Image Processing Using C/C++

1 - Introduction

In this project, you will be doing some image processing using C/C++. You will be required to implement the following image processing operations.

- *Negation*
- *Image Thresholding*
- *Histogram Stretching*
- *Median Filter*
- *Edge Detection*

2 - Background

2.1 - PGM Files

Images used in this project will be provided in PGM (Portable Gray Map) because of their ease of reading and writing. Code is provided to handle reading and writing of the PGM files where all image processing procedures will be done on images that have been loaded in memory of your program. The provided functions for reading/writing are written in a C-style format since more complex C++ methods have yet to be presented in the course.

A PGM file is in grayscale image format. Each file contains a header specifying information about the file followed by the image data itself. Below is a PGM file example (reference: http://netpbm.sourceforge.net/doc/pgm.html).

```
P2
# feep.pgm
24 3
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
```

One of the easiest ways to open and view a PGM file is to use an online PGM viewer, such as https://bytes.usc.edu/~saty/tools/PGMViewer/viewer.html.

2.2 - Negation

Image negation makes lighter areas appear darker and darker areas appear lighter. This is done by subtracting each pixel value in the original image from a constant, which is the maximum value that can be represented using given bits. In this project, each pixel is represented by 8 bits, so it can have a value from 0 (black) to 255 (white). Therefore, the negation formula is:

output_pixel_value = 255 - original_pixel_value.

2.3 - Thresholding

Image thresholding is a technique used to segment an image into distinct regions based on pixel intensity values. The process involves converting a grayscale image into a binary image, where each pixel is classified as either black or white depending on whether its pixel value is above or below a certain threshold. In this project, a threshold value of 127 is used. Therefore, the output pixel value is set to 255 (white) if the value of an input pixel is greater than or equal to 128; it is set to 0 (black) otherwise.

2.4 - Histogram Stretching

Histogram stretching improves the contrast of an image by expanding the range of pixel values. It detects the minimum pixel value (min) and the maximum pixel value (max) of an original image and converts the pixel range from [min, max] to [0, 255]. The formula for histogram stretching is:

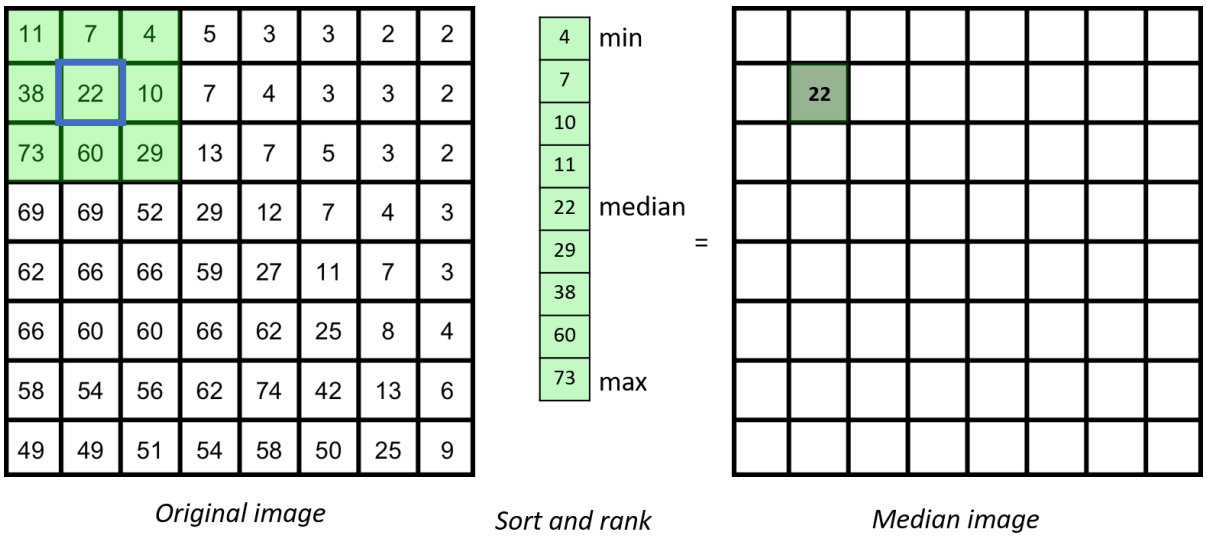
output_pixel_value = 255 x (original_pixel_value - min) / (max - min).

2.5 - Median Filter

The median filter is commonly used for noise reduction and highly effective at the removal of “salt and pepper” noise. The main concept of the median filter is to create a window of neighbors. Within a single window, the middle neighbor is replaced with the median value of all the neighbors in the window. The median filter is executed by sliding the window across the entire image.

For this project, you will be implementing a 3x3 median filter and applying it to an image. In particular, the filter window slides not only from left to right but also from top to bottom. **The boundaries of the resulting image (i.e., the first row, the last row, the first column, and the last column) are to be filled with the original values for simplicity.**

Figure 1 (image source: https://neubias.github.io/training-resources/median_filter/index.html) shows a 3x3 median filter example. A method for determining the median value in the window is to flatten the window into a 1D array, sort the values, and pick the middle value.



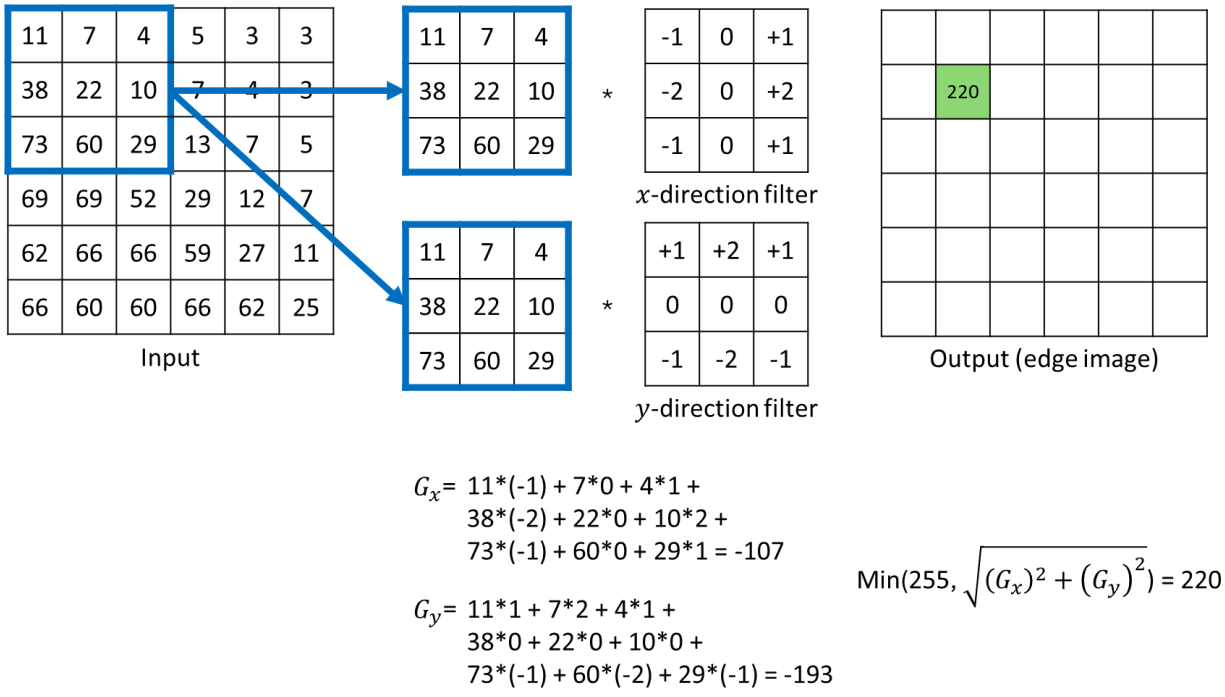
[**Figure 1**. Median filter example. The 3x3 window slides over the original image **by one pixel in a Z-order (from left to right; from top to bottom)**.]

2.6 - Edge Detection

Edge detection is used to identify edges within an image, which represent areas where there is a notable change in pixel intensity. This process is fundamental for various image processing tasks, such as object detection, image segmentation, and feature extraction.

In this project, the Sobel edge detection algorithm is used. It utilizes two 3x3 constant filters (also known as kernels) to detect edges: one for horizontal edges and the other for vertical edges. The algorithm works by convolving each filter with the image. As each filter slides over

the image, similar to the median filter, it computes a weighted sum of pixel values, G_x and G_y. To determine the overall edge strength, G_x and G_y are combined as $G = \sqrt{(G_x)^2 + (G_y)^2}$, which represents an edge intensity of a pixel. **To avoid overflow, if the G value exceeds 255, it is capped at 255**. Note that applying thresholding improves the quality of the resulting edge image, but it is not used in this project for the sake of simplicity. An example of this process is shown in Figure 2.



[**Figure 2**. Edge detection example. The 3x3 window slides over the original image **by one pixel in a Z-order (from left to right; from top to bottom)**.]

For this project, the boundaries of the resulting image (i.e., the first row, the last row, the first column, and the last column) are to be filled with the original values for simplicity. The 3x3 filter window slides not only from left to right but also from top to bottom. The filters you will be using for this project are the same as those shown in Figure 2. These filters are already implemented in the edgeDetect() function.

You can use the sqrt() function from the cmath standard library, which returns a double-typed value. To convert the result into an int-typed value, you can use the following syntax:
“int-typed variable = static_cast<int>(sqrt() result);”.

3 - Tasks

Implement the **imgNegation**(), **thresholding**(), **histogramStretching**(), **medianFilter**(), and **edgeDetect**() functions in the ImageProcessing.cpp file.

- You can find information about the parameters and return data of these functions in the ImageProcessing.h file.
- **To match the precision of golden images with your resulting images, do not use the *double* and *float* data types (use the *int* data type).**
- PGM.cpp and PGM.h are used to read and write PGM files. There is nothing to do with these files.
- main.cpp receives the name of an input image, the type of operation (negation, thresholding, histogram stretching, median filter, or edge detection), and the name of an output image. There is nothing to do with this file.
- You can define additional functions in ImageProcessing.cpp if you want.

4 - Compile and Test

```
g++ -o main *.cpp *.h
./main
```

- ***Negation***
    - Inputs to the program

```
Enter Original File Name: input_images/cameraman.pgm
…
Enter Selection: 0
…
Enter Save File Name: output_images/negation.pgm
```

    - Original and resulting images

- ***Thresholding***
    - Inputs to the program

```
Enter Original File Name: input_images/cameraman.pgm
…
Enter Selection: 1
…
Enter Save File Name: output_images/thresholding.pgm
```

    - Original and resulting images

- ***Histogram Stretching***
    - Inputs to the program

```
Enter Original File Name: input_images/balloons.pgm
…
Enter Selection: 2
…
Enter Save File Name: output_images/stretching.pgm
```

    - Original and resulting images

- ***Median Filter***
    - Inputs to the program

```
Enter Original File Name: input_images/noisy.pgm
…
Enter Selection: 3
…
Enter Save File Name: output_images/median.pgm
```

    - Original and resulting images

- ***Edge Detection***
    - Inputs to the program

```
Enter Original File Name: input_images/cameraman.pgm
…
Enter Selection: 4
…
Enter Save File Name: output_images/edgedetect.pgm
```

    - Original and resulting images

- ***Check all together***

```
chmod 777 run.sh
./run.sh
```

You must see the following messages:

```
IMAGE NEGATION - PASS
IMAGE THRESHOLDING - PASS
HISTOGRAM STRETCHING - PASS
MEDIAN FILTER - PASS
EDGE DETECTION - PASS
```

5 - Submit

- Please upload your **ImageProcessing.cpp** file to **myCourses > Assignments > Project1**.
- Do not modify other files or create new files so that the grader can easily grade your submissions.
- Do not submit *.docx, *.pdf, *.txt, or *.zip file.

6 - Grading

- [90 pts] - Functions to be implemented
    - [10 pts] imgNegation()
    - [10 pts] thresholding()
    - [15 pts] histogramStretching()
    - [25 pts] medianFilter()
    - [30 pts] edgeDetect()
- [10 pts] - Program execution
    - **The program execution portion of the grade is ALL or nothing.**