Assignment 1: Image Mosaic

Nursultan Jubatyrov

Introduction

Photo mosaic is a picture that have been done using other photos of the small sizes. When the final result is viewed it is actually similar to the initial target image and when it is viewed at low magnifications, the individual small pictures appear as a pixels of the photo.

Implementation

In order to implement this assignment I am using C++ and additional library called "*CImg*". *CImg* is a small, open-source, and modern C++ toolkit for image processing. It is actually one header file and it is easy to read/save, access pixel values of an image using it. The reason of C++ instead of Matlab is, coding on C++ is more convenient to me and it is faster in performance than the Matlab.

Solution

To construct the target image I used two ways. The first way is to calculate error using average values of an images and the second way is to calculate error using Euclidean distances between pixels.

Average values

After processing all images from the database. I calculated the average values of the red, green and blue colors of all images. Additionally, I preprocessed the average values of all patches of the target image. The next step is to find the most suitable image to the given patch. There are different ways to find them and the easiest one is to use brute force solution. We iterate through all images from the database and calculate an error of that image with the patch. Then we will choose a candidate with the minimum error. To calculate the performance of this solution let W and H will be the width and height of the target image respectively. Let k be the size of the patch and N be the number of the images in database. Then, the time complexity of the program will be $O(N * \frac{W*H}{k^2})$.

Euclidean distances

This method is similar to the previous one. Instead of calculating an average error of the patch, here we calculate error between each pixel. The easiest way finding the error is to calculate Euclidean norm. The final result using this method is more accurate than using average values, however the time performance will decrease since we need to compare each pixel. Again, if we are given W, H, k, N. Then the performance is $O(N * W * H)$.

Here is the table of the comparison between this two methods.

Method	Target image size	Patch Size	Number of picture	Time in seconds
Average values	920x1420	20	926	22.83 sec
Euclidean distance	920x1420	20	926	122.54 sec

Another patch shape

Additionally, instead of using square patch I changed it to be a triangle. I divided the square images into four triangles. Then in a similar way I calculated the averages of each triangle of each candidate image. Then by choosing random triangles I tried to match them.

Conclusion

For me, the most difficult part was to get used to represent, read/save images in *CImg* library. Other parts are quite easy. Even I implement the task, the final results are not perfect. This is because my algorithm uses the same pictures several times making the photo redundant. Anyway, it was fun to me to do this assignment.