

Project Report

Code Architecture

The primary code files for the project are a2.cpp and Sift.h. The structure of files are as follows:

Sift.h

This is a header file which contains a list of commonly used functions that are called throughout the program. This is used to generate the SIFT descriptors for a particular image after computing the orientations and descriptors. It uses a class called SiftDescriptor which initializes the values for the descriptors which contain vectors in 128 dimensions.

a2.cpp

This is the main code file. It is used to perform functions like Sift matching, image warping, homography estimation etc. The various functions listed in this file are as follows:

- **comparingvalues (CImg<double> input_image3,CImg<double> input_image1,vector<SiftDescriptor> &descriptorsip, vector<SiftDescriptor> &descriptors, int taskno) :** This function is used to calculate the best features based on distance matching. It takes two images, their sift descriptors and a taskno. parameter and returns the no. of matching features as an output. Additionally if the taskno. is '1' then it also saves the matched features as a single image.
- **drawSingleImage(vector<SiftDescriptor> &descriptors1,vector<SiftDescriptor> &descriptors2,CImg<double> input_image3,CImg<double> input_image1,CImg<double> input_image2) :** This function calls comparingvalues function with the appropriate parameters and is used to match the features between two images and draw these features on a combined appended image.
- **findGaussian(string queryimg, string fileName, vector<SiftDescriptor> &descriptorsip, vector<SiftDescriptor> &descriptors, CImg<double> queryimg) :** This method is used to summarize the SIFT descriptors for a given query and sample image and summarize them using randomly generated Gaussian vectors in k dimensions. It finds the count of matching features in the two images.
- **retrievalAlgorithm(vector<string> fileNames, string queryimg, vector<SiftDescriptor> &descriptorsip, int menu) :** This function finds the precision value for all images that are correctly matched from the same attraction. It takes a vector of file names, the input image name and its sift descriptor as input.
- **inverse_warp (CImg<double> &input) :** This function takes in an image and a transformation matrix as an input and gives the transformed image as an output. It uses the inverse warping method to eliminate holes in the output image. It is also based on the nearest neighbour interpolation.
- **inverse_warp (CImg<double> input, CImg<double> warpingMatrix) :** This function takes in an image and a transformation matrix as an input and gives the transformed

image as an output. It uses the inverse warping method to eliminate holes in the output image. It is also based on the nearest neighbour interpolation.

- **CImg<double> invertMatrix(double h[3][3])** : It takes a 3x3 matrix as an input and gives its inverse as output.
- **featureDistance(SiftDescriptor idescriptor, SiftDescriptor cdescriptor)** : It takes in two sift descriptors as input and gives the euclidian distance between them as the output.
- **getMatchingFeature (vector <double> dRow, int features)**: It finds the closest neighbour to a given feature. It takes sift descriptor and its distance from the input sift descriptor as an input and return the closet neighbour only if it is closer than the second closest neighbour by some threshold value.
- **allFeatureDistance(vector<SiftDescriptor> idescriptors, vector<SiftDescriptor> cdescriptors)** : This function takes a sift descriptors of input and comparing image as an input and return the list of matching features.
- **findTransformationMatrix(vector<pair<int,int> > matches, vector<SiftDescriptor> descriptors, vector<SiftDescriptor> cmpdescriptors)** : It takes in the input and output sift descriptors and the set of matching features and finds the transformation matrix.
- **getVote (CImg <double> transformationmatrix, vector<SiftDescriptor> descriptors,vector<SiftDescriptor> cmpdescriptors, matches, vector<SiftDescriptor>)** : It get a vote on a transformation matrix form all the matching descriptors to find out the number of inliners and outliners.
- **ransac (vector<pair<int,int> > matches, vector<SiftDescriptor> descriptors, vector<SiftDescriptor> cmpdescriptors)**: It gives the transformation matrix with the most inliners.

STEPS TAKEN IN PART 1:

Part 1 deals with querying two images for matching features and plotting them or ranking them based on the maximum number of features detected.

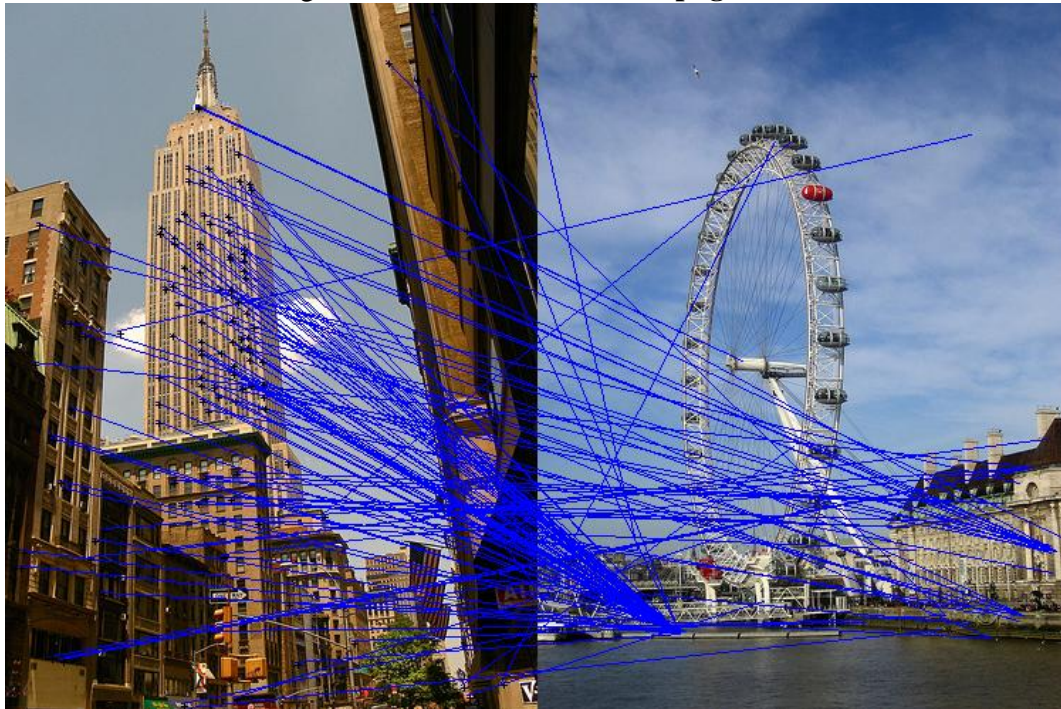
1.1

Aim

It generates one single image, superposition of the two input images, with lines connecting the matched features. A feature is matched only if the smallest Euclidian distance divided by the second smallest one is inferior to the threshold value 0.8.

- The descriptors are generated for the two images using SIFT vectors.
- The descriptors are the nqueried against each other and compared with the help of Euclidian distance to detect the closes features.
- The best and the second best features are the divided to check if they are indeed the best and no repeated feature occurs.
- The threshold maintained is 0.8 and below. The distances above the threshold are discarded.

The following is obtained on matching the two images empirestate_14.jpg and londoneye_2.jpg. The similar features with descriptors are plotted and lines connecting the commonly identified features are drawn. The image is saved in the name of **sift.png**.



1.2

Aim

It takes in a series of images and queries them against a single image to find matching features. A feature is matched only if the smallest Euclidian distance divided by the second smallest one is inferior to the threshold value 0.8. The count of the matching features per image is then used to rank them.

- The descriptors are generated for the two images using SIFT vectors.
- The descriptors are the queried against each other and compared with the help of Euclidian distance to detect the closes features.
- The best and the second best features are the divided to check if they are indeed the best and no repeated feature occurs.
- The threshold maintained is 0.8 and below. The distances above the threshold are discarded.
- The count of the matching features is stored and then sort them and display the name of the common images.

Output

On querying the images empirestate_14.jpg empirestate_9.jpg with empirestate_14.jpg we get these in order of matched features :

empirestate_14.jpg
empirestate_9.jpg

1.3

Aim

It takes in a series of images and queries them against a single image to find matching features. A feature is matched only if the smallest Euclidian distance divided by the second smallest one is inferior to the threshold value 0.8. The count of the matching features per image is

then used to rank them and the top 10 images are considered with the attraction that the query image belongs to calculate the precision of measurement.

- The descriptors are generated for the two images using SIFT vectors.
- The descriptors are the queried against each other and compared with the help of Euclidian distance to detect the closes features.
- The best and the second best features are the divided to check if they are indeed the best and no repeated feature occurs.
- The threshold maintained is 0.8 and below. The distances above the threshold are discarded.
- The count of the matching features is stored and then sort them and take the top 10 images, classify them based on the attraction they belong to and if they belong to the same attraction as the query image , they are considered for the precision value of estimation. . For a scenario of 100% precision, the ten first sorted images should correspond to the actual ten images of the attraction. We run the retrieval algorithm on the 100 images. A precision number is calculated as the percentage of images from the correct attraction in the ten first images.

Results

Image	Precision
colosseum_3.jpg	0.3
bigben_8.jpg	0.8
Louvre_13.jpg	0.3
Tatemodern_16.jpg	0.5

Observations

It was noticed that attractions like bigben had features that were very easy to identify and had a much greater precision than certain attractions like Colosseum and louvre.

1.4

Aim

Another approach is used to retrieve images with a shorter running time. First, a projection of the descriptor in the first image with a small ($k \ll 128$) set of vectors (summarization) is calculated. These are generated with help of Gaussian distribution. Then, the same projection is calculated for all descriptors in the second image. A small subset of descriptors is pre-selected based on how close their projection is to the projection of the descriptor in the first image. Then, the feature matching is applied to this small subset.

- The descriptors are generated for the two images using SIFT vectors.
- A randomly generated vector of the same dimension i.e. 128 is generated using Gaussian distribution.
- The image is then summarized using these vectors in k dimensions. The Euclidian distance between these two descriptors are then used to figure out a subset of nearest neighbours.
- This small subset is then further used to again identify the matching features among the two images using their SIFT descriptors in 128 dimensions.
- The best and the second best features are the divided to check if they are indeed the best and no repeated feature occurs.
- The threshold maintained is 0.8 and below. The distances above the threshold are discarded.

- The count of the matching features is stored and then sort them and take the top 10 images, classify them based on the attraction they belong to.
- A threshold is applied in 'w' and 'k' to control the number of matching features and the speed and accuracy of the method.

To execute : ./a2 part1fast img1.jpg img2.jpg

Observations

It is observed that this algorithm is much faster than the previous algorithms in implementing the feature matches because of reduced complexity. The values were seen to be fluctuating with the threshold values of 'w' and 'k' but the values remained to be ideal for a value of 200 for w and 10 for k.

STEPS TAKEN IN PART 2 :

2.1

Aim

First, an inverse warping function is implemented. Inverse warping has two main advantages over forward warping :

- The output image has no cracks or holes.
- All resulting pixels are within the domain of definition because of how the algorithm is constructed.

Because the corresponding pixels in the original image can be non-integer, this function uses the nearest neighbour interpolation.

- The inverse warping function is tested with a picture under the filename Lincoln.png and a 3*3 transformation matrix.
- The transformation matrix is inverted and then applied to given the warped image.
- The first image in the sequence (input image) is always stored as lincoln.jpg



The image is saved as **part2_1_warped.png** once generated.

2.2

Aim

The aim is to find the homographic estimation between the images. RANSAC is used to implement this:

- First it takes 4 random matched keypoints and solves the resulting matrix to give a homographic estimation.
- Then this homographic estimation is put to vote against all matched keypoints (as opposed to against remaining keypoint)
- A threshold variable is used since the calculated co-ordinates using the transformation matrix may not be exact. This value is maintained at 0.02 to get an accuracy of with 0.8 of the expected value.
- The number of inliers is the count of votes
- The process is repeated multiple times and the transformation matrix with the highest vote is select as the transformation matrix
- A parameter 'repetitions' is used to tune the number of repetitions (trials). In the code it is set to 10000. This has a running time of less than 2 minutes.

2.3

Aim

The aim here is to create an image warping sequence for the images in the query.

First a homographic estimation is calculated using step 2.2. The first image is taken as the input image and the estimation is calculate with respect to it.

This is homographic estimation is then used as a transformation matrix to get the warped image. The warped image is saved as 'with the suffix '_warped.png'

This process is repeated for all the images to get the warping sequence.

Observations

As the number of repetitions is increased, the homographic estimation improves but at the cost of running time. The threshold of 0.8 for keypoint matching gives the best estimation. Increasing this value gives less matches (over fitting) and worsens the estimation.

In general, seq1 images give better output than seq2 as the number of keypoint match is higher. In terms of individual images, front facing images with the plane of the tourist attraction parallel to the image plane (plane of camera projection) gave better results. For e.g.

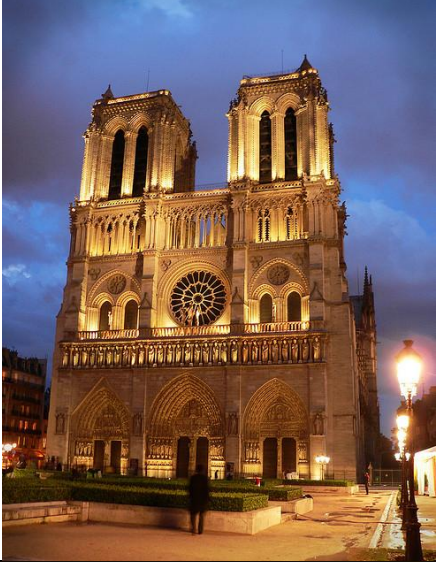



'2298146191_888de5b755_z_d.jpg' from seq1 and '2386920642_11da778137_z_d.jpg' from seq2 gave the best results in their respective sequences.

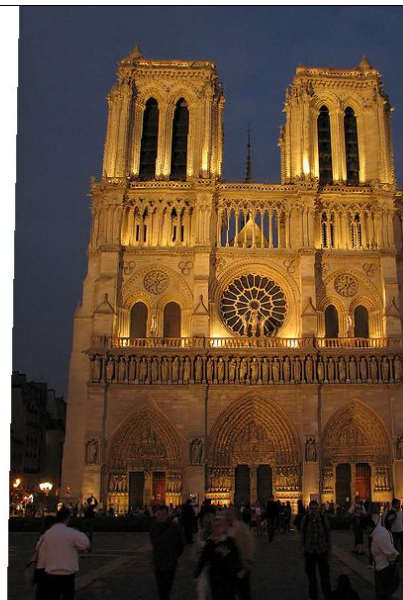
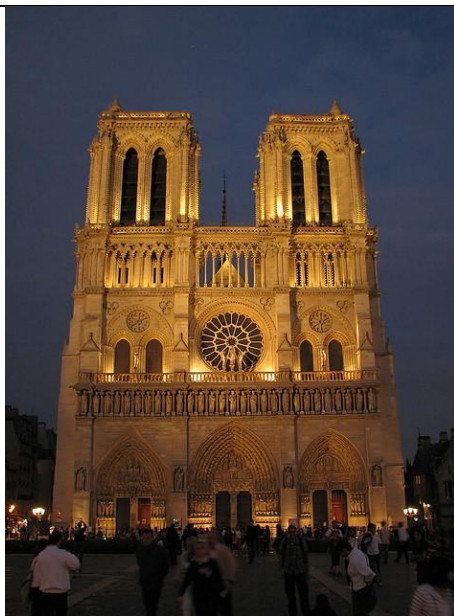
Sample output :

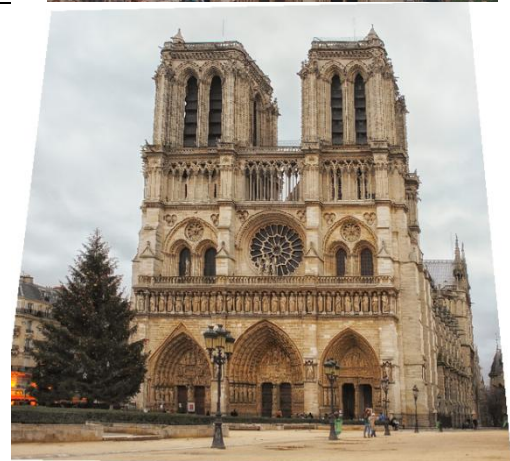
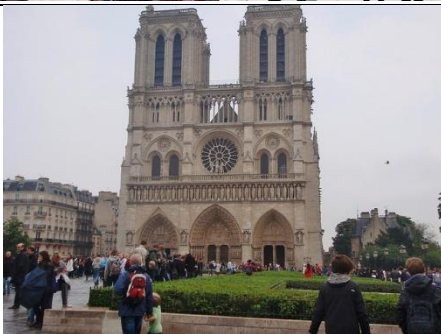
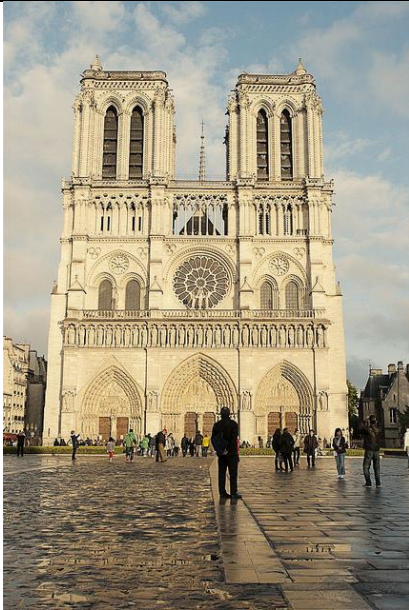
Input image '2298146191_888de5b755_z_d.jpg'



Result

Input Image	Warped image
	
	





References:

- Szeliski, Computer Vision: Algorithms and Applications. Springer, 2011
- <http://www.cse.psu.edu/~rtc12/CSE486/lecture16.pdf>
- http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf
- <http://ncalculators.com/matrix/matrix-determinant-calculator.htm>
- http://www.cs.unc.edu/~lazechnik/research/fall08/lec08_faces.pdf