International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Multimedia Processing with FFMPEG

Karthick Kumaran Ayyalluseshagiri Viswanathan[1], **Karthik Poduval**[2]

[1]Member, IEEE

[2]Member, IEEE

Abstract: ***Often the multimedia engineers face the challenge of converting one format or standard of multimedia content to another format. This is because not all multimedia devices support all types of multimedia formats and thus there is a need to convert from one format to another. The main challenge that engineers face is the right tool to do this format conversion. In this paper we will cover various commands to convert one format to another format of various multimedia contents like images, videos, audio etc.***

Keywords: Multimedia, Color space, FFMPEG

1. Introduction

The process of converting one format of multimedia content to another format is called transcoding. There are various forms of multimedia content like image, video, audio, graphics etc., Multimedia engineers working in different operating systems like Linux, Windows, Mac OS, Android are looking for a cross-platform tool to perform this conversion. FFMPEG is a complete cross platform, open-source tool that can be used to record, convert and stream audio, video, voice and image.

1) FFMPEG

FFMPEG is a complete cross platform, open source tool that consists of a set of libraries for supporting various audio and video codecs and different image formats. It is also a command line tool for encoding and decoding audio, video and images. It is a free software project for processing multimedia data. Many open source players like VLC Media player are built with the FFMPEG libraries. The popular video streaming application YouTube uses FFMPEG in the backend for decoding video and audio streams. FFMPEG has a rich set of support for various popular image, audio and video codecs and so it is used in various transcoding applications. FFMPEG source can be downloaded from https://github.com/FFmpeg/FFmpeg. FFMPEG is developed under Linux but it can be compiled under most operating systems including Mac OS, Microsoft Windows, Android. For windows OS, it is preferred to download the pre-built executable from the FFMPEG site [5].

For the Linux operating system, we can either build the ffmpeg executable from source code or install the ffmpeg from the command line. For example on an Ubuntu Linux machine we can use the following command to install ffmpeg.

```
sudo apt-get install ffmpeg
```

FFMPEG also includes other tools like ffplay, ffprobe. ffplay is used as a simple media player and ffprobe is used to display media information.

2) Color Spaces

There are a lot of color spaces available like RGB, CMYK, HSV, YUV, CIELAB etc., The one that is widely used for video processing is YUV color space. YUV was invented when engineers wanted color television to use the same broadcasting transmitters that were used for black and white TVs. They needed a signal transmission method that was compatible with black and white TV being able to add color information. The luma component already existed as the black and white signal, they added the UV signal. Inputs to the image and video processing will be generally RAW or YUV files. Input videos are usually in the YUV 4:2:0 or NV12 format.

3) Codecs

A codec is a process that compresses and decompresses the multimedia data. There are different codecs available for images, videos and audio data. Popular image codecs are JPEG, Lossless JPEG, MJPEG, HEIF etc., Most widely used video codecs are MPEG-4, H.264/AVC, HEVC, H.263, AV-1, VC-1 etc., MP3, AAC are the most popular and widely used audio codecs.

4) Video capture with FFMPEG

Capturing or recording videos with FFMPEG has never been so easy on different OS.

On Linux and Windows, video can be captured using the following command respectively

```
ffmpeg -f video4linux2 -r 30 -s 640x480 -i /dev/video0 output.mp4
ffmpeg -f dshow -i video=”screen-capture-recorder” output.mp4
```

On Linux and Windows, if you need audio along with the video recording use the following commands respectively

```
ffmpeg -f video4linux2 -i /dev/video0 -f alsa -ac 2 -i hw:0
output.mp4
ffmpeg -f dshow -i video=”screen-capture-recorder” -f dshow -i
audio=”Microphone” output.mp4
```

5) Video processing with FFMPEG

This command lists all the supported pixel formats by FFMPEG

```
ffmpeg -pix_fmts
```

The below command resizes the input YUV video of resolution 1920x1080 in YUV 4:2:0 format to 1280x720 resolution in the same YUV 4:2:0 format

```
ffmpeg -s:v 1920x1080 -i input.yuv -vf scale=1280:720 -c:v
rawvideo -pix_fmt yuv420p output.yuv
```

The below command decodes the video in YUV 4:2:2 (YUYV) format
ffmpeg -i input.mp4 -pix_fmt yuyv422 output.yuv

The below command decodes the video in YUV 4:2:0 NV12 format
ffmpeg -i input.mp4 -pix_fmt nv12 output.yuv

The below command decodes the video file in YUV 4:2:2 format and scales to 1280x720
ffmpeg -i input.mp4 -pix_fmt yuyv422 -vf scale=1280:720 output.yuv

For decoding in other pixel formats we can change the pix_fmt to one of the supported pixel formats by ffmpeg. For example if we want to decode the video in YUV 4:2:0 planar format we can change the pix_fmt to yuv420p
ffmpeg -i input.mp4 -pix_fmt yuv420p output.yuv

For resizing to any arbitrary resolution change the width and height assigned to the parameter "scale" in the below command. The below command decodes the video file in YUV 4:2:0 planar format and scales to 640x480
ffmpeg -i input.mp4 -pix_fmt yuv420p -vf scale=640:480 output.yuv

To compress/encode the YUV file of resolution 1280x720 with a frame rate of 30 fps in H.264 video format at 5 Mbps
ffmpeg -s:v 1280x720 -i input.yuv -vcodec h264 -r 30 -pix_fmt yuv420p -b:v 5000k output.h264

To compress the video in MPEG-4 video format we can use the following command
ffmpeg -s:v 1280x720 -i input.yuv -vcodec mpeg4 -r 30 -pix_fmt yuv420p -b:v 5000k output.m4v

Where the parameters
- s:v represents input resolution
- i represents input file
- vcodec represents output codec type
- r represents frame rate of the output video
- pix_fmt represents the video pixel format
- b:v represents the bitrate of the output video

6) Image processing with FFMPEG

The following command decompresses the JPEG image file and stores it in YUV format
ffmpeg -i image.jpg -pix_fmt yuv420p image.yuv

The following command can be used to resize the JPEG image to 1280x720 resolution
ffmpeg -i image.jpg -vf scale=1280:720 output.jpg

To convert the JPEG image to a BMP format use the below command
ffmpeg -i image.jpg image.bmp

To convert a single YUV frame into a JPEG image
ffmpeg -s:v 1280x720 -pix_fmt nv12 -i input.yuv image.jpg

To convert each frame in the video file into individual images use the following command
ffmpeg -i input.mp4 %03d.jpg

To convert YUV frames into individual JPEG images use the below command
ffmpeg -s:v 1280x720 -pix_fmt nv12 -i input.yuv -q 1 %03d.jpg

The parameter -q can vary from 1 to 100, lower value produces high quality JPEG images.

Cropping an image can also be performed with the below command
ffmpeg -i inputfile -filter:v "crop=out_w:out_h:left:top" output file
ffmpeg -i input.yuv -filter:v "crop=1280:720:100:100" output.yuv

In order to stitch two images side-by-side
ffmpeg -i image1.jpg -i image2.jpg -filter_complex hstack output.jpg

ARGB888 can be converted using the following command.
ffmpeg -f rawview -pixel_format rgba -video_size 2592x1944 -i input.rgba output.jpg

RGB24/BGR24 can be converted using the following commands.
ffmpeg -f rawview -pixel_format rgb24 -video_size 2592x1944 -i input.rgb output.jpg
ffmpeg -f rawview -pixel_format bgr24 -video_size 2592x1944 -i input.rgb output.jpg

Camera sensors connected to the embedded processor using the MIPI interface often produce RAW images in 8 bits-per-pixel (bpp). When these images are stored in the file we need a raw image viewer to look at the captured contents. We can use the below command to convert the RAW grayscale camera frame to a JPEG image file
ffmpeg -s:v 2592x1944 -pix_fmt gray -i input.raw output.jpg

For RAW16 BAYER, the following command can be used.
ffmpeg -f rawvideo -s 2592x1944 -pix_fmt bayer_rggb16be -i input.raw output.jpg

7) Audio Processing with FFMPEG

FFMPEG can also be used to convert/transcode audio files compressed with MP3 codec to AAC using the following command
ffmpeg -i input.mp3 -acodec aac -ab 128k -strict experimental output.m4a

To decode the MP3 audio into uncompressed audio PCM samples in the WAV format use the following command
ffmpeg -i input.mp3 output.wav

To compress or encode the uncompressed raw PCM audio samples in WAV file to compressed MP3 format use the below command
ffmpeg -i input.wav -c:a mp3 -b:a 128k output.mp3

Mixing audio and video streams can also be performed with FFMPEG
ffmpeg -i input.h264 -i input.mp3 -c:v copy -c:a copy -strict experimental output.mp4

2. Conclusion

This paper provided details about various commands to help the multimedia engineers for encoding, decoding and processing different types of multimedia data. The list of commands provided in this paper is only a short list that are widely used and required for the multimedia developers and engineers. FFMPEG has a variety of libraries namely libavcodec, libavformat, libavutil, libavfilter, libavdevice, libswscale, libswresample etc., which can be used for much more complex multimedia processing.

References

[1] "FFMPEG" [Online]. Available: https://www.ffmpeg.org/
[2] https://www.linkedin.com/pulse/multimedia-processing-ffmpeg-karthick-kumaran-a-s-v?trk=public_profile_article_view
[3] https://github.com/FFmpeg/FFmpeg
[4] https://en.wikipedia.org/wiki/Y%E2%80%B2UV
[5] https://www.ffmpeg.org/download.html
[6] https://www.ffmpeg.org/