HAWKEYE

COMPUTER VISION CS-GY-6643

PROJECT REPORT

BY:

MONIL SHAH (mds747)

TABISH SADA (mts480)

SAMVID JHAVERI (snj268)

Abstract:

The objective of the project is to track the movement of ball and predict their trajectory. This technique is known as HawkEye. This technique is implemented in this project using computer vision techniques like Optical Flow using Lucas kanade algorithm and Kalman Filter Path prediction. In addition, we also calculate and display the Optical Flow vector field results in this project. For tracking the object or ball we used HSV color detection for the green color ball detection. Furthermore, we used input videos to perform the experiment. Finally, the result videos are also provided along with this report.

Introduction:

In this modern era, the overlap of technology and sports is at its peak. The advent of technology makes the sports and entertainment experience better day-to-day. One of the concerns in modern sports like Football, Cricket, Tennis, etc is the possibility of human error. One of the most common human errors is that tracking of the ball and measuring its trajectory to provide proper referral decisions. In our project we plan to mitigate this errors using computer vision techniques.

Computer Vision is an interdisciplinary branch, which aims to design system that automates the tasks which could have some error implications when performed by human due to large volume of data. One of the techniques in the field of computer vision is optical flow which deals with object tracking. The implementations of object tracking are used in the fields of surveillance, Sports, medical Imaging ,etc.

In this project, we deal with the implementation of object tracking in the field of sports. Hawkeye is one of the most used techniques in modern day sports like Football, Cricket, Tennis, etc. as a referral system. In this experiment we deal with tracking the object using computer vision technique like optical flow and strong algorithm like Kalman Filtering.

In this approach, we discuss about the two approaches to track the ball and provide hawk eye results. Mainly we use i) Optical Flow vector field Calculation ii) Trajectory prediction using Kalman Filtering. In the first approach , we use the green ball detection technique using the HSV color range for the green ball. Then, calculating the optical flow using the optical flow Farneback technique. The input provided is a video input, so the video analysis provides the frame rate analysis.

The second approach consists of the algorithm devised by Rudolf Kalman which takes a series of measurement observed overtime, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by using Bayesian inference and estimating a joint probability distribution over the variables for each timeframe.

We display this results by showing the optical flow vector field for the entire video. For the second approach we plot the ball tracking trajectory and after that the prediction from the Kalman filter.

Theoretical Calculations:

The theoretical calculations are also divided into two parts: First part is consist of the simple one dimensional case using the Lucas Kanade algorithm and second part consists of the calculation for Kalman filters.

In the one-dimensional registration problem, we wish to find the horizontal disparity h between two curves F(x) and G(x) = F(x + h). This is illustrated in Figure 1.

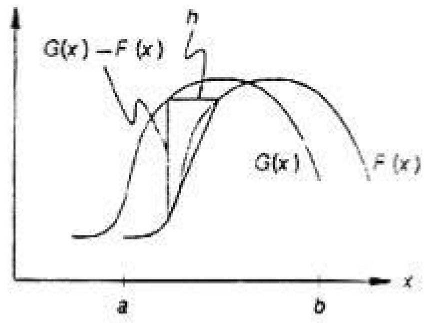

Fig 1

Our solution to this problem depends on a linear approximation to the behavior of F(x) in the neighborhood of x, as do all subsequent solutions in this paper. In particular, for small h,

$$F'(x) = \frac{F(x+h) - F(x)}{h} = \frac{G(x) - F(x)}{h},$$

So that,

$$h = \frac{G(x) - F(x)}{F'(x)}$$

But this h is approximated so if we use the weighing function inside this and integrate this over the period of the time we will get the equation as,

$$h_{k+1} = h_k + \sum_x \frac{w(x)[G(x) - F(x+h_k)]}{F'(x+h_k)} / \sum_x w(x).$$

This next part takes to the theoretical approach of the Kalman filter,

The Kalman filter uses a system's patterns (of relationships, movement, or sound) model (e.g., physical laws of movement), known control inputs to that system, and multiple (one after the other) measurements (such as from sensors) to form a guess (of a number) of the system's different amounts (its state) that is better than the guess (of a number) received/got by using only one measurement alone. So, it is a common sensor fusion and data fusion set of computer instructions.

Noisy sensor data, close guesses in the equations that describe the system (change for the better, over time), and external factors that are not here all place limits on how well it is possible to decide/figure out the system's state. The Kalman filter deals effectively with the doubt due to noisy sensor data and a little bit also with random external factors. The Kalman filter produces a guess (of a number) of the state of the system as an average of the system's (described a possible future event) state and of the new measurement using a weighted average. The purpose of the weights is that values with better (i.e., smaller) guessed (a number) doubt are "trusted" more. The weights are calculated from the covariance, a measure of the guessed (number) questions about the (statement about a possible future event) of the system's state. The result of the weighted average is a new state guess (of a number) that lies between the (described a possible future event) and measured state, and has a better guessed (number) doubt than either alone. This process is repeated at every time step, with the new guess (of a number) and its covariance informing the (statement about a possible future event) used in the following cycle. This means that the Kalman filter works recursively and needs/demands only the last "best guess", rather than the whole history, of a system's state to calculate a new.

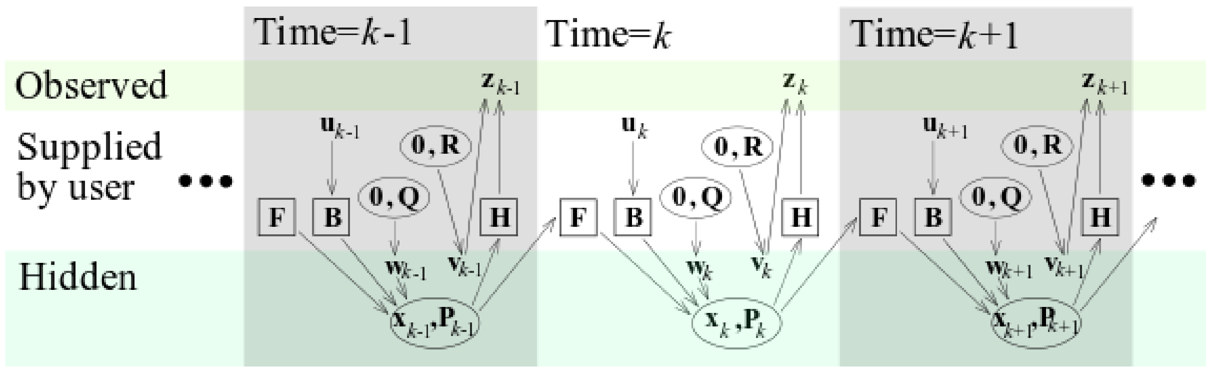

Fig 2 (Courtesy: Wikipedia.org; User:HeadlessPlatter)

Model underlying the Kalman filter. Squares represent matrices. Ellipses represent multivariate normal distributions (with the mean and covariance matrix enclosed). Unenclosed values are vectors. In the simple case, the various matrices are constant with time, and thus the subscripts are dropped, but the Kalman filter allows any of them to change each time step.

<u>Implementation</u>:

Ball Detection and Tracking:

To detect and track the ball, first the video input is taken and is divided into frames. Once the frame is arrived, the image is filtered using the Gaussian blur. This reduces the noise from the image to provide smoother results. Once the image is filtered, this image is converted to HSV from BGR. The HSV thresholding is used to detect green ball which is mentioned as below. Once the image is received after thresholding for the HSV range, the Gaussian blur image is masked with it to find the parts where ball is detected. Once the object is detected the centroid of the object is calculated. This centroid is used to track the ball. Once the ball is detected and tracked, it is displayed using the full color image . This results are forwarded to the algorithms for approach 1 and approach 2.

Green ball tracking and HSV constraints:

The green ball is tracked using color thresholding a range of HSV values for the Green Color. HSV is the most common cylindrical-coordinate representations of points in an RGB color model. **HSV** stands for *hue*, *saturation*, and *value*. HSV are used today in color pickers, in image editing software, and in image analysis and computer vision. HSV has cylindrical geometries, with hue, its angular dimension, starting at the red primary at 0°, passing through the green primary at 120° and the blue primary at 240°, and then wrapping back to red at 360°. In each geometry, the central vertical axis comprises the *neutral*, *achromatic*, or *gray* colors, ranging from black at lightness 0 or value 0, the bottom, to white at lightness 1 or value 1, the top. We had to identify a range of HSV values for greens to use in thresholding. The spread of HSV green values formed a conical distribution of values. This is because of the necessity to identify light green (bright side) and dark green (with shadow). We then identified the contours using OpenCV findContours method and also tried to detect the number of contours pertaining to the ball to keep track of i.e. the number of balls detected and tracked.

Approach 1 (Optical Flow Calculation):

Once the ball is detected and tracked, the optical flow is calculated using the optical flow Farneback Algorithm. The Lucas Kanade Algorithm was implemented but the Dense Optical Flow algorithm like Farneback proved to be better algorithm compared to the Lucas Kanade Algorithm.

The equation used to implement the algorithm is as given below:

$$\texttt{prev}(y, x) \sim \texttt{next}(y + \texttt{flow}(y, x)[1], x + \texttt{flow}(y, x)[0])$$

Using this equation we can calculate flow on basis of the prev frame provided from the video. In this project a video was taken as input and divided into frames. This frames were used to detect and track the ball. Once the green ball is tracked, the flow calculation is done using the above equation. The frames used in this is the previous and the current frame to determine the vector field. The language used here was OpenCV python with the library cv2.

Approach 2 (Trajectory estimation using Kalman Filtering):

Once the green ball is detected and tracked, we have a measurement of its current detection. Kalman filter was implemented to predict/estimate the next position and correct based on its current state. Kalman filter works in two steps: PREDICTION and UPDATE. The PREDICTION step allows to predict the position of the object knowing its history, the speed of its movements and knowing the equations that identify its movements. The PREDICTION is corrected every time a measure of the state of the object is available, this correction makes the UPDATE step. Unfortunately, the measure is not perfect, each measure has errors, so PREDICTION and UPDATE are weighted using information about measure and prediction errors. We used the OpenCV implementation of Kalman Filter and its methods. This had all the necessary setup matrices (Transition matrix, Measurement Matrix, Process Noise Covariance etc.) as attributes and the KalmanFilter was corrected with a simple KalmanFilter.correct() where we passed the recent measurements as arguments. We then plotted the trajectory of the ball using detected points and estimated kalman points (at certain time t) where the detection was not available. The code was implemented using C++ and OpenCV libraries.

Results:

Approach 1 (Optical Flow Calculation):

As we can see from the results below, the vector field for the optical flow is obtained after the implementation of approach 1. In this approach, the flow vector field calculated describes the motion of the moving object. The final video for results is attached along with this report in the results folder.

Constraints:

It is visible from the image and video that as the video taken from the camera is not stabilized the flow vector field detects the flow in the parts, where actually camera showed slight movement instead of object. In the future implementation, this can be avoided by shooting a video with camera in static position.

Goodness of results:

In this section, we would like to discuss about the results and its value for the project. The resulting video shows the vector field of the moving object which is tracked. This portion is used in the hawkeye project not just to determine the trajectory of the moving object, but with the flow vector field we can actually determine the direction of trajectory. This can be helpful in the velocity calculation. This velocity and gravity calculation from the flow field is required to estimate the trajectory in the sports where bounce of the ball matters to provide a decision. Regarding the correctness of results, we can see the entire vector field in the video provided, excluding the constraint mentioned, there is no improper vector fields detected. So we can see the changes in vector field only for the moving object. Fig.3 and Fig.4 below are the part of results.

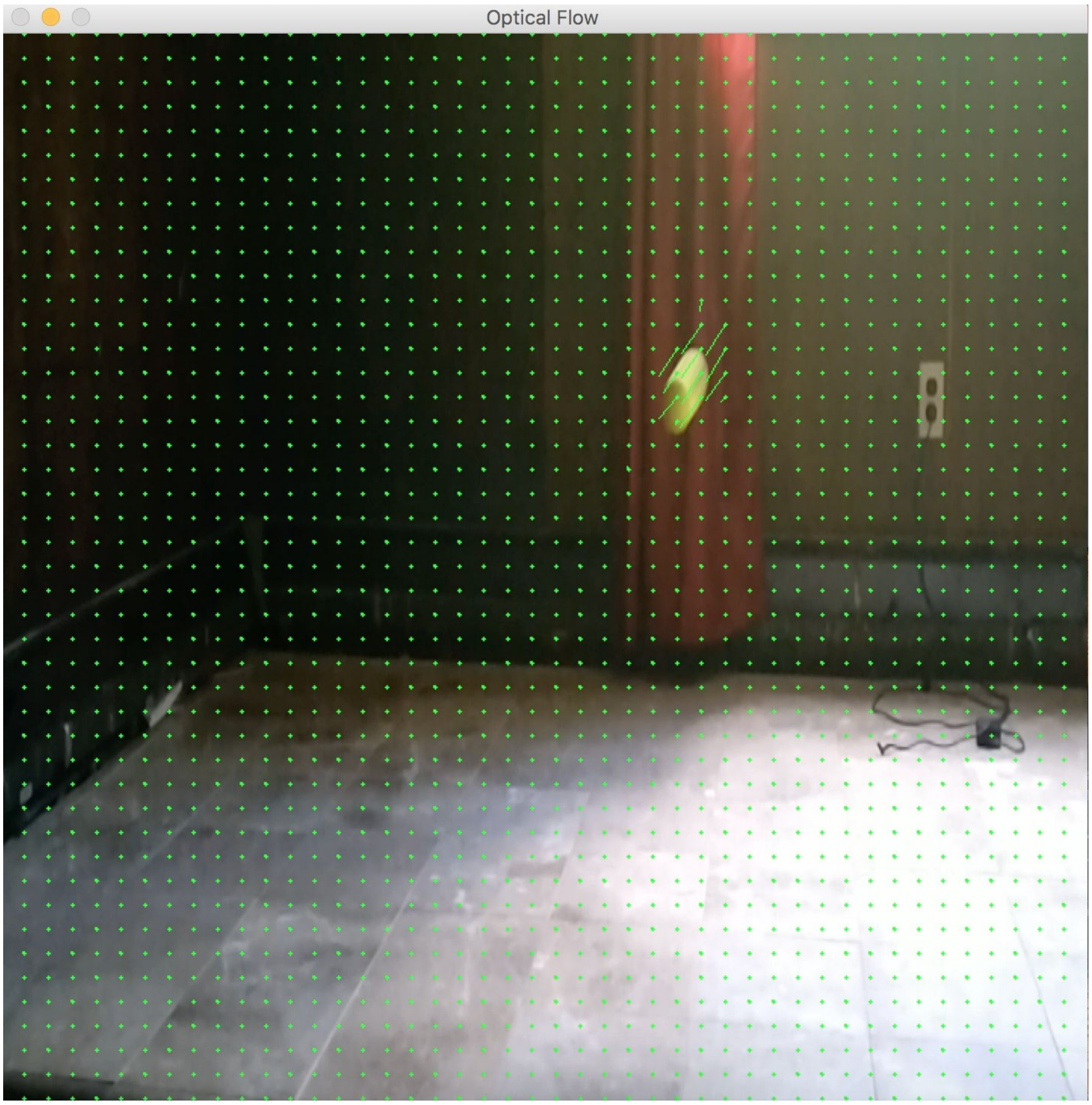

Fig 3. Vector field for object moving up

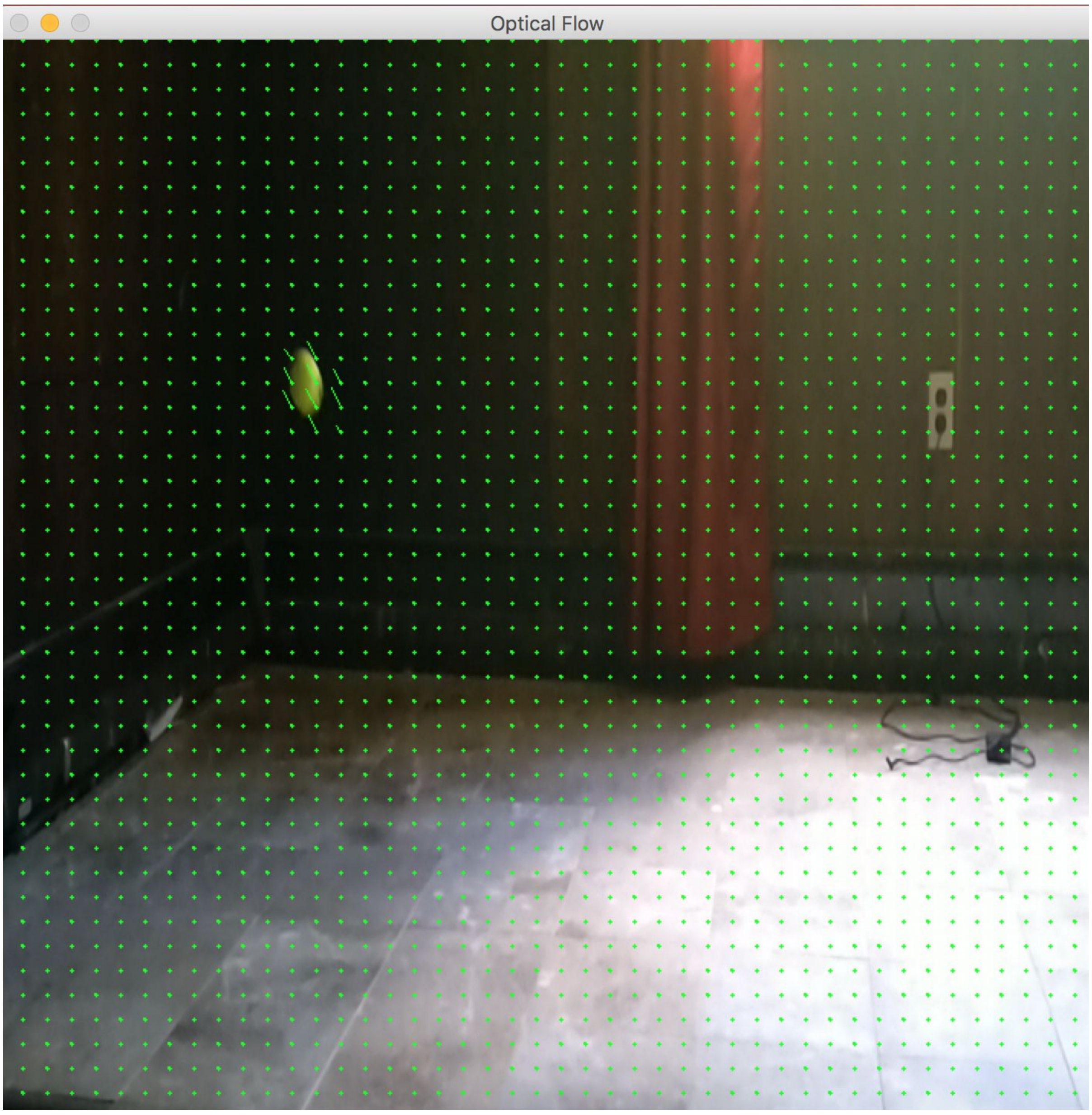

Fig 4. Vector field for object moving down

Approach 2 (Trajectory estimation using Kalman Filtering):

- The green rectangle identifies the bounding box of the ball detected during the measure step or detection step.
- The red rectangle shows the “estimated state” of the ball, the result of the prediction step of the Kalman Filter. It is really interesting to note how the estimated state overlaps the measured state meanwhile the ball moves linearly, but overall it is interesting to note that the estimation of the state stays valid even if the ball is behind the bottle.
- Fig.5 and Fig.7 are examples of detection and prediction applied on two videos Sample.mov and finalvid.mov.
- As you can see from Fig.6 and Fig.8 the kalman filter provides close to perfect estimations of the trajectories of the ball when displayed along with the detected points. We use both sets of points to get a smooth trajectory of the ball.
- Major drawbacks of this approach is the detection mechanism, if any other patch that may visibly appear as green patches due to shadows or any other phenomena could be detected as a ball also. Future work would be to improve the ball detection technique and use kalman filter to estimate the trajectory and plot a smooth trajectory.

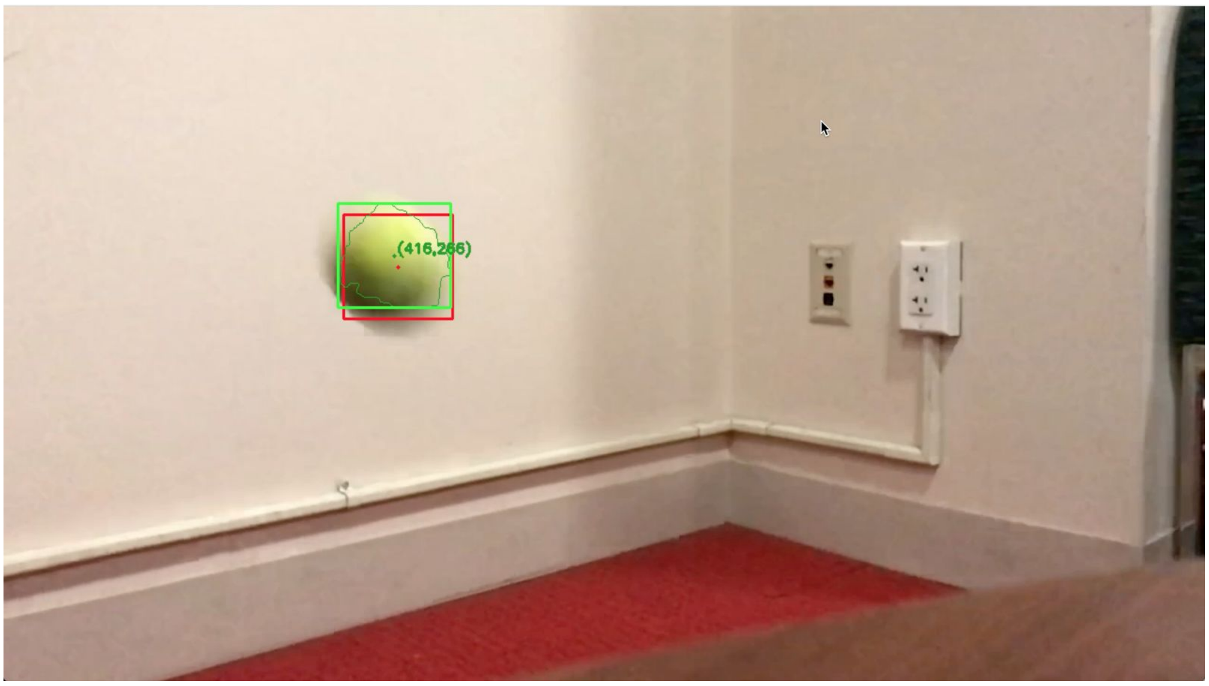

Fig. 5 - Green rectangle detects the green ball and red rectangle shows the estimated state of the ballin Sample.mov

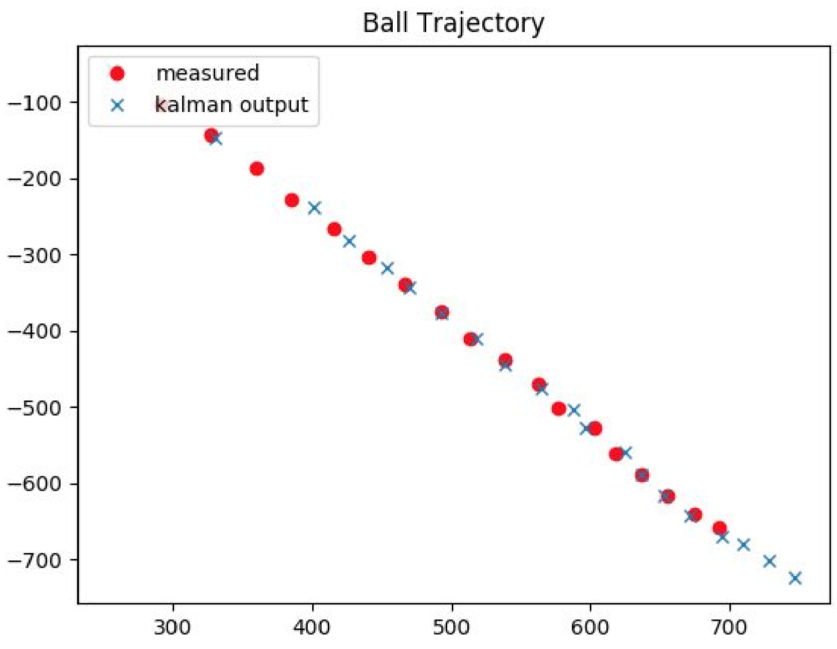

Fig.6 - Trajectories of detected points (red) and kalman estimated points of ball in Sample.mov

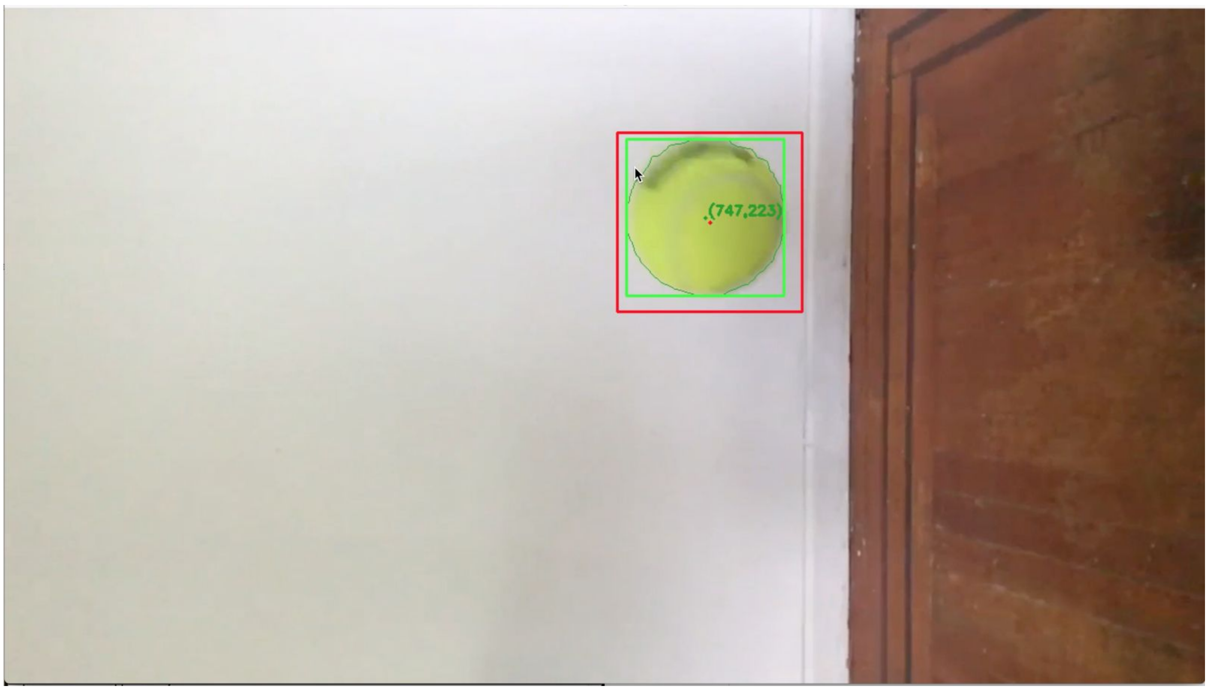

Fig. 7 - Green rectangle detects the green ball and red rectangle shows the estimated state of the ball in finalvid.mov

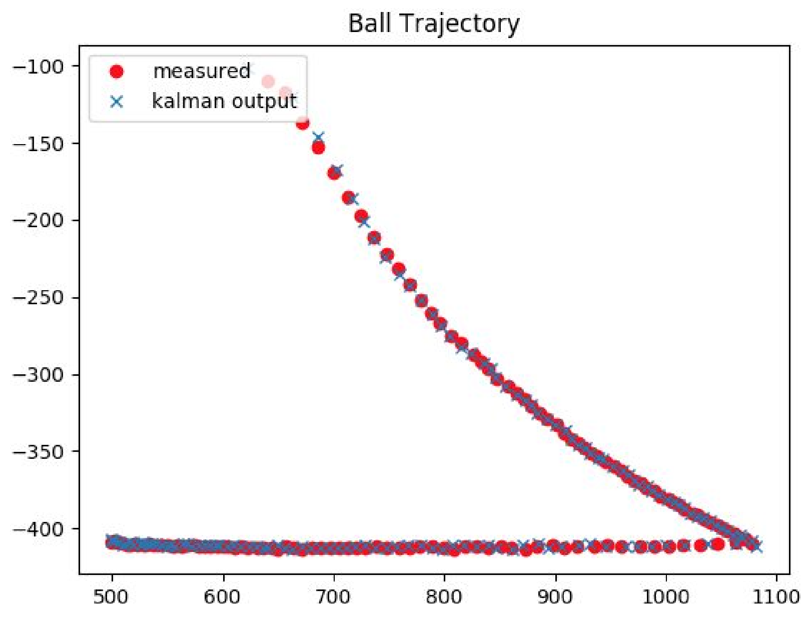

Fig.8 - Trajectories of detected points (red) and kalman estimated points of the ball in finalvid.mov

Conclusion:

We can conclude that with the results of optical flow vector field and Kalman filter point predicting we can actually estimate the trajectory of the ball i.e. Hawkeye, which can be used in many sports as referral systems. In particular, we observe that combining the results of Kalman Filter with the originally tracked points helps us understand the trajectory better in the case of small disturbances in the video. The optical flow vector field calculation helps us better understand the direction of trajectory which important in some of the sports like cricket.

In addition, we can conclude image processing like tracking a green ball can be more helpful in tracking the trajectory properly. In the future, better detection technique could be used for tracking the ball instead of tracking the color.

Furthermore, it is interesting to notice that we have used a basic Kalman Filter algorithm not any variations (EKF,UKF,etc). This is due to simplicity of the ball movements in the video. In the future it would be helpful to use videos with complicate movements, in cases where using other variations should be feasible.

References:

1. Bruce D Lucas (1984) General Image Matching through the Method of Difference (Doctoral Dissertation)
2. B. D. Lucas and T. Kanade (1981), An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop
3. Yuanyuan Zhang, (2015), Robust Object tracking based on Masked Camshift Algorithm.
4. R.E. Kalman, A new Approach to Linear Filtering and Prediction Problems
5. Theoretical Estimation of Kalman Algorithm User: HeadlessPlatter, wikipedia.org
6. Paul Zarchan, Fundamentals of Kalman Filtering: A Practical Approach, ISBN: 978-1-56347-455-2
7. Roydon C.S.; Moore K.D.(2012), Use of speed cues in the detection of moving objects and moving observers
8. Two-Frame Motion Estimation Based on Polynomial Expansion by Gunnar Farneback
9. http://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html

Acknowledgements:

We have partially used techniques considered in the class and homeworks. For this, we would like to Acknowledge Prof. Guido Gerig and course TA's Rupanta Dutta and Guangqi Chen.

List of Figures:

Appendix:

Work Division:

Monil Shah:

- Green Ball tracking and trajectory calculation
- Optical Flow Vector field using Lucas Kanade and Farneback Algorithm
- Visualizing and displaying the results (videos of optical flow results in results folder)
- Source code in OpenCV Python

Tabish Sada:

- Green ball tracking and HSV constraints
- Trajectory estimation using Kalman Filtering
- Plotting detected and estimated points to determine ball trajectory.
- Source code in OpenCV C++ and plotting trajectory points using matplotlib Python.

Samvid Jhaveri:

- Kalman Filter theoretical implications and research techniques
- Code for visualizing results and video to frames
- Plotting and comparing both approaches
- Source code in openCV Python