Ball Tracking Robot

by

TAN YENG LEE

FINAL YEAR PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme

in Partial Fulfillment of the Requirements

for the Degree

Bachelor of Engineering (Hons)

(Electrical & Electronics Engineering)

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Ball Tracking Robot

By

Tan Yeng Lee

A project dissertation submitted to the

Electrical & Electronics Engineering Programme

Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

Bachelor of Engineering (Hons)

(Electrical & Electronics Engineering)

Approved by,

[illegible]

(Mr. Patrick Sebastian)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

January 2012

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

TAN YENG LEE

ABSTRACT

Ball Tracking Robot is a project covering robotic, computer vision and image processing, microcontroller, and data communication. It is programmed in such a way that web camera will detect both ball and robot in the field, analyzing and determining the position of the ball with reference to the robot, where this information will be transmitted to the microcontroller. Microcontroller will receive the data and the robot will move towards the location of the ball.

The objective of this project is to apply knowledge learnt in image processing and microcontroller courses into practical. By using Matlab software, image captured by webcam will be analyzed and the ball and robot will be detected. Next, vector between ball and robot will be processed to determine the position of the ball with reference to the robot. Finally, output data will be transmitted to the microprocessor to guide the robot towards the ball.

For image processing part, HSI colour detection is applied to differentiate the targeted colour object from the surrounding. Since we are dealing with vector and direction, robot head position is vital for precise robot movement. For wireless data transmission, parallel data radio frequency (RF) wireless transmission model is used as medium for data to be transmitted from transmitter toreceiver located on the robot. Microcontroller is programmed in the way that it will response to the data received and direct the robot to move towards the ball.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Computer vision and image processing technology is widely used recently for research and development purposes. An image, such as photograph or video frame, is analyzed and processed with software to generate output either in image form or other parameters. For instance, ball tracking robot relies heavily on image processing to locate the position of ball so that robot is able to move towards it.

Ball tracking robot project can be broken down to two parts. The software part is about the ball detection using software such as MATLAB, with specific algorithms designed to locate and track a ball during the process. Based on these algorithms, the system will accurately distinguish between stationary and moving objects in a stream of video frames and is able to consistently detect which objects is the targeted ball and robot. A webcam is placed on top of the field, facing down and capturing the view of the field. The system should be able to detect both ball and robot all the time, obtaining the direction for the robot movement precisely, and transmit signal to move the robot towards the ball. Overall, knowledge of colour filtering, edge detection, background subtraction is applied.

After analyzing the video frame and locating the position of both the ball and the robot, instructions are generated in MATLAB to enable to lead the robot towards the location of the ball. On the other hand, a microcontroller is placed on the robot, where it is programmed to read and run the program generated by MATLAB. In this project, we can either use assembly language or C programming language to program the microcontroller, but C programming is preferred, as it is easier to understand and master. After the correct output signals are generated, the signals will be sent to the robot using wireless connection.

1.2 Problem Statements

In sport based competitions, there is need to locate the exact position of the ball. This is because it is hard for human eye to identify whether a soccer ball crosses the goal line, or whether a baseball ball is pitched outside or inside the strike zone. This leads to researches on implementing computer vision and image processing in determining the location of the ball, with technology such as *hawk eye* technology widely used in tennis games nowadays to calculate the outcome of the ball landing accurately.

In even advanced category, many algorithms were written and tested to detect and locate other specific objects, which contributing not only in sports, but also others fields such as security and manufacturing. Later on, merely object detection is not sufficient, as people request to have the system analyzing and responses to the output results. Microcontrollers are brought in to complement with the image processing, to be able to control the machines or robot to complete certain tasks depending on the situation. Microcontrollers can be designed in such a way that they are controlled manually, or automatically based on the data retrieved from the programs itself. Applications of image processing and robot controlled manually are bomb disarming robot and emergency rescue robot, while plenty others products are designed to run automatically, such as ball tracking robot.

For ball tracking robot, challenges faced are problems related to occlusions, shadows, and real time processing. Existing researches are done in such a way that the webcam is placed on the robot, which gives user a first person viewpoint; however in this project, the webcam will be placed on top of the field, just like a live broadcast viewpoint, where the webcam captures the images continuously, detecting both the ball and the robot, and lead the robot towards the ball. For robot games, it is extremely vital for the correct vector calculation between ball and robot, as an error in analysis will lead to huge different in robot response. This is because all the decision makings are preprogramming and automated.

1.3 Objective

This report studies the methods and algorithms used for ball tracking as well as locating the robot and calculating the distance between the two objects to achieve the following objectives:

1. To accurately identify and locate the position of the ball and the robot.
2. To identify the vector between the ball and the robot precisely.
3. To transmit vector data to robot to intercept ball as it moves in target area

1.4 Scope of Study

The scope of the research works is summarized as follows in order to achieve the objectives within the time frame and funds allocated:

1. The focus of the study is based on *Ball Tracking Robot*, where moving the robot towards the ball automatically is prioritized.
2. MATLAB software is used as the part of the image processing, where algorithms are designed and tested to detect the ball and the robot.
3. After vector between the ball and the robot is calculated, data is programmed so that it will be sent to the robot to move towards the ball automatically.

CHAPTER 2
LITERATURE REVIEW

2.1 Circle Detection

T. Orazio et al. [1] stressed on edge detection to detect and identify the ball from the video frame. Circle Hough Transform (CHT) is implemented to find circular patterns of a given radius R within an image. This edge algorithm can be formulated as convolutions applied to an edge magnitude image Circle detection operator applied over all the image pixels produces a maximal value when a circle is detected with a radius in the range [R_{min}, R_{max}]. *Hongying Zhang, et al.* [2] used Hough transform to locate the possible center pixels of the balls. After the center of the ball is targeted using Hough Transform, the entire ball can be easily computed using weighted average of the center pixel with its 8-neighbourhood pixels about their gray world normalization values.

2.2 Colour Detection

Xiao-Feng Tong et al. [3] showed that colour detection is another alternative for ball detection. Field extraction can be done with HSI (Hue, Saturation, and Intensity) colour space. Those background sections are set with dark colour. After field region extraction, region analysis can be done with only the objects in it need to be considered and examined. A coarse-to-fine search strategy is used to identify a unique ball as well as the robot. Finally, colour evaluation is applied to get the objects targeted. *Jusoh, R.M.* [4] captured single frame image every second in RGB (Red, Green, and Blue) format. Image processing starts with color thresholding where pixel values below the threshold value will be treated as black colour. Colour thresholding can easily separate the targeted objects from its background.

2.3 Wireless Data Transmission

Manigandan, M et al. [5] focused on the implementation of a wireless Mobile Robot control with Object detection based on coordinates and to process the images using MATLAB. To transfer the data to mobile robot, serial port COM is used for transferring the data generated from MATLAB based on the position of the detected object byte by byte. Wireless control eliminates the constraint in distance between the color object and robot for better tracking from remote location. The serial / parallel binary data is received from the computer. The data is transferred to the microcontroller. The receiver receives the transmitted data and is sent to the decoder IC. The decoder converts the serial data into parallel data and is transmitted to the microcontroller. The microcontroller drives the motor to perform the function based on data received.

2.4 Robot Movement Control

Alauddin Al-Omary [6] broke down the step by step instruction on how to control the robot automatically based on the data programmed in microcontroller. With the decision made, the robot can move forward, backward, or turn to left or right. The robot can be programmed to stop in front of the ball automatically when it reached the location of the ball. PIC 16F and 18F Series are example of microcontrollers that can be used where we can either program it in assembly language or C programming language.

CHAPTER 3
METHODOLOGY

This project can be categorized into three main parts:

1. Matlab Analysis
2. Serial Port Data Communication
3. Microcontroller Programming

3.1 Matlab Analysis

3.1.1 Colour Detection

Colour detection is preferred in this project as colour, when converted to digital value, can be detected and identified clearly. HSI (*Hue, Saturation, Intensity*) image is processed to further highlight the particular colour assigned to both robot and ball. Compared to RGB (*Red, Green, Blue*), HSI works better for those looking for specific colour that does not have matches in the database, with the brightness and dilution taken into account. With the knowledge in image processing, HSI algorithm can be constructed to detect certain colour range. For example, if the ball is green in colour, we can easily locate the ball in the image captured by highlight pixel with hue value range in between 120 ±10% to be white, while other out of range pixels black. (Noted: Hue value for pure green is 120, with the ±10% as the tolerance for the green colour to be recognized.) Same goes to the *Saturation* and *Intensity*, where we can modify based on the dilution and brightness level.

3.1.2 Noise Reduction

After correctly setting the HSI range for the ball and robot to be recognized, there will still be some noise inserted in the output image. Noise is the random unwanted signal that might affect the quality of image processing. As a result, noise reduction is a must in image processing to enhance the quality of the project. Most common noise happened in webcam image/video capturing is the salt and pepper noise. Erosion masking is applied to help to generate more accurate output result.

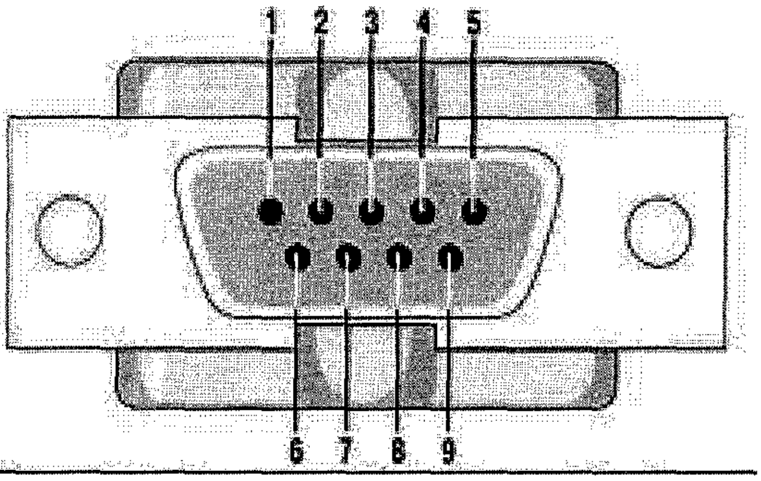
3.1.3 Position Identification and Direction Analysis

With two separate output results generated (one for the ball detection and the other one for robot detection), the pixel location for both ball and robot are recorded. Those pixel location is important as they are used to determine which direction the robot will move so that it can reach the ball. Six possible robot movements are identified, where the robot can move forward, backward, left, right, or even 45 degree to front or back, -45 degree to front or back, and last but not least, stop when it detect that the ball is a few pixel near the robot. In this project the time of the robot movement is a non-factor as the web camera will get a snapshot continuously, for example, every 3 seconds. Such real time data analysis allows us to be able to exclude the delay time for the robot movement as long as we are able to identify the location for ball and robot and manage to determine on which direction the robot need to move to approach the ball. This algorithm is also applicable to both static and moving ball. This is because MATLAB will always update the latest location of the ball and robot, and will lead the robot towards the current location of the ball.

3.2 Serial Port Data Communication

3.2.1 Serial Port Communication

Serial port communication is a common communication protocol used to interact between MATLAB and microcontroller. Serial port is easier to implement as it transmits multi-bit word bit after bit (when at any given moment only one bit will pass). Wired and wireless serial port communication will be tested out before setting up the whole transmission module. The common serial port used consists of 9 pins, with function of each port shown below:



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

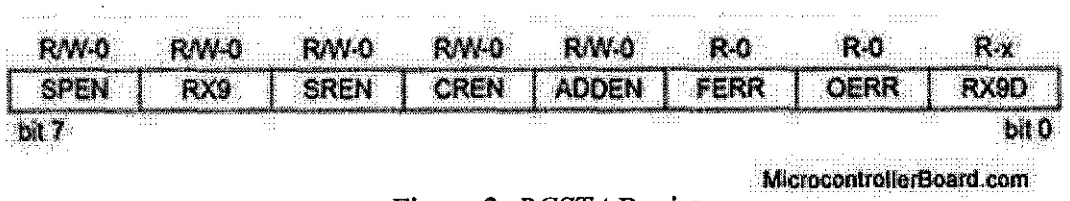
Figure 1: Serial Port Diagram

3.2.2 Serial Port Hardware Connection

Since this project requires communication between a microcontroller and a PC (sending output which consists the direction for the robot to move toward the ball from MATLAB in PC to a microcontroller), differences between PC and microcontroller need to be taken into consideration. PIC microcontroller has ***Universal Synchronous Asynchronous Receiver Transmitter (USART)*** which operates using CMOS logic levels changing between +5V and 0V to represent logic 1 and 0. However computer serial port (RS232C) on the other hand, operates in different voltage levels. It represents logic 0 with -10V and logic 1 with +10V. Direct connection between microcontroller USART pin to PC is not allowed due to the defense in logic voltage level. As a result, specially designed serial level converter ICs such as MAX232 is used to convert signals from -10 to +10V received from computer side into 0 and 5V which can be used with microcontrollers. The schematic of this serial port data communication connection can be viewed in **Appendix A.**

3.2.3 Microcontroller Asynchronous Serial Data Communication

Since there is no a full synchronization between the transmitter (PC), that sends the data, and the receiver (microcontroller), that receives the data, asynchronous serial data Communication is applied. As mentioned earlier, PIC microcontroller uses ***Universal Synchronous Asynchronous Receiver Transmitter (USART)*** to communicate with external components such as computer. Microcontroller can be either transmitter, receiver, or both. However, in this project microcontroller functions merely as receiver to get the direction input from the PC and response by moving towards the robot, only *Receive Status and control register* (*RCSTA)* register need to be enabled.



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

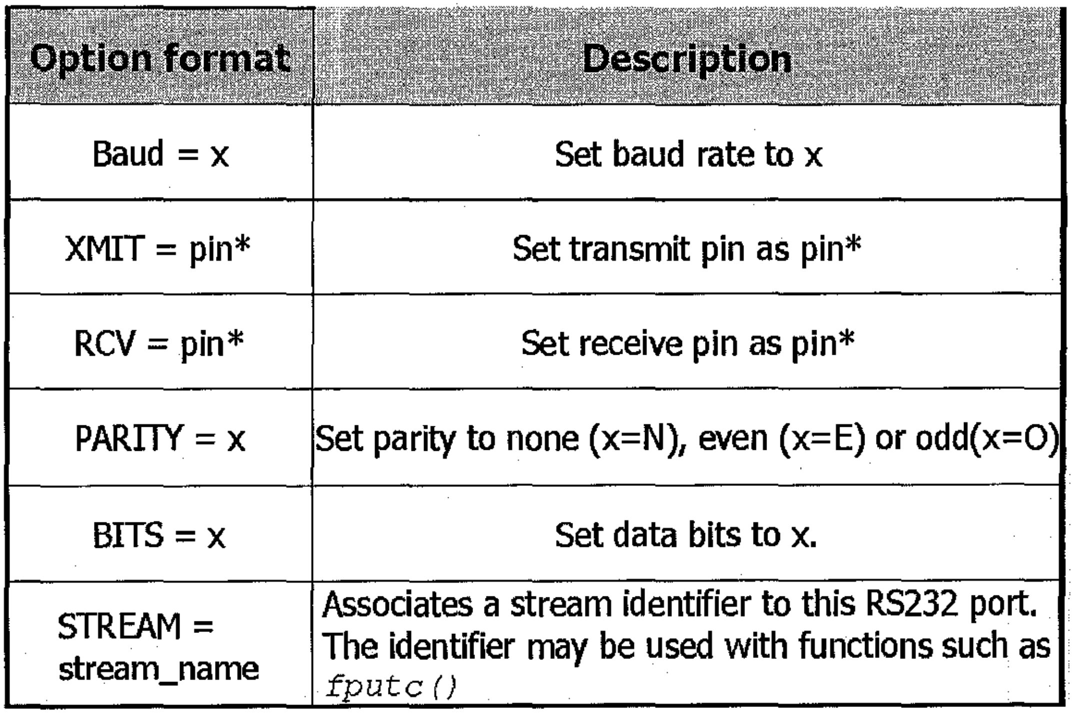
MicrocontrollerBoard.com

Figure 2: *RCSTA* Register

3.3 Microcontroller Programming

3.3.1 Asynchronous Serial Data Communication Configuration

For hardware serial data communication in PIC microcontroller, port C6 is assigned for transmitter (TX) while port C7 is assigned as receiver (RX). Before that, RS232 (Recommended Standard 232) port is called with command *#use rs232 [option].* RS232 topology is a point-to-point protocol mainly used for serial port communication. The option for the PIC microcontroller *#use rs232 [option]* command are as below:



Option format	Description
Baud = x	Set baud rate to x
XMIT = pin*	Set transmit pin as pin*
RCV = pin*	Set receive pin as pin*
PARITY = x	Set parity to none (x=N), even (x=E) or odd(x=O)
BITS = x	Set data bits to x.
STREAM = stream_name	Associates a stream identifier to this RS232 port. The identifier may be used with functions such as `fputc()`

Figure 3: *#use rs232* Command

Example of hardware based *#use rs232* command is provided:

#use RS232 (baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7, stream=, bits=8)

In our project, since PIC microcontroller functions only as receiver, only PORT C7 is needed.

3.3.2 Microcontroller C Language Programming

C language can be used to program a PIC microcontroller. In this project, microcontroller will receive data sent by MATLAB and response toward the ball based on the input received. PIC C Compiler software is used for programming purpose. The program is built and compiled for error checking. After that a HEX file will be generated. The HEX file can be loaded in PIC Simulator IDE to test its functionality in simulation mode. With programmer we can load the HEX file into microcontroller and can be tested in hardware.

3.4 Tool

Since this is a project combining both image processing and microcontroller programming in robotic field, plus data communication in serial port data communication, the list of the tools required is listed as below:

1. MATLAB 7.10.0 (R2010a)
2. Web Camera
3. Serial Port with MAX233 Chip
4. Radio Frequency (RF) Transmitter / Receiver Module
5. Robot
6. PIC 16F877 Microcontroller

3.5 Project Flow Chart

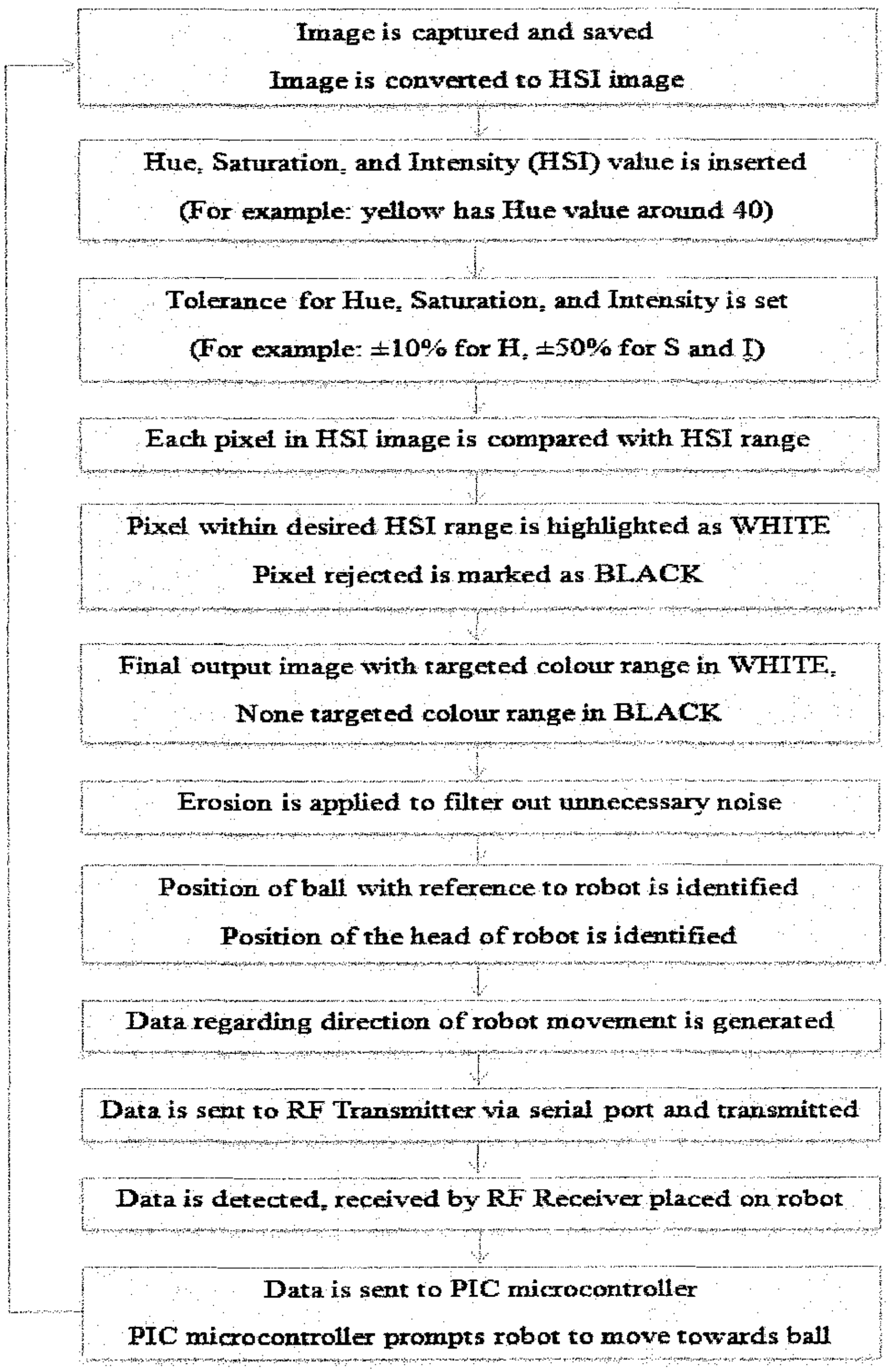


Image is captured and saved
Image is converted to HSI image
Hue, Saturation, and Intensity (HSI) value is inserted
(For example: yellow has Hue value around 40)
Tolerance for Hue, Saturation, and Intensity is set
(For example: ±10% for H, ±50% for S and I)
Each pixel in HSI image is compared with HSI range
Pixel within desired HSI range is highlighted as WHITE
Pixel rejected is marked as BLACK
Final output image with targeted colour range in WHITE,
None targeted colour range in BLACK
Erosion is applied to filter out unnecessary noise
Position of ball with reference to robot is identified
Position of the head of robot is identified
Data regarding direction of robot movement is generated
Data is sent to RF Transmitter via serial port and transmitted
Data is detected, received by RF Receiver placed on robot
Data is sent to PIC microcontroller
PIC microcontroller prompts robot to move towards ball

CHAPTER 4
RESULT AND DISCUSSION

4.1 Colour Detection

Accepted range of the HSI value will be highlighted. Those accepted pixel will be changed to white colour while other rejected pixel will turn out to be black colour.

Table 1: HSI Colour Detection Algorithm Flow Chart

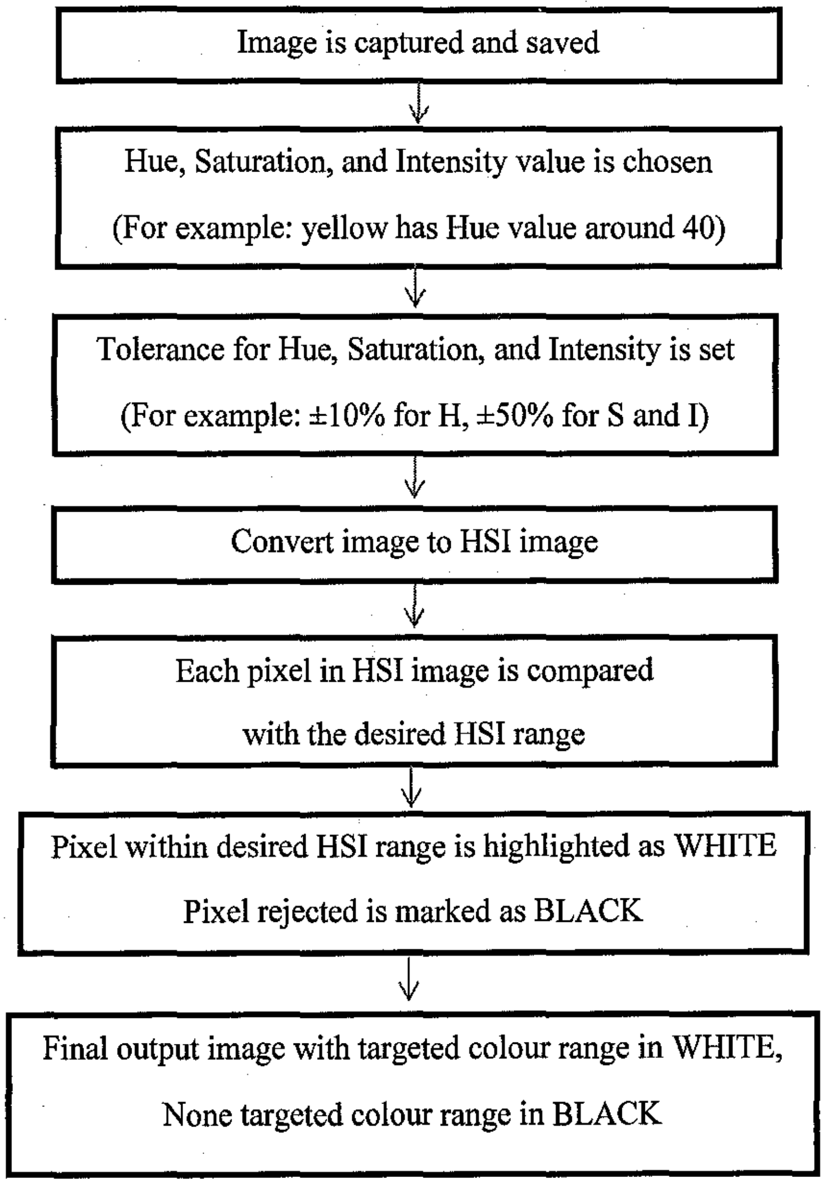


Image is captured and saved
Hue, Saturation, and Intensity value is chosen
(For example: yellow has Hue value around 40)
Tolerance for Hue, Saturation, and Intensity is set
(For example: ±10% for H, ±50% for S and I)
Convert image to HSI image
Each pixel in HSI image is compared
with the desired HSI range
Pixel within desired HSI range is highlighted as WHITE
Pixel rejected is marked as BLACK
Final output image with targeted colour range in WHITE,
None targeted colour range in BLACK

Example of the HSI colour detection algorithm can be viewed in **Appendix B**.

Figure 4 shows an example where yellow colour detection algorithm is applied to detect area which match the ±10% tolerance of yellow colour. (Hue value for yellow: 720)

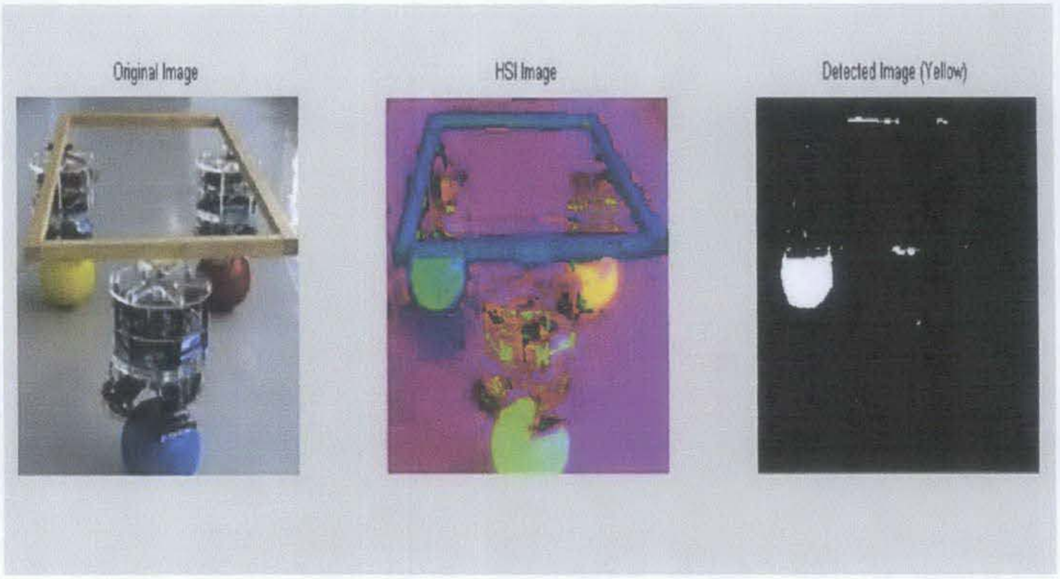


Figure 4: HSI Colour Detection Algorithm Output Image

With the Hue value = 0.2 ± 0.1 = 0.2*360 ± 0.1*360 = 72 ± 36. Those pixel with hue value in the range of [72-36 72+36] = [36 108] will be accepted and marked as while pixel. Other non-targeted pixel will be set as black colour. For information, [36 108] is the colour range in between YELLOW and LIGHT GREEN. There will always be some small non targeted areas that belong to those particular colour ranges. As noticed, there are some small portion which also get accepted. We need to get rid of this unwanted portion. One way to do it is to apply erosion method to squeeze the white area to get more accurate center point.

Erosion is a great method to get rid of some small selected area (unnecessary noise).

The code is as below:

```
SE = strel('disk',10);
g = imerode(I,SE);

subplot(1,2,1);imshow(I);title('Original Detected Area');
subplot(1,2,2);imshow(g);title('Image after Erosion');
```

The output image is shown in Figure 5 below:

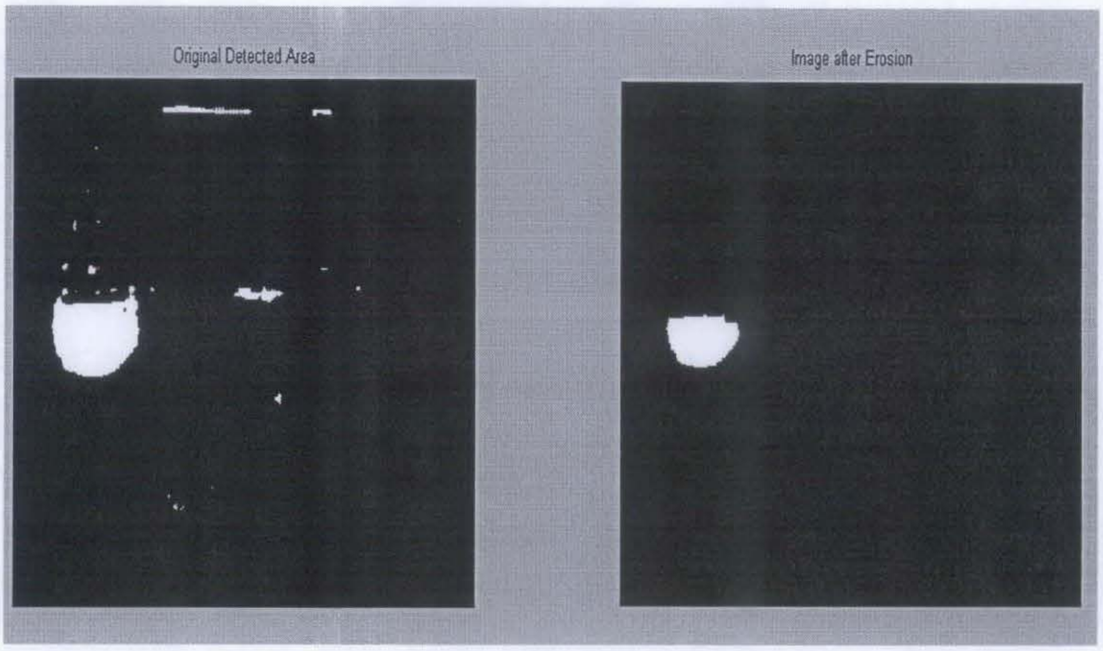


Figure 5: Noise Reduction Output Image

Erosion 'shrinks' or 'thins' object in a binary image. With the structuring element (SE) set by user, user can decide how much he/she would like to shrink the original image. Based on codes provided above, a disk shape SE with radius of 10 pixels is applied. It proves to be enough to delete those thin and random white spot notice on the original colour detection image (right) with the image after erosion (left) clearly show the exact location of the yellow ball.

Note: For real time image capturing and processing, more noise will appear. Common noise such as salt and peppers noise can be filtered using median filter. Thus, more filtering is needed for better data collection.

4.2 Vector Identification and Direction Analysis

With the same algorithm, robot can be detected by changing the hue value accordingly. After getting both robot and ball detected, the direction need to be analyzed and determined so that robot can move towards the ball.

First of all, save the pixel location of the location of ball and robot. Next, perform normal subtraction to get the pixel difference between the ball and robot. Let robot be the reference point, we can easily know which direction the ball is from the robot. Finally, for each particular direction, different bit is generated to send to microcontroller.

Assume yellow ball as the ball, while blue ball as the robot, perform the HSI colour algorithm, *I* as the detected image for ball (yellow); while *I* as the detected image for robot (blue). The detailed step by step instruction is shown in Table 3.

Table 2: Vector Identification and Direction Analysis Flow Chart

Location of ball is saved as [ballx bally]

Location of robot is saved as [robotx roboty]

↓

Robot as reference point,

Perform the subtraction of xy point of robot from ball

```
xdiff = robotx - ballx;
ydiff = roboty - bally;
```

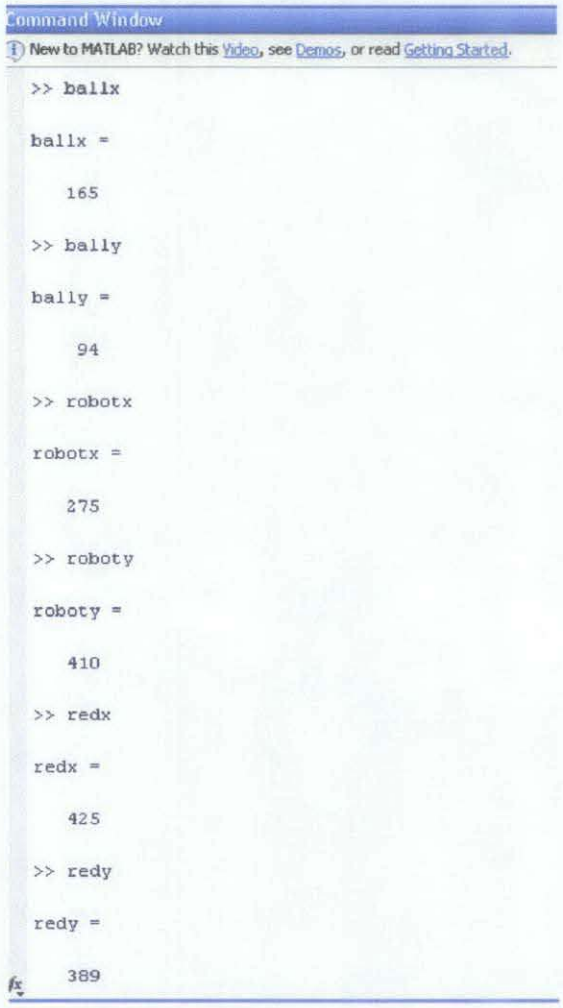
↓

Based on the result of subtraction,

Determine the vector / direction of the ball from the robot

Vector Identification and Direction Analysis algorithm can be viewed in **Appendix C**.

In command window:


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> ballx

ballx =

   165

>> bally

bally =

    94

>> robotx

robotx =

   275

>> roboty

roboty =

   410

>> redx

redx =

   425

>> redy

redy =

   389
```

Figure 6: Ball and Robot Location Display in Command Windows

We determine the direction of the ball from robot by *dir* (direction), where we check the x axis difference as well as y axis difference between ball and robot with robot as reference.

From info in Figure 6, we notice that ball is located at [165 94] while robot is detected at [275 410]. With y axis increases from left to right while x axis increases from top to bottom, we can easily conclude that ball is on top left of the robot, on the other hand, means that ball is located top left of the robot. (top left as in top left of the webcam)

There are four possibilities for *dir*, as shown in Figure 7 below:

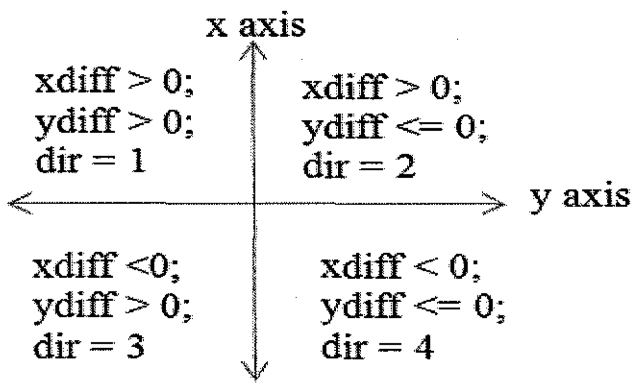


Figure 7: Analysis on Direction of The Ball from The Robot

After determining the location of ball from robot, we must then determine where the head of the robot is facing. This is crucial as sometime the robot's tail will face the ball instead of its head. At that point, moving the robot forward is actually pulling the robot further away from the ball. Thus two different colour sets are used to determine the head and tail of the robot. In this project, the head of robot will be covered by blue colour paper; while tail is covered with red colour paper. The same algorithm is applied, only that this time the vector difference between blue colour (robot head) and red colour (robot tail) is determined. The vector difference is stored, and the four possibilities are shown in Figure 8 below:

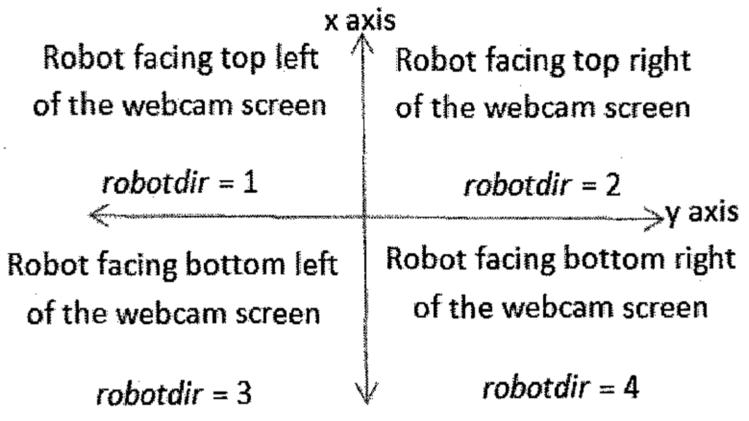

x axis
Robot facing top left of the webcam screen
robotdir = 1
Robot facing top right of the webcam screen
robotdir = 2
y axis
Robot facing bottom left of the webcam screen
robotdir = 3
Robot facing bottom right of the webcam screen
robotdir = 4

Figure 8: Analysis on Position of Robot Head

The step by step instruction on how to locate the position of robot head is shown in Table 4.

Table 3: Identification on Head of Robot Flow Chart

Location of robot head is saved as [robotx roboty]

Location of robot tail is saved as [redx redy]

↓

Robot head (blue colour) as reference point,

Perform the subtraction of xy point of robot head from robot tail

```
xrobot = redx - robotx;
yrobot = redy - roboty;
```

↓

Based on the result of subtraction,

Determine the position of robot head.

```
if (xrobot>0) && (yrobot>0)
    robotdir = 1; %robot facing top left corner
end

if (xrobot>0) && (yrobot<=0)
    robotdir = 2; %robot facing top right corner
end

if (xrobot<=0) && (yrobot>0)
    robotdir = 3; %robot facing bottom left corner
end

if (xrobot<=0) && (yrobot<=0)
    robotdir = 4; %robot facing bottom right corner
end
```

The position of the head of robot is vital as without identifying it, the robot might move away (opposite) form the robot instead of moving towards it. This is because all the image analyses are based on the webcam screen viewpoint.

As we know, we can set the robot to move on any of these six directions: forward, forward left, forward right, backward, backward left; and backward right. The final output, *sig*, which content the direction of robot movement is shown in Figure 9 below:

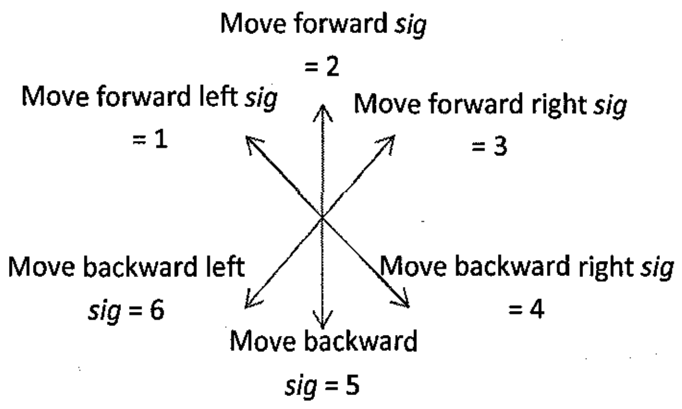

Move forward *sig* = 2
Move forward left *sig* = 1
Move forward right *sig* = 3
Move backward left *sig* = 6
Move backward right *sig* = 4
Move backward *sig* = 5

Figure 9: Analysis on Direction The Robot Move

Table 4 shows the output of direction of robot (*sig*) with respect to the location of ball from robot (*dir*) and the head of the robot facing (*robotdir*).

Table 4: Analysis on Direction The Robot Move Towards The Ball

robotdir	*dir* = 1	*dir* = 2	*dir* = 3	*dir* = 4
1	2 Forward	3 Forward Right	6 Backward Left	5 Backward
2	1 Forward Left	2 Forward	5 Backward	4 Backward Right
3	4 Backward Right	5 Backward	2 Forward	1 Forward Left
4	5 Backward	6 Backward Left	3 Forward Right	2 Forward

Final result of the complete HSI algorithm (Detected ball and robot image) is shown in Figure 10.

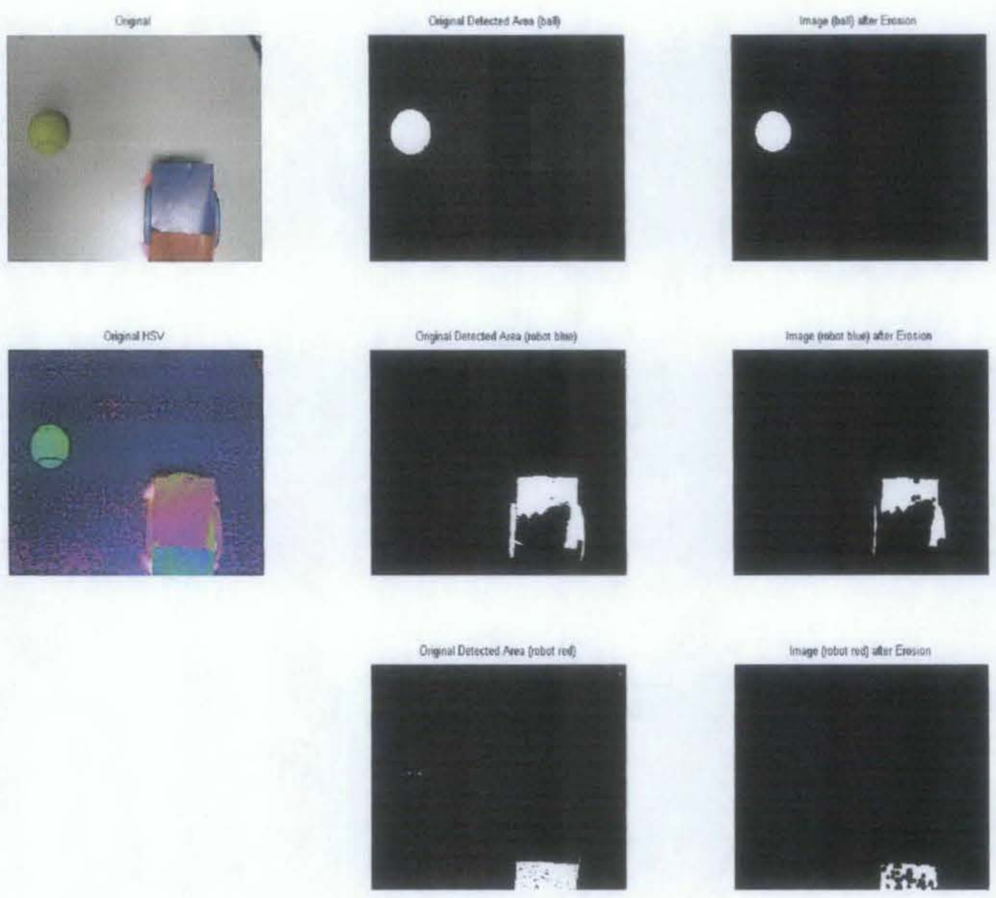
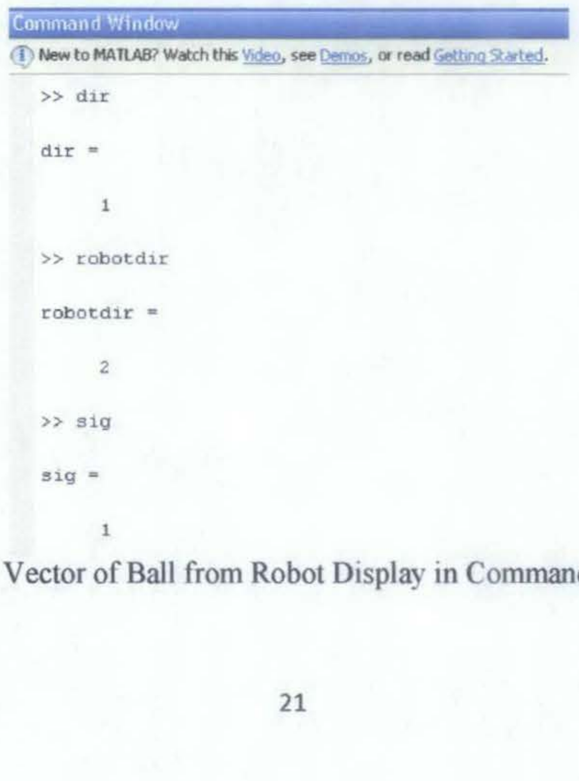

Original
Original Detected Area (ball)
Image (ball) after Erosion
Original HSV
Original Detected Area (robot blue)
Image (robot blue) after Erosion
Original Detected Area (robot red)
Image (robot red) after Erosion

Figure 10: Detected Ball and Robot Output Image


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> dir

dir =

     1

>> robotdir

robotdir =

     2

>> sig

sig =

     1
```

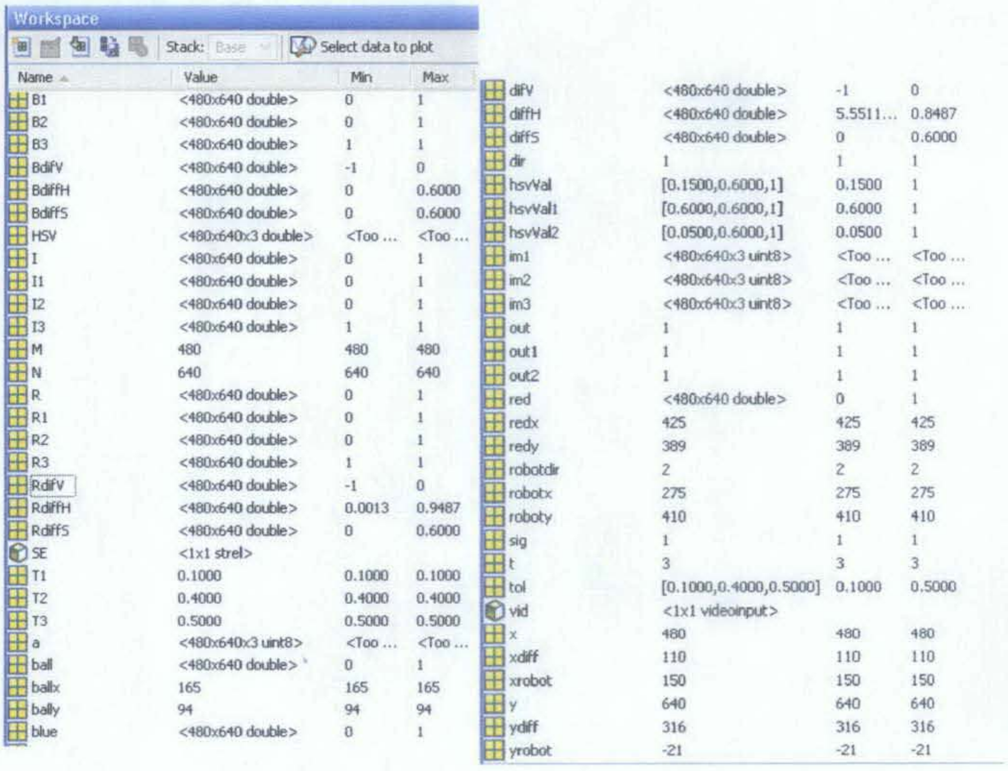
Figure 11: Vector of Ball from Robot Display in Command Windows

As shown in Figure 11, MATLAB manages to get *dir* '1', which means the ball is located on the top left of the robot (webcam view). This direction analysis works perfectly as long as the ball and robot can be detected correctly.

Next, by analyzing the blue (head) and red (tail) cover on the robot, we get *robotdir* = 2, which indicate the robot is facing top / top right on the screen. And finally, *sig* = 1 (move forward left) indicates that MATLAB manages to generate correct direction of the robot to move towards the ball.

Overall, since the two algorithms works perfectly in selected image, the only problem left is the unexpected image quality in real time image capturing. More filters and more checking are needed to enhance the accuracy of the data collection.

The data generated by MATLAB throughout the image processing process is shown in Figure 12:

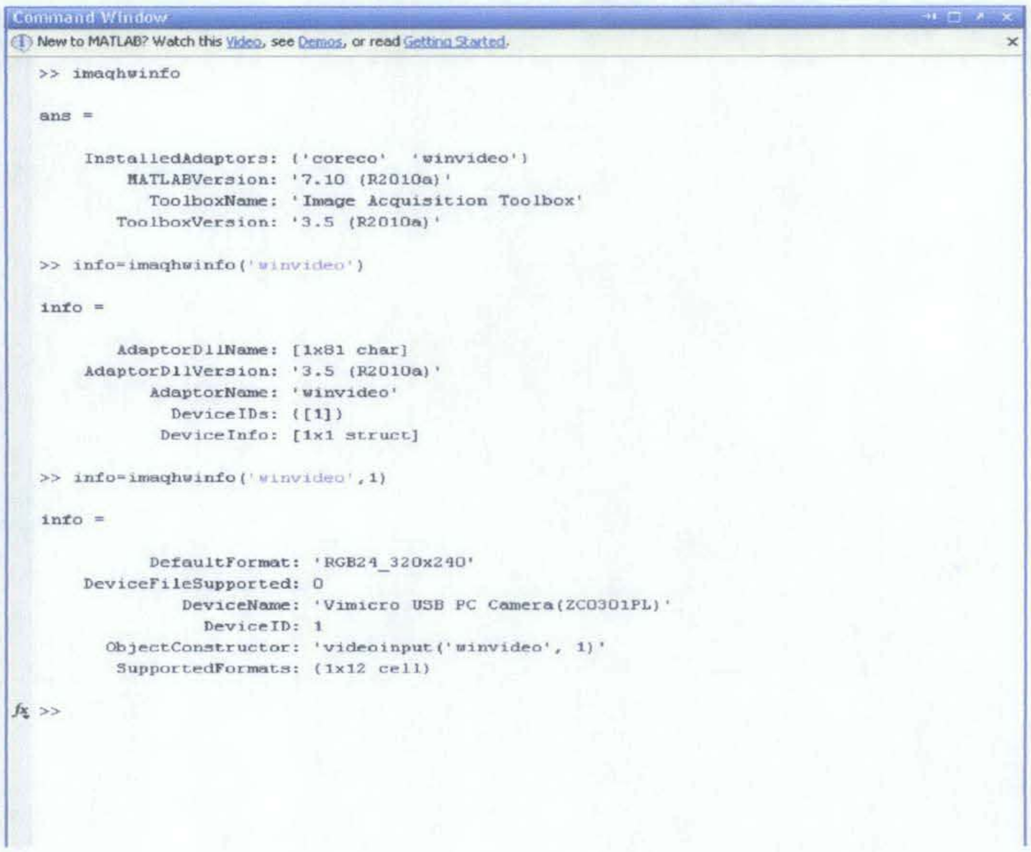


Name	Value	Min	Max
B1	<480x640 double>	0	1
B2	<480x640 double>	0	1
B3	<480x640 double>	1	1
BdifV	<480x640 double>	-1	0
BdiffH	<480x640 double>	0	0.6000
BdiffS	<480x640 double>	0	0.6000
HSV	<480x640x3 double>	<Too ...	<Too ...
I	<480x640 double>	0	1
I1	<480x640 double>	0	1
I2	<480x640 double>	0	1
I3	<480x640 double>	1	1
M	480	480	480
N	640	640	640
R	<480x640 double>	0	1
R1	<480x640 double>	0	1
R2	<480x640 double>	0	1
R3	<480x640 double>	1	1
RdifV	<480x640 double>	-1	0
RdiffH	<480x640 double>	0.0013	0.9487
RdiffS	<480x640 double>	0	0.6000
SE	<1x1 strel>		
T1	0.1000	0.1000	0.1000
T2	0.4000	0.4000	0.4000
T3	0.5000	0.5000	0.5000
a	<480x640x3 uint8>	<Too ...	<Too ...
ball	<480x640 double>	0	1
ballx	165	165	165
bally	94	94	94
blue	<480x640 double>	0	1
difV	<480x640 double>	-1	0
diffH	<480x640 double>	5.5511...	0.8487
diffS	<480x640 double>	0	0.6000
dir	1	1	1
hsvVal	[0.1500,0.6000,1]	0.1500	1
hsvVal1	[0.6000,0.6000,1]	0.6000	1
hsvVal2	[0.0500,0.6000,1]	0.0500	1
im1	<480x640x3 uint8>	<Too ...	<Too ...
im2	<480x640x3 uint8>	<Too ...	<Too ...
im3	<480x640x3 uint8>	<Too ...	<Too ...
out	1	1	1
out1	1	1	1
out2	1	1	1
red	<480x640 double>	0	1
redx	425	425	425
redy	389	389	389
robotdir	2	2	2
robotx	275	275	275
roboty	410	410	410
sig	1	1	1
t	3	3	3
tol	[0.1000,0.4000,0.5000]	0.1000	0.5000
vid	<1x1 videoinput>		
x	480	480	480
xdiff	110	110	110
xrobot	150	150	150
y	640	640	640
ydiff	316	316	316
yrobot	-21	-21	-21

Figure 12: MATLAB Workspace Data

4.3 Real Time Image Capturing

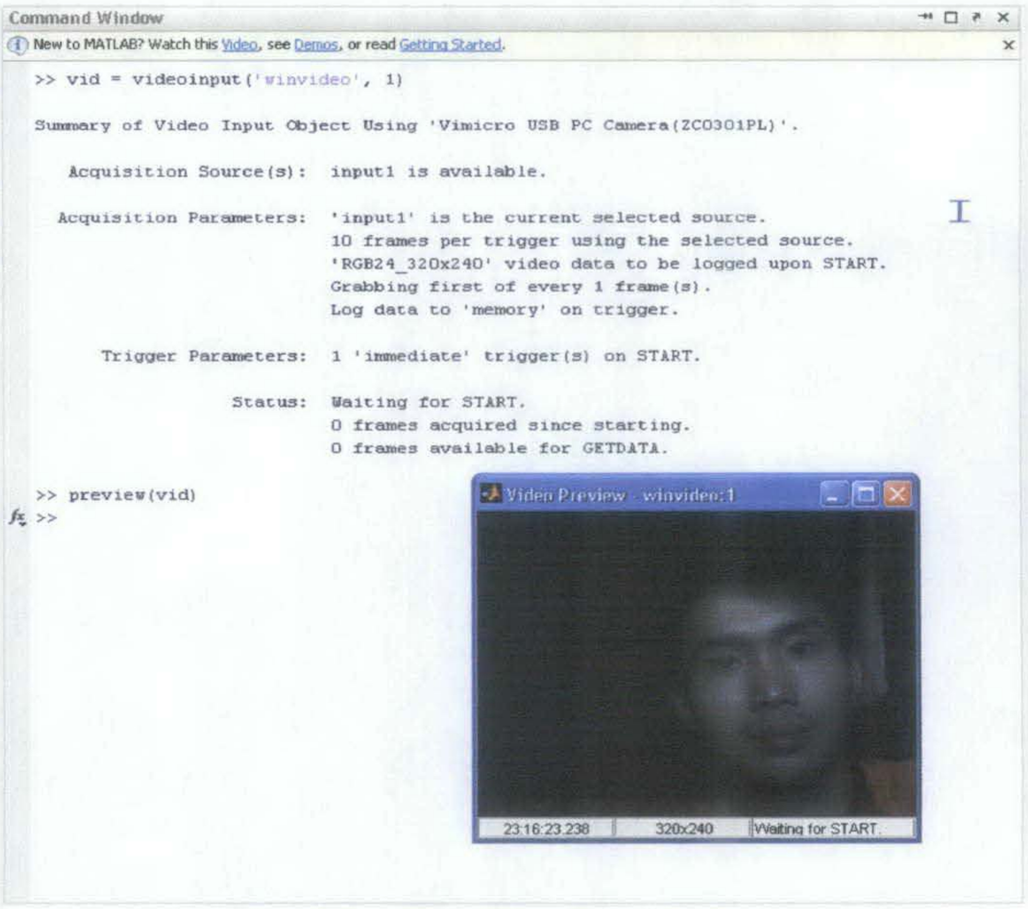
In MATLAB, there is video / image capturing toolbox available for real time image acquisition. The step by step procedure to enable the image acquisition toolbox is as below (Figure 13):


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> imaqhwinfo

ans =

    InstalledAdaptors: {'coreco'  'winvideo'}
        MATLABVersion: '7.10 (R2010a)'
          ToolboxName: 'Image Acquisition Toolbox'
       ToolboxVersion: '3.5 (R2010a)'

>> info=imaqhwinfo('winvideo')

info =

       AdaptorDllName: [1x81 char]
    AdaptorDllVersion: '3.5 (R2010a)'
          AdaptorName: 'winvideo'
            DeviceIDs: {[1]}
           DeviceInfo: [1x1 struct]

>> info=imaqhwinfo('winvideo',1)

info =

          DefaultFormat: 'RGB24_320x240'
    DeviceFileSupported: 0
             DeviceName: 'Vimicro USB PC Camera(ZC0301PL)'
               DeviceID: 1
      ObjectConstructor: 'videoinput('winvideo', 1)'
       SupportedFormats: {1x12 cell}

fx >>
```

Figure 13: MATLAB Image Acquisition Hardware Information

Image Acquisition Hardware Information (imaqhwinfo) allows us to check the available web camera. Format for the device determine the size of the web camera window.


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Video Preview - winvideo:1
23:16:23.238
320x240
Waiting for START.

```
>> vid = videoinput('winvideo', 1)

Summary of Video Input Object Using 'Vimicro USB PC Camera(ZC0301PL)'.

   Acquisition Source(s):  input1 is available.

  Acquisition Parameters:  'input1' is the current selected source.
                           10 frames per trigger using the selected source.
                           'RGB24_320x240' video data to be logged upon START.
                           Grabbing first of every 1 frame(s).
                           Log data to 'memory' on trigger.

      Trigger Parameters:  1 'immediate' trigger(s) on START.

                  Status:  Waiting for START.
                           0 frames acquired since starting.
                           0 frames available for GETDATA.

>> preview(vid)
>>
```

Figure 14: MATLAB Video Preview

In Figure 14, *videoinput* command start the summary of the video input (it is video stream collection, with *getsnapshot* command is needed to capture a single image). In this case we notice that the web camera 'Vimicro USB PC Camera' is identified. A video preview with exactly same size as the default format will pop out when *preview(vid)* command is called.

With the HSI colour detection algorithm discussed in Section 7.1 applied, we manage to get yellow ball detected area, but the quality is low as real time processing always encounters unwanted noise.

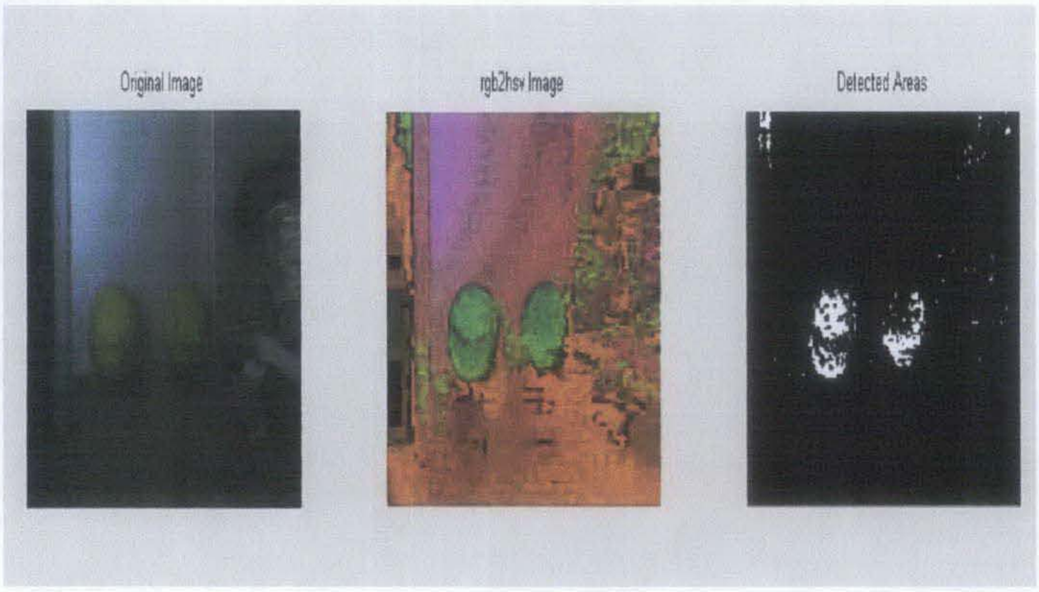


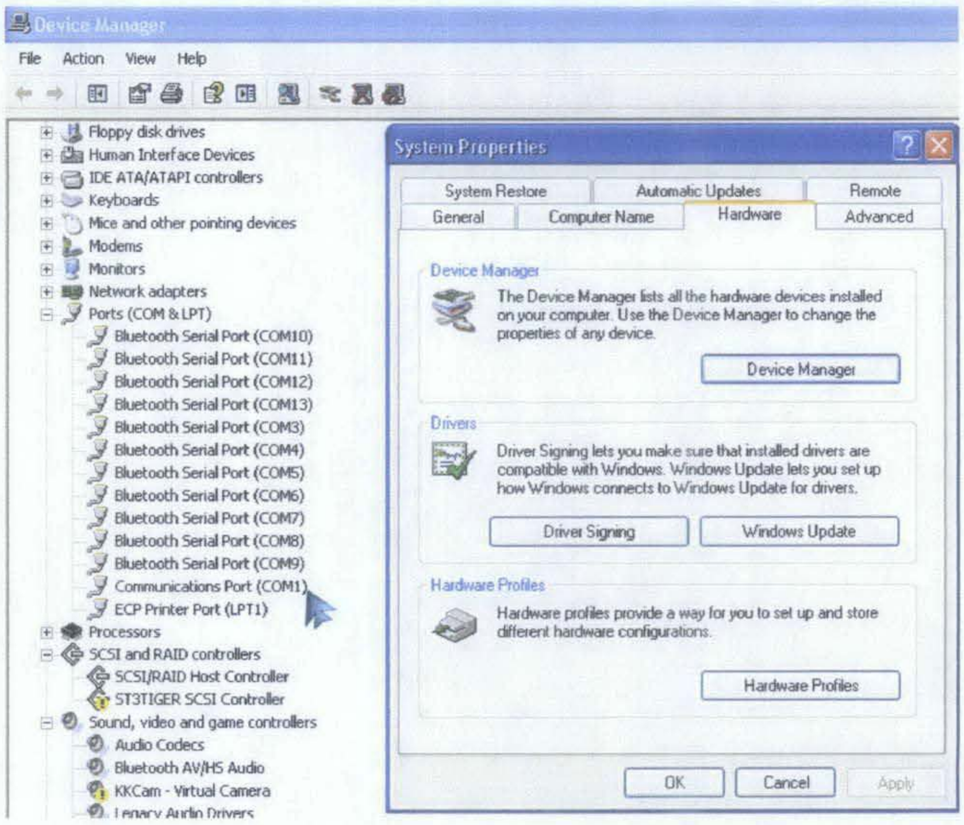
Figure 15: Real Time HSI Colour Detection Algorithm Output Image

rgb2hsv command is light sensitive. Same object under different light coverage will generate different hsv value, shown clearly in Figure 15. It makes image analysis challenging. Web cam is not consistent as it might give too much noise or blur lens focus. As a result, a higher pixel and better focus webcam is needed for image quality improvement.

4.4 Serial Port Communication Testing

Serial port communication serves as the data transmission medium. A string of character (8 bits) at a time. For testing purpose, Port 2 (Receive, RX) with Port 3 (Transmit, TX) is connected together. With the aid of AccessPort software, we can detect the data transmitted and received via serial port.

First of all, we need to make sure the serial port setting is correct. Right click *My Computer*, *Properties*, *Hardware*, *Device Manager*, *Ports (COM & LPT)*, check the available COM port.


Device Manager
File Action View Help
Floppy disk drives
Human Interface Devices
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
Modems
Monitors
Network adapters
Ports (COM & LPT)
Bluetooth Serial Port (COM10)
Bluetooth Serial Port (COM11)
Bluetooth Serial Port (COM12)
Bluetooth Serial Port (COM13)
Bluetooth Serial Port (COM3)
Bluetooth Serial Port (COM4)
Bluetooth Serial Port (COM5)
Bluetooth Serial Port (COM6)
Bluetooth Serial Port (COM7)
Bluetooth Serial Port (COM8)
Bluetooth Serial Port (COM9)
Communications Port (COM1)
ECP Printer Port (LPT1)
Processors
SCSI and RAID controllers
SCSI/RAID Host Controller
ST3TIGER SCSI Controller
Sound, video and game controllers
Audio Codecs
Bluetooth AV/HS Audio
KKCam - Virtual Camera
Legacy Audio Drivers
System Properties
System Restore
Automatic Updates
Remote
General
Computer Name
Hardware
Advanced
Device Manager
Device Manager
Drivers
Driver Signing
Windows Update
Hardware Profiles
Hardware Profiles
OK
Cancel
Apply

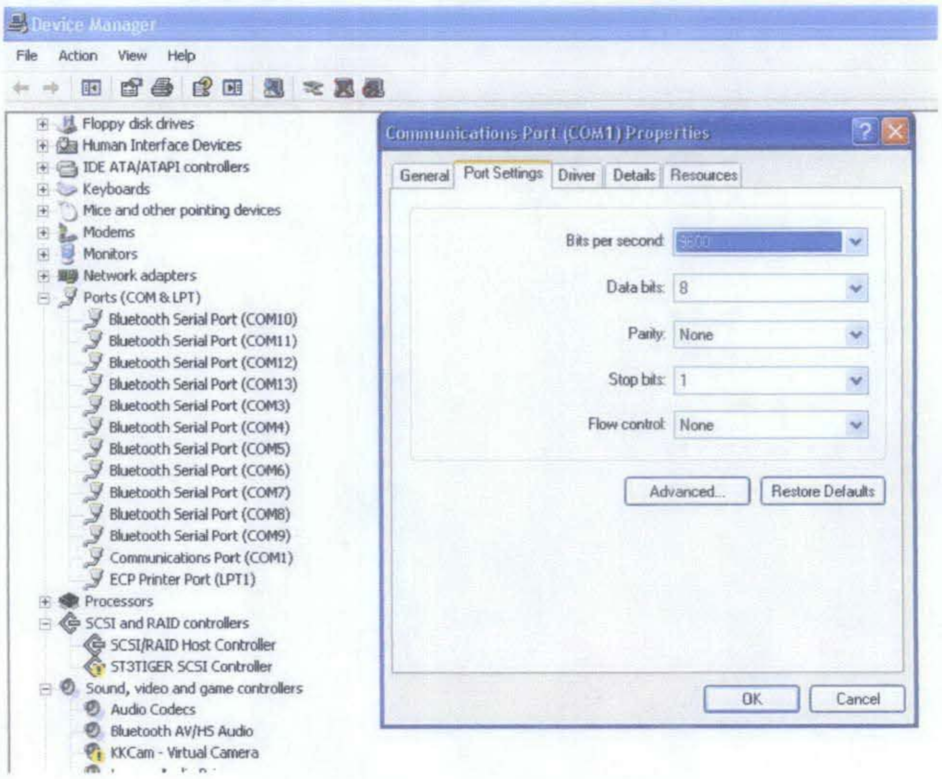
The Device Manager lists all the hardware devices installed on your computer. Use the Device Manager to change the properties of any device.

Driver Signing lets you make sure that installed drivers are compatible with Windows. Windows Update lets you set up how Windows connects to Windows Update for drivers.

Hardware profiles provide a way for you to set up and store different hardware configurations.

Figure 16: Serial Port Configuration

Right click the preferred Serial Port (in this case, *Communication Port, COM1*), choose *Properties*. Go to *Port Setting*. Record the detail of the baud rate, data bits, parity, stop bits, and flow control. Those information need to be matched in Matlab programming. Visual description is shown in Figure 17.


Device Manager

File Action View Help

- Floppy disk drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Modems
- Monitors
- Network adapters
- Ports (COM & LPT)
  - Bluetooth Serial Port (COM10)
  - Bluetooth Serial Port (COM11)
  - Bluetooth Serial Port (COM12)
  - Bluetooth Serial Port (COM13)
  - Bluetooth Serial Port (COM3)
  - Bluetooth Serial Port (COM4)
  - Bluetooth Serial Port (COM5)
  - Bluetooth Serial Port (COM6)
  - Bluetooth Serial Port (COM7)
  - Bluetooth Serial Port (COM8)
  - Bluetooth Serial Port (COM9)
  - Communications Port (COM1)
  - ECP Printer Port (LPT1)
- Processors
- SCSI and RAID controllers
  - SCSI/RAID Host Controller
  - ST3TIGER SCSI Controller
- Sound, video and game controllers
  - Audio Codecs
  - Bluetooth AV/HS Audio
  - KKCam - Virtual Camera

Communications Port (COM1) Properties

General | Port Settings | Driver | Details | Resources

Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Advanced... Restore Defaults

OK Cancel

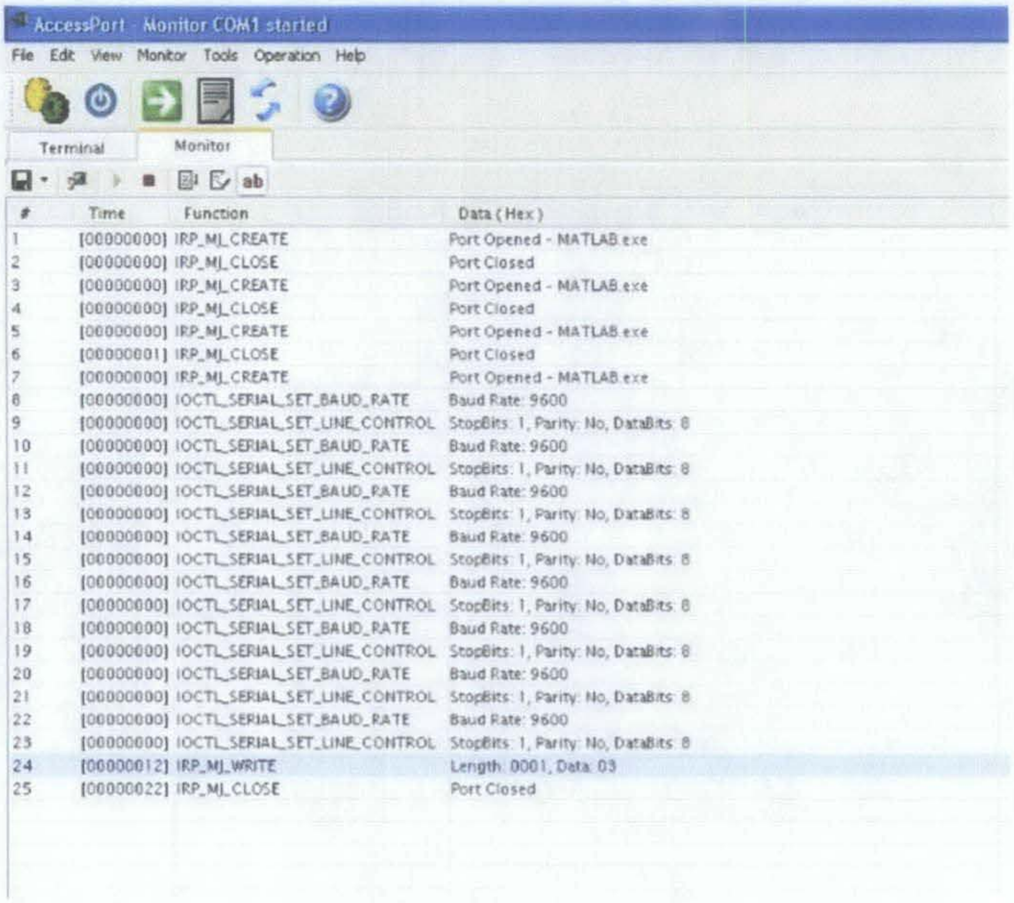
Figure 17: Serial Port Configuration

Note: Same configuration is needed in AccessPort software.

In Matlab, serial port connection is setup using `obj = serial('port')` syntax. As noticed, serial port is setup exactly the same with information stated in the Communication Port (COM1) hardware. The code is as below:

```
inp = 3;

SerPIC = serial('com1');                %<--change this appropriately
set(SerPIC, 'BaudRate', 9600, 'DataBits', 8, 'Parity',
'none','StopBits', 1, 'FlowControl', 'none');

fopen(SerPIC);
fprintf(SerPIC, '%c', inp);
fclose(SerPIC);                     %--close the serial port when done
delete(SerPIC);
clear SerPIC;
```

We can monitor data transmitted and received via AccessPort (Port 2, RX and Port 3, TX of serial port are connected.). Screenshot of the result collected from AccessPort is shown in Figure 18.


AccessPort - Monitor COM1 started
File Edit View Monitor Tools Operation Help
Terminal
Monitor
ab

#	Time	Function	Data (Hex)
1	[00000000]	IRP_MJ_CREATE	Port Opened - MATLAB.exe
2	[00000000]	IRP_MJ_CLOSE	Port Closed
3	[00000000]	IRP_MJ_CREATE	Port Opened - MATLAB.exe
4	[00000000]	IRP_MJ_CLOSE	Port Closed
5	[00000000]	IRP_MJ_CREATE	Port Opened - MATLAB.exe
6	[00000001]	IRP_MJ_CLOSE	Port Closed
7	[00000000]	IRP_MJ_CREATE	Port Opened - MATLAB.exe
8	[00000000]	IOCTL_SERIAL_SET_BAUD_RATE	Baud Rate: 9600
9	[00000000]	IOCTL_SERIAL_SET_LINE_CONTROL	StopBits: 1, Parity: No, DataBits: 8
10	[00000000]	IOCTL_SERIAL_SET_BAUD_RATE	Baud Rate: 9600
11	[00000000]	IOCTL_SERIAL_SET_LINE_CONTROL	StopBits: 1, Parity: No, DataBits: 8
12	[00000000]	IOCTL_SERIAL_SET_BAUD_RATE	Baud Rate: 9600
13	[00000000]	IOCTL_SERIAL_SET_LINE_CONTROL	StopBits: 1, Parity: No, DataBits: 8
14	[00000000]	IOCTL_SERIAL_SET_BAUD_RATE	Baud Rate: 9600
15	[00000000]	IOCTL_SERIAL_SET_LINE_CONTROL	StopBits: 1, Parity: No, DataBits: 8
16	[00000000]	IOCTL_SERIAL_SET_BAUD_RATE	Baud Rate: 9600
17	[00000000]	IOCTL_SERIAL_SET_LINE_CONTROL	StopBits: 1, Parity: No, DataBits: 8
18	[00000000]	IOCTL_SERIAL_SET_BAUD_RATE	Baud Rate: 9600
19	[00000000]	IOCTL_SERIAL_SET_LINE_CONTROL	StopBits: 1, Parity: No, DataBits: 8
20	[00000000]	IOCTL_SERIAL_SET_BAUD_RATE	Baud Rate: 9600
21	[00000000]	IOCTL_SERIAL_SET_LINE_CONTROL	StopBits: 1, Parity: No, DataBits: 8
22	[00000000]	IOCTL_SERIAL_SET_BAUD_RATE	Baud Rate: 9600
23	[00000000]	IOCTL_SERIAL_SET_LINE_CONTROL	StopBits: 1, Parity: No, DataBits: 8
24	[00000012]	IRP_MJ_WRITE	Length: 0001, Data: 03
25	[00000022]	IRP_MJ_CLOSE	Port Closed

Figure 18: Serial Port Transmit / Receive Testing

Based on Matlab code shown above, we send an 8-bit character with value 3, which is 0x03 in hex. Matlab writes/transmits the data trough Port 3 (TX) and received by Port 2 (RX). As we can see from line #24 in AccessPort, it manages to receive the data correctly.

4.5 Wired Serial Port Data Transfer

After making sure that Matlab can transmit the data correctly, the receiver side, PIC microcontroller, needs to be configured to receive the serial port input. In PIC16F877A, the *USART (Universal Synchronous / Asynchronous Receiver Transmitter)* is utilized for asynchronous serial communication. A built in function *#use rs232 ([option])* can be applied to allow serial data communication.

The code for PIC microcontroller (receiver) is as below:

```
#include <16F877A.h>
#include <stdio.h>
#fuses HS, NOWDT
#use delay(clock=20000000)

#use RS232(baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7,
stream=COM1, bits=8)

char c;

#INT_RDA
void receive_isr()
{
   c= 0;
   output_bit(PIN_D2,1);    // Indicator of interrupt occurred
   delay_ms(1000);
   c= fgetc(COM1);
   delay_ms(1000);

   output_d(c);      // Port C as output for serial port data
received
}

void main()
{
   set_tris_d(0x00);
   output_d(0x00);
   enable_interrupts(INT_RDA);
   enable_interrupts(global);

   while(1)
   {
   }
}
```

The detail of the *#use rs232* is based on the serial port communication, with Port C6 always acts as receiver while Port C7 as transmitter.

Interrupts is used so that whenever there is new data received, it will jump to the interrupt loop, where LEDs on Port D2 will be turned on, indicating the data being received (system interrupted). The data received will be in binary form, and it will be shown as the output via port D, coded by *output_d(c)* command.

Since PIC16F (TTL) operates with logic High (2V - 5V) and Low (0V - 0.8V) while rs232 is not, MAX 233 chip is needed for PIC to interact with rs232. For rs232 in between -15V to -3V, MAX 233 will convert the potential difference to 2V – 5V, with logic High (1); on the other hand, rs232 with 3V – 15V will be converted to logic Low (0V – 0.8V):

```
      RS-232                  TTL                Logic
------------------------------------------------------
  -15V ...   -3V  <->  +2V ... +5V      <->  1
   +3V ... +15V   <->   0V ... +0.8V    <->  0
```

After receiving the data, PIC microcontroller will convert the serial data into 3-bit parallel data based on the code provided in **Appendix D**.

4.6 Parallel Data Radio Frequency (RF) Wireless Transmission

As a result of financial limitation, parallel data RF transmitter and receiver modules are used as the medium for the data output from PC serial port to be transmitted to the received located on the robot. TX9902B, an 8-bit trinary address, 6-bit binary data RF transmitter is paired with RX9926, an 8-bit trinary address, 4-bit binary data RF receiver. The addresses on both transmitter and receiver need to be matched to enable any wireless communication. In my project, those addresses are left floating.

There are only six possibilities for the robot to move, thus only THREE bits data (000 for 0 while 110 for 6) are required. TX9902B will feed the 3-bit data through data port D0 (least significant bit), D1, and D2 (most significant bit). With TX9902B turns on and off every time there is new data ready to be sent, the data can be received by RX9926 receiver on the robot.

RX9926 can receive four bits data at a time, which is enough for this project. The data received will be the same with the data on transmitter site, as long as both addresses are same. Valid Transmission pin is available so that we can check the transmission and connection between transmitter and receiver. For both transmitter and receiver, 18cm wire is used as the antenna for the communication purpose.

The dimensions and pins out of TX9902B and RX9926 are shown in Figure 19 and 20.

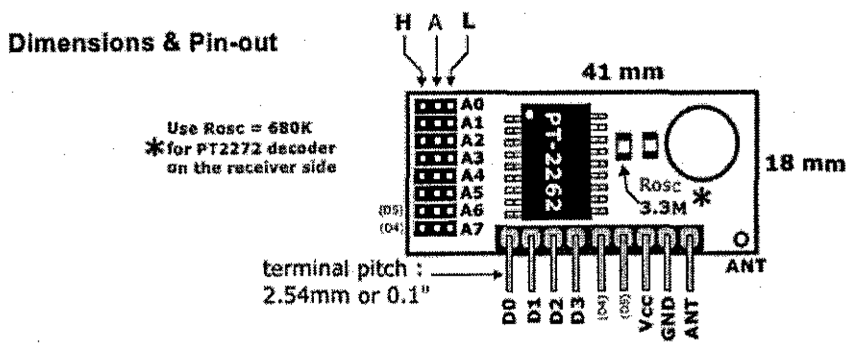


Figure 19: TX9902B RF Transmitter

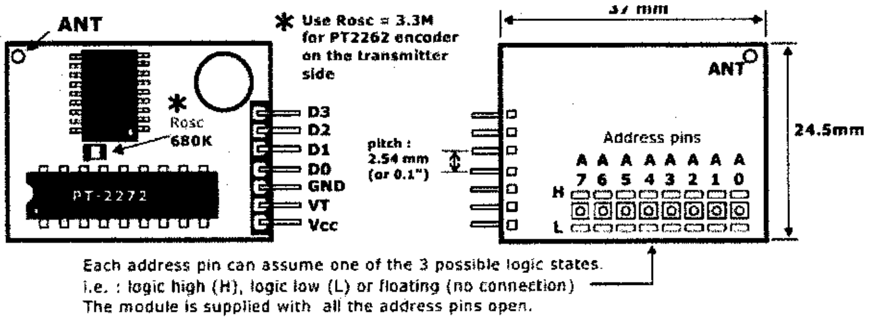

ANT
* Use Rosc = 3.3M for PT2262 encoder on the transmitter side
Rosc 680K
PT-2272
D3
D2
D1
D0
GND
VT
Vcc
pitch : 2.54 mm (or 0.1")
37 mm
ANT
Address pins
A A A A A A A A
7 6 5 4 3 2 1 0
H
L
24.5mm
Each address pin can assume one of the 3 possible logic states.
i.e. : logic high (H), logic low (L) or floating (no connection)
The module is supplied with all the address pins open.

Figure 20: RX9926 RF Receiver

The schematic diagram of the transmitter side is as below:

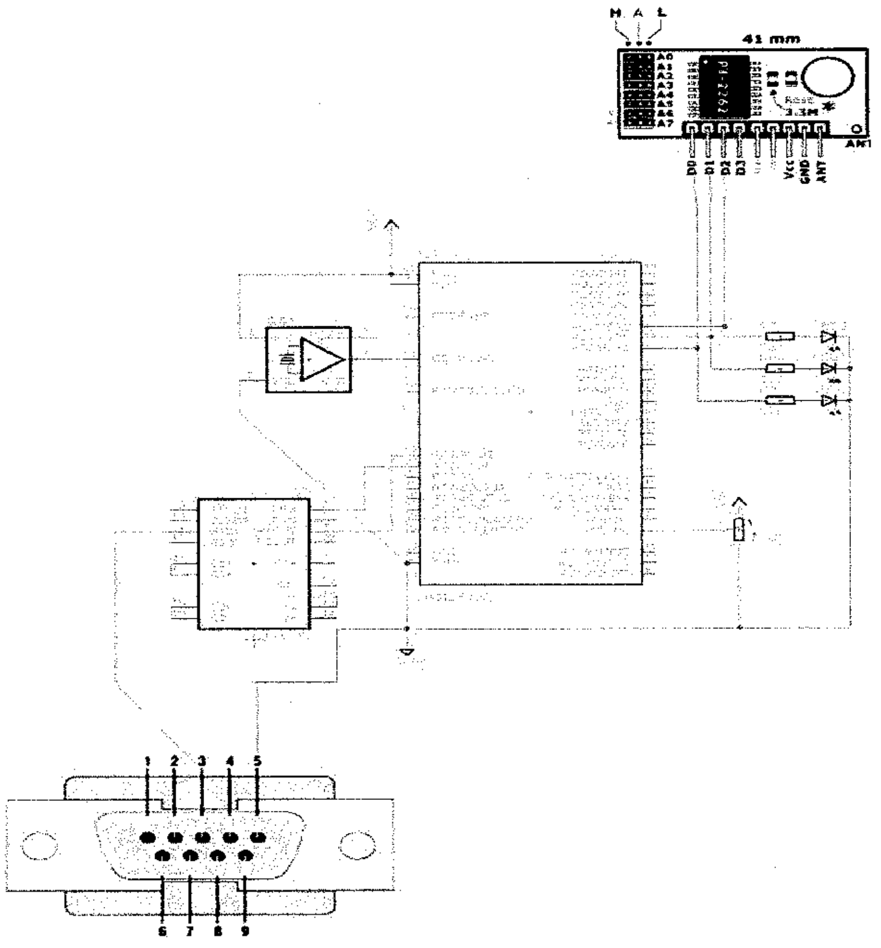


Figure 21: Transmitter (Serial Port, MAX233, RF Transmitter) Schematic

Serial Port pin 3 (TX) is connected to pin 4 (R1IN) of MAX233 chip. Pin 3 (R1OUT) of MAX233 is then connected to Port C7 (RX) in PIC microcontroller.

CHAPTER 5
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The algorithm for HSI colour detection as well as direction analysis for robot to move towards the ball is constructed and tested on different set of images. Real time processing is challenging as the surrounding light effect and web camera resolution quality will result in different image capture. There are more random and unwanted noises in real time image processing, and filtering need to be done from time to time to filter out those unnecessary signals.

Data transmission to transmit processed data from MATLAB to microcontroller is vital. This is because the data is required so that the robot embedded with microcontroller can response and move towards the ball. Serial data communication is preferred as it is simple to be implemented. Wired data transmission is tested to make sure the data can be transmitted via serial port.

Radio frequency (RF) transmitter and receiver modules are used as medium for wireless data communication. Serial data transmitted from serial port is converted to parallel data by PIC microcontroller, and the parallel data is transmitted wirelessly to the RF receiver placed on the robot. Another PIC microcontroller located on the robot will then response with the data received, prompting the robot to move towards the ball.

PIC microcontroller is programmed in the way it will analyze data received, and response based on the programming code written. Microcontroller will be programmed in C programming language.

5.2 Recommendation

The project proposed can still be improved in many ways. The following will discuss about aspects that can be improved:-

- Serial USB wireless data transmission module such as XBEE is recommended for better and more stable wireless data communication. This is because serial port data transmission is slower compared to USB data transmission.
- For the object detection algorithm, colour detection algorithm proposed is one of the techniques used. There are still other suggestions such as shape detection, field extraction, or combination of both. Object detection algorithm can be further enhanced with surrounding factors such as light taken into consideration to come out with more accurate image analysis.
- Microcontroller section can be improved with more decision makings such as speed of robot, controlled by Pulse Width Modulation (PWM), and angle of robot movement so that robot can move towards the targeted ball faster and smoother.

REFERENCES

[1] T. Orazio, N. Ancona, G. Cicirelli and M.Nitti, "*A ball detection algorithm for real soccer image sequences*", 16th Int'l Conf. on Pattern Recognition, Quebec, Canada, Aug. 2002, pp: 210-213.

[2] Hongying Zhang, et al., "*Ball Detection Based on Color Information and Hough Transform*", 2009 International Conference on Artificial Intelligence and Computational Intelligence, pp. 393-397.

[3] Jusoh, R.M., "*Application of Vision Target Localization for Mobile Robot* " Research and Development, 2006. SCOReD 2006. 4th Student Conference on Digital Object Identifier, pp. 144 - 146.

[4] Xiao-Feng Tong; Han-Qing Lu; Qing-Shan Liu, "*An effective and fast soccer ball detection and tracking method* " in Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on Volume: 4

[5] Manigandan, M.; Malathi, G.; Madhavi, N., "*Wireless vision based moving object tracking robot through perceptual color space*" , Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on Digital Object Identifier, Page(s): 20 - 25

[6] Al-Omary, A., "*Autonomous object seeking robot based on FPGA and a single chip microcontroller* ", Computer and Communication Engineering (ICCCE), 2010 International Conference on Digital Object Identifier:, Page(s): 1 - 6

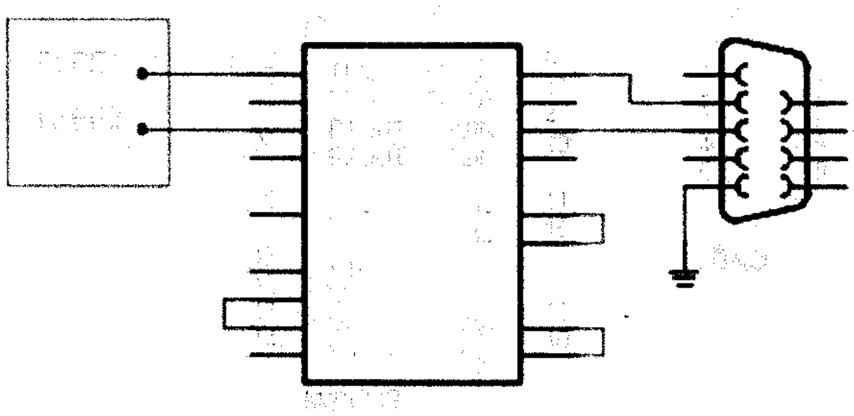
[7] Online PIC Serial Communication Tutorial <http://www.mcuexamples.com/PIC-Serial-Communication.php>

[8] National Semiconductor. October 1999 <http://www.national.com/ds/DS/DS14C232.pdf>

APPENDICES

APPENDIX A

Serial Port Data Communication Connection



APPENDIX B

HSI Colour Detection Algorithm

```
clc;
clear all;
close all;

a = imread('bb.jpg');
hsvVal = [0.2,1,1];                % HSV value for ball = yellow
hsvVal1 = [0.6,1,1];               % HSV value for robot = blue
tol = [0.1,0.5,0.5];               % tolerance for the HIS value, +-
0.1 in HUE


HSV = rgb2hsv(a);

% find the difference between required and real Hue(H) value:
diffH = abs(HSV(:,:,1) - hsvVal(1));

[M,N,t] = size(HSV);
I1 = zeros(M,N); I2 = zeros(M,N); I3 = zeros(M,N);

T1 = tol(1);

I1( find(diffH < T1) ) = 1;

if (length(tol)>1)
    % find the difference between required and real Saturation(S)
value:
    diffS = abs(HSV(:,:,2) - hsvVal(2));
    T2 = tol(2);
    I2( find(diffS < T2) ) = 1;
    if (length(tol)>2)
        % find the difference between required and real Intensity(I)
value:
        difV = HSV(:,:,3) - hsvVal(3);
        T3 = tol(3);
        I3( find(difV < T3) ) = 1;
        I = I1.*I2.*I3;
    else
        I = I1.*I2;
    end
else
    I = I1;
End


subplot(1,3,1);imshow(a);title('Original Image');              %
original image
subplot(1,3,2);imshow(HSV);title('HSI Image');             % rgb2hsv
image
subplot(1,3,3);imshow(I);title('Detected Image
(Yellow)');              % Detected area
```

APPENDIX C

Vector Identification & Direction Analysis Algorithm

```
SE = strel('disk',5);                % erosion
ball = imerode(I,SE);
blue = imerode(B,SE);
red = imerode(R,SE);

ballx = 0; bally = 0; robotx = 0; roboty=0; redx=0; redy=0;
out = 0; out1 = 0; out2=0;

for x=1:M
    for y=1:N
        if out==0
            if ball(x,y) == 1
            ballx = x;
            bally = y;
            out=1;
            end
        end

        if out1==0
            if blue(x,y) == 1
            robotx = x;
            roboty = y;
            out1=1;
            end
        end

        if out2==0
            if red(x,y) == 1
            redx = x;
            redy = y;
            out2=1;
            end
        end
    end
end


% 4 possible directions where the ball is located with robot as
reference
xdiff =0; ydiff =0;
xdiff = robotx - ballx;
ydiff = roboty - bally;
if xdiff>0
    if ydiff>0
        dir = 1;
    end
    if ydiff<=0
        dir = 2;
    end
end

if xdiff <=0
    if ydiff>0
```

```
        dir = 3;
    end

    if ydiff<=0
        dir = 4;
    end
end

% Where the head of robot is facing?
xrobot = redx - robotx;      % tail (red) - head (blue)
yrobot = redy - roboty;
robotdir = 0;

if (xrobot>0) && (yrobot>0)
    robotdir = 1; %robot facing top left corner
end

if (xrobot>0) && (yrobot<=0)
    robotdir = 2; %robot facing top right corner
end

if (xrobot<=0) && (yrobot>0)
    robotdir = 3; %robot facing bottom left
end

if (xrobot<=0) && (yrobot<=0)
    robotdir = 4; %robot facing right
end


% Getting direction for robot to move towards the ball
sig = 0;
if robotdir == 1
    if dir == 1
        sig = 2; % move forward
    end
    if dir == 2
        sig = 3; % forward right
    end
    if dir == 3
        sig = 6; % backward left
    end
    if dir == 4
        sig = 5; % backward
    end
end


if robotdir == 2
    if dir == 1
        sig = 1; % move forward left
    end
    if dir == 2
        sig = 2; % forward
    end
    if dir == 3
        sig = 5; % backward
    end
    if dir == 4
```

```
        sig = 4; % backward right
    end
end

if robotdir == 3
    if dir == 1
        sig = 4; % move backward right
    end
    if dir == 2
        sig = 5; % backward
    end
    if dir == 3
        sig = 2; % forward
    end
    if dir == 4
        sig = 1; % forward left
    end
end

if robotdir == 4
    if dir == 1
        sig = 5; % move backward
    end
    if dir == 2
        sig = 6; % backward left
    end
    if dir == 3
        sig = 3; % forward right
    end
    if dir == 4
        sig = 2; % forward
    end
end


subplot(3,3,1);imshow(a);title('Original');
subplot(3,3,2);imshow(I);title('Original Detected Area (ball)');
subplot(3,3,3);imshow(ball);title('Image (ball) after Erosion');
subplot(3,3,4);imshow(HSV);title('Original HSV');
subplot(3,3,5);imshow(B);title('Original Detected Area (robot
blue)');
subplot(3,3,6);imshow(blue);title('Image (robot blue) after
Erosion');
subplot(3,3,8);imshow(R);title('Original Detected Area (robot red)');
subplot(3,3,9);imshow(red);title('Image (robot red) after Erosion');
```

APPENDIX D

PIC Code for Serial / Parallel Data Conversion

```
#include <16F877A.h>
#include <stdio.h>
#fuses HS, NOWDT
#use delay(clock=20000000)

#use RS232(baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7, stream=COM1,
bits=8)

char c;

#INT_RDA
void receive_isr()
{
   c= 0;
   output_bit(PIN_D3,1);
   delay_ms(1000);
   c= fgetc(COM1);
   delay_ms(1000);

   if (c==6)
   {
   output_bit(PIN_D2,1);
   output_bit(PIN_D1,1);
   output_bit(PIN_D0,0);
   delay_ms(1000);
   }

   else if (c==5)
   {
   output_bit(PIN_D2,1);
   output_bit(PIN_D1,0);
   output_bit(PIN_D0,1);
   delay_ms(1000);
   }

   else if (c==4)
   {
   output_bit(PIN_D2,1);
   output_bit(PIN_D1,0);
   output_bit(PIN_D0,0);
   delay_ms(1000);
   }

   else if (c==3)
   {
   output_bit(PIN_D2,0);
   output_bit(PIN_D1,1);
   output_bit(PIN_D0,1);
   delay_ms(1000);
   }

   else if (c==2)
   {
   output_bit(PIN_D2,0);
   output_bit(PIN_D1,1);
   output_bit(PIN_D0,0);
   delay_ms(1000);
   }

   else if (c==1)
   {
   output_bit(PIN_D2,0);
   output_bit(PIN_D1,0);
   output_bit(PIN_D0,1);
   delay_ms(1000);
   }

   else
   {
   output_bit(PIN_D2,0);
   output_bit(PIN_D1,0);
   output_bit(PIN_D0,0);
   delay_ms(1000);
   }
}

void main()
{
   set_tris_d(0x00);
   output_d(0x00);
   enable_interrupts(INT_RDA);
   enable_interrupts(global);

   while(1)
   {
   }
}
```

APPENDIX E

PIC Code for Motor Response

```
#include <16F877A.h>
#include <stdio.h>
#fuses HS, NOWDT
#use delay(clock=20000000)

#define bit0   PIN_C6
#define bit1   PIN_D5
#define bit2   PIN_D6

#define m1_1   PIN_C3 //left
#define m1_2   PIN_C0 //left
#define m2_1   PIN_C5 //right
#define m2_2   PIN_C4 //right

void straight()
{
output_bit(m1_1,0);//left
output_bit(m1_2,1);//left
output_bit(m2_1,0);//right
output_bit(m2_2,1);//right
}

void reverse()
{
output_bit(m1_1,1);//left
output_bit(m1_2,0);//left
output_bit(m2_1,1);//right
output_bit(m2_2,0);//right
}


void right()
{
output_bit(m1_1,0);
output_bit(m1_2,1);
output_bit(m2_1,0);
output_bit(m2_2,0);
}

void left()
{
output_bit(m1_1,0);
output_bit(m1_2,0);
output_bit(m2_1,0);
output_bit(m2_2,1);
}

void revright()
{
output_bit(m1_1,1);
output_bit(m1_2,0);
output_bit(m2_1,0);
output_bit(m2_2,0);
}
```

```
void revleft()
{
output_bit(m1_1,0);
output_bit(m1_2,0);
output_bit(m2_1,1);
output_bit(m2_2,0);
}

void stop()
{
output_bit(m1_1,0);
output_bit(m1_2,0);
output_bit(m2_1,0);
output_bit(m2_2,0);
}

void main()
{
   set_tris_d(0x60);    // Port D5, D6 as input, others are output
   set_tris_c(0x40);    // Port C6 as input, others are output
   output_bit(PIN_C0,0);
   output_bit(PIN_C1,0);
   output_bit(PIN_C2,0);
   output_bit(PIN_C3,0);
   output_bit(PIN_C4,0);
   output_bit(PIN_C5,0);

   setup_timer_2(T2_DIV_BY_4,255, 1);
   setup_ccp1(CCP_PWM);
   setup_ccp2(CCP_PWM);

   while(1)
   {

      if(input(bit2) && input(bit1) && !input(bit0))
      {
         revleft();delay_ms(2000);    //sig = 6
         stop();delay_ms(1000);
      }

      else if(input(bit2) && !input(bit1) && input(bit0))
      {
         reverse();delay_ms(2000);    // sig = 5
         stop();delay_ms(1000);
      }

      else if(input(bit2) && !input(bit1) && !input(bit0))
      {
         revright();delay_ms(2000);    // sig = 4
         stop();delay_ms(1000);
      }

      else if(!input(bit2) && input(bit1) && input(bit0))
      {
         right();delay_ms(2000);    // sig = 3
         stop();delay_ms(1000);
      }

      else if (!input(bit2) && input(bit1) && !input(bit0))
      {
            straight();delay_ms(2000);  // sig = 2
            stop();delay_ms(1000);
      }

      else if(!input(bit2) && !input(bit1) && input(bit0))
      {
         left();delay_ms(2000);    // sig = 1
         stop();delay_ms(1000);
      }

      else
      {
            stop();delay_ms(2000);      // sig = 0
      }
}
```