Every Solution is Wrong:
Normalizing Ambiguous, Broken, and Pants-on-Head Crazy Media
New phone, who dis?

• Senior Video Engineer @ Vimeo
  • Some things I’ve worked on:
    • Transcoding pipeline (pre- and post-chunking)
    • Edge stateless segmenting (DASH, CMAF, HLS, etc.)
    • On-the-fly image recompression
    • Captions stack
• Open source developer (FFmpeg, FFMS2, etc.)
• VideoLAN non-profit board member
• Professional Twitter Sh*tposter
Users Gonna User

• Comprehensive ingest guidelines followed by majority of users
  • Important to have; most users/clients will follow these if present
  • The users that don’t will send you massively varying media
Users Gonna User

• Comprehensive ingest guidelines followed by majority of users
  • Important to have; most users/clients will follow these if present
  • The users that don’t will send you massively varying media
• We don’t have the luxury of demanding they upload correct/perfect media
Users Gonna User

• Comprehensive ingest guidelines followed by majority of users
  • Important to have; most users/clients will follow these if present
  • The users that don’t will send you massively varying media
• We don’t have the luxury of demanding they upload correct/perfect media
• We need to be able to ingest this vast array of media as best possible
  • Must be consistent
  • Result must be widely playable while best maintaining the user’s intents
  • Anger the least amount of users
• Provide users with a easy to digest recommendations based our analysis
Garbage In, Sanitized Garbage Out

• Archival purists, please look away now!
Garbage In, Sanitized Garbage Out

• Archival purists, please look away now!
• We need to do analysis and run heuristics before transcoding to make informed choices:
  • Does this file have enough well distributed RAPs to efficiently chunk? Can we even seek?
  • Do we need to convert colorspace, and if so, to what? What about HDR?
  • Do we need to scale (SAR/DAR), and to what? Cropping?
  • Do we need to un-screw timestamps, and how? Is there concept of a frame or field rate?
  • Is the file interlaced? Telecined? Is it tagged as such?
  • Do we need to resample audio? Downmix? How?
  • How should we sync audio? Do we need to pad? Silence-fill?
  • Other misc stuff like Apple “Slow-mo”, spherical video, MVC, etc.
• More, more, more…
Pick Your Poison

• Before we can do **anything**, we need to select which streams we’re doing the thing to
  • If there’s a spec (e.g. iTunes deliverables), it’s easy
Pick Your Poison

• Before we can do **anything**, we need to select which streams we’re doing the thing to
  • If there’s a spec (e.g. iTunes deliverables), it’s easy
• Otherwise: Heuristics! Yay!
Pick Your Poison

• Before we can do **anything**, we need to select which streams we’re doing the thing to
  • If there’s a spec (e.g. iTunes deliverables), it’s easy
• Otherwise: Heuristics! Yay!
• Video:
  • Derive a per-stream score based on various factors
    • Is it marked as a thumbnail stream? Is it marked as default / on?
      • Some media such as slideshows **only** have a timed thumbnail stream
    • Is it (M)JPEG?
    • Total duration?
    • Bitrate? (taking codec into account!)
The document contains a section titled "Pick Your Poison" which outlines criteria for choosing audio streams. Here are the bullet points:

- **Audio:**
  - If downmixing, prefer the official downmixed version, if present, over downmixing ourselves.
  - Prefer streams with earlier start times.
  - If they all start at the same time, prefer longer durations.
  - If everything else fails, go by lowest index.
Indexing and you

• Every file / stream is indexed, info collected includes frame types, timestamps, etc.
  • The file is not decoded during this phase, only demuxed in memory
  • Can pass around and store info needed for analysis
  • Seek easily for containers that may not have indexes
Indexing and you

• Every file / stream is indexed, info collected includes frame types, timestamps, etc.
  • The file is not decoded during this phase, only demuxed in memory
  • Can pass around and store info needed for analysis
  • Seek easily for containers that may not have indexes
• Much faster than a full decode, especially for things like JPEG2000
• Essentially building a packet-to-frame mapping
  • Harder than it sounds due to things like:
    • alt-refs – VP8 requires packet inspection, VP9 may, AV1 doesn’t
    • NVOPs – Need to be skipped.
    • PAFF – Need to handle field packets
    • Virtual timelines (edit lists, ordered chapters)
Go Chunk Yourself

- We don’t transcode to a mezzanine format before chunked encoding
Go Chunk Yourself

• We don’t transcode to a mezzanine format before chunked encoding
• Can analyse how reasonable seeking will be based on frame types
  • We don’t want to try seeking in files with one or no keyframes (no block-level ref analysis)
  • Something like a weighted quantile in the time domain based on keyframe distances (90th percentile)
    • How likely are we to end up with a RAP within N frames of a seek?
• No block / ref level analysis; deemed not worth it
Go Chunk Yourself

- We don’t transcode to a mezzanine format before chunked encoding
- Can analyse how reasonable seeking will be based on frame types
  - We don’t want to try seeking in files with one or no keyframes (no block-level ref analysis)
  - Something like a weighted quantile in the time domain based on keyframe distances (90th percentile)
    - How likely are we to end up with a RAP within N frames of a seek?
  - No block / ref level analysis; deemed not worth it
- Preferable to chunk based on shot (or rather, cost effective) boundaries, if available
  - Note that source RAP placement is not necessarily indicative of a Good™ chunk boundary
Scaling: How to Make Everyone (with Janky Media) Hate You

• Goal is to normalize for playback on as many devices as possible
Scaling: How to Make Everyone (with Janky Media) Hate You

• Goal is to normalize for playback on as many devices as possible
• Everything is resized to be mod 2 (no bitstream cropping) with a 1:1 SAR, rotation applied
Scaling: How to Make Everyone (with Janky Media) Hate You

• Goal is to normalize for playback on as many devices as possible
• Everything is resized to be mod 2 (no bitstream cropping) with a 1:1 SAR, rotation applied
• Resampling method depends on if we are upscaling (e.g. SAR compensation) or downscaling or both
  • Also depends on how we intend to compress
Scaling: How to Make Everyone (with Janky Media) Hate You

• Goal is to normalize for playback on as many devices as possible
• Everything is resized to be mod 2 (no bitstream cropping) with a 1:1 SAR, rotation applied
• Resampling method depends on if we are upscaling (e.g. SAR compensation) or downscaling or both
  • Also depends on how we intend to compress
• Some poor encoders pad odd resolution 4:2:2 or 4:2:0 files with grey or green, but don’t set crop params
  • Detect and remove this as an edge case
Scaling: How to Make Everyone (with Janky Media) Hate You

- Goal is to normalize for playback on as many devices as possible
- Everything is resized to be mod 2 (no bitstream cropping) with a 1:1 SAR, rotation applied
- Resampling method depends on if we are upscaling (e.g. SAR compensation) or downscaling or both
  - Also depends on how we intend to compress
- Some poor encoders pad odd resolution 4:2:2 or 4:2:0 files with grey or green, but don’t set crop params
  - Detect and remove this as an edge case
- Some decoders will output impossible things like non-mod 4 4:1:0
  - Need to special case (different decoders do different things) and pad/crop as appropriate
Fields and Friends

- Cringe warning: We deinterlace or apply inverse telecine
• Cringe warning: We deinterlace or apply inverse telecine
• Correctly tagged files are great
  • Except a non-trivial portion of these tags are missing or just wrong
Fields and Friends

- Cringe warning: We deinterlace or apply inverse telecine
- Correctly tagged files are great
  - Except a non-trivial portion of these tags are missing or just wrong
- Try to detect interlaced or telecined content based on frame content
  - Try to detect temporally distinct fields by how different even/odd lines are
    - Running sum spatially for field detection
    - Running sums temporally for previous, current, and future for added field order detection
  - Pattern detection for telecined content
    - e.g. CCNNC (reset at RAPs) for 3:2 pulldown
Fields and Friends

• Cringe warning: We deinterlace or apply inverse telecine
• Correctly tagged files are great
  • Except a non-trivial portion of these tags are missing or just wrong
• Try to detect interlaced or telecined content based on frame content
  • Try to detect temporally distinct fields by how different even/odd lines are
    • Running sum spatially for field detection
    • Running sums temporally for previous, current, and future for added field order detection
  • Pattern detection for telecined content
    • e.g. CCNNC (reset at RAPs) for 3:2 pulldown
• Try and detect “fake” 50i and handle appropriately
Colors\textit{spaces}  

- More cringe warnings!  
- Almost every non-HDR device requires BT.709 or SMPTE170M matrix/transfer/primaries  
  - Try and convert based off tags if possible, try and fudge it otherwise (devices \textcolor{red}{need} color info)  
  - Conversions need to be gamma-correct (swscale is terrible, use zimg\textsuperscript{1} instead)  
  - Last resort guessing based on various identifying characteristics (PAL/NTSC matters!)
More cringe warnings!

Almost every non-HDR device requires BT.709 or SMPTE170M matrix/transfer/primaries
  • Try and convert based off tags if possible, try and fudge it otherwise (devices need color info)
  • Conversions need to be gamma-correct (swscale is terrible, use zimg\(^1\) instead)
  • Last resort guessing based on various identifying characteristics (PAL/NTSC matters!)

10-bit input (not necessarily HDR!) is dithered accordingly with a random dither algorithm

HDR input is ingested no matter what it is, although we only output HDR10-style media
  • Would have liked to use HLG, but Apple and Dolby exist
  • Need to make SDR versions, of course
    • Nominal peak luminance detection (zimg plays nice here)
    • In-house tonemapping like libplacebo’s Möbius transform-based algorithm\(^2\)
Colors

- Need to choose the “correct” colorspace metadata to use
  - Different containers have different precedence for bitstream vs container color information
Colors

- Need to choose the “correct” colorspace metadata to use
  - Different containers have different precedence for bitstream vs container color information
- ICC profiles may be shipped with some codecs or containers
  - Extracted from e.g. MJPEG frames
  - ISOBMFF colr boxes may contain ICC profiles… in theory

Derek Buitenhuus @daemon404 · 22 Aug 2017
Anyone have a mov/mp4 with a colr atom with the 'prof' parameter type (embedded ICC profile) I can have? (@kieranjol or @dericed maybe?).

Charles Poynton, PhD @momaku · 22 Aug 2017
Aaaaargghhh I expect & hope you fail to find any such thing in the wild! Interested if you do. Expect all readers to ignore the profile.
Timestamps: Literally Just Trash Fire

- There are so many ways that timestamps can be screwed up. Too many to enumerate.
  - See my talk from Demuxed 2017 for some details³
There are so many ways that timestamps can be screwed up. Too many to enumerate.

- See my talk from Demuxed 2017 for some details\(^3\)
- Check if the file has a concept of a frame/field rate, use it if we can (can’t on e.g. AVI with NVOPs)
There are so many ways that timestamps can be screwed up. Too many to enumerate.
  - See my talk from Demuxed 2017 for some details
- Check if the file has a concept of a frame/field rate, use it if we can (can’t on e.g. AVI with NVOPs)
- We always deliver CFR content to clients/devices (come at me, bro), so need to choose a “good” rate
  - Quicktime bug-friendly, precision reduced to fit in e.g. H.264 or container fields
There are so many ways that timestamps can be screwed up. Too many to enumerate.
  
  - See my talk from Demuxed 2017 for some details\(^3\)
  
Check if the file has a concept of a frame/field rate, use it if we can (can’t on e.g. AVI with NVOPs)

We always deliver CFR content to clients/devices (come at me, bro), so need to choose a “good” rate
  
  - Quicktime bug-friendly, precision reduced to fit in e.g. H.264 or container fields

Our indexing from earlier provides us with all the timestamp info for easy analysis
  
  - Analysis on DTS/PTS and frame durations (all done in arbitrary precision rational arithmetic)
    
    - Take into account allowed positive and negative timestamp discontinuities
    
    - Smooth extreme outliers (usually corruption) based on some metrics, e.g. stddev
    
    - If the file is an amalgamation of multiple CFR segments, choose the “best” rate
    
    - If it’s “true” VFR, try and fudge a “good” middle ground rate
Virtual timelines need to be taken into account

- Applied at presentation level, so we don’t care until we need to adjust the rate during transcode
- Adjust timestamps based on these timelines
- Some frames needed to prime the decoder when repeating or seeking due to frame reordering
timestamps: literally just trash fire

- Virtual timelines need to be taken into account
  - Applied at presentation level, so we don’t care until we need to adjust the rate during transcode
  - Adjust timestamps based on these timelines
  - Some frames needed to prime the decoder when repeating or seeking due to frame reordering
- Container durations can’t be trusted
  - Except when we deem they can
    - Sanity threshold of audio/video codec sample durations vs container
  - Calculate our own durations based off coded samples when in doubt
    - Both used streams must be taken into account for padding
    - Partial uploads are incredibly common
Timestamps: Literally Just Trash Fire

• Virtual timelines need to be taken into account
  • Applied at presentation level, so we don’t care until we need to adjust the rate during transcode
  • Adjust timestamps based on these timelines
  • Some frames needed to prime the decoder when repeating or seeking due to frame reordering
• Container durations can’t be trusted
  • Except when we deem they can
    • Sanity threshold of audio/video codec sample durations vs container
  • Calculate our own durations based off coded samples when in doubt
    • Both used streams must be taken into account for padding
    • Partial uploads are incredibly common
• Make sure to take into account all things that can modify rates (e.g. mdia rate, trak rate, stts, edts)
It just sounds, like, warmer, man…

• Downmix when needed for various devices
  • If the format inherently has info for this (e.g. AC3), use it
  • If the format can be positively identified as e.g. an iTunes deliverable, mix as per the spec
It just sounds, like, warmer, man…

- Downmix when needed for various devices
  - If the format inherently has info for this (e.g. AC3), use it
  - If the format can be positively identified as e.g. an iTunes deliverable, mix as per the spec
- Silence fill gaps if needed (e.g. the format requires it)
  - Figure out when gaps are meant to be silence, and when they’re meant to be repeated samples
It just sounds, like, warmer, man…

• Downmix when needed for various devices
  • If the format inherently has info for this (e.g. AC3), use it
  • If the format can be positively identified as e.g. an iTunes deliverable, mix as per the spec
• Silence fill gaps if needed (e.g. the format requires it)
  • Figure out when gaps are meant to be silence, and when they’re meant to be repeated samples
• Handle “approximate” timestamps that some formats and encoders output if need be
  • Some formats don’t have a separate time base for audio/video
  • Some encoders oscillate, e.g. cyclical 1023/1025 sample durations for LC-AAC (1024)
It just sounds, like, warmer, man…

• Downmix when needed for various devices
  • If the format inherently has info for this (e.g. AC3), use it
  • If the format can be positively identified as e.g. an iTunes deliverable, mix as per the spec
• Silence fill gaps if needed (e.g. the format requires it)
  • Figure out when gaps are meant to be silence, and when they’re meant to be repeated samples
• Handle “approximate” timestamps that some formats and encoders output if need be
  • Some formats don’t have a separate time base for audio/video
  • Some encoders oscillate, e.g. cyclical 1023/1025 sample durations for LC-AAC (1024)
• Resample when needed to a known-good set of sampling rates
It just sounds, like, warmer, man…

- Cannot look at audio in isolation
  - Need the video to interpret audio timestamps correctly
Captions

- We ingest a lot of formats, normalize, and only output VTT
  - Thankfully no 608… yet
Craptons

• We ingest a lot of formats, normalize, and only output VTT
  • Thankfully no 608… yet
• Guessing the rate for SCC files based on the highest timecode frame value
Craptions

- We ingest a lot of formats, normalize, and only output VTT
  - Thankfully no 608… yet
- Guessing the rate for SCC files based on the highest timecode frame value
- In ancient times, a decision was made to detect the text encoding ourselves and convert to UTF-8
  - This is as awful as it sounds
  - Before applying to probe encoding, try to manually decode the file as valid UTF-8
    - If it can be decoded as valid UTF-8, it is extremely unlikely to be anything else
Craptons

• We ingest a lot of formats, normalize, and only output VTT
  • Thankfully no 608… yet
• guessing the rate for SCC files based on the highest timecode frame value
• In ancient times, a decision was made to detect the text encoding ourselves and convert to UTF-8
  • This is as awful as it sounds
  • Before applying to probe encoding, try to manually decode the file as valid UTF-8
    • If it can be decoded as valid UTF-8, it is extremely unlikely to be anything else
• Sanitize any HTML/JS/CSS/etc. from files, except valid SRT/VTT/etc. tags
• Try to accept all sorts of mangled files created by prominent software
Crapions

• We ingest a lot of formats, normalize, and only output VTT
  • Thankfully no 608… yet
• Guessing the rate for SCC files based on the highest timecode frame value
• In ancient times, a decision was made to detect the text encoding ourselves and convert to UTF-8
  • This is as awful as it sounds
  • Before applying to probe encoding, try to manually decode the file as valid UTF-8
    • If it can be decoded as valid UTF-8, it is extremely unlikely to be anything else
• Sanitize any HTML/JS/CSS/etc. from files, except valid SRT/VTT/etc. tags
• Try to accept all sorts of mangled files created by prominent software
• Detect common mistakes like pasting captions into Word and upload…

Derek Buijtenhuis @daemon404 · Oct 3
Today: Implementing .RTF detection in our captions stack. Because users.
But wait, there’s more!

- Lots of stuff that was cool once upon a time
  - Spherical video, equirectangular video, 3D, ambisonics, etc.
  - Apple “Slow-mo” (based on make tag and media rate)
But wait, there’s more!

- Lots of stuff that was cool once upon a time
  - Spherical video, equirectangular video, 3D, ambisonics, etc
  - Apple “Slow-mo” (based on make tag and media rate)
- The endless amount of objectively amazing stuff in ISOBMFF
  - The never-ending dumpster fire that is edit list support in libavformat
  - The super-amazing 90s choice to code affine matrices directly in the container
  - Displaying multiple streams at once, overlayed with alpha
But wait, there’s more!

- Lots of stuff that was cool once upon a time
  - Spherical video, equirectangular video, 3D, ambisonics, etc
  - Apple “Slow-mo” (based on make tag and media rate)
- The endless amount of objectively amazing stuff in ISOBMFF
  - The never-ending dumpster fire that is edit list support in libavformat
  - The super-amazing 90s choice to code affine matrices directly in the container
  - Displaying multiple streams at once, overlayed with alpha
- Gracefully handle mid-stream parameter changes
  - Nearly all video and audio traits are per-frame
But wait, there’s more!

• Lots of stuff that was cool once upon a time
  • Spherical video, equirectangular video, 3D, ambisonics, etc
  • Apple “Slow-mo” (based on make tag and media rate)
• The endless amount of objectively amazing stuff in ISOBMFF
  • The never-ending dumpster fire that is edit list support in libavformat
  • The super-amazing 90s choice to code affine matrices directly in the container
  • Displaying multiple streams at once, overlayed with alpha
• Gracefully handle mid-stream parameter changes
  • Nearly all video and audio traits are per-frame
• Working around plain old bugs
  • Old Quicktime’s H.264 decoder being unable to handle QPs less than 5
Links / References


[2] Info on libplacebo's Mōbius algorithm:
• https://github.com/mpv-player/mpv/commit/d8a3b10f45eb10fb34ce9da3a9a76e3bd8644e3d
• https://vimeo.com/album/5461208/video/293434018

Questions? Heckling? “Not a question, but…”?