kim hanwoong url 수정함. f608625 7 days ago

1 contributor

247 lines (204 sloc) | 9.83 KB

# FFmpeg

FFmpeg는 음성파일, 영상파일을 변환하거나 기록하는 오픈소스 프로그램입니다. 이미지 시퀀스를 동영상을 생성하는 일, 화면을 캡쳐하는 작업도 할 수 있습니다.

우리가 원본파일을 이용해서 동영상 파일을 생성하는 과정은 원본의 데이터가 효율적으로 압축되어 재생될 수 있는 구조로 바뀐다는 것을 의미합니다. 그 과정에서 정보는 일부 손실됩니다. 따라서 FFmpeg를 이용해서 만들어진 동영상은 컬러 컨펌을 하는 상황에서 보통은 사용하지 않습니다.

- 레이아웃
- 시퀀스의 흐름체크
- 에니메이션 컨펌
- 일반적인 컬러 작업 컨펌

위에 나열한 작업에등에서 많이 사용합니다.

## FFmpeg 설치

모든 기능을 사용할 수 있는 ffmpeg를 컴파일 하는 과정은 굉장히 오래걸립니다. http://johnvansickle.com 사이트에는 이미 우리가 자주 사용하는 기능을 활성화 해서 미리 빌드된 ffmpeg를 다운로드 받을 수 있게 해놨습니다. 그 댓가로 비트코인, 페이팔등의 수단을 통해 기부 받아 사이트를 운영합니다. 일단 여러분이 지금 학생이라면 그냥 사용하세요. 나중에 이러한 유틸리티들로 여러분의 경제활동이 이루어진다면 한번쯤 고마워하며 오픈소스 프로젝트에 기부 해보세요.

```
$ cd ~
$ mkdir -p app/ffmpeg
$ cd app/ffmpeg
$ wget http://johnvansickle.com/ffmpeg/builds/ffmpeg-git-amd64-static.tar.xz
$ tar xpvf ffmpeg-git-amd64-static.tar.xz --strip 1
```

## 명령어의 구성

ffmpeg 설치가 끝나면 2개의 명령어가 생성되어 있습니다.

- ffmpeg : 영상, 음성을 변환하는 툴
- ffprobe : 데이터를 분석하는 툴

## 지원하는 포멧과 코덱을 알아보기

실무에서 자주 사용하게 될 코덱과 포멧을 지원하는지 리눅스 명령어 grep을 통해서 체크해보겠습니다.

```
$ ffmpeg -codecs
$ ffmpeg -codecs | grep prores
$ ffmpeg -codecs | grep dnxhd
$ ffmpeg -codecs | grep h264
$ ffmpeg -codecs | grep hevc
$ ffmpeg -codecs | grep vp9
$ ffmpeg -codecs | grep av1
```

```
$ ffmpeg -formats | grep mov
$ ffmpeg -formats | grep mp4
$ ffmpeg -formats | grep webm
$ ffmpeg -formats | grep ogg
```

## 도움말 보기

ffmpeg는 복잡한 프로그램입니다. 기본적인 도움말을 보는 방법입니다.

```
$ ffmpeg -h
```

예) 세부적으로 dpx Encoder의 옵션을 알아보는 명령어는 아래와 같습니다. `ffmpeg -codecs` 명령어를 타이핑하면 나오는 codeclist 값의 옵션을 볼 때 사용합니다.

```
$ ffmpeg -h encoder=dpx
```

## 사용법

mp4를 avi로 변환하기

```
$ ffmpeg -i input.mp4 output.avi
```

### jpg시퀀스를 H.264, fps 23.98 .mp4로 만들기

인수를 나열할 때 주의할 점은 프레임 레이트에 해당하는 `-r` 옵션이 인풋 소스에 해당하는 `-i` 옵션 앞에 있어야 한다는 점입니다. 그렇게 해야 input 소스에 대해서 frame rate가 적용되어서 아웃풋 되는 결과물의 프레임수 오류가 없어집니다.

```
$ ~/app/ffmpeg/ffmpeg -f image2 -start_number 100 -r 24 -i ~/examples/FOO_0010/FOO_0010.%4d.jpg -vcodec
```

실습 : 앞으로 ffmpeg 명령어를 자주 사용하게 됩니다. ffmpge alias를 설정합니다.

### 비율을 유지하면서 가로픽셀을 1280으로 만들기

```
$ ffmpeg -f image2 -start_number 100 -r 24 -i ~/examples/FOO_0010/FOO_0010.%4d.jpg -vframes 101 -vcodec
```

### 뉴크에서 아웃풋되는 mov와 최대한 비슷하게 만들어보기.(완벽히 같지 않습니다.)

뉴크에서 .h264로 렌더링하고 mediainfo로 해당 미디어를 분석해보면 bt709 컬러스페이스로 설정되어있는 것을 볼 수 있습니다. ffmpeg에서도 bt709 옵션을 사용하면 mov 생성시 bt709 컬러스페이스 설정이 됩니다.

```
$ ffmpeg -f image2 -start_number 100 -r 24 -i ~/examples/FOO_0010/FOO_0010.%4d.jpg -vframes 101 -vcodec
```

응용 : 실제로 mov는 각 프로그램마다 옵션의 차이가 있습니다. mediainfo 명령어를 이용하여 데이터를 분석하면서 ffmpeg 옵션을 찾는 노력이 필요합니다.

### input.mp4 영상 분석하기

ffprobe 명령어를 사용하면 미디어를 분석할 수 있습니다.

```
$ ffprobe -v quiet -print_format json -show_format -show_streams H264_1280x720_24fps.mov
```

출력된 결과물은 아래와 같습니다.

```
{
    "streams": [
        {
            "index": 0,
            "codec_name": "h264",
            "codec_long_name": "H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10",
            "profile": "High",
            "codec_type": "video",
            "codec_time_base": "1/48",
            "codec_tag_string": "avc1",
            "codec_tag": "0x31637661",
            "width": 1280,
            "height": 720,
            "coded_width": 1280,
            "coded_height": 720,
            "has_b_frames": 2,
            "sample_aspect_ratio": "1:1",
            "display_aspect_ratio": "16:9",
            "pix_fmt": "yuvj420p",
            "level": 31,
            "color_range": "pc",
            "color_space": "bt709",
            "color_transfer": "bt709",
            "color_primaries": "bt709",
            "chroma_location": "left",
            "refs": 1,
            "is_avc": "true",
            "nal_length_size": "4",
            "r_frame_rate": "24/1",
            "avg_frame_rate": "24/1",
            "time_base": "1/12288",
            "start_pts": 0,
            "start_time": "0.000000",
            "duration_ts": 51712,
            "duration": "4.208333",
            "bit_rate": "1001861",
            "bits_per_raw_sample": "8",
            "nb_frames": "101",
            "disposition": {
                "default": 1,
                "dub": 0,
                "original": 0,
                "comment": 0,
                "lyrics": 0,
                "karaoke": 0,
                "forced": 0,
                "hearing_impaired": 0,
                "visual_impaired": 0,
                "clean_effects": 0,
                "attached_pic": 0,
                "timed_thumbnails": 0
            },
            "tags": {
                "language": "eng",
                "handler_name": "VideoHandler",
                "encoder": "Lavc58.35.100 libx264"
            }
        }
    ],
    "format": {
        "filename": "H264_1280x720_24fps.mov",
        "nb_streams": 1,
        "nb_programs": 0,
        "format_name": "mov,mp4,m4a,3gp,3g2,mj2",
        "format_long_name": "QuickTime / MOV",
        "start_time": "0.000000",
        "duration": "4.209000",
        "size": "529054",
        "bit_rate": "1005567",
        "probe_score": 100,
```

```
        "tags": {
            "major_brand": "qt  ",
            "minor_version": "512",
            "compatible_brands": "qt  ",
            "encoder": "Lavf58.20.100"
        }
    }
}
```

자주 사용되는 테크닉. 같은 경로에 같은 이름의 json을 생성할 수 있습니다.

```
$ ffprobe -v quiet -print_format json -show_format -show_streams H264_1280x720_24fps.mov > H264_1280x720
```

## Burn-in

FFmpeg를 이용해서 동영상에 글씨를 넣는 방법입니다. 기존에 뉴크 또는 그래픽스 툴을 사용하지 않기 때문에 툴 라이센스를 사용하지 않고 동영상에 Burn-In을 할 수 있는 장점이 있습니다.


예제 소스를 받습니다. 이 예제소스 내부에는 jpg 시퀀스가 포함되어 있습니다.

```
$ cd ~
$ git clone http://github.com/cgiseminar/examples
```

간단하게 정보를 입력하고 Burn-in 하겠습니다.

```
$ ffmpeg -f image2 -start_number 100 -r 24 -i ~/examples/FOO_0010/FOO_0010.%4d.jpg  -vcodec libx264 -vf
```

> 참고 : drawtext를 사용하기 위해서는 ffmpeg를 컴파일할 때 configure 옵션에 --enable-libfreetype 옵션을 달아서 컴파일 한 FFmpeg를 사용해야 합니다. 위에서 우리가 설치한 FFmpeg는 이미 위 옵션이 활성화되어 컴파일된 FFmpeg 입니다.

## Burn-in에 사용되는 모노스페이스 폰트

슬레이트에 사용하는 폰트는 모노스페이스 폰트를 보통 사용합니다. 모노스페이스 폰트는 각 글자의 가로 길이가 같은 폰트이고 프레임 정보처럼 매프레임 글씨가 애니메이션되더라도 자간이 흔들리지 않는 특징이 있습니다.

추천폰트

- CentOS
  - /usr/share/fonts/liberation/LiberationMono-Regular.ttf

    - /usr/share/fonts/gnu-free/FreeMono.ttf
- macOS
    - /Library/Fonts/Courier New.ttf

**실습1**

자신이 원하는 형태의 Burn-In 이 되도록 스크립트를 작성해 봅시다.

실습의 예)

- https://video.stackexchange.com/questions/14924/ffmpeg-drawtext-clipping-to-a-bounding-box

## FFmpeg 컴파일 설치방법

직접 ffmpeg를 설치하는 문서입니다.

https://trac.ffmpeg.org/wiki/CompilationGuide/Centos

## 실 습

- CentOS설치이후 /home/$USER/app 폴더에 ffmpeg가 자동으로 설치되도록 스크립트를 작성합니다.
- 터미널을 열었을 때 ffmpeg가 인식될 수 있도록 ffmpeg, ffprobe를 알리아스 설정합니다.

## 레퍼런스

- ffmpeg와 비슷한 명령어 libav : https://www.libav.org
- ffmpeg와 libav의 분쟁 : http://klutzy.nanabi.org/blog/2012/11/16/ffmpeg-libav/
- https://trac.ffmpeg.org/wiki/Scaling
- https://ffmpeg.org/ffmpeg-codecs.html
- https://video.stackexchange.com/questions/16682/ffmpeg-white-padding-is-light-grey-but-not-white
- VFX를 위한 FFmpeg 사용예 : https://trac.ffmpeg.org/wiki/Encode/VFX