UNSW

COMP9517
Report of Assignment 1
Student Name: Chunnan Sheng
Student Code: z5100764

All three tasks of this assignment are implemented via **C++** programming language together with **OpenCV** which is an open source library of Computer Vision.

1. The source code includes the following files:

File name	Description
OpenImage.cpp	Main function of this program
RGBSolver.cpp RGBSolver.h	This source file includes functionalities of task1, task2, and task3
SCNImage.cpp SCNImage.h	This source file includes methods of template matching, pyramid matching, combinations of RGB channels and final adjustment of images.
SubImgBlock.cpp SubImgBlock.h	A SubImgBlock shares the same data with an instance of SCNImage but behaves like a sub image of the original image. This is used for image matching.
SCNPixel.cpp SCNPixel.h	A pixel of 8bit unsigned digits of RGB color
MatchWindow.cpp MatchWindow.h	This class stores temporary coordinate information of matched location of another color channel in the original image.

2. Task1

This program tries to implement task1 in the following way:

A. Randomly select a location on the image where the variation of colors is high enough for matching. The size of the matching window can be determined by the second argument of the command line. Variation of colors is calculated this way:

$$Variation = mean\{[mean(f(x,y)) - f(x,y)]^2\}$$

B. Search all over the image to find a second and third place that is the closest to the first matching window. The distance of each step is 1/5 the size of the window.

C. Continue to search for the accurate location with offset range of -1/5 to 1/5 from the center of the matched window. The distance between the two sub image windows is calculated this way:

$$Distance = mean\left\{\left[f(x,y) - g(x',y') - mean\big(f(x,y)\big) + mean\big(g(x',y')\big)\right]^2\right\}$$

Someone may simply calculate the distance this way:

$$Distance = mean\{[f(x,y) - g(x',y')]^2\}$$

This way may not work well since the average brightness of the windows to match would be quite different. It is essential to neutralize the differences of mean color prior calculating the square error between the two windows.

3. Task2

Sometimes it is time-consuming to do a match for large resolution images. A good idea to solve this problem is to transform the large images into smaller ones, and then enlarge small images gradually. For example, if (x, y) and (x', y') are two matched coordinates on the smaller image which is half size of the original one, then we can find $(2x, 2y)$ and $(2x', 2y')$ on the larger image. ± 2 pixels

around $(2x', 2y')$ would continue to be evaluated until an accurate coordinate of the highest match is found.

The size of matching window would be 1.5 times the previous match while the image is enlarged each time.

4. Task3

The last task is to combine the three channels of R, G and B figured out by matching algorithms in Task1 and Task2. Borders of images are chopped off, and the algorithm of Histogram Equalization is implemented in the final adjustment of images.

5. Example

This program is executed via three arguments: ***file name of the original image***, the ***size of matching window*** and ***the smallest image size of the image pyramid***. If the smallest image size is roughly the size of the original image, no image pyramid is applied for matching algorithms.