###### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

###### Факультет информационных технологий

**Кафедра параллельных вычислений**

ОТЧЕТ

О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ

«Высокоуровневая работа с периферийными устройствами»

студента 2 курса, группы 20204

**Дронова Дениса Юрьевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Кандидат технических наук

А. Ю. Власенко

Новосибирск 2021

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc87189037)

[ЗАДАНИЕ 3](#_Toc87189038)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc87189039)

[ЗАКЛЮЧЕНИЕ 6](#_Toc87189040)

[Приложение 1. *Работа с OpenCV* 7](#_Toc87189041)

[Приложение 2. *Работа с LibUSB* 9](#_Toc87189042)

# ЦЕЛЬ

1. Ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV.
2. Ознакомиться с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.

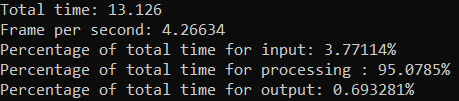
# ЗАДАНИЕ

1. Реализовать программу№1 с использованием OpenCV, которая получает поток видеоданных с камеры и выводит его на экран.
2. Выполнить произвольное преобразование изображения (кроме указанных в computerlab5.pdf сглаживания и установки значений цветовых каналов в константу).
3. Измерить количество кадров, обрабатываемое программой в секунду.
4. Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.
5. Реализовать программу №2, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатать его класс, идентификатор производителя, идентификатор изделия и серийный номер.

# ОПИСАНИЕ РАБОТЫ

**Часть 1**

Для выполнения лабораторной работы была выбрана версия OpenCV 4.5. Была написана программа на C++, выводящая изображение со встроенной (подключенной) веб-камеры в отдельное окно. В качестве преобразования видео был выбран эффект выделения границ на изображении. В последующем из исходного изображения было вычтено изображение, состоящие из границ. Также был реализован поворот изображения на произвольный угол. При измерении времени работы определенных участков программы из Приложения 1 (считывание, обработка, вывод), было получено значение FPS ≈ 4.



**Часть 2**

В ходе выполнения второй части работы был получен список устройств подключенных по USB к кафедральному компьютеру и были выведены класс устройства, id производителя, id устройства и серийный номер устройства. В полученном списке было 9 устройств, среди которых были обнаружены клавиатура, оптическая мышь, веб-камера. Для этого была реализована программа на C++, приведенная в Приложении 2.

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описаниеСтрока компиляции: g++ -o executable libUSB.cpp -lusb-1.0

# ЗАКЛЮЧЕНИЕ

В ходе выполнения первой части лабораторной работы были изучены принципы взаимодействия с веб-камерой с использованием библиотеки OpenCV. Были произведены замеры времени работы различных участков программы, и с помощью этих измерений получено значение frames per second (FPS). Также были изучены способы обработки и хранения изображений. В ходе выполнения второй части лабораторной работы были изучены способы работы с устройствами USB при помощи библиотеки libusb, а конкретно получение различной информации о подключенных к компьютеру устройств по интерфейсу USB.

# Приложение 1. *Работа с OpenCV*

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/opencv.hpp>

#include <iostream>

#include <ctime>

using namespace cv;

using namespace std;

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

{

    clock\_t time;

    clock\_t input\_time;

    clock\_t processing\_time;

    clock\_t output\_time;

    clock\_t input\_counter = 0;

    clock\_t processing\_counter = 0;

    clock\_t output\_counter = 0;

    double counter = 0;

    Mat frame; //empty matrix for storing webcam frames

    VideoCapture cap(0);

    if (!cap.isOpened())

    {

        return -1;

    }

    time = clock();

    while (true)

    {

        //input

        input\_time = clock();

        cap >> frame; //inserting the webcam frame inside frame matrix

        input\_time = clock() - input\_time;

        input\_counter += input\_time;

        processing\_time = clock();

        Mat edges;

        cvtColor(frame, edges, COLOR\_BGR2GRAY);

        Canny(edges, edges, 30, 60, 5);

        Mat sub;

        cvtColor(frame, frame, COLOR\_BGR2GRAY);

        subtract(frame, edges, sub);

        //imshow("Webcam", sub);

        //ROTATING

        double angle = 45;

        // getting the center coordinates of the image to create the 2D rotation matrix

        Point2f center((frame.cols - 1) / 2.0, (frame.rows - 1) / 2.0);

        //getting the rotation matrix

        Mat rotation\_matix = getRotationMatrix2D(center, angle, 1.0);

        Mat rotated\_webcam;

        // rotating the sub using rotation matrix

        warpAffine(sub, rotated\_webcam, rotation\_matix, edges.size());

        processing\_time = clock() - processing\_time;

        processing\_counter += processing\_time;

        output\_time = clock();

        imshow("Webcam", rotated\_webcam);

        output\_time = clock() - output\_time;

        output\_counter += output\_time;

        counter++;

        if (waitKey(33) == 27)

            break;

    }

    time = clock() - time - counter \* 33;

    double total\_time = (((double)time / CLOCKS\_PER\_SEC));

    double FPS = counter / (((double)time / CLOCKS\_PER\_SEC));

    double res\_input\_time = ((double)input\_counter / CLOCKS\_PER\_SEC);

    double res\_processing\_time = ((double)processing\_counter / CLOCKS\_PER\_SEC);

    double res\_output\_time = ((double)output\_counter / CLOCKS\_PER\_SEC);

    cout << "Total time: " << total\_time << endl;

    cout << "Frame per second: " << FPS << endl;

    cout << "Percentage of total time for input: " << res\_input\_time / total\_time \* 100 << "%" << endl;

    cout << "Percentage of total time for processing : " << res\_processing\_time / total\_time \* 100 << "%" << endl;

    cout << "Percentage of total time for output: " << res\_output\_time / total\_time \* 100 << "%"<< endl;

    return 0;

}

# Приложение 2. *Работа с LibUSB*

#include <iostream>

#include <libusb-1.0/libusb.h>

#include <stdio.h>

using namespace std;

void printdev(libusb\_device \*dev);

int main()

{

    libusb\_device \*\*devs;       // указатель на указатель на устройство

    libusb\_context \*ctx = NULL; // контекст сессии libusb

    int r;                      // для возвращаемых значений

    r = libusb\_init(&ctx);

    if (r < 0)

    {

        cout << "Initializing failed with code: " << r << endl;

        return 1;

    }

    // задать уровень подробности отладочных сообщений

    libusb\_set\_debug(ctx, 3);

    // получить список всех найденных USB- устройств

    size\_t cnt = libusb\_get\_device\_list(ctx, &devs);

    if (cnt < 0)

    {

        cout << "Failed to get device list " << r << endl;

        return 1;

    }

    cout << "Number of found devices: " << cnt << endl;

    for (size\_t i = 0; i < cnt; i++)

    {                      // цикл перебора всех устройств

        printdev(devs[i]); // печать параметров устройства

    }

    // освободить память, выделенную функцией получения списка устройств

    libusb\_free\_device\_list(devs, 1);

    libusb\_exit(ctx); // завершить работу с библиотекой libusb,

    // закрыть сессию работы с libusb

    return 0;

}

void printdev(libusb\_device \*dev)

{

    libusb\_device\_descriptor desc; // дескриптор устройства

    int r = libusb\_get\_device\_descriptor(dev, &desc);

    if (r < 0)

    {

        cout << "Failed to get device descriptor " << r << endl;

        return;

    }

    libusb\_device\_handle \*dh;

    unsigned char data[1000] = {0};

    r = libusb\_open(dev, &dh);

    if (r < 0)

    {

        cout << "Failed to open device " << endl

             << endl;

        return;

    }

    cout << "\* Number of possible configurations: " << (int)desc.bNumConfigurations << endl;

    cout << "| \* Device class: " << (int)desc.bDeviceClass << endl;

    r = libusb\_get\_string\_descriptor\_ascii(dh, desc.iManufacturer, data, 1000);

    cout << "| | \* Vendor ID: " << (int)desc.idVendor << " " << data << endl;

    r = libusb\_get\_string\_descriptor\_ascii(dh, desc.iProduct, data, 1000);

    cout << "| | | \* Product ID: " << (int)desc.idProduct << " " << data << endl;

    if(desc.iSerialNumber){

        r = libusb\_get\_string\_descriptor\_ascii(dh, desc.iSerialNumber, data, 1000);

        cout << "| | | | \* Serial Number of device: " << data << endl;

    }

    else{

    cout << "| | | | \* Failed to get Serial Number of device" << endl;

    }

    libusb\_close(dh);

    cout << endl

         << endl

         << endl;

}