

Student: Michael Grossman

Project Due Date: 02/13/2022

Algorithm Steps for Computing the Histogram Given an Input-File-Stream:

1. For Each Pixel in the file-stream:
2. $Val \leftarrow \text{Pixel}$
3. $\text{histArray}[Val]++$
4. End-Loop

Algorithm Steps for Creating the Threshold Images Given a Threshold-Value, an Input File-Stream, and Two Output File-Streams:

1. $\text{minVal} \leftarrow 0$
2. $\text{maxVal} \leftarrow 1$
3. Output-File-Stream1, Output-File-Stream2 \leftarrow output numRows, numCols, minVal, and maxVal
4. For Each Pixel in the Input-File-Stream:
5. $\text{PixelValue} \leftarrow \text{Pixel}$
6. If $\text{PixelValue} \geq \text{Threshold-Value}$:
7. Output-File-Stream1, Output-File-Stream2 \leftarrow "1 "
8. Else:
9. Output-File-Stream1 \leftarrow "0 "
10. Output-File-Stream2 \leftarrow ". "
11. End If-Else
12. Output-File-Stream1, Output-File-Stream2 \leftarrow "\n"
13. End-Loop

Main.cpp

```
#include <iostream>
#include <fstream>

/*
Program Specs:
1. Compute histogram of the input image and display the histogram in two
formats, see the output description below.
2. Perform binary threshold operation on the input image with a given
threshold value via argv[].
3. Output the result of the threshold in two formats, see the output
description below
*/

class Image{
public:

    //constructor
    Image(int nr, int nc, int mnv, int mxv, int tv);

    //destructor
    ~Image();

    //attributes
    int numRows;
    int numCols;
    int minVal;
    int maxVal;
    int* histArray;
    int thresholdVal;

    //functions
    void computeHist(std::ifstream& input);
    void printHist(std::ofstream& output);
    void dispHist(std::ofstream& output);
    void threshold(std::ifstream& input,
                  std::ofstream& output1,
                  std::ofstream& output2, int tv);
};

int main(int argc, char** argv){
```

```
if(argc > 7 || argc < 7){
    std::cout << "incorrect command line args - must always"
               << "be in the format: " << std::endl;
    std::cout << "./program file_name threshold_value "
               << "output_file1 output_file2 output_file3 "
               << "output_file_4" << std::endl;
    return 0;
}

//store command line args
std::string in_file_name = argv[1];
int threshold_value = atoi(argv[2]);
std::string out_file_names[4] = {argv[3], argv[4], argv[5], argv[6]};

//open the files
std::ifstream fin(in_file_name);
std::ofstream fout1(out_file_names[0]), fout2(out_file_names[1]),
                 fout3(out_file_names[2]), fout4(out_file_names[3]);

//get the image's header info
int image_header_info[4];
for(int i = 0; i < 4; ++i){
    fin >> image_header_info[i];
}

//apparently not allowed to use the getline / stringstream method,
//leaving here in case of errors
/*
std::string current_line;
std::getline(fin, current_line);
int image_header_info[4];
std::string working_string;
std::stringstream sstream(current_line);
for(int i = 0; i < 4; ++i){
    std::getline(sstream, working_string, ' ');
    image_header_info[i] = stoi(working_string);
}
*/

//create image object
Image image(image_header_info[0], image_header_info[1],
            image_header_info[2], image_header_info[3],
            threshold_value);
```

```
//compute the histogram and print out to files
image.computeHist(fin);
image.printHist(fout1);
image.dispHist(fout2);
fin.close();

//reopen input file
fin.open(in_file_name);

//output new thresholded image
fout3 << "The threshold value uses is " << threshold_value << "\n";
fout4 << "The threshold value uses is " << threshold_value << "\n";
image.threshold(fin, fout3, fout4, threshold_value);

//close all streams
fin.close();
fout1.close();
fout2.close();
fout3.close();
fout4.close();

return 0;
}

Image::Image(int nr, int nc, int mnv, int mxv, int tv) : numRows(nr),
    numCols(nc), minVal(mnv), maxVal(mxv), thresholdVal(tv)
{
    int arr_end = mxv + 1;
    histArray = new int[arr_end];
    for(int i = 0; i < arr_end; ++i){
        histArray[i] = 0;
    }
}

Image::~Image(){
    delete[] histArray;
}

void Image::computeHist(std::ifstream& input){

    int total_pixels = numRows * numCols, working_int = 0;
    for(int i = 0; i < total_pixels; ++i){
        input >> working_int;
        histArray[working_int]++;
    }
}
```

```
}

//apparently we cannot use the getline / stringstream method,
//leaving here in case of errors
/*
std::string working_str;
std::stringstream working_stream;
while(!input.eof()){
    std::getline(input, working_str);
    working_stream = std::stringstream(working_str);
    while(!working_stream.eof()){
        std::getline(working_stream, working_str, ' ');
        if(working_str[0] < '0' || working_str[0] > '9') break;
        histArray[std::stoi(working_str)]++;
    }
}
*/
}

void Image::printHist(std::ofstream& output){
    output << numRows << " " << numCols << " " << minVal
        << " " << maxVal << "\n";
    int arr_end = maxVal + 1;
    for(int i = 0; i < arr_end; ++i){
        output << i << " " << histArray[i] << "\n";
    }
}

void Image::dispHist(std::ofstream& output){
    output << numRows << " " << numCols << " " << minVal
        << " " << maxVal << "\n";
    int arr_end = maxVal + 1;
    int max = 0;
    for(int i = 0; i < arr_end; ++i){
        output << i << " (" << histArray[i] << "):";
        max = histArray[i] > 70 ? 70 : histArray[i];
        for(int j = 0; j < max; ++j){
            output << "+";
        }
        output << "\n";
    }
}
```

```
void Image::threshold(std::ifstream& input, std::ofstream& output1,
                     std::ofstream& output2, int tv){
    minVal = 0;
    maxVal = 1;
    int pixelVal = 0;
    output1 << numRows << " " << numCols << " " << minVal
              << " " << maxVal << "\n";
    output2 << numRows << " " << numCols << " " << minVal
              << " " << maxVal << "\n";

    //move passed the header
    for(int i = 0; i < 4; ++i){
        input >> pixelVal;
    }

    //read in the file and output depending on whether the value
    //meets the given threshold tv
    for(int i = 0; i < numRows; ++i){
        for(int j = 0; j < numCols; ++j){
            input >> pixelVal;
            if(pixelVal >= tv){
                output1 << "1 ";
                output2 << "1 ";
            }
            else{
                output1 << "0 ";
                output2 << ". ";
            }
        }
        output1 << "\n";
        output2 << "\n";
    }

    /* apparently we cant use the getline/stringstream method,
    //leaving here in case of errors
    std::string working_str;
    std::stringstream working_stream;
    std::getline(input, working_str);
    while(!input.eof()){
        std::getline(input, working_str);
        working_stream = std::stringstream(working_str);
```

```
while(!working_stream.eof()){
    std::getline(working_stream, working_str, ' ');
    if(working_str[0] < '0' || working_str[0] > '9') break;
    pixelVal = std::stoi(working_str);
    if(pixelVal >= tv){
        output1 << "1 ";
        output2 << "1 ";
    }
    else{
        output1 << "0 ";
        output2 << ". ";
    }
}
output1 << "\n";
output2 << "\n";
}
*/
}
```

Output outFile1 for data 1

```
31 40 0 9
0 309
1 288
2 194
3 64
4 0
5 2
6 12
7 106
8 124
9 141
```

Output outFile2 for data 1

```
31 40 0 9
0 (309):+++++
1 (288):+++++
2 (194):+++++
3 (64):+++++
4 (0):
5 (2):++
6 (12):+++++
7 (106):+++++
8 (124):+++++
9 (141):+++++
```


Output outFile3 for data 1

The threshold value uses is 6

31 40 0 1

[illegible]



Output outFile1 for data 2

```
45 45 1 63
```

```
0 0
```

```
1 180
```

```
2 190
```

```
3 188
```

```
4 190
```

```
5 180
```

```
6 0
```

```
7 0
```

```
8 0
```

```
9 0
```

```
10 0
```

```
11 0
```

```
12 0
```

```
13 0
```

```
14 0
```

```
15 0
```

```
16 0
```

```
17 0
```

```
18 0
```

```
19 0
```

```
20 0
```

```
21 0
```

```
22 0
```

```
23 0
```

```
24 0
```

```
25 0
```

```
26 0
```

```
27 0
```

```
28 0
```

```
29 0
```

```
30 1
```

```
31 224
```

```
32 215
```

```
33 216
```

```
34 215
```

```
35 226
```

```
37 55 220
```

```
38 36 0
```

```
39 37 0
```

```
40 38 0
```

```
41 39 0
```

```
42 40 0
```

```
43 41 0
```

```
44 42 0
```

```
45 43 0
```

```
46 44 0
```

```
47 45 0
```

```
48 46 0
```

```
49 47 0
```

```
50 48 0
```

```
51 49 0
```

```
52 50 0
```

```
53 51 0
```

```
54 52 0
```

```
55 53 0
```

```
56 54 0
```

```
57 55 0
```

```
58 56 0
```

```
59 57 0
```

```
60 58 0
```

```
61 59 0
```

```
62 60 0
```

```
63 61 0
```

```
64 62 0
```

```
65 63 0
```

```
66
```

Output outFile2 for data 2

```
45 45 1 63
0 (0):
1 (180):+++++
2 (190):+++++
3 (188):+++++
4 (190):+++++
5 (180):+++++
6 (0):
7 (0):
8 (0):
9 (0):
10 (0):
11 (0):
12 (0):
13 (0):
14 (0):
15 (0):
16 (0):
17 (0):
18 (0):
19 (0):
20 (0):
21 (0):
22 (0):
23 (0):
24 (0):
25 (0):
26 (0):
27 (0):
28 (0):
29 (0):
30 (1):+
31 (224):+++++
32 (215):+++++
33 (216):+++++
34 (215):+++++
35 (226):+++++
```

continued on next page

```
37 35 (226):
38 36 (0):
39 37 (0):
40 38 (0):
41 39 (0):
42 40 (0):
43 41 (0):
44 42 (0):
45 43 (0):
46 44 (0):
47 45 (0):
48 46 (0):
49 47 (0):
50 48 (0):
51 49 (0):
52 50 (0):
53 51 (0):
54 52 (0):
55 53 (0):
56 54 (0):
57 55 (0):
58 56 (0):
59 57 (0):
60 58 (0):
61 59 (0):
62 60 (0):
63 61 (0):
64 62 (0):
65 63 (0):
66
```

[illegible]

