**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ВТ**

**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**ПО ДИСЦИПЛИНЕ**

**«Обработка изображений и компьютерное зрение (OpenCV)»**

**ТЕМА: «ЛОГИЧЕСКИЕ ОПЕРАЦИИ И ОПЕРАЦИИ НАД ГИСТОГРАММИ»**

**Бригада №1**

| Студенты гр. 8307: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Лукьянов А.В. |
| --- | --- | --- |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Крылов А. |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Якушев Г.С. |
| Преподаватель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Костичев С.В. |
|  |  |  |

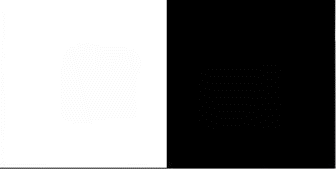
Санкт-Петербург

2023

**Задание на лабораторную работу:**

1. Логические операции

- создать изображения drawing1.jpg и drawing2.jpg вида



- выполнить логические операции: инверсия, конъюнкция, дизъюнкция, исключающее ИЛИ над изображениями drawing1.jpg и drawing2.jpg

- выполнить отображения исходного и результирующего изображений для всех логических операций

1. Операции над гистограммами
   1. Вычисление и нормализация гистограммы

Разработайте программу, обеспечивающую:

- загрузку изображения image2.jpg (функции imread, imshow)

- выделение каналов изображения (split)

- вычисление гистограммы для каждого канала (calcHist). В OpenCV каналы изображения хранятся в порядке BGR, а не в RGB

- построение и нормализация гистограмм (normalize); выполните для всех типов нормализации (NORM\_MINMAX, NORM\_INF, NORM\_L1, NORM\_L2

Выполните отображения исходной и нормализованных гистограмм.

* 1. Выравнивание (эквализация ) гистограмм

Разработайте программу, обеспечивающую:

- загрузку изображения image3.jpg (функции imread, imshow)

- преобразование в оттенки серого (cvtColor)

- вычисление гистограммы серого изображения (calcHist)

- выравнивание гистограммы (equalizeHist)

- вычисление гистограммы результирующего изображения (calcHist)

Выполните

- отображение исходного (серого) и результирующего изображений после применения выравнивания гистограмм

- отображение гистограмм исходного (серого) и результирующего изображений

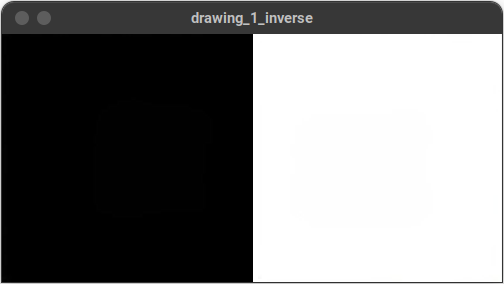
**Программное и аппаратное окружение при выполнении работы**

Поскольку члены нашей группы пользуются различными ОС, мы не смогли сделать выбор в пользу какой-то конкретной ОС, потому мы использовали машины под управлением ОС Ubuntu и ОС Windows 11, а также среду разработки CLion для языков C/C++ с поддержкой кроссплатформенной системой сборки CMake.

Аппаратное обеспечение варьируется довольно сильно - в нашем распоряжении оказались процессоры Intel Core i5-6600(4C|4T, 32 gb ram), i5-12600k(10c|16T, 32gb ram + 40 gb swap), а также AMD A10-9620P(4C|4T, 6gb ram + 20gb swap).

При работе мы использовали CLion версии 2022.3.3., CMake 3.24, libopencv 4.7.0-dev.

**Пример запуска программы**

  
Рисунок 1 - инверсия рисунка 1

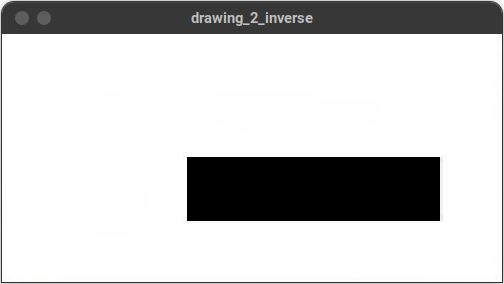
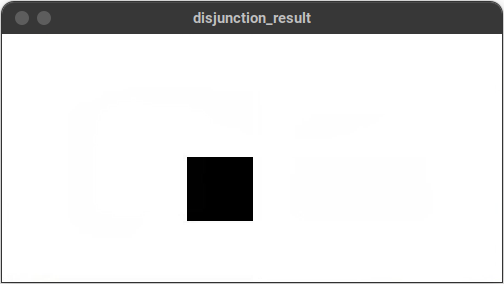


Рисунок 2 - инверсия рисунка 2

  
Рисунок 3 - операция дизъюнкции

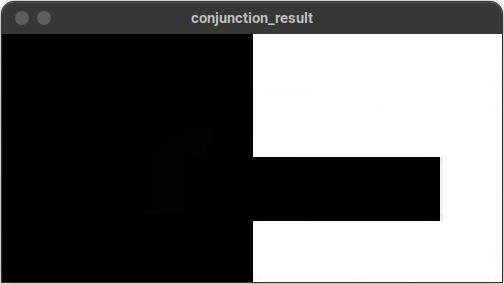
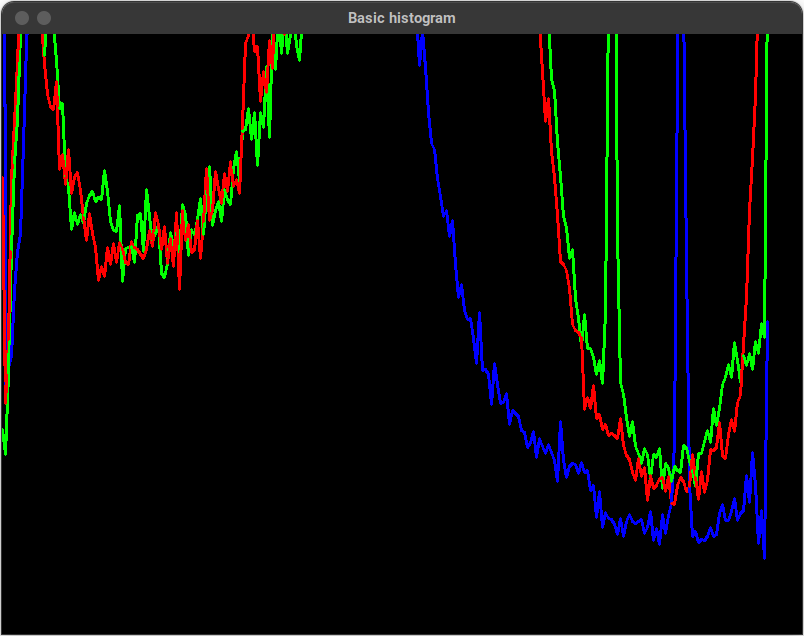
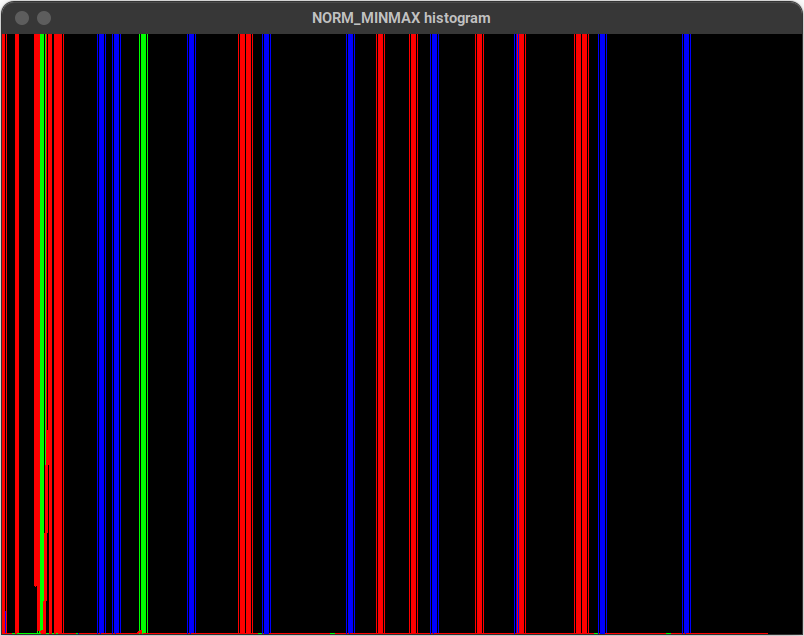
  
Рисунок 3 - операция конъюнкции

  
Рисунок 5 - оригинальное изображение “image2”

  
Рисунок 6: базовая гистограмма image2

  
Рисунок 6: NORM\_minmax гистограмма image2

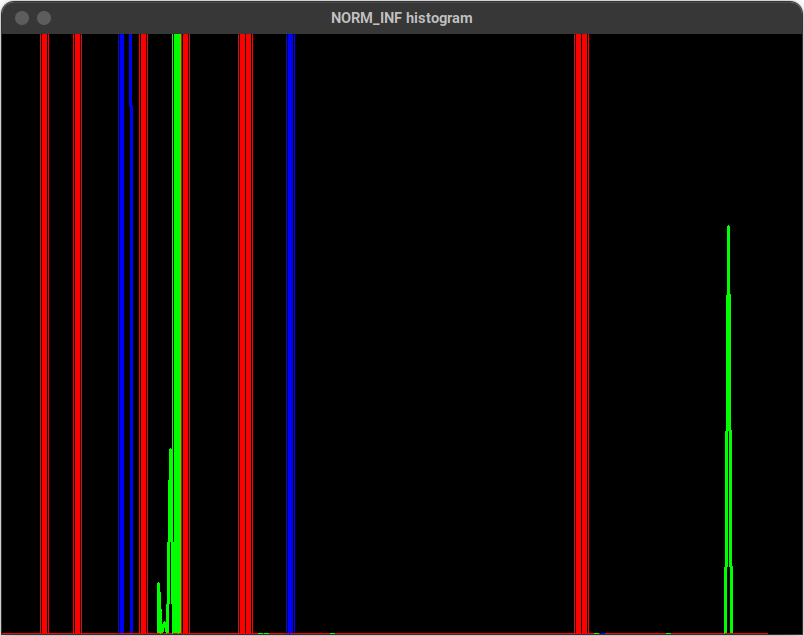
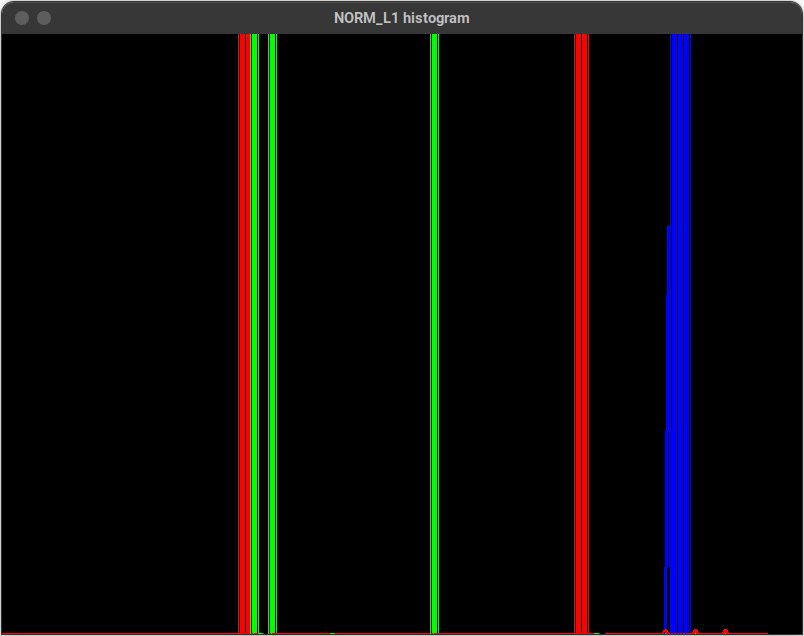


Рисунок 7: NORM\_INF гистограмма image2

  
Рисунок 8: NORM\_L1 гистограмма image2

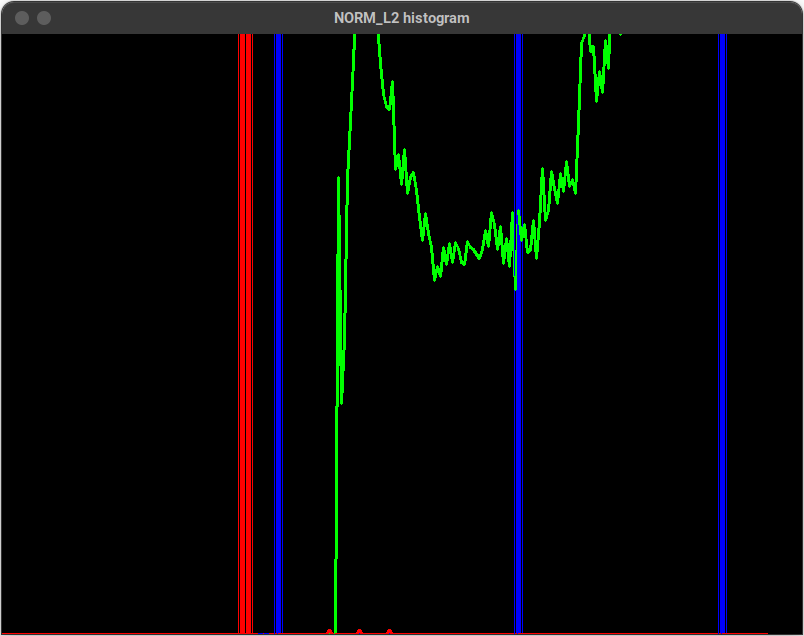
  
Рисунок 9: NORM\_L2 гистограмма image2

  
Рисунок 10 - оригинальное изображение “image3”



Рисунок 11 - изображение “image3” в градациях серого



Рисунок 12 - изображение “image3” после выравнивания гистограммы

К сожалению, гистограммы для примеров выше строиться отказались.

**Заключение**

В ходе лабораторной работы было изучено применение логических

операций над изображениями и базовых функций OpenCV для работы с

гистограммами.

**Текст программы**

**#include <iostream>**

**#include <opencv2/core.hpp>**

**#include <opencv2/imgcodecs.hpp>**

**#include <opencv2/highgui.hpp>**

**#include <opencv2/imgproc.hpp>**

**#define hist\_w 800**

**#define hist\_h 600**

**void inverse(cv::Mat image, const std::string& winName)**

**{**

**cv::bitwise\_not(image, image);**

**cv::imshow(winName, image);**

**}**

**void conjunction(const cv::Mat& first\_image, const cv::Mat& second\_image)**

**{**

**cv::Mat result;**

**cv::bitwise\_and(first\_image, second\_image, result);**

**cv::imshow("conjunction\_result", result);**

**}**

**void disjunction(const cv::Mat& first\_image, const cv::Mat& second\_image)**

**{**

**cv::Mat result;**

**cv::bitwise\_or(first\_image, second\_image, result);**

**cv::imshow("disjunction\_result", result);**

**}**

**void exclusive\_or(const cv::Mat& first\_image, const cv::Mat& second\_image)**

**{**

**cv::Mat result;**

**cv::bitwise\_xor(first\_image, second\_image, result);**

**cv::imshow("exclusive\_or\_result", result);**

**}**

**cv::Mat generateHist(cv::Mat inputOutput, std::vector<cv::Mat> normalizedVector, double histSize)**

**{**

**int bin\_w;**

**bin\_w = cvRound((double) hist\_w / histSize);**

**const std::vector<cv::Scalar> colors = {{255, 0, 0}, {0, 255, 0}, {0, 0, 255}};**

**for( int i = 1; i < histSize; i++ )**

**for(int j = 0; j < 3; j++)**

**line(**

**inputOutput,**

**cv::Point(bin\_w\*(i-1),hist\_h - cvRound(normalizedVector[j].at<float>(i-1))),**

**cv::Point( bin\_w\*(i), hist\_h - cvRound(normalizedVector[j].at<float>(i)) ),**

**colors[j], 2, 8, 0**

**);**

**return inputOutput;**

**}**

**cv::Mat generatePlainHist(cv::Mat inputOutput, cv::Mat normalizedVector, double histSize)**

**{**

**int bin\_w;**

**bin\_w = cvRound((double) hist\_w / histSize);**

**for( int i = 1; i < histSize; i++ )**

**line(**

**inputOutput,**

**cv::Point(bin\_w\*(i-1),hist\_h - cvRound(normalizedVector.at<float>(i-1))),**

**cv::Point( bin\_w\*(i), hist\_h - cvRound(normalizedVector.at<float>(i)) ),**

**cv::Scalar{0, 255, 0}, 2, 8, 0**

**);**

**return inputOutput;**

**}**

**int main() {**

***/\****

***std::string drawing\_1\_path = "/media/artyom/Новая папка/ETU/10\_sem/OpenCV/Lab\_2/drawing1.png";***

***std::string drawing\_2\_path = "/media/artyom/Новая папка/ETU/10\_sem/OpenCV/Lab\_2/drawing2.png";***

***auto drawing\_1 = cv::imread(drawing\_1\_path, cv::IMREAD\_COLOR);***

***if(drawing\_1.empty())***

***{***

***std::cout << "Could not read the image: " << drawing\_1\_path << std::endl;***

***return 1;***

***}***

***else cv::imshow("original img1", drawing\_1);***

***auto drawing\_2 = cv::imread(drawing\_2\_path, cv::IMREAD\_COLOR);***

***if(drawing\_1.empty())***

***{***

***std::cout << "Could not read the image: " << drawing\_2\_path << std::endl;***

***return 1;***

***}***

***else cv::imshow("original img2", drawing\_2);***

***inverse(drawing\_1, "drawing\_1\_inverse");***

***inverse(drawing\_2, "drawing\_2\_inverse");***

***conjunction(drawing\_1, drawing\_2);***

***disjunction(drawing\_1, drawing\_2);***

***exclusive\_or(drawing\_1, drawing\_2);***

***\*/***

**std::string image\_1\_path = "/media/artyom/Новая папка/ETU/10\_sem/OpenCV/Lab\_2/image2.jpg";**

***//std::string image\_2\_path = "/media/artyom/Новая папка/ETU/10\_sem/OpenCV/Lab\_2/drawing2.png";***

**auto image\_1 = cv::imread(image\_1\_path, cv::*IMREAD\_COLOR*);**

**if(image\_1.empty())**

**{**

**std::cout << "Could not read the image: " << image\_1\_path << std::endl;**

**return 1;**

**}**

**else cv::imshow("original img2", image\_1);**

**std::vector<cv::Mat> bgr\_horse;**

**cv::split(image\_1, bgr\_horse);**

**int histSize = 256;**

**float range[] = { 0, 256 }; *//the upper boundary is exclusive***

**const float\* histRange[] = { range };**

**bool uniform = true, accumulate = false;**

**cv::Mat histImageBasic( hist\_h, hist\_w, CV\_8UC3),**

**histImageMinMax( hist\_h, hist\_w, CV\_8UC3, cv::Scalar(0,0,0) ),**

**histImageInf( hist\_h, hist\_w, CV\_8UC3, cv::Scalar(0,0,0) ),**

**histImageL1 ( hist\_h, hist\_w, CV\_8UC3, cv::Scalar(0,0,0) ),**

**histImageL2 ( hist\_h, hist\_w, CV\_8UC3, cv::Scalar(0,0,0) );**

**std::vector<cv::Mat> basicBgrVector(bgr\_horse.size()),**

**bgrVectorNormMinmax(bgr\_horse.size()),**

**bgrVectorNormInf(bgr\_horse.size()),**

**bgrVectorNormL1(bgr\_horse.size()),**

**bgrVectorNormL2(bgr\_horse.size());**

**for(int i = 0; i < 3; i++)**

**{**

**calcHist(&bgr\_horse[i], 1, nullptr, cv::Mat(), basicBgrVector[i], 1, &histSize, histRange, uniform, accumulate);**

**normalize(basicBgrVector[i], bgrVectorNormMinmax[i], 0, histImageMinMax.rows, cv::*NORM\_MINMAX*, CV\_8UC3);**

**normalize(basicBgrVector[i], bgrVectorNormInf[i], histImageInf.rows, 0, cv::*NORM\_INF*, CV\_8UC3);**

**normalize(basicBgrVector[i], bgrVectorNormL1[i], 0, histImageMinMax.rows, cv::*NORM\_L1*, CV\_8UC3);**

**normalize(basicBgrVector[i], bgrVectorNormL2[i], 0, histImageMinMax.rows, cv::*NORM\_L2*, CV\_8UC3);**

**}**

**histImageBasic = generateHist(histImageBasic, basicBgrVector, histSize);**

**histImageMinMax = generateHist(histImageMinMax, bgrVectorNormMinmax, histSize);**

**histImageInf = generateHist(histImageInf, bgrVectorNormInf, histSize);**

**histImageL1 = generateHist(histImageL1, bgrVectorNormL1, histSize);**

**histImageL2 = generateHist(histImageL2, bgrVectorNormL2, histSize);**

**cv::imshow("Basic histogram", histImageBasic );**

**cv::imshow("NORM\_MINMAX histogram", histImageMinMax );**

**cv::imshow("NORM\_INF histogram", histImageInf );**

**cv::imshow("NORM\_L1 histogram", histImageL1 );**

**cv::imshow("NORM\_L2 histogram", histImageL2 );**

**std::string image\_3\_path = "/media/artyom/Новая папка/ETU/10\_sem/OpenCV/Lab\_2/image3.jpg";**

**auto image\_3 = cv::imread(image\_3\_path, cv::*IMREAD\_COLOR*);**

**if(image\_3.empty())**

**{**

**std::cout << "Could not read the image: " << image\_3\_path << std::endl;**

**return 1;**

**}**

**else cv::imshow("original img3", image\_3);**

**cv::Mat grayImg, basicHistImage3( hist\_h, hist\_w, CV\_8UC3), resultHist, finalHist;**

**cv::cvtColor(image\_3, grayImg, cv::*COLOR\_BGR2GRAY*);**

**cv::imshow("grayscale img3", grayImg);**

**calcHist(&grayImg, 1, nullptr, cv::Mat(), resultHist, 1, &histSize, histRange, uniform, accumulate);**

**resultHist = generatePlainHist(resultHist, resultHist, histSize);**

**cv::imshow("Image3 histogram", resultHist);**

**cv::equalizeHist(grayImg, finalHist);**

**cv::imshow("Image3 equalized histogram", finalHist);**

**cv::waitKey(0);**

**}**