\, v_{ynorm} \tag{5.9}$$

We should note that, the calculated velocities in (5.8–5.9) are relative to the global frame of reference. In order to have the robot move in the desired direction, we should transform these velocities relative to the robot's current frame of reference. This is accomplished by using the inverse of the rotation matrix in the z direction:

$$R_z^{-1}(\alpha) = R_{z^T} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}. \tag{5.10}$$

Finally, the commanded velocities are calculated as

$$\begin{bmatrix} v_{xrobot} \\ v_{yrobot} \end{bmatrix} = R_z^{-1}(\Upsilon) \begin{bmatrix} v_{xn} \\ v_{yn} \end{bmatrix}, \tag{5.11}$$

where Υ is the orientation of the robot relative to the global coordinate system. The next section discusses the transformation of the calculated velocities in 5.11 to the individual wheel velocities.

5.2. Wheel Velocity Transformation

This section is based on Raul Roja's work in [5]. In the previous section, we have calculated linear velocities for sending the robot to a point. These velocities should be transformed into individual wheel velocities for desired motion. This requires a good understanding of the kinematics of our robot.

5.2.1. Robot Kinematics

Kinematics is the branch of mechanics which describes the motion of an object without considering the forces exerted on it. There are different types of kinematics such as the translational kinematics which is for translational motion and the rotational kinematics which is for angular motion. The combination of these two is the rigid-body

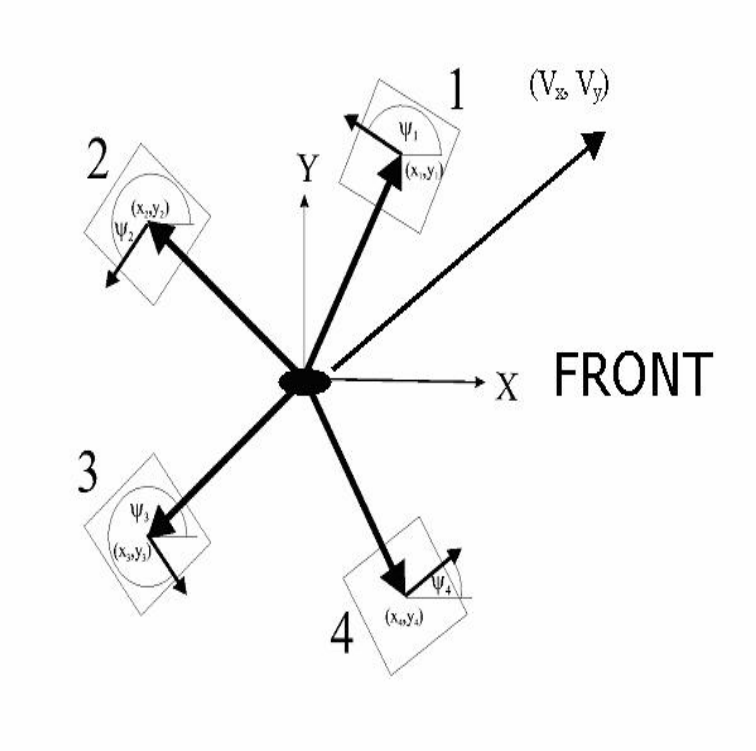

Figure 5.2. Schematic of our robot [11]

kinematics. Since our robot has both translational and rotational motion, rigid body kinematics apply to it.

We start our analysis by examining the schematic of our robot in Fig. 5.2. The point in the middle is the assumed center of mass (CM) of the robot. All of the angular and linear velocities are computed relative to this point. The robot's own frame of reference is also attached to this point. Each arrow at the wheel designates the positive rotation direction for that wheel. $\Psi_n \ , n = 1, \ldots, 4$ represent the angles of the wheels with respect to the x–axis. The translational acceleration of the CM is given by,

$$\vec{a} = \frac{1}{M}(\vec{F}_{w1} + \vec{F}_{w2} + \vec{F}_{w3} + \vec{F}_{w4}) \tag{5.12}$$

and the rotational acceleration,

$$\dot{\omega} = \frac{R}{I}(f_{w1} + f_{w2} + f_{w3} + f_{w4}) \tag{5.13}$$

where M is the mass of the robot; R is the radius of the robot's body; I is the moment of inertia; f_{wn} is the magnitude of the force vector $\vec{F}_{wn}$. Since each force vector is tangential to the robot, we can work with the magnitudes of the force vectors. Considering the geometry of the problem, we can split the components of the acceleration into x, y and $\dot{\omega}$:

$$a_x = \frac{1}{M}(f_{w1}\cos\Psi_1 + f_{w2}\cos\Psi_2 + f_{w3}\cos\Psi_3 + f_{w4}\cos\Psi_4) \tag{5.14}$$

$$a_y = \frac{1}{M}(f_{w1}\sin\Psi_1 + f_{w2}\sin\Psi_2 + f_{w3}\sin\Psi_3 + f_{w4}\sin\Psi_4) \tag{5.15}$$

$$\dot{\omega} = \frac{R}{I}(f_{w1} + f_{w2} + f_{w3} + f_{w4}) \tag{5.16}$$

or alternatively in matrix form:

$$\begin{bmatrix} a_x \\ a_y \\ \dot{\omega} \end{bmatrix} = \frac{1}{M}\begin{bmatrix} \cos\Psi_1 & \cos\Psi_2 & \cos\Psi_3 & \cos\Psi_4 \\ \sin\Psi_1 & \sin\Psi_2 & \sin\Psi_3 & \sin\Psi_4 \\ \frac{MR}{I} & \frac{MR}{I} & \frac{MR}{I} & \frac{MR}{I} \end{bmatrix}\begin{bmatrix} f_{w1} \\ f_{w2} \\ f_{w3} \\ f_{w4} \end{bmatrix} \tag{5.17}$$

The moment of inertia of the robot in the z direction is given by

$$I_z = \lambda M\, R^2,\ \ 0 \leq \lambda \leq 1 \tag{5.18}$$

where λ is the dimensionless inertia constant. If we substitute (5.18) in (5.17) and change the unit of the ω from rad/s to m/s, (5.17) becomes

$$\begin{bmatrix} a_x \\ a_y \\ R\dot{\omega} \end{bmatrix} = \frac{1}{M}\begin{bmatrix} \cos\Psi_1 & \cos\Psi_2 & \cos\Psi_3 & \cos\Psi_4 \\ \sin\Psi_1 & \sin\Psi_2 & \sin\Psi_3 & \sin\Psi_4 \\ \frac{1}{\lambda} & \frac{1}{\lambda} & \frac{1}{\lambda} & \frac{1}{\lambda} \end{bmatrix}\begin{bmatrix} f_{w1} \\ f_{w2} \\ f_{w3} \\ f_{w4} \end{bmatrix} \tag{5.19}$$

The 3×4 matrix in (5.19) is called the force coupling matrix (η) [5]. The 3×1 acceleration vector is denoted by a_r and the 4×1 force vector is denoted by f_w. The

Table 5.1. Wheel angles

Ψ_1	Ψ_2	Ψ_3	Ψ_4
157°	225°	315°	23°

wheel angles of our robot are tabulated in Table 5.1.

By using the angles in 5.1, we note that

$$\sin\Psi_2 = \cos\Psi_2 \tag{5.20}$$

$$\sin\Psi_3 = -\cos\Psi_3 \tag{5.21}$$

$$\cos\Psi_2 = \sin\Psi_3 \tag{5.22}$$

$$\sin\Psi_2 = -\cos\Psi_3 \tag{5.23}$$

$$\sin\Psi_1 = \sin\Psi_4 \tag{5.24}$$

$$\cos\Psi_1 = -\cos\Psi_4 \tag{5.25}$$

Hence the force coupling matrix η can be simplified as follows:

$$\eta = \frac{1}{M}\begin{bmatrix} \cos\Psi_1 & \cos\Psi_2 & -\cos\Psi_2 & -\cos\Psi_1 \\ \sin\Psi_1 & \sin\Psi_2 & \sin\Psi_2 & \sin\Psi_1 \\ \frac{1}{\lambda} & \frac{1}{\lambda} & \frac{1}{\lambda} & \frac{1}{\lambda} \end{bmatrix} \tag{5.26}$$

5.2.2. Transformation of Linear and Angular Velocities to Wheel Velocities

We can also find a transformation matrix between the commanded linear and angular velocities (v_{xrobot},v_{yrobot}, Rω) and the wheel velocities (w_v). From Fig. 5.3, we can decompose the velocity vector (v_{xrobot}) into wheel velocity vectors. Note that,

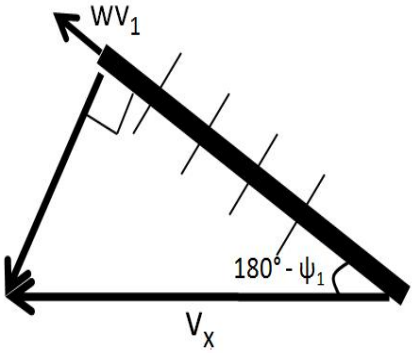

Figure 5.3. Wheel velocity vector

one of the components of v_{xrobot} gives the velocity of the main wheel while the other component gives the velocity of passive wheels (thin lines which are placed on the main wheel). We are only interested with the velocity of the main wheel, therefore, for the first wheel, the velocity vector (w_{v1}) is given by

$$w_{v1} = -\, v_{xrobot} \times \cos(\pi - \Psi_n) \tag{5.27}$$

$$= v_{xrobot} \times \cos \Psi_n \tag{5.28}$$

We can repeat the same decomposition for the remaining wheels to relate the wheel velocity vectors to (v_{yrobot}, $R\omega$) as follows: The obtained result is given in Eqn.(5.29).

$$\begin{bmatrix} w_{v1} \\ w_{v2} \\ w_{v3} \\ w_{v4} \end{bmatrix} = \begin{bmatrix} \cos\Psi_1 & \sin\Psi_1 & 1 \\ \cos\Psi_2 & \sin\Psi_2 & 1 \\ \cos\Psi_3 & \sin\Psi_3 & 1 \\ \cos\Psi_4 & \sin\Psi_4 & 1 \end{bmatrix} \begin{bmatrix} v_{xrobot} \\ v_{yrobot} \\ R\omega \end{bmatrix} \tag{5.29}$$

The 4×3 matrix in (5.29) is called the wheel velocity coupling matrix (κ). The 4×1 wheel velocity vector is denoted by w_v and the 3×1 linear and angular velocity vector

is denoted by v_r

$$\kappa = \begin{bmatrix} \cos\Psi_1 & \sin\Psi_1 & 1 \\ \cos\Psi_2 & \sin\Psi_2 & 1 \\ \cos\Psi_3 & \sin\Psi_3 & 1 \\ \cos\Psi_4 & \sin\Psi_4 & 1 \end{bmatrix} = \begin{bmatrix} \cos\Psi_1 & \sin\Psi_1 & 1 \\ \cos\Psi_2 & \sin\Psi_2 & 1 \\ -\cos\Psi_2 & \sin\Psi_2 & 1 \\ -\cos\Psi_1 & \sin\Psi_1 & 1 \end{bmatrix} \tag{5.30}$$

Using η and κ, we can write the following identities

$$a_r = \eta\, f_w \tag{5.31}$$

$$w_v = \kappa\, v_r \tag{5.32}$$

If we integrate the expression in (5.32) over time interval $\Delta\, t$, we get

$$\Delta\, w_v = \kappa\, \Delta\, v_r \tag{5.33}$$

where

$$\Delta\, v_r = \Delta\, t \times a_r \tag{5.34}$$

Using (5.34), (5.32) becomes

$$\Delta\, w_v = \Delta\, t \times \kappa\, \eta\, f_w \tag{5.35}$$

Eqn. (5.35) shows us that the wheel speed increments are proportional to the motor forces. If we expand (5.35), we obtain

$$\Delta\, w_v = \Delta\, t \left(\frac{1}{M} \begin{bmatrix} 1 & \mu & -\varpi & \varepsilon \\ \mu & 1 & \rho & -\varpi \\ -\varpi & \rho & 1 & \mu \\ \varepsilon & -\varpi & \mu & 1 \end{bmatrix} + \frac{1}{M\,\lambda} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \right) f_w \tag{5.36}$$

where $\mu = \cos(\Psi_1 - \Psi_2)$, $\varpi = \cos(\Psi_1 + \Psi_2)$, $\varepsilon = \sin^2\Psi_1 - \cos^2\Psi_1$ and $\rho = \sin^2\Psi_2 - \cos^2\Psi_2$. Eqn. (5.36) shows that motor forces contribute to the rotation of the robot as well as the translation; therefore, the translational and rotational control of the robot should be decoupled.

5.2.3. Driving Without Rotating

In the previous section, we have derived the effect of wheel forces on the motion of the robot. Our aim is to send the robot to a desired position without the robot rotating which requires equal contribution of wheel forces in order to cancel rotation. For a force vector $y_{fw} = (1, -1, -1, 1)^T$, the rotation is 0 and the robot moves towards y direction with an acceleration of $a_r = \frac{2}{M}(0, \sin\Psi_1 - \sin\Psi_2, 0)^T$ (see (3.15)). In the same way, a force vector $x_{fw} = (-1, -1, 1, 1)^T$ corresponds to a movement without rotation in the x direction with an acceleration of $a_r = -\frac{2}{M}(\cos\Psi_1 + \cos\Psi_2, 0, 0)^T$. Thus any linear combination of these two vectors x_{fw} and y_{fw} would produce a movement without rotation in the x–y plane. Using (5.36), we can also derive the wheel accelerations for the vector y_{fw}:

$$\begin{bmatrix} \Delta\, w_{v1} \\ \Delta\, w_{v2} \\ \Delta\, w_{v3} \\ \Delta\, w_{v4} \end{bmatrix} = \Delta t \frac{1}{M} \begin{bmatrix} 1 - \mu + \varpi + \varepsilon \\ \mu - 1 - \rho - \varpi \\ -\varpi - \rho - 1 + \mu \\ \varepsilon + \varpi - \mu + 1 \end{bmatrix} \tag{5.37}$$

and for the vector x_{fw}

$$\begin{bmatrix} \Delta\, w_{v1} \\ \Delta\, w_{v2} \\ \Delta\, w_{v3} \\ \Delta\, w_{v4} \end{bmatrix} = \Delta t \frac{1}{M} \begin{bmatrix} -1 - \mu - \varpi + \varepsilon \\ -\mu - 1 + \rho - \varpi \\ \varpi - \rho + 1 + \mu \\ -\varepsilon + \varpi + \mu + 1 \end{bmatrix} \tag{5.38}$$

From (5.37–5.38), we note that the final velocities of the front and rear wheels are different although the torques applied on the robot's center are the same. Taking this

into consideration, we apply velocities that satisfy the given constraints so that the robot can move to a point without rotating.

5.3. Formation Control

In this section, we present several formation control algorithms for line and circle formations. The desired formation pattern can be an arbitrary shape, and basic formation patterns such as line and circle are widely used in today's world. For example, when soldiers explore an area, they may form a line or when fighter aircrafts protect a large bomber aircraft, they may form a circle around it.

Specifically, we consider n holonomic robots whose position and orientation are characterized by the vector $R_{ip} = (x_i, y_i, \theta_i)$, $i = 1, 2, \ldots, n$. Given the current positions of the robots, the task at hand is to determine the final destinations of the robots to achieve certain geometrical shapes.

5.3.1. Line Formation

In this section, we present a heuristic line formation algorithm first and then compare its performance with the two other methods that can be used for forming a line. The first is the brute force method that minimizes the total distance that should be covered by the robots. The second one is an analytic approach that minimizes the sum of the squares of distances to be covered (see [30]).

5.3.1.1. Proposed Heuristic Algorithm. Given $R_{ip} = (x_i, y_i, \theta_i)$, $i = 1, 2, \ldots, n$, the objective is to bring these arbitrarily placed robots into a line formation. Although many solution methods may exist, we further impose the constraint that this objective is met with minimum total displacements of the robots. This is an important criteria not only to conserve battery power but also possibly to minimize time to converge to the desired formation. Our 4–step algorithm is described below:

S1: We search for the robots with maximum and minimum x coordinates. These robots are held fixed in order to reduce the time it takes to complete the formation. The position difference between these two robots in terms of the x and y coordinates can be calculated as:

$$\Delta x_l = \max_{i=1,2,\ldots,n} x_i - \min_{i=1,2,\ldots,n} x_i, \tag{5.39}$$

$$\Delta y_l = y_{\arg\max_i x_i} - y_{\arg\min_i x_i}. \tag{5.40}$$

S2: Using (5.39) and (5.40), we find the slope, m_{hl}, and the angle α_{hl} between these two robots:

$$\alpha_{hl} = \tan^{-1}(\Delta y/\Delta x), \tag{5.41}$$

$$m_{hl} = \Delta y/\Delta x. \tag{5.42}$$

We can now construct a straight line between them.

S3: We calculate the length of the line using

$$d_{line_{hl}} = \sqrt{\Delta x^2 + \Delta y^2}. \tag{5.43}$$

Then, we mark $n-2$ equidistant points on the line so that we can place the remaining $n-2$ robots on these points to form the line. This is accomplished by

$$x_{dl,j} = \min_{i=1,2,\ldots,n} x_i + j\,\frac{d_{line_{hl}}}{n-1}\cos\alpha_{hl} \tag{5.44}$$

$$y_{dl,j} = y_{\arg\min_i x_i} + j\,\frac{d_{line_{hl}}}{n-1}\sin\alpha_{hl} \tag{5.45}$$

where $(x_{dl,j}, y_{dl,j})$, $j = 1, 2, \ldots, n-2$ denote the desired locations.

S4: In order to assign the desired locations $(x_{dl,j}, y_{dl,j})$, $j = 1, 2, \ldots, n-2$ to the robots, we construct the $(n-2) \times (n-2)$ matrix whose (i,j)–th component denotes the distance between the i–th robot and the j–th desired location $(x_{dl,j}, y_{dl,j})$. Hence, each row of this matrix represents a robot and its relative distance to these $n-2$ points. After creating the matrix, we check all the distances and if any of these distances is below a certain distance threshold $(d_{t_{hl}})$, then the corresponding robot is assigned to its associated desired location. If there are multiple robots that are closer than $d_{t_{hf}}$, the one with the minimum distance is assigned first. The remaining desired locations are allocated to the robots based on their relative distances. If none of the distances is below the threshold, then each robot is assigned to its desired location with the minimum distance. In order to avoid assigning multiple robots to the same location, the assigned location's column is removed from the distance matrix.

5.3.1.2. Brute Force Method. This method minimizes total distance that should be covered for forming a line. The total perpendicular distance of a set of points $(x_i, y_i, i = 1, 2, \ldots, n)$ to a line $(y = a_l + b_l\, x)$ is given by

$$d_{l_t} = \sum_{i=1}^{n} \frac{|y_i - (a_l + b_l\, x_i)|}{\sqrt{1 + b_l^2}} \tag{5.46}$$

There is no analytic method to minimize (5.46); however, we can do a brute force search for the optimal line parameters (a_{opt} & b_{opt}) by restricting the search to the following intervals.

$$a_{l_{lt}} \leq a_l \leq a_{l_{ht}} \tag{5.47}$$

$$b_{l_{lt}} \leq b_l \leq b_{l_{ht}} \tag{5.48}$$

where $a_{l_{lt}}$ and $b_{l_{lt}}$ are lower bounds while $a_{l_{ht}}$ and $b_{l_{ht}}$ are upper bounds.

5.3.1.3. The Least Squares Method. In this section, we discuss the least squares solution of distance minimization problem. As mentioned previously, there is no analytic solution for minimizing the distance of a point to a line, but there is a closed form solution for the minimization of the square of this distance. In our case, we want to minimize sum of the the square of the total distances

$$d^2_{lsm_t} = \sum_{i=1}^{n} \frac{(y_i - (a_l + b_l\, x_i))^2}{1 + b_l^2} \tag{5.49}$$

where a_l and b_l are the parameters of the line. The optimal solution for a_l and b_l, denoted by, a_{lsm} and b_{lsm}, are given as [30]

$$a_{lsm} = \frac{\sum_{i=1}^{n} y_i - b_{lsm} \sum_{i=1}^{n} x_i}{n} \tag{5.50}$$

$$b_{lsm_{1,2}} = -B \pm \sqrt{B^2 + 1} \tag{5.51}$$

where B is given by

$$B = \frac{1}{2} \frac{\left[\sum_{i=1}^{n} y_i^2 - \frac{1}{n}(\sum_{i=1}^{n} y_i)^2\right] - \left[\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2\right]}{\frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i} \tag{5.52}$$

From (5.51), we note that there are two solutions for b_{lsm}; therefore, for each b_{lsm}, we have different a_{lsm}. In order to determine which parameter set is best for a given set of robot positions, one should check whether one of the matrices

$$\nu_1 = \begin{pmatrix} \frac{\partial^2 d^2_{lsm_t}}{\partial^2 a_{lsm_1}} & \frac{\partial d^2_{lsm_t}}{\partial a_{lsm_1} b_{lsm_1}} \\ \frac{\partial d^2_{lsm_t}}{\partial a_{lsm_1} b_{lsm_1}} & \frac{\partial^2 d^2_{lsm_t}}{\partial^2 b_{lsm_1}} \end{pmatrix} \tag{5.53}$$

$$\nu_2 = \begin{pmatrix} \frac{\partial^2 d^2_{lsm_t}}{\partial^2 a_{lsm_2}} & \frac{\partial d^2_{lsm_t}}{\partial a_{lsm_2} b_{lsm_2}} \\ \frac{\partial d^2_{lsm_t}}{\partial a_{lsm_2} b_{lsm_2}} & \frac{\partial^2 d^2_{lsm_t}}{\partial^2 b_{lsm_2}} \end{pmatrix} \tag{5.54}$$

is positive definite. Due to the computational complexity, we alternatively construct the lines for both sets of parameters and calculate the corresponding total distances. We then choose the line with the minimum total distance.

Table 5.2. Position of the robots on the x–y plane

Robots	x	y
R_{1p}	127	65
R_{2p}	189	221
R_{3p}	284	360
R_{4p}	432	184
R_{5p}	623	542

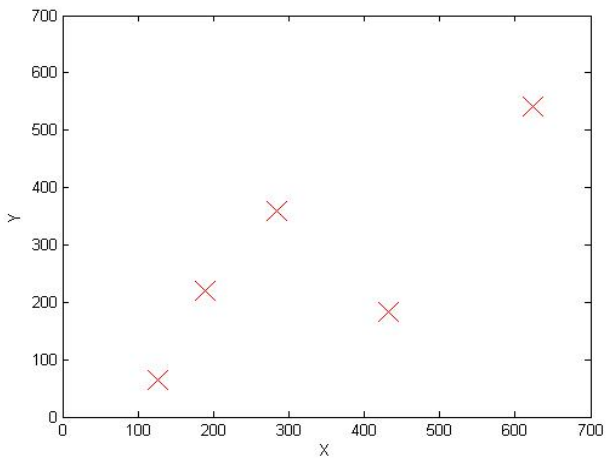

Figure 5.4. Robots on the x–y plane

5.3.1.4. Simulation Results. In this section, the simulation results for the 3 methods discussed above are given. We consider $n = 5$ robots and randomly position them on the 2–D plane (see Table 5.2 and Fig. 5.4).

The steps and the result of our proposed heuristic algorithm are given in Fig. 5.8

Table 5.3. Performance comparison table for line formation methods

Method Name	Total displacement (units)	Execution Time (msec)
Proposed Heuristic Algorithm	427	25
Brute Force Method	304	17
Least Squares Method	309	10

and Fig. 5.9 while in Fig. 5.10, the line drawn by the brute force method and the final positions of the robots are shown. Finally Fig. 5.11 shows the line drawn according to the least squares solution and the resulting robot positions. The lines obtained by each algorithm are in Fig. 5.12.

From Fig. 5.12, we note that the solutions of the brute force and least squares method are barely distinguishable. The total displacement for these two methods are nearly the same (see Table 5.3) while the proposed heuristic approach requires %35 more displacement.

If we analyze the execution time of these methods, Least Squares Method needs only 10 msec to complete the execution. The brute force method takes 17 msec to complete its execution but this value is heavily dependent on the size of the search intervals. The proposed approach takes 25 msec to complete its execution.

Comparing the performance figures in Table 5.3, we can conclude that the proposed heuristic algorithm is far from ideal in terms of total displacement. On the other hand, the robots are placed equidistant from each other; therefore, if the dimensions of the robots are taken into account, it reduces the risk of colliding robots in the formation. Moreover, only $n-2$ robots have to move instead of n robots during execution which decreases the computational load for trajectory planning; therefore, in the real system implementation, the total execution time might be less than the other two approaches.

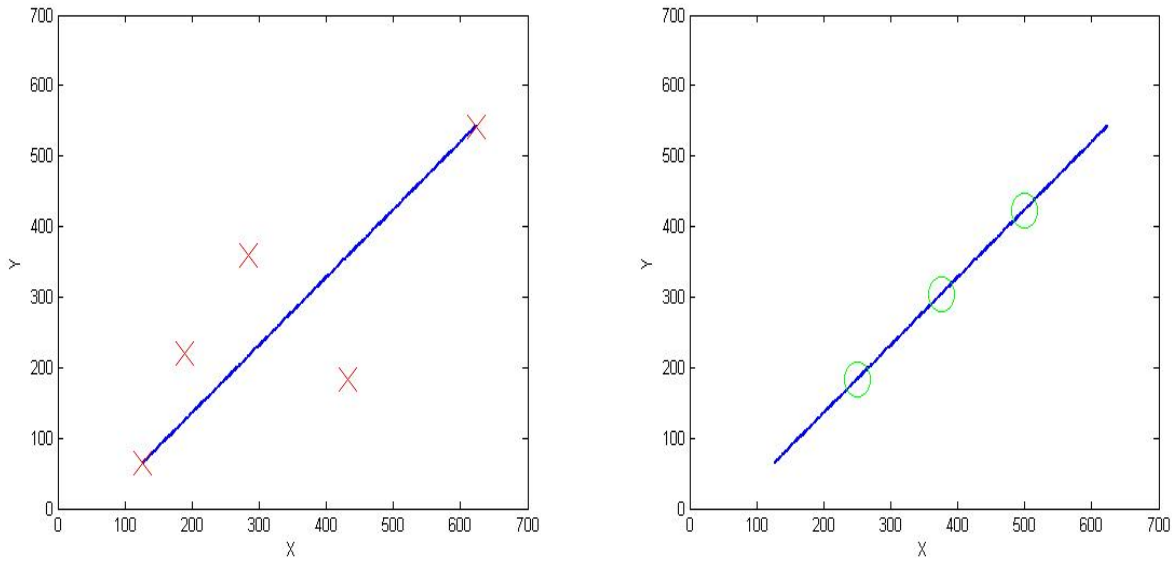

Figure 5.5. Proposed heuristic line formation algorithm steps 1, 2 and 3

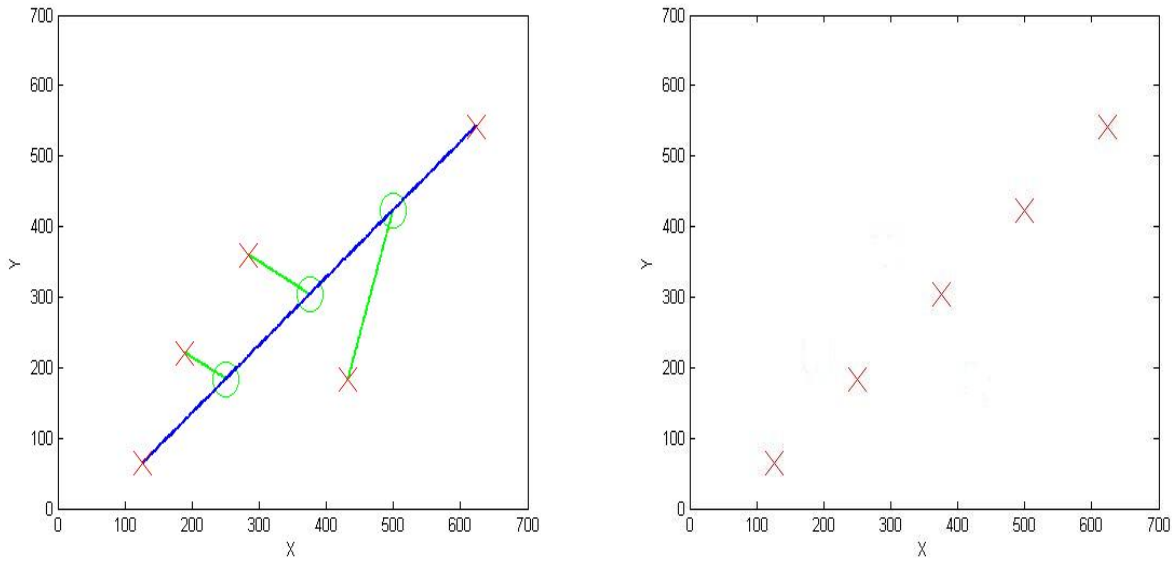

Figure 5.6. Proposed heuristic line formation algorithm step 4 and the resulting robot positions

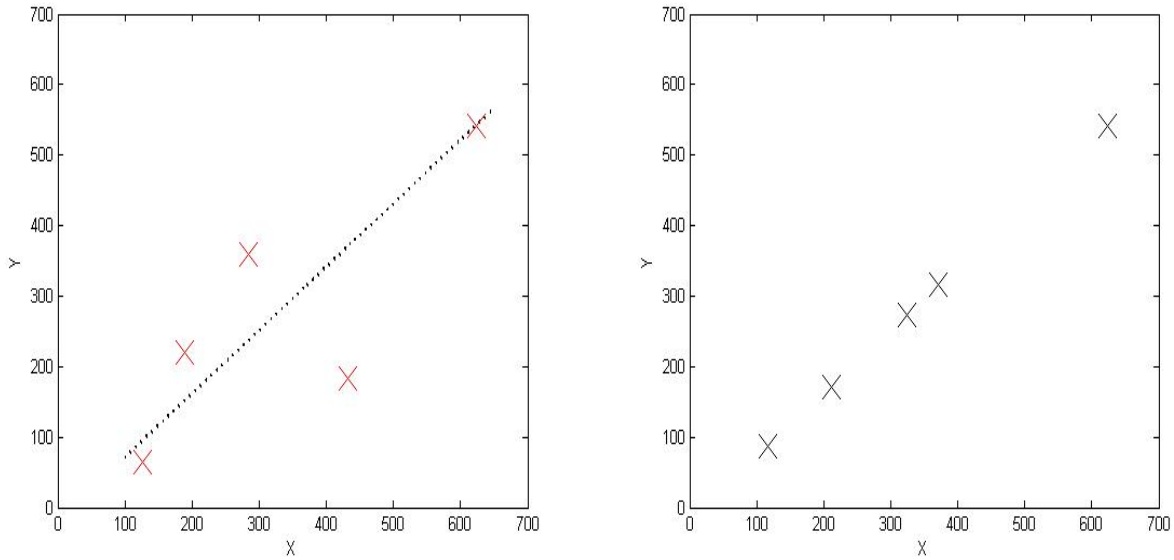

Figure 5.7. The line drawn by the brute force method and the resulting robot positions

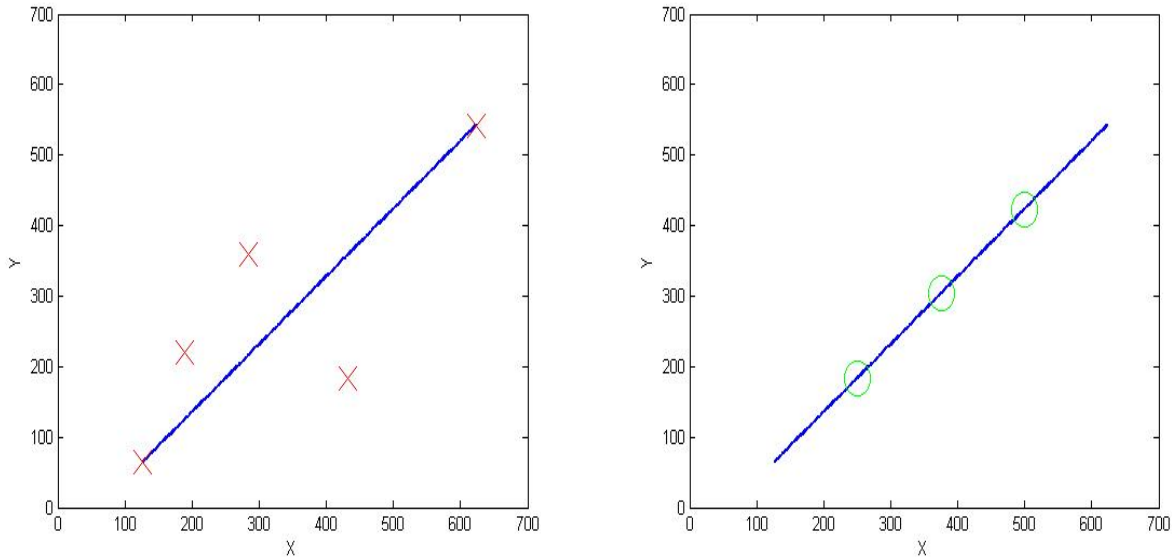

Figure 5.8. Proposed heuristic line formation algorithm steps 1, 2 (left) and 3 (right)

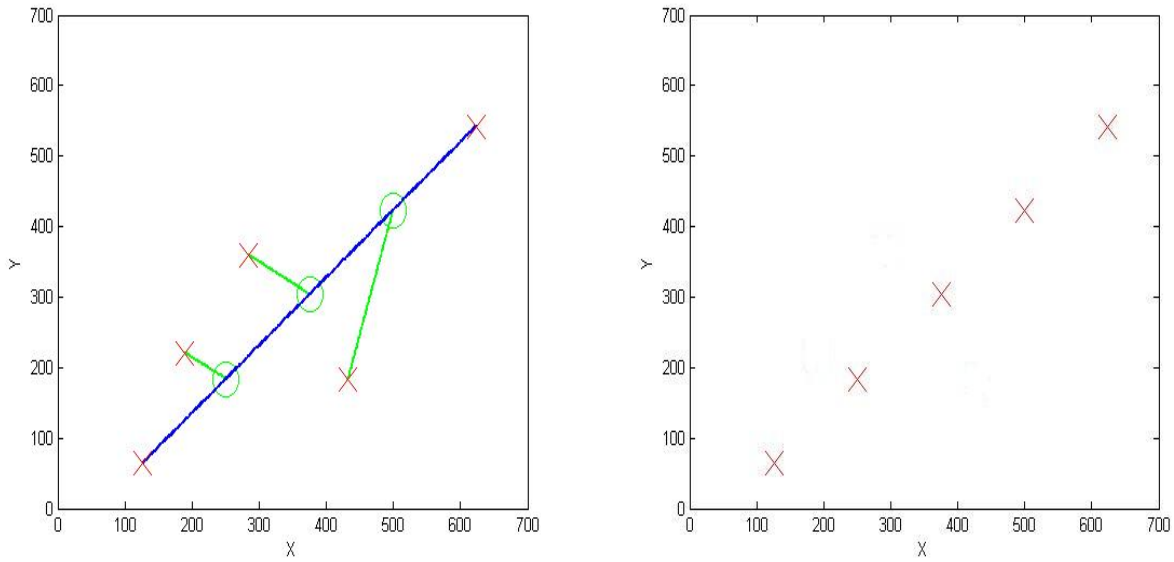

Figure 5.9. Proposed heuristic line formation algorithm step 4 and the resulting formation

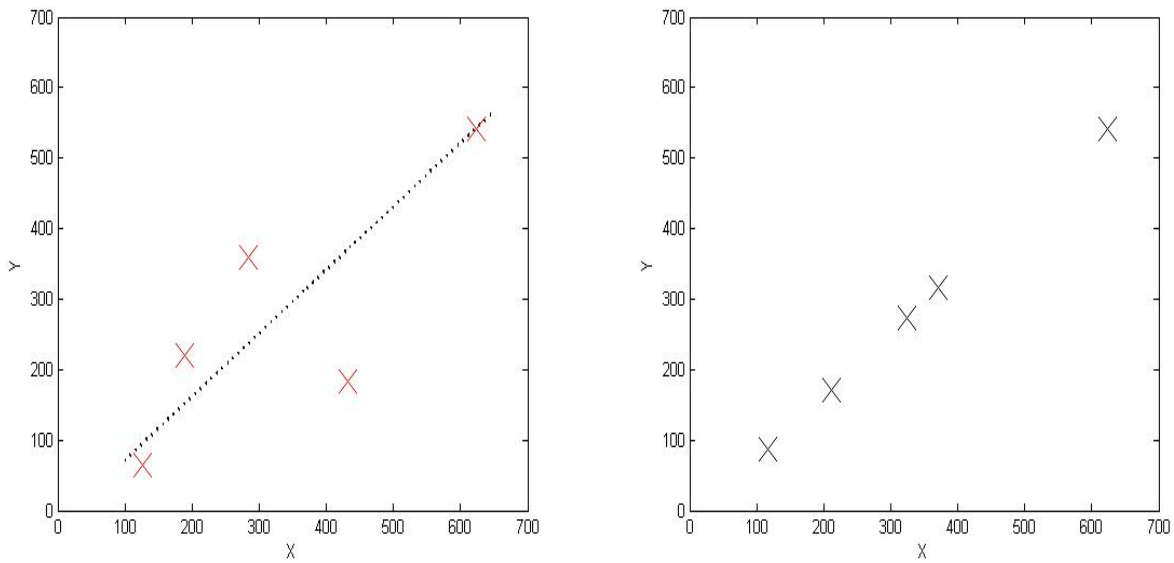

Figure 5.10. The line drawn by the brute force method and the resulting robot positions

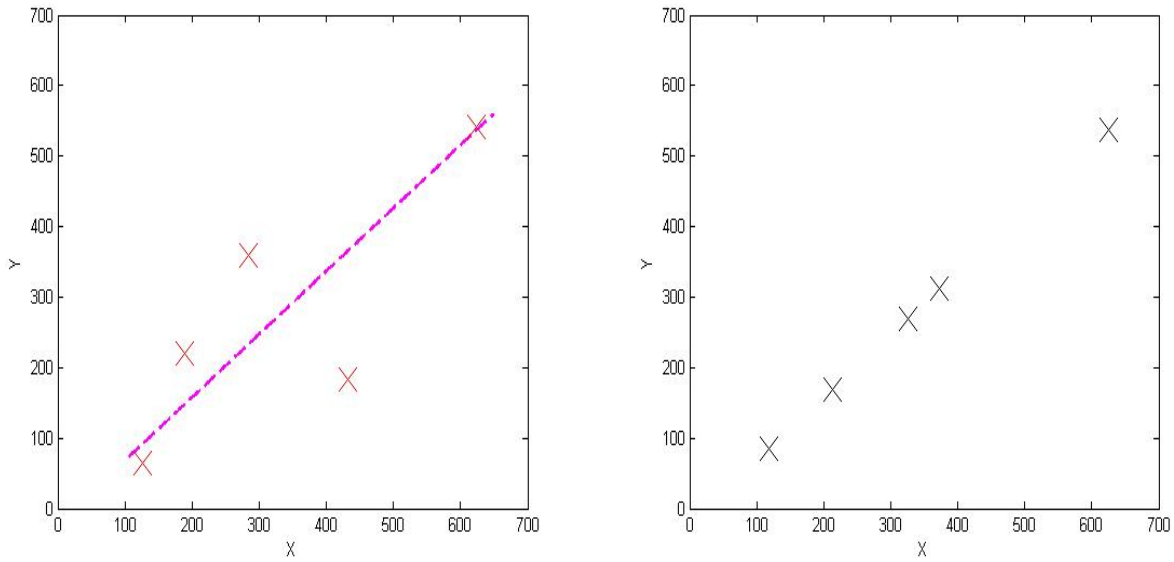

Figure 5.11. The line drawn by the least squares method and the resulting robot positions

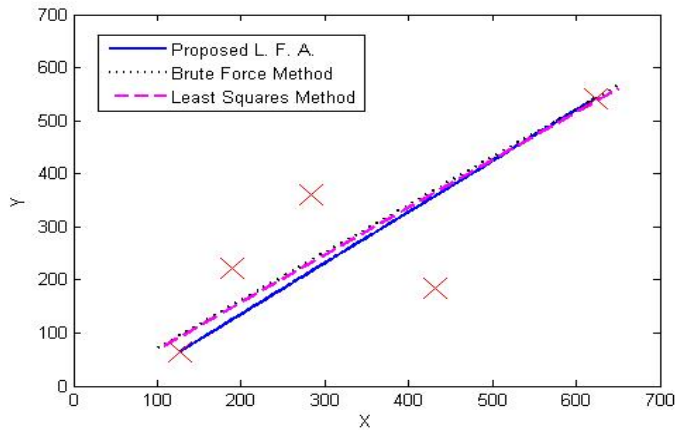

Figure 5.12. Lines created by the 3 line formation methods

5.3.2. Circle Formation

In this section, we extend our formation algorithm concept for forming a circle. Methodology used for circle formation is similar to the line formation. One approach is the self-developed heuristic algorithm and the other one is the brute force method. All of the methods are explained in detail below.

5.3.2.1. Proposed Heuristic Algorithm. Our approach for n robots represented by vector R_{ip} is as follows:

S1: As in the line algorithm, we first determine the robots with the maximum and minimum x coordinates. We then calculate the angle (α_{hc}) and distance (d_{hc}) between them using (5.41) and (5.43), respectively.

S2: The distance between the robots with minimum (R_{min}) and maximum (R_{max}) x coordinates will be the diameter of our circle. We fix R_{max} and mark $n-1$ points on our circle to place the remaining $n-1$ robots. We use the following equations to mark the corresponding locations

$$Center_{x_{hc}} = 0.5(x_{R_{max}} + x_{R_{min}}), \tag{5.55}$$

$$Center_{y_{hc}} = 0.5(y_{R_{max}} + y_{R_{min}}), \tag{5.56}$$

$$x_{dc,j} = Center_{x_{hc}} + D/2\cos(\alpha_{hc} + j2\pi/n), \tag{5.57}$$

$$y_{dc,j} = Center_{y_{hc}} + D/2\sin(\alpha_{hc} + j2\pi/n), \tag{5.58}$$

for $j = 1, 2, \ldots, n-1$, where $Center_{x_{hc}}$ and $Center_{y_{hc}}$ represents the center coordinates; D is the diameter of the circle; $x_{R_{max}}$ and $y_{R_{max}}$ denote the x and y

coordinates of R_{max}. $x_{R_{min}}$ and $y_{R_{min}}$ are defined similarly for R_{min}.

S3: After marking the points on the circle, the rest of the circle algorithm is the same as the line algorithm. Each robot is distributed one by one according to its relative distance to the desired location. The only difference is that we have $n-1$ robots to assign in the circle algorithm instead of the $n-2$ in the line algorithm.

5.3.2.2. Brute Force Method. This method is similar to the one in the line algorithm. The perpendicular distance of a point (x_p, y_p) to the perimeter of a circle (c_x, c_y, R_c) is given by

$$d_{pc} = |\sqrt{(x_p - c_x)^2 + (y_p - c_y)^2} - R_c| \tag{5.59}$$

and the total distance for multiple number of points is

$$d_{c_t} = \sum_{p=1}^{n} |\sqrt{(x_p - c_x)^2 + (y_p - c_y)^2} - R_c| \tag{5.60}$$

Similar to (5.46), (5.59) has no analytic solution for the minimization problem. A search for the 3 parameters of the circle that are involved in (5.59) is necessary to find the optimal parameter set. As a first step, search intervals are defined for these three parameters:

$$c_{x_{lt}} \leq c_x \leq c_{x_{ht}} \tag{5.61}$$

$$c_{y_{lt}} \leq c_y \leq c_{y_{ht}} \tag{5.62}$$

$$R_{c_{lt}} \leq R_c \leq R_{c_{ht}} \tag{5.63}$$

We then calculate the total distance d_{c_t} in 5.60 for each value of c_x, c_y and R_c. The parameters that minimize the total distance d_{c_t} are recorded as the optimal ones

$(c_{x_{opt}},\ c_{y_{opt}},\ R_{c_{opt}})$.

5.3.2.3. Simulation Results. We have run simulations to compare these 2 circle formation methods. The same robot positions in Table 5.2 are used for the simulations. Fig. 5.13 shows the first and the second steps of the heuristic circle algorithm. Fig. 5.14 shows the last step of the algorithm and the resulting robot positions. The results of brute force method is shown Fig. 5.15. Fig. 5.16 shows the circles drawn by these methods together with the initial robot positions.

If we compare the relative performance of these 2 methods by looking at Table 5.4, the brute force method minimizes the total displacement but its execution time is too long. The proposed heuristic circle algorithm needs much more total displacement compared to the brute force method but the robots are properly positioned, resulting in a better circle. Moreover, if the dimensions of the robots are taken into account, there is less risk of conflicting final robot positions on the circle.

Table 5.4. Performance comparison table for circle formation methods

Method Name	Total displacement (units)	Execution Time (sec)
Proposed Heuristic Algorithm	882	0.16
Brute Force Method	434	22.8

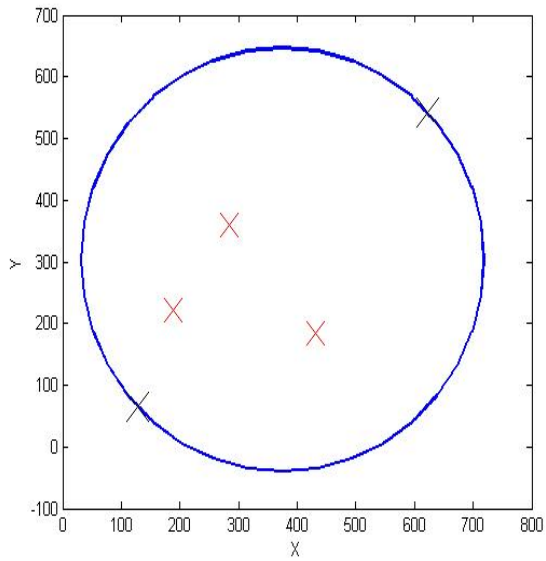

Figure 5.13. Proposed heuristic circle algorithm Steps 1 and 2

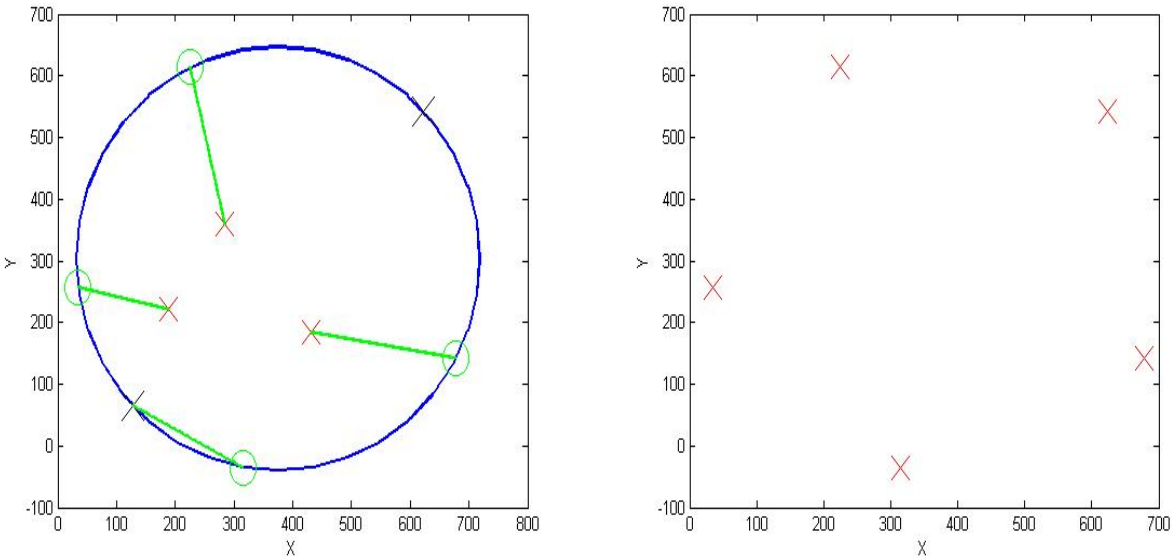

Figure 5.14. Proposed heuristic circle algorithm step 3 (left) and resulting positions of the robots (right)

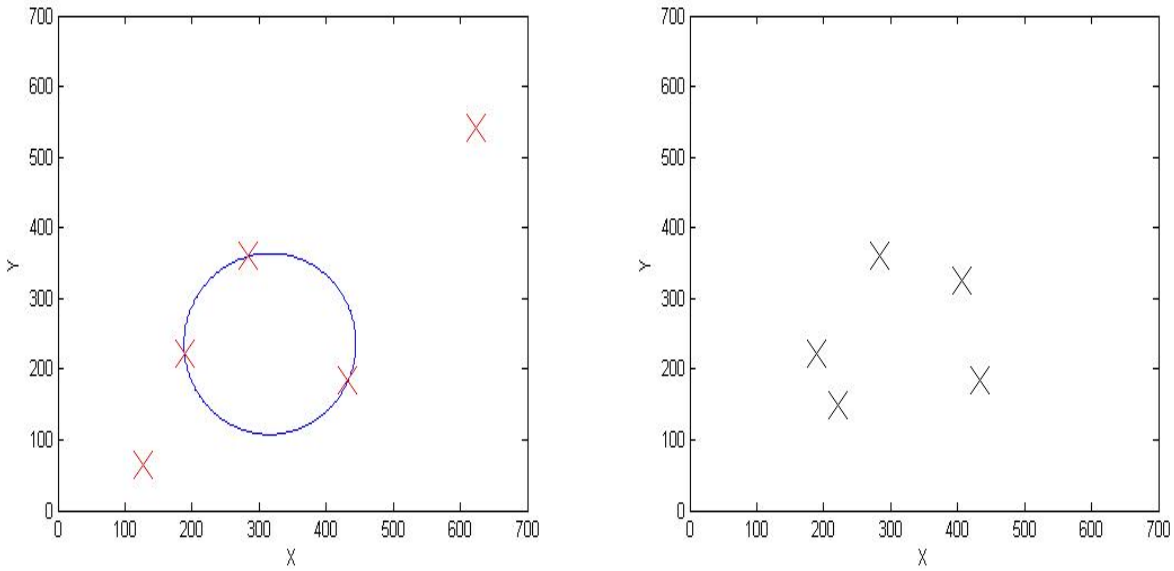

Figure 5.15. The circle drawn by the brute force method and the resulting robot positions

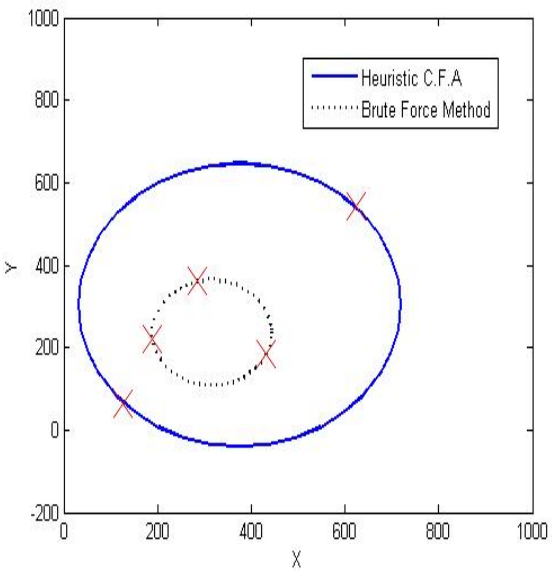

Figure 5.16. The circles drawn by 2 circle formation methods together with the initial robot positions

6. CONCLUSIONS

The multidisciplinary SSL is a challenging research platform that requires a lot of hardwork and theoretical background.

Our contributions in this project are, we have modeled and analyzed the DC motors used in robots, developed a controller for it and implemented it successfully in the real system. Vision based control of holonomic robots is implemented successfully, i.e., two robots can be controlled simultaneously using the advantage of their omni–directional drive system. Moreover, heuristic formation control algorithms are proposed in order to solve the multi–robot coordination problem for our team. Some of the performance measures of these algorithms are compared against different approaches.

Although a lot of achievements have been made towards building a competent SSL team, there are still improvements that can be made. The low level controller can be improved by adding feedback sensors such as gyroscopes or accelerometers in order to provide local feedback for the robot itself. Hence, the robots can correct or observe their own motion, decreasing the load of high level control. The image processing module can be developed further, to improve color detection and classification. Optimization of the software is also necessary in order to increase the speed of this module. Once the robots and the image processing module are optimized, the real time implementation of the proposed formation control algorithms and other game strategies can be carried out.

REFERENCES

1. Robocup Official Website, http://www.robocup.org/Intro.htm

2. Er-Force Team Description Paper, 2008.

3. B-Smart Team Description Paper, 2007.

4. CM Dragons Team Description Paper, 2008.

5. Rojas, R., *Omnidirectional Control*, Freie Universität, 2005

6. MC33035 DataSheet, http://www.onsemi.com/pub/Collateral/MC33035−D.PDF

7. MC33039 DataSheet, http://www.onsemi.com/pub/Collateral/MC33039−D.PDF

8. Ould-Khessal, N., Botnia: A Team of Soccer Playing Robots, 2^{nd} *International Conference on Autonomous Robots and Agents*, December, 2004.

9. Balch, T., J. Bruce and M. Veloso. Fast and Inexpensive Color Segmentation for Interactive Robots, *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol.3, 2061–2066, October, 2000.

10. Bruce, J., and M. Veloso. Fast and Accurate Vision-Based Pattern Detection and Identification, *In Proceedings of the IEEE International Conference on Robotics and Automation*, Vol.1, 1277–1282, September, 2003.

11. Robocup Systems Engineering Project, MS Thesis, Dept. of Electrical Engineering, Cornell University, 2002.

12. Alptekin, G., Y. Xiaoping, and O. Albayrak, "Line and Circle Formation of Distributed Physical Mobile Robots", *Journal of Robotic Systems*, 14(2), 63–76, 1997.

13. Chio, T., and T. Tarn, "Rules and Control Strategies of Multi-Robot Team Moving in Hierarchical Formation", *In Proc. IEEE International Conf. on Robotics and Automation*, vol. 2, 2701–2706, Sept. 2003.

14. Huang, J., S. Farritor, A. Qadi and S. Goddard, "Localization and Follow-the-Leader Control of a Heterogeneous Group of Mobile Robots", *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 2, 205–215, Apr. 2006.

15. Ren, W., N. Sorensen, "Distributed coordination architecture for multi-robot formation control", *Robotics and Automation Systems*, 56, 324–333, 2008.

16. Balch, T., R. C. Arkin, "Behavior-based Formation Control for Multi-robot Teams", *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, 926–939, Dec. 1998.

17. Lawton, J., R. Beard and B. Young, "A Decentralized Approach to Formation Maneuvers", *IEEE Transactions on Robotics amd Automation*, vol. 19, no. 6, 933–941, Dec. 2003.

18. De la Cruz, C., and R. Carelli, "Dynamic Modeling and Centralized Formation Control of Mobile Robots", *IEEE Annual Conference on Industrial Electronics*, 3880–3885, Nov. 2006.

19. Fredslund, J., and M. J. Mataric, "Robot Formations Using Only Local Sensing and Control", *In. Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 308–313, 2001.

20. Gazi, V., "Stability of an Asynchronus Swarm With Time-Dependent Communication Links", *IEEE Transactions on Systems Man and Cybernetics*, vol. 38, no. 1, 267–274, Feb. 2008.

21. Hsu, H., and A. Liu, "Multiagent-Based Multi-team Formation Control for Mobile Robots", *Journal of Intelligence and Robotic Systems*, 42, 337–360, 2005.

22. Lewis, M. A., and K.-H. Tan, "High Precision Formation Control of Mobile Robots Using Virtual Structures", *Autonomous Robots*, 4, 387–403, 1997.

23. Ren, W., and R. W. Beard, "Virtual Structure Based Spacecraft Formation Control with Formation Feedback", *In. Proc. AIIA Guidance, Navigation, and Control Conference and Exhibit*, no. 2002-4963, Aug. 2002.

24. Sun, D., and C. Wang, "Controlling Swarms of Mobile Robots for Switching between Formations Using Synchronization Concept", *IEEE International Conference on Robotics and Automation*, 2300–2305, Apr. 2007.

25. Ratzilla Preliminary Design Report, University of California, Davis, Nov. 2005.

26. Nise, N.S., Control Systems Engineering, 4th edition, New Jersey: John Wiley&Sons, 2004.

27. CMU Digital Camera Driver, retrieved from http://www.cs.cmu.edu/ iwan/1394/

28. Welch, G., and G. Bishop, "An Introduction to the Kalman Filter", Department of Computer Science, University of North Carolina, Jul. 2006

29. Negenborn, R., "Robot Localization and Kalman Filters", Institute of Information and Computing Sciences, Utrecht University, Sept. 2003

30. Least Squares Fitting – Perpendicular Offsets, Retrieved 10 December 2008 from http://mathworld.wolfram.com/LeastSquaresFittingPerpendicularOffsets.html