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»

студента Бородина Артёма Максимовича 2 курса, 19205 группы

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
к.т.н, доцент
А.Ю. Власенко

Новосибирск 2020

СОДЕРЖАНИЕ

ЦЕЛЬ

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

ЗАДАНИЕ

1. Реализовать программу 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. Составить отчет по лабораторной работе.

ОПИСАНИЕ РАБОТЫ

1. Изучены основы c++ библиотеки OpenCV версии 4.1.0 и функции данной библиотеки для работы с изображениями и видео файлами.

2. Была написана программа для изменения видео изображения. (Приложение 1)

3. Результирующее изображение получалось путем наложения различных модификаций на каждый кадр видео. Основными модификациями являлись: уменьшение глубины цвета, изменение порогового значения изображения, наложение цветовой карты и шума. (Приложение 2)

4. При работе программы производились вычисления количества кадров в секунду и времени затраченного на обработку кадра. Данные значения выводились на каждом кадре.

5. Видео файл имел HD(1080*720) разрешение.

6. В среднем количество кадров в секунды равнялось 9, а время затраченное на каждый кадр 110мс.

ЗАКЛЮЧЕНИЕ

После изучения библиотеки компьютерного зрения OpenCV были получены практические знания работы с изображениями и видео, последующей их обработки.

Приложение 1. *Код программы*

```
#include <iostream>
#include <ctime>
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/core.hpp"

using namespace cv;

void addNoise(Mat &img) {
    Mat gaussianNoise = Mat::zeros(img.rows, img.cols, CV_8U);
    randn(gaussianNoise, 128, 20);

    std::vector<Mat> BGR;
    split(img, BGR);

    BGR[0] = BGR[0] + gaussianNoise * 0.3;
    BGR[1] = BGR[1] + gaussianNoise * 0.3;
    BGR[2] = BGR[2] + gaussianNoise * 0.3;

    merge(BGR, img);
}



Mat createLUT(int numColors) {
    if (numColors < 0 || numColors > 256) {
        std::cout << "Invalid number of colors. It must be between 0 and 256" << std::endl;
        return Mat::zeros(1, 256, CV_8U);
    }

    Mat lookupTable = Mat::zeros(1, 256, CV_8U);

    uchar *p = lookupTable.data;
    int startIdx = 0;
    for (int x = 0; x < 256; x += 256 / numColors) {
        p[x] = x;
        for (int y = startIdx; y < x; y++) {
            if (p[y] == 0) {
                p[y] = p[x];
            }
        }
        startIdx = x;
    }
    return lookupTable;
}

Mat reduceColors(Mat img, Mat &redLUT, Mat &greenLUT, Mat &blueLUT) {
    std::vector<Mat> BGR;
    split(img, BGR);

    LUT(BGR[0], blueLUT, BGR[0]);
    LUT(BGR[1], greenLUT, BGR[1]);
    LUT(BGR[2], redLUT, BGR[2]);

    merge(BGR, img);
    return img;
}

Mat cartoon(const Mat &img, Mat &redLUT, Mat &greenLUT, Mat &blueLUT) {
    Mat reducedColorImage = reduceColors(img, redLUT, greenLUT, blueLUT);
    Mat result;

    cvtColor(img, result, COLOR_BGR2GRAY);
    medianBlur(result, result, 9);
    adaptiveThreshold(result, result, 200, ADAPTIVE_THRESH_GAUSSIAN_C, THRESH_BINARY, 9, 10);
    cvtColor(result, result, COLOR_GRAY2BGR);
    bitwise_and(reducedColorImage, result, result);

    return result;
}
```

```
int main(int argc, char **argv) {
    VideoCapture cap("../video.mp4");
    if (!cap.isOpened()) {
        std::cout << "Could not open video file" << std::endl;
        return -1;
    }

    Size size = Size(cap.get(CAP_PROP_FRAME_WIDTH), cap.get(CAP_PROP_FRAME_HEIGHT));
    double fps = cap.get(CAP_PROP_FPS);

    VideoWriter outputVideo;
    bool check = outputVideo.open("result.mp4", 1983148141, fps, size, true);
    if (!check) {
        std::cout << "Could not open output file" << std::endl;
        return -1;
    }

    Mat redLUT = createLUT(9);
    Mat greenLUT = createLUT(9);
    Mat blueLUT = createLUT(9);

    Mat reducedColors, unedited, result;
    while (true) {
        clock_t start = clock();

        cap >> reducedColors;
        if (reducedColors.empty()) {
            std::cout << "empty frame" << std::endl;
            return -2;
        }
        unedited = reducedColors.clone();

        medianBlur(reducedColors, reducedColors, 5);
        result = cartoon(reducedColors, redLUT, greenLUT, blueLUT);
        applyColorMap(result, result, COLORMAP_BONE);
        addNoise(result);

        clock_t end = clock();
        double msCount = (end - start) / double(CLOCKS_PER_SEC);
        std::string fpsStr = std::to_string(int(1 / msCount));
        std::string toPrint = "fps: " + fpsStr + "  msPerFrame: " + std::to_string(end - start) + "ms";

        putText(result, toPrint, Point(30, 30), FONT_HERSHEY_SIMPLEX, 0.6, Scalar(255, 255, 255), 1);
        imshow("unedited", unedited);
        imshow("pre-edit", reducedColors);
        imshow("edited", result);

        outputVideo << result;

        char c = (char) waitKey(33);
        if (c == 27) {
            break;
        } else if (c == 'p') {      // pause
            bool pause = true;
            while (pause) {
                char t = (char) waitKey(1000);
                if (t == 'p') {
                    pause = false;
                }
            }
        }
    }

    cap.release();
    destroyAllWindows();
    return 0;
}
```

Приложение 2. *Пример ввода/вывода*