GTU

**DEPARTMENT OF COMPUTER ENGINEERING**

**CSE 463 – Spring 2022**

**HOMEWORK 1  
REPORT**

SÜLEYMAN GÖLBOL 1801042656

1. **PROBLEM SOLUTION APPROACH AND ALGORITHMS**

My first problem that I have encountered in application was OpenCV’s window problems. For example, OpenCV doesn’t close the window when user clicks into X button. To solve this problem I named window and used the function below.

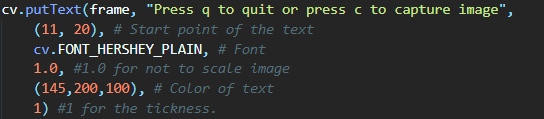


To get the 4 points from user I used if event == cv.EVENT\_LBUTTONDOWN: to get the mouse left clicks from user and in every click, I saved the position into a list and after 4 points, I sent these points to my Homography module.

To select button option between webcam or folder; I used tkinter module.



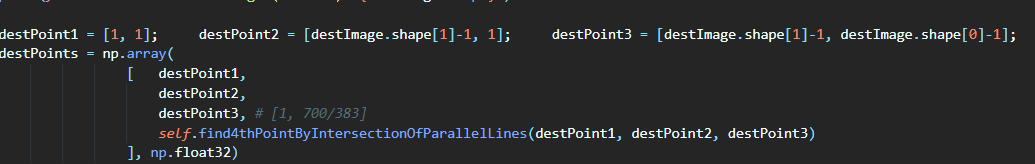
If user opens webcam to save image from it, in the left top of the screen, a text appears for the exit and capture keyboard shortcuts.



After user selects the image and save the 4 points of input image, the Homography class is being called.

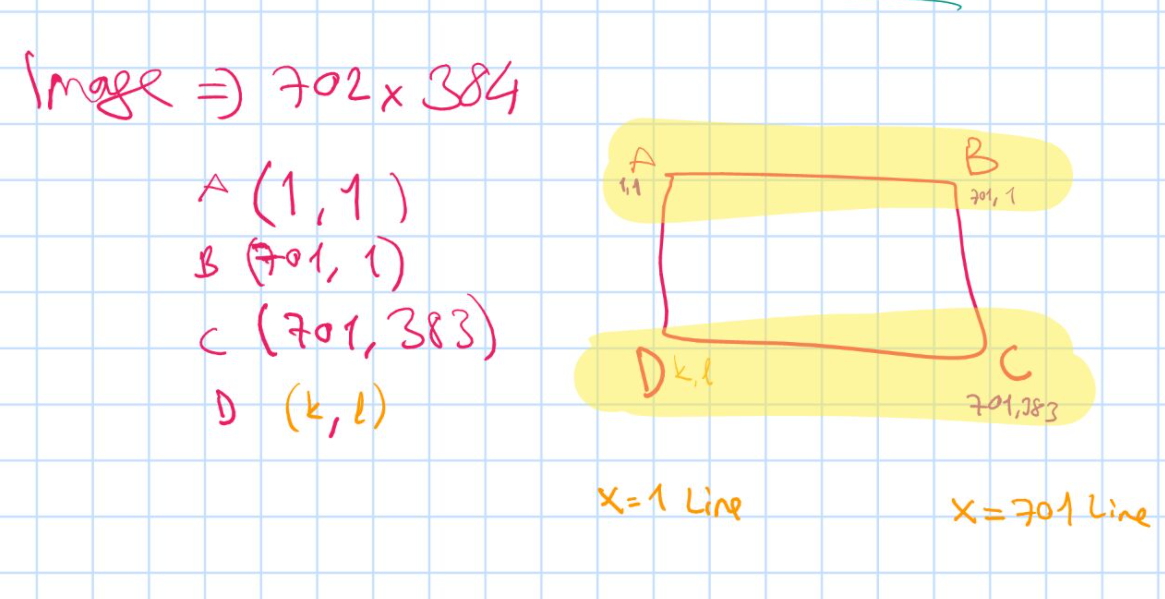
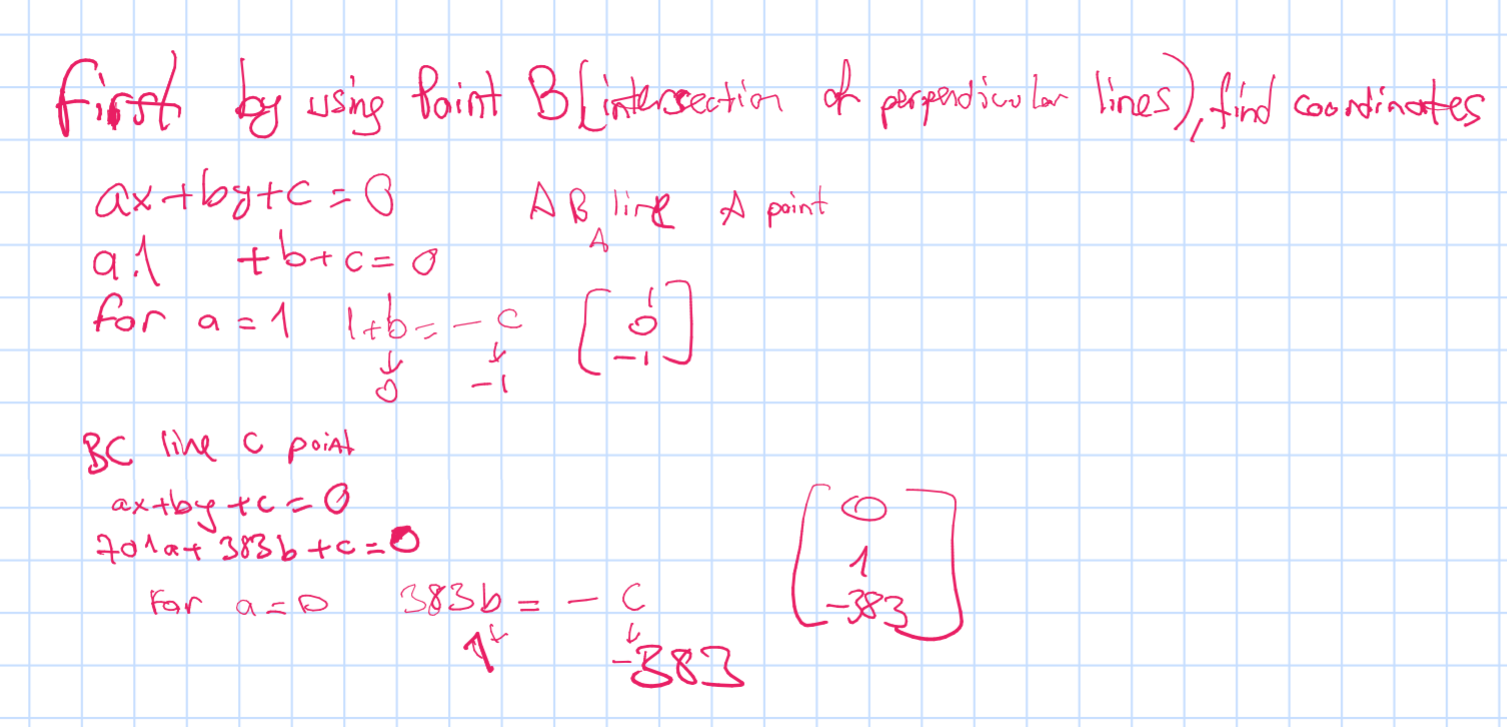
**I** **INTERSECTION OF PARALLEL LINES**

To find 4th point from the intersection of parallel lines in the soccer model field, I created a method called find4thPointByIntersectionOfParallelLines.

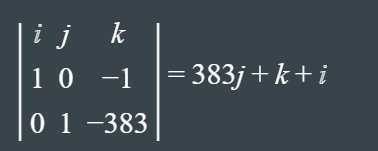


1) using P1 value from P1P2 line to get the first coordinate vector of the line  
2) using P3 value from P3P4 line to get the second coordinate vector of the line  
3) using the intersection of the two lines to get the 4th point [using determinant formula]

Example from our model soccer field:

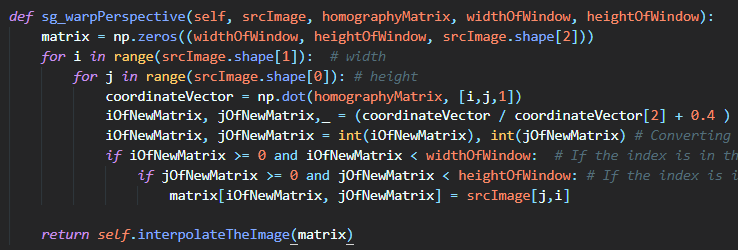
  


As you can see from our points in parallel lines, we obtained 2 vectors.  
To find the intersection point that’s on infinity, we need to use determinant. Coefficient of i is 1, and coefficient of j is 383 so I used this as 4th point of model field.

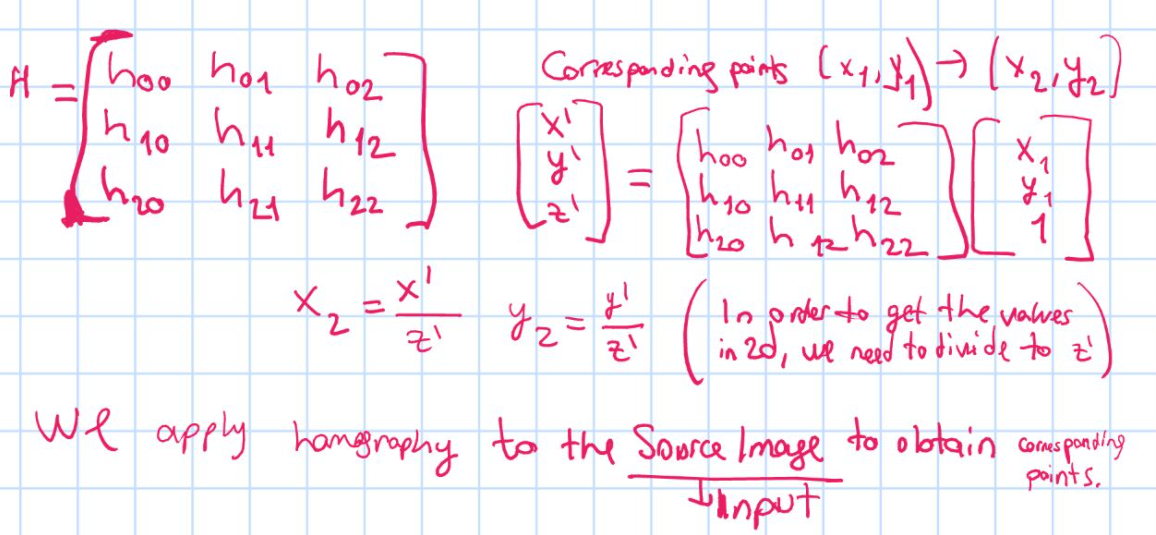


**My warpPerspective Implementation**

As it says in PDF, I didn’t use any img processing function from OpenCV expect in pdf’s. <https://docs.opencv.org/2.4/modules/imgproc/doc/imgproc.html>

And unfortunately warpPerspective is a image processing function. So I needed to implement it.  


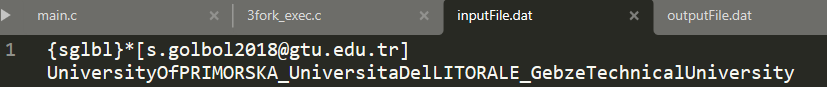
I took the source(input) image, the homography matrix, and the size of destination image.



As in the formula above, when we create a vector of [i,j,1] and we multiply with our homography matrix, we get some values, and if we divide these to z’, we get corresponding points.

And if the divide this value to n (10), we get the covariance matrix.  
  
 Covariance matrix: a‘ \* a / n

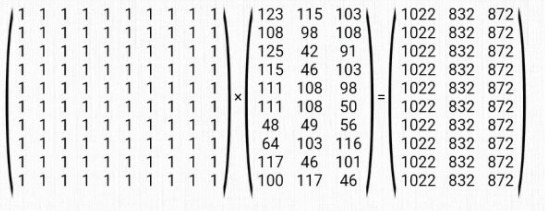
**Example** from my input file:



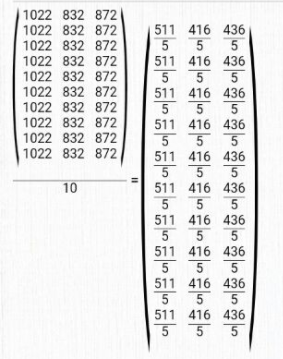
My input file starts with this line and when my program reads first 10 coordinates (30 bytes);  
It gives this output to stdout:

To calculate covariance matrix, first I needed to get the deviation score matrix.

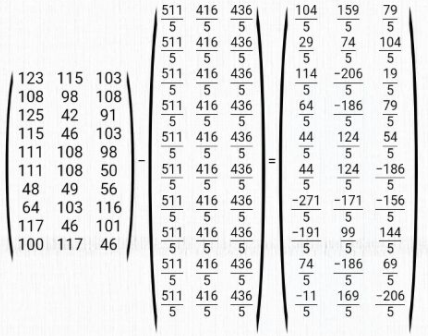
O \* A



O \* A \* ( 1 / n )

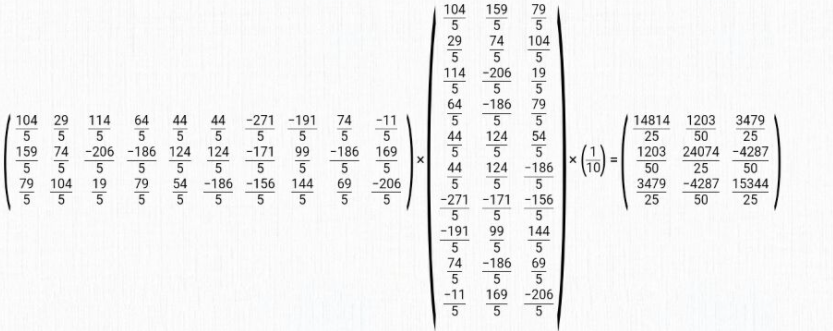


a = A – O \* A \* ( 1 / n )



Now if we multiply its transpose matrix with itself and then if we divide to the size we will get covariance matrix.

Covariance matrix: a‘ \* a / n



This value is my covariance matrix of first child.

After this, for the frobenius norm, it wasn’t difficult. I just summed the squares of output values and then get the square root of it.

**INTEGER/DOUBLE - STRING CONVERSION**

Another big problem that I was into was printing integer or double values using write(STDOUT\_FILENO, …) system call.

write() doesn’t have formatter like printf does. So it needs char\* as argument and itoa is not C standard function for every environment so I tried to create a int variable and send it’s address to function but it didn’t work while printing so I tried to cast to void\* and dereferenced it , also it didn’t work. So I wrote my own itoa function that works for nonnegative ascii integers.

Also I wrote my own double to string function for double value printing. I thought 2 parts of double(before dot, after dot) as different values then I converted them to string with putting “.” between these two.

**3 ) TESTS AND RESULTS**

# TEST SS VE JUPYTERR

# 

# GUI

# 

# 

# 

# 

# 