FFMPEG – Lab Sheet

Audiovisual Processing CMP-6026A

Dr. David Greenwood

November 8, 2021

Aims and Objectives

To introduce the command line tool FFMPEG, and provide some examples for processing video and audio, useful for the second coursework assignment.

FFMPEG

FFMPEG is very powerful and flexible, but the number of options available can make it somewhat daunting to use. I have asked for it to be installed on the lab machines, and I encourage you to install on your own computers too.

The documentation is here: https://ffmpeg.org/ffmpeg.html

There is a useful wiki here: https://trac.ffmpeg.org/wiki/Seeking on the concept of seeking to a particular time in a video. We will use the fast seeking method in this lab.

There is an example video file to do the exercises with: `lab2.mp4`. If you have your own video file, you can use it in the exercises.

I have put some clickable links to documentation throughout the exercises.

Exercise 1

Trim a video from some start time to some end time. Copy the video and audio codec to avoid recompression.

```
ffmpeg -ss 3.6 -i lab2.mp4 -t 4.5 -c copy ex1.mp4
```

Here, the options mean the following:

- -i the input file
- -ss specifies the start time, e.g. 00:01:23.000 or 83 (in seconds)
- -t specifies the duration of the clip.
- -c copy copies the video and audio bitstream from the input to the output file without re-encoding them.

Re-encode is the default behaviour.

```
ffmpeg -ss 3.6 -i lab2.mp4 -t 4.5 ex1a.mp4
```

You will notice this takes quite a bit more time to process.

The default encoder for MP4 is libx264 (H.264 video) or mpeg4 (MPEG-4 Part 2 video) depending on your ffmpeg build. See FFmpeg Wiki: H.264 Video Encoding Guide for more info.

Exercise 2

Crop an area of a video. The video retains the same duration, but is of new dimensions. Video is recompressed. Audio is retained and copied to the new video. We will use the short clip from Exercise 1 as the input.

```
ffmpeg -i ex1.mp4 -vf crop=400:400:680:450 -c:a copy ex2.mp4
```

Where the options are:

- `-vf` specifies a video filter, here we use `crop=out_w:out_h:x:y`
- out_w is the width of the output rectangle
- out_h is the height of the output rectangle
- x and y specify the top left corner of the output rectangle

crop documents here

Exercise 3

Convert a video to a series of images.

```
mkdir ex3
ffmpeg -i ex2.mp4 ex3/ex3-%03d.jpg
```

A sequence of image files don't have a framerate. If you want to undersample the video file, use `-r` before the input.

Exercise 4

Convert a series of images to a video. FFMPEG documents this process as a slideshow.

ffmpeg -framerate 24 -i ex3/ex3-%03d.jpg ex4.mp4

We use the same file pattern as in Exercise 3, only this time we are reading for input.

What do you notice about the output video? It is not as fast as the input? Why is this?

Exercise 5

Convert a series of images to a GIF animation. You might need this in your coursework presentation.

```
ffmpeg -f image2 \
-start_number 150 \
-framerate 50 \
-i ex3/ex3-%03d.jpg \
-frames:v 50 \
ex5.gif
```

- `start_number` specifies the first image to use - respecting the numbering of the files.
- The framerate of 50 is the maximum practical for the GIF specification.
- `frames:v` specifies the number of *output* frames to use.

Exercise 6

Draw a rectangle on a video. Again, this can be useful for presentation.

```
ffmpeg -i ex1.mp4 \
-vf drawbox=x=680:y=450:w=400:h=400:color=red@0.5\
-c:a copy \
ex6.mp4
```

We copied the audio as before, but must re-encode the video stream. Unfortunately, the rectangle dimension specification differ from the crop specification.

Conclusion

This set of exercises has introduced some of the possibilities for processing video and audio with FFMPEG. There are many more, and I encourage you to explore them.