Politecnico di Milano

3D Trajectory in Sports - Table Tennis

July 18, 2018

Version 1.0

Authors:

- Matheus FIM (Mat. 876069)
- Reda AISSAOUI (Mat. xxxxxx)

Prof. Vincenzo CAGLIOTI

Contents

1 Problem Formulation

Table tennis is a demanding sport where the referees have to pay attention at a complex set of rules at all moments. For this reason, much can be obtained from having in-place a system capable of reconstructing the 3D trajectory of the ball during the course of a point. For instance, eventually unclear plays can be decided correctly or even coaches can better analyse the players performance. However this is not a simple task. Many of the techniques applied to other sports cannot be used for this context, as the table tennis ball can be hard to detect due to its reduced dimension and also cannot be relied to follow a parabolic trajectory as its small mass causes it to be heavily affected by the air resistance.

The proposed system takes as input real matches video sequences and is able to overcome a wide variety of problems, such as high object motion, blurring, uneven lighting, background merging, object confusion and occlusion. Thus generating as output an estimation of the 3D trajectory of the ball. The algorithm relies on videos recorded by two non synchronized RGB cameras and uses this information to detect the table, calibrate the cameras, track the ball in between frames, synchronize the cameras and finally combine the single trajectories for the production of the 3D trajectory.

The Experiments were conducted on 60fps and 120fps video samples recorded with GOPRO cameras in an indoor environment and produced satisfactory results. In all tests the trajectory was estimated with suitable accuracy, whereas the most satisfying results were experienced in the higher fps videos. Also, the adaptive "region of interest" allowed to reduce both the time of execution of the algorithm and the incidence of false positives.

2 State of the Art

Wong et. al.[**?**] described an algorithm capable of tracking table tennis balls in services and stated the accuracy was high enough for umpiring applications. Their work handled the following problems:

- **High object motion:** The high speed of the ball may cause it to be blurred, colour faded and distorted in shaped.

- **Multiple moving objects:** Apart from the ball, the players, the rackets and background elements in the scene exhibit motion.
- **Uneven lighting:** Light sources on the ceiling tends to cause the ball to be brighter at the top than at the bottom.
- **Occlusion:** The ball can be blocked by the player, by the rackets, or even disappear from the field of view in the upper part of the video.
- **Merging:** Low contrast from the ball and the background may cause difficulties in distinguishing both.
- **Object Confusion:** Objects in the background with similar color (e.g. human skin) and shapes (e.g. a bald person head) may be confused with the ball.
- **Small size:** Size of the ball is only a fraction of the frame, which randers histogram-based detection unsuitable.
- **Time constraint:** latency from detecting to tracking the ball must be minimised.

In order to overcome these issues it is proposed a method that executes an object segmentation followed by a object evaluation algorithm. The object segmentation is composed by a two-pass thresholding (TPT) algorithm, capable of adapting the threshold according to context. The first pass highlights only elements of the frame with a highly similar color to the one of the ball, and the second thresholding relax the constraints in order to reconstruct the shape of the object. The output is a set of candidate balls to be further evaluated by the object evaluation algorithm. This second algorithm is also executed in two steps. In the first it is controlled if the individuated object displays a rounded upper contour, its location is consistent with the predicted location and if it exhibits motion both at its centre and in the predicted location. And in the second, from the set of objects that were able to pass the first step, one is chosen as the ball based on spatial geometric measurements (area, maximum width, maximum height, perimeter, roundness, and error on the rounded upper contour).

These elements are going to be further discussed in the document as they were applied in the project.

Tamaki and Saito(20..) proposed a method of reconstructing 3D trajectories of table tennis balls. They consider both the problem reconstructing the trajectory from two rgb cameras and from one rgb-d

camera (i.e. capable of providing depth information). One of the novel approaches is the method that approximates that a ball is travelling on tilted planes, and thus it is able to accurately identify which of the possible candidates is the ball. Also they leverage on the work produced by **Zhang (CITE)** to compute the camera intrinsics ... (continue)

3 Solution

The proposed solution is divided in table tennis detection, camera calibration, ball tracking, 2D trajectory estimation, camera synchronization and 3D trajectory estimation.

3.1 Set-up Phase

In the set-up phase the user selects the initial position of the ball and its diameter. The initial position allows the algorithm to understand which is the object to be tracked in case of multiple candidate objects in the scene and also allows the fine tuning of the color of the ball, done by averaging on the neighbouring pixels color. The diameter serves to define many of the constraints used in the algorithm, such as the area related constraints for detecting the ball, the size of the Region of Interest, the kernel for blurring the frame and thus smoothing the edges of objects.

3.2 Table detection

In order to take detect the table, we have to take advantage of its features. The table edges are long straight lines. It is also blue and stands out from its environment. So our approach consists of finding the lines present in the input image and then filter them according to different criteria. After that, the lines are grouped depending on which side of the table they belong to. From this group of lines, the four sides of the table is calculated

3.2.1 Lines detection

Using Hough peaks and Hough lines, the algorithm start by detecting the lines that are present in the input image. We consider as a valid lines

the ones that are on the edge of the table. The lines are therefore filtered on the following criteria:

Orientation

Given the point of view of the camera, we don't expect to have an edge of the table showing vertically on the image. Hough lines algorithm uses a polar coordinates system, therefore we filter the lines directly from θ of the polar coordinates.

Color separation

A valid line will have from one side the blue table color and then from the other side any other color (floor green, wall white...). To decide which lines are valid, two sample pixels are taken from each side of the line. The midpoint of the line along with the orthogonal line going through the midpoint are calculated. The sample pixels are equally distant points from the centroid on the orthogonal lines. The HSV values of these points are considered as it is a more robust way to deduce the color.

Given two points $p_1 = (h_1, s_1, v_1)$ and $p_2 = (h_2, s_2, v_2)$, a color is considered blue if

$$blue_1 = (h_1 > BLUEMIN) \wedge (h_1 < BLUEMAX) \wedge (h_1 > VLIGHTMIN)$$

where the thresholds (BLUEMIN, BLUEMAX and VLIGHTMIN) are defined from the hue values of the blue color. A minimum value of brightness is also imposed as blacks may be considered as blue.

A line is considered valid if

$$line = blue_1 \veebar blue_2$$

3.2.2 Lines grouping

Still needs to be developed

After detecting the lines on the edge of the table we need to cluster them using kmeans ....

Because of the distortion caused by the wide angle lens, the lines detection algorithm sometimes return many lines for one table edge.

we have multiple lines per edge $\Rightarrow$ weighted average grouping

then grouping ⇒ calculate intersects and then finished

3.3 Camera Calibration

3.4 Ball Tracking

The ball tracking is done using Wong et. al.[**?**] algorithm. The candidate balls are individuated in an object segmentation phase and then selected based on an object evaluation algorithm. The object segmentation phase is subdivided in the two-pass thresholding, automatic tuning of the thresholds, adaptive control of the Region of interest (ROI). While the object evaluation is composed of an eliminatory step, to discard all the candidates that do not have the minimum requisites to be considered the ball, and a classificatory one, to choose the best possible among the final set of candidates.

The pseudocode of the ball tracking algorithm is as follows:

Algorithm 1 Ball Tracking Algorithm

```
while Video has frame do
    frame ← Read video frame
    if First Frame then
        Positions ← User input initial ball position
        Thresholds ← Define initial color thresholds
        ROI ← Crop ROI based on frame and initial position
        Frame ← Read video frame
    else if Ball found before then
        ROI ← Crop ROI based on previous found position
    else                    ▷ In case the position was not found previously
        ROI ← Adaptive control of the ROI
    end if
    ROI ← Blurr ROI      ▷ To smooth the edges we apply a blurring filter in the ROI
    CandidateBalls ← First Pass(ROI)      ▷ First pass has a coarse threshold
    if CandidateBalls were found then
        for each CandidateBall do
            ROI ← Add the ROI of the CandidateBall
        end for
        CandidateBalls Second Pass(ROI)      ▷ Second pass has a relaxed threshold
        Objects ← Object Evaluation
        Positions ← Add Object with smallest error from Objects
    end if
end while
Trajectory ← Calculate 2D Trajectory
```

3.4.1 Two-Pass Thresholding

The objective is to segment *candidate balls* from the other elements in the frame. *Candidate balls* are objects that exhibit similar properties to the Object of Interest (OOI). To perform this segmentation it is implemented a color thresholding algorithm. The basic idea is to create a binary differential image where pixels with a color similar to the one of the ball are turned white while others are turned black, then the neighbouring white pixels are merged together to form an object. However, simple thresholding tends to be sensible to variations in color due to lighting or movement. Therefore, to increase precision this is divided in two passes. In the first, a coarse threshold is applied and only pixels with a sufficiently similar color to the ball are filtered and grouped according to the process

described above. The coarse threshold only individuates candidates that have a high probability of being the ball, but is also expected to produce a distorted object. In this matter, the second pass is applied only on the regions where *candidate balls* have been found and with a more relaxed threshold. This allows the recovery of the original shape of the object, while still excluding the majority of false positives.

Below it is possible to obseve the pseudocode for the **First Pass:**

Algorithm 2 First Pass Thresholding

while $CandidateBall$ not found AND $i < MaxIterations$ **do**
  $BinaryROI \leftarrow$ Color threshold the ROI ▷ returns a binary image with the white pixels representing the color of the ball
  if Only one object is found in the $BinaryROI$ **then**
    $CandidateBall \leftarrow$ set as found
    $BestFirstThreshold \leftarrow$ set current thresholds as best
  else if More then one object found in the $BinaryROI$ **then**
    if Less objects were found than in the previous iteration **then**
      $BestNumberOfObjects \leftarrow NumberOfObjects$
      $BestFirstThreshold \leftarrow$ set current thresholds as best
    end if
    $FirstThreshold \leftarrow$ Increase the Threshold
  else
    $FirstThreshold \leftarrow$ Decrease the Threshold
  end if
  $i \leftarrow$ increase the iteration number
end while
$FirstThreshold \leftarrow BestFirstThreshold$
return FirstThreshold, BinaryROI

And for the **Second Pass:**

Algorithm 3 Second Pass Thresholding

```
for each ROI do
    SecondThreshold ← restore initial thresholds ▷ Each ROI cannot be affected by the other, so they all start with the same thresholds
    while ObjectArea not OK AND i < MaxIterations do
        BinaryROI ← Color threshold the ROI
        for each of the objects in the BinaryROI do
            Error ← Calculate the percentage error between the areas of the object and the ball
            if Error is within threshold then
                ObjectArea ← OK
                BestSecondThreshold ← set current threshold
            else if Object area is greater than the ball area then
                if smallest percentage error so far then
                    BestPercentageError ← Error
                    BestSecondThreshold ← set current threshold
                end if
                SecondThreshold ← Increase the Threshold
            else
                if smallest percentage error so far then
                    BestPercentageError ← Error
                    BestSecondThreshold ← set current threshold
                end if
                SecondThreshold ← Decrease the Threshold
            end if
        end for
        i ← increase the iteration number
    end while
    BinaryROI ← recalculates the best ROI found
    Binaries ← Add the best BinaryROI of each object
    Thresholds ← Add the best BestSecondThreshold of each object
end for
return Threshold, Binaries
```

3.4.2 Automatic Tuning of the Two-Pass Thresholding

As it can be noticed, the method is highly dependent on the configuration of the TPT thresholds. They may vary depending on the specific environment conditions in which the video was recorded, and even in between the video frames. Hence, to increase the generalisability of the algorithm, it is implemented an automatic tuning of the TPT threshold. The automatic tuning leverages on the reference location for the ball selected in

the set-up phase by defining a region of interest (ROI) around the last known position of the ball. In the ROI it is applied the first pass thresholding. If zero *candidate balls* are found the threshold is decreased, while if it is found more than one, it is increased. The first pass is repeated either until the maximum number of iterations is reached or only one candidate is left. In the event that no candidate was found, the current frame is discarded and the adaptive control of the ROI is triggered. However, if there is at least one candidate the second pass is executed. The second pass is similar to the first, but instead of comparing the number of OOI found, it is controlled their area. As before, if the area is greater than the acceptable percentage difference from the known ball area, the threshold is increased. In the opposite event, it is decreased. When the maximum number of iterations is reached or the percentage difference is within the acceptable difference, the current threshold is selected as optimum and the ball selection algorithm is triggered.

3.4.3 Adaptive Control of the Region of Interest

The table tennis ball occupies only about 0.06% of the frame. Hence, searching the entire frame is not only computationally expensive but also is counter-productive as it is increases the possibility of false positives. To solve this issue only the part of the frame that is likely to contain the ball is examined. Wong et. al.[**?**] suggest the use of interpolation based on at least the two last known ball locations to determine where is the probable next ball position and set it as the centre of the ROI. We, however, choose to derogate from this approach. In our approach, the adaptive control of the ROI is based only on the of the previous search successes. I.e. if the ball was found in the previous frame, it is set as a small square of m times the diameter of the ball. For each frame the ball is not detected, the size of the ROI is increased by a constant factor k, until the limit of the frame size.

3.4.4 Object Evaluation

The object segmentation phase individuates a set of candidates that have a similar color to the ball and reconstructs their shape. Each of these *candidate balls* is analysed in a two stage evaluation. The first is eliminatory and checks if it has *rounded upper contour* (RUC), if the location is consistent with the predicted location (T) and if it exhibits motion at its centre (M). The parameters RUC, T, M are formally defined in the table 1.

Rounded Upper Contour (RUC)	$RUC = \begin{cases} 1, & \text{if } E_{RUC} < t_{RUC} \\ 0, & \text{if } E_{RUC} \geq t_{RUC} \end{cases}$ where t_{RUC} is a preset threshold and E_{RUC} is an objective (error) function defined as: $E_{RUC} = \frac{\sum \lvert \frac{d_i - r}{r} \rvert}{N}$ and $d_i = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}$ where $(x_i y_i)$ is a pixel in the upper contour, i the pixel index, N the number of pixels, r the radius and (x_c, y_c) the centre obtained by solving the equation of a circle for (x_i, y_i).
Position (T)	$T = \begin{cases} 1, & \text{if } L_{diff} < t_T \\ 0, & \text{if } L_{diff} \geq t_T \end{cases}$ Where L_{diff} is the Euclidean distance between the OOI actual and predicted positions and t_T is a threshold. The predicted location is the linear extrapolation of OOI locations in previous frames.
Motion (M)	$M = \begin{cases} 1, & \text{if } OC_{diff} < t_M \\ 0, & \text{if } OC_{diff} \geq t_M \end{cases}$ Where OC_{diff} is the Euclidean distance between the candidate ball and the OOI in the previous frame and t_M is a threshold.

Table 1: First Stage Evaluation

Those that were able to fulfill at least two of the above criteria are kept while the other *candidate balls* are discarded. In the second stage, it is evaluated which of the remaining objects has the best fit according to a set of spatial geometric measurements, which are area (A), maximum width (W), maximum height (H), perimeter (P), roundness (R), and error on the rounded upper contour E_{RUC}. The definition of the parameters can be found in the table 2 and are combined in an error function E, calculated as:

$$E = \frac{E_{RUC} + w_A \frac{|A_c - A_b|}{A_b} + w_W \frac{|W_c - W_b|}{W_b} + w_H \frac{|H_c - H_b|}{H_b} + w_P \frac{|P_c - P_b|}{P_b} + w_R \frac{|R_c - R_b|}{R_b}}{(n_p + 1) n_c} \quad (1)$$

where w are the weightings, n_p is the number of spatial parameters, n_c is the number of conditions a *candidate ball* satisfied in the stage 1 test. The X_c and X_b are the spatial parameters of the *candidate ball* and OOI respectively.

Area (A)	Number of pixels in the object.
Maximum width (W)	Horizontal distance between left and rightmost pixels of the object.
Maximum height (H)	Vertical distance between top and bottom most pixels of the object.
Perimeter (P)	Length of object boundary
Roundness (R)	Given by: $R = \frac{4\Pi A}{P^2}$

Table 2: Second Stage Evaluation

The *candidate ball* with the smallest error E is selected as OOI.

To better exemplify the above algorithm, below it is possible to observe the pseudocode:

Algorithm 4 Object Evaluation

```
for each Object do                                   ▷ Start with the first stage
    RUC ← True if it has Rounded Upper Contour       ▷ according to described formula in table 2
    if More than one position of the ball in previous frames is available then
        Predict ← Linear extrapolate the next ball location
        T ← True if Object is within a maximum distance from prediction
        M ← True if candidate show a minimum movement from last known position
    else                                             ▷ no prediction is made
        M ← True if candidate show a minimum movement from last known
    end if
    if more than one condition is true then          ▷ Second stage
        Calculate spatial geometric constants
        Error ← calculate the error as described in formula 1
        if Error is smaller current MinimumError then
            OOI ← Object
            MinimumError ← Error
        end if
    end if
end for
```

3.5 2D Trajectory Estimation

The above process is repeated for every frame of the video, resulting in a sequence of ball positions in time. From this is possible to reconstruct the trajectory of the ball according to the single cameras. As a possible excess of caution, first an outlier removal process is done using the Hampel identifier. The Hampel filter calculates the median of the window composed by the six surrounding positions, three per side. Then it estimates the standard deviation of each sample with its window median, and replaces it with the median if it is more than three standard deviations. From the derived set of positions, it is applied a cubic-Spline interpolation to reconstruct the curve. This method allows to estimate successfully even the points in which the ball was out of the frame.

3.6 Camera Synchronization

To synchronize the camera we use as reference time instants of the bounces of the ball. From the estimation of these instants it is possible to calculate the *delta* between both cameras and correct it. The result is a set of equivalent points in time. To estimate the bounce instants, it is observed the changes in the vertical axis for a window of six positions. If it is found that there are three consecutive downward movements followed by three upward, it is checked that there is no change in the vertical axis in the same period (preventing from considering players striking from down to up). From this is recognized a pair frames in which a bounce has occured. The exact moment of the bounce is estimated with the cubice-Spline interpolation. With the bounces time estimation, it is calculated which are the possible matches in the cameras. This is done by the comparison of the time difference for bounces in the first camera with the ones in the second. In case of doubt it is also considered the interval between two bounces at the same camera and compared with the multiple possibilities in the second camera.

3.7 3D trajectory Estimation

4 Experimental Results

5 Conclusions