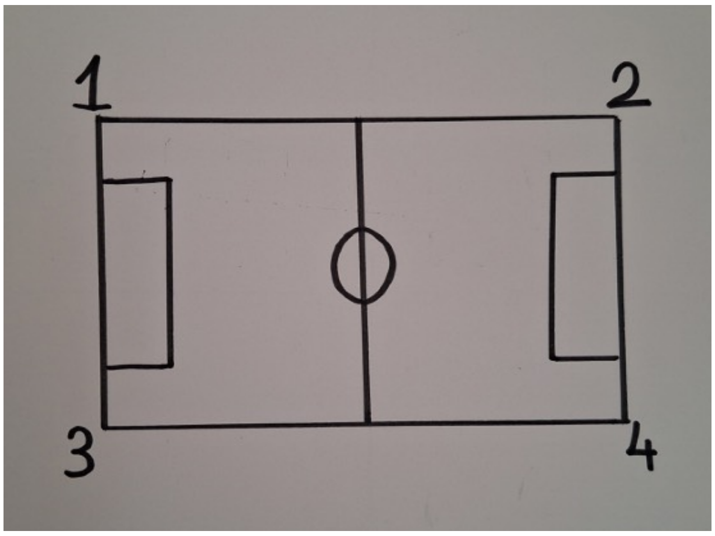
CSE 463 - Homework 1 Report

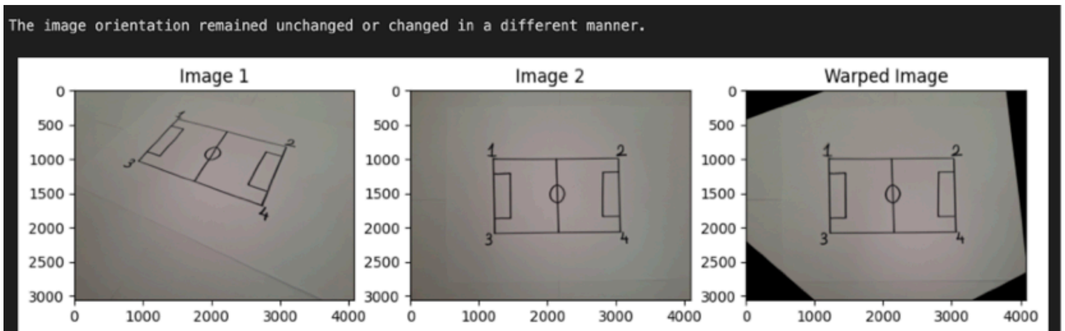
Sena Özbelen
1901042601
7 April 2024

Part 1

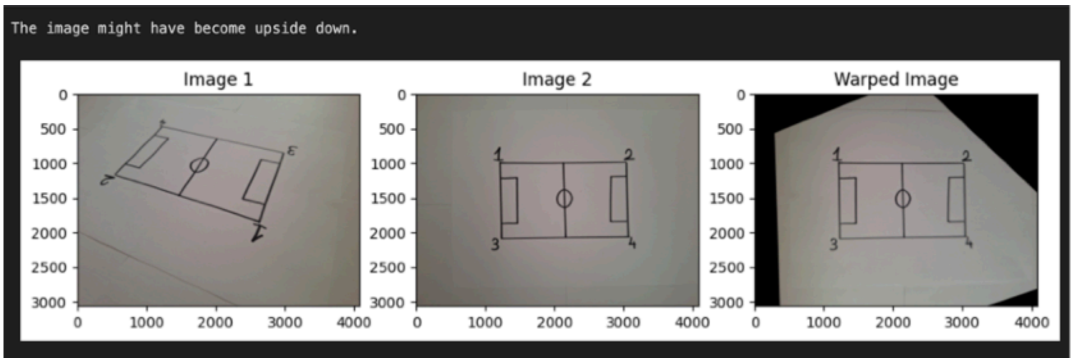
- **Calculate the homography:** I redrew the soccer field on a paper. To distinguish the corners, I put digits to each corner as seen in the picture below.



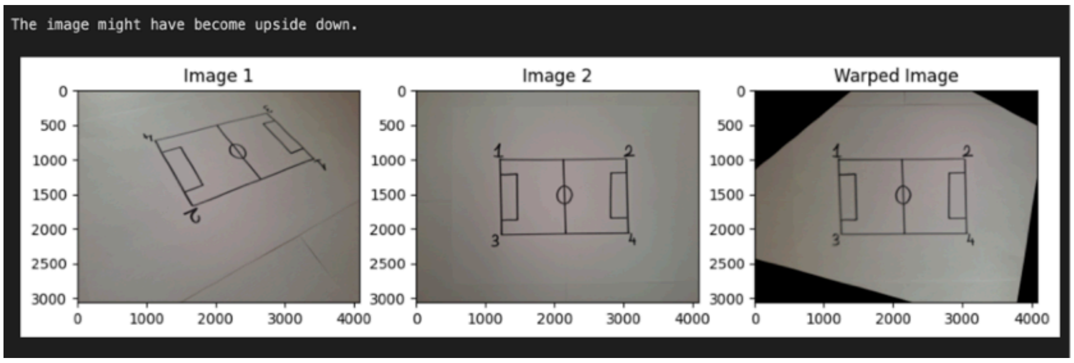
For homography part, I utilized the OpenCV tutorial "**Feature Matching + Homography to find Objects**". After loading the images, it basically uses SIFT to find the points which are used in homography calculations. Since we need points from both images and they should be matched, we need a matcher to find those pairs of points. This tutorial uses FlannBasedMatcher. After getting points, it calculates homography with findHomography function. Then, I added a part to check if the image is flipped. If it is flipped, then the circle position in the second part will differ. It basically transforms the corners of the image with the homography matrix and checks the corner positions (top-left and bottom-right). Then, it uses warpPerspective to get the final result of homography.



```
The image orientation remained unchanged or changed in a different manner.
```



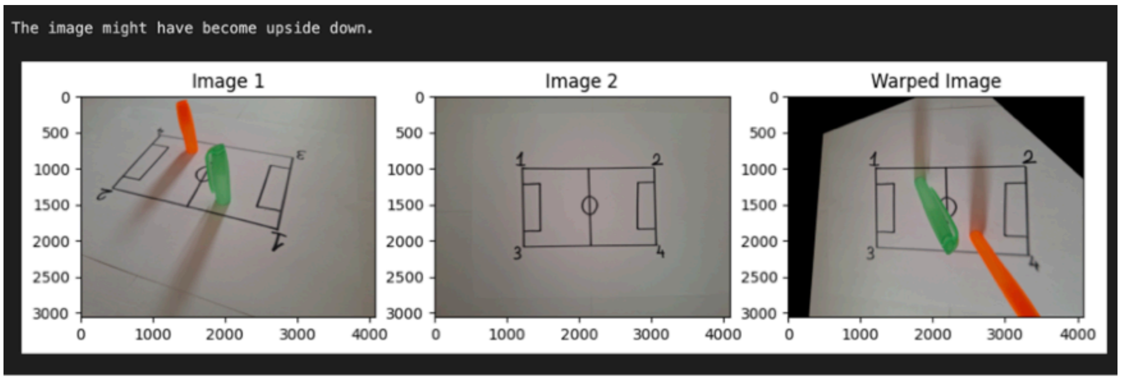
```
The image might have become upside down.
```



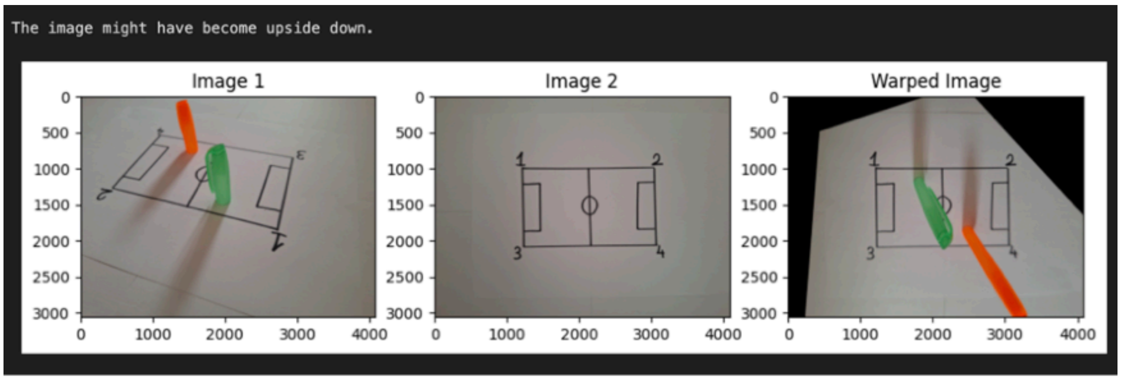
```
The image might have become upside down.
```

Part 2

- **Apply homography:** Apply the first part for the image with player. Sometimes, it might not produce the exact proper transformed image since its result depends on the matcher. The result might change in each run. If the corners are quite blurry as the one below, it confuses the matcher.

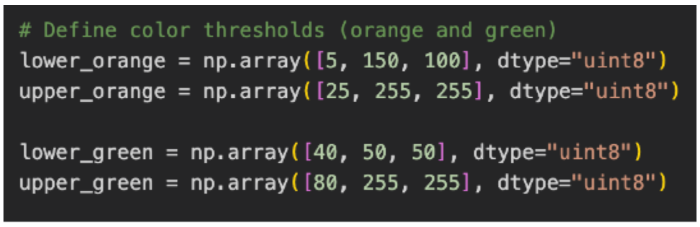


```
The image might have become upside down.
```



```
The image might have become upside down.
```

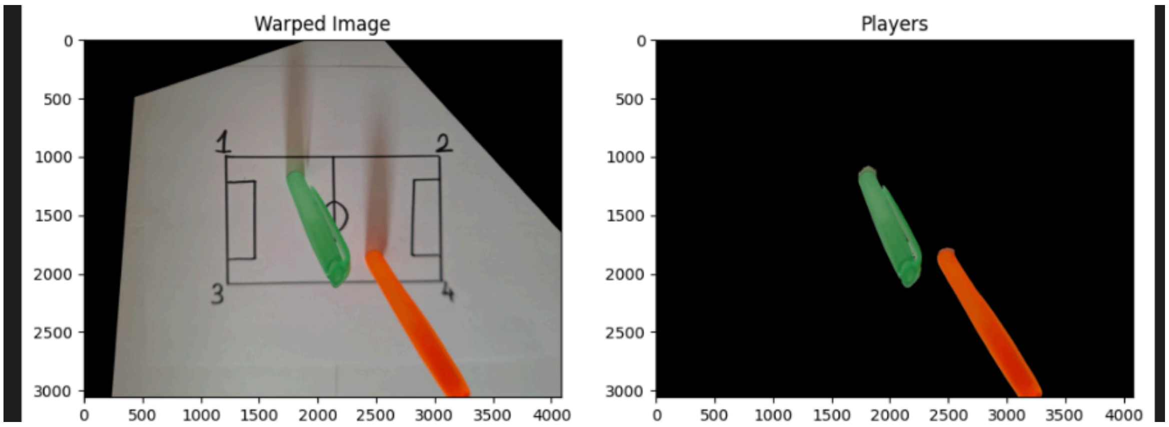
- **Extract the players:** It converts the image to HSV format to extract the players. The extraction is done by using colors. Since these players are orange and green, the values in the code are chosen for these colors.



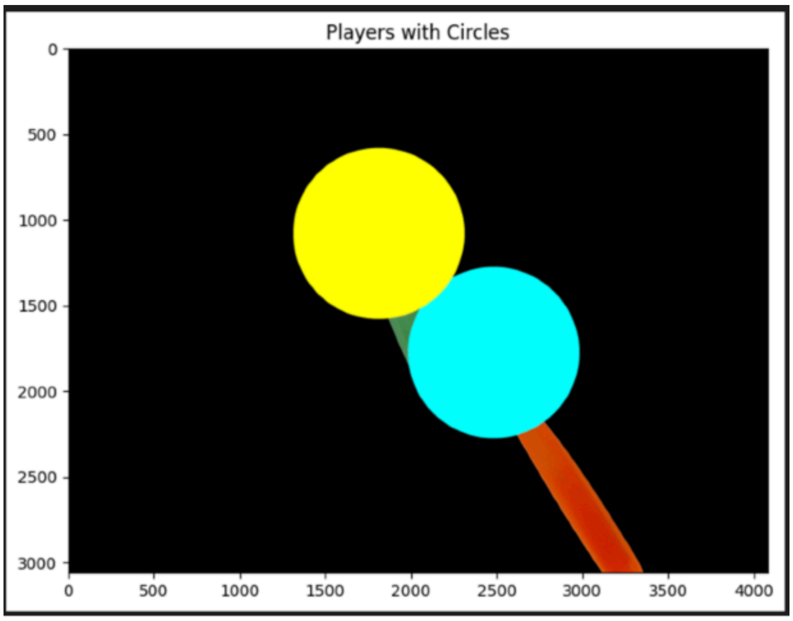
```
# Define color thresholds (orange and green)
lower_orange = np.array([5, 150, 100], dtype="uint8")
upper_orange = np.array([25, 255, 255], dtype="uint8")

lower_green = np.array([40, 50, 50], dtype="uint8")
upper_green = np.array([80, 255, 255], dtype="uint8")
```

Then, color masks are created with these values and they are combined by bitwise or. Once we apply this mask to the image, we will get the colored objects which are the players for our case.

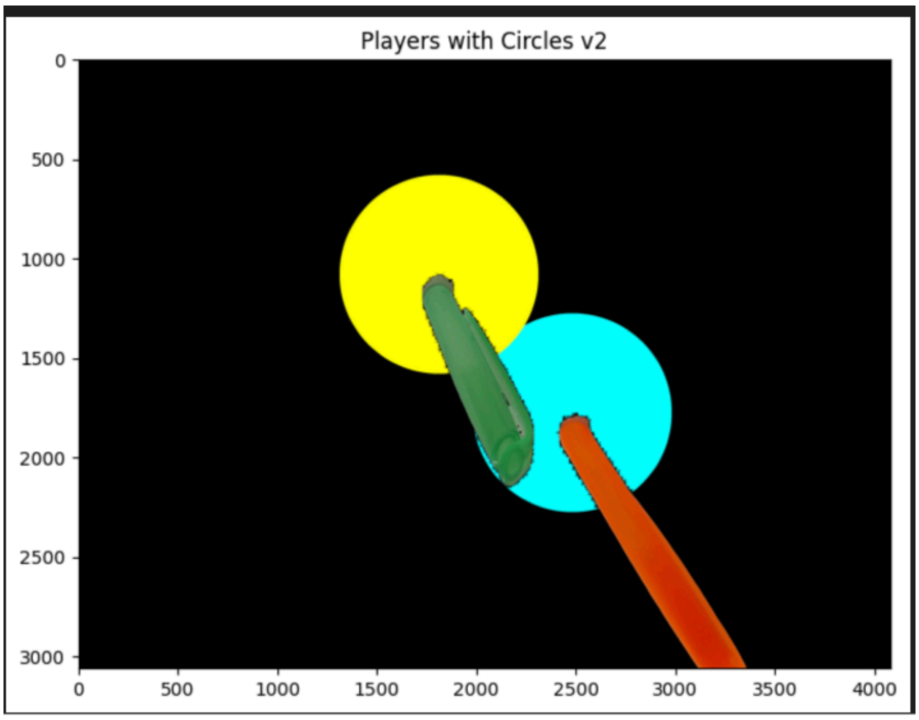


- **Put the circles under the players:** It uses threshold to convert the image to binary image so that we can find the contours of the image and this will help us find the players. We use flip information for this part. If it is not flipped during the homography transformation, we should find the

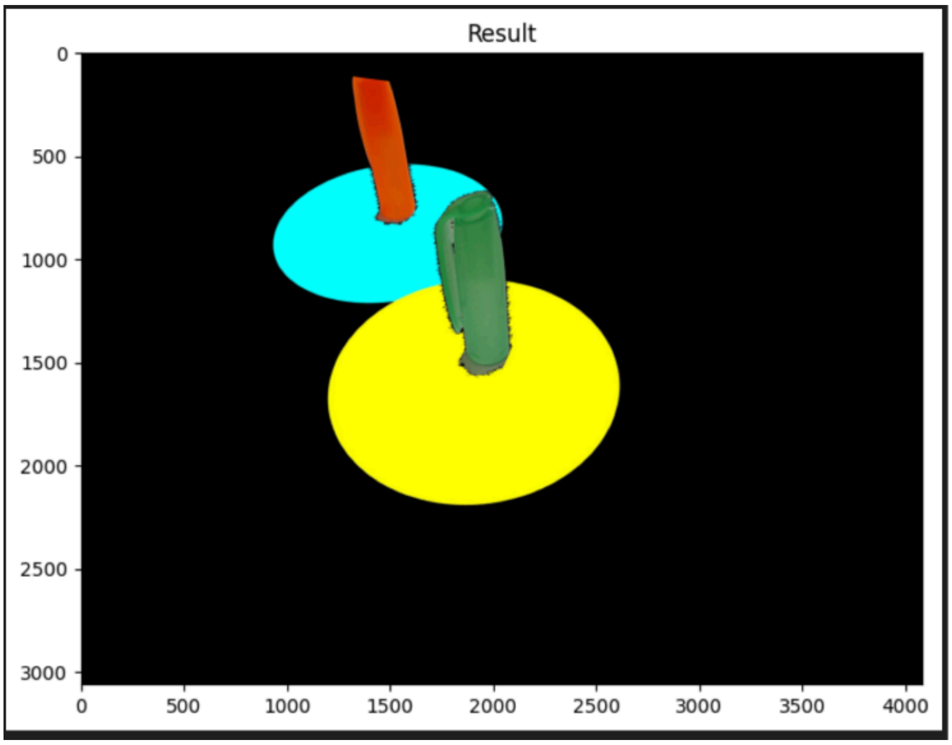


bottommost point of the player. If it is flipped, then we should find the topmost point. When we find these points, it puts a circle with a random color (they might have the same color since it is randomly selected).

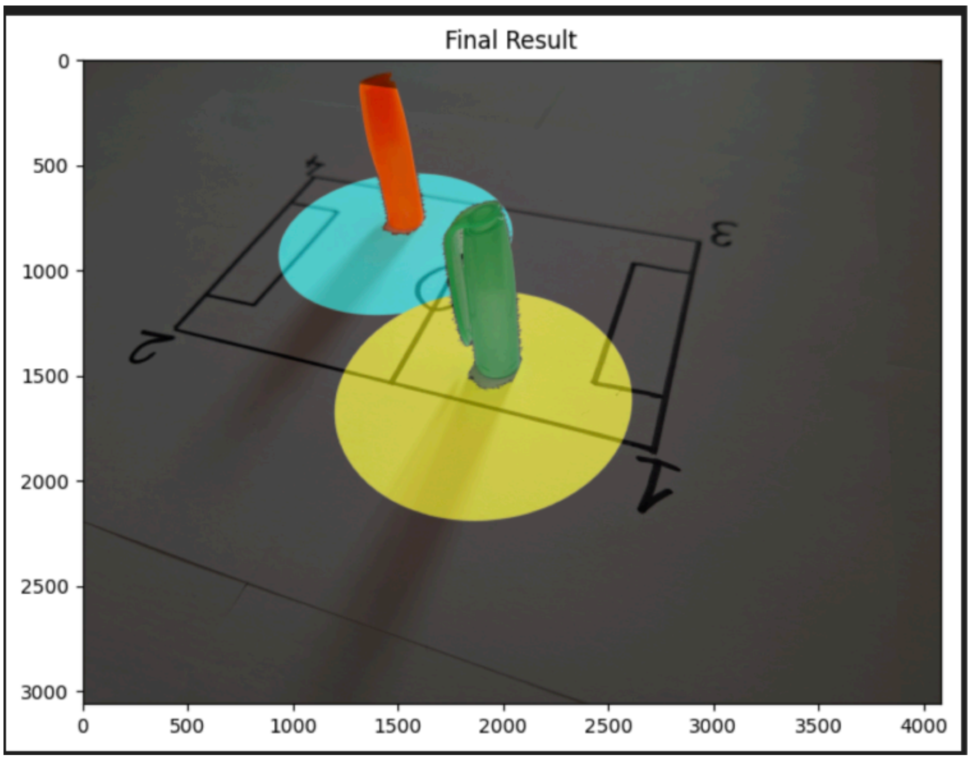
- **Remove the parts of the circles which should be behind the players:** There is a need to remove the parts which should be behind the players. Therefore, we need to extract the visible parts of circles and add to the players' image. To do that, it creates a mask with players. Then, it inverts it to keep the background which is the visible part of circles. When we apply this mask to the players with circles as a binary image, we get the visible part of circles as a mask. It applies this mask to the non-binary image of players with circles and we get the visible parts. When we add these parts to the players' image, we will get the circles behind the players.



- **Apply inverse homography to get the original perspective:** It computes the inverse of the homography matrix and applies it to the image of players with circles.



- **Add the soccer field to the background:** Lastly, it adds the players with circles to the original image.


Final Result