PART I

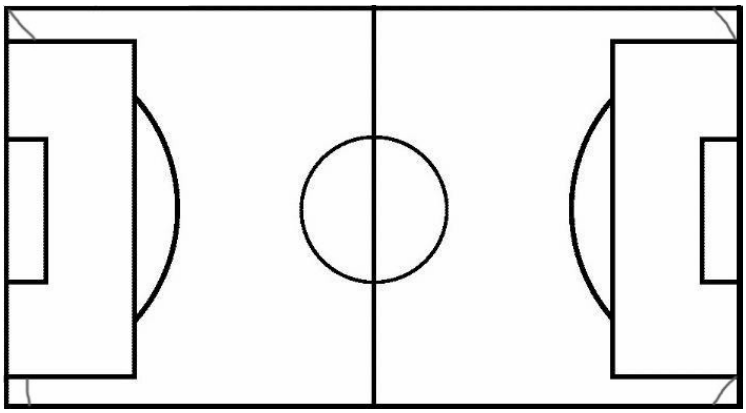
MODEL IMAGE OPERATIONS

For find tomography matrix, we should 4 point in model image and that take a picture of the soccer field with your cell phone. Lets start model points.

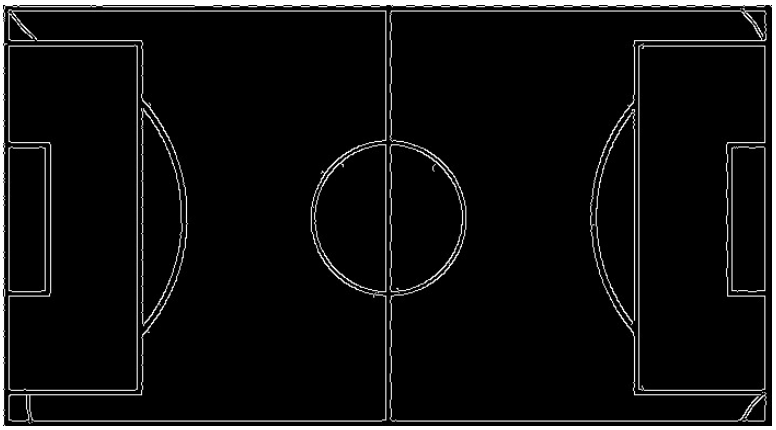
ROAD MAP

1. Read the model image,
2. Apply gray filter to image,
3. Detect canny edges,
4. Detect hough line from canny edges,
5. Detect parallel line intersections by k-means center,

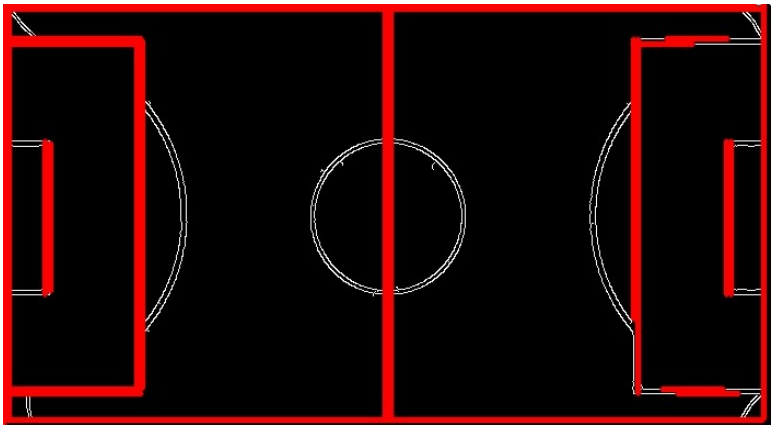
1. Read the model image,
2. Change color mode to gray scale,



3. Detect canny edges,



4. Detect hough line from canny edges,



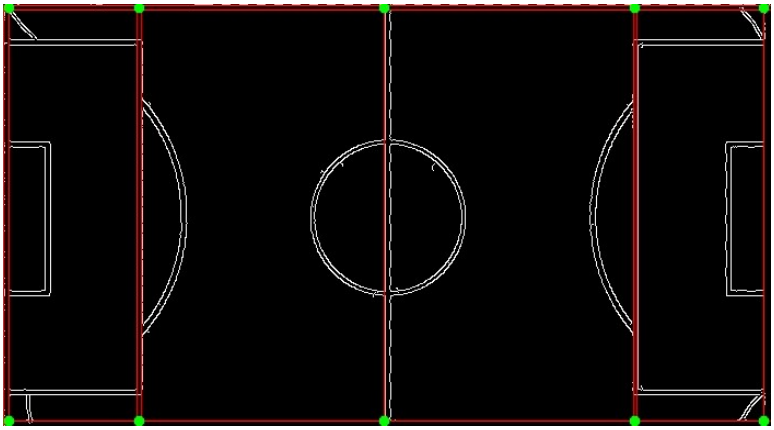
5. Detect parallel line intersections,

All intersections;

[[5, 5], [348, 5], [695, 5], [122, 5], [579, 5], [126, 5], [576, 5], [5, 379], [348, 379], [695, 379], [122, 379], [579, 379], [126, 379], [576, 379], [5, 1], [348, 1], [695, 1], [122, 1], [579, 1], [126, 1], [576, 1]]

Applies K-Means on the intersections;

[[124. 379.], [577.5 3.], [124. 3.], [577.5 379.], [348. 3.], [348. 379.], [5. 3.], [695. 3.], [5. 379.], [695. 379.]]



Select min/max points;

Points on top: [[***5, 3***], [***695, 3***]]

Points on bot: [[***5, 379***], [***695, 379***]]

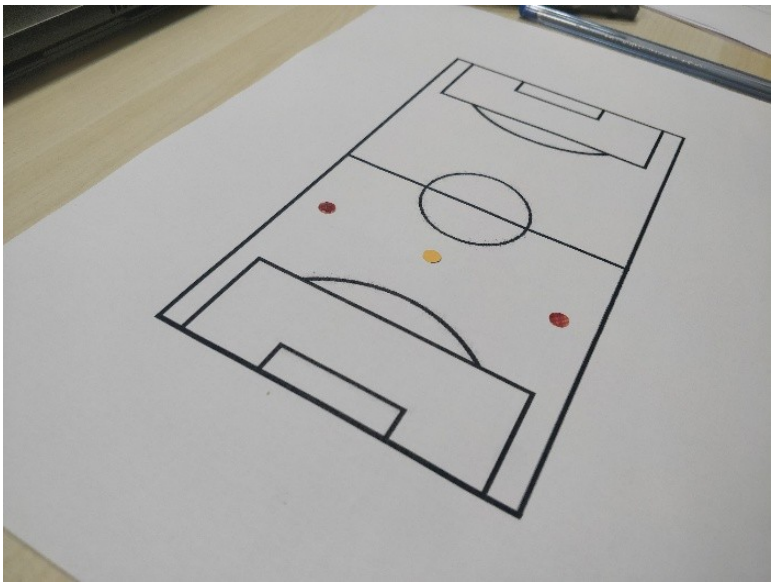
Now, Pictures on homework approximately 4000*2000px, so line height and pixels contained in a line are very huge. I did measured line pixels, that is about 25px which is different other images. But the model image's line is 4px. So we must resize input image about 700px width with aspect ratio. This point is important to understand all operations.

Up to now, i don't start input image operation. Nearly same operations we have to apply input image. I can describe a road map;

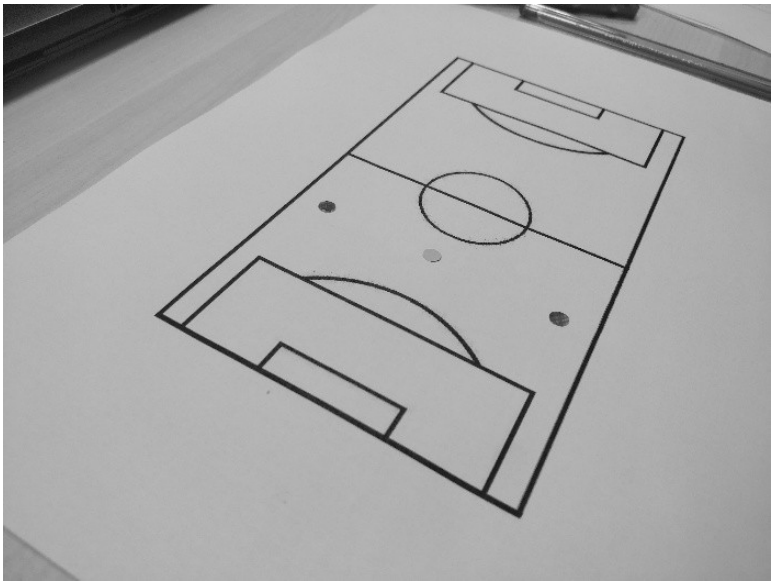
ROAD MAP

1. Read the input image,
2. Check size, resize image if needed
3. Apply gray filter to image,
4. Detect *canny* edges,
5. Detect *hough line* from canny edges,
6. Detect parallel line intersections by k-means center (get 4 corner points),
7. Estimate a homography image with using ***findHomography()*** function in open cv library,
8. Transform the input image with using ***warpPerspective()*** function in open cv library.

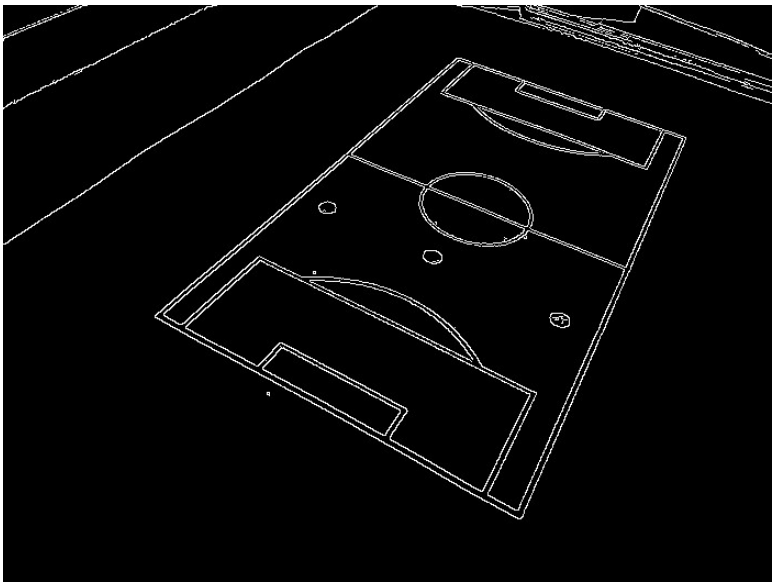
2. Resized image,



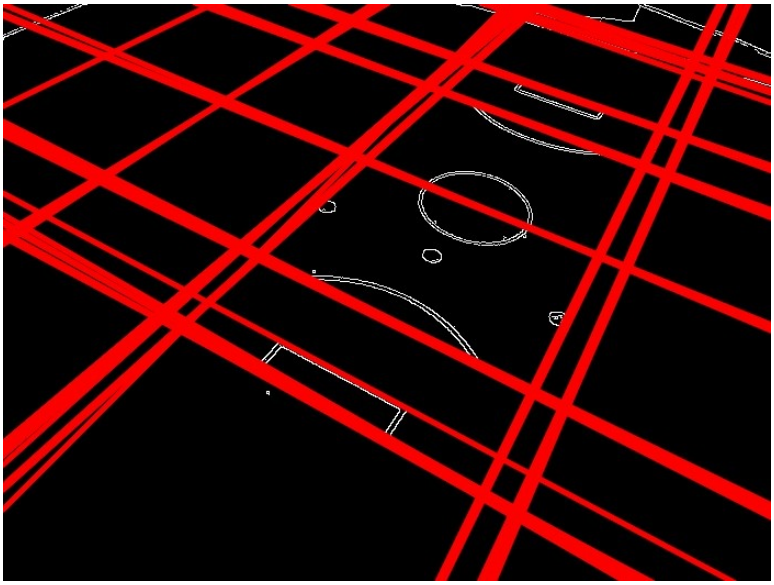
3. Gray filter,



4. Detect canny edges,



5. Detect hough line,



6. Detect parallel line intersections

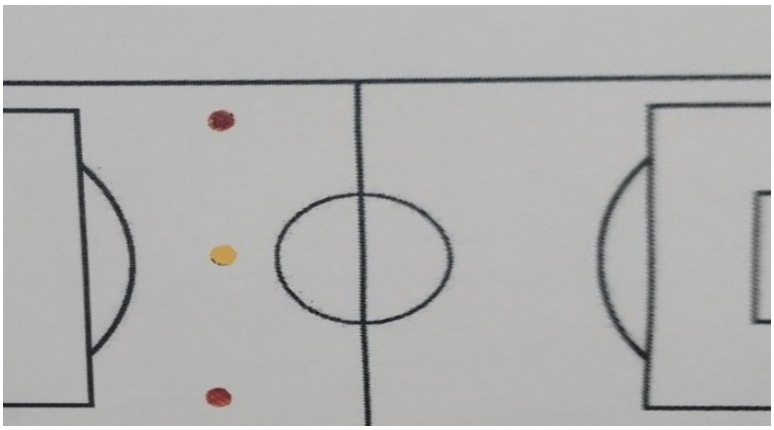
All;

[378.52147239 42.50920245], [-340547. 307048.], [186959. 67891.], [-283879. 256024.], [71161. 22938.], [-1520.5   -641.66326531], [-12606. -6780.], [12717. 3969.], [-3822. 10095.], [-19418. -11009.], [-2949.53333333 -1173.8   ], [5936. -4751.], [-4506.125 -1853.   ], [819.55833333 -316.75833333], [137.26896552 213.69655172], [-2932. 2860.], [5288. 2145.], [-8083. -3823.], [-1989.22916667 -830.33333333], [475.06944444 410.86111111], [-1203.15517241 -494.12068966], [610.27173913 120.83695652], [-568.11111111 -233.33333333], [-11251. -4157.]

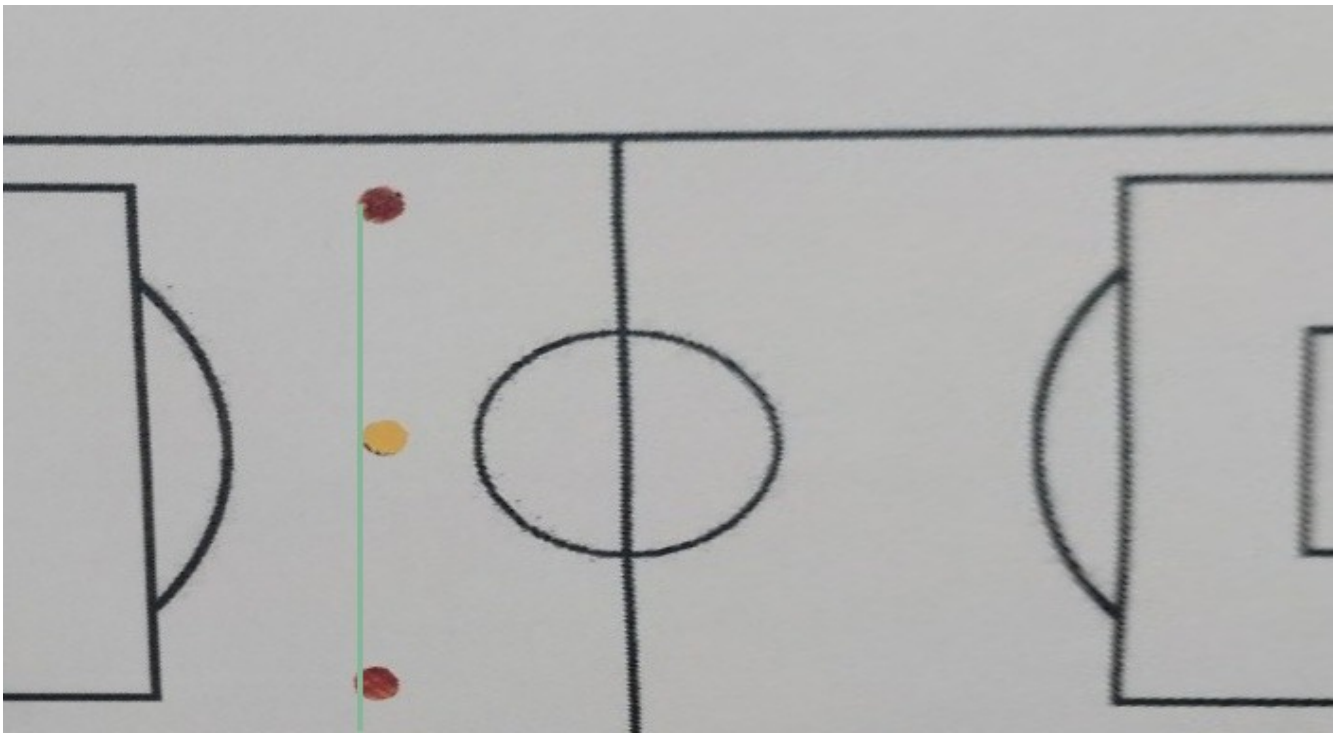
Top & Bottom Intersections;

Points on top: [***array([137, 213]), array([378, 42])***]
Points on bot: [***array([475, 410]), array([610, 120])***]

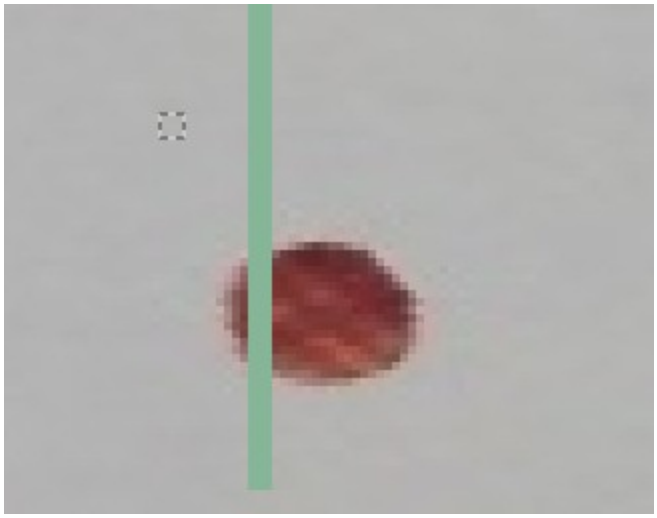
7. Estimate a homography image,



So we can detect some violation like offside,



The third player is a bit ahead than other two player, so that’s offside.



PART II

OPERATIONS

First off all, we should find colorful pixels. Lets start !

ROAD MAP

1. Find colorful pixels,
2. Some operations where describe above section,
    2.1. Apply gray filter,
    2.2. Detect canny edges,
    2.3. Detect *hough line* from canny edges,
    2.4. Detect parallel line intersections,
    2.5. Estimate a homography image,
3. Draw circle to colorful image’s starting place,
4. Get reverse homography,
5. Put colorful pixel old place.

1. Find colorful pixels,