VMG Assignment: Object detection

Parthasarathi Khirwadkar

Overview

This assignment aimed to detect and track the basketball in a video taken from the given dataset. Since all the videos in the dataset are similar in the sense that every video has the same background and the same color of the ball, under various simplifying assumptions, the required object can be detected using traditional computer vision techniques. Due to the lack of ground truth object detection, only qualitative results are provided. The various parameters involved in this method were also hand-tuned to obtain qualitatively best possible results.

Method

The nature of the data allows us to make the following assumptions:

- The ball is approximately of a uniform color which is different from the background.
- The background does not change with time.
- The ball always follows a projectile motion.
- The radius of the ball, as seen in the video, is approximately the same in every video.

Under these assumptions, the approach taken to solving the object detection problem was via color thresholding and background subtraction. These techniques were coupled with shape-based filtering and Kalman filtering to reduce noise and to predict the location of the ball if it goes out of the frame for a few frames. The proposed method can be broken down into the following steps:

1. Obtain a patch of the ball from the user and find a range of RGB values in the path. This task needs to be done only once for the entire dataset since the ball has the same color in every video.
2. Obtain the frame and remove the background using the MOG2 background subtractor implemented in OpenCV.
3. Threshold the foreground to obtain a binary mask. Apply erosion and dilation to remove noise.

4. Find all the connected components in the binary mask, and sort them with respect to their area.

5. The estimate the center and the radius of the minimum enclosing circle corresponding to each connected component.

6. The largest connected component with the radius of the minimum enclosing circle within a known range and with the center closest to the previous estimate of the location of the ball is the contour corresponding to the ball.

7. Update the Kalman filter and use its estimate to reduce the noise in the prediction of the center of the basketball.

Other methods, including Histogram Backprojection and HSV color thresholding, were also tried out. However, RGB thresholding gave the best qualitative results.

Code files

The code was fully implemented in Python 3 and OpenCV 3. The `pykalman` package is required for Kalman Filtering. It should also be noted that this code was tested on macOS and not Linux, although the code should replicate on Linux as well.

main.py

This is the main code which is used to perform the object detection. It calls other utility functions implemented in the `track_utils.py`. To run the `main.py` code file, use the following command:

```
python main.py -v path/to/video.avi -o path/to/output.avi -l limits.npy -s -vis
```

The estimated ball location will be printed on the terminal. Following optional arguements can be provided to `main.py`

- `-v, --video` : path to the input video file (necessary)
- `-o, --output` : path for the output video file
- `-s, --save` : save the ball detections as `measurements.npy`
- `-vis, --visualise` : the code will visualise each frame while the code is running
- `-l, --limits` : path to the `limits.npy` file. If not provided then the user is prompted to input a patch of the ball using mouse.
- `-bg, --background` : path to extracted background image. By default `background.png` is used, if arguement is not provided

track_utils.py

Implements all the required utility functions for `main.py`

visualise.py

This script is used to generate a image visualisation of the annotated video. Internally, it reads `measurements.npy` to obtain annotations (make sure to use `--save` arguement while running `main.py`). Use following command to generate new visualisation:

```
python visualise.py -v path/to/video.avi -o path/to/output.png
```

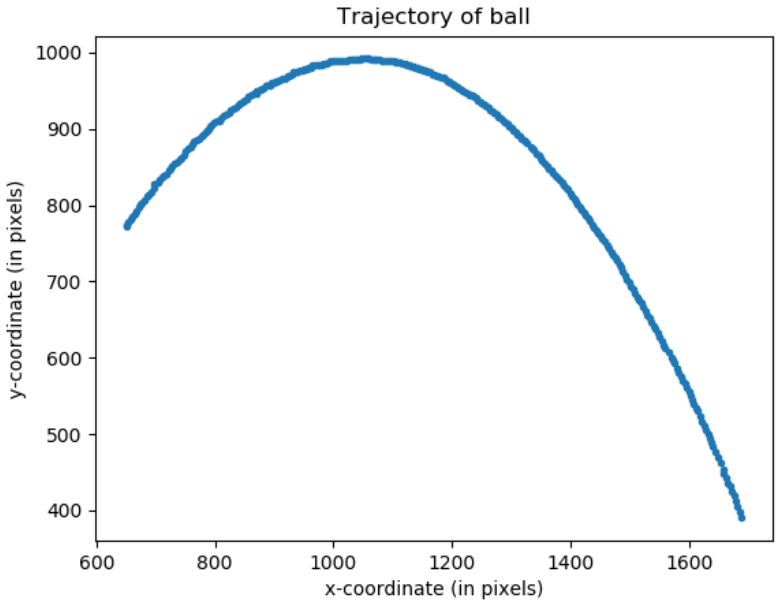
obtain_background.py

This code is used to obtain the background from a video, which is taken as the median of pixel value at each pixel taken across all frames. The optional arguement are the same as `visualise.py`.

Results

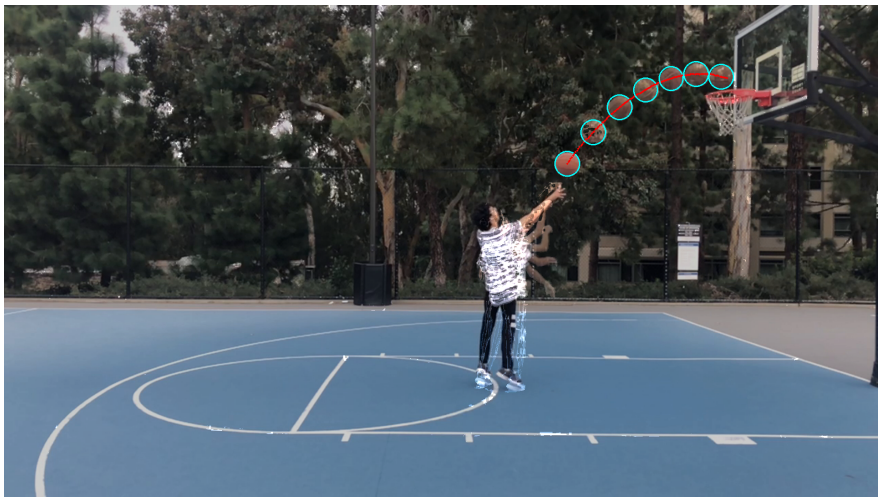
The following results were obtained by using `limits.npy` and `background.png` provided in the repository. Note that the parameters for Kalman filter and other morphological operations were NOT tuned using these videos. These videos were picked at random from the rest of the dataset.
The y-coordinate in the plots below is = frame height − y-coordinate of the ball with respect to image coordinate system. The origin of the image coordinate system lies on the top-left corner, but for the purpose of plotting, the origin was shifted to bottom left corner and the y-axis was inverted along with the above mentioned transformation of the data.
The annotated videos for the following are available at : link

(a) Visualisation of ball and bounding circles

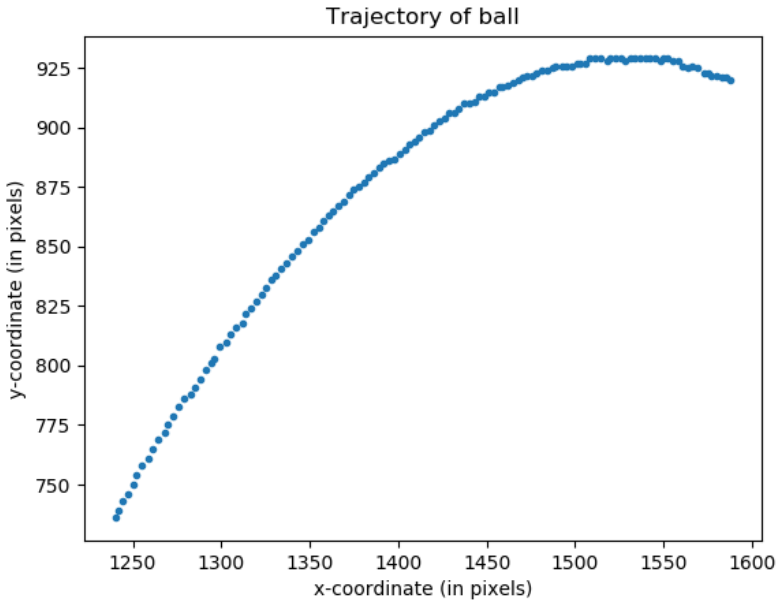


(b) Plot of the coordinates of the ball detection

Figure 1: Results for the video 4355-4651.avi in Set1

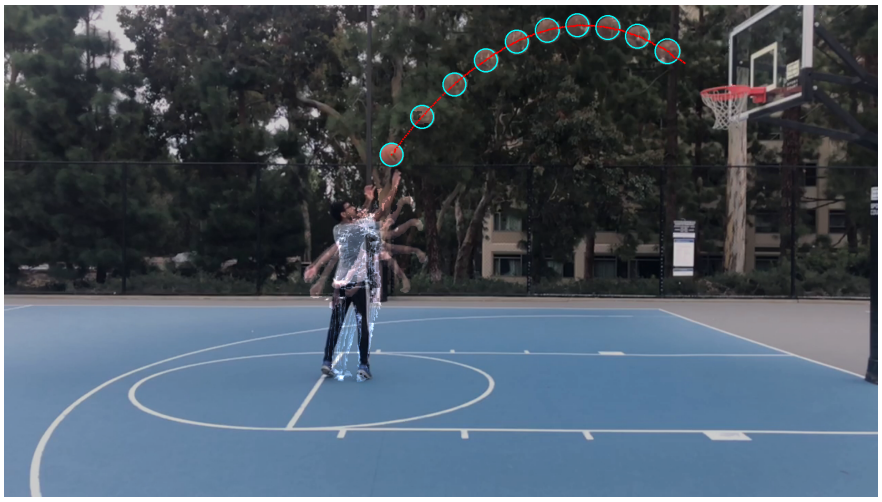


(a) Visualisation of ball and bounding circles

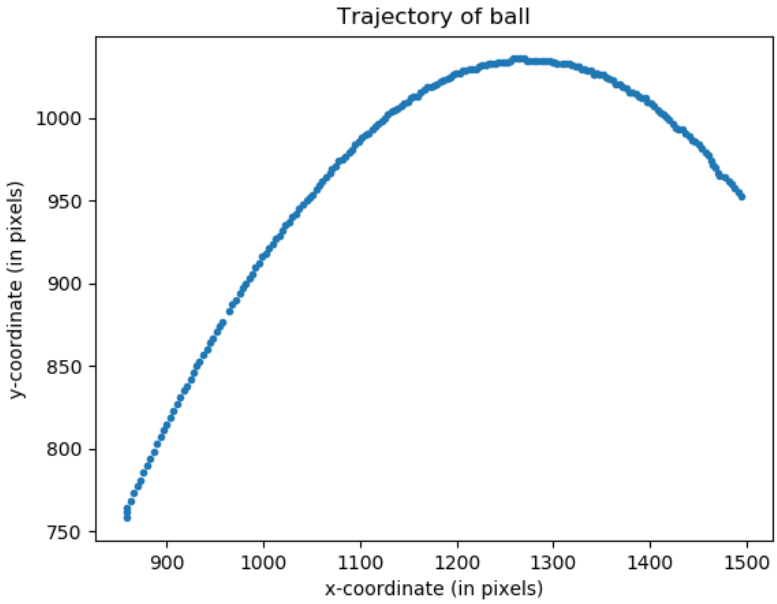


(b) Plot of the coordinates of the ball detection

Figure 2: Results for the video 14573-14698.avi in Set1



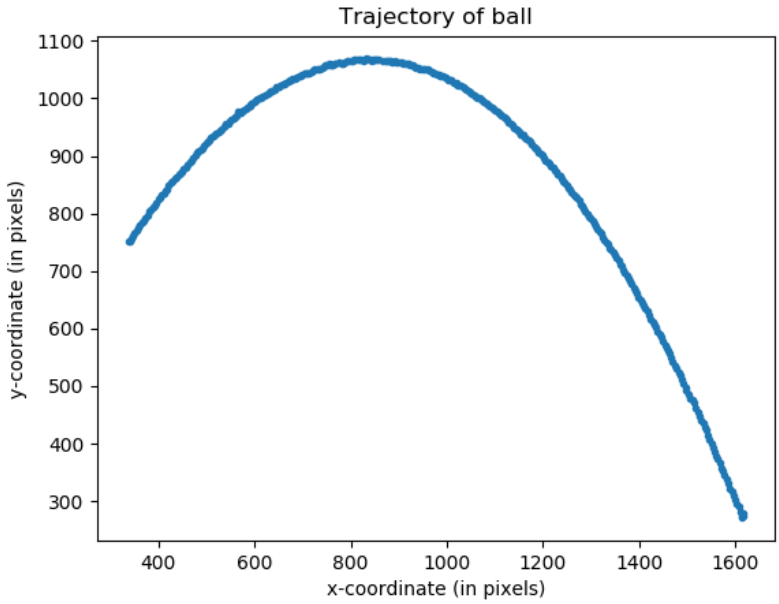
(a) Visualisation of ball and bounding circles



(b) Plot of the coordinates of the ball detection

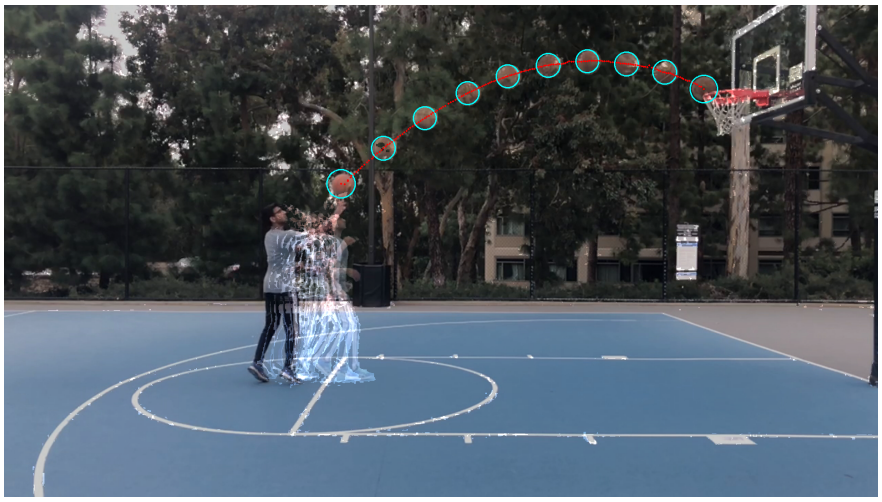
Figure 3: Results for the video 73686-73878.avi in Set5

(a) Visualisation of ball and bounding circles

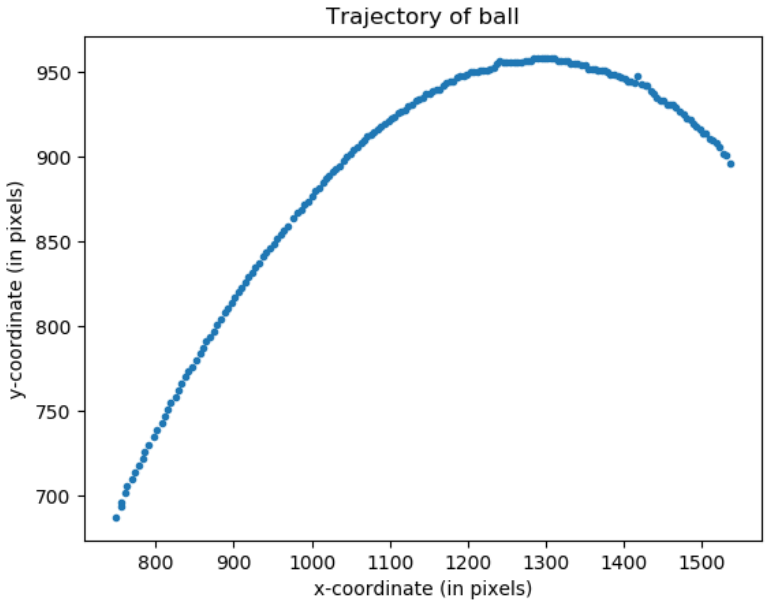


(b) Plot of the coordinates of the ball detection

Figure 4: Results for the video 77595-77946.avi in Set5



(a) Visualisation of ball and bounding circles

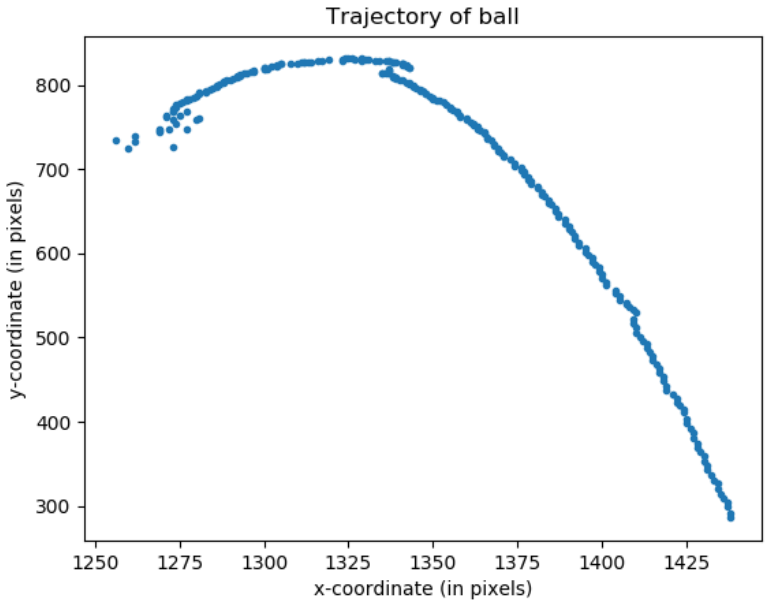


(b) Plot of the coordinates of the ball detection

Figure 5: Results for the video 94026-94206.avi in Set23

The case below shows that these assumptions can give noisy results in some situations. However, it was observed that the assumptions and the tuned parameters worked reasonably well for most videos.

(a) Visualisation of ball and bounding circles



(b) Plot of the coordinates of the ball detection

Figure 6: Results for the video 6720-6978.avi in Set1

Conclusion

A method for object detection using traditional computer vision techniques is demonstrated. Under certain assumption on the dataset, it is shown qualitatively that this approach work reasonably well for most of the videos in the dataset. Excluding the time taken to read each frame and displaying it, the time taken for the complete detection operation is around 0.04 seconds per frame, which is about 25 fps on a Dual-Core Inter i5 processor.