###### 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

ПРАКТИЧЕСКАЯ РАБОТА №3

студента 2 курса, группы 22204

Соломенникова Николая Александровича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

доцент

А.Ю.Власенко

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 2](#__RefHeading___1)

[ЗАДАНИЕ 2](#__RefHeading___2)

[ОПИСАНИЕ РАБОТЫ 3](#__RefHeading___3)

[ЗАКЛЮЧЕНИЕ 6](#__RefHeading___4)

[Листинг кода программы 1 6](#__RefHeading___5)

[Листинг кода программы 2 9](#__RefHeading___6)

# ЦЕЛЬ

Ознакомиться с библиотеками OpenCV и libusb и научиться работать с некоторыми их функциями.

# 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.

Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.

4. 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.

5. Требования к содержанию отчета:

- «чистое» неоткорректированное изображение, полученное с камеры;

- это же изображение в преобразованном виде;

- полный код программы №1, выполняющей преобразование

изображения;

- оценку скорости обработки видео (кадров в секунду) и долю времени, затрачиваемого процессором на ввод, обработку и показ видеоданных;

6. Полный код программы №2, выводящей информацию по USB-устройствам.

7. Описание обнаруженных USB-устройств,

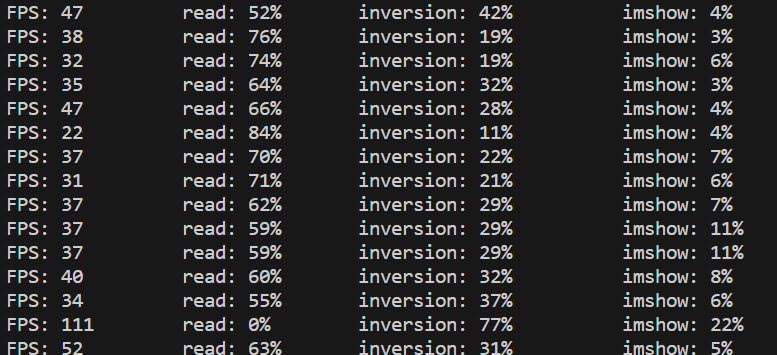
# ОПИСАНИЕ РАБОТЫ

В качестве преобразования была выбрана инверсия цветов (негатив).

И ещё дополнительно был сделан вывод FPS и цвета пикселя, на который указывает мышка, на экран.



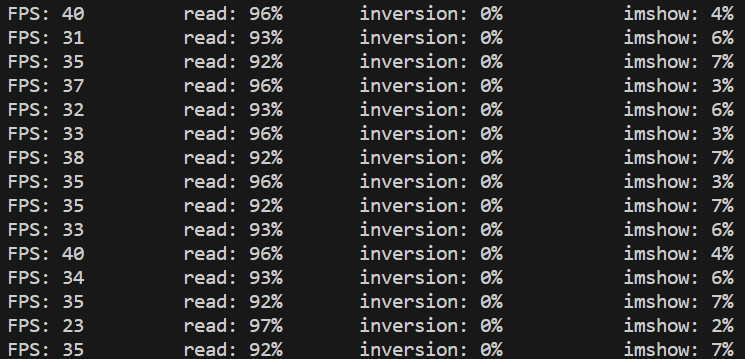
Показатели FPS и долей времени с включённым преобразованием:



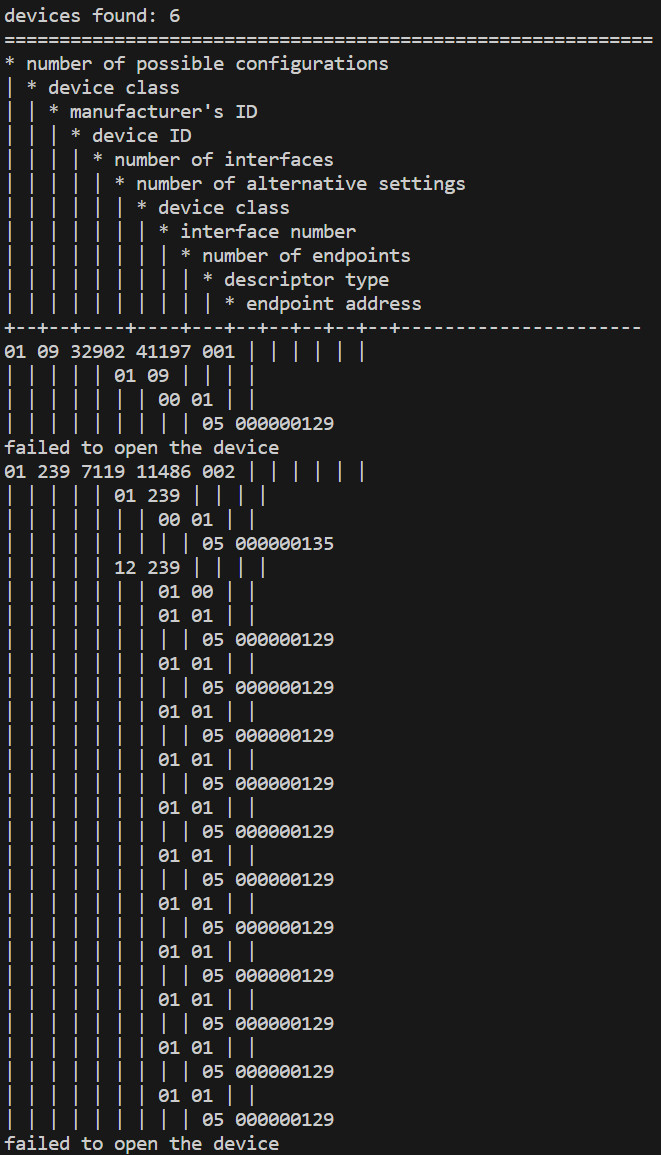
Чистое изображение без преобразований:

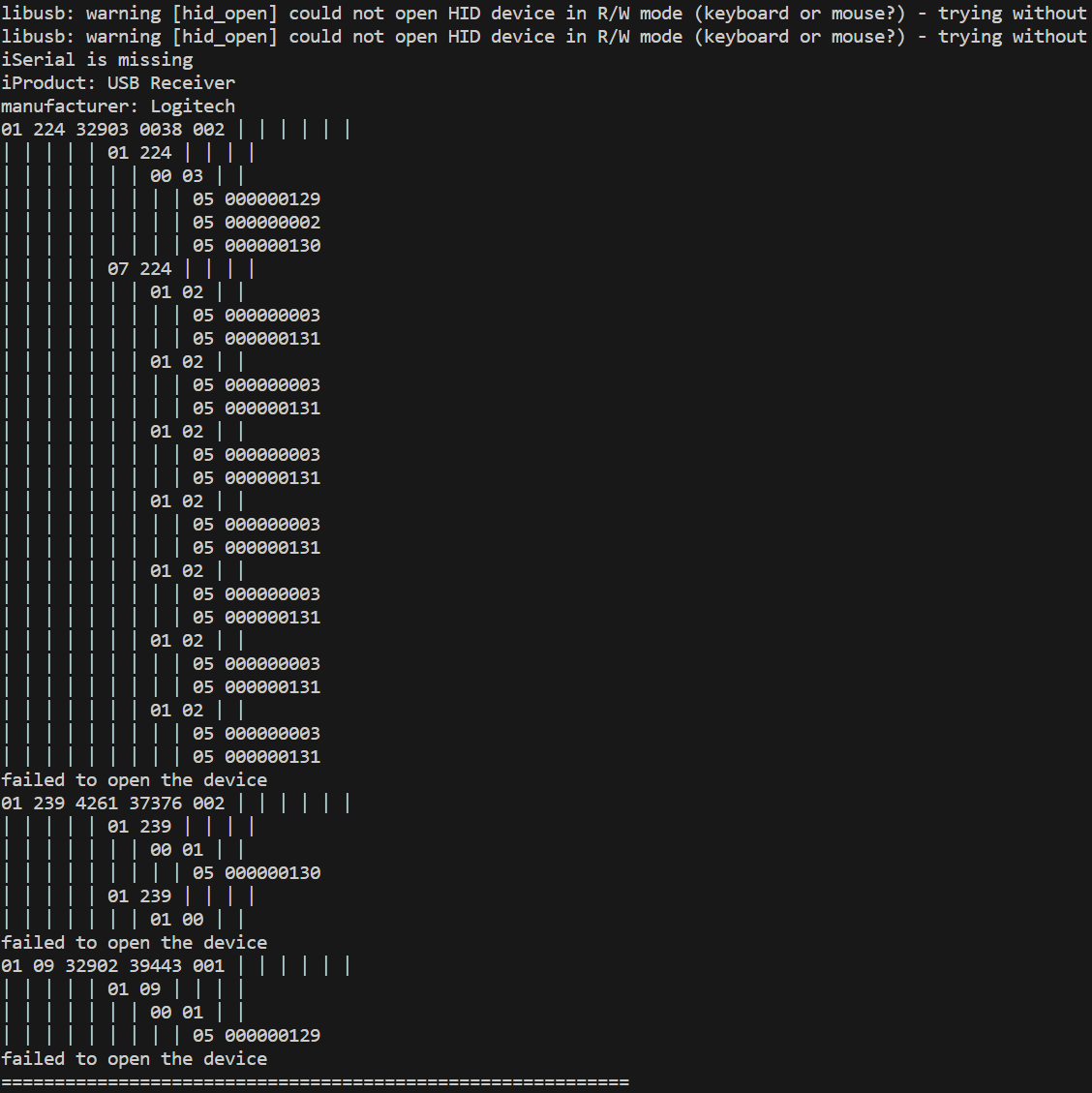


Показатели FPS и долей времени с выключённым преобразованием:



Программа для работы с libusb показала следующий набор устройств:





# ЗАКЛЮЧЕНИЕ

Библиотека OpenCV даёт очень много возможностей для работы с изображениями, видео, окнами, различными обработками и др. В целом это очень мощная библиотека с разными модулями. Библиотека libusb позволяет работать с интерфейсом usb устройств, получать разную информацию о устройствах.

# Листинг кода программы 1

#include <iostream>

#include <opencv2/core.hpp>    // Basic OpenCV structures (cv::Mat, Scalar)

#include <opencv2/highgui.hpp> // OpenCV window I/O

#include <opencv2/opencv.hpp>  // for putText

#include <time.h>

using namespace std;

using namespace cv;

// global variables

int mouseX = 0, mouseY = 0;

void callBackFunc(int event, int x, int y, int flags, void \*userdata)

{

    if (event == EVENT\_MOUSEMOVE)

    {

        mouseX = x;

        mouseY = y;

    }

}

void inversion(Mat img)

{

    int height = img.rows;

    int width = img.cols;

    for (int y = 0; y < height; y++)

    {

        uchar \*p = img.ptr(y);

        for (int x = 0; x < width \* 3; x++)

        {

            // R:

            \*p = 255 - \*p;

            p++;

            // G:

            \*p = 255 - \*p;

            p++;

            // B:

            \*p = 255 - \*p;

            p++;

        }

    }

}

void showInfo(Mat img, int curFPS)

{

    string s = to\_string(curFPS);

    putText(img, "FPS: " + s, Point2d(5, 20),

FONT\_HERSHEY\_SIMPLEX, 0.5, Scalar(0, 0, 0, 255));

    uchar \*p = img.ptr(mouseY) + (mouseX) \* 3;

    s = to\_string((int)\*p);

    p++;

    s += " ";

    s += to\_string((int)\*p);

    p++;

    s += " ";

    s += to\_string((int)\*p);

    putText(img, "Pixel color: " + s, Point2d(5, 40),

FONT\_HERSHEY\_SIMPLEX, 0.5, Scalar(0, 0, 0, 255));

}

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

{

    VideoCapture cap(0); // argument can be either the device index or

the name of a video file

    Mat img;

    namedWindow("Image", 1);                 // create a window

    setMouseCallback("Image", callBackFunc); // binding the mouse to the window

    int curFPS = 0;

    while (true)

    {

        clock\_t timeTable[4]; // table to detect time intervals

        timeTable[0] = clock();

        cap.read(img);

        timeTable[1] = clock();

        // inversion(img);        // this is my frame conversion

        // showInfo(img, curFPS); // show text information in Window

        timeTable[2] = clock();

        imshow("Image", img);

        timeTable[3] = clock();

        char c = waitKey(1); // 27 = esc

        if (c == 27)

            break;

        double dt = timeTable[3] - timeTable[0];

        curFPS = int(CLOCKS\_PER\_SEC / dt);

        cout << "FPS: " << curFPS << "    \t";

        cout << "read: " << int((timeTable[1] - timeTable[0]) / dt \* 100) << "%\t";

        cout << "inversion: " << int((timeTable[2] - timeTable[1]) / dt \* 100)

<< "%    \t";

        cout << "imshow: " << int((timeTable[3] - timeTable[2]) / dt \* 100) << "%\n";

    }

    return 0;

}

Файл для сборки первой программы (CMakeLists.txt):

cmake\_minimum\_required(VERSION 3.10)

project(DisplayImage)

set(OpenCV\_DIR C:/Software/opencv/build)

find\_package(OpenCV REQUIRED)

add\_executable(${PROJECT\_NAME} capt\_camera.cpp)

target\_link\_libraries(${PROJECT\_NAME} ${OpenCV\_LIBS} )

# Листинг кода программы 2

#include "libusb/libusb.h"

#include <iostream>

#include <stdio.h>

using namespace std;

void printdev(libusb\_device \*dev);

int main()

{

    libusb\_device \*\*devs;

    libusb\_context \*ctx = NULL;

    int r;

    ssize\_t cnt;

    r = libusb\_init(&ctx);

    if (r < 0)

    {

        fprintf(stderr, "Error: initialization failed, code: %d.\n", r);

        return 1;

    }

    libusb\_set\_debug(ctx, 3);

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

    if (cnt < 0)

    {

        fprintf(stderr, "Error: the list of USB devices was not received, code: %d.\n", r);

        return 1;

    }

    printf("devices found: %d\n", (int)cnt);

    printf("===========================================================\n");

    printf("\* number of possible configurations\n");

    printf("| \* device class\n");

    printf("| | \* manufacturer's ID\n");

    printf("| | | \* device ID\n");

    printf("| | | | \* number of interfaces\n");

    printf("| | | | | \* number of alternative settings\n");

    printf("| | | | | | \* device class\n");

    printf("| | | | | | | \* interface number\n");

    printf("| | | | | | | | \* number of endpoints\n");

    printf("| | | | | | | | | \* descriptor type\n");

    printf("| | | | | | | | | | \* endpoint address\n");

    printf("+--+--+----+----+---+--+--+--+--+--+----------------------\n");

    for (ssize\_t i = 0; i < cnt; i++)

    {

        printdev(devs[i]);

    }

    printf("===========================================================\n");

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

    libusb\_exit(ctx);

    return 0;

}

void printdev(libusb\_device \*dev)

{

    libusb\_device\_descriptor desc;

    libusb\_config\_descriptor \*config;

    const libusb\_interface \*inter;

    const libusb\_interface\_descriptor \*interdesc;

    const libusb\_endpoint\_descriptor \*epdesc;

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

    if (r < 0)

    {

        fprintf(stderr, "Ошибка: дескриптор устройства не получен, код: %d.\n", r);

        return;

    }

    libusb\_get\_config\_descriptor(dev, 0, &config);

    printf("%.2d %.2d %.4d %.4d %.3d | | | | | |\n",

           (int)desc.bNumConfigurations,

           (int)desc.bDeviceClass,

           desc.idVendor,

           desc.idProduct,

           (int)config->bNumInterfaces);

    for (int i = 0; i < (int)config->bNumInterfaces; i++)

    {

        inter = &config->interface[i];

        printf("| | | | | %.2d %.2d | | | |\n",

               inter->num\_altsetting,

               (int)desc.bDeviceClass);

        for (int j = 0; j < inter->num\_altsetting; j++)

        {

            interdesc = &inter->altsetting[j];

            printf("| | | | | | | %.2d %.2d | |\n",

                   (int)interdesc->bInterfaceNumber,

                   (int)interdesc->bNumEndpoints);

            for (int k = 0; k < (int)interdesc->bNumEndpoints; k++)

            {

                epdesc = &interdesc->endpoint[k];

                printf("| | | | | | | | | %.2d %.9d\n",

                       (int)epdesc->bDescriptorType,

                       (int)(int)epdesc->bEndpointAddress);

            }

        }

    }

    libusb\_device\_handle \*handle = nullptr;

    unsigned char description[256];

    libusb\_open(dev, &handle);

    if (!handle)

    {

        printf("failed to open the device\n");

        libusb\_close(handle);

        return;

    }

    int result = libusb\_get\_string\_descriptor\_ascii(handle, desc.iSerialNumber, description, 256);

    if (result < 0)

        printf("iSerial is missing\n");

    else

        printf("iSerial: %s\n", description);

    result = libusb\_get\_string\_descriptor\_ascii(handle, desc.iProduct, description, 256);

    if (result < 0)

        printf("missing iProduct\n");

    else

        printf("iProduct: %s\n", description);

    result = libusb\_get\_string\_descriptor\_ascii(handle, desc.iManufacturer, description, 256);

    if (result < 0)

        printf("manufacturer is missing\n");

    else

        printf("manufacturer: %s\n", description);

    libusb\_free\_config\_descriptor(config);

    libusb\_close(handle);

}

Файл для сборки второй программы (CMakeLists.txt):

cmake\_minimum\_required(VERSION 3.10)

project(libusbTest)

add\_executable(${PROJECT\_NAME} main.cpp)