B490/659 Project 4: Image matching and warping

Submitted By,

Rahul Pasunuri (rahupasu@indiana.edu)

Jothin Vedre (jvedre@indiana.edu)

2Q)

We have implemented two approaches to solve this problem. Both the approaches have their merits and de-merits.

Approach-1:

Here, we iterate through all the source coordinates and find the corresponding location in the target image. We, then copy the pixel RGB intensity values from source location to target location.

Pros:

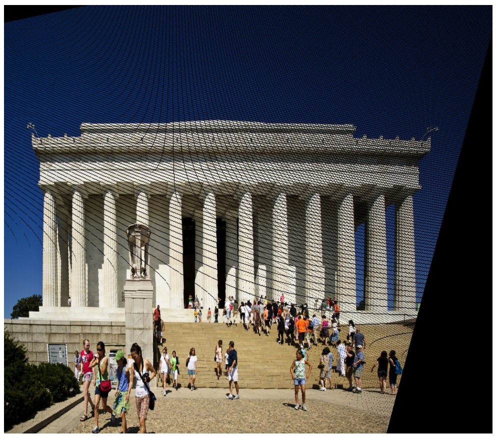
The transformation matrix need not be invertible.

Cons:

Some, of the pixels in the target may not get mapped to a source pixel, and will remain black. So, we will see some black pixels in the final output.

Source Image

Target Image



Approach-2:

First we compute the matrix inverse of the transformation matrix. We then iterate through the target pixels, and then map this pixel to a source pixel. We, then copy the pixel RGB intensity values from source location to target location.

Pros:

All the pixels in the target will have a mapping to a source pixel.

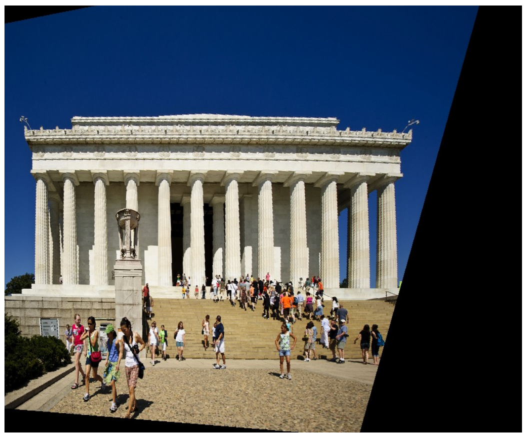
Cons:

This approach cannot be followed when the matrix is not invertible.

We, first check whether, the matrix is invertible or not. If the matrix is invertible, we follow approach-2, else, we follow approach-1.

Source Image

Target Image



3Q)

3.1)

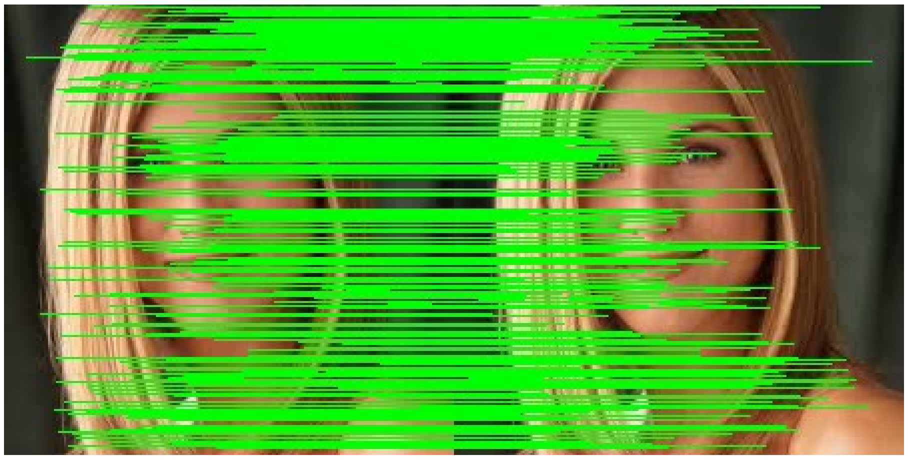
Below results are obtained using a minimum SIFT distance value of 100, for it be match.

Example-1

Source Image1

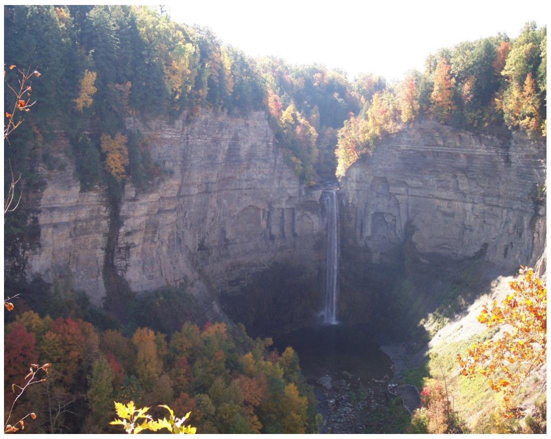
Source Image2

SIFT Matches:

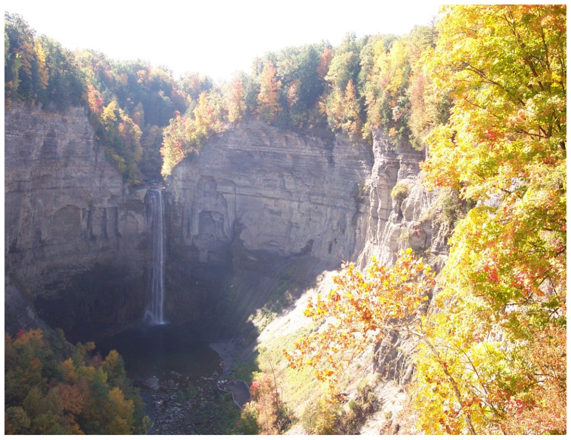


<u>Example-2</u>

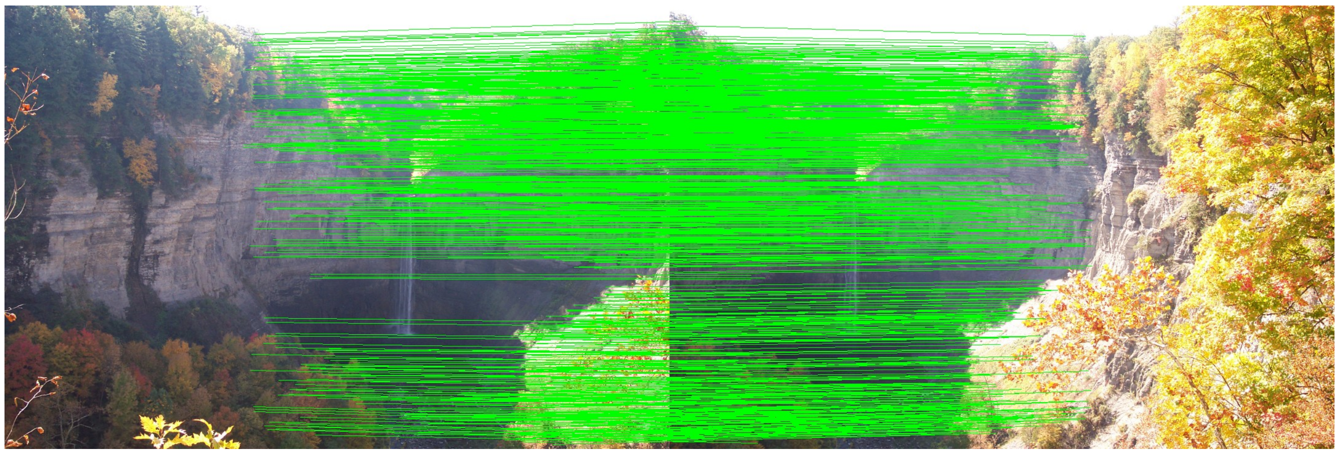
Source Image1



Source Image2



SIFT Matches:



3.3)

We have written a python script file(with the file name "commandline.py") to randomly pick a attraction image from every category and to generate the command to run "3.2".

The precision results are shown below.

Query Image	**Precision (in %)**
tatemodern_24.jpg	20
notredame_20.jpg	30
louvre_13.jpg	20
eiffel_22.jpg	30
londoneye_16.jpg	20
sanmarco_18.jpg	10
empirestate_22.jpg	30
bigben_16.jpg	40
trafalgarsquare_20.jpg	20
colosseum_11.jpg	50

A detailed list of results are present in the file "precision.txt".

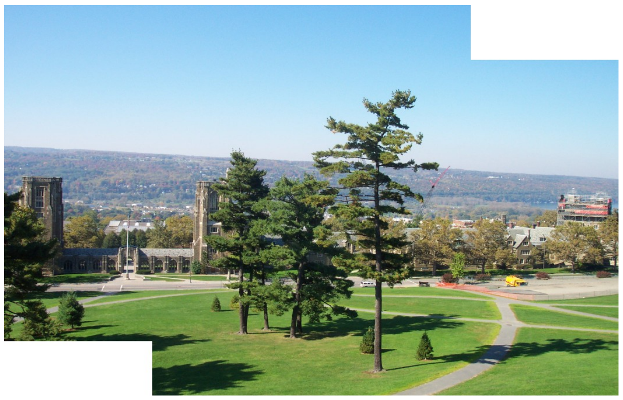
The results above show that colosseum and bigben, are the easiest attractions. And the attraction sanmarco is the hardest to find.

4Q)

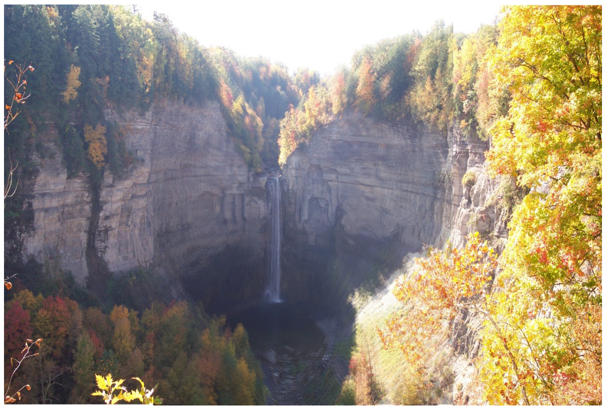
We used a displacement ratio threshold of **0.8** between the closest match and the second closest match, to filter some of the duplicate SIFT matches. We obtained the following results:

Note: falls_1.png and falls_2.png were taken from different positions. But notice that the cliff / view is concave to the camera's position, maybe because the camera is too close to it. As a result, the positioning of the camera affects the appearance of the cliff / view in the image. Hence, to stitch these images, translation is not enough. We need to apply more transformations.

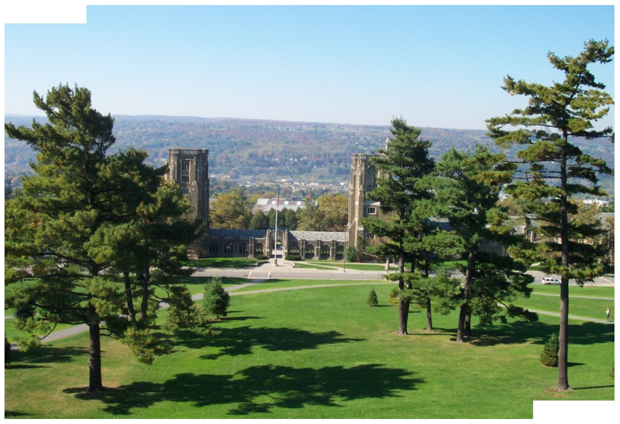
Cayuga:



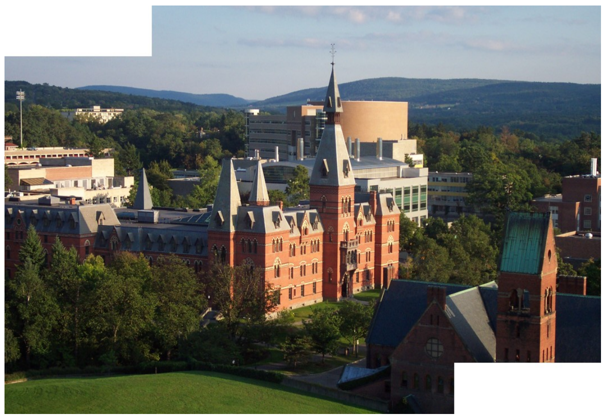
Falls:



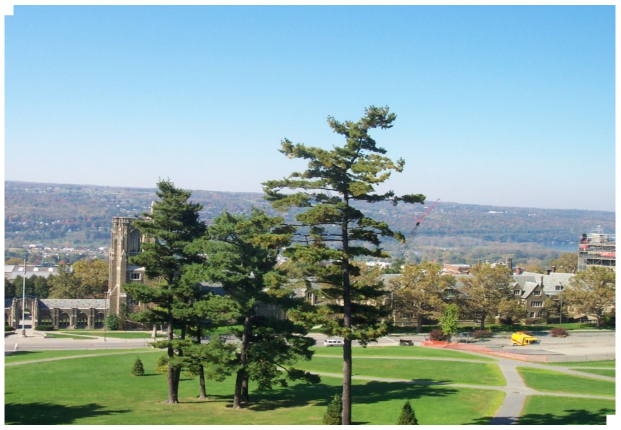
McFaddin:



Sage:



Start:



5Q)

The constants used in this implementation are:

a) k=5

b) bin_size= 100

c) hash_table_size = 101

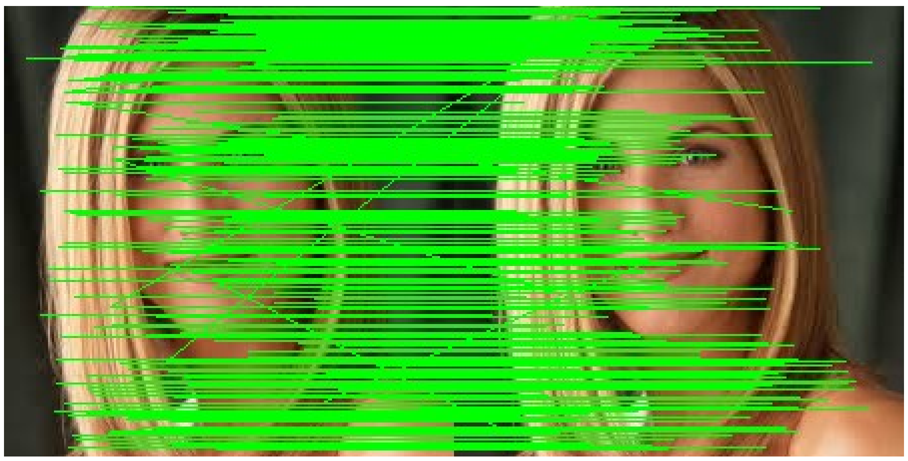
d) fingerprint_hash_table_size = 73

Example-1

Source Image1

Source Image2

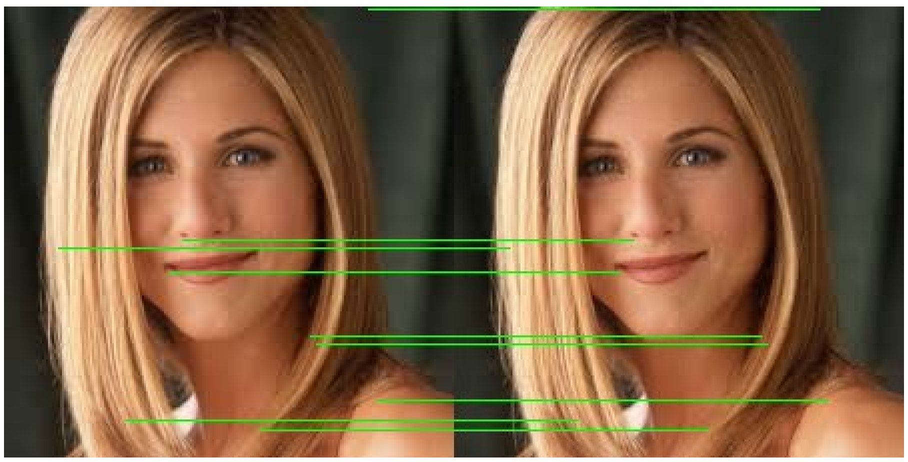
Approximate SIFT Matches:



Verifying Correctness:

In order to verify the correctness of the program, we have randomly sampled 10 matches out of all matches and resulting matches are:

Random 10 matches:



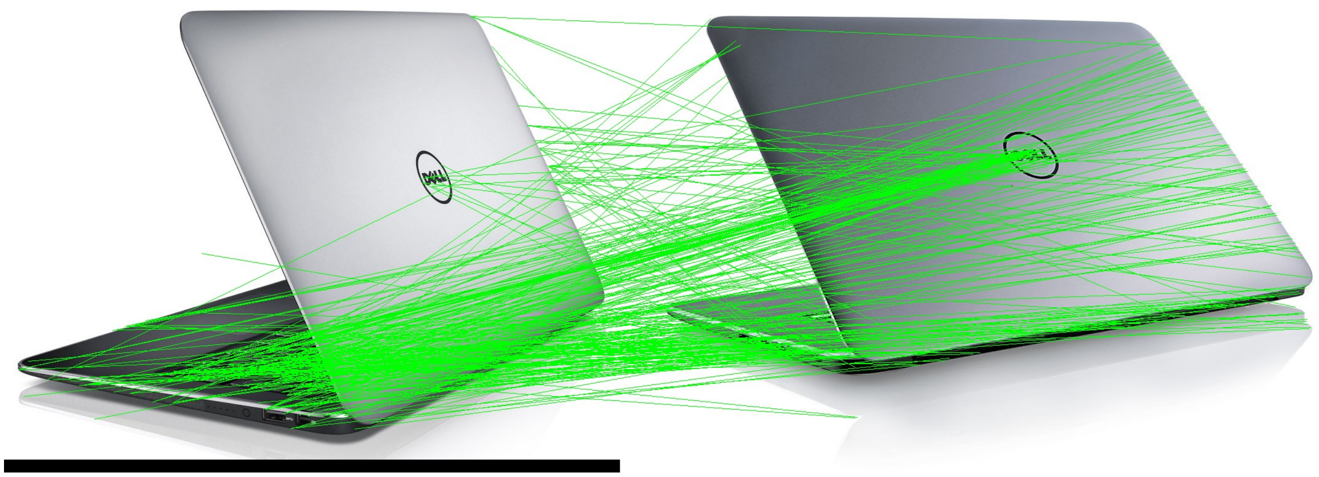
We can see that the approximate matches perform equally well.

Example-2:

Source Image1

Source Image2

Approximate SIFT Matches:



Approximate match vs Exact Match

Image-1	Image-2	Exact Match Time (in seconds)	Approx. Match Time (in seconds)	Ratio (Exact/Approx.)
dell1.jpg	dell2.jpg	3.39	3.21	1.06
jenn.jpg	jenn_copy.jpg	0.37	0.36	1.03
falls_1.png	falls_2.png	20.03	8.24	2.43

The decrease in computational time is less significant, when the number of SIFT features is very low. Hashing might cause an extra over head in such cases. The reduction in computation time can be clearly seen in the example 3 of the above table, where the number of SIFT descriptors are far greater than the other two examples (almost a factor 10).

Percentage of "Accurate" edges detected by LSH:

Image-1	Image-2	Percentage Accurate Match
dell1.jpg	dell2.jpg	3.92
jenn.jpg	jenn_copy.jpg	99

This approach is giving very accurate matches, if both the images are very much similar. But, the performance is not good in all the cases. Carefully selecting all the parameters used might improve the performance.

References

1) Inverse of a Matrix (http://way2cplusplus.blogspot.com/2014/03/inverse-of-matrix.html)