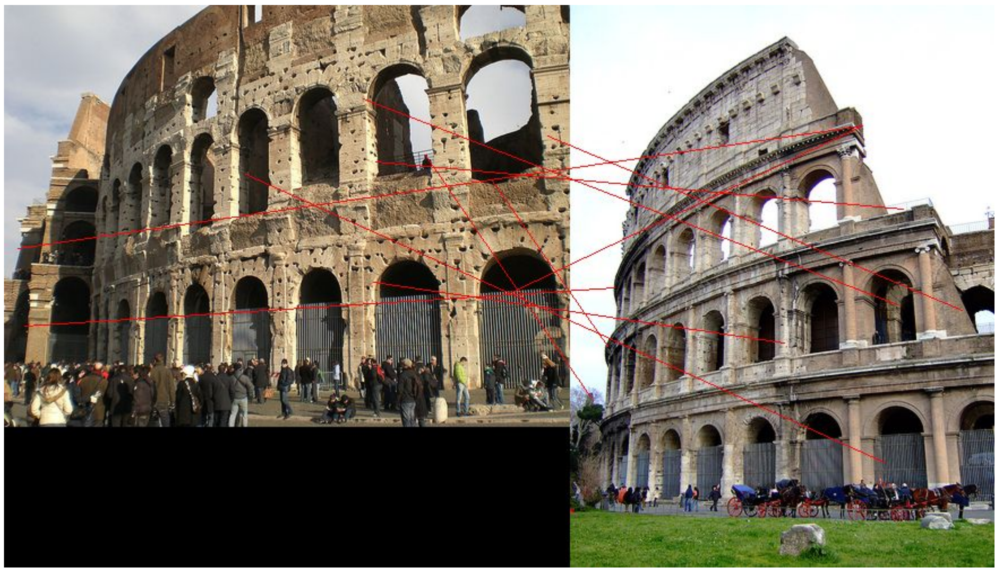
Assignment 2
Final Report

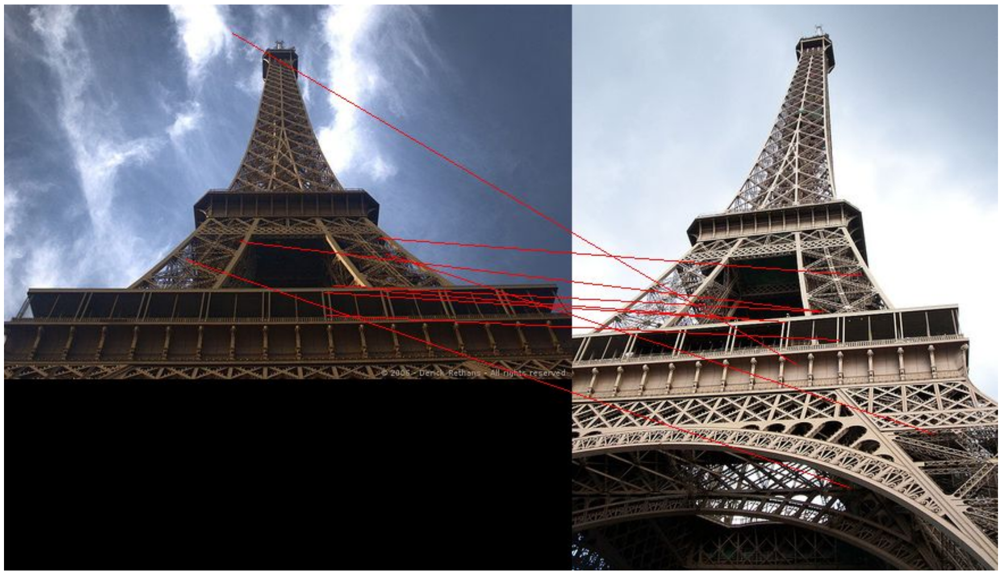
Rohil Bansal bansalro@indiana.edu
Ishita Ganjoo iganjoo@indiana.edu
Archana Molasi molasia@umail.iu.edu

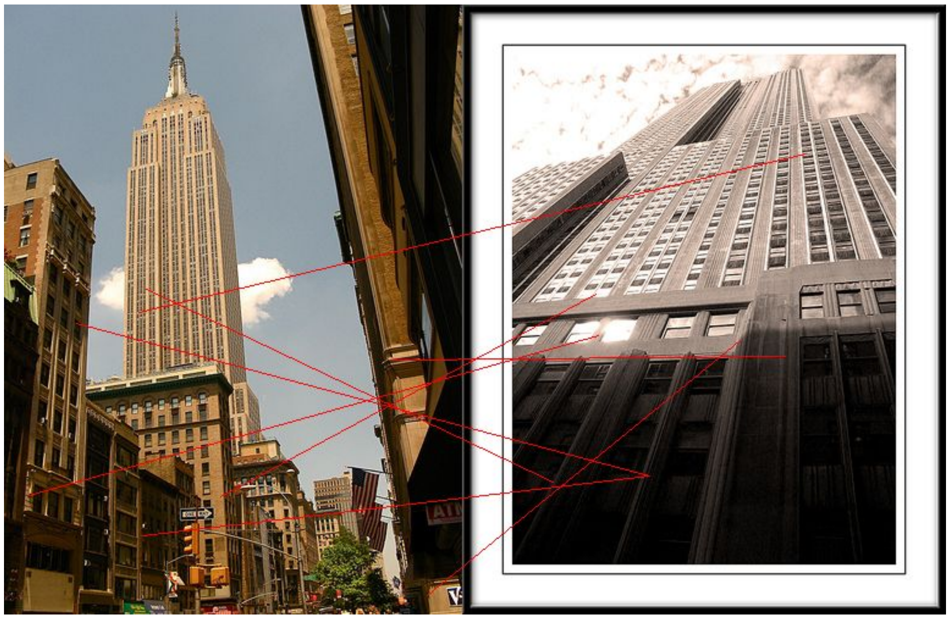
PART 1 : IMAGE MATCHING

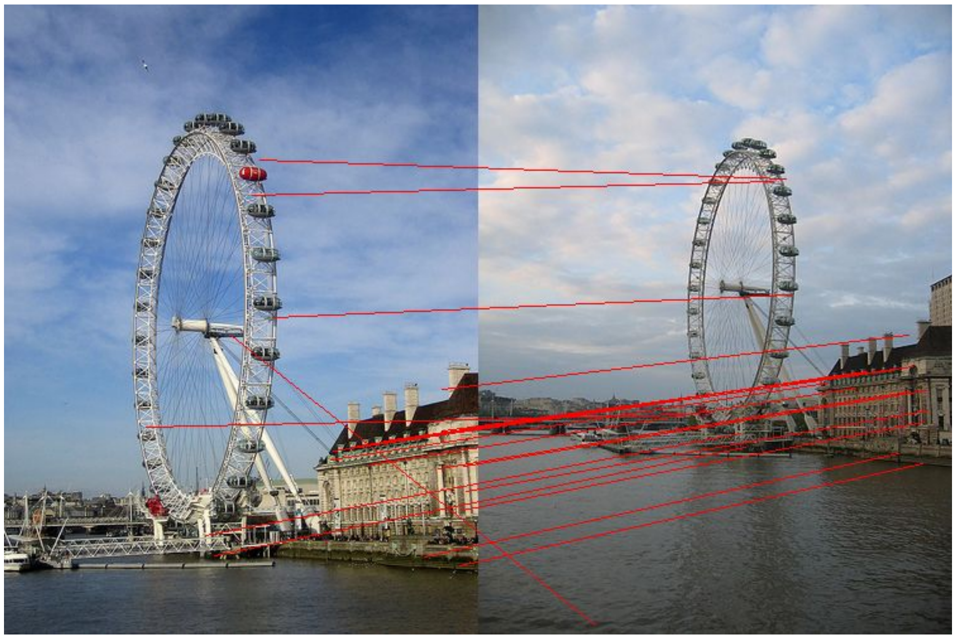
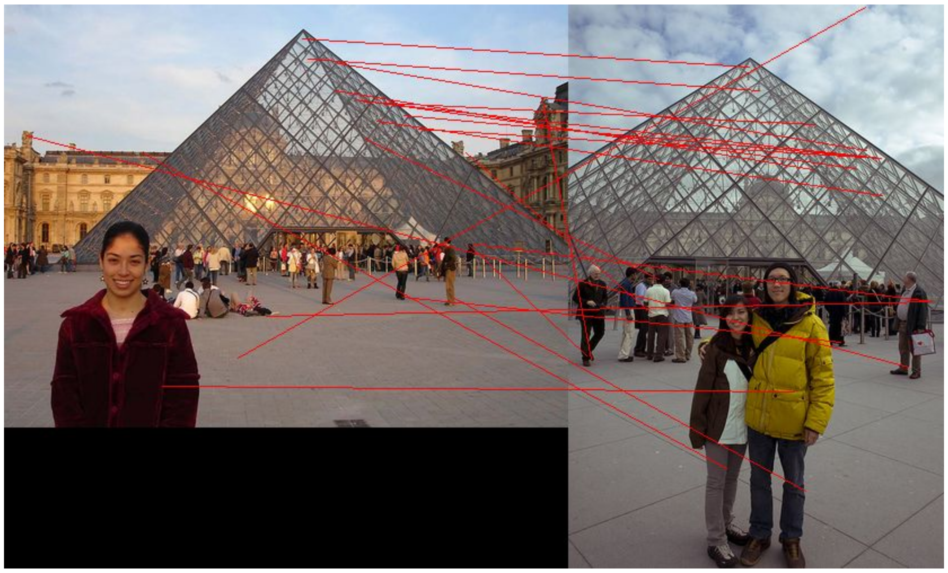
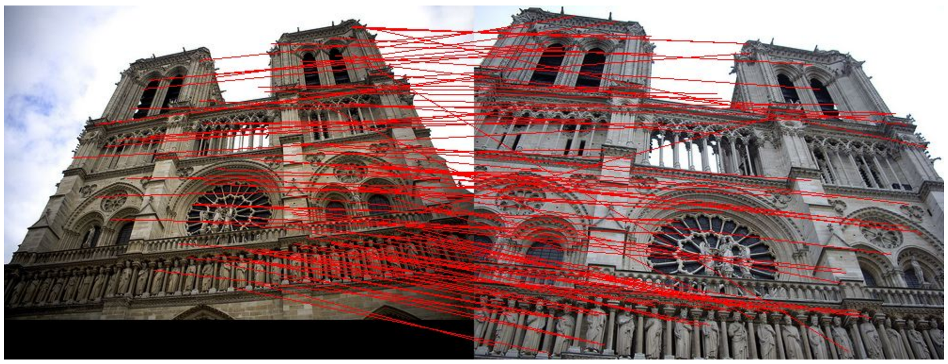
1. To match the two images, we calculate the descriptors of the two images and call a function findMatches. Inside findMatches, we start iterating on the descriptors of image 1, and pass the ith descriptor and all the descriptors of image 2 to another function findNearestNeighbour, which compares the euclidean distance of image 1 descriptor to all the descriptors in image 2, and based on a certain threshold, returns the index of the nearest neighbour to descriptor of image 1.

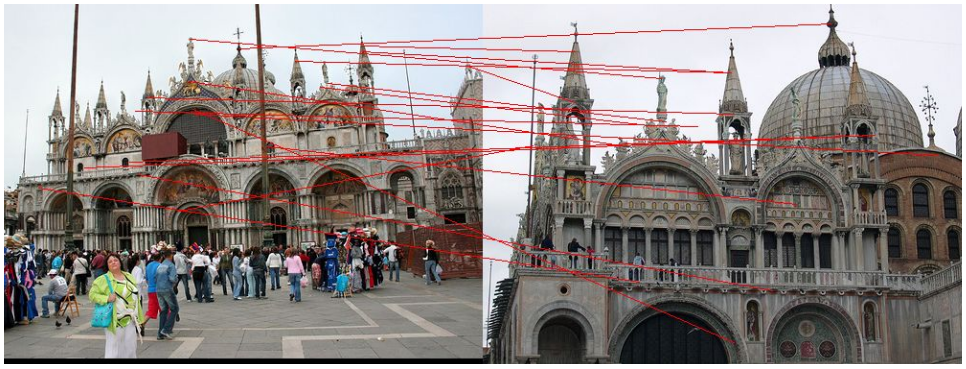
   Then we draw lines between the matching descriptors in image 1 and image 2.
   To run this part, use the below command:
   ./a2 part1 image1.jpg image2.jpg

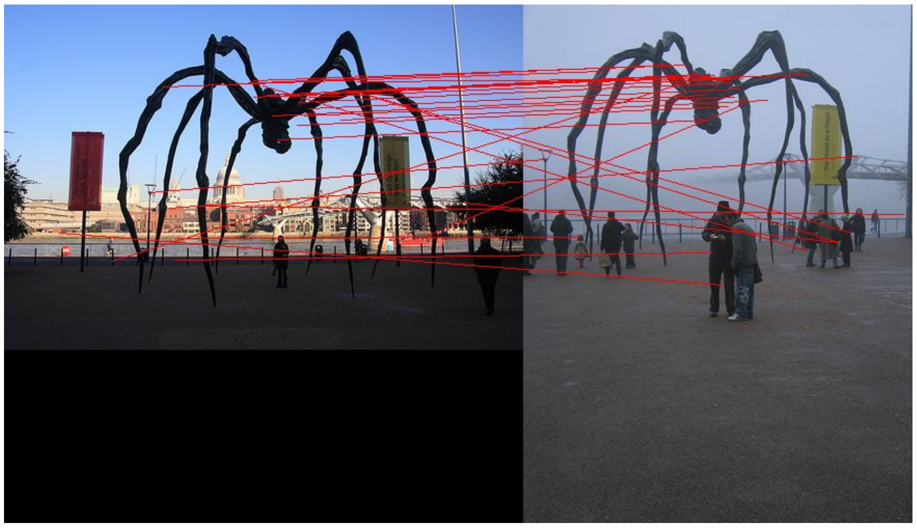


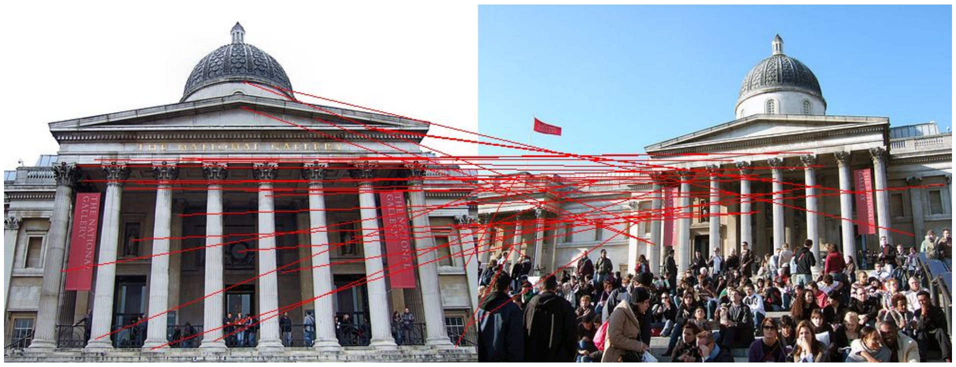

© 2006 - Derick Rethans - All rights reserved









2. Image Retrieval Program:

If there are more than two images passed, we need to rank the images in decreasing order of the number of matched features. We create a map called countMap<int, vector<string>> which stores the count of the matched features between the query image and one of the passed images, and the name of the corresponding image. If two images return the same count, then we access the value corresponding to that count and append the current image name to the vector.

To get the count of matched features, we call a method findBestMatchForImage which gets the count using the method findMatches. We increment the count in the method findMatches if the index of the nearest neighbor returned by the method findNearestNeighbour is not equal to -1. We then display the images in decreasing order of matched features.

To run this part, use the below command:

```
./a2 part1 query_image.jpg image2.jpg image3.jpg …… imagen.jpg
```

3. Testing the image retrieval program:

We test the above image retrieval program on all the 10 attractions provided and calculate the precision of the program.
We used the below command to test the precision, we used each of the below images as a random image and tested on all the remaining images, we got the precision as indicated in front of each image name, and later we calculated the cumulative precision for each attraction as indicated below each attraction. Based on our results, we found that **Big Ben** gives the highest precision and is easiest to recognize, whereas **Empire State** gives us the lowest precision and is the most difficult to recognize.

londoneye_21 0.2
londoneye_12 0.2
londoneye_16 0.1
londoneye_23 0.3
londoneye_12 0.2
londoneye_17 0.2
londoneye_23 0.4
londoneye_9 0.4
londoneye_8 0.1

londoneye = 0.233

bigben_10 0.1
bigben_12 0.3
bigben_13 0.2
bigben_14 0.6
bigben_16 0.3
bigben_10 0.5
bigben_2 0.5
bigben_3 0.6
bigben_6 0.5

bigben = 0.4

tatemodern_11 0.2
tatemodern_14 0.5
tatemodern_16 0.2
tatemodern_13 0.2
tatemodern_4 0.2
tatemodern_8 0.2

tatemodern = 0.25

colosseum_11 0.1
colosseum_13 0.1
colosseum_18 0.1
colosseum_4 0.3
colosseum_8 0.2
colosseum_15 0.3

colosseum = 0.18

eiffel_15 0.1
eiffel_18 0.2
eiffel_22 0.2
eiffel_7 0.2
eiffel_6 0.1

eiffel = 0.16

notredame_14 0.6
notredame_25 0.1
notredame_4 0.4
notredame_19 0.3
notredame_8 0.2

notredame = 0.32

trafalgarsquare_16 0.4
trafalgarsquare_1 0.4
trafalgarsquare_22 0.3
trafalgarsquare_25 0.1

trafalgarsquare= 0.3

empirestate_10 0.1

empirestate_16 0.1
empirestate_27 0.1
empirestate_9 0

empirestate= 0.075

louvre_10 0.2
louvre_13 0.1
louvre_16 0.4
louvre_11 0.1

louvre = 0.2

sanmarco_13 0.1
sanmarco_18 0.1
sanmarco_4 0.3
sanmarco_19 0.2

sanmarco= 0.175

```
./a2 part1 query_image.jpg path/*.jpg
```

We used the below results to find the precision of our retrieval program:

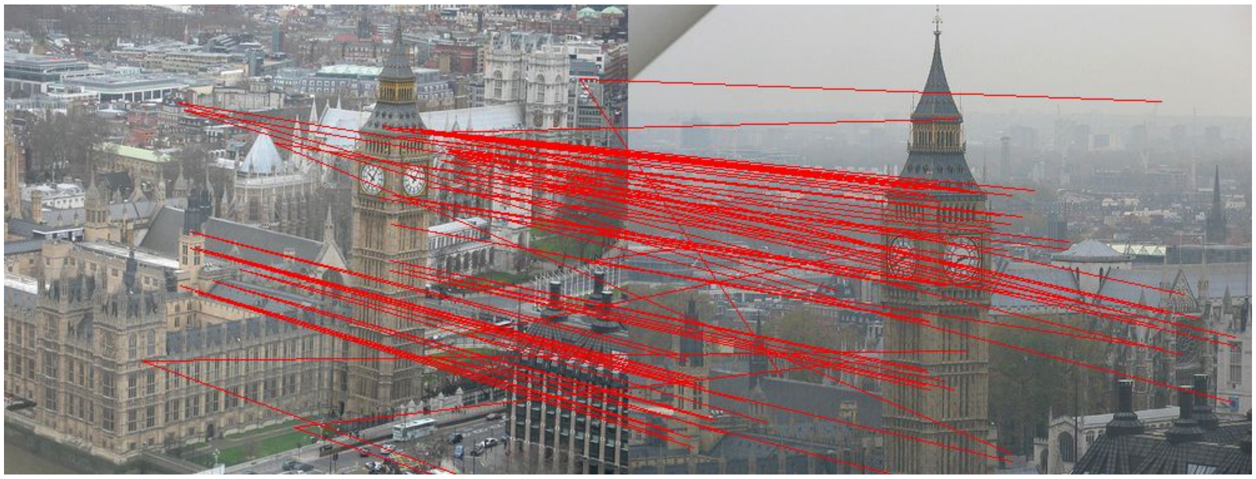
Part 2: Estimating Homographies and Speeding Things Up

1. The approximation has produced much cleaner correspondences as compared to that from the correspondences produced by the question 1. The inliers are shown in the figure below which is the output of the RANSAC algorithm.

The precision also improved as compared to the precision in the 1st question's part3. The precision is almost the same as compared to that achieved in the 1st question. We did not find any difference in the precision using the RANSAC algorithm. Its identical to the one found in 1st question where **Big Ben** has the highest precision and **Empire State** has the lowest precision.

To run this algorithm, following commands are used:

```
./a2 part2 image1.jpg image2.jpg
```



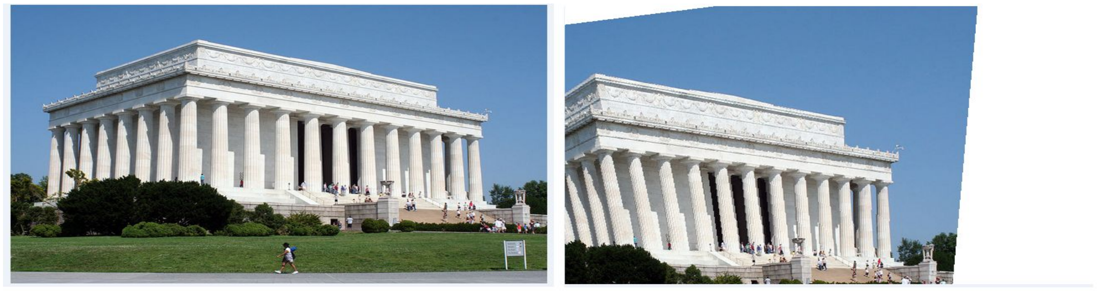
2. After using the quantized projection function, the algorithm speeds up as compared to in the part above where RANSAC was implemented. The average time calculated when the RANSAC algorithm was run was approximately 3.6 seconds and the average time calculated when the quantized projection function was implemented was approximately 2.7 seconds.
So, after using the quantized projection function, the image retrieval speeds up by almost 1 second.
The results that we are getting after the approximation are deteriorated as compared to the results that we were getting from the first question.
The best trade off between running time and accuracy can be achieved by adjusting w, k and number of trials.
If we make k relatively very small (say 3), then the quality becomes really poor and a value of k around 20 gives somewhat enough dimensions to differentiate between various descriptors.
Also, if w is fairly large (say 50), it narrows down the dimensions and the floor function makes a lot of descriptors similar which defeats our purpose of quantization. Hence, w should be relatively small so as to achieve difference between various descriptors and hence, the usefulness of quantization can be seen.
To run the program using the quantized projection function, run the following command:

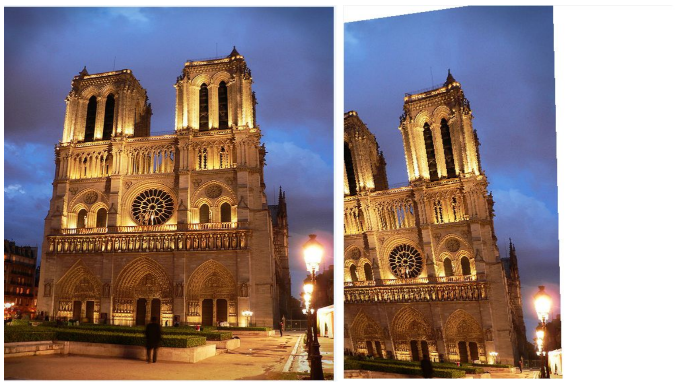
```
./a2 part2.2 image1.jpg image2.jpg
```

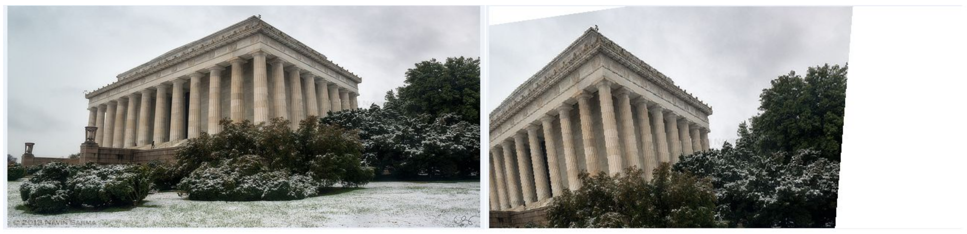
<u>Part 3 : Image Warping</u>

1) We applied the given transformation matrix on some sample images as shown below.
   To run the program using the quantized projection function, run the following command:

```
./a2 part3 image1.jpg
```







2) We computed homography between the first input image and the each of the image in the sequence. The homography matrix is used as the the transformation matrix to project the sequence of images onto the projection plane of the first image.
Below are some sample examples with first image as the query image and the sequence of images with their warped images.
To run the program using the quantized projection function, run the following command:

```
./a2 part3 image1.jpg image2.jpg ...imagen.jpg
```

Query image: