Computer vision homework 2

By Achal Shah, Shubham Kumar, Kushal Kokje

Part 1

Approach used

1. We apply SIFT on given two input images and find matching descriptors by by looking at the ratio of the Euclidean distance between the closest match and the second-closest match and comparing it with some threshold.
2. We have written code so that program take input one query image and compare it with another set of images and generates a list in the decreasing order of number of matching descriptors.

Decisions

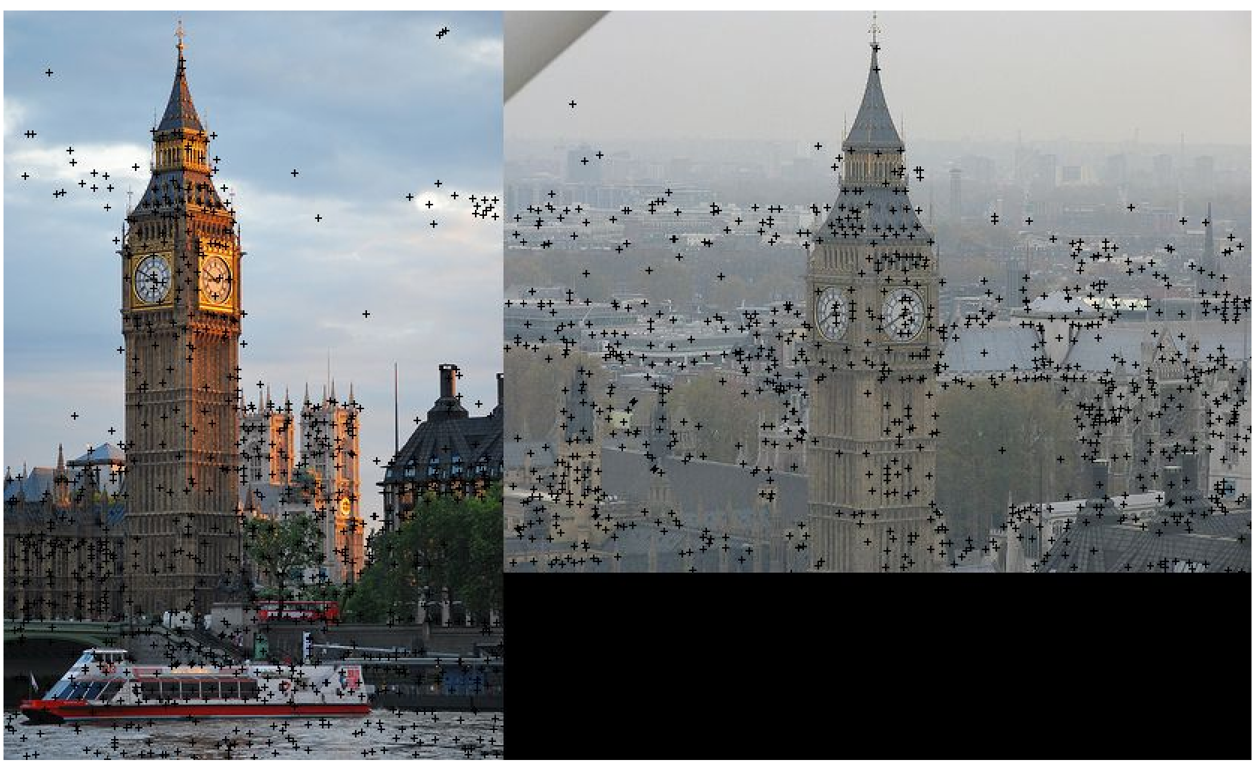
1. Threshold for ratio of closest match and the second-closest match is set to 0.85.

How to run Code

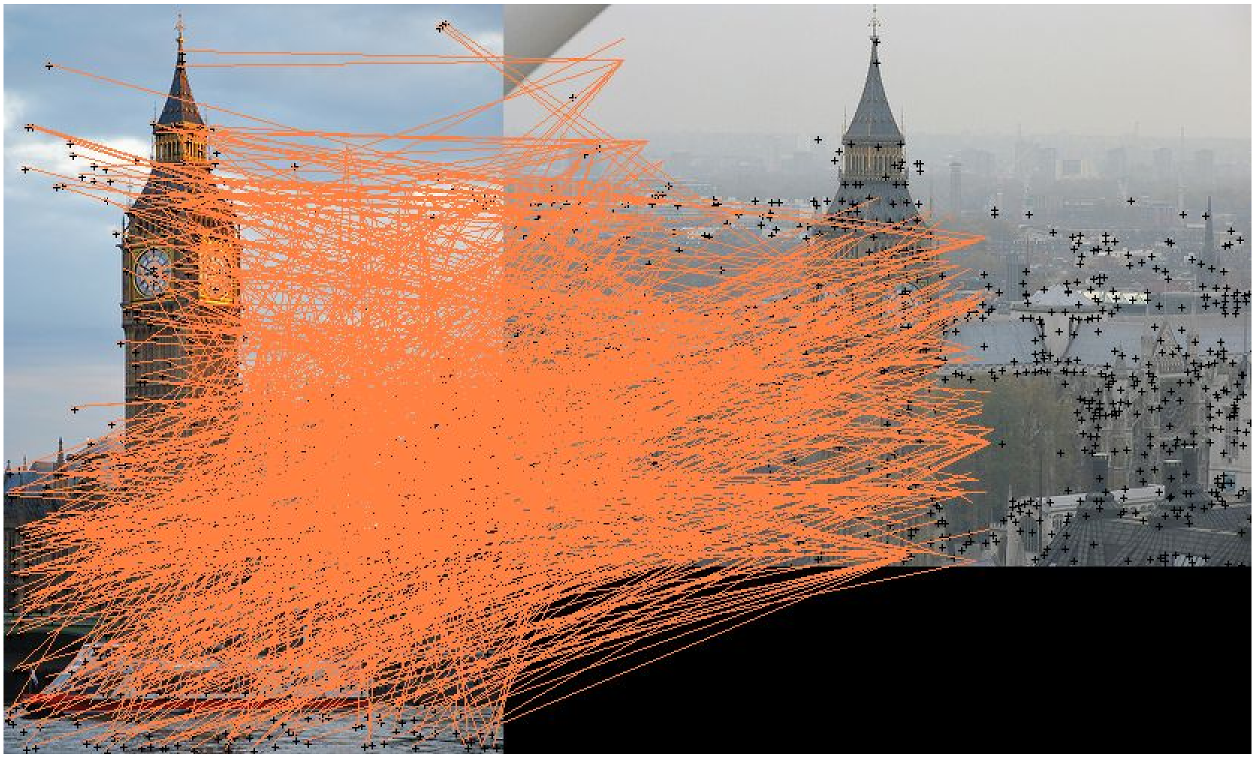
Part 1.1 Visualization of SIFT matches

Command ./a2 part1 input_image1.png input_image2.png

Example test on bigben_14.jpg and bigben_3.jpg



Sift correspondences



Part 1.2 Find best matches for query image

Command ./a2 part1 query_image.png img_1.png img_2.png ... img_n.png

Part 1.3 Checking precision

Command ./a2 part1 query_image.png <full path of folder part1_images>/*.jpg

Conclusion

- Sift matching is not a good measure because we got very low precision
- Bigben attraction was easy to find with the tuned parameter
- sanmarco was difficult attractions to find

Note: We could have find above attractions as well with different thresholds

Retrieval (Example log, find full log file on github **part1_evaluation.log**)

For query image: images/part1_images/bigben_14.jpg

81 matches for: images/part1_images/bigben_10.jpg
55 matches for: images/part1_images/bigben_14.jpg
51 matches for: images/part1_images/bigben_12.jpg
46 matches for: images/part1_images/tatemodern_13.jpg
45 matches for: images/part1_images/louvre_4.jpg
45 matches for: images/part1_images/colosseum_12.jpg
43 matches for: images/part1_images/tatemodern_16.jpg
43 matches for: images/part1_images/sanmarco_20.jpg
42 matches for: images/part1_images/tatemodern_24.jpg
42 matches for: images/part1_images/londoneye_21.jpg

Attraction	Precision
Bigben_14	0.3
colosseum_3	0.1
eiffel_6	0.3
empirestate_12	0.1
londoneye_9	0.1
louvre_9	0.1
notredame_8	0.1
sanmarco_5	0
tatemodern_6	0.3
trafalgarsquare_6	0.3

Part 2: Estimating homography

1. **Ransac**

Input:

We have sift correspondences from part 1. We will run RANSAC on that to eliminate false matching

Approach Used:

1. Pick 4 matching pairs at random
2. Estimate homography using following equation:

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ m_8 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix}^{-1} \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}.$$

3. Calculate error by multiplying homography matrix with other SIFT correspondences.
4. Declare an inlier if error is below threshold
5. Calculate number of inliers and outliers and find outlier ratio
6. Monitor outlier ratio and maintain homography for best inliers
7. Repeat steps 1 to 6 for n number of trials

Good model parameters:

1. Number of trials, n = 4000
2. Error threshold to declare inlier, e = 70

Observation:

1. If we increase error threshold then we are getting more inliers with many false positives
2. If we decrease error threshold then we get less inliers with a few false positives
3. We tuned these parameters by experience and finally these are the parameters which worked well for some images

Command:
./a2 part2_1 n_iterations error_threshold query_img target_images

i.e. ./a2 part2_1 4000 100 bigben_14.jpg bigben_3.jpg

Challenges faced:

- CImg solver did not work for me because B vector was multi dimensional
    - We used CImg inverse and dot functions to calculate homography
- Fix value of ransac parameters doesn't generalize well

Conclusion

- Ransac reduced some of the false matching but precision is still not good
- Bigben attraction was easy to find with the tuned parameter
- colosseum was difficult attractions to find

Note: We could have find above attractions as well with different ransac parameters

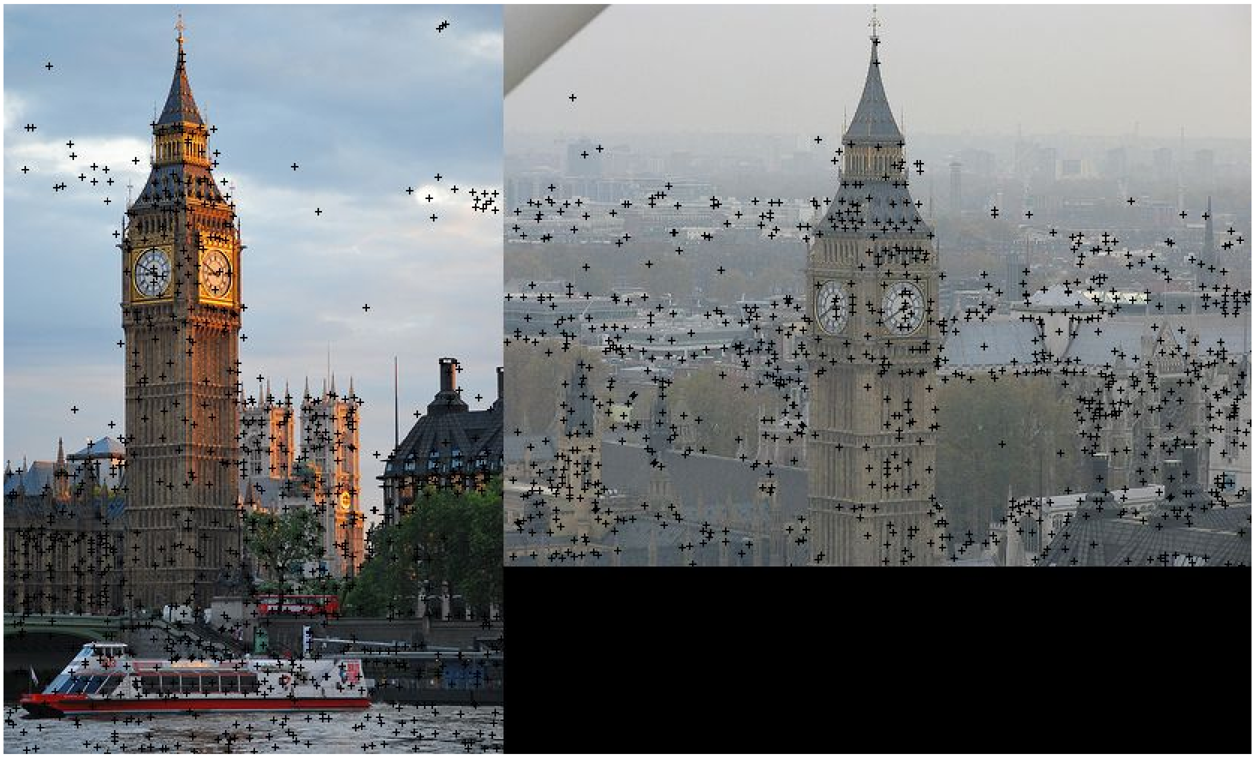
For query image: images/part1_images/bigben_14.jpg

30 matches for: images/part1_images/bigben_14.jpg
10 matches for: images/part1_images/bigben_12.jpg
9 matches for: images/part1_images/trafalgarsquare_25.jpg
8 matches for: images/part1_images/eiffel_7.jpg
7 matches for: images/part1_images/tatemodern_24.jpg
7 matches for: images/part1_images/tatemodern_16.jpg
7 matches for: images/part1_images/tatemodern_11.jpg
7 matches for: images/part1_images/eiffel_1.jpg
7 matches for: images/part1_images/bigben_10.jpg
6 matches for: images/part1_images/trafalgarsquare_15.jpg

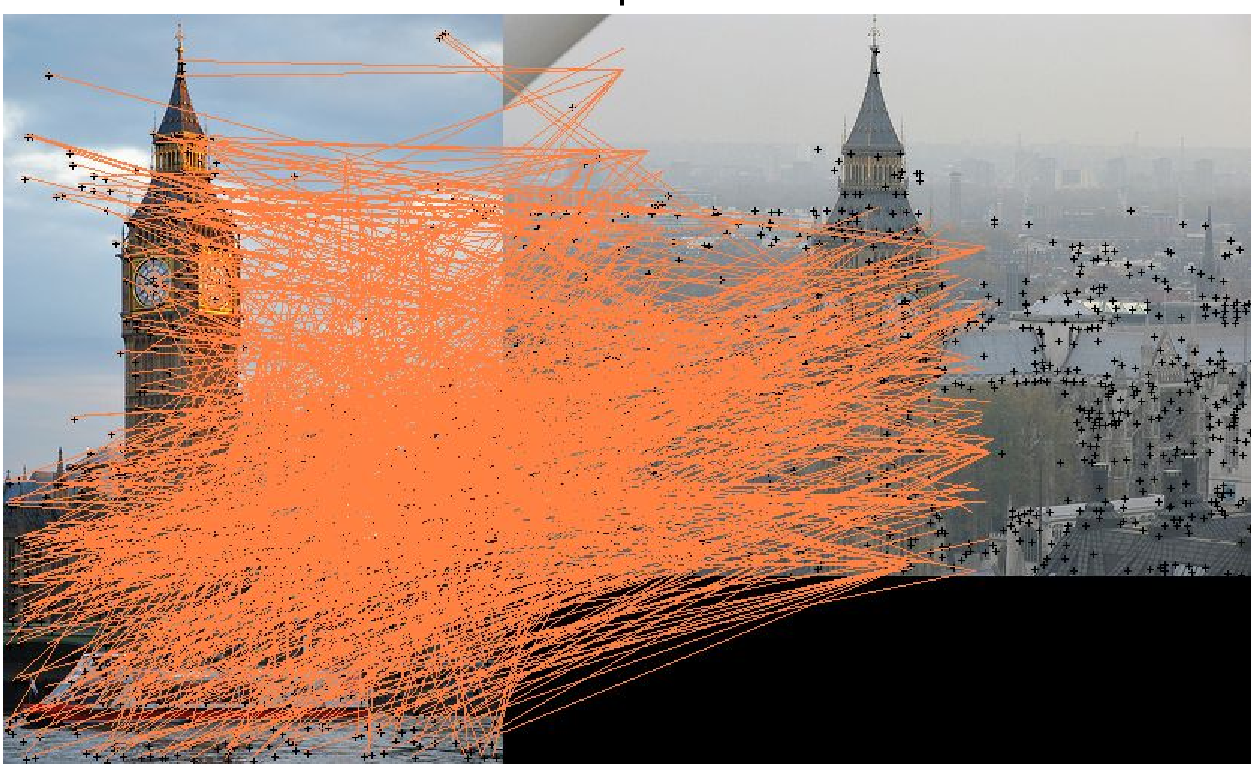
Retrieval (Example log, find full log file on github **part2_1_evaluation.log**)

Attraction	Precision
Bigben_14	0.4
colosseum_3	0
eiffel_6	0.2
empirestate_12	0.2
londoneye_9	0.1
louvre_9	0.1
notredame_8	0.1
sanmarco_5	0.1
tatemodern_6	0.2
trafalgarsquare_6	0.4

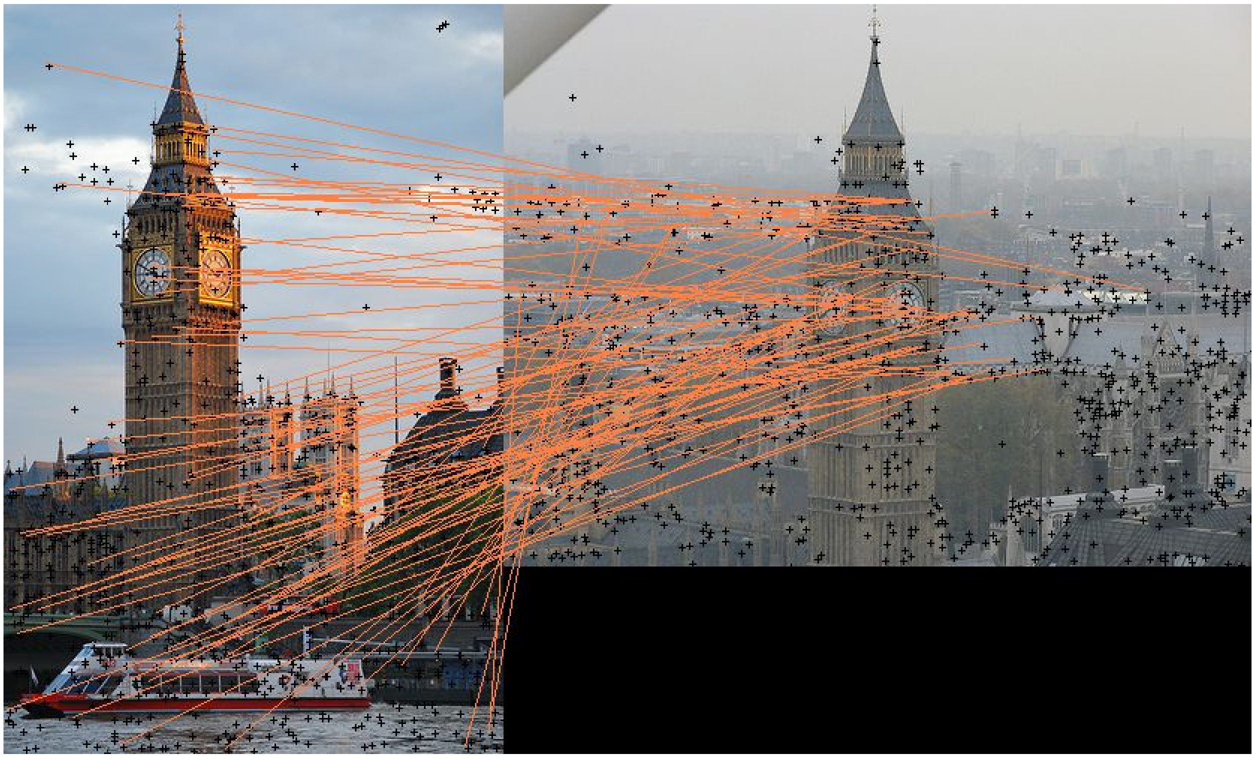
Example test on bigben_14.jpg and bigben_3.jpg



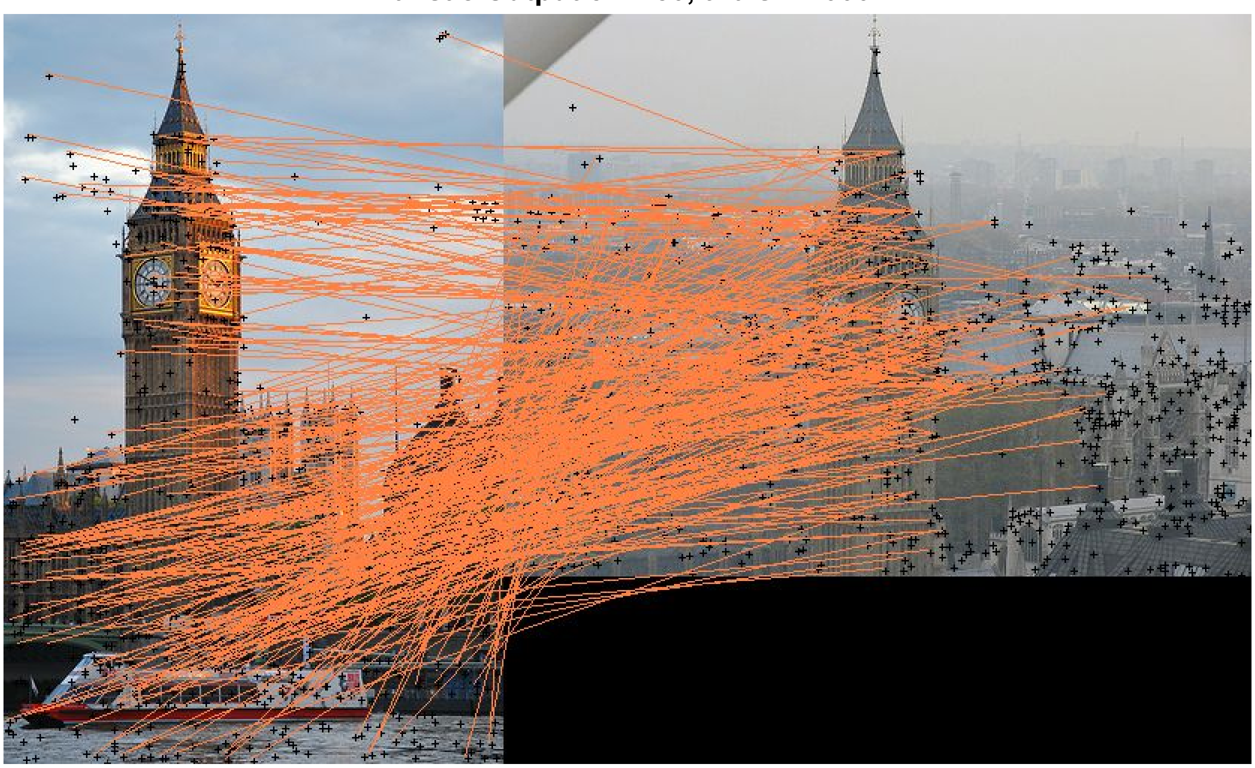
Sift correspondences



Ransac Output e = 70, trials = 4000



Ransac Output e = 150, trials = 4000



2. **Summary matches**

Input:
We have sift descriptors for both query and target images

Approach used:
1. Take some random value of quantization factor w and number of levels k
2. Initialize xi vector with a random value between 0 to 1 in uniform distribution
3. Multiply xi with sift descriptors of both images and divide them by quantization factor 2
4. For each sift descriptors we have k dimensional summary vector
5. Find summary vectors in target images which are identical to summary vectors in query image (nearest neighbours)
6. Repeat step 5 for n number of trials and pick all unique nearest neighbours from all rounds.
7. Take these nearest neighbours and calculate euclidean distance between them and query image in 128D sift space
8. Find closest points as per minimum distance and declare them a match if it is below error threshold

Good model parameters:
1. Number of trials, n = 10
2. Length of summary vector, k = 5
3. Quantization factor, w = 300

Observation:
1. If we increase value of k, we don't get any nearest neighbour because probability of getting identical vectors is very low.
2. If we decrease value of w, we don't get any nearest neighbour because probability of getting identical vectors is very low
4. Higher values of w and lower values of k worked best for us
5. Number of trials also helped us to improve calculation of nearest neighbours in random space.

Command:
./a2 part2_2 w k num_trials query_img target_images
i.e. ./a2 part2_2 500 3 100 bigben_14.jpg bigben_3.jpg

Challenges faced:
- With large k and too small w, we did not get any match

Conclusion

- Summary quantization improved retrieval speed and accuracy for some of the attractions.
- notredame attraction was easy to find with the tuned parameter
- londoneye was difficult attractions to find
- With small k, we get a noticeable improvement in speed.

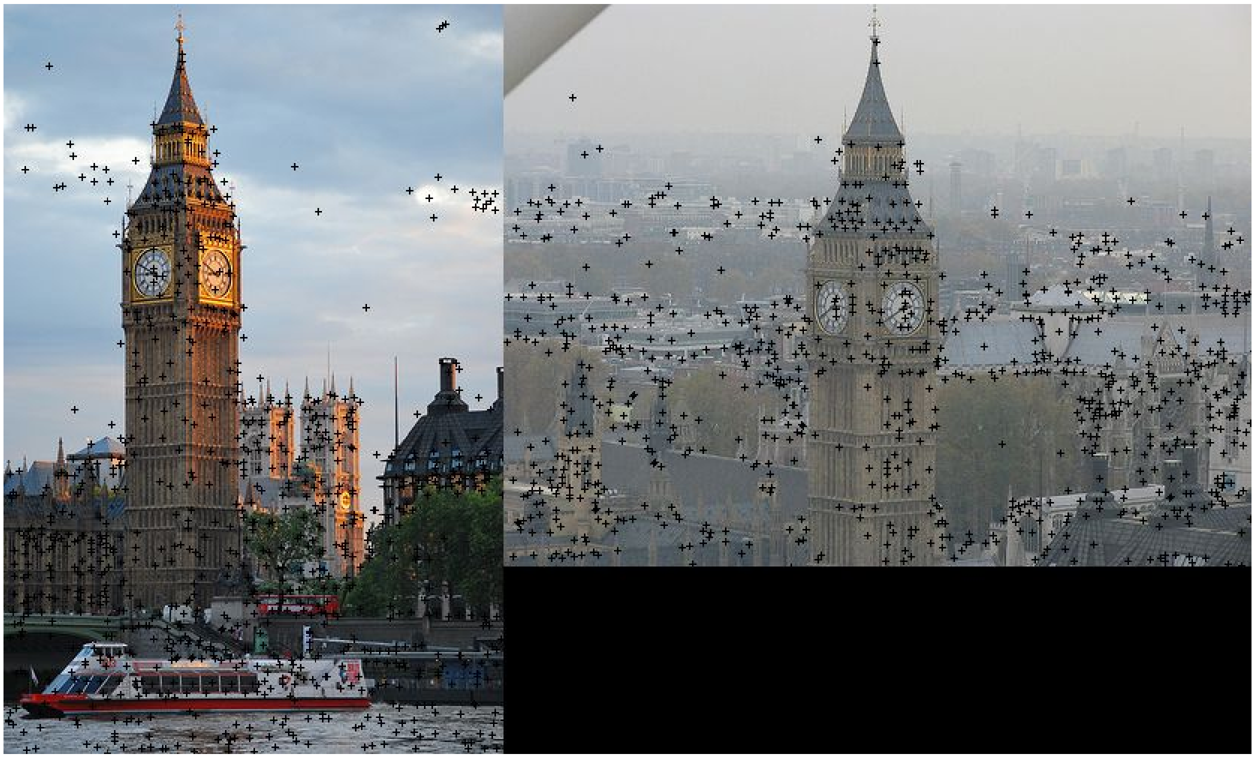
Note: We could have find above attractions as well with different w and k

Retrieval (Example log, find full log file on github **part2_2_evaluation.log**)

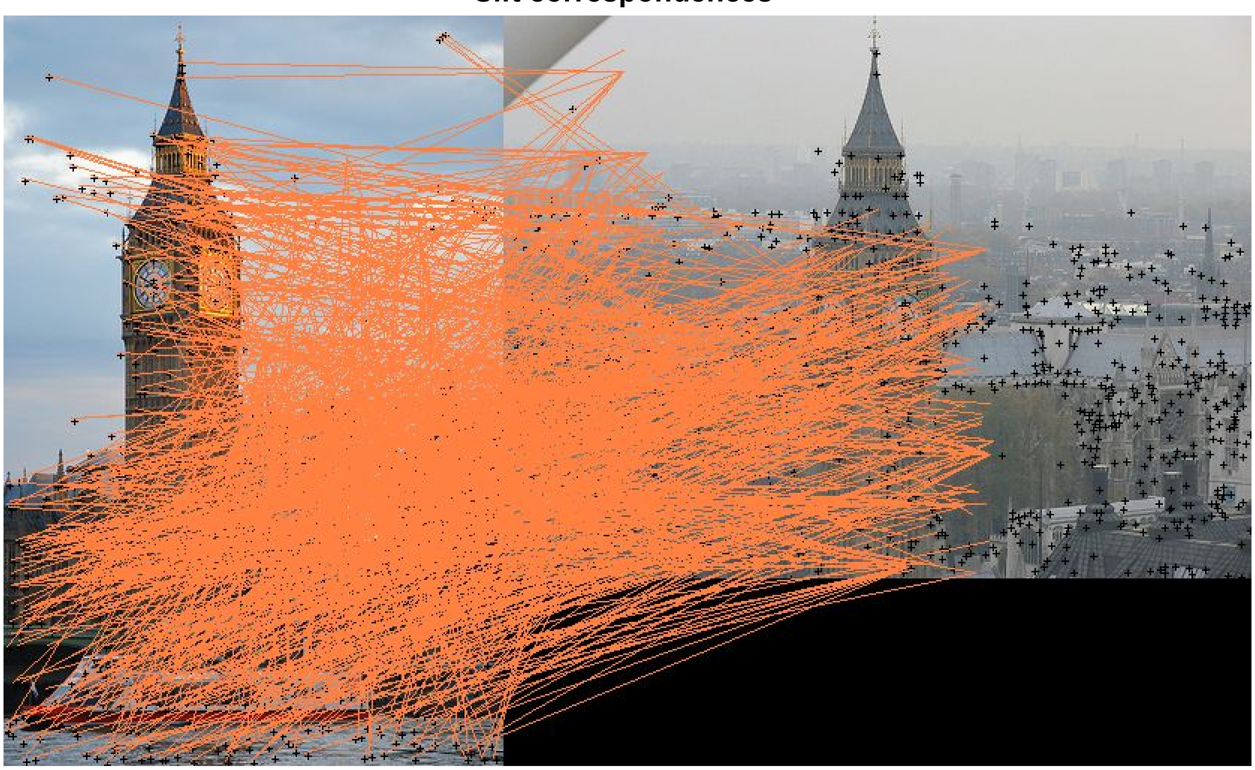
For query image: images/part1_images/notredame_8.jpg
101 matches for: images/part1_images/notredame_3.jpg
97 matches for: images/part1_images/colosseum_4.jpg
87 matches for: images/part1_images/bigben_2.jpg
84 matches for: images/part1_images/notredame_5.jpg
83 matches for: images/part1_images/notredame_14.jpg
82 matches for: images/part1_images/sanmarco_4.jpg
82 matches for: images/part1_images/louvre_15.jpg
75 matches for: images/part1_images/trafalgarsquare_15.jpg
73 matches for: images/part1_images/sanmarco_19.jpg
73 matches for: images/part1_images/eiffel_5.jpg

Attraction	Precision	SIFT Matching time	Summary Quantization time K = 3 & w= 100
Bigben_14	0.3	~220 seconds	~150 seconds
colosseum_3	0.2		
eiffel_6	0.1		
empirestate_12	0.2		
londoneye_9	0.1		
louvre_9	0.1		
notredame_8	0.4		
sanmarco_5	0.1		
tatemodern_6	0.2		
trafalgarsquare_6	0.4		

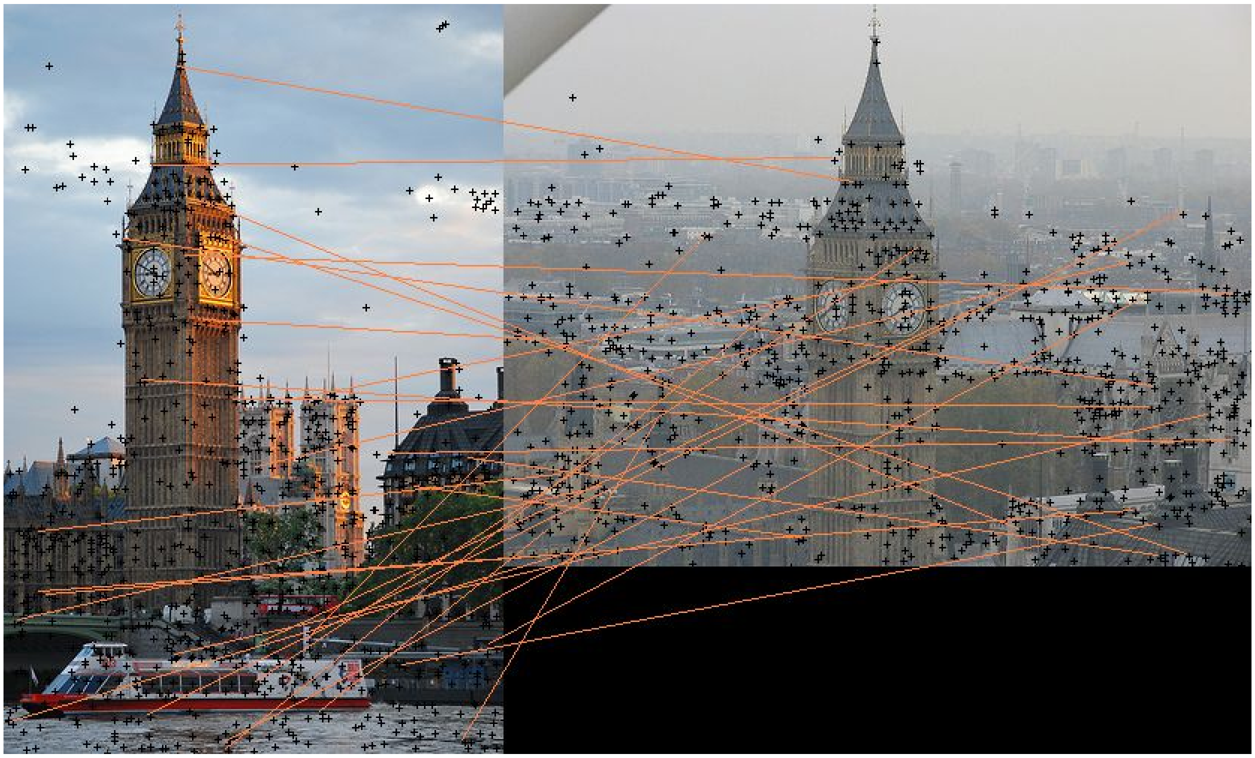
Example test on bigben_14.jpg and bigben_3.jpg



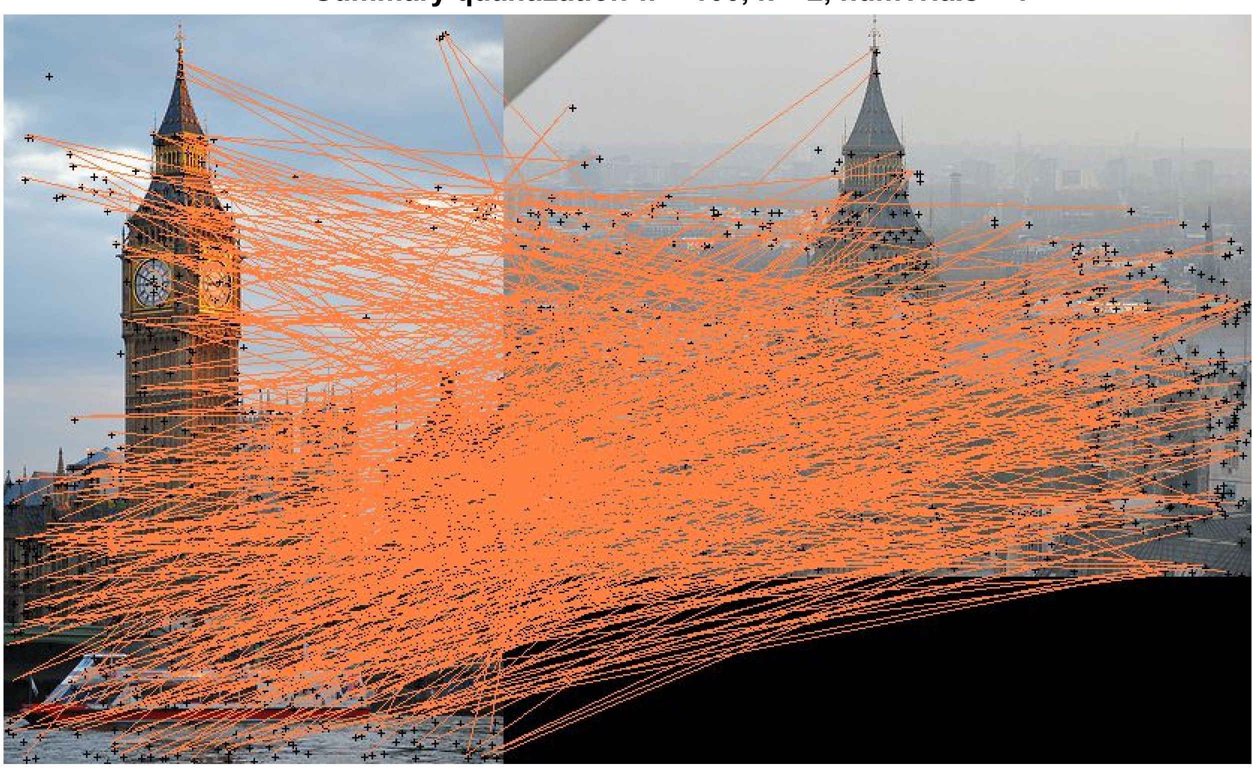
Sift correspondences



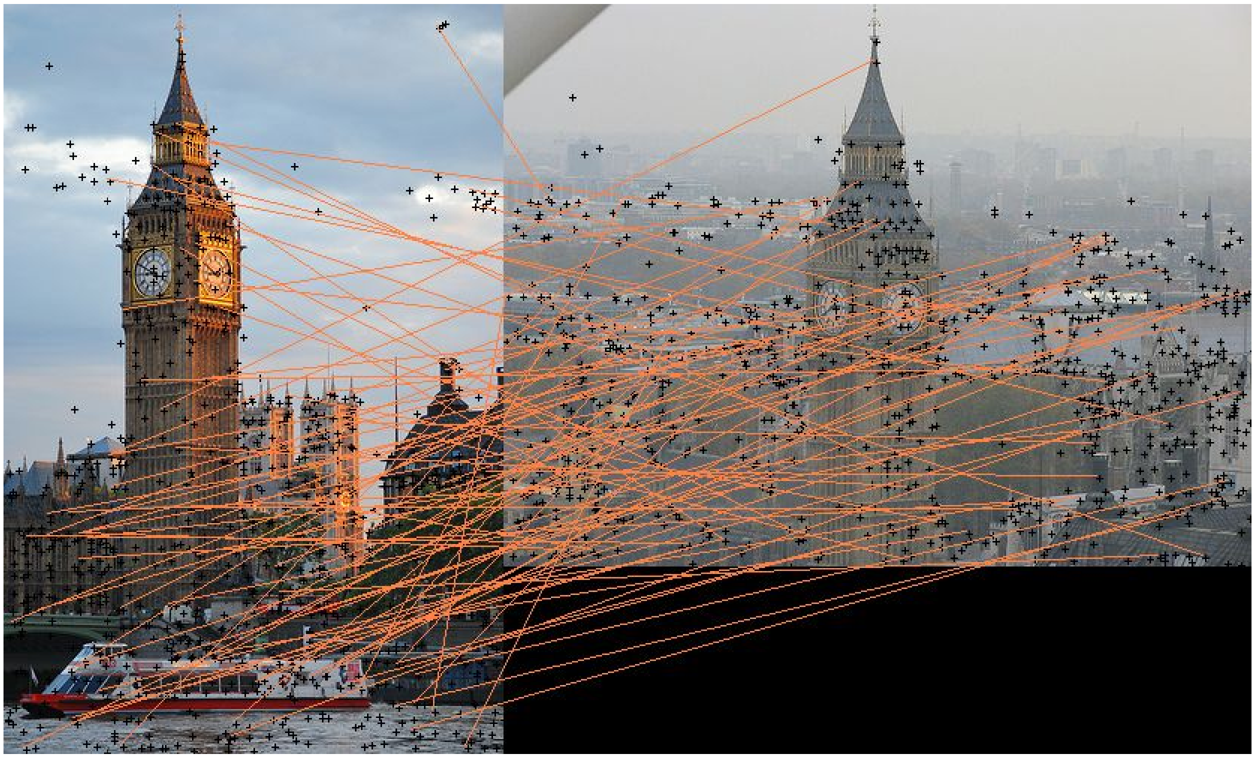
Summary quantization w = 100, k = 3, numTrials = 1



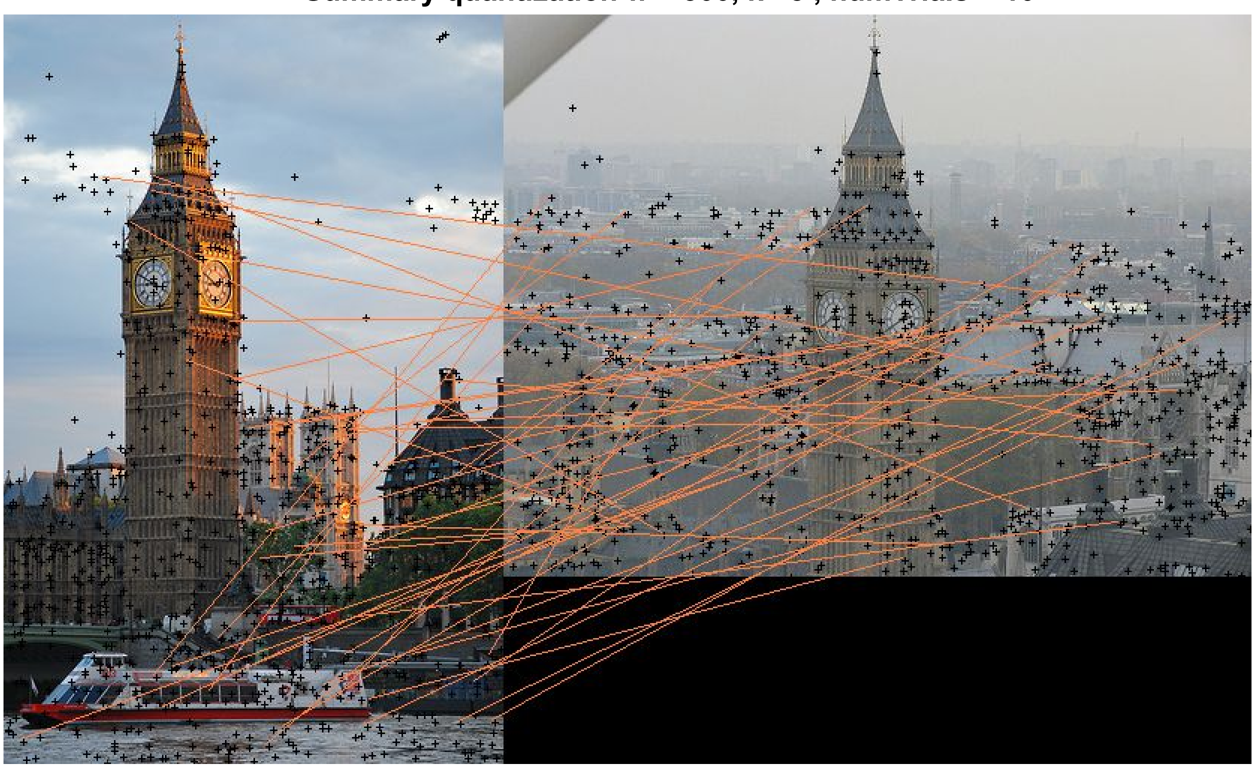
Summary quantization w = 100, k = 2, numTrials = 1



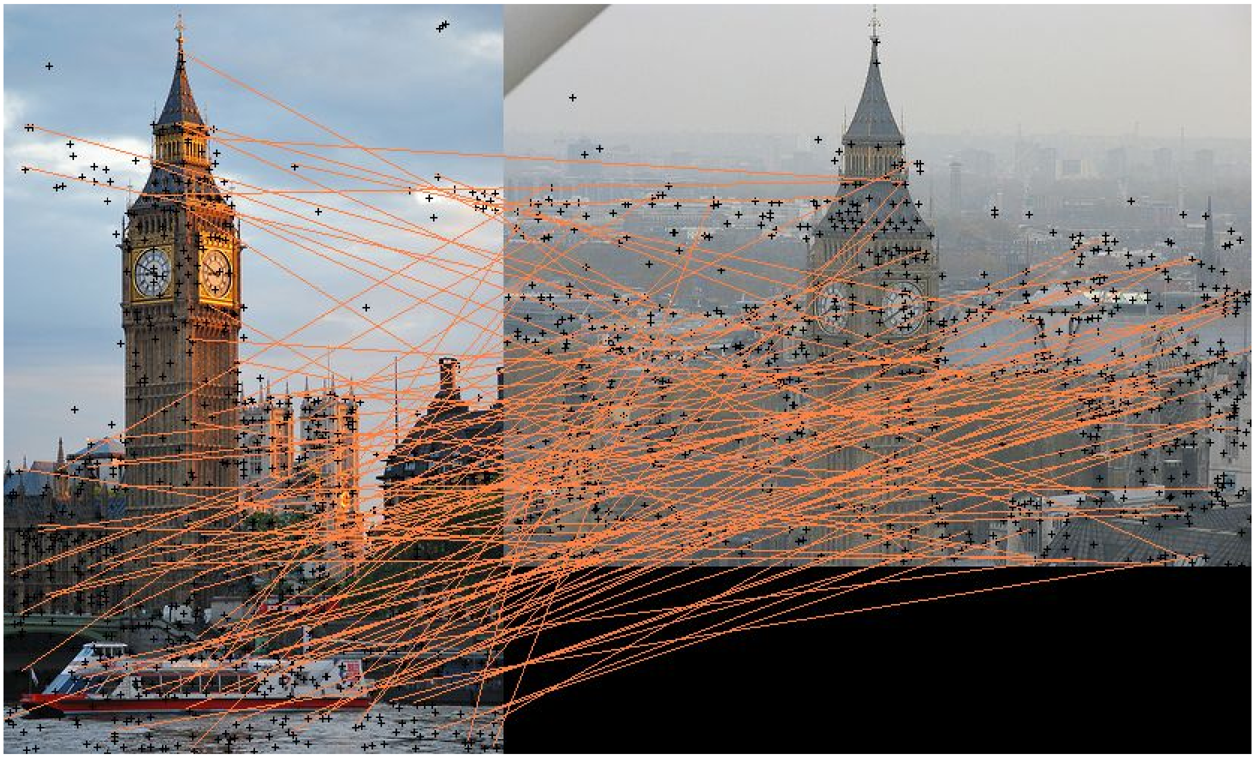
Summary quantization w = 10, k = 2, numTrials = 10



Summary quantization w = 300, k =5 , numTrials = 10



Summary quantization w = 1000, k =10 , numTrials = 100



Part 3 : Image Warping

1. Apply a given 3x3 transformation matrix to an input image (affine transformation)

Approach used : Inverse warping with bilinear interpolation

Execution step :

./a2 part3 lincoln.png

Input Image : lincoln.png

Output Image :

2. Image Sequence Warping Application

Approach used:

Get the Transformation coordinates between reference image and target image from part 2 and apply the returned transformation on the target image.

Execution Step :

./a2 part3 2298146191_888de5b755_z_d.jpg 3268706748_0d2c67f3c3_z_d.jpg 4085417699_cf3c254916_z_d.jpg 9623070553_683d9a28c4_z_d.jpg

Input Image :

2298146191_888de5b755_z_d.jpg <------ first Image (reference)

3268706748_0d2c67f3c3_z_d.jpg
4085417699_cf3c254916_z_d.jpg
9623070553_683d9a28c4_z_d.jpg

Output Image :

3268706748_0d2c67f3c3_z_d-warped.png
4085417699_cf3c254916_z_d-warped.png
9623070553_683d9a28c4_z_d-warped.png