**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Телевидения и Видеотехники**

**курсовой проект**

**по дисциплине «Информационные технологии»**

**Тема: Разработка редактора изображений**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2105 |  | Левонюк А.С. |
| Преподаватель |  | Чиркунова А.А. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на курсовой проект**

|  |  |  |
| --- | --- | --- |
| Студент Левонюк А.С. | | |
| Группа 2105 | | |
| Тема проекта: Разработка редактора изображений | | |
| Исходные данные:  Изображение формата .bmp; 0.0(Перемещение прямоугольной области);1.3(вырез участка изображения,); 2.2(Увеличение масштаба по горизонтали); 3.5(гамма-коррекция); 4.2(нормализация гистограммы.); 5.9(пороговая сегментация желтый RGB). | | |
| Содержание пояснительной записки:  Содержание, Введение, 1. Теоретическая часть, 2. Программные документы, Заключение, Список использованных источников | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 24.03.2023 | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Левонюк А.С. |
| Преподаватель |  | Чиркунова А.А. |

**Аннотация**

Курсовая работа основывается на обработке изображений. Решение заданий заключается в загрузке изображений, использовании фильтров, применении различных методов работы с ними и выводе измененного результата изображения. Курсовое задание включает в себя создание/загрузку изображения, изменение его визуальных свойств. В курсовой работе используются следующие фильтры:

1) масштабирование изображения

2) изменение средней яркости изображения

3) линейное контрастирование изображения

4) растяжка гистограммы

5) пороговая сегментация цветных изображений

**содержание**

|  |  |  |
| --- | --- | --- |
| 1. | Введение | 6 |
| 1.0 | Фильтр «Перемещение прямоугольной области» | 7 |
| 1.1 | Фильтр «Вырез участка изображения» | 7 |
| 1.2 | Фильтр «Увеличение масштаба по горизонтали» | 7 |
| 1.3 | Фильтр «Гамма-коррекция» | 7 |
| 1.4 | Фильтр «Нормализация гистограммы» | 7 |
| 1.5 | Фильтр «Пороговая сегментация» | 7 |
| 2. | Программные документы | 8 |
| 2.1 | Спецификация | 8 |
| 2.2 | Описание программы | 8 |
| 2.2.1 | Общие сведения | 8 |
| 2.2.2 | Функциональное назначение | 8 |
| 2.2.3 | Описание логической структуры | 8 |
| 2.2.4 | Используемые технические средства | 8 |
| 2.2.5 | Вызов и загрузка | 8 |
| 2.2.6 | Выходные данные | 8 |
| 2.3 | Текст программы | 10 |
| 2.4 | Эксплуатационные документы | 14 |
| 2.4.1 | Ведомость эксплуатационных документов | 14 |
| 2.4.2 | Описание применения | 14 |
| 2.4.2.1 | Назначение программы | 14 |
| 2.4.2.2 | Условия применения | 15 |
| 2.4.2.3 | Описание задачи | 15 |
| 2.4.2.4 | Входные и выходные данные | 15 |
| 2.4.3 | **Руководство оператора** | 15 |
| 2.4.3.1 | Назначение программы | 15 |
| 2.4.3.2 | Условия выполнения программы | 15 |
| 2.4.3.3 | Выполнение программы | 15 |
| 2.4.3.4 | Выполнение программы | 15 |
| 2.4.3.5 | Сообщения оператору | 15 |
| 2.4.4 | **Описание языка** | 15 |
| 2.4.4.1 | Общие сведения | 15 |
| 2.4.4.2 | Элементы языка | 15 |
| 2.4.4.3 | Средства обмена данными | 15 |
| 2.4.4.4 | Встроенные элементы | 15 |
| 2.4.4.5 | Средства отладки программы | 15 |
|  | Заключение | 16 |
|  | Список использованных источников | 17 |
|  | Приложение А. Ссылочные нормативные документы | 18 |

**введение**

Цель работы: разработка программного обеспечения для редактирования изображений, настройка программного обеспечения.

Условия: создание фигуры, изменение средней яркости при помощи ползунков; масштабирование изображения (масштабирование всего изображения); линейное контрастирование изображения; линейная растяжка гистограммы; пороговая сегментация цветных изображений (YCbCr желтый).

**1. тЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**1.0 Фильтр** «Перемещение прямоугольной области»

Пользователь нажатием левой кнопки мыши выделяет прямоугольную область, после чего нажатием правой кнопки мыши осуществляется перенос выделенной области вблизи курсора.

* 1. **Фильтр** «Вырез участка изображения»

Пользователь задает ширину и высоту, после чего нажатием левой кнопки мыши производится «вырезание» участка изображения. Вырезанная часть изображения выводится в новое окно.

**1.2 Фильтр** «Увеличение масштаба по горизонтали»

Пользователь с помощью ползунка меняет кратность изменения размера изображения по горизонтали.

**1.3 Фильтр** «Гамма-коррекция»

Пользователь с помощью ползунков, количество которых зависит от типа изображения, меняет значение гамма.

**1.4 Фильтр** «Нормализация гистограммы»

При запуске программы на экран пользователя выводится нормализированное изображение и гистограммы каналов этого изображения. Количество гистограмм зависит от типа изображения.

**1.5 Фильтр** «Пороговая сегментация цветных изображений»

С помощью ползунков пользователь изменяет пороги сегментации. Пиксели, значения которых находятся в пределах этих порогов, перекрашиваются в желтый цвет в пространстве цветов YCbCr.

**2. пРограммная документация**

**2.1 Спецификация**

- Компоненты – windows 10, visual studio 22, c++(opencv)

**2.2 Описание программы**

2.2.1 Общие сведения Windows 11, С++(opencv), visual studio 22.

2.2.2 Функциональное назначение Oбработка и изменение изображений с помощью фильтров. Oграничение: обработка и изменение текстовых файлов.

2.2.3 Описание логической структуры Блок-схема алгоритма программы.

2.2.4 Используемые технические средства Компьютер на windows 11 с 8 гб ОЗУ и четырехъядерным процессором.

2.2.5 Вызов и загрузка Способ вызова программы: запуск программы с помощью отладчика, объем оперативной памяти 8 гб, объем программы: 243 строчки кода (35.2 озу).

**2.3 Текст программы**

#include <iostream>

#include <opencv2/core.hpp>

#include <opencv2/imgproc.hpp>

#include <opencv2/highgui.hpp>

#include<algorithm>

#include<vector>

using namespace std;

using namespace cv;

Mat img, tmpImg, tmpImg1;

int h = 1;

int l = 1;

void on\_mouse(int EVENT, int x, int y, int flags, void\* ustc)

{

int x1, y1;

Mat img21(h, l, CV\_8UC3, Scalar(0, 0, 0));

switch (EVENT)

{

case EVENT\_LBUTTONDOWN:

{

x1 = x - (l / 2);

y1 = y - (h / 2);

vector<Mat> channels{};

split(tmpImg, channels);

vector<Mat> channels1{};

split(img21, channels1);

for (int i = 0; i < l; i++)

{

for (int j = 0; j < h; j++)

{

channels1.at(0).at<uchar>(j, i) = channels.at(0).at<uchar>(x1 + j, y1 + i);

channels1.at(1).at<uchar>(j, i) = channels.at(1).at<uchar>(x1 + j, y1 + i);

channels1.at(2).at<uchar>(j, i) = channels.at(2).at<uchar>(x1 + j, y1 + i);

}

}

merge(channels1, img21);

imshow("123", img21);

rectangle(tmpImg, Point(x - (l / 2), y - (h / 2)), Point(x + (l / 2), y + (h / 2)), Scalar(255, 255, 255), FILLED);

imshow("Paint\_img", tmpImg);

break;

}

}

}

Mat img3(1000, 1000, CV\_8UC3, Scalar(255, 255, 255));

vector <int> b;

vector <int> b1;

void on\_mouse1(int EVENT, int x, int y, int flags, void\* ustc)

{

switch (EVENT)

{

case EVENT\_LBUTTONDOWN:

{

b.push\_back(x);

b.push\_back(y);

break;

}

case EVENT\_LBUTTONUP:

{

b1.push\_back(x);

b1.push\_back(y);

vector<Mat> channels{};

split(tmpImg1, channels);

vector<Mat> channels1{};

split(img3, channels1);

for (int i = b[b.size() - 2], i1 = 0; i < x; i++, i1++) {

for (int j = b[b.size() - 1], j1 = 0; j < y; j++, j1++) {

channels1.at(0).at<uchar>(j1, i1) = channels.at(0).at<uchar>(j, i);

channels1.at(1).at<uchar>(j1, i1) = channels.at(1).at<uchar>(j, i);

channels1.at(2).at<uchar>(j1, i1) = channels.at(2).at<uchar>(j, i);

channels.at(0).at<uchar>(j, i) = 255;

channels.at(1).at<uchar>(j, i) = 255;

channels.at(2).at<uchar>(j, i) = 255;

}

}

merge(channels, tmpImg1);

imshow("Paint\_img1", tmpImg1);

merge(channels1, img3);

break;

}

case EVENT\_RBUTTONDOWN: {

vector<Mat> channels{};

split(tmpImg1, channels);

vector<Mat> channels1{};

split(img3, channels1);

for (int i = x, i1 = 0; i < (x + b1[b1.size() - 2] - b[b.size() - 2]); i++, i1++) {

for (int j = y, j1 = 0; j < (y + b1[b1.size() - 1] - b[b.size() - 1]); j++, j1++) {

channels.at(0).at<uchar>(j, i) = channels1.at(0).at<uchar>(j1, i1);

channels.at(1).at<uchar>(j, i) = channels1.at(1).at<uchar>(j1, i1);

channels.at(2).at<uchar>(j, i) = channels1.at(2).at<uchar>(j1, i1);

}

}

merge(channels, tmpImg1);

imshow("Paint\_img1", tmpImg1);

break;

}

}

}

int main()

{

setlocale(LC\_ALL, "rus");

int n;

cout << "Введите номер фильтра" << endl;

cout << "0. Перемещение прямоугольной области" << endl << "1. Вырез участка изображения" << endl << "2. Увеличение масштаба по горизонтали" << endl;

cout << "3. Гамма-коррекция" << endl << "4. Нормализация гистограммы " << endl << "5. Пороговая сегментация" << endl;

cin >> n;

cout << "Введите номер желаемого изображения(1-цветное,2-черно-белое):" << endl;

int k;

cin >> k;

if (k == 1)

img = imread("09.bmp");

if (k == 2)

img = imread("009.bmp");

switch (n)

{

case 0:

{

img = imread("9.jpg");

tmpImg1 = img.clone();

imshow("Paint\_img1", tmpImg1);

while (true)

{

setMouseCallback("Paint\_img1", on\_mouse1, (void\*)(&tmpImg1));

waitKey(0);

}

destroyWindow("Paint\_img1");

break;

}

case 1:

{

tmpImg = img.clone();

imshow("Paint\_img", tmpImg);

createTrackbar("Высота", "Paint\_img", &h, 300);

createTrackbar("Ширина", "Paint\_img", &l, 300);

while (true)

{

setMouseCallback("Paint\_img", on\_mouse, (void\*)(&tmpImg));

waitKey(0);

}

break;

}

case 2:

{

Mat img1;

int k = 0;

namedWindow("Уменьшение размера", WINDOW\_AUTOSIZE);

createTrackbar("Кратность", "Уменьшение размера", &k, min(img.size().width, img.size().height));

while (true)

{

resize(img, img1, Size(img.size().width + k, img.size().height), 0, 0, INTER\_LINEAR);

if ((img.size().width - k) <= 0 or (img.size().height - k) <= 0)

break;

imshow("Уменьшение размера", img1);

waitKey(1);

}

break;

}

case 3:

{

if (k == 1)

{

img = imread("09.bmp");

Mat img2 = img.clone();

namedWindow("Гамма-Коррекция");

vector<Mat> channels{};

split(img, channels);

int qr = 10, qg = 10, qb = 10;

createTrackbar("Синий", "Гамма-Коррекция", &qr, 20);

createTrackbar("Зелёный", "Гамма-Коррекция", &qg, 20);

createTrackbar("Красный", "Гамма-Коррекция", &qb, 20);

while (true)

{

double qrr = static\_cast<double>(qr) / 10;

double qgg = static\_cast<double>(qg) / 10;

double qbb = static\_cast<double>(qb) / 10;

vector<Mat> channels{};

split(img2, channels);

vector<Mat> channelss{};

split(img, channelss);

for (int i = 0; i < img.rows; i++)

{

for (int j = 0; j < img.cols; j++)

{

if (pow(channelss.at(0).at<uchar>(i, j), qrr) < 255)

{

channelss.at(0).at<uchar>(i, j) = pow(channels.at(0).at<uchar>(i, j), qrr);

}

else

channelss.at(0).at<uchar>(i, j) = 255;

if (pow(channelss.at(1).at<uchar>(i, j), qgg) < 255)

{

channelss.at(1).at<uchar>(i, j) = pow(channels.at(1).at<uchar>(i, j), qgg);

}

else

channelss.at(1).at<uchar>(i, j) = 255;

if (pow(channelss.at(2).at<uchar>(i, j), qrr) < 255)

{

channelss.at(2).at<uchar>(i, j) = pow(channels.at(2).at<uchar>(i, j), qbb);

}

else

channelss.at(2).at<uchar>(i, j) = 255;

}

}

merge(channelss, img);

imshow("Гамма-Коррекция", img);

waitKey(1);

}

}

else

{

img = imread("009.bmp", 0);

Mat img2 = img.clone();

namedWindow("Гамма-Коррекция");

int qr = 10;

createTrackbar("Гамма", "Гамма-Коррекция", &qr, 20);

while (true)

{

double qrr = static\_cast<double> (qr) / 10;

for (int i = 0; i < img.rows; i++)

{

for (int j = 0; j < img.cols; j++)

{

if (pow(img2.at<uchar>(i, j), qrr) < 255)

{

img.at<uchar>(i, j) = pow(img2.at<uchar>(i, j), qrr);

}

else

img.at<uchar>(i, j) = 255;

}

}

imshow("Гамма-Коррекция", img);

waitKey(1);

}

}

break;

}

case 4:

{

if (k == 1)

{

img = imread("09.bmp");

Mat histR, histG, histB, hist, img2;

int channels[] = { 0 };

int histSize[] = { 256 };

float hr1[] = { 0,255 };

const float\* ranges[] = { hr1 };

vector<Mat> ch;

split(img, ch);

calcHist(&ch.at(2), 1, channels, Mat(), histR, 1, histSize, ranges, true, false);

calcHist(&ch.at(1), 1, channels, Mat(), histG, 1, histSize, ranges, true, false);

calcHist(&ch.at(0), 1, channels, Mat(), histB, 1, histSize, ranges, true, false);

Mat histimgR(600, 900, CV\_8UC3, Scalar(0, 0, 0));

Mat histimgG(600, 900, CV\_8UC3, Scalar(0, 0, 0));

Mat histimgB(600, 900, CV\_8UC3, Scalar(0, 0, 0));

normalize(histR, histR, 0, 255, NORM\_MINMAX, -1, Mat());

normalize(histG, histG, 0, 255, NORM\_MINMAX, -1, Mat());

normalize(histB, histB, 0, 255, NORM\_MINMAX, -1, Mat());

for (int i = 0; i < 256; i++)

{

line(histimgR, Point(i, 600), Point(i, 600 - cvRound(histR.at<float>(i))), Scalar(255, 255, 255), 1, LINE\_AA);

line(histimgG, Point(i, 600), Point(i, 600 - cvRound(histG.at<float>(i))), Scalar(255, 255, 255), 1, LINE\_AA);

line(histimgB, Point(i, 600), Point(i, 600 - cvRound(histB.at<float>(i))), Scalar(255, 255, 255), 1, LINE\_AA);

}

normalize(img, img, 0, 255, NORM\_MINMAX);

imshow("img", img);

imshow("HistR", histimgR);

imshow("HistG", histimgG);

imshow("HistB", histimgB);

waitKey(0);

}

else

{

img = imread("009.bmp", IMREAD\_GRAYSCALE);

Mat img4 = img.clone();

Mat hist;

int channels[] = { 0 };

int histSize[] = { 256 };

float hr1[] = { 0,255 };

const float\* ranges[] = { hr1 };

calcHist(&img, 1, channels, Mat(), hist, 1, histSize, ranges, true, false);

Mat histimg(600, 900, CV\_8UC3, Scalar(0, 0, 0));

normalize(hist, hist, 0, 255, NORM\_MINMAX, -1, Mat());

for (int i = 0; i < 256; i++)

{

line(histimg, Point(i, 600), Point(i, 600 - cvRound(hist.at<float>(i))), Scalar(255, 255, 255), 1, LINE\_AA);

}

imshow("Hist", histimg);

normalize(img, img4, 0, 255, NORM\_MINMAX);

imshow("new", img4);

waitKey(0);

}

break;

}

case 5:

{

Mat img1;

Mat img2(Size(img.size()), CV\_8UC3, Scalar(0));

cvtColor(img, img1, COLOR\_RGB2YCrCb);

int y\_min = 0, y\_max = 0, Cr\_min = 0, Cr\_max = 0, Cb\_min = 0, Cb\_max = 0;

namedWindow("picha", WINDOW\_AUTOSIZE);

createTrackbar("y\_min", "picha", &y\_min, 1000);

createTrackbar("y\_max", "picha", &y\_max, 1000);

createTrackbar("Cr\_min", "picha", &Cr\_min, 1000);

createTrackbar("Cr\_max", "picha", &Cr\_max, 1000);

createTrackbar("Cb\_min", "picha", &Cb\_min, 1000);

createTrackbar("Cb\_max", "picha", &Cb\_max, 1000);

while (true)

{

Mat img1;

img1 = img.clone();

cvtColor(img1, img1, COLOR\_BGR2YCrCb);

vector<Mat> channels{};

split(img1, channels);

for (int i = 0; i < img1.rows; i++)

{

for (int j = 0; j < img1.cols; j++)

{

if (channels.at(0).at<uchar>(i, j) > y\_min && channels.at(0).at<uchar>(i, j) < y\_max &&

channels.at(1).at<uchar>(i, j) > Cr\_min && channels.at(1).at<uchar>(i, j) < Cr\_max &&

channels.at(2).at<uchar>(i, j) > Cb\_min && channels.at(2).at<uchar>(i, j) < Cb\_max)

{

channels.at(0).at<uchar>(i, j) = 100;

channels.at(1).at<uchar>(i, j) = 97;

channels.at(2).at<uchar>(i, j) = 115;

}

}

}

merge(channels, img1);

cvtColor(img1, img1, COLOR\_YCrCb2BGR);

imshow("picha", img1);

waitKey(1);

}

}

}

}

**2.4 Эксплуатационные документы**

**2.4.1 Ведомость эксплуатационных документов**

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение | Наименование | Кол. экз. | Местонахождение |
|  | Документы на программу |  |  |
| А.В.ХХХХХ-ХХ 31 ХХ-ХХ | Обработка изображений  Описание применения | 1 | Каф. ТВ |
| А.В.ХХХХХ-ХХ 34 ХХ-ХХ | Обработка изображений  Руководство оператора | 1 | Каф. ТВ |
| А.В.ХХХХХ-ХХ 35 ХХ-ХХ | Обработка изображений  Описание языка | 1 | Каф. ТВ |

**2.4.2 Описание применения**

2.4.2.1 Назначение программы Назначение: обработка и изменение изображений при помощи фильтров. Возможности ПО: создание изображения и их обработка. Ограничения: создание и обработка текстовых файлов.

2.4.2.2 Условия применения Загрузка файлов в формате png, jpg. Минимальный объем оперативной памяти ПК 4гб, четырехъядерный процессор, windows 10,11, место на жестком диске не менее 1 гб свободного места.

2.4.2.3 Описание задачи Изменение изображения c применением различных фильтров. Метод решения - использовать фильтры и библиотеку opencv.

2.4.2.4 Входные и выходные данные: изображения.

**2.4.3 Руководство оператора**

2.4.3.1 Назначение программы Назначение: обработка изображений с помощью фильтров. Возможности ПО: создание изображения и обработка, вывод результата изображения. Ограничения: создание и обработка текстовых файлов.

2.4.3.2 Условия выполнения программы Компьютер с 4 гб ОЗУ, установленный visual studio 2022, установленный c++, подключенная библиотека opencv.

2.4.3.3 Выполнение программы Последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, вывод результата, описание функций, формата, возможных вариантов команд и ответы программы на эти команды.

2.4.3.4 Сообщения оператору В случае загрузки изображения не того формата компилятор выдаст ошибку открытия изображения. В случае попытки открытия текстового файла компилятор выдаст ошибку.

**2.4.4 Описание языка**

2.4.4.1 Общие сведения Язык предназначен для решения поставленных вопросов. Главными отличительными чертами являются: применение условных операторов, разнообразные шаблоны и функции, возможность использования различных библиотек. С++ используется с целью создания и обработки изображений, 3D проектов, для работы с крупными объемами данных.

2.4.4.2 Элементы языка В программах используется стандартный синтаксис с++.

2.4.4.3 Средства обмена данными Cтандартные файлы ввода и вывода (#include <iostream>).

2.4.4.4 Встроенные элементы Подключены библиотеки (opencv, iostream), библиотека opencv - открытая библиотека с целью работы с алгоритмами компьютерного зрения, машинным обучением и обработкой изображений, iostream - заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++, vector - библиотека для работы с векторами.

2.4.4.5 Средства отладки программы: Пошаговое перемещение по коду и просмотр значений, хранящихся в переменных, задание контрольных значений с целью отслеживания изменений значений, изучение пути выполнения программного кода и т.д.

**заключение**

В ходе выполнения курсовой работы создано программное обеспечение для обработки изображений и вывода результата обработанного изображения. Выполнен запуск программы, написанной на языке программирования С++ с применением вышеперечисленных фильтров и использовании библиотеки OpenCV.

**список использованных источников**

# Книга Практический OpenCV (технология в действии).

# [Самарт Брамбхатт](https://www.amazon.com/Samarth-Brahmbhatt/e/B00IXSSZD0/ref=dp_byline_cont_book_1).: Apress; 1-е изд. издание (20 ноября 2013). 256 с.

# 2. Книга Практическое введение в компьютерное зрение с помощью OpenCV (Wiley-IS & T Series in Imaging Science and Technology). [Кеннет Доусон-Хоу](https://www.amazon.com/Kenneth-Dawson-Howe/e/B00K1I80QW/ref=dp_byline_cont_ebooks_1).: 1-е издание, издание Kindle. 304 с.

**приложение А**

**ССылочные нормативные документы**

|  |  |
| --- | --- |
| Название | Номер раздела, подраздела, в котором используется |
| ГОСТ 19.202-78.ЕСПД.Спецификация | 2.1 |
| ГОСТ 19.401-78.ЕСПД.Текст программы | 2.3 |
| ГОСТ 19.402-78.ЕСПД.Описание программы | 2.2 |
| ГОСТ 19.502-78.ЕСПД.Описание применения | 2.4.2 |
| ГОСТ 19.505-79.ЕСПД.Руководство оператора | 2.4.3 |
| ГОСТ 19.506-79.ЕСПД.Описание языка | 2.4.4 |
| ГОСТ 19.701-90.ЕСПД.Схемы алгоритмов, программ, данных и систем | 2.2.3 |
| ГОСТ 19.507-79.ЕСПД.Ведомость эксплуатационных документов | 2.4.1 |