###### 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 ЛАБОРАТОРНОЙ РАБОТЫ**

«ВЫСОКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ»

студентки 2 курса, группы 19201

Хаецкой Дарьи Владимировны

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Власенко Андрей Юрьевич

Новосибирск 2020

**Цель**

1. Ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV;

2. Применить полученные знания в реализации программы вывода в окно искаженного изображения с камеры.

**Задание**

Изучить принципы работы OpenCV, ознакомиться с работой с растровыми изображениями и их форматами, работой с видеоданными и их форматами, интерфейсом для работы с камерами, реализацией упрощенного оконного интерфейса, операций над векторами и матрицами. Реализовать с помощью полученных знаний об этой библиотеке программу, которая будет принимать видеосигнал с камеры, распознавать лица на видеопотоке и искажать полученное изображение путем сдвига цветовых каналов. Замерить время, затрачиваемое на получение, обработку и вывод кадров на окно программы. Провести анализ полученных данных.

**Листинг программы**

**#include <opencv4/opencv2/opencv.hpp>**

**#include <vector>**

**#include <fstream>**

**#include <iostream>**

**using namespace cv;**

**using namespace std;**

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

**double scale = 1.0;**

**CascadeClassifier faceCascade;**

**faceCascade.load("/home/rey/opencv-master/data/haarcascades/haarcascade\_frontalface\_alt.xml");**

**VideoCapture cap(0);**

**if(!cap.isOpened())**

**return -1;**

**long double totalTime = 0, inputTime = 0, procTime = 0, outputTime = 0;**

**long int frameCount = 0;**

**for (;;){**

**Mat frame;**

**clock\_t c\_start = clock();**

**cap >> frame;**

**clock\_t c\_end = clock();**

**double timeElapsed0 = 1000.0 \* (c\_end - c\_start) / CLOCKS\_PER\_SEC;**

**totalTime += timeElapsed0;**

**inputTime += timeElapsed0;**

**imshow("original", frame);**

**c\_start = clock();**

**Mat grayscale; // так как алгоритмы машинного обучения работают с чёрно-белыми изображениями**

**// нужно конвертировать изображение в черно-белое**

**cvtColor(frame, grayscale, COLOR\_BGR2GRAY);**

**resize(grayscale, grayscale, Size(grayscale.size().width / scale,**

**grayscale.size().height / scale));**

**vector<Rect> faces;**

**faceCascade.detectMultiScale(grayscale, faces, 1.1,**

**3, 0, Size(30, 30));**

**for (Rect area: faces){**

**Scalar drawColor = Scalar(127, 255, 112);**

**rectangle(frame, Point(cvRound(area.x \* scale), cvRound(area.y \* scale)),**

**Point(cvRound((area.x + area.width - 1) \* scale),**

**cvRound((area.y + area.height - 1) \* scale)), drawColor);**

**}**

**Mat hsv, mask;**

**cvtColor(frame, hsv, COLOR\_BGR2HSV);**

**vector<int> lower**White **{ 0, 0, 0 };**

**vector<int> upperWhite { 10, 30, 20 };**

**inRange(hsv, lowerWhite, upperWhite, mask);**

**Mat result;**

**bitwise\_and(frame, frame, mask);**

**c\_end = clock();**

**double timeElapsed1 = 1000.0 \* (c\_end - c\_start) / CLOCKS\_PER\_SEC;**

**totalTime += timeElapsed1;**

**procTime += timeElapsed1;**

**c\_start = clock();**

**imshow("face detection", frame);**

**imshow("converted", hsv);**

**c\_end = clock();**

**double timeElapsed2 = 1000.0 \* (c\_end - c\_start) / CLOCKS\_PER\_SEC;**

**outputTime += timeElapsed2;**

**totalTime += timeElapsed2;**

**frameCount++;**

**if(waitKey(30) == 27) {**

**break;**

**}**

**}**

**ofstream TimeFile;**

**TimeFile.open("/home/rey/CLionProjects/opencvtest/time.txt");**

**double percent = totalTime / 100.0;**

**TimeFile << " CPU time for reading frames: " << inputTime / percent << “%” << endl;**

**TimeFile << " CPU time for processing frames:" << procTime / percent << “%” << endl;**

**TimeFile << "time for output frames: " << outputTime / percent << “%” << endl;**

**TimeFile << "Total framerate: " << ((frameCount - 1) / (totalTime / 1000.0)) << endl;**

**TimeFile.close();**

**return 0;**

**}**

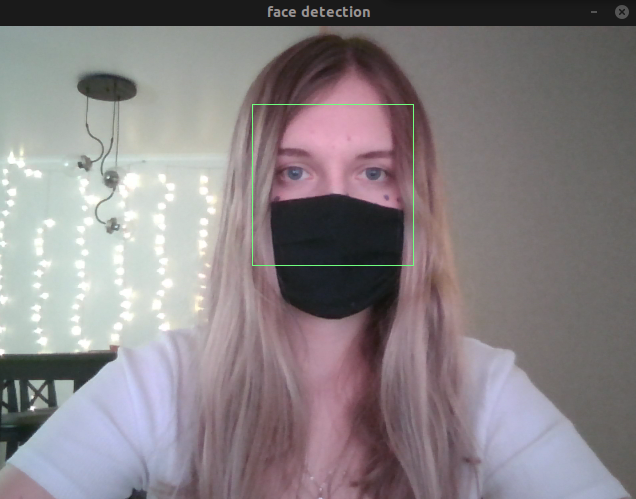
**Результат работы программы:**

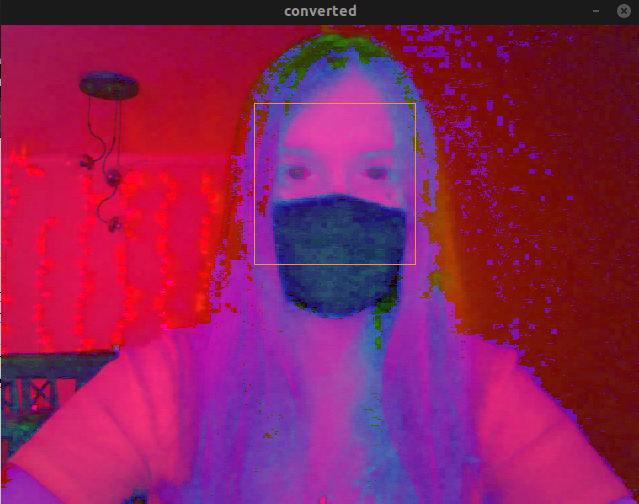
**Изображение до обработки:**

****

**Изображение после работы алгоритма распознавания лиц:**

**(как мы можем видеть, работает даже в маске)**

****

**Изображение после сдвига белого канала:**

**Оценка времени работы**

Содердимое файла time.txt:

CPU time for reading frames: 0.802917%

CPU time for processing frames:98.817%

time for output frames: 0.380082%

Total framerate: 1.52128

Эти данные отображают процентное соотноение времени, затрачиваемого на ввод, обработку и вывод изображения на экран. Видно что программа обрабатывает 1-2 кадра в секунду. Такая маленькая скорость обусловлена следующими факторами:

1. Вебкамера низкого качества;
2. В программе используются алгоритмы машинного обучения которые в настоящее время очень ресурсозатратны.

**Выводы**

В ходе работы с OpenCV были освоены:

работа с камерой;

работа с изображениями, полученными с камеры;

работа с функциями, работающими с предобученными моделями распознавания лиц;

вывод обработанного изображения на экран и работа с окнами в OpenCV.