Projectile course prediction

Sagi Habani; Yaron Hever; Ziv Remba

ABSTRACT

IN THIS PROJECT WE HAVE BUILT A SYSTEM THAT DETECTS A RED BALL IN MID-AIR, PREDICTS ITS REMAINING COURSE, AND ILLUSTRATES ITS PREDICTED COURSE ON 2-D ORIGINAL IMAGE AND 3-D GRAPH, IN REAL-TIME. WE ASSUME THE BALL SUBJECTED TO CONSTANT EXTERNAL FORCE ONLY (GRAVITY FORCE) THROUGHOUT ITS WHOLE MOTION. THE SYSTEM CONTINUES TRACKING THE BALL THROUGHOUT ITS ENTIRE FLIGHT AND MAKES ADJUSTMENTS TO ITS PREDICTED COURSE IF NECESSARY.

THE SYSTEM INCORPORATES STEREO VISION AND SEVERAL DETECTION METHODS.

COMMERCIAL USES COME TO MIND: THE SYSTEM CAN BE EMPLOYED IN THE WORLD OF SPORTS AND SPORTS MEDIA. VIEWERS OF BASKETBALL, SOCCER AND FOOTBALL CAN ENJOY A REPLAY (OR THE LIVE GAME ITSELF!) WITH THE TRAJECTORY OF A SHOT MADE SUPERIMPOSED. IN FACT, A QUICK INTERNET SEARCH WILL REVEAL THAT COMPANIES THAT OFFER SUCH FEATURES DO EXIST (EVEN IN ISRAEL) AND ARE ENDURING.

MOST IMPORTANT IS THAT THE IDEA OF DETECTION-CONTINUOUS TRACKING-PREDICTION CAN BE EMPLOYED TO MANY OTHER (COMMERCIAL) USES.

1. The overall system at a glance

The system was built to produce results for distances of about 0-10 meters, because farther than that the ball will be too small to be caught on camera.

The system handles well ball speeds as high as those that are generated by a kick.

The system consists of four main stages (see fig 1):

1. A preparatory stage of stereo calibration
2. Detection and tracking of the ball in real-time (for each camera separately)
3. Calculating the predicted course using stereo triangulation
4. Illustrating the predicted course on 2-D original image and 3-D graph

Stages 2 to 4 are repeated endlessly in real-time.

We also built a function which enables us to record the motion and display it in slow motion for better viewing.

Stereo vision is needed in this project because without two cameras, depth cannot be inferred and hence a 3-D trajectory cannot be computed. The stereo triangulation also requires the intrinsic and extrinsic parameters of the cameras, which are found in stage 1 using the camera calibration toolbox for Matlab which was developed by Caltech.

Detection yields the ball's center of mass' pixel coordinates in each image (left and right, which were of course captured simultaneously). Using these coordinates, epipolar geometry and the camera pinhole model, a "world" 3D reference frame coordinate is extracted for the ball for that instant.

Once two 3-D coordinates are extracted, a course can be predicted using Newton's laws.

Because of working in real time, a significant challenge we encountered was increasing frames per second as much as we can. This has been done mainly by decreasing exposure time (under illumination constraints) and “finding algorithm” based on predicted course (as will be depict in chapter 3), till we achieved approximately 27 frames per second.

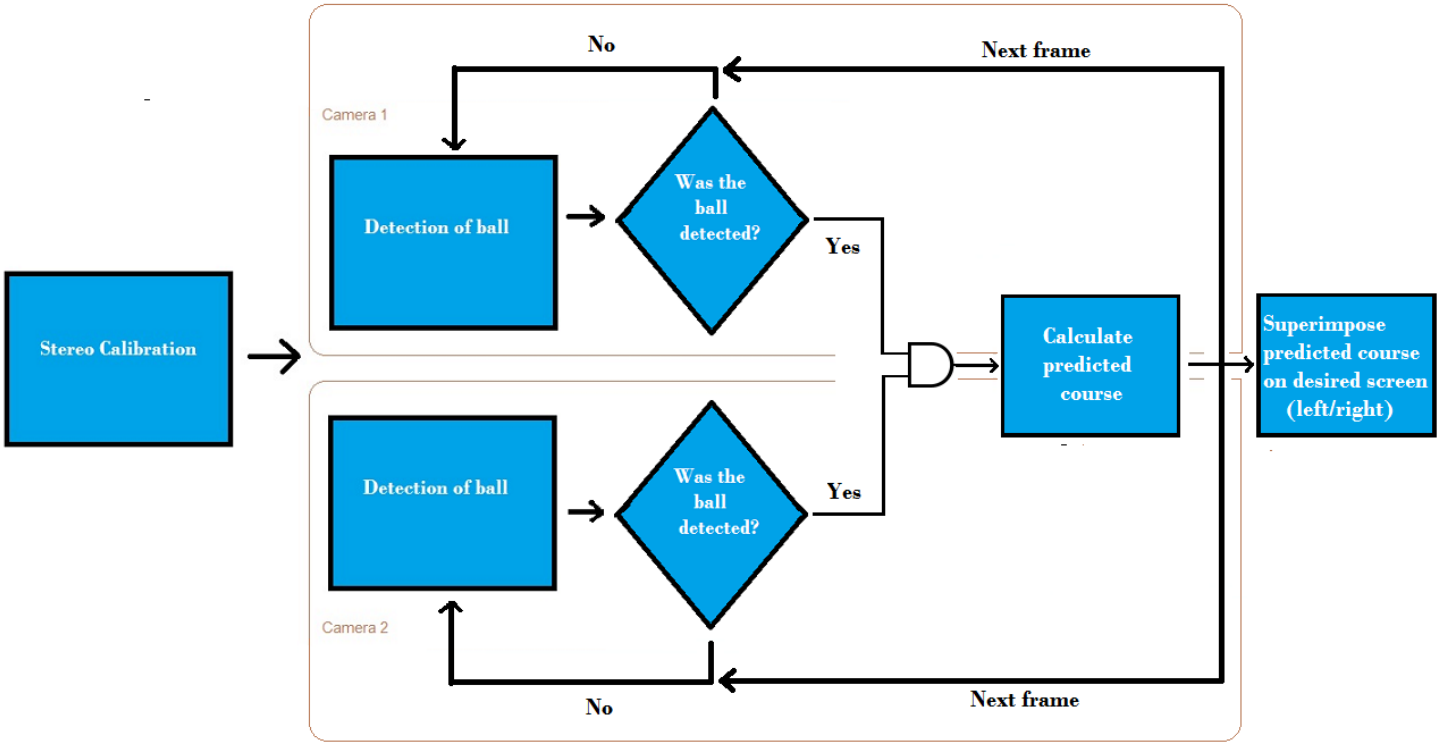


Fig 1. Flow chart describing the system's work flow

Fig 2. a stereo image with the ball's center of mass marked with a cursor and the detected perimeter of the ball contained in a box. (a) left camera (b) right camera. As seen in this image, although there are several red objects, only the ball is tracked.

2. Stereo configuration

Stereo vision is essential, without it we cannot derive the 3D coordinate of the ball.

The transformation from two 2D pixels (one pixel from the left camera and one from the right camera) to a single 3D location vector requires the intrinsic and extrinsic parameters of the cameras. The predicted course is calculated in 3D, and afterwards projected back into 2D suitable pixel of the left camera , this back projection also requires the intrinsic and extrinsic parameters, therefore it is essential that the calibration will be as precise as possible otherwise the system will give faulty results at each step of the process.

2.1. Calibration process

First of all we created a function which enables us to take 20 pairs (left and right cameras) of pictures of the same scenes simultaneously. Next we calibrated each camera to acquire the intrinsic parameters, using the suitable pairs of pictures. Last step in the process was stereo-calibration of those two cameras together to acquire the extrinsic parameters (the pose between them).

2.2.The physical problem: Pose recovery; stereo triangulation

Pose extraction: Given a set of pictures of the same scene, we need to extract the pose of the cameras regarding each other (these are called the extrinsic parameters).

“Stereo triangulation”: given two pixel coordinates of a corresponding point (the ball's center of mass) from the left and right camera, this function extracts the ball’s 3D location vector (see fig. 4).

“Project points”: we build a function which gets one 3D location vector as input, and returns the suitable two 2D suitable location pixel in the chosen camera image.

2.3.Essential matrix estimation and Epipolar geometry

The R matrix and T vector which relate the two cameras were found using the camera calibration toolbox develop by Caltech (see fig 3).

2.4.Calibration Verifying

In order to verify calibration’s validation we used some simple tests. The first one was locating the ball in a well measured location regarding the camera position, and check if the computed location (using “Stereo-triangulation” function) is the same as the measured location. The second one was to project backwards the 3-D computed

location vector into two 2-D pixels (using “Project-points” function) and check if we get the same first pixels (where the ball was represented in the left/right camera).

2.5.What have you learned?

We learned how to calibrate a digital camera and how the intrinsic and extrinsic parameters fit in the model.

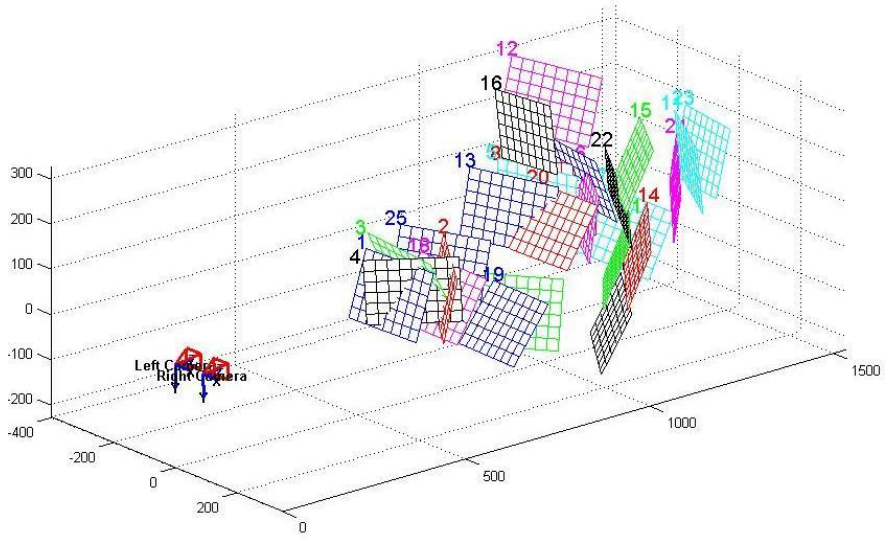


Fig 3. Depiction of the extrinsic parameters calibration. Each colored plane is shown with respect to the two cameras.

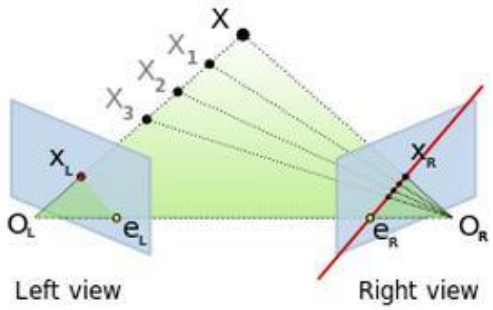


Fig 4. Epipolar geometry: Based on the geometry in the figure, constraints are derived. These constraints are called epipolar geometry and they dictate the relations between corresponding points in the right and left images. In the images, the points O represent the pinholes of the cameras.

3. Detection

The task of detection is the task of yielding the ball's center of mass in the pixel coordinates cameras.

This task was a challenging one, in fact the one without a solid algorithm to solve it. We tried many methods to tackle the problem but none seemed to work until finally we found an ingenious way. Our first attempts were naive – subtracting the background image from the current image to obtain a (very noisy) foreground, which was supposed to be just the ball (at first we reduced the problem to a static background, which will later be dropped). To that image we applied the median filter (which is a good cleanser of "salt and pepper" noise), morphological operations and finally, Because we know we are looking for a ball, we tried the circular Hough transform on the image (by the “imfindcircles” function, which actually works on the thresholded, Absolute value of the gradient of the image, and not the image itself) but it gave horrible results.

Because that didn’t work we tried a different approach. What do we know of the ball besides its shape? Its color. Our last and successful algorithm – subtract the intensity image from the R component, and threshold the image to receive a binary image. To that binary image we preform blob analysis to extract the ball's center of mass. The results were excellent and fast!

2.6.The physical problem: Detection

For reasons mentioned above, the task of detection was diminished to detecting a red ball appearing over a non-red containing background. Therefore we assume that there will be only a single detected object in the image - the ball. In reality, more than one blob might be present.

The detection process is based upon the ball's color. The main algorithm’s stages are shown below using example for better understanding:

The original image:

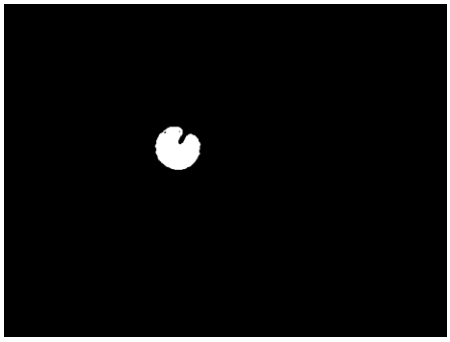
Subtracting the grey scaled image (on the right) from the red component of the image (on the left):

The result is:

Using median filter for cancelling noise we get:

Inverting into binary image with threshold 0.18:

Removing all the small objects (less than 300 pixels):



At last we using “region props” function we get center of mass and bounding frame.

2.7. Difference photo and blob analysis

The detection process is founded on subtracting the intensity image from the red component. It is a well-known method, which we first saw in the lectures and homework #1 and later on projects from past years.

Because after thresholding the difference photo many blobs are present. Blob analysis is held (by the “bwlabel” followed by “regionprops”) from which we take the blob with the biggest area as the detected ball.

2.8. Detection algorithm based on Predicted course

As we mentioned before, one of our challenges was increasing number of frames per second. In order to achieve that, we used the trajectory prediction in order to increase detection algorithm efficiency. After calculating the trajectory (as will be shown next chapter) the detection function is activated only on a frame of 100X100 pixels where the ball expected to be. If the ball still wasn’t detected, then the detection function is activated on the whole image. Assuming the whole image consists of 300,000 pixels, that way we decrease our area of searching 30 times less, to only 100x100=10,000 pixels.

2.9.What have you learned, and what left open?

We have learned much about detection – which the straight forward way may not always work (that is why we reduced the problem to red colored balls). We found that Circular Hough transform is completely unreliable; we learned allot about noise reduction by morphological operations and the median filter which give very nice results.

3. Trajectory calculation

In This part we use two series (length m) of 2D pixels (left/right camera) which describes the detected ball's location in the past. In order to get the real 3D location vector we use "stereo triangulation" function (from the calibration toolbox).

As we know from classic physics, the course of flying object (subjected to gravity force only) must be in a parabolic form- which dictating 3 unknowns in each axis (totally 9 unknowns).

Analytically this problem is very simple: Using 'm' (where m can be at least 3) last 3D samples of the detected ball's location and suitable times, we can calculate the 3 parameters of the parabola (here in X-axis):

$$t = [t_1, \ldots, t_m]$$

$$x = [x_1, \ldots, x_m]$$

$$A_x, B_x, C_x \in R^1$$

$$[A_x, B_x, C_x] \begin{bmatrix} t^2 \\ t \\ 1 \end{bmatrix} = x$$

$$\Downarrow$$

$$[A, B, C] = x * pseudo\_inverse(\begin{bmatrix} t^2 \\ t \\ 1 \end{bmatrix})$$

The aforementioned equations are used similarly in order to find the 3 parameters in Z,Y axis too:

$$A_y, B_y, C_y \in R^1$$

$$A_z, B_z, C_z \in R^1$$

In fact we encountered big deviations because of numerical reasons and noise. On one hand our time series of samples should be very condensed. For example If we managed to reach 25 images per second, it means that dt=0.04 sec, where dt is the time difference between two consecutive images. On the other hand it means that our time matrix become very singular with huge condition numbers. For example we checked in matlab the next matrix of time:

$$T = \begin{bmatrix} 100^2 & 100.04^2 & 100.08^2 & 100.12^2 & 100.16^2 \\ 100 & 100.04 & 100.08 & 100.12 & 100.16 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

And got cond(T)= 3.7480e+10, means that every small change in input (because of noise) will cause huge change in output. In order to decrease it we realize that it is better to use time serial which begins in t_0=0. For example checking the suitable matrix of above we got:

$$T = \begin{bmatrix} 0^2 & 0.04^2 & 0.08^2 & 0.12^2 & 0.16^2 \\ 0 & 0.04 & 0.08 & 0.12 & 0.16 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

And cond(T)= 379.5018.

It didn't solve the problem. We still encountered lots of noise in our unknown.

At last we decided to minimize the problem by forcing the acceleration in each axis to be as we know according to classic physics, namely:

$$A_X = 0 \qquad A_Y = -4.9 \qquad A_Z = 0$$

For doing that we had to coordinate between the Cartesian axis system of the left camera (reference camera) to the Cartesian axis system of "earth".

After having the 3D parabolic trajectory we drew it in 3-D graph. Then using "project points" function (in order to project it back again) we drew the trajectory on the 2-D plane of the left camera image.

3.1.The physical problem: Trajectory calculation

I am receiving location of the ball and from it calculating a trajectory according to Newton's laws.

3.2.What have you learned, and what left open?

We learned a lot about the pinhole camera model and the intrinsic parameters with which we projected a point in space to the image plane corresponding to it. Also we learned how noise problem become significant in real practical problems.

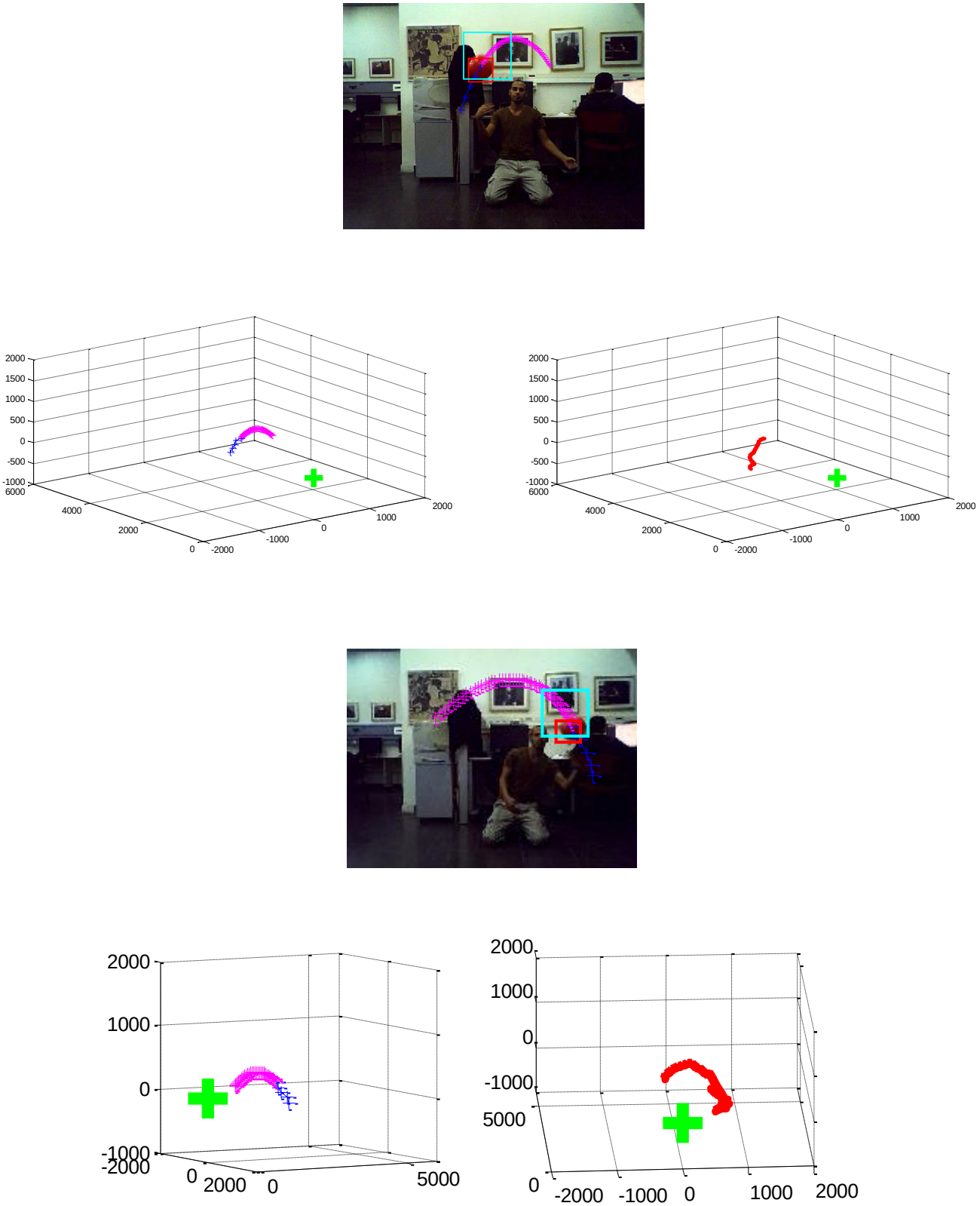
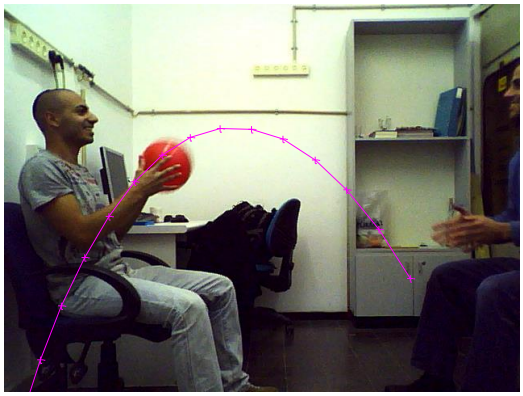


Fig 5. Trajectory course depicted on both left camera image and in 3-D graph. The green cross symbolizes the camera location. The blue course symbolizes the last 5 samples of the ball in past, and the pink course symbolizes the future predicted trajectory. The red frame is where the ball had been detected in present frame, and the azure frame shows the area of searching the ball in next frame, based upon expected location of the ball.



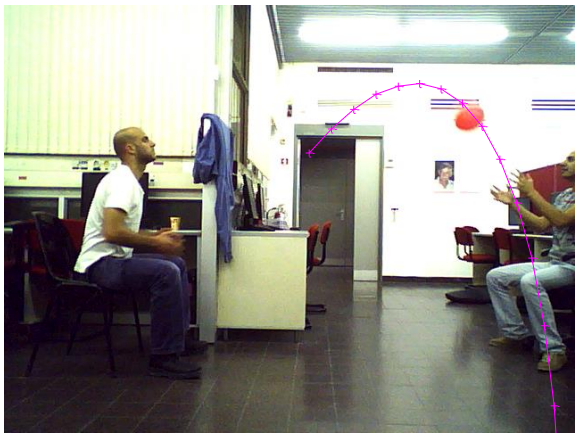


Fig 6. Snapshots of the ball with predicted course projected on left camera image.

4. Summary:

Overall the project turned out to be a lot harder than we first thought it will be. But we achieved a fine product.

Future developments may be to use smarter (and consequently more computationally complex) algorithms to differ the ball from the background. Under the assumption of a slowly changing background, one might use the MOG approach, albeit it demands a number of initial frames for initializing, which might make it incompatible to a video of a ball in the air for only a few short seconds.

5. References:

[1]- Camera calibration toolbox. caltech
http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example5.html

[2] – Epipolar geometry article , Wikipedia
http://en.wikipedia.org/wiki/Epipolar_geometry