Supporting niche formats and hardware in open source software and operating systems

Kieran Kunhya <kieran@kunhya.com> - @kierank_

Who am I?

- Work on Open Source Multimedia in both a personal and professional capacity: FFmpeg and x264 in particular.
- Also interested in reverse engineering of codecs and hardware
- Presentation in reverse chronological order for ease of understanding
- **This presentation given in a personal capacity**

Why FFmpeg?

- De-facto open source multimedia processing tool and library
  - Many web browsers implement complex specifications like HTML and CSS
  - Nothing comparable in multimedia
  - Basis of video players such as VLC, browsers (Chrome + Firefox), Smart TVs etc

- Written in C, not in newfangled language of the week
  - Widely supported across computer architectures
  - Likely tens of decades or more of support
  - Works on RPI, watch, other weird device

Standing on the Shoulders of Giants

- Kostya Shishkov and Paul Mahol
- Master codec reverse engineers
  - Willingness to teach and help (in their own special way…)
- Both reverse engineered dozens of codecs
  - Monumental impact on media playback

Why support niche formats

- Someday, someone might discover and want to play these formats.
  - An impact now or long into the future
  - Most of you in the room are that impact

- Quite a lot of multimedia low-hanging fruit is “done”, good way to stay familiar with concepts (entropy coding, DCT).
  - Something you can do over Easter/Christmas.
  - Also work for new students

This ticket

- There are so many variants of MPEG-4, why are these ones not supported?
- Why are they 4:2:2 and 4:4:4?
- Only 3 samples available, to this day no real-world samples found.

#4447 closed enhancement (fixed)

Opened 4 years ago
Closed 7 months ago

Support MPEG-4 Studio Profile

Reported by:	cehoyos	Owned by:	
Priority:	wish	Component:	avcodec
Version:	git-master	Keywords:	sstp
Cc:	kierank	Blocked By:	
Blocking:		Reproduced by developer:	no
Analyzed by developer:	no		

Description

http://ffmpeg.org/pipermail/libav-user/2015-February/007875.html
A user provided three MPEG-4 Simple Studio Profile files that are not supported by FFmpeg.

23.98p SR Lite (4:2:2 @ 220 Mbps)

```
$ ffmpeg -i A003C002_SR_Lite_23.98p.mxf
ffmpeg version N-71321-g6fd300a Copyright (c) 2000-2015 the FFmpeg developers
  built with gcc 4.7 (SUSE Linux)
  configuration: --enable-gpl
  libavutil      54. 22.100 / 54. 22.100
  libavcodec     56. 34.100 / 56. 34.100
  libavformat    56. 30.100 / 56. 30.100
  libavdevice    56.  4.100 / 56.  4.100
  libavfilter     5. 13.101 /  5. 13.101
  libswscale      3.  1.101 /  3.  1.101
  libswresample   1.  1.100 /  1.  1.100
```

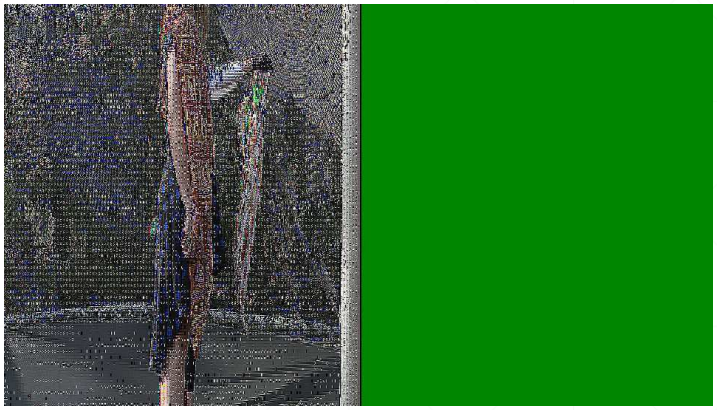
MPEG-4 Sstp (1)

- Spent a lot of time trawling the internet looking for recent copy of paid spec
  - Common in industry but a big problem for independent developers.
  - Eventually found it on some Chinese website

- Woah this is different
  - Up to 12-bit data, 4:2:2, 4:4:4, RGB – none supported in existing MPEG-4 code
  - > 16-bit coefficients (not supported in FFmpeg IDCT)
  - Crazy DPCM mode with vertical block scanning
- Only one program on Windows, no ability to extract raw data

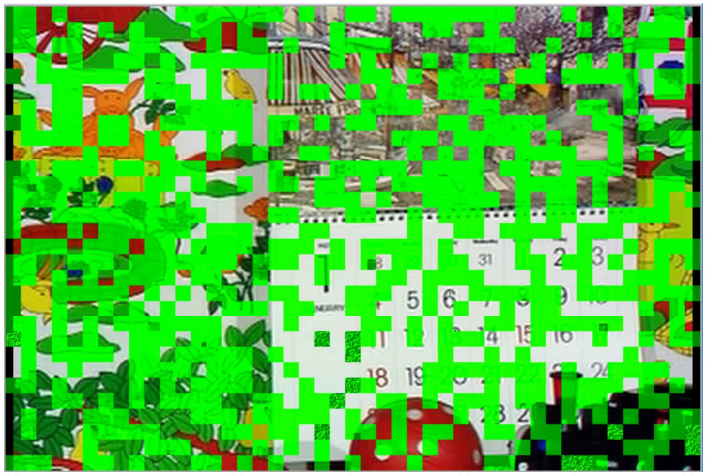
MPEG-4 Sstp (2)

- Entropy coding nice, either works or it doesn't
- Used float IDCT to get to working picture
- No reference decoder, tweaked picture until it looked ok.
- Took another year to get round to boring work of implementing templated integer IDCT.
- Also tedious work to hack-in 10/12-bit into FFmpeg, not ideal method.

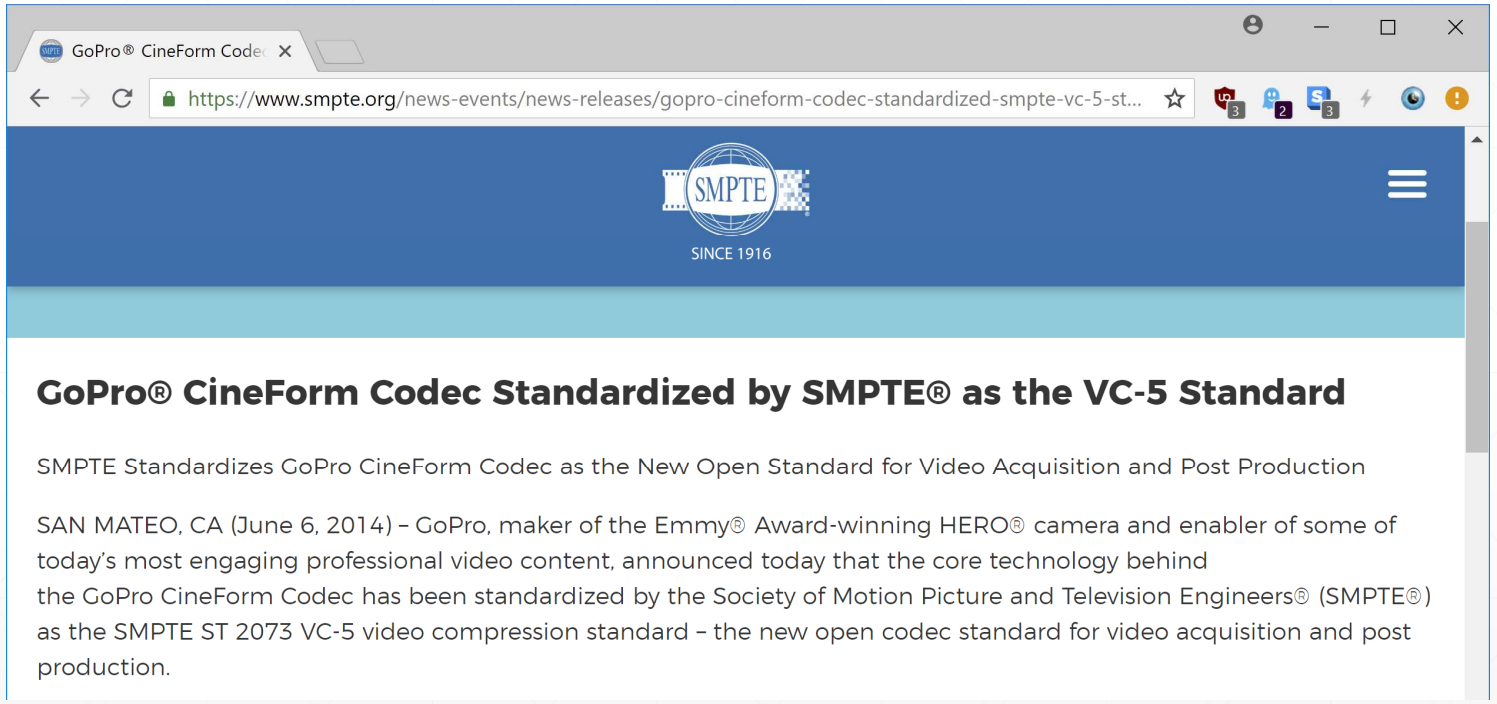


MPEG-4 Sstp (3)

- Was quite hard to verify DPCM blocks
- After a lot of digging online found conformance MPEG bitstreams, weird raw file format.
- Got to position where nearly all of image worked but some part looked weird.
- After even more digging found reference software. Typo found: -(x >> n) != ((-x) >> n)
- Might be some minor errors, probably IDCT precision problem, maybe > 32-bit intermediates?

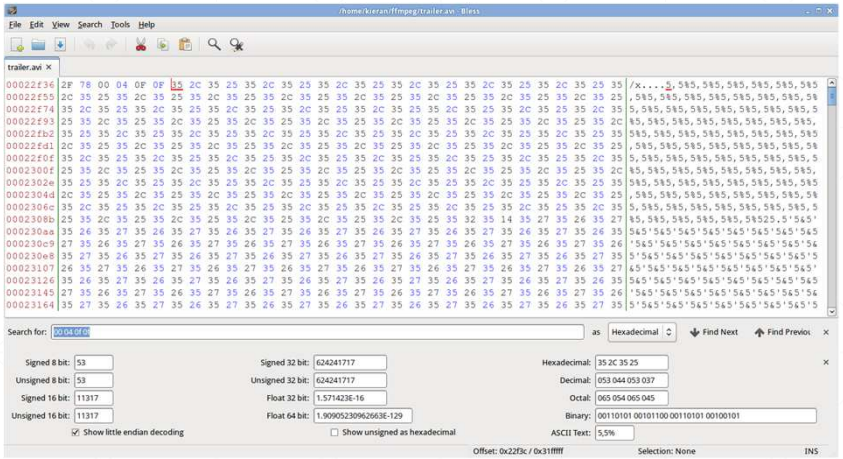


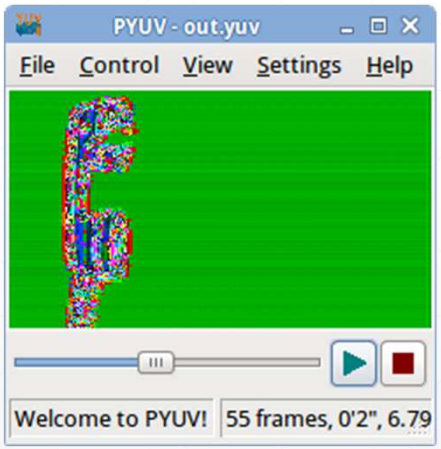
This press release



Cineform

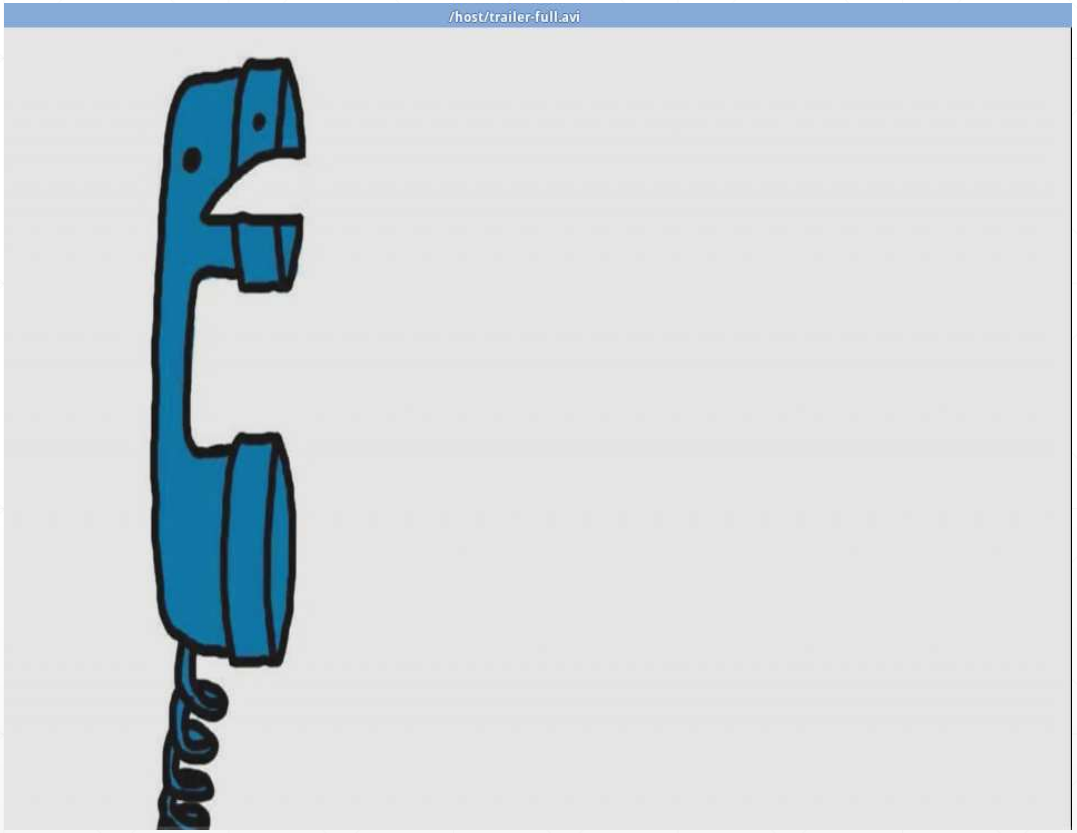
- Downloaded SMPTE VC-5 specification
  - Some things similar but some things in real world completely different
- Hints that lowpass coefficients were raw, by luck one sample had quite a lot of flat colours.
  - Helped identify region
- Lucky also this sample used the published codebook, though could have reverse engineered from binary decoder





Cineform (2)

- Continue reverse engineering tags, lots of people contributed samples.
  - Aligned coefficient layout
  - Simple and fast codec
  - Eventually get to working decoder
- Some samples had more complicated structures, “3D-transform” frames, interlaced, bayer.



Cineform (3)

- https://medium.com/@kierank_/reverse-engineering-the-gopro-cineform-codec-7411312bfe1c – 36k readers!
- Eventually led to Cineform being open-sourced by GoPro!
- Few missing pieces implemented by Google Summer of Code 2018 student
  - Not integrated into FFmpeg yet

But what about Physical Media files come on

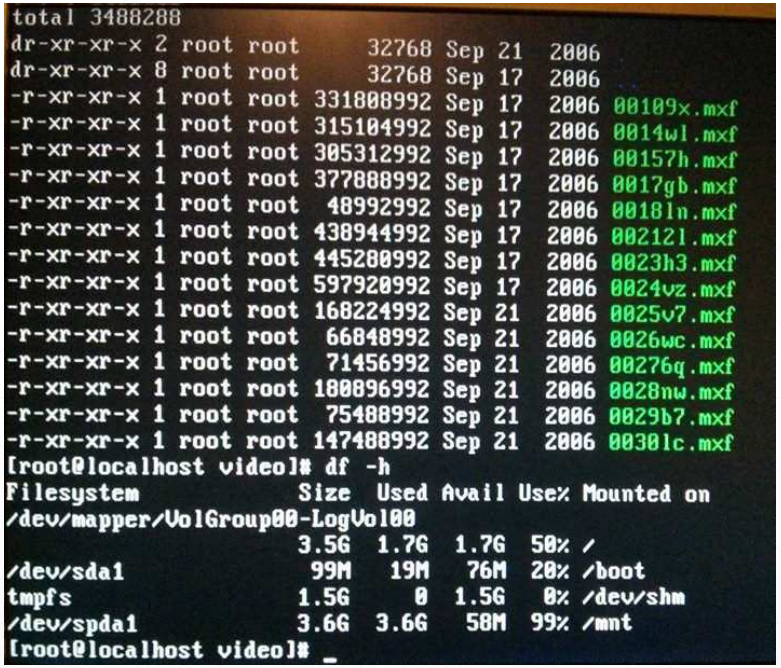
- One day someone will come across these and wonder how to read files
- Panasonic P2 and Sony SxS cards
  - PCI and PCI Express-based respectively
  - Both common and widely used, precursors to modern NVMe storage
- Closed source Windows and Mac drivers, or proprietary readers.
  - Likely won’t work in 100 years
  - Linux driver will last, nothing comparable hardware support

Panasonic P2 (1)

- PCMCIA based solid stage storage introduced in 2004
- Modern PCI Express backwards compatible with legacy PCI
- Bought cheap card off eBay.

Panasonic P2 (2)

- Request source code for Linux on camera
  - Dig through thousands of lines and find ancient Linux 2.4 driver.
- Go to parents house, find my high school PC from 2008 with legacy PCI slot.
  - Will it work without official hardware?
- A lot of messing with CentOS and wow, it worked!
- Might have newer cameras with more reasonable driver:
  - https://github.com/kierank/p2card



Sony SxS

- Also requested source code, no driver to be found ☹
- Instead looked into reverse engineering.
- Built rig to easily use card.
- Setup Windows XP QEMU to sniff memory mapped IO, how the driver is reading memory from the card.
  - https://hakzsam.wordpress.com/2015/02/21/471/

Sony SxS (2)

- At the time needed newest CPU with IOMMU
- Use “dd” to read block by block and see how the driver reads and writes. Change the block index, and continue
  - Very basic by modern NVMe standards
  - Request block, get an interrupt with buffer
- https://github.com/kierank/sxs-linux
  - No working DMA yet so very slow
  - Need time to get it into mainline Linux (Christmas?)

Conclusions

- Audio and Video codec reverse engineering is tending to completion
  - Codecs are being reverse engineered at a faster rate than they are created
  - This has remarkable historical consequences for media
    - Both the fact that users can “store” their content on sites like YouTube
    - Or that discovered media will be playable for decades

- That said there are still many proprietary storage formats out there
  - May not be easily possible to use commodity reader and reverse engineer