中華大學資訊工程學系

系統開發專題報告

Lego NXT-藍芽足球機器人

Lego NXT-The Robot Plays the Soccer By Using Bluetooth Device

專題生:洪智賢

指導教授:周智勳　教授

專題編號 PRJ2011-CSIE-10000

執行時間:100 年七月至 101 年 1 月

目錄

背景與目的

背景:結合 Lego　NXT 主機及超音波 SENSOR、視訊裝置、藍芽裝置，利用電腦上視訊裝置所擷取之圖片，C#讀取之並進行影像處理，把運算結果轉換成機器人之動作命令，再透過藍芽裝置，把要傳達給機器人之動作命令傳給 Lego　NXT 機器人

目的:將上述的機器人動作命令傳給 Lego　NXT 機器人,使之能夠找到足球,並且邊踢足球邊移動還要能夠閃避障礙物,最後在距離球門一定之距離內,能夠成功把球踢到球門之中。

Figure 1.1:LEGO NXT主機- Intelligent NXT Brick

Figure 1.2:超音波SENSOR- Ultrasonic Sensor

Figure 1.3:藍芽裝置-Bluetooth Device

Figure 1.4:視訊裝置- Web Camera

Figure 1.5:組裝過後的 Lego NXT 足球機器人(主機+超音波 SENSOR)

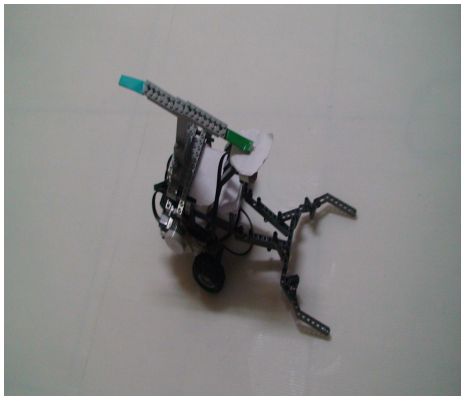
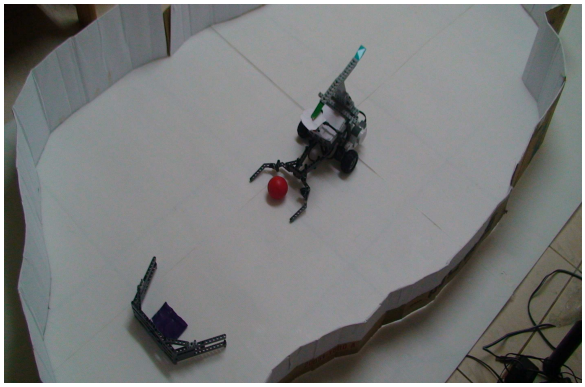


Figure 1.6:Lego NXT 足球機器人與球場之全景圖



(一) 專題系統架構

1.1 實作前之規劃

a.使用 Web Camera 在每一個單位時間內對 Lego NXT 足球機器人、足球、球門同時進行拍照動作。

b.利用C#來讀取 Web Camera 每一個單位時間所擷取之圖片。

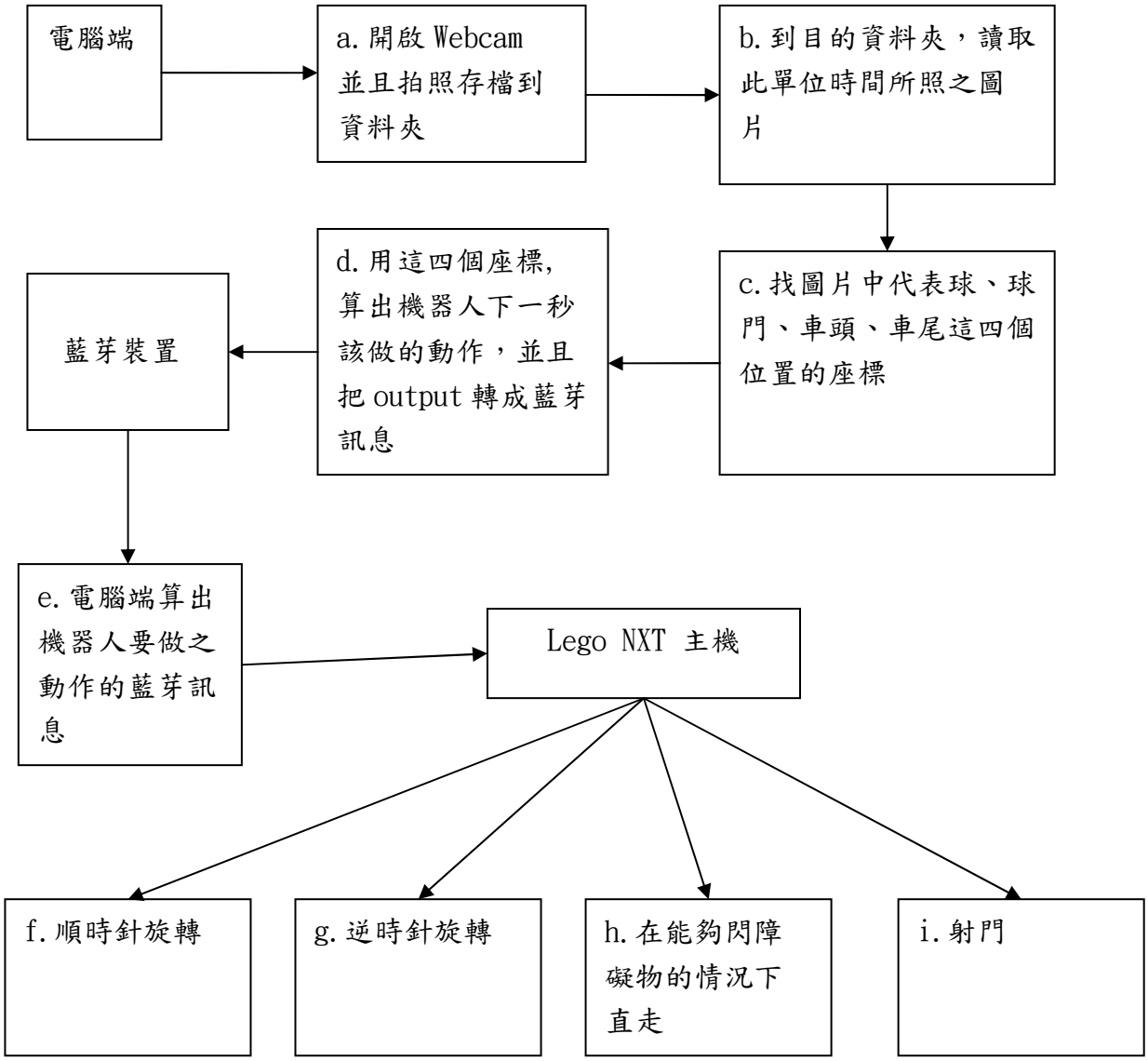
c.再利用讀取之圖片進行影像處理，讀取相關Pixel之座標進行運算

,最後根據運算完之結果，並決定Lego NXT機器人下一次單位時間要做的動作。

d.透過電腦上藍芽裝置,把決定好的動作命令傳給Lego NXT主機。

e.當Lego NXT主機接受到命令時,會把命令轉換成自己能夠辨識的程式碼,並且執行這單位時間所辨識之程式碼。

1.2 系統運作流程圖



(二)方法說明

說明:這裡將介紹系統架構流程圖中，各個流程要如何實作出來。

==========

a.開啟 Webcam 並且拍照存檔到資料夾

說明：利用 C#的 MSDN 內建函式實作出來，以及利用自己寫的類別所創之物件來存檔

Figure 2.1:開啟/關閉 Webcam

```
//Ctrl+A : 開始視訊 與 停止視訊
private void startTheWebcamToolStripMenuItem_Click(object sender, EventArgs e)
{
    //配置記憶體給capture
    if capture is not created, create it now


    if (_capture != null)
    {
        if (_captureInProgress)
        {  //stop the capture
            Application.Idle -= new EventHandler(ProcessFrame);
            startTheWebcamToolStripMenuItem.Text = "Start The Webcam";
        }
        else
        {
            //start the capture
            startTheWebcamToolStripMenuItem.Text = "Stop";
            Application.Idle += new EventHandler(ProcessFrame);
        }

        _captureInProgress = !_captureInProgress;
    }

}
```

Figure 2.2:用 Webcam 拍照

```
//Ctrl+C :開始擷取圖片 與  停止擷取圖片
private void startCapturingToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (timer1.Enabled == true)
    {
        timer1.Enabled = false;
        startCapturingToolStripMenuItem.Text = "Stop Capturing";
        Application.Idle += new EventHandler(ProcessFrame);
        Application.Exit();
    }
    else
    {
        //每1000毫秒(1秒)存取一張圖片
        this.timer1.Interval = 1000;
        timer1.Enabled = true;
    }
}
```

Figure 2.3:將拍照之圖存到目的資料夾

```
// 建立一個專門秀圖的 Form 類別
class ImageForm : Form
{
    Image image;

    // 預設建構子建構子
    public ImageForm(int Filename)
    {
        if (Filename >= 0)
        {
            image = Image.FromFile(@"d:\\123\\" + Filename + ".jpg");
            Filename++;
        }
    }
    public void LoadImage(String Filename)
    {
        // Step 1: 載入影像
        image = Image.FromFile(Filename);
        this.Text = Filename;

        // Step 2: 調整視窗的大小
        this.Height = image.Height;
        this.Width = image.Width;
    }



//所存之圖片,都放在D槽內名為123的資料夾
this.captureImageBox.Image.Save("d:\\123\\" + num + ".jpg");
ImageForm MyImage = new ImageForm(num);
```

==========

b.到目的資料夾，讀取此單位時間所照之圖片

說明：把照之圖片 show 出來，用 C#內建函式實作在的主程式的類別

Figure 2.4:加在主程式的類別中

```
public void LoadImage(String Filename)
{
    // Step 1: 載入影像
    image = Image.FromFile(Filename);
    this.Text = Filename;

    // Step 2: 調整視窗的大小
    this.Height = image.Height;
    this.Width = image.Width;
}

protected override void OnPaint(PaintEventArgs e)
{
    // Step 3: 顯示出影像
    e.Graphics.DrawImage(image, 0, 0, image.Width, image.Height);
}
```

c.找圖片中代表球、球門、車頭、車尾這四個位置的座標

說明:

(1)先把 C#主程式的屬性改成 unsafe,因為讀圖片中之 pixel 座標會牽涉到記憶體修改。

(2) 在主程式的類別中，先分析圖片中 RGB 資訊，利用 Bitmap 將 image 包起來，取得 pixel 顏色資訊。

(3) 在主程式的類別中實作出以下功能，I.先鎖住存放圖片的記憶體 ，II.取得 pixel 資料的起始位址 ，III.計算每行的像點所佔據的 byte 總數，IV.直接利用指標, 把影像資料取出來。

(4)先使用 C#工具箱的 timer1_Tick，再利用創到類別之物件把讀取之圖拿來分析 RGB 資訊

(5)宣告要代表球、車頭、車尾、球門的四個座標，並且找出四個顏色的 RGB，利用兩個 FOR 迴圈找出此圖所有的座標之 RGB 資料,如果有符合有座標符合這四個顏色的 RGB 中其

中一個顏色的 RGB，既將此座標當作該位置的座標（EX:找到球的座標，球的顏色的 RGB 為 [255,0,0]）

Figure 2.5:如何把讀取之圖，將之分析其 RGB 資訊

```
// 傳回 RGB 陣列資訊
public int[, ,] getRGBData()
{
    MessageBox.Show("開始 RGB 彩色資訊讀取");

    // Step 1: 利用 Bitmap 將 image 包起來
    Bitmap bimage = new Bitmap(image);
    int Height = bimage.Height;
    int Width = bimage.Width;
    int[, ,] rgbData = new int[Width, Height, 3];


    // Step 2: 取得像點顏色資訊
    for (int y = 0; y < Height; y++)
    {
        for (int x = 0; x < Width; x++)
        {
            Color color = bimage.GetPixel(x, y);
            rgbData[x, y, 0] = color.R;
            rgbData[x, y, 1] = color.G;
            rgbData[x, y, 2] = color.B;
        }
    }

    MessageBox.Show("RGB 彩色資訊讀取完成");

    return rgbData;
}
```

Figure 2.6:用高效率的方式，把圖的資訊完整分析出來(pixel 數、定義所有 pixel 之 RGB)

```
public static int[, ,] getRGBData(Bitmap bimage)
{
    // Step 1: 先鎖住存放圖片的記憶體
    BitmapData bmData = bimage.LockBits(new Rectangle(0, 0, bimage.Width, bimage.Height),
                                        ImageLockMode.ReadOnly,
                                        PixelFormat.Format24bppRgb);
    int stride = bmData.Stride;

    // Step 2: 取得像點資料的起始位址
    System.IntPtr Scan0 = bmData.Scan0;

    // 計算每行的像點所佔據的byte 總數
    int ByteNumber_Width = bimage.Width * 3;

    // 計算每一行後面幾個 Padding bytes
    int ByteOfSkip = stride - ByteNumber_Width;
    int Height = bimage.Height;
    int Width = bimage.Width;
    int[, ,] rgbData = new int[Width, Height, 3];

    // Step 3: 直接利用指標，把影像資料取出來
    unsafe
    {
        byte* p = (byte*)(void*)Scan0;
        for (int y = 0; y < Height; y++)
        {
            for (int x = 0; x < Width; x++)
            {
                rgbData[x, y, 2] = p[0];     // B
                ++p;
                rgbData[x, y, 1] = p[0];     // G
                ++p;
                rgbData[x, y, 0] = p[0];     // R
                ++p;
            }
            p += ByteOfSkip; // 跳過剩下的 Padding bytes
        }
    }

    bimage.UnlockBits(bmData);
    return rgbData;
}
```

Figure 2.7:宣告四個座標，並且定義四個顏色之 RGB，然後讀取此圖所有的 pixel
把符合球、車頭、車尾、球門代表這四個部份的顏色座標一一找出來

```
for (int y = 0; y < Height; y++)
{
    for (int x = 0; x < Width; x++)
    {
        if (rgbData[x, y, 0] >= 120 && rgbData[x, y, 1] <= 40  && rgbData[x, y, 2] <= 40 && (rgbData[x, y, 0] != 255 && rgbData[x, y, 1] != 25
        {
            temp1_X = x;
            temp1_Y = y;
            tempG1 = rgbData[x, y, 0];
            tempG2 = rgbData[x, y, 1];
            tempG3 = rgbData[x, y, 2];
        }
        if (rgbData[x, y, 0] < 70 && rgbData[x, y, 1] >= 100 && rgbData[x, y, 2] >= 100 && (rgbData[x, y, 0] != 255 && rgbData[x, y, 1] != 255
        {
            temp3_X = x;
            temp3_Y = y;
            tempG4 = rgbData[x, y, 0];
            tempG5 = rgbData[x, y, 1];
            tempG6 = rgbData[x, y, 2];
        }
        if (rgbData[x, y, 0] >= 145 && rgbData[x, y, 1] >= 130 && rgbData[x, y, 2] >= 20 && rgbData[x, y, 2] < 70 && (rgbData[x, y, 0] != 255
        {
            temp2_X = x;
            temp2_Y = y;
            tempG7 = rgbData[x, y, 0];
            tempG8 = rgbData[x, y, 1];
            tempG9 = rgbData[x, y, 2];
        }
        if (rgbData[x, y, 0] < 40 && rgbData[x, y, 1] >= 60 && rgbData[x, y, 1] <= 90 && rgbData[x, y, 2] >= 10 && rgbData[x, y, 2] <= 50 && (
        {
            temp8_X = x;
            temp8_Y = y;
            tempG10 = rgbData[x, y, 0];
            tempG11 = rgbData[x, y, 1];
            tempG12 = rgbData[x, y, 2];
        }
    }
}
```

---

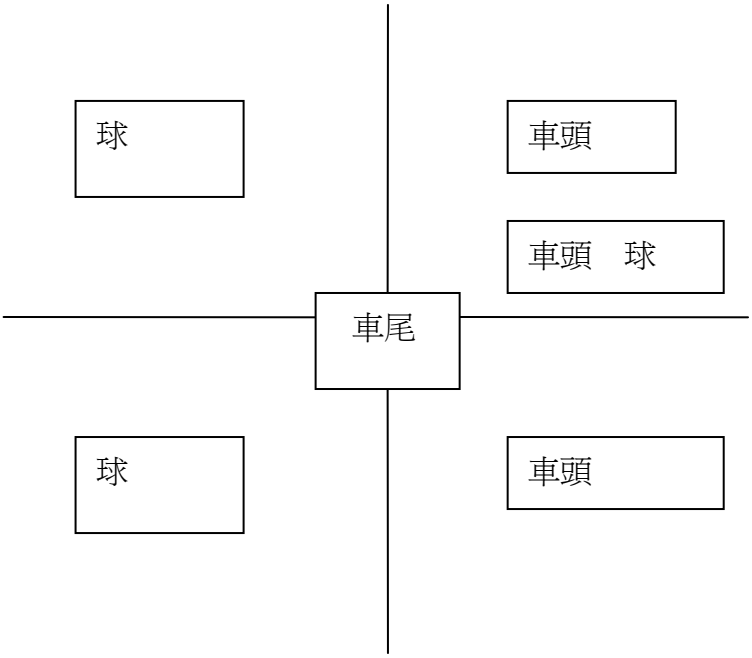
d.用這四個座標,算出機器人下一秒該做的動作，並且把 output 轉成藍芽訊息

說明：算出機器人的動作總共有四種

(1) 順/逆時針旋轉

利用球/球門、車頭、車尾這三個座標,用車尾當原點在利用剩下兩個座標所在之象限的分布位置來觀察是否順/逆時針

先分成三種情形討論



I.球/球門跟車頭之象限差為 1/-1

由 I、II 象限來看，象限差為 1，所以就是車頭需逆時針找球,在看 III、IV 象限，象限差為-1，所以就是車頭需順時針找球

II. 球/球門跟車頭之象限差為|2|

計算球跟原點(車尾)(X,Y)跟創的新座標(X,Y+5)或者(X,Y-5),這三點算出夾角,如果此角度>車尾車頭 X 軸之角度,則逆時針,否則順時針

III. 球/球門跟車頭之象限差為 0

再上圖之第一象限來看，計算球跟原點(車尾)(X,Y)跟創的新座標(X,Y-5),如果（新座標＋原點＋車頭三點算兩個向量）的夾角＜（新座標＋原點＋球三點算兩個向量）的夾角，則順時針，反之逆時針。其他向量亦可用此觀念推倒來求出，機器人如何順/逆時針轉到球之正面。

Figure 2.8:先定義象限位置

```
if ((temp2_X > temp3_X) && (temp2_Y > temp3_Y)) { quadrant1 = 1; }//++
else if ((temp2_X < temp3_X) && (temp2_Y > temp3_Y)) { quadrant1 = 2; }//-+
else if ((temp2_X < temp3_X) && (temp2_Y < temp3_Y)) { quadrant1 = 3; }//--
else if ((temp2_X > temp3_X) && (temp2_Y < temp3_Y)) { quadrant1 = 4; }//+-

if ((temp1_X > temp3_X) && (temp1_Y > temp3_Y)) { quadrant2 = 1; }//++
else if ((temp1_X < temp3_X) && (temp1_Y > temp3_Y)) { quadrant2 = 2; }//-+
else if ((temp1_X < temp3_X) && (temp1_Y < temp3_Y)) { quadrant2 = 3; }//--
else if ((temp1_X > temp3_X) && (temp1_Y < temp3_Y)) { quadrant2 = 4; }//+-
```

Figure 2.9:實作利用象限差判斷順/逆時針

```
//y+5
temp4_X = temp3_X;
temp4_Y = temp3_Y + 15;
vector_U_X = temp4_X - temp3_X;
vector_U_Y = temp4_Y - temp3_Y;
vector_V_X = temp1_X - temp3_X;
vector_V_Y = temp1_Y - temp3_Y;
norm2_U = System.Math.Sqrt(vector_U_X * vector_U_X + vector_U_Y * vector_U_Y);
norm2_V = System.Math.Sqrt(vector_V_X * vector_V_X + vector_V_Y * vector_V_Y);
innerProduct = vector_U_X * vector_V_X + vector_U_Y * vector_V_Y;
theValueofAngleFromCos = innerProduct / (norm2_U * norm2_V);
angle2 = System.Math.Acos(theValueofAngleFromCos) * 180 / Math.PI;
```

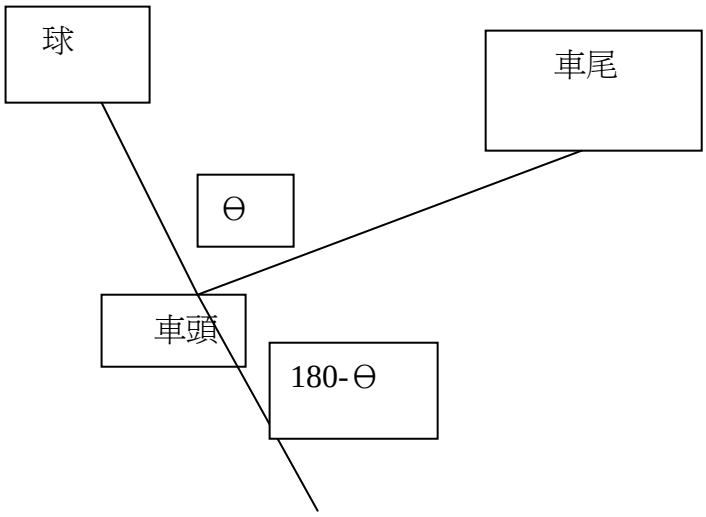
```
//y-5
temp5_X = temp3_X;
temp5_Y = temp3_Y - 15;
vector_U_X = temp4_X - temp3_X;
vector_U_Y = temp4_Y - temp3_Y;
vector_V_X = temp1_X - temp3_X;
vector_V_Y = temp1_Y - temp3_Y;
norm2_U = System.Math.Sqrt(vector_U_X * vector_U_X + vector_U_Y * vector_U_Y);
norm2_V = System.Math.Sqrt(vector_V_X * vector_V_X + vector_V_Y * vector_V_Y);
innerProduct = vector_U_X * vector_V_X + vector_U_Y * vector_V_Y;
theValueofAngleFromCos = innerProduct / (norm2_U * norm2_V);
angle3 = System.Math.Acos(theValueofAngleFromCos) * 180 / Math.PI;
```

Figure 2.10:最後把符合條件集中再逆時針,反之順時針

```
if (quadrant1 == 0 || quadrant2 == 0)
{
    clockwise = 3;
}
else if ((quadrant2 == 1 && quadrant1 == 2) || (quadrant2 == 2 && quadrant1 == 3) || (quadrant2 == 3 && quadrant1 == 4) || (q
{
    clockwise = 1;
}
```

(2) 直走

當機器人跟球/球門成一直線，則直走找球/球門



Solve:利用科西不等式概念延伸之算式

I.先利用球/球門　車頭　車尾 三個座標，算出兩個向量

U1=(x1-x2,y1-y2)=(X1,Y1)

U2=(x3-x2,y3-y2)=(X2,Y2)

II.在取兩個 NORM

NORM1=X1^2+Y1^2

NORM2=X2^2+Y2^2

III.取內積

內積=X1*X2+Y1*Y2

IV.取 COSΘ

$$\cos\theta = \frac{\text{內積}}{\text{NORM1}*\text{NORM2}}$$

V.180-Θ就是轉的角度，使車子能夠正面向球，達到直走找求目的

Figure 2.11:判斷機器人如何轉到正面

```
//紅球辨識
vector_U_X = temp1_X - temp2_X;
vector_U_Y = temp1_Y - temp2_Y;
vector_V_X = temp3_X - temp2_X;
vector_V_Y = temp3_Y - temp2_Y;
norm2_U = System.Math.Sqrt(vector_U_X * vector_U_X + vector_U_Y * vector_U_Y);
norm2_V = System.Math.Sqrt(vector_V_X * vector_V_X + vector_V_Y * vector_V_Y);
innerProduct = vector_U_X * vector_V_X + vector_U_Y * vector_V_Y;
theValueofAngleFromCos = innerProduct / (norm2_U * norm2_V);
angle = System.Math.Acos(theValueofAngleFromCos) * 180 / Math.PI;

double antiAngle = 0;
antiAngle = 180 - angle;
```

(3)射門

當機器人帶球找球門時,如果轉到跟球門適當的角度便會直走找球門,如果再射程範圍內便會射門

Figure 2.12:如何判斷射門條件

```
//用來判斷紅球跟球門的距離是否夠近
temp4 = temp1_X - temp8_X;
temp5 = temp1_Y - temp8_Y;
temp6 = System.Math.Sqrt(temp4 * temp4 + temp5 * temp5);
temp7 = (int)temp6;


if (readyToGoal == 1)
{
    if (antiAngle10 < 15 && temp7 < 150)
    {
        label29.Text = "射門!!!!!!!";
        int_mes(6,0);
        // Application.Exit();
    }
    else if (antiAngle10 < 7 && temp7 >= 150)
```

==========

e.電腦端算出機器人要做之動作的藍芽訊息

說明: 主程式會把要傳的訊息用 BYTE 的方式存在一個正確的通訊協定的陣列(長度:2BYTE) 然後傳給機器人,其中這通訊協定包含了是否回傳訊息,傳到第幾個信箱,以及要傳的訊息(EX:數字)轉成 16 進位 code

Figure 2.13:C#上要傳的訊息轉成認可的藍芽通訊協定的訊息

```
SerialPort nxt = new SerialPort();
void int_mes(double s, int mail)
{
    int tmp = Convert.ToInt32(s);
    byte[] mes = { 0x00, 0x09, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00 };

    mes[2] = (byte)(mail);
    for (int i = 0; i <= 3; i++)
    {
        mes[i + 4] = (byte)tmp;
        tmp >>= 8;
    }
    sendmes(mes);
}

void sendmes(byte[] mes)
{
    Byte[] mes_h = { 0x00, 0x00 };

    mes_h[0] = (byte)mes.Length;
    nxt.Write(mes_h, 0, mes_h.Length);
    nxt.Write(mes, 0, mes.Length);
}

string link(string s)
{
    if (nxt.IsOpen)
    {
        try
        {
            nxt.Close();
            return "end";
        }
        catch
        {
            return "Error";
        }
    }
    else
    {
        nxt.PortName = "COM" + s;
        try
        {
            nxt.Open();
            nxt.ReadTimeout = 500;
            return "true";
        }
        catch
        {
            return "Error";
        }
    }
}
```

---

f、g、h、i：ＮＸＴ機器人收到訊息做的四個動作

說明：機器人如何把收之訊息轉成動作

(1) 用 BrixCC 內建的函式來接收訊息

(2) 再利用 switch 把收到的訊息一一對應該動作

Figure 2.14:NXT 接收訊息

```
task main()
{
   byte b = 5;

   //BTCommCheck(0);
   while(true)
   {
      ReceiveRemoteNumber(0,true,b);
      nxt_walk(b);
   }
}
```

Figure 2.15:利用 switch 把收到的訊息轉成動作

```
void nxt_walk(int select)
{

   switch(select)
   {
      case 1://順時針轉50
        RotateMotorEx(OUT_AC,40,50,100,true,false);
        break;
      case 2://順時針轉30
        RotateMotorEx(OUT_AC,40,30,100,true,false);
        break;
      case 3://逆時針轉50
        RotateMotorEx(OUT_AC,40,50,-100,true,false);
        break;
      case 4:
        Wait(100);
        break;
      case 5://順時針轉15
        RotateMotorEx(OUT_AC,40,15,100,true,false);
        break;
      case 6://射門
        RotateMotorEx(OUT_AC,-30,50,0,true,false);
        Wait(100);
        RotateMotorEx(OUT_AC,100,300,0,true,false);
        Wait(100);
        break;
      case 7://順時針轉10
        RotateMotorEx(OUT_AC,80,10,100,true,false);
        break;
```

```
  case 8://大前進
     SetSensorLowspeed(IN_3);
     if(SensorUS(IN_3) <=23)
     {
        PlayTone(344,400);
        Wait(100);
        RotateMotorEx(OUT_AC,-30,10,0,true,false);
        Wait(100);
        RotateMotorEx(OUT_AC,40,120,-100,true,false);
        Wait(100);
        RotateMotorEx(OUT_AC,40,500,0,true,false);
        Wait(100);
        RotateMotorEx(OUT_AC,40,120,100,true,false);
        Wait(100);
     }
     else
     {
        RotateMotorEx(OUT_AC,40,30,0,true,false);
     }
    break;
  case 9://逆時針轉30
    RotateMotorEx(OUT_AC,40,30,-100,true,false);
    break;
  case 10://逆時針轉15
    RotateMotorEx(OUT_AC,40,15,-100,true,false);
    break;
  case 11://逆時針轉10
    RotateMotorEx(OUT_AC,80,10,-100,true,false);
    break;
  case 12:
     SetSensorLowspeed(IN_3);
     if(SensorUS(IN_3) <=23)
     {
        PlayTone(344,400);
        Wait(100);
        RotateMotorEx(OUT_AC,-30,10,0,true,false);
        Wait(100);
```

```
        }
        else
        {
           RotateMotorEx(OUT_AC,100,50,0,true,false);
        }
       break;
     default:
       break;
  }
  NumOut(5,LCD_LINE2,select);
}
```

(三)實驗過程與結果

3-1 架構環境

i.先準備好材料

Webcam*1

藍芽裝置*1

Lego NXT 主機*1

Lego 材料數個

超音波 SENSOR*1

紅色塑膠球*1

用 Lego 材料做成之球門

用瓦楞紙做的圍牆*1

用四個白色瓦楞紙做的草皮*1

ii.場地製作流程

a. 先把 Webcame 跟藍芽裝置安裝再電腦上

b. 超音波 SENSOR 安裝再機器人身上

c. 把 Lego 材料安裝在機器人身上

-輪子*3(兩個大輪子當作左右胎,小輪子當輔助腳)

-馬達*2(用來前進,後退,轉彎)

-綠色以及水藍色色紙+塑膠板(用來當作車頭.車尾之區分)

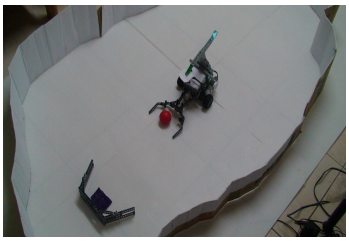
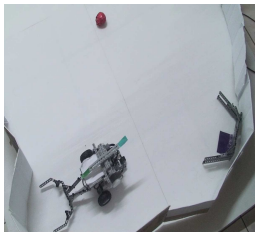
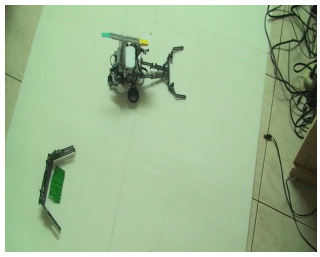
-彎曲塑膠零件數個,直線塑膠零件數個(組裝控球把手)

d. 紅色塑膠球放置任意點

e. 用 Lego 零件組成的球門放置任意點,並放上紫色色紙(用來給 Webcam 擷取圖片用以判別位置)

f.架上瓦楞紙做的圍牆跟鋪上四片瓦楞版做的草皮

Figure 3.1:Lego 足球機器人與場地全景圖*3

iii.開始主程式 C#,並利用 C#開啟視訊準備擷取照片並下達指令

Figure 3.2:執行主程式並開啟視訊準備要下達指令

3-2 為何進行實驗

目的:測試所寫之 Lego NXT 藍芽足球機器人是否可以執行下列功能

a. 能夠找球

b. 閃避障礙物

c. 當找到球把球控在自己身邊

d. 成功把球踢進球門

3-3 進行實驗與檢討

實驗 I:

目的:希望機器人能夠閃避障礙物並且找足球,最後把球順利的踢進球門

靜態實驗圖:12 張（觀看順序為由左到右,由上到下）











檢討與改進:

機器人雖然能夠順利把球踢進球門而且成功的閃避障礙物,可是筆者在圖片 12 時,是設定射門的時候自動把程式關掉,停止指令,所以沒有成功拍到射門鏡頭,這是因為射門的指令,所花的動作時間超過一秒,筆者設定的時間是 1 秒鐘擷取一張照片所以機器人是要在 1 秒鐘內接收完指令並且執行完動作,如果沒有辦法達成,會造成訊息無法傳進機器人,而機器人會一直重複執行上一秒所作之動作

解決方法:

i. 因為射門動作超過 1 秒,所以可以讓擷取圖片的時間加長,或者減少射門動作的時間,減少至 1 秒內。

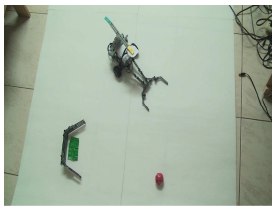
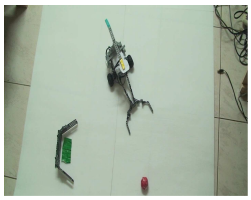
ii. 設定機器人收到射完門指令的同時結束程式(不是讓機器人射完門才結束程式),讓

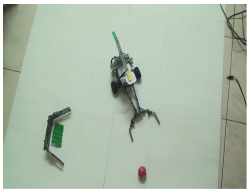
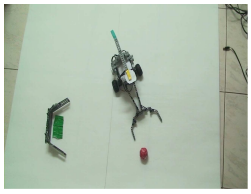
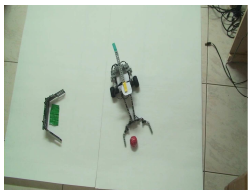
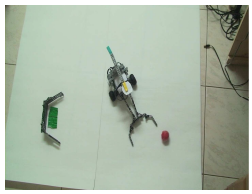
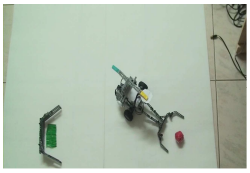
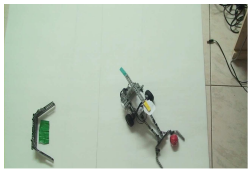
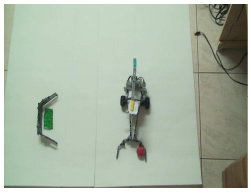
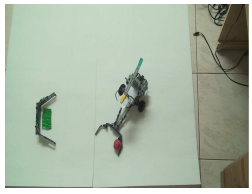
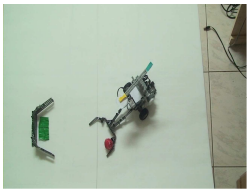
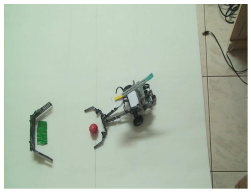
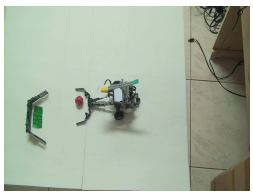
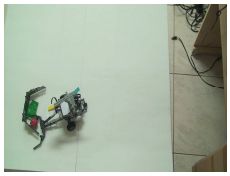
機器人不能在接收到下一秒指令。

實驗 II

目的:希望機器能夠把球踢進左邊球門，並且能夠順利的拍到射門的畫面，因為射門的動作仍超過一秒，所使用的方法為當機器人接受到射門的指令時，立刻把程式關掉，不會讓機器人收到下一秒指令

靜態實驗圖　16 張

檢討與改進:

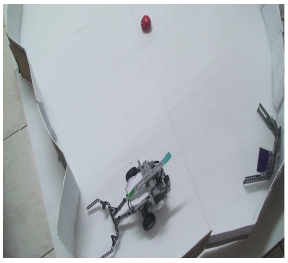
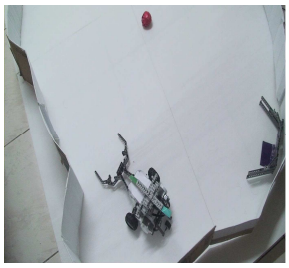
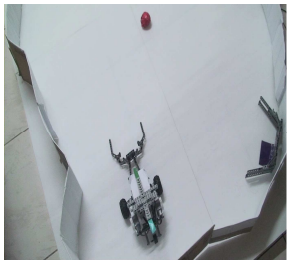
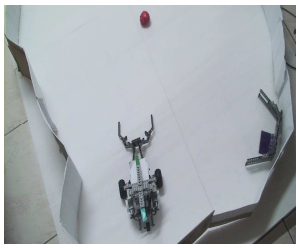
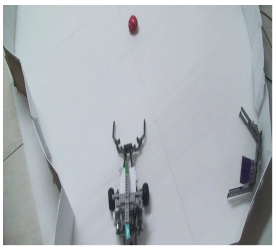
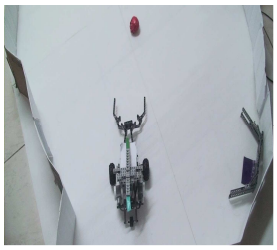
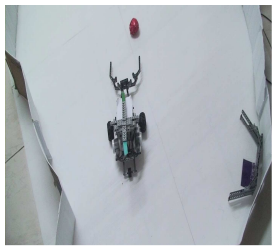
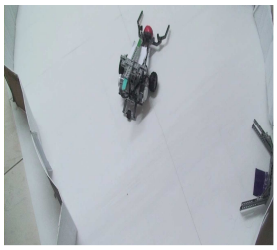
其實拍這部實驗影片,前 30 秒都卡在同一個點,之後才順利把球踢到球門

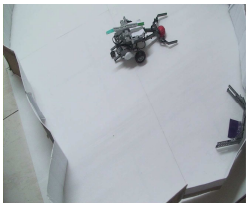
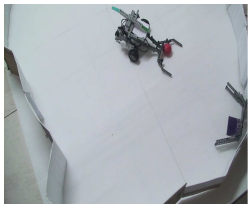
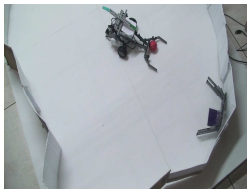
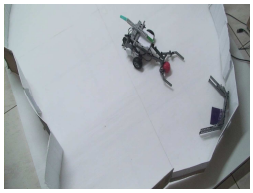
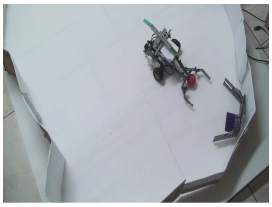
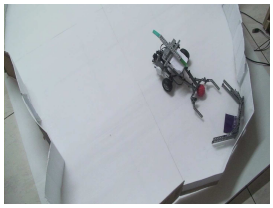
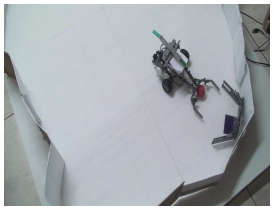
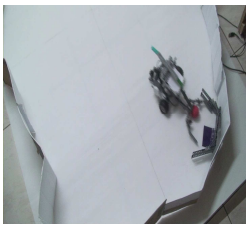
改進:下一個時間將加大機器人的馬力,並且加重球的重量使之不會卡點

實驗 III

目的:希望機器能夠把球踢進右邊球門，並且能夠順利的拍到射門的畫面

靜態實驗圖 16 張

檢討與改進:

機器人成功的找到球，並且能夠控球然後把球踢到球門，成功的地方在於轉彎的馬力增強了約 15,而且利用紅色色紙加重了紅色塑膠球的重量，使之更穩定推進球門

缺點：如果球的重量加太多，射門會不容易成功，必須斟酌。

(四)專題心得

I. 遇到的困難跟解決的方法

a. 機器人如何知道要拿到球

困難:因為機器人再 360 度跟球之絕對距離=((X1-X2)ˆ2+(Y1-Y2)ˆ2) ˆ ½),幾乎都完全不同

所以沒辦法用只有一個固定的絕對距離來讓機器人知道他拿到球

解決方法:用建表的方式把所有角度跟機器人與球的絕對距離一一記錄起來

Figure 4.1:計算車頭跟球之距離

```
//用來判斷紅球跟藍色區域的距離是否夠近
temp = temp2_X - temp1_X;
temp1 = temp2_Y - temp1_Y;
//(X^2+Y^2)^1/2
temp2 = System.Math.Sqrt(temp * temp + temp1 * temp1);
//車頭跟球之絕對距離
temp3 = (int)temp2;
temp8 = System.Math.Abs(temp1_X - temp2_X);
temp9 = System.Math.Abs(temp1_Y - temp2_Y);
```

Figure 4.2:判斷機器人是否拿到球

```
if (antiAngle < 30 && ((quadrant1 == 4 && quadrant2 == 4 && temp3 <= 50) || (quadrant1 == 3 && quadrant2 == 3 && temp3 <= 65
{
    label11.Text = "找球門";
    readyToGoal = 1;
}
```

優點:當建表完整度夠,就可以讓機器人順利把球推進球門

缺點:i.時間長

ii.換場地跟視訊裝置拉高高度就要重新寫

b.場地會因為燈光的因素而使球,球門,車頭,車尾的 RGB

變化,嚴重可能會使運算結果錯誤,導致很難把球踢進球門

解決方法:利用 ColorPix 軟體,在現場調整 RGB 既可

優點:可以使運算恢復

缺點:換場地就要重新算 RGB,耗時

c.場地可能造成機器人卡點

解決方法:買平滑場地地板(EX.壓克力板)

優點:不容易卡點

缺點:球可能容易甩出去(調弱機器人馬力既可)

II.學習心得

心得:能夠學到藍芽通訊協定跟影像處理跟機器人的 AI 實作,很開心,未來希望能夠繼續研究機器人 AI 的領域,雖然要計算的東西跟場地的狀況研究要思考,有時候 AI 並不是只靠單純的演算法推倒,而還要考慮環境會不會影響變數運算

==========

(五)總結與未來展望

寫完這次的樂高機器人踢足球,遇到的困難都要好好的思考,而不是土法鍊鋼的把條件寫死,像機器人跟場地會受燈光的因素,而造成 RGB 混亂,使運算錯誤,這可以用觀察顏色跟亮度的關係來寫出一套演算法來解決,可是因為時間問題無法完成,還有機器人跟球之絕對距離也可以用演算法算出圖中紅色足球的中心跟車頭黃色的中心,使絕對距離可以固定,這些未來應該好好思考。

(六)參考文獻

[1]樂高藍芽通訊協定: http://www.codeproject.com/KB/cs/nxtBluetooth.aspx

[2]視訊拍照影像處理
http://ppt.cc/ZWb0

[3]樂高所有運作之技巧 BLOG

http://www.blogger.com/profile/12376832463978326433

[4]OPENCV 安裝教學
http://www.dotblogs.com.tw/chou/category/2293.aspx?Show=All

[5]樂高論壇 1
http://note.tcc.edu.tw/515.html

[6]藍芽 BLOG
http://mey10101.pixnet.net/blog/post/4978803

[7]專門討論藍芽的論壇
http://it360.tw/forum/forum_posts.asp?TID=7556&KW=%E8%97%8D%E8%8A%BD

[8]OPENCV 影像處理論壇
http://www.opencv.org.cn/index.php

[9]Robot fun 機器人論壇
http://www.robofun.net/forum/forumdisplay.php?fid=68

致謝

感謝周老師的指導,周老師提供的演算法以及解決方法都很實用,能夠碰到機器人 AI 實作跟藍芽通訊協定以及影像處理這三個領域很充實。