Thông tin đào tạo

Thông báo
Lịch khai giảng
Đăng ký học

Nội dung mới cập nhật

Thông báo tạm dừng các khóa đào tạo lập trình nhúng

USB Device Drivers, Phần 1. USB driver trên Linux (1)
USB Device Drivers, Phần 2. USB driver trên Linux (2)
USB Device Drivers, Phần 3. Trao đổi dữ liệu với thiết bị USB

Device Drivers, Phần 1. Cơ bản về driver trên Linux
Device Drivers, Phần 2. Viết driver đầu tiên trên Linux
Device Drivers, Phần 3. Lập trình C ở tầng nhân
Device Drivers, Phần 4. Linux Character Drivers
Device Drivers, Phần 5. Character Device Files
Device Drivers, Phần 6. Các thao tác đối với file thiết bi

Lập trình mạng > Xử lý ảnh OpenCV >

Thu nhận ảnh từ camera sử dụng V4L2 trên Linux

OpenCV cung cấp các hàm cần thiết để thu nhận (capture) ảnh từ thiết bị usb camera, webcam. Các hàm này khá thuận tiện và dễ dàng sử dụng trong nhiều tình huống khác nhau. Tuy nhiên, khi phát triển ứng dụng làm việc với thiết bị video trên Linux (PC, Arm Linux), chúng ta còn có thể sử dụng các API mà hệ điều hành Linux cung cấp, đó là V4L2 (OpenCV hỗ trợ V4L2 thông qua lựa chọn khi biên dịch). Sử dụng các API của Linux tuy phức tạp hơn các hàm của OpenCV nhưng cũng có nhiều ưu điểm hơn (dễ xác định thiết bị camera muốn làm việc, khắc phục tình huống với các thiết bị camera mà hàm cvCaptureFromCAM của OpenCV cho ARM không mở được, trả về NULL)

Bài viết này hướng dẫn sử dụng các API cơ bản của V4L2 để làm việc với thiết bị camera trên Linux. Dữ liệu ảnh thu nhận được sẽ được chuyển đổi sang định dạng thích hợp của thư viện OpenCv để tiếp tục xử lý bằng các hàm, thuật toán của OpenCV.

Giới thiệu V4L2

V4L2 (Video for Linux 2) là phiên bản 2 của thư viện Video For Linux cung cấp các hàm API trên hệ điều hành Linux. Các tài liệu mô tả về các API này có thể tìm hiểu ở đây

Các bước chính để sử dụng V4L2 API

Bước 1. Open capture device (Mở thiết bị camera)

Trên Linux, thiết bị camera (đã nhận driver tương thích) thường có tên file thiết bị dạng /dev/video0 (hoặc /dev/video1, 2, … tùy thuộc số thiết bị hoặc cổng usb kết nối)

Dùng hàm open() để mở file thiết bị với device file tương ứng

Trang web được xây dựng bởi

Embedded247
Bộ môn Kỹ thuật máy tính,
Viện Công nghệ Thông tin và Truyền thông,
Đại học Bách Khoa Hà Nội
Phòng 502, B1, ĐH BKHN
Tel: 04-38696125


```
int fd;

fd = open("/dev/video0", O_RDWR);

if (fd == -1)

{

    // couldn't find capture device

    perror("Opening Video device");

    return 1;

}
```

Bước 2. Query Capture (Truy vấn thiết bị)

Cần kiểm tra xem thiết bị được capture có sẵn sàng hay không. V4L2 không hỗ trợ một số loại thiết bị usb camera, khi đó sẽ phát sinh lỗi. Chúng ta sử dụng cấu trúc v4l2_capability và VIDIOC_QUERYCAP để truy vấn thiết bị. (chi tiết thêm ở đây)

```
struct v4l2_capability caps = {0};

if (-1 == xioctl(fd, VIDIOC_QUERYCAP, &caps))

{

    perror("Querying Capabilites");

    return 1;

}
```

Ở đây hàm xioctl là một "hàm đóng gói" (wrapper function) của hàm ioctl. Hàm ioctl có chức năng thao tác với các tham số của file thiết bị. (Chi tiết ở đây)

```
#include <sys/ioctl.h>
```

```
static int xioctl(int fd, int request, void *arg)

{

    int r;

        do r = ioctl (fd, request, arg);

        while (-1 == r && EINTR == errno);

        return r;

}
```

Bước 3. Image Format (Định dạng ảnh thu nhận)

V4L2 cung cấp API để kiểm tra định dạng ảnh và không gian màu sử dụng mà thiết bị camera hỗ trợ và cung cấp. Cấu trúc v4l2_format được sử dụng để thay đổi định dạng ảnh

```
struct v4l2_format fmt = {0};

fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;

fmt.fmt.pix.width = 320;

fmt.fmt.pix.height = 240;

fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_MJPEG;

fmt.fmt.pix.field = V4L2_FIELD_NONE;


if (-1 == xioctl(fd, VIDIOC_S_FMT, &fmt))

{

    perror("Setting Pixel Format");
```

```
    return 1;

}
```

Trong trường hợp này, chúng ta thiết lập kích thước khung hình là 320x240 (width x height). Kiểm tra khuôn dạng dữ liệu mà camera hỗ trợ (MJPEG, YUV, …)

Bước 4. Request Buffers

Một buffer được sử dụng để chứa dữ liệu trao đổi bởi ứng dụng và driver thiết bị sử dụng phương pháp streaming I/O. Sử dụng cấu trúc v4l2_requestbuffers để cấp phát buffers này.

```
struct v4l2_requestbuffers req = {0};

req.count = 1;

req.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;

req.memory = V4L2_MEMORY_MMAP;

if (-1 == xioctl(fd, VIDIOC_REQBUFS, &req))

{

    perror("Requesting Buffer");

    return 1;

}
```

Hàm ioctl được sử dụng để khởi tạo một memory mapped (mmap)

Bước 5. Query buffer

Sau khi yêu cầu buffer từ thiết bị, chúng ta sẽ truy vấn đến buffer này để lấy dữ liệu thô (raw data)

```
struct v4l2_buffer buf = {0};

buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
```

```
buf.memory = V4L2_MEMORY_MMAP;

buf.index = bufferindex;

if(-1 == xioctl(fd, VIDIOC_QUERYBUF, &buf))

{

    perror("Querying Buffer");

    return 1;

}


buffer = mmap (NULL, buf.length, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, buf.m.offset);
```

Bước 6. Capture Image

Sau khi truy vấn buffer thành công, vấn đề còn lại là thu nhận một khung hình và lưu nó vào trong buffer.

```
if(-1 == xioctl(fd, VIDIOC_STREAMON, &buf.type))

{

    perror("Start Capture");

    return 1;

}

fd_set fds;

FD_ZERO(&fds);
```

```
FD_SET(fd, &fds);

struct timeval tv = {0};

tv.tv_sec = 2;

int r = select(fd+1, &fds, NULL, NULL, &tv);

if(-1 == r)

{

    perror("Waiting for Frame");

    return 1;

}

if(-1 == xioctl(fd, VIDIOC_DQBUF, &buf))

{

    perror("Retrieving Frame");

    return 1;

}
```

Bước 7. Store data in OpenCV datatype

Dữ liệu thu nhận được chuyển đổi sang cấu trúc ảnh của OpenCV, và lưu lại thành một file ảnh:

```
CvMat cvmat = cvMat(480, 640, CV_8UC3, (void*)buffer);

IplImage * img;

img = cvDecodeImage(&cvmat, 1);
```

Chương trình minh họa đơn giản các bước trên viết bằng c có thể tham khảo file đính kèm.

Để thuận tiện sử dụng phương pháp này trong các ứng dụng làm việc với thiết bị video trên hệ nhúng ARM Linux, chúng ta sẽ viết một lớp chuyên biệt làm việc với thiết bị camera sử dụng các API V4L2. Dữ liệu ảnh thu nhận được sẽ được chuyển đổi sang cấu trúc ảnh của OpenCV và tiếp tục sử dụng với các hàm xử lý của thư viện này. Chương trình Qt minh họa phương pháp này xem trong file đính kèm.

QtCaptureV4L2.tar.gz (666k)	DCE- SOICT, 06:30 30-12-2013	v.2	
capturev4l2.c (6k)	DCE- SOICT, 06:57 27-12-2013	v.2	

Comments

You do not have permission to add comments.