TCP/IP 비디오 스트리밍

With Video4Linux & FFmpeg

VEDA 송현준

한화비전
Hanwha Vision

Contents

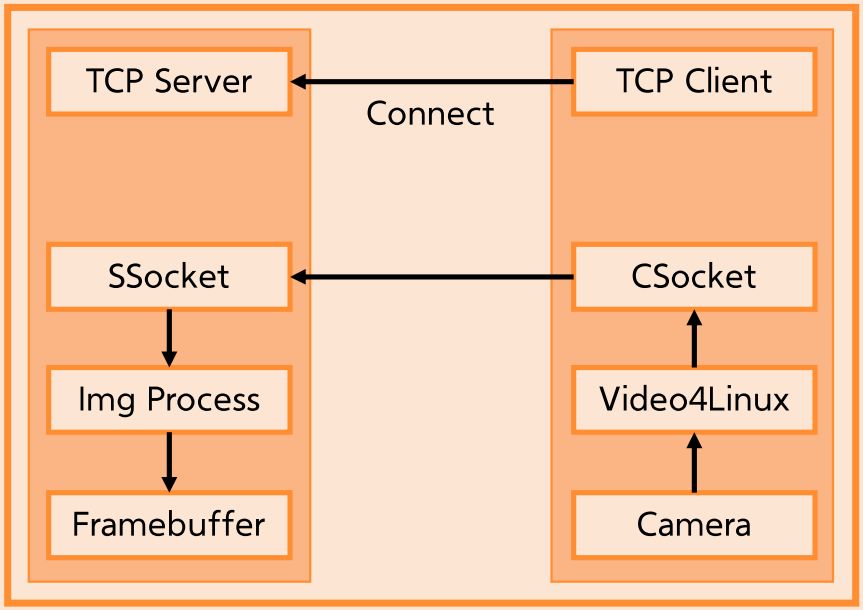
01

개요

01 – 개요

전체 구조

❖ 개요



- Server
  Client가 넘겨준 프레임을
  YUYV->RGB24->RGB565 변환을 거쳐
  프레임 버퍼에 Display.

- Client
  Video4Linux를 통해 Camera로부터 프레임을
  받아 Server로 전송

02

Video4Linux

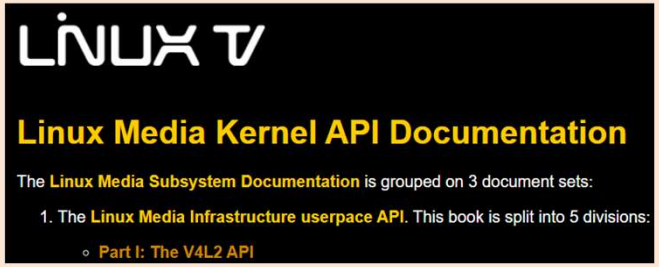
- Video4Linux
- Scan

02 – Video4Linux

Video4Linux

❖ **개요**

Video4Linux(V4L)은 Linux 시스템에서 실시간 비디오 캡처를 지원하기 위한 디바이스 드라이버와 API 모음이다.



❖ **프로젝트에서의 사용**

ioctl( )과 함께 사용하며, 디바이스에게 특정 명령을 수행하게 하거나, 디바이스에 대한 정보를 얻을 수 있다.

ioctl (fd, VIDIOC_REQBUFS, *v4l2_requestbuffers);	비디오 장치에 mmap I/O를 위한 공간 할당 요청
ioctl (fd, VIDIOC_QUERYBUF, *v4l2_buffer);	mmap에 사용할, 버퍼의 상세 정보 호출
ioctl (fd, VIDIOC_QUERYCAP, *v4l2_capability);	비디오 장치의 기능 조회 호출
ioctl (fd, VIDIOC_STREAMON/VIDIOC_STREAMOFF, *v4l2_buf_type);	스트림을 열고/닫아 버퍼 큐에서 버퍼 수신
ioctl (fd, VIDIOC_QBUF/VIDIOC_DQBUF, *v4l2_buffer);	버퍼 큐에서 버퍼 하나를 Deque/Queue

Video4Linux

❖ **비디오 장치에 비디오 포맷 적용**

```
struct v4l2_format fmt;
memset(&fmt, 0, sizeof(fmt));                    // fmt 구조체를 0으로 초기화
fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;          // 싱글 프레임 캡처 타입으로 설정
fmt.fmt.pix.width = WIDTH;                       // 캡처 해상도 너비
fmt.fmt.pix.height = HEIGHT;                     // 캡처 해상도 높이
fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV;     // 픽셀 포맷 설정 (YUYV)
fmt.fmt.pix.field = V4L2_FIELD_NONE;             // 필드 설정 (인터레이스가 아닌 프로그레시브 스캔을 사용)
ioctl(fd, VIDIOC_S_FMT, &fmt);                   // 설정한 포맷을 장치에 적용하기 위한 ioctl 호출
```

Scan

❖ Interaced Scan VS Progressive Scan

– Interaced Scan : 한 프레임은 짝수 줄, 다음 프레임은 홀수 줄을 송신해 대역폭을 절반으로 줄이는 방법.

– Progressive Scan : 전체 프레임을 한 번에 송신하는 방법.

03

YUV, BT.601

- YUV
- YUV422
- BT.601
- How process_image work

03 – YUV, BT.601

YUV

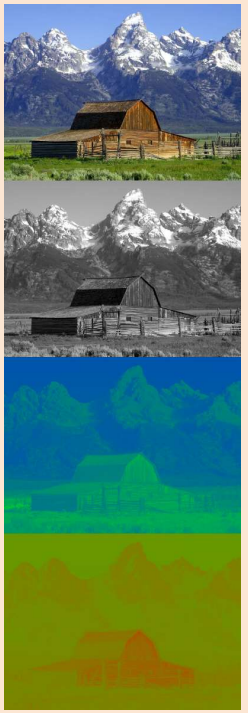
❖ **개요**

인간의 눈은 색상보다는 밝기에 민감. 이에 착안하여 색을 밝기에 해당하는 luma(Y) 성분과
2개의 색차 성분 U(B-Y), V(R-Y)로 색공간을 정의하는 색 인코딩 시스템.
YUV는 다양한 의미로 혼용. 오늘날엔 YCbCr을 사용하여 인코딩되는 파일 포맷 기술용으로 사용.

❖ **Chroma Subsampling**

인간의 눈은 색상보다는 밝기에 민감하여 색차에 대한 정보량을 줄여도 큰 위화감을 느끼지 못 함.
이에 기반에 색차에 대한 정보량을 줄이는 방법론들을 통칭하는 용어가 Chroma Subsampling.
e.g. YUV422, YUV411, YUV420 …

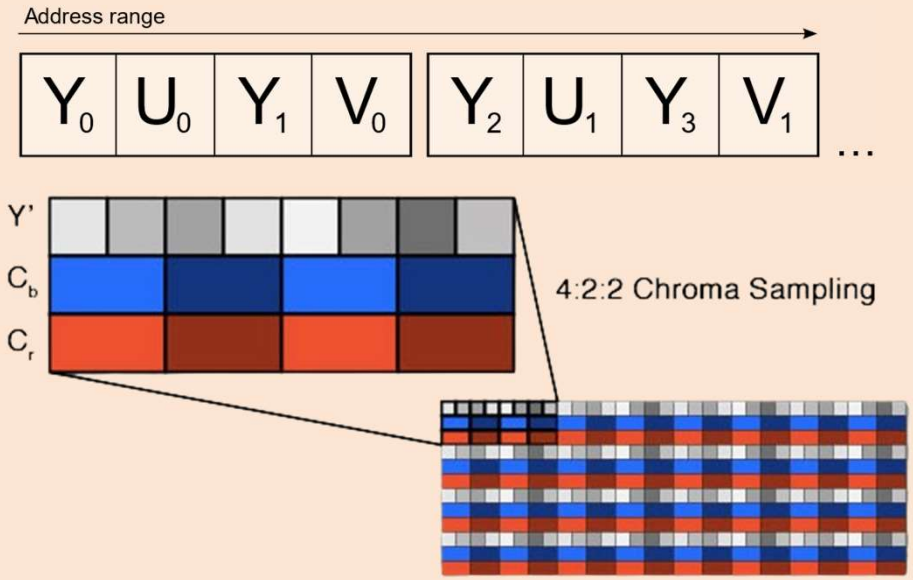
현재 사용하는 라즈베리 파이 카메라는 YUV422 – YUYV 포맷을 사용한다.



03 – YUV, BT.601

YUV422

❖ **컨테이너 및 픽셀 구조**



YUV422 중 YUYV or YUY2라고 불리는 포맷.
2개의 픽셀을 하나의 컨테이너로 묶는 구조.

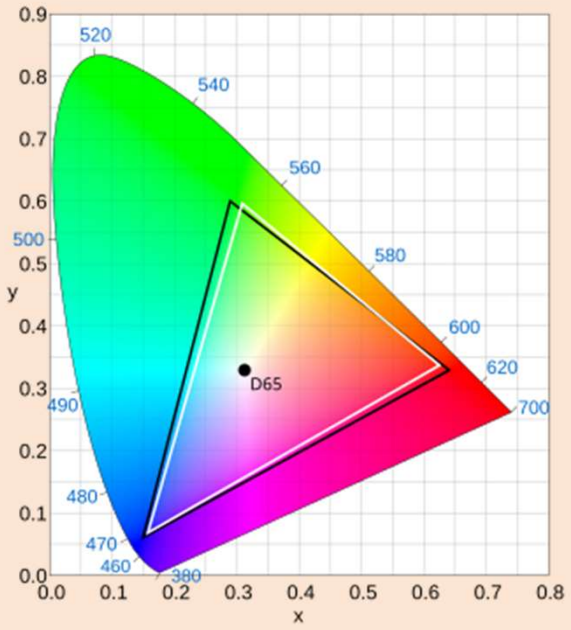
픽셀 2개에 대한 Cr, Cb 정보를 하나로 서브샘플링.
Luma 정보는 2개로 샘플링.

카메라의 YUYV 데이터를 프레임버퍼에
Display하기 위해선 RGB565로의 변환이 요구된다.

03 – YUV, BT.601

BT.601

❖ 개요



국제전기통신연합 ITU(International Telecommunication Union)에서 마련한 색공간 규격.

영상 포맷 형식이나 신호 포맷 규격 등이 정의되어 있다.

❖ BT.601

Luma : 8bit 샘플에서 Black을 16, White를 235로 정의.

Cb, Cr : 8bit 샘플에서 unsigned로 0~255의 값을 가지며 128이 Neutral 색차.

BT.601 문서를 기반으로 YUYV –> RGB24 –> RGB565로의 변환.

03 – YUV, BT.601

```
static void process_image(const unsigned char *in)
{
    int x, y, j;
    int y0, u, y1, v, r, g, b;
    unsigned short pixel;
    long location = 0;
    int istride = WIDTH * 2;

    for(y = 0; y < HEIGHT; ++y) {
        for(j = 0, x = 0; j < vinfo.xres * 2; j += 4, x += 2) {
            if(j>=WIDTH*2) {
                location++;
                location++;
                continue;
            }

            y0 = in[j] - 16;
            u = in[j + 1] - 128;
            y1 = in[j + 2] - 16;
            v = in[j + 3] - 128;

            r = clip((298 * y0 + 409 * v + 128) >> 8, 0, 255);
            g = clip((298 * y0 - 100 * u - 208 * v + 128) >> 8, 0, 255);
            b = clip((298 * y0 + 516 * u + 128) >> 8, 0, 255);
            pixel = ((r>>3)<<11)|((g>>2)<<5)|(b>>3);
            fbp[location++] = pixel;

            r = clip((298 * y1 + 409 * v + 128) >> 8, 0, 255);
            g = clip((298 * y1 - 100 * u - 208 * v + 128) >> 8, 0, 255);
            b = clip((298 * y1 + 516 * u + 128) >> 8, 0, 255);
            pixel = ((r>>3)<<11)|((g>>2)<<5)|(b>>3);
            fbp[location++] = pixel;
        }
        in += istride;
    }
}
```

How process_image work

❖ **BT.601 기반 YCbCr to RGB24**

$$R = 1.164(Y - 16) + 1.596(Cr - 128)$$

$$G = 1.164(Y - 16) - 0.813(Cr - 128) - 0.392(Cb - 128)$$

$$B = 1.164(Y - 16) + 2.017(Cb - 128)$$

❖ **핵심 Point**

1. YUYV 한 컨테이너는 2개의 픽셀의 정보를 가지고 있다.
2. YUYV 한 컨테이너의 크기는 4바이트.
3. 프레임버퍼 출력을 위한 RGB24 –> RGB565 포맷으로 변환.

04

FFmpeg

- FFmpeg
- H.264
- YUV420

04 – FFmpeg

FFmpeg

❖ **개요**

디지털 음성 스트림과 영상 스트림에 대해 다양한 종류의 형태로 기록하고 변환하는 컴퓨터 프로그램

여러 Free-Software와 오픈 소스 라이브러리로 구성됨.

❖ **핵심 라이브러리**

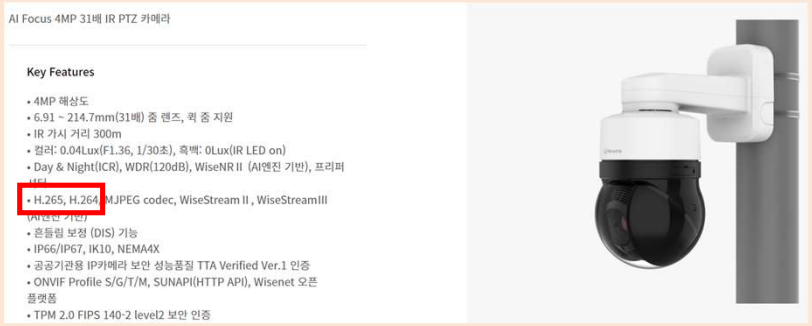
- libavcodec : 음성 및 영상 데이터의 인코딩과 디코딩.
- libavformat : 멀티미디어 파일 포맷을 읽고 쓰는데 사용.
- libavutils : 여러 유틸리티 함수 및 자료 구조.
- libswscale : 스케일링, 색 공간 변환 등을 수행.

H.264

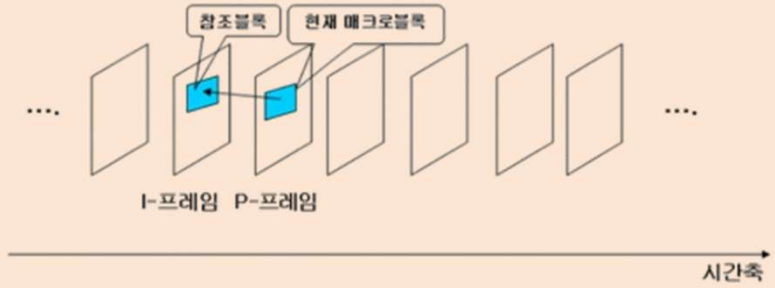
❖ **개요**

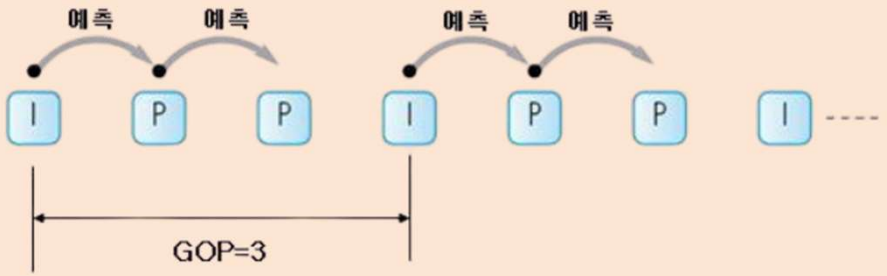
가장 일반적으로 사용되는 동영상 녹화, 압축, 배포를 위한 포맷.

움직임 보상(Motion compensation) 기반으로 영상을 압축한다.



❖ **움직임 보상**





H.264

❖ **In FFmpeg**

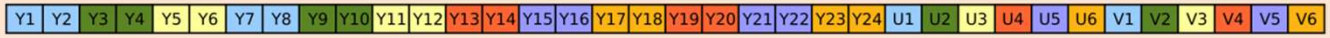
```
AVCodecContext *c = avcodec_alloc_context3(codec);
c->bit_rate = 1000000;                          // 비디오 품질과 파일 크기를 조정
c->width = WIDTH;
c->height = HEIGHT;
c->time_base = (AVRational){1, 25};             // 프레임 간 시간 간격 조정
c->framerate = (AVRational){25, 1};             // 1초 당 Display하는 프레임 개수
c->gop_size = 10;                               // 압축률과 지연 시간을 결정
c->max_b_frames = 0;                            // 실시간 스트리밍에서는 필요 X
c->pix_fmt = AV_PIX_FMT_YUV420P;                // H.264 압축을 위해 YUV420P 변환 필요
```

YUV420

❖ **컨테이너 및 픽셀 구조**

Single Frame YUV420:

Position in byte stream:



하나의 패킷에 Y 성분이 전부 나열된 후 U, V 성분이 나열된다.

픽셀 4개에 대한 Cr, Cb 정보를 하나로 서브샘플링.

❖ **L420**

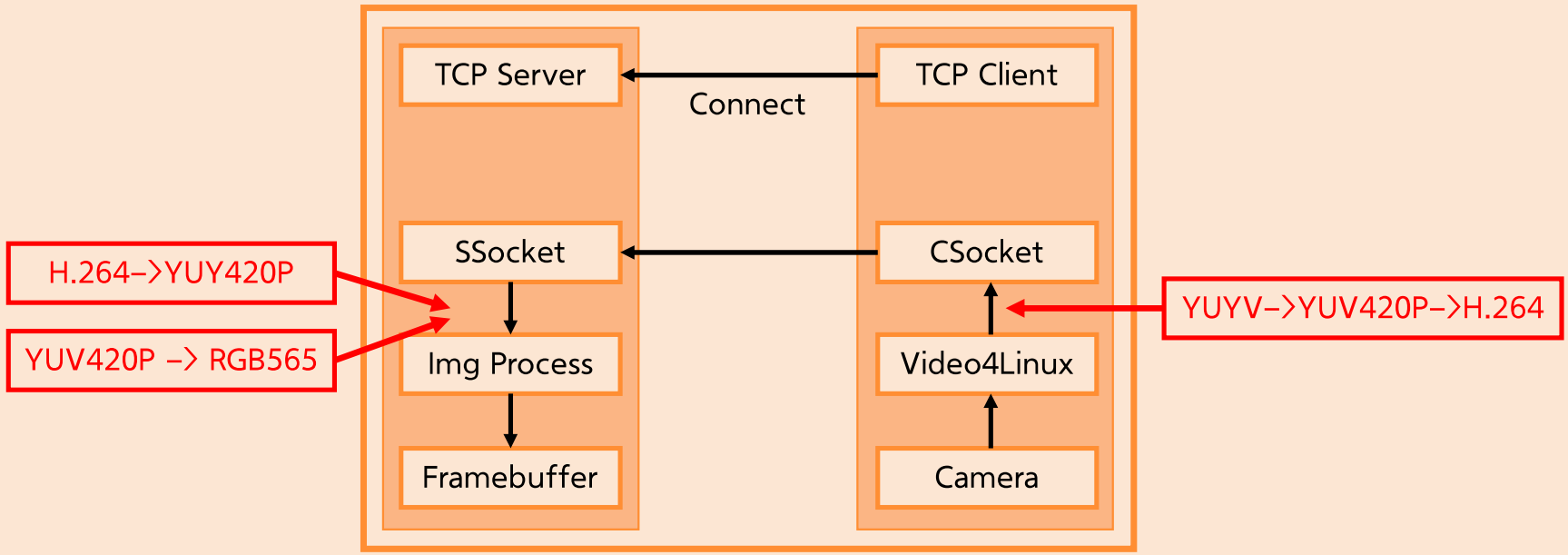
L420은 가장 널리 사용되는 YUV420 디자인.

안드로이드 카메라에서 널리 사용되는 NV12도 이 디자인을 따른다.

아래와 같이 패킷이 오면 2X2 픽셀 당 하나의 U, V값을 이용하여 색상을 표현한다.

How process_image work

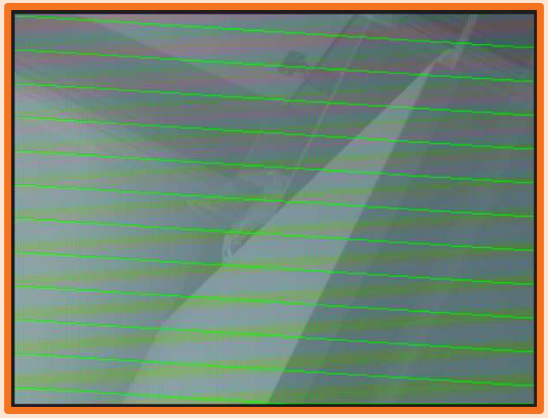
❖ 동작 구조



How process_image work

❖ What's problem?

❖ Expected Output



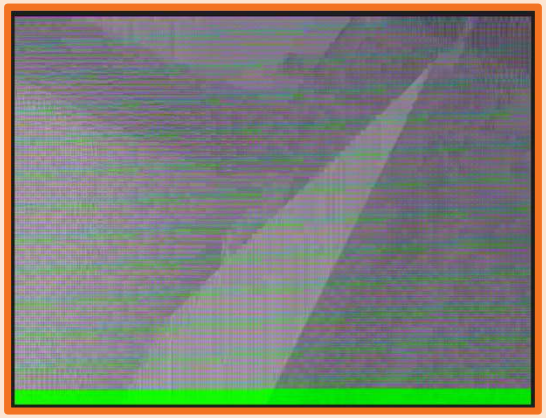
❖ Displayed Output

YUYV –> YUV420P로의 변환 과정에서 문제 발생.

How process_image work

❖ What's problem?

❖ Expected Output



❖ Displayed Output

디코딩 한 H.264의 YUV420P 프레임을 RGB565로 변환하는 과정에서 문제 발생.

How process_image work

❖ What's problem?

❖ Expected Output

❖ Displayed Output

YUV420P 프레임들을 H.264로 인코딩하는 과정에서 문제 발생.

How process_image work

❖ 시연 영상

어느 단계에서 문제가 생겼는지,

각 단계에서 디버깅은 어떻게 할 것인지,

발견한 문제를 어떻게 해결할 것인지,

전체적인 프로세스에 대한 지식과 이해가 필요하다.

❖ Processing…

- oRTP를 활용하여, rtp 프로토콜로 h264 데이터 송수신.
- New: 더블 버퍼링 구조로 개선하기.

EOF

들어 주셔서 감사합니다.

송현준

한화비전
Hanwha Vision