COMP-SCI 201L (FS17) - **Section 2**

Lab 9 (11/07/17)

Remove the Pesky Tourist

Task Description

One day, you visited UMKC Volker Campus and wanted to take some photos of the sculpture in front of the Flarsheim Hall. Unfortunately, Prof. Brian Hare kept wandering around and would not get out of the way. Eventually, you had to give up and leave.

However, when you looked at the pictures later, you realized that **the tourist were never standing in the same part of the shot**. If you could use the unobstructed parts of the image to cover him up, you could effectively remove the tourist from the image. Therefore, in today's Lab, you task is to generate a **clean** photo of the sculpture with the peaky tourist removed (See Figure 1).

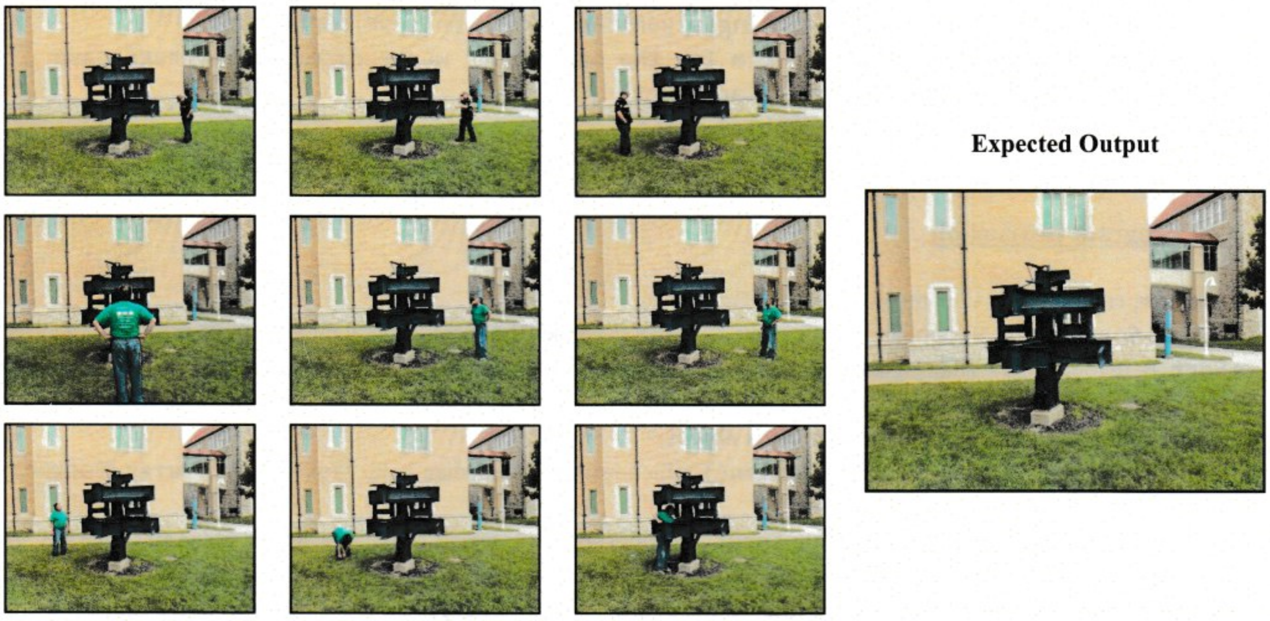

Figure 1. From the left nine pictures with tourist, generate a clean photo of the sculpture with no tourist.

Image in PPM Format

You are given a collection of image files in **PPM** format. This is a lowest-common-denominator format in which **the image data is stored in a text file**. The format is as follows:

```
P6
800 600
255
N
N
N
...etc...
```

The "P6" is a magic constant identifying this as a PPM color image. The next line gives the dimensions of the image, which is **800 pixels per row and 600 rows**. The next line gives the color depth; in this image, the red, green, and blue values range from 0 to 255. Then the pixel data follows, which is in the row-major order: the red value of the first pixel, green value of the first pixel, blue value of the first pixel, red value of the second pixel, green value of the second pixel, and so on. All numeric values are decimal-coded ASCII. All values are separated by whitespace. Please note that the PPM standard does **not** require line breaks any place in particular (or at all); spaces, tabs, or newlines, in any combination, can separate items.

The following link is a fast online PPM image viewer. You can use it to see your PPM files as images. The link is also posted in Blackboard.

http://paulcuth.me.uk/netpbm-viewer

Sorting Algorithms

In this Lab, you are using **sorting** to find out the **median** (**not average!**) value of the color for each pixel. You have learned many sorting algorithms in your lecture, e.g. selection sort, insertion sort, merge sort, and so on. **In this Lab, you can use any sorting algorithm you like**. However, in your code documentation (comment), I would like to see the following information.

```
/** In your code documentation (comment), include the following information.
    1. Sorting algorithm you used in this Lab
    2. Briefly describe how your sorting algorithm works (in your own words)
    3. Big-O notation of your algorithm for the best case, worst case, and average case
    4. Is your sorting algorithm in-place sort?
    5. Is your sorting algorithm stable?
*/
```

`Main()` Program Instruction

1. Start a new, empty Visual C++ project in MS Visual Studio.
2. Download `Sort.h`, `Lab_9.cpp`, and all the input `PPM` files from the Blackboard. Load the files in your project.
3. In `Sort.h`, implement a sorting algorithm (choose any sorting algorithm you like) which sorts an array (`vector`) of integers in **non-decreasing order**. Do **not** forget to include the required information in your code comment. **You will get zero for this part if the required code documentation is absent**.
4. In `Sort.h`, implement a function called **`Median()`** which returns the median value from a **sorted** **array** (`vector`) of integers.
5. The `main()` program first reads a **text file** `input_filenames.txt`, which contains **a list of filenames**, one per line. Those files are the PPM images you are going to combine into a "cleaned" image. Your output file will be a **"cleaned" PPM** in which you have applied a **median filter** to the image. With a median filter, pixel values from multiple images are compared, and the median is used as the final value. This is applied separately for each color value (red, green, blue) for each pixel. If you view the output file in the PPM viewer, it should be the sculpture with the tourist removed.
6. Consider your program logic carefully. **PPM files are large**, and your program may have to deal with several of them. However, it is not necessary to have all the data for all the files in memory at once. Since you do **not** know exactly how many images your program might have to deal with, you should probably use a dynamic structure to hold them; perhaps a `vector`.

Lab Submission and Grading

1. Submit your work **no later than Thursday, Nov 9th, 2017, by 11:59 PM**.
2. Submit two files only, `Sort.h` and `Lab_9.cpp` (**unzipped**).
3. Your code may be graded using another set of PPM pictures to see if it can generate the expected output. The grader will also look at your code manually to see if it meets our other requirements.
4. Although the Lab instructor may give you extra days to complete the Lab assignment, the Lab is designed to be completed within the period of the Lab session. Therefore, finishing the Lab within the Lab period is highly expected. **Unless you complete the Lab assignment and submit your work via Blackboard, you cannot leave the Lab early!** You will get zero for the Lab if you leave the Lab early, without the submission of your code, nor an acceptable reason.