Assignment 6

Part 1 source code

Get more details about codes and comments at HistogramEqualization.cpp.

Part 2 explaination

(1)

```
void drawHistogram(IplImage* img, string histName)
```

This function aims to dram a histogram and save it into a file named histName.

for example:

```
//draw the histogram of origin image
drawHistogram(img, "OriginHistogram.png");

//draw the histogram of result image
drawHistogram(img, "ResultHistogram.png");
```

(2)

```
//init the histogram
double hist[MAX_GRAY_VALUE];
for (int i = 0; i < MAX_GRAY_VALUE; i++)
{
    hist[i] = 0;
}

//get the data of histogram
for (int row = 0; row < img->height; row++)
{
    uchar* ptr = (uchar*)img->imageData + row*img->widthStep;
    for (int col = 0; col < img->width; col++)
    {
        int temp_gray_value = ptr[col];
        hist[temp_gray_value]++;
    }
}
```

This part initializes the hist array with 0 which stores the histogram data.

And it looks up the point of image one by one to count the number of each gray value from 0 to 255.

(3)

```
//normalization
int totalpoints = img->height * img->width;
for (int i = 0; i < MAX_GRAY_VALUE; i++)
{
    hist[i] = hist[i] / totalpoints;
}

//sum the normalized histogram
double sum_hist[MAX_GRAY_VALUE];
for (int i = 0; i < MAX_GRAY_VALUE; i++)
{
    if (i == 0)
    {
        sum_hist[i] = hist[0];
    }
    else
    {
        sum_hist[i] = sum_hist[i - 1] + hist[i];
    }
}
```

It calculates the totalpoint to invert the hist array from occurences of gray value to probability of gray value. This operation is also called normalization.

Then it sum the normalization hist array for the next operation.

(4)

```
//find the max gray and the min gray
int max_gray = 0;
int min_gray = 255;
for (int i = 0; i < MAX_GRAY_VALUE; i++)
{
    if (hist[i] > 0 && i > max_gray)max_gray = i;//find the max index that hist[index] > 0
    if (hist[i] > 0 && i < min_gray)min_gray = i;//find the min index that hist[index] < 0
}

//equalize the origin img
for (int row = 0; row < img->height; row++)
{
    uchar* ptr = (uchar*)img->imageData + row*img->widthStep;
    for (int col = 0; col < img->width; col++)
    {
        int result_gray_value = sum_hist[ptr[col]] * (max_gray - min_gray) + min_gray;
        ptr[col] = result_gray_value;
    }
}
```

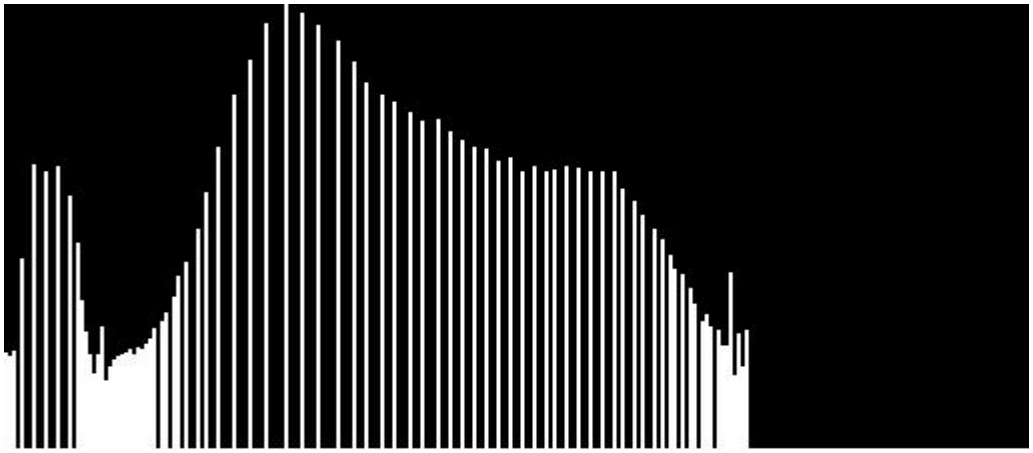
First of all, it find the max&min gray value of the origin image. Then it equalizes the origin image with several variables.

I think equalization just like to reset the gray value with the weight of the point multiplies the gap between max and min gray value.

Part 3 output

(1)origin histogram

(2)result histogram



(3)result image