Real Time Object (Ball) Tracking System using OpenCV

ECE-GY 6183 Digital Signal Processing Laboratory

Name: Arunkumar NP **NetID:** an3333

Name: Vishwanath Babu CA **NetID:** vc2173

Link to Code: https://github.com/arunkumar1531/ECE-GY-6183-Project/tree/main

Table of Contents

Introduction

A ball tracking system is a technology that is used to detect and track the movement of a ball in a given environment. The purpose of this technology is to provide real-time information about the location and movement of the ball, which can be used in various applications such as sports analysis, entertainment, and robotics. One of the most popular tools for ball tracking is OpenCV, which is an open-source computer vision library that provides a wide range of functions for image and video processing. OpenCV has been widely used in various fields such as computer vision, machine learning, and robotics, and it has become a standard tool for many researchers and developers.

In this project, we will be using OpenCV to build a ball tracking system that can detect and track the movement of a ball in a video stream. The system will use image processing techniques to detect the ball in the frame, and then it will use object tracking algorithms to track the movement of the ball over time. The goal of this project is to build a robust and efficient ball tracking system that can be used in a variety of applications. To achieve this goal, we will be using a combination of image processing techniques and object tracking algorithms to detect and track the ball in real-time. [1]

Overall, the ball tracking system is a challenging and exciting project that involves a combination of computer vision, machine learning, and robotics. Ball tracking is mainly used in sports like tennis, soccer, cricket, etc. Since there is always the aspect of human error, it is common for the above mentioned sports to use machines to track objects. So, in the recent past there have been many research in the field of object tracking and detection. In this project, we will be implementing the ball tracking system using OpenCV. [2]

Common Challenges

Real-time object tracking is a challenging task that involves detecting and tracking the movement of objects in a video stream. There are several factors that can affect the performance of an object tracking system, and it is important to understand these challenges in order to design and develop effective tracking algorithms. One of the main challenges in real-time object tracking is the variability of the objects being tracked. Objects can vary in size, shape, color, and texture, and these variations can make it difficult to accurately detect and track the objects. [4][5] For example, an object tracking system designed to track a car may struggle to track a bike or a pedestrian, due to differences in shape and size.

Another challenge is the presence of clutter and occlusion in the environment. Clutter refers to the presence of multiple objects in the same frame, which can make it difficult to distinguish the target object from the background. Occlusion occurs when the target object is partially or fully obscured by another object, which can make it difficult to detect and track the object. [4] A third challenge is the presence of noise and variations in the image or video data. Noise can be caused by factors such as lighting conditions, camera motion, and image compression, and it can interfere with the accuracy of the object tracking system. Similarly, variations in the image or video data, such as changes in resolution or frame rate, can also affect the performance of the object tracking system.

In summary, real-time object tracking is a complex task that involves a range of challenges, including variability in the objects being tracked, clutter and occlusion in the environment, and noise and variations in the image or video data. To overcome these challenges, object tracking algorithms must be robust and adaptable, and they must be able to handle a wide range of conditions and variations in the data.

Libraries

The following packages were used in this project.

- OpenCV
- Numpy
- OS Module
- Time
- Glob
- Deque
- VideoStream
- Imutils
- Argparse

Code

This code predominantly tracks balls that are used in sports like tennis, cricket, etc. We can track balls of different colors based on the range of their HSV values that can be chosen via command line arguments. There are totally 4 different command line arguments that can be used to run the code.

- **Object Color (-c or –object_color)**
  The argument lets you choose the color of the ball that you can track based on the HSV range of the color. The available color options are black, white, red, green, blue, yellow, purple, orange, gray, and cyan. The default value for this argument is blue.

- **Trail length (-t or –trail length)**
  This argument lets you choose the length of the tracking trail which contains the values of past location values of the ball. The larger the value, the larger is the memory buffer. This in turn increases the length of the trail. The default value of the length is 64.

- **Input video (-i or –input_video)**
  This argument lets you track the ball in a video rather than real-time tracking. You have to enter the absolute path of the video file as the input. Then, the ball in the video is tracked and the tracked frames are stored in “frames” folder along with the tracked video file. When no video input is given to the program, it automatically defaults to real-time tracking and opens the webcam in your computer.

- **Trail width (-w or –trail_width)**
  This argument is similar to trail length. It controls the width of the trail line or the tracking path. The default value for this parameter is 5.

The program starts with initializing the trail path using the variable trail. This is of deque type rather than list because it is much faster when performing operations like pop(), append(), etc. Also, the complexity of such methods is O(1) in deque when compared to O(N) of lists.

Now, each frame of the webcam input or input video is captured based on the command line arguments and few image processing techniques like Gaussian blur, resizing etc. is performed which is followed by conversion of RGB scale to HSV scale. This is followed by applying morphological methods like dilation and eroding is performed to identify the object of chosen color in the frame.

Now, the contours are identified and we initialize the bounding circle for the object detected. Next, we use Moments in OpenCV to obtain the radius and centre of the object. Now, two circles are formed – one enclosing the object to be tracked and another circle to mark the centre of the object. The centroid value of the object is appended to the trail deque for every passing frame. Finally we define the thickness of the trail and display the tracked frames.

In the case of real-time tracking, the tracked frames are directly displayed to the user. In case of tracking objects in a video, the tracked frames are saved in "frames" folder before the tracked frames are displayed to the user. Finally, using the saved frames, the Images are converted to an mp4 video and it is stored in the same "frames" folder.

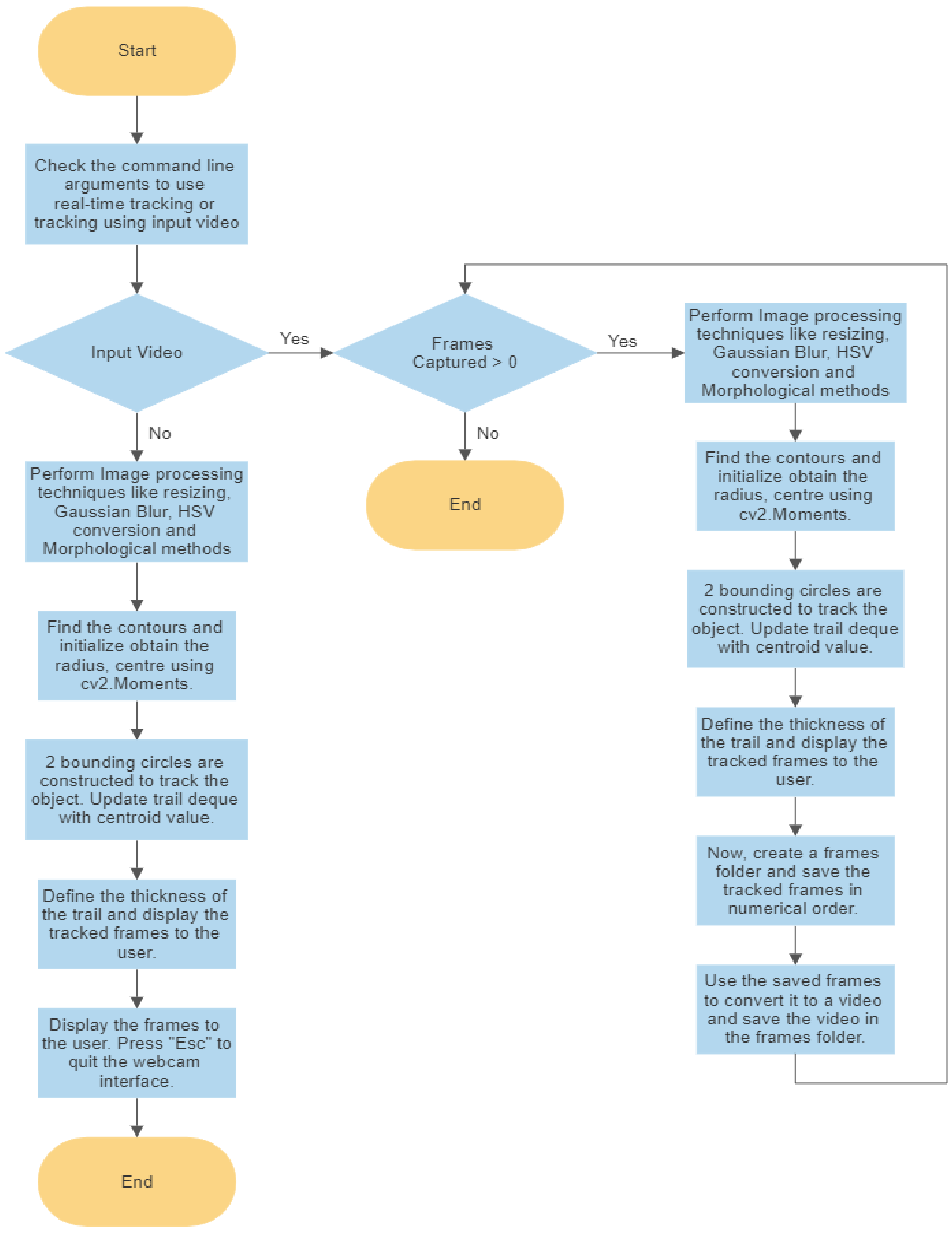
Start
Check the command line arguments to use real-time tracking or tracking using input video
Input Video
Yes
No
Perform Image processing techniques like resizing, Gaussian Blur, HSV conversion and Morphological methods
Find the contours and initialize obtain the radius, centre using cv2.Moments.
2 bounding circles are constructed to track the object. Update trail deque with centroid value.
Define the thickness of the trail and display the tracked frames to the user.
Display the frames to the user. Press "Esc" to quit the webcam interface.
End
Frames Captured > 0
Yes
No
End
Perform Image processing techniques like resizing, Gaussian Blur, HSV conversion and Morphological methods
Find the contours and initialize obtain the radius, centre using cv2.Moments.
2 bounding circles are constructed to track the object. Update trail deque with centroid value.
Define the thickness of the trail and display the tracked frames to the user.
Now, create a frames folder and save the tracked frames in numerical order.
Use the saved frames to convert it to a video and save the video in the frames folder.

Figure – 1 Code Flow

Running the Code

Example 1: Run the tracking code in real-time to track a blue coloured ball.
Python ball_tracking.py –c “blue”

Example 2: Run the tracking code to track a tennis ball in a video
Python ball_track.py –i “path/to/video/file.mp4” –c “cyan”

Example 3: Run the tracking code with a specific trail length and width.
Python ball_tracking.py –t “128” –w “8”

Results

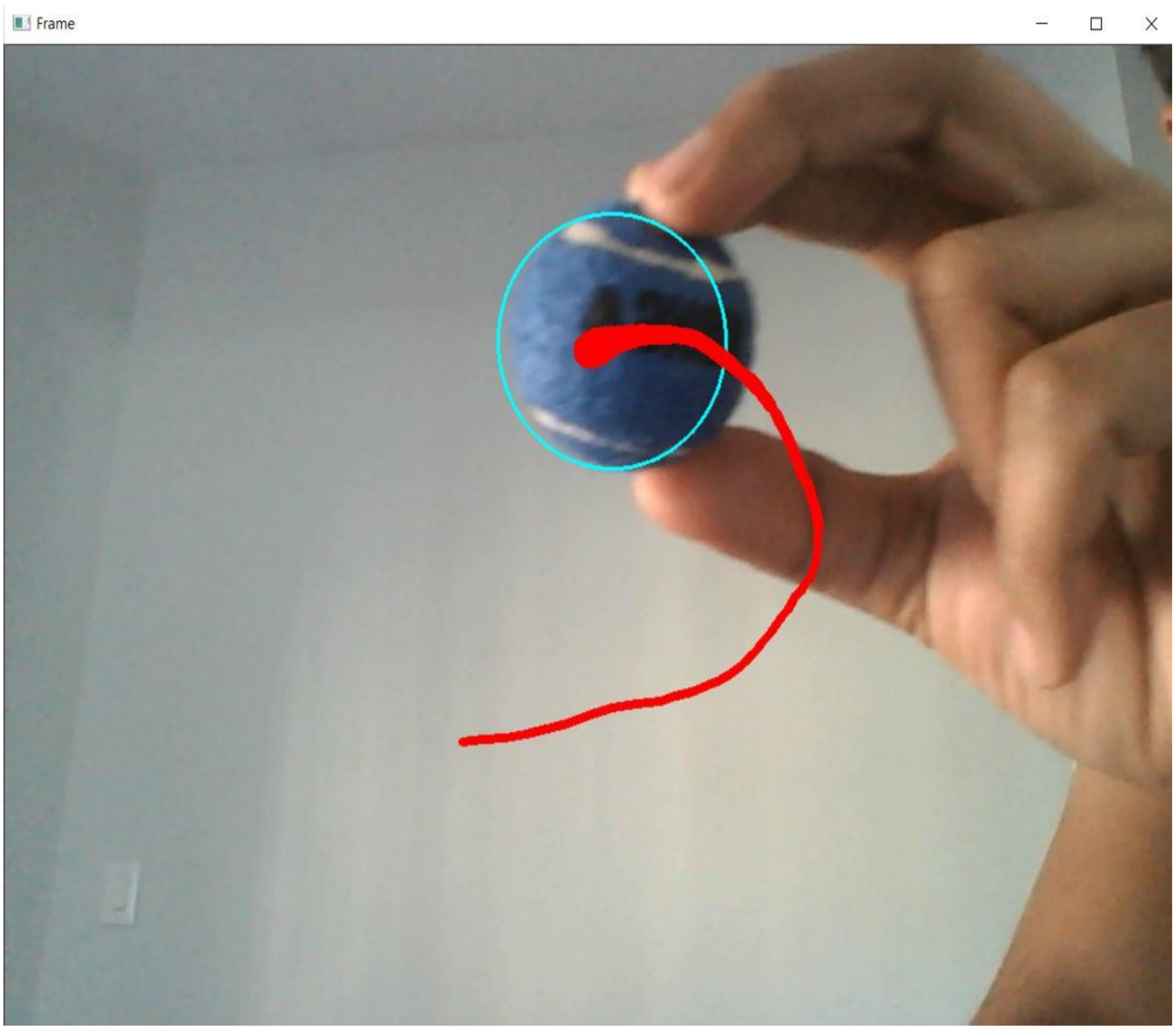

Figure 2 – Webcam Output (tracking a blue ball)

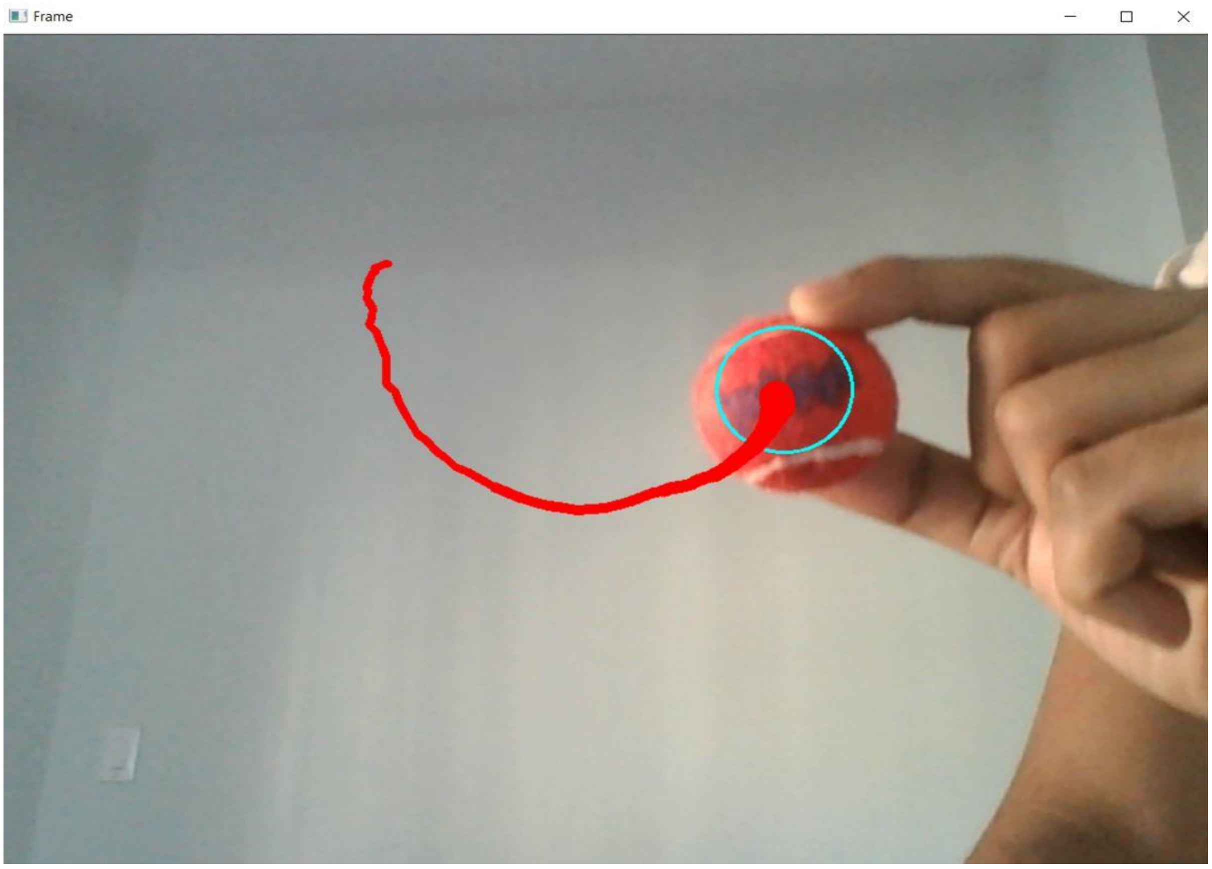

Figure 2 – Webcam Output (tracking a red ball)

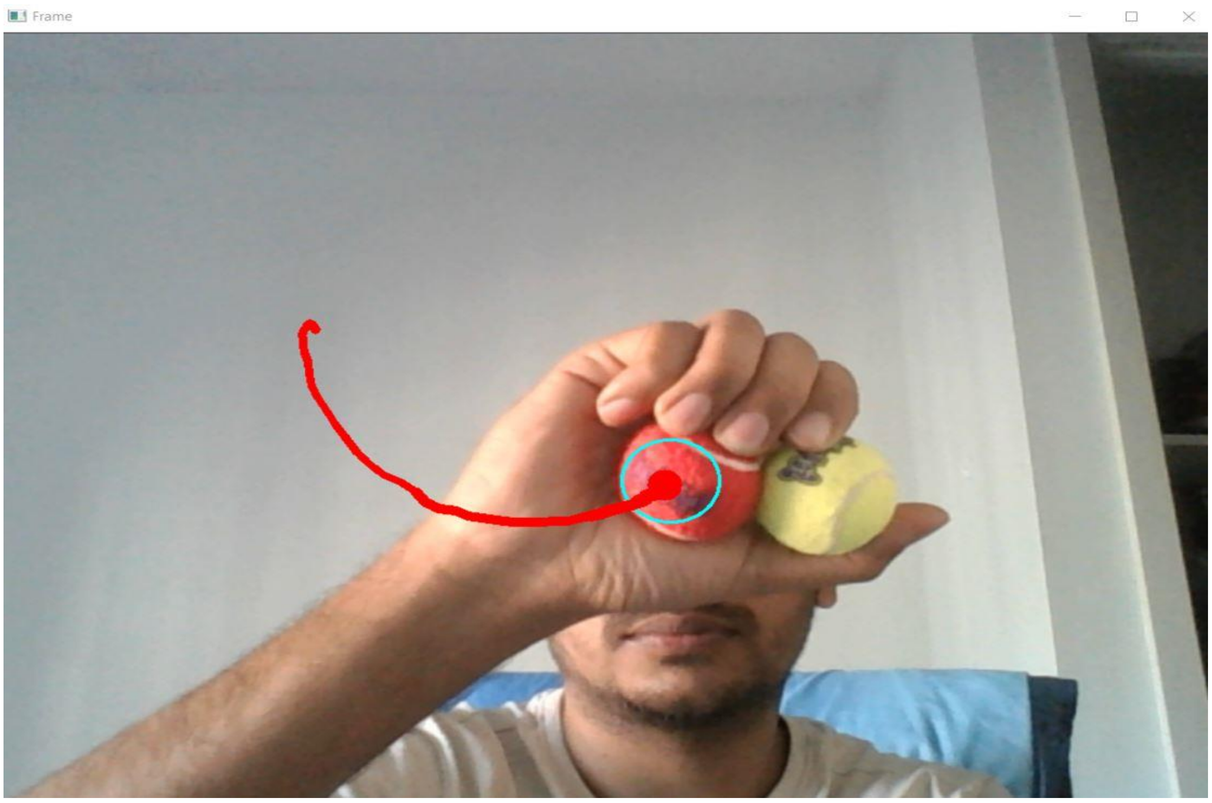

Figure 4 – Webcam Output (Tracking multiple balls at once)

Figure 2-4 depicts the tracking process in real-time webcam video. It can be seen that the algorithm is capable of tracking different colored objects. However when multiple objects are present in a single frame the algorithm fails to recognize both the objects. It is capable of dealing with only one ball in a given frame.

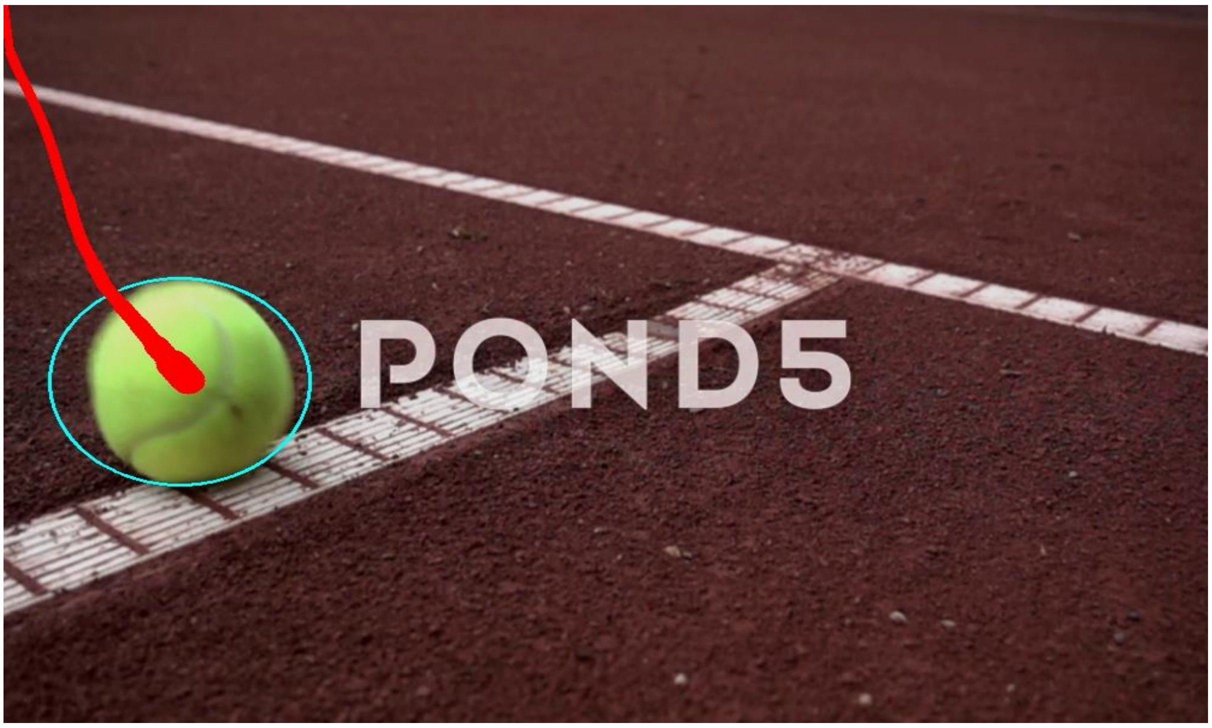

Figure 5 - Tracking object in an input video (Frame 34)

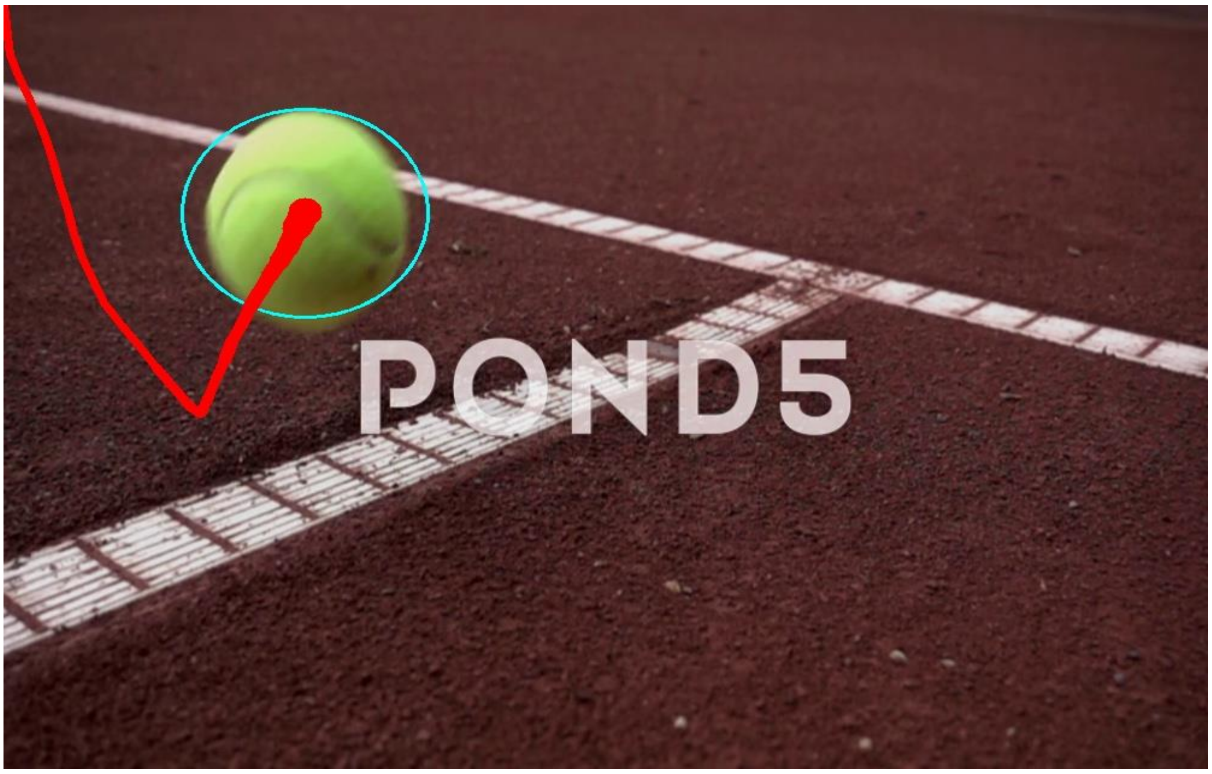

Figure 6 - Tracking object in an input video (Frame 74)

Figure 5-6 shows the tracking output when using an input video instead of real-time tracking. The bounding circles are well established and the path of the ball is tracked with good accuracy.

Future Work

Since the current algorithm has a few limitations there is still room for development.

- The algorithm can be modified to track multiple objects in a given frame.
- As of now, the code accepts only one input video file. This could be changed to accept multiple video files and process them in any specific order.
- Can be modified to track objects of different sizes – Example: a tennis ball, football, soccer ball, etc.
- Integrate the algorithm with Machine learning to improve the accuracy of the tracking and predict the position of the ball in the presence of clutter.

Reference

1. Qadir, J., Shafique, M., & Hassan, A. (2014). Real-time object tracking using Adaboost and Kalman filter. International Journal of Computer Applications, 98(14), 15-20.
2. Zhao, Y., & Chellappa, R. (2012). OpenCV 2 computer vision application programming cookbook. Packt Publishing Ltd.
3. Kalal, Z., Mikolajczyk, K., & Matas, J. (2012). Real-time object tracking using the median flow algorithm. International Journal of Computer Vision, 98(2), 168-182.
4. Lin, Y., & Chen, S. (2014). Real-time ball tracking using Kalman filter and mean shift algorithm. Measurement, 47, 51-58.
5. Hsiao, H., Chen, T., & Lai, J. (2013). Real-time ball tracking using multiple cues. IEEE Transactions on Automation Science and Engineering, 10(3), 823-828.