Lab Instructions

**CS4243: Computer Vision and Pattern Recognition**

Fall 2020

NUS, School of Computing
**Prof. Angela Yao**

# Lab3: Image Stitching

**Teaching assistant**: Abhinav Rai
e0403959@u.nus.edu

## Objective

The goal of this assignment is to do image stitching – how multiple images can form a panorama. To do this, you would need to implement feature detectors, feature descriptors, keypoints matching and homography computation algorithms .

# 1 Task 1 Keypoint Detection (10 points)

## 1.1 Task 1 Description

- For this task we will implement the Harris Corner Detector

## 1.2 Steps Task 1

- Compute Image Gradient $I_x$ and $I_y$ at each point in the image.
- Compute the Hessian Matrix H for each window.

$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{1}$$

- Compute Corner Response.

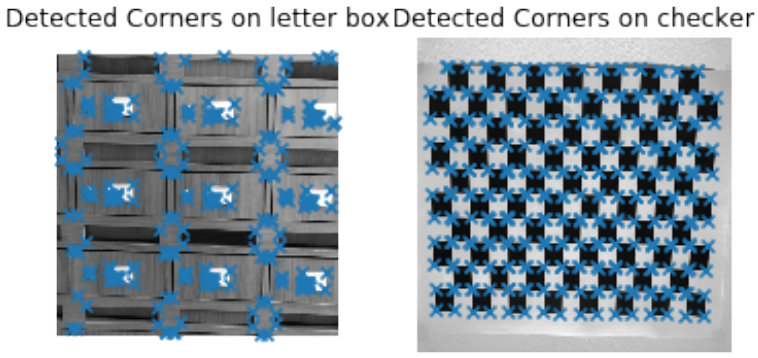
$$R = Det(H) - k\left(\mathrm{Tr}\,a\,\mathrm{ce}(H)^2\right) \tag{2}$$

- Find the points whose surrounding windows has a larger corner response (R ¿ threshold). Take the points of local maxima i.e., perform Non-Maximum Supression. This again has already been implemented.

## 1.3 Hints

- Use skimage.filters.sobel_v / sobel_h
- Use scipy.ndimage.filters.convolve

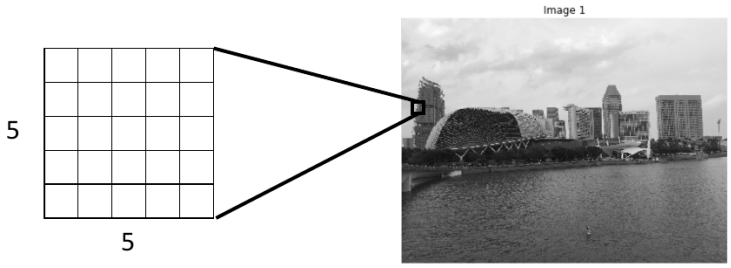
### 1.4 Expected Result Task 1




## 2 Task 2a Keypoint Description (14 points)

### 2.1 Task 2a Description

- Implement a simple descriptor that describes each keypoint with normalized intensity in a patch (e.g. size of 5*5 pixels) around it.




- Given the Feature F, perform normalization: $\mathrm{F} = \frac{F-\mu}{\sigma}$ if $\sigma > 0$ else $F - \mu$

## 3 Task 2b Keypoint Matching (14 points)

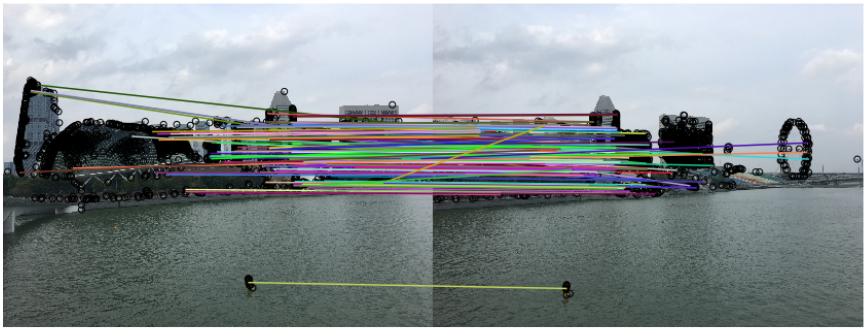
### 3.1 Task 2b Description

- Calculate Euclidean Distance between all pairs of descriptors from image 1 and image 2.
- If the distance to the closest vector is significantly (by a given factor) smaller than the distance to the second-closest, we consider it a match. The output of the function is an array where each row holds the indices of one pair of matching descriptors.

### 3.2 Hints

- Use scipy.spatial.distance.cdist()

### 3.3 Expected Results Task 2



## 4 Task 3 Homography Estimation (18 points)

### 4.1 Task 3 Description

- For this task we will be using DLT algorithm to compute the homography matrix. Specifically, we will use the normalized DLT algorithm.

### 4.2 Hints

- Use np.linalg.svd()
- Use transform_homography function in image_stiching.py
- You will not get accurate stitching results in this step, as there are a lot of erroneous matches which need to be filtered. We will implement RANSAC in the next task for that.

## 5 Task 4 RANSAC for Descriptor Matching (18 points)

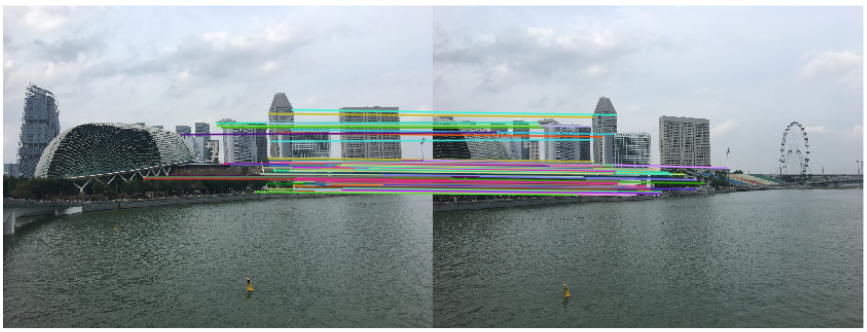
### 5.1 Task 4 Description

- RANSAC is an algorithm that is used to exclude outliers, by iteratively finding out a set of good fits and then using the best fit to do keypoint matching.

### 5.2 Steps Task 4

- Select a random set of n_samples of matches.
- Compute Homography Matrix - Refer to def compute_homography($p1, p2$)
- Find inliers using the provided threshold - Compute Sum of Squared Difference ($SSD$) on transformed M2 ($i.e. matched2$) using homography and M1 ($i.e. matched1$), and pick those lower than the given threshold as inliers.
- Repeat the above steps for n_iters and keep the largest set of inliers.
- Recompute the least squares estimate using only the inliers.

### 5.3 Expected Results Task 4 (22 points)



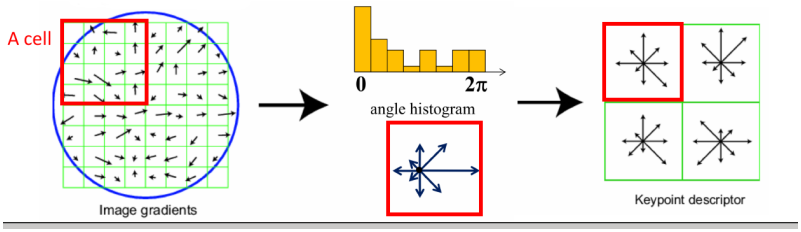

### 5.4 Hints

- Use compute_homography and transform_homography
- You will need to tune the parameters sampling_ratio, n_iters and threshold to generate accurate results.
- The final result may vary across iterations because of random sampling at each step.

## 6 Task 5 Scale Invariant Feature Transform(SIFT) (18 points)

### 6.1 Task 5 Description

- Implement a simplified version of SIFT descriptor.
- For each keypoint, take a 16x16 patch and divide it into 4x4 grid cells each of length 4.
- For illustration 8x8 patch and 2x2 grid of cells are shown below.



- Each cell should have a histogram of the local distribution of gradients in 8 orientations.
- Appending these histograms together will give you a 4*4*8 = 128-dimensional vector.
- For an image sample, the gradient magnitude m and orientation $\theta$ are computed using pixel differences. Also, features should be normalized to unit length.
  m(x, y)=$\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2}$
  $\theta$(x, y)=atan $2(L(x,y+1)-L(x,y-1), L(x+1,y)-L(x-1,y))$

## 6.2 Expected Results Task 5