Новосибирский Государственный Университет

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Курс “ЭВМ и переферийные устройства”

Лабораторная работа №5

«ВЫСОКОУРОВНЕВАЯ РАБОТА С ПЕРЕФЕРИЙНЫМИ УСТРОЙСТВАМИ»

Выполнил:
Пятаев Егор, гр. 15206
Преподаватель:
Городничев Максим Александрович

Новосибирск 2016

Цели работы

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

Листинг реализованной программы

main.cpp:

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <cmath>
#include <time.h>

using namespace cv;
using namespace std;

int main(int argc, char** argv )
{
  struct timespec start, end;

  double t = 1;
  CvCapture *capture = cvCreateCameraCapture(0);

  if(!capture) return 0;


  while(1){
    clock_gettime(CLOCK_MONOTONIC_RAW, &start);
    double st = clock();
    IplImage * image = cvQueryFrame(capture);
    IplImage * frame = cvCloneImage(image);

    if(!image) break;

    for (int y=0; y<image->height / 2; y++) {
      uchar *ptr = (uchar*)(image->imageData + y*image->widthStep);
      for(int x=0; x<image->width / 2;x++){
        ptr[3*x] = 64;
        ptr[3*x+2] = 64;
      }
    }

    for (int y = image->height/2; y<image->height; y++) {
      uchar *ptr = (uchar*)(image->imageData + y*image->widthStep);
      for(int x = image->width/2; x<image->width;x++){
        ptr[3*x] = 192;
        ptr[3*x+2] = 0;
      }
    }
```

```
    for (int y = image->height/2; y<image->height; y++) {
      uchar *ptr = (uchar*)(image->imageData + y*image->widthStep);
      for(int x = 0; x<image->width /2;x++){
        ptr[3*x] = 192;
        ptr[3*x+2] = 192;
      }
    }
    for (int y = 0; y<image->height / 2; y++) {
      uchar *ptr = (uchar*)(image->imageData + y*image->widthStep);
      for(int x = image->width/2; x<image->width;x++){
        ptr[3*x] = 0;
        ptr[3*x+2] = 224;
      }
    }
    for (int y = image->height / 2; y < (image->height / 2) + 128; y++) {
      uchar *ptr = (uchar*)(image->imageData + y*image->widthStep);
      for(int x = (image->width / 2) - sqrt(16384 - ((y - image->height / 2))*(y -
(image->height / 2))); x < image->width / 2 + sqrt(16384 - ((y - image->height /
2))*(y - (image->height / 2))); x++){
        ptr[3*x] = 12;
        ptr[3*x+2] = 122;
      }
    }

    for (int y = image->height / 2; y > (image->height / 2) - 128; y--) {
      uchar *ptr = (uchar*)(image->imageData + y*image->widthStep);
      for(int x = (image->width / 2) - sqrt(16384 - ((y - image->height / 2))*(y -
(image->height / 2))); x < image->width / 2 + sqrt(16384 - ((y - image->height /
2))*(y - (image->height / 2))); x++){
        ptr[3*x] = 128;
        ptr[3*x+2] = 128;
      }
    }

    cvFlip(image, image, 1);

    cvShowImage("test1", image);
    cvShowImage("test2",frame);

    char c = cvWaitKey(33);

    double fi = clock();
    clock_gettime(CLOCK_MONOTONIC_RAW, &end);
```

```
    if(c == 27) break;

    cout \
    << "All time: " << (end.tv_sec-start.tv_sec+ 0.000000001*(end.tv_nsec-
start.tv_nsec)) << "\n" \
    << "Prog time: " << ((fi - st) / CLOCKS_PER_SEC) << "\n"\
    << "FPS: ~" << t/(end.tv_sec-start.tv_sec+ 0.000000001*(end.tv_nsec-
start.tv_nsec)) << "\n" \
    << "Part: "<< ((end.tv_sec-start.tv_sec + 0.000000001*(end.tv_nsec-start.tv_nsec))
- ((fi - st) / CLOCKS_PER_SEC))/(end.tv_sec-start.tv_sec +
0.000000001*(end.tv_nsec-start.tv_nsec)) \
    * 100 << "%"<< endl << endl;

  }

  cvReleaseCapture(&capture);
  cvDestroyWindow("test");

  return 0;
}
```

CMakeLists.txt:

```
cmake_minimum_required(VERSION 2.8)
project( DisplayImage )
find_package( OpenCV REQUIRED )
add_executable( DisplayImage main.cpp )
target_link_libraries( DisplayImage ${OpenCV_LIBS} )
```

Компиляция:

```
~$ cmake .
~$ make
```

Запуск:

```
~$ ./DisplayImage
```

Формулы расчета

Кадры в секунду:

1 кадр : (Т завершения итерации цикла – Т начала итерации цикла)

Доля времени:

$((T_{зав.\ итер.\ ц.} - T_{нач.\ итер.\ ц.}) - (T_{зав.\ проц.\ в\ итер.} - T_{нач.\ проц.\ в\ итер.})) : (T_{зав.\ итер.\ ц.} - T_{нач.\ итер.\ ц.})$

Выводы

При выполнении работы была реализована программа с использованием библиотеки OpenCV, которая получает поток видеоданных с камеры и выводит его на экран.

Были выполнены некоторые преобразования изображения (изменение параметров цветовых каналов разных частей изображения, зеркальное преобразование изображения).

Замеры производились в один цикл обработки потока с камеры, для замера кадров в секунду использована функция clock_gettime, так замеряет время работу процессора, а для оценки доли времени, затрачиваемого процессором на обработку данных дополнительно использован таймер времени процесса clock(), чтобы замерить время одного процесса без учета времени затраченного на работу с библиотекой.

Кадры в секунду: ~ 25.

Доля времени: ~ 84%.