B657
Spring 2016
Assignment 2
Image Warping,Matching and Stitching

necsashi-vraizada-li514

February 28, 2016

1 Introduction

In this assignment we work on Image matching,warping and stitching.We match images based on the interest points detected by applyng SIFT.SIFT interest points are 128-dimensional co-ordinates.Based on the SIFT point matches betwen the images we find the right image match for a given image and rank the matched images in the order of best matches.We further use the SIFT descriptors to generate a homography matrix between two images and then use this matrix to generate a warped image of the first image from the viewpoint of the second.

2 PART I

2.1 Visiualization of SIFT Matches

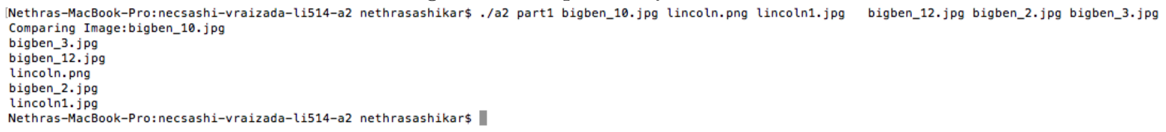
We take two images and apply SIFT on each of the images.After applying SIFT, we pass the descriptors to find the matching descriptor points across two images.For each descriptor on the query image we calculate the first minimum and second minimum distance with the descriptor in input image and calculate the ratio of it.We do this for all the descriptors in query image and sort them.We take the 1/40th value as the thershold.Then we draw lines for each of the decriptors whose calculated ratio is less than the threshold.
Output Image:sift.png

2.2 Simple Image Retrieval

We do the image retrieval based on the threshold we find in the above step. We compare the query image with the list of images given for comaprison by passing it as command line arguements and find the threshold for each image.Based on threshold we order the image and lower the threshold value,higher its chances of getting matched.We rank the image based on the threshold value and return the image rank.
Output Image:PartIQ2.png

.

Figure 1: Output of Question2



```
Nethras-MacBook-Pro:necsashi-vraizada-li514-a2 nethrasashikar$ ./a2 part1 bigben_10.jpg lincoln.png lincoln1.jpg   bigben_12.jpg bigben_2.jpg bigben_3.jpg
Comparing Image:bigben_10.jpg
bigben_3.jpg
bigben_12.jpg
lincoln.png
bigben_2.jpg
lincoln1.jpg
Nethras-MacBook-Pro:necsashi-vraizada-li514-a2 nethrasashikar$ 
```

2.3 Performance Experiment for Image Retrieval

We do image matching for each type of ten attractions and try to find the top ten matches for each image.

- We store one image for each attraction in a array of strings and compare it with every other image in the folder part1_images.
- For every image we find ten images that are the closest matches.

2.4 Image Matching based on quantized function

- In this part we are trying to reduce the descriptor dimensions from 128 to some reduced value k.
- we pass the descriptors of the image for which the dimensions have to reduced to the function **compute()**.
- We assume a value for **w,k and Number of trails**.
- Now we generate a random vector of 128 dimensions whose values are between 0 and 1
- We compute the function

$$f_i(v) = \left\lfloor \frac{x^i.v}{w} \right\rfloor$$

- **w** controls the degree of quantization
- x is randomly generated vector of 128 dimensions whose value ranges from [0,1]
- We compute this function k times to reduce the vector dimension to k from 128 dimension.
- we carry out this procedure for certain number of trails for the random projections to cause nearby SIFT vectors and also adjusting the value of w.

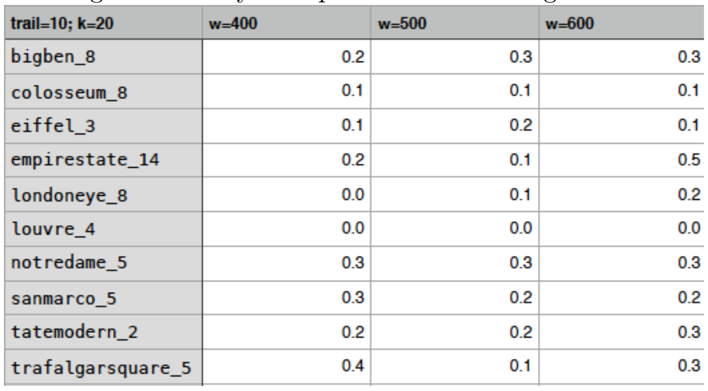
2.5 Analysis of time taken by question 3 and question 4

- Question 3 takes 985.282 seconds for 10 attractions to be matched with 100 images
- Question 4 takes 927.596 seconds for 10 attractions to be matched with 100 images
- Thus reducing the dimensions of the vector reduces the time taken for computation of image matching

2.6 Analysis of precision based on w,k and number of trails

.

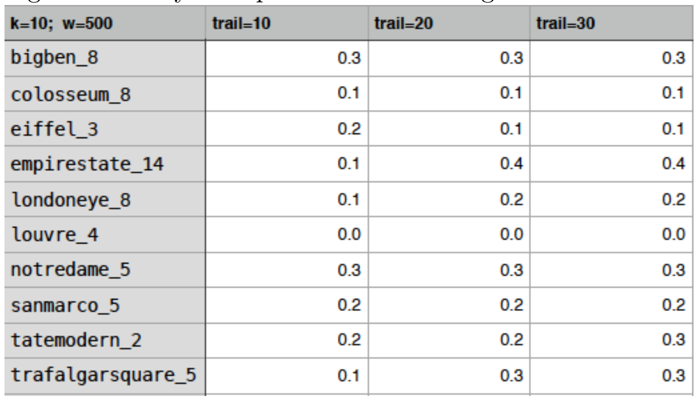
Figure 2: Analysis of precision with change in 'w'



trail=10; k=20	w=400	w=500	w=600
bigben_8	0.2	0.3	0.3
colosseum_8	0.1	0.1	0.1
eiffel_3	0.1	0.2	0.1
empirestate_14	0.2	0.1	0.5
londoneye_8	0.0	0.1	0.2
louvre_4	0.0	0.0	0.0
notredame_5	0.3	0.3	0.3
sanmarco_5	0.3	0.2	0.2
tatemodern_2	0.2	0.2	0.3
trafalgarsquare_5	0.4	0.1	0.3

.

Figure 3: Analysis of precision with change in trails and 'k'



k=10; w=500	trail=10	trail=20	trail=30
bigben_8	0.3	0.3	0.3
colosseum_8	0.1	0.1	0.1
eiffel_3	0.2	0.1	0.1
empirestate_14	0.1	0.4	0.4
londoneye_8	0.1	0.2	0.2
louvre_4	0.0	0.0	0.0
notredame_5	0.3	0.3	0.3
sanmarco_5	0.2	0.2	0.2
tatemodern_2	0.2	0.2	0.3
trafalgarsquare_5	0.1	0.3	0.3

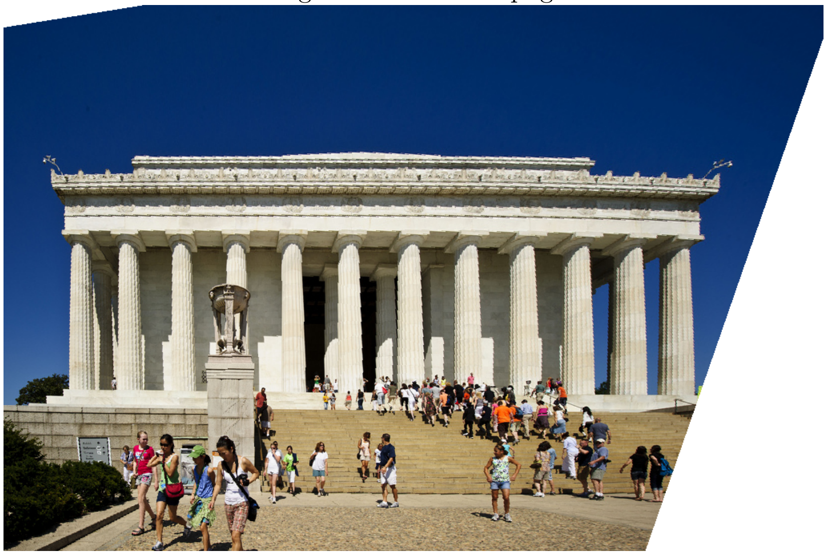
3 PART II

3.1 Image Warping

- Function name:getWarpedImg (matrix , input_image)
- This function takes a 3x3 matrix and warps the input_image given as parameter according to the matrix using inverse warping. A warped image is returned.

.

Figure 4: transform.png



3.2 Homography Estimation

- In this function sift descriptors are compared between two images and all descriptors below threshold are stored in a list.
- Then randomly four points are selected from the list and a homography matrix is generated.
- Inliers are counted for the matrix.
- The process is repeated N (Magic Number :P) times, and the matrix with maximum inliers is returned by the function.

3.3 Combination

- For all images given through command line, first the homography matrix is generated using getHomography(). Then this matrix is passed to getWarpedImg() to get the warped image.

3.4 Problems Faced

- The matrices generated are not always best. This can be due to the quality of SIFT matches as the performance of RANSAC depends on the number of outliers.
- The CImg library uses column coordinate as x and row as y which is not intuitive and I was always confused as to use matrix or its transpose.

4 Outputs

4.1 Steps to run the code:

Part I
Question 1

1. Pass the parameter **"./a2 part1 lincoln.png lincoln1.jpg"**
2. Part I gives the output image **sift.png**,which has the sift lines drwan between the first two images **lincoln.png lincoln1.jpg**

Question 2

1. Pass the parameter **./a2 part1 lincoln.png lincoln1.jpg lincoln.png bigben_10.jpg bigben_12.jpg bigben_2.jpg bigben_3.jpg** for question 2.
2. It gives the list of images in the order of matching.
3. lincoln.png is matched with rest of the images and image ranking is made.

Question 3

1. Although Question 3 is independent of the parameter passed, we should send minimum of two images by parameter for the code to execute part I.
2. Pass the parameter as we did for question 1 or question 2

Question 4

1. Pass the parameter as we did for question 1 or question 2.
2. It is independent of the parameters passed.

Part II
Question 1

1. The code is commented, just pass the image name(Lincoln.png) via command line. The output file will be generated by the name transform.png.

Question 2

1. The code is commented, pass name of two images via command line. The output will be generated on the console.

Question 3

1. Pass the constant image as first input to command line and follow it with the rest of the images.