Санкт-Петербургский политехнический университет Петра Великого   
Высшая школа программной инженерии

**ОТЧЕТ ПО**

**ИНДИВИДУАЛЬНОЙ РАБОТЕ**

По дисциплине «ЭВМ и периферийные устройства»

«Пороговое преобразование и сегментация готовых изображений и кадров с web-камеры»

Выполнил  
студент гр. 23531/21 А.В. Терехин

Руководитель С.А. Молодяков

«I22i» в маяiiii 2018 г.

Санкт-Петербург

2018

ЗАДАНИЕ

НА ВЫПОЛНЕНИЕ ИНДИВИДУАЛЬНОЙ РАБОТЫ

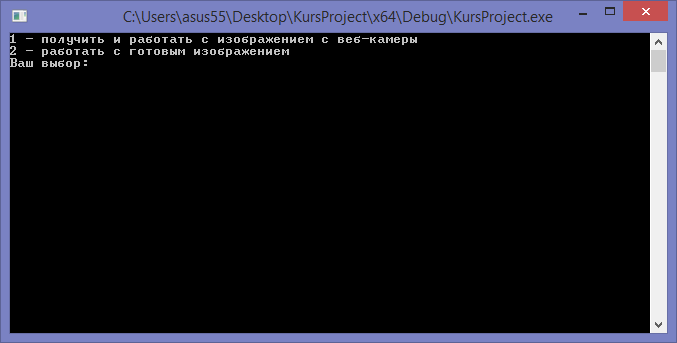
студенту группы 23531/21 Терехину Андрею Валерьевичу

*(номер группы) (фамилия, имя, отчество)*

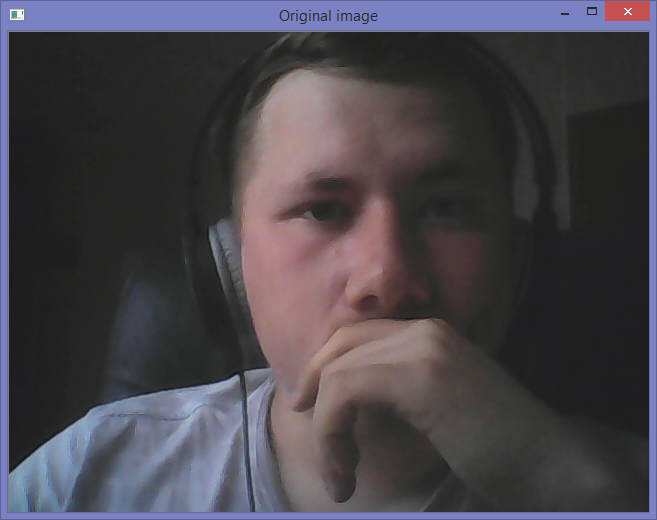
Программа для порогового преобразования и сегментации готовых изображений и кадров с web-камеры. Данная программа при выборе пользователя получает изображение с web-камеры или использует готовое изображение для дальнейшей работы с ним. После выбора способа получения изображения программа создаёт окно с оригинальным изображением и выводит меню команд в консоли. При нажатии определённой клавиши происходит создание нового окна с результатом выбранного преобразования. При манипуляции мышью можно изменять свойства выбранного преобразования. В случае изменения изображения создаётся новый файл с изменениями в папке с оригиналом.

**Результаты работы программы**

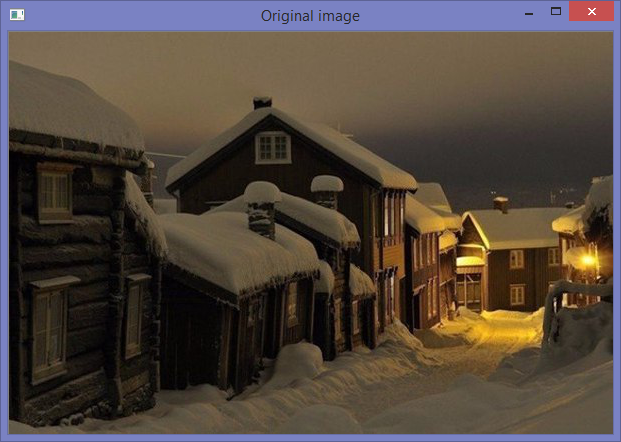
При запуске программы происходит вывод окна с выбором способа получения изображения, с которым в дальнейшем будут проводиться преобразования



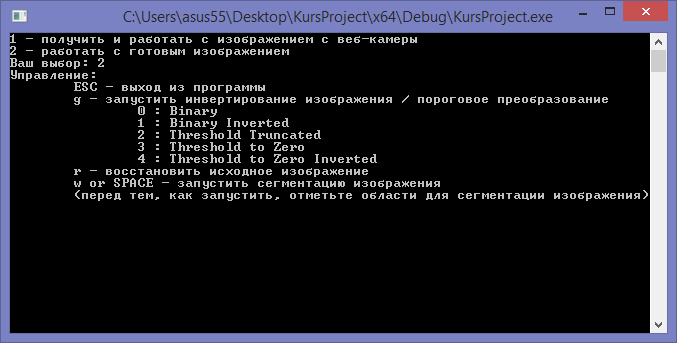
В случае выбора первого пункта появляется окно с полученным изображением с web-камеры



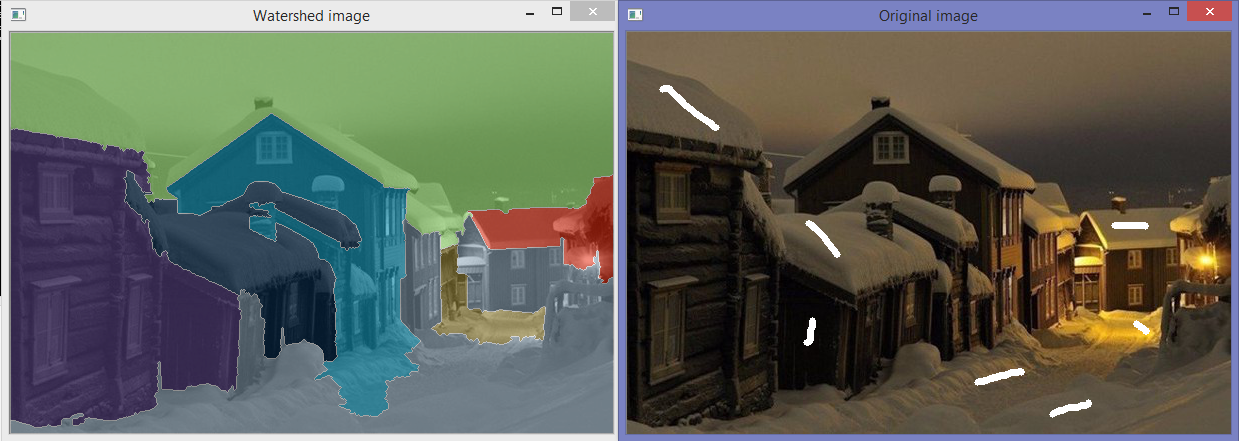
В случае выбора второго пункта открывается файл с подготовленным изображением

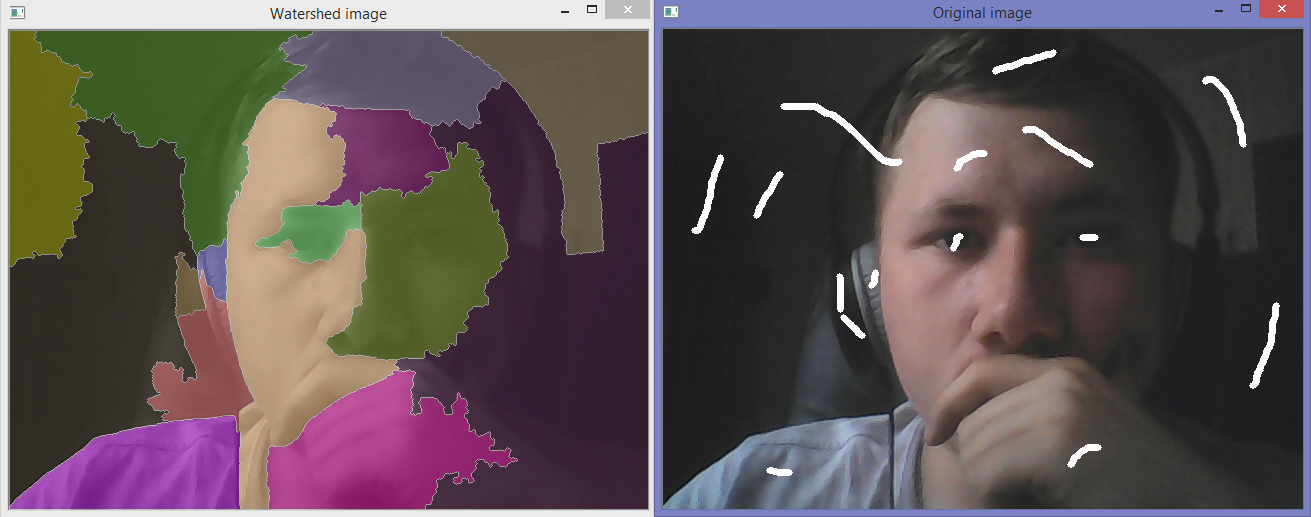


Помимо окна с оригиналом отображается меню управления

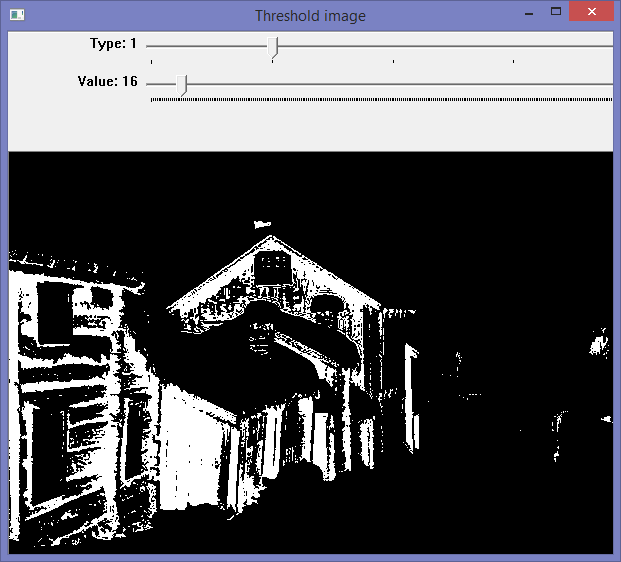
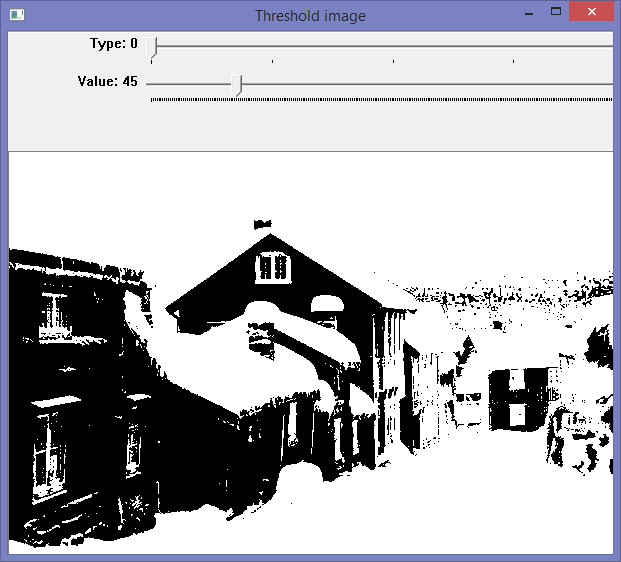


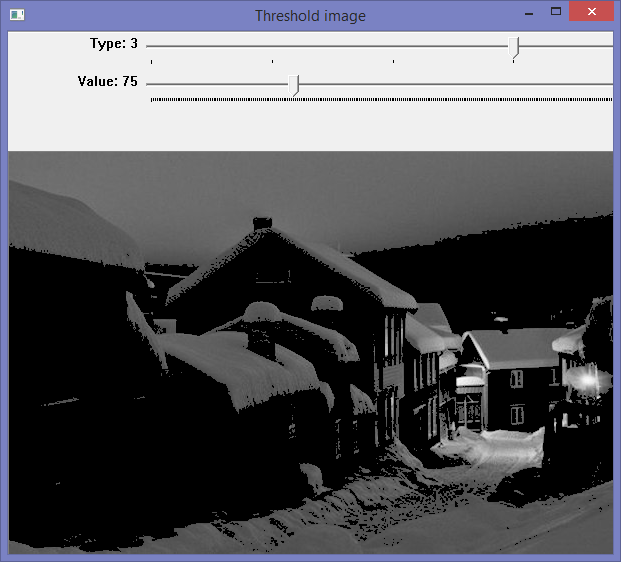
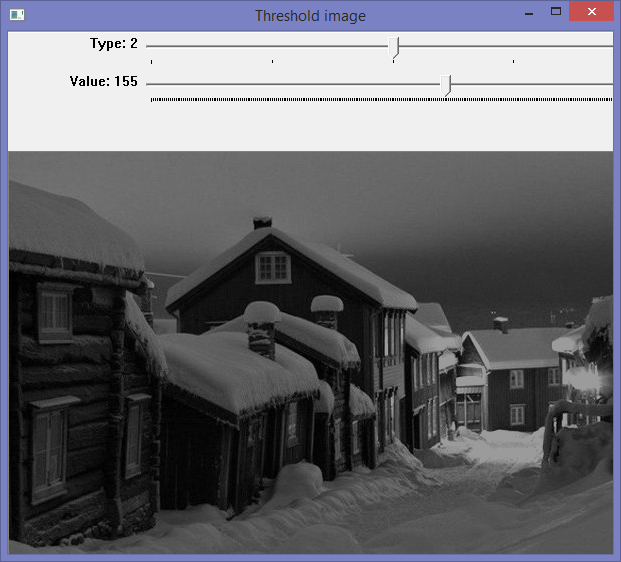
Сегментация изображений:

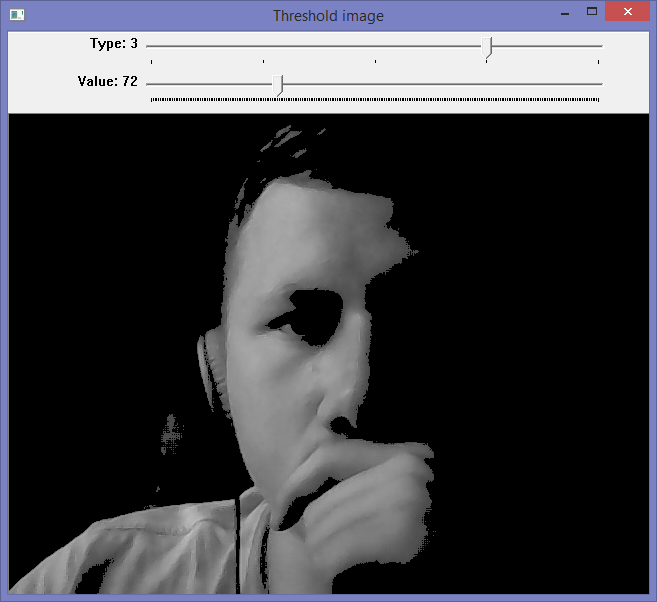
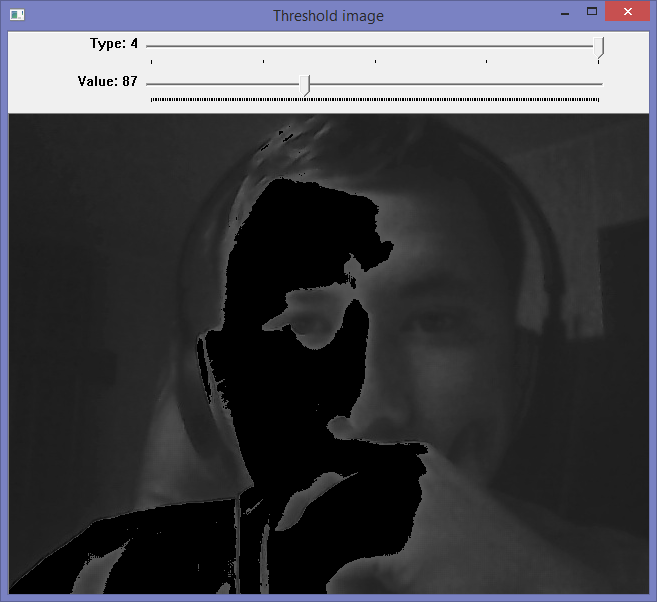


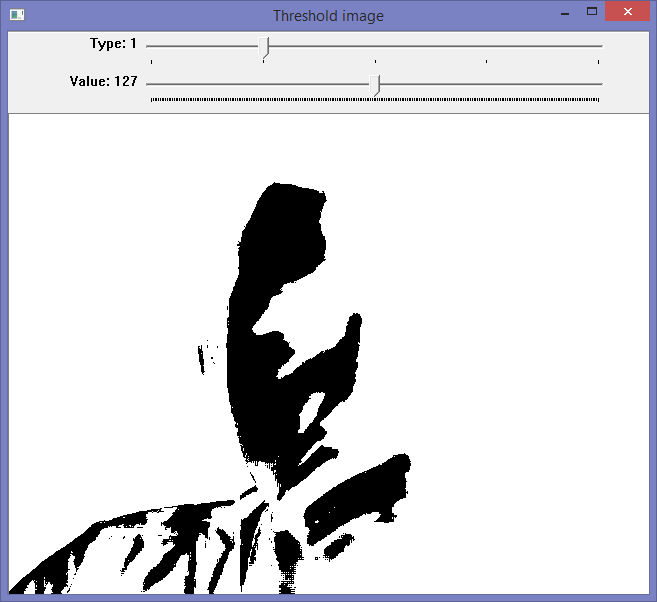
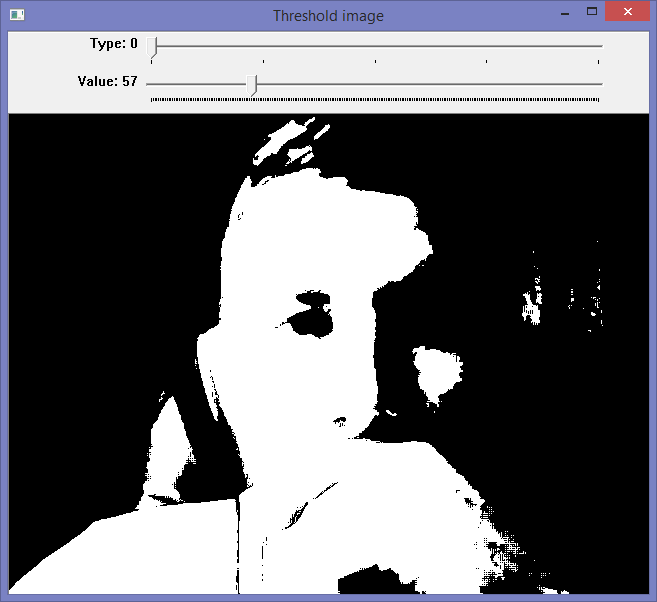


Пороговое преобразование изображений:









В данной программе была использована библиотека «OpenCV».

**Что такое OpenCV?**

**OpenCV** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Open Source Computer Vision Library*, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов [компьютерного зрения](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B7%D1%80%D0%B5%D0%BD%D0%B8%D0%B5), [обработки изображений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) и численных алгоритмов общего назначения с [открытым кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

**В программе используются следующие модули библиотеки OpenCV:**

В версии 2.2 библиотека была реорганизована. Вместо универсальных модулей cxcore, cvaux, highGUI и других было создано несколько компактных модулей с более узкой специализацией:

* opencv\_core — основная функциональность. Включает в себя базовые структуры, вычисления (математические функции, генераторы случайных чисел) и линейную алгебру, [DFT](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A4%D1%83%D1%80%D1%8C%D0%B5), [DCT](https://ru.wikipedia.org/wiki/DCT), ввод/вывод для XML и YAML и т. д.
* opencv\_imgproc — обработка изображений (фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.).
* opencv\_highgui — простой UI, ввод/вывод изображений и видео.
* opencv\_ml — модели машинного обучения (SVM, деревья решений, обучение со стимулированием и т. д.).
* opencv\_features2d — распознавание и описание плоских примитивов ([SURF](https://en.wikipedia.org/wiki/SURF) (англ.)[русск.](https://ru.wikipedia.org/w/index.php?title=SURF&action=edit&redlink=1), FAST и другие, включая специализированный фреймворк).
* opencv\_video — анализ движения и отслеживание объектов ([оптический поток](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA), шаблоны движения, устранение фона).
* opencv\_objdetect — обнаружение объектов на изображении (нахождение лиц с помощью [алгоритма Виолы-Джонса](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%92%D0%B8%D0%BE%D0%BB%D1%8B_%E2%80%94_%D0%94%D0%B6%D0%BE%D0%BD%D1%81%D0%B0), распознавание людей HOG и т. д.).
* opencv\_calib3d — калибровка камеры, поиск стерео-соответствия и элементы обработки трёхмерных данных.
* opencv\_flann — библиотека быстрого поиска ближайших соседей (FLANN 1.5) и обертки OpenCV.
* opencv\_contrib — сопутствующий код, ещё не готовый для применения.
* opencv\_legacy — устаревший код, сохранённый ради обратной совместимости.
* opencv\_gpu — ускорение некоторых функций OpenCV за счет [CUDA](https://ru.wikipedia.org/wiki/CUDA), создан при поддержке [NVidia](https://ru.wikipedia.org/wiki/NVidia" \o "NVidia).

В данной программе используются следующие модули: opencv\_core, opencv\_imgproc и opencv\_highgui.

**В программе используются следующие функции OpenCV:**

|  |  |
| --- | --- |
| imshow | Отображение результата |
| imwrite | Запись в файл |
| imread | Чтение из файла |
| cvtColor | Конвертация матрицы |
| createTrackbar | Создание trackbar |
| threshold | Пороговое преобразование |
| findContours | Поиск контуров |
| drawContours | Отображение контуров |
| watershed | Сегментация |

**Код программы:**

#include <opencv2/core/utility.hpp>

#include <opencv2/imgproc.hpp>

#include <opencv2/imgcodecs.hpp>

#include <opencv2/highgui.hpp>

#include <cstdio>

#include <iostream>

#include <conio.h>

using namespace cv;

using namespace std;

static void help()

{

cout << "Управление: \n"

"\tESC - выход из программы\n"

"\tg - запустить инвертирование изображения / пороговое преобразование\n"

"\t\t0 : Binary\n"

"\t\t1 : Binary Inverted\n"

"\t\t2 : Threshold Truncated\n"

"\t\t3 : Threshold to Zero\n"

"\t\t4 : Threshold to Zero Inverted\n"

"\tr - восстановить исходное изображение\n"

"\tw or SPACE - запустить сегментацию изображения\n"

"\t(перед тем, как запустить, отметьте области для сегментации изображения)\n";

}

Mat markerMask, img;

Point prevPt(-1, -1);

int threshold\_value = 0;

int threshold\_type = 3;

int const max\_value = 255;

int const max\_type = 4;

int const max\_BINARY\_value = 255;

string filename\_main;

Mat src, src\_gray, dst;

char\* window\_name = "Threshold image";

char\* trackbar\_type = "Type";

char\* trackbar\_value = "Value";

void Threshold\_Im(int, void\*);

static void onMouse(int event, int x, int y, int flags, void\*)

{

if (x < 0 || x >= img.cols || y < 0 || y >= img.rows)

return;

if (event == EVENT\_LBUTTONUP || !(flags & EVENT\_FLAG\_LBUTTON))

prevPt = Point(-1, -1);

else if (event == EVENT\_LBUTTONDOWN)

prevPt = Point(x, y);

else if (event == EVENT\_MOUSEMOVE && (flags & EVENT\_FLAG\_LBUTTON))

{

Point pt(x, y);

if (prevPt.x < 0)

prevPt = pt;

line(markerMask, prevPt, pt, Scalar::all(255), 5, 8, 0);

line(img, prevPt, pt, Scalar::all(255), 5, 8, 0);

prevPt = pt;

imshow("Original image", img);

}

}

int main(int argc, char\*\* argv)

{

setlocale(LC\_CTYPE, "rus");

int select = 1;

cout << "1 - получить и работать с изображением с веб-камеры\n2 - работать с готовым изображением\nВаш выбор: ";

cin >> select;

if (!cin)

{

cout << "Ошибка ввода!";

\_getch();

exit(0);

}

if (select == 1)

{

VideoCapture cap(0);

// Получение кадра с веб-камеры

Mat save\_img; cap >> save\_img;

if (save\_img.empty())

{

std::cerr << "Ошибка в работе веб-камеры, нет возможности получить изображение!" << std::endl;

}

// Сохранение кадра в файл

filename\_main = "test\_web.jpg";

imwrite(filename\_main, save\_img);

}

else if (select == 2)

{

filename\_main = "test.jpg";

}

else

{

cout << "Ошибка ввода!";

\_getch();

exit(0);

}

cv::CommandLineParser parser(argc, argv, "{help h | | }{ @input | test.jpg | }");

if (parser.has("help"))

{

help();

return 0;

}

Mat img0 = imread(filename\_main, 1), imgGray;

if (img0.empty())

{

cout << "Невозможно открыть файл " << filename\_main << ".\n";

\_getch();

return 0;

}

help();

namedWindow("Original image", 1);

img0.copyTo(img);

cvtColor(img, markerMask, COLOR\_BGR2GRAY);

cvtColor(markerMask, imgGray, COLOR\_GRAY2BGR);

markerMask = Scalar::all(0);

imshow("Original image", img);

setMouseCallback("Original image", onMouse, 0);

for (;;)

{

int c = waitKey(0);

if ((char)c == 27)

exit(0);

if ((char)c == 'r')

{

markerMask = Scalar::all(0);

img0.copyTo(img);

imshow("Original image", img);

}

if ((char)c == 'g')

{

src = imread(filename\_main, 1);

/// Конвертирование изображения в серый

cvtColor(img, src\_gray, CV\_BGR2GRAY);

/// Создание окна для отображения результатов

namedWindow(window\_name, CV\_WINDOW\_AUTOSIZE);

/// Создание Trackbar для выбора типа порогового преобразования

createTrackbar(trackbar\_type,

window\_name, &threshold\_type,

max\_type, Threshold\_Im);

createTrackbar(trackbar\_value,

window\_name, &threshold\_value,

max\_value, Threshold\_Im);

/// Вызов функции инициализации

Threshold\_Im(0, 0);

}

if ((char)c == 'w' || (char)c == ' ')

{

int i, j, compCount = 0;

vector<vector<Point> > contours;

vector<Vec4i> hierarchy;

findContours(markerMask, contours, hierarchy, RETR\_CCOMP, CHAIN\_APPROX\_SIMPLE);

if (contours.empty())

continue;

Mat markers(markerMask.size(), CV\_32S);

markers = Scalar::all(0);

int idx = 0;

for (; idx >= 0; idx = hierarchy[idx][0], compCount++)

drawContours(markers, contours, idx, Scalar::all(compCount + 1), -1, 8, hierarchy, INT\_MAX);

if (compCount == 0)

continue;

vector<Vec3b> colorTab;

for (i = 0; i < compCount; i++)

{

int b = theRNG().uniform(0, 255);

int g = theRNG().uniform(0, 255);

int r = theRNG().uniform(0, 255);

colorTab.push\_back(Vec3b((uchar)b, (uchar)g, (uchar)r));

}

double t = (double)getTickCount();

watershed(img0, markers);

t = (double)getTickCount() - t;

printf("Время выполнения = %gms\n", t\*1000. / getTickFrequency());

Mat wshed(markers.size(), CV\_8UC3);

// создание сегментированного изображения

for (i = 0; i < markers.rows; i++)

for (j = 0; j < markers.cols; j++)

{

int index = markers.at<int>(i, j);

if (index == -1)

wshed.at<Vec3b>(i, j) = Vec3b(255, 255, 255);

else if (index <= 0 || index > compCount)

wshed.at<Vec3b>(i, j) = Vec3b(0, 0, 0);

else

wshed.at<Vec3b>(i, j) = colorTab[index - 1];

}

wshed = wshed\*0.5 + imgGray\*0.5;

imshow("Watershed image", wshed);

imwrite("result\_watershed.jpg", wshed);

}

}

\_getch();

return 0;

}

void Threshold\_Im(int, void\*)

{

threshold(src\_gray, dst, threshold\_value, max\_BINARY\_value, threshold\_type);

imshow(window\_name, dst);

imwrite("result\_threshold.jpg", dst);

}