Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Дисциплина: Методы и средства цифровой обработки информации

Выполнили студенты гр. 43501/1 А.А. Дроздовский
Руководитель Н.А. Абрамов

«___» _____ 2017г.

Санкт -Петербург
2017

СОДЕРЖАНИЕ

1. Цель работы

Изучить методы цифровой обработки изображений:

- Линейное растяжение гистограммы;
- Эквализация гистограммы.

2. Ход работы

На рисунке 2.1. и 2.2. представлены исходные изображения:

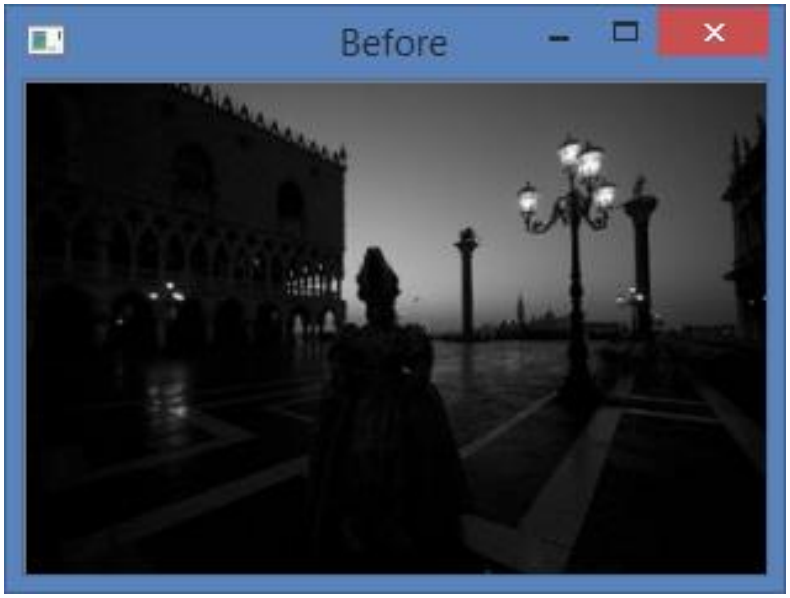

Рис. 2.1. Затемненное изображение (Изображение 1).

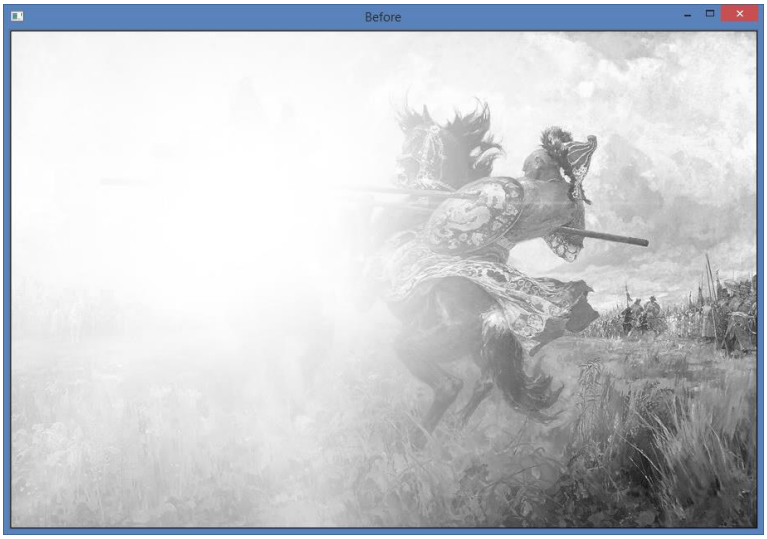

Рис. 2.2. Засвеченное изображение (Изображение 2).

На рисунках 2.3. и 2.4. представлены исходные гистограммы изображений на рисунках 2.1. и 2.2.:

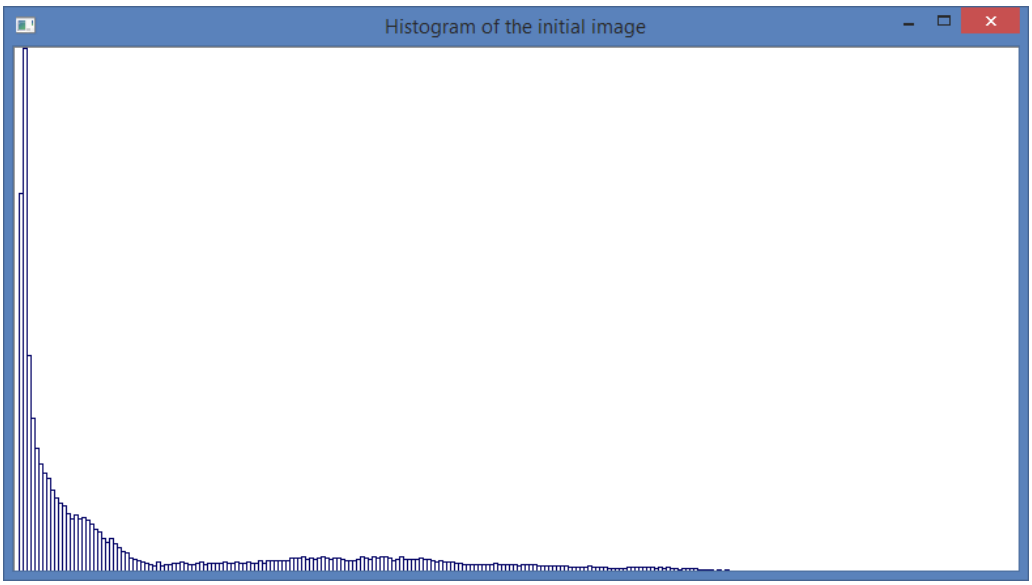

Рис. 2.3. Гистограмма затемненного изображения.

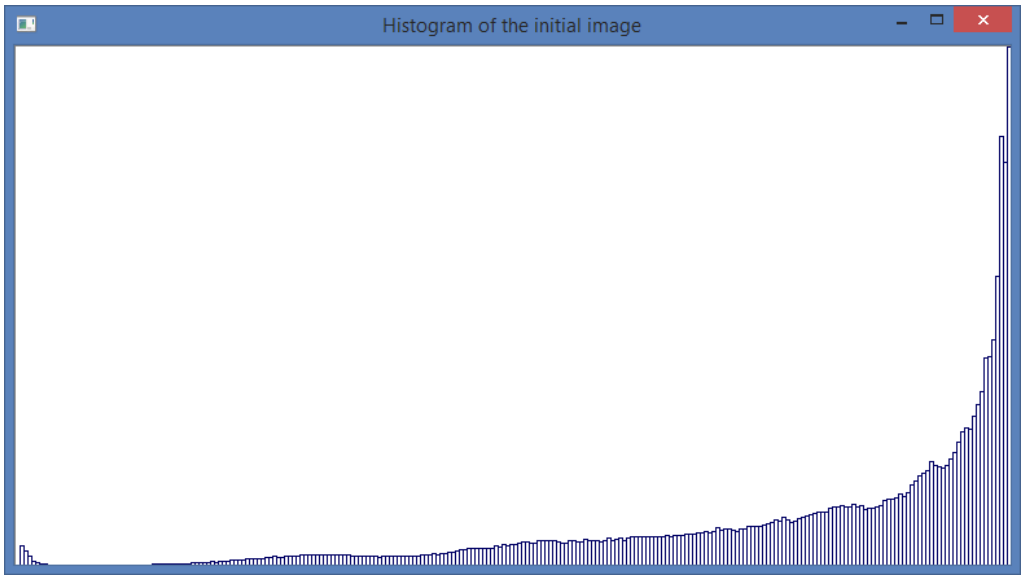

Рис. 2.4. Гистограмма засвеченного изображения.

2.1. Линейное растяжение гистограммы

Отбросим 5% от общего числа пикселей из исходных гистограммы, на рисунках 2.5. и 2.8. представлены результаты. Затем выполним линейное растяжение гистограммы, заменив исходные цвета в исходном изображение по формуле:

$S_{out} = (S_{in} - a) \cdot \left(\frac{d-c}{b-a}\right) + c$, где

S_{out} – результирующий цвет;

S_{in} – исходный цвет;

a,b – нижняя и верхняя граница исходного диапазона;

c,d – нижняя и верхняя граница результирующего диапазона.

Гистограммы после линейного растяжение представлены на рисунках 2.6. и 2.9., на рисунках 2.7. и 2.10. представлены результирующие изображения:

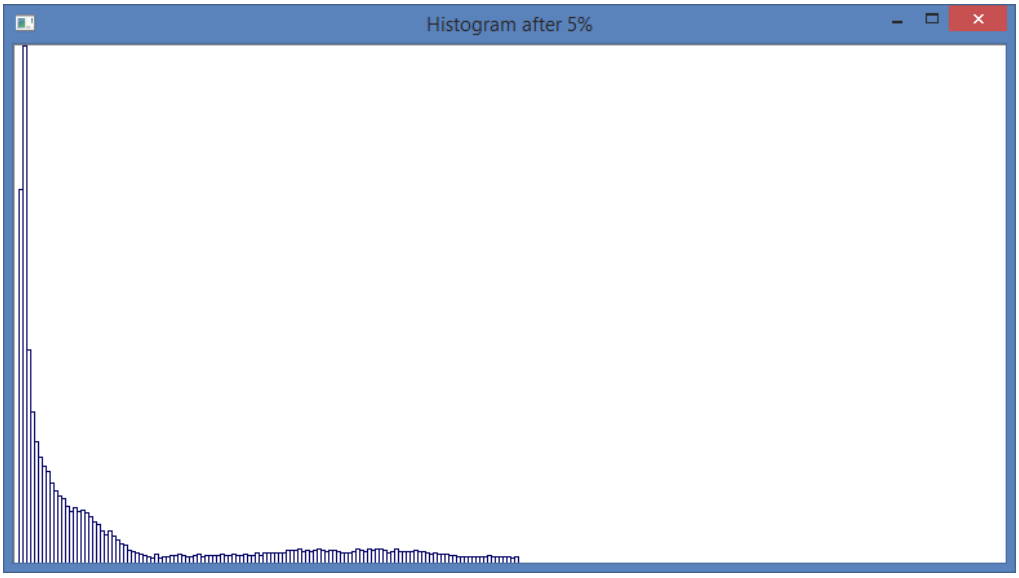

Рис. 2.5. Исходная гистограмма изображения 1 (После удаления 5% пикселей).

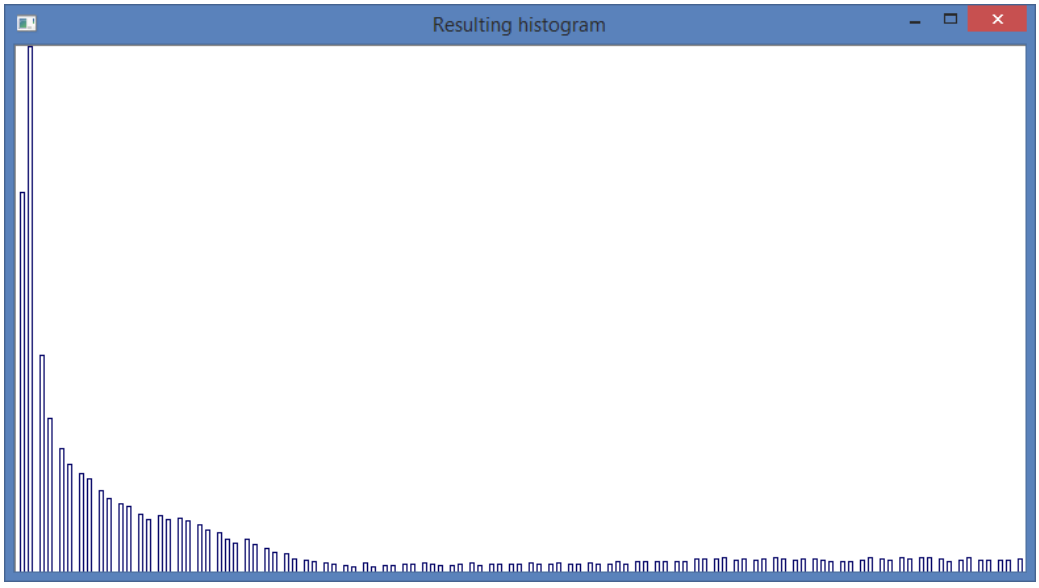

Рис. 2.6. Результирующая гистограмма изображения 1 после линейного растяжения.

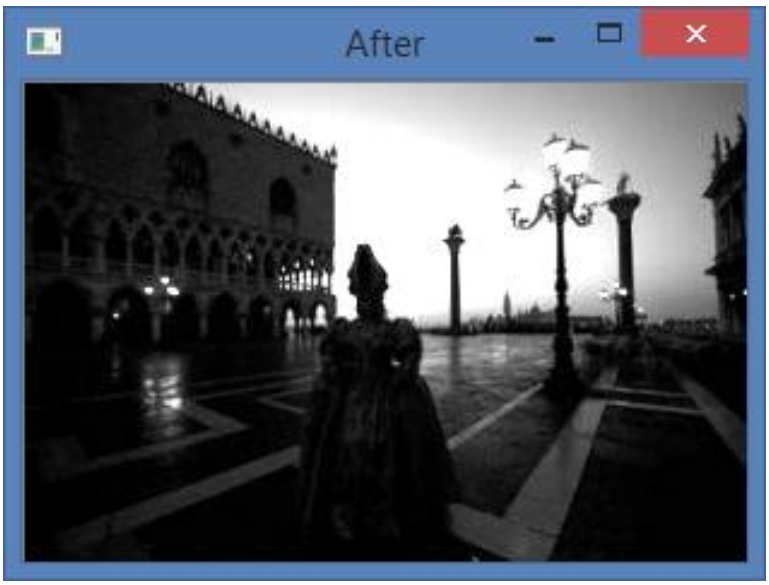

Рис. 2.7. Изображение 1 после линейного растяжения.

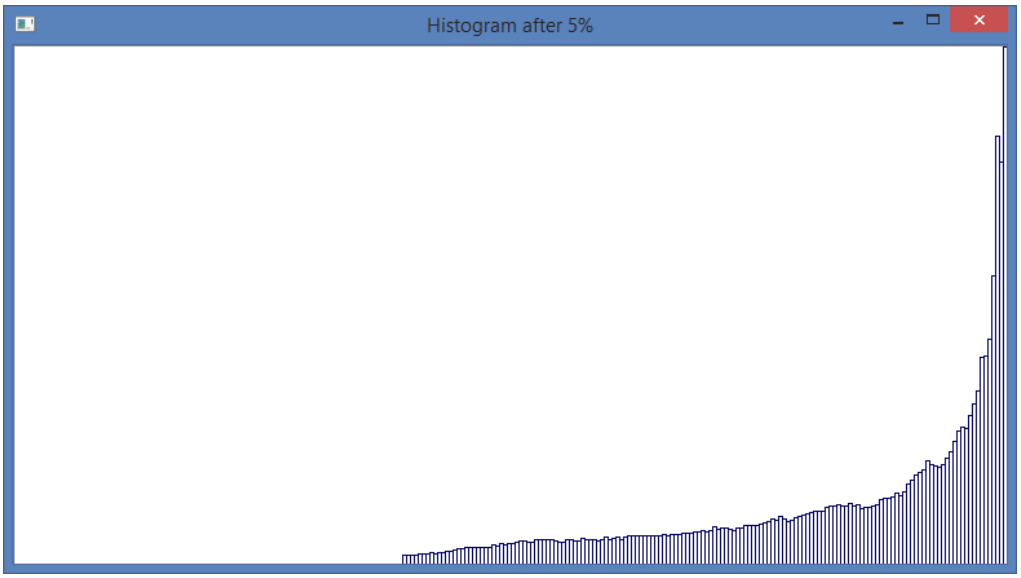

Рис. 2.8. Исходная гистограмма изображения 2 (После удаления 5% пикселей).

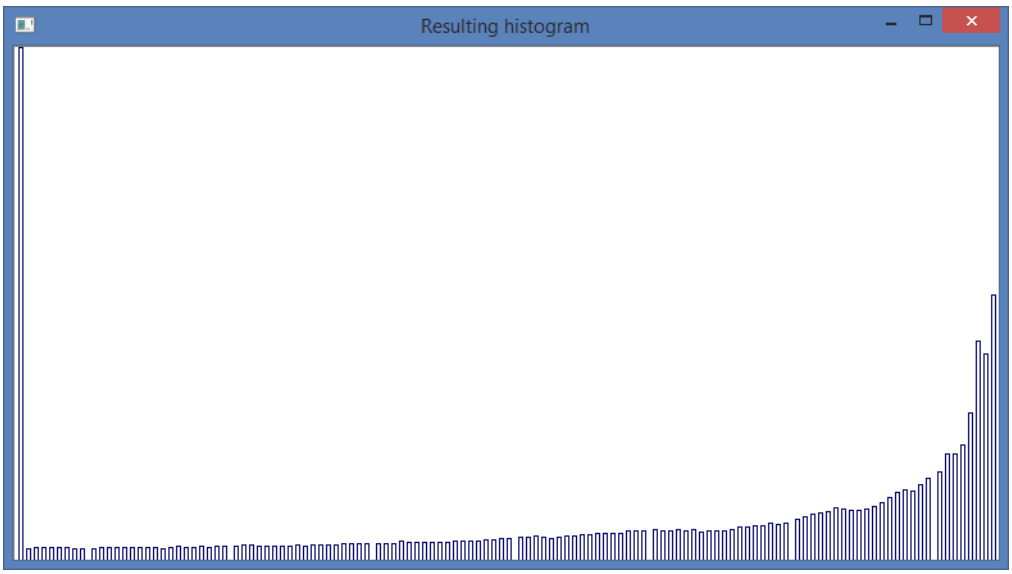

Рис. 2.9. Результирующая гистограмма изображения 2 после линейного растяжения.

Рис. 2.10. Изображение 2 после линейного растяжения.

2.2. Эквализация гистограммы

Заменим исходные цвета в исходном изображение по формуле:

$S_{out} = round\left(\frac{CDF(S_{in})-CDF(min)}{M \cdot N-1} \cdot (L-1)\right)$, где

S_{out} – результирующий цвет;

S_{in} – исходный цвет;

$CDF(i) = \sum_{j=0}^{i} n_j$;

n_j – количество пикселей оттенка j;

L – количество оттенков;

$M \cdot N$ – количество пикселей.[1]

Гистограммы после эквализации представлены на рисунках 2.11. и 2.13., на рисунках 2.12. и 2.14. представлены результирующие изображения:

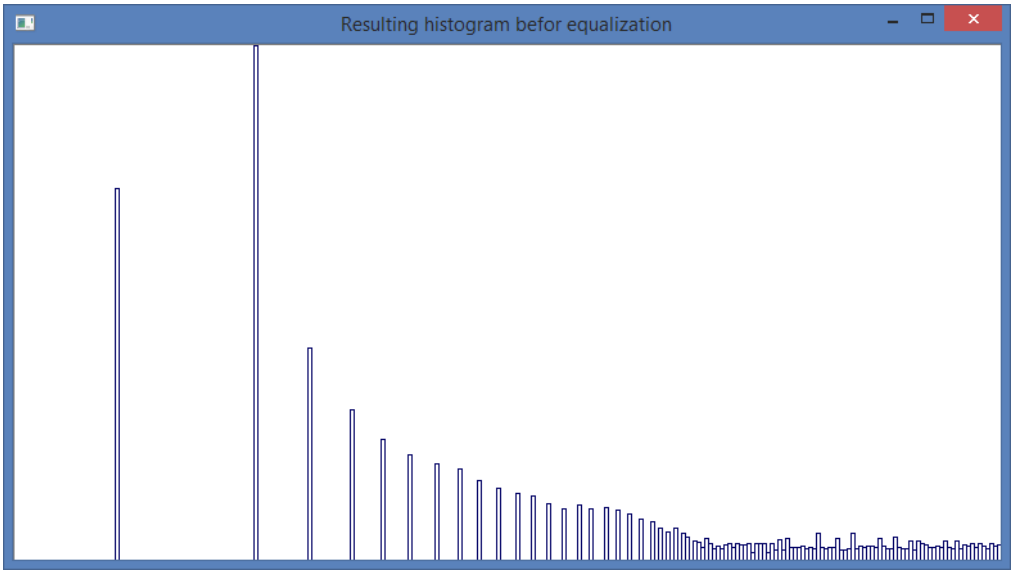

Рис. 2.11. Результирующая гистограмма изображения 1 после эквализации.

Рис. 2.12. Изображение 2 после эквализации.

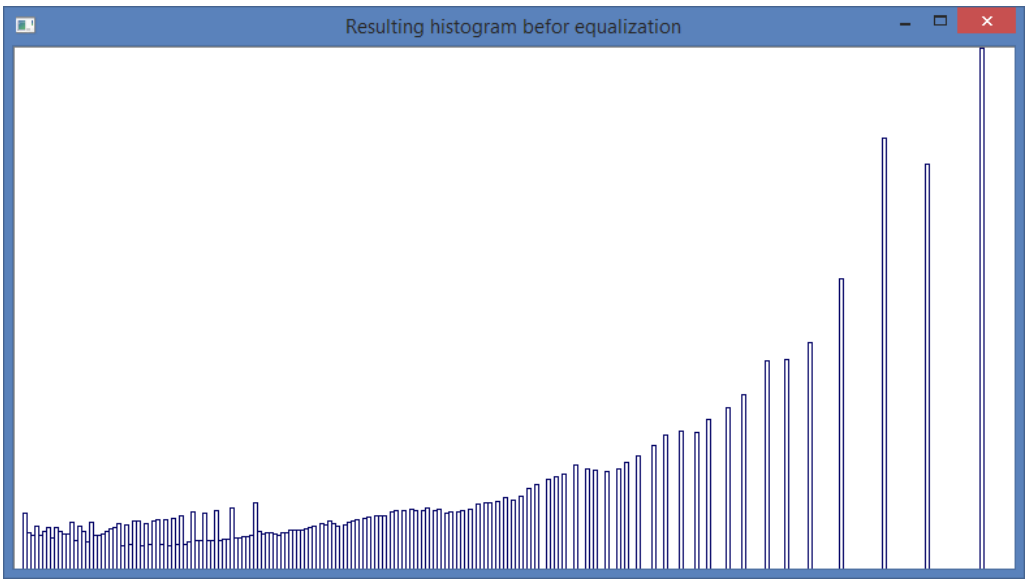

Рис. 2.13. Результирующая гистограмма изображения 2 после эквализации.

Рис. 2.14. Изображение 2 после эквализации.

3. Выводы

В ходе проделанной работы было протестировано два метода цифровой обработки изображений. Оба метода работают и применимы для работы с изображениями. Наглядно видно, что применение эквализации гистограммы при работе с засвеченным изображением наиболее эффективно. При работе с затемненным изображение лучше показывает себя линейное растяжение гистограммы, при применении эквализации видны значительные искажения в сравнение с исходным файлом.

Список используемой литературы

1. Histogram equalization [Электронный ресурс] // Википедия : свободная энцикл., 2017. – URL: https://en.wikipedia.org/wiki/Histogram_equalization (Дата обращения: 19.09.2017).

Текст программы

```
#include "stdafx.h"
#include <opencv2\opencv.hpp>
#include <iostream>
#include <string>
#include <vector>
using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
	/*------------------== Часть 1 ==------------------*/
	namedWindow("LR1");
	String imageName("LR1_3.jpg");
	if (argc > 1)
	{
		imageName = argv[1];
	}
	/// Считывание изображения в оттенках серого
	Mat image = imread(imageName.c_str(), IMREAD_GRAYSCALE);
	/// Проверка входных значений
	if (image.empty())
	{
		cout << "Could not open or find the image" << endl;
		return -1;
	}
	/// Вывод на экран первоначального изображения
	imshow("Before", image);

	int hist_size = 256;
			// Число элементов в гистограмме
	int hist_width = 3;
				// Для автоподбора ширины изображений
	float range_0[] = { 0,256 };
	const float* ranges = { range_0 };
	int hist_w = hist_size*hist_width; int hist_h = 400;                    //
Ширина и высота полотна для нанесения гистограммы
	int bin_w = round((double)hist_w / hist_size);
	// Ширина элементов гистограммы

	/// Построение гистограммы
	Mat hist;
	bool uniform = true, accumulate = false;
	calcHist(&image, 1, 0, Mat(), hist, 1, &hist_size, &ranges, uniform,
accumulate);
	Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(255, 255, 255));

	Mat copy_hist,hist_without_Norm;
		// Копии гистограммы
	hist.copyTo(copy_hist);
				// Для брасывания 5%
	hist.copyTo(hist_without_Norm);
			// Для второй части работы
	normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
	for (int i = 1; i < hist_size; i++)
		rectangle(histImage, Point(i*bin_w, hist_h),
			Point((i + 1)*bin_w, hist_h - round(hist.at<float>(i - 1))),
			Scalar(100, 0, 0), 1, 8, 0.5);
	/// Вывод гистограммы исходного изображения
	imshow("Histogram of the initial image", histImage);
```

```
	float b = 255, a = 0;
				// b - верхняя граница, a - нижняя граница исходной
гистограммы
	float d = 255, c = 0;
				// d - верхняя граница, c - нижняя граница результирующей
гистограммы
	double FivePercent = round((double)(image.cols*image.rows)*0.05);   // 5% от
общего числа пикселей исходного изображения
	double sum_for_del = 0;
				// количество удаленных пикселей

	for (; sum_for_del < FivePercent;)
		if (copy_hist.at<float>(a) > copy_hist.at<float>(b))                //
Сравнивнение последнего и первого элемент
		{
			sum_for_del += copy_hist.at<float>(b);
			for (int i = 0; i < image.cols*image.rows; i++)
				if (abs(double(image.at<unsigned char>(i)) - b)<1 )
					image.at<unsigned char>(i) = b-1;
			copy_hist.at<float>(b) = 0;
			b--;
		}
		else
		{
			for (int i = 0; i < image.cols*image.rows; i++)
				if (abs(double(image.at<unsigned char>(i)) - a)<1)
					image.at<unsigned char>(i) = a+1;
			sum_for_del += copy_hist.at<float>(a);
			copy_hist.at<float>(a) = 0;
			a++;
		}	Mat histImage2(hist_h, hist_w, CV_8UC3, Scalar(255, 255, 255));
	normalize(copy_hist, copy_hist, 0, histImage2.rows, NORM_MINMAX, -1, Mat());
	for (int i = 1; i < hist_size; i++)
		rectangle(histImage2, Point(i*bin_w, hist_h),
			Point((i + 1)*bin_w, hist_h - round(copy_hist.at<float>(i - 1))),
			Scalar(100, 0, 0), 1, 8, 0.5);
	/// Вывод гистограммы после отбрасывания 5%
	imshow("Histogram after 5%", histImage2);
	/// Замена пикселей в исходном изображение
	for (int i = 0; i < image.rows*image.cols; i++)
	{
		image.at<unsigned char>(i) = (double(image.at<unsigned char>(i)) - a)*((d
- c) / (b - a)) + c;
	}
	/// Вывод полученного изображения
	imshow("After", image);
	/// Построение результирующей гистограммы
	Mat hist_rez;
				// Результирующая гистограмма
	calcHist(&image, 1, 0, Mat(), hist_rez, 1, &hist_size, &ranges, uniform,
accumulate);
	Mat histImage3(hist_h, hist_w, CV_8UC3, Scalar(255, 255, 255));
	normalize(hist_rez, hist_rez, 0, histImage3.rows, NORM_MINMAX, -1, Mat());
	for (int i = 1; i < hist_size; i++)
		rectangle(histImage3, Point(i*bin_w, hist_h),
			Point((i + 1)*bin_w, hist_h - round(hist_rez.at<float>(i - 1))),
			Scalar(100, 0, 0), 1, 8, 0.5);
	/// Вывод результирующей гистограммы
	imshow("Resulting histogram", histImage3);
	/*------------------== Часть 2 ==------------------*/
	Mat image2 = imread(imageName.c_str(), IMREAD_GRAYSCALE);
	vector<float> CDF;
	CDF.push_back(hist_without_Norm.at<float>(0));
	for (int i = 1; i < hist_without_Norm.rows*hist_without_Norm.cols; i++)
```

```
	{
		CDF.push_back(CDF.at(i - 1) + hist_without_Norm.at<float>(i) );
	}
	/// Замена исходных пикселей
	for (int i = 0; i < image2.rows*image2.cols; i++)
	{
		image2.at<unsigned char>(i) = round(((CDF.at(double(image2.at<unsigned
char>(i)))-CDF.at(0))/ (image2.rows*image2.cols -1))*255);
	}
	/// Вывод полученного изображения
	imshow("After Histogram Equalization", image2);
	/// Построение результирующей гистограммы после эквализации
	Mat hist_equalization;
				// Результирующая гистограмма
	calcHist(&image2, 1, 0, Mat(), hist_equalization, 1, &hist_size, &ranges,
uniform, accumulate);
	Mat histImage4(hist_h, hist_w, CV_8UC3, Scalar(255, 255, 255));
	normalize(hist_equalization, hist_equalization, 0, histImage4.rows, NORM_MINMAX,
-1, Mat());
	for (int i = 1; i < hist_size; i++)
		rectangle(histImage4, Point(i*bin_w, hist_h),
			Point((i + 1)*bin_w, hist_h - round(hist_equalization.at<float>(i
- 1))),
			Scalar(100, 0, 0), 1, 8, 0.5);
	/// Вывод результирующей гистограммы
	imshow("Resulting histogram befor equalization", histImage4);
	waitKey(0);
	return 0;
}
```