Real-time Shuttlecock Trajectory Estimation for Badminton Playing Robot

Shubham Paul, Chirag Parikh, Naman Pandey, and Anirvan Dutta
Birla Institute of Technology, Mesra, India

***Abstract*— This paper deals with the computer vision algorithms developed for automating a badminton playing robot. The shuttlecock is segmented out from any other moving object through its unique parabolic curve fitting and its drop is finally predicted by trajectory estimation of its motion using Kalman filter while considering air resistance and random noise during its time of flight. The badminton playing robot is positioned at the estimated dropping point before the shuttlecock actually reaches at that point so as to make a hit. The position of the robot was also detected through special colored markers and controlled with a PID controller in real time.**

I. Introduction

In robotics, many research papers have been published for automating different sports games like cricket, table tennis, baseball, etc., [1], [2]. Though very few works have reported to automate a badminton game via automated badminton playing robot, some work has been done on the shuttlecock trajectory prediction but they have not been reported of being practically implemented in a badminton game. Shuttlecock detection is challenging using the existing image processing algorithms because of its irregular shape and very small size (approx. 6cm in height), so it is not easily visible in low resolution image processing operations. Also, because of its irregular shape and low mass, its motion is largely affected by air resistance. The shuttlecock has a very high initial velocity, hence its exact position cannot be determined in real-time unless the algorithm for its segmentation is efficient and fast. Hideshiko et al., [3] proposed a method for trajectory estimation or prediction of a shuttlecock using motion blur. They used this feature to determine the real time velocity and position of a moving shuttlecock during the game and fed these values as inputs to the three-dimensional (3D) Kalman filter model for its periodic update, thereby predicting its trajectory. The velocity estimation using motion blur was noise-prone at higher velocities due to inaccuracies in the segmentation of the blurred region. Yongkui et al. [4], have suggested a robust algorithm to accurately and quickly predict the actual trajectory of shuttlecock using Kalman filter considering air resistance and Earths gravity. They have assumed a constant velocity of shuttlecock in its initial time of flight, but there is a considerable decrease in its high initial velocity due to the large deceleration caused by the air drag. The method proposed in this paper segmented out the shuttlecock from all other objects in motion based on its parabola-like trajectory, thereby resulting an improved and reliable detection of shuttlecock. The Kalman Filter model (accounting the air drag) was then initialized and the next position of the shuttlecock was predicted. In the next frame, the shuttlecocks position was determined by searching for moving objects inside a Region Of Interest (ROI) around the predicted shuttlecock position. The KF (Kalman Filter) simultaneously was periodically updated and accordingly the shuttlecocks trajectory estimation was refined, ultimately predicting an accurate position of its drop on the opponent teams badminton court much before its actual fall (giving sufficient time for positioning and hitting of the shuttlecock by the badminton playing robot). The position of the badminton playing robot was also determined by identifying a color-pattern marker placed near the hitting actuator. This algorithm gave a maximum of $1/52$ second of execution time which was enough for real time shuttlecock tracking. Thus the position of shuttlecock could be accurately tracked even when its initial velocity was high. The algorithm was executed independently for two different camera views perpendicular to each other in two different computers. The two-dimensional (2D) images captured from these two separate camera views were independently processed and accordingly the position feedback was communicated to the robot for its localization in the two perpendicular axes. This method is very flexible to use since it does not require any 3D calibration to get the 3D world coordinates of the shuttlecock.

II. Methodology

A. Camera Positioning

The 3D position of the shuttlecock was estimated at any given instant based on the information obtained from two different view points or camera planes as discussed by [3], shown in the Fig. 1. This methodology is termed as 3D projection/mapping. Three dimensional mapping to two dimensional planes is done without any loss of information. Thus it is possible to unambiguously project points from the physical 3D world to points in the image. The trajectory of the shuttlecock was estimated in both the view points independently and the approximated trajectory data points were sent to the controllers. Thus the trajectory of the shuttlecock could be tracked quite accurately. Also the computation efficiency was increased as all the algorithm had to be developed for 2D plane instead of 3D space.

[1]The authors are undergraduate students of BIT Mesra, Ranchi. shubhampaul14@gmail.com

Preprint submitted to IEEE International Conference on Robotics and Automation for Humanitarian Applications. Received June 24, 2016.

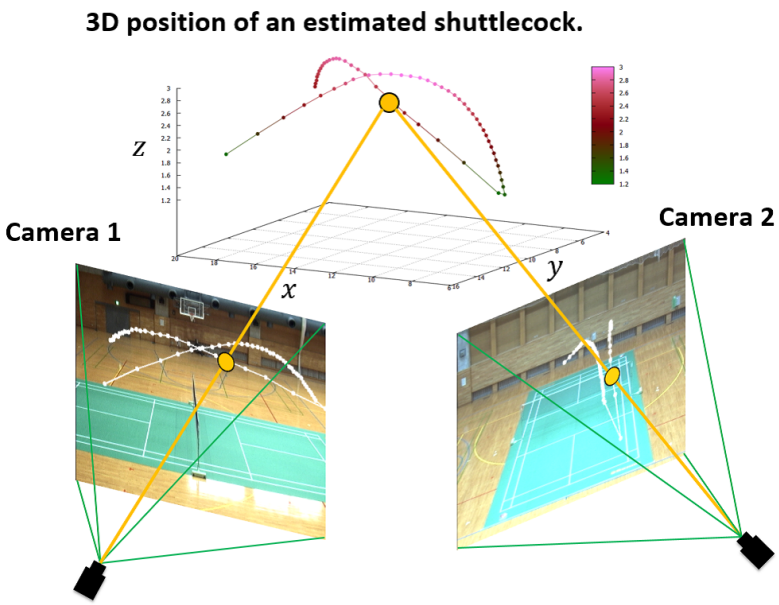

Fig. 1. Positioning of the PS3 Eye Cameras, [3]

The following subsections are in the order in which they appear in our overall shuttle-tracking algorithm.

B. Motion Detection

Frame differencing was used to detect the motion of the shuttlecock. This process can efficiently catch the edges of moving objects. If $F1$ is a gray-scale at time T and similarly $F2$ at time $T+1$, then the frame difference method gives $F = F2 - F1$. The frame F was converted into binary image based on the difference value of the pixels. Since pixels exhibit fluctuations, small differences are ignored and large differences are set. The output of frame difference method is shown in Fig. 2, where only the moving objects including the shuttle is visible. This frame is labeled using connected-

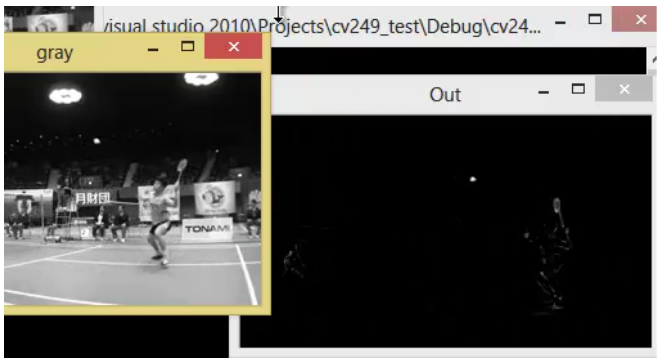

Fig. 2. Frame-differencing before binarization.

component approach. A connected component in a binary image is a set of pixel that form a group or blob. Thus neighboring pixels were identified by their label. A pixel may belong to a particular group or blob if it had the same label as that of all the pixels in that group. Neighboring pixels were labeled on the basis of 4-connectivity and a flood fill approach. Further, area of each blob was computed based on the number of pixels and area-based thresholding was implemented to improve the frame difference output. Since the noise due to fluctuations in pixel values may be present, a minimum area of the blob in frame differencing method was kept. This removes the erroneous noisy blobs. Further, to ignore the player movements, a maximum area threshold was also kept referring to the player blob-size in the frame. Earlier, Lucas-Kanade optical flow tracker method, [5] was also attempted, but due to lack of well-defined and sharp features of shuttle due to motion blurring, during implementation it was not used.

C. Parabolic Segmentation

This method of segmentation separated out the objects which are following a parabolic or a nearly parabolic trajectory. The blob analysis of the frame returned the centroid and area of the blobs from the frame subtraction method . Based on the area of the blobs (largest area first), the first N blobs were selected where N (30-50) is a variable and the x and y co-ordinates of the blobs are saved in an n-by-m array where m = 6. The reason to take m = 6 has been explained later. This step was repeated for 6 consecutive frames and the entire array was filled column by column, starting with the leftmost column as shown in the Fig. 3. With their centroids stored in this way, consistently moving objects were required to be tracked in all the consecutive 6 frames, for e.g. players, tip of the racket, the shuttlecock itself etc. This was done by calculating the minimum distance of a blob in the oldest frame, represented here by the leftmost column, to every blob in the next frame. The point which was found to have

Preprint submitted to IEEE International Conference on Robotics and Automation for Humanitarian Applications. Received June 24, 2016.

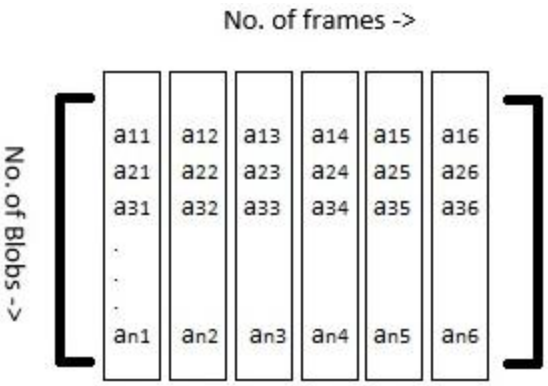

Fig. 3. Matrix illustrating the way centroids are stored.

the minimum distance, lying within a particular range, to the point in the previous frame was 'marked' and the operation was repeated for the next frame with respect to the currently marked point. If the minimum distance did not fall within this range, the current operation was canceled and started afresh with the next point in the first column.

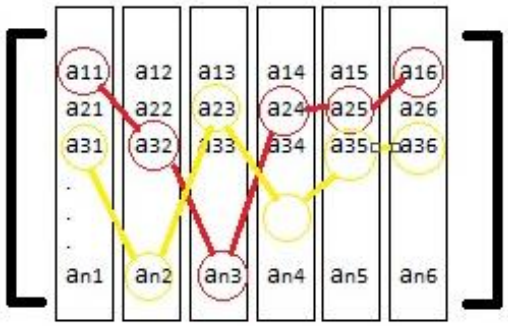

Fig. 4. Object tracking based on minimum-distance matching (join-the-dots) operation.

If this *joining-the-dots* operation was not canceled till the last column, then the 6 marked points were stored. These points possibly represent the motion of a moving object, which could be of a shuttlecock or any other object. Next it was to be classified whether this moving object was a shuttlecock or not. If it was a shuttlecock, its motion would resemble a parabola to a good degree. To find if the trajectory found was of a shuttle, the points of trajectory were fitted to the general equation of a parabola: $y = ax^2 + bx + c$. Also, as the trajectory of a shuttle is a downward opening parabola, value of $a < 0$ (in Cartesian system, it is $a > 0$, but in digital images, the y-axis is inverted). To find the parabola, a set of three consecutive points in trajectory was taken and the values of the three coefficient were calculated. In this way, using all the six marked points, the coefficients of four similar parabolic equations were evaluated. All these corresponding coefficient values would be nearly same if all the six points belong to the same parabolic path. The standard deviation of a, b and c were calculated to quantify their randomness. Now, if the marked points satisfied the above criteria, then it was assumed to be a shuttlecock and the parabolic motion based segmentation algorithm would be terminated and Kalman filter based tracking would begin. Else otherwise, in case of noise or any other moving object, its random motion would give non similar values of the coefficients (i.e. high standard deviations of a, b and c)for the four parabola equations. This method also returned an array of the 6 marked points, representing the history of the shuttlecocks motion, to the tracking algorithm for initialization of the Kalman filter function in OpenCV. Likewise, if this step failed then the entire operation was repeated for the next point in the first column. And if all the points in the first column failed this test then a new column of points after grabbing a new frame was pushed in, in a FIFO fashion , where each column represented the elements of a FIFO and the insertion occurred from the leftmost column. Thus, there was a delay of six frames only in the beginning to fill up the matrix.

D. Kalman filter based trajectory estimation and future path prediction

The Kalman filter in the algorithm served the following purposes

1) Use the corrected (filtered) Kalman states to find the future course of shuttlecock flight trajectory
2) To use the filtered states to calculate the position and size of the adaptive ROI (to drastically reduce the computation).
3) To reduce jitter in the raw detection.
4) To protect against missed detection and occlusion for a few frames(by skipping the prediction step).

After the Parabolic segmentation step, the Kalman filter was initialized with approximate states (namely x, v_x, a_x, y, v_y, a_y and g) using the history returned by the parabolic segmentation step. The Kalman filter soon converged after a first few frames, normally 3 to 4, and hence started getting good values to begin the prediction.

For the Kalman Filter, the discrete State Transition equations as described by [4] were used. The paper also considered drag proportional to the velocity. To this, a 7^{th} state, i.e. the apparent acceleration due to gravity g was added by the authors. The apparent g changes and decreases as one moves farther away from the camera. This variation in g is more prominent in the camera which is looking at the court length-wise. For simplicity, the authors have taken the value of $T = 1$ as described in the paper by [4]. The new State Transition matrix used in our code thus became

$$X_{t+1} = \begin{bmatrix} 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -k & -k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -k & -k & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} X_t \quad (1)$$

Preprint submitted to IEEE International Conference on Robotics and Automation for Humanitarian Applications. Received June 24, 2016.

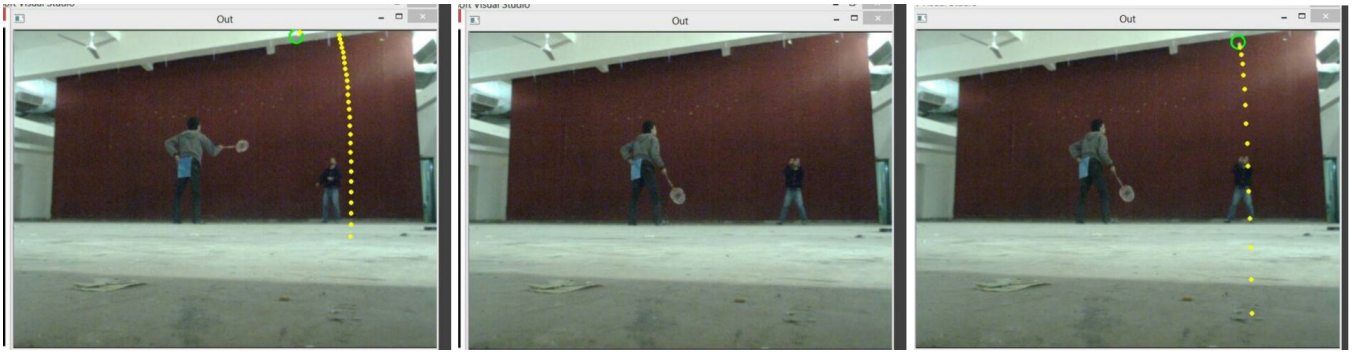

Fig. 5. Images showing a case of occlusion and subsequent tracking recovery by using the predictions made by Kalman filter.

where $X_t = \begin{bmatrix} x & v_x & a_x & y & v_y & a_y & g \end{bmatrix}^T$ and X_{t+1} are the predicted state and k is the constant b/m, where b and m are viscosity coefficient and mass, respectively. In our case, after some tuning, a value of 0.0109 worked satisfactorily. Therefore, the value of apparent g also adapted and corrected itself with each surpassing frame, thereby making our algorithm accurately track the shuttlecock in the longitudinal view also.

As mentioned earlier, the convergence of KF was accelerated by initializing it with values calculated from the centroid-history of the shuttlecock returned after the parabolic motion-segmentation step as follows:

$$\begin{aligned} x' &= x(t) \\ v'_x &= x(t) - x(t-1) \\ a'_x &= -k(x(t) - x(t-1)) \\ y' &= y(t) \\ v'_y &= y(t) - y(t-1) \\ a'_y &= g - k(y(t) - y(t-1)) \end{aligned} \tag{2}$$

where, $g = y(t) - 2y(t-1) + y(t-2)$ and, $g' = g$.

The dashed quantities were the elements of the initial State vector. These difference equations were derived by approximation of derivatives. The values thus obtained were noise-affected and hence were approximate. As the KF iterated and converged, good, noise-immune values of the shuttlecock states were eventually obtained. For predicting the future flight path, the following equations of kinematics in both x and y axis were iterated by setting $t = 1, 2, 3, \cdots$ and so on.

$$\begin{aligned} y_{predicted} &= y + v_y t + \frac{1}{2} a_y t^2 \\ x_{predicted} &= x + v_x t + \frac{1}{2} a_x t^2 \end{aligned} \tag{3}$$

Where, y, y_t, a_y, x, v_x and a_x are Kalman-corrected states. Thus, higher value of t represents the motion of the shuttlecock farther in the future.

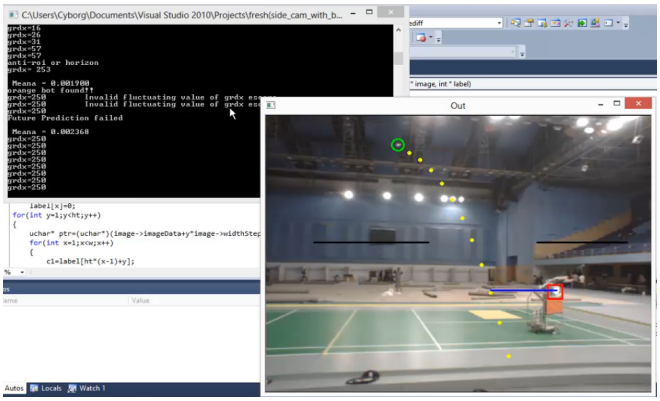

Fig. 6. Predicted flight of the shuttlecork and tracking of the robot.

E. ROI prediction and its adaptive sizing

The corrected Kalman states (from the previous measurement step) x and y were used to calculate the centroid of the predicted ROI. Also, the frame differencing occurred only in that relatively small ROI to find the shuttlecock which considerably reduced the computation. That centroid was initially the x and y itself and if the cork was not found in that ROI, the new ROI was calculated (by prediction) in the next frame (by setting $t = 1$ in (3)). If the shuttlecock was not found even in this frame then the new ROI was calculated two frames in the future (by setting $t = 2$ in (3) and so on). If the cork was not found within a set limit of future iterations, the shuttlecock was assumed to be lost and the parabolic segmentation step started over again.

The size of the ROI was also variable about a base size depending upon the cork x and y velocity. The size of the ROI changed breadth-wise and lengthwise by increasing or decreasing about the base dimension proportional to v_x and v_y respectively.

F. Localization of Badminton playing Robot and Estimation of the drop point

In addition to the shuttlecock, the program also tracked the robot in real-time (the red box shown in the Fig. 6). Figure 6 also shows the yellow trace for the predicted flight path of the

Preprint submitted to IEEE International Conference on Robotics and Automation for Humanitarian Applications. Received June 24, 2016.

shuttlecock (within the green circle). The blue line between the robot and the yellow trace represents the error value to be sent to their respective Proportional-Integral-Derivative (PID) controllers. The horizontal separation between the robot and the nearest horizontal point in the future path was measured and this separation was used as the error to be fed to the PID controller. The reason the authors came up with this scheme was because it allowed imperfections in setting up the camera tripods (due to limited setting time available in ABU Robocon 2015). This technique does not require elaborate camera placements and time-taking calibrations. All that's required was that both the shuttle and the robot's marker should be within the camera's field-of-view (FOV).
The robot was identified in the frame using a colored pattern segmentation technique. An orange sheet was pasted at the back of the robot and on top of it a white colored light-emitting-diode (LED) was installed. To find the robot in the frame, the pattern of orange color with a white color on top of it was searched.
Orange color was detected in the frame using a matrix subtraction method. The blue matrix of the frame was subtracted from the red matrix of the frame. Orange color has a high red color intensity and a low blue color intensity. When blue matrix was subtracted from red matrix, the gray-scale matrix returned only those blobs which were red in color or were a shade of red (for example, orange). White color was detected in the frame by checking for high intensity of RGB values of the frame. A pixel to pixel traversal was done for the whole frame and the pixels which had high values (above a threshold) were considered to be white colored pixel. Each laptop running the code controlled the motion of one axis and they worked independently. A PID controller was used to minimize the error (error calculation method described in the beginning of this subsection) in both the axes. Also a small 'anti-ROI' was defined around the colored marker (as close to the hitting actuator) to stop the tracking and restart the entire process to look for an incoming shuttlecork. More on the experimental method has been described in the following section.

G. Hardware and Experimental Setup

The algorithm was implemented in C using OpenCV, an open-source computer vision library using Microsoft's Visual Studio 2010. The computing platform consisted of two laptops running on Intel Core i3-3110M CPU @ 2.40GHz and 4GB RAM. The camera used was PS3 Eye Camera having a resolution 640x480 @ 60 fps and 320x240 @ 120 Frames Per Second (FPS). Their high frame-rate made it an excellent choice for capturing the motion of the shuttlecock without losing any frame. The cameras were connected to the laptop via active USB extension cables. The command to the robots after processing were sent via USART to an Arduino connected to each of the Laptops. The Arduinos, in turn were communicating with each other using SPI (because their Serial port were in use with the computers). The Arduino connected to the laptop doing the longitudinal processing sent the motion commands to the robot via a HC-05 Bluetooth module.

H. Performance and Results

The algorithm was tested for 50 test cases where the shuttlecock was segmented and trajectory was estimated with a decent accuracy for about 45 cases. Thus the essential component of tracking of the shuttlecock which was segmentation worked quite efficiently with the presented algorithm. It took approximately six frames, which was within $0.19s$, to precisely segment the shuttlecock from all other moving objects with the parabolic segmentation technique thus proving that the presented algorithm would work in real time scenarios as well. During Kalman tracking, the code executed at 52 FPS due to processing limited to a small ROI. Also since the total time of flight was mostly about $1-2$ s, there was enough time remaining for accurate fall point detection and hence it could give a robot enough time to position itself accurately at the fall point before the shuttlecock would actually reach that point if its locomotion is fast enough. The fall point accuracy was determined by the extent to which robot could accurately position itself to the falling point of the shuttlecock. This was constrained by two factors: 1) communication between the controllers and 2) the speed of the robot. Since the system analyzed the shuttlecock trajectory in two frames, the wheeled mobile robot with omni-wheels simultaneously received two values to traverse - one in x direction and other in y direction. Based on the above two factors, the certainty of the robot reaching the fall point before the shuttlecock directly depended on the distance to be traveled by the robot to reach the fall point from its current position in the game field. The ROI based approach of finding the shuttlecock in successive frames ensured not only less computational load on the system but also reduced the chances of noisy blob detection due to other moving objects around. Another feature of the algorithm was that it could very well handle cases of occlusion or when shuttle is lost due to prediction state of Kalman filter. A video [6] of our project in action and its source code [7] can be found in the web citations section.

III. Conclusion

In this paper, a novel shuttlecock trajectory estimation method using parabolic motion segmentation and Kalman filter was proposed and implemented in software. The badminton shuttlecork trajectory estimation problem was studied and a fairly robust and efficient algorithm to track the same was presented. All the aspects of the shuttle-tracking and prediction starting from the methodology to implementation were covered. This method was very light on the CPU since it scanned a very small area of the frame due to which a frame rate of 52 FPS during tracking could be achieved. Also since the robot is being tracked too, the need for elaborate and precise camera placement was not needed (like in the case of stereo vision), hence the vision setup consisting of just 2 separate cameras can be up and running in no

Preprint submitted to IEEE International Conference on Robotics and Automation for Humanitarian Applications. Received June 24, 2016.

time. One observation can be made from this paper that by changing the type of motion to segment or by changing the effect of air-resistance, the algorithm can be made to work for any game that involves a projectile motion or even rapid motions in a straight line (e.g. carrom, snooker etc). Thus all the essentials required to build a software capable of tracking and predicting a shuttlecock's motion were presented which can be used to develop a badminton-playing robot.

ACKNOWLEDGMENT

The authors would like to thank Mr. Arun Dayal Udai, Assistant Professor in the Department of Mechanical Engineering, BIT Mesra for his valuable inputs to our project and his untiring efforts and enthusiasm to promote the culture of robotics and innovation in our college.

REFERENCES

[1] Collins, Harry and Evans, Robert, 'You cannot be serious! Public understanding of technology with special reference to "Hawk-Eye" ',Public Understanding of Science, Vol: 17(3), 2008, pp. 283-308.

[2] Wei Chen and Yu-Jin Zhang (2006) 'Tracking Ball and Players with Applications to Highlight Ranking of Broadcasting Table Tennis Video', Computational Engineering in Systems Applications, IMACS Multiconference on (Volume:2), pp. 1896 - 1903 DOI: 10.1109/CESA.2006.4281948

[3] Hidehiko Shishhido, Itaru Kitahara, Yoshinari Kameda, and Yuichi Ohta, 'A Trajectory Estimation Method for Badminton Shuttlecock Utilizing Motion Blur', Image and Video Technology, Lecture Notes in Computer Science, Vol.(8333), 2014, pp 325-336.

[4] Man Yongkui, Zhao Liang, Hu Jingxin, 'Application of Kalman Filter in Track Prediction of Shuttlecock', Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics, December 19 -23, 2009, Guilin, China

[5] Lurong Shen, Xinsheng Huang, Wanying Xu, Yongbin Zheng, 'Robust Visual Tracking by Integrating Lucas-Kanade into Mean-shift', Proceedings of the IEEE 2011 Sixth International Conference on Image and Graphics, 12-15 August 2011, pp. 660 - 666, E-ISBN : 978-0-7695-4541-7,Hefei,Anhui.

WEB CITATIONS

[6] https://www.youtube.com/watch?v=7vyOqhcb11M

[7] https://github.com/Pratyunmis/Badminton-Shuttle-Tracker

Preprint submitted to IEEE International Conference on Robotics and Automation for Humanitarian Applications. Received June 24, 2016.