# Preprocessing movies with FFmpeg

(For use with movie\_split\_balance version 0.1)

Prior to use in movie watching paradigms, the movies we use must be preprocessed to a standard size and volume. PsychoPy (which we use to run these experiments) doesn’t play well with MP4 files with sound and will often play movies with the audio and video components out of sync. To circumvent this, we provide PsychoPy with separate video (mp4) and audio (mp3) files, which will then play in sync (for some reason this works better with PsychoPy’s built-in movie player).

The bash script **movie\_split\_balance** uses FFmpeg to achieve both of these goals!

In order to run this script, you must first install FFmpeg. On a Mac, can do this using the Homebrew installer or by downloading a static file from the FFmpeg documentation site:

brew install ffmpeg

<https://ffmpeg.org/download.html>

Test your install by opening a command line (Terminal on Mac) and entering:

ffmpeg -version

Which should return something like:

ffmpeg version 6.1 Copyright (c) 2000-2023 the FFmpeg developers

built with clang version 16.0.6

With FFmpeg installed we can now preprocess our movies.

In the command line, navigate to the folder containing the **movie\_split\_balance.sh** file:

cd Google\ Drive/Shared\ drives/DSCLab/Precision\ Imaging\ Stimuli/Movie\_Preprocessing\_Scripts/

Verify that you are in the right folder by calling the script with **bash** and the **-v** (version) parameter:

bash movie\_split\_balance.sh -v

If you have navigated to the correct folder and the script is properly callable, it will return the version of the script you are running:

movie\_split\_balance version 0.1

Last tested with FFmpeg version 6.1

Processing a video is simple: call the script followed by the path to your video. For example, here I am processing a video that is saved to my Documents folder:

bash movie\_split\_balance.sh ~/Documents/ModernFamily\_ep1\_1\_720p.mp4

The script will first measure the loudness of the input video, in terms of *Loudness Units to Full Scale* (LUFS). LUFS is a standard loudness measure that is similar to dB, but accounts for variance in perceived loudness across the audio spectrum. The measurements are saved to a file named **<movie\_name>\_loudness.txt**. This step usually takes <30 seconds.

Next the audio is re-encoded and adjusted to a mean loudness of -**17 LUTS**. This is a standard that has been used in the Patel lab for other movie stimuli in the past. The resulting audio is saved as **<movie\_name>\_audio.mp3**. This step takes a minute or two depending on the length of the input movie.

Finally, the video portion of the movie is processed. The output dimensions are set to 1920x1080, and a black border is added to center the original movie (the borders are set such that the original movie is the same size, in pixels). The audio is also removed. The video-only portion is saved as **<movie\_name>\_video.mp4**.Note that some media players, including QuickTime, will not play an mp4 file without sound (though VLC player will). This step takes the longest, between 5 and 15 minutes for a ~25 minute video.

All of the outputs are saved in a folder named after the original file and within the same directory as the original file:

A screenshot of a computer

Description automatically generated

You can now use the separate mp3 and mp4 files in PsychoPy.

## Additional Parameters:

Script version:

bash movie\_script\_balance.sh -v

Script usage:

bash movie\_script\_balance.sh -h

Specify integrated LUTS value:

bash movie\_script\_balance.sh -I <LUTS\_Value> <movie\_name.mp4>

Note that LUTS must be a negative number between -70 and -5, with larger numbers resulting in louder output.

## How it works:

The bash script **movie\_split\_balance** runs a sequence of FFmpeg commands to measure and re-encode the audio and video components of the input movie. Here are some details about what is going on behind the scenes:

*Measuring Audio:*

FFmpeg can adjust the LUTS values of a movie “dynamically” by streaming the audio and adjusting volume as it goes. However, this can introduce inconsistencies in loudness over the course of the movie, especially in longer movies. It is better to give FFmpeg measurements of the loudness of the input movie before adjusting, so that we can adjust the volume of the entire audio recording equally. Thus, the first step is to measure the loudness over the entire video.

ffmpeg -i "$INPUTFILE" -af loudnorm=I="$LOUDNESS\_I":print\_format=summary -vn -nostats -f null - 2>"$OUTPUT\_DIR/$MEASUREMENTSFILE"

This initial command uses the [loudnorm](https://ffmpeg.org/ffmpeg-filters.html#loudnorm) filter to measure the LUTS values over the entire recording. By setting the *print\_format* parameter to *summary*, the command returns the measurements as text. “-f null” suppresses any audio file output and “-vn” suppresses any video encoding. The final “- 2> “ specifies that we want to save the text that is usually output to the command line to a file instead; this saves the measurements from loudnorm to a txt file.

The measurements we need are “Input Integrated”, “Input True Peak”, “Input LRA”, and “Input Threshold.” The bash script reads these from the txt file generated above. These are saved as variables and passed to the next FFmpeg command:

ffmpeg -y -i "$INPUTFILE" -af loudnorm=I="$LOUDNESS\_I":measured\_I="$INPUTINTEGRATED":measured\_LRA="$INPUTLRA":measured\_TP="$INPUTTRUEPEAK":measured\_thresh="$INPUTTHRESHOLD":linear=true -ar 48000 "$OUTPUT\_DIR/$AUDIOFILEOUT"

Note that the “-ar” command specifies that the output audio should be sampled at 48kHz. The audio file is now saved as an mp3 normalized to -17 LUTS.

Finally, the video stream is re-encoded with the black border added and the audio removed:

ffmpeg -y -i "$INPUTFILE" -filter\_complex "pad=1920:1080:(ow-iw)/2:(oh-ih)/2:black" -c:v libx264 -crf 23 -preset slow -an "$OUTPUT\_DIR/$VIDEOFILEOUT"

The [pad](https://ffmpeg.org/ffmpeg-filters.html#pad-1) command adds the black border such that the final video is 1920x1080. The “ow”, “iw”, “oh”, and “ih” represent the *output* video width, the *input* video width, and likewise for the height. They are used to center the video within the border. “-c:v libx264” specifies the video codec; [libx264](https://ffmpeg.org/ffmpeg-codecs.html#toc-libx264_002c-libx264rgb) is a standard encoder for MP4 files. “-crf 23” and “-preset slow” are arguments that specify how to compress the video.