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

«Библиотеки OpenCV и libusb»

студента 2 курса, группы 21204

Осипова Александра Александровича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
А. Ю. Власенко

Новосибирск 2022

СОДЕРЖАНИЕ

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.

3. Измерить количество кадров, обрабатываемое программой в секунду. Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.

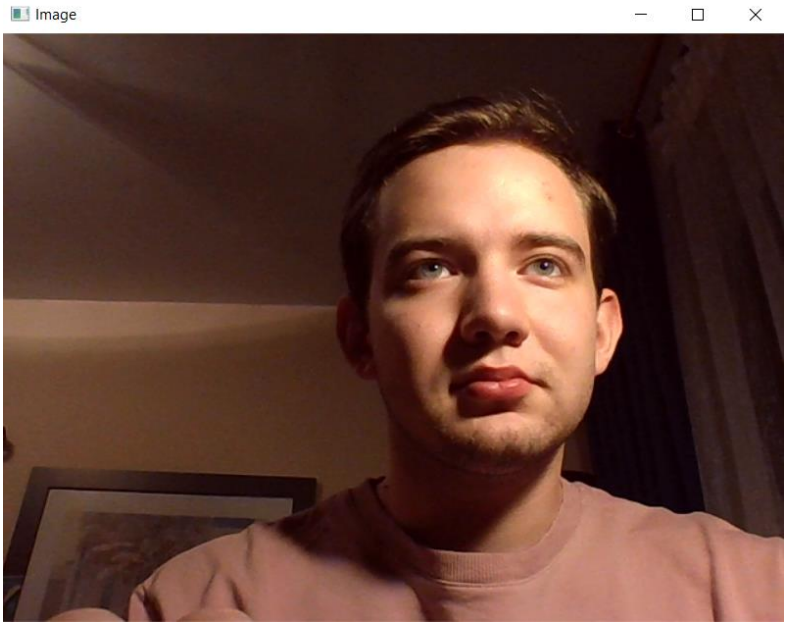
4. Реализовать программу, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатать его класс, идентификатор производителя, идентификатор изделия и серийный номер.

5. Составить отчет по лабораторной работе.

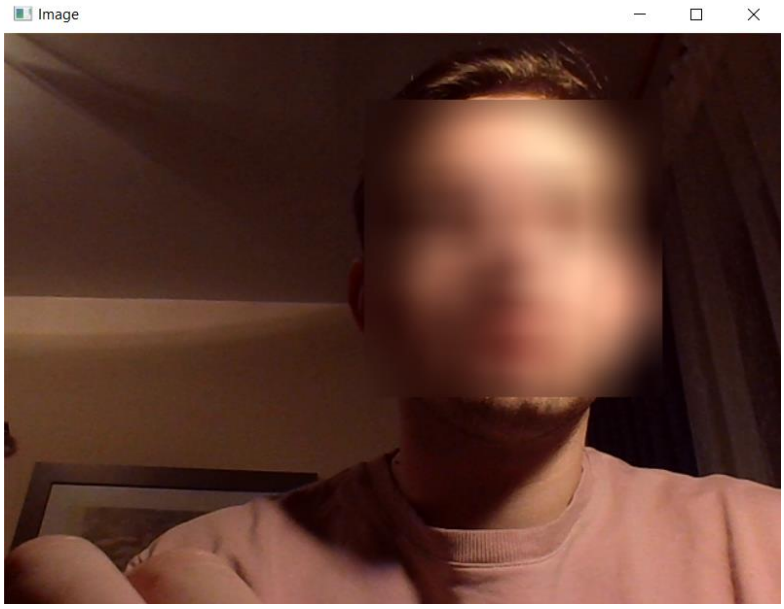
ОПИСАНИЕ РАБОТЫ

Часть 1. OpenCV

1. Была локально установлена библиотека OpenCV на ноутбук.
2. С помощью функций библиотеки OpenCV была написана программа (см. Приложение 1), реализующая захват лица с веб-камеры и его размытие.



Изображение без преобразований



Изображение с преобразованиями

3. Было измерено количество кадров, обрабатываемое программой в секунду (как с преобразованиями, так и без них). Так же была оценена доля времени, затрачиваемая процессором на обработку видеоданных, получаемых с камеры (см. Приложение 2).

Без преобразований:

- ~30 FPS

С преобразованиями:

- 2-6 FPS
- ~0.1% времени затрачено на ввод
- ~98% времени затрачено на обработку
- ~1% времени затрачено на вывод
- остальное время затрачено реализацией измерения времени

Часть 2. libusb

1. Была создана папка lab3 на сервере кафедры, куда был добавлен исходный файл usb.cpp
2. За основу была взята программы, листинг которой прикреплен к описанию лабораторной работы.
3. Для печати серийного номера, текстовой строки изделия и производителя исходная программа была дополнена следующими строчками кода:

```
libusb_device_handle *handle = nullptr;
if (libusb_open(dev, &handle) == LIBUSB_SUCCESS){
    unsigned char str[256];
    r = libusb_get_string_descriptor_ascii(handle, desc.iSerialNumber, str, sizeof(str));
    if (r > 0){
        cout << "Serial number: " << str << endl;
    }
    else {
        cout << "Serial number: empty" << endl;
    }
    r = libusb_get_string_descriptor_ascii(handle, desc.iProduct, str, sizeof(str));
    if (r > 0){
        cout << "Product: " << str << endl;
    }
    else {
        cout << "Product: empty" << endl;
    }
    r = libusb_get_string_descriptor_ascii(handle, desc.iManufacturer, str, sizeof(str));
    if (r > 0){
        cout << "Manufacturer: " << str << endl;
    }
    else {
        cout << "Manufacturer: empty" << endl;
    }
}
else {
    cout << "FAILURE" << endl;
}
cout << endl;
libusb_close(handle);
```

4. Программа была скомпилирована с помощью команды:
   g++ -o usb.out -I/usr/include/libusb-1.0 usb.cpp -lusb-1.0
5. Были получены количество обнаруженных устройств, их серийный номер, текстовая строка изделия и производителя (см. Приложение 4)

ЗАКЛЮЧЕНИЕ

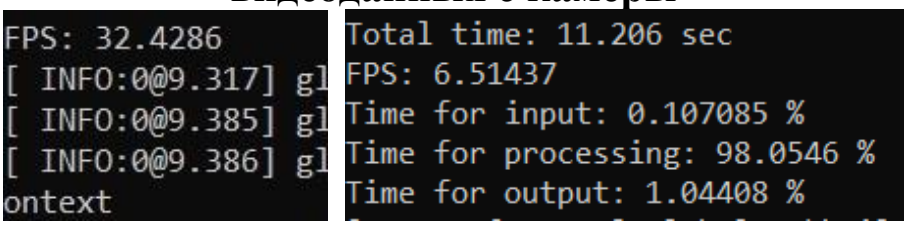
В рамках данной лабораторной работы были изучены такие библиотеки для программирования периферийных устройств, как OpenCV и libusb. Работа с OpenCV показала, как сильно может снизиться количество кадров, обрабатываемых в секунду, если добавить всего два преобразования: захват лица с веб-камеры и его размытие. Так же было зафиксировано, что почти 100% времени от обработки видеоданных занимает преобразование изображения. Программа usb.out обнаружила и напечатала информацию о 12 usb-устройствах. Большинство из них — это хост-контроллеры. Также нашлись клавиатура, мышь и картридер.

Приложение 1. Листинг программы для захвата и размытия лица

```
#include <iostream> // for standard I/O
#include <string> // for strings
#include <iomanip> // for controlling float print precision
#include <sstream> // string to number conversion
#include <opencv2/core.hpp> // Basic OpenCV structures (cv::Mat, Scalar)
#include <opencv2/imgproc.hpp> // Gaussian Blur
#include <opencv2/videoio.hpp>
#include <opencv2/highgui.hpp> // OpenCV window I/O
#include <opencv2/imgcodecs.hpp>
#include <opencv2/objdetect.hpp>
#include <time.h>
#include <chrono>
using namespace std;
using namespace cv;

using std::chrono::duration_cast;
using std::chrono::milliseconds;
using std::chrono::system_clock;

void getClearImg(Mat img, VideoCapture &cap, int numFrames) {
	time_t start, end;

	time(&start);
	while (true) {
		cap.read(img);
		imshow("Image", img);
		char c = waitKey(1);
		if (c == 'z') break;
		++numFrames;
	}
	time(&end);
	double seconds = difftime(end, start);
	double fps = numFrames / seconds;
	cout << "FPS: " << fps << endl;
}

void getConvertedImg(Mat img, VideoCapture &cap, int numFrames) {
	CascadeClassifier faceCascade;
	faceCascade.load("Resources/haarcascade_frontalface_default.xml");
	assert(!faceCascade.empty());

	double timeInput = 0;
	double timeProc = 0;
	double timeOutput = 0;

	auto start =
duration_cast<milliseconds>(system_clock::now().time_since_epoch()).count();

	while (true) {
		auto startInput =
duration_cast<milliseconds>(system_clock::now().time_since_epoch()).count();
		cap.read(img);
		auto endInput =
duration_cast<milliseconds>(system_clock::now().time_since_epoch()).count();

		auto startProc =
duration_cast<milliseconds>(system_clock::now().time_since_epoch()).count();
		vector<Rect> faces;
		faceCascade.detectMultiScale(img, faces, 1.1, 2);
		for (int i = 0; i < faces.size(); ++i) {
			GaussianBlur(img(faces[i]), img(faces[i]), Size(55, 55), 100);
```

```
            }
            auto endProc =
duration_cast<milliseconds>(system_clock::now().time_since_epoch()).count();

            auto startOutput =
duration_cast<milliseconds>(system_clock::now().time_since_epoch()).count();
            imshow("Image", img);
            auto endOutput =
duration_cast<milliseconds>(system_clock::now().time_since_epoch()).count();

            char c = waitKey(1);
            if (c == 'z') break;

            ++numFrames;
            timeInput += (endInput - startInput);
            timeProc += (endProc - startProc);
            timeOutput += (endOutput - startOutput);
      }
      auto end =
duration_cast<milliseconds>(system_clock::now().time_since_epoch()).count();
      double seconds = (end - start)/1000.0;
      double fps = numFrames / seconds;
      cout << "Total time: " << seconds << " sec" << endl;
      cout << "FPS: " << fps << endl;
      cout << "Time for input: " << ((timeInput/1000.0)/seconds)*100 << " %" <<
endl;
      cout << "Time for processing: " << ((timeProc / 1000.0) / seconds) * 100 << "
%" <<endl;
      cout << "Time for output: " << ((timeOutput / 1000.0) / seconds) * 100 << "
%" << endl;
}


int main(int argc, char* argv[]) {
      VideoCapture cap(0);
      Mat img;

      int numFrames = 0;
      getClearImg(img, cap, numFrames);
      getConvertedImg(img, cap, numFrames);

      return 0;
}
```

Приложение 2. Результаты измерений времени обработки видеоданных с камеры



Приложение 3. Листинг программы для печати информации о usb-устройствах

```
#include <iostream>
#include <libusb.h>
#include <stdio.h>
using namespace std;
```

```c
void printdev(libusb_device* dev);
int main() {
	libusb_device** devs; // указатель на указатель на устройство,
	// используется для получения списка устройств
	libusb_context* ctx = NULL; // контекст сессии libusb
	int r; // для возвращаемых значений
	ssize_t cnt; // число найденных USB-устройств
	ssize_t i; // индексная переменная цикла перебора всех устройств
	// инициализировать библиотеку libusb, открыть сессию работы с libusb
	r = libusb_init(&ctx);
	if (r < 0) {
		fprintf(stderr,
			"Ошибка: инициализация не выполнена, код: %d.\n", r);
		return 1;
	}
	// задать уровень подробности отладочных сообщений
	libusb_set_debug(ctx, 3);
	// получить список всех найденных USB- устройств
	cnt = libusb_get_device_list(ctx, &devs);
	if (cnt < 0) {
		fprintf(stderr,
			"Ошибка: список USB устройств не получен.\n", r);
		return 1;
	}
	printf("найдено устройств: %d\n", cnt);
	printf("=============================="
		"============================\n");
	printf("* количество возможных конфигураций\n");
	printf("| * класс устройства\n");
	printf("| | * идентификатор производителя\n");
	printf("| | | * идентификатор устройства\n");
	printf("| | | | * количество интерфейсов\n");
	printf("| | | | | * количество "
		"альтернативных настроек\n");
	printf("| | | | | | * класс устройства\n");
	printf("| | | | | | | * номер интерфейса\n");
	printf("| | | | | | | | * количество "
		"конечных точек\n");
	printf("| | | | | | | | | * тип дескриптора\n");
	printf("| | | | | | | | | | * адрес "
		"конечной точки\n");
	printf("+--+--+----+----+---+--+--+--+"
		"--+--+--------------------\n");
	for (i = 0; i < cnt; i++) { // цикл перебора всех устройств
		printdev(devs[i]); // печать параметров устройства
	}
	printf("=============================="
		"============================\n");
	// 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
	libusb_config_descriptor* config; // дескриптор конфигурации объекта
	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;
```

```
	}
	// получить конфигурацию устройства
	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_free_config_descriptor(config);

	libusb_device_handle* handle = nullptr;
	if (libusb_open(dev, &handle) == LIBUSB_SUCCESS) {
		unsigned char str[256];
		r = libusb_get_string_descriptor_ascii(handle, desc.iSerialNumber, str,
sizeof(str));
		if (r > 0) {
			cout << "Serial number: " << str << endl;
		}
		else {
			cout << "Serial number: empty" << endl;
		}
		r = libusb_get_string_descriptor_ascii(handle, desc.iProduct, str,
sizeof(str));
		if (r > 0) {
			cout << "Product: " << str << endl;
		}
		else {
			cout << "Product: empty" << endl;
		}
		r = libusb_get_string_descriptor_ascii(handle, desc.iManufacturer, str,
sizeof(str));
		if (r > 0) {
			cout << "Manufacturer: " << str << endl;
		}
		else {
			cout << "Manufacturer: empty" << endl;
		}
	}
```

```
	else {
		cout << "FAILURE" << endl;
	}
	cout << endl;
	libusb_close(handle);

}
```

Приложение 4. Результат работы программы usb.out

```
найдено устройств: 12
=============================================================
* количество возможных конфигураций
| * класс устройства
| | * идентификатор производителя
| | | * идентификатор устройства
| | | | * количество интерфейсов
| | | | | * количество альтернативных настроек
| | | | | | * класс устройства
| | | | | | | * номер интерфейса
| | | | | | | | * количество конечных точек
| | | | | | | | | * тип дескриптора
| | | | | | | | | | * адрес конечной точки
+--+--+----+----+---+--+--+--+--+--+---------------------
```

```
01 00 7247 0038 002 | | | | |
| | | | | 01 00 | | |
| | | | | | | 00 01 | |
| | | | | | | | | 05 000000129
| | | | | 01 00 | | |
| | | | | | | 01 01 | |
| | | | | | | | | 05 000000130
Serial number: empty
Product: USB Keyboard
Manufacturer: SIGMACHIP

01 00 1112 0058 001 | | | | |
| | | | | 01 00 | | |
| | | | | | | 00 01 | |
| | | | | | | | | 05 000000129
Serial number: empty
Product: USB Optical Mouse
Manufacturer: Genius

01 09 7531 0001 001 | | | | |
| | | | | 01 09 | | |
| | | | | | | 00 01 | |
| | | | | | | | | 05 000000129
Serial number: 0000:00:1d.1
Product: UHCI Host Controller
Manufacturer: Linux 5.15.0-52-generic uhci_hcd
```