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

ВЫСОКО- И НИЗКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИИ

Студентки 2 курса, группы 21205

Евдокимовой Дари Евгеньевны

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук, доцент
А.Ю. Власенко

Новосибирск 2022

СОДЕРЖАНИЕ

ЦЕЛЬ

1. Ознакомление с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV.
2. Изучение работы высокоуровневых периферийных устройств.
3. Ознакомление с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.
4. Изучение работы низкоуровневых периферийных устройств.

ЗАДАНИЕ

1. Реализовать программу, которая произвольно преобразует изображения с использованием библиотеки OpenCV, которая получает поток видеоданных с камеры и выводит его на экран.
2. Измерить количество кадров, обрабатываемое программой в секунду. Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.
3. Реализовать программу, 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

1. Была скачана библиотека OpenCV и установлена в Visual Studio 2022.
2. По документации[1] были изучены основные методы библиотеки, которые позволяют работать с камерой, преобразованием полученного изображения и его выводом.
3. Был создан файл *webcamera.cpp*, в котором реализована программа по преобразованию изображения, полученного с веб-камеры. Листинг программы представлен в Приложении 1.
4. Изображение до обработки представлено в Приложении 2.
5. Была произведена оценка скорости обработки видео, оценена доля времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры. Результаты представлены в Таблице 1.

Таблица 1. Результаты вычисления программы

Критерий	Полученный результат
Среднее значение FPS	7.44444 кадров/сек
Время чтения	77.7964%
Время преобразования	6.75731%
Время показа (вывода изображения)	0.49036%

6. Был создан файл *usb.cpp*, в который была написана программа, получающая список всех подключенных к машине USB устройств с использованием библиотеки libusb. Для каждого найденного устройства выведен его класс, идентификатор производителя, идентификатор изделия и серийный номер. Результаты представлены на рис. 1.
7. Команды для компиляции и запуска программы:
   g++ usb.cpp -o usb.o `pkg-config --libs --cflags libusb-1.0`

   /.usb.o

```
================================
|* класс устройства
|  | * идентификатор производителя
|  |    | * идентификатор устройства
|  |    |    | * серийный номер
+--+----+----+------------------+
Устройство 1\6
 09 1d6b 0003 null
Устройство 2\6
 e0 8087 0a2a null
Устройство 3\6
 ef 0bda 57ed null
Устройство 4\6
 00 1c4f 0034 null
Устройство 5\6
 00 09da 2268 null
Устройство 6\6
 09 1d6b 0002 null
================================
```

Рис.1. Выведенные устройства

9. Был произведен поиск информации по каждому из устройств с помощью Интернет-ресурсов и команды *lsusb* (см. Рис. 2.) в терминале. Данная команда позволяет узнать идентификатор производителя и устройства, а также распознать что за устройство подключено. Только с помощью lsusb была идентифицирована веб-камера и клавиатура, так как в Интернете не было информации об устройствах с такой конфигурацией. Устройства и их описания представлены в Приложении 4.

```
[redacted]$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 8087:0a2a Intel Corp.
Bus 001 Device 003: ID 0bda:57ed Realtek Semiconductor Corp. USB2.0 VGA UVC WebCam
Bus 001 Device 006: ID 1c4f:0034 SiGma Micro Usb Mouse
Bus 001 Device 005: ID 09da:2268 A4Tech Co., Ltd. USB Keyboard
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Рис. 2. Информация об устройствах из терминала

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены основные принципы работы с библиотекой OpenCV, веб-камерой и способами преобразования изображения. Так же были получены данные об измерении скорости обработки видео. По полученным измерениям можно сделать вывод о том, что веб-камера не способна создавать большое количество изображений в промежуток времени, поэтому FPS низкий.

Также были изучены основы работы с библиотекой libusb и получен список всех подключенных к компьютеру USB устройств с дополнительными данными.

СПИСОК ЛИТЕРАТУРЫ

1. Документация библиотеки OpenCV [Электронный ресурс]. URL: https://docs.opencv.org/4.x/
2. Документация библиотеки libusb [Электронный ресурс]. URL: http://www.usb.org/

Приложение 1. Листинг программы с использованием библиотеки OpenCV

```
#include <time.h> //for time info
#include <iostream> // for standard I/O
#include <opencv2/core.hpp>     // Basic OpenCV structures (cv::Mat, Scalar)
#include <opencv2/highgui.hpp>  // OpenCV window I/O, works with capturing from a cam
#include <opencv2/imgproc.hpp>  // Works with images, e.g. Gaussian Blur.

using namespace std;
using namespace cv;

int main(int argc, char **argv) {
    int frameCounter = 0;
    double totalRead = 0;
    double totalProcess = 0;
    double totalWrite = 0;
    int tick = 0; //counts how many milliseconds last from the last call
    int totalFPS = 0;

    time_t time_begin = time(nullptr);
    clock_t time_begin_clock = clock();
    clock_t tmpReadTime, tmpProcessTime, tmpWriteTime;

    VideoCapture cap(0); //open the default camera using default API
    if (!cap.isOpened()) {
        cout << "Can not open webcam" << endl;
        return 0;
    }

    Mat input; //itit video capture
    while (true) {
        // reading
        tmpReadTime = clock();

        // wait for a new frame from camera and store it into 'input'
        cap.read(input);
        if (input.empty()) {
            cout << "ERROR! blank frame grabbed" << endl;
            break;
        }
        totalRead += clock() - tmpReadTime;

        // proccessing
        tmpProcessTime = clock();
        imshow("No Filters Image", input); //display an image in window

        char key = waitKey(1);
        if (key == 27) {
            break;
        }
```

```
        Mat outFlippered,outErode, outBorder;

        //flip the capture
        flip(input, outFlippered, 1); // 1 means flip in Oy

        /*  About kernel
            OpenCV blurs an image by applying what's called a Kernel.
            A Kernel tells you how to change the value of any given pixel
            by combining it with different amounts of the neighboring pixels.
            The kernel is applied to every pixel in the image one-by-one to
            produce the final image (this operation known as a convolution).
            erode(outFlippered, outErode, kernel);
        */
        Mat kernel = getStructuringElement(MORPH_RECT, Size(7, 7));

        /*
            @param outFlippered - input iamge
            @param outErode - output iamge
            @param kernel - structuring element used for erosion
        */
        erode(outFlippered, outErode, kernel);

        //make a border
        int top = (int)(0.15 * input.rows); //input.rows =  number of rows in a matrix
        int bottom = top;
        int left = (int)(0.05 * input.cols); //input.cols =  number of columns in a matrix
        int right = left;
        int borderType = BORDER_CONSTANT;
        Scalar color(186, 85, 211); // RGB for purple color

        /*
            @param outErode - source image
            @param outBorder - destination image
            @params top, bottom, left, right - length in pixels of the borders at each side of the image
            @param BORDER_CONSTANT - type of the border
            @param color - color of the border
        */
        copyMakeBorder(outErode, outBorder, top, bottom, left, right, BORDER_CONSTANT,
color);

        double alpha = 1.1; // argument for contrast control, [1.0-3.0]
        int beta = 5;       // for brightness control, [0-100]

        /*  We perform the operation: g(y, x) = alfa * f(y, x) + beta
            alfa is gain (усиления) parameter to control contrast
            beta is bias (смещения) parameter to control brightness
            f(y, x) - src image, g(y, x) - dst image.

            To access each pixel in the images we are using this syntax:
            outBorder.at<Vec3b>(y,x)[c] where y is the row, x is the column and c is B, G or R (0, 1
or 2).
            We use cv::saturate_cast to make sure the values are valid
```

```
            chanels() - number of matrix chanels
        */

        for (int y = 10; y < input.rows / 1.5; y++) {
            for (int x = input.cols / 1.5 - 30; x > 0 ; x--) {
                for (int c = 0; c < input.channels(); c++) {
                    outBorder.at<Vec3b>(y, x)[c] =
                        saturate_cast<uchar>(alpha * input.at<Vec3b>(y, x)[c] + beta);
                }
            }
        }

        Mat res = outBorder;

        //include a text on the image
        String text = "Sharp Image";
        int fontFace = FONT_HERSHEY_PLAIN;
        int fontScale = 3;
        Scalar colorText = (128, 0, 128);
        int thickness = 3;

        putText(res, text, Point(10, 50), fontFace, fontScale, colorText, thickness);

        totalProcess += clock() - tmpProcessTime;

        // writing
        tmpWriteTime = clock();
        imshow("With Filters", res);
        totalWrite += clock() - tmpWriteTime;

        // FPS counter
        frameCounter++;
        auto time_now = time(nullptr) - time_begin;
        if (time_now - tick >= 1) {
            tick++;
            totalFPS += frameCounter;
            cout << "FPS: " << frameCounter << endl;
            frameCounter = 0;
        }
    }

    clock_t total_time_clock = clock() - time_begin_clock;
    cout << "Average FPS: " << totalFPS * 1.0 / tick << endl;
    cout << "Time for reading: " << totalRead * 100 / total_time_clock << "%" << endl;
    cout << "Time for processing: " << totalProcess * 100 / total_time_clock << "%" << endl;
    cout << "Time for writing: " << totalWrite * 100 / total_time_clock << "%" << endl;
    return 0;
}
```

Приложение 2. Изображения, полученные с веб-камеры

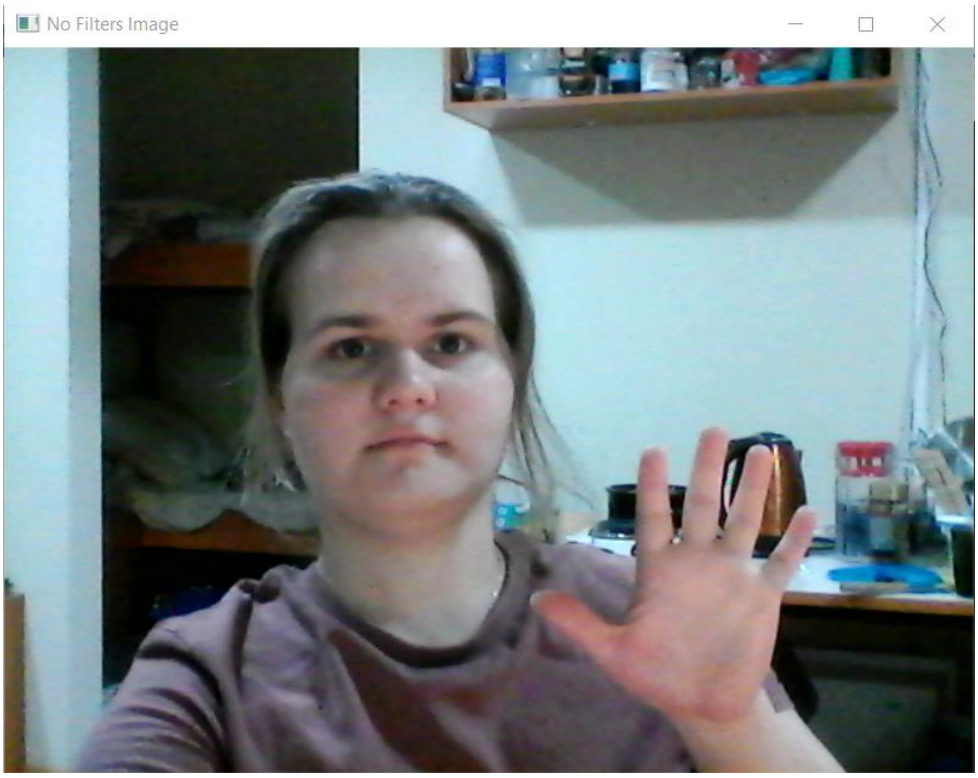

Рис 1. Изображение без обработки фотографии

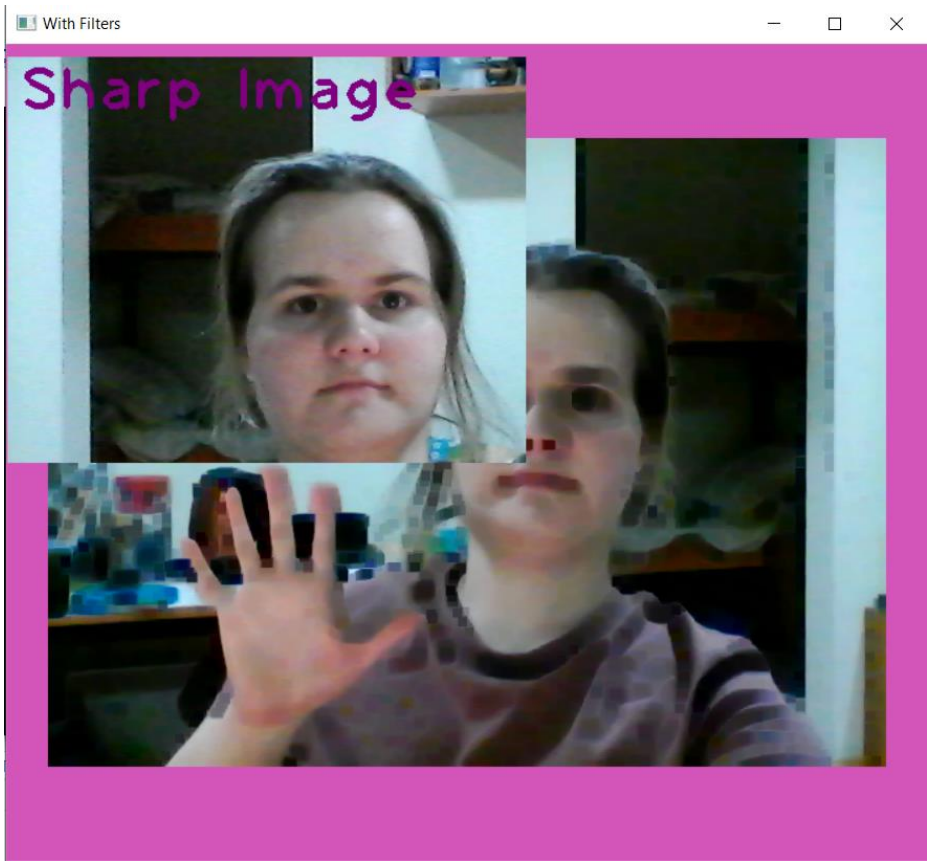

Рис 2. Изображение с обработкой фотографии

Приложение 3. Листинг программы для определения подключенных USB устройств

```
#include <cstdio>
#include <iostream>
#include <libusb-1.0/libusb.h>

void PrintHead(){
   printf("=================================\n");
   printf("|* класс устройства\n");
   printf("|  | * идентификатор производителя\n");
   printf("|  |    | * идентификатор устройства\n");
   printf("|  |    |    | * серийный номер\n");
   printf("+--+----+----+-------------------+\n");
}

void printDevices(libusb_device *dev){
   libusb_device_descriptor desc{}; // дескриптор устройства
   libusb_device_handle *handle = nullptr; // хэндл устройства
   unsigned char str[256]; // строка для хранения серийного номера

   int r = libusb_get_device_descriptor(dev, &desc); // получить дескриптор
   if (r < 0){
      fprintf(stderr,"Ошибка: дескриптор устройства не получен, код: %d.\n", r);
      return;
   }

   printf("%.2x %.4x %.4x ",
      desc.bDeviceClass,
      desc.idVendor,
      desc.idProduct);

   libusb_open(dev, &handle);
   if (handle && desc.iSerialNumber){
      r = libusb_get_string_descriptor_ascii(handle, desc.iSerialNumber, str, sizeof(str));
      printf("%s", str);
   }
   else
      printf("null");
   std::cout << "\n";
}

int main(){
   libusb_device **devs; // указатель на указатель на устройство, используется для
получения списка устройств
   libusb_context *ctx = nullptr; // контекст сессии libusb
   int r; // для возвращаемых значений
   size_t cnt; // число найденных USB-устройств
```

```
    r = libusb_init(&ctx); // инициализировать библиотеку libusb, открыть сессию работы с
libusb
    if (r < 0){
        fprintf(stderr,"Ошибка: инициализация не выполнена, код: %d.\n", r);
        return 1;
    }

    cnt = libusb_get_device_list(ctx, &devs);
    if (cnt < 0){
        fprintf(stderr,
                "Ошибка: список USB устройств не получен. Код: %d\n", r);
        return 1;
    }

    PrintHead();

    for (size_t i = 0; i < cnt; i++){ // цикл перебора всех устройств
        printf("Устройство %ld\\%ld\n", i + 1, cnt);
        printDevices(devs[i]); // печать параметров устройства
    }
    printf("====================================\n");

    // освободить память, выделенную функцией получения списка устройств
    libusb_free_device_list(devs, 1);

    // завершить работу с библиотекой libusb, закрыть сессию работы с libusb
    libusb_exit(ctx);
    return 0;
}
```

Приложение 4. Описание подключенных USB-устройств

Таблица 1. Информация о подключенных устройствах

Номер устройства	Данные об устройстве	Описание устройства	Комментарии	Источник информации
Устройство 1\6	09 1d6b 0003 null	Linux Foundation 3.0 root hub	Рут-хаб 3.0	https://linux-hardware.org/?id=usb:1d6b-0003
Устройство 2\6	e0 8087 0a2a null	Драйвер USB\VID_8087&PID_0A2A Intel Corp.	Драйвер	https://linux-hardware.org/?id=usb:8087-0a2a
Устройство 3\6	ef 0bda 57ed null	Realtek Semiconductor Corp. USB2.0 VGA UVC WebCam	Веб-камера	Команда lsusb и терминал
Устройство 4\6	00 1c4f 0034 null	SiGma Micro Usb Mouse	Мышь	https://linux-hardware.org/?id=usb:1c4f-0034
Устройство 5\6	00 09da 2268 null	A4Tech Co., Ltd. USB Keyboard	Клавиатура	Команда lsusb и терминал
Устройство 6\6	09 1d6b 0002 null	Linux Foundation 2.0 root hub	Рут-хаб 2.0	https://linux-hardware.org/?id=usb:1d6b-0002