Mustafa Cömert

151044028

CSE 463 Hw1 Report

Homograpy Matrix:

To find a good homography matrix I have used 3 methods one is built-in opencv function getPerspectiveTransform and I have written other 2 methods, first one creates 8x9 matrix than multiplies it with the 9x1 homography matrix result would be 9x1 zero matrix.

:
$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y'_1 & y_1y'_1 & y'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 & x'_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y'_2 & y_2y'_2 & y'_2 \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3x'_3 & y_3x'_3 & x'_3 \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3y'_3 & y_3y'_3 & y'_3 \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4x'_4 & y_4x'_4 & x'_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4y'_4 & y_4y'_4 & y'_4 \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = 0$$

To solve this equation I have used singular value decomposition, I didn’t wrote code for SVD instead I have used built-in numpy function linalg.svd.

In second one I have used a simple trick as taking advantage of homogenous coordinate system

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y'_1 & y_1y'_1 & y'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 & x'_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y'_2 & y_2y'_2 & y'_2 \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3x'_3 & y_3x'_3 & x'_3 \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3y'_3 & y_3y'_3 & y'_3 \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4x'_4 & y_4x'_4 & x'_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4y'_4 & y_4y'_4 & y'_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

And it turns out the simple linear system equation, since 9x9 matrix is invertible, we can come up with below equation,

$A*H = B \Rightarrow H = A^{-1} * B$

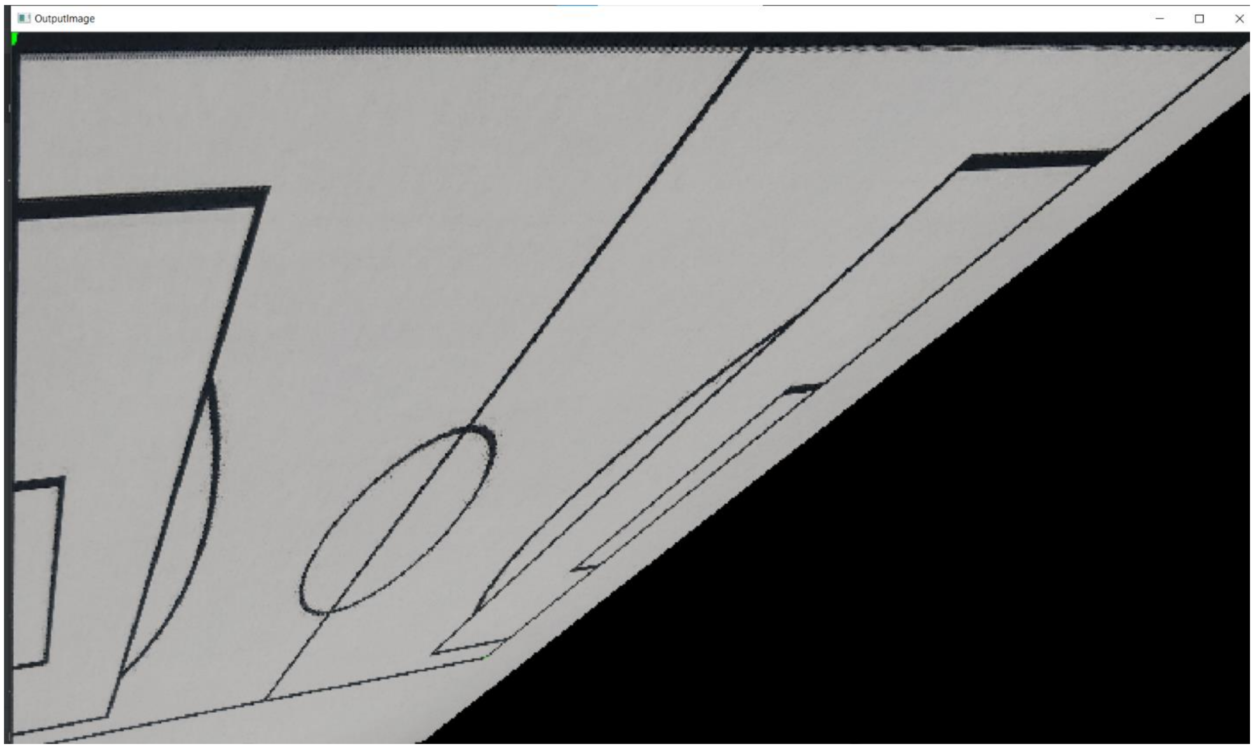
I didn’t write code for matrix multiplication and inversion instead I have used numpy.matmul and numpy.linalg.inv methods.

Point Selection:

Program show source image and expects for user selects four point by mouse clicks one by one. Then, automatically selects corner points of soccer field image as corresponding points.

Intersection Point of Lines:

I couldn’t work it out, when I used intersection of two side lines as a fourth point I can’t get good results. Below result show when I used intersection of two side lines as fourth point.

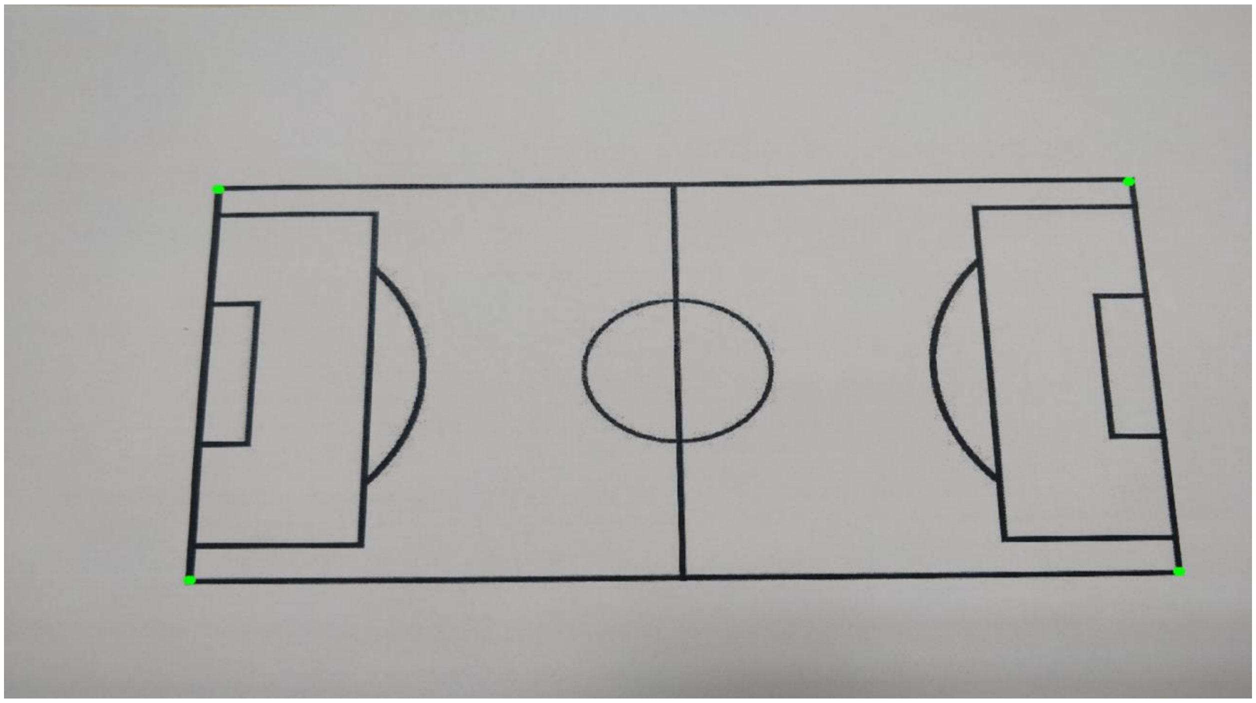


This doesn’t quite seem right, but I can’t figure it out.

Therefor I didn’t use intersection of two side lines as fourth point,

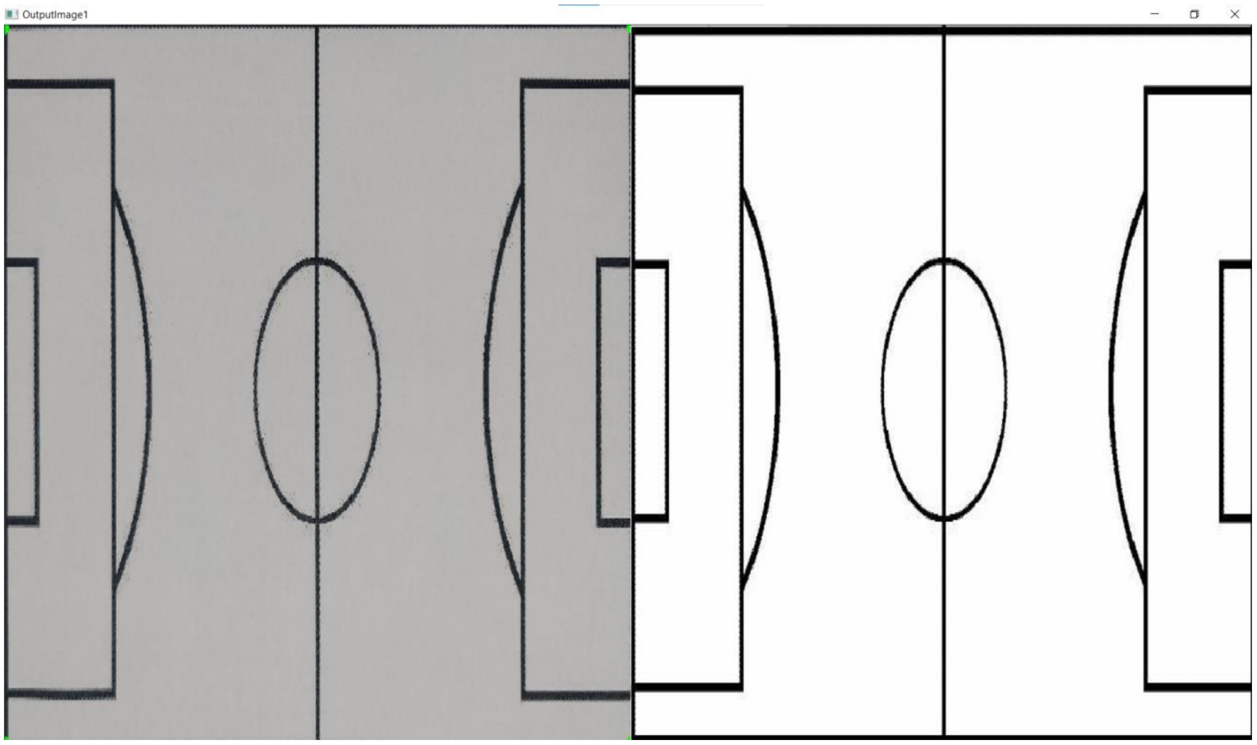
I have used user’s fourth point as a fourth point.

Source Image With Selected Points:



Green dots selected points by user

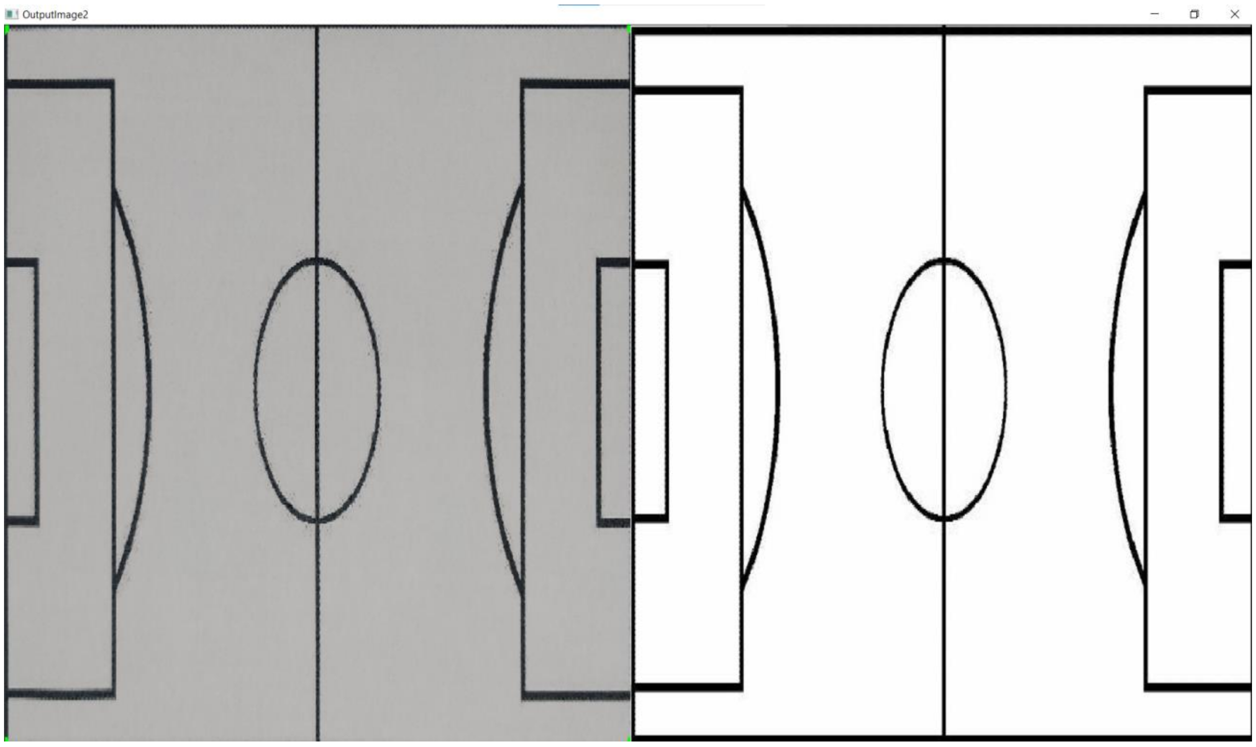
Output Of getPerspectiveTransform:



Homograpy matrix1

```
[[ 2.15099755e-01  1.12245878e-02 -1.80298493e+02]
 [-4.12382965e-03 -2.12907434e-01  6.03461437e+02]
 [-5.50244595e-06  4.72316150e-05  1.00000000e+00]]
```

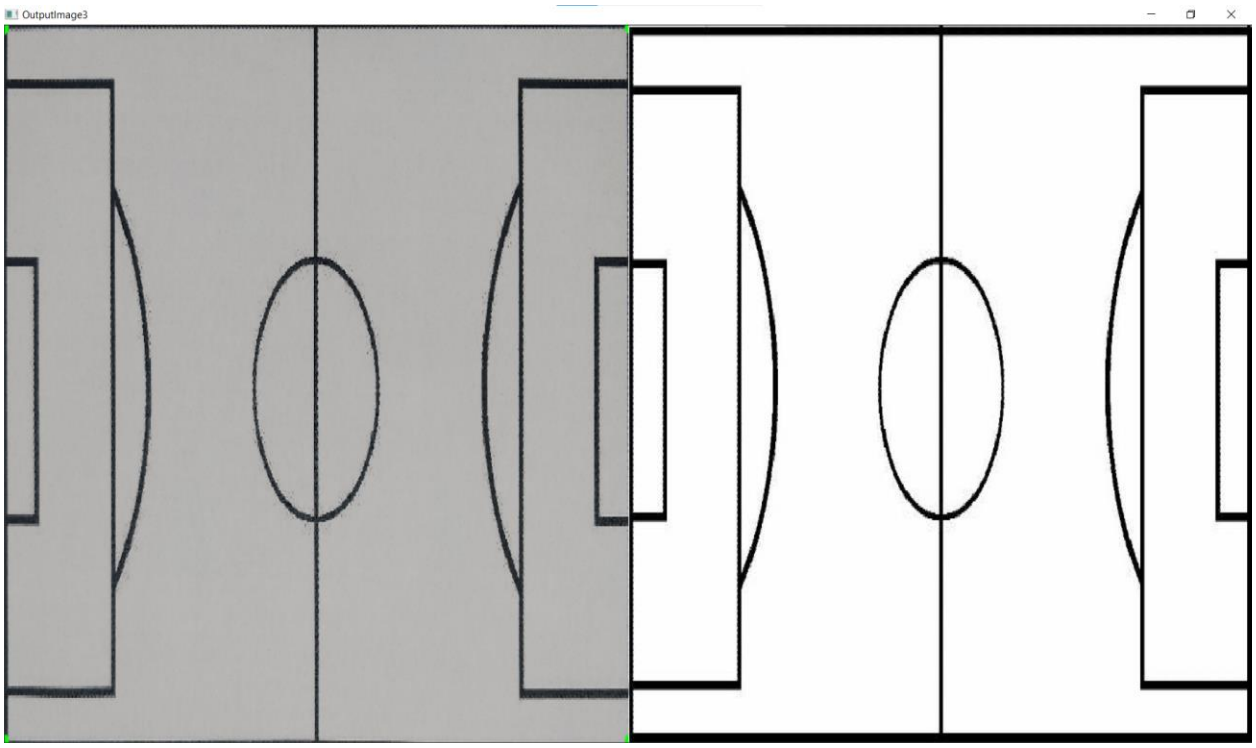
Output For my SVD solution(DLT):



Homography matrix2:

```
[[ 2.15099755e-01  1.12245878e-02 -1.80298493e+02]
 [-4.12382965e-03 -2.12907434e-01  6.03461437e+02]
 [-5.50244597e-06  4.72316150e-05  1.00000000e+00]]
```

Output of 9x9 matrix solution:



Homograpyh Matrix3:

```
[[ 2.15099755e-01  1.12245878e-02 -1.80298493e+02]
 [-4.12382965e-03 -2.12907434e-01  6.03461437e+02]
 [-5.50244595e-06  4.72316150e-05  1.00000000e+00]]
```

Conclusion: Homography matrix that is obtained by getPerspectiveTransform() opencv method(matrix1) and my 9x9 matrix solution(matrix3) all the same, on the other hand my SVD solution(matrix2) gives slightly different result than other two.