###### 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

«Лабораторная работа 5»

Студента 1 курса, 19210 группы

**Пирожков Андрей Константинович**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

(ученая степень, звание)

Д.С.Иванишкин

Новосибирск 2020

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc56767257)

[ЗАДАНИЕ 3](#_Toc56767258)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc56767259)

[ЗАКЛЮЧЕНИЕ 5](#_Toc56767260)

[Приложение *Листинг файла Lab5.c* 6](#_Toc56767261)

# ЦЕЛЬ

* 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. Отчет должен содержать следующее.
   * Титульный лист.
   * Цель лабораторной работы.
   * Полный компилируемый листинг реализованной программы и команды для ее компиляции.
   * Оценку скорости обработки видео (кадров в секунду) и долю времени, затрачиваемого процессором на обработку (ввод, показ) видеоданных
   * Вывод по результатам лабораторной работы.

# **ОПИСАНИЕ РАБОТЫ**

За основу этой лабораторной работы был взят код из примера. Поэтому программа была реализована без проблем.

Использовал эту команду для компиляции (код программы в [Приложении](#_Приложение_1)):

* gcc -o lab5.exe lab5.c -lopencv\_core -lopencv\_highgui

На основе примера обработки изображения реализовал эффект «негатив». Т.к. глубина изображения составляла 8, это означало, что на каждый цвет пикселя уходил 8 битов. Чтобы получить изображение в противоположном цвете, надо заменить битики на противоположное значение. Поэтому я просто из 255 вычитал само значение цвета. Таким образом я получил эффект «негатив».

Чтобы вычислять количество кадров в секунду я применил знания из самой первой лабораторной работы. Время выполнения участков программы фиксировал с помощью clock\_gettime. В моей программе было несколько участков программы:

* Общее количество времени показа видеоданных. Это время нужно для вычисления среднего общего показателя FPS. (Выводится в конце программы).
* Фиксировал время полной итерации цикла while. Для общего интереса и представления, сколько по продолжительности длится один полный цикл while. (Вывод времени в конце каждой итерации: «»).
* Замер времени на извлечении картинки с камеры. (Вывод времени и доли в конце каждой итерации: «»).
* Фиксирование времени на редактирование и применение эффекта «негатив». (Вывод времени и доли в конце каждой итерации: «»).
* Замер времени на вывод изображение на монитор, точнее на Xming. (Вывод времени и доли в конце каждой итерации: «»).
* Фиксирование времени на ожидание нажатия пользователя клавиши «Ecs» для остановки и выхода из программы. Зачем нужно фиксирование этого отрезка времени, я напишу далее. (Вывод времени и доли в конце каждой итерации: «»).

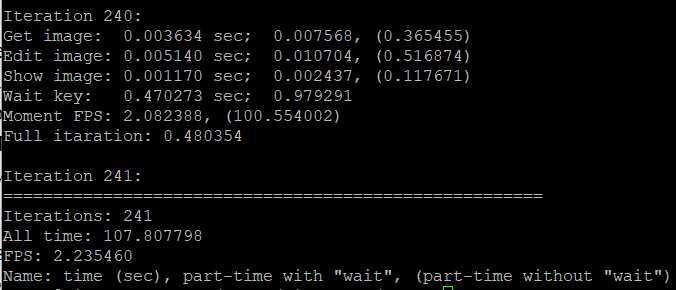
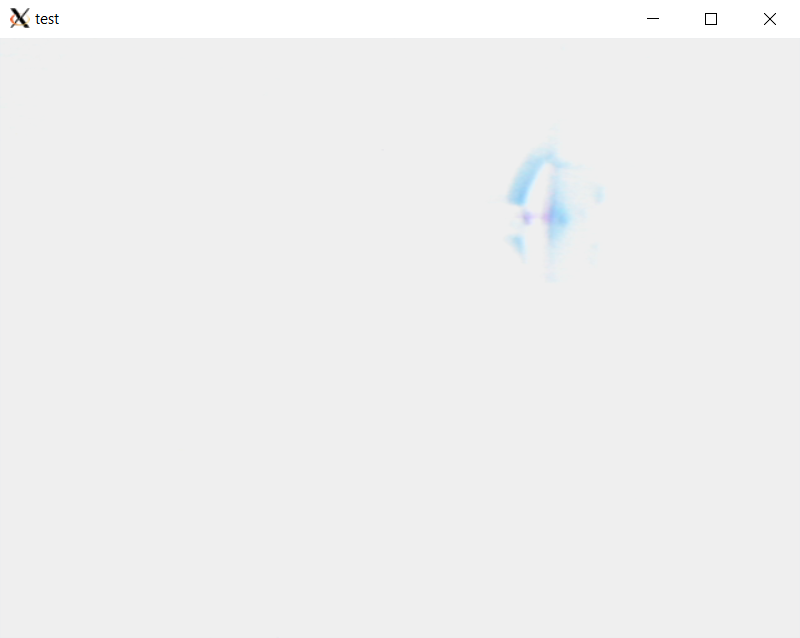
Далее я использовал эти участки времени для вычисления количество FPS. У меня их 2 типа. Один тип я выводил в конце. Он рассчитывается из количество полученных кадров, делённое на общее время. Тем самым я получаю средний показатель FPS и вывожу его перед завершением программы.

Второй тип я рассчитывал на каждый получаемый кадр. Т.е. я рассчитывал текущий FPS или как я его назвал «Moment FPS» и вычислялся он следующим образом: я делил одну секунду на время выполнения получения, обработки, вывода, и ожидания. Тем самым получая количество возможных обработанных кадров за одну секунду – что и есть FPS.

Рядом с вычислением текущего FPS присутствует значение в скобочках. Это значение FPS без учёта ожидания нажатия пользователя клавиши. Оно во много раз превосходит значение текущего FPS. Дело в том, что время ожидания занимает более 95% времени на получения одного кадра. Но, если убрать эту строчку, то FPS взлетает до 15. Почему так происходит мне не совсем ясно.

Ещё хотел упомянуть про вычисления долей времени. У меня сначала идёт вывод количества времени, затрачиваемого на определённый участок в программе. Затем есть ещё два значения, обозначающие долю времени на текущий участок. Первое значение доли идёт из расчётов с учётом ожидания нажатия клавиши пользователя. Из-за того, что доля времени ожидания занимает 95% сложно понять и оценить остальные три доли. Для них я сделал ещё отдельный расчёт и указал его в скобочках. Там я не включал время ожидания пользователя, поэтому их легче между собой сравнивать.

Вот скриншот результатов в терминале и скриншот камеры с Xming:



# **ЗАКЛЮЧЕНИЕ**

В ходе этой лабораторной работы я лабораторной работы я ознакомился с программированием устройств ввода данных Web-камеры с использованием библиотеки OpenCV. Также я реализовал программу, которая получает поток видеоданных с камеры и выводит его на экран. Применил преобразование получаемой картинки с эффектом «негатив». Также измерил FPS двумя способами, а также рассчитал доли времени затрачиваемыми процессором на обработку ввода, преобразования, показа и ожидания видеоданных. И проанализировал их.

# Приложение

*Листинг файла Lab5.c*

#include <opencv2/highgui/highgui.hpp>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

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

{

CvCapture\* capture = cvCreateCameraCapture(0);

if (!capture)

{

printf("Error in Video data\n");

return 0;

}

int i = 0;

struct timespec start, end, st0, st1, st2, st3, st4, st5, temp;

printf("Name: time (sec), part-time with \"wait\", (part-time without \"wait\")\n");

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

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

while (1)

{

temp = st0;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &st0);

printf("Full itaration: %lf\n\n", (st0.tv\_sec - temp.tv\_sec + 0.000000001 \* (st0.tv\_nsec - temp.tv\_nsec)));

i++;

printf("Iteration %d: \n", i);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &st1);

IplImage\* frame = cvQueryFrame(capture);

if (!frame)

{

printf("Error in Photo data\n");

break;

}

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &st2);

for (int y = 0; y < frame->height; y++)

{

uchar\* ptr = (uchar\*)(frame->imageData + y \* frame->widthStep);

for (int x = 0; x < frame->width; x++)

{

ptr[3 \* x] = 255 - ptr[3 \* x]; // Blue

ptr[3 \* x + 1] = 255 - ptr[3 \* x + 1]; // Green

ptr[3 \* x + 2] = 255 - ptr[3 \* x + 2]; // Red

}

}

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &st3);

cvShowImage("test", frame);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &st4);

char c = cvWaitKey(1);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &st5);

if (c == 27) break;

double part\_time = st5.tv\_sec - st1.tv\_sec + 0.000000001 \* (st5.tv\_nsec - st1.tv\_nsec);

double part\_fps = 1.0 / part\_time;

double part\_time\_without\_wait = st4.tv\_sec - st1.tv\_sec + 0.000000001 \* (st4.tv\_nsec - st1.tv\_nsec);

double time1 = st2.tv\_sec - st1.tv\_sec + 0.000000001 \* (st2.tv\_nsec - st1.tv\_nsec);

double time2 = st3.tv\_sec - st2.tv\_sec + 0.000000001 \* (st3.tv\_nsec - st2.tv\_nsec);

double time3 = st4.tv\_sec - st3.tv\_sec + 0.000000001 \* (st4.tv\_nsec - st3.tv\_nsec);

double time4 = st5.tv\_sec - st4.tv\_sec + 0.000000001 \* (st5.tv\_nsec - st4.tv\_nsec);

double sum\_time1 = time1 + time2 + time3;

double sum\_time2 = sum\_time1 + time4;

printf("Get image: %lf sec; %lf, (%lf)\n", time1, (time1 / sum\_time2), (time1 / sum\_time1));

printf("Edit image: %lf sec; %lf, (%lf)\n", time2, (time2 / sum\_time2), (time2 / sum\_time1));

printf("Show image: %lf sec; %lf, (%lf)\n", time3, (time3 / sum\_time2), (time3 / sum\_time1));

printf("Wait key: %lf sec; %lf\n", time4, (time4 / sum\_time2));

printf("Moment FPS: %lf, (%lf)\n", 1.0/ sum\_time2, 1.0/sum\_time1);

}

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

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

printf("Iterations: %d\n", i);

double time = end.tv\_sec - start.tv\_sec + 0.000000001 \* (end.tv\_nsec - start.tv\_nsec);

printf("All time: %lf\n", time);

double fps = (double)i / time;

printf("FPS: %lf\n", fps);

printf("Name: time (sec), part-time with \"wait\", (part-time without \"wait\")\n");

cvReleaseCapture(&capture);

cvDestroyWindow("test");

return 0;

}