IACV Project

Ingvild Christoffersen Fleisje, Zhengchen Xu and Simen Piene Fløtaker

February 15, 2022

Contents

1 Introduction and motivation

This project has been concerned with detecting and tracking the balls in a game of Bocce. In this game there are nine balls and two teams. There are four balls for each team, and a smaller ball ("pallino"). The smaller ball is thrown onto the court, after this each team will throw their balls, trying to get as close to the pallino as possible. There has been two main goals for this project. Firstly, estimating the 3D-trajectory of the balls from they are thrown to they hit the ground. Secondly, estimate the 2D ball positions in the goal region of the court.

1.1 Motivation

Estimating the 2D ball positions would allow the program to keep track of the intermediate score of the game. A motivation for creating such a program could be to increase spectator engagement in the sport. Televised sport often shows data such as the current scores, predicted outcomes and overviews of the game status. In addition, such a program could aid with difficult measurements. If many of the balls have similar distances to the pallino, it might be difficult or time consuming to correctly measure the distances and calculate the score. If the program is made accurate enough such measurements could happen in real-time, without human interaction. Estimating the 3D trajectory could of course also be used to digitize the game and display for audience. But it could also serve as useful information for the players, to practice and improve their techniques. Being able to analyze ones throw in 3D, could give insight into what was done right or wrong in a throw.

1.2 Current approaches to similar problems

In recent years, several ball detection cutting-edge technologies have been put into practice, for instance, tennis ball detection achieved by implementing convolution neural network has been proved to work very well with high accuracy in more complex systems, it can decide whether a ball is being observed in every single frame. Back to the years when AI technology had not been widely used, ping-pong ball detection could be achieved by Hough circle transform, and ball position can be detected based on Kalman filter. Overall, since so many methods have been used in ball detection, we are trying to use an approach more in line with the course content.

2 Ball Detection

When detecting the Bocce balls, since they have the same round shape, we need to use the information of their colors to identify the different balls. Since the balls all have a single bright color, this information was also used to detect their locations. And due to the reflection of nearby environment, such as sunlight, the color of balls might change slightly, so the masks of balls can not be always detected in the round shape, then we need to define a method to reshape the contour of mask as much as possible like a circle. During this process, we need to add some constraints to make sure that we only detect the desirable balls so that the balls positions in different frames can be easily award.

2.1 Approach

The ball detection was implemented by mainly three steps. Firstly, we define the color of all the balls, and mask out these colors from each frame. Secondly, we look for contours in these color masks. Lastly, we add constraints on the shape, size and movements of the balls to select the appropriate contours belonging to each individual ball.

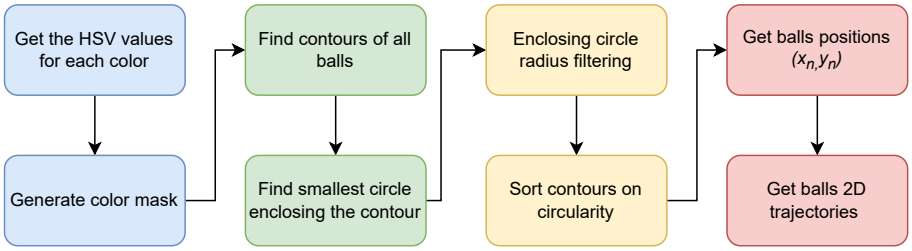

Figure 1: The ball detection process

2.1.1 HSV color model

To track the individual balls in each frame, we want to use their color properties. Ideally we could find their exact colors experimentally, and then mask out the pixels containing only those values. However, the lighting and environment will change their appearances slightly, so we need to define a range of colors to be masked out. To create such a range the frame is first converted to HSV color space.

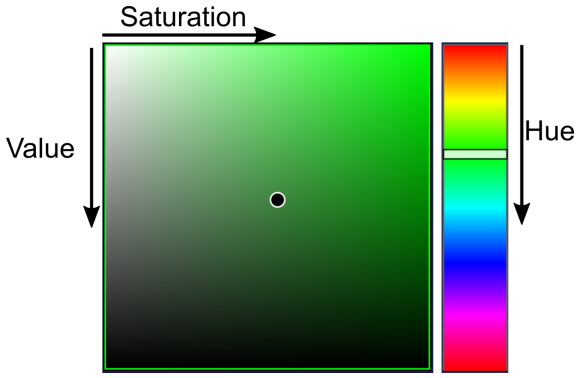

Figure 2: A visualization of the HSV color space

The HSV (Hue Saturation Value) color model is an alternative to RGB. A color represented by HSV can be seen as a mix of a pure color, white and black. For reference see

figure 2. It has three parameters which (for conceptual understanding) can be defined as:

- **Hue:** The pure color
- **Saturation:** The ratio color/white
- **Value:** The ratio color/black

This color model is practical when we want to mask out known colors. As the differences in the ball colors will mainly be in lightness and darkness, not in its pure color. To perform the HSV color masking we follow three steps. First, find the HSV value of each ball in nominal conditions experimentally. Second, create an upper and lower bound in HSV range for each of the colors. Third, create a binarized version of the frame that is white only in regions within the given range. In opencv, the mask is created by the function `inRange` taking in the frame and the HSV ranges. In figure 3 all the five color masks are combined to demonstrate the result of this process.

Figure 3: A frame and the combination of all its color masks

2.1.2 Contour detection

After obtaining the color masks of balls, all the points on the boundary of balls can be detected as a contour, typically, a specific contour refers to boundary pixels that have the same color and intensity. In opencv, the function `findContours` will return a list of all the contours in a binary image.

Considering the situation of random noise in the court, chances are that there will always be some objects sharing color with our detected balls. Therefore, we need to have some method for selecting the correct contours (or no contours) out of all the contours in the color mask. Two important attributes of the contours we are looking for should be **circularity** and **radius**.

Evaluating the radius of a contour can be done by finding the smallest circle containing all points of the contour and using its radius. The center and radius of such a circle can be found with the function `findMinEnclosingCircle` in opencv. Doing this for all the contours in a mask, we can filter out only the contours with radius inside a certain range. This certain range is derived from the true radius of the ball. In the image however, the balls will change their radius based on how far away they are from the camera. We therefore define the furthest and closest position we expect a ball to be in, with respect to the camera. Using the camera projection matrix, found later in section 3, we can find the radius of the balls, in image space, at the extreme positions, thus lower radius γ_{min} and upper radius γ_{max} are introduced. So we discard the contours with minimum enclosing circle not satisfying the given radius range.

The circularity of a contour can be evaluated by the relationship between its area and its perimeter:

$$\text{circularity} = 4\pi \frac{\text{area}}{\text{perimeter}^2} \tag{1}$$

Since this formula will return 1 given a perfect circle, it can be used to determine whether the contour is more likely to be a ball or not. In general, there will be at most two balls with same color in the court, except the single yellow ball. To choose the best contours, we sort the contours based on their circularity, and then select the two contours with the largest circularity. For the goal yellow ball we select only one. A lower threshold for circularity was also added, to ensure that no balls will be detected if there are none in the image. After adding the constraints on both radius and circularity, only round objects of similar size and color should be able to produce false poisitve-detections.

2.1.3 Ball Classification

After the contour detection the position of the balls are just given by the detected contour and its center. But between each frame the program has no way of separating between balls of the same color. This is crucial information for creating the trajectories of the balls, so we have to define a way to classify two balls with the same color. Since the frame rate of the video is quite high with respect to the velocity of the ball, there is a limit to how much the balls will move from one frame to the next. In figure 4 a situation is where two balls are dected in subsequent frames, and need to be classified. Since the time difference

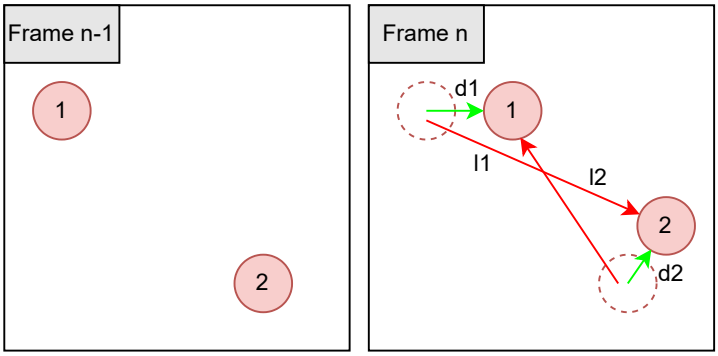

Figure 4: two situations of ball movements

between two frames is so small, it is more probable to have ball 1 and 2 move distance d_1 and d_2, than l_1 and l_2. By defining $D = d_1 + d_2$ and $L = l_1 + l_2$, we can create a rule for selecting the ball identities. We seek to minimize the total distances two balls move between two frames, so we select the identities corresponding to $min(D, L)$.

2.2 Results

For the first part of ball detection, the color masking, the color ranges had to be tuned quite thoroughly. In the end the results were quite successful, but it became evident that this method is quite vulnerable to changes in light. In some of our tests due to the reflection of light on the balls, their color in the image space was saturated to white, which meant some of the ball ended up outside the color range. In addition, the colors of the balls, with their extended ranges, cover quite a lot of the color space, so several objects in the environment was included.

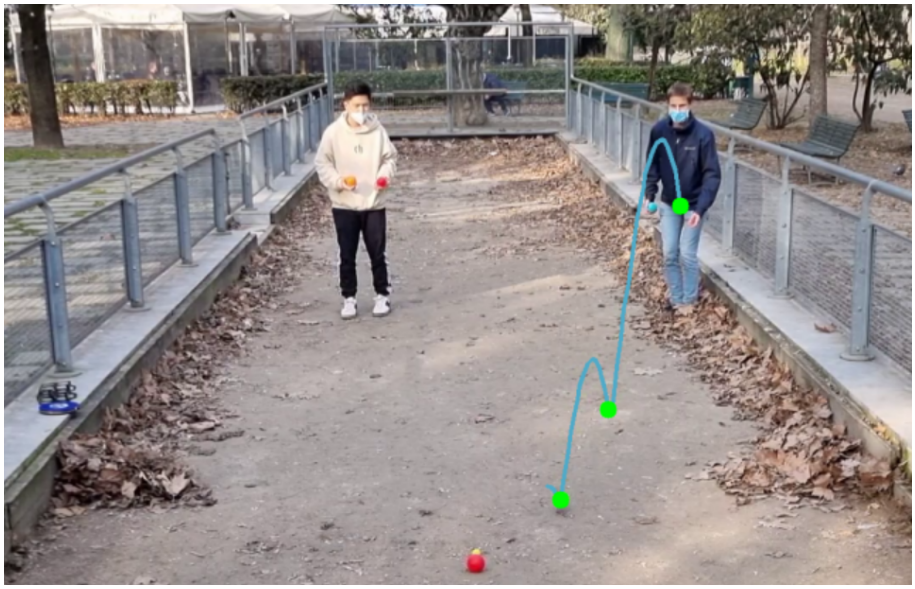

Figure 5: Results of ball tracking

In order to suppress the disturbance of random objects, we gradually added constraints on the contours. At the very start, we tried to sort the contours from large to small based on their area, however we found that some objects were also detected with similar color and area, like pattern on shoes, facial mask and moss. Therefore, circularity and radius constraints were introduced, in order to discard the unwanted objects. Lastly, functionality was added allowing the user to select an arbitrary polygon by clicking in the video. This was then used as a mask, restricting the detection to only areas within this polygon. This was effective for eliminating false positives outside the court, however it does put an extra element of supervision to be done by the user of the program.

Overall, we can conclude that after adding all the constraints, the results are quite good for the specific environment it is tuned for. However, there are several problems remaining to be solved and improved:

- As an alternative to contour detection, we could use other approaches to detect the balls. For instance, we tried to use the Hough circle method, which is a model based algorithm for detecting circles. However, this did not yeild satisfactory results, and the number of false positives were to many to filter away.

- Some objects was never completely suppressed. The blue face mask wore by one of the players shares almost the same color with blue balls, and also has a semi circular shape and similar size. To solve an issue such as this it might help to use edge detection rather than color masking. Of course controlling the environment, ensuring for instance that no one wears such masks, is the most straightforward way to solve such issues.

- The goal yellow ball is often not detected, or its detection is suppressed by noise. This is due to its tiny size and near white color.

- With respect to the ball classification part, when two balls of the same color overlap in the image, we may have the issue that their contours are merged into one, in this case our program will either discard the contour, or classify it as one ball. This problem, and outlines for a solution are discussed in section 5.

- Our way of classifying balls of the same color based on movement is very primitive, and subject to several problems. Most importantly when two same color balls is close to each other, their classification might be wrong. To improve this more complete physical constraints to the balls movements can be introduced.

  Figure 5 shows the results of detecting balls, and all the approaches mentioned above can be accomplished quite well, and the accuracy seems to be good enough to serve the purpose of this project.

3 Camera Calibration

In order to estimate where the balls are in the real world, we need to find the camera projection matrix. It is used for relating positions in the three-dimensional world reference frame and positions in the image space. It is a 3×4 homogeneous matrix on the form:

$$P = K \begin{bmatrix} R_{3\times3} & T_{3\times1} \end{bmatrix} \tag{2}$$

This matrix can be used to project any point X in the world space onto its point x' in image space, by the transformation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = x' = P_{3\times4}X = P_{3\times4} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3}$$

3.1 Approach

The projection matrix was found in two steps. First finding the camera matrix K, containing the intrinsics of the camera. After this the extrinsics R and T was found. Then the projection matrix was computed with the relation in equation 2. A diagram giving an overview of this approach is shown in figure 6.

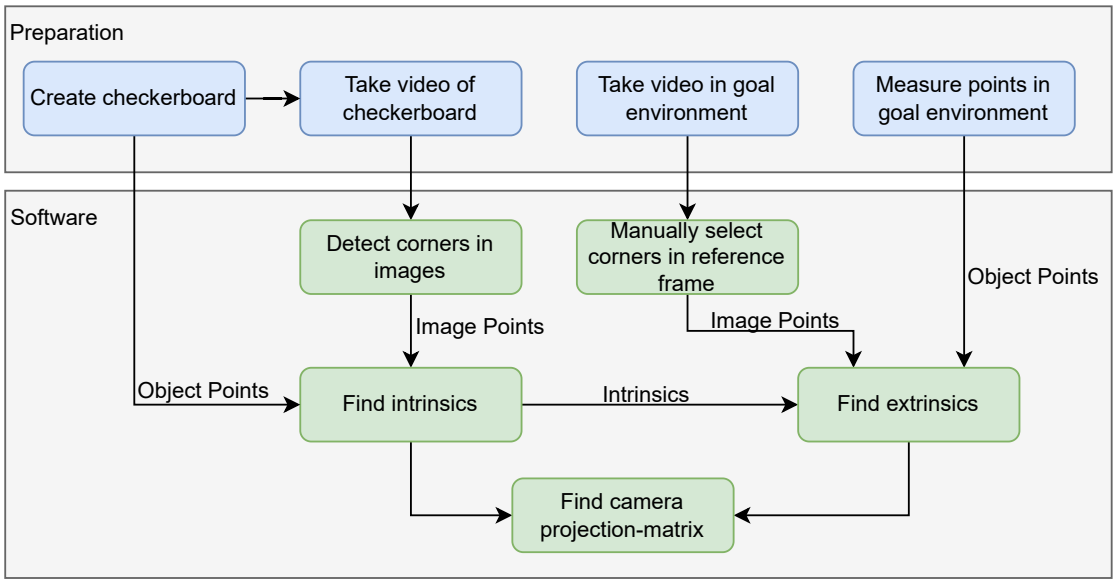

Figure 6: The camera calibration process

3.1.1 Finding the camera intrinsics

The camera intrinsics are

$$K = \begin{bmatrix} f_x & \gamma & U_0 \\ 0 & f_y & V_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where f_x and f_y are the focal distances and (U_0, V_0) is the principal point. The parameter γ represents a skew between x and y axes in the image, however this can often be assumed to be zero. These are values unique to all cameras, which means that we only need to

Figure 7: Checkerboard used for calibration, with tracked corners.

calculate them once for the camera used in this project. Since the camera intrinsics are defined only by the camera itself, they can be found independent of the specific situation in which the program will be used. Since a phone was used to capture all the videos in this project, the camera was readily available for calibration in controlled environments. For this reason the intrinsics were found using a calibration-checkerboard. An alternative would be to use known locations in the bocce-court the games were played on, but the nature of the court made such measurements difficult to obtain.

Obtaining camera intrinsics with a calibration-checkerboard is based on taking several images from different angles of a checkerboard of known dimensions. The checkerboard used in this project can be seen in figure 7. By defining an origin in one of the corners we can define the true three-dimensional positions of all the corners on the board, we call them object points. Since all the corners are on a common plane, they can all be defined with $z = 0$. Then all these corners can be found in the images using functions in opencv. The tracking of these points is also shown in figure 7

Performing the camera calibration based on these sets of image points and corresponding real points is done with Zhangs Method. Since there are five unknowns in K we need to find five constraints on them to solve the system. Since all the object points have $z = 0$, equation 3 can be reduced to a 2D transformation, described by a 3x3 homography:

$$x' = H_{3\times3}X = K \begin{bmatrix} R_{3\times2} & T_{3\times1} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5}$$

This homography can be estimated by direct linear transformation (DLT), with a overdetermined system. The homography matrix can be presented by three column vectors, and using the knowledge that $R_{3\times2}$ consists of two orthonormal 3x1 vectors:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{6}$$

The estimated homography can then be used to impose two constraints on K derived from

equation 5, since r_1 and r_2 are orthonormal, we can deduce that:

$$r_1^T r2 = 0 \quad \rightarrow \quad h_1^T K^{-T} K^{-1} h_2 = 0 \tag{7}$$

$$||r_1|| = ||r_2|| \quad \rightarrow \quad h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 \tag{8}$$

since each picture yields two constraints on K (5 unknowns), we need 3 or more pictures. In our case 30 images was used, so the system is overdetermined, but this is solved with a least squares approximation. The Zhang method is already implemented in opencv in the function `calibrateCamera`, so this was used. This function will also return the estimated distortion parameters of the camera.

3.1.2 Finding the camera extrinsics

Camera extrinsics describe the rotation and translation between the camera and a chosen world frame (the world frame used in this case is shown in figure 8. As a result these parameters will change every time the camera moves or rotates. Since the cameras were fixed during the recordings, we only need to calculate the extrinsics once for each video. Since the camera extrinsics are defined by a rotation R (three unknowns) and translation

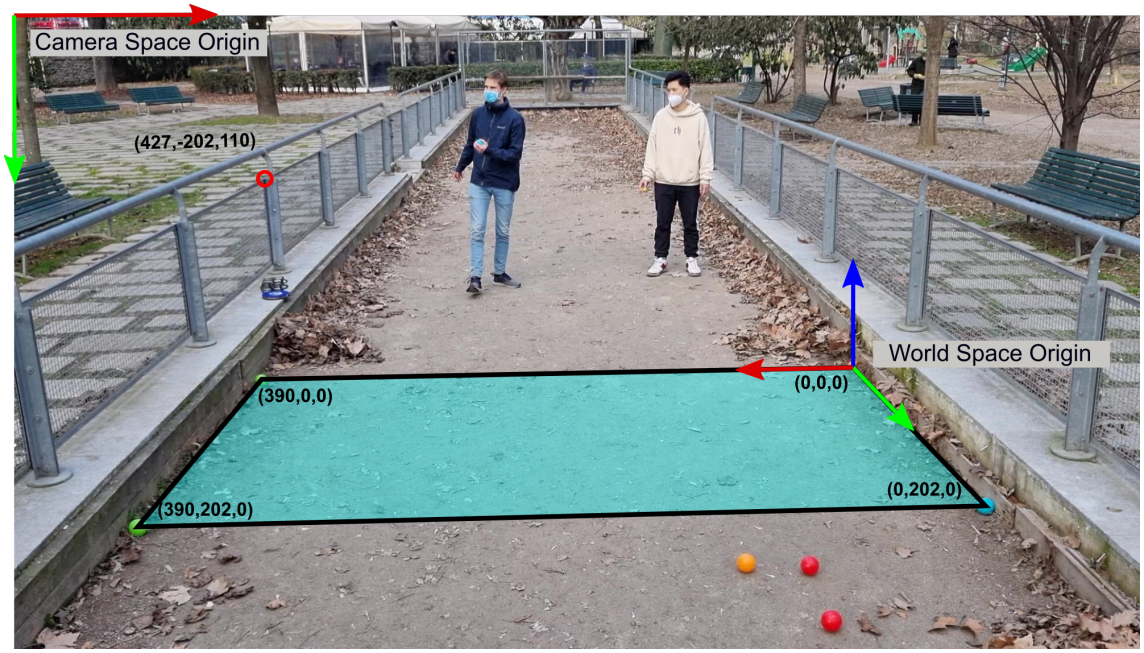

Figure 8: The chosen world frame

T (three unknowns), we have six unknowns to solve for. Using the intrinsics K and known points in the scene, we can use equation 3 to get:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{9}$$

Each known point in the scene contributes with two constraints. So we need at least three points. By measurement, the four corners of the goal region are known, and can thus be used to estimate the extrinsics. In opencv there is a function `solvePnP` for exactly this purpose.

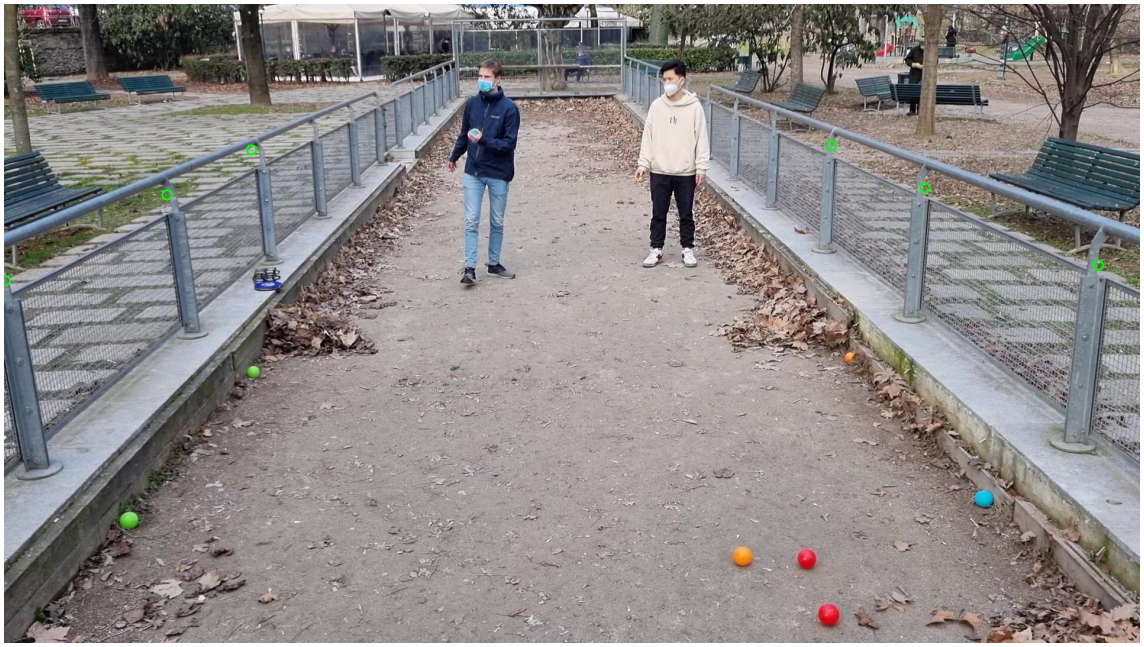

Figure 9: Projected points for evaluation of the calibration process

3.2 Results

After the projection matrix is obtained, it is of interest to verify whether it is correct (or at least a good enough estimation). One way to quickly verify if the calibration is successful is to project a known object point onto the image, and verify by visual inspection if the projection was correct. In our case we had also measured the three-dimensional position of a point on the fence, relative to the origin. This point can be seen in figure 8. This point was not used in the calibration process, and it is not in the same plane as the other known points, so it serves as a good test for verifying the calibration. This point was projected into the image using equation 3. Assuming that the court is symmetric, five other object points on the fence was calculated based on the measured point, projected and drawn. Figure 9 shows the resulting position of these drawn points (green circles). The results are quite good, but clearly not perfect. There can be several reasons for this result:

- Inaccuracy in the measurements of real-world distances.
- The manual selection of the balls is subject to some inaccuracy, as well as the balls not serving as perfect points for the corners of the court.
- The court is not perfectly planar.
- The assumption that the court fence is symmetric might not be completely correct. For instance the right side might be a bit higher than the left.

Most likely the inaccuracy in the drawn points (and more importantly the projection matrix) is a result of a combination of all these effects. However the accuracy seems to be enough to serve the purpose of this project.

4 Trajectory Estimation

In order to estimate the 3D trajectory of a ball from just one fixed camera, three elements are needed; the 2D trajectory of image points u, v found in section 2, the estimated camera parameters matrix $\boldsymbol{P}$ found in section 3 and a physical model of the ball's real world motion.

4.1 Approach

4.1.1 Physical model

For given initial position (x_0, y_0, z_0) and velocity components (V_x, V_y, V_z), the physical motion of a ball, only affected by gravitational forces, follows a parabolic trajectory. The position in the real world coordinates X, Y, Z is dependent on time t and can be modeled by the system of equations 10.

$$\begin{aligned} X(t) &= x_0 + V_x t \\ Y(t) &= y_0 + V_y t \\ Z(t) &= z_0 + V_z t + \frac{1}{2} g t^2 \end{aligned} \tag{10}$$

where $g = -981[cm/s^2]$ is the gravitational constant defined along the z-direction. By determining the initial positions x_0, y_0, z_0 and velocities V_x, V_y, V_z of the ball, these equations can be used to calculate the full 3D trajectory of the ball motion.

4.1.2 3D trajectory estimation

To calculate the 3D trajectory, the six unknown parameters, i.e. initial position and velocities, must be determined. A sufficient amount of points u, v in the image frame is needed. Each point can be transformed into real world coordinates (X, Y, Z) with the transformation relation 3 using the projection matrix $\boldsymbol{P}$.

$$\boldsymbol{P} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{33} & C_{24} \\ C_{31} & C_{32} & C_{33} & 1 \end{bmatrix} \tag{11}$$

Setting (X, Y, Z) equal to the physical model equations 10 we get the set of equations 12.

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{33} & C_{24} \\ C_{31} & C_{32} & C_{33} & 1 \end{bmatrix} \begin{bmatrix} x_0 + V_x t \\ y_0 + V_y t \\ z_0 + V_z t + \frac{1}{2} g t^2 \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{12}$$

Multiplying out and inserting the last equation into the two first, we get the following two equations:

$$\begin{aligned} C_{11}x_0 + C_{11}V_x t + C_{12}y_0 + C_{12}V_y t + C_{13}z_0 + C_{13}V_z t + \frac{1}{2}C_{13}gt^2 + C_{14} = \\ u(C_{31}x_0 + C_{31}V_x t + C_{32}y_0 + C_{32}V_y t + C_{33}z_0 + C_{33}V_z t + \frac{1}{2}C_{33}gt^2 + 1) \end{aligned} \tag{13}$$

$$C_{21}x_0 + C_{21}V_x t + C_{22}y_0 + C_{22}V_y t + C_{23}z_0 + C_{23}V_z t + \frac{1}{2}C_{23}gt^2 + C_{24} =$$
$$v(C_{31}x_0 + C_{31}V_x t + C_{32}y_0 + C_{32}V_y t + C_{33}z_0 + C_{33}V_z t + \frac{1}{2}C_{33}gt^2 + 1) \quad (14)$$

For n image points measured at a rate $1/t$, the two above equations can be solved at each time instant t to create a set of $2n$ equations.

Gathering the six unknown parameters, i.e. the initial position and the initial velocity components, in a vector $\boldsymbol{q} = [x_0, V_x, y_0, V_y, z_0, V_z]^T$ it is possible to build a system of the form 15.

$$\boldsymbol{Aq} = \boldsymbol{a} \quad (15)$$

where $\boldsymbol{A}$ is a 6-by-2n matrix and $\boldsymbol{a}$ is a vector of length 2n created from the left and right side of equations 13 and 14 respectively.

Solving 15 for $\boldsymbol{q}$ 16 using the pseudo inverse of $\boldsymbol{A}$ we get an estimate of the initial position and velocity components of the 3D trajectory.

$$\boldsymbol{q} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{a} \quad (16)$$

The estimated $\boldsymbol{q}$ can then be used to calculate the full 3D trajectory for the ball by using the physical model equations 10.

The process structure of calculating the 3D trajectory can be seen in figure 10.

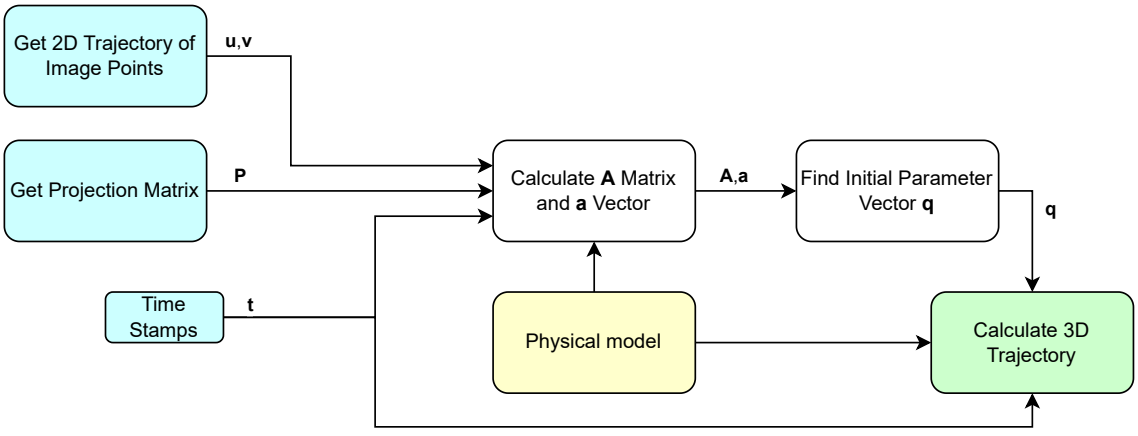

Figure 10: Structure of 3D trajectory estimation process. Inputs from other modules are marked in blue while the output from the 3D trajectory module is marked green.

4.2 Results

The real world ball trajectories were calculated based on the 2D trajectory of detected ball positions in the image frame, the camera specific projection matrix and the simplified physical model described. The resulting 3D trajectories, showing throws from three different angles, can be seen in figure 11.

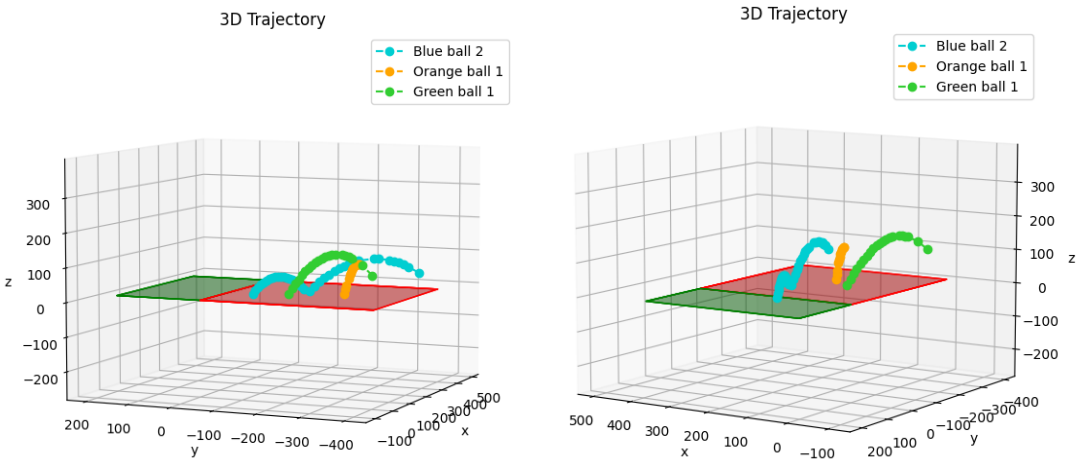

Figure 11: 3D trajectories from two different angles

Comparing the obtained 3D trajectories in figure 11 to their corresponding 2D image trajectories in figure 12 it can be verified that the 3D trajectories have both reasonable heading and shape.

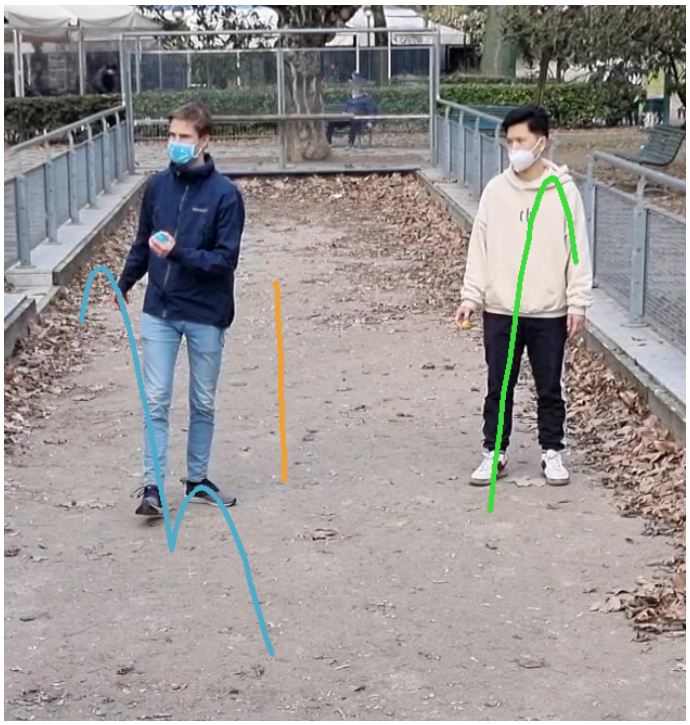

Figure 12: 2D trajectories from ball detection

5 Points of improvements

5.1 Ball detection

5.1.1 Solution for partially overlapping balls

As mentioned in section 2, no matter how the camera is placed, there is always the possibility of two balls of the same color partially obscuring one another with respect to the camera. With our current solution this leads to the two balls creating one continuous contour. This results in the balls either being classified as one, or they will both be discarded if their joined contour is too big. A complete solution to this problem was not developed, but parts of it was explored. The problem consists of two parts:

- Detect that a contour is the joined contour of two balls.
- Estimate the exact location of each ball

The first of these points has not been tackled. However a method has been developed to solve the second point. The algorithm is as follows (see 13 for a reference figure):

- Find the smallest circle enclosing the joined contour
- Find the two points on the circle with the largest distance from the circle
- Split the original contour at these two points
- Find the smallest circles enclosing these two contours

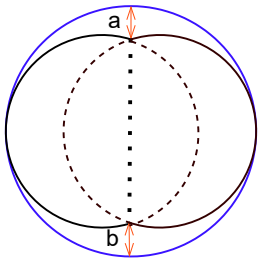

Figure 13: The problem of overlapping balls

The result of a program running such an algorithm is shown in figure 14.

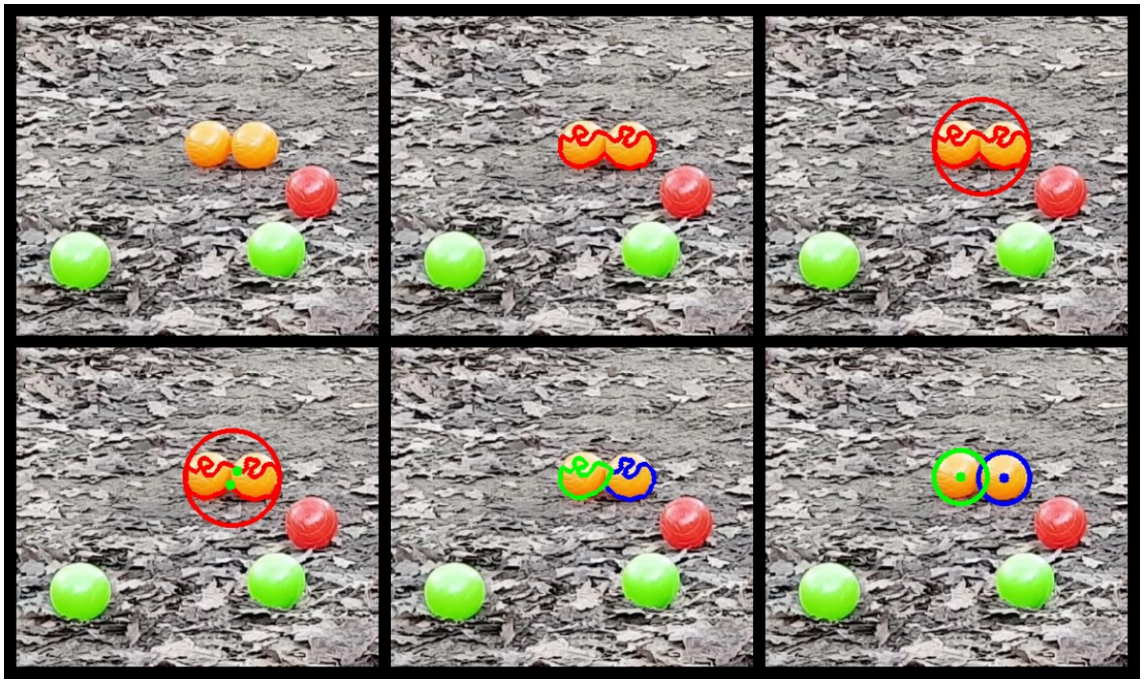

Figure 14: Partial solution for solving the problem of overlapping balls

5.2 Game Scores

After the camera calibration the positions of the balls in 2d could be estimated by assuming that they are on the court (z=0). Displaying the results, however showed to be quite time consuming. As the user interface is not directly correlated to the goals of the course, only a unrefined example was made.

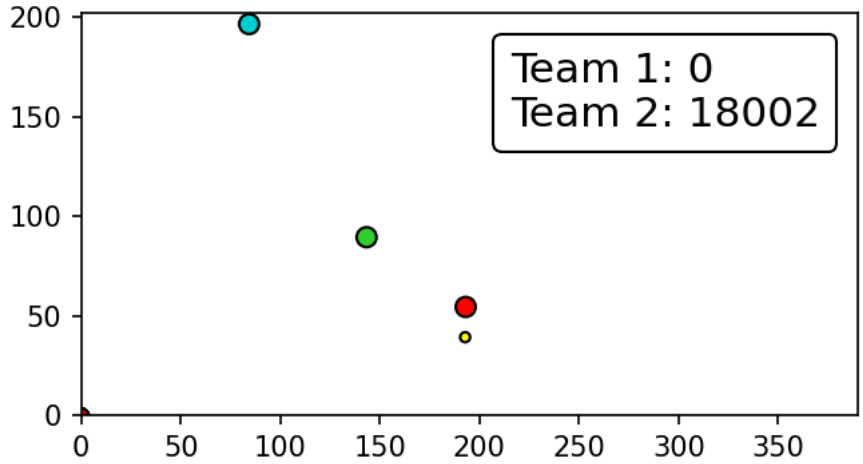

Figure 15: Score estimation

5.3 3D trajectory estimation

5.3.1 Measuring

Due to the sub-optimal environment and the lack of proper measuring gear it was not possible to find the true real world trajectories, and hence numerical comparisons could not be provided.

5.3.2 Camera location

It was seen that the results vary in accuracy based on the angle of the trajectories, with greater angle yielding better results. I.e. a different camera placement, showing the trajectories from the side, would yield more accurate results for the 3D trajectory estimation.

5.3.3 Physical model

The physical model used assumed only gravitational forces acting on the ball. The results can therefore be improved by using a more realistic model taking into account air friction.