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, 24202 группы

Корсун Дарьи Андреевны

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
В.А.Перепёлкин

Новосибирск 2025

СОДЕРЖАНИЕ

ЦЕЛЬ

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

ЗАДАНИЕ

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

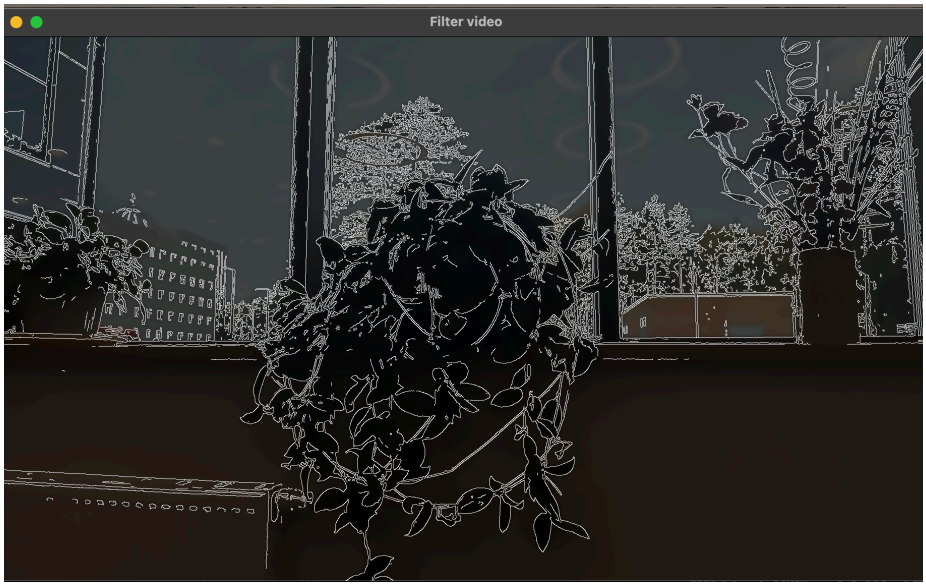
Для начала работы с Web-камеры с использованием библиотеки OpenCV. Эта библиотека была установлена на ноутбук с помощью ввода в терминал команды «-brew install opencv»

Далее был написан код, изменяющий поток видеоданных с камеры. Ниже представлен код:

```
#include <time.h>

#include <iomanip>
#include <iostream>
#include <opencv2/opencv.hpp>

using namespace cv;
using namespace std;

int main() {
  VideoCapture video(0);
  if (!video.isOpened()) return 0;

  int frames = 0;
  double total_capture_time = 0;
  double total_processing_time = 0;
  double total_display_time = 0;
  double total_wait_time = 0;

  struct timespec start, end;
  struct timespec program_start, program_end;

  clock_gettime(CLOCK_MONOTONIC_RAW, &program_start);

  while (true) {
    clock_gettime(CLOCK_MONOTONIC_RAW, &start);

    Mat frame;
    video >> frame;
    if (frame.empty()) break;

    clock_gettime(CLOCK_MONOTONIC_RAW, &end);
    double capture_time = (end.tv_sec - start.tv_sec) +
                          0.000000001 * (end.tv_nsec -
start.tv_nsec);

    clock_gettime(CLOCK_MONOTONIC_RAW, &start);

    Mat edges;
    cvtColor(frame, edges, COLOR_BGR2GRAY);
    edges.convertTo(edges, -1, 1.5, 10);
    Canny(edges, edges, 50, 90, 3);
```

```
Mat blured_backgroung;
    GaussianBlur(frame, blured_backgroung, Size(21, 21), 0);
    blured_backgroung = blured_backgroung * 0.3;
    blured_backgroung.setTo(Scalar(255, 255, 255), edges);

    clock_gettime(CLOCK_MONOTONIC_RAW, &end);

    double processing_time = (end.tv_sec - start.tv_sec) +
                             0.000000001 * (end.tv_nsec -
start.tv_nsec);

    clock_gettime(CLOCK_MONOTONIC_RAW, &start);

    imshow("Filter video", blured_backgroung);

    clock_gettime(CLOCK_MONOTONIC_RAW, &end);
    double display_time = (end.tv_sec - start.tv_sec) +
                          0.000000001 * (end.tv_nsec -
start.tv_nsec);

    clock_gettime(CLOCK_MONOTONIC_RAW, &start);

    int key = waitKey(5);

    clock_gettime(CLOCK_MONOTONIC_RAW, &end);

    double wait_time = (end.tv_sec - start.tv_sec) +
                       0.000000001 * (end.tv_nsec - start.tv_nsec);

    total_capture_time += capture_time;
    total_processing_time += processing_time;
    total_display_time += display_time;
    total_wait_time += wait_time;
    frames++;

    if (key == 27) break;
  }

  clock_gettime(CLOCK_MONOTONIC_RAW, &program_end);
double total_program_time =
      (program_end.tv_sec - program_start.tv_sec) +
      0.000000001 * (program_end.tv_nsec - program_start.tv_nsec);

  double real_fps = frames / total_program_time;
```

```
cout << "Capture time: " << fixed << setprecision(2)
       << total_capture_time * 1000.0 << " ms (" << fixed <<
setprecision(1)
       << (total_capture_time * 100.0 / total_program_time) << "%)"
<< endl;
  cout << "Processing time: " << fixed << setprecision(2)
       << total_processing_time * 1000.0 << " ms (" << fixed <<
setprecision(1)
       << (total_processing_time * 100.0 / total_program_time) <<
"%)" << endl;
  cout << "Display time: " << fixed << setprecision(2)
       << total_display_time * 1000.0 << " ms (" << fixed <<
setprecision(1)
       << (total_display_time * 100.0 / total_program_time) << "%)"
<< endl;
  cout << "Wait time: " << fixed << setprecision(2) <<
total_wait_time * 1000.0
       << " ms (" << fixed << setprecision(1)
       << (total_wait_time * 100.0 / total_program_time) << "%)" <<
endl;
  cout << "FPS: " << fixed << setprecision(2) << real_fps << endl;

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



Программа реализует затемнение и размытие изначального видео с наложением границ.

Далее были проведены измерения количество кадров FPS (обрабатываемое программой в секунду).Также были посчитаны доли времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) потока видеоданных, получаемых с камеры. Показания брались среднее для нескольких десятков секунд работы программы, так как за это время программа успевает достаточно

Process:	Proportion
Capture time:	25.7%
Processing time:	22.4%
Display time:	1.8%
Wait time:	49.0%

FPS = 30 кадров в секунду.
Вывод программы, значения из которого занесены в таблицу:

```
Capture time: 86227.95 ms (25.7%)
Processing time: 75228.62 ms (22.4%)
Display time: 6088.16 ms (1.8%)
Wait time: 164481.68 ms (49.0%)
FPS: 29.95
```

Как видно примерно половину времени программы провела в ожидании следующего кадра. Следующее по затратам было времени необходимое для получения потока видеоданных, в то время как вывод видеоданных занял менее 2%

ЗАКЛЮЧЕНИЕ

Получилось ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV. Также были выполнены измерения, касающиеся количества кадров, обрабатываемых программой в секунду, долях времени затрачиваемые процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.