**Text

Description automatically generatedA picture containing qr code

Description automatically generatedGraphical user interface, application

Description automatically generatedA picture containing text, screenshot

Description automatically generatedGraphical user interface, text

Description automatically generated Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated**

**Question 02**

**Image processing Assignment II 190285J Prabhashwara Kahawalage**

**GitHub:** [**https://github.com/r3ki3g/Image-processing-fundamentals/tree/main/assignment-2**](https://github.com/r3ki3g/Image-processing-fundamentals/tree/main/assignment-2)

**(All the codes and full python notebook is in the GitHub/)**

Corner Cords of flag

Corner Cords of flag

However what I got was the affine transformation with [0 0 1] as the bottom row. Therefore the warping did not fit the wall.

Super imposing

Warping

Findingthehormography

**Experimented with finding the hormographic matrix myself**

Important points:

1. “Mouse click” and getting the coordinates was not possible because my notebook (Jupytor) froze every time I tried to do so. **So I had to hardcode the coordinates**.
2. First hardcoded the 4 corner coordinates of the flag then the 4 corner coordinates of the spot we need to super impose.
3. Then calculated the matrix using the cv.findHormography method and warped the flag as needed. Finally super imposed with fine weights using cv.addWeighted.

**Text

Description automatically generatedA picture containing text

Description automatically generated**

**Question 03**

It was impossible to find matches between 1.ppm and 5.ppm directly. So idea was to find the hormography between 1.ppm and 2.ppm then 2.ppm and 3.ppm , and like so finally 4.ppm and 5.ppm. Then we can pre-multiply the hormographies and get the composition of hormography which is equivalent to hormography between 1.ppm and 5.ppm

**siftMatchFinder() function is defined to calculates sift matches between images. (It is used multiple times later in the code)**

**Text

Description automatically generated**

Creating the data matrix to find the least error transform with RANSAC algorithm. Using matrix and Numpy library helps do the calculations fast and ends up letting us do much more iterations for the RANSAC.

**Text

Description automatically generatedText

Description automatically generated**

getDistanceBtwMat() function calculated the geometric distance between given vectors. It is used to see how the destination points are deviated from the requires positions in the image.

By selecting the case of minimal eigen values, in each iteration we can calculate the hormography matrix . The result is shown here:

**A picture containing text

Description automatically generatedText

Description automatically generated with medium confidence**

**Final stitched image and warped version are shown here**

Text, letter

Description automatically generated

**Question 01**

Circle eq. from 3 points, rough work

RANSAC algo for circle detection

A piece of paper with writing on it

Description automatically generated with medium confidenceText

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Chart

Description automatically generated

Calculating inlier count for each random sample of size 3, then finding the maximum case as the solution

Graphical user interface, text, application, email

Description automatically generated

Used Best circle finding code is from

<https://scipy-cookbook.readthedocs.io/items/Least_Squares_Circle.html#Using-scipy.optimize.leastsq>

Best circle finder with scipy.optimize

Important points:

1. Instead of looping through a random index set, here I am using three random numpy vector s of indexes to select 3 points randomly for RANSAC. All the calculations are done as vectors to make the process faster.
2. 10,000 samples are taken for the RANSAC
3. Both RANSAC and best fit (least square) are almost overlapping **for threshold of 2.** Here shown is a most deviated instance I got.
4. Mistakenly did the line finding part (not asked in the assignment) and it Is in the python notebook.(GITHUB)

Results