

Student: Trisha Espejo

Project Due Date: 2/14/2020

Computing The Histogram steps:

Step 0: Obtain header from the input file and place into output file

Step 1: allocate memory for an array to store pixel and create a counter

Ex: `pixel[i][j];`

Step 2: Create a double for loop which would iterate through each pixel

Step 3: in the most inner for loop obtain the value from input file and place it to the pixel array

Ex: `Input >> pixel[i][j];`

Step 4: use the value of the pixel array obtain from Step 3 for index of histogram array and add the histogram

Ex: `histArray[pixel[i][j]]++;`

Step 4 delete the allocated array for pixel from the heap

Step 5 print the histogram using the values obtain from hisArray

Step 6 delete the allocated array for hisArray from the heap

Performing Threshold Operation:

Threshold (input, output, thrVal)

Step 0: obtain header from input file.

Step 1: replace the min value with 0 and max with 1

Ex: `minVal = 0` and `maxVal = 1`

Step 2: place the new header into the output file

Step 3: allocate memory for an array to store pixel and create a counter

Ex: `pixel[i][j];`

Step 4: Create a double for loop which would iterate through each pixel

Step 5: in the most inner for loop obtain the value from input file and place it to the pixel array

Ex: `Input >> pixel[i][j];`

Step 6: if the threshold value less than or equal to pixel value place 1 into the Output file else place 0 in output file

Ex: `If (thrVal <= pixel[i][j]) output << "1";`
`else output << "0"`

Step 7: delete the allocated array for pixel from the heap

Trisha Espejo

SOURCE CODE:

```
/*
    Computer Vision
    Histogram
    Author : Trisha Espejo
    Due Date: 2/14/2020
    Project 1
*/
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>

using namespace std;

class Image{

public :
    int numRows = -1, numCols= -1, minVal = -1, maxVal = -1;
    int *hisArray;

    int thresholdValue = -1;

public:
    Image(ifstream & input){
        read_header(input);

        this -> hisArray = new int[this->maxVal + 1];
        for (int i = 0; i < this -> maxVal + 1; ++i){
            hisArray[i] = 0;
        }
    }

    void read_header(ifstream & input)
    {
        input >> this->numRows >> this->numCols >> this->minVal >>
this->maxVal;
    }

    void computeHist (ifstream & input){

        int **pixel;
        int pixelValue = 0;

        pixel = new int*[this -> numRows];
        for (int i = 0; i < this -> numRows; ++i){
            pixel[i] = new int[ this -> numCols ];
        }
    }
}
```

```

        for (int j = 0; j < this->numCols; ++j)
            pixel[i][j] = 0;
    }

    for (int i = 0; i < this->numRows; ++i){
        for (int j = 0; j < this->numCols; ++j)
        {
            input >> pixel[i][j];
            pixelValue = pixel[i][j];
            hisArray[pixelValue]++;
        }
    }

    for (int i = 0; i < this->numRows; ++i)
        delete[] pixel[i];
    delete[] pixel;
}

void printHist(ofstream & output)
{
    output <<" " << this->numRows <<" " << this->numCols <<" " <<
this->minVal <<" " << this->maxVal << endl;

    for (int i = 0; i < this->maxVal + 1; ++i)
    {
        output << i <<" ";
        output << hisArray[i] << endl;
    }
}

void displayHist (ofstream & output2){

    int count = 0;
    int temp = 0;

    output2 <<" " << this->numRows <<" " << this->numCols <<" " <<
this->minVal <<" " << this->maxVal << endl;

    for (int i = 0; i < this->maxVal + 1; ++i){
        temp = hisArray[i];
        if (count < 10)
            output2 << count <<" " << "("<< temp << ")" << ": ";
        else
            output2 << count <<" " << "("<< temp << ")" << ": ";
        if ( temp > 0 )
        {

```

```

        if (temp > 70)
            temp = 70;
        for (int j = 0; j < temp; j++)
            output2 << "+" ;
    }
    output2 << endl;
    count++;
}
}

```

```

void free_Heap (){
    delete[] this->hisArray;
}

```

```

void threshold (ifstream & input, ofstream & output3, ofstream &
output4, int thrVal)
{
    thresholdValue = thrVal;
    int **pixel;
    int pixelValue = 0;
    minVal = 0;
    maxVal = 1;

    output3 << " " << this->numRows << " " << this->numCols << " " <<
this->minVal << " " << this->maxVal << endl;

    output4 << " " << this->numRows << " " << this->numCols << " " <<
this->minVal << " " << this->maxVal << endl;

    pixel = new int*[this -> numRows];
    for (int i = 0; i < this -> numRows; ++i){
        pixel[i] = new int[ this -> numCols ];
        for (int j = 0; j < this -> numCols; ++j)
            pixel[i][j] = 0;
    }

    for (int i = 0; i < this -> numRows; ++i){
        for (int j = 0; j < this -> numCols; ++j)
        {
            input >> pixel[i][j];
            pixelValue = pixel[i][j];
            if (thresholdValue <= pixelValue){
                output3 << "1" << " ";
                output4 << "1" << " ";
            }
        }
    }
}

```

```

        else{
            output3 << "0" << " ";
            output4 << "." << " ";
        }
    }
    output3 << endl;
    output4 << endl;
}

for (int i = 0; i < this->numRows; ++i)
    delete[] pixel[i];
delete[] pixel;
}

};

```

```

int main(int argc, const char * argv[]) {

    // input 1
    string inputName = argv[1];
    ifstream input;
    input.open(inputName);

    // output 1
    string outputName1 = argv[3];
    ofstream outputFile1;
    outputFile1.open(outputName1);

    //output 2
    string outputName2 = argv[4];
    ofstream outputFile2;
    outputFile2.open(outputName2);

    // ouput 3
    string outputName3 = argv[5];
    ofstream outputFile3;
    outputFile3.open(outputName3);

    //output 4
    string outputName4 = argv[6];
    ofstream outputFile4;
    outputFile4.open(outputName4);

    if (input.is_open()){
        if (outputFile1.is_open()){
            if (outputFile2.is_open()){

                Image* read_img = new Image(input);
                read_img -> computeHist(input);
                read_img -> printHist(outputFile1);
            }
        }
    }
}

```

```

        read_img -> displayHist(outputFile2);
        input.close();
        outputFile1.close();
        outputFile2.close();
        read_img -> free_Heap();
        delete read_img;
    } else cout << "Error Output 2"<< endl;
} else cout << "Error Output 1"<< endl;
}
else cout << "Error Input"<< endl;

input.open(inputName);
int thrVal = atoi(argv[2]);

if (input.is_open()){
    if (outputFile3.is_open()){
        if (outputFile4.is_open()){
            Image* read_img = new Image(input);
            outputFile3 << "The threshold value is " << thrVal <<
endl;
            outputFile4 << "The threshold value is " << thrVal <<
endl;
            read_img -> threshold(input, outputFile3, outputFile4,
thrVal);
            input.close();
            outputFile3.close();
            outputFile4.close();

        } else cout << "Error Output 4"<< endl;
    } else cout << "Error Output 3"<< endl;
}
else cout << "Error Input"<< endl;
return 0;
}

```

DATA 1:

Output File:

a) Compute the Histogram OutputFile 1 (argv[3]):

31 40 0 9

0 309

1 288

2 194

3 64

4 0

5 2

6 12

7 106

8 124

9 141

b) Display the Histogram OutputFile 2 (argv[4]) :

31 40 0 9

0 (309): ++++++

1 (288): ++++++

2 (194): ++++++

3 (64): ++++++

4 (0):

5 (2): ++

6 (12): ++++++

7 (106): ++++++

8 (124): ++++++

9 (141): ++++++

c) Display the Histogram for visual OutputFile 3 (argv[5]) :
The threshold value is 5

31 40 0 1

[illegible]

d) Display the Histogram OutputFile 4 (argv[6]) :

The threshold value is 5

31 40 0 1

```

.....
...1...1.....1.....11..1..
.....111.....
....1.....11111.....
.....1111111.....
.....111111..11.....
.....111111..1111.....
.....11111111111111.....
.....11111111111111.....
.....11111111111111.....
.....11111111111111.....
.....111111111111..111111.....
.....11111111111111111111.....
.....11111111111111..11111111111111111111.....
.....111111111111111111111111.....
...1...11111..1111111111..11111...1...
..1.....1111..1111111111..1111....1...
.....1111111111..11111111..11.....
..1.....11111111111111111111..11.....
...1.....1111111111111111.....
.....11111111111111.....
.....1111111111.....11.....
.....11111111.....11.....
..11.....11111.....
...1.....111.....1.....
.....1.....1.....1....
.....1.....
.....

```

Better image of the argv[6]

The threshold value is 5

31 40 0 1



DATA 2:

Output:

a) Display the Histogram OutputFile 1 (argv[3]) :

46 46 1 63

0 0

1 277

2 278

3 270

4 319

5 278

6 7

7 6

8 35

9 4

10 5

11 7

12 8

13 6

14 9

15 3

16 3

17 0

18 12

19 1

20 3

21 4

22 7

23 3

24 7

25 3

26 0

27 3

28 15

29 3

30 7

31 7

32 7

33 2

34 10

35 10

36 0
37 0
38 25
39 1
40 7
41 19
42 18
43 18
44 13
45 8
46 2
47 2
48 313
49 0
50 0
51 8
52 2
53 1
54 2
55 11
56 0
57 0
58 25
59 0
60 9
61 1
62 2
63 10

b)Display the Histogram for vis OutputFile 2(argv[4]) :

46 46 1 63

0 (0):

1 (277): ++++++

2 (278): ++++++

3 (270): ++++++

4 (319): ++++++

5 (278): ++++++

6 (7): ++++++

7 (6): ++++++

8 (35): ++++++

9 (4): +++++

10 (5): +++++

11 (7): ++++++

12 (8): ++++++

13 (6): ++++++

14 (9): ++++++

15 (3): +++

16 (3): +++

17 (0):

18 (12): ++++++

19 (1): +

20 (3): +++

21 (4): +++++

22 (7): ++++++

23 (3): +++
24 (7): +++++++
25 (3): +++
26 (0):
27 (3): +++
28 (15): ++++++++++++
29 (3): +++
30 (7): +++++++
31 (7): +++++++
32 (7): +++++++
33 (2): ++
34 (10): ++++++++
35 (10): ++++++++
36 (0):
37 (0):
38 (25): ++++++++++++
39 (1): +
40 (7): +++++++
41 (19): ++++++++++++
42 (18): ++++++++++++
43 (18): ++++++++++++
44 (13): ++++++++++++
45 (8): +++++++
46 (2): ++
47 (2): ++
48 (313): ++++++++++++

49 (0):

50 (0):

51 (8): ++++++++

52 (2): ++

53 (1): +

54 (2): ++

55 (11): ++++++++

56 (0):

57 (0):

58 (25): ++++++

59 (0):

60 (9): ++++++++

61 (1): +

62 (2): ++

63 (10): ++++++++

c) Display the Histogram for visual OutputFile 3(argv[5]) :

The threshold value is 38

46 46 0 1

[illegible]

d)Display the Histogram OutputFile 4 (argv[6]) :

The threshold value is 38

46 46 0 1

```
.....
...1.....1.....
.....11.....1...1.....
.....11.....1...1.....
.....1.....1...1.....
.....1.....1...1.....
.....1.....1...1.....
.....1.....1...1.....
.....1.....1...1.....
.....111...1.....
...1.....1.111.1.....
.....1...111.1.....
....11.....111.1111.....
....11.....1.....
.....11111111.....
.....1111.111.1111.....
.....111111111111.11.....1.....
.....11111111.11111111...1.....
.....1111111111111111...1.....
.....1111111111111111...1.....
.....1111111111111111...111.....
.....111.111.11111.1111.1111.....
..1.....111111111.11111111.11.1111.....
.....111111111111111111...11111111.....
.....11111111.....
.....111111.11111111111111...11111111.....
...111111.111111111111111111...1.111.111111.11..
.....111111111111111111.111..11111.111.....
.....11.11111.111111111111..111.111.....
.....11111111...11111111...11111.....
.....11111111..111111.11...111.....
.....1111.111111.11111111..11.....
.....11111111.....
.....11111.1111111111.11..1.....
.....1111111111111111.....
.....1111111111111111.11.....1.....
....11.1.....
.....111...1...1.....
.....11.1...11.1.....
....1.....1111.111.....
.....1111111.....
.....11.11.....
...1.....111.....1...1.....
...1....11.....1.....
....11.....1.....1.....
.....1.....1.....11...1.....
.....1.....1.....
```

Better image of the argv[6]

[illegible]