Rapsodo Assignment

Sriram Kumar K

September 2017

1 Introduction

The aim is to measure the speed vector of the baseball with given information. There are 15 frames which are captured at 240 FPS. The camera is tilted 10 degree upwards towards the sky, sitting from the ground. It is placed 4 meters away from where the ball starts.

There could be various solutions to solve the problem. In this document, I will be discussing the approach that I chose to solve this problem, its pros, cons and potential alternate methods.

The steps are to locate the ball in image, estimate the 3D position of the ball and calculate the speed vector of ball between consecutive frames. The algorithm is shown below in figure 1. First, I use Adaptive Gaussian Mixture-based Background/Foreground Segmentation Algorithm. On this Foreground segmented image, I locate circles using Hough circles. I have taken a ball image from the samples given as a template. Using Local Binary Pattern (LBP) features, I do feature matching between the circles and template. This will give me the Region of Interest (ROI) of the ball. Canny image of ROI is Convoluted with a circle of known radius to correct the circle center and radius. At this point, we should be able to locate all the circles with proper radius and center, but there could be outliers. So in-order to remove that, I used Random Sample Consesus (RANSAC) with Linear least square to fit line. Then to correct the position of the outliers I again fit a curve using Linear least squares and got the approximate x and y ROI. With this predicted ROI, I again correct the circle radius as earlier explained earlier using convolution. Now all the balls are detected with accurate center and radius. With this, the speed vector is calculated. Details about speed calculation will be discussed in below sections.

2 Foreground segmentation and circle detection

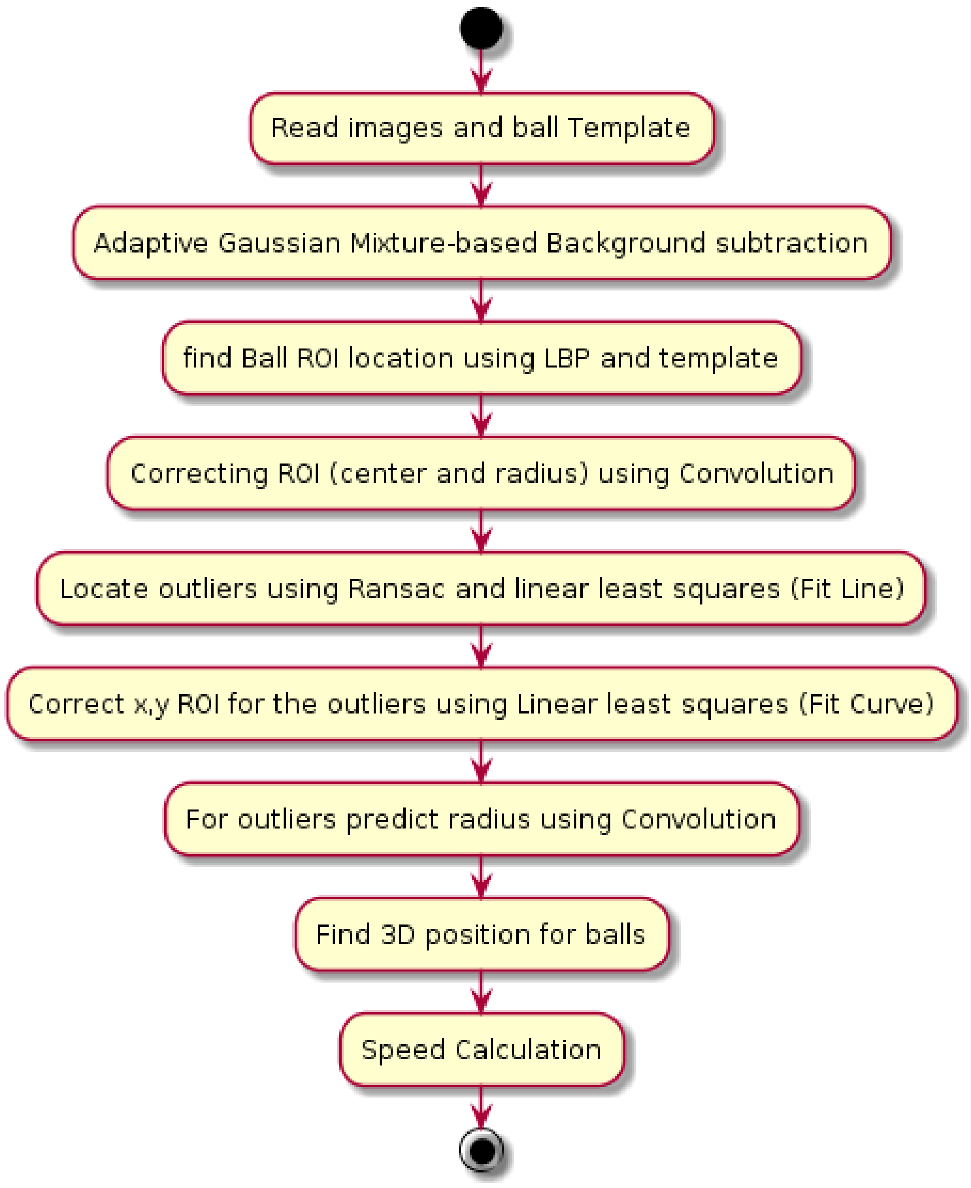
As the camera is fixed, I used background subtraction to identify the independent motion in the scene. I can use image difference with threshold, or running average over background subtraction. But I chose to use the Adaptive Gaussian mixture based background subtraction. It models each background pixel by a mixture of K Gaussian distributions (K = 3 to 5). It also selects the appropriate number of Gaussian distribution for each pixel. The weights of the mixture represent the time proportions that those colours stay in the scene. The probable background colours are the ones which stay longer and more static. 0 are the ones which are static and 255 are the ones which change. The results of this method was more accurate and reliable. The result of this method is shown in figure 2

As the ball is sphere, the projection of a sphere will be circle. So using circle detection will be ideal. Over this foreground segmented image I apply Hough circle detection. This will give me a list of circles for each frame. It might contain ball and false positives. The result of Hough circles are shown in figure 3.

3 Feature Matching / Template Matching

For us to know what is a ball, prior knowledge about the ball is necessary. So I took a cropped image of the ball from the images as a template. With this template image, I can apply template matching to find which circle is a ball. Initially I tried sum of absolute difference but it was not accurate enough due to the seams of ball and illumination change. So I used LBP for texture description. See figure 4. LBP is one of the most commonly used features for texture discrimination tasks like face, facial expressions, gesture, scene

Figure 1: Ball ball speed vector calculation algorithm - activity diagram

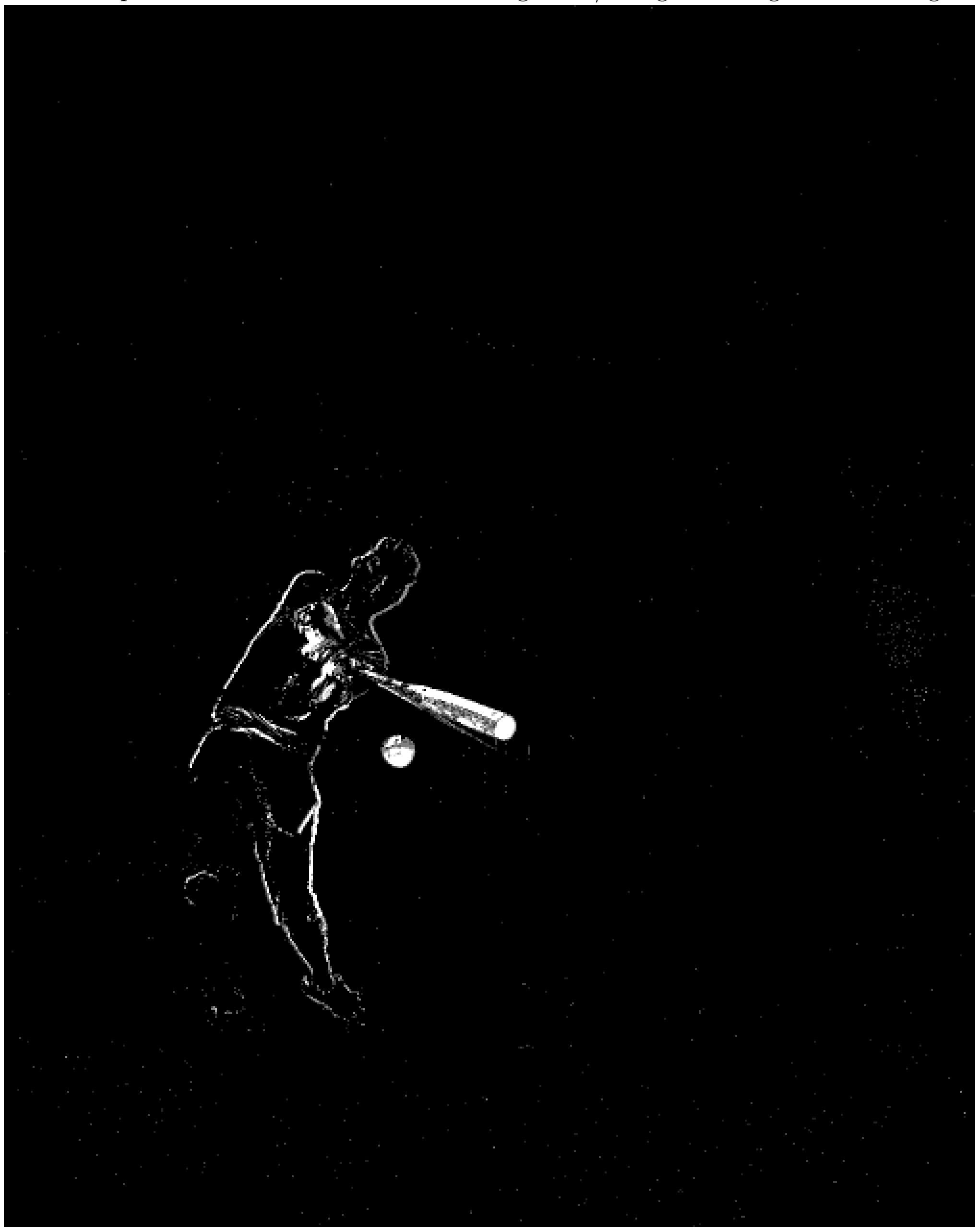

Read images and ball Template
Adaptive Gaussian Mixture-based Background subtraction
find Ball ROI location using LBP and template
Correcting ROI (center and radius) using Convolution
Locate outliers using Ransac and linear least squares (Fit Line)
Correct x,y ROI for the outliers using Linear least squares (Fit Curve)
For outliers predict radius using Convolution
Find 3D position for balls
Speed Calculation

and object recognition. LBP was conceived well before SIFT and HOG. LBP texture of template and circles are compared. The Chi-Square distance between the corresponding LBP histograms are computed and the circle with the least distance is selected as a ball. HOG can also be used instead of LBP. The result after template matching is shown in figure 5.

After this step, still we don't have accurate data to compute the speed. There can be outliers and there could be error in center and radius of the circle detected. For this reason I do ROI correction and outlier detection.

More accurate method would be to use supervised binary classification. For which many samples for ball with different size, lighting and seam are collected as positive training set. A negative training set is also collected. Take LBP or HOG or PCA or any good global descriptor of the image. We can use SVM or neural network to train a model the separating hyper plane between positive and negative samples. With this model, we can do sliding window to use the classifier to detect the balls. Kalman filter can also be used to reduce the search area and fasten the algorithm. Classification method is more accurate as we have a

Figure 2: Adaptive Gaussian Mixture-based Background/Foreground Segmentation Algorithm



model of different scenarios of the ball.

4 Radius and center correction

Now we have a rough position of where the ball is located, it is important to compute the radius and center of the ball accurately. The radius of the ball is used to compute the depth of the ball from the camera and the center of the ball is used to compute the X and Y position of the ball in the world.

Till this point we were using the temporal and spatial information to locate the ball. Now we can use the spatial information and get the accurate center and radius of the ball.

We have the rough position of the ball, with which I take bigger window around the ball (which is

Figure 3: Hough circles on Foreground image

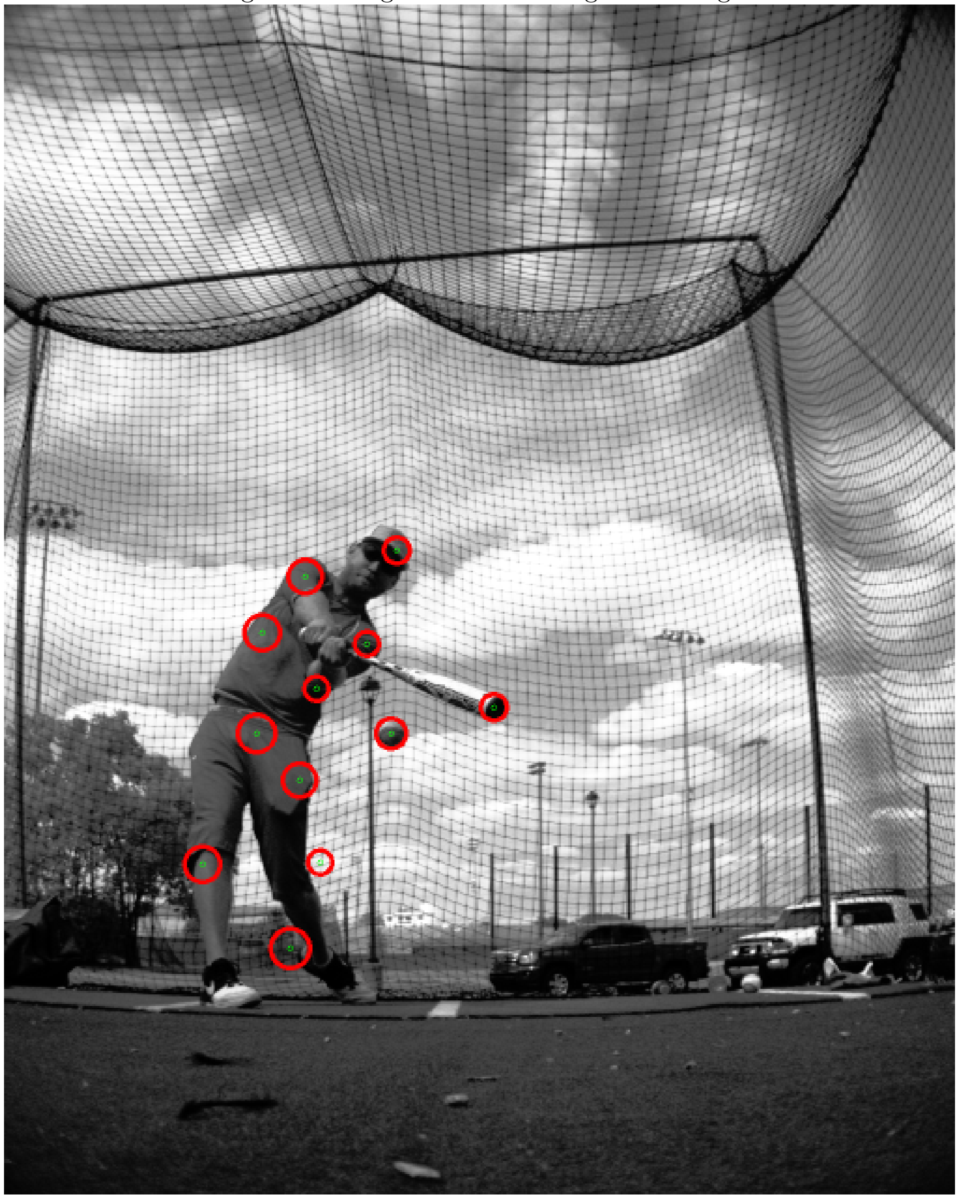
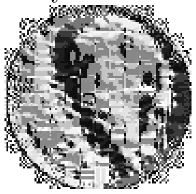


Figure 4: Local Binary Pattern image for Template



twice the diameter). Do Canny edge detection of the cropped ROI and mask it with foreground image to avoid unwanted background getting detected. With potential ball radius, I drew a circle mask. Then do convolution with different radius and size of this circle mask in the ROI to locate the circle accurately. We have a lower and upper radius and center ranges in which we iterative search for correct radius and center triplet. See figure 6 for the result of radius and center correction step.

Figure 5: Result after template Matching

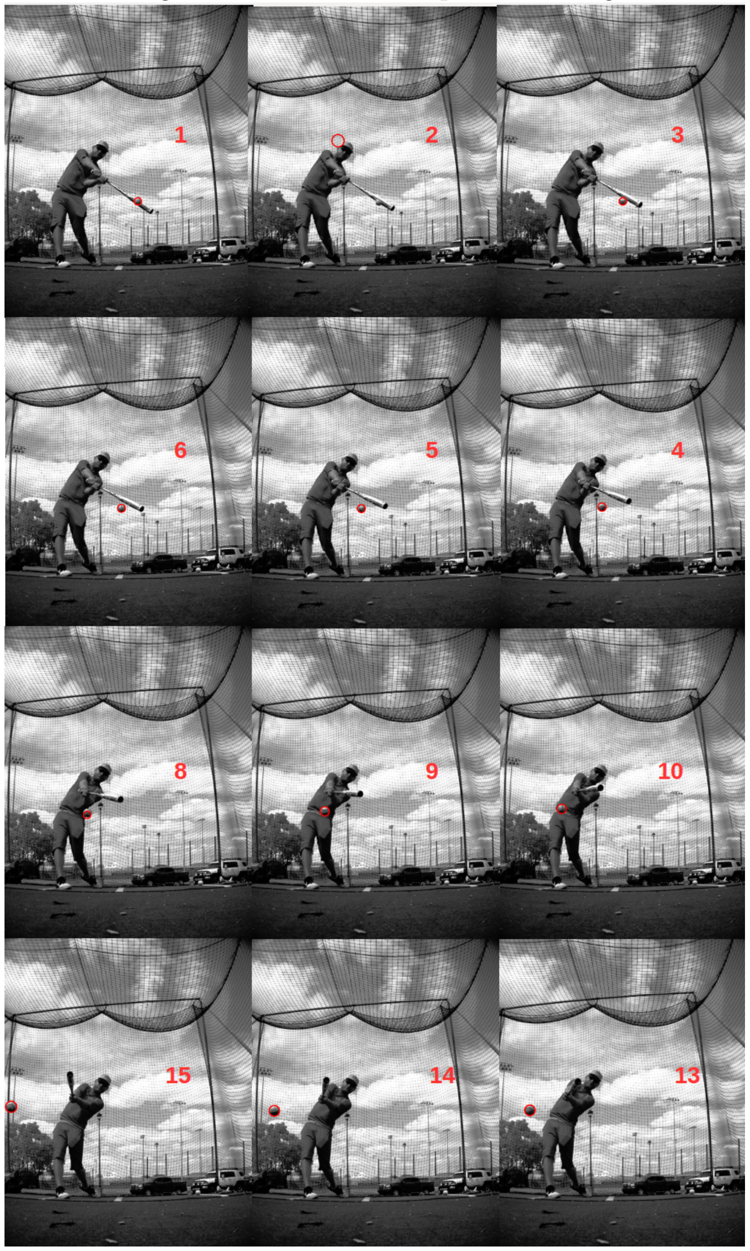
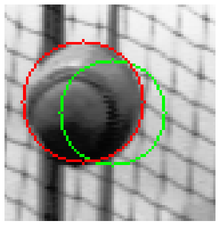


Figure 6: Result of Radius of center correction step. Green circle is circle detected by Hough circle and red circle is the corrected circle.

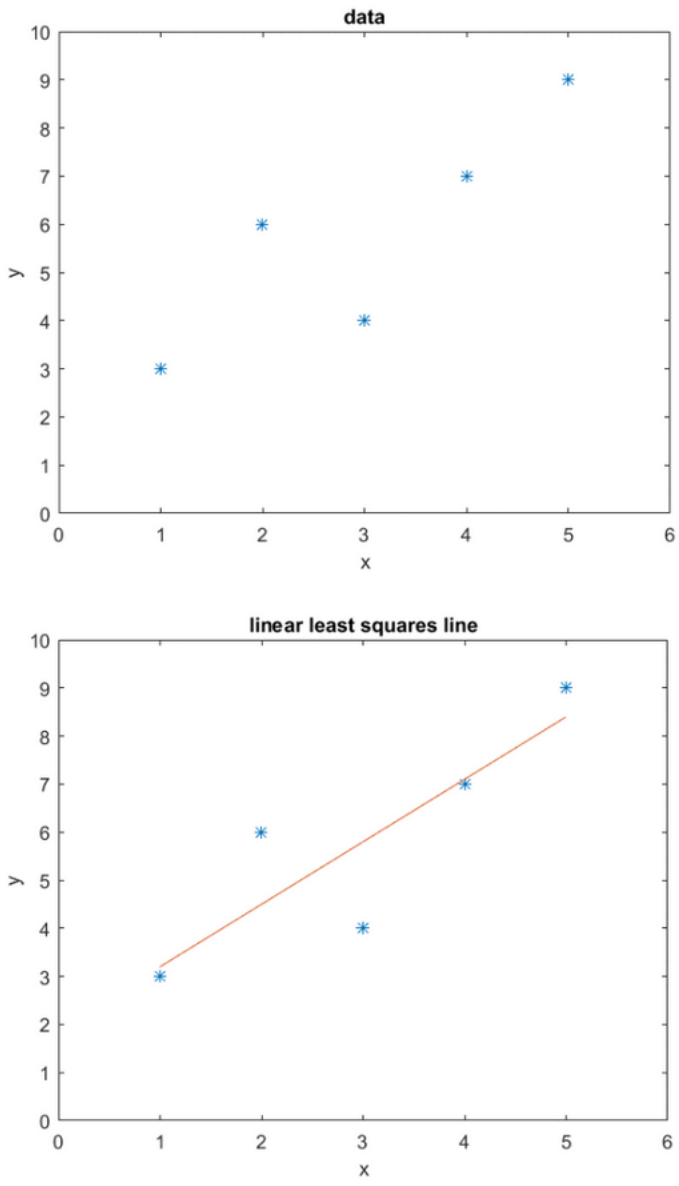


5 Outlier detection using regression

There can be outliers like the frame 2 in Figure 5. To correct this we will need to validate the relationship of the centers among the frames. I assume that the ball will travel in a straight or slightly curved trajectory.

So with this assumption, I fit a line through the result given by previous step. I use Linear least squares to fit the line (regression). To get a good optimal fit, I did RANSAC of sample size 6 and threshold at 2. The error for RANSAC is the distance from the original ball center to the line. See Figure 7 to see how line is fit on a 2D data using linear least squares. Similar to this, a line is fit for x and y coordinates of the ball position in 2D. After this, I assume all the balls should be within a certain distance from the line. The balls which are further away are outliers.

Figure 7: Linear Least Squares Example



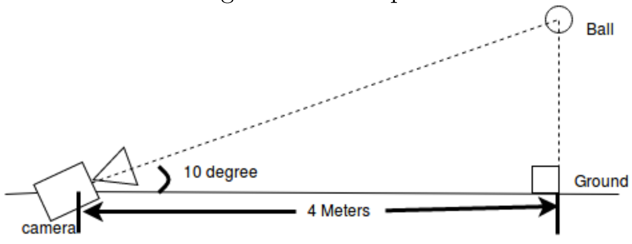
To find the right x and y position for the outlier frames, I again fit a curve for x and frame number using linear least squares. So now given a frame number I predict the x value and then with x, I can predict the y with the 2 line models above. Now position of all the balls will be corrected and again we repeat the step as in section 4 to correct the radius and center for the outliers alone.

This final data is accurate enough to calculate the 3D position of the ball.

6 Camera Model and speed vector calculation

First assumption that I have is that it is a pin hole camera model. Second assumption is that the optical center is at the center of the image. Based on the given information, I have shown the assumptions in the image 8.

Figure 8: Assumptions



As per figure 8, camera to ball distance (z) = camera to ball on ground distance / cos(10). ball to ground distance (y) = sin(10) / camera to ball distance (z). x is assumed to be at zero. so the 3D world position of the first ball is at (0,y,z) with respect to the camera coordinate system. From now on this point will be the reference point to calculate other balls 3D position.

For a given ball the depth z is calculated as we know the ball radius (0.0373 meter), focal length (8mm) and pixel size (0.0048mm).

Then we compute in image pixel. Referring real world ball as base Ball and its projection on image plain as ball. See figure 9 to see the visual description of camera model and base ball relationship.

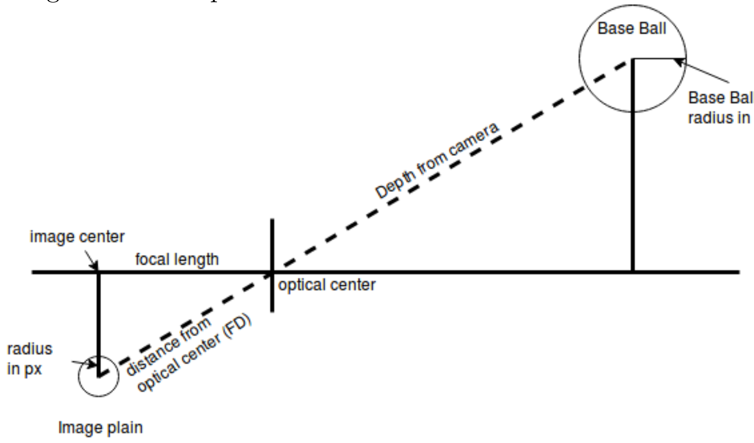
First,

$$BallDistanceFromOpticalCenter(FD) = \sqrt{MagnitudeOfOpticalCenterToBallCenterInMM + FocalLength} \quad (1)$$

then,

$$BallDistanceFromCam(D) = (BaseBallRadiusInMM * FD)/(BallRadiusInMM) \quad (2)$$

Figure 9: Description for camera model and base ball relationship



From the about 2 equation I get the distance at which a ball is from the camera (Z). In section 4 we found accurate radius and center of the ball, But it is not sub pixel accurate. So there are multiple frames in same depth. This leads to wrong calculation of speed. So in order to get a smooth speed, I applied Linear

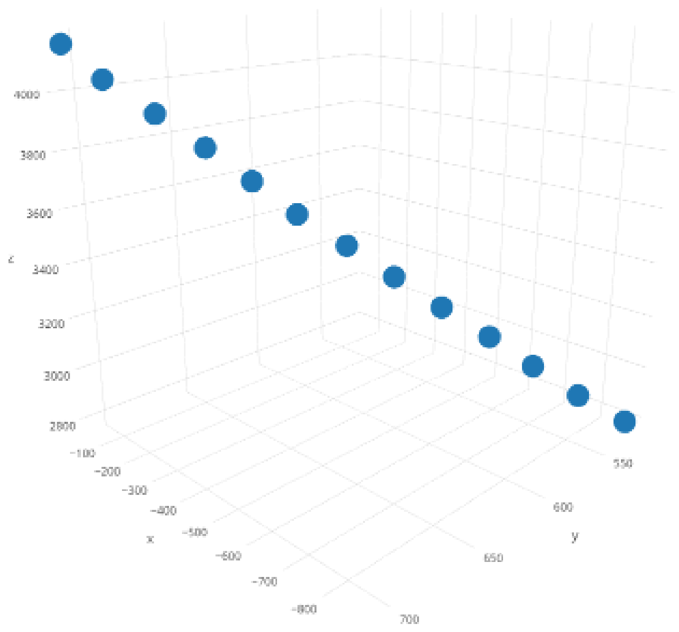
Table 1: Ball speed - Results

Frame	**X Pos**	**Y Pos**	**Z Pos**	**Speed (Km/hr)**
1	-50.0189	707.801	4155.7	92.015
2	-99.8348	695.61	4040.63	108.847
3	-165.301	679.398	3925.68	115.151
4	-233.634	664.151	3810.84	116.204
5	-299.784	649.866	3696.12	115.078
6	-367.85	636.543	3581.52	115.737
7	-437.297	620.019	3467.04	116.568
8	-509.296	604.728	3352.67	117.51
9	-581.321	588.725	3238.42	117.507
10	-654.561	572.214	3124.28	118.035
11	-724.594	557.203	3010.26	116.336
12	-799.708	541.953	2896.36	118.618
13	-871.803	523.327	2782.58	117.49
			Average Speed	**114.238**

least squares to correct the Z for each ball. We know the position of first ball that we calculated earlier. With this first ball as reference point and the known Z we can calculate X and Y position of the ball in real world. We do this iterative to all balls and get 3D position.

Once we have the 3D positions of all the balls. We can calculate speed of the balls by (distance between 2 points of ball) / time between 2 frame. As the fps is 240. The time to capture between 2 frame is 1/240. With this the speed can be calculated. The trajectory of the ball is shown in figure 10. The 3D positions of the ball, speed and average speed are shown in table 1.

Figure 10: Ball trajectory in 3D space



4000
3800
3600
z
3400
3200
3000
2800
−100
−200
−300
−400
x
−500
−600
−700
−800
550
600
y
650
700