2-D AND 3-D (STEREO) VISION USING OPENCV

By: Yazhini Priyadharshini

Objective:

1. 2-D Vision sensing using USB camera
2. 3-D Stereo Vision using two cameras
3. Colour Sensing

1. Brief Introduction:

A standard 2D machine vision image is flat, calibrated to allow the measurement of length and width, but it does not provide any height information. The next step, 2.5D, includes the Z axis or height information in addition to X and Y axes; it also provides information that allows the machine vision system to estimate the objects' rotation (pitch and yaw) around two of the three dimensions. True 3D provides X, Y, and Z information as well as rotational information around all three axes (rX, rY, and rZ).

If you want to inspect an object that is not moving on a conveyor, as we have seen before, or if you want to use a camera for a pick and place operation, a 2D camera will probably meet your requirements. If this is your case, another question you need to ask yourself is if you need a colour camera or if a black & white one would be sufficient. Again, our brain is so used to see in colour, we tend to think we need the same for our camera. It is much simpler, though, to analyse black and white images. Using a colour camera would leave you with more data, but would you know what to do with it? Using a black & white camera would still leave you with multiple shades of grey to work with. The processing time would be far less than with colour images, the data transferring time would be decreased and the need for data storage also.

Another thing with 2D cameras: some cameras are sensors that capture the image and let you transfer it to a computer for extensive image processing. Some other cameras have the sensor and processor all embedded in the camera: these are called intelligent cameras or smart cameras. If you need basic image processing, for pick & place operations, for example, this type of camera will probably be sufficient, as you will not use funky image processing functions, and will not require to add a computer to do so.

Sensor size / Field of view = Focal length / Working distance

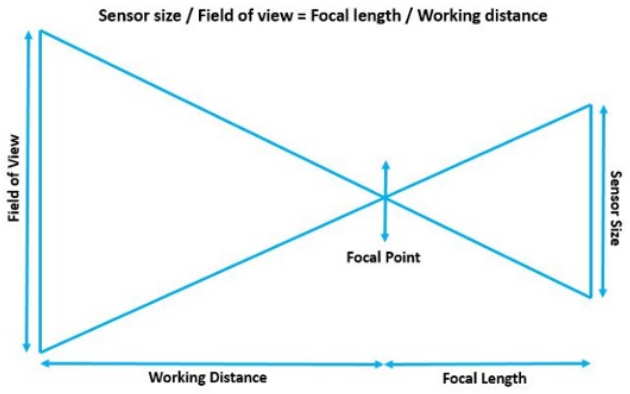


Figure 1.1- Field of View

Sensor size: Actual size of the CCD sensor inside the camera. Various dimensions are available, ranging from ¼" to 1.2". The sensor size vs its name is rather confusing: a 2/3" sensor will actually measure 8.6 x 6.6 mm (if you're interested in historical reasons for the 2/3" name, consult this page).

Field of view: The physical size of the area you will be looking at with your camera. This area will probably be rectangular, since most CCDs are rectangular. Typically, it will follow a 4:3 rule, so if you have 12cm in width, you will have 9 cm in height.

Focal length: The ‘strength’ of the light convergence. 25mm, 50mm, etc. The focal length is inversely proportional to the field of view, so an image taken with a 50mm lens will be more zoomed than if it were taken with a 25mm lens.

Working distance: Physical distance from the sensors to the object you want to look at.

Computer stereo vision is the extraction of 3D information from digital images, such as obtained by a CCD camera. By comparing information about a scene from two vantage points, 3D information can be extracted by examination of the relative positions of objects in the two panels.

In traditional stereo vision, two cameras, displaced horizontally from one another are used to obtain two differing views on a scene, in a manner like human binocular vision. By comparing these two images, the relative depth information can be obtained in the form of a disparity map, which encodes the difference in horizontal coordinates of corresponding image points. The values in this disparity map are inversely proportional to the scene depth at the corresponding pixel location.

For a human to compare the two images, they must be superimposed in a stereoscopic device, with the image from the right camera being shown to the observer's right eye and from the left one to the left eye.

In a computer vision system, several pre-processing steps are required.[1]

The image must first be undistorted, such that barrel distortion and tangential distortion are removed. This ensures that the observed image matches the projection of an ideal pinhole camera.

The image must be projected back to a common plane to allow comparison of the image pairs, known as image rectification.

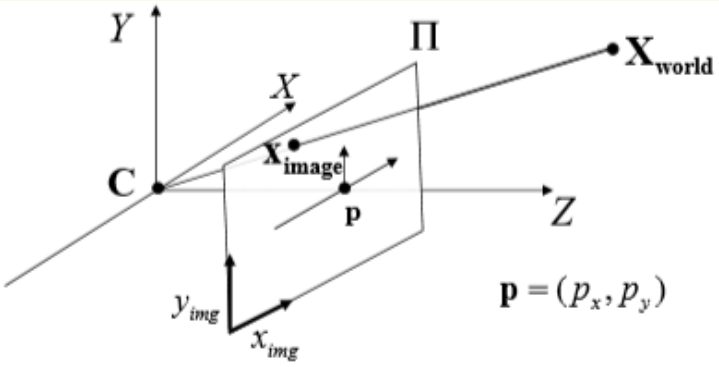
An information measure which compares the two images is minimized. This gives the best estimate of the position of features in the two images, and creates a disparity map.

Optionally, the received disparity map is projected into a 3d point cloud. By utilising the cameras' projective parameters, the point cloud can be computed such that it provides measurements at a known scale.

We use OpenCV for this purpose. For 2-D vision sensing, one camera is used. It is first calibrated and then using a grid, kept at various positions to get the x coordinate and y coordinate of the object. In 3-D vision sensing, two cameras are used. Two images received are then put together to form one disparity image from which using the intrinsic and extrinsic parameters, point cloud image is taken. The intrinsic and extrinsic parameters are:

Extrinsic camera parameters: the parameters that define the location and orientation of the camera reference frame with respect to a known world reference frame. These are the parameters that identify uniquely the transformation between the unknown camera reference frame and the known world reference frame. - Typically, determining these parameters means: (1) finding the translation vector between the relative positions of the origins of the two reference frames. (2) finding the rotation matrix that brings the corresponding axes of the two frames into alignment (i.e., onto each other)

Intrinsic camera parameters: the parameters necessary to link the pixel coordinates of an image point with the corresponding coordinates in the camera reference frame. These are the parameters that characterize the optical, geometric, and digital characteristics of the camera: (1) the perspective projection (focal length f). (2) the transformation between image plane coordinates and pixel coordinates. (3) the geometric distortion introduced by the optics.



Camera internal parameters

$$K = \begin{bmatrix} \alpha_x & s & p_x \\ & \alpha_y & p_y \\ & & 1 \end{bmatrix}$$

α_x Scale factor in x- coordinate direction

α_y Scale factor in y- coordinate direction

s Camera skew

$\frac{\alpha_x}{\alpha_y}$ Aspect ratio

Camera matrix, $P = K[R \mid \mathbf{t}]$

R Rotation

$\mathbf{t}$ Translation vector

Camera skew factor/parameter, s: The parameter "s" accounts for a possible nonorthogonality of the axes in the image plane. This might be the case if the rows and columns of pixels on the sensor are not perpendicular to each other.

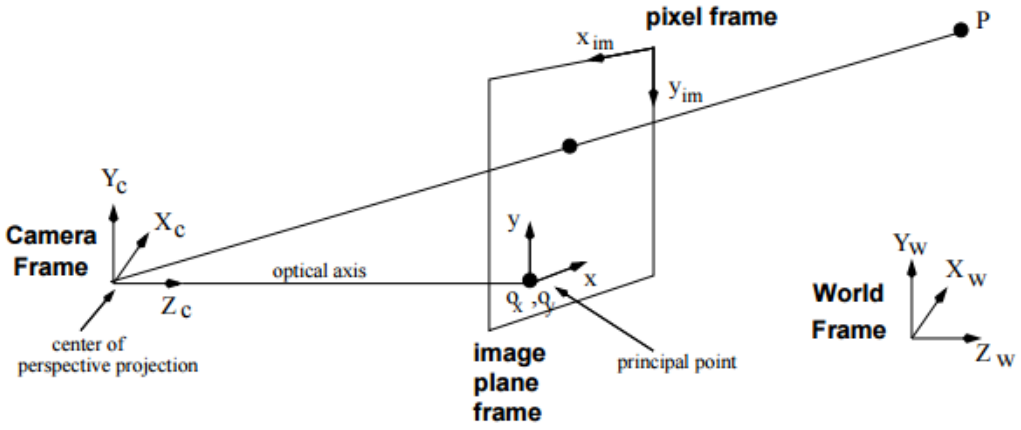


Figure 1.2- Camera and pixel frame

Two images coming from the two cameras are rectified to form one single image. The disparity of the images is then found out.

3D stereo vision can be understood with the following example- Most human beings use what is known as binocular vision to perceive depth and see the world in 3D. The binocular vision system relies on the fact that we have two eyes, which are approximately 3 in apart. This separation causes each eye to see the world from a slightly distinct perspective. The mind combines the two images by matching up the similarities and adding in the slight differences. The minor differences between the two images add up to an enormous difference in the final picture! The combined image is more than the sum of its parts. It is a three-dimensional *stereo* picture. It understands the differences and uses them to calculate distance

creating our sense of depth and ability to gauge distance. An uncomplicated way to understand this principle is to hold your thumb up at arm's length and close one eye. Then try closing the other eye. As you switch between open eyes you should see your thumb "jumping" back and forth against the background. To see how much of a difference the binocular vision system makes, have a friend throw you a ball and try to catch it while keeping one eye closed.

In a computer vision system, several pre-processing steps are required.

1. The image must first be undistorted, such that barrel distortion and tangential distortion are removed. This ensures that the observed image matches the projection of an ideal pinhole camera.
2. The image must be projected back to a common plane to allow comparison of the image pairs, known as image rectification.
3. An information measure which compares the two images is minimized. This gives the best estimate of the position of features in the two images, and creates a disparity map.
4. Optionally, the received disparity map is projected into a 3d point cloud. By utilising the cameras' projective parameters, the point cloud can be computed such that it provides measurements at a known scale.

The diagram of a simplified stereovision setup, where both cameras are mounted perfectly parallel to each other, and have the exact same focal length is given below.

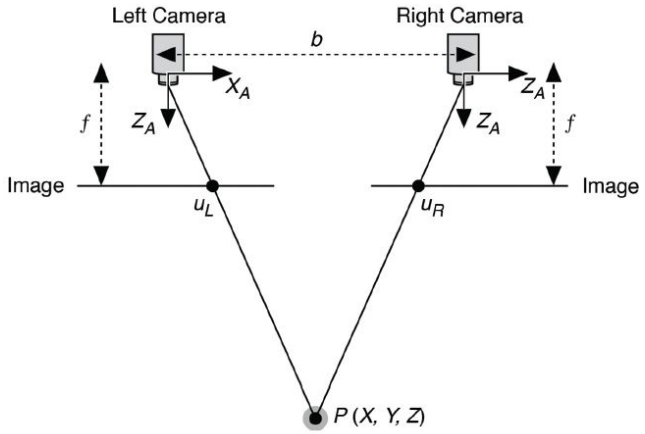


Figure 2- simplified stereo system

The variables in Figure are: b is the baseline, or distance between the two cameras, f is the focal length of a camera, X_A is the X-axis of a camera, Z_A is the optical axis of a camera, P is a real-world point defined by the coordinates X, Y, and Z, u_L is the projection of the real-world point P in an image acquired by the left camera, u_R is the projection of the real-world point P in an image acquired by the right camera.

Since the two cameras are separated by distance "b," both cameras view the same real-world point P in a different location on the two-dimensional images acquired. The X-coordinates of points u_L and u_R are given by:

$$u_L = f * X/Z \text{ and } u_R = f * (X-b)/Z$$

Distance between those two projected points is known as "disparity" and we can use the disparity value to calculate depth information, which is the distance between real-world point "P" and the stereo vision system.

$$\text{disparity} = u_L - u_R = f * b/z$$

$$\text{depth} = f * b/\text{disparity}$$

The calibration process involves using a calibration target — for example, a grid of dots or a checkerboard — and acquiring images at different angles to calculate image distortion, as well as the exact spatial relationship between the two cameras.

To optimize the accuracy of a stereovision system setup, and accurately relate calculated image disparity to true depth data, there are several considerations and parameters to keep in mind.

For a simple stereo system, the depth of a point (z) is given by:

$$Z = f * b/d$$

where f is the focal length, b is the baseline, or distance between the cameras, and d the disparity between corresponding points.

When relating depth to disparity, it is important to note that as depth decreases, disparity increases exponentially, as illustrated in Figure 5.

Depth resolution refers to the accuracy with which a stereovision system can estimate changes in the depth of a surface. Depth resolution is proportional to the square of the depth and the disparity resolution, and is inversely proportional to the focal length and the baseline, or distance between the cameras. Good depth resolution requires a large baseline value, a large focal length value, and a small depth value for a given disparity resolution.

Stereovision Applications:

Stereovision is well suited to applications that require locating objects or obstacles, and this location data can be used to guide the movement of a robot or robotic arm. For navigating autonomous vehicles, depth information is used to measure the size and distance of obstacles for accurate path planning and obstacle avoidance. Stereovision systems can provide a rich set of 3D information for navigation applications, and can perform well even in changing light conditions.

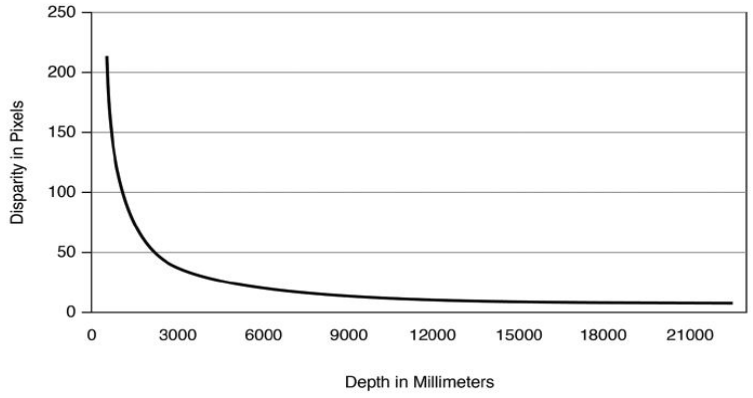


Figure 3. Disparity values as a function

A stereovision system is also useful in robotic industrial automation of tasks such as bin-picking or depalletization. A bin-picking application requires a robot arm to pick a specific object from a container that holds several distinct kinds of parts. A stereovision system can provide an inexpensive way to obtain 3D information and determine which parts are free to be grasped. It can also provide precise locations for individual products in a crate.

2. Results:

2.1 . 2-D VISION SENSING:

2-D vision sensing has been performed by tracking the movements of a single red ball using one camera. First a grid was drawn on a big white board. By moving the red ball and keeping it at various locations, the movement was recorded. The radius of the ball along with the x and y coordinates were noted.

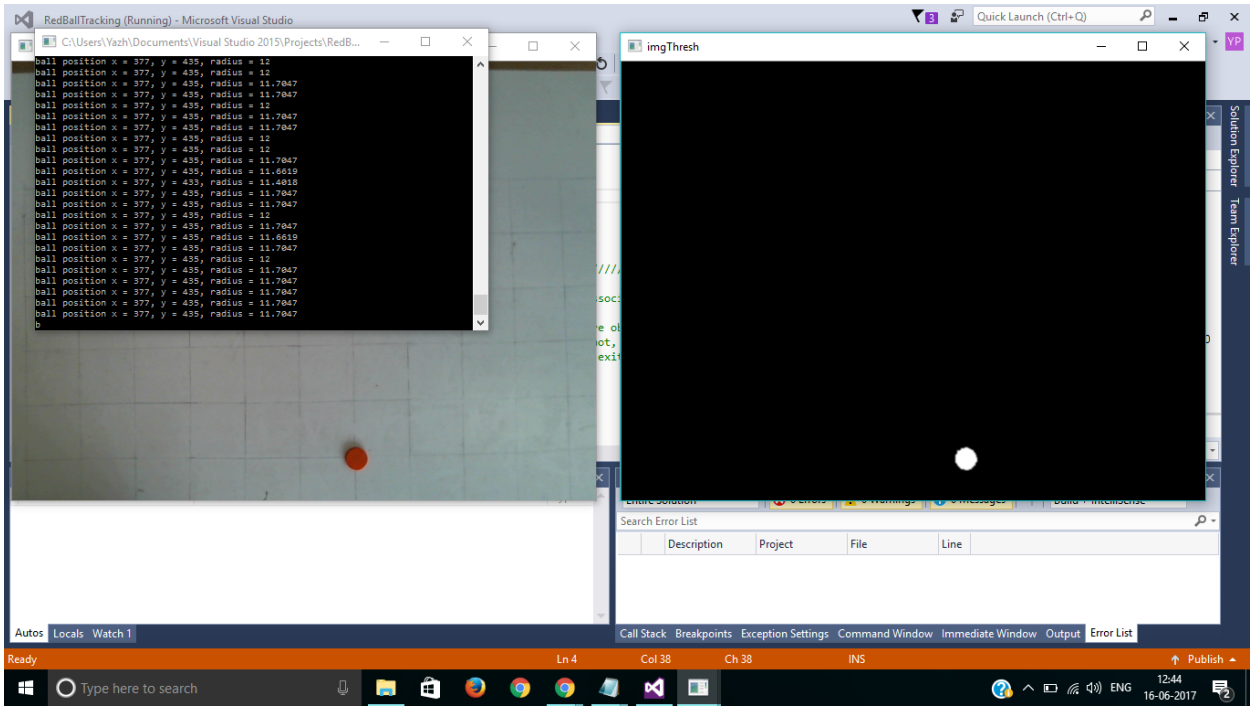


Figure 4.1- 2D Vision Sensing

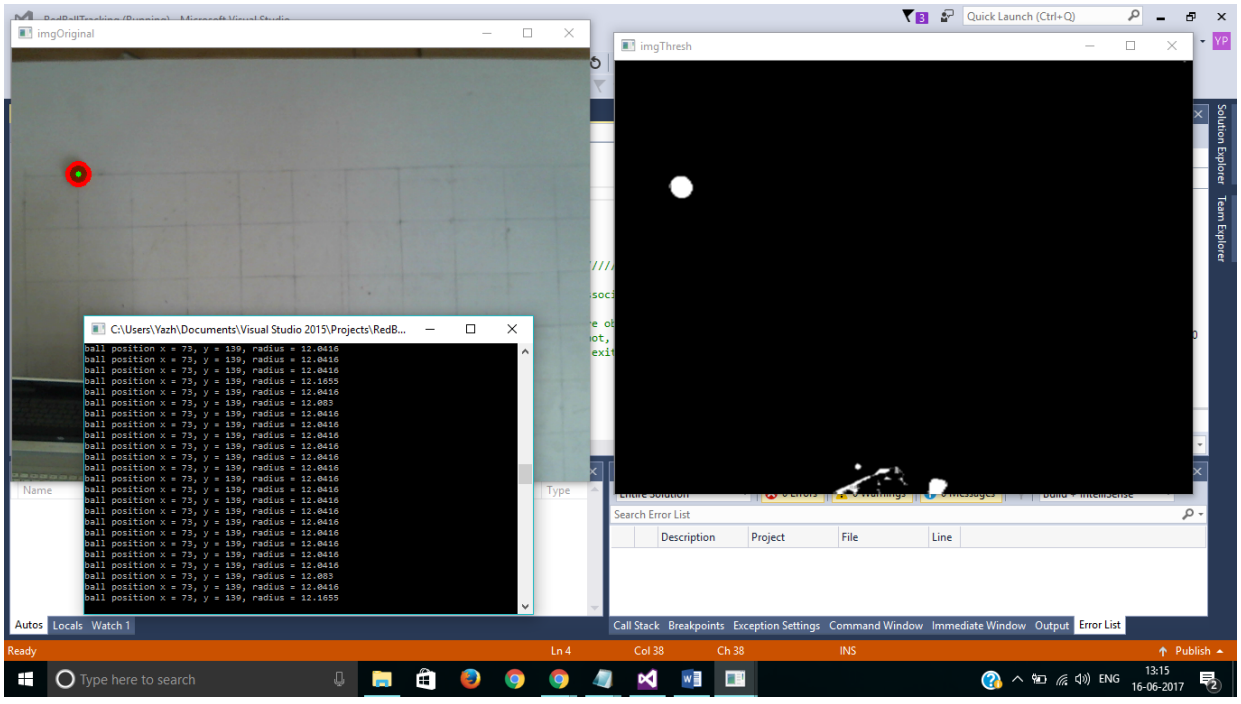


Figure 4.2- 2D Vision Sensing

Then the practical value was calculated using the map function.

long map (long x, long in_min, long in_max, long out_min, long out_max)

{ return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min; }

Here x- value of x/y coordinate; in_min - the lower bound of the value's current range; in_max: the upper bound of the value's current range; out_min: the lower bound of the value's target range; out_max: the

upper bound of the value's target range. The scatter-plot for the x and y value along with the theoretical values were plotted.

x_pixel	y_pixel	x_Prac	x_Thero	y_Prac	y_Thero	Diff(x)	Diff(y)
135	375	9.173154	10	9.863446	10	0.826846	0.136554
197	375	20.13423	20	9.863446	10	-0.13423	0.136554
255	375	29.86577	30	9.863446	10	0.13423	0.136554
315	375	39.93289	40	9.863446	10	0.06711	0.136554
375	375	50	50	9.863446	10	0	0.136554
435	373	60.06711	60	9.863946	10	-0.06711	0.136054
497	373	70.4698	70	9.863946	10	-0.4698	0.136054
555	375	80.2013	80	9.863446	10	-0.2013	0.136554
135	315	9.173154	10	20.40816	20	0.826846	-0.40816
195	315	19.79866	20	20.40816	20	0.20134	-0.40816
255	315	29.86577	30	20.40816	20	0.13423	-0.40816
315	315	39.93289	40	20.40816	20	0.06711	-0.40816
375	315	50	50	20.40816	20	0	-0.40816
375	315	50	50	20.40816	20	0	-0.40816
435	313	60.06711	60	20.06803	20	-0.06711	-0.06803
497	313	70.4698	70	20.06803	20	-0.4698	-0.06803
555	313	80.2013	80	20.06803	20	-0.2013	-0.06803
135	255	9.173154	10	30.27211	30	0.826846	-0.27211
195	255	19.79866	20	30.27211	30	0.20134	-0.27211
255	255	29.86577	30	30.27211	30	0.13423	-0.27211
315	253	39.93289	40	29.5302	30	0.06711	0.4698
375	253	50	50	29.5302	30	0	0.4698
435	253	60.06711	60	29.5302	30	-0.06711	0.4698
497	253	70.4698	70	29.5302	30	-0.4698	0.4698
555	253	80.2013	80	29.5302	30	-0.2013	0.4698
135	195	9.173154	10	40.47619	40	0.826846	-0.47619
195	195	19.79866	20	40.47619	40	0.20134	-0.47619
255	195	29.86577	30	40.47619	40	0.13423	-0.47619
315	193	39.93289	40	40.47619	40	0.06711	-0.47619
375	191	50	50	40.47619	40	0	-0.47619
435	191	60.06711	60	40.47619	40	-0.06711	-0.47619
497	191	70.4698	70	40.47619	40	-0.4698	-0.47619
555	191	80.2013	80	40.47619	40	-0.2013	-0.47619
133	139	9.395973	10	50.68027	50	0.604027	-0.68027
195	137	19.79866	20	50.34014	50	0.20134	-0.34014
253	139	29.5302	30	50.68027	50	0.4698	-0.68027
315	139	39.93289	40	50.68027	50	0.06711	-0.68027
373	139	49.66443	50	50.68027	50	0.33557	-0.68027
435	139	60.06711	60	50.68027	50	-0.06711	-0.68027
497	137	70.4698	70	50.34014	50	-0.4698	-0.34014
555	139	80.2013	80	50.68027	50	-0.2013	-0.68027

X-THEORETICAL VALUES VS X-PRACTICAL VALUES:

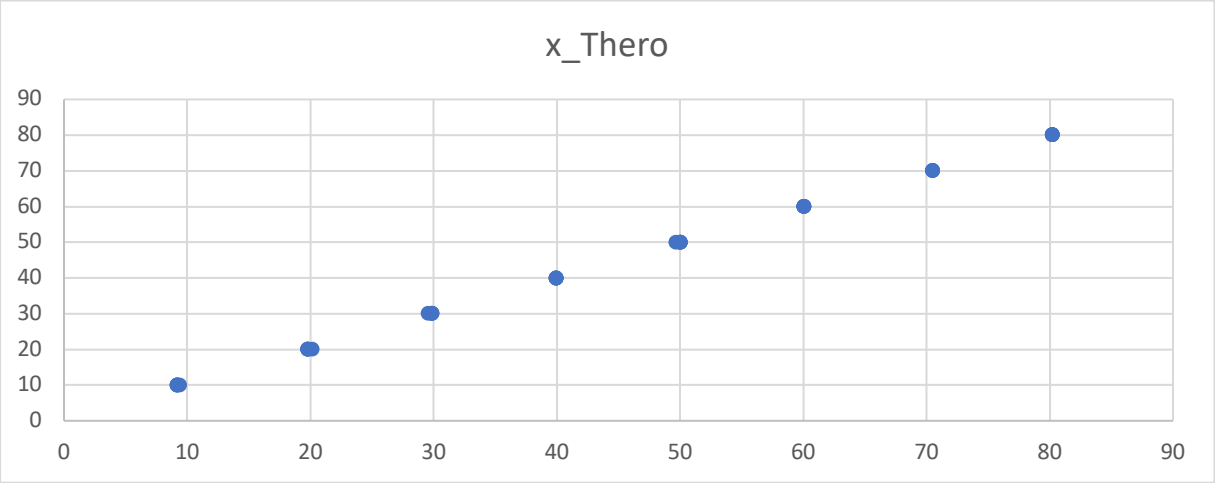


Figure 4.3- X Scatterplot

Y-THEROTICAL VALUES VS Y-PRACTICALVALUES:

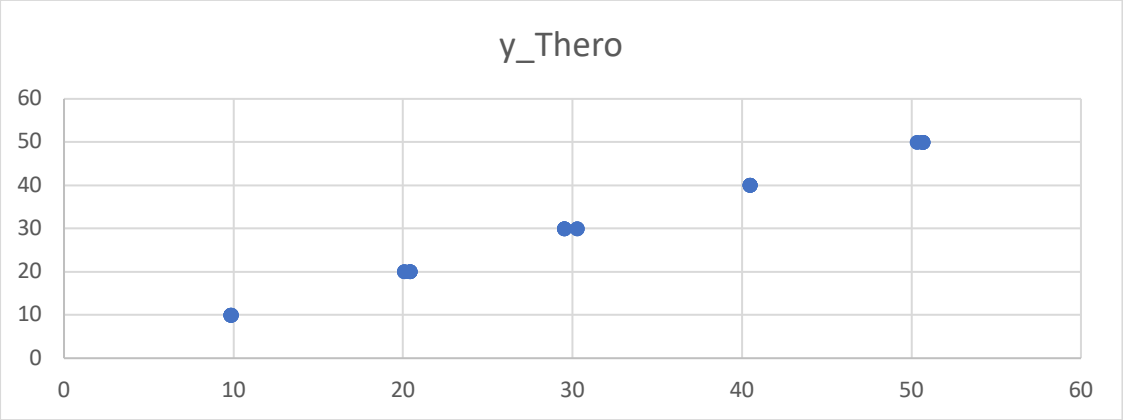


Figure 4.4- Y Scatterplot

2.2 3-D STEREO VISION:

For 3 D stereo vision 3 operations are required.

The cameras are to be calibrated like we did in 2D. For that images from the 2 cameras were first taken and used in the stero_calib function. This extracts the intrinsic and extrinsic parameters of the cameras and stores them in two separate yml files. This is used to produce point cloud images.

First the disparity image of two random images from the internet was taken then the camera was used to produce two images.

2.2.1 CHECKBOARD CALIBRATION:

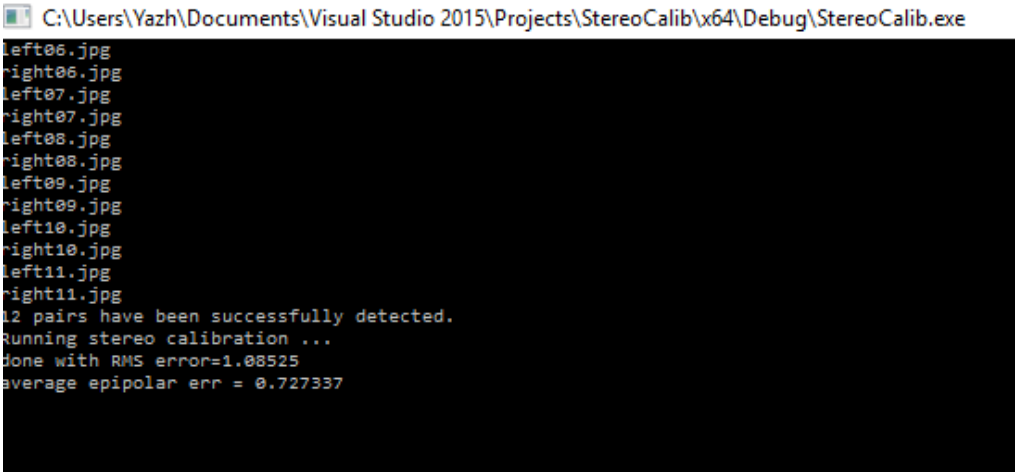


Figure 5.1.1- Calibration

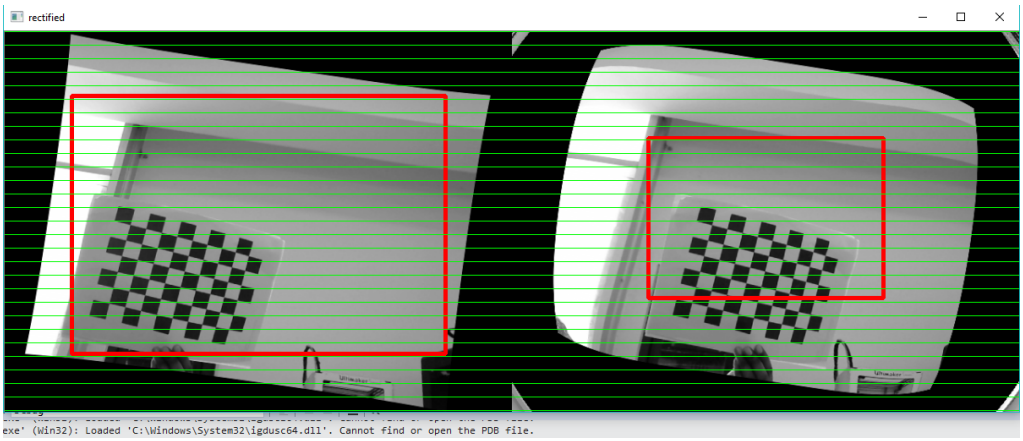


Figure 5.1.2- Calibration

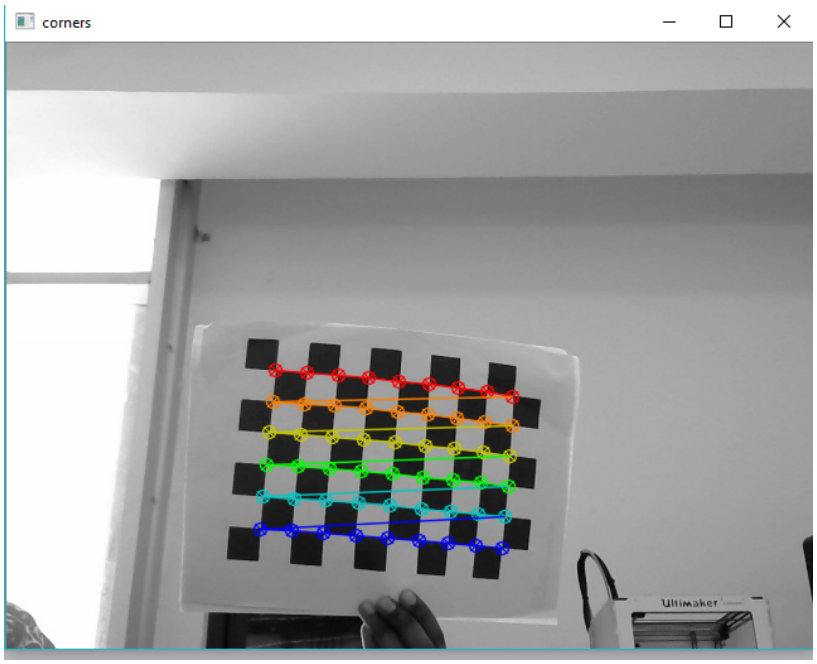


Figure 5.1.3- Calibration

2.2.2 DISPARITY IMAGE FROM TWO GIVEN IMAGES (left camera and right camera):

Left Camera Image:

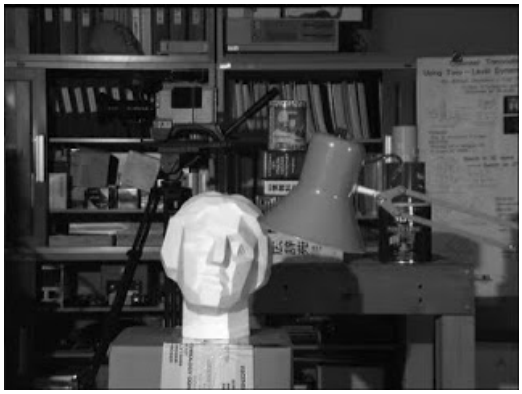


Figure 5.2.1- Left Image

Right Camera Image:

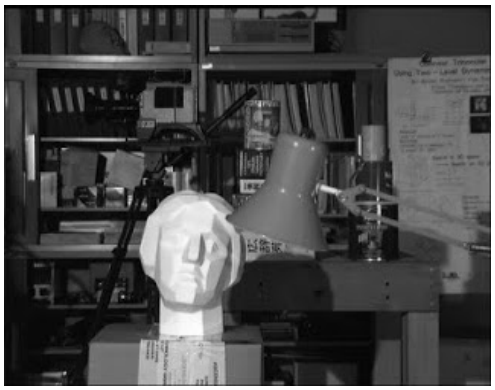


Figure 5.2.2.- Right Image

Disparity Image: As we increase the block size the clarity of the objects increases.

Block Size 21

Figure 5.2.3- Block Size 21

Block Size 25

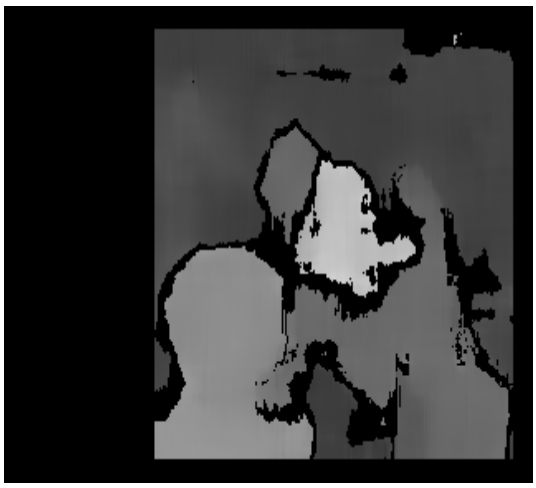


Figure 5.2.4- Block Size 25

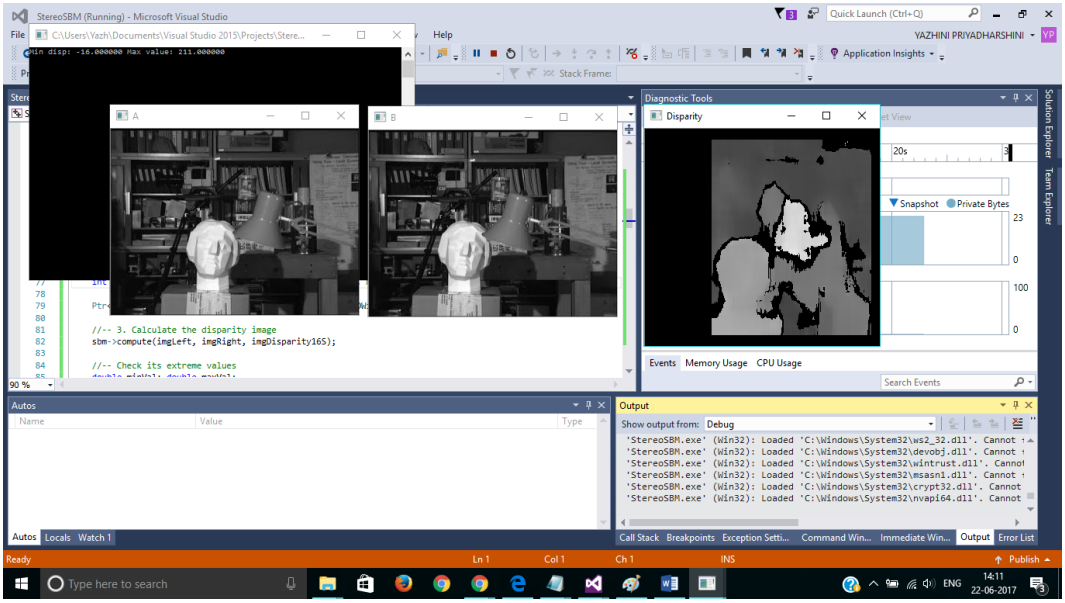


Figure 5.2.5- Block Size 27

2.2.3. POINT CLOUD OF 3-D STEREO VISION (OF ABOVE TWO IMAGES)

The point cloud image was then calculated by calling the intrinsic and extrinsic file in the disparity program.

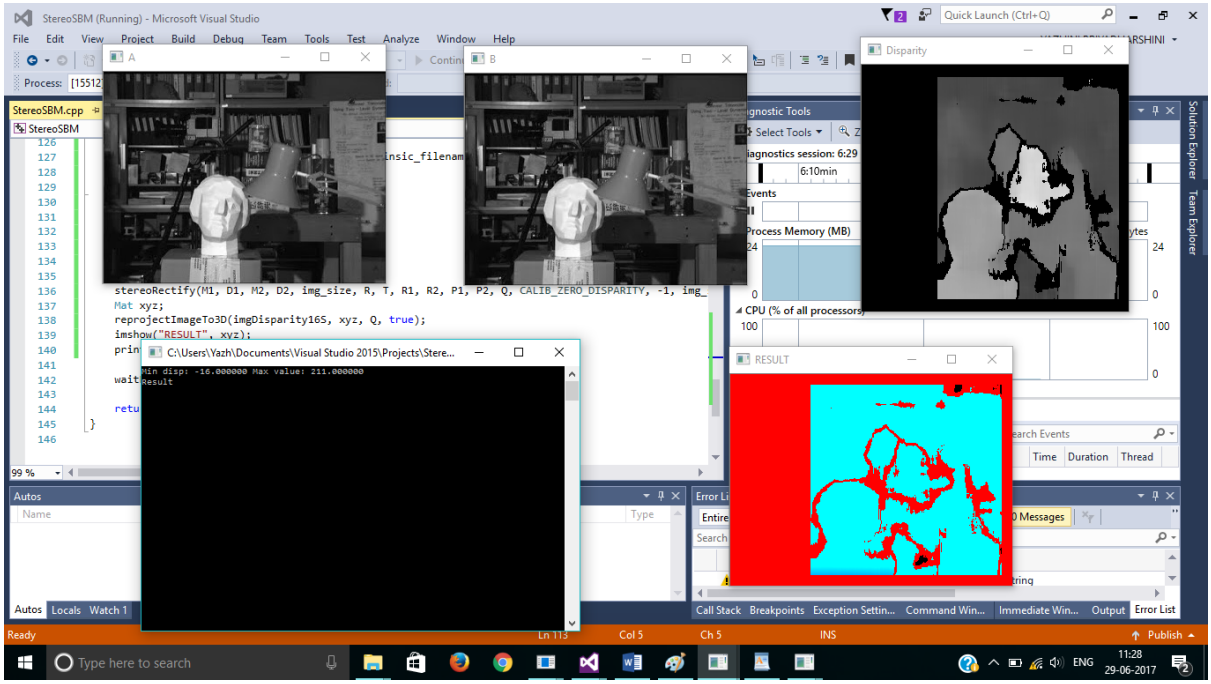


Figure 5.3.1- Point Cloud of image

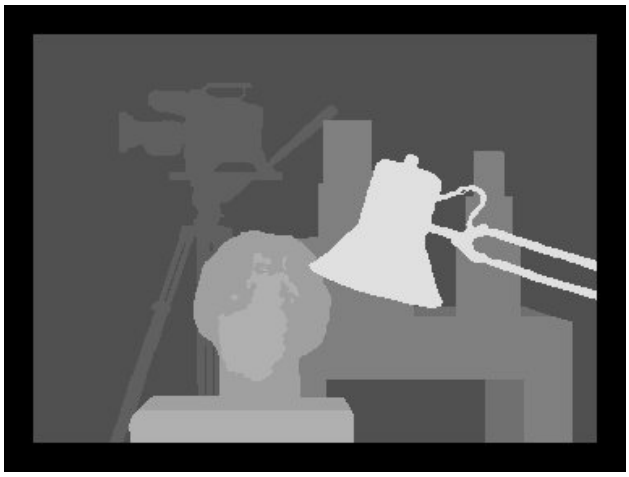


Figure 5.3.2- Point Cloud of image 1

Using the left and right camera, I took two images of myself. The disparity of that was also taken.

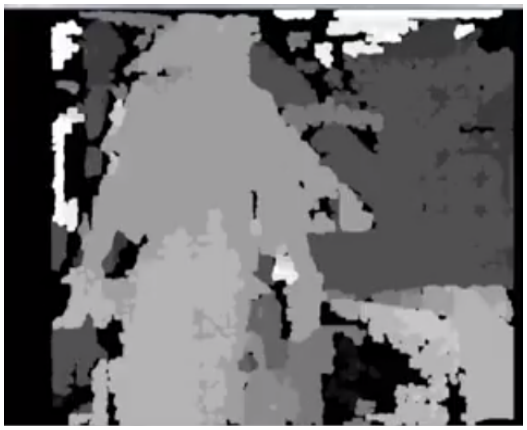


Figure 5.3.3- Point Cloud of camera image

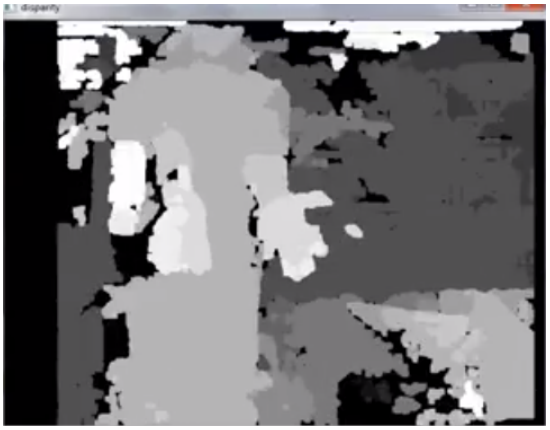


Figure 5.3.4- Point Cloud of camera image (closer)

The closer I went the brighter the image was.

2.3. COLOR SENSING:

Colour sensing has been performed by adjusting the hue, saturation, and value of each colour. First red, blue and green were sensed separately. The radius and position of the same was too calculated. Afterwards they were combined and the position of the colours were found jointly.

According to Hue, Saturation and Value:

RED:

Hue: (0,82); Saturation: (142, 256); Value: (82, 206)

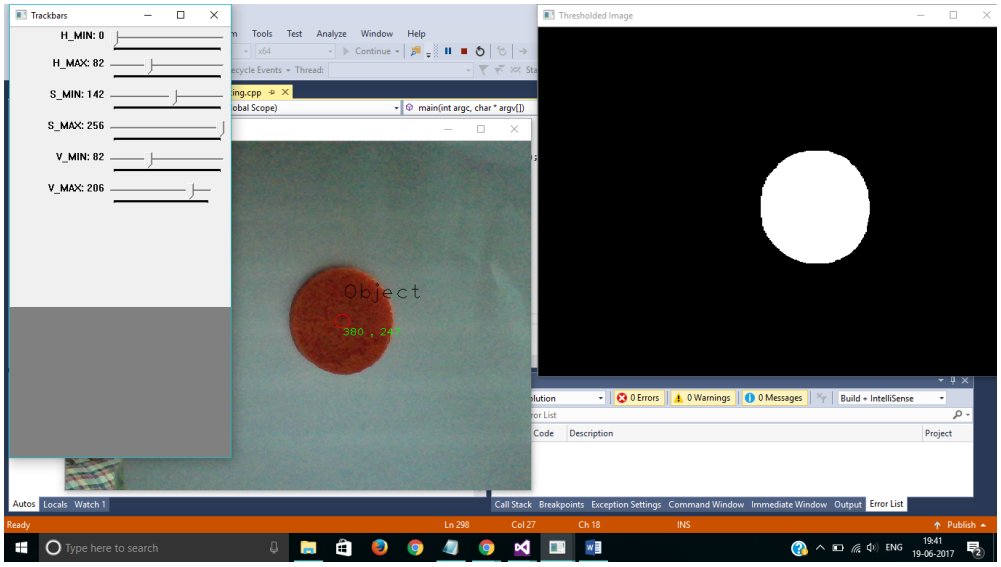


Figure 5.4.1- Red Colour Ball Detection

BLUE:

Hue: (72,147); Saturation: (107, 256); Value: (47, 206)

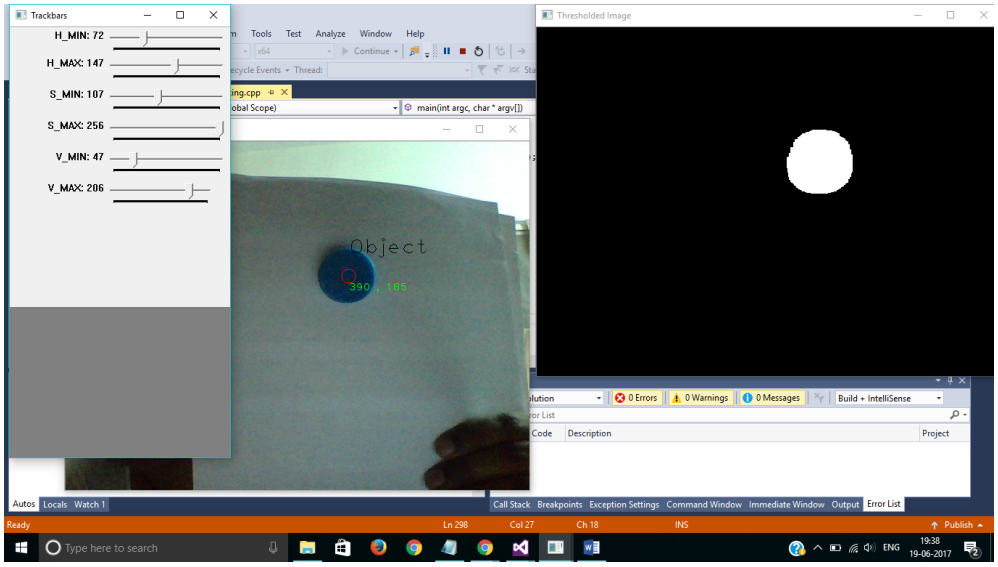


Figure 5.4.2- Blue Colour Ball Detection

GREEN:

Hue: (58,116); Saturation: (98,256); Value: (21, 206)

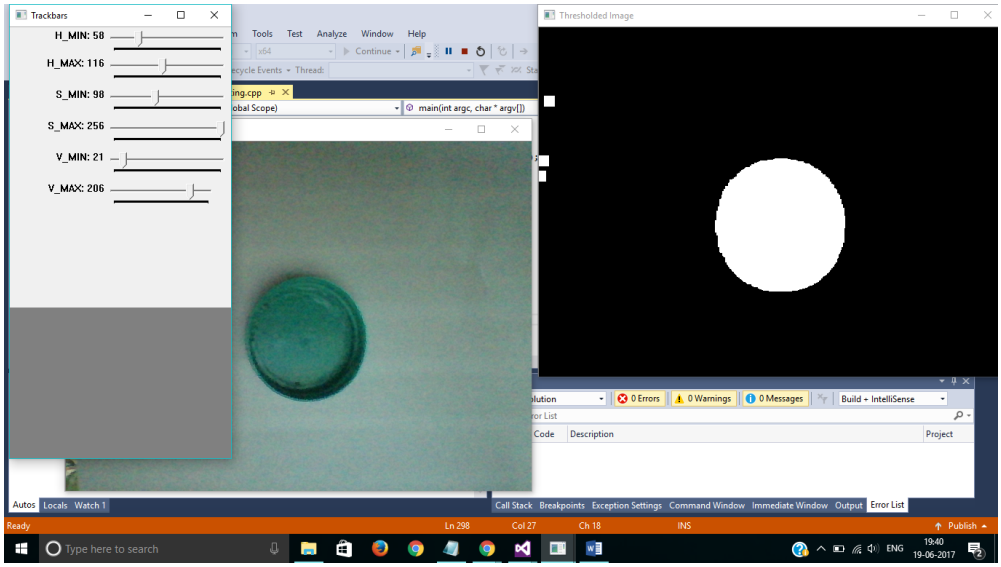


Figure 5.4.3- Green Colour Ball Detection

COLORS SENSING TOGETHER: (Blue Ball and Red ball)

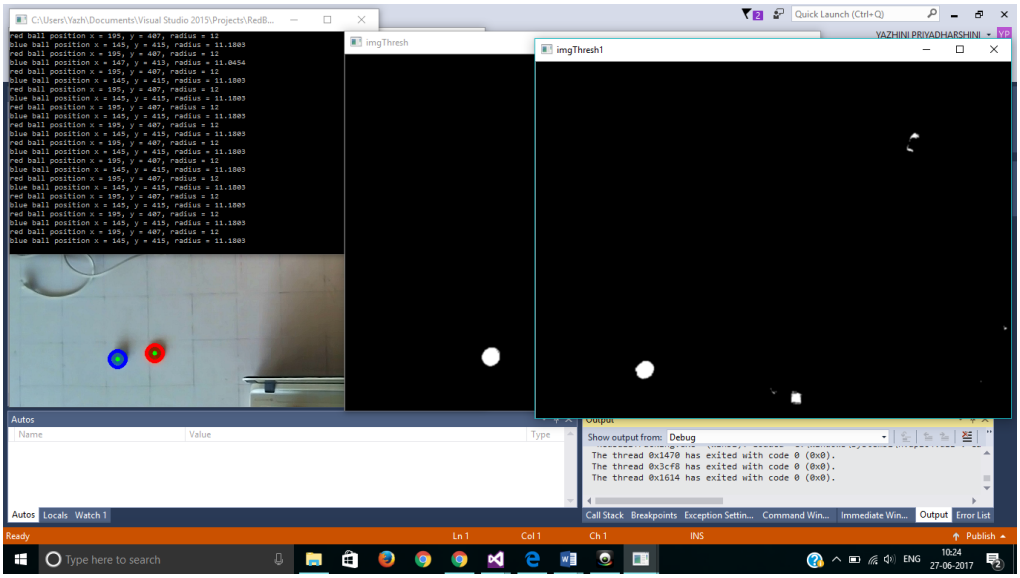


Figure 5.4.4- Red and Blue Colour Ball Detection

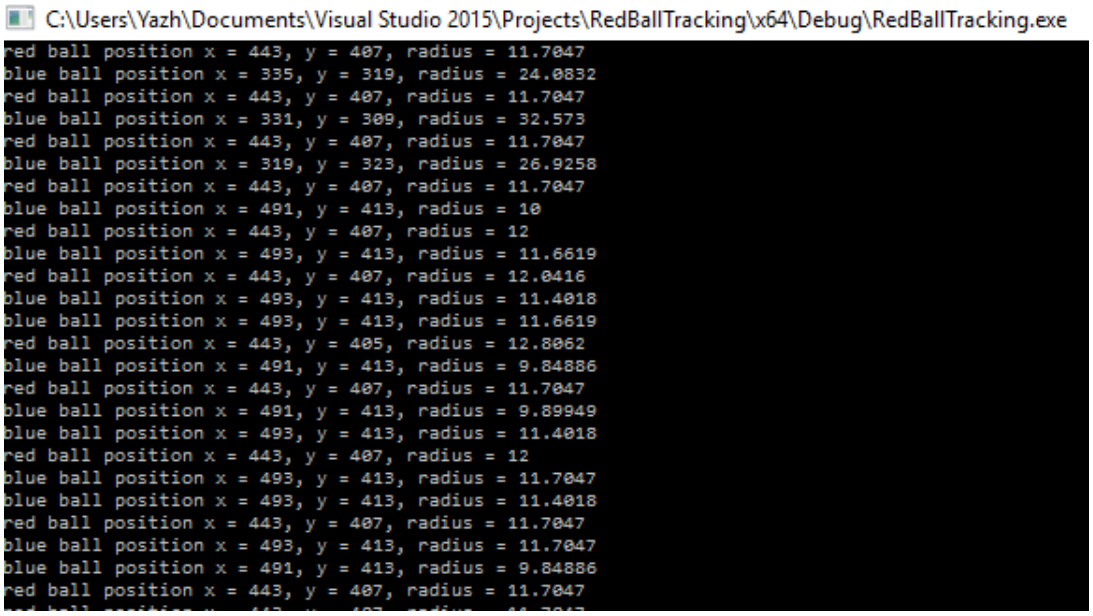


Figure 5.4.6- Red and Blue Colour Ball Detection values

3.Some Links to Documentation:

1. Documentation to OpenCV- http://docs.opencv.org/3.2.0
2. http://sourishghosh.com/2016/stereo-calibration-cpp-opencv/
3. Map function - https://www.arduino.cc/en/Reference/Map
4. Calibration-http://ais.informatik.uni-freiburg.de/teaching/ws10/robotics2/pdfs/rob2-10-camera-calibration.pdf
5. Calibration- https://boofcv.org/index.php?title=Tutorial_Camera_Calibration
6. Calibration- http://www.ics.uci.edu/~majumder/VC/classes/cameracalib.pdf
7. Calibration- https://in.mathworks.com/help/vision/ug/camera-calibration.html?s_tid=gn_loc_drop
8. Disparity - https://abhishek4273.com/2014/07/09/playing-with-stereo-images-and-depth-map/
9. http://www.jayrambhia.com/blog/stereo-calibration
10. Stereo Camera- https://erget.wordpress.com/2014/02/01/calibrating-a-stereo-camera-with-opencv