DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

EFFICIENT MULTIPLE ROBOT PATH PLANNING ALGORITHM

by
Çağdaş Yetişenler

September, 2005
İZMİR

EFFICIENT MULTIPLE ROBOT PATH PLANNING ALGORITHM

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical&Electronics Engineering

by
Çağdaş Yetişenler

September, 2005
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"EFFICIENT MULTIPLE ROBOT PATH PLANNING ALGORITHM"** completed by **Çağdaş YETİŞENLER** under supervision of **Asst. Prof. Dr**. **Ahmet ÖZKURT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Asst. Prof. Dr. **Ahmet ÖZKURT**

Supervisor

.....

(Jury Member)

.....

(Jury Member)

Prof.Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to thank to my supervisor, Asst. Prof. Dr. Ahmet ÖZKURT, Yüksek Teknoloji AŞ engineers, Recai YETİŞENLER, Dr. Serdar AKDURAK and Yavuz ÖZDEMİR and my parents for their help and patience.

Çağdaş YETİŞENLER

EFFICIENT MULTIPLE ROBOT PATH PLANNING ALGORITHM

ABSTRACT

This thesis describes the necessary hardware and software developments to prepare a group of mobile robot vehicles to participate at RoboCup. Robotic soccer is a highly complex domain that has become a significant challenge problem in both mobile robotics and artificial intelligence. This project describes the global vision, path planning algorithm, behavior based control for strategy and robot architecture for robot soccer.

The aim of the vision system for robot soccer is to track and predict the motion states of six agents and a ball. The work about the tracking includes: the color-based segmentation which is used to locate the colored objects in the image, the blob analysis which is adopted to calculate the positions of the objects in the image and to filter noise. Robot recognition and labeling is also done in vision system.

The application of Cellular Automata to the problem of robot path planning is presented. It is shown that a Cellular Automata allows the efficient computation of an optimal collision free path from an initial to a goal configuration on a physical space cluttered with obstacles. The cellular space represents a discrete version of the workspace for a mobile robot.

For Strategy of robot soccer Team strategy architecture is divided into three levels: (1) at the lowest level, agents must be able to follow simple behaviors. (2) In second level it uses combination of these behaviors to build some complex abilities roles (to get behind a ball, move toward a goal, take the Ball from corner, and so on). (3) In level three, Robots are ordered with individual and team-oriented strategies, this level controls role selection.

In this study, an agent based multi robot path planning algorithm and required fundamental structure for robotic structure is designed. It has been planned that the

information gathered using this study can be used in multiple agent robotics applications.

Keywords: RoboCup, Robot Soccer, Tracking, Strategy, Global Vision, Path Planning, Robot vehicles.

ETKiN ÇOKLU ROBOT YOL PLANLAMA ALGORiTMASI

ÖZ

Bu tez RoboCup'a katılacak bir grup mobil robot üretiminde gereken mekanik ve yazılımsal geliştirmeleri açıklamaktadır. Robot futbol müsabakaları, mobil robot tekniği ve yapay zeka alanında birçok zorluğu aşmayı gerektiren, karmaşık bir alandır. Bu proje, robot futbol maçlarındaki kameralı görüntü sistemi, güzergah belirleme algoritması, strateji için davranış bazlı kontrol ve robot yapısını açıklamaktadır.

Kameralı görüntü sisteminin amacı altı robot ve bir toptan oluşan elemanların yerlerini belirlemek ve sonraki hareketlerini tahmin etmektir. Yer belirleme işlemi, görüntüdeki renkli cisimleri tanımlamak için kullanılan renk bazlı ayırt edim işleminden ve nesnelerin yerini belirlemek ile gürültüyü filtre etmek için kullanılan blob analizinden oluşur. Robot tanımlama ve isimlendirme işlemleri de görüntü sistemi tarafından gerçekleştirilir.

Robotlardaki güzergah belirleme sorunu için Cellular Automata metodu sunulmuştur. Cellular Automata metodu, robotun belirli bir hedefe sahadaki engellere çarpmadan en uygun şekilde gitmesini sağlar. Cellular Automata yöntemiyle hücrelere bölünmüş saha, robotlar için yeni bir çalışma alanı oluşturmaktadır.

Robot futbolu için oluşturulan strateji mimarisi üç seviyeden oluşmaktadır: (1) En alt seviye elemanların basit hareketlerini kontrol eder. (2) İkinci seviyede, birinci seviyede kullanılan basit hareketlerin birleşmesiyle daha karmaşık roller oluşturulur.(Topun arkasına geçmek, kaleye doğru ilerlemek, topu köşelerden çıkarmak, vb). (3) Üçüncü seviyede robotlara bireysel ve takım bazlı stratejiler atanır. Bu seviye rol atama işlemlerini gerçekleştirir.

Bu çalışmada, küçük robot araçları için yol takibi algoritması ve program yapıları tasarlanmıştır. Planlanan diğer bir amaç ise bu çalışmada elde edilen bilgilerin ve tecrübelerin başka çoklu robot uygulamalarında kullanılmasıdır.

Anahtar Sözcükler: RoboCup, Robot Futbol'u, Kameradan Görüntü Takibi, Strateji, Görüntü Sistemi, Güzergah belirleme, Robot araçları.

CONTENTS

Page

CHAPTER ONE

INTRODUCTION

1. Introduction

Robotic soccer is a highly complex domain that has become a significant challenge problem in both mobile robotics and artificial intelligence. Two well-known annual competitions, ROBOCUP (Originally called as Robot World Cup Initiative) and FIRA (Federation of International Robosoccer Association) allow teams to compete in a number of different leagues distinguished by robot size and hardware restrictions. Because the domain is based on teamwork in a real-time environment, it has been found agent-based control of individual robots to be a very convenient approach to designing a robotic soccer team.

1.1 Robocup

RoboCup (Originally called as Robot World Cup Initiative) is an international research and education initiative. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education. For this purpose, RoboCup chose to use soccer game as a primary domain, and organizes RoboCup: (The Robot World Cup Soccer Games and Conferences). In order for a robot team to actually perform a soccer game, various technologies must be incorporated including: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup is a task for a team of multiple fast-moving robots under a dynamic environment. RoboCup also offers a software platform for research on the software aspects of RoboCup.

The concept of soccer-playing robots was first introduced in 1993. Following a two-year feasibility study, in August 1995, an announcement was made on the introduction of the first international conferences and football games. In July 1997, the first official conference and games were held in Nagoya, Japan. Followed by

Paris, Stockholm, Melbourne and Seattle where the annual events attracted many participants and spectators. The 6th RoboCup2002 was held in Fukuoka, Japan in cooperation with Busan, Korea, while the 7th edition in 2003 took place in Padua, Italy, then in 2004 in Lisbon, Portugal. The events were covered by national and international media all over the world.

While soccer game is used as a standard problem where broad-range of efforts will be concentrated and integrated, competition is only a part of RoboCup activity. Current activities of the RoboCup consists of:

- Technical Conferences
- RoboCup International Competitions and Conferences
- RoboCup Challenge Programs
- Education Programs
- Infrastructure Development

Nevertheless, RoboCup International Competitions and Conferences is the central pillar of our activity, where researchers can get together and evaluate research progress.
RoboCup has three major domains:

RoboCupSoccer:

- Simulation League
- Small Size Robot League (f-180)
- Middle Size Robot League (f-2000)
- Four-Legged Robot League
- Humanoid League (from 2002)

RoboCupRescue:

- Rescue Simulation League
- Rescue Robot League

RoboCupJunior:

- Soccer Challenge
- Dance Challenge
- Rescue Challenge

1.1.1 RoboCupSoccer

The main focus of the RoboCup activities is competitive football. The games are important opportunities for researchers to exchange technical information. They also serve as a great opportunity to educate and entertain the public. RoboCupSoccer is divided into the following leagues:

Figure 1.1 Robocup Small size League

Simulation league: Independently moving software players (agents) play soccer on a virtual field inside a computer. Matches have 5-minute halves. This is one of the oldest fleet in RoboCup Soccer.

Small-size robot league (f-180): Small robots of no more than 18 cm in diameter play soccer with an orange golf ball in teams of up to 5 robots on a field with the size of bigger than a ping-pong table. Matches have 10-minute halves.

Middle-size robot league (f-2000): Middle-sized robots of no more than 50 cm diameter play soccer in teams of up to 4 robots with an orange soccer ball on a field the size of 12x8 meters. Matches are divided in 10-minute halves.

Four-legged robot league: Teams of 4 four-legged entertainment robots (SONY's AIBO) play soccer on a 3 x 5 meters field. Matches have 10-minute halves.

Humanoid league: This league was introduced in 2002 and the robots will have their third appearance ever in this year's RoboCup. Biped autonomous humanoid robots play in "penalty kick," and "1 vs. 1", "2 vs. 2" matches. "Free style" competitions are to be expected as well.

The Dream:

The ultimate goal of the RoboCup Initiative to be stated as follows:

By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup.

1.2 Fira Cup

Robot soccer can be portrayed as a competition of advanced robot technology within a confined space. It offers a challenging arena to the young generation and researchers working with autonomous mobile robotic systems. It is hoped that FIRA's flagship event, called the FIRA Robot World Cup (or the FIRA Cup in short), which started in 1996, together with many other FIRA events, will help generate interests in robotics in the young minds. Through these events, FIRA hopes to help them better understand and appreciate, with interests, the scientific concepts and technological developments involved. FIRA believes that some of these interests will fuel scientific and engineering skills that ultimately develop into research outcomes to serve mankind in a variety of ways. Ever since its establishment, FIRA has had venues for its annual FIRA Cup in Australia, Brazil, China, France and Korea. Making progress over successive years since 1996, FIRA Cup has now attained world recognition as a robot festival.

1.2.1 Fira Cup Competition Categories

Like soccer, robot-soccer has well-defined game rules. The FIRA Cup event is organized into several categories, including the Micro-Robot Soccer Tournament (MiroSot), the Simulated Robot Soccer Tournament (SimuroSot) and the Humanoid Robot Soccer Tournament (HuroSot). These games are played under the watchful

eyes of a human referee and the participants who are the robot players' managers and trainers.

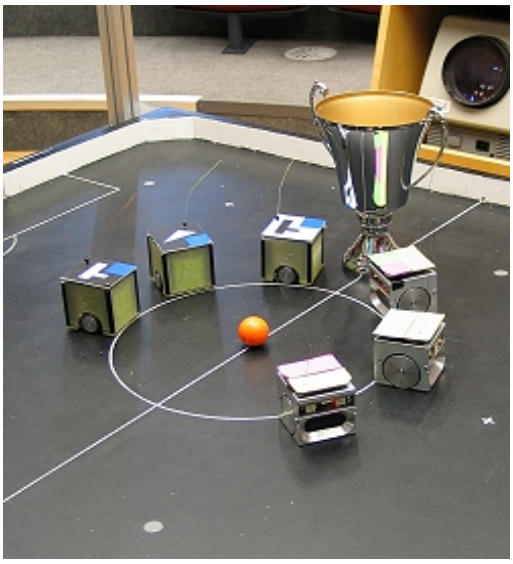

Figure 1.2 Fira Cup MicoSot

MicroSot: In MiroSot, participants need to devise good strategies using artificial intelligence (AI) techniques, and develop sharp sensing and precise real-time control for the physical robot-soccer players. These basic capabilities are needed for the robot-soccer players to cooperate and coordinate autonomously (i.e., with "human hands-off"), and are crucial to winning the game against an opponent team. As many who have witnessed a MiroSot game will testify, the excitement always runs high especially when two strong robot-soccer teams meet. During the match, the robot players autonomously tackle many unfamiliar situations that arise due to the different strategies, hardware and control software technologies employed in the opponent robot players. Like in a FIFA World Cup soccer match, no one knows for sure which team will win until the final whistle.

SimuroSot: In SimuroSot, the game is played on a computer between two teams. With no physical robot involved, the game is decidedly one of complex strategy development using advanced AI techniques.

HuroSot: In HuroSot, a robot player is more human-like in that it has two legs, hence the term humanoid. Given the current state of the art, the participants are only expected to endow their humanoid robot with, for instance, the ability to walk steadily, avoid obstacles simulating stationary opponent players and take penalty shots, all under the remote guidance of its human trainer.

1.3 Overall Structure of the Project Team

One specific characteristics of the RoboCup small size league is the high speed of the robots. They move at above 2 m/sec, crossing the whole field in about 2.5 sec. This fact demands a highly adaptive and reactional behavior of the robots, with a minimized system delay. Figure 1.3 gives a brief description of how the whole system of the Team small size robots is functioning. In Robocup, the small size architecture uses as single sensor one or more global cameras, positioned over the field of play. The dimensions of the robots, the field, the markers on the field and the height of the camera are defined in the small size league rules (visit www.robocup.org for more information and the current rules).

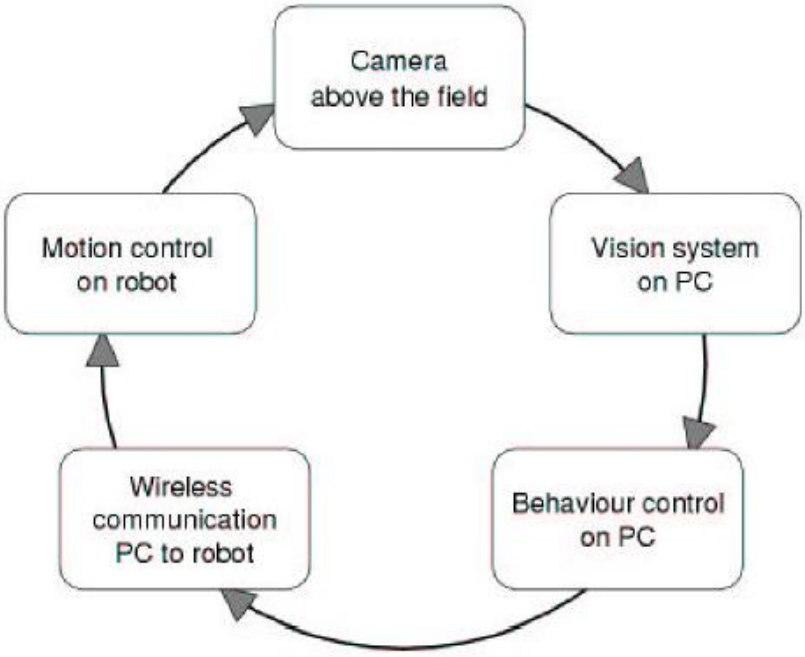

Figure 1.3 The Overall Control Circle

The Project team uses one global camera at 25 frames/sec to sensor the field of play. The images are sent to a central processing unit (PC), which undertakes the calculations. The first step is the processing and evaluating the images from the camera. In every frame the robots and the ball are found, using the color information. The ball is, for example, always red, and the robots have one mandatory team marker (blue or yellow in the middle of the robot) and other color markers on the cover, which help to identify each exact position and direction. Figure 1.4 shows the view from global camera.

Figure 1.4 The view from global camera.

The evaluated information from the camera is being merged and the final positions and orientations of the robots and the ball are computed. Vision system and Robot recognition parts are examined in Chapter 3. This “world” is then being passed to the next module, the Strategy module (behavior control). The behavior control uses the positions and orientations of the robots together with the position of the ball to calculate the best behavior step for every robot. The behavior is planned in a top-down matter, deciding first what the team as a whole has to do and then precising these decisions down to the robots. The behavior control is explained in detail in Chapter 5. The Path planning algorithm is also used in strategy but to understand better it is examined in Chapter 4.Path Planning is an important task for moving robots efficiently in match field, there are several method are developed on

this purpose. The next step is to pass the information to the robots. The commands for every direction (X and Y axis, rotation) and motor power values are being calculated and passed to the robots via wireless communication. On the robot itself, only a minimal intelligence is provided by a Pic16F877 microcontroller. Robot architecture is examined in Chapter2. Figure 1.5 shows the soccer field, test and improvement areas.

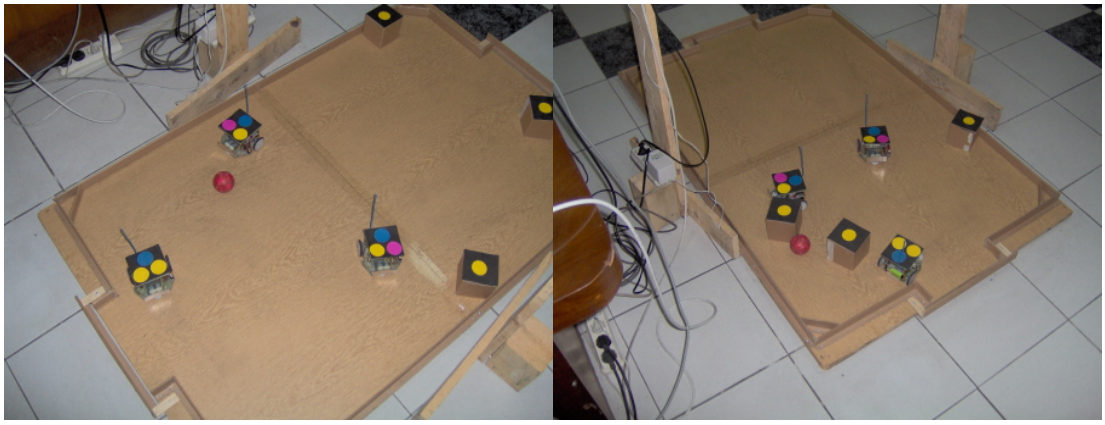

Figure 1.5 The view of Match Field

CHAPTER TWO

ROBOT ARCHITECTURE

2. Robot Architecture

In this chapter Hardware and Software of Robot structure is examined.

2.1 Robot Hardware

Robot's hardware is constructed with three wheels, an electronic card, two batteries, a switch and two empty cards for the body. Two back wheels simply built up with plastic and rubber, front tires is a ball caster which is a small steel ball that rests on metal rollers inside a plastic frame. Robot has two batteries one is for electronic card (+9 volt) other one is for DC motor which is a mobile phone battery +3,6 v 1600mAh Li-ion. +9 volt battery is mounted back side of the robot, behind the electronic card. +3,6 volt battery is mounted under the robot body which is shown in Figure 2.1. Robots body is constructed with 3 layers first layer is consist of power switch, Gear Box, Wheels, Batteries and ball holder. Second layer includes the electronic card, the Rf module and the antenna, this layer has connections to Dc motors and to batteries. Third layer is used for a place to stick the identification cartoon.

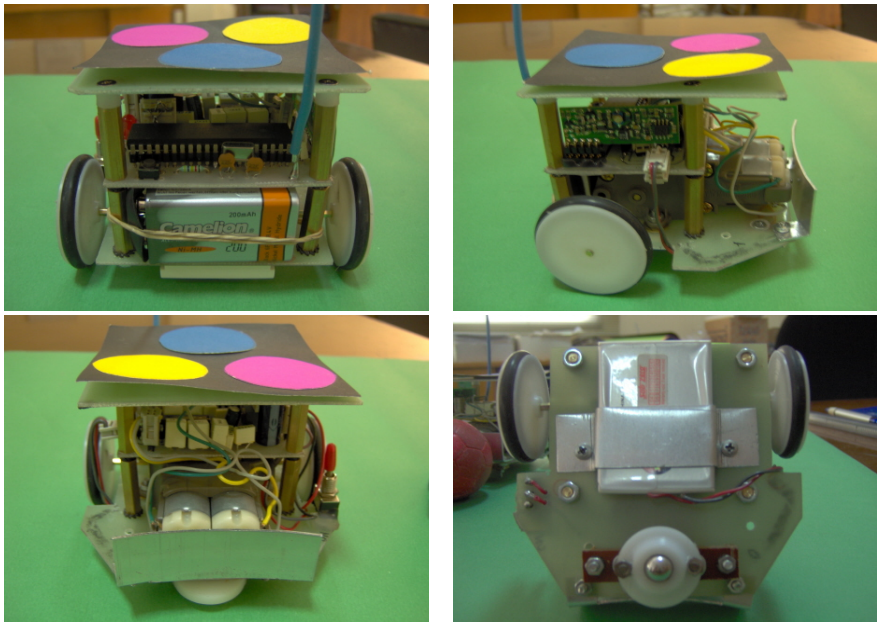

Figure 2.1 Robot Architecture

2.1.1 Robot Gear box

A twin motor gearbox is used for the robot movement Figure 2.2. The Tamiya twin-motor gearbox is a small (3in long) plastic gearbox. It contains two small DC motors that drive separate 3mm hexagonal output shafts. There are two ways to put the kit together: with a high-speed 58:1 gear ratio or with a slower 203:1 gear ratio. Either way, the motors provide plenty of power to drive any small robot.

The two motors in the twin-motor gearbox are rated for 3 volts but will typically work fine up to 6 volts. When using this product with the dual serial motor controller (or any motor driver based on L293D integrated circuit), keep in mind that up to a few volts (depending on the load) are lost on the transistors of the L293D, so that a 4.5 volt battery delivers about 3 volts to the motors, giving them a reasonable amount of power. Motor overheating can be caused by excessive stalling, even at very low voltages.

Figure 2.2 Twin Motor Gear Box

2.1.2 Robot Electronics

Robot's electronic card has two important parts; first part is the microcontroller part, it is preferred to use Pic16F877 microcontroller which has the PWM module. Motor power is controlled with pulses (PWM) by this method motor's speed become variable. The first process after power command is received from the Rf module, is to create a Pulse Width Modulation control. The PWM basically controls the speed

of the motor; the switches of the Motors which are opened and closed at different rates in order to apply different average voltages across the motor. Figure 2.3 shows a sample for PWM

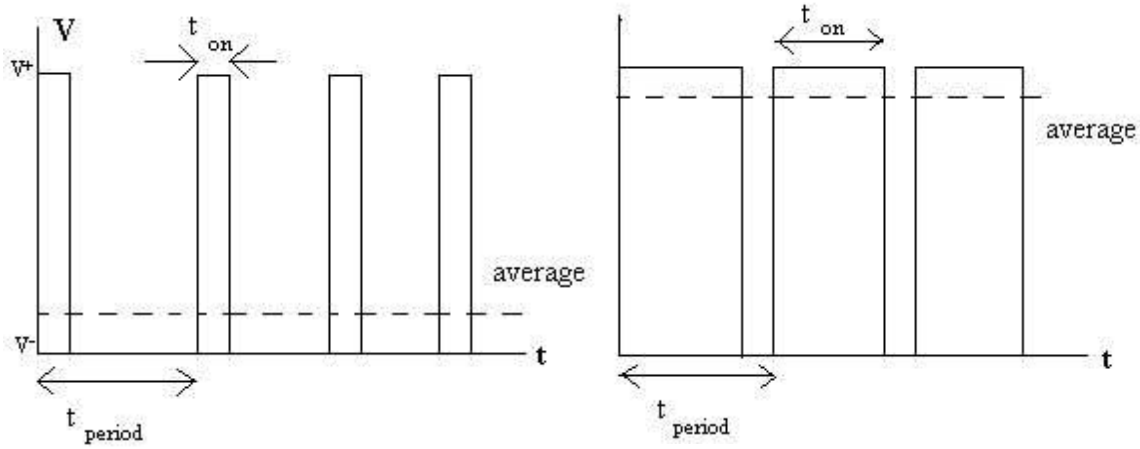

Figure 2.3 Pulse Width Modulation.

V is the voltage across the motor and t is time. By switching quickly, an average voltage can be created across the motor. The speed of the motor (average voltage applied) can be adjusted by changing the pulse-width ratio: Pulse-Width Ratio = t (on) / t (period).

Second important part of the robot electronic is the Rf module. It's preferred to use The ARX-34 UHF ASK (UDEA) data receiver which is developed to cover a band plan ERC Recommendation on Short Range Device (SRD) in the range of 434MHz ISM band. ARX-34 is designed as a module that will be integrated into a user system. It can be regarded more like a special component for part of an electronic system. The user needs basic knowledge about electronics. Special knowledge about RF technology is helpful, but the most difficult parts are integrated into the modules to enable easy operation. Technical specifications are given in Table 2.1.

Most important for effective data transmission is selection of a good antenna, and RF grounding, both for the transmitter and receiver. Without an antenna it is impossible to transmit data over a long distance range. The ATX-34 has a simple antenna input pin. Any suitable UHF antenna can be connected to it. The easiest way

to connect an antenna to the ATX-34 is to solder a 17.3 cm. (434MHz) wire directly to antenna input. If the receiving antenna is installed away from the module, a 50-Ohm Coax antenna wire can be used. The shielding of the antenna wire should be soldered to the case near the antenna input of the module. For serial communication baud rate is selected as 2400 bit/sec which is the maximum speed of the Rf communication. Speed is an important task in Robocup for three robots serial communication will take a time of 30ms (3 robots 9 bytes). This time will increase with robot number, for faster application this Rf module needs to be changed.

Table 2.1 Technical Features of RF module.

	Min.	Typ.	Max.	Unit
Frequency		433.920		Mhz
Bandwidth		±2		Mhz
Data Rate	0.3		2.4	Kbit/s
RF Sensitivity		-108		dBm
Voltage Supply	4.9		5.1	Vcc
Supply Current		5		mA
Logic ”0” DOUT Voltage	0		0.1 * Vcc	Vdc
Logic ”1” DOUT Voltage	0.8 * Vcc		Vcc	Vdc
RX on time		10		ms
Working Temp.	-10		+55	°C

Rf receiver module has 5 pins these are; Antenna, Gnd, Vcc, Aout and Dout. Dout pin is connected to microcontrollers serial port, Rf modules are working transparent signal is modulated with Ask and then demodulated for transforming data to RS232 serial communication format. Analog out pin is used as a test pin. The demodulated signal can be seen on 1,5 VDC level.

2.2 Robot Software

Robot program is developed in Pic Basic Pro compiler. Algorithm has three parts; first part includes the port reading (Robot number) and the periodic lighting signal

for the led (which is good for online battery test). Second part is the serial communication part, Pic16f877 has a serial interrupt and when a command is received it will be buffered and will be put in an array. Three byte command group will be buffered, if any command fill that buffer program will call the identification algorithm. First it controls the Robot number, second controls the Way number and third controls the Power number, if any wrong data is received this process will be canceled and Robot will not do any movements. If command is received correctly, program will choose and load the PWM register with specific power periods and also selects the way for dc motors. In Figure 2.4 this algorithm is shown.

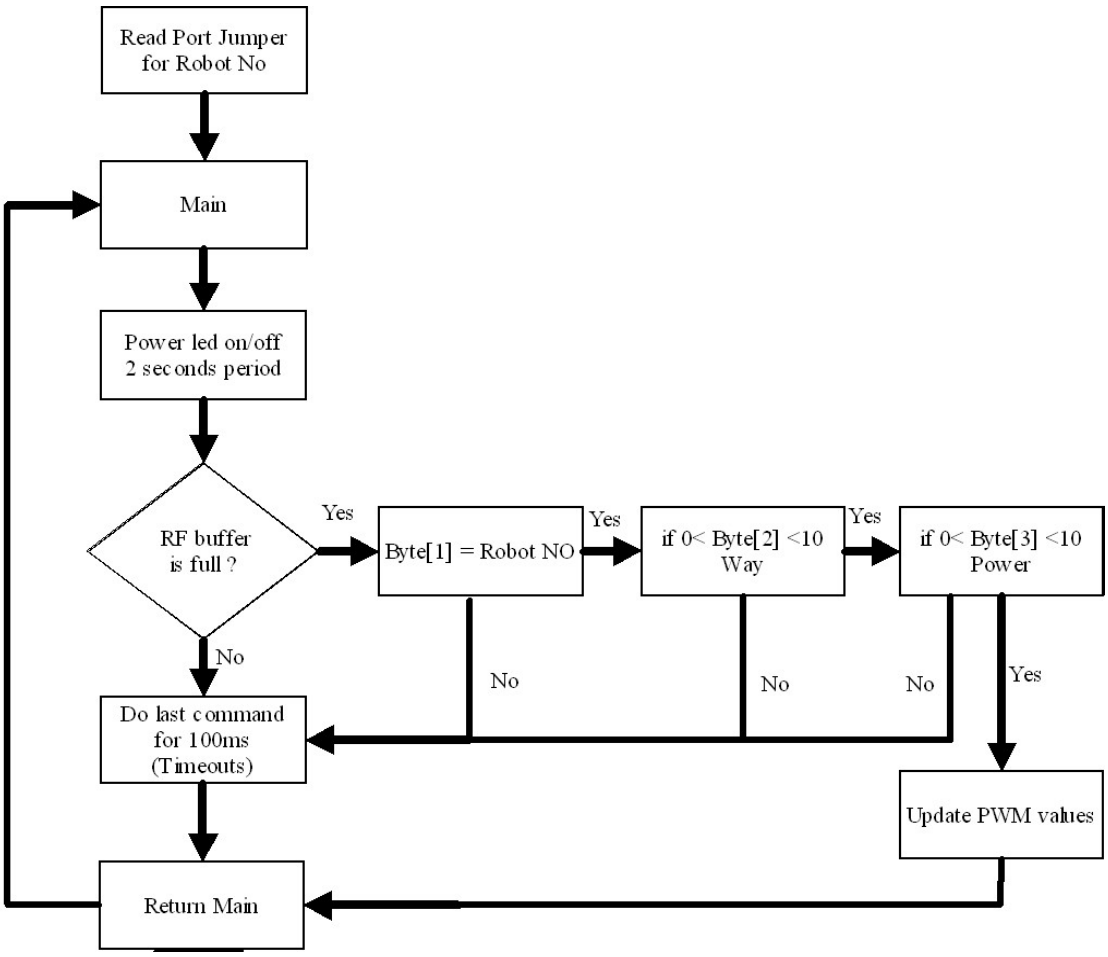

Figure 2.4 Robot's Pic16F877 Program

2.3 Computer Communication Module

The transmitter is connected to the computer through a RS-232 connection cable and mounted on the serial interface board. In Transmitter part The ATX-34 UHF ASK (UDEA) data transmitter is used which is developed to cover a band plan ERC Recommendation on Short Range Device (SRD) in the range of 434MHz ISM band.

Data protocol should be as below: TX: preamble + sencron + data1 +.....+ dataX. Preamble is used in order to establish synchronization of the frames required for the transmission of data between transmitter and receiver, this signal is added to the front of the frame, and may consist of a bit stream of alternately repeated 0s and 1s. Data is sent to transmitter with 2400 bit/sec, communication is done in one direction. Robot doesn't have the ability to send data back to computer and also computer can't receive any data from robot, command is send in one direction. This method will provide some errors but in this application speed is the key, so while in match field computer doesn't care if robot took that command or not, it keeps sending until it does its work. Figure 2.5 shows the computer Rf module.

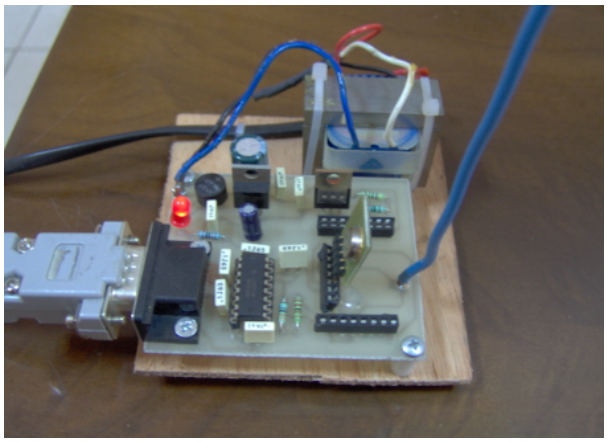

Figure 2.5 Computer Rf Transmitter.

Robot card scheme and Computer RF transmitter card scheme are shown in Figure 2.6 and Figure 2.7, these cards are developed in Protel 99se program.

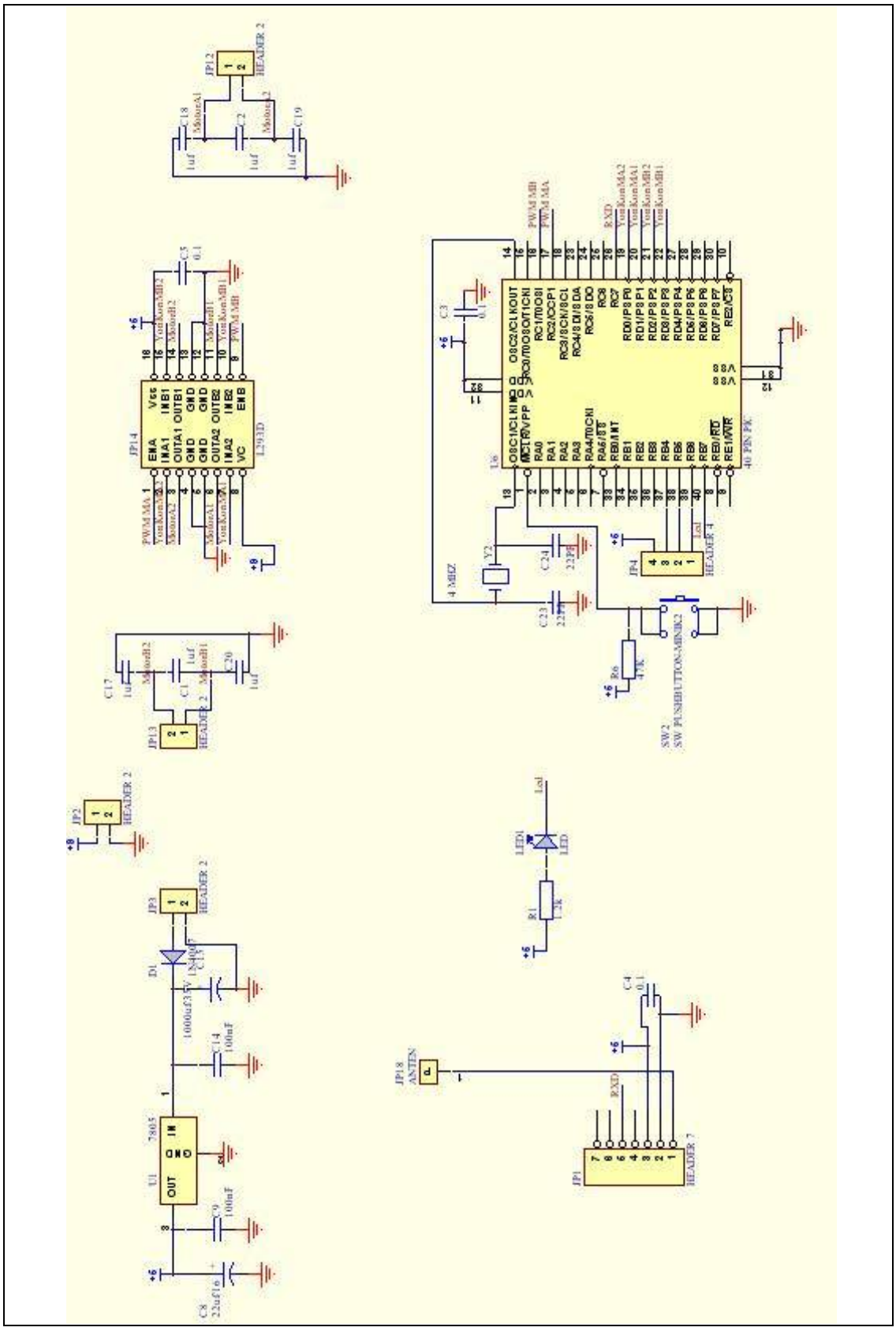

Figure 2.6 Robot Electronic Card Scheme.

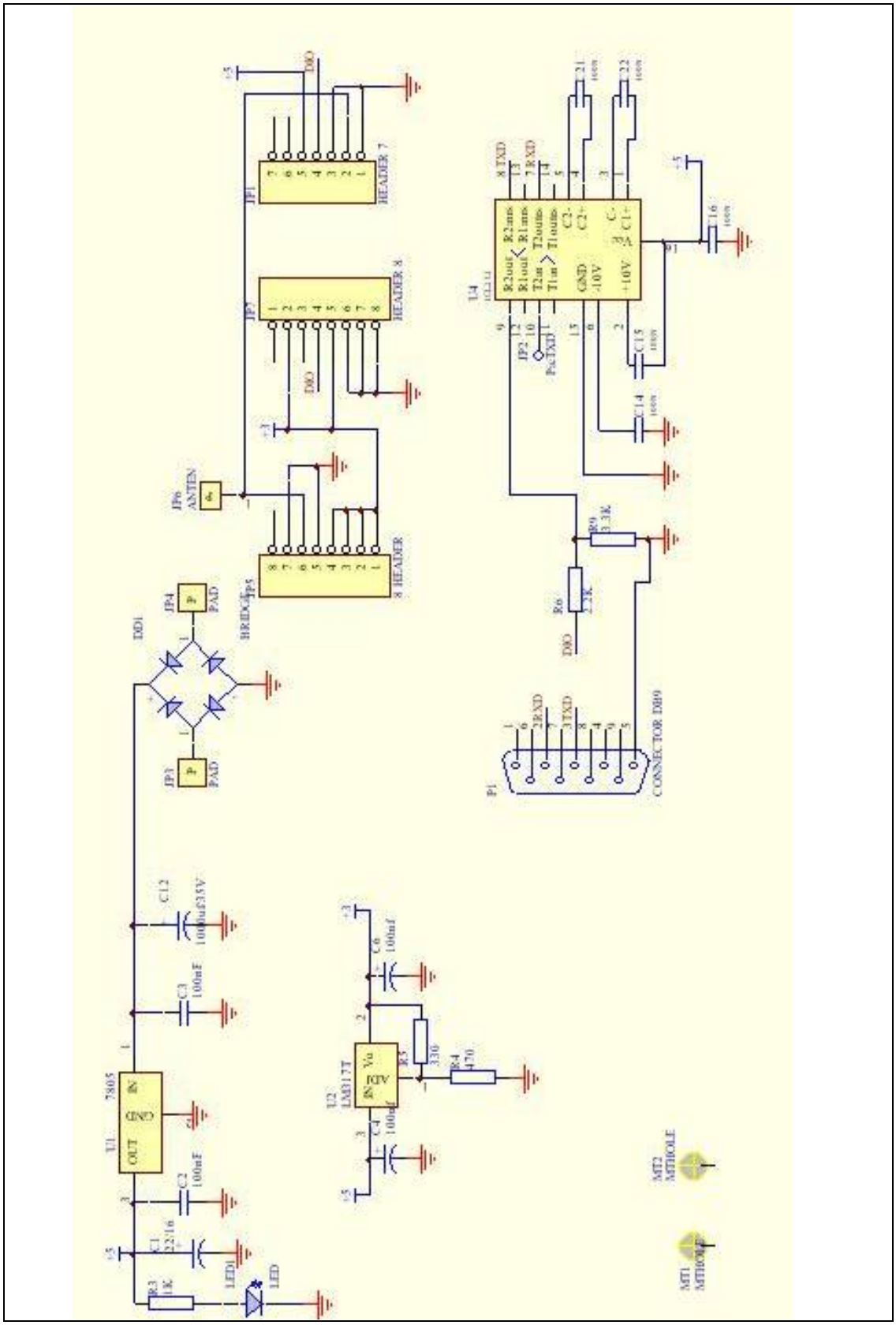

Figure 2.7 Computer Rf transmitter card scheme.

CHAPTER THREE

VISION SYSTEM

3. Vision System

The RoboCup Small Size League (SSL) is also a challenging environment for computer vision applications. Ball and Robots which move at speeds up to 2 m/s and the need of tracking them precisely requires the implementation of a high performance vision system. In vision system also Robot and Ball identification and orientation process is done, vision system is used for a feedback to strategy.

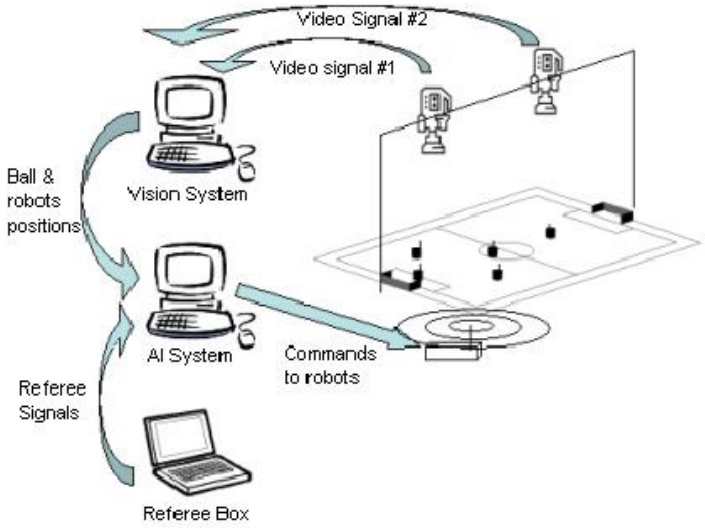

Figure 3.1 RoboCup Vision and Field Architecture

3.1 Introduction to RoboCup Small Size Vision

RoboCup (Luis A. Martinez-Gomez & Alfredo Weitzenfeld, 2004) is an international joint project to promote AI, robotics and related field. In the Small Size League, two teams of five robots up to 18 cm in diameter play soccer on a 4 by 5.4 m carpeted soccer field. The typical architecture of a team in the Small Size League (SSL) has four main components: the vision system, the AI system, five robots and the referee box. The vision system digitally process two video signals from the cameras mounted on top of the field. It computes the position of the ball and robots

on the field, including orientation of the robots in its own team transmitting the information back to the AI system.

The AI system receives the information and makes strategic decisions. The actions of the team are based in a set of roles (goalkeeper, defense, forward) that exhibit behaviors according to the current state of the game. To avoid collision with robots of the opposite team an obstacle avoidance module is used. The decisions are converted to commands that are sent back to the robots via a wireless link. The robots execute these commands and produce mechanical actions as ordered by the AI system. This cycle is repeated 60 times per second. Finally the referee can communicate additional decisions (infraction, goal scored, start of the game, etc.) sending a set of predefined commands to the AI system through a serial link.

The vision system (Luis A. Martinez-Gomez & Alfredo Weitzenfeld, 2004) is the only source of feedback in the whole architecture, if the data given by the vision system is wrong the overall performance of the team will be severely affected. That's why the vision system should be robust enough to compensate for any possible mistakes. The main object characteristics used by the vision system are the colors defined in the rules of the SSL. The ball is a standard orange golf ball. The robots of one team must have on top of them a 50 mm blue colored circle while the other team must have a yellow patch.

3.2 Color Identification and Vision system

In this project it is preferred to use two teams which's made of three robots up to 10 cm in diameter and a play soccer on a 1,4 by 1,8 m wood soccer field. The vision system digitally process one video signal from the camera mounted on top of the field with 25 fps. It computes the position of the ball and robots on the field, including orientation of the robots and transmitting the information (photo) to computer.

The main tasks of the vision system are, (Luis A. Martinez-Gomez & Alfredo Weitzenfeld, 2004):

1. Capture video from the cameras mounted on top of the field in real time.
2. Recognize the set of colors assigned in the rules to the objects of interest in the field (robots and ball).
3. Identify and compute the orientation and position of the robots in the team
4. Compute the position of the robots of the opposite team.
5. Transmit the information back to the AI system.
6. Adapt to different light conditions (color calibration procedure)(This can't be handled, I have to use a standard lightning).

Vision system architecture has several modules, each module it's a functional block with a specific task**.** Figure 3.2 shows the vision system architecture.

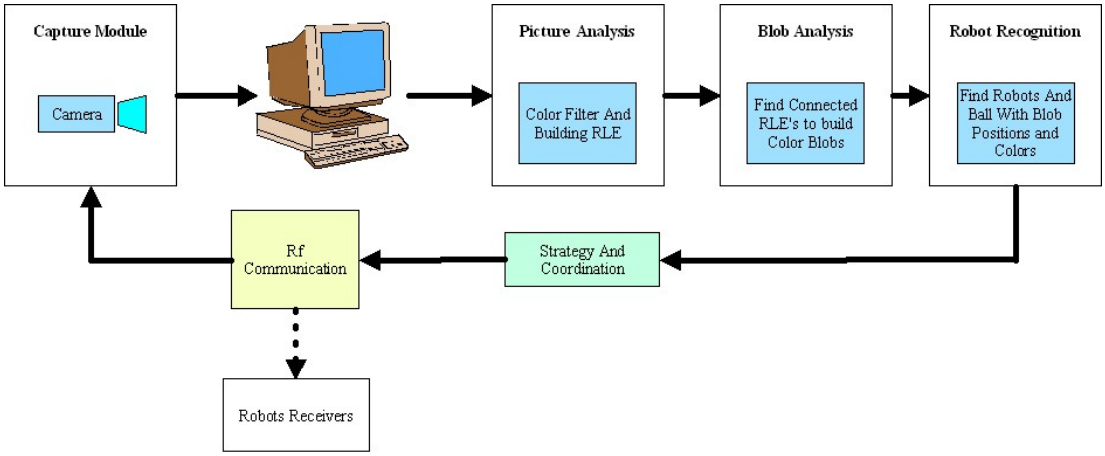

Figure 3.2 Vision System Architecture of the Project

The main blocks of the vision system are:

1. Capture Module; in which is assigns a physical capture device to the cameras, the type of connection (S Video, Composite, etc), the resolution of the image, and the frame rate.
2. Picture Analysis; in which separates each pixel of the images into object classes. Filter using the thresholds values assigned and Separate them into RLE (Run Length

Element) which is color lines. Here are colors shown in Table 3.1 which are used in RoboCup and in My Project.

3. Blob analysis part; in which program calculates and labels separated circular color papers which are mounted on top of the Robots.

4. Robot recognition part; in which robots are identified and numbered according to blobs configuration which will be examined in fallowing parts.

Table 3.1 Used Colors in Robot Soccer

Color	**Representation of the object In RoboCup**	**This Project**
Orange	Ball	Red Color For Ball
Yellow	One team	Opponents and mark for identification (My Team).
Blue	The other team	My Robot Team
Green	Playing Field	Not used (Hard to Capture)
White	Field markings including border, middle line, center circle, and defense zone lines	Not used (Reflections are dangerous)
Black	Physical Robot ‘s cover	Physical Robot ‘s cover
Magenta	Orientation Mark(Optional)	Orientation Mark for identification
Cyan	Orientation Mark(Optional)	Not Used

3.2.1 Color Filtering

In filtering process RGB space distance to desired color is used, color identifications and thresholds are given in Table3.2.Filtering works like a sphere around the desired threshold color this idea is shown in Figure 3.3. For example to filter the red colored pixel, program calculates the distance between the examined pixel (RGB values) and the full red color (255, 0, 0). Distance to desired color can be found by simple equation which finds the distance between two points.

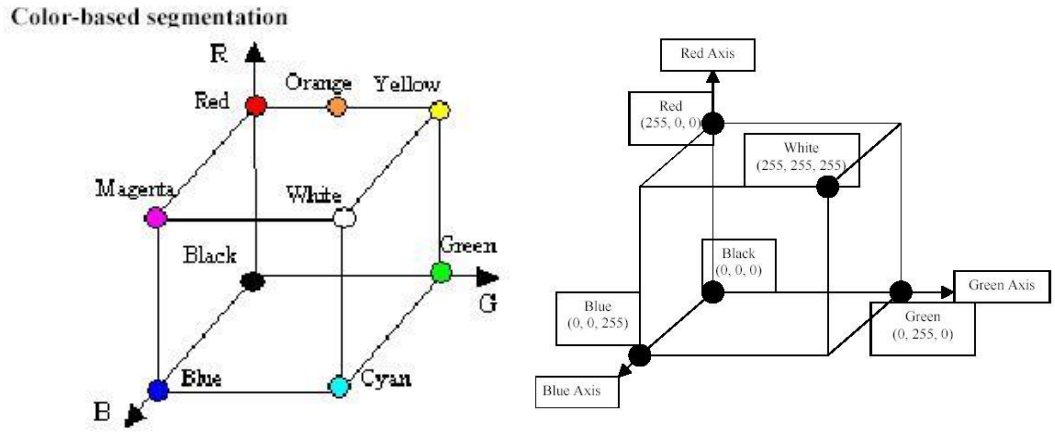

Figure 3.3 RGB Color space. (Chunmiao Wang and others, 2001).

Table 3.2 RGB codes and their own thresholds (R Radial).

Red	Green	Blue	Threshold (R)	Color
255	0	0	160	Red
0	0	255	150	Blue
255	255	0	170	Yellow
255	0	255	160	Magenta

Lets look at maximum Red (255, 0, 0) and a Point which has Red = 200 Green = 50 Blue = 45 when the distance between two points is calculated it has been found as 86 which is less then threshold R = 160 so that pixel is labeled as Red (Classification). This process continues until all pixels (240 x 320) finished. Next step is to group this dot pixels to lines (Run length Element) and then after to blobs.

The matching color partition for each color identifier is created based on the expected color of the object in the image, which is dependent on the actual color of the object and the lighting conditions. Given a fixed set of objects and controlled uniform lighting, a constant set of convex color partitions can be defined.

3.2.2 Run Length Encoding

The RLE algorithm (C. H. Messom and others, 2002) takes a line of pixels in the image and identifies the matching color identifier for each pixel. The color identifiers represent a set of RGB values that form a convex subspace of the 24-bit RGB color space. A partition of the RGB color space must be provided for each color identifier before this can be achieved as it is examined before.

Once each pixel has been classified as being part of a particular object's color partition, a line of pixels is run length encoded by identifying the start and end position of each object .The start and end position of the line of pixels that are from a single object and the color identifier of the object forms the components of the run length element given in Figure 3.4

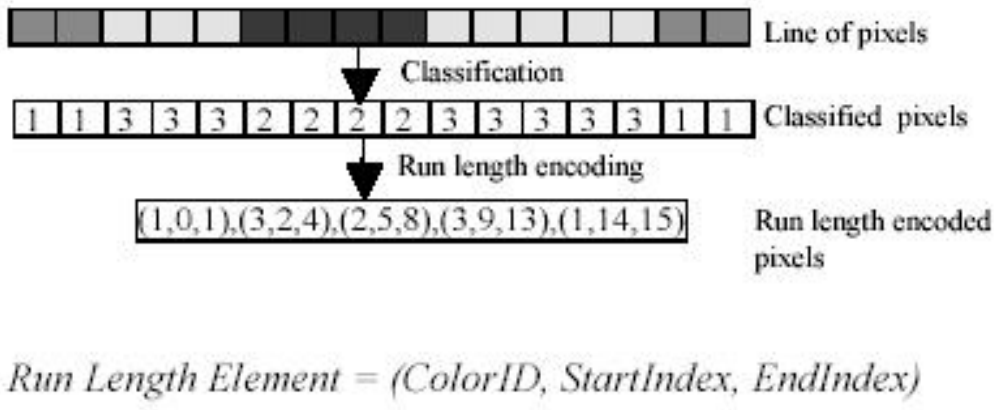

Figure 3.4 Run Length Encoding of a line of pixels (C. H. Messom and others, 2002)

3.2.3 Blob Analysis

In this part program connects the segmented pixels (RLE) into blobs. Before reaching this module the image is composed of separate pixels. The compressed image is then processed to identify the objects in the image, their sizes and their positions. The run length encoded lines are compared to identify the adjacent elements with identical color classifications. These run length elements are tagged with object identifiers shown in Figure 3.5 that are unique for each element.

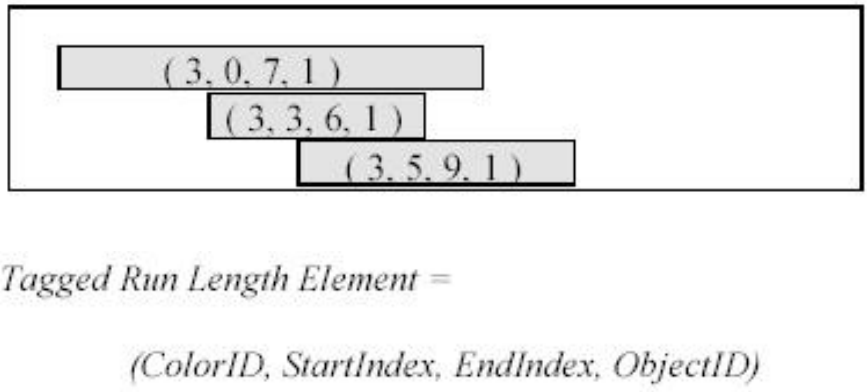

Figure 3.5 Tagged Run Length Encoding of a line of pixels (C. H. Messom and others, 2002)

The run length elements that have no neighbors of the same color are allocated a new object identifier (Object ID). If the run length element is adjacent to a run length element of the same color it is assigned the same object identifier as the neighbor. This approach guarantees that each object of a particular color is tagged with a unique object identifier.

Starting from the top of the image the object identifiers can be consistently assigned. However when a run length element has more than one adjacent element in the previous row as seen in Figure 3.6 two object identifiers can exist for a single object. This is resolved by re labeling the second neighboring element with the correct object identifier.

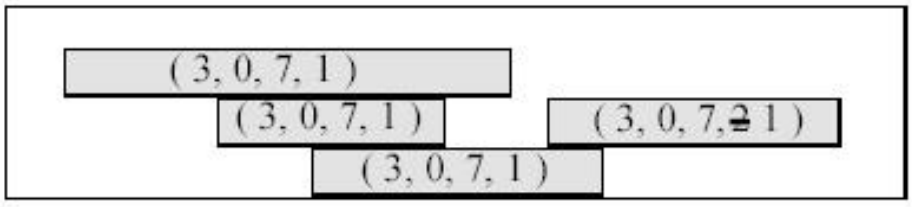

Figure 3.6 Reassigning object identifiers (C. H. Messom and others, 2002)

Once the elements have been correctly classified with unique object identifiers, they are grouped together by totaling up the run length elements that share the same object identifier. This provides the size of each object. The position of each object is

determined using a center of gravity calculation based on the size and position of the run length elements. The size and position of the run length elements is given by the start and end position of the run lengths and the line in the image in which they occur.

3.2.3.1 Blob Numbering Algorithm

The image analysis algorithm has been implemented using two data structures, 'RLE_element' and 'Neighbors'. (C. H. Messom and others, 2002) Each instance of the first structure, 'RLE_element', holds the information for a single sequence of pixels of the same color. If a run of pixels of a particular color occurs, an instance of the 'RLE_element' structure stores the location and length of the run of pixels, the object ID number, the color partition that object falls into, and whether it has any neighbors of the same classification Figure 3.7

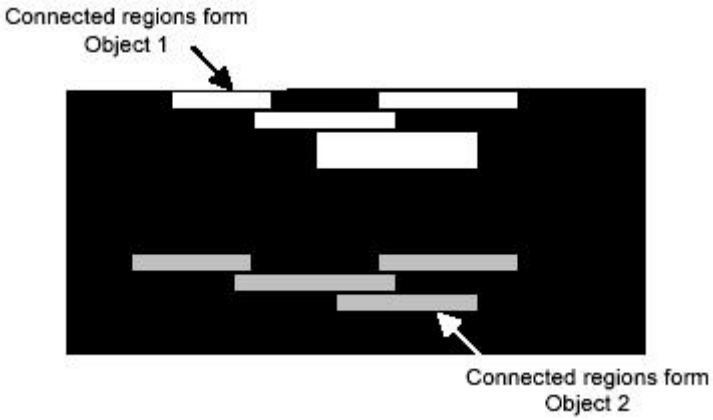

Figure 3.7 Neighboring RLE elements forming objects. (C. H. Messom and others, 2002)

An instance of the second structure, 'Neighbors', is used to store for each RLE_element, the locations of all neighboring elements falling into the same color partition. Thus, when a run of pixels is recorded in a RLE_element structure, the region around the run of pixels is also checked for other runs of pixels, with the same color categorization. Every run that falls into the same color category is stored in the current object's corresponding 'Neighbors' structure. The structure holds all the

neighboring objects' locations in an array Figure 3.8. These structures and arrays are shown in Figure 3.9.

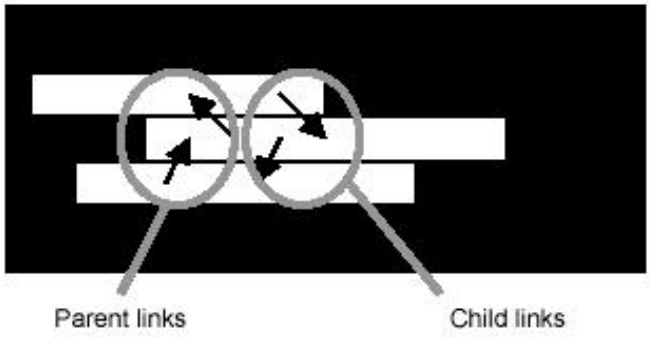

Figure 3.8 Links to neighboring RLE elements of identical color.

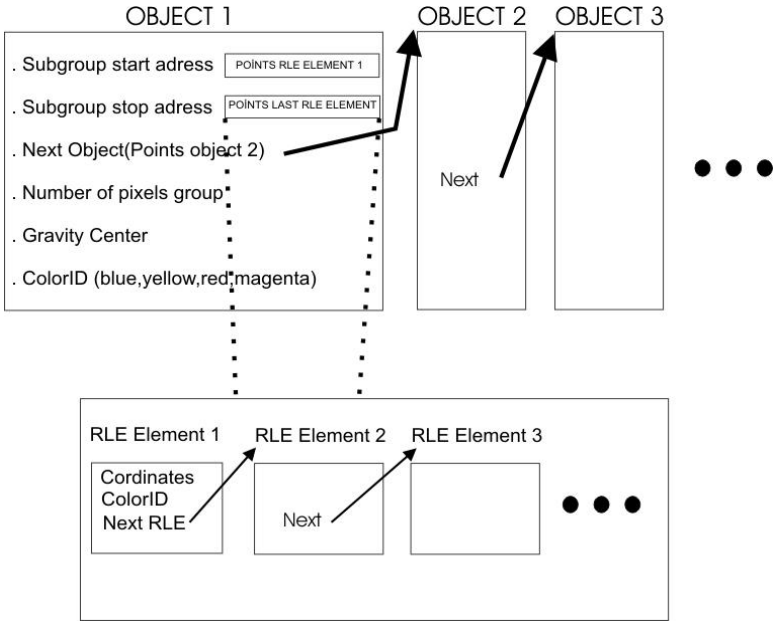

Figure 3.9 Using Pointers for Connecting Element together.

The whole program passes through the image twice: firstly compressing the image and secondly - processing the encoded image. The first pass analyses each row of pixels, until the end of the captured image is reached. In each row, the runs of pixels are classified into their respective color partitions. When an object other than the background is encountered, two processes are executed. Firstly, the object is assigned the next available object ID. Next, the surrounding region of the current

object is checked for other objects of the same color category. If any exist, they are considered connected, and belong to be the same object. The connected region is recorded as a neighbor of the current object, and vice versa. The background, or playing field, is not recorded with the 'neighbors' field.

The second pass through the image selects each object, and uses it as the root of a tree. From this root, a recursive depth first traversal is performed on all the connected regions that have been recorded as 'Neighbors' in the first pass. In this process, all connected regions will be assigned the same object ID, as the tree root. If the connected regions already have been assigned an object ID, the tree overwrites its object ID. The result of this second pass through the image is that all connected portions are identified with the same object. The flow chart in Figure 3.10 illustrates the processes in the algorithm.

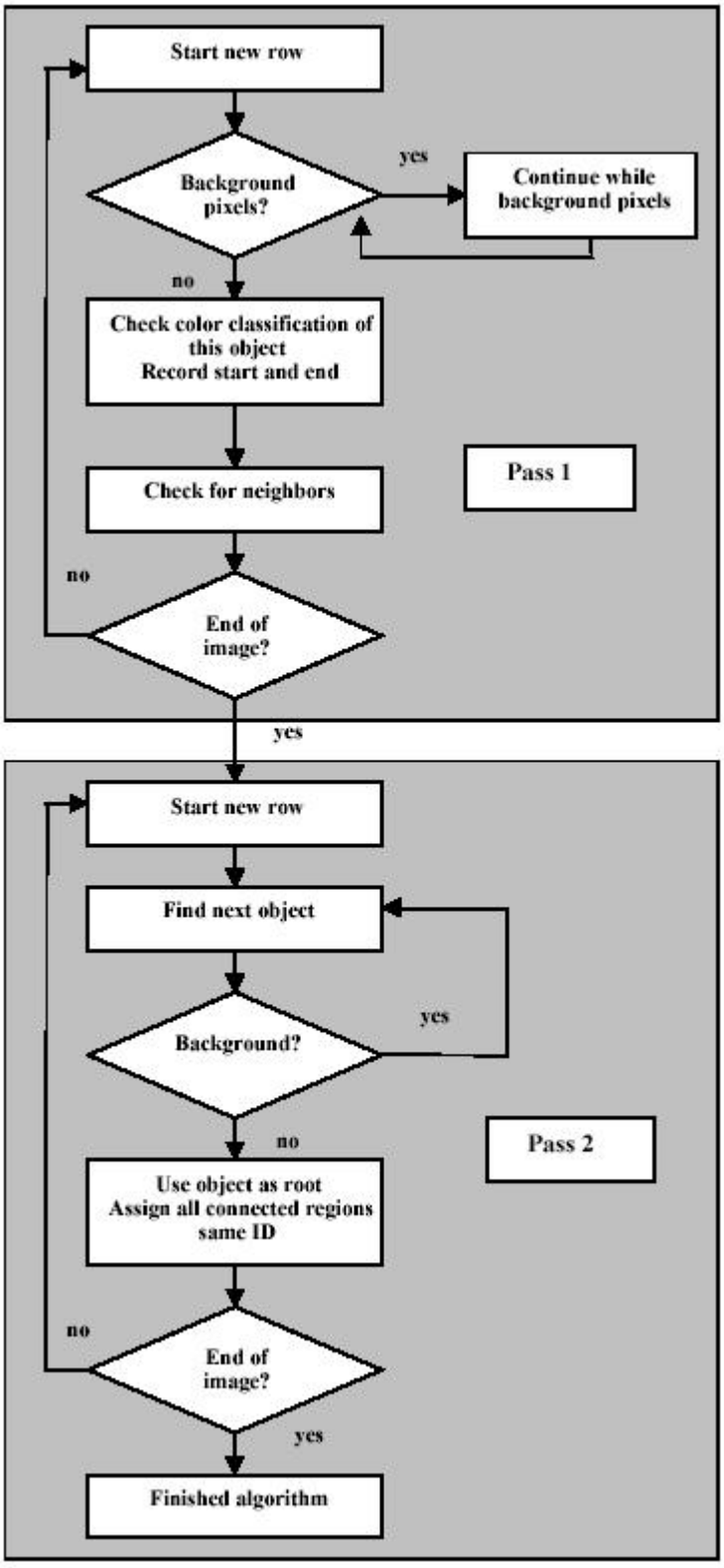

Figure 3.10 Blob Numbering Processing Flow Chart. (C. H. Messom and others, 2002)

When a blob is constructed useful information is computed such as the area, centroid, bounding box, etc. selects the regions that adjust better to the objects searched. It has a selection criteria for every kind of object. For the ball red blob is selected which is nearest to an area of 300 pixels (with an image resolution of 240x320). For the robots of the opposite team the selection criteria consists in selecting the blobs of the corresponding color of the central patch with an area nearest to150 pixels (the area of the patch is smaller than the ball). The number of selected blobs are determined in the Robot Recognition. For home team the procedure is similar to the one used for the robots of the opponent team, but additional to the central patch, a search for extra patches is necessary. The extra patches are employed for Robot Recognition they computes the position of all objects in the field.

3.3 Robot Recognition

When all colors are filtered and labeled now vision program has lots of information about the match field, this information is valuable for strategy and robot identification. The combination of these color blobs gives an information but first of all this code has to be solved, this chapter examines this blob order decoding.

3.3.1 Ideas Used in Robocup for Robot identification

Now that the basis for choosing patches has been established, this knowledge can be used to evaluate tracking patterns. One source for many different ideas are the various patterns used by the over 20 teams in the RoboCup F180 League. The rules for patterns which are on the top of the robots in that league have naturally led to many tracking and identification patterns being tried. Examples of some of the more popular designs can he seen in Figure 3.11. The approaches taken thus far generally fall into one of three broad categories. By far the most common type is like that shown above, which is called patch based, where in addition to the team marker patch in the center, one or more additional circular or rectangular patches are used to encode position and orientation.

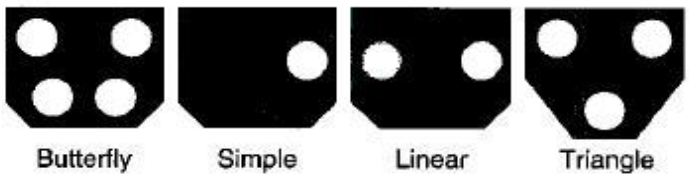

Figure 3.11 Examples of common tracking patterns from the RoboCup180.

Patch based systems. (James Bruce & Manuela Veloso, 2003) have the advantage that position and orientation detection can combine several features (patches in this case) with sub-pixel accuracy. One alternative to this is to have a key patch marking the center of the pattern, surrounded by radial "pie slices" of two or more colors. Orientation and identification can be performed by scanning at some constant radius from the central patch. Unfortunately, by depending on features (color edges in this case) that are difficult to quickly detect with sub-pixel accuracy, it is difficult to get very accurate orientation using this method. Another type of tracking pattern that has been tried is to use a central patch with a nearby line feature. By finding the edge points of that feature, a least squares fit can be made to get an accurate orientation measurement.

Unfortunately, both these classes of patterns do not offer a straightforward way to improve the position estimate. (James Bruce & Manuela Veloso, 2003) For a given pattern size, it would be like to make the most of the space. Ideally, it would be liked that all of the patches to contribute to position, orientation, and identification detection. In this regard, patch based systems tend to do quite well, which has led to them becoming the most common class of pattern used for tracking.

If Figure 3.11 is cosidered, it can be noticed that similarities can aid in creating a generic detection algorithm. All are keyed by a colored patch (in the center of the pattern) indicating the presence of a pattern. Each patch occurs at some unambiguous angle radially from the key patch.

Thus there is a distinct circular ordering of the non-key patches that can be calculated even in the presence of moderate noise. This means a generic detection

algorithm can start by searching for the key patch, and then detecting and sorting the additional patches radially. All that needs to be done after that is to find the rotational correspondence of the additional patches with the geometric model of the pattern. In the case of the Simple pattern, the correspondence is trivial. For the Linear and Triangle patterns, the colors of patches can he used to disambiguate geometrically similar correspondences. Finally, in the case of the Butterfly pattern, geometric asymmetry can be used to find the rotational correspondence, assuming the distances between patches can be measured accurately enough. Table 3.3 Shows the number of robots which can be identified with its specific pattern and color number.

Table 3.3 The Number of Uniquely Identifiable Patterns That Can Be Detected Using a Certain Number of Colors. (Excluding The Key Patch And Key Patch Color).(James Bruce & Manuela Veloso, 2003)

Pattern	2 Colors	3 Colors	4 Colors	N colors
Butterfly	16	64	256	4^n
Simple	2	3	4	n
Linear	1	3	6	n * (n-1)/2
Triangle	2	6	12	n * (n-1)

3.3.2 Robot Identification Using Triangle Pattern

Once the lists of regions are generated the recognition process begins. As mentioned before, to recognize the ball program simply takes the region of red color whose area is nearest to 300 pixels (ball area with a 240x320 resolution). Every team decides which pattern of patches to use; in this project it is decided to use a Triangle pattern. Program searches the additional patches in a window around the central patch.

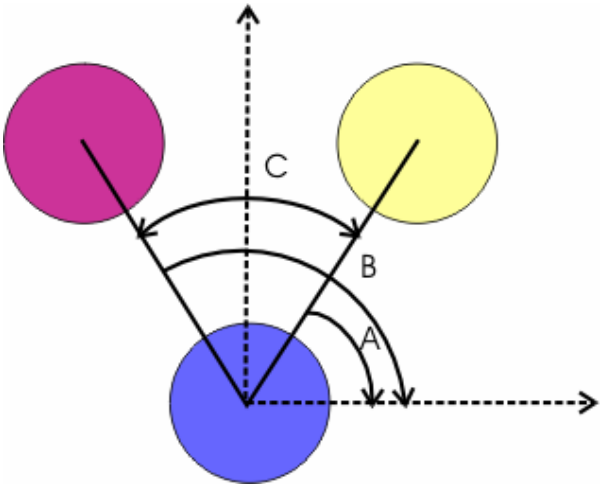

Figure 3.12 Color Patches and Using Angles for Identification.

Once two patches are found in the window area around the central patch **Blue** (Team color) then next step is to order them in a clockwise direction to obtain the color encoded ID of the robot. Every robot has a unique ID encoded in the additional patches, one color **(Magenta)** is interpreted as a **1** and the other color **(Yellow)** as a **0**. So if it has been found a patch pattern of Magenta and Yellow as in Figure 3.12 it has been interpret as 010 or a decimal 2 (which is Robot Number 1). The gravity center of 3 colors is Center of Robot and Program calculates the direction of the Robot by using two front patches (Magenta and Yellow) gravity centers.

Table 3.4 Color Orders and Robot Numbering

Robot Number	**Left Front**	**Right Front**	**Center, Team Color**
Robot 0 = 00	Yellow = 0	Yellow = 0	Blue
Robot 1 = 01	Yellow = 0	Magenta = 1	Blue
Robot 2 = 10	Magenta = 1	Yellow = 0	Blue

Blue is the team color and after that program searches for other color (Yellow or Magenta).First Program take Blue(1) in consideration and search other color blobs which are in h=20 pixel far from it. If program finds two color blobs and these are

labeled as Yellow that robot can be assigned as Robot0. In assigning process Robot's center and Front parts are calculated and written to that Robots program structure.

Program will continue with Blue(2) and search for blobs when it finds one magenta and one yellow it needs to look at their orders which will give the robot number. As shown in Figure 3.12 it can be seen that yellow is in right and magenta is in left side of the central patch. To decode the order of color blobs, angle difference is considered. As it seen in Figure 3.12 Yellow blob's angle (A) is smaller then Magenta blob's angle (B) by this information it can be understood that Yellow is in left side. But in this case there will be some problems as it is shown in Figure 3.13.

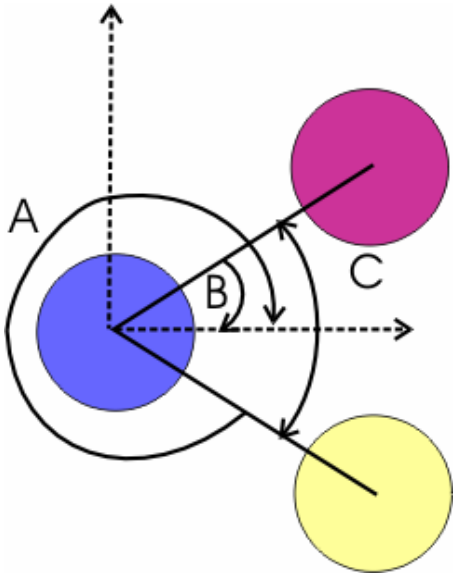

Figure 3.13 Some Problems Using Angles for Identification.

In some conditions its is not so easy to consider angles, if Figure 3.13 is considered program will calculate orders wrongly, to handle this problem program needs to look at difference between A and B which is C = A-B. Absolute value of C must be smaller then 90 degrees if its not program fixes it with in the idea in Table 3.5 and fixes the order combination of colors.

Table 3.5 Searching clockwise direction by using angles.

Angle C	**Robot 1**	**Robot 2**
<= 90 (**Figure 1)**	B > A	A > B
> 90 **(Figure2)**	A > B	B > A

Figure 3.14 shows this robot color orders. As it seen opponent team members are using single yellow colors program will find their places easily. Lonely Yellow blobs are labeled as opponent team member. Red blob is directly labeled with Ball.

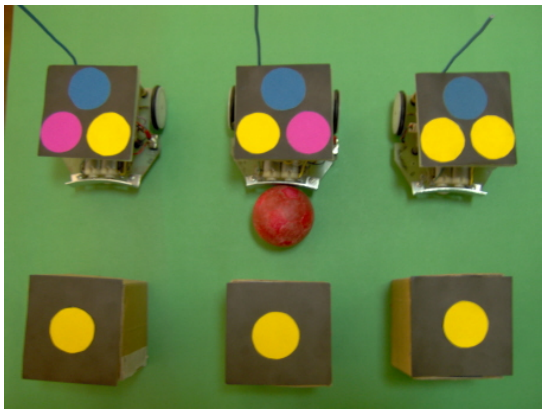

Figure 3.14 Robot and Ball Color In Project.

3.3.3 Result of Robot Identification

Identification process is so critical which depends on colors and lightning, in some conditions to minimize these faults it is preferred to use a constant lightning system. This system consist of four lamps which place top of the filed. Figure 3.15 shows all these color blobs and their orders, because of the lightning conditions there will be same faults. If one of these yellow blobs is miss identified the owner Robot of that color will be lost and similarly if the vision system lost the red blob all strategy will stop until the ball is reappeared. Lonely Yellow blobs are identified as Opponent team, more then three Opponent team members are reported as error, this will not fail the strategy but one of the yellow blob will be miss matched. Opponent robot which is near to enemy goal is identified as enemy goal keeper this information is used for Forward role and for best shooting program will calculate empty parts of the enemy goal. Red color is used for ball identification. It is the only red blob in field if vision finds more then one blob this is reported as error and strategy calculation will stop

(Robots will stop until vision find the ball again). In the beginning of the project instead of Magenta a Green Blob is used but because of some identification problems (lightning) Green color is changed with magenta. Color classification and robot identification handle efficiently but in real life Vision program has to handle some other problems, lightning is one of the biggest problem in vision system. Light conditions can change in all day (morning, evening and night). All these changes need some calibrations and because of these changes sometimes identification process can't work well. In Figure 3.15 camera view of the field can be seen, vision system needs to handle some lens distortions these nonlinearities are handle by some program calculations.

Vision program of this project is seen in Figure 3.15. Team member are identified and their names are labeled near to their image, opponent team member are showed as yellow circle and finally it can be seen that the ball with red circle in the middle of the field. Goal centers and field size is know by computer which's values are constants and can't be changed. Camera is 1.5m high from the ground and it is placed for seeing the whole filed.

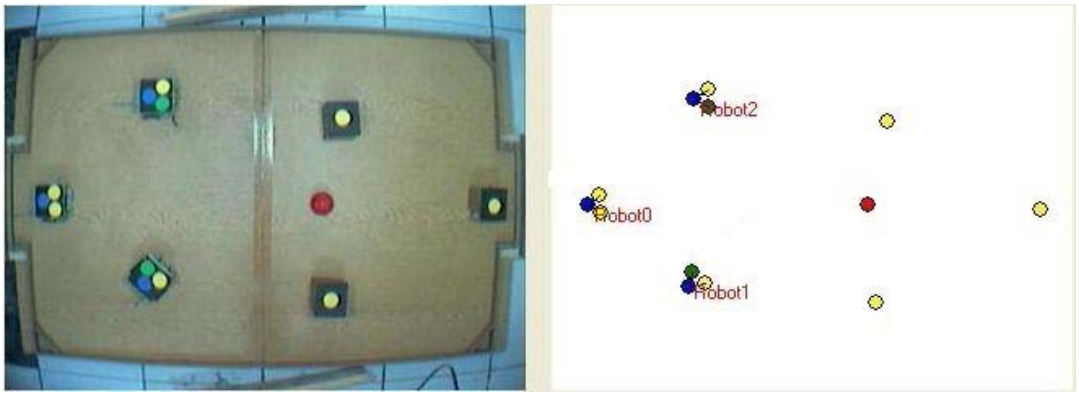

Figure 3.15 Blue Team, Opp. Team and Ball Colors in Project Vision System.

CHAPTER FOUR

PATH PLANNING

4. Path Planning

In this chapter; The Real time planning of a free-collision route for the robot to move from an initial to a goal position is considered. This problem is well known in robotics and is called the basic motion planning problem. Such problems involve searching the system configuration space for a collision-free path that connects a given start and goal configurations, while satisfying constraints imposed by a complicated obstacle distribution. This definition of the problem simplifies some of the aspects of robot motion planning. The dynamic properties of the robot are ignored and the problem is transformed to a purely geometrical path planning problem.

4.1 Ideas Used In Robot Soccer

In (T. H. Lee and others, 2001) different layers are used and then these layers are combined to find the path. Every object in the stadium is given a working layer which is decomposed into a fixed number of small areas. They propose to assign values to different small areas of the layer according to the object's movement. The final path is obtained by combining the small areas of the layers and getting the path with the smallest values. Since the host computer uses only a simple equation to calculate the value of each small area, this algorithm requires only a low computing power, and the speed of path generation is very fast.

Another idea which is used in (John Anderson &Jacky Baltes, 2004) that is the Quad tree data structure to model the environment and Flexible Binary Space Partitioning. Instead of consistently partitioning the space into components of precise sizes (i.e. dividing in half for binary space partitioning), this approach attempts to make a logical choice for a partitioning point in any partitioning decision. An entropy-based measurement is employed to examine the potential information gain

that would be made by choosing a particular point to further divide the space, and the point with the largest information gain is chosen. Once the space is partitioned, path planning ensues based on this decomposition. That idea uses “learning” mode of environment mapping. It uses the distance transform for path planning. Field Partitions are not constant they change time to information valuable partitions are divided more and updated. I didn’t choose this method because its hard to program and it takes long computing time.

In (Ching-Chang Wong and others, 2001) a Fuzzy-Obstacle-Avoidance-Path (FOAP) algorithm is used .Which is for the obstacle avoidance and a procedure for the shooting action of the robot soccer. This algorithm contains a fuzzy system that is used to estimate the rotational velocity of the robot soccer.

In (T. H. Lee and others, 2001) Another idea presented for a fast method to plan the path, and control the position of wheeled mobile robots (WMR) in robot soccer game. In a WMR system with position control, when the target position is given, the controller will generate the reference path and then the WMR will move to the target position following the reference path. To tackle the difficulties of the nonlinear control problem, the motion of the WMR is realized via changing the linear displacement and angular displacement in a separate manner such that the WMR is either rotating or moving in a straight line at any one time. This method takes advantage of a proposed fast path-planning algorithm**.** The complexity of design is thus significantly reduced.

In the literature diverse algorithms have been proposed to tackle this problem: Some of them, such as the randomized potential field methods (Thilo Weigel and others, 2001). represent the robot as a particle moving under the influence of an artificial potential field produced by the sum of a repulsive potential, generated by the obstacles, and an attractive potential, generated by the goal configuration. The path is obtained by a descent along the negative gradient of the total potential. Others are based on roadmaps (T. H. Lee and others, 2001) in which a network of one-dimensional curves, called the roadmap, lying in the free space of the workspace is

constructed. The final path results from the concatenation of three sub paths, one connecting the initial configuration to the roadmap, another belonging to the roadmap and a final one from the roadmap to the goal configuration.

Another general approach is based on a division of the free space into a set of exact or approximate cells Cellular Automata (M. Bracho and others 2001), (C. Behring, M. Bracho, M. Castro & J. A. Moreno, 2000). The path is a sequence of cells with the following properties: (1) The first cells contains the initial configuration. (2) The final cell contains the goal positions. (3) Neighboring cells in the sequence are cells in free space with a common boundary. In spite of the diversity of available path planning methods their computational complexity, is an important limiting factor in their practical applicability. In contrast the path planning algorithm proposed in this work is Cellular Automata.

4.2 Cellular Automata

After recognition part is complete it is time to think about the movements and basic behaviors of our team Robots. First it has been tried to build the Robot movement ability which will help for commanding Robots one destination to target. Path Planning is used so often in Robot strategy, for this purpose (C. Behring, M. Bracho, M. Castro & J. A. Moreno, 2000) Cellular Automata and Potential Field method combination is used for path planning algorithm.

For initial condition of Cellular Automata the workspace is decomposed in a finite collection of non intercepting square cells. Assuming a square of 8 x 8 pixel size, in the present case a 240x320 grid where each cell corresponds to approximately 4 cm2 of real space, the discrete configuration space can be defined by the coordinates of each cell (30x40). Before starting to Cellular Automata it will be useful to look at some concepts that are used in CA, these are Moore Neighborhood and Manhattan Distance.

4.2.1 Moore Neighborhood

The Moore neighborhood (also known as the 8-neighbors or indirect neighbors) of a pixel, P, is the set of 8 pixels which share a vertex or edge with that pixel. These pixels are namely pixels P1, P2, P3, P4, P5, P6, P7 and P8 shown in Figure 4.1.

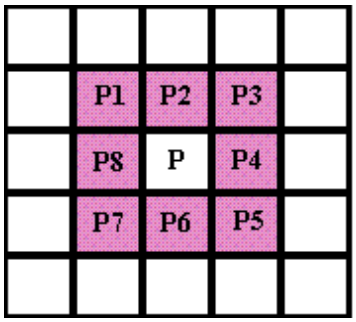

Figure 4.1 Moore Neighborhood

4.2.2 Manhattan Distance

The distance between two points measured along axes at right angles. In a plane with p_1 at (x_1, y_1) and p_2 at (x_2, y_2), it is $|x_1 - x_2| + |y_1 - y_2|$.

4.2.3 Phases of Cellular Automata

Cellular Automata has 3 phases;

In phase one; all field is divided into 8x8 pixel boxes and they placed in a structure with their 0 potentials and coordinates. After this separation program has a new field which has 30 boxes in Y direction and 40 boxes in X direction in a 240x320 pixel field. In this part, program also updates corners and out side of the field with potential 100 (which is maximum). Robots will not choose those places because they always choose to travel to low potentials. These are the initial conditions for the potential field, after three phases are finished this initial potentials will be reloaded for next application. This process is shown in Figure 4.2.

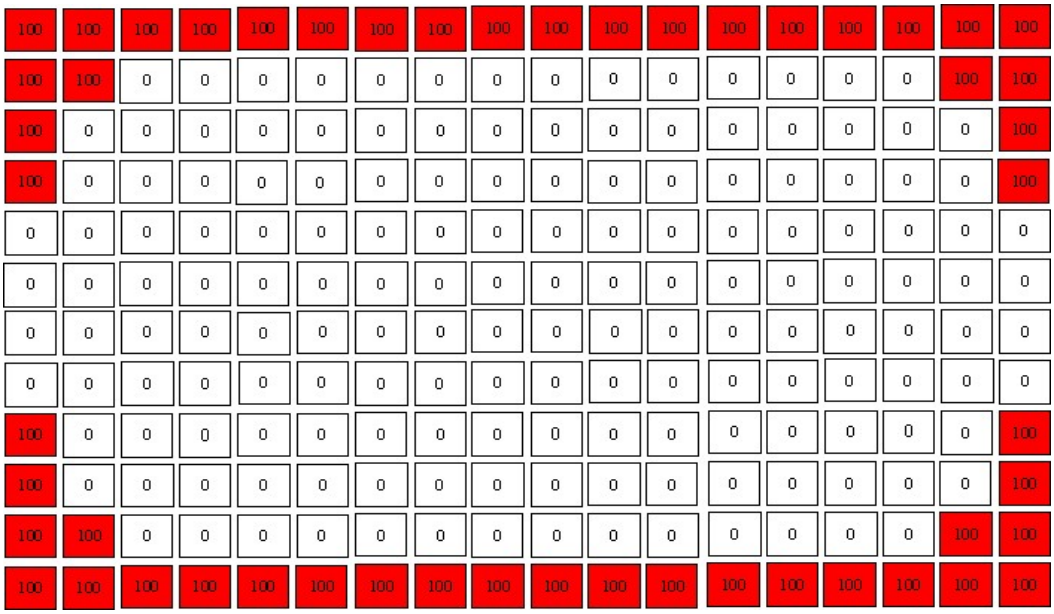

Figure 4.2 Initial Conditions of Cell Potentials.

In phase two; obstacles are placed and their potentials are updated. In Figure 4.3 it can be seen that opponent team members, home (Blue) team members and ball center potentials are updated with 40 potential and also their Moore neighborhoods are updated with 30 potential. The ball and all robots are updated with this method except the robot which's path is going to be planned (Robot2 in Figure 4.3). Program also updates home team robots because they are also obstacles and they can collide with each other. Potential fields are also updated for the ball because it is also an obstacle, otherwise robots won't be able to drag the ball and may push or shoot wrongly. It is preferred to use 40 potential for center and 30 potential for protection band, these values are given just to show the process much simply in Figure 4.3. These potentials and number of protection bands can be changed, in this project three protection bands are used and their potentials are updated as follows: center with 60, first band with 55, second band with 50, third band with 45. These potentials are used when any wrong interceptions are occurred. Robots will choose the low potential to get away from the obstacle center. A predefined template (protection band) of cells in state is placed over the position of the obstacle objects in order to account for the physical size of the robot. The size of the template depends on the

size of the robot and of the cell, for the typical resolutions used the template is about 7x7cells. Figure 4.5 shows these processes in computer vision program.

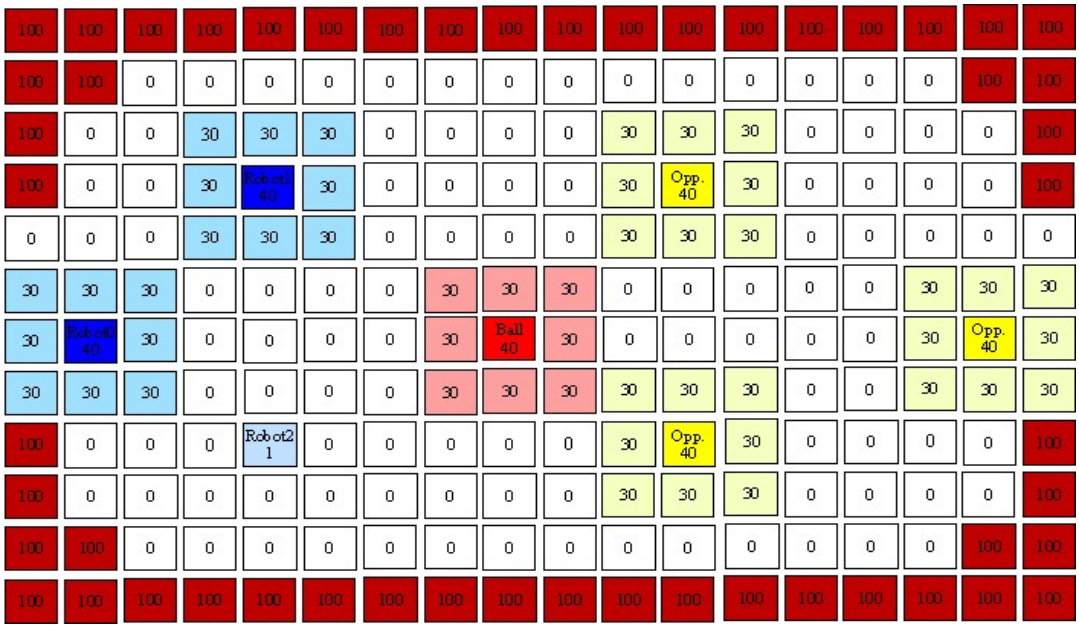

Figure 4.3 Obstacle Potentials and Protection Bands

In phase three; all free space is updated with its Manhattan distance to goal. As it seen in Figure 4.4 potential of destination box is loaded with 1 and its Moore neighbors (which's have only 0 potential) are updated with the destination potential plus one (Center +1) which is 2. After this process program will search for boxes (cells) which have 2 potential and when it finds 2 potential boxes it applies the same method to update its neighbors with 3 potential (2 + 1 = 3). Next step is 3 potential boxes and this loop will continue until no more cells (boxes) left with 0 potential. Potential Cell update algorithm is showed in Figure 4.6.

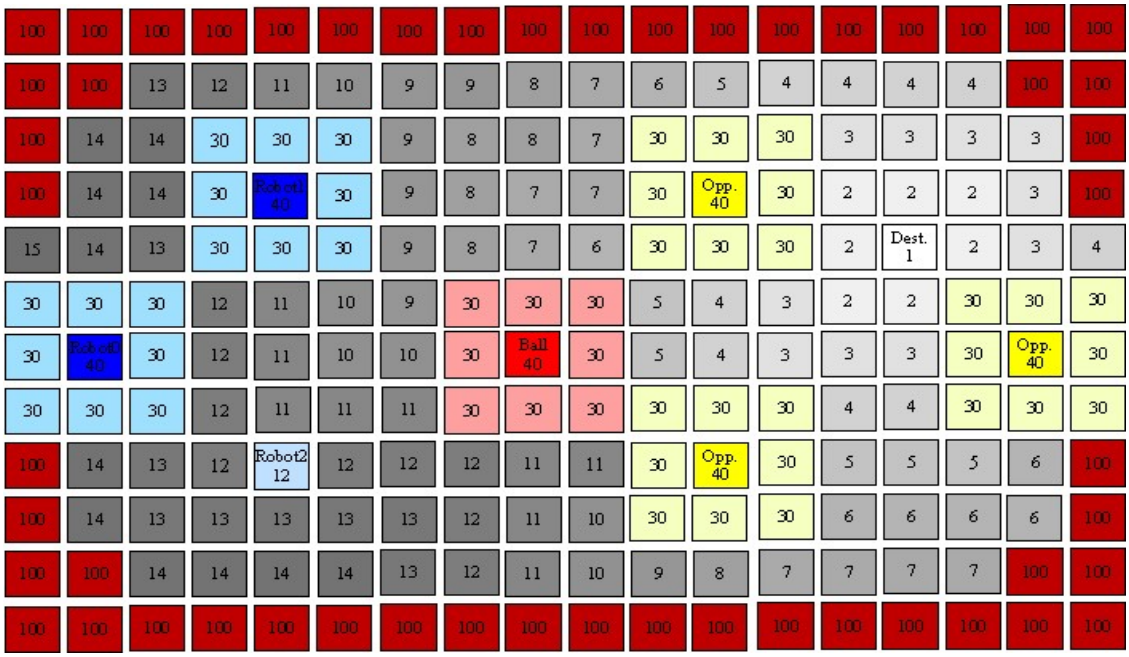

Figure 4.4: Each cell is updated with its Manhattan distance to goal

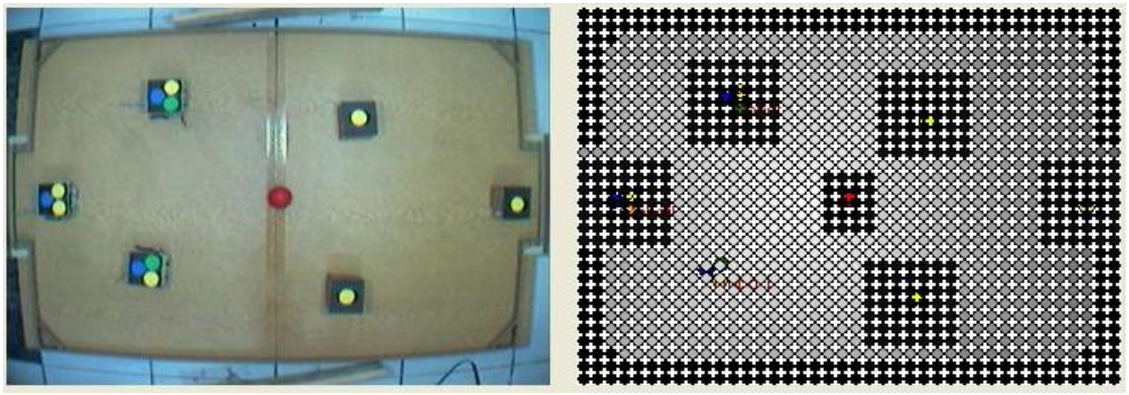

Figure 4.5: Potential update in computer program**.**

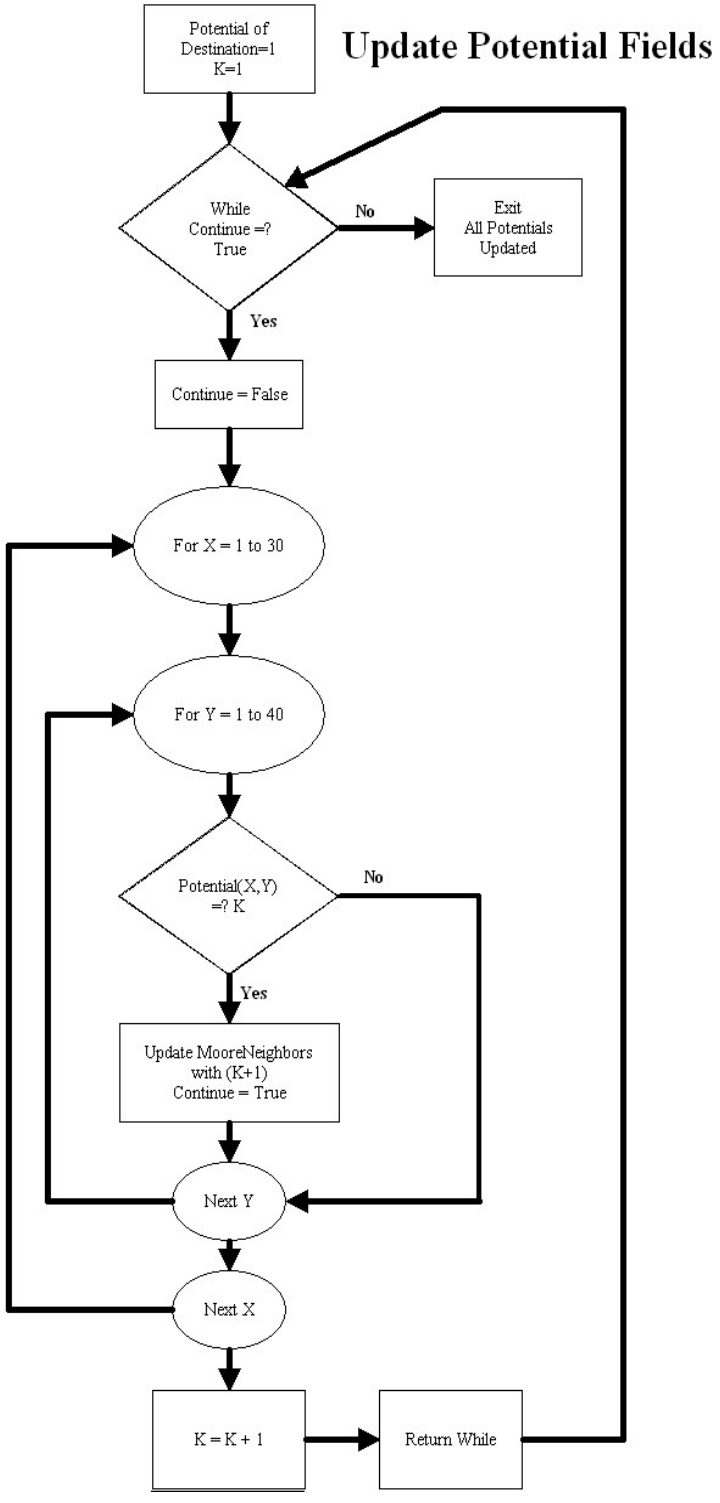

Figure 4.6 Algorithm of Updating Potentials.

4.3 Path Drawing and Robot Movements

After potential update part is finished, it will be easy to select the correct path. Robot will start searching its path from the cell which it occupies and select the smallest potential from its neighbor cells. Again moore neighborhood method is used for scanning connected cells. In Figure 4.7 this process is shown. Robot (Blue Cell) starts to search its neighbor cells which's potentials are 7, 50 and 5, it will choose the lowest potential (which is 5) and will travel to that direction in a method which will be explained in following chapters. After the robot's movement, its center cell will be changed and for next step again it will search its new neighbor cells. When Robot moves to 5 potential it will have new neighbor cells which have been loaded with 6, 7, 50, 3, 4 potentials and considering the rule, it will choose 3 potential and will move to that cell. This process continues until it reaches to 1 potential target (destination).

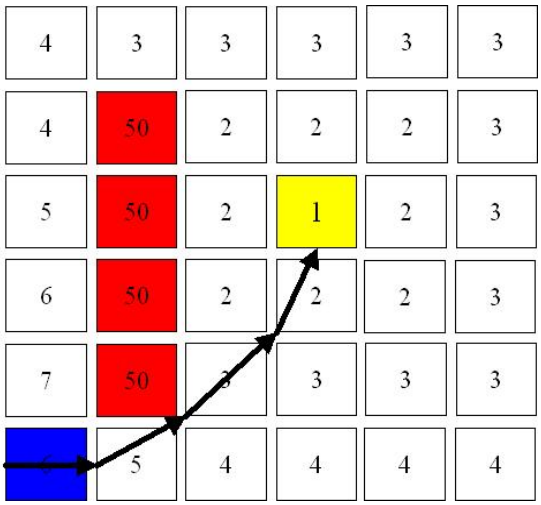

Figure 4.7 Path Planning

In some conditions it's not so simple to choose the lowest potential cell because there will be more cells which have been loaded with same potentials. As the Robot advances closer to target cell (potential 1) it will occupy the 3 potential cell (Figure 4.7 Third Arrow in Route) and under this condition it will have neighbor potentials like 5, 50, 50, 2, 2, 3, 4, 4. Considering the rule the robot will try to choose the lowest potential, however there are two same low potentials. In this case program will consider the distance angles to find the closest path to target. In Figure 4.8 this

problem can be seen with in Box A, Box B, Box C (Cells) which have 8 potentials. Box A has an angle of 90 degree, Box B has an angle of 45 degree, Box C has an angle of 0 degree and target Cell has an angle of "t" according to Robot's center cell. Robot will choose the box which has the closest angle to "t". If it is considered that "t" = 20 degree, then Box A would be | 90 – 20 | = 70 degree far from target, Box B would be | 45 – 20 | = 25 degree far from target, Box C would be | 0 – 20 | = 20 degree far from target. As it is seen Box C has the closest angle to target, it will be selected for the path (Yellow cell in Figure 4.8).

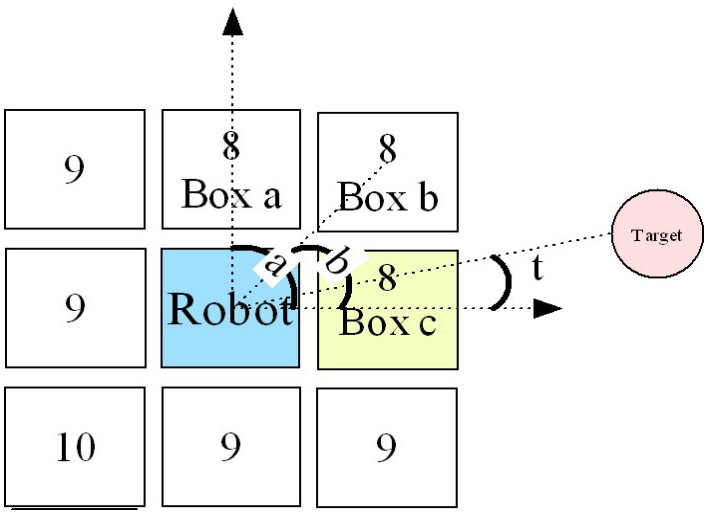

Figure 4.8 Problem with same potentials.

4.3.1 Basic Robot movements

When path cell is selected Robot will travel to that direction efficiently with in the idea (Jacky Baltes and Robin Otte, 1999) before starting, it is better to look at Robot movements. Robot's electronic card has Pic16f877 microcontroller and it is programmed to communicate with 3 bytes control command. This command block includes the Robot number and it continues with Robot way and Motor power. Command block is shown in Figure 4.9.

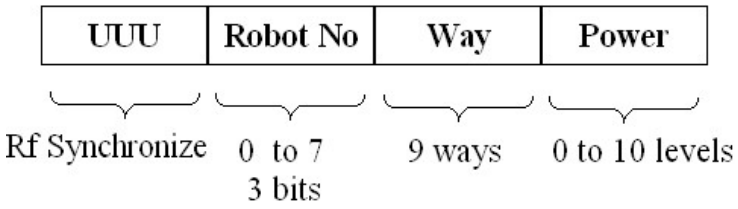

Figure 4.9 Robot Commands.

Robots have 9 basic movements and these are the simple abilities which are loaded to Robot microcontroller program. All other movements, behaviors and strategy build with combinations of these simple movements which are shown in Figure 4.10. For an example if Robot 0 wanted to be turned left with a specific power, its communication word is loaded with UUU [40] [4] [10] which means Robot 0 (Address of 40) turn left [4] with power 10 [10] (Full power).After command received Robot continues doing its job for a time period of 100ms and after this period it stops, if a new command has received in 100 ms it changes its job according to that command. In Figure 4.10 nine Robot movements are shown. It is important to focus on Left Forward and Right Forward movements which are different from the Left and Right movements. In Left and Right movement robot doesn't change its center it just turns on a constant point, but in Left Forward and Right Forward Robot turns while moving to forward. These movements (7 and 9) give a smooth moving angle to Robot in path movements however it also increases the risk of crashes. To escape from this collusions program uses an extra speed control algorithm which will be examined in previous sections. Left backward and right backward movements are not used in computer program because in real life applications it so hard to control and also not so efficient in path planning.

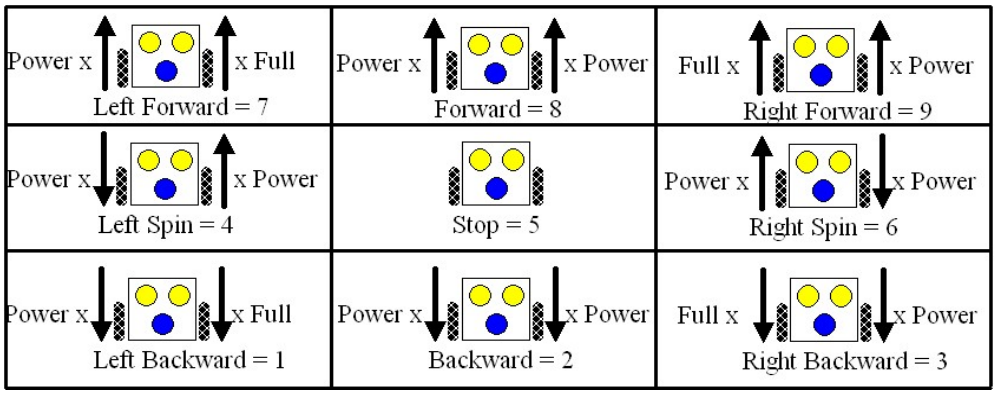

Figure 4.10 Robot simple movements programmed in Pic16f877

4.3.2 Moving Robot on a field

Moving robot on a path has three parts:

In part one program calculates the risk of movement speed of the Robot which will increase the smoothness and speed of movement and also increase the risk of crashing. This control is different from Robot power command because it controls the area near the Robot which may include other physical objects (Robots, Ball and Field Corners). According to that knowledge it increases the speed of movement. If there are no obstacles around, robot will move faster and also will use left forward and right forward (7 and 9) commands instead of left and Right (4 and 6). This application is so dangerous in experimental results, if high speed movement is selected without any control robot would be stuck or crush to other robots. Speed control (Run Faster) works like a switch, if there are any obstacles around then it will be off (Robot will move slow and use left and Right) if there are no obstacles then it will be on. For a better ball controlling Robot will move slower when it reaches near to it. Robot also will move slower and turn sharper when it comes near to field corners and opponent team members. This ability will decrease the risk of collusions. This process is shown in Figure 4.11 as Part one of path planning.

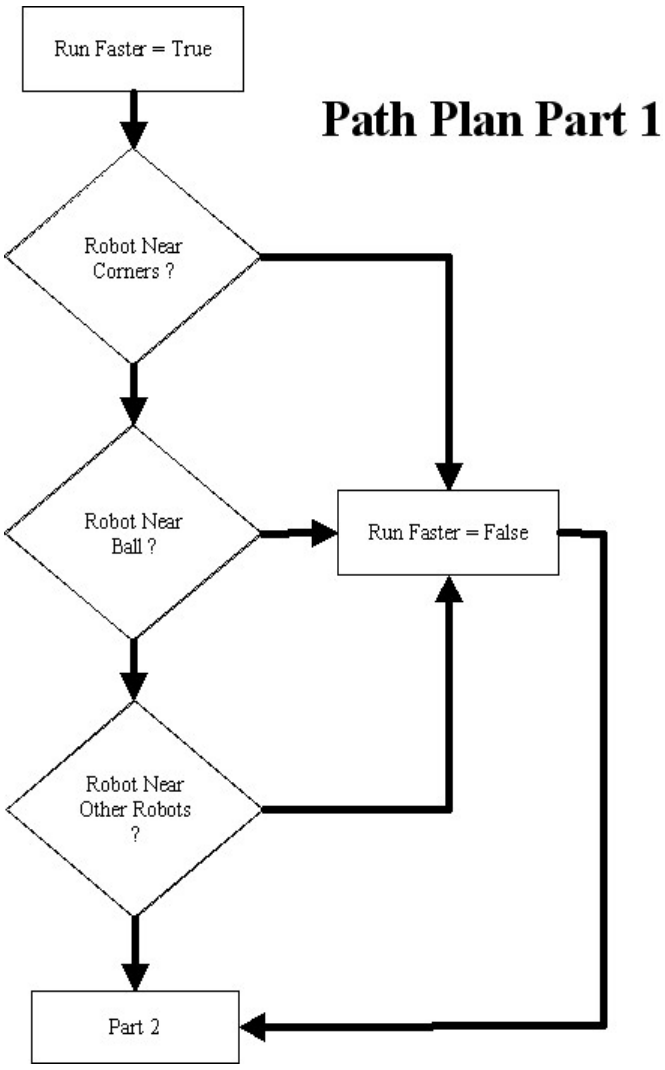

Figure 4.11 Path Planning Part One

In Part two; program calculates the direction of the way (left, right, forward, backward) with considering the path cell and the Robot center. Program calculates the angle “c” in Figure 4.12 which is absolute difference between Robot forward angle “a” and Path cell angle “b”.

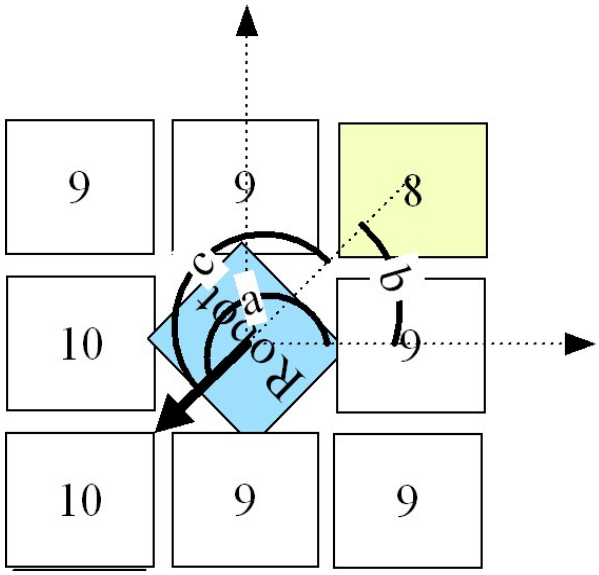

Figure 4.12 Moving Robot on a field.

There will be two conditions according to angle “c”, it will be smaller then 180 degree or not, according to that selection program will consider about angles “a” and “b”. While choosing a turning way program needs to consider about speed control (run fast). If Run Fast selection is true then program will choose 7 or 9 if it is not true then it will chose 4 or 5 ways. As an example (Figure 4.12), if angle “c” is smaller than 180 degree and angle “a” is bigger then angle “b” then selection of turning way will be the right direction, according to speed control (run fast) it will be Forward Right (7) or Right (4). This algorithm is shown in Figure 4.13 path planning part 2. To choose a way of forward or backward again angle “c” is considered. If angle “c” is smaller then 15 degree it means that path cell is almost in front of the robot, then Robot way is loaded with 8(Forward) and Robot power is loaded with 10 (Max. power). When Forward or Backward way is selected program will not enter to Part three, there is no need to calculate the motor power it is already loaded with max speed. Similarly if angle “c” is between 165 and 180 degree which means path cell is almost in backwards of robot, then Robot way is loaded with 2 (Backward) and Robot power is loaded with 10(Max power).

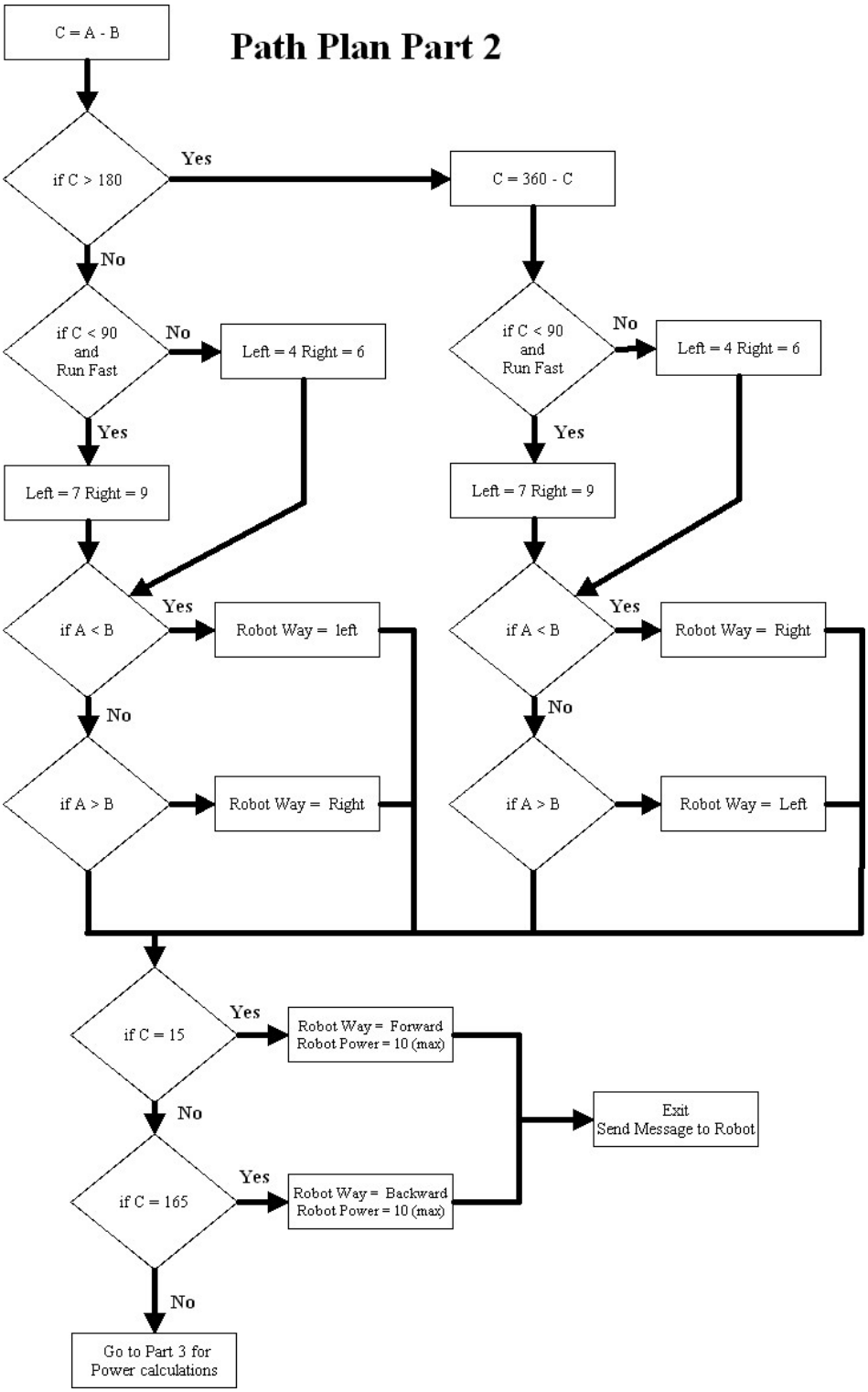

Figure 4.13 Path Planning Part 2

In Phase Three; Robot motor powers are calculated. When Robot turning way is selected, motor powers needs to be calculated for turning robot to that direction efficiently. The idea is so simple, if angle “c” has a small angle value like 30 degree Robot needed to turn with a small angle to path cell with a small motor power %30 percent power. It is like a linear equation which is Robot Power = (Angle C) * n. Figure 4.14 shows this selection algorithm.

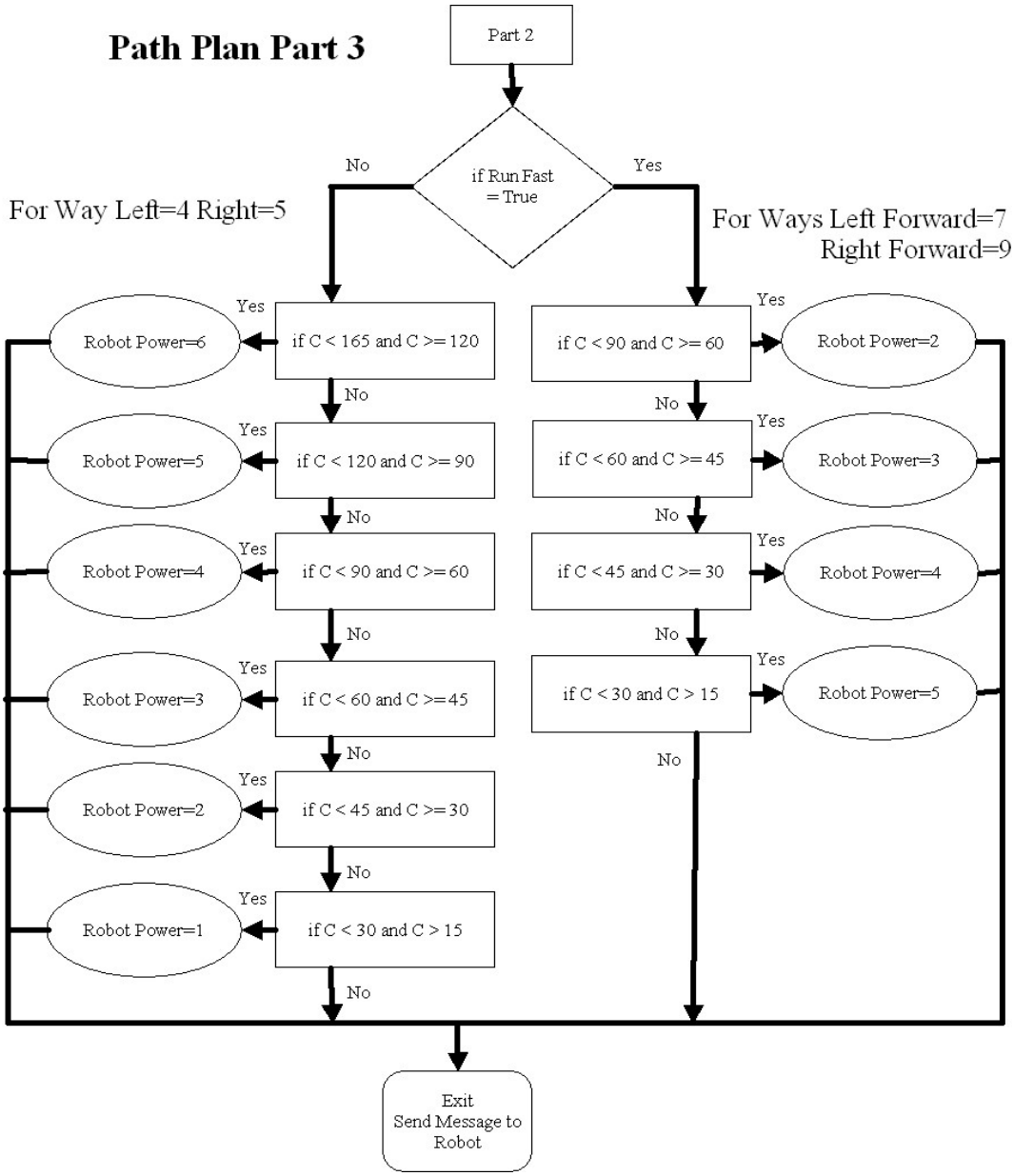

Figure 4.14 Path Planning Part 3

For example if Figure 4.14 is considered it can be seen that Robot's motor power is calculated according to angle "c"(Figure 4.12). If angle "c" is between 120 degree and 90 degree Robot's motor power is selected as 4 which's nearly half of full power (10). With help of these power selections robot can move more accurately. If run fast selection is true program will select different power values because of the different way selections (7 and 9). As it was shown in Figure 4.10 left forward and right forward works different from other way directions, in these ways Robot power byte just controls one wheel, other wheel always works with full power. So if we load 1 power to left forward direction, left tire will turn with power 1 (min power) and right tire will turn with 10 (full power), in this condition robot will turn with max sharpness to left forward. Simply it can be told that Left Forward and Right Forward (7 and 9) power idea works (opposite) different then the other way directions.

4.4 Experimental results

After these three phases Robot program structure is loaded with way and power information. This information is send to Robots through the Rf communication. In real soccer field Robots reach their destination efficiently and with minimum distance error, but of course some collusions may occur with obstacles. Controlling Robots movements is a hard job because of the voltage changes in batteries. At low power all calibration will be changed. Potential updates and path planning process takes 5 ms with Pentium 4 1.4 Ghz processor which is enough and so quick for real time applications. In Figure 4.15 Robot1 has a path to a target which is behind the ball. This path is draw while considering Robot's center so when Robot stared to move through this path, path will be updated and reconstructed for every new frame. While moving robot looks to its neighbor cells not the whole path.

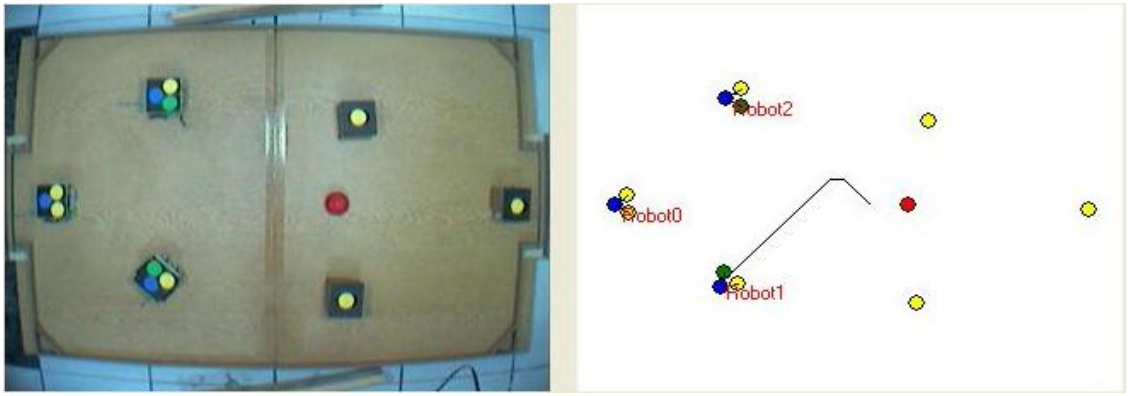

Figure 4.15 Path Planning in Computer Program.

CHAPTER FIVE

STRATEGY

5. Organization and Strategy

The functionality of Team robots are divided into three levels based on the level of abstraction from basic motion: (1) at the lowest level, agents must be able to follow specified behaviors. (2) In second level it uses combination of these behaviors to build some complex abilities roles.(to get behind a ball, move toward a goal, take the Ball from corner, and so on). (3) Robots must have individual and team-oriented strategies and roles that create goals for the Robot (places the agent desires to be and courses of action the Robot desires to take). The following subsections describe each of these levels in turn.

5.1 Options on Strategy

In (H. K. Lam and others, 2001) they present a decision marker, which will select the appropriate tactics and actions for soccer robots according to the condition of a tournament. The selected tactic will be employed to assign each home robot an action to play the game. The decision marker consists of a tactic database, an action database, a tactic selection algorithm, a tactic selector and an action selector. This idea also uses probability for tactic selections.

In (Bernhard Nebel & Yuliya Babovich, 2002) behavior networks is used, which support real-time decision making. Robotic soccer appears to be one domain where behavior Networks have been proven to be particularly successful. In reference they analyze the reason for the success by identifying conditions that make behavior networks goal converging, i.e., allow them to reach the goals regardless of which particular action selection scheme is used. In terms of STRIPS domains one could talk of self-solving planning domains. In (Prof. Dr. Thomas Bräunl and Birgit Graf, 1999) and (Manuela Veloso and others, 1998) strategy a behavior based control is used which is preferred used in this project.

5.2 Behavior Based Control

To build a strategy, for the beginning it will be much easy to use some simple movements and abilities, which can be added together to build up much complex abilities and coordination. Behavior Based Control is already used in Robocup soccer, every team built up its own strategy and uses this method in their team architecture. It is tried to build a strategy which is based on Behaviors and with their combinations Roles. Strategy algorithm is constructed with three layers. Lowest layer is Behaviors (Level 1) second level is Roles (Level 2) and last level is the Director level which gives duties to robots (Level 3), all levels can control its sub level and will change its application. In following chapters all simple behaviors and their combination (Roles) will be examined. Figure 5.1 shows this strategy levels.

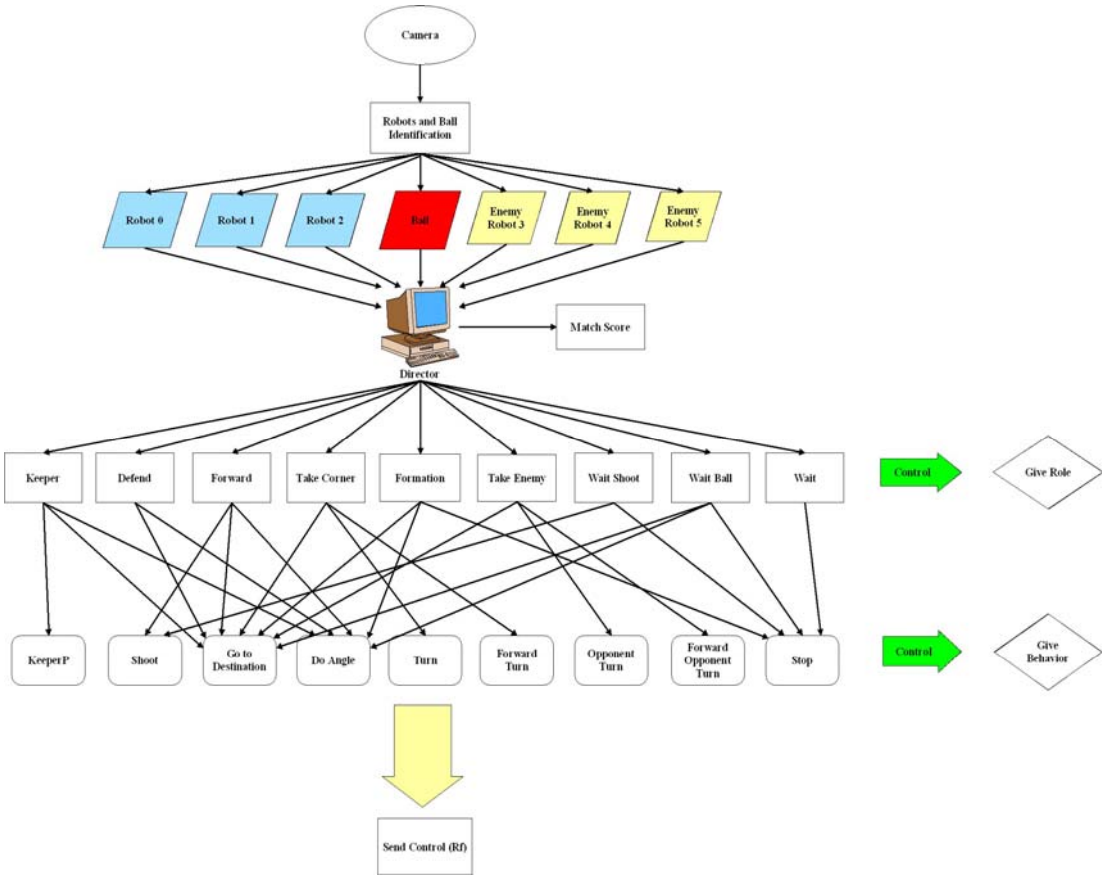

Figure 5.1 Levels of strategy

5.2.1 Simple Behaviors

Behaviors are constructed with combinations of basic robot movements (Turning, Running forward and so on). In strategy algorithm robots have nine behaviors these behaviors are basic and also have the ability to handle some hard situations. These behaviors are; Shoot, Go to Destination, Do Angle, Turn, Forward Turn, Opponent Turn, Forward Opponent Turn, Stop and Keeper Path. All these behaviors are controlled, set and cleared by Behavior manager (Level 1). But it has to be knowen that this is not the Strategic selection, that part is done in the Director Level (Level 3) which is the intelligence of strategy.

5.2.1.1 The Behavior Manager

The Behavior manager controls behavior changes. It calculates the applicability and reward for all behaviors. The behavior manager may also terminate a behavior which is not making progress. The switching of behaviors is constrained by a time function, which limits the behavior manager from switching behaviors too often or too quickly. This way, the agent deals with large changes in the highly dynamic domain of robot soccer. The Behavior manager is shown in Figure 5.2.

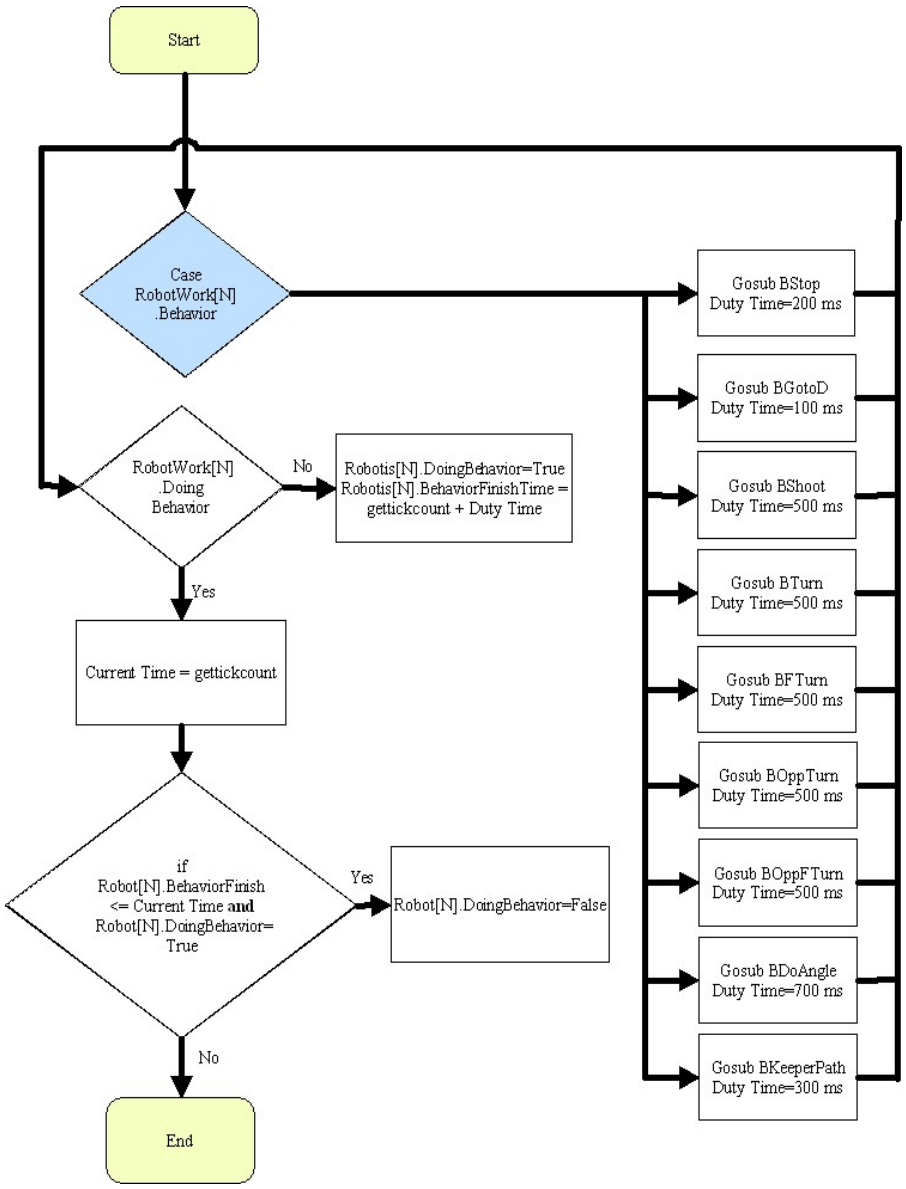

Figure 5.2 The Behavior Manager.

All Behaviors have a specific time period. When a Robot is set to one behavior that behavior can't be switched off until it's time period is over. This period application is a must for strategy if this idea isn't used Robot will change behaviors so quickly so often and this will be very bad in real life applications. Let's examine robot behaviors.

5.2.1.2 Shoot Behavior

When Robot is set to shooting behavior from Level 2 (Role) Robot will shoot with forward way and maximum power. This behavior will continue and can't be changed for 500ms, because of this Robot continues running forward until 500ms is over. In algorithm Robot way is loaded with 8 (Forward) and Robot Power is loaded with 10 (Max Power) and this information is sent to robot by Rf communication.

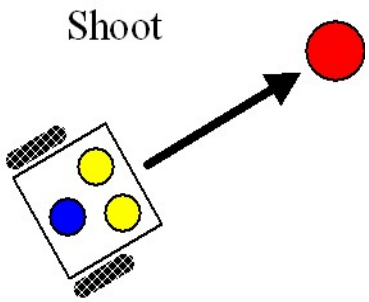

Figure 5.3 Shoot Behavior

5.2.1.3 Go to Destination Behavior

This behavior is the most complex behavior in robot team. if this behavior is selected it can't be switched off for 100 ms. Go to destination algorithm has been examined in Chapter 4. This algorithm is used as a behavior for moving the robot to a destination. When Robot loaded with Go To Destination behavior then BGotoD (Figure 5.2) subroutine is called and this subroutine delivers the desired destination to Potential update and Path Planning Algorithms. As it is mentioned before, path planning algorithm calculates the robot movements. Robot will follow the path with its new way and power calculations. This behavior is used in %90 percent of strategy, as it is mentioned before going to a destination behavior is so important and also so much used in Robot soccer.

5.2.1.4 Do Angle Behavior

This Behavior looks similar to the path fallowing in part 3.3.2 (Moving Robot on a field). This behavior is used for controlling Robot's front looking direction. If this

behavior is selected it can't be switched off for 700ms which is controlled by the behavior manager. For example when robot comes near the ball, according to its Role (Level 2) it selects the Do angle behavior for turning Robot's front towards the ball and keeps controlling it for 700ms. In Figure 5.4 is an example of Do angle behavior, target will be selected as ball and Robot will turn left side to turn its face to ball.

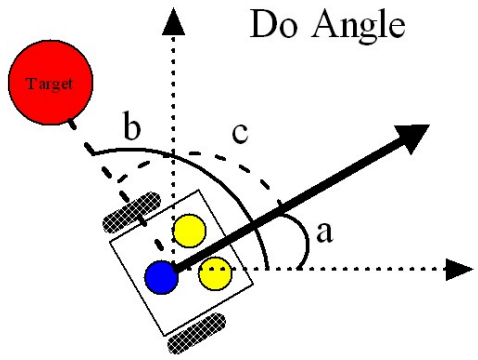

Figure 5.4 Do Angle Behavior

Target can be changed either to ball or to enemy goal point. This behavior is also so useful for robot control. It may be used in many conditions; for example while shooting, the ball has to be in front of the robot. In this behavior algorithm again three angles are considered 'a' is robots forward angle 'b' is target angle and 'c' is the difference between them. Program calculates the values of these angles to find the destination and speed of the movement. Speed (Power) is based on angle 'c' and turning way is determined by angle values of 'a' and 'b' this algorithm is shown in Figure 5.5.

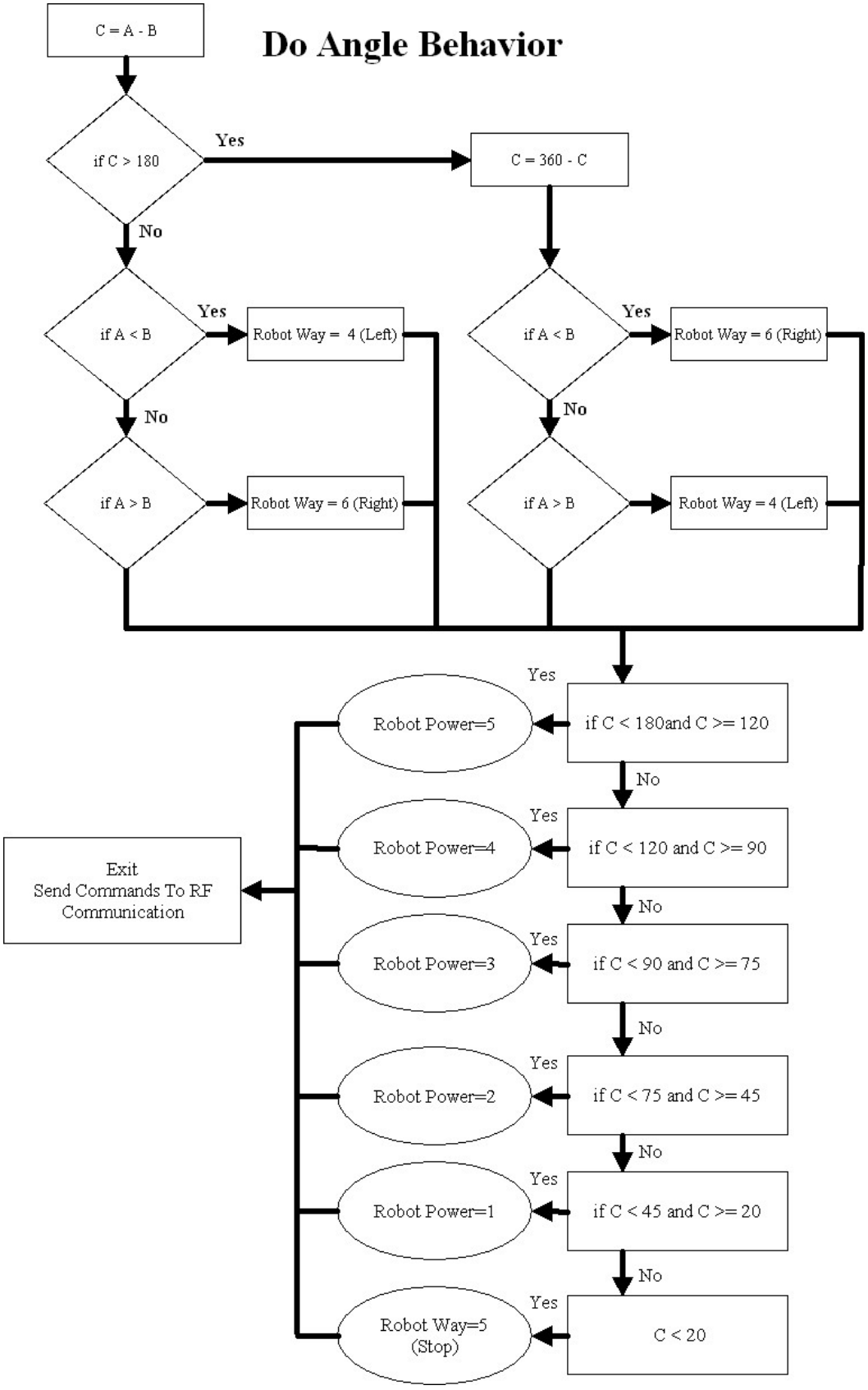

Figure 5.5 Do Angle Behavior Algorithm

5.2.1.5 Turn Behavior

This behavior is used for two purposes; taking ball from corners and intercepting opponent team members. Robot will turn left or right with full speed and drive circles. This movement is done with high speed to increase the chance of touch the ball and to change its direction. If this behavior is selected it can't be switched off for 500ms this is controlled by the behavior manager. Figure 5.6 shows this movement. If ball is stuck at corner, according to Role selection Turn Behavior is loaded to Robot. After this, Robot will start rotating to hit the ball and take it from corner.

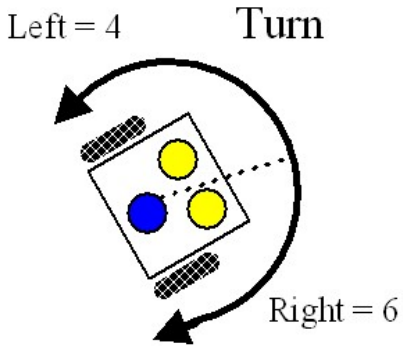

Figure 5.6 Turn Behavior

5.2.1.6 Forward Turn Behavior

Forward turn behavior is an alternative to the Turn behavior, instead of rotating with center circle Robot turns a circle with a bigger radius. It uses left forward (7) and right forward (9) for the robot movements and sets the robot power to 1 (minimum) for a sharp turn. For example if a left turn movement is selected left wheel will move with power 1(min) and right wheel will move with power 10(max) this configuration will make robot turn left with a radios which is equal robot size (10cm). If this behavior is selected it can't be switched off for 700ms this is controlled by the behavior manager. If the ball is stuck for two second in a corner or near an edge, Forward Turn behavior is activated and robot will leave the turn behavior and try to hit the ball with an another movement (Forward Turn). This behavior is shown in Figure 5.7.

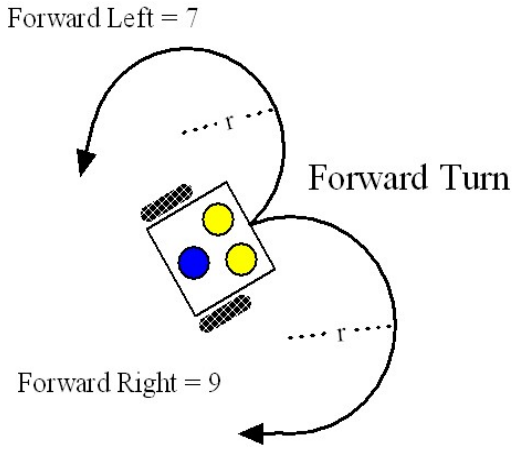

Figure 5.7 Forward Turn Behavior

5.2.1.7 Opponent Turn Behavior

This behavior is similar to Turn behavior, the only difference is the usage. Opponent Turn behavior is used for intercepting the opponent team members. In Figure 5.8 there are four regions that ball can be take place. According to the ball place, program will choose a turning way; in A2 robot will turn to Right (6), in B2 robot will turn to Left (4), in A1 robot will turn to Left (4), in B1 robot will turn Right (6). In these turning actions robot will rotate with maximum power and increase the chance of intercepting the ball. This behavior is done for 500ms and can't be changed in that period.

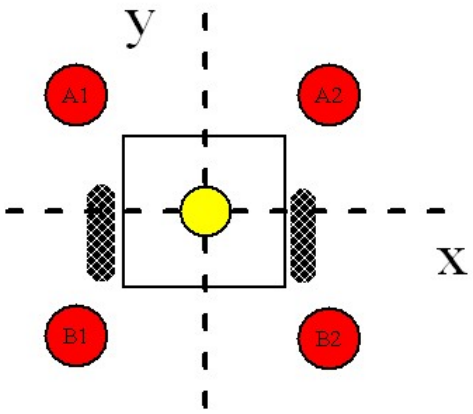

Figure 5.8 Opponent Turn Behavior

5.2.1.8 Forward Opponent Turn Behavior

This behavior is a combination of Opponent Turn behavior and Forward turn behavior Robot will turn with a big radios and try to hit the ball which is in front of the opponent team member. Again turning way selection is similar; if ball is in A2 robot will turn to Forward Right (9), in B2 robot will turn to Forward Left (7), in A1 robot will turn to Forward Left (7), in B1 robot will turn Forward Right (9). If the ball is stuck beside to an opponent member for two second, according to role selection level this behavior is activated. Robot will leave the Opponent turn behavior and try to hit the ball with an another action (Forward Opponent Turn). This behavior is done for 500ms and can't be changed in that period.

5.2.1.9 Stop Behavior

This behavior simply stops the robot from moving and loads the robot structure with Robot way to 5 and Robot power to 0. This behavior used in all Roles (level 2) but commonly used for waiting.

5.2.1.10 Keeper Path Behavior

This behavior is only used by keeper. After keeper is placed parallel to goal line by the Role Level, this behavior is activated which will move the robot parallel to goal line. In Figure 5.9 blue dot is the target of the keeper and according to target position program will select the way of movement. If the target is placed in upper side (Target y is smaller then Keeper center y. Y axis operates mirrored in Delphi) of the keeper as in Figure 5.9 program will chose forward direction (8) other wise it will chose backwards direction(2). Power selection is done according to distance of target (h). If "h" is smaller then 16 pixel Robot's Power will be selected as 3, other wise it is understood that target is far from keeper and Robot's power will be selected as 6 which will move robot faster. By these speed controls keeper will catch the ball much quicker and also will be much constant when blocking.

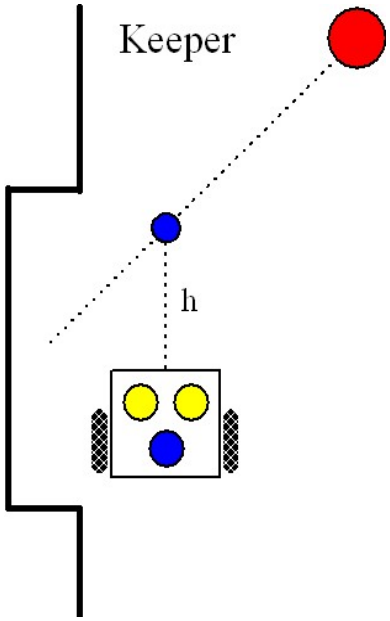

Figure 5.9 Keeper Path Behavior

5.2.2 Combination of Behaviors Roles

After defining behaviors now some Roles can be constructed which are combinations of these behaviors. Role selection part is level 2 in strategy program according to Director Level (level 3) any robot can take any role in the match field.

5.2.2.1 The Role Manager

Role selection algorithm is controlled with the Role Manager which controls role timeouts and subroutines. It works similar to Behavior manager only difference is Role manager controls Roles. Again Role manager is not the brain of the team it just controls the organization of roles. Role assignment is done in level 3 Director Level part. Figure 5.10 shows this algorithm. It also controls robot if it's doing behavior or not, if Robot is doing a behavior there is no need to go to Role selection part because as it has been discussed before all behaviors take time and until it is finish they can't be changed. This idea is also used in Role selection if a Robot is assigned to a role it can't be changed for a specific time this method helps strategy for not changing Roles so often and so quickly. Lets examine all Roles these are nine roles in strategy program these are; Forward, Defend, Keeper, Take Corner, Formation, Take Enemy, Wait Shoot, Wait Ball, Wait.

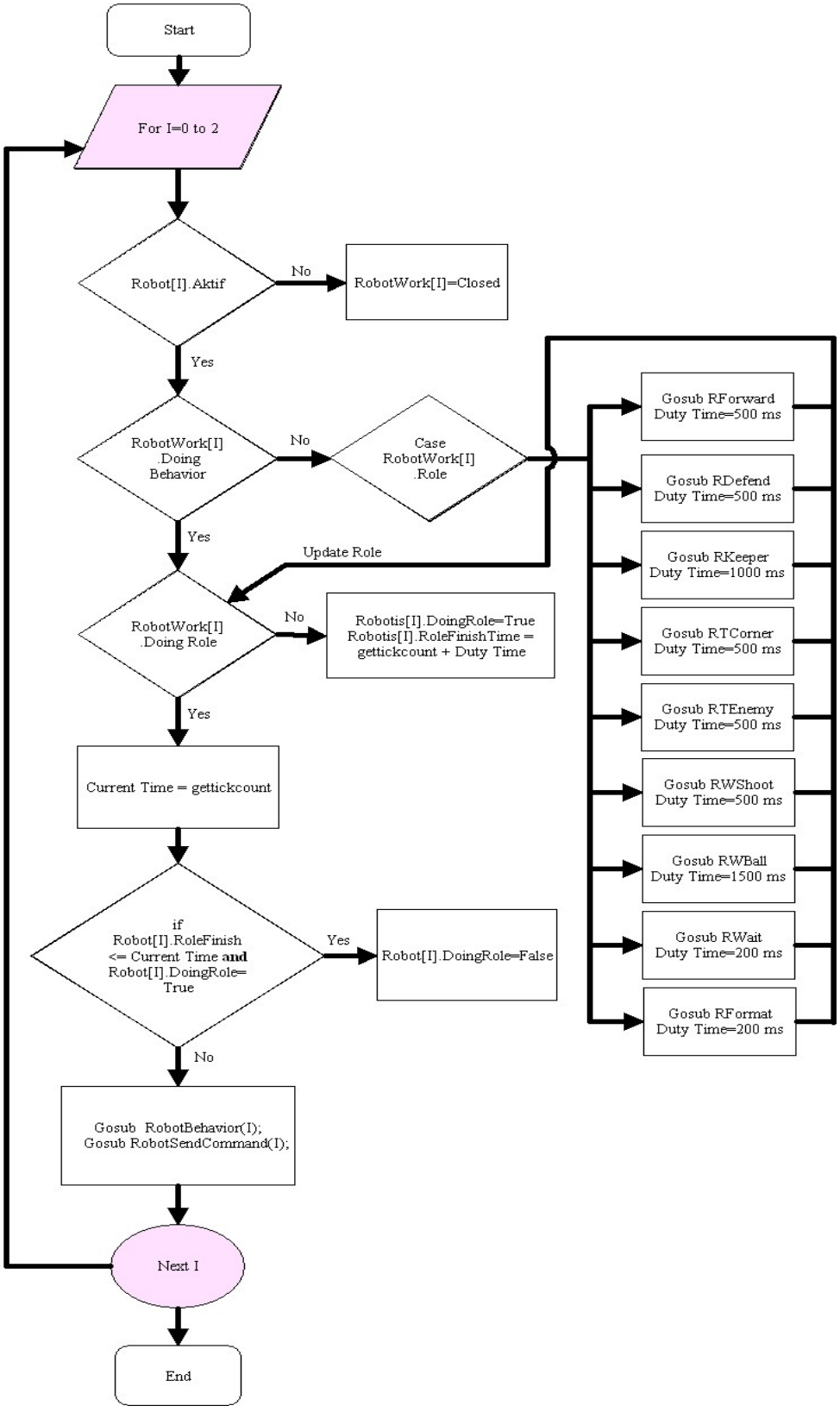

Figure 5.10 The Role Manager

5.2.2.2 Forward

When forward role is selected for a Robot there will be one condition in which the Robot must occupy in opponent's half field. In Figure 5.11 Robot is in opponent team's half field and is preparing for scoring. As it is mentioned before when robot takes a role it will do some basic movements (behaviors) and use them in some order. In Forward behavior Robot can use three behaviors, these are Shoot, Do Angle and Go to Destination. Considering the Figure 5.11 this process can be seen, in first part (1) Robot travels to a position which is behind the ball, to do this it uses Go To Destination behavior (That point is calculated by subroutine Take Score). In second part (2) robot uses Do angle behavior to turn its face to the ball. In third part (3) robot will use shoot behavior for hitting the ball. Take Score subroutine has another duty which is a calculation that finds the empty sides of the opponent's goal. It controls the enemy keeper (which is the nearest opponent robot to the goal) and determines the empty corners which can be G1, G2, G3 in Figure 5.11.

This Role also have a panic shoot ability, if the ball some how passes between the goal and Robot it will shoot instantly without any movements. This ability will increase the scoring chance. This Role takes 500ms and can't be changed by Role manager. Figure 5.12 shows the Forward Role program algorithm.

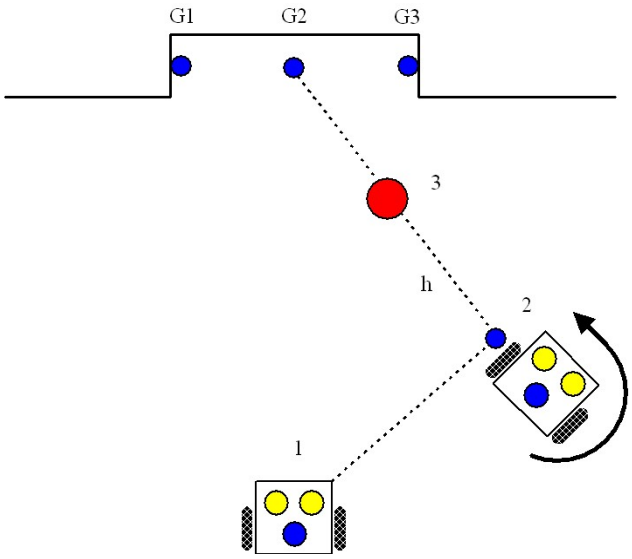

Figure 5.11 Forward Role

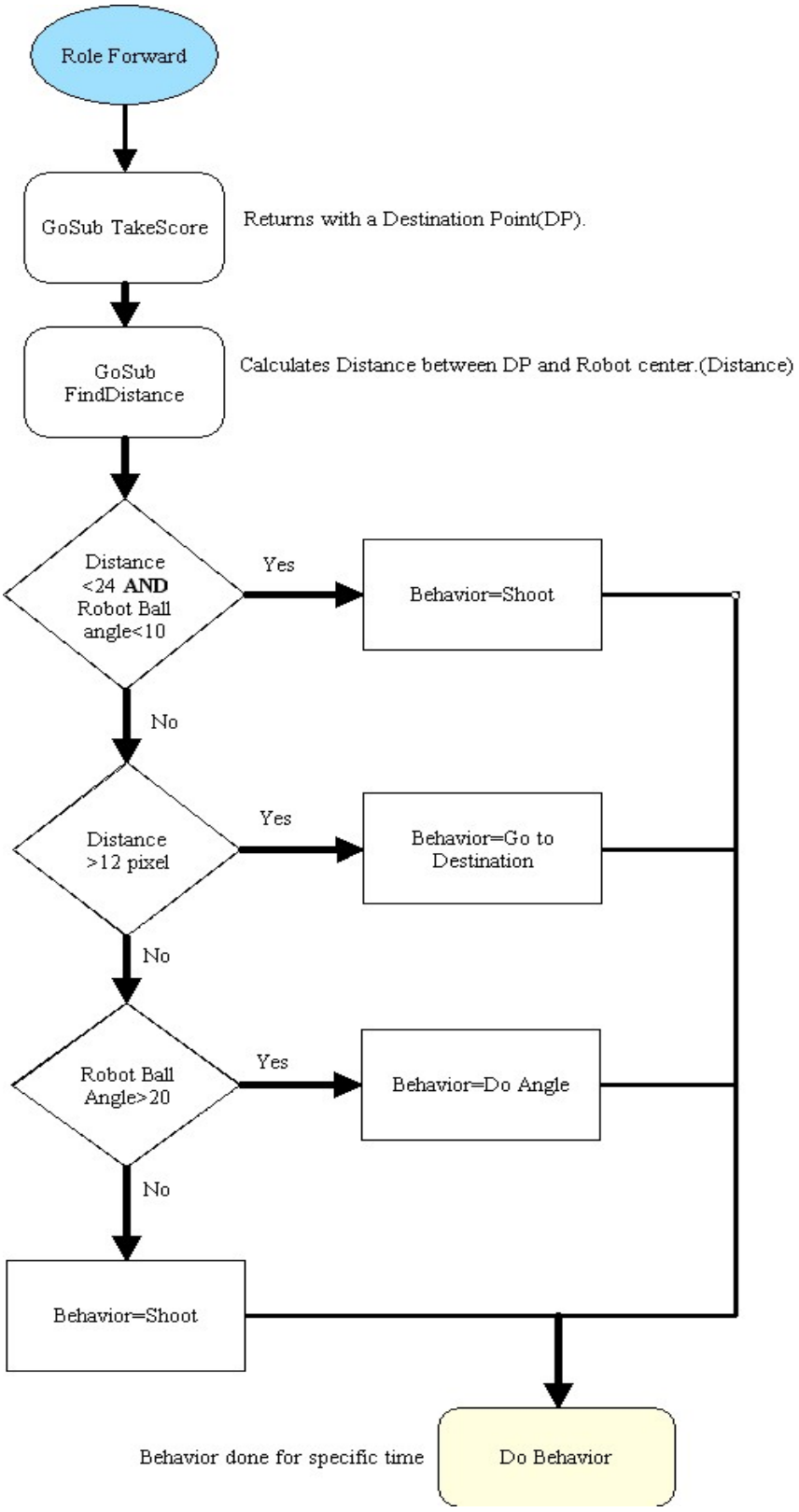

Figure 5.12 Role Forward Algorithm

5.2.2.3 Defend

Defend Role is similar to Forward Role main difference is the pace of the ball in the field. Defend Role is selected in home team's own field side. Take score subroutine calculates a point which is back side of the Ball, Robot will travel to that point and after it takes Do Angle behavior to face to ball. For now on all these process are look similar to Forward Role, but instead of shooting Defend Role uses Go to Destination behavior to drag the ball to enemy field. After Do angle behavior is done Robot tries to drag the ball to enemy goal. This role uses two behaviors Go To Destination and Do angle.

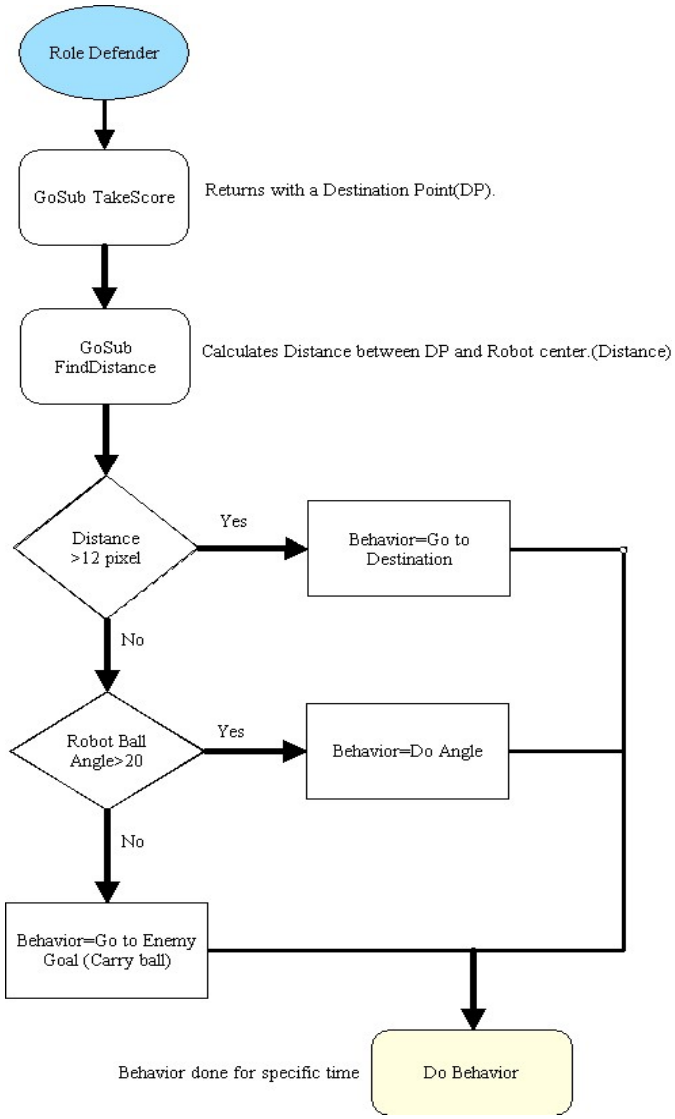

Figure 5.13 Role Defend Algorithm

5.2.2.4 Keeper

Keeper Role uses four behaviors; Go to Destination, Do angle, Stop and Keeper Path. Keeper's duty is limited with three conditions which are shown in Figure 5.14; first (1) if robot is out of the Keep Area, Robot uses Go to destination behavior for returning back to Keeper area. This ability is also good in some conditions which robot forced to leave its own goal by opponent team members (hit and crash). Second (2) Keeper Robot uses Do Angle behavior to make its body parallel to goal line, this movement helps the robot keeping goals much better, simply it moves forward and backwards to keep the goal. In third part (3) Robot will move to destination which is between the goal center and the ball, that point is calculated with Keeper Place subroutine in which algorithm uses line equations to find the center blue dot in Figure 5.14 Keeper Role controls the "h" distance and according to that it chooses one of the behaviors Stop or Keeper path. If ball is in danger position (h = 16 pixel) role will choose Keeper path and quickly it closes the empty corners of goal line. Figure 5.15 shows this Role algorithm.

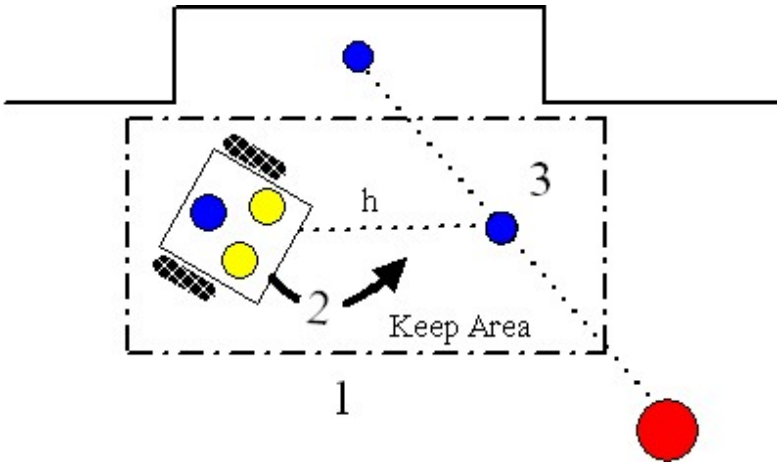

Figure 5.14 Role Keeper

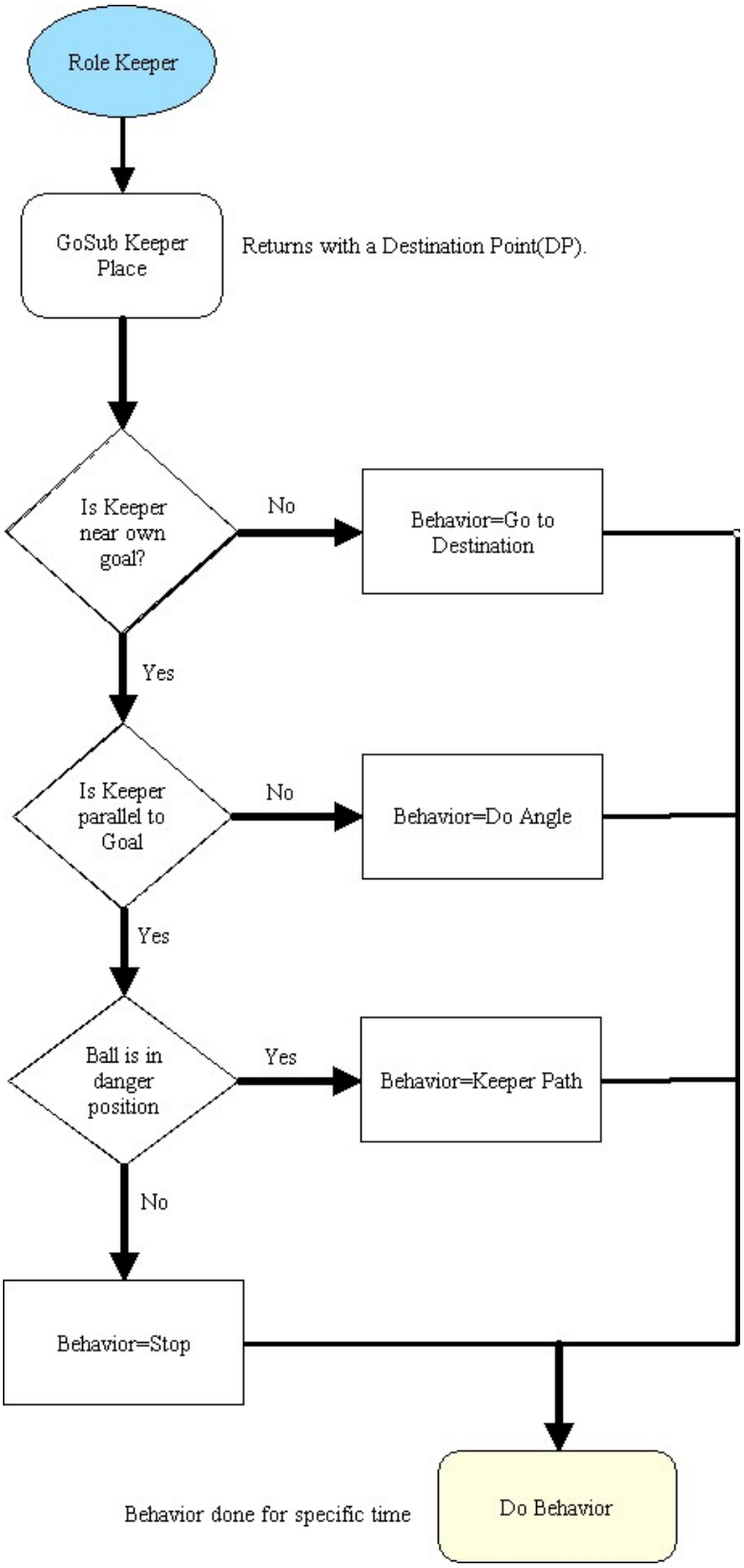

Figure 5.15 Role Keeper Algorithm

5.2.2.5 Take Corner

Take Corner Role uses two subroutines (Ball Stuck and Ball in Corner) and four behaviors. In Figure 5.16 it is shown that Robot has two conditions; first robot uses go to destination behavior to go to a target which is behind the ball and also not too close to corner line. Second Robot will use the turn behavior and try to hit the ball to get it out of corner. This condition is done for two seconds after this program will calculate the ball is stuck or not (Ball Stuck Subroutine). If ball is not moved for two seconds robot will change behavior to Do angle behavior to face the ball and after that uses Forward Turn behavior to take The Ball from its stuck position. Forward Turn behavior increase the chance of shooting, taking ball from corners is a hard job turning behavior sometimes is lack for this but Forward turn will handle this problem. There two subroutines are used; one is Ball Stuck which controls the ball center, if it is not moved for 2 second Ball Stuck subroutine gives an alarm to Take Corner Role program. Second subroutine is Ball in Corner Sub. this algorithm returns with a target point which is usually behind the ball (Considered to opponent goal). It also chooses a target which is helpful for the ball recovery from own goal and a shooting point when the ball is beside the enemy goal. When ball is near own goal line Ball Corner Subroutine calculates a target which is safe for ball recovery (Try to pretend from having a goal) because when using the Turn behavior Robots will take score to own goal. When ball is near to enemy goal line Ball Corner algorithm calculates a target which will increase the chance of scoring. Turning behavior can be used for shooting. Figure 5.17 shows the Take Corner Algorithm.

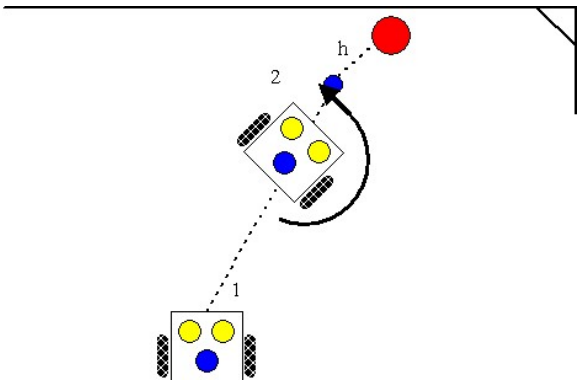

Figure 5.16 Role Take Corner

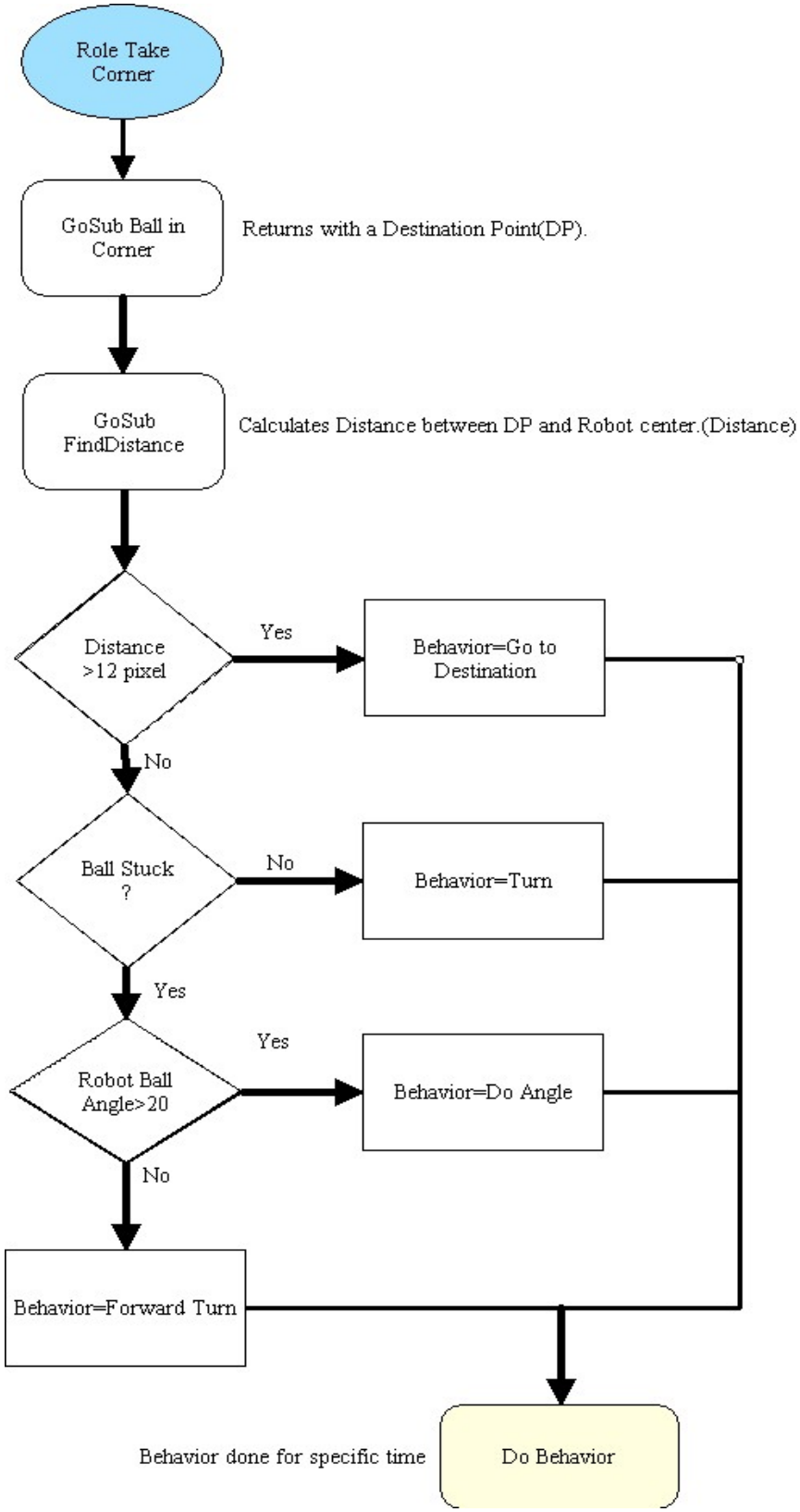

Figure 5.17 Role Take Corner Algorithm

5.2.2.6 Formation

This Role is used when a team is scored a goal. It uses three behaviors (Go to Destination, Do angle and Stop) and a subroutine which is for Formation Points. Formation point subroutine gives specific targets to robots which are constant regions in the field. These regions are; for keeper it's the center of own goal line and for other two robots it's the middle side of the filed. First of all Robots will move to their targets and after that they will face to enemy goal and stop. Match will restart if ball is placed to center line.

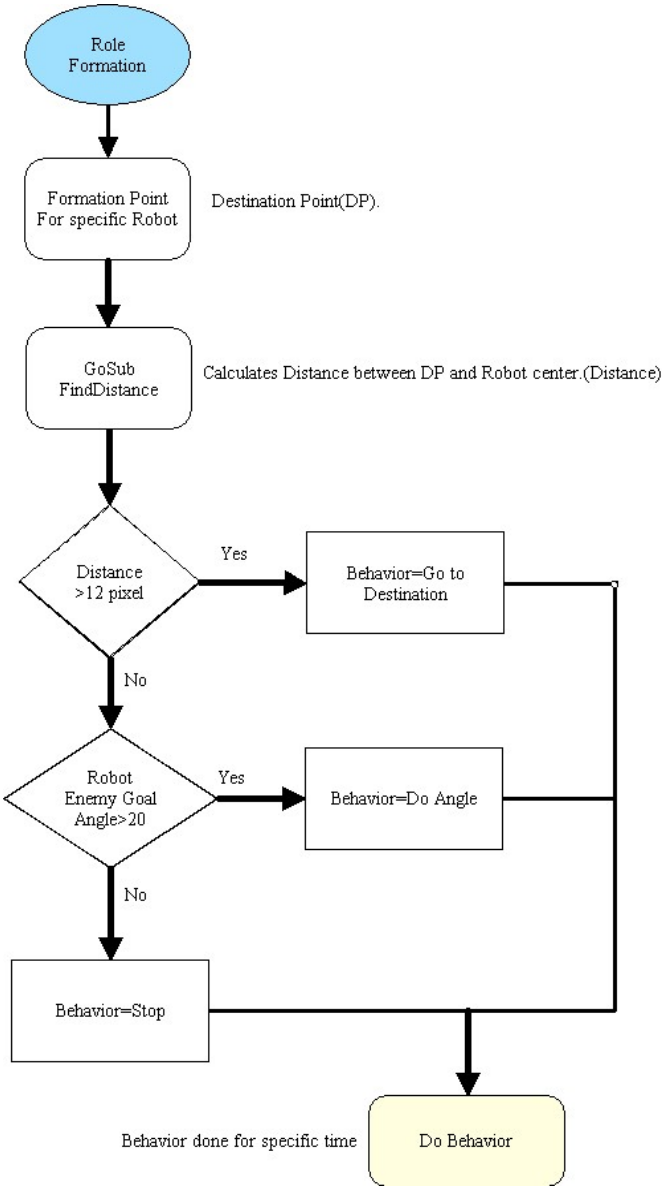

Figure 5.18 Role Formation Algorithm

5.2.2.7 Take Enemy

This Role is similar to the Take Corner Role. This Role uses two subroutines one is Take ball which returns with a target near to ball but also not so close to opponent robot which is seen in Figure 5.19 second is The Ball Stuck subroutine which returns with true or false. This Role uses four behaviors (Go to destination, Do angle, Opponent Turn and Forward Opp. Turn). In Figure 5.19 it is shown that Robot has two conditions; first robot uses go to destination behavior to go to a target which is behind the ball and also not too close to opponent robot. Second Robot will use the Opponent turn behavior to hit the ball and to get it away from the opponent robot. Opponent turn behavior will continue for two seconds after this program will calculate the ball is stuck or not (Ball Stuck Subroutine). If ball is not moved for two seconds robot will change behavior to Do angle behavior to face the ball, after that it uses Opponent Forward Turn behavior to take the Ball from its stuck position. Opponent Forward Turn behavior increase the chance of shooting, taking ball from opponent robot is a hard job Opponent turn behavior sometimes is lack for this but Opponent Forward turn will handle this problem. In this role two subroutines are used; one is Ball Stuck which controls the ball center, if it is not moved for 2 second Ball Stuck subroutine gives an alarm to Take Enemy Role program. Second subroutine is Take Ball Sub. this algorithm gives a target point to Take Enemy Role program. Target is usually behind the ball (Considered to opponent Robot) and far enough from the Opponent Robot that place is a chosen for good intercepting this point is marked as blue point in Figure 5.19.

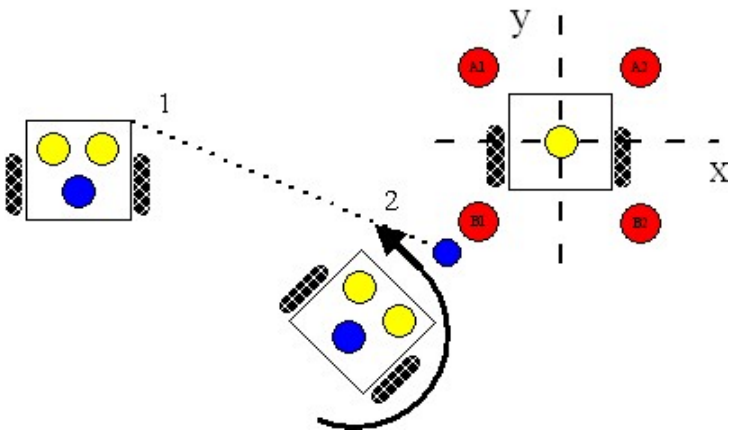

Figure 5.19 Role Take Enemy

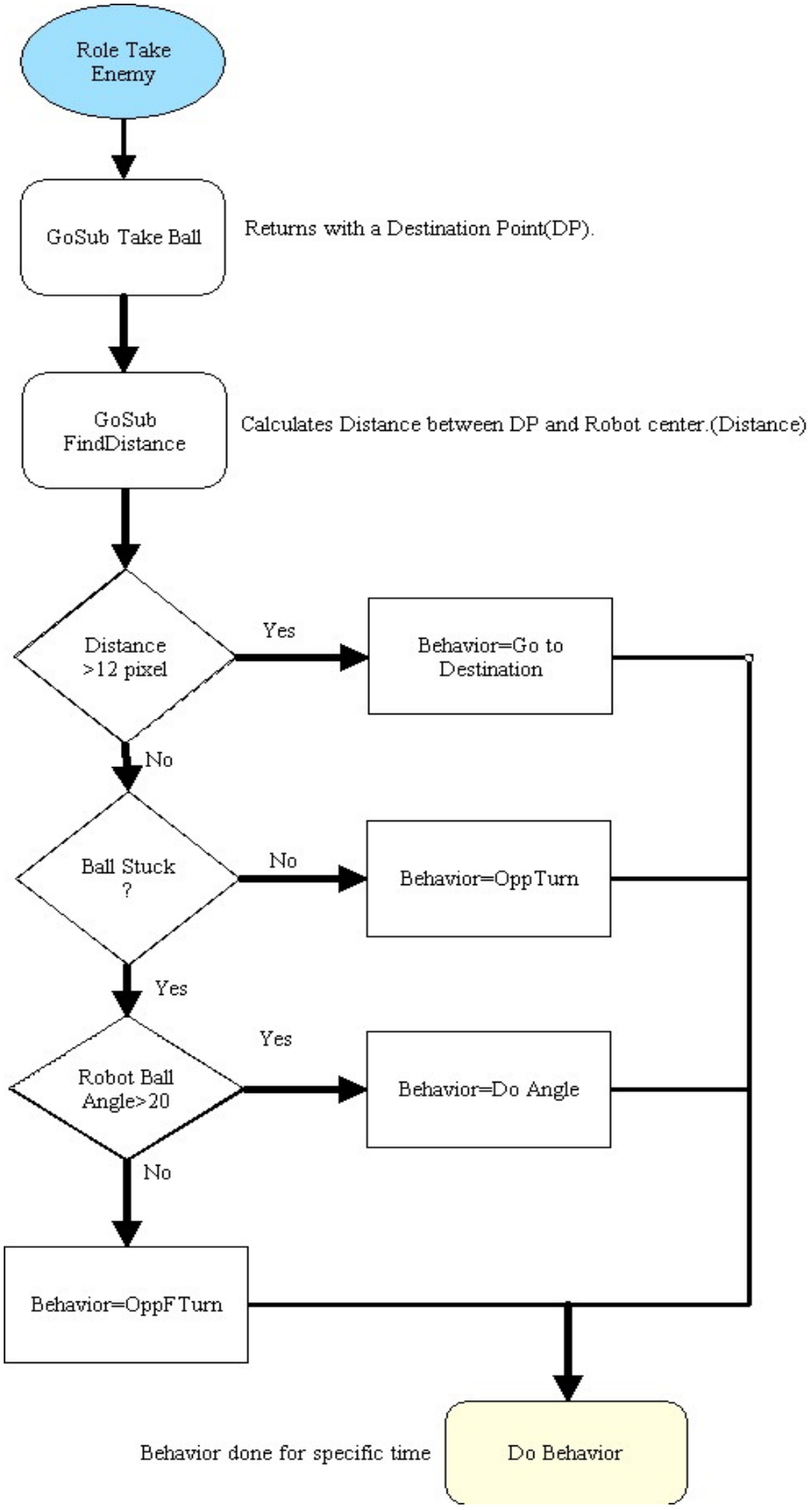

Figure 5.20 Role Take Enemy Algorithm

5.2.2.8 Wait Ball

Wait ball role uses one subroutine (Robot Wait Ball) which calculates a waiting point in field. This role uses three behaviors (Go to Destination, Do angle, Stop). Wait Ball role is used when second team robot is in an aggressive role, for example when it was doing a role of Forward or Defender ect. If Figure 5.21 is considered in the left half size, Robot 1 is close to ball and it will be charged with a role of Defender, according to the Director Level Robot 2 will be charged with Wait Ball role. Robot2 will move to target (which is calculated by Robot Wait Ball subroutine). Target's X axis point is h1 pixel bigger ball's center X point (closer to opponent goal) and target's Y axis direction is same with Robot's Y axis (Target 1 in Figure 5.21). Robot will move to Target 1 with Go to destination algorithm and after that it will use do angle behavior to turn it's face to opponents goal and starts to wait with Stop Behavior. Same process is done in opponents half field only difference is the target point. When right side of the field is considered it can be seen that Robot 1 is near to the ball which will be charged with Forward Role and Robot 2 will be charged with Wait Ball Role. Target 2 is calculated h2 pixel smaller then Ball center x, this idea is used for increasing the chance of shooting to hoping ball. Robot 2 will hit to ball which is approaching to it's front, that will be explained in Wait Shoot Role. Safe areas are used for not to change the waiting point so often and so quickly, once Target is calculated it was not changed until Ball gets out of these safe areas. This idea is good for wait shoot role because shooting to a running ball is a hard job and waiting point must be stable. Figure 5.22 shows the algorithm.

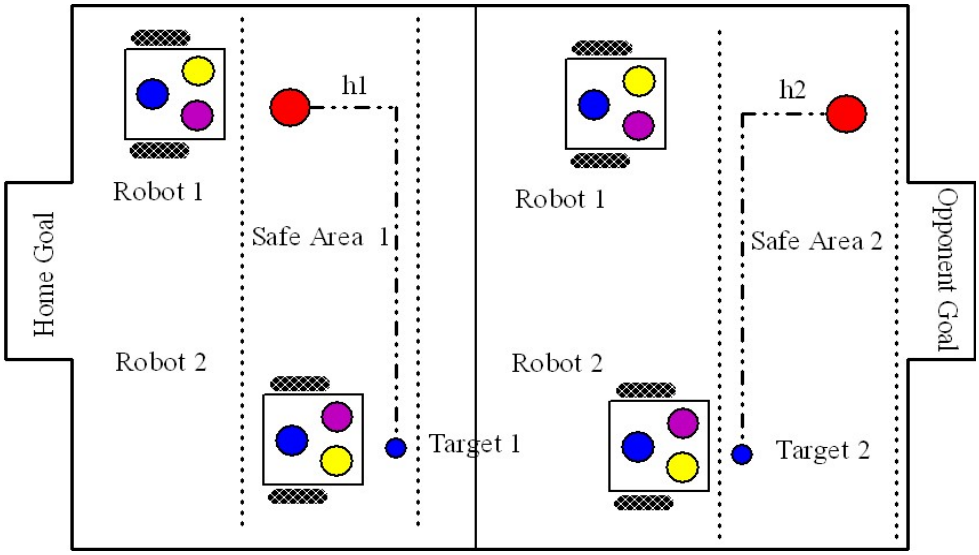

Figure 5.21 Role Wait Ball

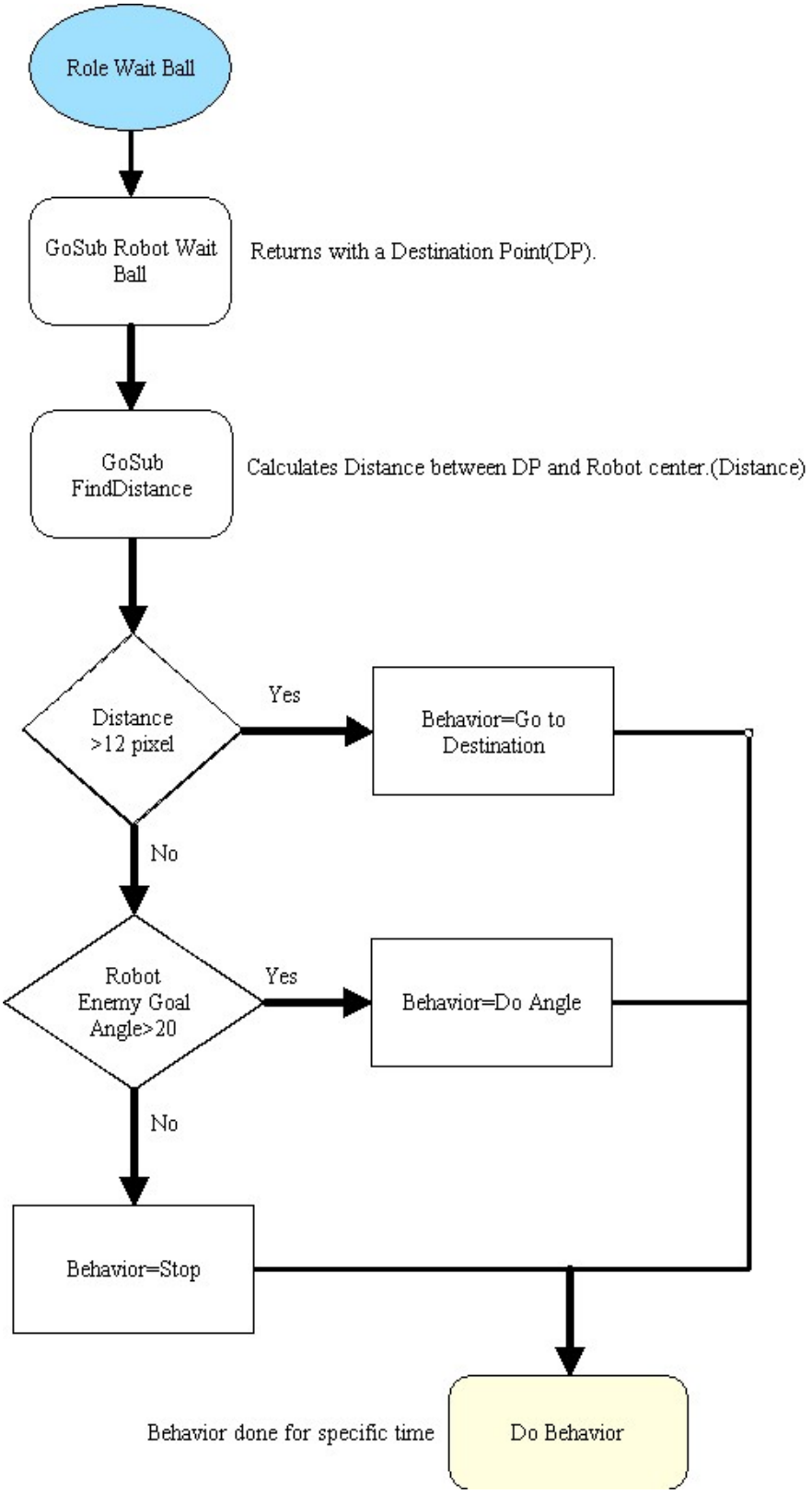

Figure 5.22 Role Wait Ball Algorithm

5.2.2.9 Wait Shoot

Wait shoot behavior uses one subroutine (Hit running ball) which calculates the connection of two direction equations (Target in Figure 5.23) and uses two behaviors (Shoot and wait).The Director Level also considers Hit Running ball subroutine to decide giving a wait shoot role to a robot. Let's consider figure 5.23 there are four variables these are Ball's speed (Ball v), ball distance to target (h1), Robot's speed (Robot v) and Robot distance to target. As a rule (Ball v) / h1 = (Robot V) / h2 equation must be correct for an interception on target. Hit Running ball subroutine uses two equations Ball line and Robot line which will help to find their connection point. After this subroutine controls the distances and velocities which can occupy an interception on target, it also gives some time tolerance of 500 ms for interception. When Robot is charged with Wait Shoot behavior (Director Level) it is known that ball is approaching to Robot's front, but the exact time isn't know. To avoid miss shooting Wait Shoot role algorithm again considers Hit Running ball subroutine and waits, when connection is confirmed (all distances and velocities are ok) Robot will shoot. Figure 5.24 shows this algorithm. If connection is not confirmed robot will leave waiting and will take another role from the Director Level.

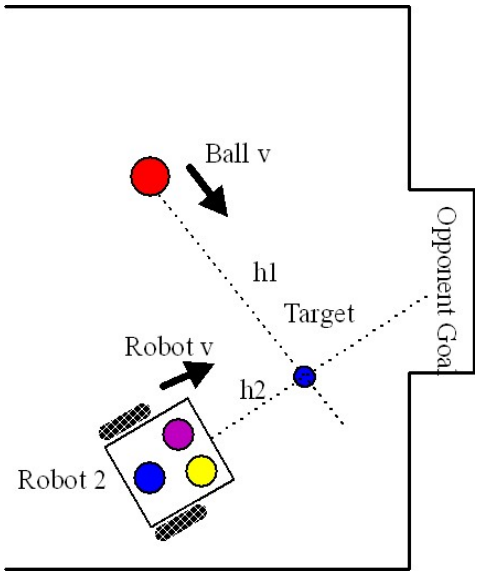

Figure 5.23 Role Wait Shoot

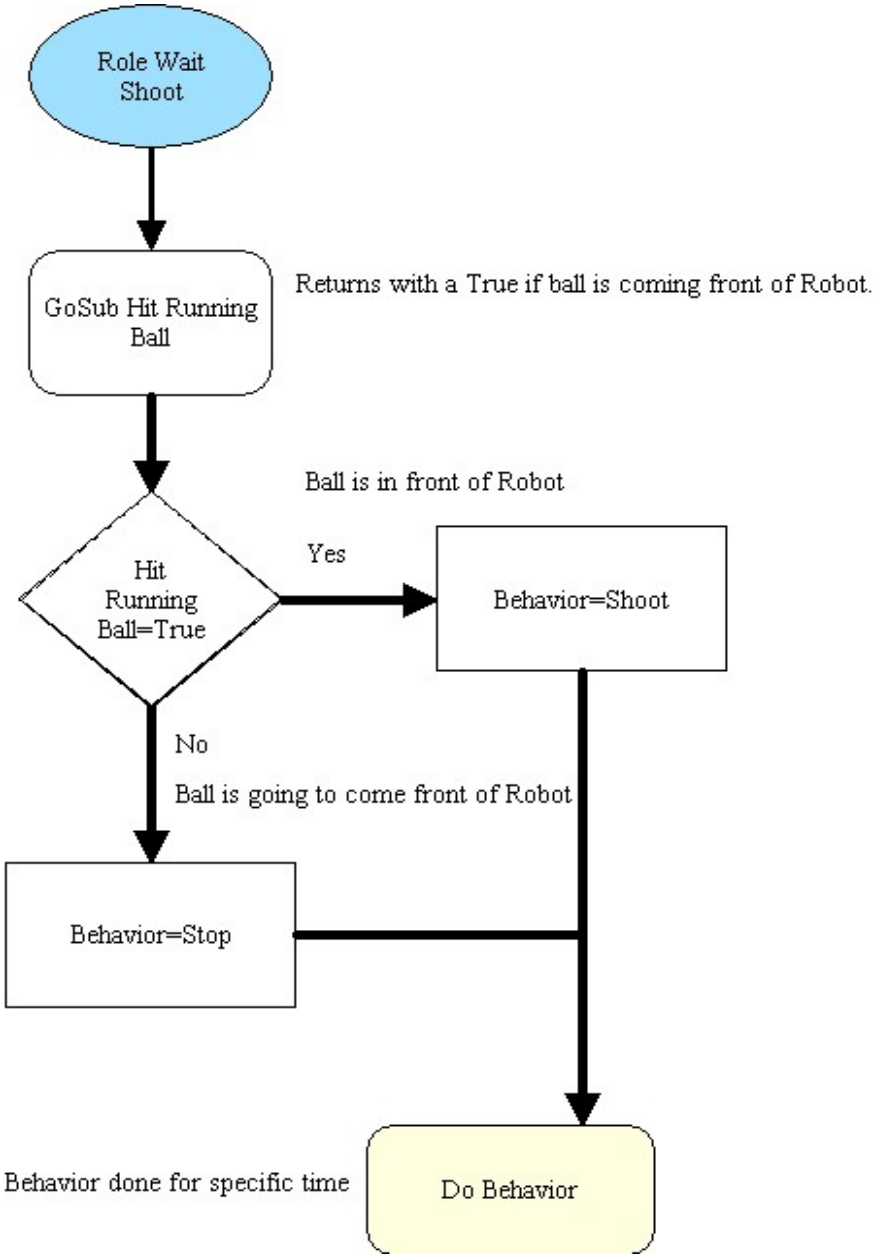

Figure 5.24 Role Wait Shoot Algorithm

5.2.2.10 Wait

Wait Role is just used when a goal is scored, all robots are stopped for 200 ms and after that they set to Formation Role. Figure 5.25 shows this algorithm. In strategy wait role is not so often used but wait behavior used in all Roles.

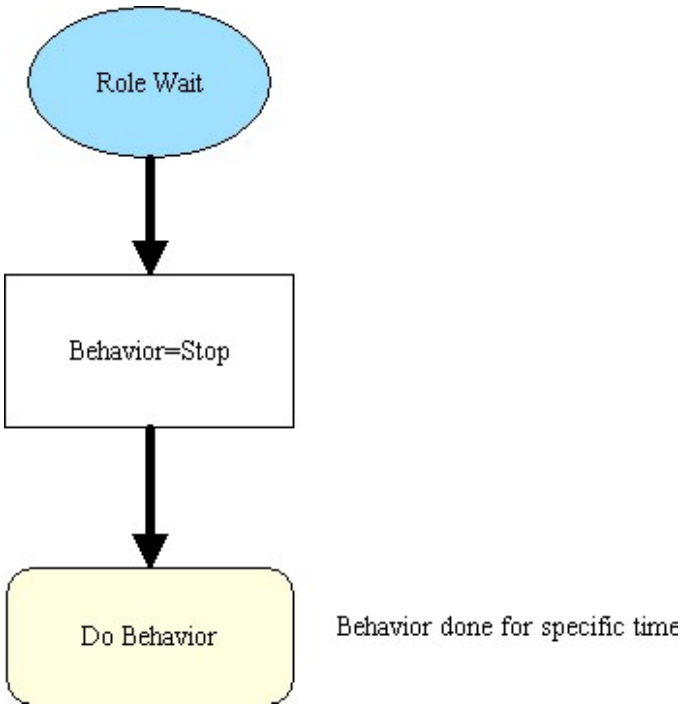

Figure 5.25 Role Wait Algorithm

5.3 Strategy Director Level

When all Behaviors and all Roles are constructed it is time use a role selector algorithm. Main idea in Role selection is based on Match field regions which are shown in Figure 5.26. There are 9 regions and according to ball's place, strategy will be developed. For declarations it is preferred to use some extra identification names which are N1, N2, NK. Robot N1 is assigned to a robot which is the nearest robot to the ball. Most aggressive roles are given to Robot N1 these are the Forward, Defend, Take Corner and so on. Any Robot in the team members can take this identification. N2 name identification is used for the robot which will help to Robot N1 for any of its duty. For example if Robot N1 is shooting or doing an aggressive attack Robot N2 will wait for the ball. Robot N2 will also wait for shooting when Robot N1 has the ball. RobotN2's duty is waiting the ball this will protect home robots for running to same target (ball). Again any robot in team can take label N2. Robot NK is assigned to keeper it will be given to any robot in team but for constant it is preferred using it as Robot 0 (Robot 0 is labeled as Robot NK). Robot 1 and Robot 2 will be set to

either N1 or N2 but this decision is made by Director Level in Figure 5.27 according to place of the ball and place of the robots.

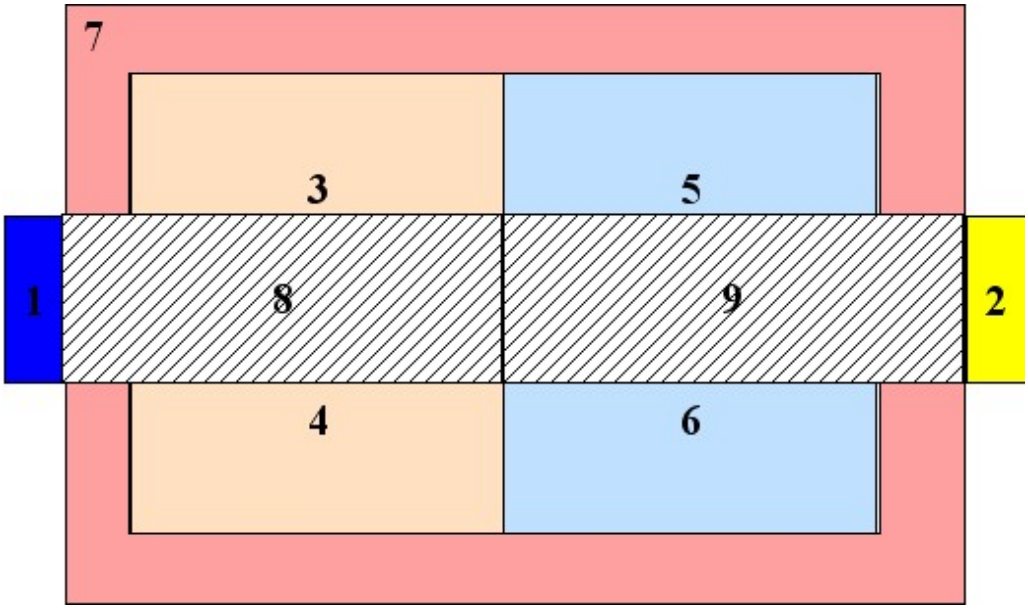

Figure 5.26 Match Field Areas for Strategy

The director algorithm is shown in Figure 5.27 first part of the program Robot N1 and Robot N2 is assigned, this is done by looking to Ball and its place in field. If ball is in regions 8 or 9 (according to Robot Near Ball Subroutine) nearest Robot to ball is selected as Robot N1 and other player robot is selected as Robot N2 as it mentioned before Robot NK is always selected as Robot 0. If the ball is not in regions 8 or 9 again some constant declarations are used if ball in upside of the field (regions 3 or 5) Robot 1 is labeled as Robot N2 and Robot 2 is labeled as Robot N1 if ball is in regions 4 or 6 Robot 1 is labeled as Robot N1 and Robot 2 is labeled as Robot N2.

After Robot labeling part is finished it is time to look at robot role selection algorithm. This process is shown in Figure 5.27 According to the ball field regions (Figure 5.26) role assignments are chosen. If ball is in region 1 or 2, match score subroutine is called (for score board calculations) and Director level assigns Formation role to all robots. If an opponent Robot is near to ball, Robot N1 is

charged with Take Enemy Role and Robot N2 is charged with Wait Ball Role. If ball is in regions 3 or 4 Robot N1 is charged with Defend Role and Robot N2 is charged with Wait Ball Role. If ball is in regions 5 or 6, when a condition of the ball which is hoped from the opponent keeper and approaching to Robot N2, Director manager will charge Robot N2 with Wait Shoot role and Robot N1 will be charged with Wait Ball role. If ball is not approaching to Robot N2 then Robot N1 is charged with Forward and Robot N2 is charged with Wait Ball Role. This algorithm is shown in Figure 5.27.

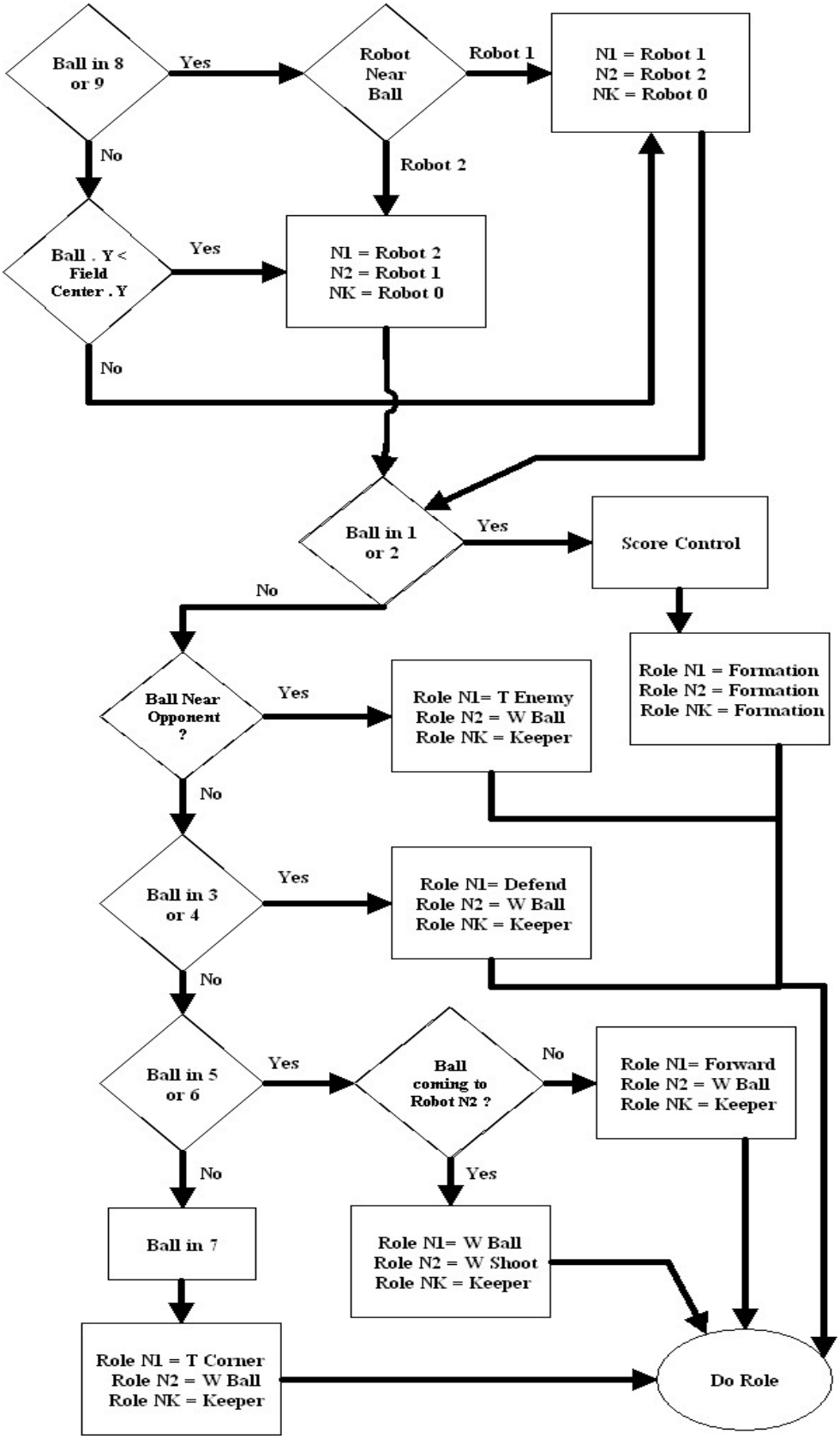

Figure 5.27 Director Level Strategy Decision

5.4 Some used Procedures and Structures

5.4.1 Robot and Ball Program Structures

Two Robot Structures are used; one is shown is Figure 5.28 all of the robots in field have this structure (also opponent team members). Three color inputs (Yellow, Magenta, Blue) are used for robot identification as it is considered in previous sections. Center is a point of pixel in the field which is the center of the robot in 240x320 pixel format. Center B is considered as Blob number that covers Robot center which is the place in 30x40 Blob space, this center is used for path planning and in potential updates. Field place is used for strategy and it's the place of the robot in strategic field part Figure 5.26. Angle element is the Robot's direction angle which is found by front color blobs. Ball angle is the angle between the ball and the robot, this identification is commonly used in Do angle behavior. New Ball angle is similar to Ball angle it uses future position of the ball and calculates the angle difference between the Robot direction angle and Future ball center. Goal angle is the angle between the opponent goal and the robot this is used for Wait Ball and Wait shoot roles. Active instruction is used when Robot is identified by vision system, if Robot's colors blobs can't be recognized active will be false (Because of lightning).

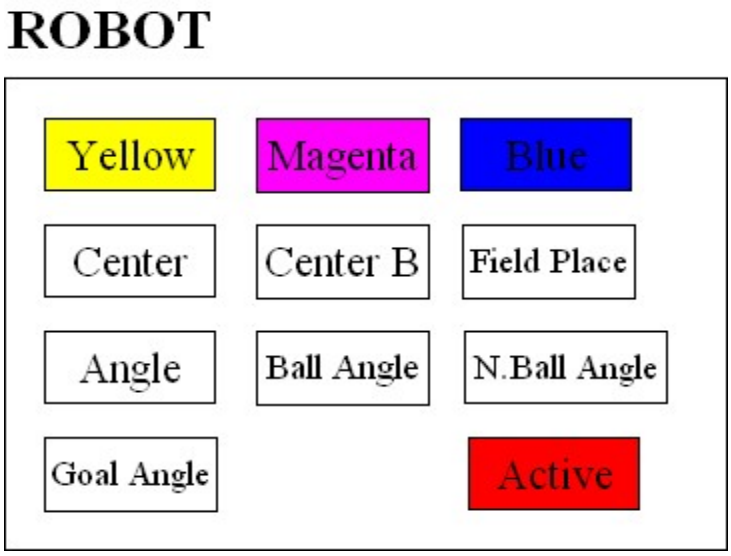

Figure 5.28 Robot structure

Second robot structure is build for robot work coordination which is shown in figure 5.29. According to figure 5.29 “Behavior and Role identifications” are activated when Robot is doing a Work or a Role. “Time behavior” and “Time role” identifications are used for time calculations which hold the process times of role and behavior. “Robot Direction” is used for Robot way selection, this value is sent to Rf communication. “Robot Power” is used for Robot dc motor power, this value is sent to Rf communication. “Robot Direction and Robot power” are the final part of the strategy Robot will move according to these values. “Line equation” is updated by two points which are the Robot center and opponent goal center, this equation is used for Wait Shoot Role. “Behavior and Role selection arrays” are loaded with roles and behaviors which are currently in process, these arrays used in to Behavior and Role Managers.

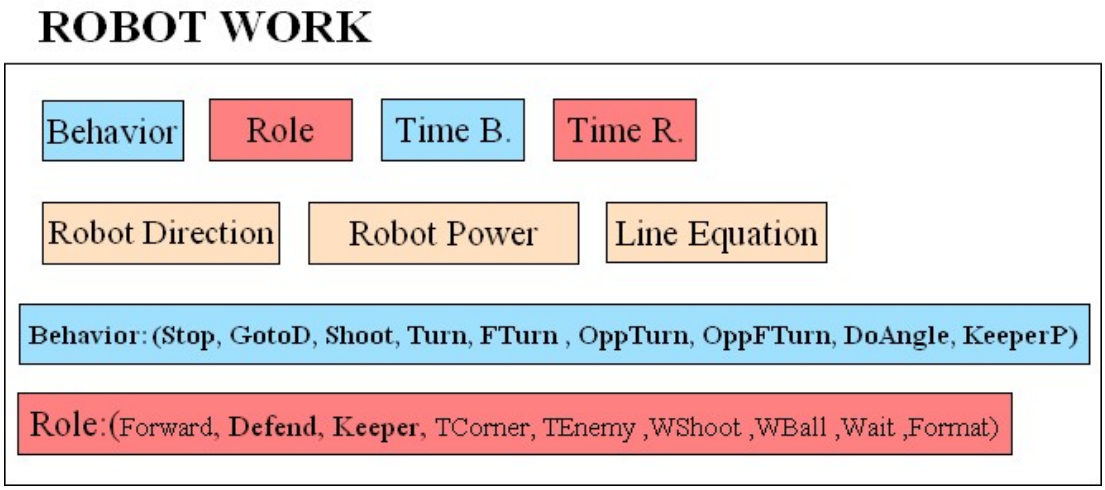

Figure 5.29 Robot Work Structure

Ball structure is used by whole roles and in strategy Figure 5.30. “Center” shows the position of the ball in 240x320 pixel space. “Center Blob” shows the position in 30x40 Blob space. “Field place” is used for strategy and it’s the place of the ball in match field (Figure 5.26). “Old center” is used for calculating the Ball equation. “New center” is the future position of the ball (1 second later). “Equation” is calculated with two points which are “old center and center” this equation gives the direction of the ball. “Velocity” of the ball is calculated with time and displacement difference. “Angle” is the inclination for the ball equation which is the ball

movement angle. “Stuck” Boolean is set when ball is stuck for 2 seconds. “Active” instruction is used when ball is lost by Vision system, if ball can’t be recognized by camera “Active” will be false.

All these structures are used in strategy. These structures are global valuables and all subroutines and Roles can use these structures efficiently.

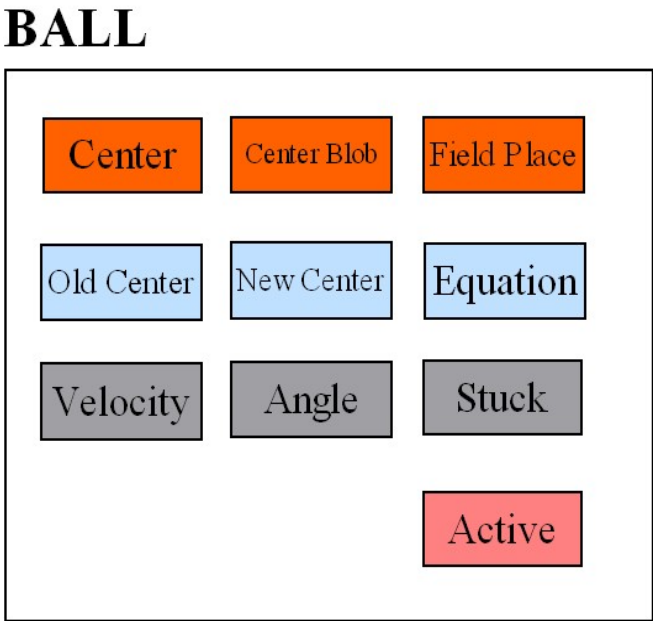

Figure 5.30 Ball Structure

5.4.2 Linear Line Equations

In strategy program two line equations are used; one is used for calculating the line equation with two points Home team robots and also the ball have a direction and a line equation which will be used in some Roles (Forward, Defend). Second subroutine finds the intersection of two lines this point is used in wait shoot role. Program calculates the interception point of the ball and robot’s front.

5.5 Experimental Results

The behavior based strategy is (Prof. Dr. Thomas Bräunl and Birgit Graf, 1999) and (Manuela Veloso and others, 1998) used efficiently in strategy. The main idea of the Role selection is demands on the place of the Ball, Team Robots and Opponent Robots. Robots have nine roles which can command nine simple behaviors. These behaviors are consisted of simple robot movements and combinations of these behaviors can handle some complex abilities. Role selection is the second level of strategy and the idea of role usage consists of the combinations of behaviors. Behavior and Role managers are used for controlling time periods of these robot movements, these periods are so important for switching behaviors and roles. Time periods are important for strategy, otherwise in some conditions robots may change roles and behaviors so often so quickly, by this idea strategy stabilize the Role and Behavior switching. For robot soccer it is hard to build a specific strategy, there are many conditions that robots have to be handle. In this project robots can handle nine strategic conditions; keeping ball, taking ball from corner, forward, defend, take ball from enemy, wait for ball, wait ball for shooting, take formation and waiting, all these conditions are handled efficiently. Wait ball for shooting role is the hardest shooting ability in this project. Robot will change its conditions for shooting to hopping ball. Passing is an important role for strategy but because of the field size (so small for passing) and robot movements this Role will not work properly. For an example robot has to shoot for passing but in experimental results it has seen that robot will not shoot so fast and will fallow the ball for 500 ms this will make passing impossible field is so small for this condition.

CHAPTER SIX

CONCLUSION

In this study, a cellular automata approach for solving robot path planning problem which requires an efficient experimental performance on robocup type multiple robot path planning applications. This work presents a design and implementation of the real-time vision system and required multilevel strategy and simple behavior algorithms.

In this thesis the necessary hardware and software developments are prepared for a group of mobile robot vehicles to participate at RoboCup. Robotic soccer is a highly complex domain that has become a significant challenge problem in both mobile robotics and artificial intelligence. In this project a global vision, path planning algorithm, behavior based control for strategy and robot architecture for robot soccer is developed and carefully researched.

The main idea in this system is to design an algorithm in an dynamic environment which includes multiple robots. User robots can be managed according to positions of the team members, the ball and the opponent robots.

Identification process is so critical which depends on colors and lightning, in some conditions to minimize these faults it is preferred to use a constant lightning system. This system consist of four lamps which place top of the filed. In the beginning of the project instead of Magenta a Green Blob is used but because of some identification problems (lightning) Green color is changed with magenta. Color classification and robot identification handle efficiently but in real life Vision program has to handle some other problems, lightning is one of the biggest problem in vision system. Light conditions can change in all day (morning, evening and night). All these changes need some calibrations and because of these changes sometimes identification process can’t work well. Vision system also needs to handle some lens distortions these nonlinearities are handled by some program calculations. After these processes color blobs identification process is done efficiently.

A cellular automata approach (M. Bracho and others 2001), (C. Behring, M. Bracho, M. Castro & J. A. Moreno, 2000) for solving the robot path planning problem that yields very efficient experimental performance on real life path planning situations is proposed. The cellular automata algorithms were tested with different workspace configurations and cellular space sizes over real time images from a digital camera. The computer effort depends on the size of the cellular space and the length of the resulting path. Path planning algorithm has two parts these are the Cellular automata for Potential fields and the Path planning algorithm for choosing the correct path; after these two parts Robot program structure is loaded with way and power information. In real soccer field Robots reach their destination efficiently and with minimum distance error, but of course some collusions may occur with obstacles. Controlling Robots movements is a hard job because of the voltage changes in batteries. At low power all calibration will be changed.

The behavior based strategy is (Prof. Dr. Thomas Bräunl and Birgit Graf, 1999) and (Manuela Veloso and others, 1998) used efficiently in strategy. The main idea of the Role selection is demands on the place of the Ball, Team Robots and Opponent Robots. Robots have nine roles which can command nine simple behaviors. These behaviors are consisted of simple robot movements and combinations of these behaviors can handle some complex abilities. Role selection is the second level of strategy and the idea of role usage consists of the combinations of behaviors. Behavior and Role managers are used for controlling time periods of these robot movements, these periods are very important for switching behaviors and roles. Time periods are vital for strategy, otherwise in some conditions robots may change roles and behaviors so often and so quickly. By this idea strategy stabilizes the Role and Behavior switching. For robot soccer it is hard to build a specific strategy, there are many conditions that robots have to handle. In this project robots can handle nine strategic conditions; keeping ball, taking ball from corner, forward, defend, take ball from enemy, wait for ball, wait ball for shooting, take formation and waiting, all these conditions are handled efficiently.

The performance of the system is based on the response of the algorithm in the dynamical system. For this purpose, a self-tuned and robust system is aimed to design. As a conclusion, the practical performance is satisfactory, however, further studies including more improvements and new strategies are necessary.

REFERENCES

Luis A. Martinez-Gomez, & Alfredo Weitzenfeld. (2004). *Real Time Vision System for a Small Size League Team*. Comp. Eng. Dept, ITAM

James Bruce, & Manuela Veloso.(2003). *Fast and Accurate Vision-Based Pattern Detection and Identification.* Computer Science Department Camegie Mellon University.

Chunmiao Wang, Hui Wang, Han Wang and William Y.C. Soh.(2001). *The Vision System For Robocup Small Robot League*. Division of Control & Instrumentation, School of EEE, Nanyang Technological University.

C. H. Messom, S. Demidenko, K. Subramaniam, & G. Sen Gupta. (2002). *Size/Position Identification in Real-Time Image Processing using Run Length Encoding.*

M. Bracho, M. Castro, & J. A. Moreno. (2001) . *A Robotic Architecture for RoboCup*

C. Behring, M. Bracho, M. Castro, & J. A. Moreno.(2000). *An Algorithm for Robot Path Planning with Cellular Automata*

John Anderson, & Jacky Baltes. (2004). *Agent-Based Control In A Global-Vision Robotic Soccer Team.*

Thilo Weigel, Jens-Steffen Gutmann, Markus Dietl, Alexander Kleiner, Bernhard Nebel. (2001). *CS Freiburg: Coordinating Robots for Successful Soccer Playing.*

Ching-Chang Wong, Ming-Fong Chou, Chin-Po Hwang, Cheng-Hsin Tsai, & Shys-Rong Shyu. (2001). *A Method for Obstacle Avoidance and Shooting Action of the Robot Soccer.*

T. H. Lee, H. K. Lam, F. H. F. Leung P. K. S. Tam. (2001). *A Path Planning Method For Micro Robot Soccer.*

Jacky Baltes, & Robin Otte. (1999). *A Fuzzy Logic Controller for Car-like Mobile Robots.*

Prof. Dr. Thomas Bräunl, & Birgit Graf. (1999). *Diploma Thesis No. 1700 Robot Soccer.*

Bernhard Nebel, & Yuliya Babovich. (2002). *Goal-Converging Behavior Networks and Self-Solving Planning Domains.*

Manuela Veloso, Peter Stone, & Michael Bowling. (1998). *Anticipation: A Key for Collaboration in a Team of Agents.*

H. K. Lam ,T. H. Lee, F. H. F. Leung P. K. S. Tam. (2001). *Decision Maker for Robot Soccer.*

T. H. Lee, H. K. Lam, F. H. F. Leung, P. K. S. Tam. (2001). *A Fast Path Planning and Tracking Control for Wheeled Mobile Robots.*