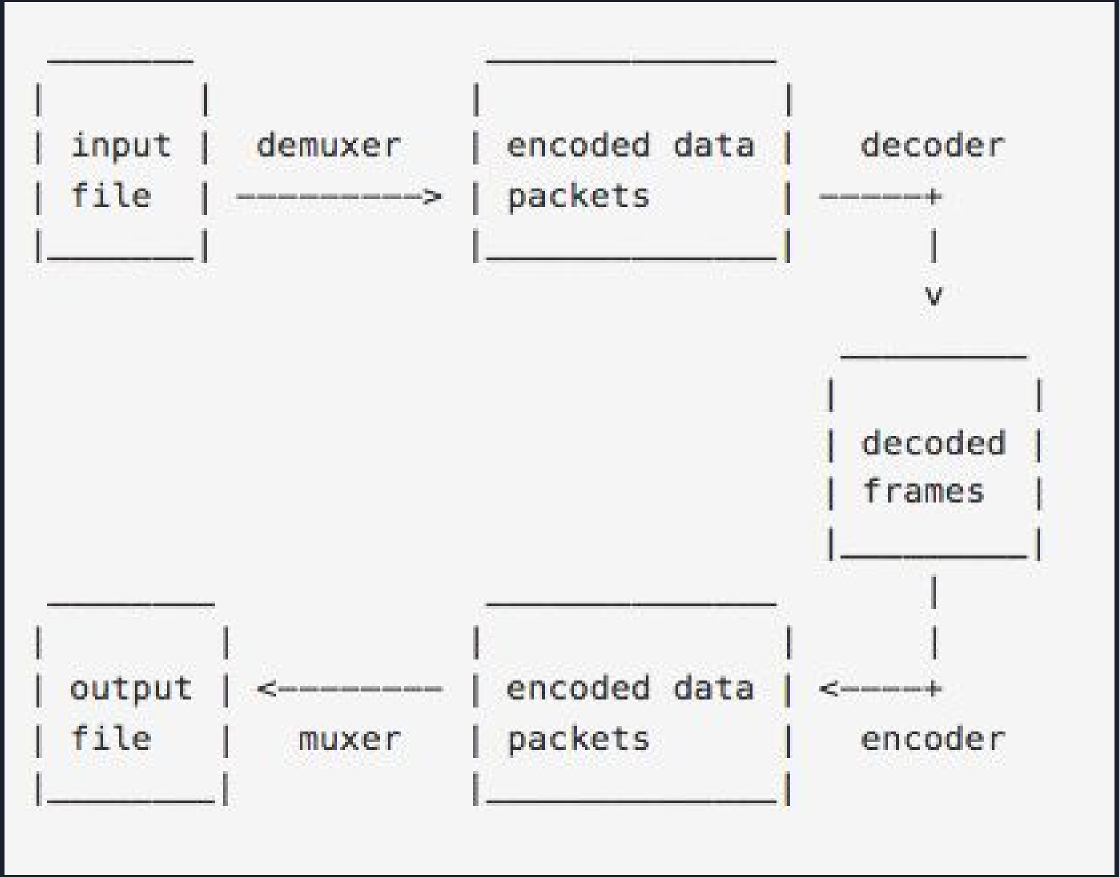
ffsake: further fluff with ffmpeg

Talking Tech, Spring 2017
2/5/17
Ethan Gates

What is ffmpeg?

- Leading open-source framework for processing and manipulating audiovisual material
- Aims to decode, encode, transcode, mux, demux, stream, filter and play “anything that humans and machines have created”
- A combination of tools and libraries
  - ffmpeg - media conversion
  - ffprobe - metadata analysis/extraction
  - ffplay - playback
    - libavutil - various utilities (e.g. math equations)
    - libavcodec - codec library
    - libavformat - format (wrapper/container) library
    - libavdevice - device library (drivers for common/open hardware)
    - libavfilter - filter library
    - libswscale - scaling library
    - libswresample - audio sampling library



```
 _______              ______________
|       |            |              |
| input |  demuxer   | encoded data |   decoder
| file  | ---------> | packets      | -----+
|_______|            |______________|      |
                                           v
                                       _________
                                      |         |
                                      | decoded |
                                      | frames  |
                                      |_________|
 ________             ______________       |
|        |           |              |      |
| output | <-------- | encoded data | <----+
| file   |   muxer   | packets      |   encoder
|________|           |______________|
```

Who’s using it?

- VLC Media Player
- Media Player Classic
- Handbrake
- Audacity
- DCP-o-matic
- YouTube
- Chromium/Google Chrome
- Facebook
- DaVinci Resolve
- Axle

But what can it do?

- **Make derivative copies**
- **Re-wrap files**
- **Edit files**
- **Generate checksums**
- **Create test files/streams**
- **Extract/insert metadata**
- **Generate informative (or just plain cool) audio/video effects**
- **Livestream video/screencapture**

* Thanks for this slide, Andrew Weaver

Installation

All about that configuration!

```
$ brew info ffmpeg
```

```
$ brew install ffmpeg --with-sdl2 --with-freetype --with-openjpeg
--with-x265 --with-rubberband --with-tesseract
```

Translation: with ffplay (playback), with added text/titles support, with JPEG 2000 codec, with H.265 (HEVC) codec, with extra audio filters, with OCR support

Tasks to Cover

I. Making a test file/signal

II. Encoding

  A. Transcoding

  B. Changing properties

III. Multiplexing/Muxing/Re-wrapping

IV. Editing

V. Fixity and Metadata

  A. framemd5

  B. Extracting metadata

VI. Filtering

  A. Deinterlacing

  B. Color Correction

  C. Temporal Difference

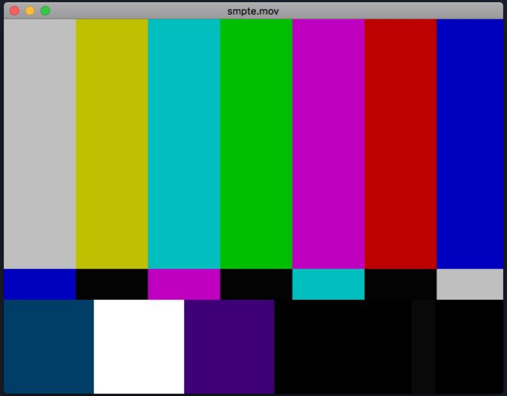
VII. Fuzzing

VIII. OCR

IX. Perceptual Hashing

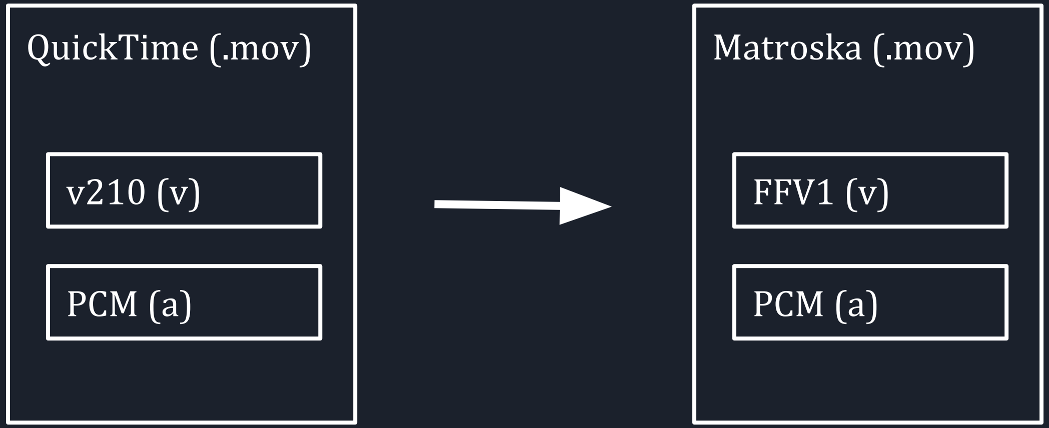
Generate a test file (SMPTE bars + 1 kHz tone)

```
$ ffmpeg -f lavfi -i
smptebars=size=720x486:rate=29.97 -f lavfi
-i “sine=frequency=1000:sample_rate=48000” -c:v
v210 -aspect 4:3 -c:a pcm_s24le -t 10 smpte.mov
```



Transcode and re-wrap to FFV1/MKV (preservation master)

```
$ ffmpeg -i smpte.mov -c:v ffv1 -level 3 -g 1
-slicecrc 1 -slices 16 -c:a copy smpte_ffv1.mkv
```


QuickTime (.mov)

v210 (v)

PCM (a)

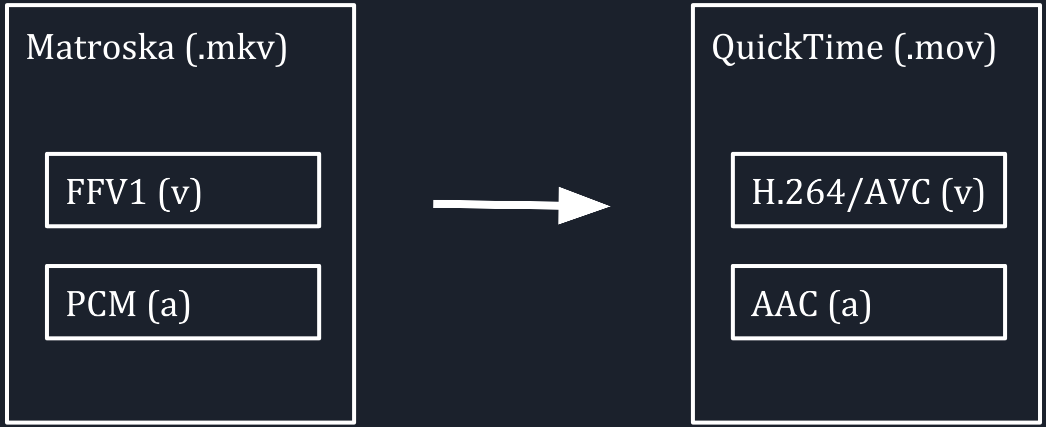
Matroska (.mov)

FFV1 (v)

PCM (a)

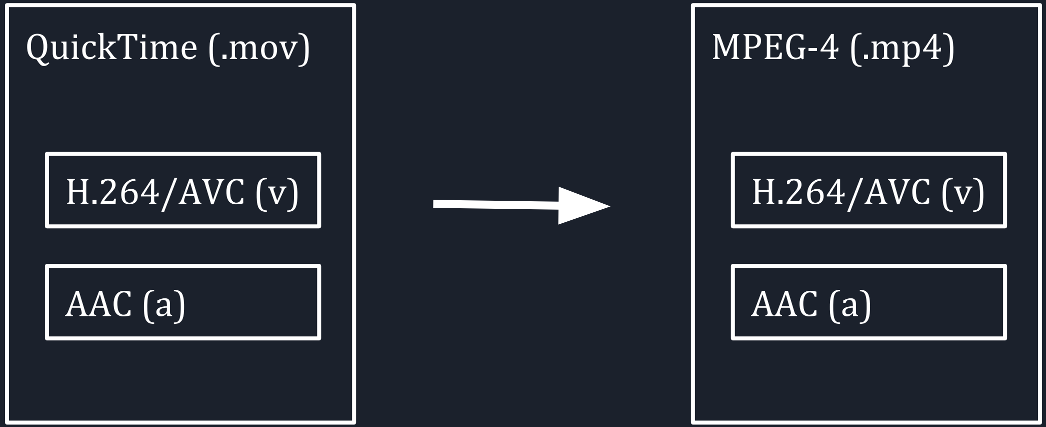
Transcode to H.264 (access copy)

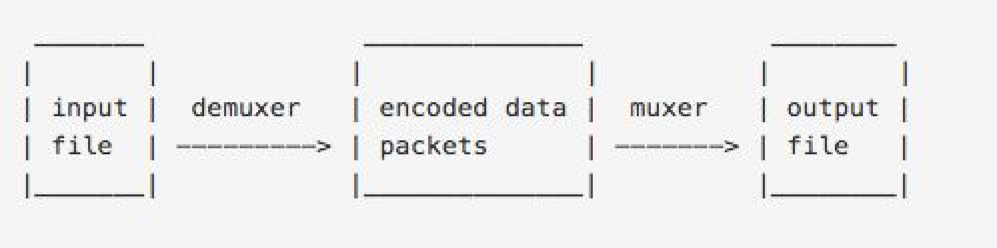
```
$ ffmpeg -i smpte_ffv1.mkv -c:v libx264 -pix_fmt
yuv420p -c:a aac smpte_h264.mov
```



Re-wrap to MP4

```
$ ffmpeg -i smpte_h264.mov -c:v copy -c:a copy
smpte_rewrap.mp4
```

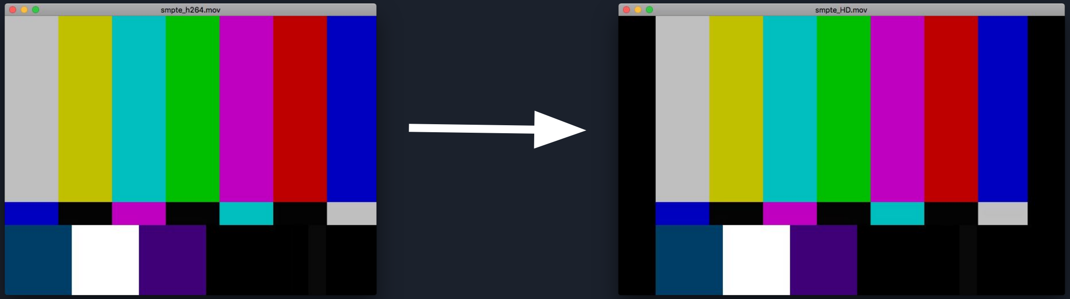




```
 _______              ______________            ________
|       |            |              |          |        |
| input |  demuxer   | encoded data |  muxer   | output |
| file  | ---------> | packets      | -------> | file   |
|_______|            |______________|          |________|
```

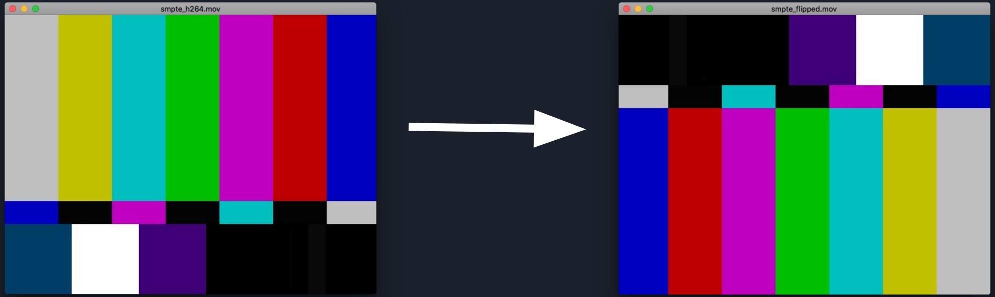
Pad to HD Resolution

```
$ ffmpeg -i smpte_h264.mov -filter:v
"pad=ih*16/9:ih:(ow-iw)/2:(oh-ih)/2" -c:a copy
smpte_HD.mov
```



Flip Image

```
$ ffmpeg -i smpte_h264.mov -filter:v “hflip,vflip”
-c:a copy smpte_flipped.mov
```



Trim Video

From beginning:

```
$ ffmpeg -i smpte_ffv1.mkv  -t 1 -c:v copy -c:a
copy smpte_trimmed.mkv
```

Or from 5 seconds into the video:

```
$ ffmpeg -i smpte_ffv1.mkv -ss 00:00:05 -t 1 -c:v
copy -c:a copy smpte_trimmed.mkv
```

Split Into Still Image Sequence + Audio Track

```
$ mkdir frames

$ ffmpeg -i smpte_trimmed.mkv frames/%03d.dpx

$ ffmpeg -i smpte_trimmed.mkv -c:a copy -vn
frames/audio.wav
```

Reconstruct into Video

```
$ cd frames

$ ffmpeg -f image2 -framerate 29.97 -i %03d.dpx -i
audio.wav -map 0:v -map 1:a -c:v ffv1 -level 3 -g
1 -slicecrc 1 -slices 16 -pix_fmt yuv422p10le -c:a
copy smpte_reconstructed.mkv
```

Automation: Presets

```
$ nano preservation.ffpreset
```

```
vcodec=ffv1
level=3
g=1
slicecrc=1
slices=16
acodec=pcm_s16le
ar=48000
```

```
$ ffmpeg -i input_file -fpre preservation.ffpreset
output_file
```

Fixity: Creating framemd5

```
$ ffmpeg -i smpte_ffv1.mkv -f framemd5 -an
video.framemd5
```

```
$ ffmpeg -i smpte_ffv1.mkv -af
“asetnsamples=n=48000” -f framemd5 -vn
audio.framemd5
```

```
$ sh check_[video/audio]framemd5.sh -i
smpte_ffv1.mkv -m [video/audio].framemd5
```

Fixity: Checking FFV1 Embedded CRC

```
$ ffmpeg -report -i smpte_ffv1.mkv -f null -
```

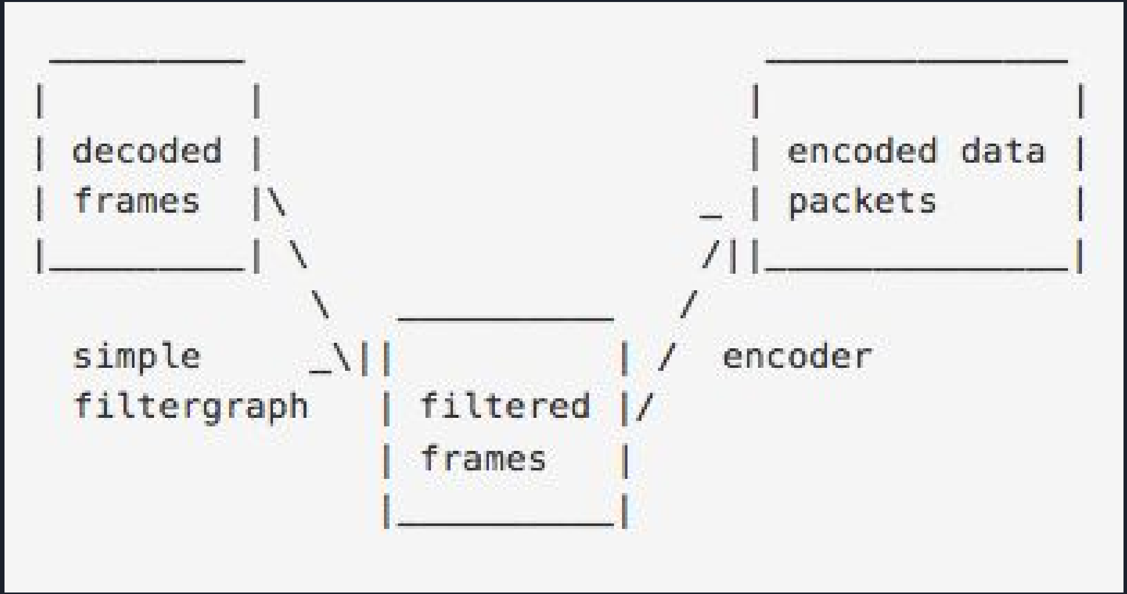
Metadata: ffprobe

```
$ ffprobe smpte_ffv1.mkv -show_format
-show_streams

$ ffprobe smpte_ffv1.mkv -show_format
-show_streams -print_format json

$ ffprobe smpte_ffv1.mkv -show_format
-show_streams -print_format xml > smpte_ffv1.json
```

Filtering



```
 _________                        ______________
|         |                      |              |
| decoded |                      | encoded data |
| frames  |\                   _ | packets      |
|_________| \                  /||______________|
             \   __________   /
  simple     _\||          | /  encoder
  filtergraph   | filtered |/
                | frames   |
                |__________|
```

Filtering: Deinterlacing

From an interlaced source format (e.g. digitized analog video, DV, older DVD/VOB files):

```
$ ffmpeg -i Stoney_visit.dv -c:v libx264 -vf
“yadif,format=yuv420p” Stoney_deinterlaced.mov
```

To reverse telecine pulldown:

```
$ ffmpeg -i Hot_Dogs.VOB -c:v libx264 -vf
“fieldmatch,yadif,decimate,format=yuv420p”
Hot_Dogs_deinterlaced.mov
```

Filtering: Color Correction

LUT = lookup table

- Binds pixel components to specific output values
- Can work with RGB or YUV colorspaces
- lutyuv requires a YUV-encoded input file, lutrgb works for RGB-encoded input

Back up

what the hell is YUV

A Super-Brief History of Color and “YUV”

- Television needed a system for displaying color that took up less bandwidth than additive RGB
- Invention of differential system: YUV
  - Y = luminance value
  - U = B - Y (blue value - luminance)
  - V = R - Y (red value - luminance)
- Basic idea refined into a number of different systems:
  - Y’IQ = NTSC television signals
  - Y’UV= PAL television signals
  - $Y'D_BD_R$ = SECAM television signals
  - $Y'P_BP_R$ = Analog “component” video
  - $Y'C_BC_R$ = Digital video *and* many still-image encodings

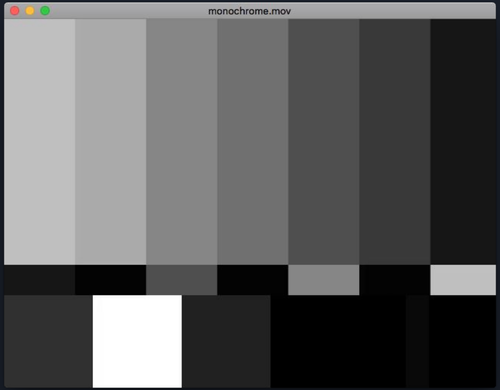
****What to keep in mind:

1) You may see *any* of these systems imprecisely referred to as “YUV”

2) In the digital age, many (including FFmpeg) use “YUV” to refer to a family of *file formats* that use $Y'C_BC_R$ encoding (as opposed to RGB or anything else)

Filtering: Color Correction

```
$ ffmpeg -i smpte_h264.mov -vf lutyuv=u=128:v=128
monochrome.mov
```



Filtering: Color Correction

```
$ ffplay smpte_h264.mov -vf lutyuv=y=negval

$ ffplay smpte_h264.mov -vf lutyuv=y=0

$ ffplay smpte_h264.mov -vf lutyuv=u=256:v=0
```

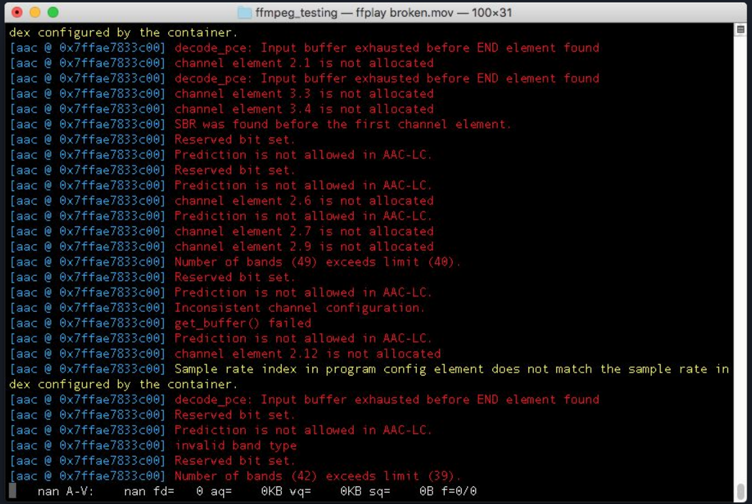
Or try your own combos!

Filtering: Fun!

```
$ ffmpeg -i hedgehog.mov -vf
tblend=all_mode=difference128 -f nut -c:v rawvideo
- | ffplay -
```

Fuzzing

```
$ ffmpeg -i input_file -bsf noise=1 -c:v copy -c:a copy broken.mov
```



```
dex configured by the container.
[aac @ 0x7ffae7833c00] decode_pce: Input buffer exhausted before END element found
[aac @ 0x7ffae7833c00] channel element 2.1 is not allocated
[aac @ 0x7ffae7833c00] decode_pce: Input buffer exhausted before END element found
[aac @ 0x7ffae7833c00] channel element 3.3 is not allocated
[aac @ 0x7ffae7833c00] channel element 3.4 is not allocated
[aac @ 0x7ffae7833c00] SBR was found before the first channel element.
[aac @ 0x7ffae7833c00] Reserved bit set.
[aac @ 0x7ffae7833c00] Prediction is not allowed in AAC-LC.
[aac @ 0x7ffae7833c00] Reserved bit set.
[aac @ 0x7ffae7833c00] Prediction is not allowed in AAC-LC.
[aac @ 0x7ffae7833c00] channel element 2.6 is not allocated
[aac @ 0x7ffae7833c00] Prediction is not allowed in AAC-LC.
[aac @ 0x7ffae7833c00] channel element 2.7 is not allocated
[aac @ 0x7ffae7833c00] channel element 2.9 is not allocated
[aac @ 0x7ffae7833c00] Number of bands (49) exceeds limit (40).
[aac @ 0x7ffae7833c00] Reserved bit set.
[aac @ 0x7ffae7833c00] Prediction is not allowed in AAC-LC.
[aac @ 0x7ffae7833c00] Inconsistent channel configuration.
[aac @ 0x7ffae7833c00] get_buffer() failed
[aac @ 0x7ffae7833c00] Prediction is not allowed in AAC-LC.
[aac @ 0x7ffae7833c00] channel element 2.12 is not allocated
[aac @ 0x7ffae7833c00] Sample rate index in program config element does not match the sample rate in
dex configured by the container.
[aac @ 0x7ffae7833c00] decode_pce: Input buffer exhausted before END element found
[aac @ 0x7ffae7833c00] Reserved bit set.
[aac @ 0x7ffae7833c00] Prediction is not allowed in AAC-LC.
[aac @ 0x7ffae7833c00] invalid band type
[aac @ 0x7ffae7833c00] Reserved bit set.
[aac @ 0x7ffae7833c00] Number of bands (42) exceeds limit (39).
    nan A-V:    nan fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

Optical Character Recognition (OCR)

Play calculated OCR data over video:

```
$ ffplay ocr.mov -vf
“ocr,drawtext=fontfile=/Library/Fonts/Andale
Mono.ttf:text=%{metadata\\\:lavfi.ocr.text}:
fontcolor=white”
```

Save OCR data to file:

```
$ ffprobe -show_entries frame_tags=lavfi.ocr.text
-f lavfi -i “movie=ocr.mov,ocr” > ocr.txt
```

Perceptual Hashing
(all the cool kids are doing it □)

```
$ ffmpeg -i hedgehog.mov -i Hot_Dogs.VOB -filter_complex
signature=detectmode=full:nb_inputs=2 -f null -

$ ffmpeg -i hedgehog.mov -i yozhik.mov -filter_complex
signature=detectmode=full:nb_inputs=2 -f null -

$ffmpeg -i fog.mov -i hedgehog.mov -filter_complex
signature=detectmode=full:nb_inputs=2 -f null -
```

Resources

- ffmprovisr
- Explain Shell
- Official ffmpeg documentation
- A trio of presentations by:
    - Ashley Blewer
    - Ben Turkus
    - Andrew Weaver
- "Learn FFmpeg libav the hard way", Leandro Moreira

  (it deceptively has good intro concepts too!)
- FFmpeg Basics by Frantisek Korbel ($9.50 for PDF)