Frame Interpolation and Motion Blur for Film Production and Presentation

2013 GTC Conference, San Jose

Keith Slavin, isovideo LLC
(slides 20 to 22 by Chad Fogg)

➢

What we have today

- 24 frames/sec is too low to avoid judder on fast moving camera pans with detail
- To avoid judder (a perception of uneven motion), low frame rates require some combination of:
  - low light levels,
  - low camera motion,
  - out-of-focus backgrounds, or
  - motion blur (larger shutter angles).
- Existing deblur algorithms assume global camera motion and are heavily iterative (slow)
- No algorithms known today will undo local motion blur – nor (with noise and Cramer-Rao bounds analysis) should we assume they will ever exist!
- Films made today don't look good on the best display systems of today – definitely not near limits of human perception, and not “future proof”
- Europe is also subjected to a 4% speedup from 24 -> 25, requiring damaging audio pitch conversion for some viewers

What we have available Today

Motion-Compensated Frame Rate Conversion

- Motion vector quality from 24 to higher frame rates is strongly scene dependent
- Obtaining high quality motion vectors from occlusion and revelation is an open problem
- Motion aliasing is common in repetitive man-made objects and wagon-wheels
- Frame-to-frame motion where an object does not overlap itself is very problematic
- Up-converted 24 looks smooth-and-blurry, with effective shutter angles >> 360°

And the Consequences Are...

- Most people end up viewing movies with duplicated frames, and many with additional 3:2 judder on fixed 60HZ refresh displays.

This talk is about judder, frame rates, blur, motion estimation, interlacing & deinterlacing, and also their effects on compression.

Judder Perception

For a display refresh frequency and refresh period F_R,$=1/T_R$ and a video frame-rate F_v, we expect each video frame to be presented an average of n times, given by:

$$\frac{F_R}{F_v}=\frac{T_v}{T_R}=n=a\,p+b\,q$$

Where p,q represent the two nearest distinct integer repeat rates, given by

$$p=\lfloor n\rfloor,\text{and }\; q=p+1$$

and a,b are the relative weightings, such that

$$a+b=1$$

This gives:

$$a=1+p-n\,\text{and }\; b=n-p$$

The repeat values have associated presentation times T_p= p T_R and T_q= q T_R,
where $T_p < T_q$. We associate an overall weighted judder score for these time periods:

$$Judder=J_{Tp}\,a+J_{Tq}\,b$$

Where J_T is a judder measure as a function of presentation time. We expect that:

$$J_{Tp}<J_{Tq}$$

Experiment involved:
1) using our Legato MCFRC to convert to many frame-rates, no added blur
2) using refresh rates that are an exact multiple of frame rates.

120Hz: 0.0083, 0.0166, 0.0250, 0.0333, 0.0417, 0.0500, 0.0583, 0.0666

The perceptual judder function was found empirically to be a sigmoid function of the frame presentation time J_T :

$$J_T = \frac{1}{q+1}\text{, where } q = e^{(T_{center} - T)/T_{gain}}$$

where for “crowd-run” clip at viewing distance 1.5 x picture height:

$T_{center} \approx 42$ ms (about 24Hz)

$T_{gain} \approx 6.25$ ms (about 160Hz)

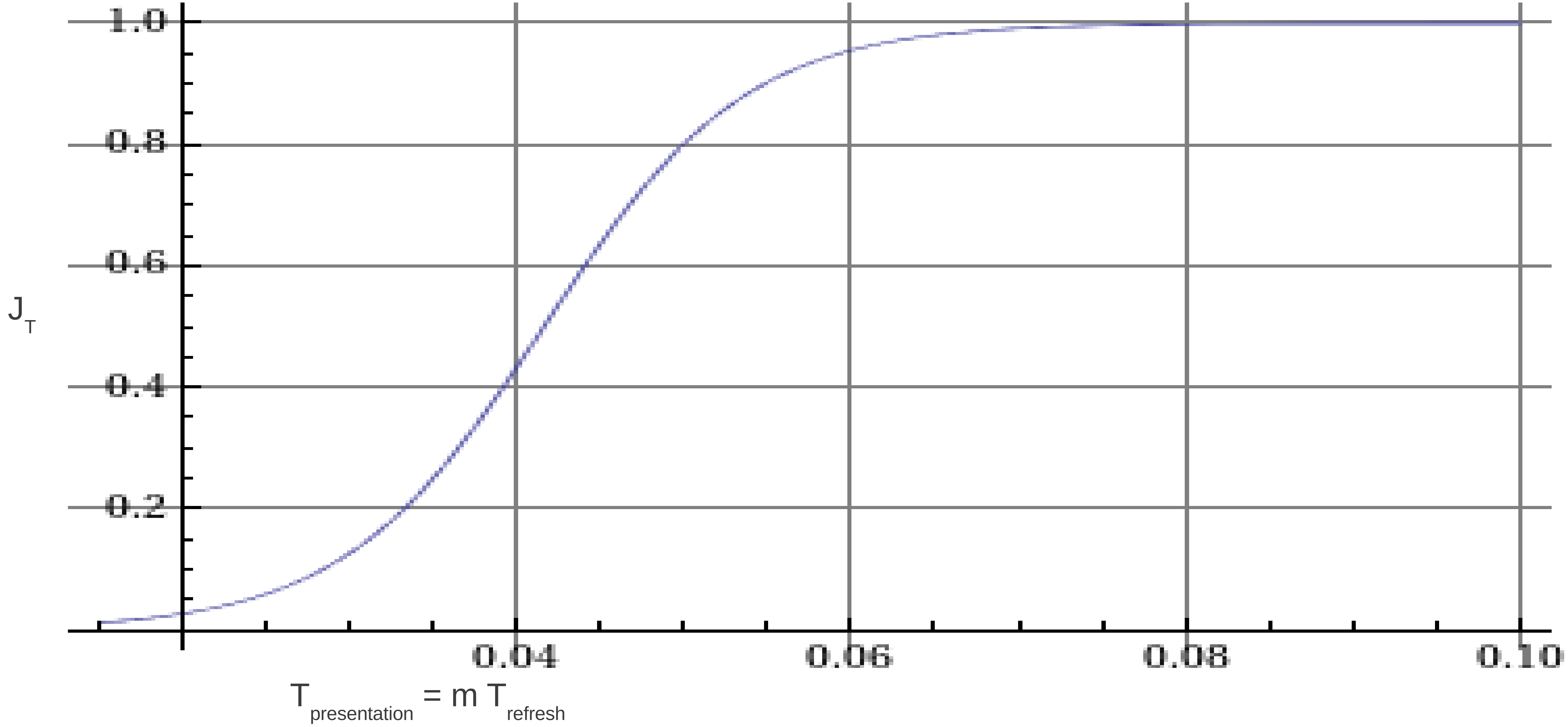
and presentation time T is integer multiple of monitor refresh $T_{refresh} = 1/F_{refresh}$.

T_{center} is proportional to a global measure of object velocity for a scene.

Sigmoid Judder Perception Function

"Crowd-run", 1.5 x picture height

T_{center} = 0.042 seconds, T_{gain}=0.00625



Example 1: if F_R=60, F_v=25, then n=2.4, p=2, q=3, a=0.6, b=0.4.
We obtain two presentation times: T_p=2/60=0.033, T_q=3/60=0.050 seconds.
From the graph, $J_{0.033}$ = 0.19, $J_{0.050}$ = 0.80, so:

$$Judder = 0.6\, J_{0.033} + 0.4\, J_{0.050} = 0.6 \times 0.19 + 0.4 \times 0.80 = 0.434$$

Example 2: if F_R=60, F_v=24, then n=2.5, p=2, q=3, a=0.5, b=0.5.
The same presentation times apply, so:

$$Judder = 0.5\, J_{0.033} + 0.5\, J_{0.050} = 0.5 \times 0.19 + 0.5 \times 0.80 = 0.495$$

The above shows simple linear interpolation from one point on the sigmoid to another.
The set of points is determined by the monitor refresh rate.

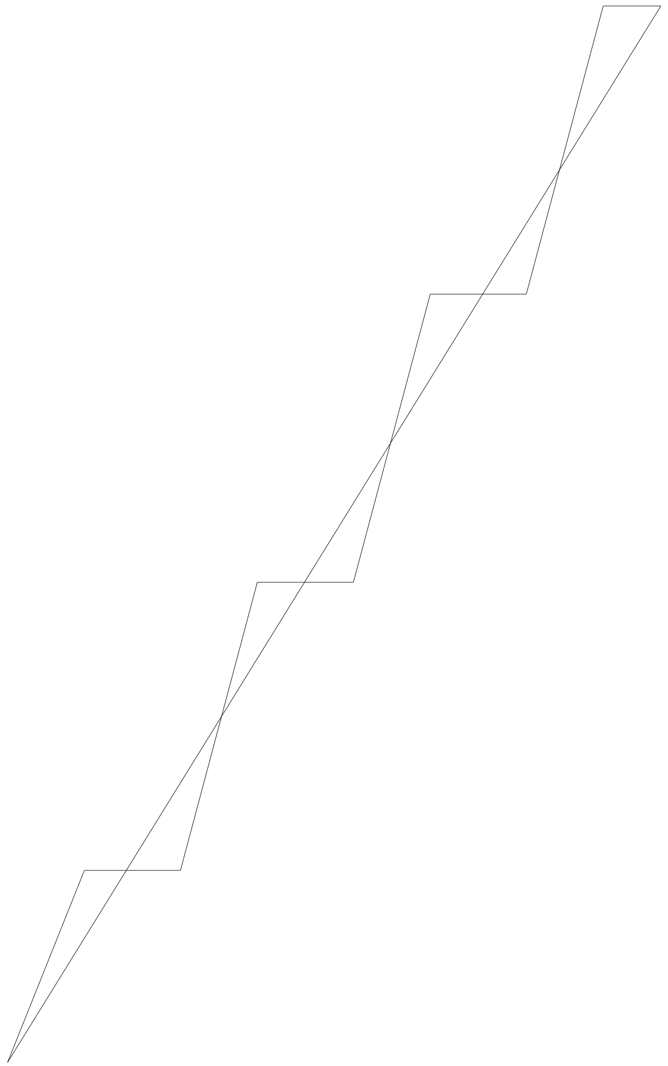
Example 3: if F_R=60, F_v=50, then n=1.2, p=1, q=2, a=0.8, b=0.2.
The presentation times are: 1/60=0.0167, and 2/60=0.033 seconds.

$$Judder = 0.8\, J_{0.0167} + 0.2\, J_{0.033} = 0.8 \times 0.015 + 0.2 \times 0.19 = 0.05$$

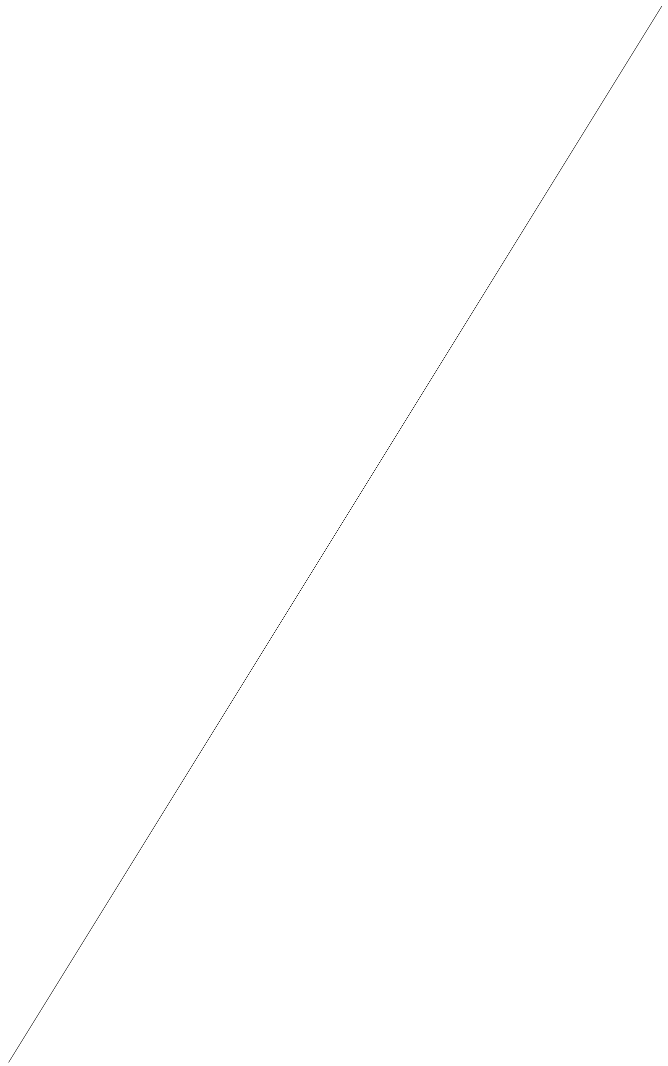
Judder Perception & Blur

Judder perception is discordance between natural (smooth) and perceived motion in our vision system

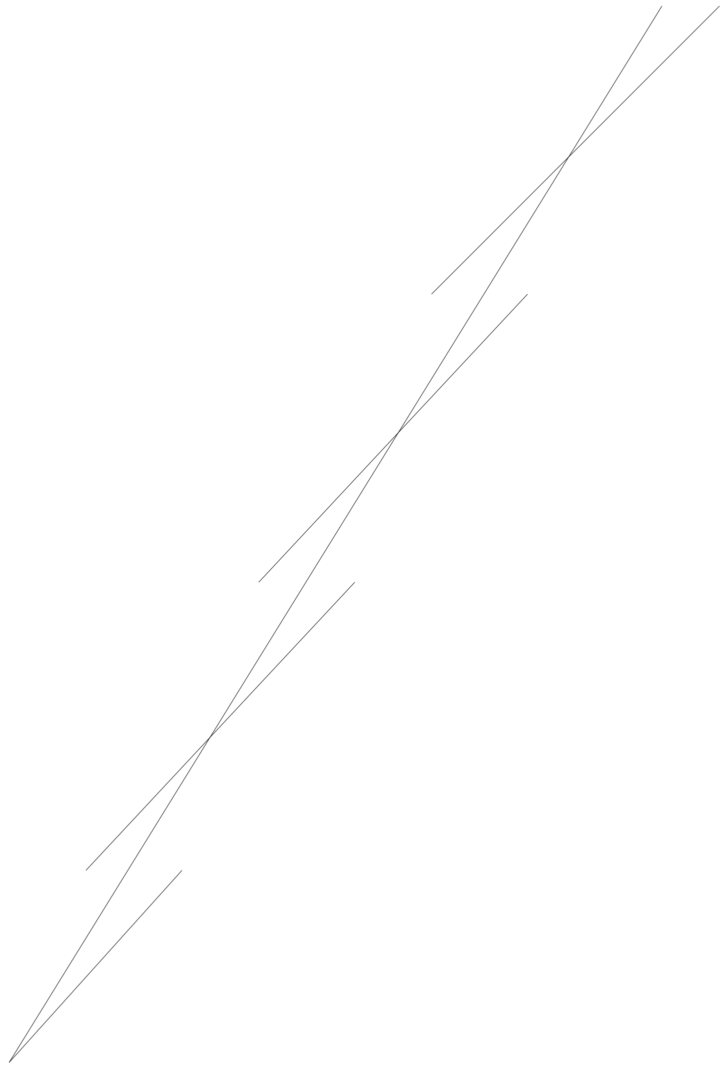
Shutter angle 120º –
may see judder



360º shutter - no judder-but may
look blurry at low frame rates



Upsampled - Shutter angle 720º
No judder but blurry



Judder Perception – a Summary:

- Judder perception is discordance between natural (smooth) and perceived motion in our vision system
- Perception varies between people
- Perception is proportional to speed of object motion
- Perception decreases with motion blur
- Perception increases with amount of detail/texture/edges in an object
- Perception increases with light levels and contrast
- Perception increases with solid angle of moving object to eye
- Perception increases with longer presentation times of individual images
- Perception increases for any larger integer multiples of refresh time in a rendering cadence
- “Frequencies” are not an issue – this is all about presentation time

Where Movie Production needs to go

- Large subtended angle, bright, high refresh rate displays are becoming common
- Sufficient movie information needs to be captured to meet human vision limitations
- Improved cameras and increased compute power allow new solutions
- Movies need to be near human perception limits

Solution: high frame rates → min. blur from cameras, none from graphics/effects

- Motion-compensated frame rate conversion then works well
- Simulated motion blur can be added for those who like 24 frames/sec

Reference: High Frame Rates Solve all Conversion Problems

- HFR reduces MCFRC problems associated with occlusions.
- HFR reduces the likelihood of aliasing in the reference.
- HFR uses faster shutter speeds, with much less motion blur.
- Motion deblurring is never required.
- Reduced motion blur improves edge detail - allows MCFRC to work better.
- Motion blur is small enough that it doesn't adversely MCFRC algorithms.
- In down-conversion, simulated motion blur can be added.
- New, fast technologies allows high quality derivatives to be created at will.
- For post-production, the “product” is the reference work.
- Derivatives can be automatically generated - some might prefer to adjust blur scene-by-scene.
- Experiments are on-going to add blur automatically based on scene analysis.

GPU acceleration of Motion-Compensated Frame-Rate Conversion

- Legato-cinema is our CUDA-based MCFRC product, with simulated motion blur.
- Without blur: 90 frames/sec output rates for 1080p50 to 60 conversions.
- Motion blur is implemented by upsampling to a higher frame rate and averaging groups of frames.
- Blur typically slows output to around 20 frames/sec.
- “Simulated” shutter angles are used to control motion blur – familiar paradigm for the movie industry.
- Estimate of input shutter angle can be used to control oversampling.
- Smaller input angles (higher oversampling) is visually safer, but mostly just slows conversions.
- The output angle controls the added output motion blur as expected.
- Motion deblur will probably never be supported!

System Issues

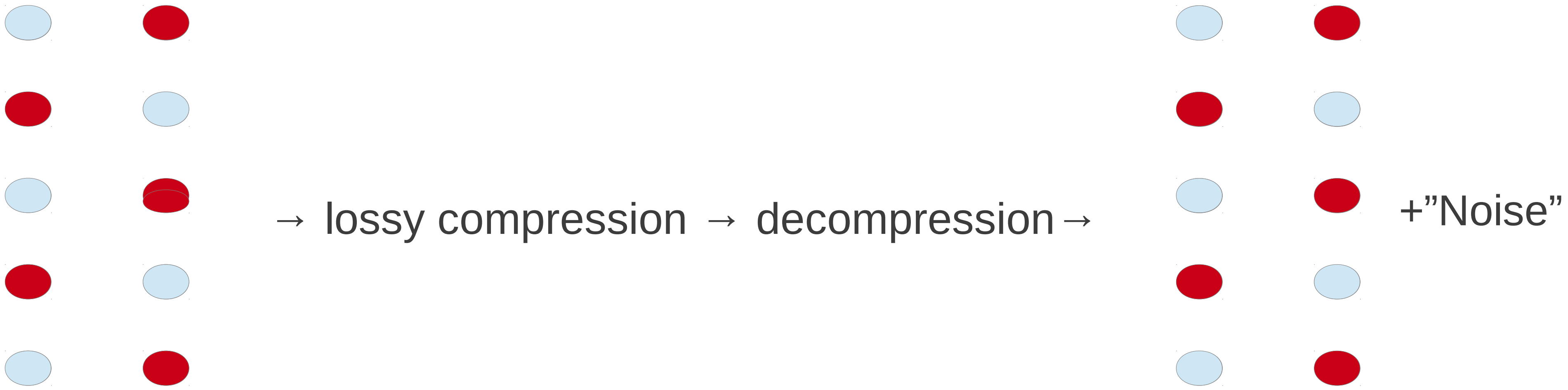
- 16-bit CUDA processing → improved SNR and simpler workflow
- Dynamic GPU resource allocation: multi-GPU systems avoid bottlenecks.
- Frame-grained parallelism achieves efficient conversions in multi-user systems and conversion pipelines.
- Our lossless 2:1 super-fast compression tool can be used to help preserve quality over many operations, while doubling storage bandwidth and halving file sizes.
- Relatively low CPU usage allows CPU intensive tools like x264 encoding to be in a processing pipeline.
- On our 3.8GHz over-clocked Intel 3930K reference machine with Samsung SSD 830, and VDPAU, we have been able to smoothly display 3840x2160 clips at 50 frames/sec.

Deinterlacing

- Same raw video bandwidth, each frame → two fields, double temporal sample rate
- Doubling temporal sampling can reduce the perception of judder, but...
- Sampling is spatially damaged by discarding alternate odd/even lines
- “Tearing” occurs from motion, so at some stage, deinterlacing for progressive displays is required
- Computation grows exponentially for an asymptotic improvement as more input samples become involved in reconstruction
- Excellent deinterlacing quality is computationally very expensive
- OK results are possible for 1080i if the output is spatially low-pass filtered – but why not use 720p?
- Human vision limitations → don't sit too close
- Spatial damage means modern compression algorithms (H.264 and HEVC) can do better with the same frame rate vs field rate at the same SNR

- A compressed interlaced transmission system can be replaced with:

  deinterlace → compress → decompress → reinterlace



- Interlaced (blue) samples on left → noisy channel to right
- Noise from (a) influence of interpolated (red) samples on compression, and (b) lossy compression
- Deinterlacing provides “progressive” video at the end of transmission for future-proof system integration
- Deinterlacing provides “progressive” video for archives where the original interlaced can be extracted with an improved SNR vs compressing raw interlaced directly.

TV Distribution & Broadcast

“piecemeal” replacement of interlaced capture/production/distribution systems is possible.

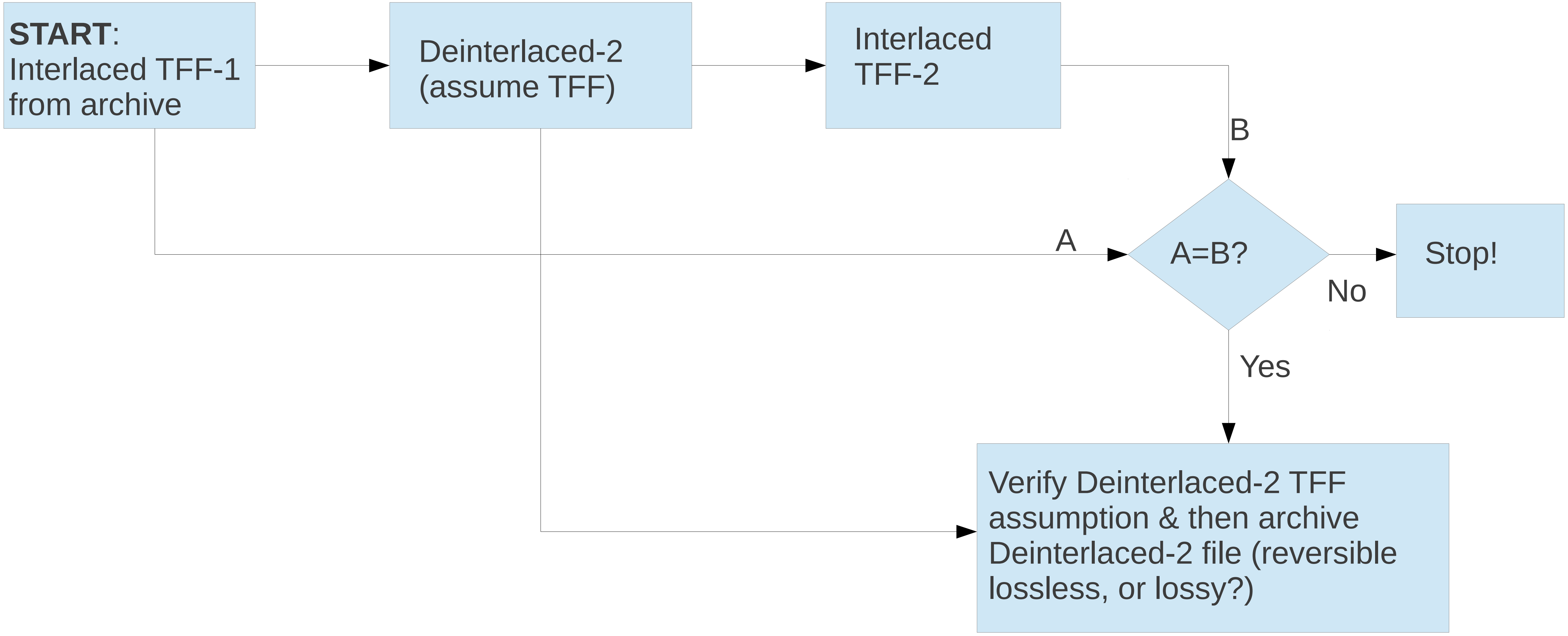
May take a *long* time. No technical barriers remain. Motives for migration include:

- Lower bit rates – lower costs
- Progressive systems or better deinterlacers → improved distributed image quality
- Better control of final quality (no deinterlacers “in the wild”)
- Lower transmission bandwidth/channel in future ATSC (>=2.0) broadcast
- Better integration with Internet and computer-based display systems.
- Better access to portable devices (which can't/don't deinterlace)
- Simpler production and editing
- Simpler conversion between formats (scaling, frame-rate-conversion, etc)

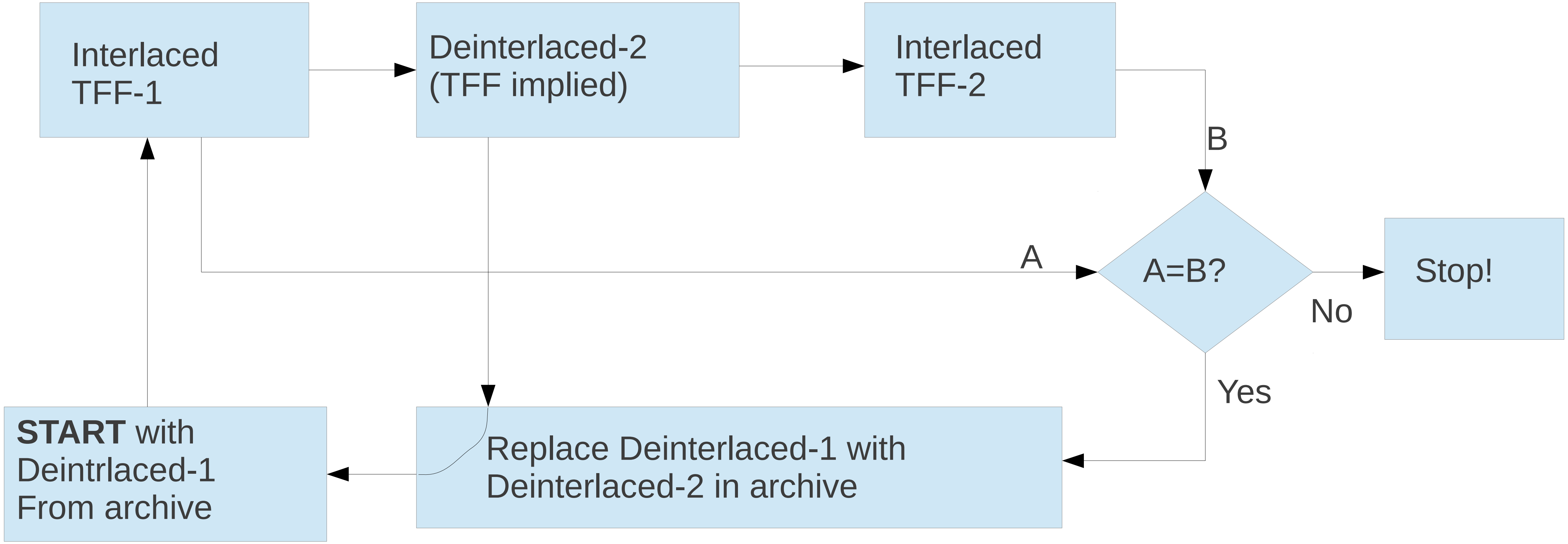
Demeler Deinterlacer

- CUDA-based motion estimation is particularly effective for most of picture area
- CUDA: diagonal interpolation - improves results in some situations
- CUDA: any failures in motion estimation and diagonal interpolation (resulting in combing) are detected and patched
- Faster than real-time performance is possible with two GTX 580s or GTX 690s.
- Demeler has low flicker and ***no output filtering*** → Low Flicker Field Pass-Through (LFFPT)
- LFFPT → lossless compressed deinterlaced archives can recover the original interlaced video.
- LFFPT → an average 15% bandwidth reduction when used before H.264 or HEVC compression, instead of compressing interlaced directly.

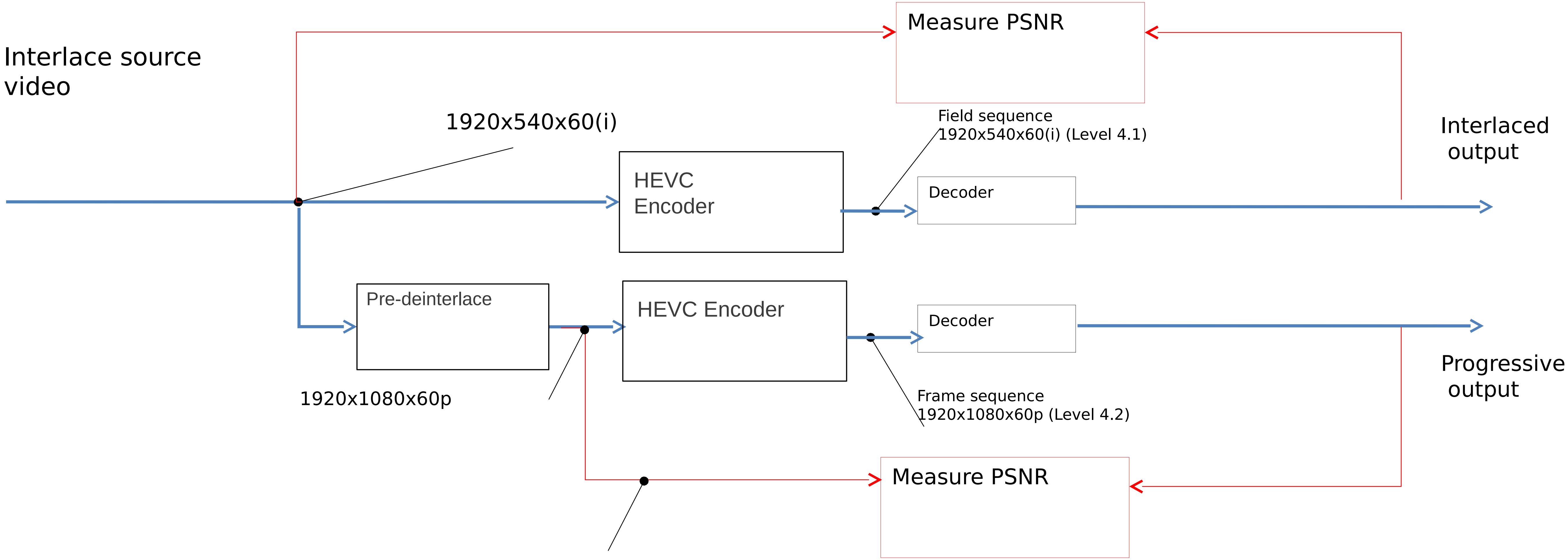
Moving an Interlaced File from Interlaced -> Deinterlaced Archive



Updating a Deinterlaced Archive (new Deinterlacer)



Interlaced vs. deinterlaced (HEVC)



*fine detail is preserved, **and input fields passed through unchanged**.*

Results so far..

On the diverse but challenging test sequence set chosen..

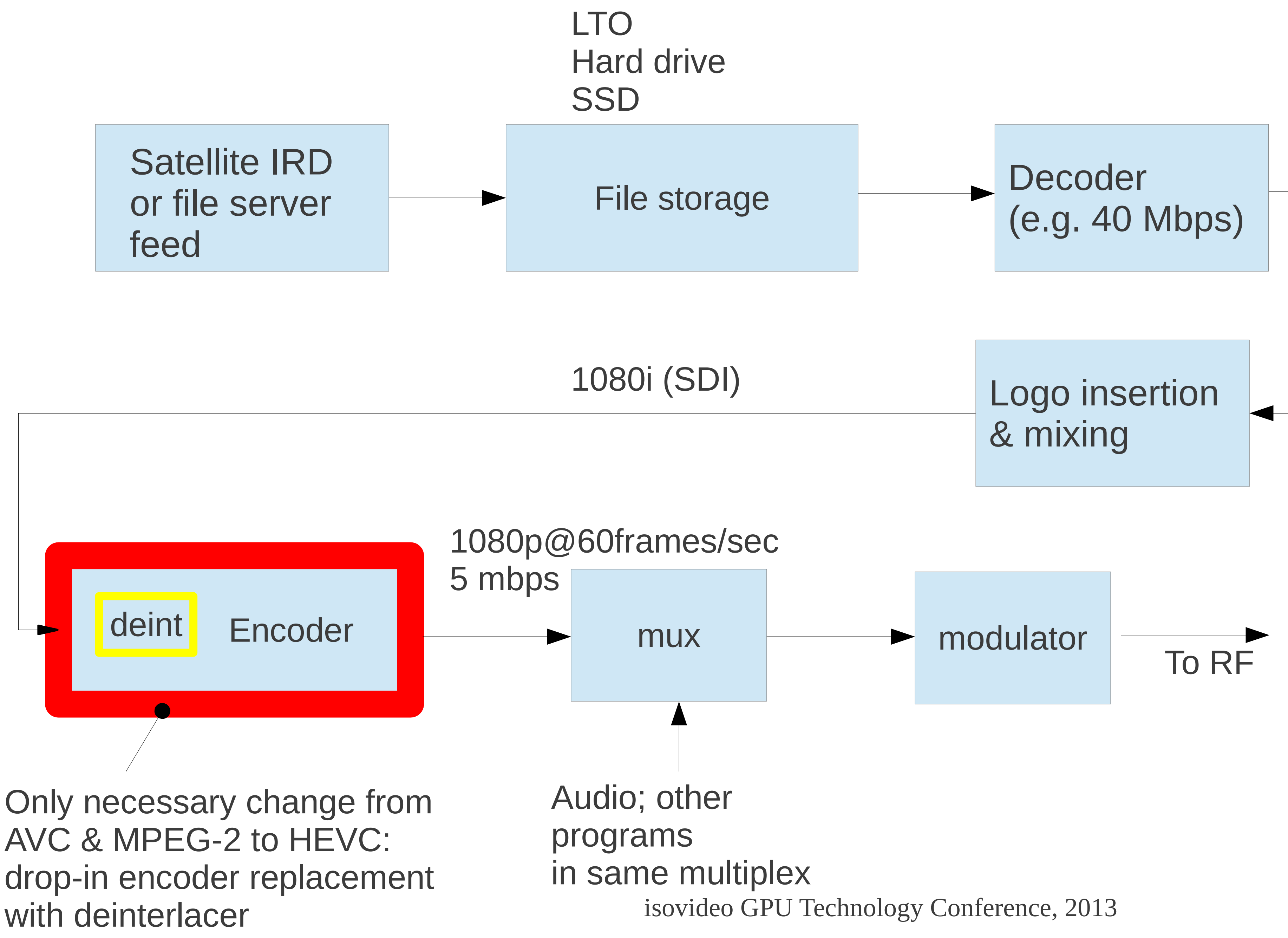
- deinterlaced HEVC coded frame sequences average -15% (lower bit-) rates than HEVC coded field sequences (fixed QP=22,27,32,37, HM 8.0). Range is -39% to +32%

- Pre-encoding deinterlaced AVC coded sequences average lower rate than AVC MBAFF coded frame sequences (-18%).

  Range [-40%,+22%]

- Bdrate() suggests deinterlacing prior to encoding is better than deinterlacing after decoding.

Upgrading to HEVC



Viarte Professional Quality Standards-Conversion/Transcoding Server

- Simple deployment – Viarte is file-based and mountable as a shared drive,
- Scalable to multiple servers,
- Configurable drag-and-drop triggers one or more conversions,
- Faster-than-realtime full-HD throughput via i) load-balanced multi-GPU acceleration and ii) an intelligent optimizatiion (that speeds up throughput by up to 250%).
- Bit-rate reduction achieved by customizing frame rates and images sizes for distribution to mobile networks, while maintaining or improving picture quality.