ЛАБОРАТОРНАЯ РАБОТА №5
ВЫСОКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

Цели работы

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

1. БИБЛИОТЕКА OPENCV

Библиотека OpenCV предоставляет высокоуровневый интерфейс для разработки прикладных программ в области машинного зрения. OpenCV включает следующие группы функций: работа с растровыми изображениями и их форматами, работа с видеоданными и их форматами, интерфейс для работы с камерами, машинное обучение с помощью нейронных сетей и других моделей, реализация упрощенного оконного интерфейса, операции над векторами и матрицами и другие.

2. ВЫВОД ВИДЕОПОТОКА С КАМЕРЫ НА ЭКРАН С ПОМОЩЬЮ БИБЛИОТЕКИ OPENCV

Рассмотрим последовательность деклараций и действий в программе, которая вводит изображение с камеры или из файла и показывает его в окне.

1. Создается поток ввода видеоданных с первой камеры (нумерация начинается с нуля). Результат операции – указатель на дескриптор обрабатываемого потока видеоданных.

```
CvCapture *capture = cvCreateCameraCapture(0);
if (!capture) return 0;
```

2. Запускается циклическая обработка потока видеоданных.

```
while(1) {
```

3. На каждой итерации извлекается очередной кадр (изображение) из открытого потока ввода видеоданных. Результат операции – указатель на дескриптор растрового изображения, которое будет содержать один кадр из потока видеоданных.

```
IplImage *frame = cvQueryFrame(capture);
if(!frame) break;
```

4. Текущий кадр выводится в окно с именем test (при первом вызове создается окно с таким именем):

```
cvShowImage("test", frame);
```

5. В течение 33 миллисекунд ожидается нажатие клавиши пользователем (если клавиша не была нажата, вызов срабатывает как задержка на 33 мс, что обеспечивает частоту показа примерно 30 кадров в сек):

```
char c = cvWaitKey(33);
```

6. Если пользователь нажал клавишу Esc, выйти из цикла обработки и показа кадров видеопотока:

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

7. Перейти к следующей итерации цикла.

```
}
```

8. В эту точку программы можно попасть в двух случаях: во-первых, когда в видеопотоке больше нет кадров (конец видео); во-вторых, когда пользователь нажал клавишу Esc. Здесь производится удаление потока ввода видеоданных, освобождение занятых им ресурсов, а также удаление окна, в которое выводились кадры потока видеоданных:

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

Пример программы ввода изображения с камеры и показа в окне:

```
#include <opencv2/highgui/highgui.hpp>
int main(int argc,char *argv[])
{ CvCapture *capture = cvCreateCameraCapture(0);
  if (!capture) return 0;
```

```
  while(1) {
    IplImage *frame = cvQueryFrame(capture);
    if(!frame) break;
    cvShowImage("test", frame);
    char c = cvWaitKey(33);
    if(c == 27) break;
  }
  cvReleaseCapture(&capture);
  cvDestroyWindow("test");
}
```

Пример команды компиляции программы с использованием OpenCV:

```
gcc -o prog prog.c –lopencv_core –lopencv_highgui
```

3. ПРЕОБРАЗОВАНИЕ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ БИБЛИОТЕКИ OPENCV

Библиотека OpenCV предоставляет набор функций, позволяющих выполнять различные преобразования изображений, например, сглаживание, морфологические преобразования (сужение и расширение), пороговые преобразования и другие. Следующий пример демонстрирует создание копии изображения `frame`, выполнение над ним библиотечной операции простого сглаживания и вывод на экран в окне:

```
IplImage *image = cvCloneImage(frame);
cvSmooth(frame,image,CV_BLUR,3,3);
cvShowImage(“smooth”,image);
```

Кроме того, OpenCV предоставляет прямой доступ к данным изображения, что позволяет производить редактирование изображения «вручную». Следующие поля дескриптора изображения (структуры `IplImage`) необходимы для корректного доступа к данным изображения:

- nChannels – число цветовых каналов,

- depth – глубина цвета в битах,
- width – ширина изображения в пикселях,
- height – высота изображения в пикселях,
- widthStep – расстояние между данными соседних пикселей по вертикали в байтах,
- imageData – указатель на данные изображения.

Данные изображения представляют собой массив пикселей с числом строк height и числом столбцов width, начинающийся с адреса imageData. Объем данных для одного пикселя в битах определен как nChannels*2^depth.

Далее приводится пример, в котором происходит обнуление данных красного и синего каналов изображения (nChannels: 3, depth: 8):

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

4. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

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.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое OpenCV? Какие задачи решает OpenCV?
2. Как с помощью OpenCV получить изображение с видеокамеры и вывести его на экран?
3. Какие способы преобразования изображений можно использовать, применяя OpenCV.

ЛАБОРАТОРНАЯ РАБОТА №6
НИЗКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

Цели работы

Ознакомиться с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.

1. БИБЛИОТЕКА LIBUSB

USB (Universal Serial Bus) – последовательный интерфейс передачи данных для среднескоростных и низкоскоростных периферийных устройств. Стандарт USB ввел единообразие в работу с широким спектром периферийных устройств. До его появления использовалось множество разных интерфейсов для подключения периферийных устройств (последовательный порт RS-232C для модема, мыши, параллельный порт для принтера и дисководов IOMEGA ZIP, PS/2 для мыши и клавиатуры, специализированные интерфейсы для подключения сканеров и т. д.).

Программирование взаимодействия с USB-устройствами достаточно трудоемко. Для упрощения работы с USB-устройствами из прикладных программ была создана многоплатформенная библиотека libusb. Она реализует универсальный низкоуровневый интерфейс, который позволяет взаимодействовать со всеми возможными USB-устройствами. С помощью библиотеки libusb можно получить список доступных USB-устройств, узнать их параметры, обмениваться данными с устройствами в синхронном и асинхронном режимах, реагировать на подключение и отключение устройств.

Каждое конкретное устройство имеет свой специальный протокол взаимодействия, и прикладная программа должна взаимодействовать с

устройством в соответствии с его протоколом, используя низкоуровневые операции библиотеки libusb.

2. ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ДОСТУПНЫХ USB-УСТРОЙСТВАХ С ПОМОЩЬЮ БИБЛИОТЕКИ LIBUSB

Библиотека libusb предоставляет следующие функции, которые позволяют получить информацию об имеющихся в системе USB-устройствах:

- libusb_init – инициализация работы с libusb,
- libusb_exit – завершение работы с libusb,
- libusb_set_debug – установка уровня подробности отладочных сообщений (рекомендуется использовать уровень 3),
- libusb_get_device_list – получение списка подключенных к машине USB устройств,
- libusb_free_device_list – освобождение памяти, выделенной в функции libusb_get_device_list для хранения данных со списком устройств,
- libusb_get_device_descriptor – получение дескриптора USB устройства,
- libusb_get_config_descriptor – получение дескриптора конфигурации USB устройства,
- libusb_free_config_descriptor – освобождение памяти, выделенной в функции libusb_get_config_descriptor,
- libusb_ref_device – увеличение счетчика числа пользователей устройства на 1 (при первом вызове функции libusb_get_device_list после подключения устройства его счетчик устанавливается в 1),
- libusb_unref_device – уменьшение счетчика числа пользователей устройства на 1 (если счетчик уменьшается до нуля, дескриптор устройства удаляется),

- libusb_open – открыть устройство (начать работать с устройством) и получить дескриптор устройства, который далее можно использовать для ввода/вывода данных,
- libusb_open_device_with_vid_pid – открыть устройство по его идентификаторам производителя и изделия,
- libusb_close – закрыть устройство после его использования,
- libusb_get_string_descriptor_ascii – получить дескриптор устройства в виде строки символов,
- lib_usb_error_name – преобразование кода ошибки библиотеки libusb в строковое сообщение об ошибке.

Пример программы получения состава и параметров USB-устройств с помощью библиотеки libusb приведен в листинге 1.

Листинг 1. Получение параметров USB-устройств

```
#include <iostream>
#include <libusb.h>
#include <stdio.h>

using namespace std;

void printdev(libusb_device *dev);

int main(){
  libusb_device **devs; // указатель на указатель на устройство,
                        // используется для получения списка устройств
  libusb_context *ctx = NULL; // контекст сессии libusb
  int r;                // для возвращаемых значений
  ssize_t cnt;          // число найденных USB-устройств
  ssize_t i;            // индексная переменная цикла перебора всех устройств
  // инициализировать библиотеку libusb, открыть сессию работы с libusb
  r = libusb_init(&ctx);
  if(r < 0){
    fprintf(stderr,
      "Ошибка: инициализация не выполнена, код: %d.\n", r);
    return 1;
  }
  // задать уровень подробности отладочных сообщений
  libusb_set_debug(ctx, 3);
  // получить список всех найденных USB- устройств
```

```
  cnt = libusb_get_device_list(ctx, &devs);
  if(cnt < 0){
    fprintf(stderr,
      "Ошибка: список USB устройств не получен.\n", r);
    return 1;
  }
  printf("найдено устройств: %d\n", cnt);
  printf("=============================="
         "==============================\n");
  printf("* количество возможных конфигураций\n");
  printf("|  * класс устройства\n");
  printf("|  |  * идентификатор производителя\n");
  printf("|  |  |    * идентификатор устройства\n");
  printf("|  |  |    |    * количество интерфейсов\n");
  printf("|  |  |    |    |   * количество "
         "альтернативных настроек\n");
  printf("|  |  |    |    |   |  *  класс устройства\n");
  printf("|  |  |    |    |   |  |  * номер интерфейса\n");
  printf("|  |  |    |    |   |  |  |  * количество "
         "конечных точек\n");
  printf("|  |  |    |    |   |  |  |  |  * тип дескриптора\n");
  printf("|  |  |    |    |   |  |  |  |  |  * адрес "
         "конечной точки\n");
  printf("+--+--+----+----+---+--+--+--+"
         "--+--+----------------------\n");
  for(i = 0; i < cnt; i++) {  // цикл перебора всех устройств
    printdev(devs[i]);        // печать параметров устройства
  }
  printf("=============================="
         "==============================\n");
  // освободить память, выделенную функцией получения списка устройств
  libusb_free_device_list(devs, 1);
  libusb_exit(ctx);           // завершить работу с библиотекой libusb,
                              // закрыть сессию работы с libusb
  return 0;
}

void printdev(libusb_device *dev){
  libusb_device_descriptor desc;    // дескриптор устройства
  libusb_config_descriptor *config; // дескриптор конфигурации объекта
  const libusb_interface *inter;
  const libusb_interface_descriptor *interdesc;
  const libusb_endpoint_descriptor *epdesc;
  int r = libusb_get_device_descriptor(dev, &desc);
  if (r < 0){
    fprintf(stderr,
      "Ошибка: дескриптор устройства не получен, код: %d.\n",r);
    return;
  }
  // получить конфигурацию устройства
  libusb_get_config_descriptor(dev, 0, &config);
  printf("%.2d %.2d %.4d %.4d %.3d |  |  |  |  |  |\n",
```

```
    (int)desc.bNumConfigurations,
    (int)desc.bDeviceClass,
    desc.idVendor,
    desc.idProduct,
    (int)config->bNumInterfaces
  );
  for(int i=0; i<(int)config->bNumInterfaces; i++){
    inter = &config->interface[i];
    printf("|  |  |    |    |   "
      "%.2d %.2d |  |  |  |\n",
      inter->num_altsetting,
      (int)desc.bDeviceClass
    );
    for(int j=0; j<inter->num_altsetting; j++) {
      interdesc = &inter->altsetting[j];
      printf("|  |  |    |    |   |  |  "
        "%.2d %.2d |  |\n",
        (int)interdesc->bInterfaceNumber,
        (int)interdesc->bNumEndpoints
      );
      for(int k=0; k<(int)interdesc->bNumEndpoints; k++) {
        epdesc = &interdesc->endpoint[k];
        printf(
          "|  |  |    |    |   |  |  |  |  "
          "%.2d %.9d\n",
          (int)epdesc->bDescriptorType,
          (int)(int)epdesc->bEndpointAddress
        );
      }
    }
  }
  libusb_free_config_descriptor(config);
}
```

Для компиляции программы в листинге 1 можно использовать команду:

```
g++ -o executable -I/usr/include/libusb-1.0 main.cpp -lusb-1.0
```

Примечание. При запуске на собственном компьютере может понадобиться разрешить доступ к USB-устройствам по чтению и записи. Например:

```
chmod 777 /dev/bus/usb/001/003
```

Альтернативный вариант заключается в запуске программы с правами администратора:

```
sudo ./executable
```

3. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

1. Реализовать программу, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатать его класс, идентификатор производителя и идентификатор изделия. За основу для разработки можно взять программу, приведенную в листинге 1.
2. Изучить состав и характеристики обнаруженных с помощью реализованной программ USB устройств.
3. Дополнить программу, реализованную в п. 2 функцией печати серийного номера USB устройства. Для написания функции рекомендуется использовать функции libusb_open, libusb_close, libusb_get_string_descriptor_ascii для печати поля iSerialNumber дескриптора устройства.
4. Составить отчет по лабораторной работе. Отчет должен содержать следующие пункты:
    - титульный лист,
    - цель лабораторной работы,
    - полный компилируемый листинг реализованной программы и команды для ее компиляции,
    - описание обнаруженных USB-устройств,
    - вывод по результатам лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое libusb? Какие задачи решает libusb?
2. На какие группы можно разбить функции libusb?
3. Как с помощью библиотеки libusb получить состав и конфигурацию USB-устройств?

5. ЛИТЕРАТУРА

1. http://www.dreamincode.net/forums/topic/148707-introduction-to-using-libusb-10/
2. http://www.usb.org/

ПРИЛОЖЕНИЕ

Коды классов USB устройств

00h – код отсутствует (информацию о классе нужно получать в дескрипторе интерфейса)

01h – аудиоустройство (если код получен из дескриптора интерфейса, а не устройства)

02h – коммуникационное устройство (сетевой адаптер)

03h – устройство пользовательского интерфейса

05h – физическое устройство

06h – изображения

07h – принтер

08h – устройство хранения данных

09h – концентратор

0Ah – CDC-Data

0Bh – Smart Card

0Dh – Content Security

0Eh – видеоустройство

0Fh – персональное медицинское устройство

10h – аудио- и видеоустройства

DCh – диагностическое устройство

E0h – беспроводный контроллер

EFh – различные устройства

FE – специфическое устройство