VMG Candidate Project Report

Arda Okan

UCLA Reference Number: 475042366

GitHub: https://github.com/ArdaOkan97/Arda_Okan-VMG_Candidate_Project

Approach

Since we only need to detect the basketball, I decided that using a data driven approach would be unnecessary. I used the OpenCV library in Python for this project.

I first set a range for the orange color of the basketball in HSV color domain. After experimenting with the range for a little, I decided that the color range should be between (2, 60, 60) and (10, 255, 255). After setting the range, I added gaussian blur for smoothing to each frame of the video and converted the color domain from BGR to HSV for applying mask to the frames.

Next, I created the mask for each frame according the color range that I set before. To remove small noises, I also added erosion to the mask. You can see the original frame and the generated mask below for that frame.

Fig 1: Original frame

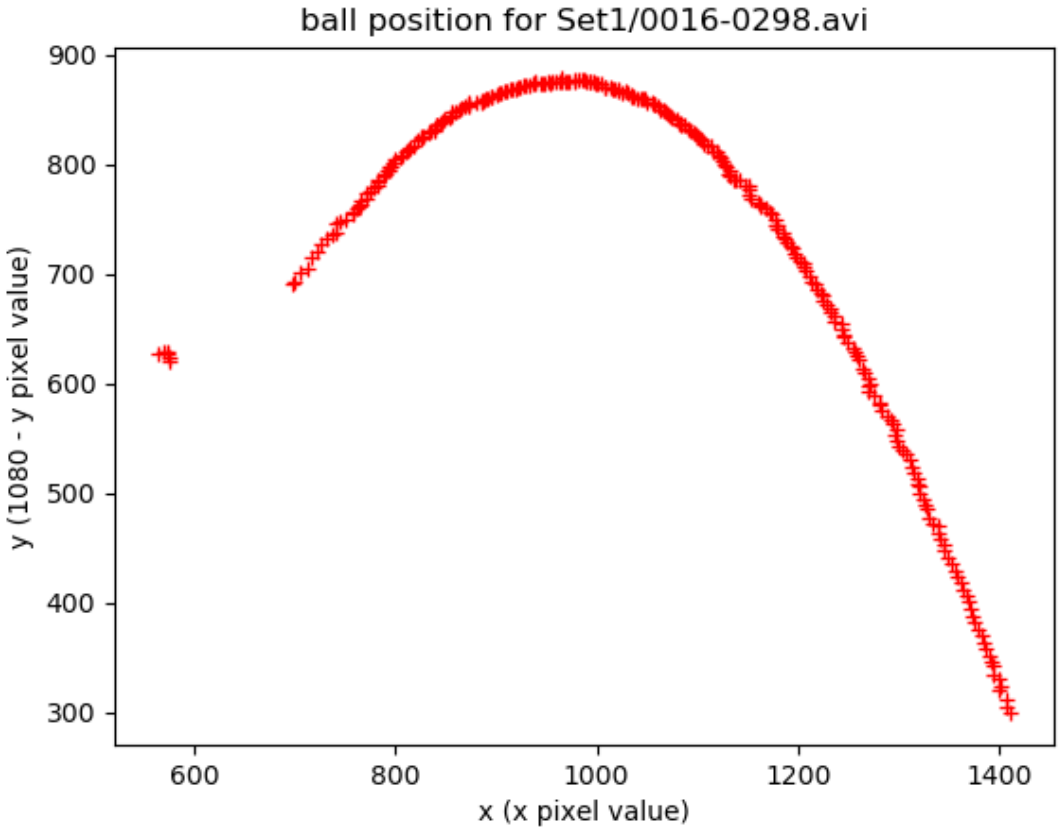
Fig 2: Created mask according the frame and color range

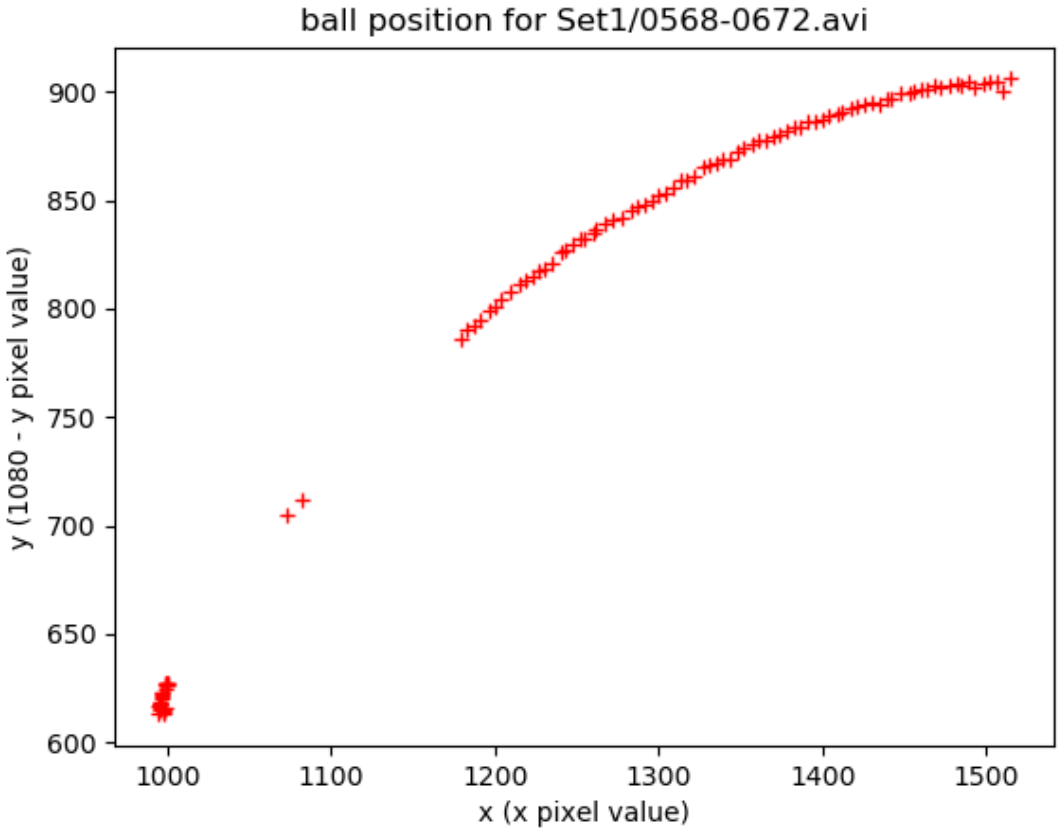
After generating the mask, I find the white contours in the mask. Then, I get the biggest one from the found contours and calculated the smallest circle that is enclosing this contour. I calculated the radius and the center of this contour. Finally, according the calculated circle, I bounded the basketball with a red box which can be seen below.

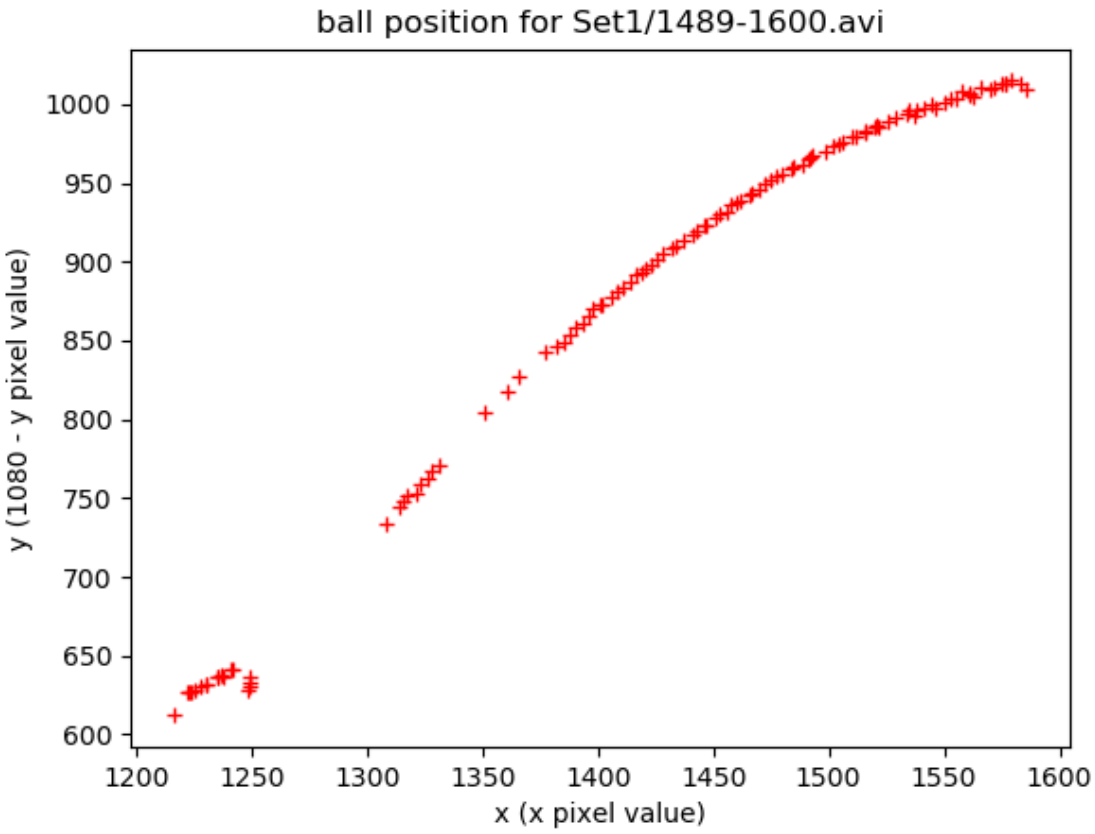
Fig 3: Detected basketball according to the mask and calculated contours

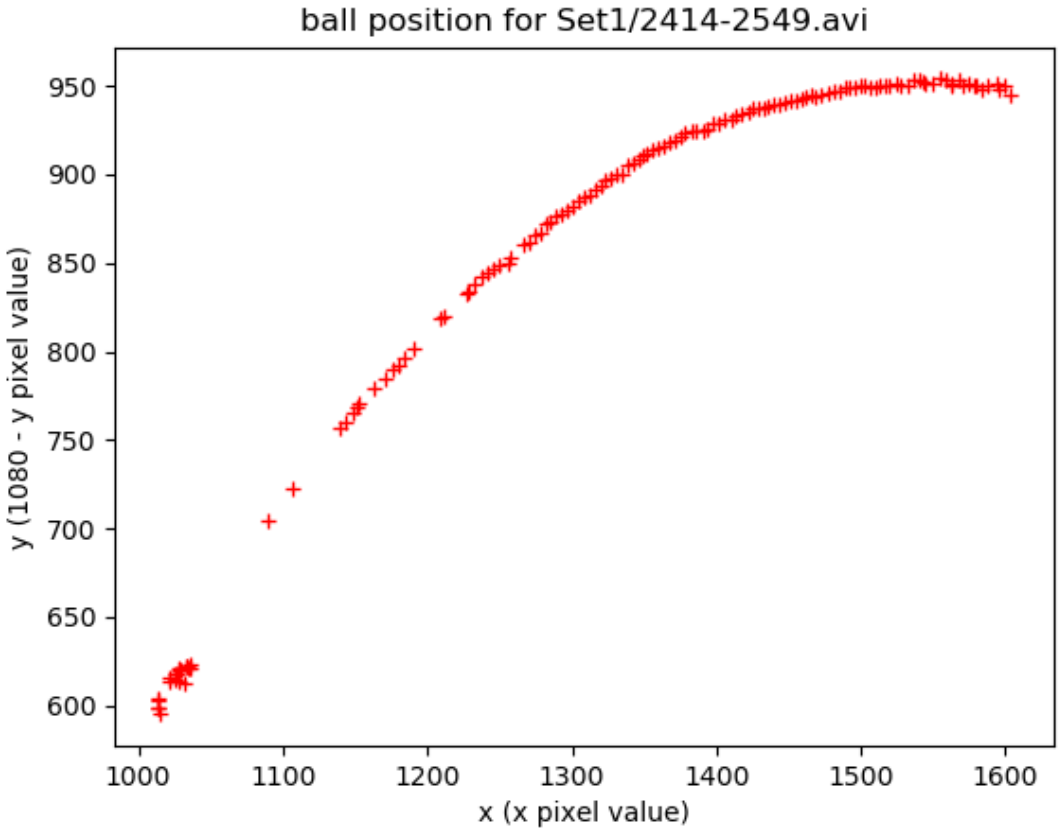
Results

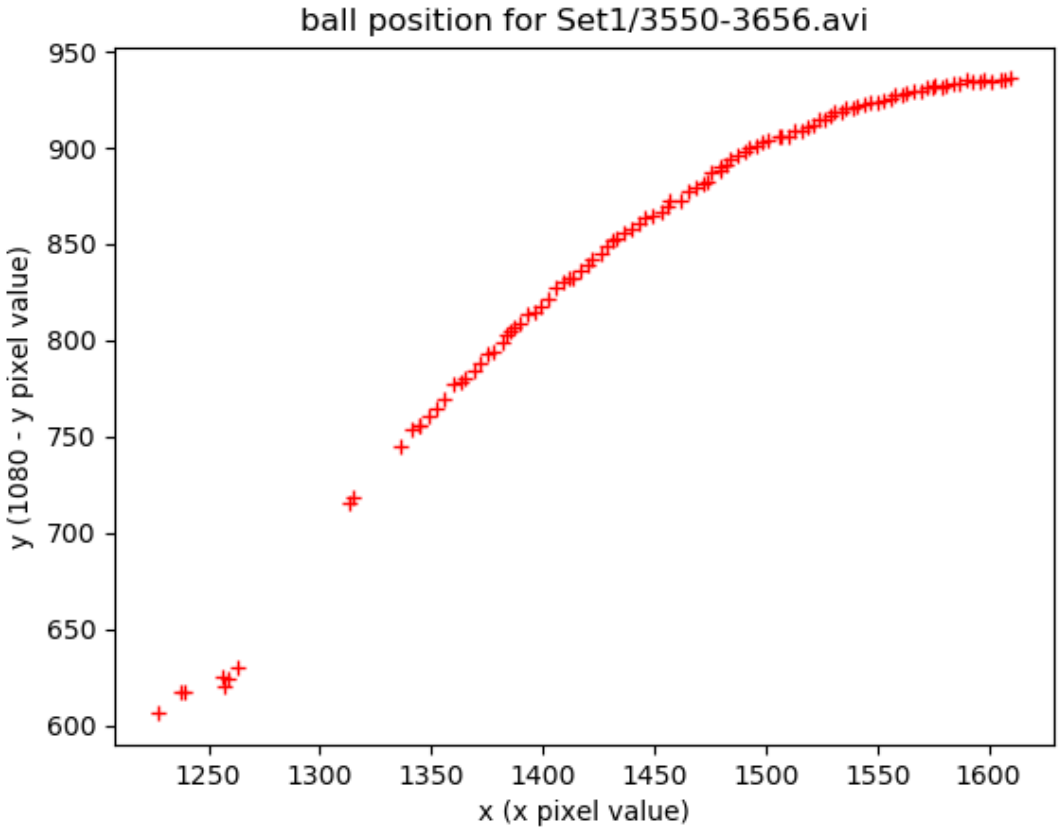
I tested my approach with the 20 videos in Set1 folder. I worked mostly fine, but in some frames, the program detected the arms of the thrower rather than detecting the ball. To reduce this error, I set a range for the found radius of the biggest contour, which is between 10 and 55 pixels for these set of videos. After setting this range, I plotted the x-y coordinates of the detected ball. Since the y-coordinates in frames increases from top to bottom, I subtracted the calculated y-coordinate from the frame height, which is 1080 in this case. I got the following results for the first 8 videos in Set1. The results for all 20 videos in Set1 folder can be found in GitHub.

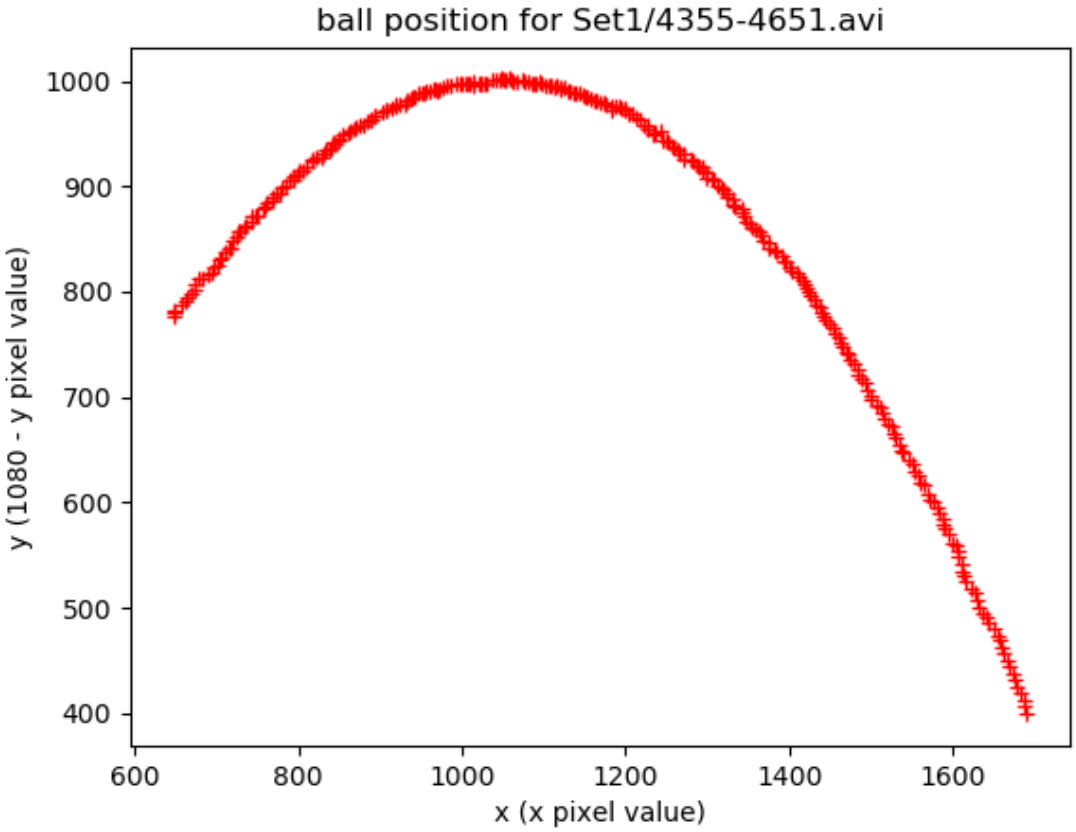

ball position for Set1/0016-0298.avi
900
800
700
600
500
400
300
y (1080 - y pixel value)
600
800
1000
1200
1400
x (x pixel value)

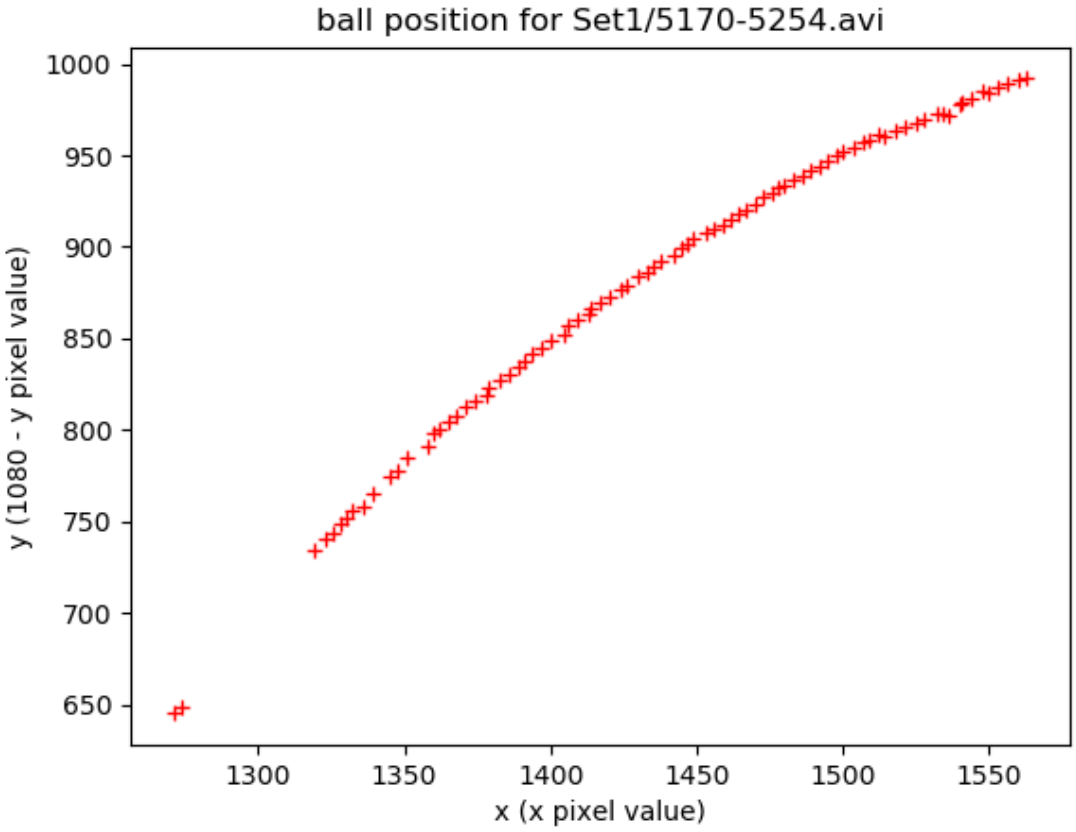

ball position for Set1/0568-0672.avi
900
850
800
750
700
650
600
y (1080 - y pixel value)
1000
1100
1200
1300
1400
1500
x (x pixel value)

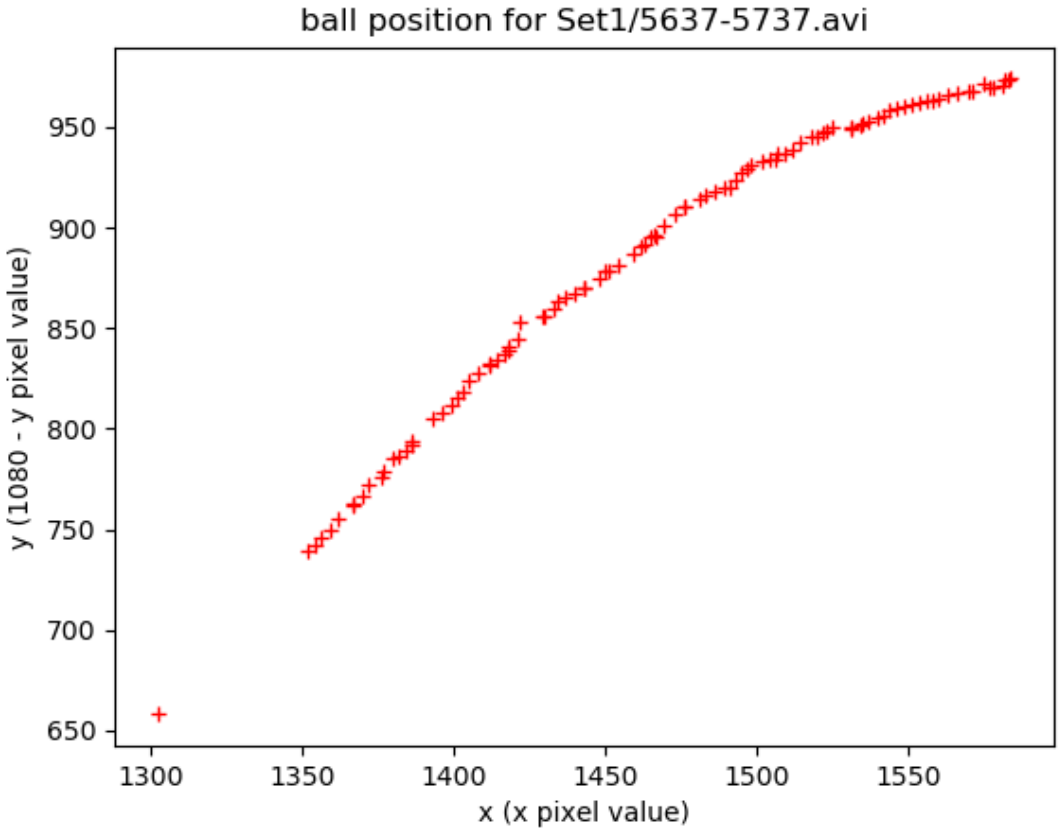

ball position for Set1/1489-1600.avi
y (1080 - y pixel value)
x (x pixel value)
1000
950
900
850
800
750
700
650
600
1200
1250
1300
1350
1400
1450
1500
1550
1600


ball position for Set1/2414-2549.avi
y (1080 - y pixel value)
x (x pixel value)
950
900
850
800
750
700
650
600
1000
1100
1200
1300
1400
1500
1600


ball position for Set1/3550-3656.avi
950
900
850
800
750
700
650
600
y (1080 - y pixel value)
1250
1300
1350
1400
1450
1500
1550
1600
x (x pixel value)


ball position for Set1/4355-4651.avi
1000
900
800
700
600
500
400
y (1080 - y pixel value)
600
800
1000
1200
1400
1600
x (x pixel value)


ball position for Set1/5170-5254.avi
y (1080 - y pixel value)
x (x pixel value)
1000
950
900
850
800
750
700
650
1300
1350
1400
1450
1500
1550


ball position for Set1/5637-5737.avi
y (1080 - y pixel value)
x (x pixel value)
950
900
850
800
750
700
650
1300
1350
1400
1450
1500
1550

As you can see from the figures above, the program got the main trajectory of the toss correctly. In some on the videos, there are some clusters between y=600 and y=650 range in the left side of the figures. This is mainly caused due to the false positive detection of the arms of the thrower. It may be possible to adjust the color range of the mask to reduce this error.

Since this approach doesn't require much calculation load for the computer, it can detect the ball quickly.

Code

vmg.py

```
import numpy as np
import cv2
import imutils


def detectBall(frame):
    # orange range for ball in HSV color space
    orangeLower = (2, 60, 60)
    orangeUpper = (10, 255, 255)

    # add gaussian blur and convert color to HSV
    gausBlur = cv2.GaussianBlur(frame, (11,11), 0)
    hsv = cv2.cvtColor(gausBlur, cv2.COLOR_BGR2HSV)

    # create mask according to color range
    mask = cv2.inRange(hsv, orangeLower, orangeUpper)
    mask = cv2.erode(mask, None, iterations=2)

    # find contour according to the created mask
    contours = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)
    center = None

    # check for found contour
    if len(contours) > 0:
```

```
            # calculate the center and radius of biggest contour
            c = max(contours, key=cv2.contourArea)
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            M = cv2.moments(c)
            center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

            # check radius range
            if (radius > 10) & (radius < 55):

               # draw the bounding box
               cv2.rectangle(frame, (int(x)-int(radius), int(y)-int(radius)),
            (int(x)+int(radius), int(y)+int(radius)),
                        (0, 0, 255), 2)

   cv2.imshow('frame (press "q" to quit)',frame)
   cv2.imshow('mask (press "q" to quit)', mask)
   return center
```

main.py

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import vmg

videoLoc = 'Set1/4355-4651.avi'

vid = cv2.VideoCapture(videoLoc)

posList = []
height, width, channels = [0,0,0]

while(vid.isOpened()):
   ret, frame = vid.read()
```

```
    if not(ret):
        break

    height, width, channels = frame.shape

    # press "q" to exit
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

    # save detected location of ball
    posList.append(vmg.detectBall(frame))



npPosList = np.array(posList)
xPosList = npPosList[:,0]
yPosList = npPosList[:,1]

vid.release()
cv2.destroyAllWindows()

# plot position of the ball
plt.figure()
plt.plot(xPosList, height-yPosList, 'r+')
plt.xlabel('x (x pixel value)')
plt.ylabel('y (' + str(height) + ' - y pixel value)')
plt.title('ball position for ' + videoLoc)
plt.show(block=True)
```