**EN3160 - Image Processing and Machine Vision**

**Assignment 1 - Intensity Transformations and Neighborhood Filtering**

**Name :** A.A.H. Pramuditha

**Index No. :** 200476P

**GitHub Link :** [hashirupramuditha/EN3160-Image-Processing-and-Machine-Vision (github.com)](https://github.com/hashirupramuditha/EN3160-Image-Processing-and-Machine-Vision)

Question 1: Detect and Draw Circles in an Image using Blob Detection

A computer screen with colorful text

Description automatically generated

**Range of 2 - 40 sigma values are used.**

**The maximum estimated standard deviation: 50.91168824543143**

**A field of sunflowers with blue circles

Description automatically generated**

Here, blob\_log() function is used to apply Laplacian of Gaussian Blob detection method to find blobs in the selected image by setting the standard deviation of the gaussian kernel () in between 2 and 40. Also, the blobs with intensities below the threshold (0.1) are ignored.

Question 2: Line & Circle Estimation Using RANSAC Algorithm

The output results from the code can be viewed as follows:

**-----------------------------Implement Line Estimation--------------------------------**

**Iteration: 3**

**Line best error: 7.495263964720616**

**Best line indices: [97 55]**

**Best model for the line: [0.69596607 0.71807469 1.37351009]**

**Total number of inliers: 44**

**Iteration: 7**

**Line best error: 6.141079915399064**

**Best line indices: [94 50]**

**Best model for the line: [0.70248673 0.71169685 1.49143588]**

**Total number of inliers: 41**

**------------------------------------Implement Circle Estimation-----------------------**

**Number of inliers: 41**

**Number of remnants: 59**

**Iteration: 0**

**Circle best error: 8.757072175486158**

**Best circle indices: [33 21 44]**

**Best model for circle: [ 2.40389922 2.72387709 10.16752722]**

**Total number of inliers: 40**

**Iteration: 4**

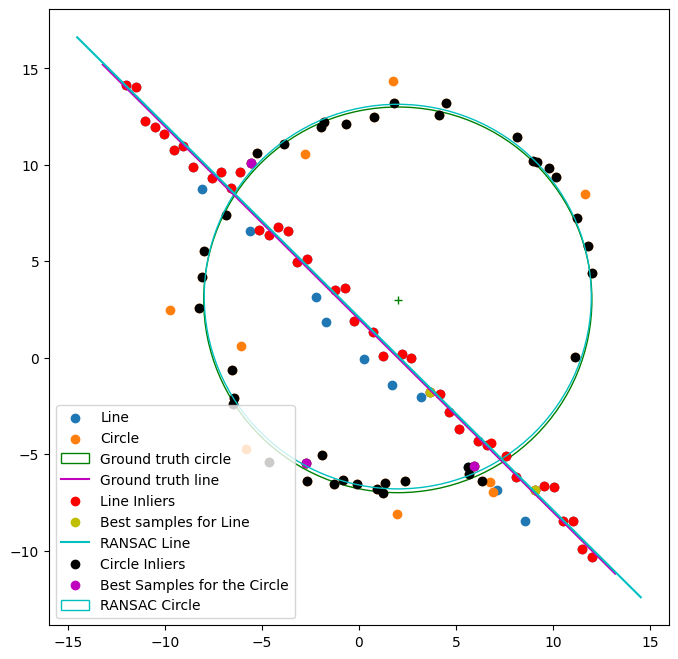
**...**

**Circle best error: 5.3997389308266435**

**Best circle indices: [24 29 40]**

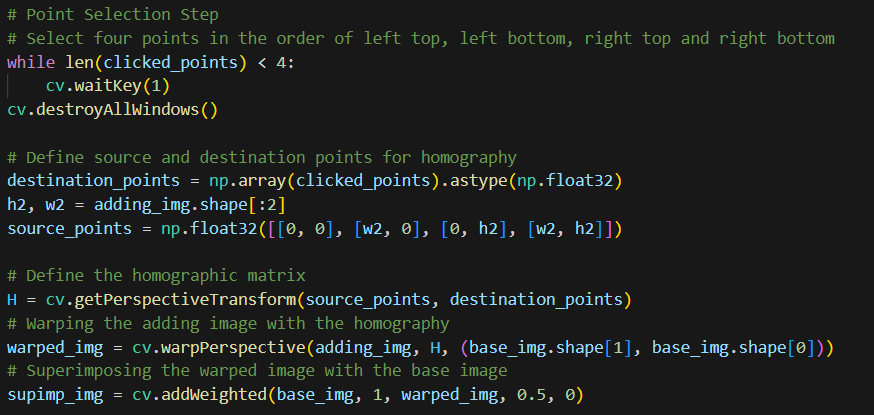
**Best model for circle: [1.99807947 3.16815845 9.97198408]**

**Total number of inliers: 41**

****

2.3) If we fit the circle first, the algorithm will create a model that fits the circular points well. But it won't be able to capture the remnants for line model effectively. So, when we try to obtain line parameters that need to account for these remnants, it would be difficult. So, in RANSAC-based models, it is better to fit much simpler models (like lines) first, and then handle the remaining points (remnants) with more complex models.

Question 3: Superimposing a Flag with an Architectural Image



**Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).**

**Image shape: (768, 1024, 3)**

**Flag shape: (150, 225, 3)**

**Selected Points: [(280, 242), (270, 449), (501, 292), (495, 461)]**

**Homographic Matrix:**

**[[-5.85557918e-02 2.22712812e+00 2.80000000e+02]**

**[ 8.96533352e-01 7.72670814e-01 2.42000000e+02]**

**[-5.22642494e-05 1.50458041e-03 1.00000000e+00]]**

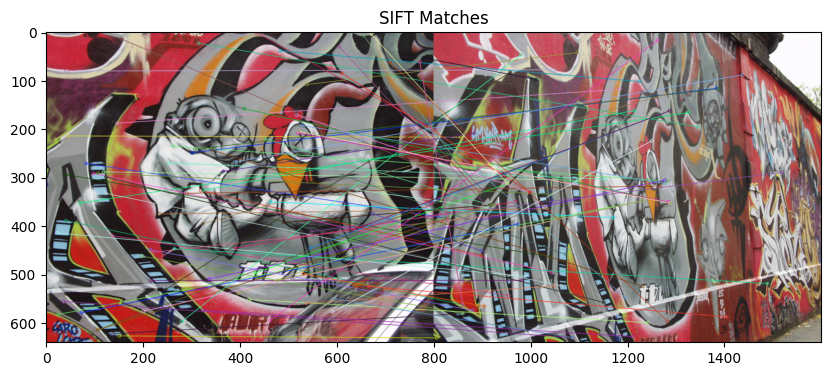
A building with a flag on the side

Description automatically generated

In this section, for the superimposing purpose, I chose a building with a much larger surface area and a flag to superimpose with. First, four points on the surface of the building are selected. Then the transformation is applied. In the transformation, a homography is calculated by using cv.getPerspectiveTransform() function. Then for the warping purpose, cv.warpPerspective() is used with the above homography. At last, the building image is blended with the flag image for the selected location with enough bias.

Question 4: Stitching Two Graffiti Images

4.1) SIFT (Scale Invariant Feature Transformation)



4.3) Stitching two images and observe the difference.

A wall with graffiti on it

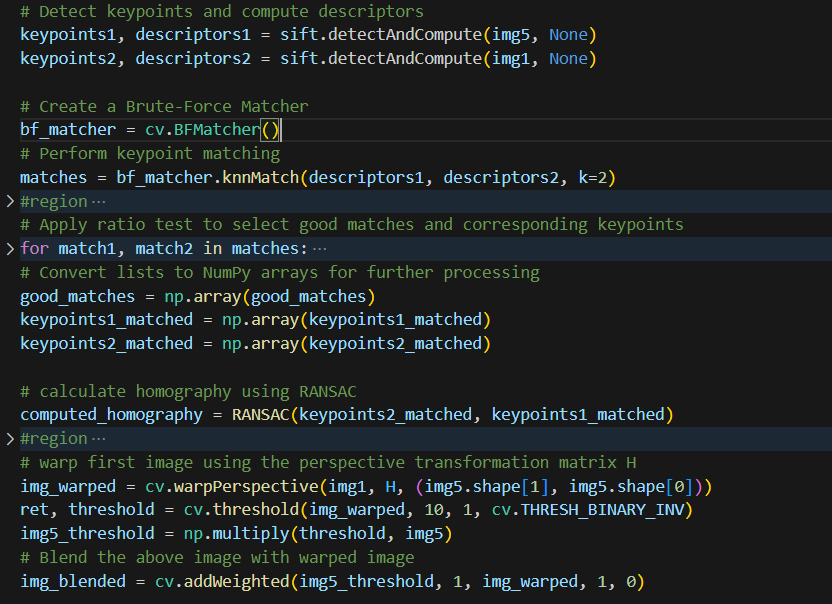
Description automatically generated

**Computed Homography Matrix:**

**[[ 6.12465611e-01 1.99327915e-01 2.11475491e+02]**

**[ 3.88459007e-01 1.25350118e+00 -3.94509825e+01]**

**[ 2.31859193e-04 3.96481577e-04 1.00000000e+00]]**

In section 1, the SIFT algorithm is used to detect distinctive key points of an image that’re robust to rotation, scaling, and affine transformations. Using that, several best key points were identified and matched using their descriptors.

In section 2 & 3, a homography matrix is calculated and perspective transformation is applied to stitch two images with overlapping fields to create a photo or a panorama with a higher resolution.