Student number 10525757

Animal Visual Perception

In this report I will outline, describe and discuss the various ways I have used the owl robot to solve various computer vision-based tasks as well as use cameras to control the servos. To do this I will implement open cv libraries to complete my task.

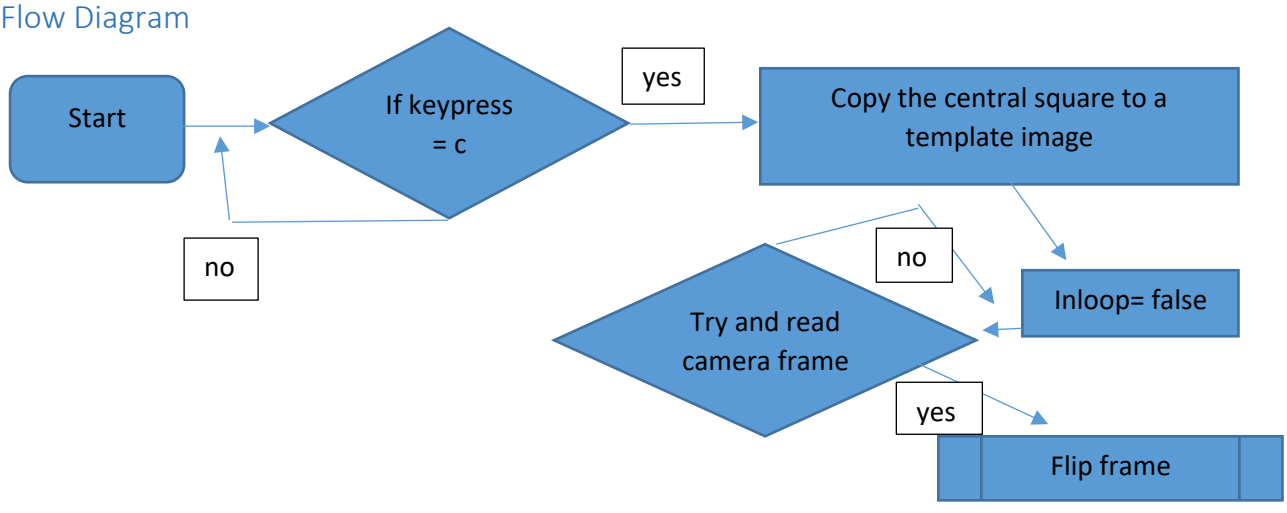
Contents

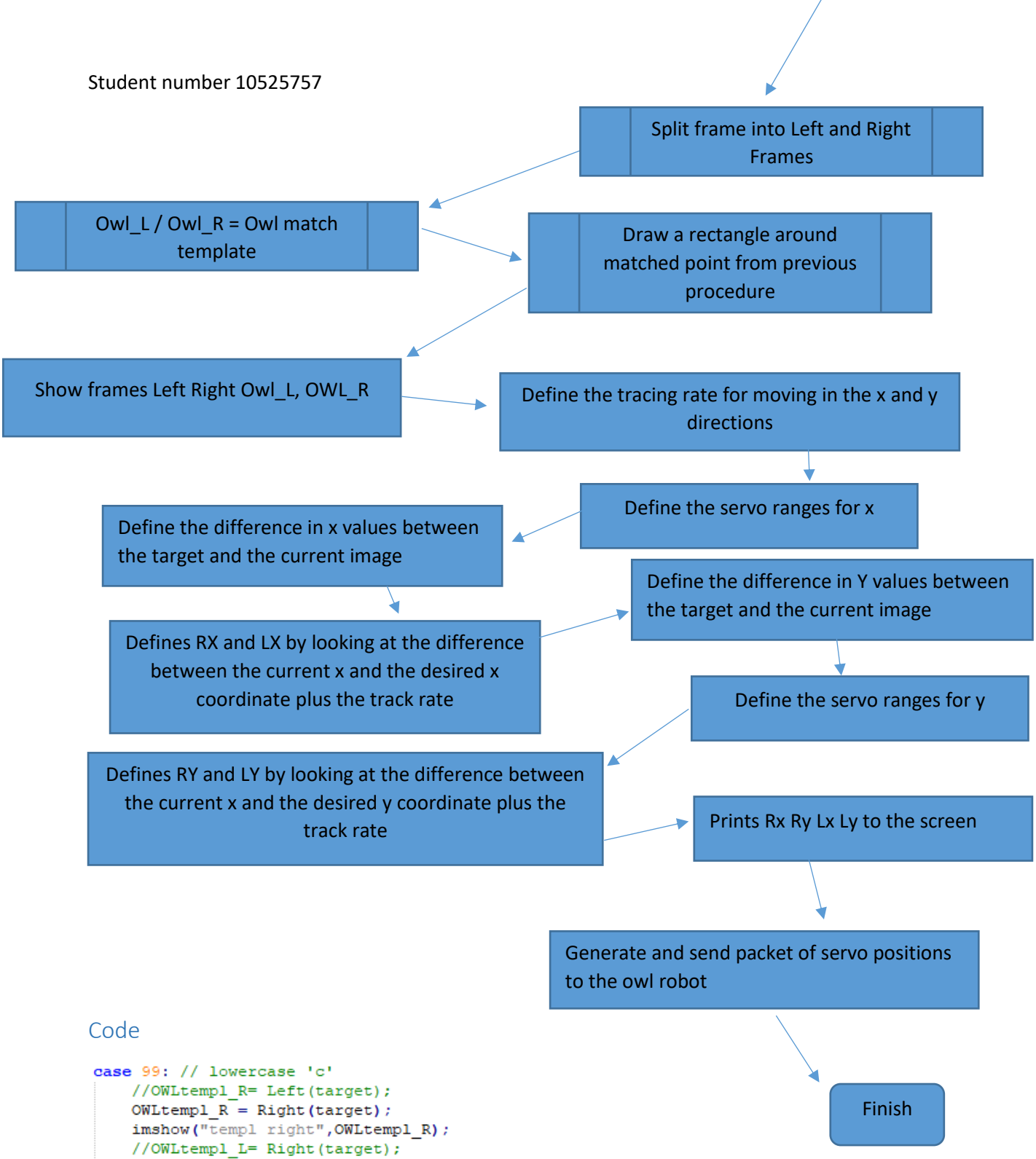
1. Verging onto a slow moving target with both eyes

Introduction

Vergence angle is very important when assessing distance using stereo cameras. In order to lock on to a target you need to make both cameras fixed. The angle to allow the cameras to be fixed to a common point is called a vergence angle. To do this I will use the keypress ‘c’ which enables matching the central frame to the live image stream and moves the eyes accordingly.

Flow Diagram





Code

```
case 99: // lowercase 'c'
    //OWLtempl_R= Left(target);
    OWLtempl_R = Right(target);
    imshow("templ right",OWLtempl_R);
    //OWLtempl_L= Right(target);
    Right.copyTo(LeftCopy);
    OWLtempl_L = LeftCopy(target);
    imshow("templ left",OWLtempl_L);
    waitKey(1);
    inLOOP=false; // quit loop and start tracking target
    break; // left
default:
    key=key;
    //nothing at present
}
```

```
inLOOP=true; // run through the loop until decided to exit
while (inLOOP) {
    if (!cap.read(Frame))
    {
        cout  << "Could not open the input video: " << source << endl;
        break;
    }
    Mat FrameFlpd; cv::flip(Frame,FrameFlpd,1); // Note that Left/Right are reversed now
    //Mat Gray; cv::cvtColor(Frame, Gray, cv::COLOR_BGR2GRAY);
    // Split into LEFT and RIGHT images from the stereo pair sent as one MJPEG iamge
    Left= FrameFlpd( Rect(0, 0, 640, 480)); // using a rectangle
    Right=FrameFlpd( Rect(640, 0, 640, 480)); // using a rectangle

    //Rect target= Rect(320-32, 240-32, 64, 64); //defined in owl-cv.h
    OwlCorrel OWL_L;
    OwlCorrel OWL_R;
    OWL_L = Owl_matchTemplate( Right,  Left, OWLtempl_L, target);
    OWL_R = Owl_matchTemplate( Left,   Right, OWLtempl_R, target);
    /// Show me what you got
    //Mat LeftCopy;
    //Left.copyTo(LeftCopy);
    rectangle( Left, OWL_L.Match, Point( OWL_L.Match.x + OWLtempl_L.cols , OWL_L.Match.y + OWLtempl_L.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( Right, OWL_R.Match, Point( OWL_R.Match.x + OWLtempl_R.cols , OWL_R.Match.y + OWLtempl_R.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( OWLresult_R, OWL_R.Match, Point( OWL_R.Match.x + OWLtempl_R.cols , OWL_R.Match.y + OWLtempl_R.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( OWLresult_L, OWL_L.Match, Point( OWL_L.Match.x + OWLtempl_L.cols , OWL_L.Match.y + OWLtempl_L.rows), Scalar::all(255), 2, 8, 0 );

    imshow("Owl-L", Left);
    imshow("Owl-R", Right);
    imshow("Correl",OWL_L.Result );
```

```
imshow("Correl",OWL_R.Result );
if (waitKey(10)== 27) inLOOP=false;
```

Conclusions

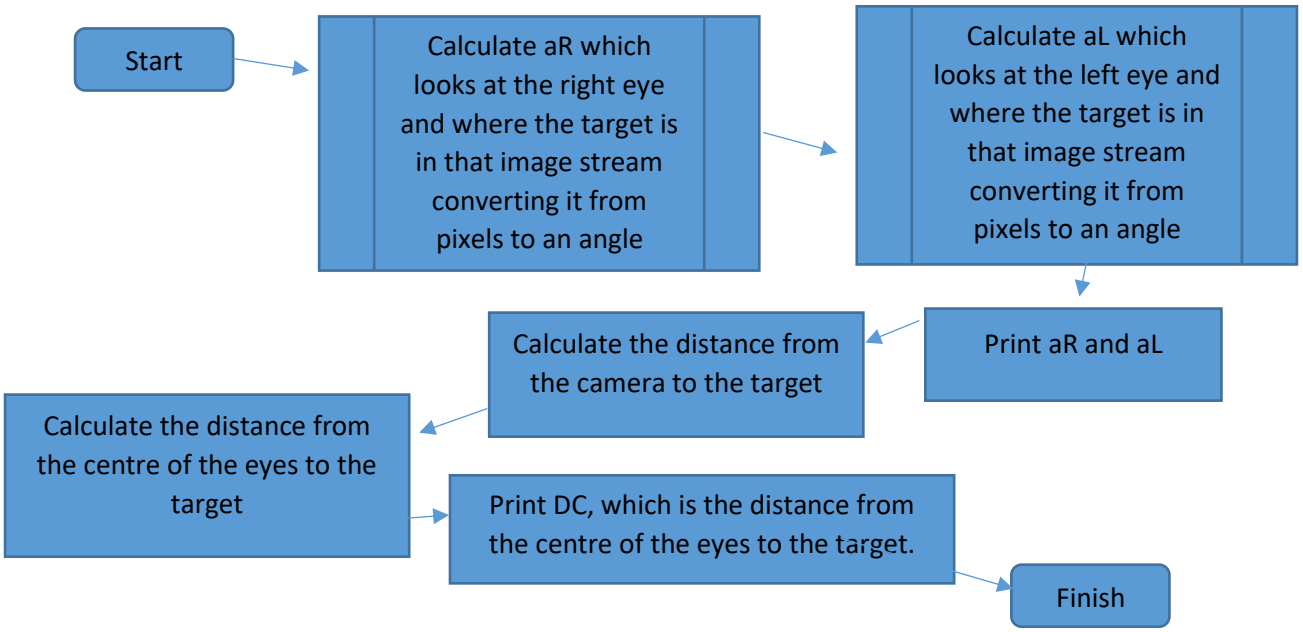
The owl is able to track a target with reasonable accuracy while in its field of view whilst utilising motion of the servo controllers. This enables the owl to track something very well, the downside is that it will not be able to control the servos if the target goes off out of the camera's field of view. This causes the cameras to move erratically around and fail to continue to track even if the target is present again.

2. Estimating the distance to a target

Introduction

Distance calculation requires that you track a target with both cameras. The first step is to look at where the servos are pointing which enables you to get a rough direction of the target. The next step is to evaluate where the target box is drawn in the field of view then servo position is able to be measured.

Flow Diagram



Student number 10525757

This section of the code is for tracing a target in its field of view.

```
inLOOP=true; // run through the loop until decided to exit
while (inLOOP) {
    if (!cap.read(Frame))
    {
        cout  << "Could not open the input video: " << source << endl;
        break;
    }
    Mat FrameFlpd; cv::flip(Frame,FrameFlpd,1); // Note that Left/Right are reversed now
    //Mat Gray; cv::cvtColor(Frame, Gray, cv::COLOR_BGR2GRAY);
    // Split into LEFT and RIGHT images from the stereo pair sent as one MJPEG iamge
    Left= FrameFlpd( Rect(0, 0, 640, 480)); // using a rectangle
    Right=FrameFlpd( Rect(640, 0, 640, 480)); // using a rectangle

    //Rect target= Rect(320-32, 240-32, 64, 64); //defined in owl-cv.h
    //Mat OWLtempl(Right, target);
    OwlCorrel OWL_L;
    OwlCorrel OWL_R;
    OWL_L = Owl_matchTemplate( Right,  Left, OWLtempl_L, target);
    OWL_R = Owl_matchTemplate( Left,   Right, OWLtempl_R, target);
    /// Show me what you got
    //Mat LeftCopy;
    //Left.copyTo(LeftCopy);
    rectangle( Left, OWL_L.Match, Point( OWL_L.Match.x + OWLtempl_L.cols , OWL_L.Match.y + OWLtempl_L.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( Right, OWL_R.Match, Point( OWL_R.Match.x + OWLtempl_R.cols , OWL_R.Match.y + OWLtempl_R.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( OWLresult_R, OWL_R.Match, Point( OWL_R.Match.x + OWLtempl_R.cols , OWL_R.Match.y + OWLtempl_R.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( OWLresult_L, OWL_L.Match, Point( OWL_L.Match.x + OWLtempl_L.cols , OWL_L.Match.y + OWLtempl_L.rows), Scalar::all(255), 2, 8, 0 );

    imshow("Owl-L", Left);
    imshow("Owl-R", Right);
    imshow("Correl",OWL_L.Result );
```

The next section looks at the servo positions as well as the where in the image the matched template is found. This will then be able to print the distance as DC in the code.

```
double aR;
double aL;
double Dl;
double Dc;
```

```
aR = 160 - (((Rx+OWL_R.Match.x) - RxLm)/8.125);
aL = ((Lx+OWL_L.Match.x) - LxLm)/8.125;
//aL = (Lx - LxLm)/4.375;
cout << aR << "right angle - " << OWL_R.Match.x << endl;
cout << aL << "left angle - " << OWL_L.Match.x << endl;

Dl = (67*cos(aR))/sin(aL - aR);
cout << Dl << "DL distance" << endl;
Dc = sqrt(pow(Dl,2) + (pow(67,2)/4)- Dl * 67 * sin(90-aL));
cout << Dc << "final distance" << endl;
```

```
imshow("Correl",OWL_R.Result );
if (waitKey(10)== 27) inLOOP=false;

double KPx=0.1; // track rate X
double KPy=0.1; // track rate Y
double RxScaleV = RxRangeV/(double)640; //PWM range /pixel range
double Xoff_R= 320-(OWL_R.Match.x + OWLtempl_R.cols)/RxScaleV ; // compare to centre of image
int RxOld=Rx;
double LxScaleV = LxRangeV/(double)640; //PWM range /pixel range
double Xoff_L= 320-(OWL_L.Match.x + OWLtempl_L.cols)/LxScaleV ; // compare to centre of image
int LxOld=Lx;

Rx=RxOld-Xoff_R*KPx; // roughly 300 servo offset = 320 [pixel offset
Lx=LxOld-Xoff_L*KPx; // roughly 300 servo offset = 320 [pixel offset


double RyScaleV = RyRangeV/(double)480; //PWM range /pixel range
double Yoff_R= (250-(OWL_R.Match.y + OWLtempl_R.rows)/RyScaleV)*KPy ; // compare to centre of image
int RyOld=Ry;
Ry=RyOld+Yoff_R; // roughly 300 servo offset = 320 [pixel offset
double LyScaleV = LyRangeV/(double)480; //PWM range /pixel range
double Yoff_L= (250+(OWL_L.Match.y + OWLtempl_L.rows)/LyScaleV)*KPy ; // compare to centre of image
int LyOld=Ly;
Ly=LyOld-Yoff_L; // roughly 300 servo offset = 320 [pixel offset

cout << Rx << " " << Xoff_R << " " << RxOld << endl;
cout << Ry << " " << Yoff_R << " " << RyOld << endl;
cout << Lx << " " << Xoff_L << " " << LxOld << endl;
cout << Ly << " " << Yoff_L << " " << LyOld << endl;
```

Conclusions

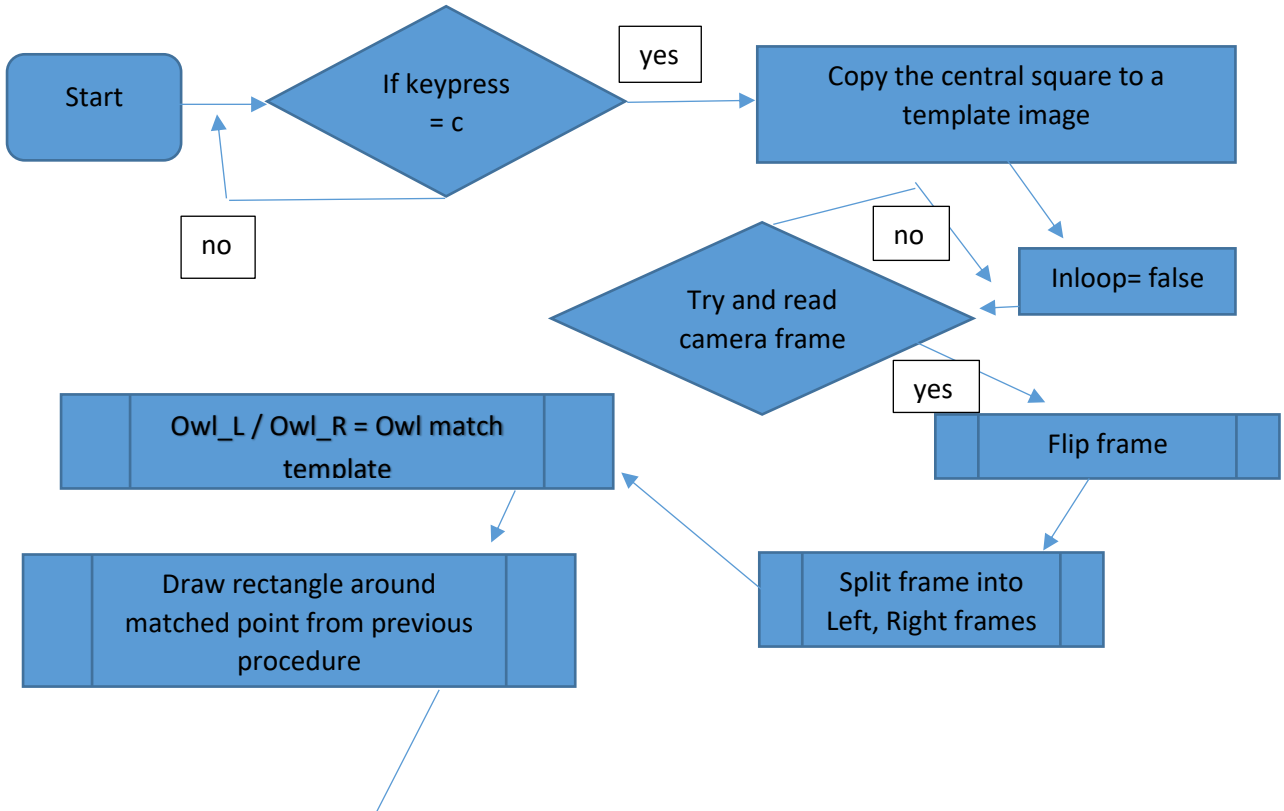
This method enables the user to approximate the distance of an object, however you will get errors carried over because the servos are not totally accurate. Another source of error is the mis-matching of the target boxes, however this will produce a distance measurement fit for purpose.

3. Track a target for up to ten seconds

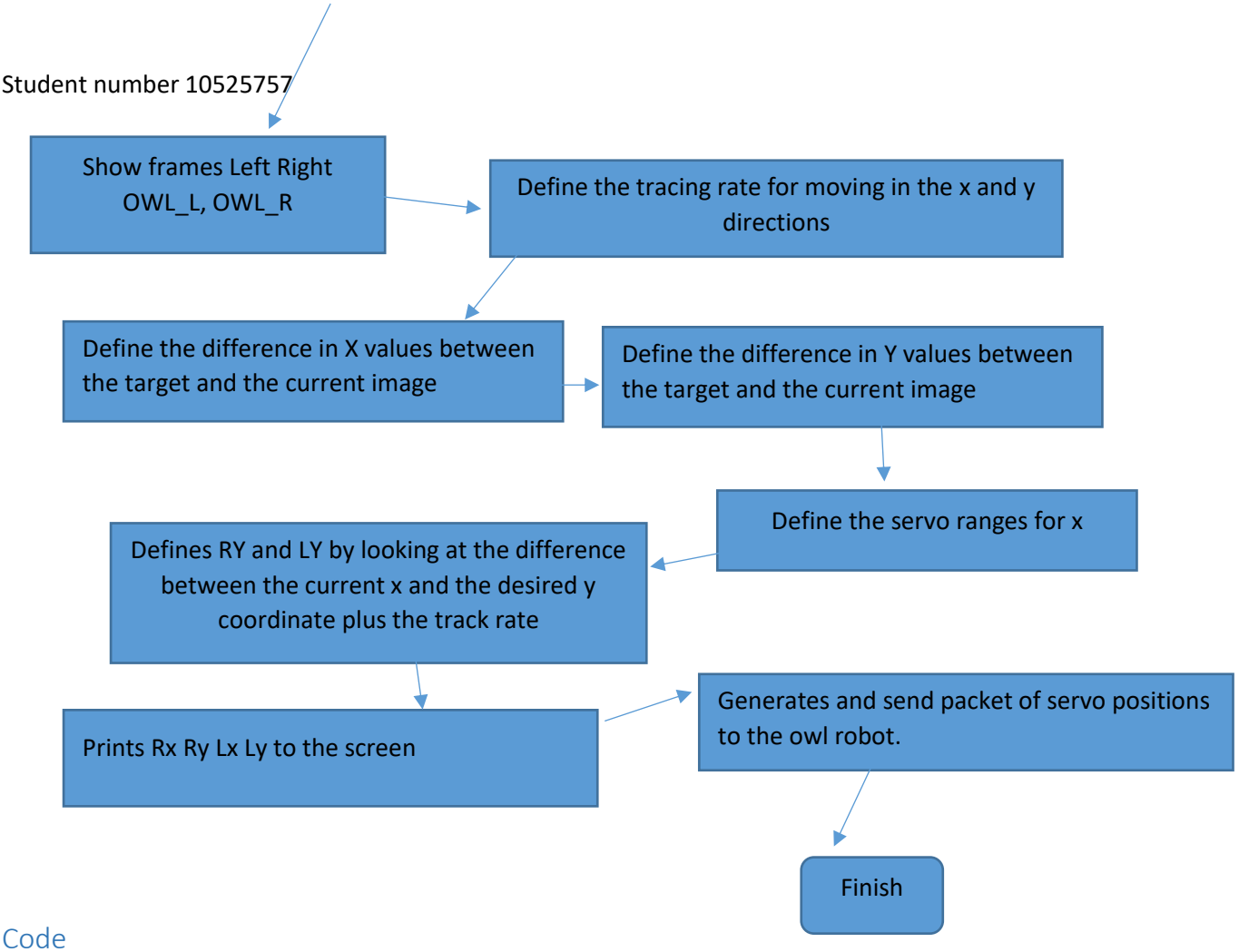
Introduction

Tracking a target is a very human and animal trait, your eyes are naturally able to not only estimate speed and distance of many targets but also able to predict their position. Animals can also do this incredibly well predicting when to strike. The following code is a very basic version of what an animal can do by matching a target to the current image, you can see the direction of travel and follow it.

Flow Diagram



Student number 10525757



Code

```
case 99: // lowercase 'c'
    //OWLtempl_R= Left(target);
    OWLtempl_R = Right(target);
    imshow("templ right",OWLtempl_R);
    //OWLtempl_L= Right(target);
    Right.copyTo(LeftCopy);
    OWLtempl_L = LeftCopy(target);
    imshow("templ left",OWLtempl_L);
    waitKey(1);
    inLOOP=false; // quit loop and start tracking target
    break; // left
```

```
inLOOP=true; // run through the loop until decided to exit
while (inLOOP) {
    if (!cap.read(Frame))
    {
        cout  << "Could not open the input video: " << source << endl;
        break;
    }
    Mat FrameFlpd; cv::flip(Frame,FrameFlpd,1); // Note that Left/Right are reversed now
    //Mat Gray; cv::cvtColor(Frame, Gray, cv::COLOR_BGR2GRAY);
    // Split into LEFT and RIGHT images from the stereo pair sent as one MJPEG iamge
    Left= FrameFlpd( Rect(0, 0, 640, 480)); // using a rectangle
    Right=FrameFlpd( Rect(640, 0, 640, 480)); // using a rectangle

    //Rect target= Rect(320-32, 240-32, 64, 64); //defined in owl-cv.h
    //Mat OWLtempl(Right, target);
    OwlCorrel OWL_L;
    OwlCorrel OWL_R;
    OWL_L = Owl_matchTemplate( Right,  Left, OWLtempl_L, target);
    OWL_R = Owl_matchTemplate( Left,   Right, OWLtempl_R, target);
    /// Show me what you got
    //Mat LeftCopy;
    //Left.copyTo(LeftCopy);
    rectangle( Left, OWL_L.Match, Point( OWL_L.Match.x + OWLtempl_L.cols , OWL_L.Match.y + OWLtempl_L.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( Right, OWL_R.Match, Point( OWL_R.Match.x + OWLtempl_R.cols , OWL_R.Match.y + OWLtempl_R.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( OWLresult_R, OWL_R.Match, Point( OWL_R.Match.x + OWLtempl_R.cols , OWL_R.Match.y + OWLtempl_R.rows), Scalar::all(255), 2, 8, 0 );
    rectangle( OWLresult_L, OWL_L.Match, Point( OWL_L.Match.x + OWLtempl_L.cols , OWL_L.Match.y + OWLtempl_L.rows), Scalar::all(255), 2, 8, 0 );

    imshow("Owl-L", Left);
    imshow("Owl-R", Right);
    imshow("Correl",OWL_L.Result );
```

```
imshow("Correl",OWL_R.Result );
if (waitKey(10)== 27) inLOOP=false;

double KPx=0.1; // track rate X
double KPy=0.1; // track rate Y
double RxScaleV = RxRangeV/(double)640; //PWM range /pixel range
double Xoff_R= 320-(OWL_R.Match.x + OWLtempl_R.cols)/RxScaleV ; // compare to centre of image
int RxOld=Rx;
double LxScaleV = LxRangeV/(double)640; //PWM range /pixel range
double Xoff_L= 320-(OWL_L.Match.x + OWLtempl_L.cols)/LxScaleV ; // compare to centre of image
int LxOld=Lx;

Rx=RxOld-Xoff_R*KPx; // roughly 300 servo offset = 320 [pixel offset
Lx=LxOld-Xoff_L*KPx; // roughly 300 servo offset = 320 [pixel offset


double RyScaleV = RyRangeV/(double)480; //PWM range /pixel range
double Yoff_R= (250-(OWL_R.Match.y + OWLtempl_R.rows)/RyScaleV)*KPy ; // compare to centre of image
int RyOld=Ry;
Ry=RyOld+Yoff_R; // roughly 300 servo offset = 320 [pixel offset
double LyScaleV = LyRangeV/(double)480; //PWM range /pixel range
double Yoff_L= (250+(OWL_L.Match.y + OWLtempl_L.rows)/LyScaleV)*KPy ; // compare to centre of image
int LyOld=Ly;
Ly=LyOld-Yoff_L; // roughly 300 servo offset = 320 [pixel offset

cout << Rx << " " << Xoff_R << " " << RxOld << endl;
cout << Ry << " " << Yoff_R << " " << RyOld << endl;
cout << Lx << " " << Xoff_L << " " << LxOld << endl;
cout << Ly << " " << Yoff_L << " " << LyOld << endl;
```

Conclusions

From the code already made in the given code I have added the ability to track targets with both eyes separately. This is using the fact that I am using the target box with each camera and then running the owl match template procedure for each camera on the eye robot and generating separate x and y coordinate in the image of the owl robot. The implementation of this code is demonstrated by the video below.

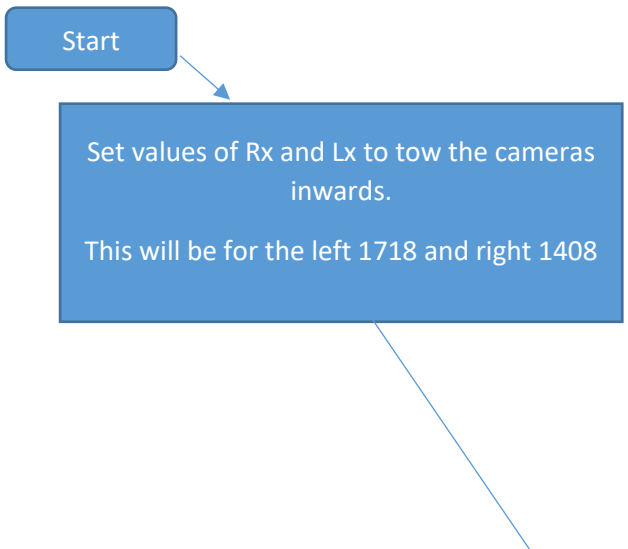
https://www.youtube.com/watch?v=Bbsjc8ZlbD0

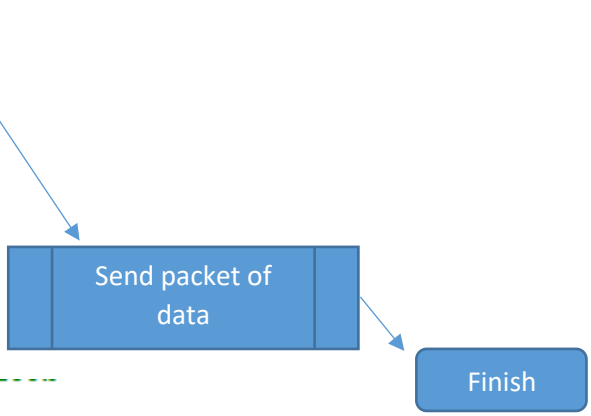
4. Calibrate cameras in orthographic mode

Introduction

Orthographic mode is a term used by artists and engineers as an easy way of drawing a 3 dimensional shape. It is used to add perspective to drawings. Orthographic projection is a means of representing a 3d object in a two dimensional view. Orthographic projections are different in the fact that the axes are approximately 120 degrees apart and the vertical lines on a drawing always remain vertical.

Flow Diagram





Code

```
case 122://
    Rx = 1718;
    Lx = 1408;
    Ry = 1560;
    Ly = 1560;

    CMDstream.str("");
    CMDstream.clear();
    CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
    CMD = CMDstream.str();
    RxPacket= OwlSendPacket (u_sock, CMD.c_str());
```

Conclusions

Orthographic mode is a great way to view an object and really start to understand how a target is moved into place. In the next sections when I look into disparity, depth maps and scene projection this will become essential to produce something with better accuracy.

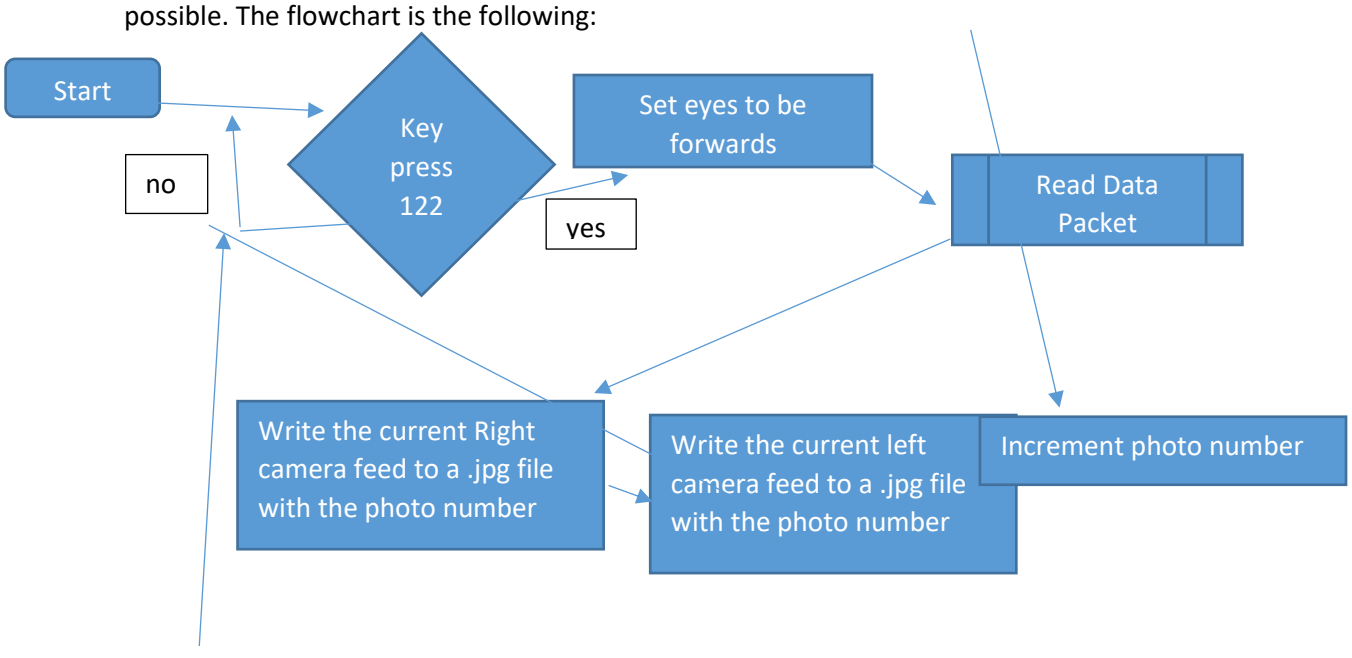
5. Calculate disparity of a target

Introduction

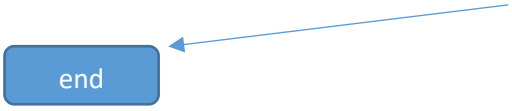
Similar to the previous section where it looks at the orthographic projection instead the disparity runs on taking a series of 20 images with both of the cameras and then map the distance to every target. This will not only look for the perspective of the eyes and the positioning of the images but also the location of the grid on the images. By using a series of images and mapping the board it is able to transport the two cameras into a depth stream. Not just by looking at the differences in the board position but also looking how large the board is largely because the size of the board and the red shapes remain constant during the calibration process and the matching process which enables the depth streams to be more accurate.

Flow Diagram

To take the images I am have chosen to use code which enables the user to take as many photos as possible. The flowchart is the following:



Student number 10525757



The following code is used to take the photos for the owl correlation calibration:

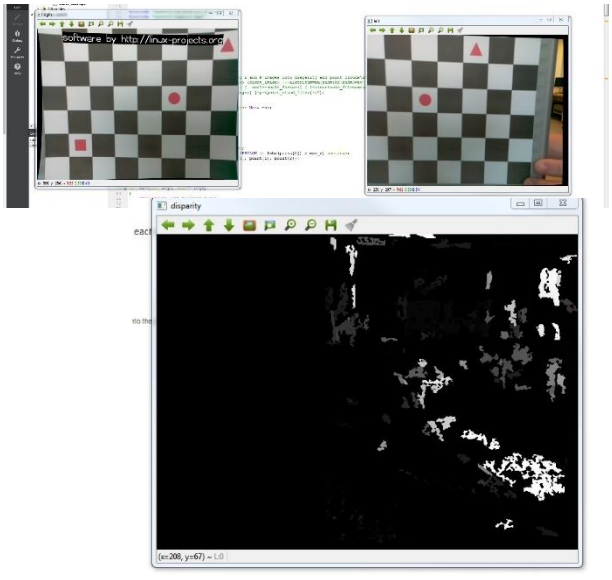
Code

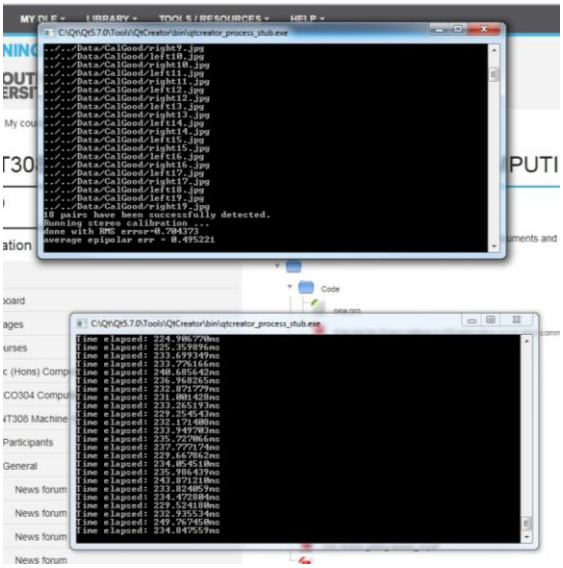
```
        case 122://
            Rx = 1550;//1525
            Lx = 1550;//1575
            Ry = 1390;
            Ly = 1595;

            CMDstream.str("");
            CMDstream.clear();
            CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
            CMD = CMDstream.str();
            RxPacket= OwlSendPacket (u_sock, CMD.c_str());
            waitKey(20);

            image = Right;
            imwrite( "C:/Repository/imgs/rightimg"+ std::to_string(i) +".jpg", image );

            image = Left;
            imwrite( "C:/Repository/imgs/leftimg"+ std::to_string(i) +".jpg", image );

            i++;
            break;
**********************end calibrate and take picture
```

Stereo_callib.cpp and stereo_match.cpp and the code is the same as the one prepared previously and was given to me in the course material and the first calibrates the cameras for this individual owl robot. The second match program uses the Stereo_callib.cpp and produces a depth stream from using these two cameras.

Conclusions

The following images shows at the left and right frames and then below that is the disparity map produced by the process.





6. Produce a depth map and calibration readings

Introduction

In the previous section you saw how I was able to create a depth stream from the two cameras. This is done using OpenCV application in order to look at the differences in the pixels. This is of course by the programs provided. I will first look at the actual value of the graph before going on to look at the shape of the graphs made along with their equations.

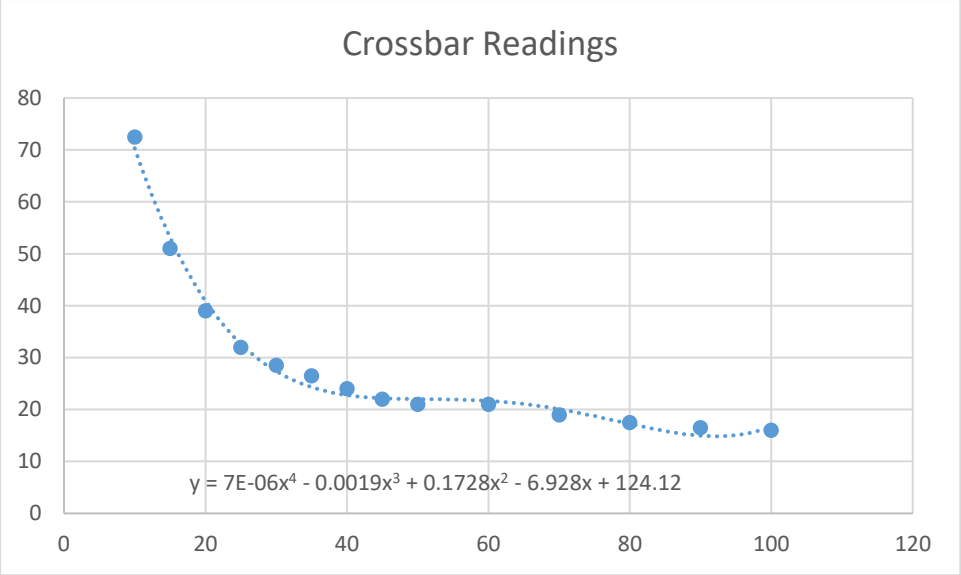
The following table shows different depth readings. The first is looking at the cross bar readings basically hovering over the cross bar and reading the result. The second looks at hovering over the square on the box and the rounding values '—' in the table represents where the data meaning the object cannot be seen in the image. The final Colum represent the values of the vertical bars on the image.

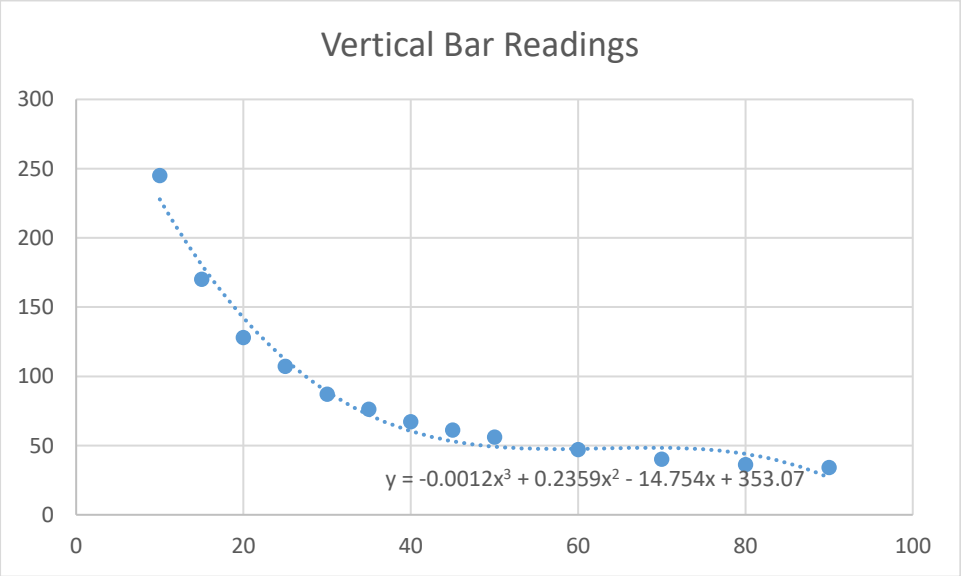
Conclusions

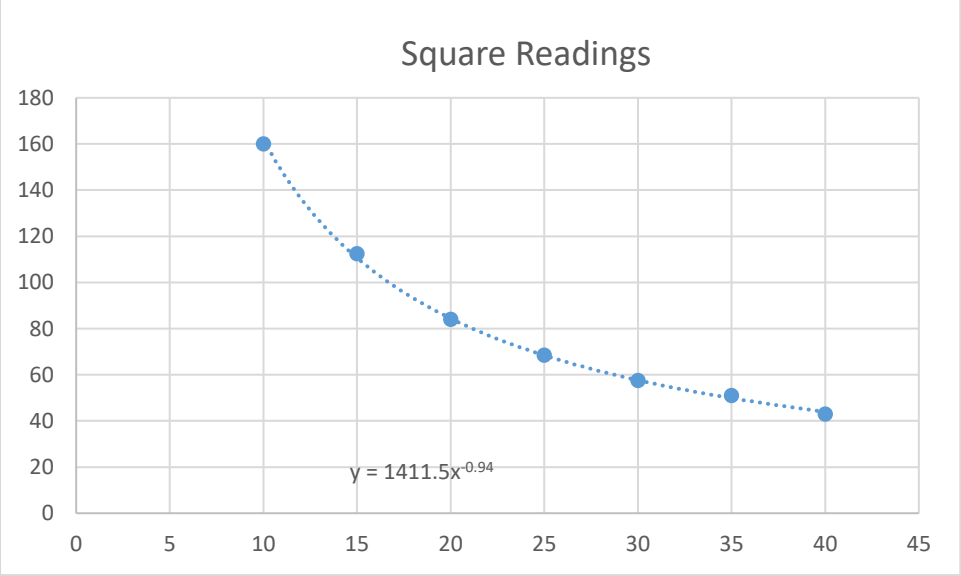
number	actual distance	cross bar distance	rounding	squares	rounding	vertical bars
1	5	--	-	--	--	--
2	10	72/73	72.5	160	160	245
3	15	51	51	112/113	112.5	170
4	20	39	39	84	84	128
5	25	32	32	68/69	68.5	107
6	30	28/29	28.5	57/58	57.5	87
7	35	26/27	26.5	51	51	76
8	40	24	24	43/44	43	67
9	45	22	22	--	--	61
10	50	21	21	--	--	56
11	60	21	21	--	--	47
12	70	19	19	--	--	40
13	80	18/17	17.5	--	--	36
14	90	17/16	16.5	--	--	34
15	100	16	16	--	--	---

I then have made graphs comparing the various visible parts of the box to the actual values I have recoded from a tape measures the graphs are the following:

Student number 10525757







As you can see from the graph and the table reading from the square was by far the leas visible after 40cm. However it is also the closest reading from the short time period which follows a straight line. Although the vertical bars are visible for a lot longer than the square, the horizontal bars do provide a lot more accuracy in terms of looking at the depth readings. Looking at the readings in terms of what the algorithm provides you get 3 formulas for converting between the two.

For horizontal bar reading the equation is: $Y = 7E\text{-}06x^4 - 0.0019x^3 + 0.1728x^2 - 6.928x + 124.12$

Square reading the equation is: $y = 1411.5x^{-0.94}$

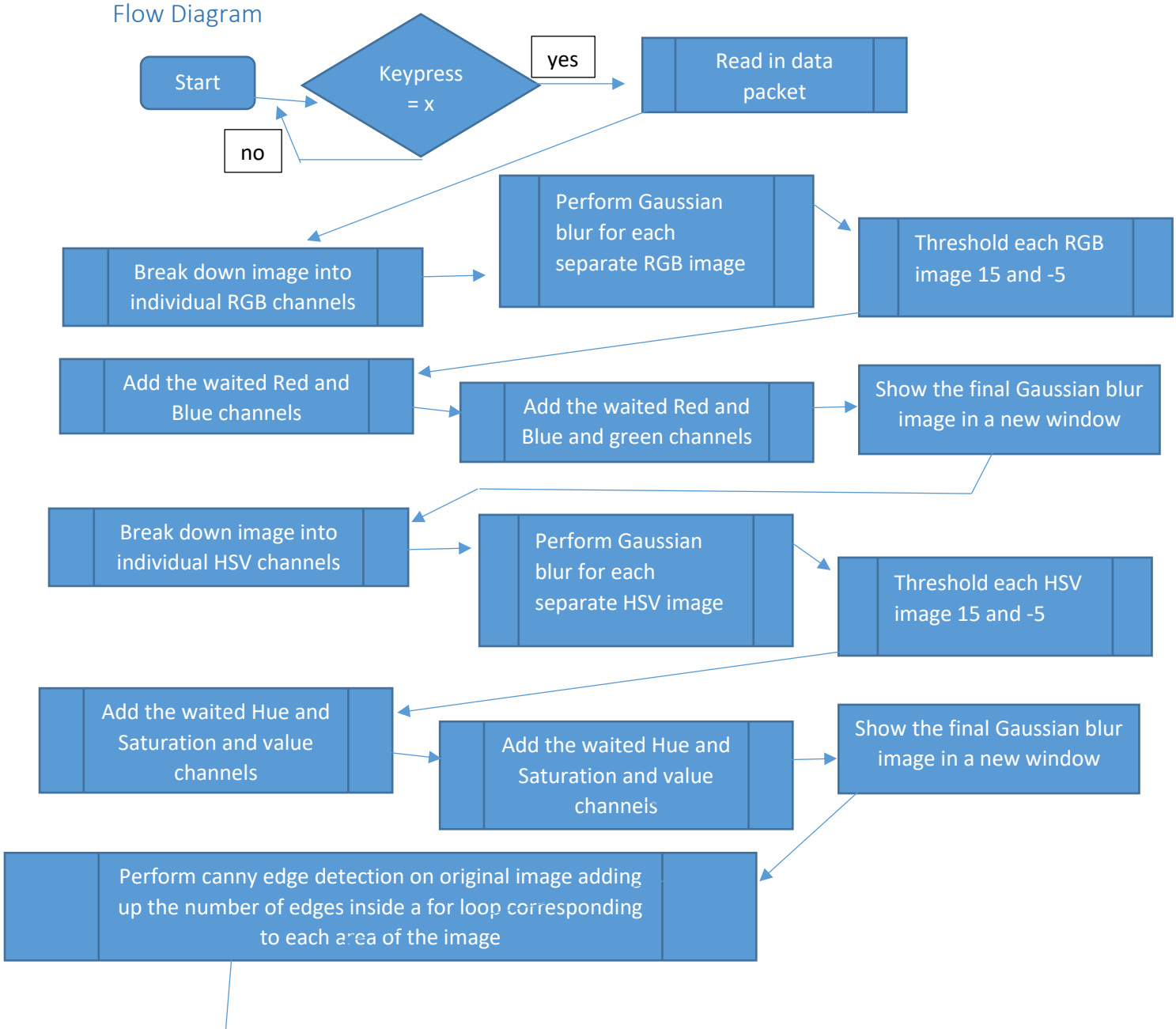
For vertical bar reading the equation is: $y = -0.0012x^3 + 0.2359x^2 - 14.754x + 353.07$

7. Developing a processing model and apply it to a scene

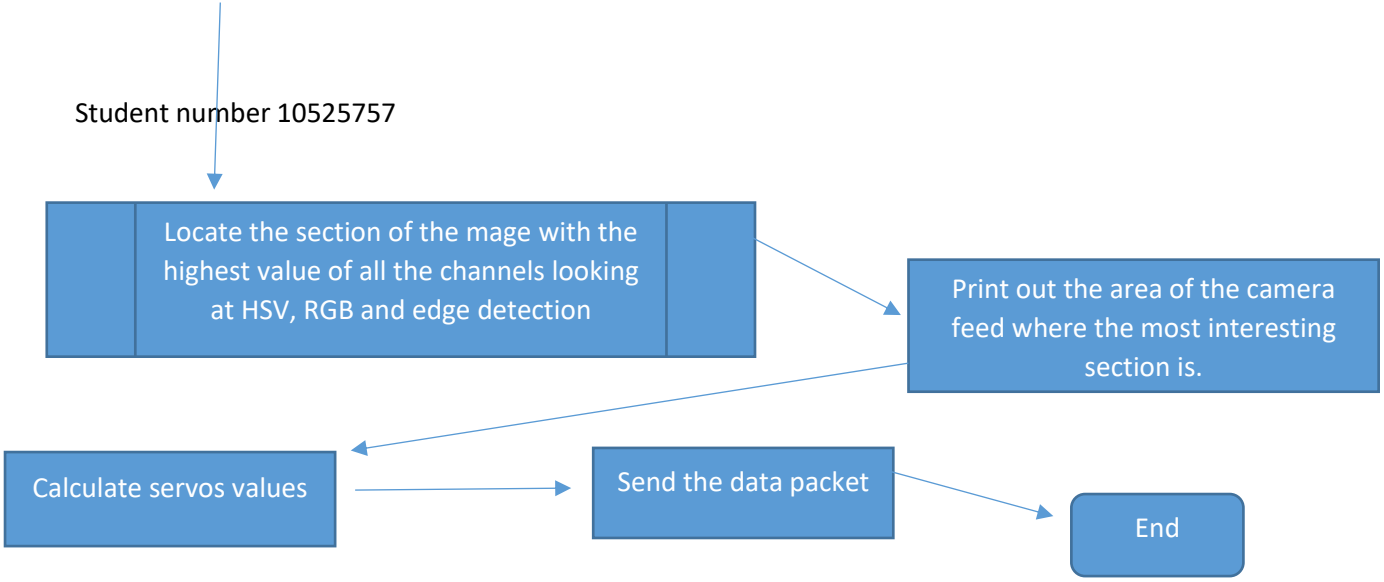
Introduction

I this section I plan to use various OpenCV tools to process the scene from looking at the separate RGB images to see if an image has a large amount of any one colour to using HSV to look at the particular brightness of any one section of the area to see if any stand out or to look at large amounts of motion of and target in the field of view. Any number of ways to process a scene in this section I hope to declare interest points to investigate further at a later date.

Flow Diagram



Student number 10525757



Code

```
case 120: // case x detecting interesting features ------------------------------------------------------------------
    inLOOP=true; // run through the loop until decided to exit
    destroyAllWindows();

    while (inLOOP) {
        if (!cap.read(Frame))//reads and image frame
        {
            cout  << "Could not open the input video: " << source << endl;
            break;
        }
        Mat FrameFlpd; cv::flip(Frame,FrameFlpd,1); // Note that Left/Right are reversed now

        // Split into LEFT and RIGHT images from the stereo pair sent as one MJPEG iamge
        Left= FrameFlpd( Rect(0, 0, 640, 480)); // creates the image Left by not only
        //splitting the camera frame recieved by the camera but also flipt
        Right=FrameFlpd( Rect(640, 0, 640, 480)); // creates the image Right by not only
        //splitting the camera frame recieved by the camera but also flipt

        LeftCopy = Left;// method uses a copy of the camera image not a live stream
        RightCopy = Right;

        imshow("Owl-L", LeftCopy);//shows the image of the left eyes owl camera
        imshow("Owl-R", RightCopy);//shows the image of the Right eyes owl camera

        Rect observe_area; //is the rectangle for the target area to observe
        vector<Mat> channels;
        image = RightCopy;   //copys the right frame into a mat image to enable us to process the image

        split(image, rgb_breakdown);  // this takes the image and breaks it down into its individual bgr chanles

        // use Gaussian Blur to blur out the contours and focus on the colour
        GaussianBlur(rgb_breakdown[0], rgb_breakdown[0], Size(), 4, 1, BORDER_DEFAULT );//represents the red image
        GaussianBlur(rgb_breakdown[1], rgb_breakdown[1], Size(), 4, 1, BORDER_DEFAULT );//represents the green image
        GaussianBlur(rgb_breakdown[2], rgb_breakdown[2], Size(), 4, 1, BORDER_DEFAULT );// represents the Blue image

    /*waitKey(20);  // used for testing
    imshow("Red",rgb_breakdown[0]);
    waitKey(20);
    imshow("Green",rgb_breakdown[1]);
    waitKey(20);
    imshow("Blue",rgb_breakdown[2]);
    waitKey(20);*/

    //adaptive thresholding is a way of looking for only the highist amound of color in the image
    adaptiveThreshold(rgb_breakdown[0], rgb_breakdown[0],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);//thresholds the red image
    adaptiveThreshold(rgb_breakdown[1], rgb_breakdown[1],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);//thresholds the green image
    adaptiveThreshold(rgb_breakdown[2], rgb_breakdown[2],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);//thresholds the blue image

    addWeighted(rgb_breakdown[0],0.7,rgb_breakdown[1],0.3,0, gblur);// this is how to combine the red and green images after the
    //  threshold has been applied
    addWeighted(gblur,0.7,rgb_breakdown[2],0.3,0,gblur);// this adds the thresholded blue image to the previos combined image

    imshow("RGB GaussianBlur",gblur); // shows the completed RGB image showing the areas of the most color only

    cvtColor(image, image_HSV, cv::COLOR_RGB2BGR);//converts the RGB image to Hue, Saturation, Brightness

    channels.clear();
    channels.resize(image_HSV.channels());  //resize channels
    cv::split(image_HSV, &channels[0]);// splits the images into its individual channels

    /*waitKey(20); // used for testing
    imshow("Hue",channels[0]);
    waitKey(20);
    imshow("Saturation",channels[1]);
    waitKey(20);
    imshow("Lightness",channels[2]);
    waitKey(20);*/
```

Student number 10525757

```
//adaptive thresholding is a way of looking for only the highist amound of color in the image
adaptiveThreshold(channels[0], channels[0],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);// thresholds the Hue in the image
adaptiveThreshold(channels[1], channels[1],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);// thresholds the Saturation in the image
adaptiveThreshold(channels[2], channels[2],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);// thresholds the Lightness in the image

addWeighted(channels[0],0.7,channels[1],0.3,0, dst);// combines the Hue and Saturation channels after they have been thresholded
addWeighted(dst,0.7,channels[2],0.3,0,dst);// combines the Hue, Saturation and Lightness channels after they have been thresholded

imshow("HSV adaptiveThreshold",dst);// displays the combined HSV image after it has been thresholded

Canny(image, edge_det, 50, 200, 3);//performs canny edge detection of the original image

vector<Vec2f> lines1;
HoughLines(edge_det, lines1, 1, CV_PI/180, 100, 0, 0 );
for( size_t i = 0; i < lines1.size(); i++ ){
    float rho = lines1[i][0], theta = lines1[i][1];
    Point pt1, pt2;
    double a = cos(theta), b = sin(theta);
    double x0 = a*rho, y0 = b*rho;
    pt1.x = cvRound(x0 + 1000*(-b));//declare the first x cordernate of the line
    pt1.y = cvRound(y0 + 1000*(a));//declares the first y cordernate of the line
    pt2.x = cvRound(x0 - 1000*(-b));//declare the last x cordernate of the line
    pt2.y = cvRound(y0 - 1000*(a));//declares the last y cordernate of the line
    line( edge_det2, pt1, pt2, Scalar(0,0,255), 3, CV_AA);//draws the line
}

vector<Vec4i> lines2;
HoughLinesP(edge_det, lines2, 1, CV_PI/180, 50, 50, 10 );
for( size_t i = 0; i < lines2.size(); i++ ){
    Vec4i l = lines2[i];
    line( edge_det2, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0,0,255), 3, CV_AA);
}


waitKey(1000);
imshow("detected lines", edge_det);
waitKey(20);

int lx = 0;// looks at the horizontal position of the image the algorithm is useing
int ly = 0;// looks at the vertical position of the image the algorithm is useing
int y = 1;//counter for the next loop on the position of the y cordernate
int x = 1;//counter for the next loop on the position of the x cordernate
highest_score = 0;//area of the image with the highest overall value

for(int j = 0; j <image_grouping;j++){
    while(y <= rect_locations_y[ly])
    {
        while(x <= rect_locations_x[lx])
        {
            //gets the rgb value of the selected pixel for processing
            Vec3b colour = gblur.at<Vec3b>(Point(x, y));
            Vec3b contrast = dst.at<Vec3b>(Point(x, y));
            Vec3b edgedet = edge_det.at<Vec3b>(Point(x, y));

            //looks for the largest value in terms of RGB for any one element indicating a strong presence on any one color
            if(colour[0] >=150 or colour[1] >=150 or colour[2] >=150){
                combined_image_score[0] = combined_image_score[0]+1;
            }
            //looks for the largest value in terms of HSV for any one element indicating a strong presence of any one color
            if(contrast[0] >=150 or contrast[1] >=150 or contrast[2] >=150){
                combined_image_score[1] = combined_image_score[1]+1;
            }
            // looks for the most edges in any one section of the images indicating large changes in the image
            if(edgedet[0] >=150 or edgedet[1] >=150 or edgedet[2] >=150){
                combined_image_score[2] = combined_image_score[2]+1;
            }
            x++;
        }
        y++;
```

```
            }

            blur_score[j] = combined_image_score[0];//tracks blur score
            if(highest_blur == 0 or highest_blur < combined_image_score[0]){//looks for the largest value in terms of HSV for any one elemend indecating a
                //sroge presence on any one color
                highest_blur = combined_image_score[0];//records this value for the highest_blur value in the array sofar
            }

            hsv_score[j] = combined_image_score[1];//tracks hsv score
            if(highest_hsv == 0 or highest_hsv < combined_image_score[1]){//looks for the largest value in terms of HSV for any one elemend indecating a
            //sroge presence on any one color
                highest_hsv = combined_image_score[1];//records this value for the highest HSV value in the array sofar
            }

            edge_score[j] = combined_image_score[2];//tracks edge score
            if(highest_edge == 0 or highest_edge < combined_image_score[2]){// looks for the most edges in any one section of the images indicating
            //large changes in the image
                highest_edge = combined_image_score[2];//records this value for the highest  value in the array sofar
            }

            memset(combined_image_score, 0x00, sizeof combined_image_score);//resizes the image score

            /*cout << "location :"  << j << endl;   //used in testing
            cout << "blur_score :" << blur_score[j] << endl;
            cout << "hsv_score :" << hsv_score[j] << endl;
            cout << "edge_score :" << edge_score[j] << endl;

            cout << "highest_blur :" << highest_blur << endl;//looks at only the highest values of RGB
            cout << "highest_hsv :" << highest_hsv << endl;//looks at only the highest values of HSV
            cout << "highest_edge :" << highest_edge << endl;//looks at only the highest amount of lines in an image group*/

            total_score[j] = blur_score[j] + hsv_score[j] + edge_score[j];//combines all the scores to find the best group according to all 3 elements indicating the
            //strongest colour, brightness and density of the image
            if(highest_score == 0 or highest_score < total_score[j]){

                highest_score = total_score[j];//declares a new highest score
                move_location[0] = ly;//highest score location y coordinates
                move_location[1] = lx;//highest score location x coordinates
            }

            //cout << "Total :" << total_score[j] << endl;// prints the current total score to the screen if the score dose not changes it will
            //be the same score as before

            lx++;
            if(lx == 8){//indicating the end of a row
                ly++;//changes the section of the image to the next row down
                lx = 0;//resets this to shift it to the first image on the left
                x = 0;//resets the x cordernate on the image
            }
            else if( ly > 0){//looks Ly is in the image and it not the end of the row
                y = rect_locations_y[(ly - 1)];//set y to the next segment of pixles as shown by the values in the array rect_locations_y
            }
            else{
                y = 0;
            }

        }
        observe_area= Rect(rect_locations_x[move_location[1]]-30, rect_locations_y[move_location[0]]-40, 80, 60);//creates target area

        double KPx = 0.1; // track rate X
        double KPy = 0.1; // track rate Y

        cout << "highest_score :" << highest_score << "location: " << move_location[0] << move_location[1] << endl;//prints out the highest overall
        //score and x,y group location

        //calculates x movement towards target location
        double RxScaleV = RxRangeV/(double)640; //PWM range /pixel range
        double Xoff_R = 320-((rect_locations_x[move_location[1]] - 30))/RxScaleV ; // compare to centre of image
        int Rxold = Rx;
        Rx = Rxold-Xoff_R*KPx;
        Lx = Rxold-Xoff_R*KPx;//Left camera follows the primary right camera. this is due to possible conflict if implemented in both cameras
```

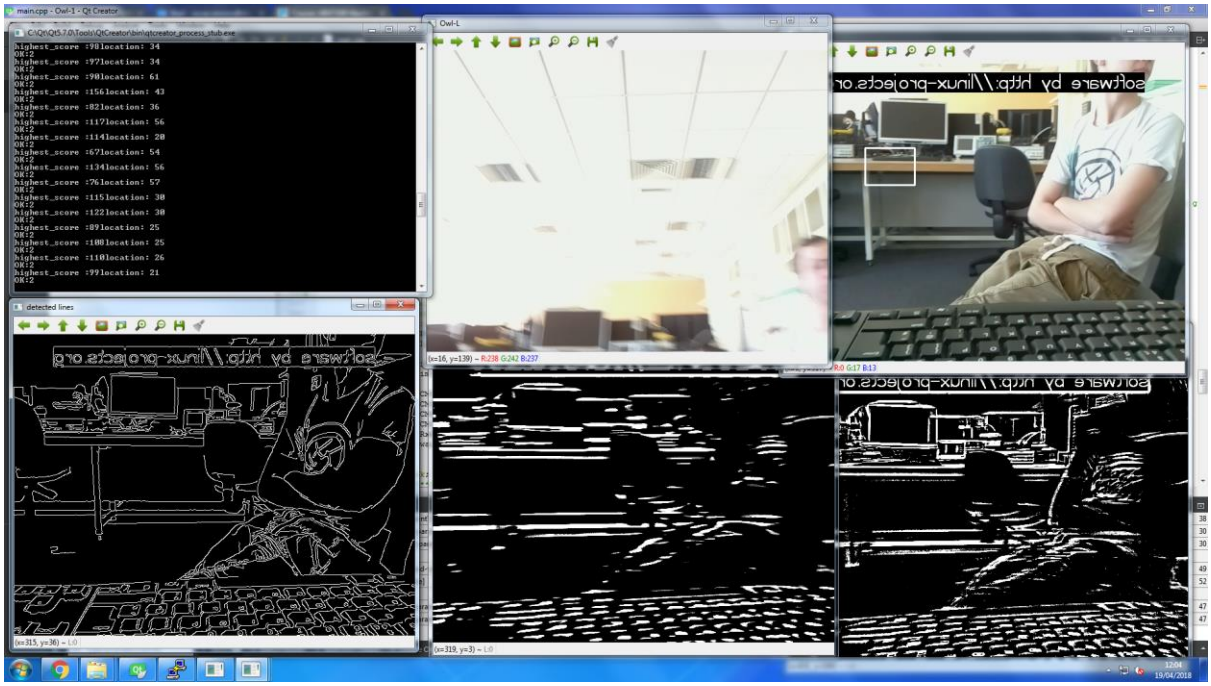
Conclusions

I believe that this way of using the RGB and HSV channels is the best way to analyse a scene. Also the edge detection allows the algorithm to look at highly dense area of the image feed. Also it is able to find the most interesting place in the image feed.

Please see the video or the image bellow for the results.

https://www.youtube.com/watch?v=ouudWMIVj-Q&feature=youtu.be

Student number 10525757



8. Review a computer vision based toy

I have chosen to review Cozmo. It is a small toy developed by Anki that uses a number of combinations of computer vision and robotics to make a fun user experience. Cozmo uses a number of computer vision tools including edge detection facial recognition and trying to understand and move around its own environment. Other features of the toy include showing facial expressions through two LED screens to represent the eyes. There are also other aspects like moving round blocks, repeating sounds and words, allowing the user to drive it through their own smart phone and allowing Cozmo to be programed using scratch to make it accessible to anyone. Now how does this compare with my own robot owl. The owl can show emotions shown in assignment one, move its eyes and neck to look at areas of interest. The owl robot does indeed use computer vision like edge detection seen in the processing model although I'm sure slightly more complex than my implementation. The main reason this toy looks so interesting however is its AI component playing games and entertaining people which is where this toy exceeds the capabilities of the owl robot I worked with due to the large number of pre-programmed actions and the flexibility of it having batteries and connecting wirelessly. This makes it an impressive toy.

9. Links:

https://www.techradar.com/reviews/anki-cozmo

https://www.tomsguide.com/us/anki-cozmo,review-4515.html

https://techcrunch.com/2016/10/14/anki-cozmo-review/

http://www.euronews.com/2016/08/29/cozmo-a-high-tech-toy-for-the-present-and-future

10. Annex:

```
// owl.cpp : Defines the entry point for the console application.

/* Phil Culverhouse Oct 2016 (c) Plymouth UNiversity

 *

 * Uses IP sockets to communicate to the owl robot (see owl-comms.h)

 * Uses OpenCV to perform normalised cross correlation to find a match to a template

 * (see owl-cv.h).

 * PWM definitions for the owl servos are held in owl-pwm.h

 * includes bounds check definitions

 * requires setting for specific robot

 *

 * This demosntration programs does the following:

 * a) loop 1 - take picture, check arrow keys

 *          move servos +5 pwm units for each loop

 *          draw 64x64 pixel square overlaid on Right image

 *          if 'c' is pressed copy patch into a template for matching with left

 *           exit loop 1;

 * b) loop 2 - perform Normalised Cross Correlation between template and left image

 *          move Left eye to centre on best match with template

 *          (treats Right eye are dominate in this example).

 *          loop

 *          on exit by ESC key

 *              go back to loop 1

 *

 * First start communcations on Pi by running 'python PFCpacket.py'

 * Then run this program. The Pi server prints out [Rx Ry Lx Ly] pwm values and loops

 *

 * NOTE: this program is just a demonstrator, the right eye does not track, just the left.

 */

//https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html-------------------------!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


#include <iostream>
```

Student number 10525757

```
#include <fstream>
#include <sys/types.h>
#include <unistd.h>

#include "owl-pwm.h"
#include "owl-comms.h"
#include "owl-cv.h"

#include <math.h>
#include <iostream> // for standard I/O
#include <string>   // for strings

using namespace std;
using namespace cv;

int main(int argc, char *argv[])
{
   char receivedStr[1024];
   ostringstream CMDstream; // string packet
   string CMD;
   int N;

   Rx = RxLm; Lx = LxLm;
   Ry = RyC; Ly = LyC;
   Neck= NeckC;

   string source ="http://10.0.0.10:8080/stream/video.mjpeg"; // was argv[1];          // the source file name
   string PiADDR = "10.0.0.10";

   //SETUP TCP COMMS
   int PORT=12345;
   SOCKET u_sock = OwlCommsInit ( PORT, PiADDR);
```

Student number 10525757

```
   /**********************
 * LOOP continuously for testing
 */
   Rx = RxC; Lx = LxC;
   Ry = RyC; Ly = LyC;
   Neck= NeckC;

   const Mat OWLresult_R;// correlation result passed back from matchtemplate
   const Mat OWLresult_L;// correlation result passed back from matchtemplate

   cv::Mat Frame;
   Mat Left, Right; // used for images from right and left camera

   bool inLOOP=true; // run through cursor control first, capture a target then exit loop

   Mat rgb_breakdown[2], edge_det, edge_det2, image_HSV, image, gblur, dst;//images used in processing the scene
   int image_grouping = 64;//number of grouped area to be observed
   int blur_score[image_grouping], hsv_score[image_grouping], edge_score[image_grouping], total_score[image_grouping];//arrays used to keep track of the scoring system
   int highest_blur = 0, highest_hsv = 0, highest_edge = 0, highest_score = 0, move_location[2] = {0,0};//used to keep track of the best location depending on that aspect and all aspects
   int rect_locations_x[8] = {80,160,240,320,400,480,560,640};//declares grouping size, if changed image_grouping would need to be updated
   int rect_locations_y[8] = {60,120,180,240,300,360,420,480};

   int working=1;
   int i = 0;

   while (inLOOP){
      // move servos to centre of field
      CMDstream.str("");
      CMDstream.clear();
      CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
```

```
CMD = CMDstream.str();

string RxPacket= OwlSendPacket (u_sock, CMD.c_str());
VideoCapture cap (source);            // Open input

if (!cap.isOpened())
{
   cout  << "Could not open the input video: " << source << endl;
   return -1;
}

while (inLOOP){
   if (!cap.read(Frame))
   {
      cout  << "Could not open the input video: " << source << endl;
      //       break;
   }

   Mat FrameFlpd; cv::flip(Frame,FrameFlpd,1); // Note that Left/Right are reversed now

   // Split into LEFT and RIGHT images from the stereo pair sent as one MJPEG iamge
   Left= FrameFlpd( Rect(0, 0, 640, 480)); // using a rectangle
   Right=FrameFlpd( Rect(640, 0, 640, 480)); // using a rectangle

   Mat RightCopy;
   Right.copyTo(RightCopy);
   rectangle( RightCopy, target, Scalar::all(255), 2, 8, 0 ); // draw white rect

   Mat LeftCopy;
   Left.copyTo(LeftCopy);
   rectangle( LeftCopy, target, Scalar::all(255), 2, 8, 0 ); // draw white rect

   imshow("Left",LeftCopy);
```

```
        imshow("Right", RightCopy);
        waitKey(10); // display the images


        int key = waitKey(0); // this is a pause long enough to allow a stable photo to be taken.
        printf("%d",key);//mrs added 01/02/2017 to diagnose arrow keys returned code
**************************************************
        if (working!='0'){
           key=7;
        }


        int z = 100;//counter for the z direction
        int y = 200; //counter for the y direction
        int x = 400;//counter for the x direction
        int combined_image_score[3] = {0x00};
        int arrayX [10] = {1280,1830,1400,1800,1300,1400,1500,1350,1700,1500}; //the array of positions to
use for the geco eyes
        int arrayY [10] = {1180,1700,1300,1200,1950,1350,1750,1150,1150,1400}; //the array of positions to
use for the geco eyes
        int counter = 0;    //counter is used in the dog of same and is used to move betwen moveing the eyes
and the neck


        switch (key){
        case 7://means it is currently moveing
           Neck=Neck+10;
           working='0';
           break;
//***************left eye
        case 119://2490368: Changed BILL//up arrow
           Ly=Ly-5; // was Ly=+5 Changed BILL
           break;
        case 115://2621440: Changed BILL//down arrow
           Ly=Ly+5; // was Ly=-5 BILL
           break;
        case 97://2424832: Changed BILL//left arrow
```

```
            Lx=Lx-5;
             break;
          case 100://2555904: Changed BILL// right arrow
             Lx=Lx+5;
             break;
//*****************end of left eye
            //start of right eye
          case 105://2490368: Changed BILL//up arrow
             Ry=Ry+5; // was Ly=+5 Changed BILL
             break;
          case 107://2621440: Changed BILL//down arrow
             Ry=Ry-5; // was Ly=-5 BILL
             break;
          case 106://2424832: Changed BILL//left arrow
            Rx=Rx-5;
             break;
          case 108://2555904: Changed BILL// right arrow
             Rx=Rx+5;
             break;
//***********************end of right eye
          case 102://2490368: Changed F: Head Rotation
            for(int i=0;i<=(x+3);i++){
                Neck=1530 + (sin((2*M_PI*i)/x)*425);

                // code to send data to the owl
                CMDstream.str("");
                CMDstream.clear();
                CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
                CMD = CMDstream.str();
                RxPacket= OwlSendPacket (u_sock, CMD.c_str());
                // end of sending data to the owl
                waitKey(10);//puase function stopping the robot for a set time
            }
```

```
            break;
//**********************end of head rotation
        case 116://2621440: Changed t: Eye horizontal
            for(int i=0;i<=(x+3);i++){
                Rx=1545 + (sin((2*M_PI*i)/x)*425);
                Lx=1545 + (sin((2*M_PI*i)/x)*425);

                // code to send data to the owl
                CMDstream.str("");
                CMDstream.clear();
                CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
                CMD = CMDstream.str();
                RxPacket= OwlSendPacket (u_sock, CMD.c_str());
                waitKey(10);
                //end of sending data to the owl
            }
            break;
        case 103://2621440: Changed g: GECKO! eyes
            for(int i=0;i<10;i++)
            {
                Rx=arrayX[i];//rx ry lx ly used a series of preset points in an array to move the eyes in a gecho like
manner
                Ry=arrayY[i];
                Lx=arrayX[(10-i)]; // this hapens in an oposite direction the right eye
                Ly=arrayY[i];

                //code to send data to the owl
                CMDstream.str("");
                CMDstream.clear();
                CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
                CMD = CMDstream.str();
                RxPacket= OwlSendPacket (u_sock, CMD.c_str());
                waitKey(1000);
```

```
            //end of sending data to the owl
        }

        Rx=1545;//these 4 value are used to center the eyes
        Ry=1460;
        Lx=1545;
        Ly=1560;
        break;
//**********************end of gecko eyes
      case 104://2621440: Changed h: dog in shame
        for(int i=0;i<=y;i++){
          if((Ry <=1200 || Ly <=1200) && counter == 0){//makes sure the eyes point towards the floor and left corner, then neck will move left
            for(int i=0;i<=x;i++){
              if(i==(y/2)){
                for(int j=0;j<=y;j++){
                  Neck=1530 - (sin((2*M_PI*j)/x)*425);// allows neck rotation

                  //used to send data to the owl
                  CMDstream.str("");
                  CMDstream.clear();
                  CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
                  CMD = CMDstream.str();
                  RxPacket= OwlSendPacket (u_sock, CMD.c_str());
                  //end of sending data to the owl
                  waitKey(10);
                }
              }
              else{//neck moves back to the right and eyes return to center
                Rx=1545 + (sin((2*M_PI*i)/x)*425);//uses sin wave to make a smooth transition for both eyes
                Lx=1545 + (sin((2*M_PI*i)/x)*425);

                //send data to the owl robot
```

```
                    CMDstream.str("");
                    CMDstream.clear();
                    CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
                    CMD = CMDstream.str();
                    RxPacket= OwlSendPacket (u_sock, CMD.c_str());

                    //end of sending data
                    waitKey(10);
                    if(i==(x/2)){
                       break;// breaks out of the for loop if the owl returns to the center
                    }
                  }
                }
                counter = 1;// escape condition for the for loop
              }
              else{
                Ry=1510 - (sin((2*M_PI*i)/x)*425);//centers the eyes verticaly
                Ly=1510 + (sin((2*M_PI*i)/x)*425);//centers the eyes verticaly
              }
              //sends the paket data
              CMDstream.str("");
              CMDstream.clear();
              CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
              CMD = CMDstream.str();
              RxPacket= OwlSendPacket (u_sock, CMD.c_str());
              //end of packet data
              waitKey(10);
            }
            break;
//**********************end of dog in shame
        case 121://2621440: Changed y: Joeys intense apology look
            for(int i=0;i<=z;i++){
              if(i<(z/2)){
```

```
            Rx = Rx - 5;// moves the eyes to the top and left
            Lx = Lx - 5;
            Ry = Ry + 5;
            Ly = Ly - 5;

            if(i > z/4){
               Neck = Neck - 10; // moves the head to the right
            }
         }
         else{
            if(i==(z/2)){
               waitKey(1000);// pasues halfway through to stair sidways on for the target
            }

            Neck = Neck + 5;

            if(i > (z- (z/4))){//moves the neck and eyes back to the center
               Rx = Rx + 10;
               Lx = Lx + 10;
               Ry = Ry - 10;
               Ly = Ly + 10;
            }
         }
         //sends the packet data

         CMDstream.str("");
         CMDstream.clear();
         CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
         CMD = CMDstream.str();
         RxPacket= OwlSendPacket (u_sock, CMD.c_str());
         //end of sending the packet data
         waitKey(20);//pauses
      }
```

```
            break;
//**********************end of Joeys intense apology look
        case 122:// case z : used to take photos for the calibration
            Rx = 1550;// centers cameras
            Lx = 1550;
            Ry = 1390;
            Ly = 1595;

            CMDstream.str("");
            CMDstream.clear();
            CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;
            CMD = CMDstream.str();
            RxPacket= OwlSendPacket (u_sock, CMD.c_str());
            waitKey(20);

            image = Right;
            imwrite( "C:/Repository/imgs/rightimg"+ std::to_string(i) +".jpg", image );
            image = Left;
            imwrite( "C:/Repository/imgs/leftimg"+ std::to_string(i) +".jpg", image );
            //takes images from left and right cameras
            i++;
            break;
//**********************end calibrate and take picture
        case 120: // case x detecting interesting features ----------------------------------------------------------------------------
---------------------------------
            inLOOP=true; // run through the loop until decided to exit
            destroyAllWindows();

            while (inLOOP) {
              if (!cap.read(Frame))//reads and image frame
              {
                 cout  << "Could not open the input video: " << source << endl;
                 break;
```

```
}

Mat FrameFlpd; cv::flip(Frame,FrameFlpd,1); // Note that Left/Right are reversed now


// Split into LEFT and RIGHT images from the stereo pair sent as one MJPEG iamge

Left= FrameFlpd( Rect(0, 0, 640, 480)); // creates the image Left by not only splitting the camera frame recieved by the camera but also flipt

Right=FrameFlpd( Rect(640, 0, 640, 480)); // creates the image Right by not only splitting the camera frame recieved by the camera but also flipt


LeftCopy = Left;// method uses a copy of the camera image not a live stream

RightCopy = Right;


imshow("Owl-L", LeftCopy);//shows the image of the left eyes owl camera

imshow("Owl-R", RightCopy);//shows the image of the Right eyes owl camera


Rect observe_area; //is the rectangle for the target area to observe

vector<Mat> channels;

image = RightCopy;   //copys the right frame into a mat image to enable us to process the image


split(image, rgb_breakdown);  // this takes the image and breaks it down into its individual bgr chanles


// use Gaussian Blur to blur out the contours and focus on the colour

GaussianBlur(rgb_breakdown[0], rgb_breakdown[0], Size(), 4, 1, BORDER_DEFAULT );//represents the red image

GaussianBlur(rgb_breakdown[1], rgb_breakdown[1], Size(), 4, 1, BORDER_DEFAULT );//represents the green image

GaussianBlur(rgb_breakdown[2], rgb_breakdown[2], Size(), 4, 1, BORDER_DEFAULT );// represents the Blue image


/*waitKey(20);  // used for testing

imshow("Red",rgb_breakdown[0]);

waitKey(20);

imshow("Green",rgb_breakdown[1]);

waitKey(20);
```

```
imshow("Blue",rgb_breakdown[2]);

waitKey(20);*/


//adaptive thresholding is a way of looking for only the highist amound of color in the image

adaptiveThreshold(rgb_breakdown[0], rgb_breakdown[0],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);//thresholds the red image

adaptiveThreshold(rgb_breakdown[1], rgb_breakdown[1],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);//thresholds the green image

adaptiveThreshold(rgb_breakdown[2], rgb_breakdown[2],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);//thresholds the blue image


addWeighted(rgb_breakdown[0],0.7,rgb_breakdown[1],0.3,0, gblur);// this is how to combine the red and green images after the threshold has been applied

addWeighted(gblur,0.7,rgb_breakdown[2],0.3,0,gblur);// this adds the thresholded blue image to the previos combined image


imshow("RGB GaussianBlur",gblur); // shows the completed RGB image showing the areas of the most color only


cvtColor(image, image_HSV, cv::COLOR_RGB2BGR);//converts the RGB image to Hue, Saturation, Brightness


channels.clear();

channels.resize(image_HSV.channels());  //resize channels

cv::split(image_HSV, &channels[0]);// splits the images into its individual channels


/*waitKey(20); // used for testing

imshow("Hue",channels[0]);

waitKey(20);

imshow("Saturation",channels[1]);

waitKey(20);

imshow("Lightness",channels[2]);

waitKey(20);*/


//adaptive thresholding is a way of looking for only the highist amound of color in the image
```

```
adaptiveThreshold(channels[0], channels[0],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);// thresholds the Hue in the image

adaptiveThreshold(channels[1], channels[1],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);// thresholds the Saturation in the image

adaptiveThreshold(channels[2], channels[2],255,ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,15,-5);// thresholds the Lightness in the image


addWeighted(channels[0],0.7,channels[1],0.3,0, dst);// combines the Hue and Saturation channels after they have been thresholded

addWeighted(dst,0.7,channels[2],0.3,0,dst);// combines the Hue, Saturation and Lightness channels after they have been thresholded


imshow("HSV adaptiveThreshold",dst);// displays the combined HSV image after it has been thresholded


Canny(image, edge_det, 50, 200, 3);//performs canny edge detection of the original image


vector<Vec2f> lines1;
HoughLines(edge_det, lines1, 1, CV_PI/180, 100, 0, 0 );
for( size_t i = 0; i < lines1.size(); i++ ){
   float rho = lines1[i][0], theta = lines1[i][1];
   Point pt1, pt2;
   double a = cos(theta), b = sin(theta);
   double x0 = a*rho, y0 = b*rho;
   pt1.x = cvRound(x0 + 1000*(-b));//declare the first x cordernate of the line
   pt1.y = cvRound(y0 + 1000*(a));//declares the first y cordernate of the line
   pt2.x = cvRound(x0 - 1000*(-b));//declare the last x cordernate of the line
   pt2.y = cvRound(y0 - 1000*(a));//declares the last y cordernate of the line
   line( edge_det2, pt1, pt2, Scalar(0,0,255), 3, CV_AA);//draws the line
}


vector<Vec4i> lines2;
HoughLinesP(edge_det, lines2, 1, CV_PI/180, 50, 50, 10 );
for( size_t i = 0; i < lines2.size(); i++ ){
   Vec4i l = lines2[i];
```

```
    line( edge_det2, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0,0,255), 3, CV_AA);
  }

  waitKey(1000);
  imshow("detected lines", edge_det);
  waitKey(20);

  int lx = 0;// looks at the horizontal position of the image the algorithm is useing
  int ly = 0;// looks at the vertical position of the image the algorithm is useing
  int y = 1;//counter for the next loop on the position of the y cordernate
  int x = 1;//counter for the next loop on the position of the x cordernate
  highest_score = 0;//area of the image with the highest overall value

  for(int j = 0; j <image_grouping;j++){
    while(y <= rect_locations_y[ly])
    {
      while(x <= rect_locations_x[lx])
      {
        //gets the rgb value of the selected pixel for processing
        Vec3b colour = gblur.at<Vec3b>(Point(x, y));
        Vec3b contrast = dst.at<Vec3b>(Point(x, y));
        Vec3b edgedet = edge_det.at<Vec3b>(Point(x, y));

        //looks for the largest value in terms of RGB for any one element indicating a strong
presence on any one color
        if(colour[0] >=150 or colour[1] >=150 or colour[2] >=150){
          combined_image_score[0] = combined_image_score[0]+1;
        }
        //looks for the largest value in terms of HSV for any one element indicating a strong
presence of any one color
        if(contrast[0] >=150 or contrast[1] >=150 or contrast[2] >=150){
          combined_image_score[1] = combined_image_score[1]+1;
        }
```

```
                    // looks for the most edges in any one section of the images indicating large changes in the
image
                    if(edgedet[0] >=150 or edgedet[1] >=150 or edgedet[2] >=150){
                       combined_image_score[2] = combined_image_score[2]+1;
                    }
                    x++;
                  }
                  y++;
                }

                blur_score[j] = combined_image_score[0];//tracks blur score
                if(highest_blur == 0 or highest_blur < combined_image_score[0]){//looks for the largest value in
terms of HSV for any one elemend indecating a sroge presence on any one color
                   highest_blur = combined_image_score[0];//records this value for the highest_blur value in the
array sofar
                }

                hsv_score[j] = combined_image_score[1];//tracks hsv score
                if(highest_hsv == 0 or highest_hsv < combined_image_score[1]){//looks for the largest value in
terms of HSV for any one elemend indecating a sroge presence on any one color
                   highest_hsv = combined_image_score[1];//records this value for the highest HSV value in the
array sofar
                }

                edge_score[j] = combined_image_score[2];//tracks edge score
                if(highest_edge == 0 or highest_edge < combined_image_score[2]){// looks for the most edges in
any one section of the images indicating large changes in the image
                   highest_edge = combined_image_score[2];//records this value for the highest  value in the
array sofar
                }

                memset(combined_image_score, 0x00, sizeof combined_image_score);//resizes the image score

                /*cout << "location :"  << j << endl;   //used in testing
                cout << "blur_score :" << blur_score[j] << endl;
                cout << "hsv_score :" << hsv_score[j] << endl;
```

```
            cout << "edge_score :" << edge_score[j] << endl;


            cout << "highest_blur :" << highest_blur << endl;//looks at only the highest values of RGB

            cout << "highest_hsv :" << highest_hsv << endl;//looks at only the highest values of HSV

            cout << "highest_edge :" << highest_edge << endl;//looks at only the highest amount of lines in an image group*/


            total_score[j] = blur_score[j] + hsv_score[j] + edge_score[j];//combines all the scores to find the best group according to all 3 elements indicating the strongest colour, brightness and density of the image

            if(highest_score == 0 or highest_score < total_score[j]){

               highest_score = total_score[j];//declares a new highest score

               move_location[0] = ly;//highest score location y coordinates

               move_location[1] = lx;//highest score location x coordinates

            }


            //cout << "Total :" << total_score[j] << endl;// prints the current total score to the screen if the score dose not changes it will be the same score as before


            lx++;

            if(lx == 8){//indicating the end of a row

               ly++;//changes the section of the image to the next row down

               lx = 0;//resets this to shift it to the first image on the left

               x = 0;//resets the x cordernate on the image

            }

            else if( ly > 0){//looks Ly is in the image and it not the end of the row

               y = rect_locations_y[(ly - 1)];//set y to the next segment of pixles as shown by the values in the array rect_locations_y

            }

            else{

               y = 0;

            }


         }

         observe_area= Rect(rect_locations_x[move_location[1]]-30, rect_locations_y[move_location[0]]-40, 80, 60);//creates target area
```

```
double KPx = 0.1; // track rate X

double KPy = 0.1; // track rate Y


cout << "highest_score :" << highest_score << "location: " << move_location[0] << move_location[1] << endl;//prints out the highest overall score and x,y group location


//calculates x movement towards target location

double RxScaleV = RxRangeV/(double)640; //PWM range /pixel range

double Xoff_R = 320-((rect_locations_x[move_location[1]] - 30))/RxScaleV ; // compare to centre of image

int Rxold = Rx;

Rx = Rxold-Xoff_R*KPx;

Lx = Rxold-Xoff_R*KPx;//Left camera follows the primary right camera, this is due to possible conflict if implemented in both cameras


//calculates y movement towards target location

double RyScaleV = RyRangeV/(double)480; //PWM range /pixel range

double Yoff_R = (250-((rect_locations_y[move_location[0]] - 40))/RyScaleV)*KPy ; // compare to centre of image

int Ryold = Ry;

Ry = Ryold+Yoff_R;

Ly = Ryold-Yoff_R;//Left camera follows the primary right camera, this is due to possible conflict if implemented in both cameras


rectangle( RightCopy, observe_area, Scalar::all(255), 2, 8, 0 );//displays target area

imshow("Target", RightCopy);//displays camera scene


CMDstream.str("");//makes the stream = ""

CMDstream.clear();//clears the stream

CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;//compiles the new streem to write the eye and neck values for the servo

CMD = CMDstream.str();// combpiles the paket data

RxPacket= OwlSendPacket (u_sock, CMD.c_str());//sends the data to the owl robot

waitKey(100);
```

```
            }


            break;
//**********************end intresting feature detection
         case 99: // lowercase 'c'
            OWLtempl_R = Right(target);
            imshow("templ right",OWLtempl_R);
            Right.copyTo(LeftCopy);


            OWLtempl_L = LeftCopy(target);
            imshow("templ left",OWLtempl_L);


            waitKey(1);
            inLOOP=false; // quit loop and start tracking target
            break; // left
         default:
            key=key;
            //no key has been pressed
         }
            CMDstream.str("");//makes the stream = ""
            CMDstream.clear();//clears the stream
            CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;//compiles the new streem to
write the eye and neck values for the servo
            CMD = CMDstream.str();// combpiles the paket data
            RxPacket= OwlSendPacket (u_sock, CMD.c_str());//sends the data to the owl robot


            if (0) {
               for (int i=0;i<10;i++){


                  Rx=Rx-50; Lx=Lx-50;


                  CMDstream.str("");//makes the stream = ""
                  CMDstream.clear();//clears the stream
```

```
            CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;//compiles the new streem to write the eye and neck values for the servo

            CMD = CMDstream.str();// combpiles the paket data

            RxPacket= OwlSendPacket (u_sock, CMD.c_str());//sends the data to the owl robot

          }

        }

    } // END cursor control loop

    // close windows down

    destroyAllWindows();

    // just a ZMCC

    // right is the template, just captured manually

    i = 0;

    inLOOP=true; // run through the loop until decided to exit

    while (inLOOP) {

      if (!cap.read(Frame))//trys to read from the camera

      {

        cout  << "Could not open the input video: " << source << endl; // this is and error mesage that will apera in the event of no image from the owl robot being recived

        break;

      }


      Mat FrameFlpd; cv::flip(Frame,FrameFlpd,1); // Note that Left/Right are reversed now


      // Split into LEFT and RIGHT images from the stereo pair sent as one MJPEG iamge

      Left= FrameFlpd( Rect(0, 0, 640, 480)); // creates the image Left by not only spliting the camera frame recived by the camera but also flipt

      Right=FrameFlpd( Rect(640, 0, 640, 480)); // creates the image Right by not only spliting the camera frame recived by the camera but also flipt


      OwlCorrel OWL_L;//used in corrolation of the left image frame

      OwlCorrel OWL_R;//used in corrolation of the right image frame


      OWL_L = Owl_matchTemplate( Right,  Left, OWLtempl_L, target);//matched the template image to the current image frame
```

```
OWL_R = Owl_matchTemplate( Left,   Right, OWLtempl_R, target);//matched the template image to the current image frame


/// Show me what you got


rectangle( Left, OWL_L.Match, Point( OWL_L.Match.x + OWLtempl_L.cols , OWL_L.Match.y + OWLtempl_L.rows), Scalar::all(255), 2, 8, 0 );//draws the square in the center of the Left image frame

rectangle( Right, OWL_R.Match, Point( OWL_R.Match.x + OWLtempl_R.cols , OWL_R.Match.y + OWLtempl_R.rows), Scalar::all(255), 2, 8, 0 );//draws the square in the center of the Right image frame

rectangle( OWLresult_R, OWL_R.Match, Point( OWL_R.Match.x + OWLtempl_R.cols , OWL_R.Match.y + OWLtempl_R.rows), Scalar::all(255), 2, 8, 0 );//draws the square around the matched image of the Left frame

rectangle( OWLresult_L, OWL_L.Match, Point( OWL_L.Match.x + OWLtempl_L.cols , OWL_L.Match.y + OWLtempl_L.rows), Scalar::all(255), 2, 8, 0 );//draws the square around the matched image of the Right frame


imshow("Owl-L", Left);//shows the image of the left eyes owl camera

imshow("Owl-R", Right);//shows the image of the Right eyes owl camera

imshow("Correl",OWL_L.Result );//shows the image of the Left image matched frame

imshow("Correl",OWL_R.Result );//shows the image of the Right image matched frame


if (waitKey(10)== 27) inLOOP=false;


double KPx = 0.1; // track rate X

double KPy = 0.1; // track rate Y


double RxScaleV = RxRangeV/(double)640; //PWM range /pixel range

double Xoff_R = 320-(OWL_R.Match.x + OWLtempl_R.cols)/RxScaleV ; // compare to centre of image

int RxOld = Rx;


double LxScaleV = LxRangeV/(double)640; //PWM range /pixel range

double Xoff_L = 320-(OWL_L.Match.x + OWLtempl_L.cols)/LxScaleV ; // compare to centre of image

int LxOld = Lx;


Rx = RxOld-Xoff_R*KPx; // roughly 300 servo offset = 320 [pixel offset

Lx = LxOld-Xoff_L*KPx; // roughly 300 servo offset = 320 [pixel offset
```

```
double RyScaleV = RyRangeV/(double)480; //PWM range /pixel range

double Yoff_R = (250-(OWL_R.Match.y + OWLtempl_R.rows)/RyScaleV)*KPy ; // compare to centre of image

int RyOld = Ry;

Ry = RyOld+Yoff_R; // roughly 300 servo offset = 320 [pixel offset


double LyScaleV = LyRangeV/(double)480; //PWM range /pixel range

double Yoff_L = (250+(OWL_L.Match.y + OWLtempl_L.rows)/LyScaleV)*KPy ; // compare to centre of image

int LyOld = Ly;

Ly = LyOld-Yoff_L; // roughly 300 servo offset = 320 [pixel offset


cout << Rx << " " << Xoff_R << " " << RxOld << endl;//prints how far the matched section of the image is off fo center in the Right frame's x direction

cout << Ry << " " << Yoff_R << " " << RyOld << endl;//... Right frame's y direction

cout << Lx << " " << Xoff_L << " " << LxOld << endl;//... Left frame's x direction

cout << Ly << " " << Yoff_L << " " << LyOld << endl;//... Left frame's y direction

cout << "#################################################################" << endl;

double aR;//angle Left

double aL;//angle right

double Dl;//distance from the eye to the target

double Dc;//final distance to the target


aR = 160 - (((Rx+OWL_R.Match.x) - RxLm)/4.375);//calculates the right angle

aL = ((Lx+OWL_L.Match.x) - LxLm)/4.375;//calculates the Left angle

cout << "-----------------------------------------------------------------" << endl;

cout << aR << "right angle :" << OWL_R.Match.x << ":" << Rx << endl;//prints the right angle to the screen

cout << aL << "left angle : " << OWL_L.Match.x << ":" << Lx << endl;//prints the left angle to the screen

cout << "-----------------------------------------------------------------" << endl;

aR = (M_PI * aR)/180;//convert degrees to radeons

aL = (M_PI * aL)/180;//convert degrees to radeons


Dl = (6.7*cos(aR))/sin(aL + aR);//calculate the distance from the eye to the target
```

```
            Dc = sqrt(pow(Dl,2) + (pow(6.7,2)/4)- Dl * 6.7 * sin(aL));//calculates the final distance to the target
            //Dc = Dc + (Dc*0.1);
            cout << "--------------------------------------------------------------------------" << endl;
            cout << Dl << "DL distance" << endl;//prints the distcance to the target on the screen
            cout << Dc << "final distance" << endl;//prints the calculated final distance to the screen
            cout << "--------------------------------------------------------------------------" << endl;
            // dc is at a final distance in cm so add a case to save images

            // ACTION
             cap.read(Frame);//reads the camera frame
            // move to get minimise distance from centre of both images, ie verge in to targe
            // move servos to position
             CMDstream.str("");//makes the stream = ""
             CMDstream.clear();//clears the stream
             CMDstream << Rx << " " << Ry << " " << Lx << " " << Ly << " " << Neck;//compiles the new streem to
write the eye and neck values for the servo
             CMD = CMDstream.str();// combpiles the paket data
             RxPacket= OwlSendPacket (u_sock, CMD.c_str());//sends the data to the owl robot

          } // end if ZMCC
      } // end while outer loop
#ifdef __WIN32__
      closesocket(u_sock);
#else
      close(clientSock);
#endif
      exit(0); // exit here for servo testing only
   }
```