ECE-GY 6123 Image and Video Processing, Spring 2021
Programming Assignment 5: Feature Detection and Image Stitching
Due: 12/06/2023 11:59PM

General Guideline:

This assignment is intended to help you better understand the concept of a) Harris feature detection; b) SIFT Feature Descriptor; c) Feature Matching and d) Panorama Stitching.

You could use either Python or Jupyter Notebook to complete the assignment. You should refer to the Jupyter TEMPLATE provided for more specific tasks assigned.

If you use Jupyter Notebook, a template is provided for you and you need to complete the TO DO part. You should add your comment/observations within the Notebook. You should submit the following as two separate attachments:

1. A pdf version of the notebbok (that includes the results/figures and comments you have for any results)

2. A zip file including all following files named **your-last-name-CA05**

- The python notebook as a *.ipynb* file with the name as **your-last-name-CA05**
- The pdf version of the notebbok
- Add all the saved images in a folder named *output-images*

If you use Python, save all the figures/plots needed to be displayed, and you should submit the following

1. A report in .pdf that includes the results/figures and comments you have for any results. Also, please add instructions on how to run your python code.

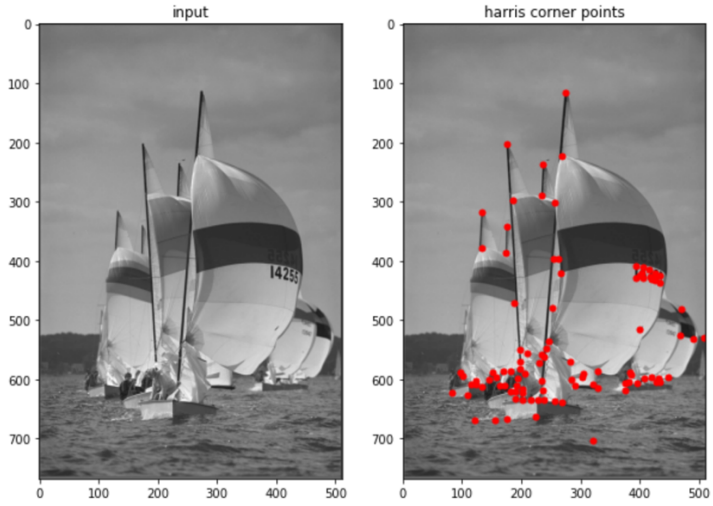
2. A zip file including all following files named **your-last-name-CA05**

- The python scripts as *.py* files.
- Anything else needed to run your python code
- Add all the saved images in a folder named *output-images*

Part A Harris Detector

In this part you will implement a function for Harris corner point detection ***at a fixed scale***. Specifically, you will:

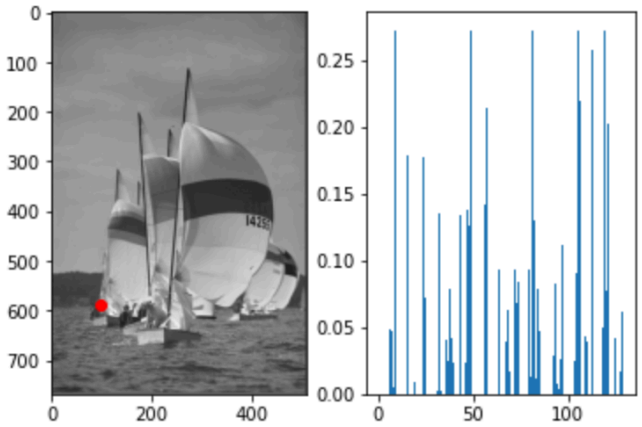
- Implement a function to generate gaussian filter and derivative of gaussian given the size and std.
- Implement a function to find Harris corner point following $H = \mathrm{Det}(A) - \alpha * \mathrm{Trace}(A)^2$
- Test your function on a provided image by visualizing Ix, Iy, Harris corner point and Harris values for the entire image. The detected Harris corner point will be similar to:



Part B SIFT Descriptor

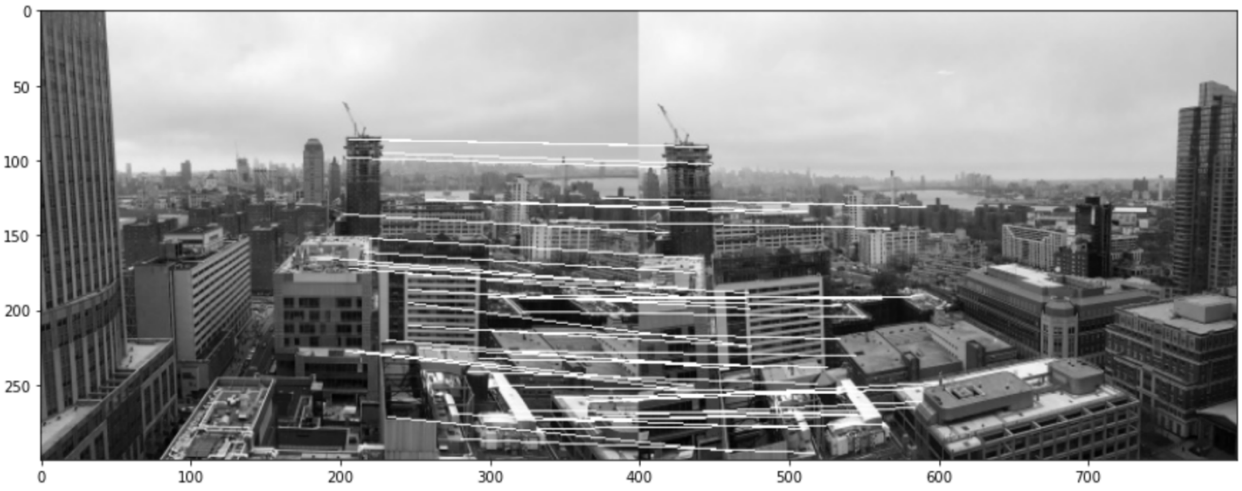
In this part we will play with the SIFT descriptor. We will use 8 bins for the histogram and extract SIFT descriptor for each Harris corner point found in Part A. We will implement the following functions:

- An atomic histogram function, which will generate a histogram given the quantized orientation and magnitude.
- A descriptor function, which find the dominant direction from the weighted gradient magnitude and construct 128d SIFT descriptor from the histograms of the splited 4x4 blocks from the 16x16 patches. The descriptor should be clipped and normalized.
- A helper function to combine the histogram and descriptor function to extract SIFT features from the images.
- You will check the correctness of your code by visualizing 10 feature points along with the extracted SIFT features using a bar plot:



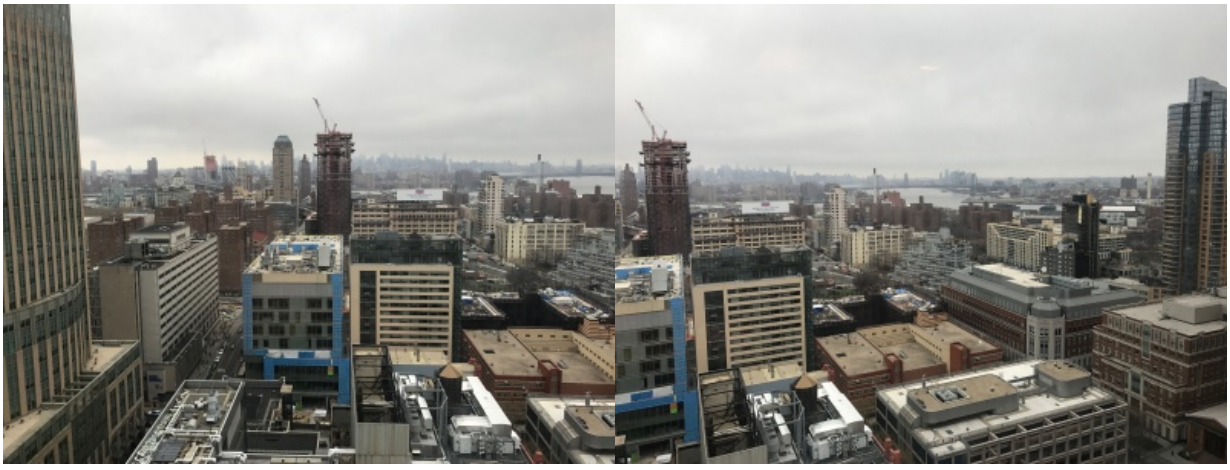
Part C Feature Matching

Next we will try to find the correspondence between the keypoints from a pair of images using the SIFT descriptor extracted with the function you implemented in Part B. For simplicity, you could measure the closeness between 2 SIFT features using L2 norm. We will **experiment on different thresholds** for the matching and visualize the correspondence by connecting the matched feature points across images. An example result will be similar to:



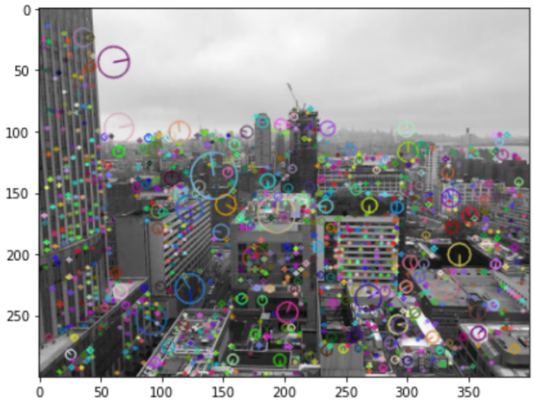
Part D Panorama Stitching

In the end, we will employ the OpenCV implemented keypoint detector and feature descriptor to perform panorama stitching. You are given two images taken from the same position but with different camera orientations. In this case, the two images are related by a homography mapping even though the imaged scene is not a plane. Can you recognize the places in these two images? (You should be able to answer this question if you have been on our Brooklyn campus).

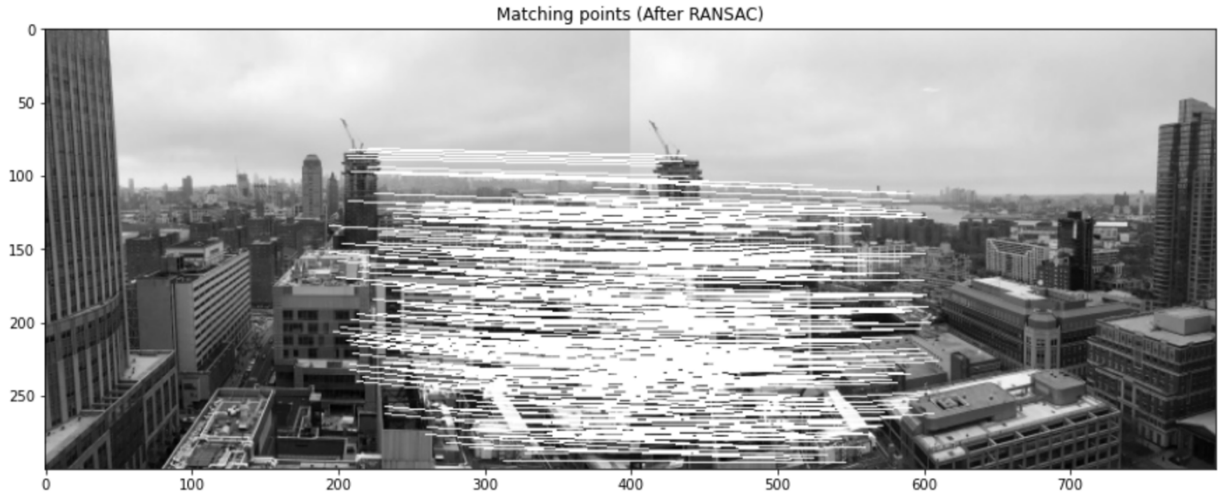


For this task, the simple single-scale Harris detector that you used earlier will not be sufficient to get good results. Instead, we will use the SIFT feature detector which is scale invariant. We will visualize the detected keypoints along with its meaningful scale in the images using the provided

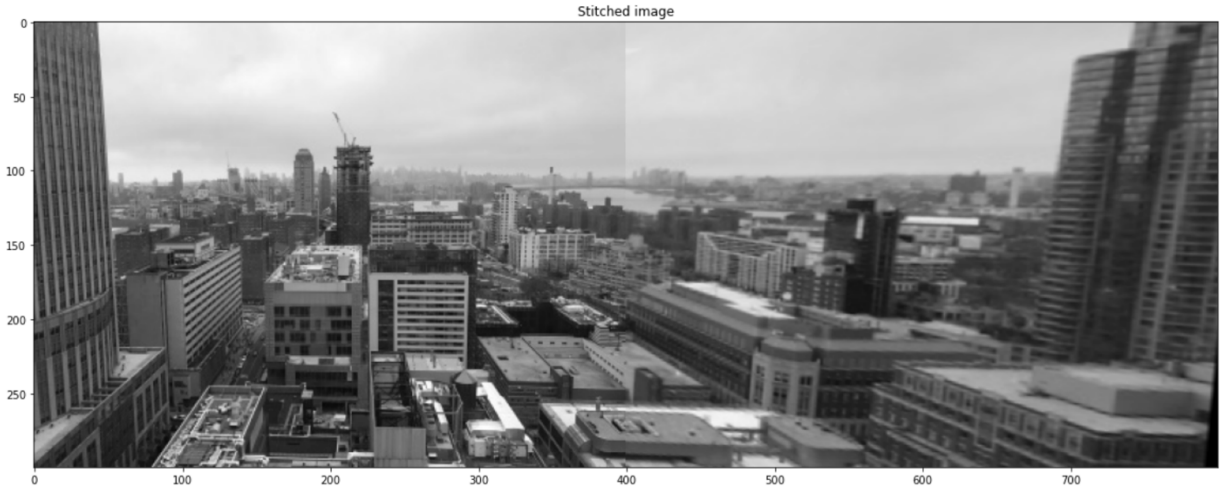
OpenCV function: cv2.drawKeypoints(), the results will look similar to the images below, where the radius of the circle characterize the scale at which the corresponding keypoints are obtained:



For feature matching, we will start with the simple feature matching approach to get the initial set of candidate matches. For this task, you could reuse your feature matching program from Part C, or you could use available Python functions. We will then use RANSAC to eliminate outliers and simultaneously determining the homography mapping between the two images. OpenCV provides handy tools to perform such matching functions for us luckily. We will compare the matching results by a visualization similar to Part C. The example results should look like:



Finally, based on the homography we found from the robustly matched keypoints using RANSAC, we could perform warping to change the second images to align with the first one and stitch the 2 images captured from different views together. This step is the key component of the panorama function that is commonly available in most smart phones nowadays. For blending, we will just simply overlay one image on top of the other for simplicity. The stitched image will look like:



Useful OpenCV functions:

- sift = cv2.SIFT_create(); keypoint = sift.detect(img); keypoint, features = sift.compute(img, keypoint)
- cv2.drawKeypoints();
- matcher = cv2.DescriptorMatcher_create("BruteForce”); rawMatches = matcher.knnMatch(features1, features2,k=2);
- cv2.findHomography(); cv2.warpPerspective()

In your solution, please first answer why the two images are related by a homography mapping.

Also, after you obtain the stitched image, please comment why the right part is blurred.