LUMA ADJUSTMENT FOR HIGH DYNAMIC RANGE VIDEO

Jacob Ström, Jonatan Samuelsson, Kristofer Dovstam
Ericsson Research

OUTLINE

- Luminance (Y) and luma (Y')
- Strange things that happen during subsampling
- How to do subsampling without artifacts
- Implementation details
- Applications

- Bonus?

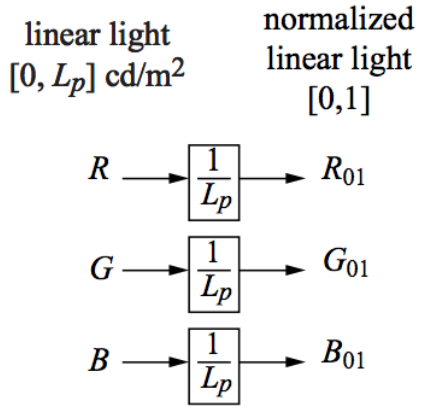
LUMINANCE

- Assume linear (R, G, B) tuple
  - (think “linear output of red source, etc”)
- Luminance is weighted average
  - $Y = w_R R + w_G G + w_B B$
  - Linear
  - “How bright”
  - Y in CEI1931 XYZ color space
  - $\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \quad 3x3 \quad \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$

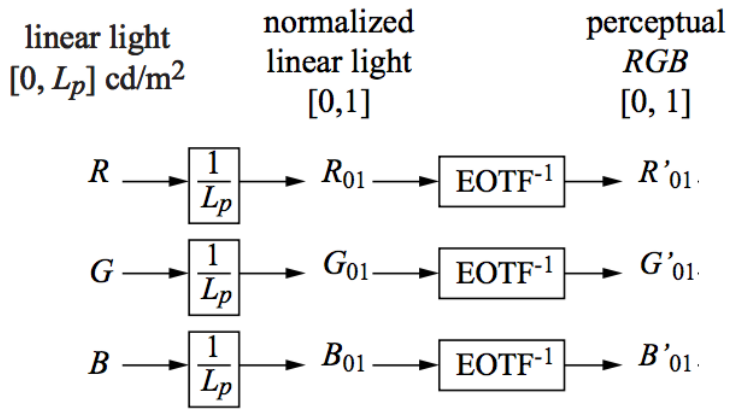
LUMA

- Non-linear values (R', G', B'):
  - $R = tf(R')$ $\quad R' = tf^{-1}(R)$
  - $G = tf(G')$ or $G' = tf^{-1}(G)$
  - $B = tf(B')$ $\quad B' = tf^{-1}(B)$
- Luma is weighted average
  - $Y' = w_R R' + w_G G' + w_B B'$
  - Non-linear
  - Not "how bright"
  - Y' in Rec.709 and BT.2020 Y'CbCr
  - $\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} \quad 3x3 \quad \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$

PREPROCESSING

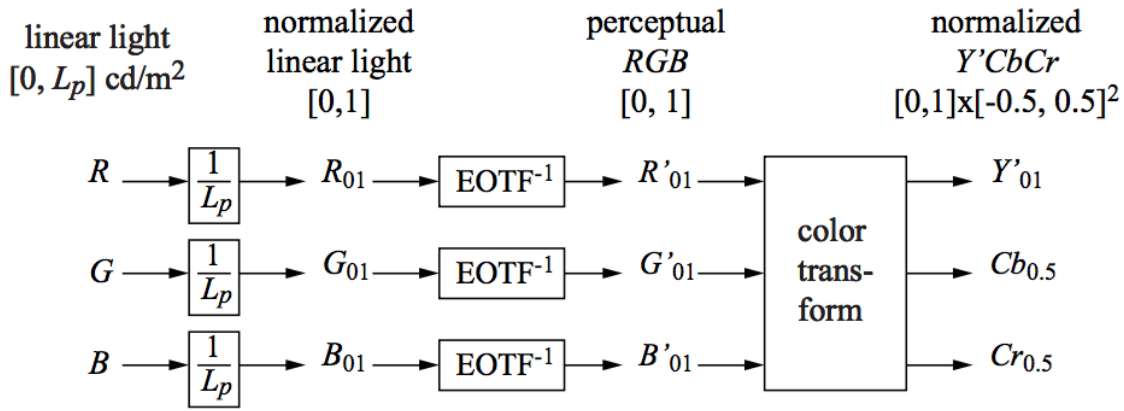


PREPROCESSING

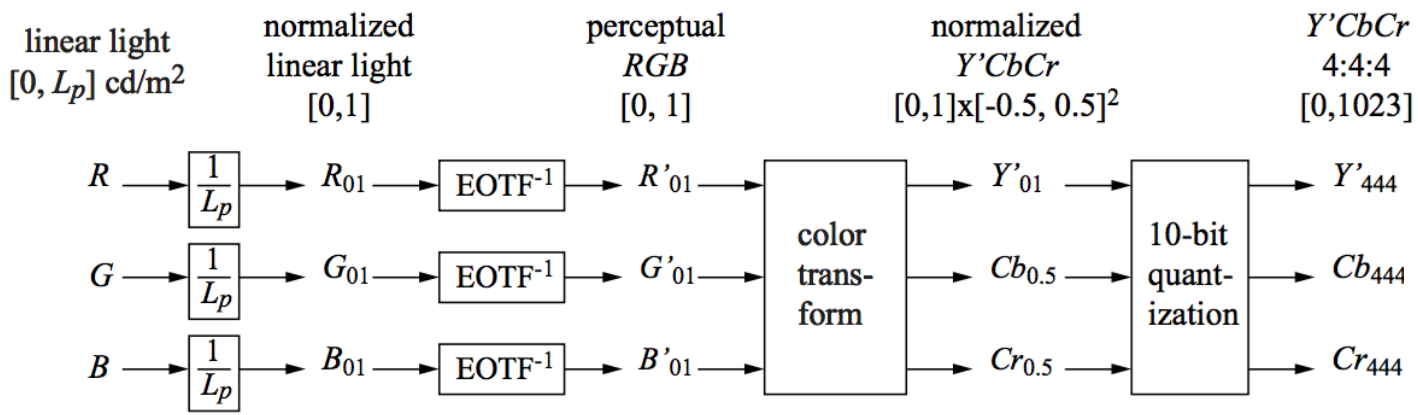


inverse gamma
a.k.a. inverse Electro Optical Transfer Function (EOTF^{-1}). We use SMPTE ST 2084 (PQ)

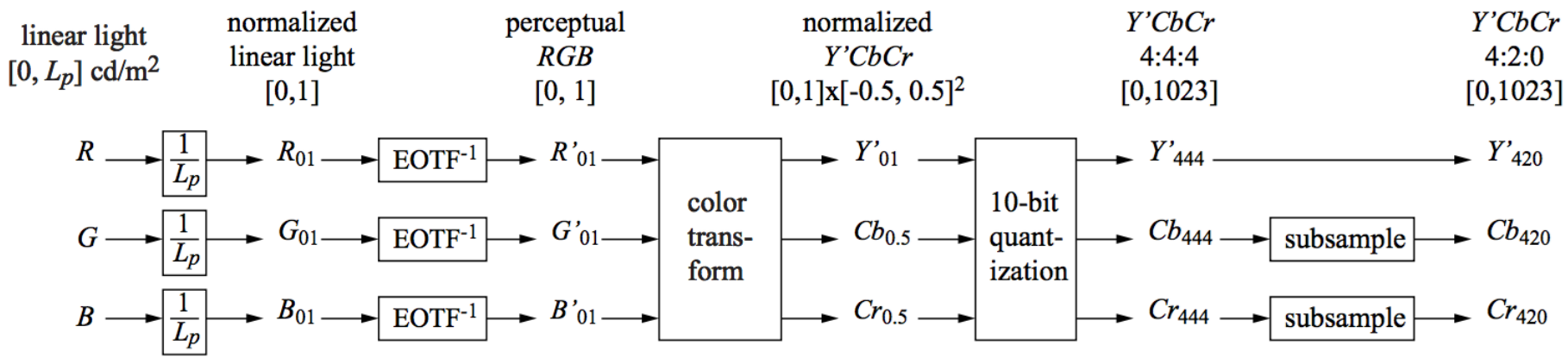
PREPROCESSING



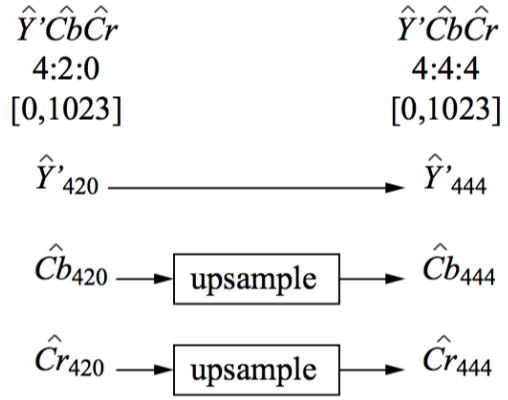
PREPROCESSING



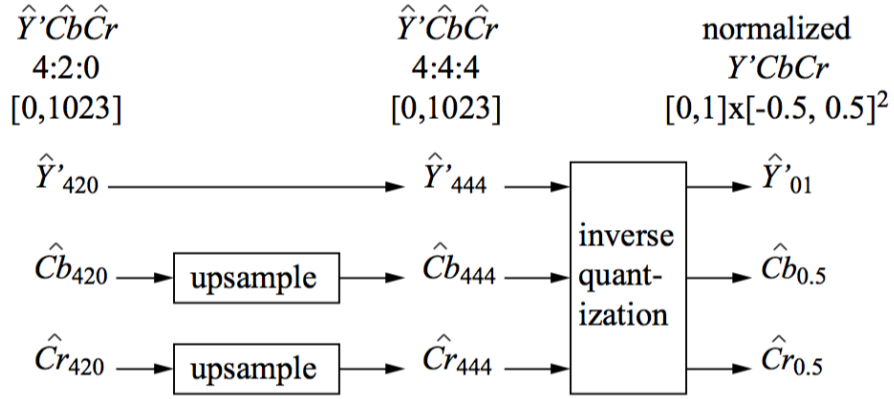
PREPROCESSING



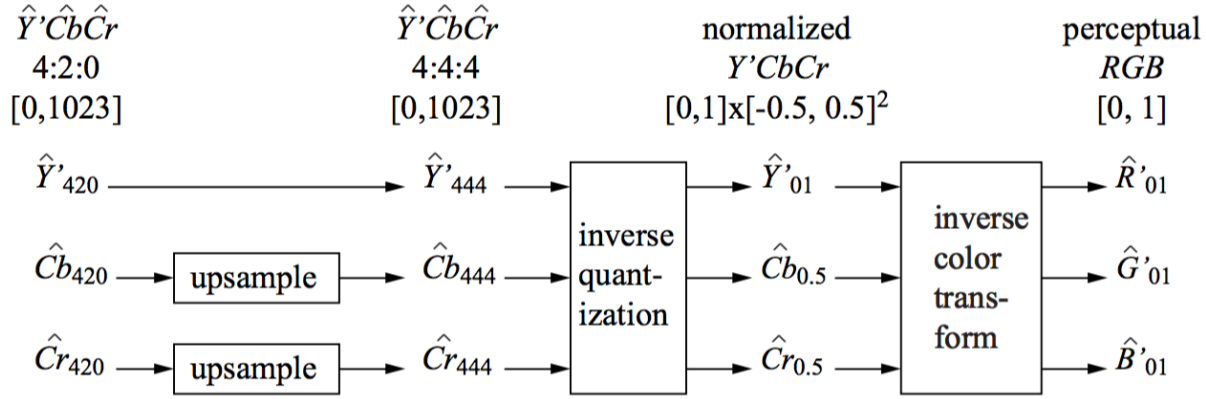
POST PROCESSING



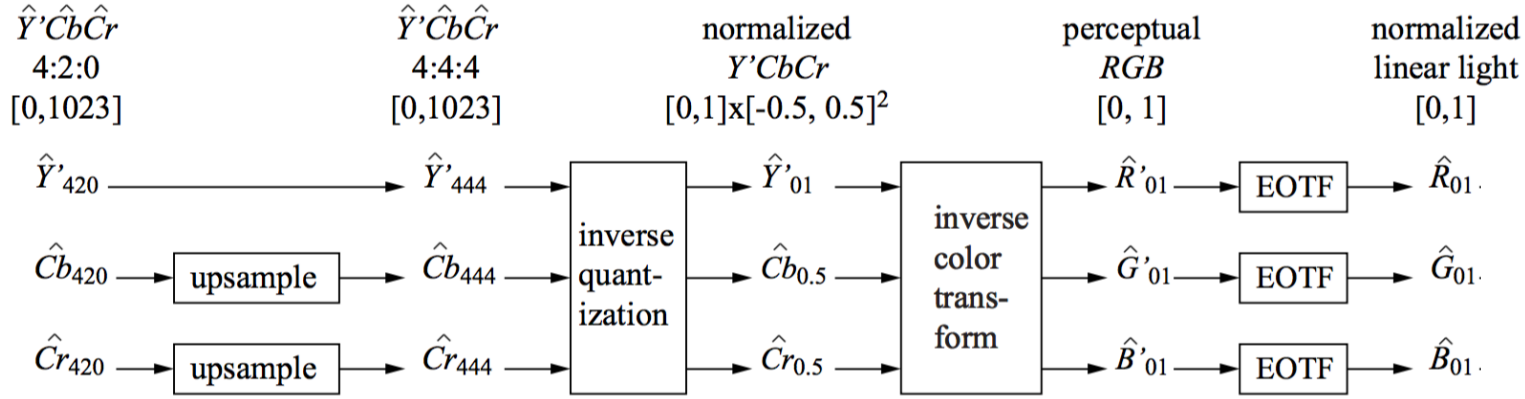
POST PROCESSING



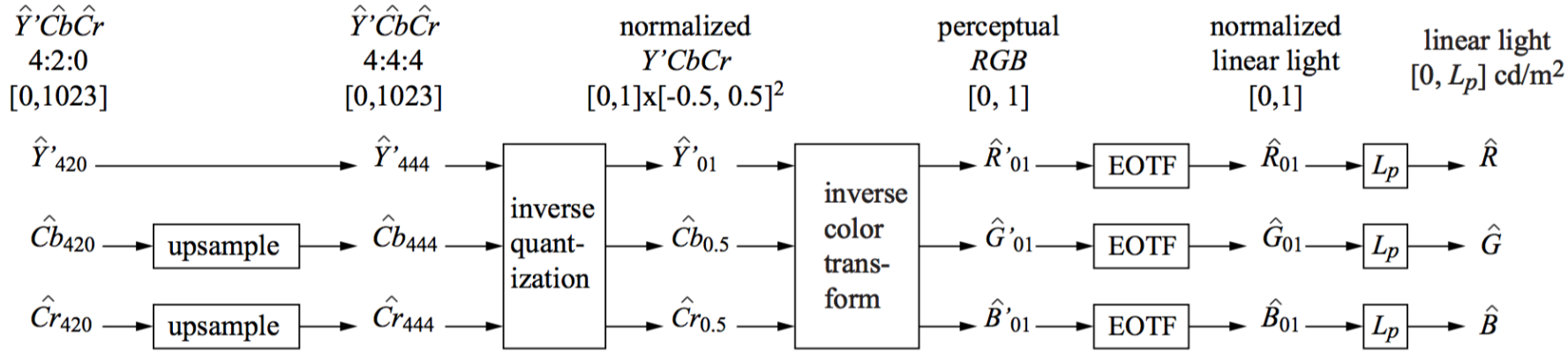
POST PROCESSING



POST PROCESSING



POST PROCESSING



ARTIFACTS

original 4:4:4

subsampled to 4:2:0
(no compression!)

Sequence courtesy of Technicolor and the NevEx project

ARTIFACTS

- Spotted by E. Francois in an MPEG contribution
- Also noted that the effect is biggest for saturated colors

WHAT IS HAPPENING?

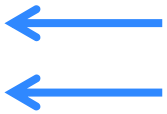
- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263,\ 646,\ 831)$
  - $Y'CbCr2 = (401,\ 571,\ 735)$

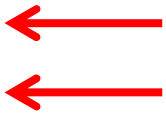
WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$



Quite similar

- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263,\ 646,\ 831)$
  - $Y'CbCr2 = (401,\ 571,\ 735)$



Quite different

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263,\ 646,\ 831)$
  - $Y'CbCr2 = (401,\ 571,\ 735)$
- After averaging
  - $Y'CbCr = (332,\ 608.5,\ 783)$

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263,\ 646,\ 831)$
  - $Y'CbCr2 = (401,\ 571,\ 735)$
- After averaging
  - $Y'CbCr = (332,\ 608.5,\ 783)$
- After converting back to RGB:
  - $RGB = (1001,\ 0.48,\ 100.5)$

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263,\ 646,\ 831)$
  - $Y'CbCr2 = (401,\ 571,\ 735)$
- After averaging
  - $Y'CbCr = (332,\ 608.5,\ 783)$
- After converting back to RGB:
  - $RGB = (1001,\ 0.48,\ 100.5)$

Quite similar

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263,\ 646,\ 831)$
  - $Y'CbCr2 = (401,\ 571,\ 735)$
- After averaging
  - $Y'CbCr = (332,\ 608.5,\ 783)$
- After converting back to RGB:
  - $RGB = (1001,\ 0.48,\ 100.5)$

Quite similar

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263,\ 646,\ 831)$
  - $Y'CbCr2 = (401,\ 571,\ 735)$
- After averaging
  - $Y'CbCr = (332,\ 608.5,\ 783)$
- After converting back to RGB:
  - $RGB = (1001,\ 0.48,\ 100.5)$

Conclusion:

Just averaging is not a problem.

Quite similar

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263, 646, 831)$
  - $Y'CbCr2 = (401, 571, 735)$
- After averaging
  - $Y'CbCr = (332, 608.5, 783)$
- Average Cb/Cr, copy Y':
  - $Y'CbCr1 =$
  - $Y'CbCr2 =$

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263, 646, 831)$
  - $Y'CbCr2 = (401, 571, 735)$
- After averaging
  - $Y'CbCr = (332, 608.5, 783)$

- Average Cb/Cr, copy Y':
  - $Y'CbCr1 = (\times, 608.5, 783)$
  - $Y'CbCr2 = (\times, 608.5, 783)$

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263, 646, 831)$
  - $Y'CbCr2 = (401, 571, 735)$
- After averaging
  - $Y'CbCr = (332, 608.5, 783)$
- Average Cb/Cr, copy Y':
  - $Y'CbCr1 = (332, 608.5, 783)$
  - $Y'CbCr2 = (332, 608.5, 783)$
- After conversion back to RGB:
  - $RGB1 = (484,\ 0.03,\ 45)$
  - $RGB2 = (2061,\ 2.2,\ 216)$

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263, 646, 831)$
  - $Y'CbCr2 = (401, 571, 735)$
- After averaging
  - $Y'CbCr = (332, 608.5, 783)$
- Average Cb/Cr, copy Y':
  - $Y'CbCr1 = (332, 608.5, 783)$
  - $Y'CbCr2 = (332, 608.5, 783)$
- After conversion back to RGB:
  - $RGB1 = (484,\ 0.03,\ 45)$
  - $RGB2 = (2061,\ 2.2,\ 216)$

Way too dark!

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263, 646, 831)$
  - $Y'CbCr2 = (401, 571, 735)$
- After averaging
  - $Y'CbCr = (332, 608.5, 783)$
- Average Cb/Cr, copy Y':
  - $Y'CbCr1 = (332, 608.5, 783)$
  - $Y'CbCr2 = (332, 608.5, 783)$
- After conversion back to RGB:
  - $RGB1 = (484,\ 0.03,\ 45)$
  - $RGB2 = (2061,\ 2.2,\ 216)$

Way too bright!

WHAT IS HAPPENING?

- Two neighboring pixels:
  - $RGB1 = (1000,\ 0,\ 100)$
  - $RGB2 = (1000,\ 4,\ 100)$
- After conversion to Y'CbCr:
  - $Y'CbCr1 = (263, 646, 831)$
  - $Y'CbCr2 = (401, 571, 735)$
- After averaging
  - $Y'CbCr = (332, 608.5, 783)$
- Average Cb/Cr, copy Y':
  - $Y'CbCr1 = (332, 608.5, 783)$
  - $Y'CbCr2 = (332, 608.5, 783)$
- After conversion back to RGB:
  - $RGB1 = (484,\ 0.03,\ 45)$
  - $RGB2 = (2061,\ 2.2,\ 216)$

Conclusion:

Subsampling of two components ***and*** copying of third component creates trouble.

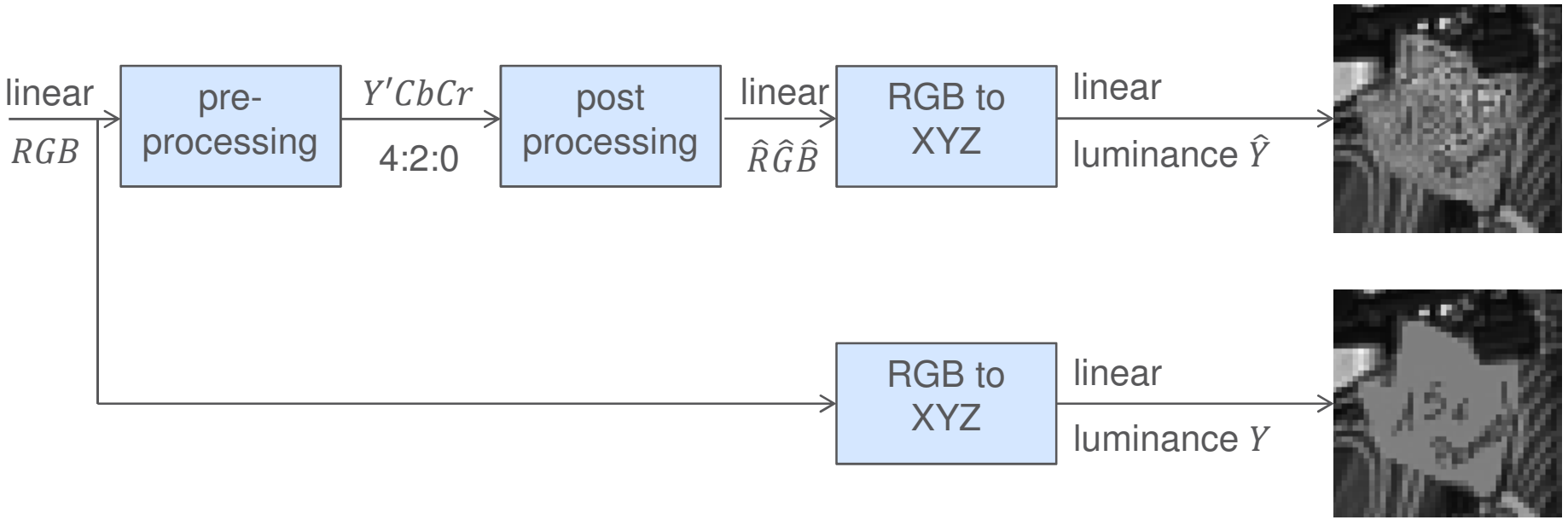
MAIN IDEA

- Y’ will make pixels too dark or too bright
- But! We can set Y’ individually for each pixel.
- Why not set it so that the pixel is just as bright as the original?
- This means matching luminance Y

- Task: Find Y’ (luma) that gives the right Y (luminance)

- We call it Luma Adjustment

OVERVIEW



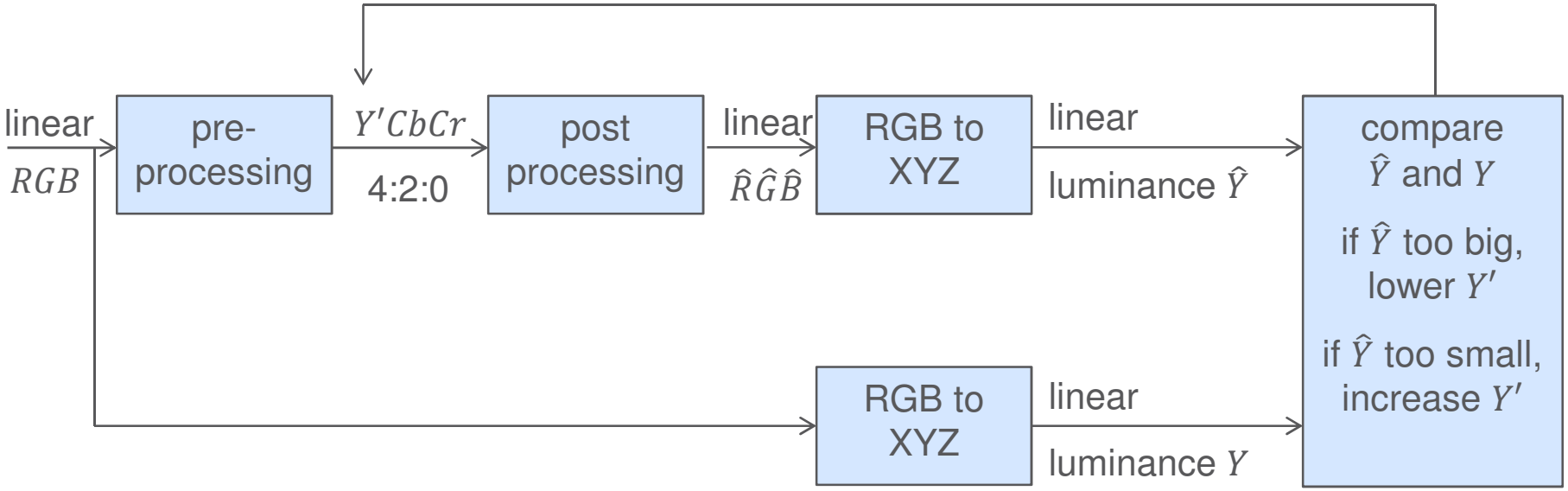
4:2:0 subsampling not really preserving Y

FIRST APPROACH

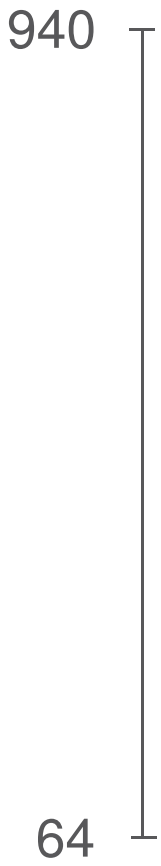
- Y’ is an integer in [0, 1023] (or [64, 940] in standard range).
- Try all Y’ values and see which generates the best luminance Y
- Very slow...

- Lucky thing:
  - Luminance Y is monotonously increasing with Y’
  - If Y’ gives something that is too bright, the optimum Y’ must be smaller than that value

LUMA ADJUSTMENT

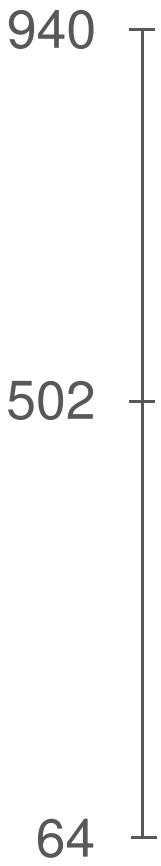


BINARY SEARCH



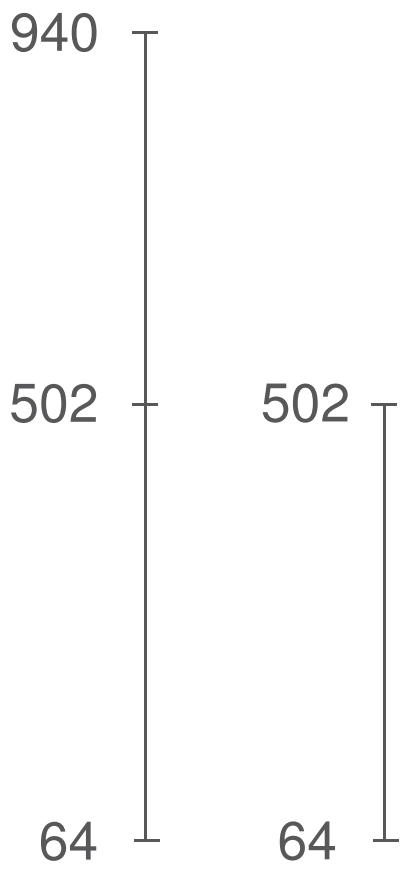
- We know the Y' value must be between 64 and 940

BINARY SEARCH



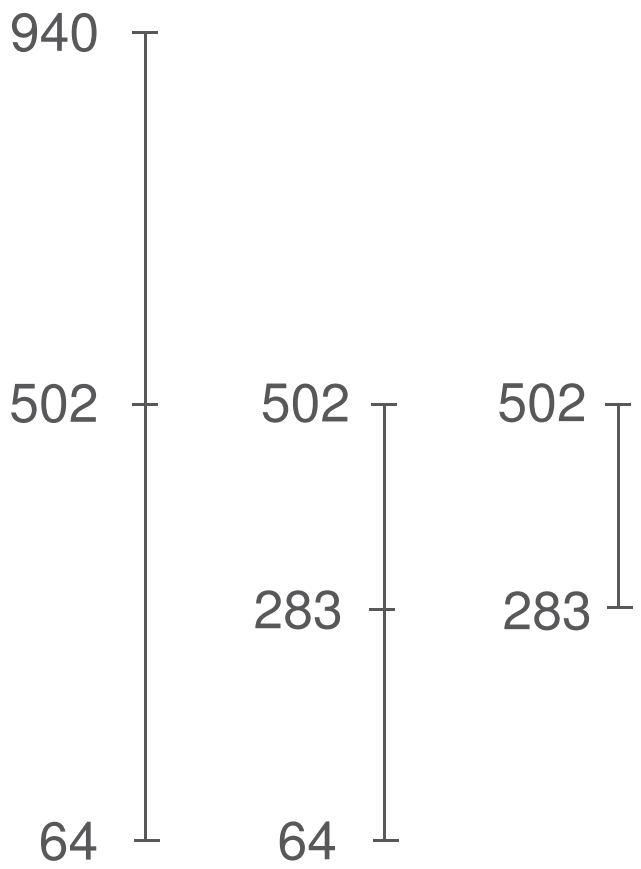
- We know the Y' value must be between 64 and 940
- We try the middle value (64+940)/2 = 502

BINARY SEARCH



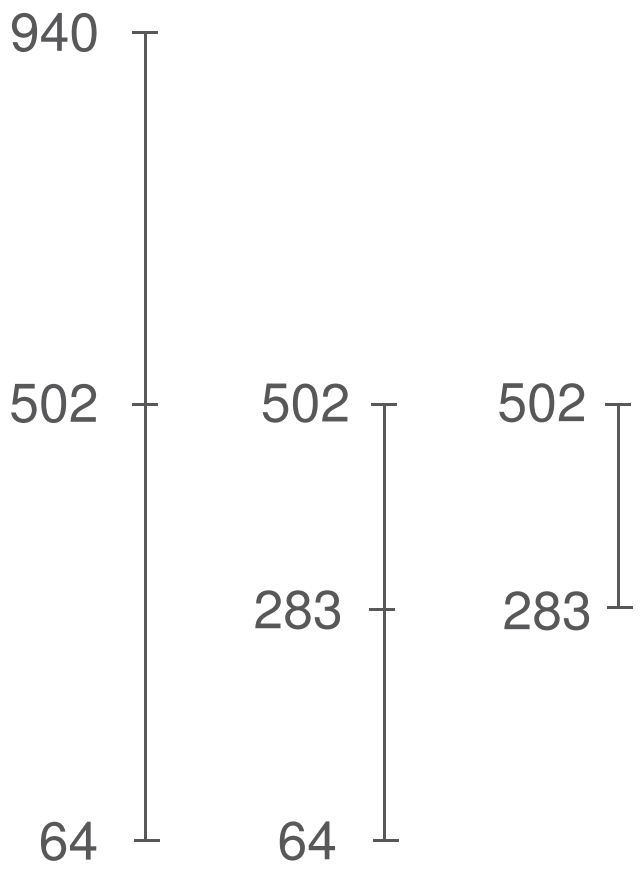
- We know the Y' value must be between 64 and 940
- We try the middle value (64+940)/2 = 502
- If Y' = 502 generates a linear luminance $\hat{Y}$ that is too bright, we know the best Y' must be in [64, 502]

BINARY SEARCH



- We know the Y' value must be between 64 and 940
- We try the middle value (64+940)/2 = 502
- If Y' = 502 generates a linear luminance $\hat{Y}$ that is too bright, we know the best Y' must be in [64, 502]
- If Y' = (64+502)/2 = 283 makes $\hat{Y}$ too dark, Y' should be in [283, 502] etc.

BINARY SEARCH



- We know the Y' value must be between 64 and 940
- We try the middle value (64+940)/2 = 502
- If Y' = 502 generates a linear luminance $\hat{Y}$ that is too bright, we know the best Y' must be in [64, 502]
- If Y' = (64+502)/2 = 283 makes $\hat{Y}$ too dark, Y' should be in [283, 502] etc.
- 10 iterations instead of 1024

FURTHER REFINEMENT: BOUNDS

- The fact that the EOTF is monotonously increasing can be used to create a lower and upper bound on Y' given Y:
- Upper

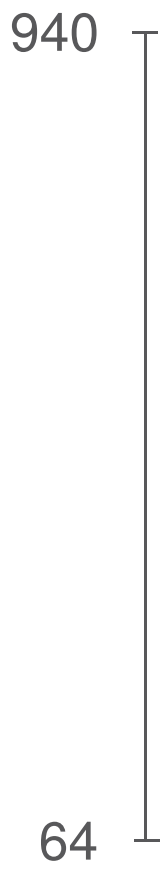
$$Y' \leq tf^{-1}\left(\frac{Y}{L_p}\right) + \max\{-a_{13}Cr, \quad a_{22}Cb + a_{23}Cr,,, -a_{22}Cb\}$$

- Lower

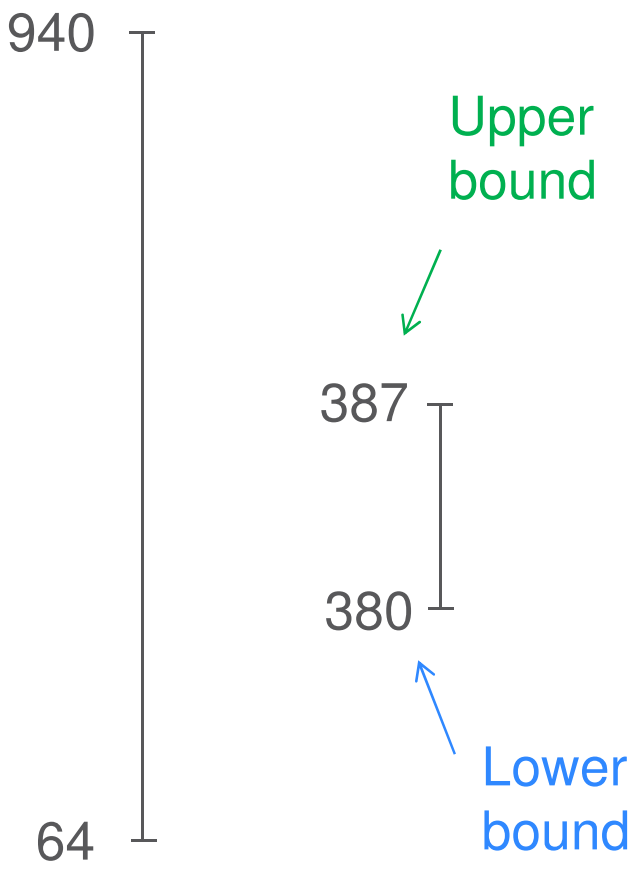
$$Y' \geq tf^{-1}\left(\frac{Y}{L_p}\right) + \min\{-a_{13}Cr, \quad a_{22}Cb + a_{23}Cr,,, -a_{22}Cb\}$$

- An even tighter upper bound can be used due to the EOTF being convex.

FURTHER REFINEMENT: BOUNDS

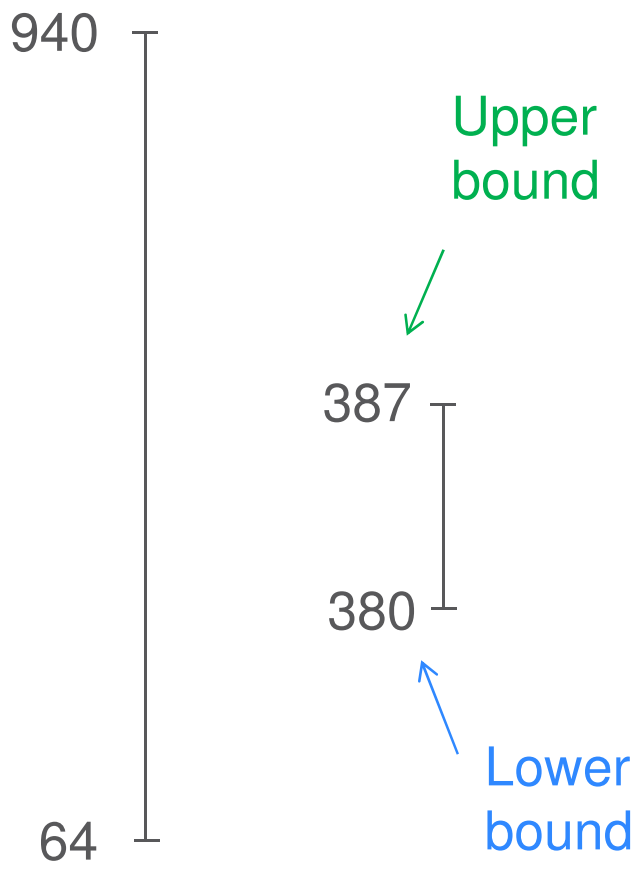


FURTHER REFINEMENT: BOUNDS



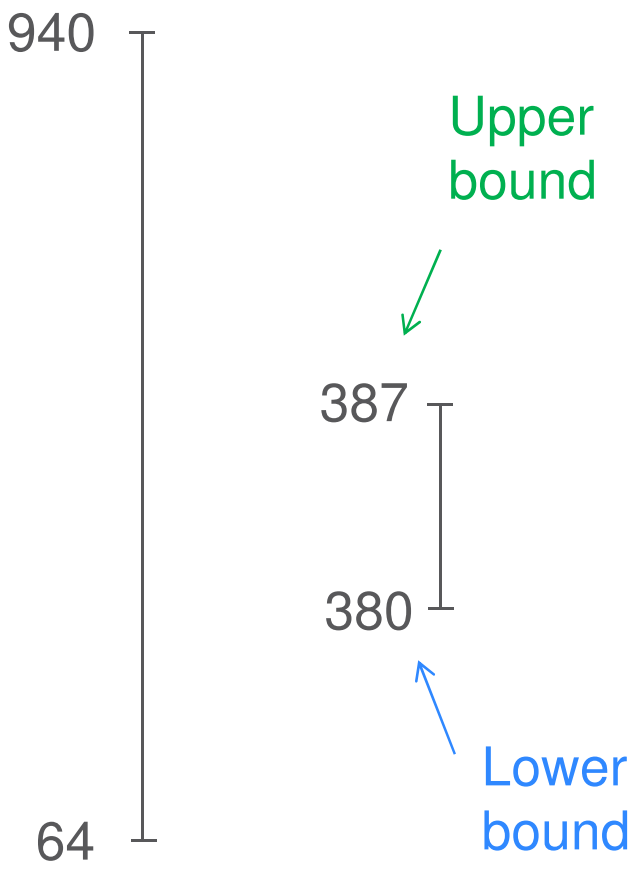
- Calculate bounds

FURTHER REFINEMENT: BOUNDS



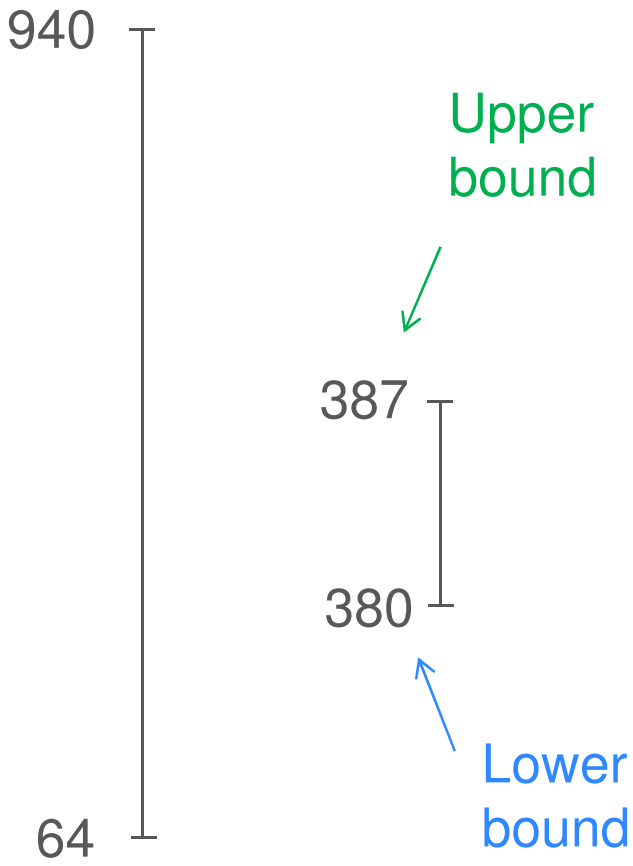
- Calculate bounds
- Continue interval halving from there

FURTHER REFINEMENT: BOUNDS



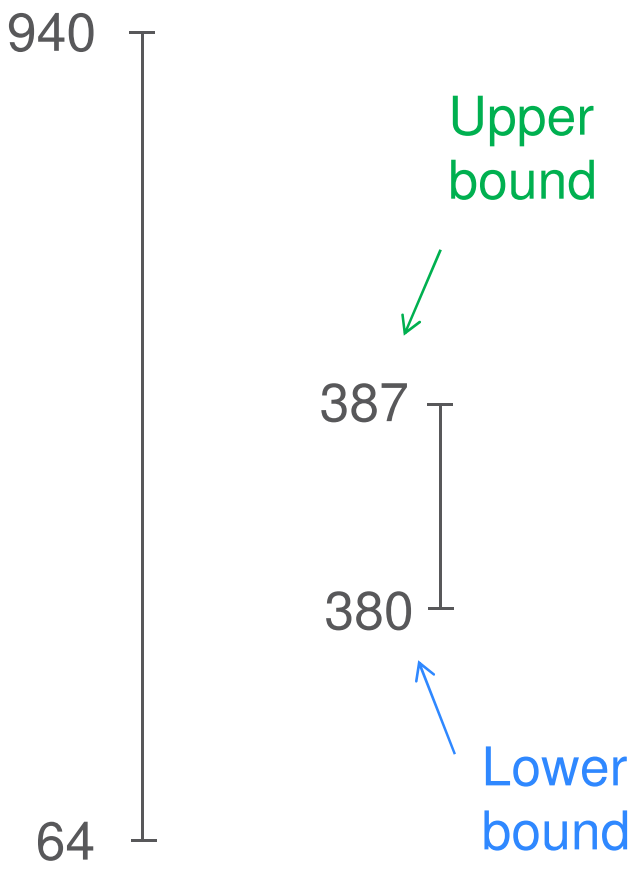
- Calculate bounds
- Continue interval halving from there
- Average number of iterations in a test sequence: < 5 iterations

FURTHER REFINEMENT: BOUNDS



- Calculate bounds
- Continue interval halving from there
- Average number of iterations in a test sequence: < 5 iterations
- Tighter bounds < 2 iterations

FURTHER REFINEMENT: BOUNDS



- Calculate bounds
- Continue interval halving from there
- Average number of iterations in a test sequence: < 5 iterations
- Tighter bounds < 2 iterations
- Other approches: LUTs, linearizations

RESULTS

- We do not have any material that fills out the BT.2020 color container.
- In the future, we will have such material.
- Example images are created using Rec.709 container and material that fills out the Rec.709 gamut.

EXAMPLES

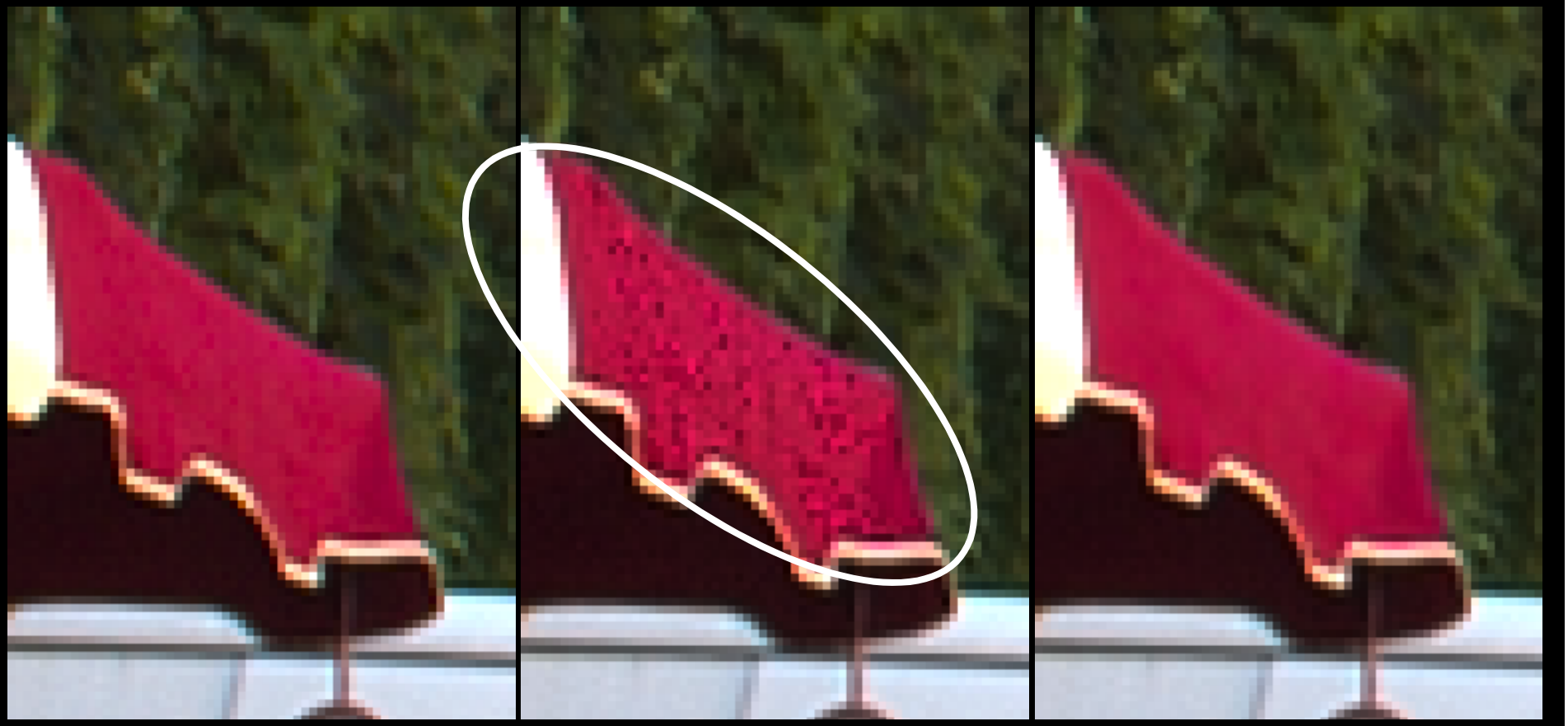
Original 4:4:4

Traditional subsampling
(no compression)

Luma Adjustment
(proposed method)

Sequence courtesy of Technicolor and the NevEx project

EXAMPLES



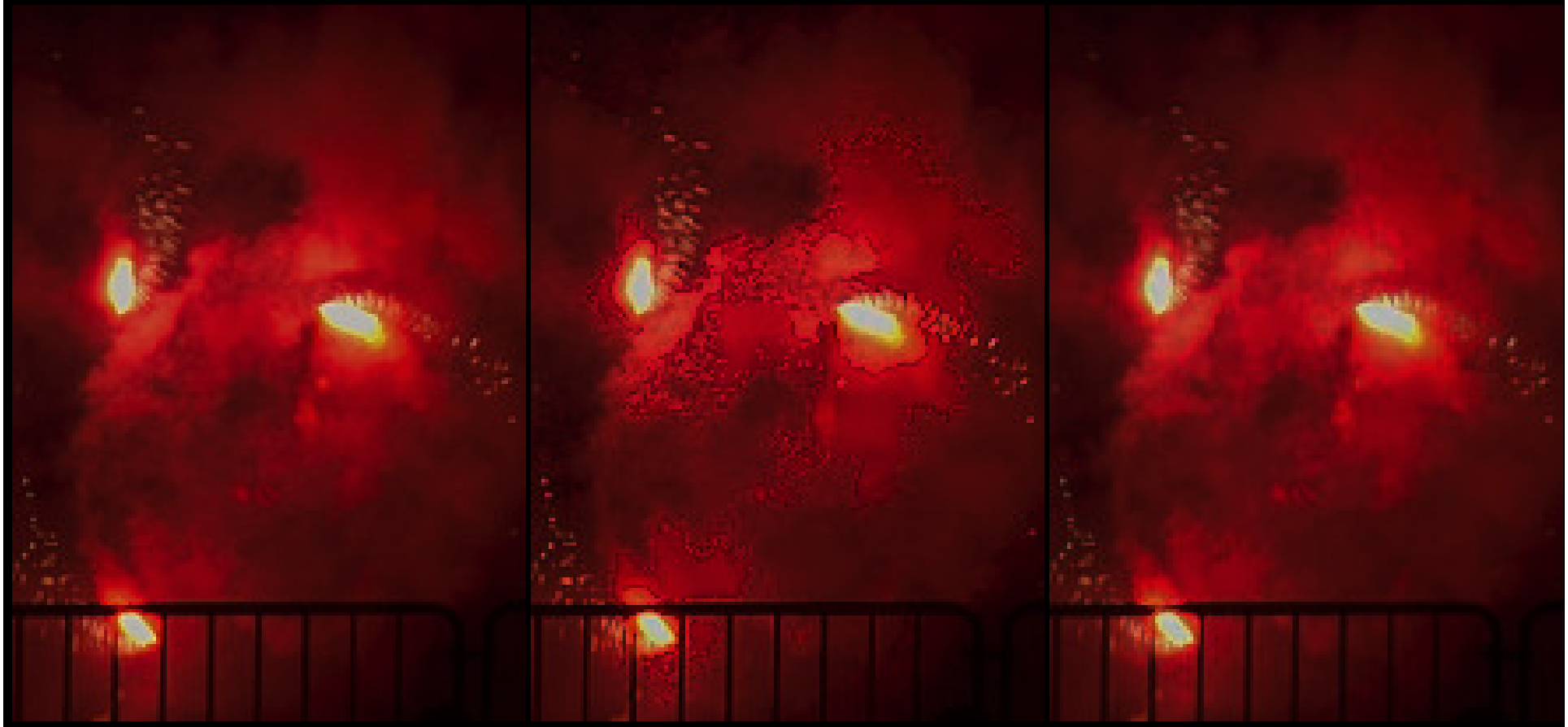
Original 4:4:4

Traditional subsampling
(no compression)

Luma Adjustment
(proposed method)

Sequence courtesy of Technicolor and the NevEx project

COMPRESSED EXAMPLES



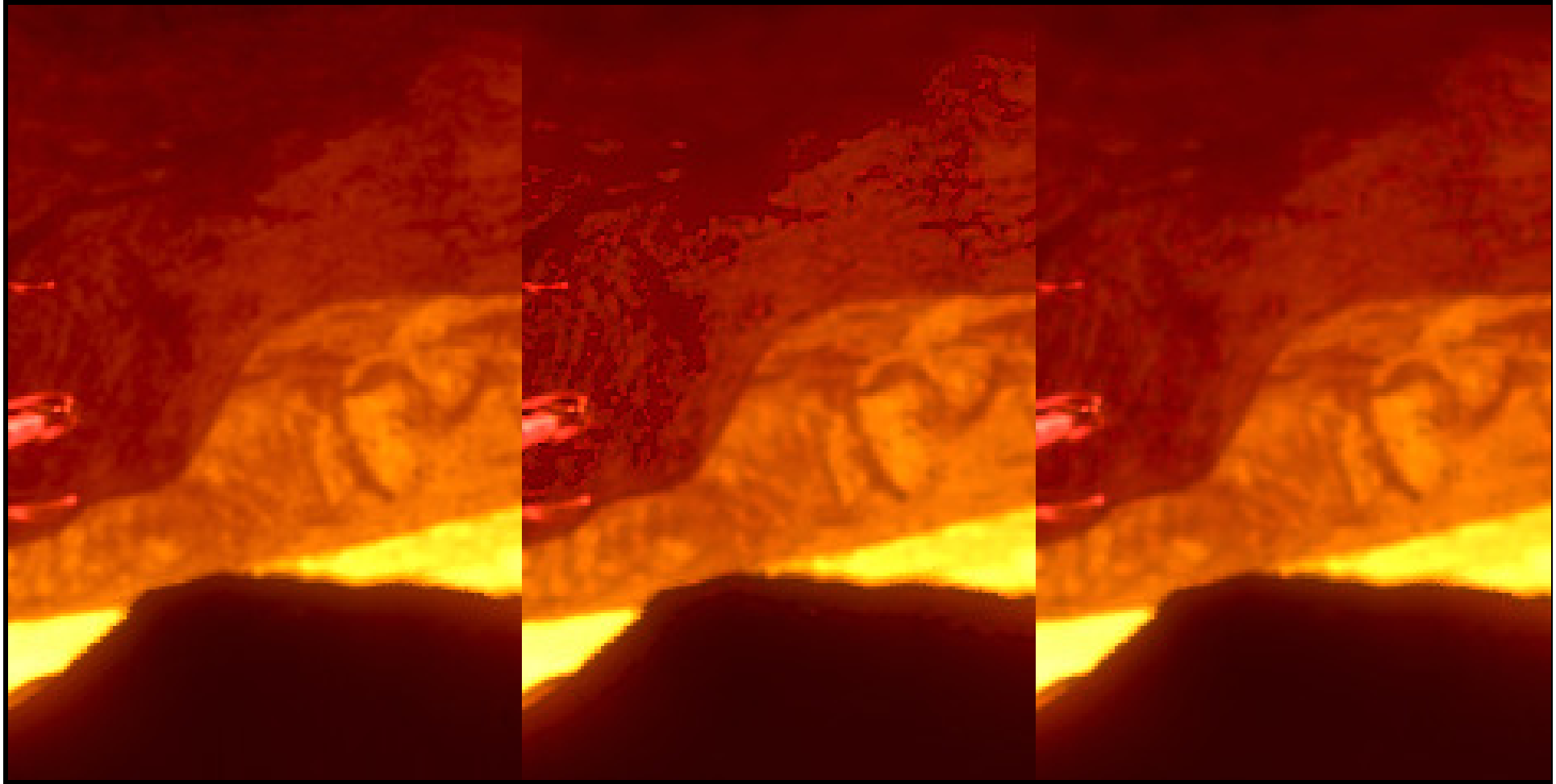
Original 4:4:4

Traditional Processing
(18752 kbps)

Luma Adjustment
(17700 kbps)

Sequence courtesy of Technicolor and the NevEx project

COMPRESSED EXAMPLES



Original 4:4:4

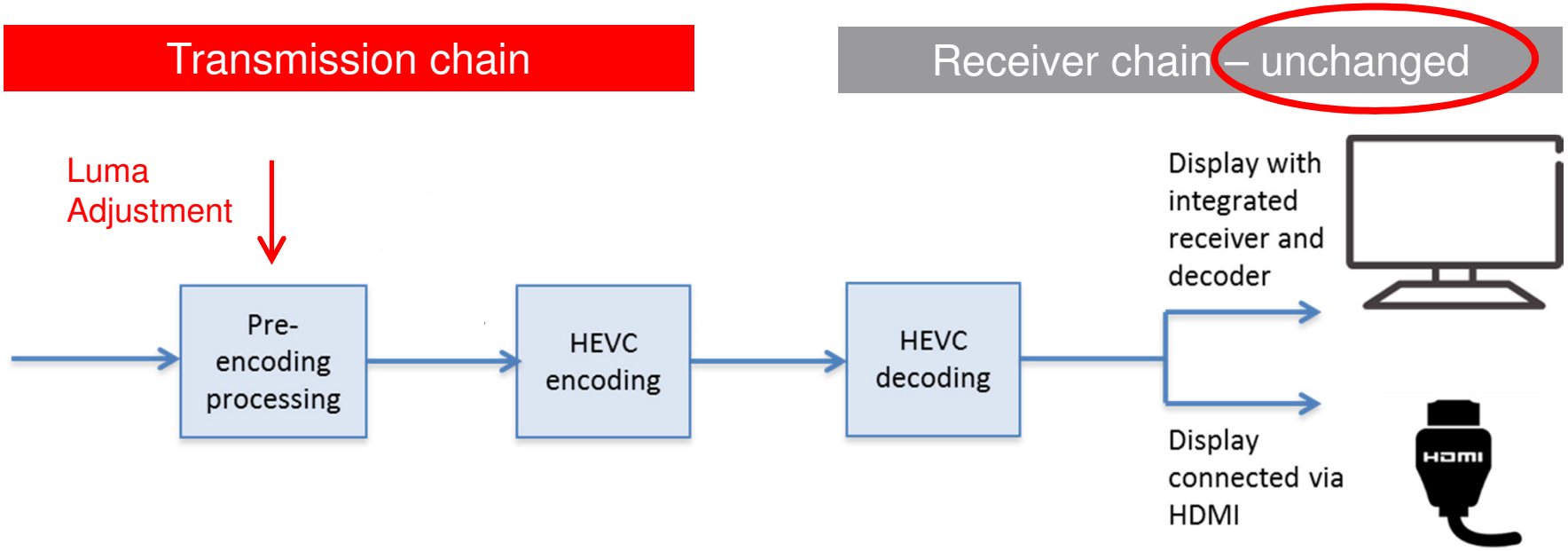
Traditional Processing
(26568 kbps)

Luma Adjustment
(26142 kbps)

Sequence courtesy of Technicolor and the NevEx project

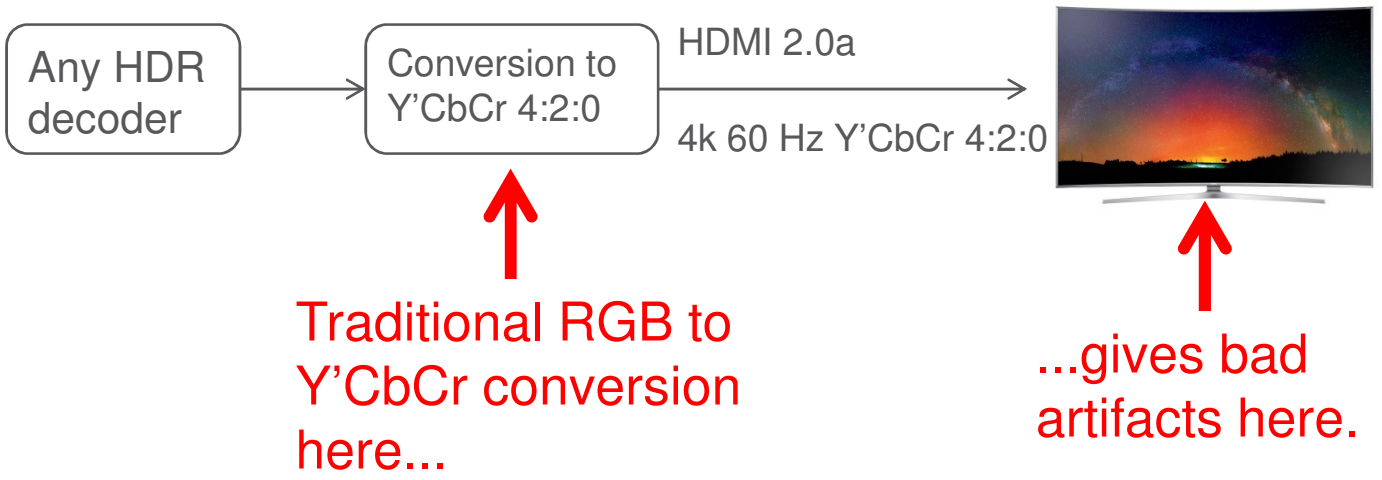
APPLICATIONS

- Encoding for “HDR10” (HEVC Main 10, ST 2084, Y’CbCr)



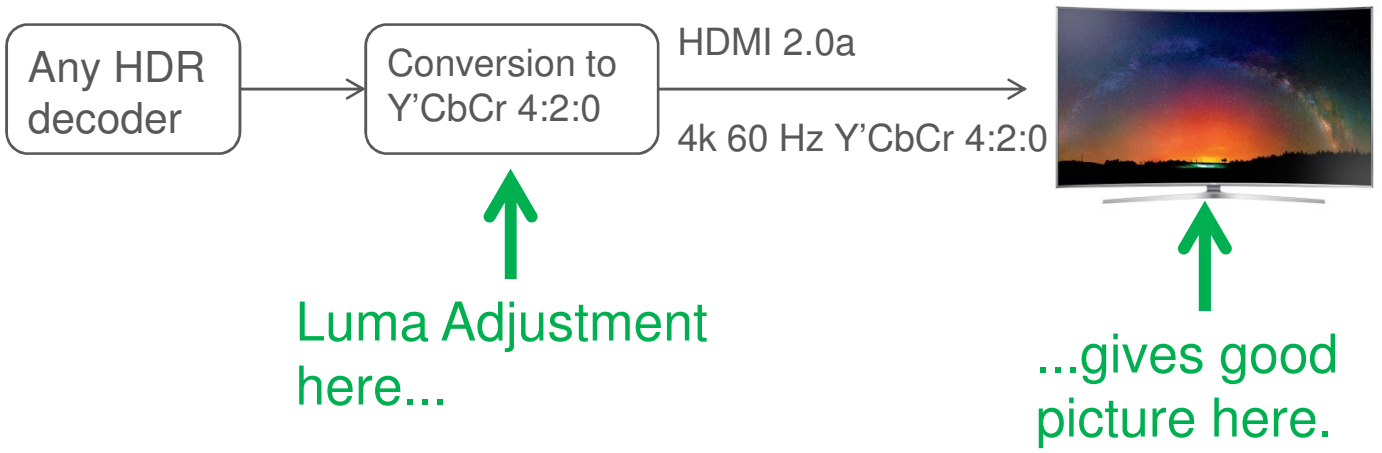
APPLICATIONS

- Conversion for display using HDMI 2.0a



APPLICATIONS

- Conversion for display using HDMI 2.0a



OTHER EOTFS

- We tried this for SMPTE ST 2084. What about other transfer functions?
- Hybrid Log Gamma (ARIB B67)

HYBRID LOG GAMMA

Original 4:4:4

Traditional subsampling
(no compression)

Luma Adjustment
(proposed method)

Sequence courtesy of Technicolor and the NevEx project

OTHER EOTFS

- We tried this for SMPTE ST 2084. What about other transfer functions?
- Hybrid Log Gamma
- What about Standard Dynamic Range (SDR) (Rec 1886)?

SDR (1886)

original 4:4:4

traditional processing

Luma Adjustment

SDR (1886)

original 4:4:4

traditional processing

Luma Adjustment

Sequence courtesy of Technicolor and the NevEx project

CONCLUSION

- Due to non-linearity of transfer function, artifacts appear during subsampling from 4:4:4 to 4:2:0
- Worse for HDR since transfer function is more non-linear
- It is possible to adjust Y’ to match luminance in each pixel
- Since Y is monotonous in Y’, binary search can be used
- Mathematical bounds further lower number of iterations
- Applications are preprocessing for compression and conversion to HDMI 2.0a.
- Some noise and blurring in 4:2:0 SDR data can be due to the processing from 4:4:4 to 4:2:0.
- MPEG anchors

ERICSSON

HDMI 2.0 AND HDMI 2.0A

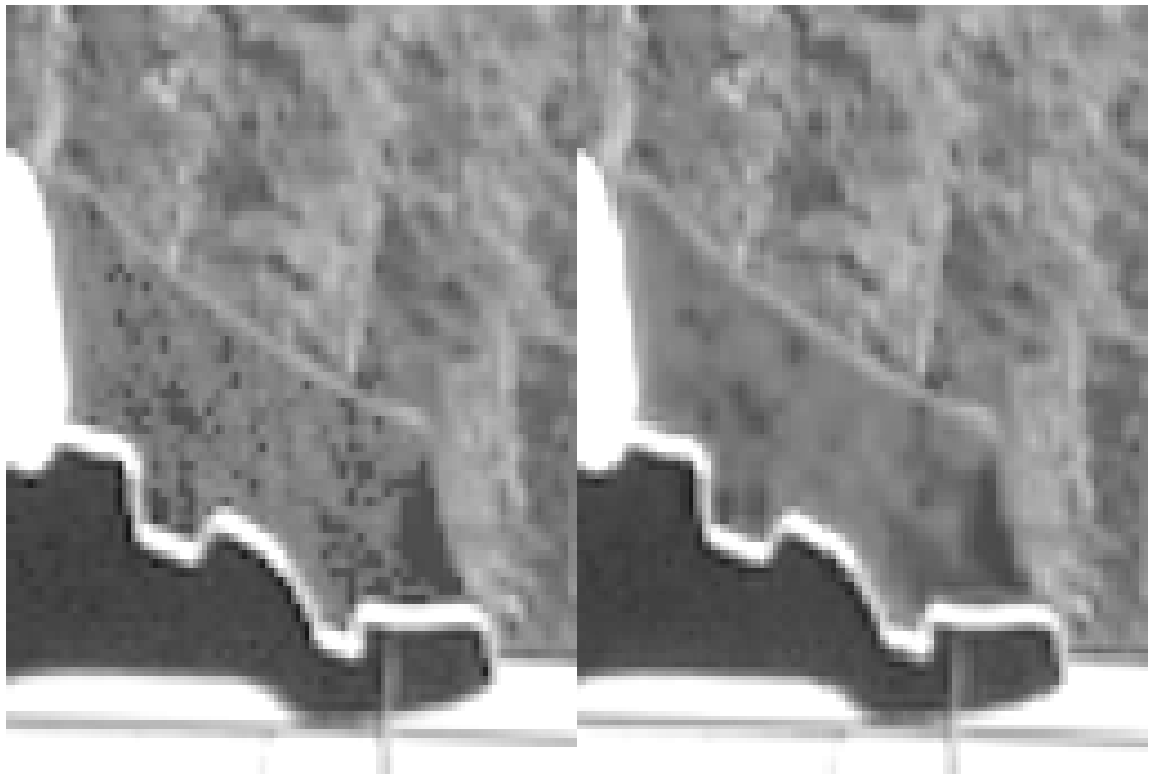
- For 4K resolutions at 50-60 Hz, 4:2:0 is necessary.
- HMDI 2.0a specifies HDR through pointing to the CEA-861.3 specification.

Overview of HDMI 2.0

What are the 4K formats supported by HDMI 2.0?

	8bit	10bit	12bit	16bit
4K@24	RGB 4:4:4	RGB 4:4:4	RGB 4:4:4 4:2:2	RGB 4:4:4
4K@25				
4K@30				
4K@50	RGB 4:4:4 4:2:0	4:2:0	4:2:2 4:2:0	4:2:0
4K@60				

Y' EASIER TO CODE



Y' traditional processing

Y' after Luma Adjustment