Lecture 15:

Image and Video Compression

Computer Graphics: Rendering, Geometry, and Image Manipulation
Stanford CS248A, Winter 2026

Recurring themes in the course

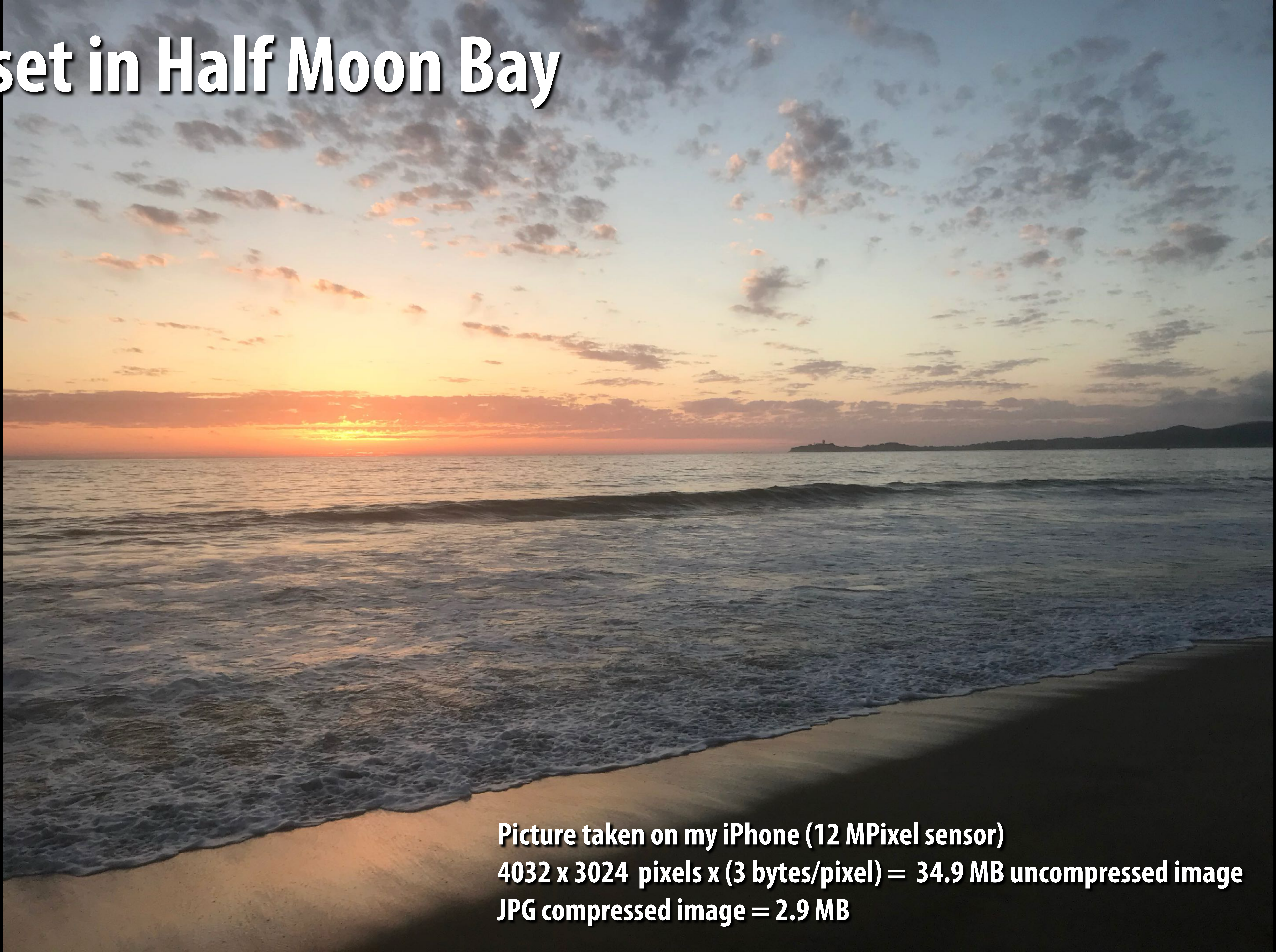
- Choosing the right representation for a task
  - e.g., choosing the right basis, choosing implicit/explicit representations of shape

- Exploiting human perception for computational efficiency
  - Approximations to the real world can be tolerable if humans do not notice

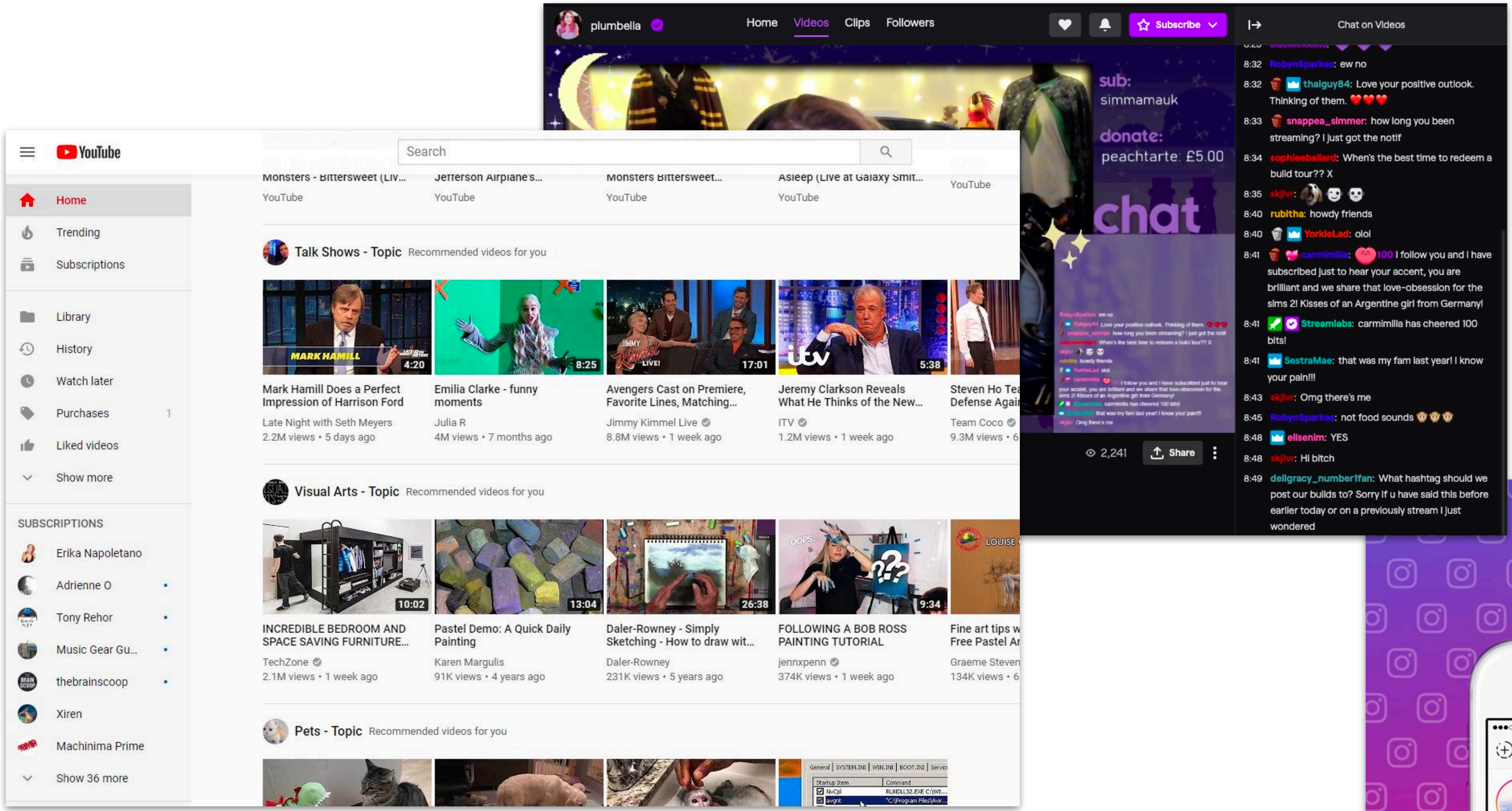
- Convolution as a useful operator
  - To remove high-frequency content from images
  - What else can we do with convolution?

Image Compression

A sunset in Half Moon Bay



Picture taken on my iPhone (12 MPixel sensor)
4032 x 3024 pixels x (3 bytes/pixel) = 34.9 MB uncompressed image
JPG compressed image = 2.9 MB



8:32 RobynSparkes: ew no
8:32 thaiguy84: Love your positive outlook. Thinking of them. ❤❤❤
8:33 snappea_simmer: how long you been streaming? I just got the notif
8:34 sophieeballard: When's the best time to redeem a build tour?? X
8:35 skjlvr:
8:40 rubitha: howdy friends
8:40 YorkieLad: olol
8:41 carmimilla: 100 I follow you and I have subscribed just to hear your accent, you are brilliant and we share that love-obsession for the sims 2! Kisses of an Argentine girl from Germany!
8:41 Streamlabs: carmimilla has cheered 100 bits!
8:41 SestraMae: that was my fam last year! I know your pain!!!
8:43 skjlvr: Omg there's me
8:45 RobynSparkes: not food sounds
8:48 ellsenim: YES
8:48 skjlvr: Hi bitch
8:49 deligracy_number1fan: What hashtag should we post our builds to? Sorry if u have said this before earlier today or on a previously stream I just wondered

Monsters - Bittersweet (Liv... YouTube
Jefferson Airplane's... YouTube
Monsters Bittersweet... YouTube
Asleep (Live at Galaxy Smit... YouTube
YouTube

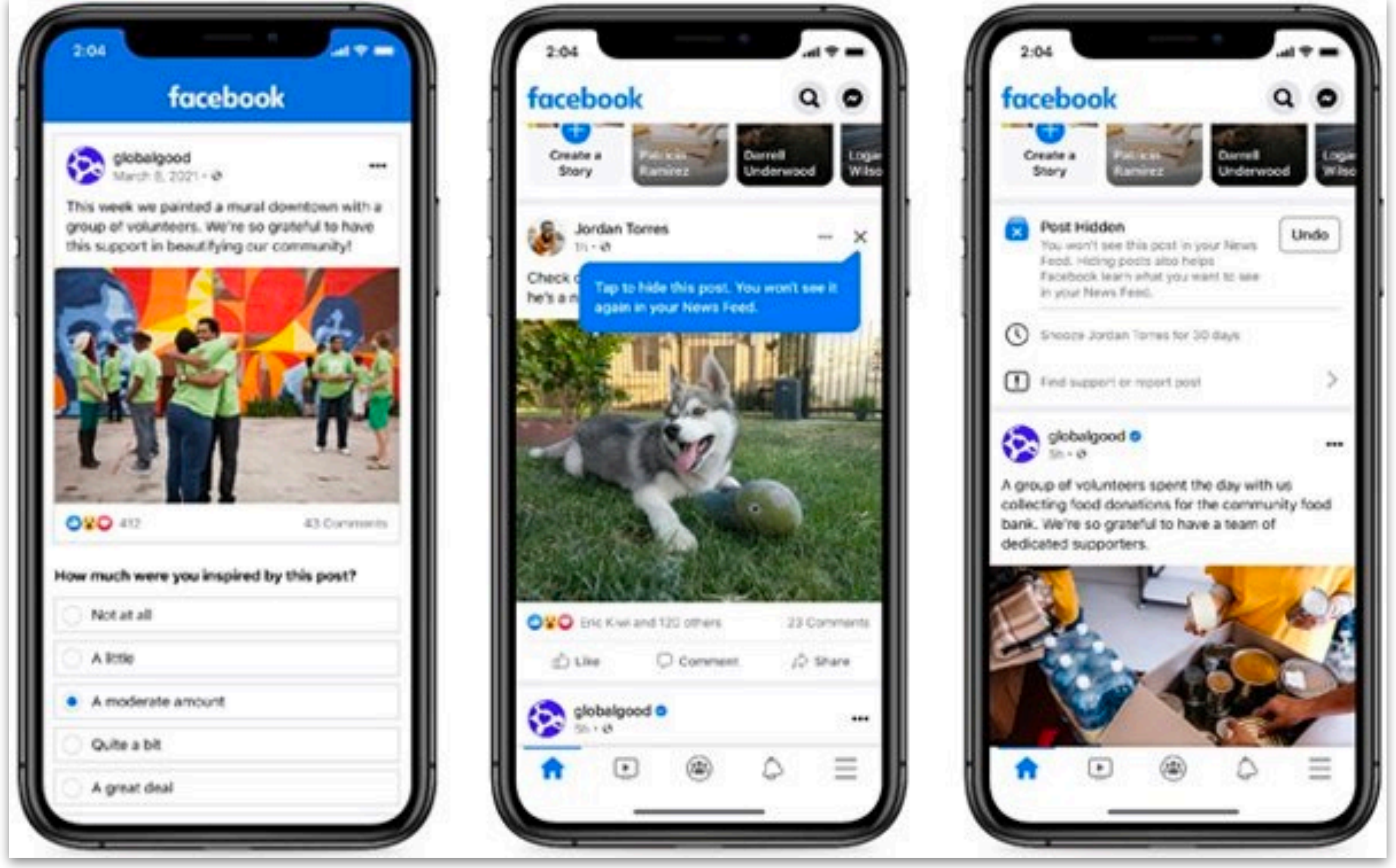
Talk Shows - Topic Recommended videos for you

Video	Channel	Views
Mark Hamill Does a Perfect Impression of Harrison Ford	Late Night with Seth Meyers	2.2M views • 5 days ago
Emilia Clarke - funny moments	Julia R	4M views • 7 months ago
Avengers Cast on Premiere, Favorite Lines, Matching...	Jimmy Kimmel Live	8.8M views • 1 week ago
Jeremy Clarkson Reveals What He Thinks of the New...	ITV	1.2M views • 1 week ago
Steven Ho Te... Defense Agai...	Team Coco	9.3M views • 6...

Visual Arts - Topic Recommended videos for you

Video	Channel	Views
INCREDIBLE BEDROOM AND SPACE SAVING FURNITURE...	TechZone	2.1M views • 1 week ago
Pastel Demo: A Quick Daily Painting	Karen Margulis	91K views • 4 years ago
Daler-Rowney - Simply Sketching - How to draw wit...	Daler-Rowney	231K views • 5 years ago
FOLLOWING A BOB ROSS PAINTING TUTORIAL	jennxpenn	374K views • 1 week ago
Fine art tips w... Free Pastel Ar...	Graeme Steven...	134K views • 6...

Pets - Topic Recommended videos for you



globalgood
March 8, 2021

This week we painted a mural downtown with a group of volunteers. We're so grateful to have this support in beautifying our community!

412 43 Comments

How much were you inspired by this post?

○ Not at all
○ A little
● A moderate amount
○ Quite a bit
○ A great deal

Create a Story
Patricia Ramirez
Darrell Underwood
Logan Wilson

Jordan Torres
1h

Tap to hide this post. You won't see it again in your News Feed.

Eric Kwi and 120 others 23 Comments
Like Comment Share

globalgood
5h

Post Hidden
You won't see this post in your News Feed. Hiding posts also helps Facebook learn what you want to see in your News Feed.
Undo

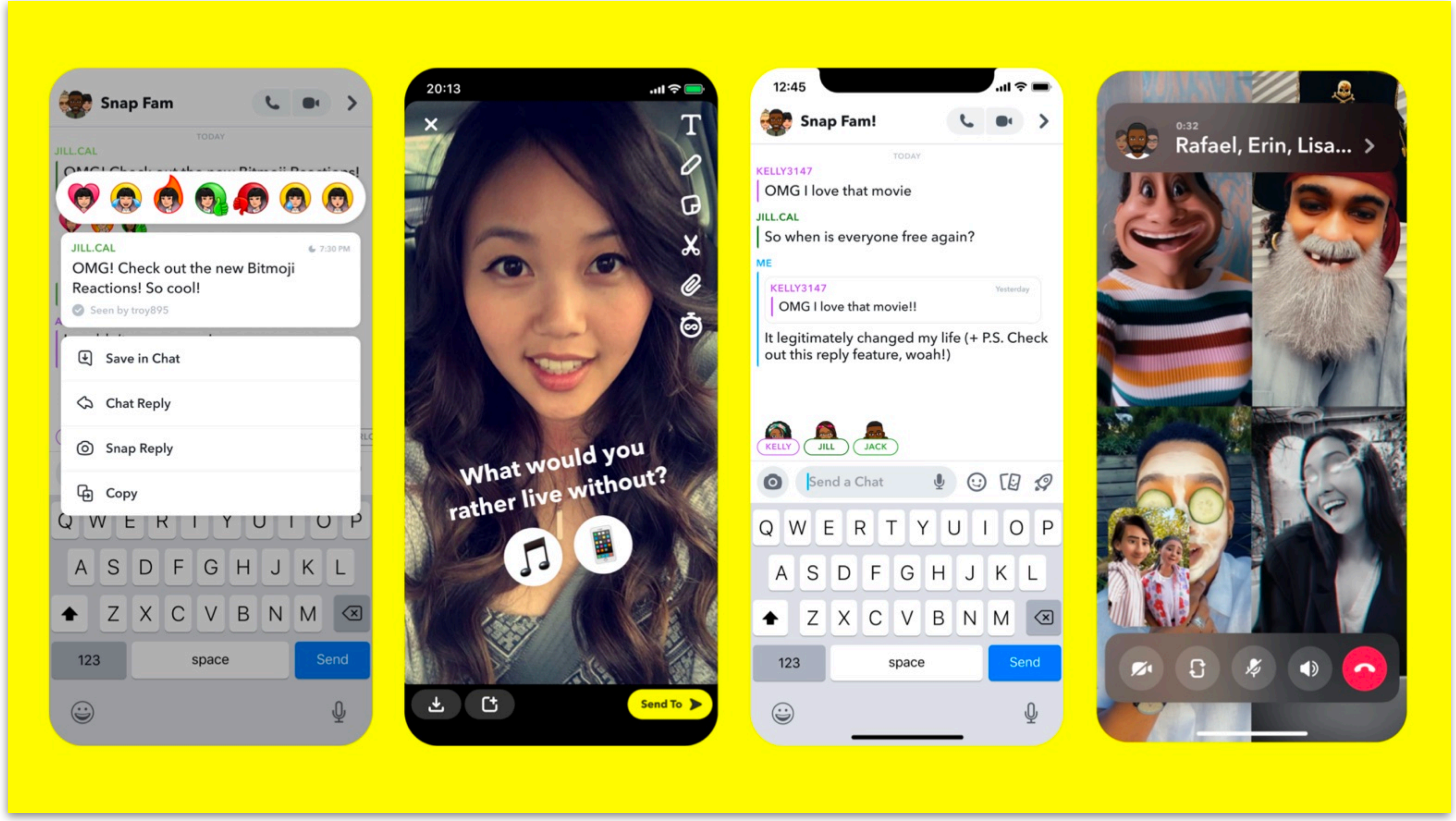
Snooze Jordan Torres for 30 days

Find support or report post

globalgood
5h

A group of volunteers spent the day with us collecting food donations for the community food bank. We're so grateful to have a team of dedicated supporters.

Instagram Stories



Snap Fam

JILL.CAL
OMG! Check out the new Bitmoji Reactions! So cool!
Seen by troy895

Save in Chat
Chat Reply
Snap Reply
Copy

What would you rather live without?

Send To

12:45
Snap Fam!

KELLY3147
OMG I love that movie

JILL.CAL
So when is everyone free again?

ME
KELLY3147
OMG I love that movie!!
It legitimately changed my life (+ P.S. Check out this reply feature, woah!)

Send a Chat

Rafael, Erin, Lisa...

Review from last class: color spaces

Last time: displays producing color

- Given a set of primary lights, each with its own spectral distribution (e.g. R,G,B display pixels):

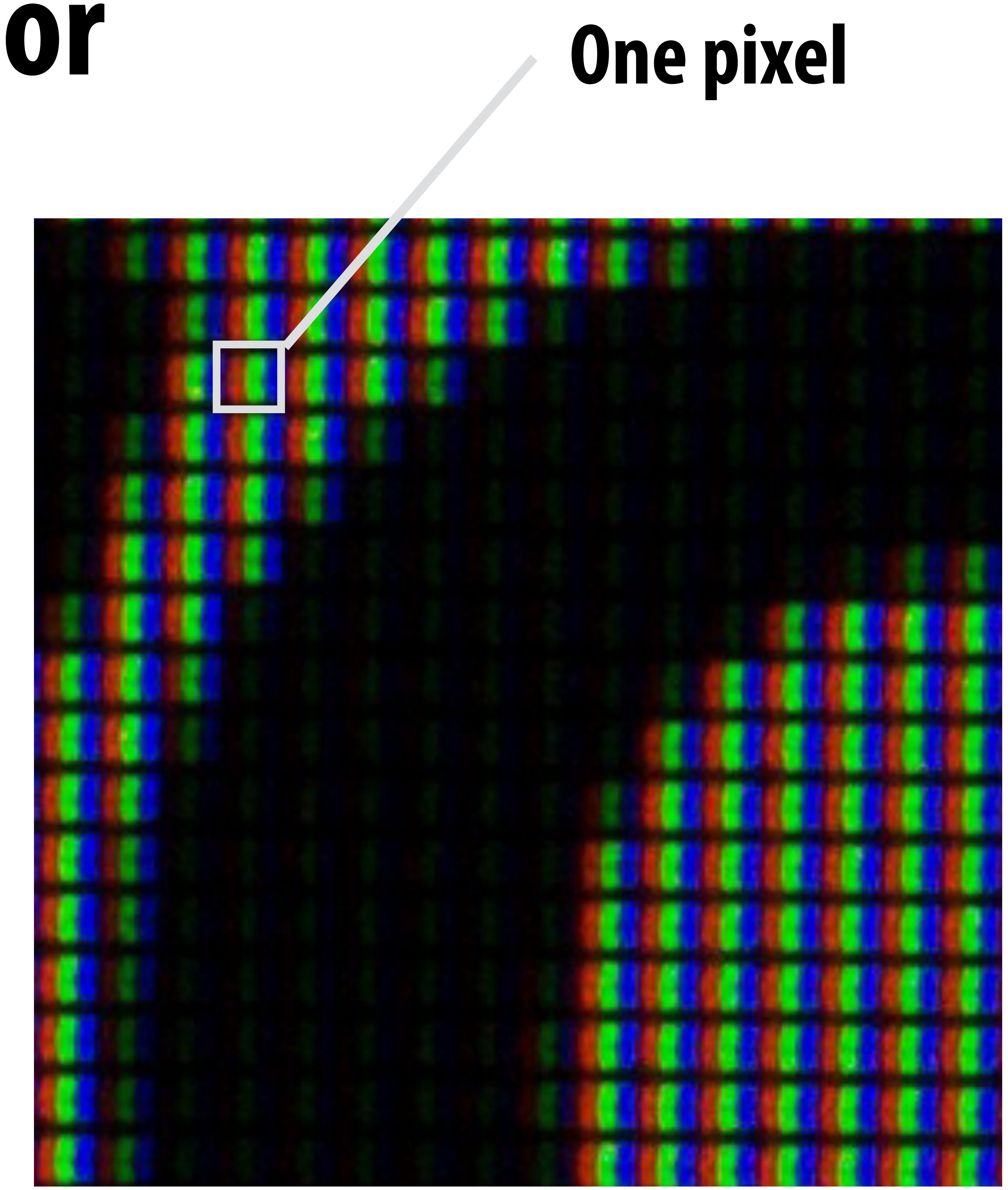
$$s_R(\lambda),\ s_G(\lambda),\ s_B(\lambda)$$

- We can adjust the brightness of these lights and add them together to produce a linear subspace of spectral distribution:

$$R\, s_R(\lambda) + G\, s_G(\lambda) + B\, s_B(\lambda)$$

- The color is now described by the scalar values:

$$R,\ G,\ B$$



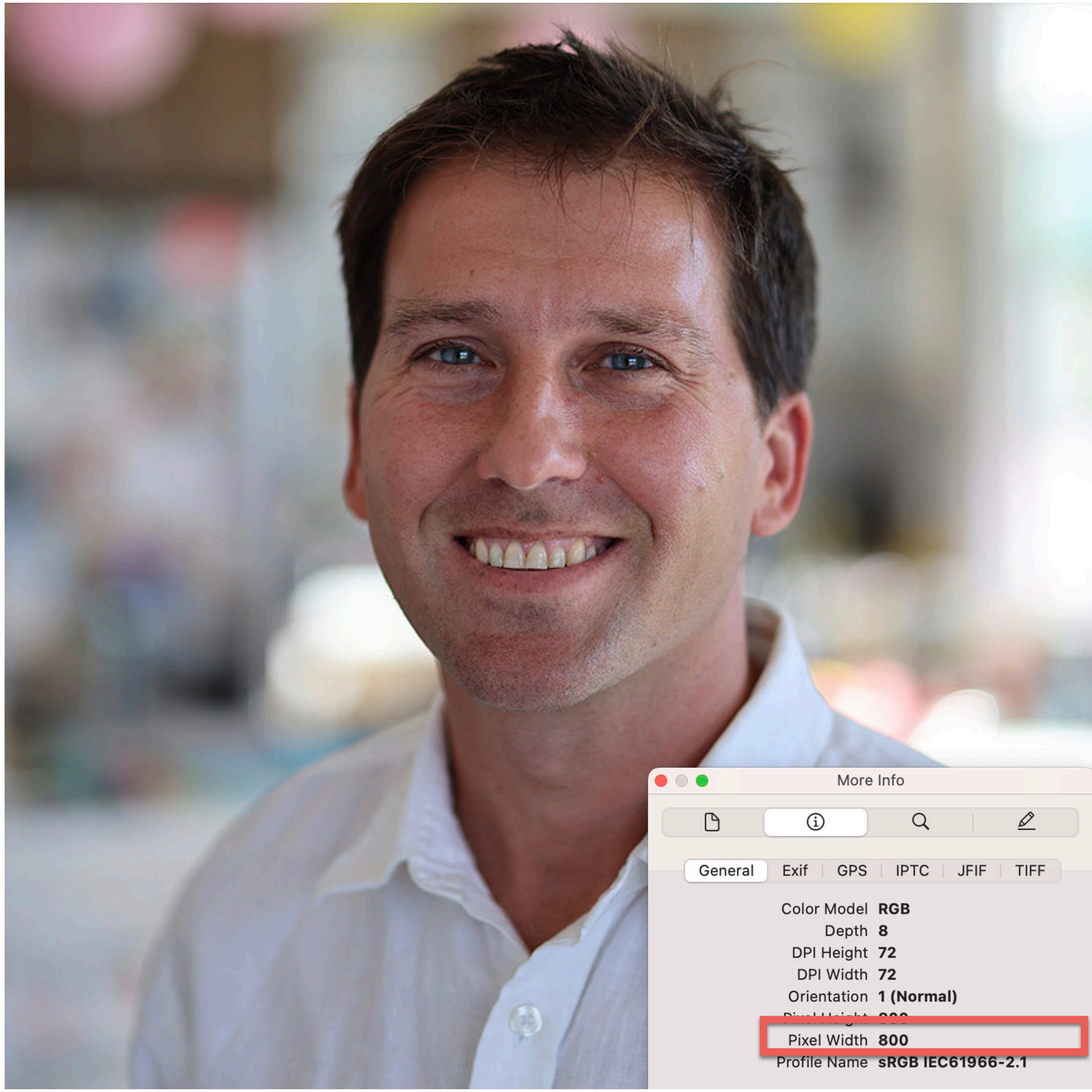
Color spaces

- Need three numbers to specify a color
  - But what three numbers?
  - A color space is an answer to this question

- Common example: color space defined by a display
  - Define colors by what R, G, B scalar values will produce them on your monitor
    - Output spectra $s = rR + gG + bB$ for some display primary spectra r, g, b
  - This a device dependent representation of color: if I choose R,G,B by looking at my display and send those values to you, you may not see the same color on your display (which might have different primaries, etc.)

Standard color spaces

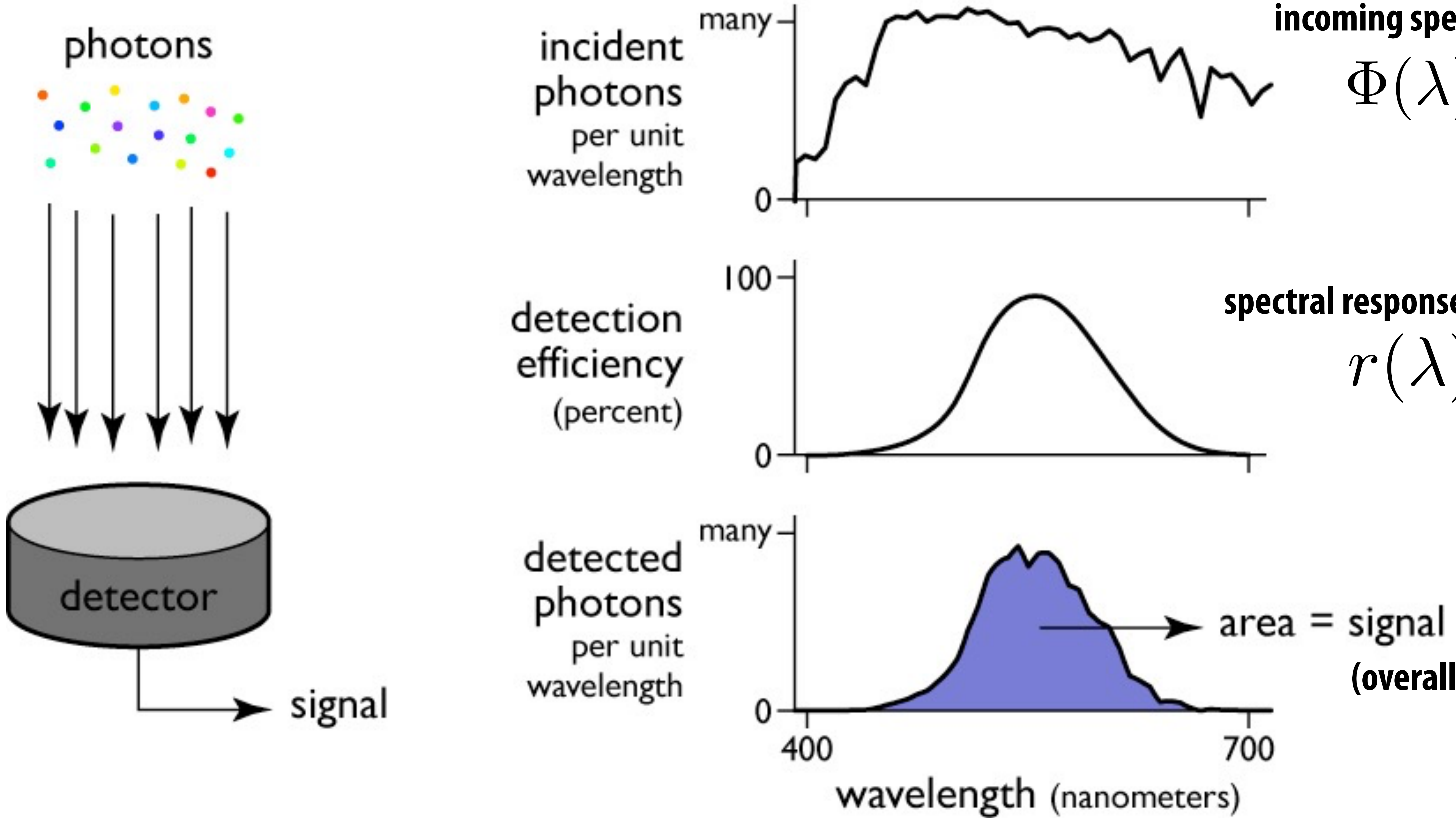
- Standardized RGB (sRGB)
  - Makes a particular monitor's primaries the RGB standard
  - Other color devices simulate that monitor by calibration
  - sRGB is usable as an interchange color space; widely adopted today



```
 Color Model  RGB
       Depth  8
  DPI Height  72
   DPI Width  72
 Orientation  1 (Normal)
Pixel Height  800
 Pixel Width  800
Profile Name  sRGB IEC61966-2.1
```

Review from last time: detectors

Sensor's response is proportional to amount of light arriving at sensor



incoming spectrum

$\Phi(\lambda)$

spectral response function

$r(\lambda)$

(overall response)

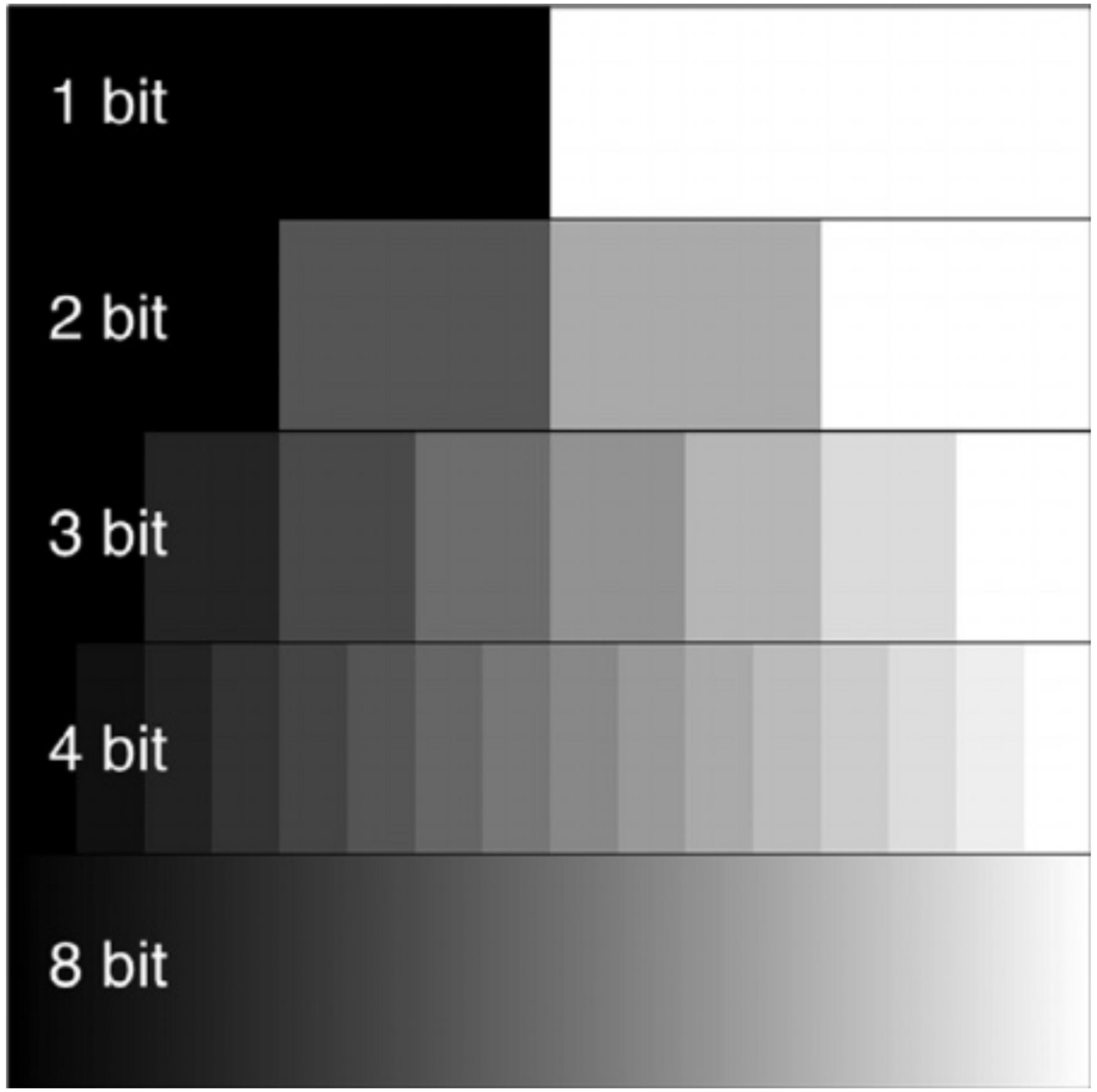
$$R = \int_\lambda \Phi(\lambda) r(\lambda) d\lambda$$

Figure credit: Steve Marschner

Encoding measurements of light

Encoding numbers

- More bits → can represent more unique numbers
- 8 bits → 256 unique numbers (0-255)



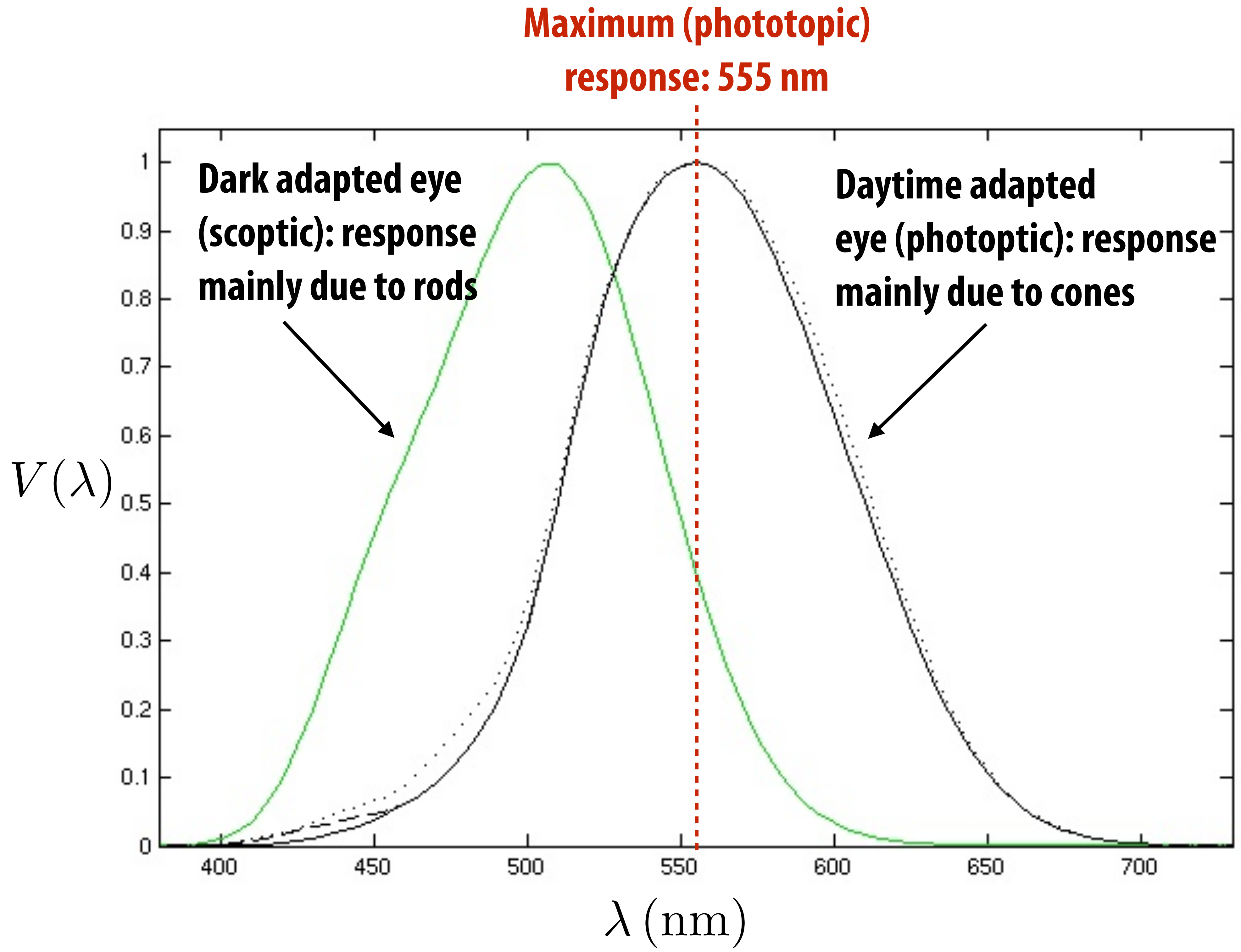
[Credit: lambert and waters]

Luminance (brightness)

Product of radiance and the eye's luminous efficiency

$$Y = \int \Phi(\lambda) V(\lambda)\, \mathrm{d}\lambda$$

- **Luminous efficiency is measure of how bright light at a given wavelength is perceived by a human (due to the eye's response to light at that wavelength)**



- **How to measure the eye's response curve $V(\lambda)$?**
  - **Adjust power of monochromatic light source of wavelength λ until it matches the brightness of reference 555 nm source (photopic case)**
  - **Notice: the sensitivity of photopic eye is maximized at ~ 555 nm**

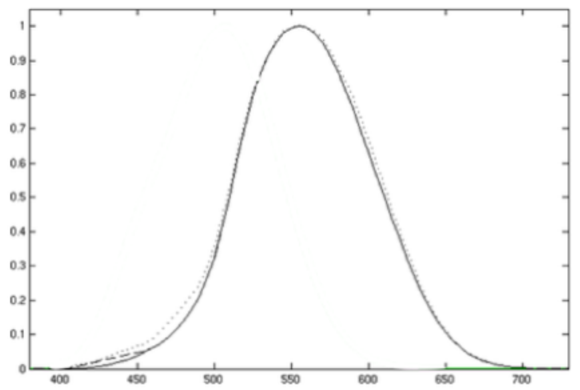
https://upload.wikimedia.org/wikipedia/commons/a/a0/Luminosity.png

Lightness (perceived brightness) aka "luma"

$$\text{Lightness } (L^*) \xleftarrow{?} \text{Luminance } (Y) = \int_{\lambda}$$

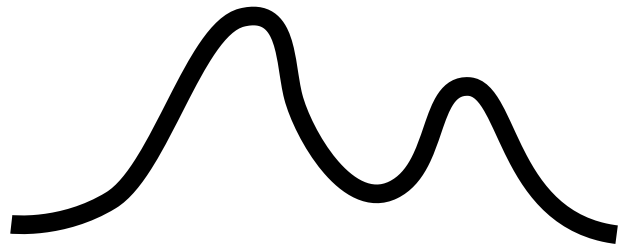
(Perceived by brain)

(Response of eye)



Spectral sensitivity of eye
(eye's response curve)

$*$



Radiance
(energy spectrum
from scene)

Dark adapted eye: $L^* \propto Y^{0.4}$

Bright adapted eye: $L^* \propto Y^{0.5}$

In a dark room, you turn on a light with luminance: Y_1

You turn on a second light that is identical to the first. Total output is now: $Y_2 = 2Y_1$

Total output appears $2^{0.4} = 1.319$ times brighter to dark-adapted human

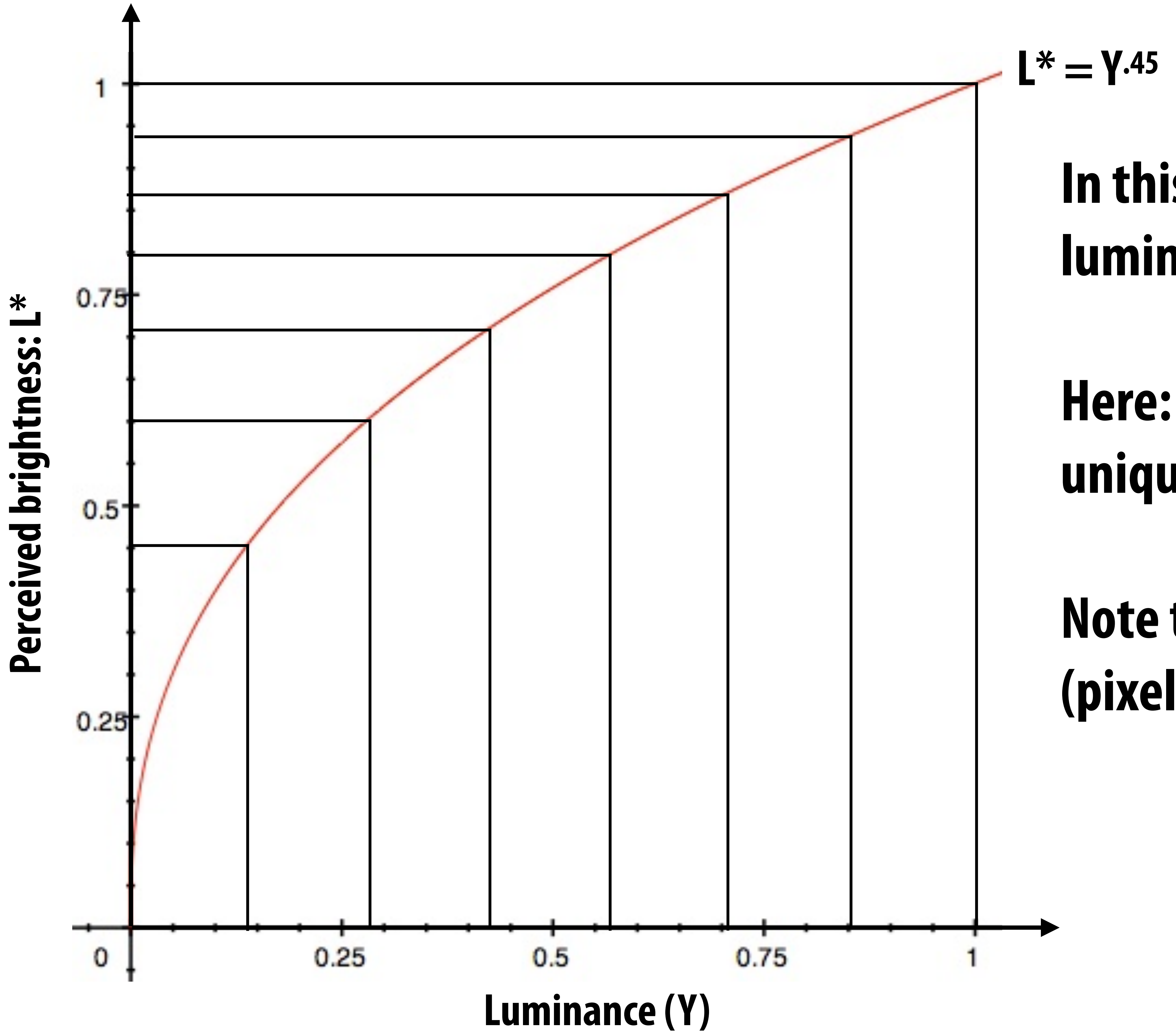
Note: Lightness (L*) is often referred to as luma (Y')

Idea 1:

- **What is the most efficient way to encode intensity values as a byte?**

- **Idea: encode based on how the brain *perceives brightness* (lightness)*,* not based on the response of eye**

Consider an image with pixel values encoding luminance (linear in energy hitting sensor)



In this visualization: A pixel can represent 8 unique luminance values (3-bits/pixel)

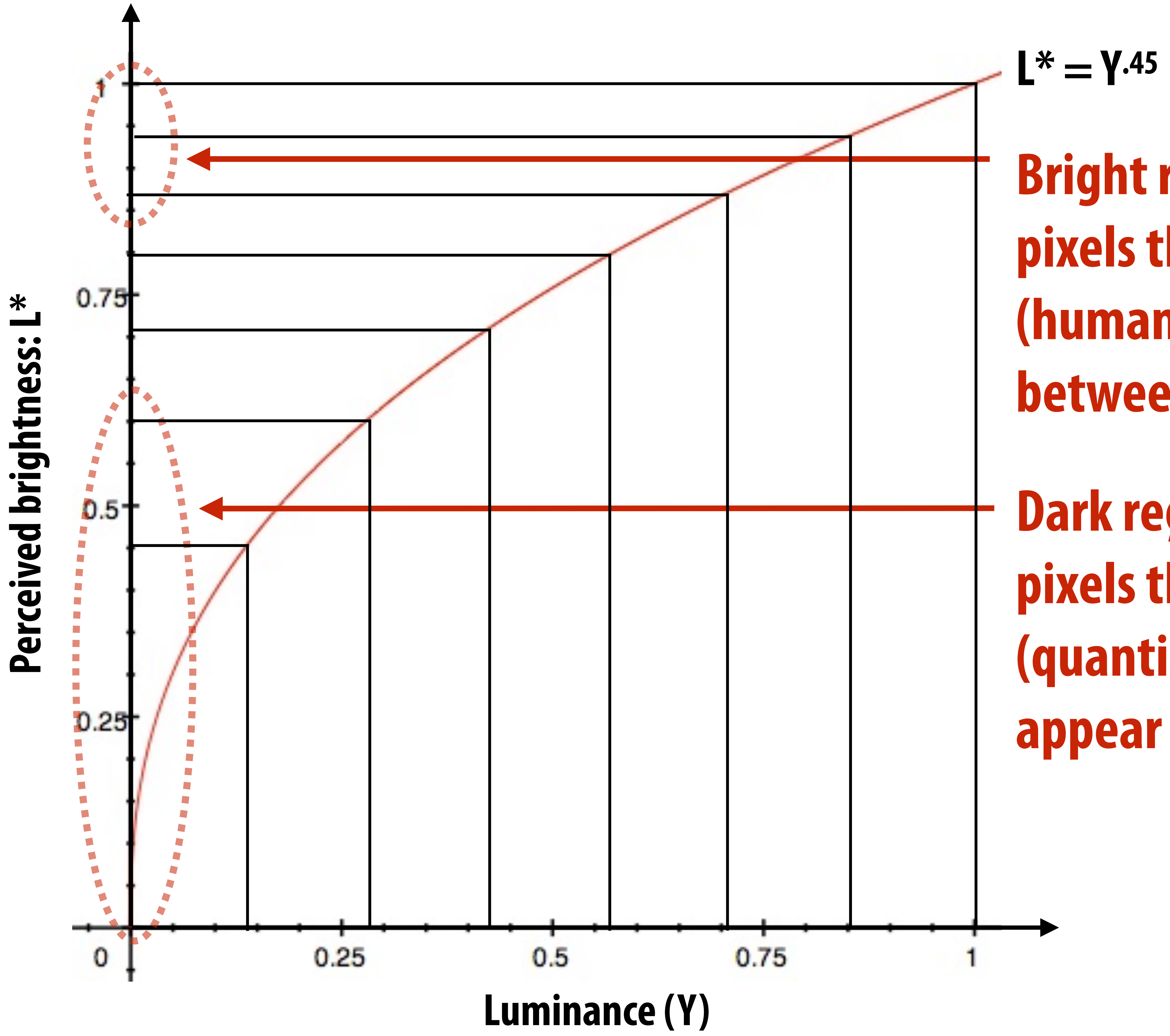
Here: lines indicate luminance associated with each unique pixel value

Note that "spacing" of pixel values is linear in luminance (pixel value encode equally spaced sensor responses)

Problem: quantization error

Many common image formats store 8 bits per channel (256 unique values)

Insufficient precision to represent brightness in darker regions of image



Bright regions of image: perceived difference between pixels that differ by one step in luminance is small! (human may not even be able to perceive difference between pixels that differ by one step in luminance!)

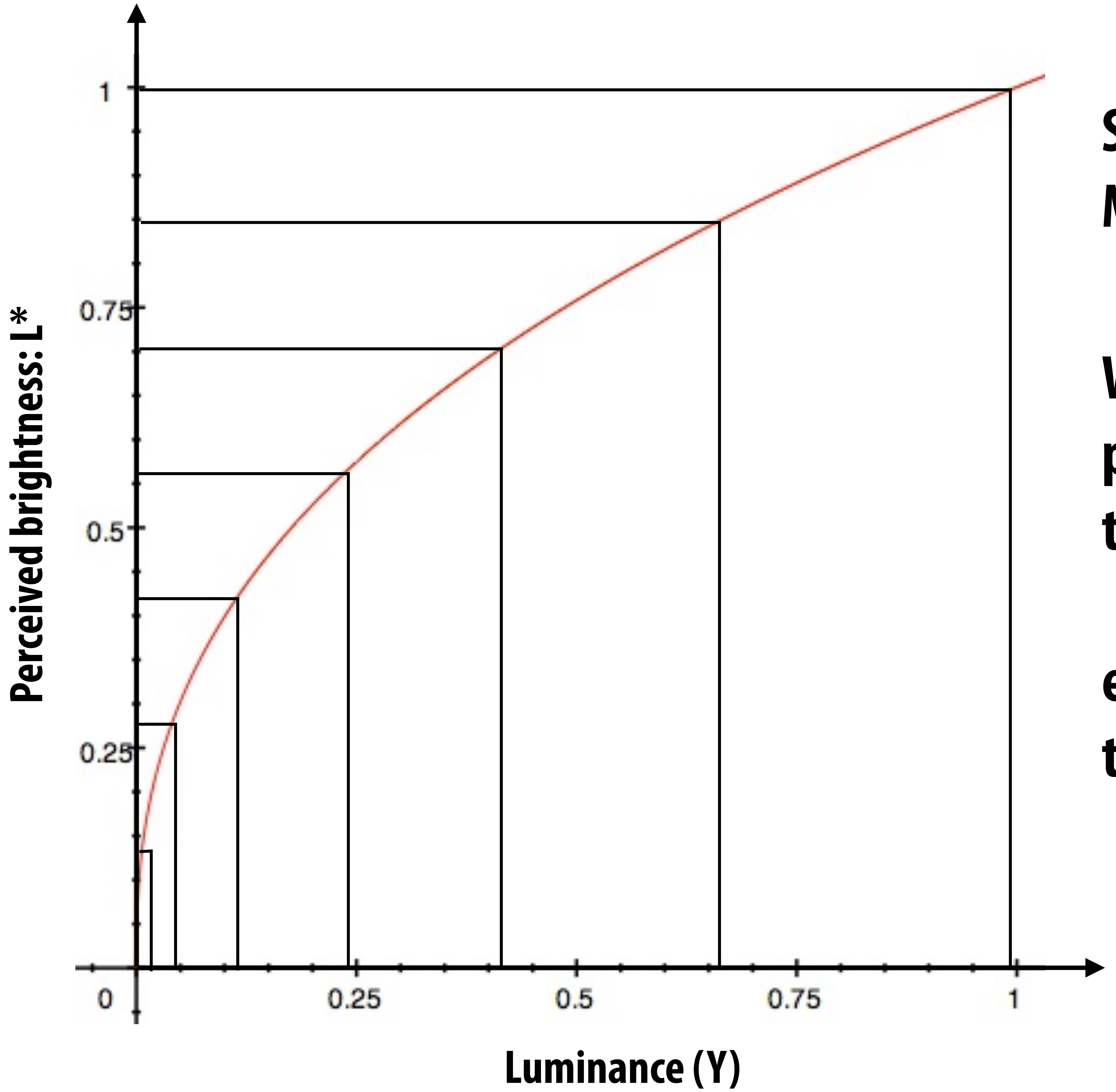
Dark regions of image: perceived difference between pixels that differ by one step in luminance is large! (quantization error: gradients in luminance will not appear smooth.)

Rule of thumb: human eye cannot differentiate <1% differences in luminance

Store lightness, not luminance

Idea: distribute representable pixel values evenly with respect to lightness (perceived brightness), not evenly in luminance

(make more efficient use of available bits)



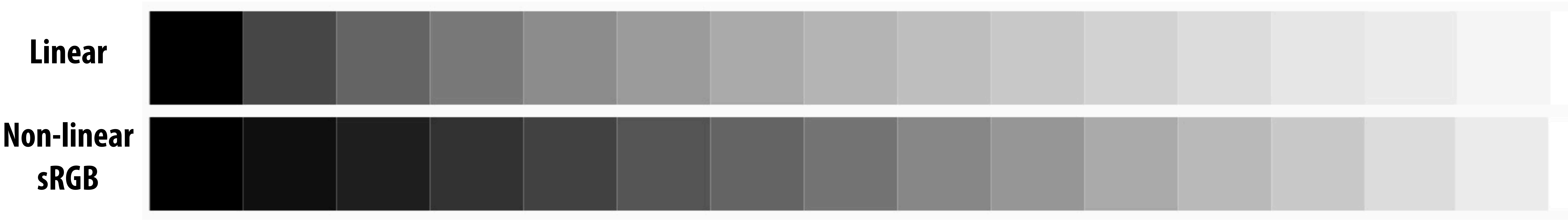
Solution: pixel stores $Y^{0.45}$

Must compute $(\text{pixel\_value})^{2.2}$ prior to display on LCD

Warning: must take caution with subsequent pixel processing operations once pixels are encoded in a space that is not linear in luminance.

e.g., When adding images should you add pixel values that are encoded as lightness or as luminance?

Linear vs. non-linear spacing of grays



Care has to be taken when processing images that are encoded in non-linear spaces.

Example: consider "adding" two images:
Want to convert to linear encoding, then perform addition, then convert back to non-linear space.

Image credit: Krita manual (https://docs.krita.org/)

Idea 2:

- Chrominance ("chroma") subsampling

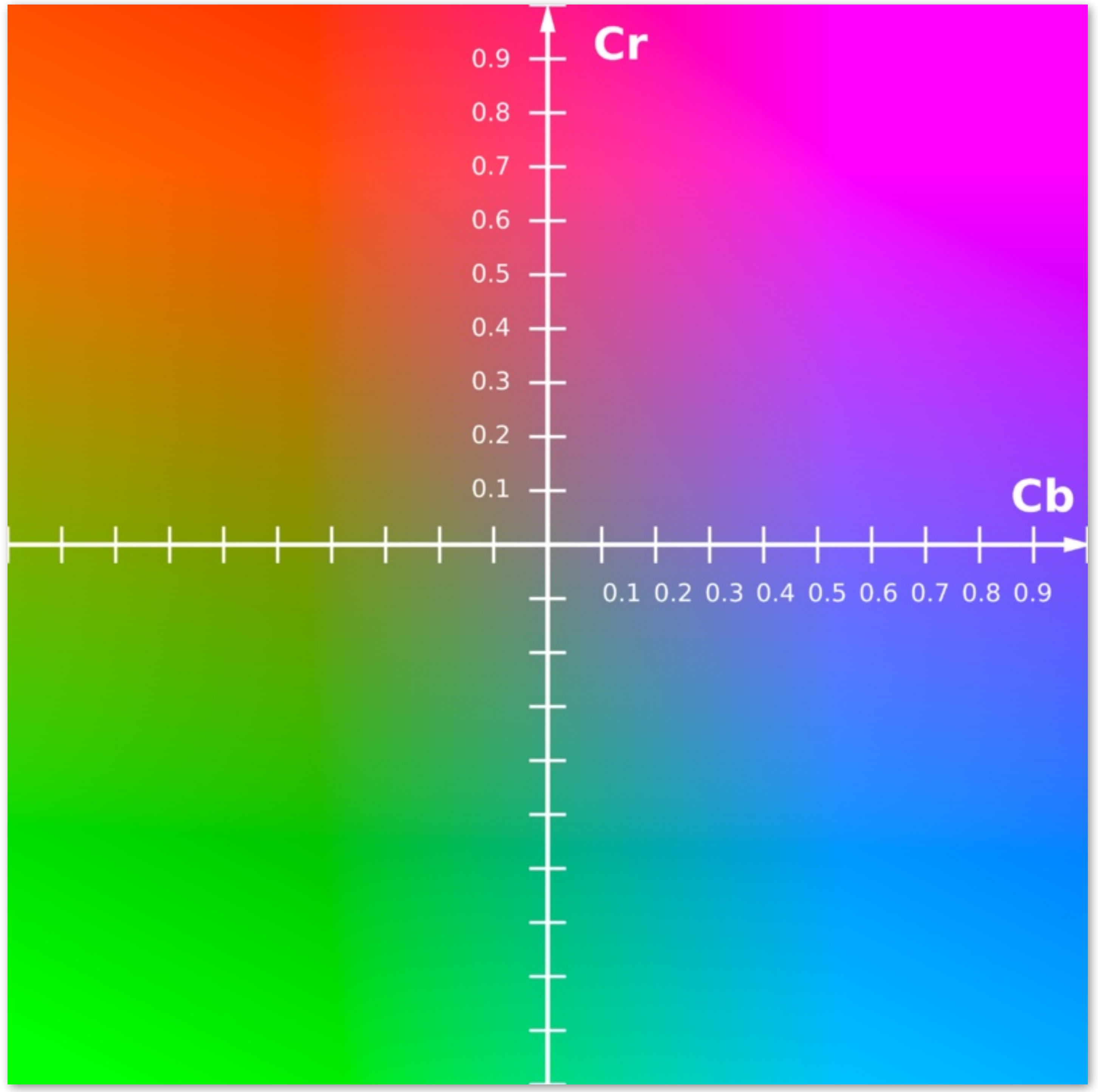
- The human visual system is less sensitive to detail in chromaticity than in luminance
    - So it is sufficient to sample chroma more sparsely in space

Y'CbCr color space

Y' = luma: perceived luminance (non-linear)

Cb = blue-yellow deviation from gray

Cr = red-cyan deviation from gray



CbCr plane at a fixed value of Y'

Non-linear RGB
(primed notation indicates
perceptual (non-linear) space)

Conversion from R'G'B' to Y'CbCr:

$$
\begin{aligned}
Y' &= 16 + \frac{65.738 \cdot R'_D}{256} + \frac{129.057 \cdot G'_D}{256} + \frac{25.064 \cdot B'_D}{256} \\
C_B &= 128 + \frac{-37.945 \cdot R'_D}{256} - \frac{74.494 \cdot G'_D}{256} + \frac{112.439 \cdot B'_D}{256} \\
C_R &= 128 + \frac{112.439 \cdot R'_D}{256} - \frac{94.154 \cdot G'_D}{256} - \frac{18.285 \cdot B'_D}{256}
\end{aligned}
$$

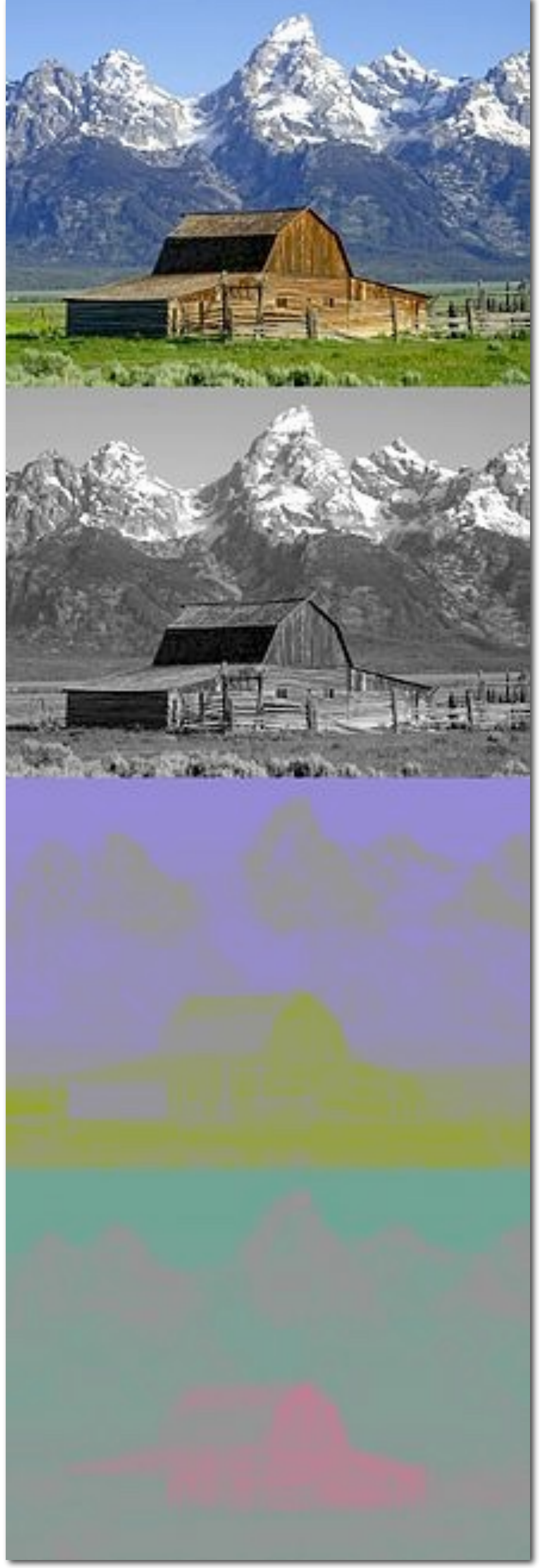
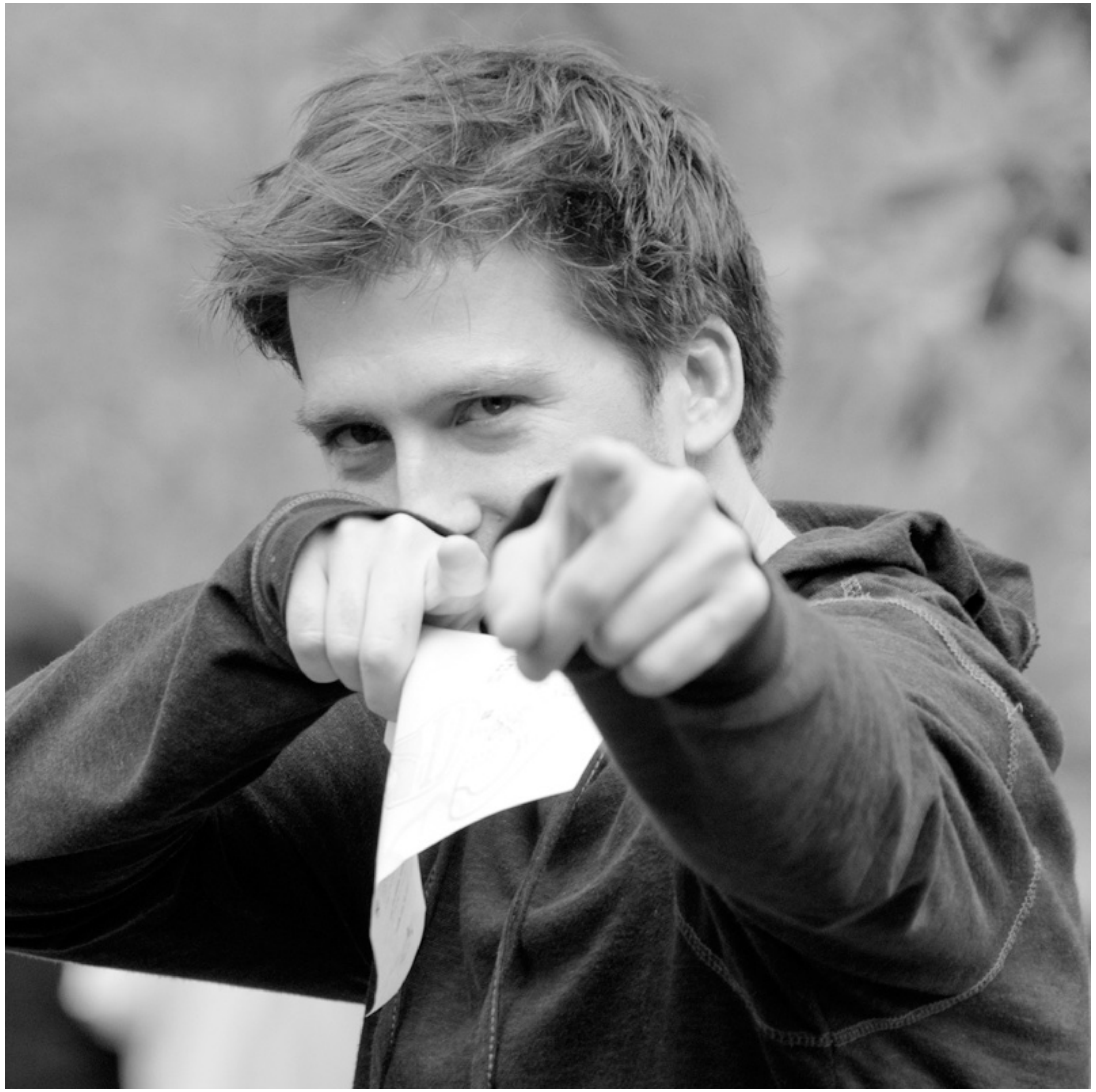


Image credit: Wikipedia

Example: compression in Y’CbCr

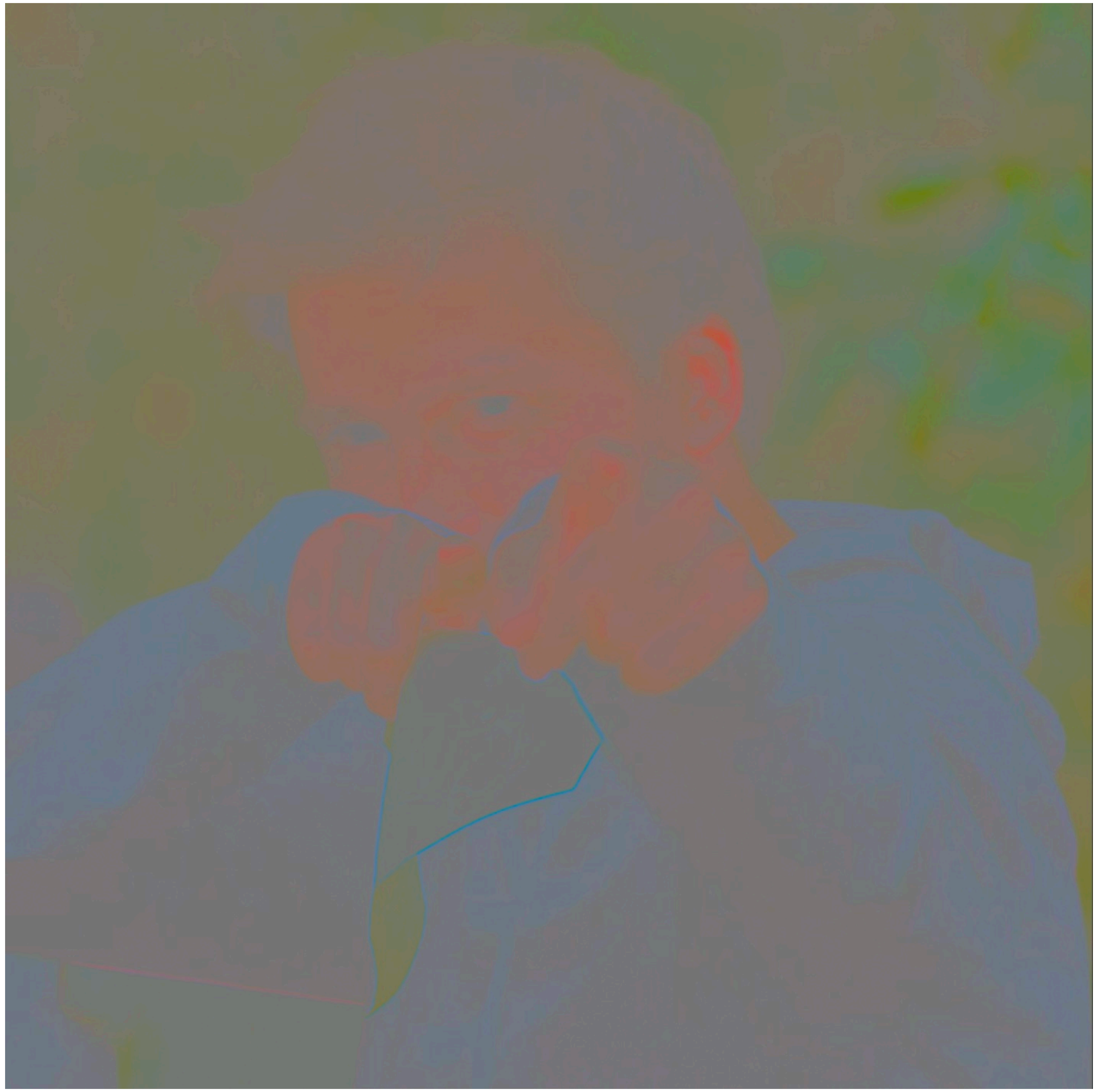
Original picture of Kayvon

Example: compression in Y'CbCr



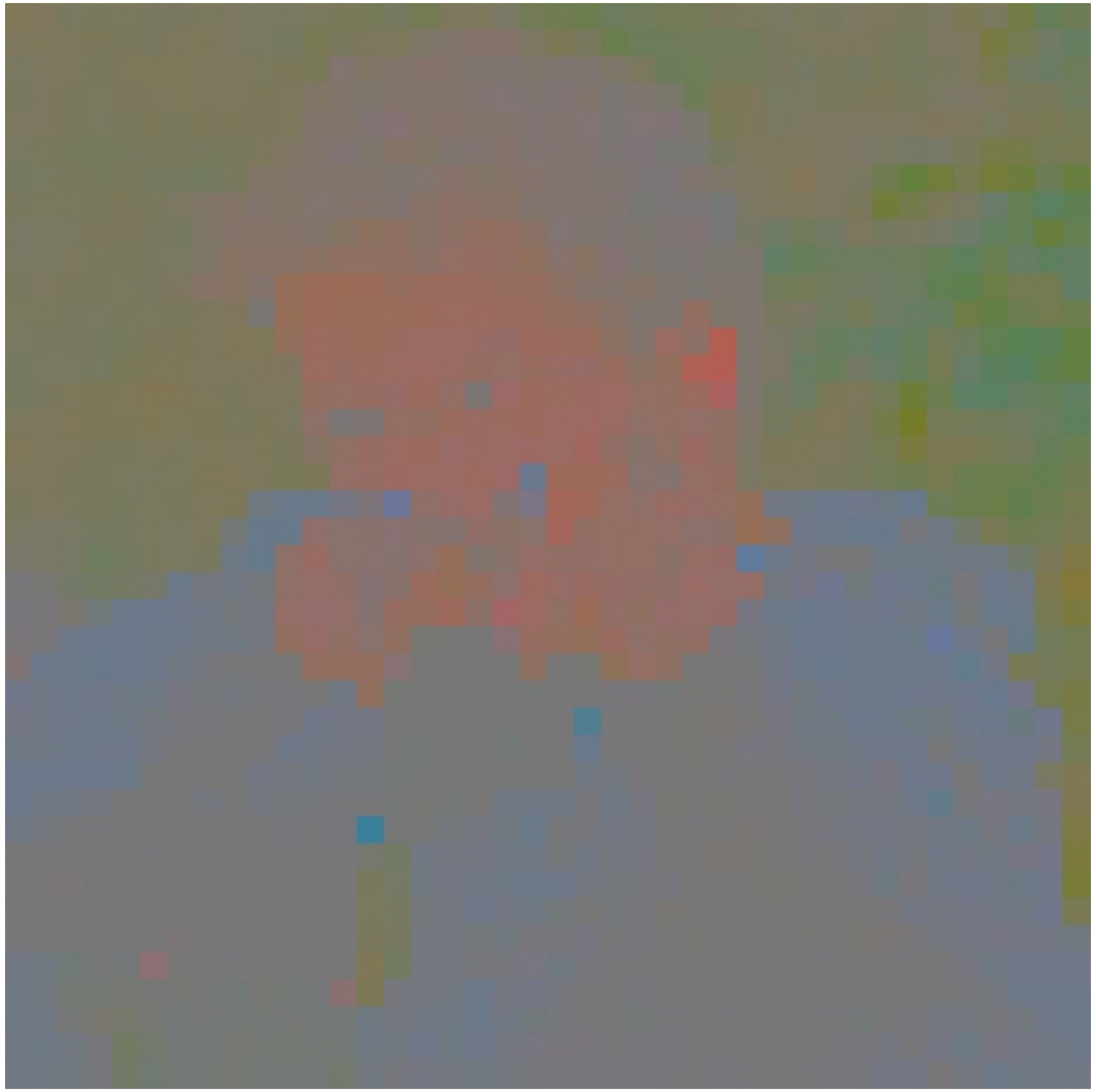
Full resolution sampling of luma (Y')

Example: compression in Y’CbCr



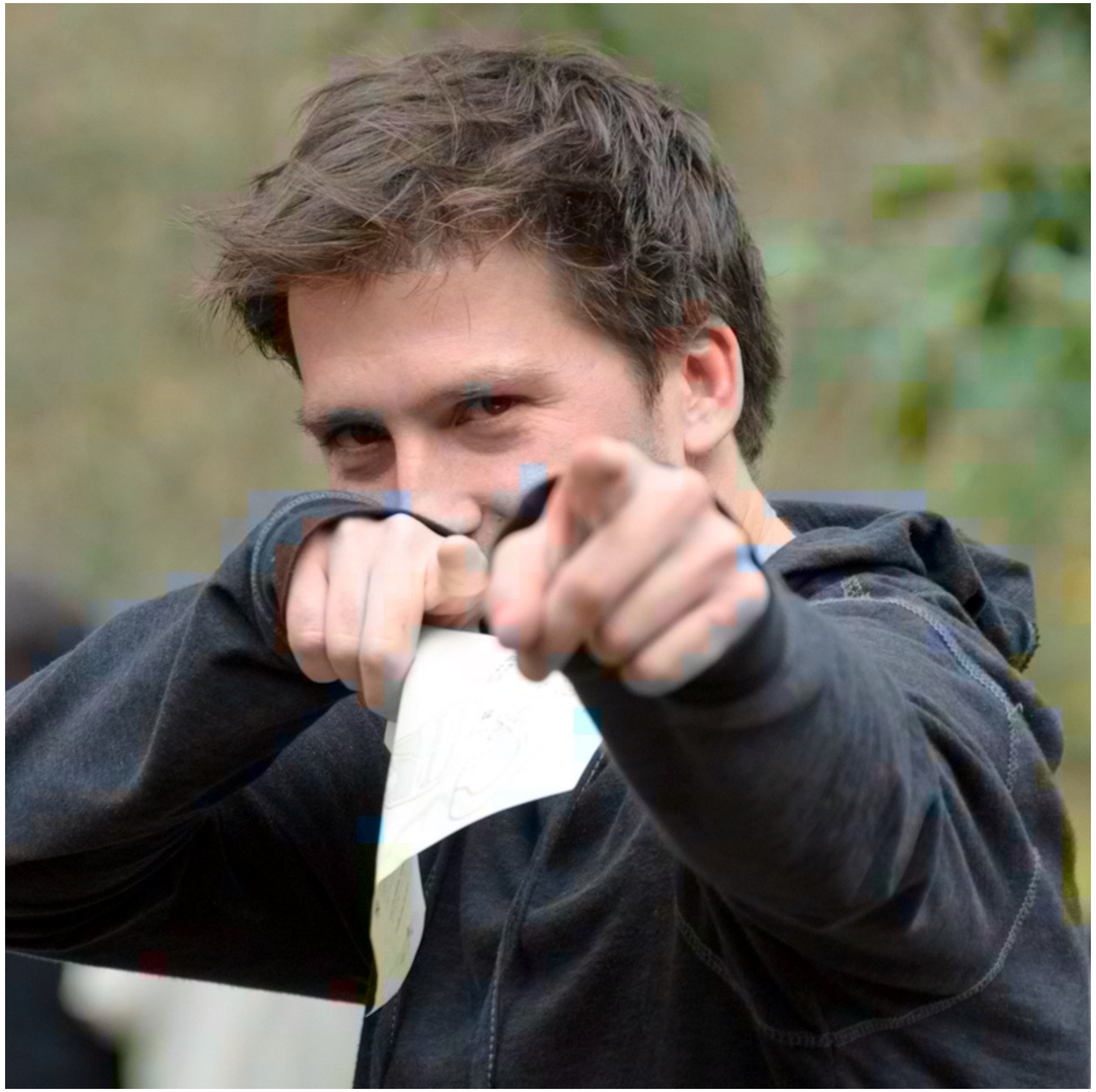
Full resolution sampling of CbCr
(Visualized with constant Y’)

Example: compression in Y’CbCr



Contents of CbCr color channels downsampled by a factor of 20 in each dimension (400x reduction in number of samples)

Example: compression in Y’CbCr



Reconstructed result
(looks pretty good)

Chroma subsampling

Y'CbCr is an efficient representation for storage (and transmission) because Y' can be stored at higher resolution than CbCr without significant loss in perceived visual quality

Y'_{00} Cb_{00} Cr_{00}	Y'_{10}	Y'_{20} Cb_{20} Cr_{20}	Y'_{30}
Y'_{01} Cb_{01} Cr_{01}	Y'_{11}	Y'_{21} Cb_{21} Cr_{21}	Y'_{31}

4:2:2 representation:

Store Y' at full resolution
Store Cb, Cr at full vertical resolution, but only half horizontal resolution

X:Y:Z notation:
X = width of block
Y = number of chroma samples in first row
Z = number of chroma samples in second row

Y'_{00} Cb_{00} Cr_{00}	Y'_{10}	Y'_{20} Cb_{20} Cr_{20}	Y'_{30}
Y'_{01}	Y'_{11}	Y'_{21}	Y'_{31}

4:2:0 representation:

Store Y' at full resolution
Store Cb, Cr at half resolution in both dimensions

Real-world 4:2:0 examples:
most JPG images and H.264 video

Idea 3:

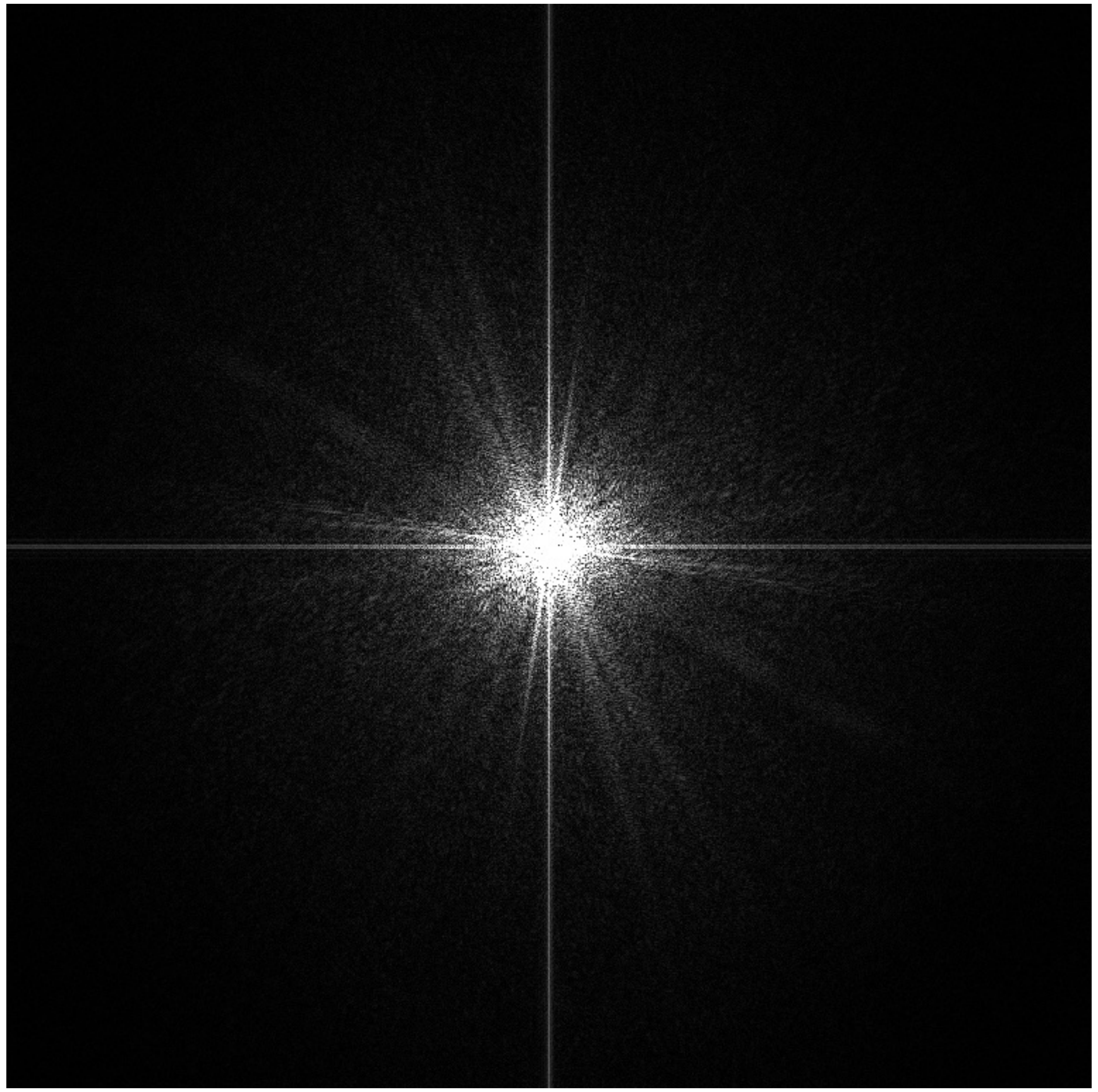
- Low frequency content is predominant in the real world

- The human visual system is less sensitive to high frequency sources of error in images

- So a good compression scheme needs to accurately represent lower frequencies, but it can be acceptable to sacrifice accuracy in representing higher frequencies

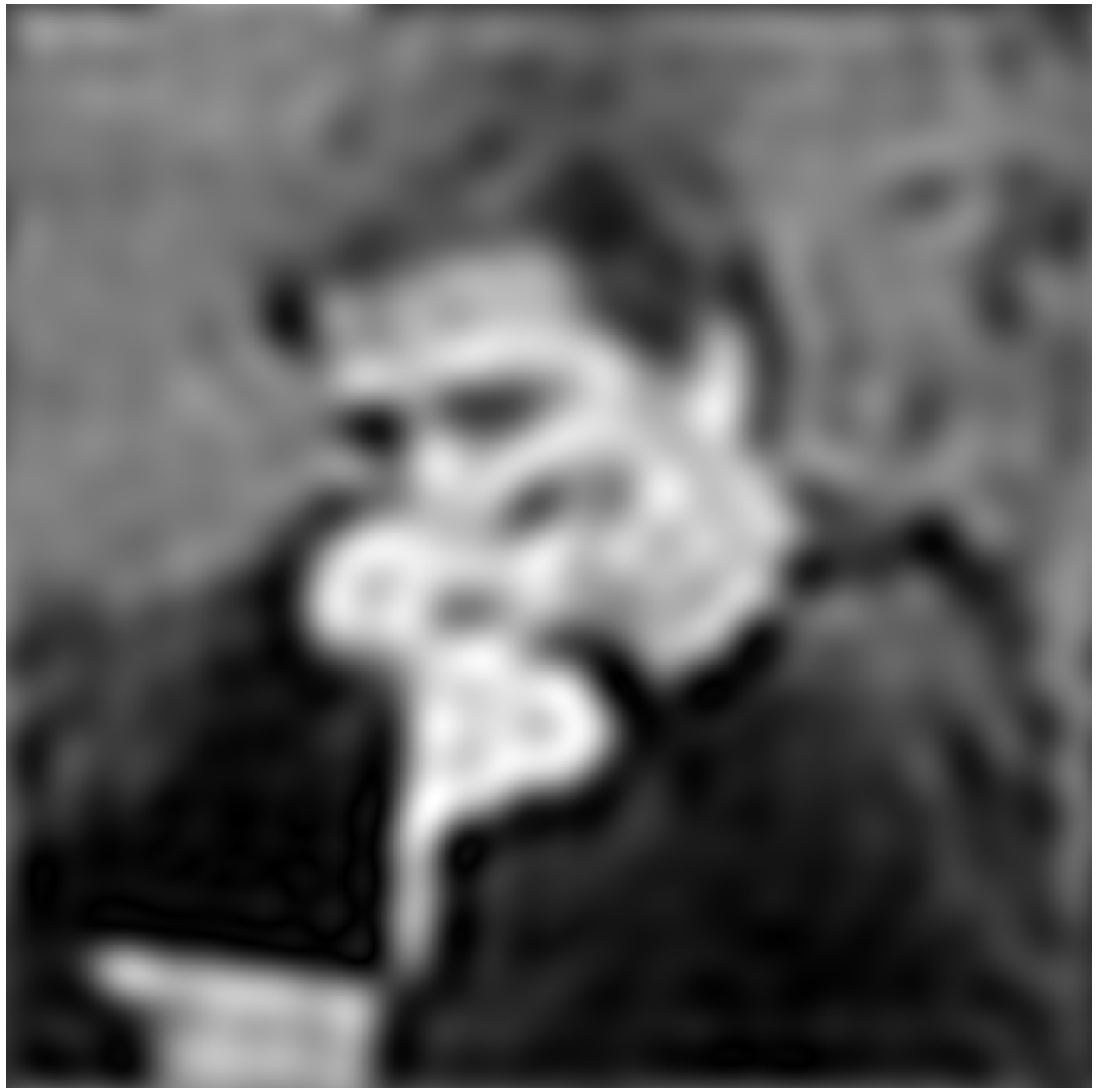
Recall: frequency content of images

Spatial domain result

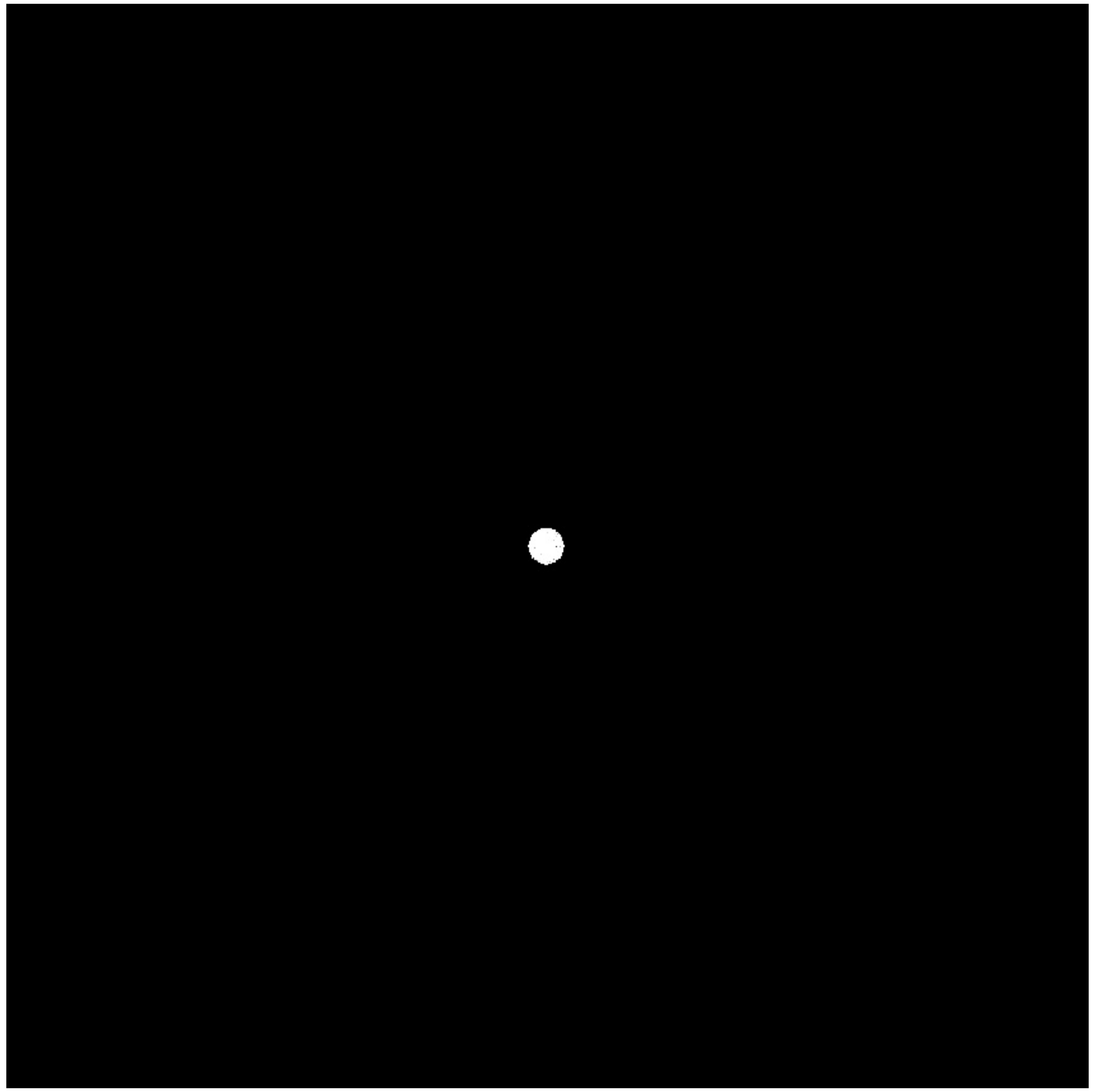


Spectrum of image

Recall: frequency content of images

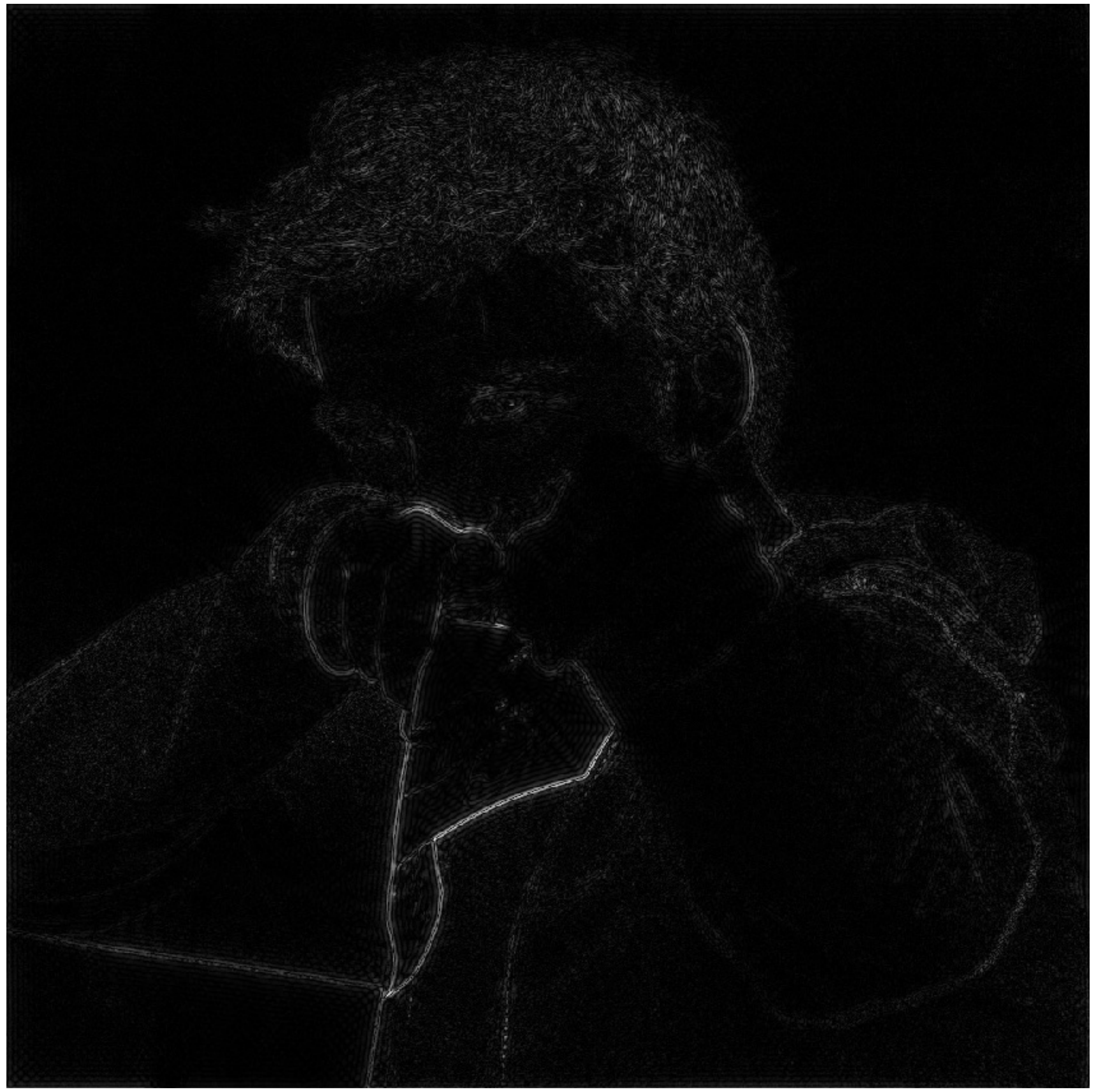


Spatial domain result

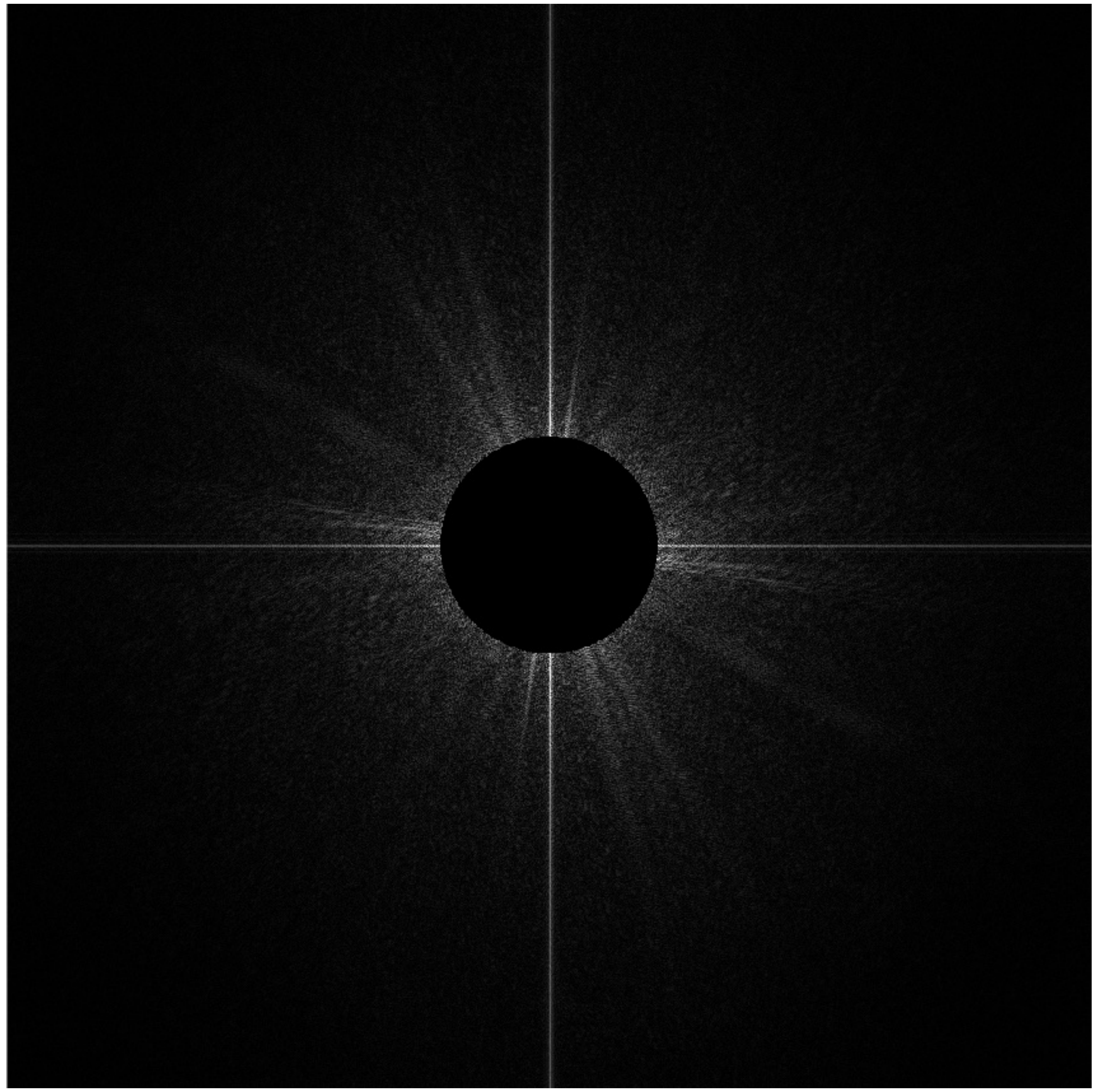


Spectrum (after low-pass filter)
All frequencies above cutoff have 0 magnitude

Recall: frequency content of images



Spatial domain result
(strongest edges)

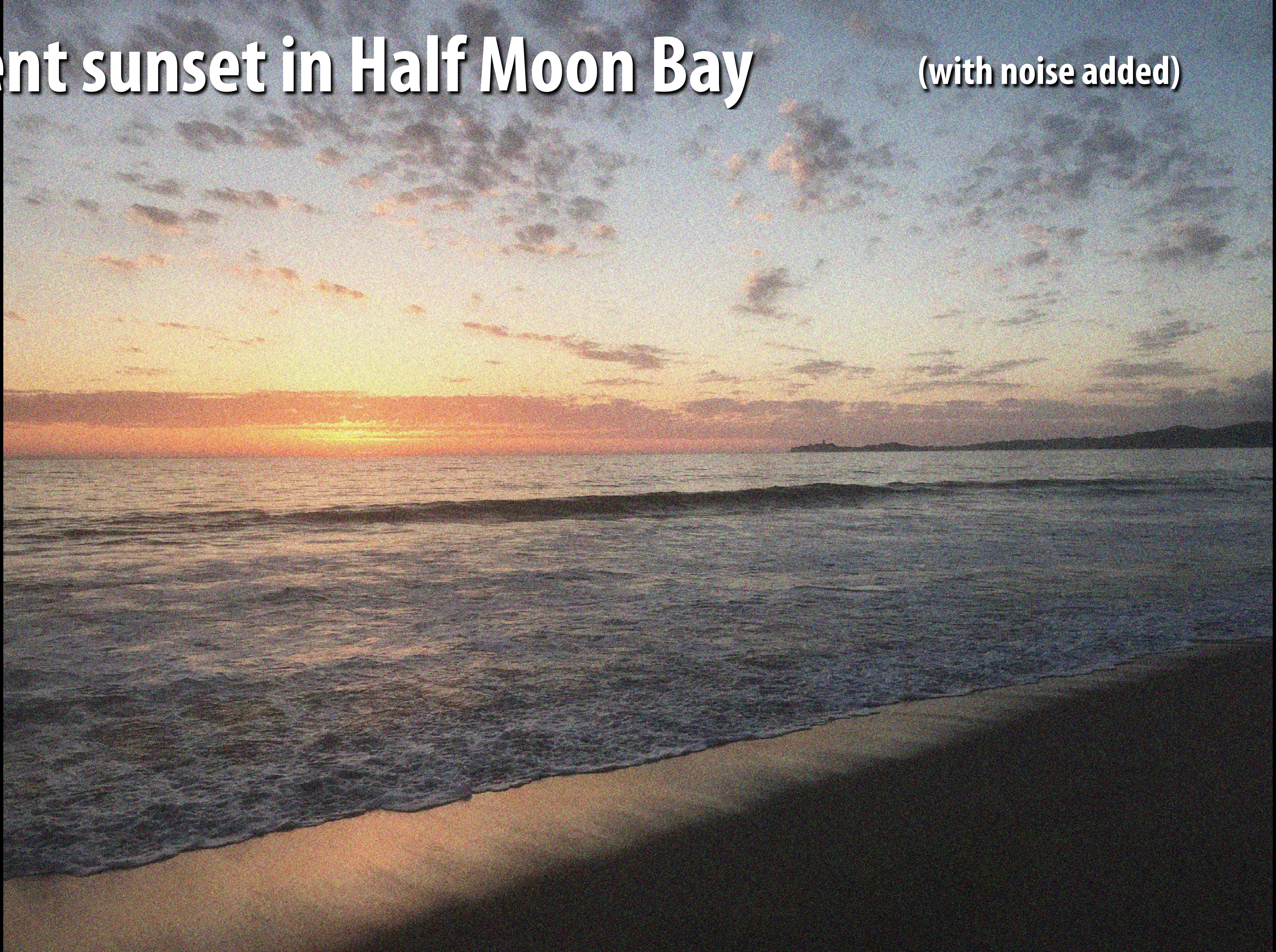


Spectrum (after high-pass filter)
All frequencies below threshold
have 0 magnitude

A recent sunset in Half Moon Bay

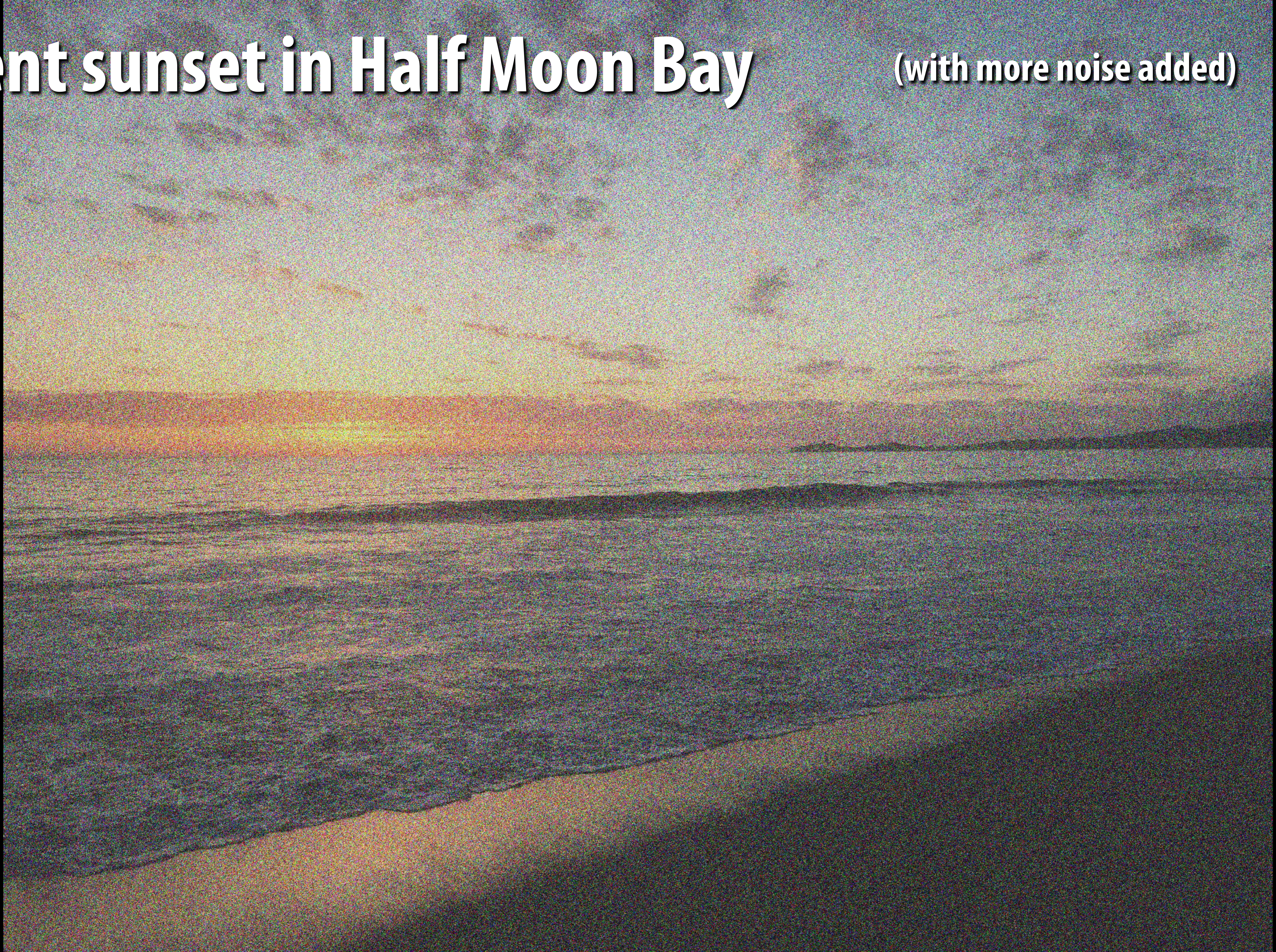
A recent sunset in Half Moon Bay

(with noise added)

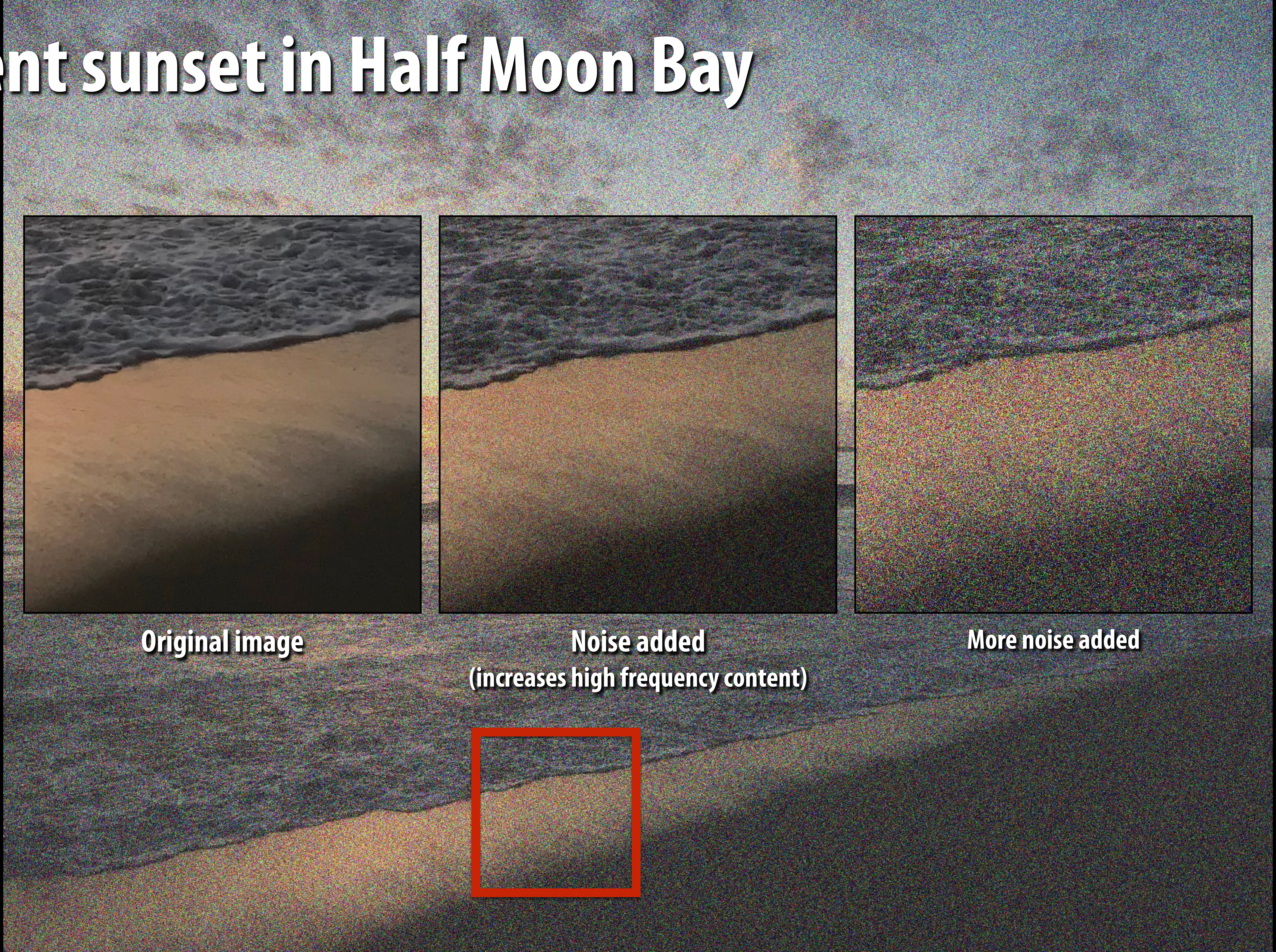


A recent sunset in Half Moon Bay

(with more noise added)



A recent sunset in Half Moon Bay



Original image

Noise added
(increases high frequency content)

More noise added

What is a good representation for manipulating frequency content of images?

Hint:

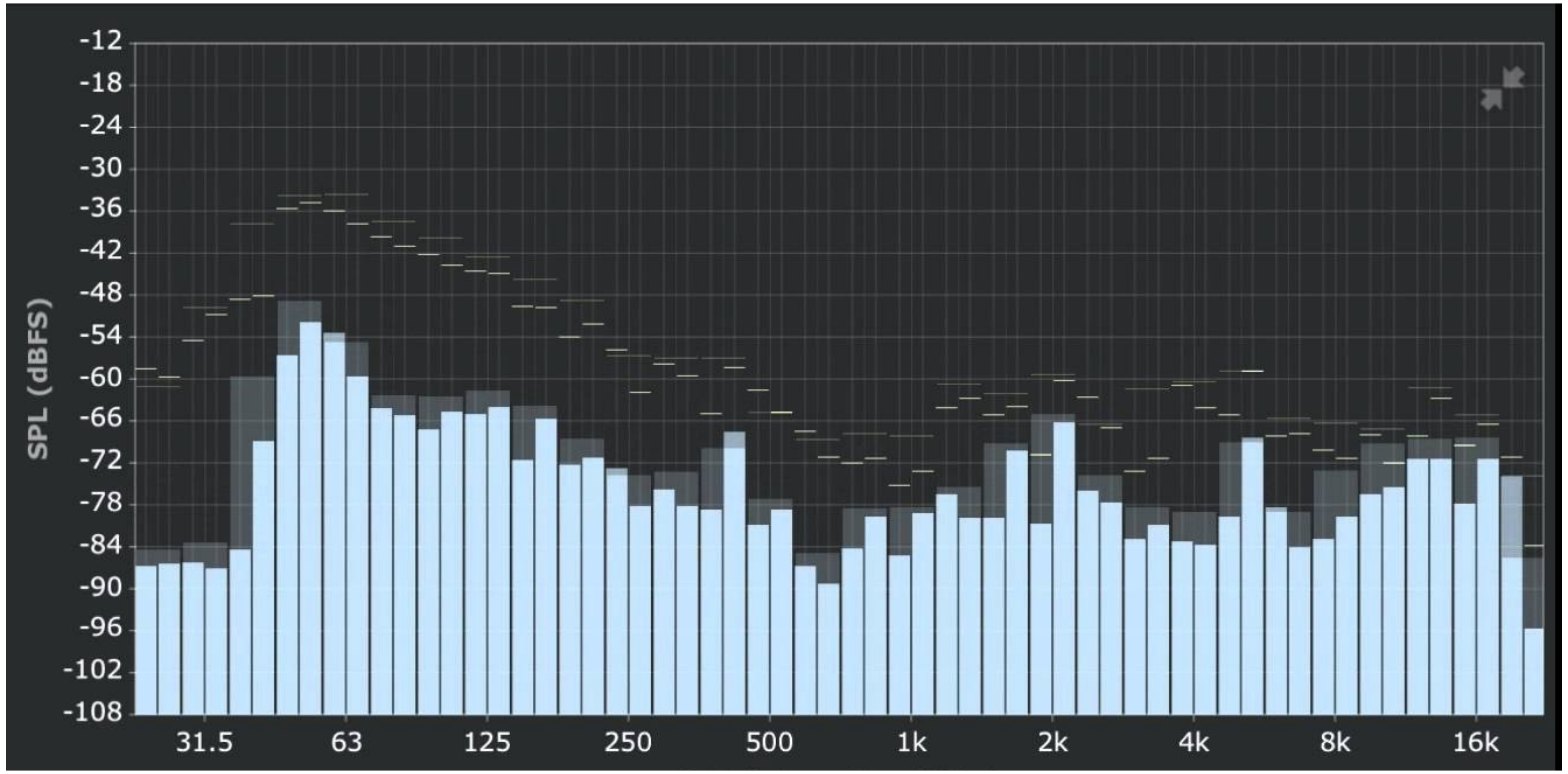


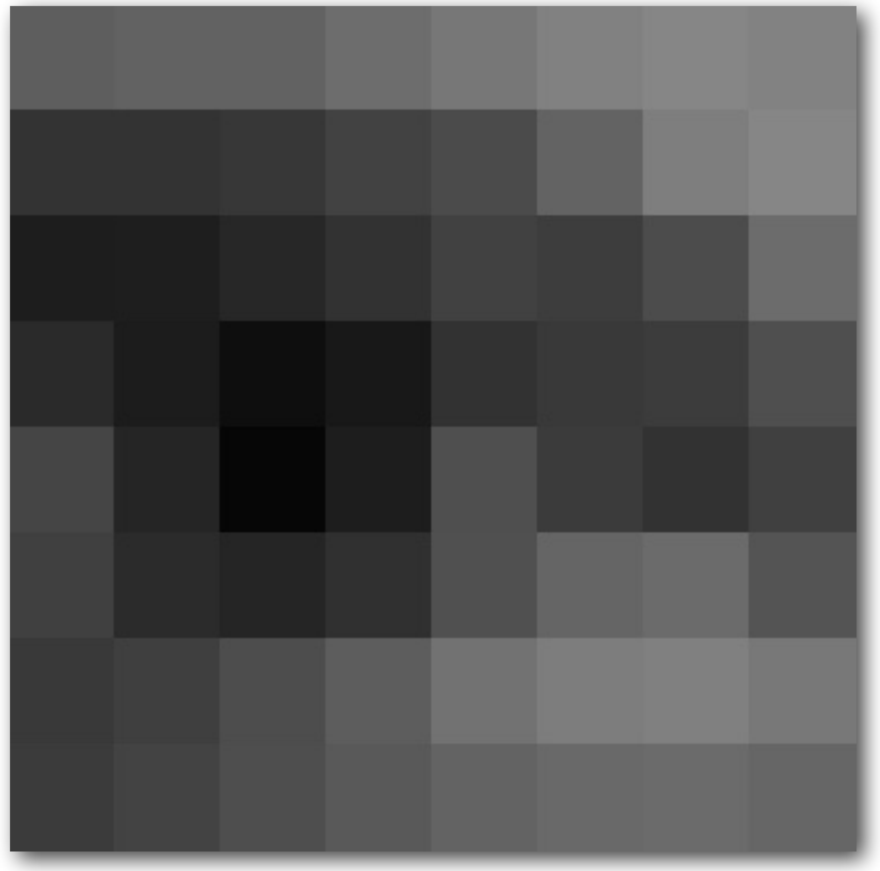
Image transform coding using the discrete cosign transform (DCT)

8x8 pixel block (64 coefficients of signal in "pixel basis")

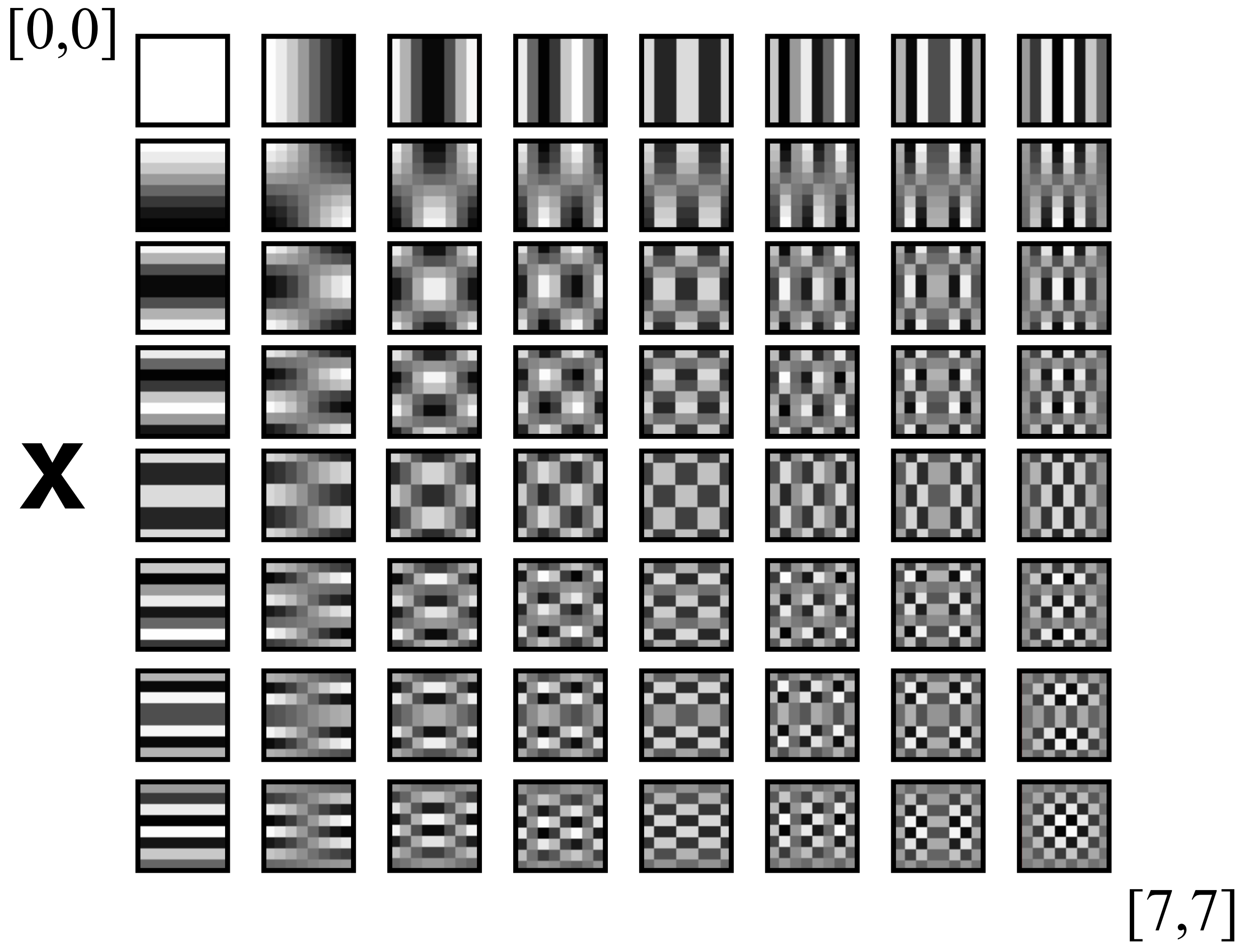
64 basis coefficients

64 cosine basis vectors (each vector is 8x8 image)

$$\text{basis}[i,j] = \cos\left[\pi \frac{i}{N}\left(x+\frac{1}{2}\right)\right] \times \underset{j}{\cos}\left[\pi \frac{j}{N}\left(y+\frac{1}{2}\right)\right]$$



$$= \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \times$$

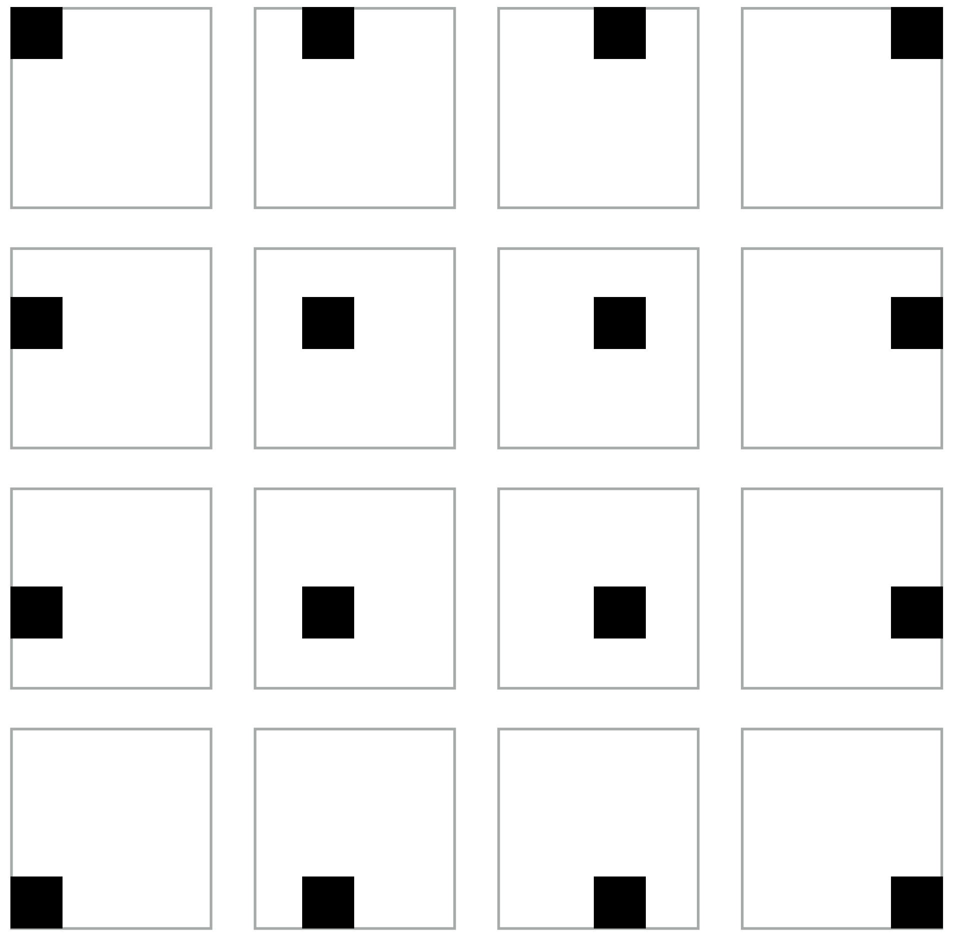


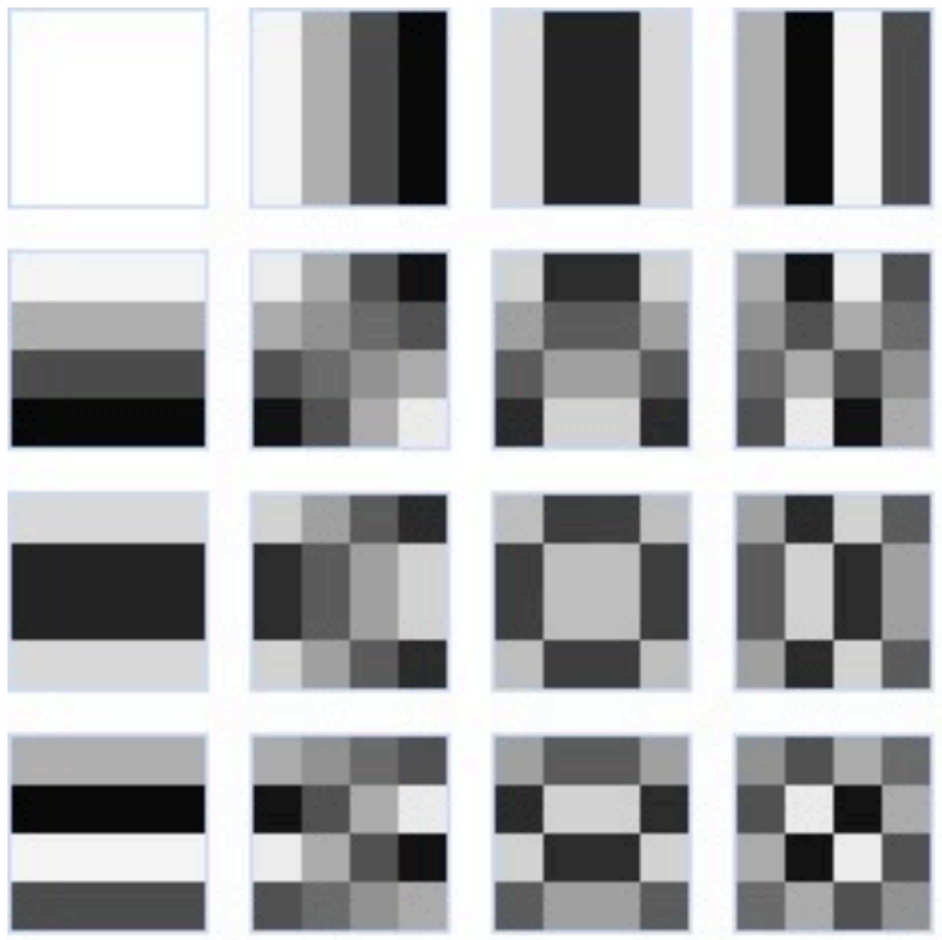
In practice: DCT is applied to 8x8 pixel blocks of Y' channel, 16x16 pixel blocks of Cb, Cr (assuming 4:2:0)

Examples of other bases

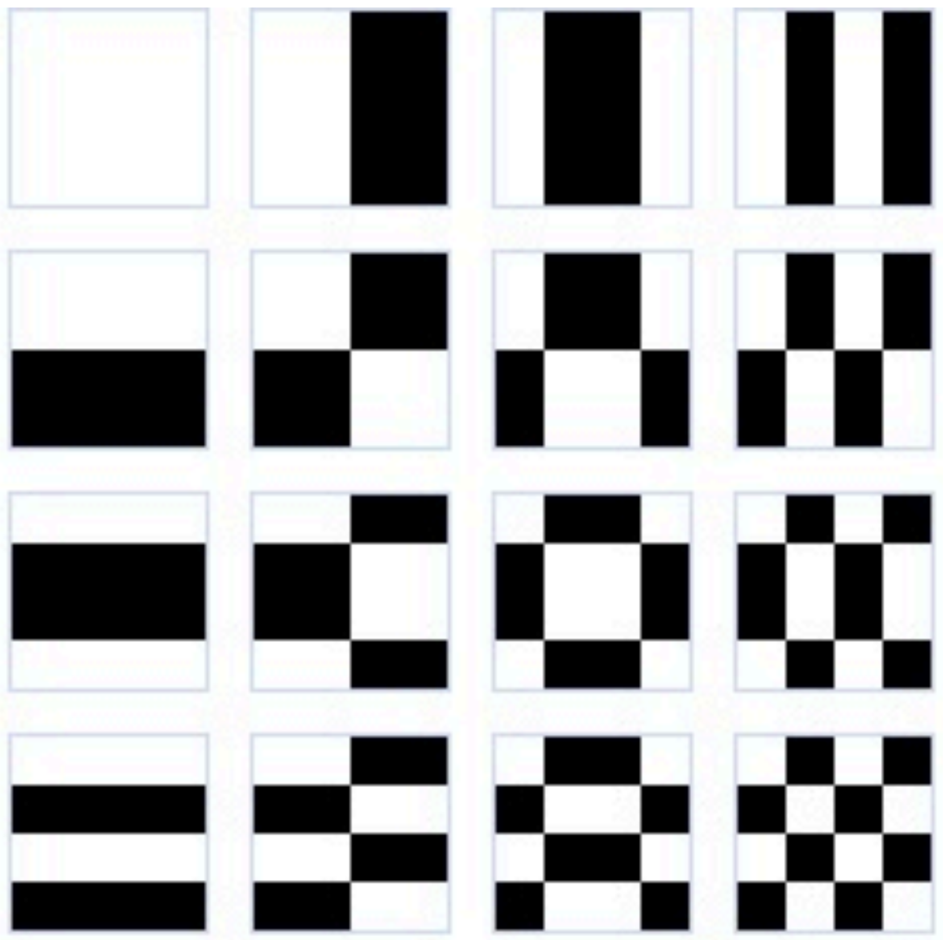
This slide illustrates basis images for 4x4 block of pixels (although JPEG works on 8x8 blocks)

Pixel Basis
(Compact: each coefficient in representation only effects a single pixel of output)

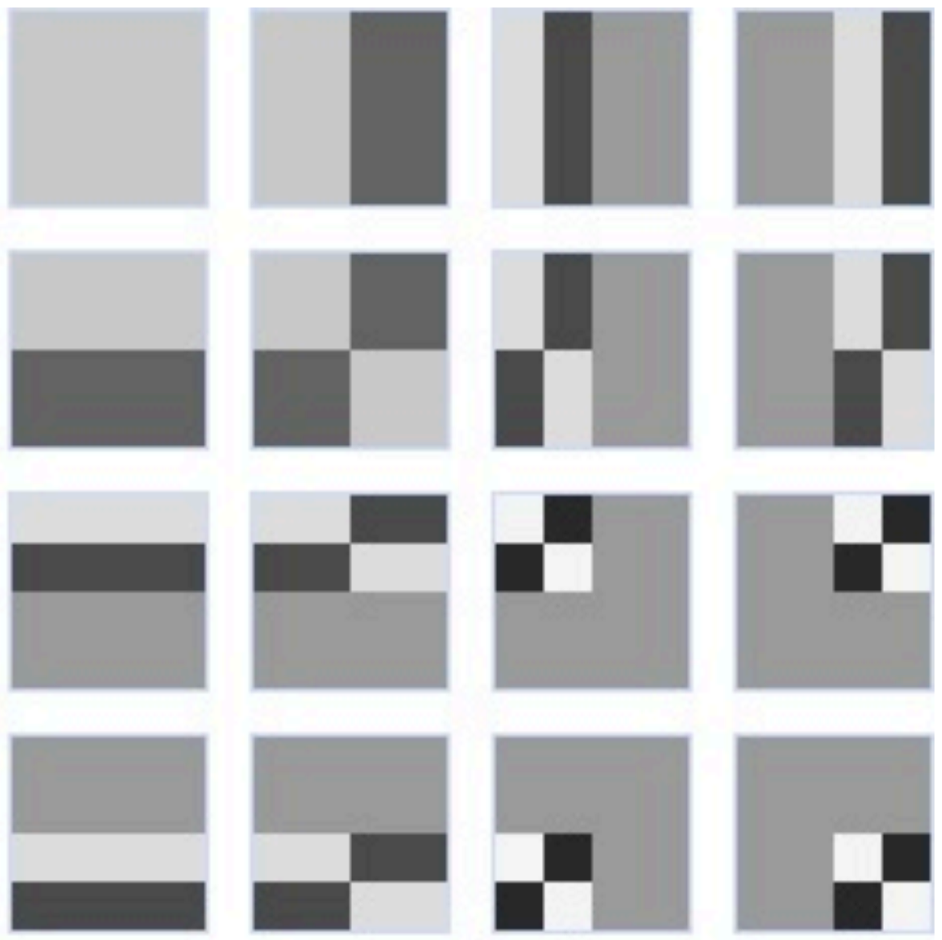




DCT



Walsh-Hadamard



Haar Wavelet

[Image credit: https://people.xiph.org/~xiphmont/demo/daala/demo3.shtml]

Quantization

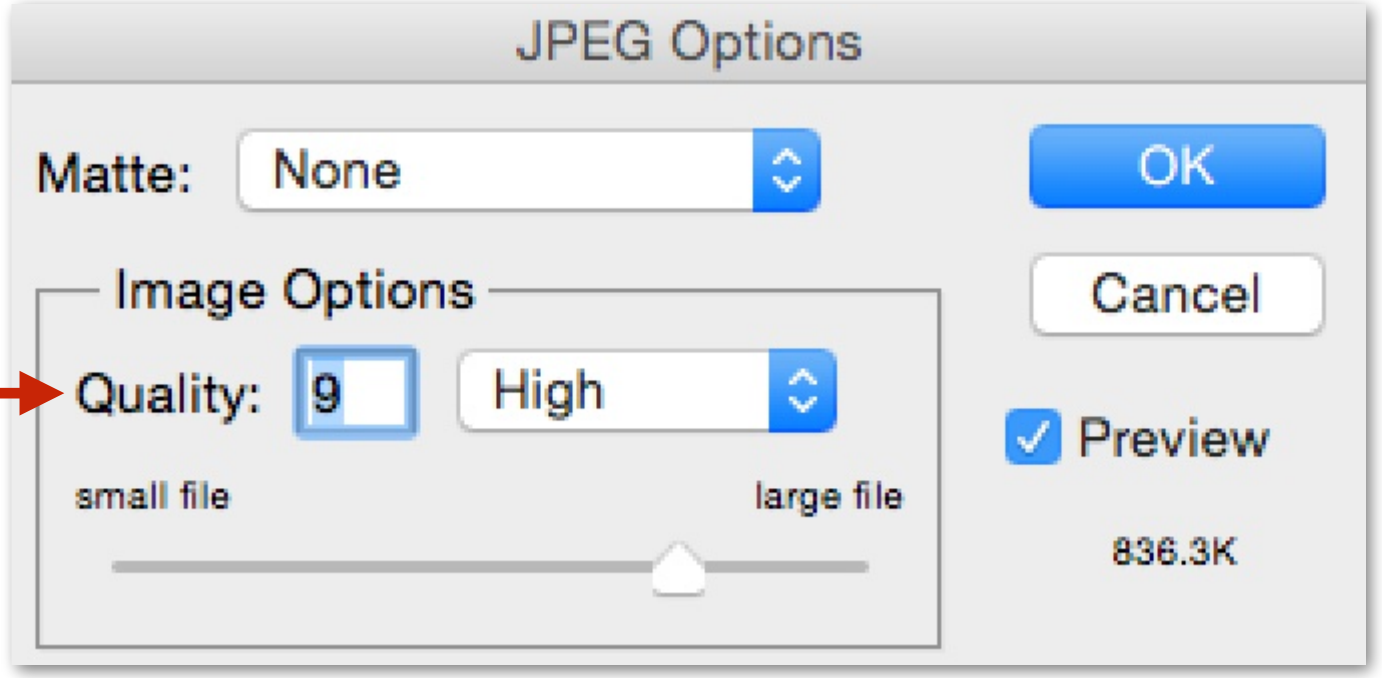
$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \Big/ \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Result of DCT
(representation of image in cosine basis)

Quantization Matrix

$$= \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Changing JPEG quality setting in your favorite photo app modifies this matrix (“lower quality” = higher values for elements in quantization matrix)



JPEG Options

Matte: None

Image Options

Quality: 9 High

☑ Preview

836.3K

Quantization produces small values for coefficients (only few bits needed per coefficient)

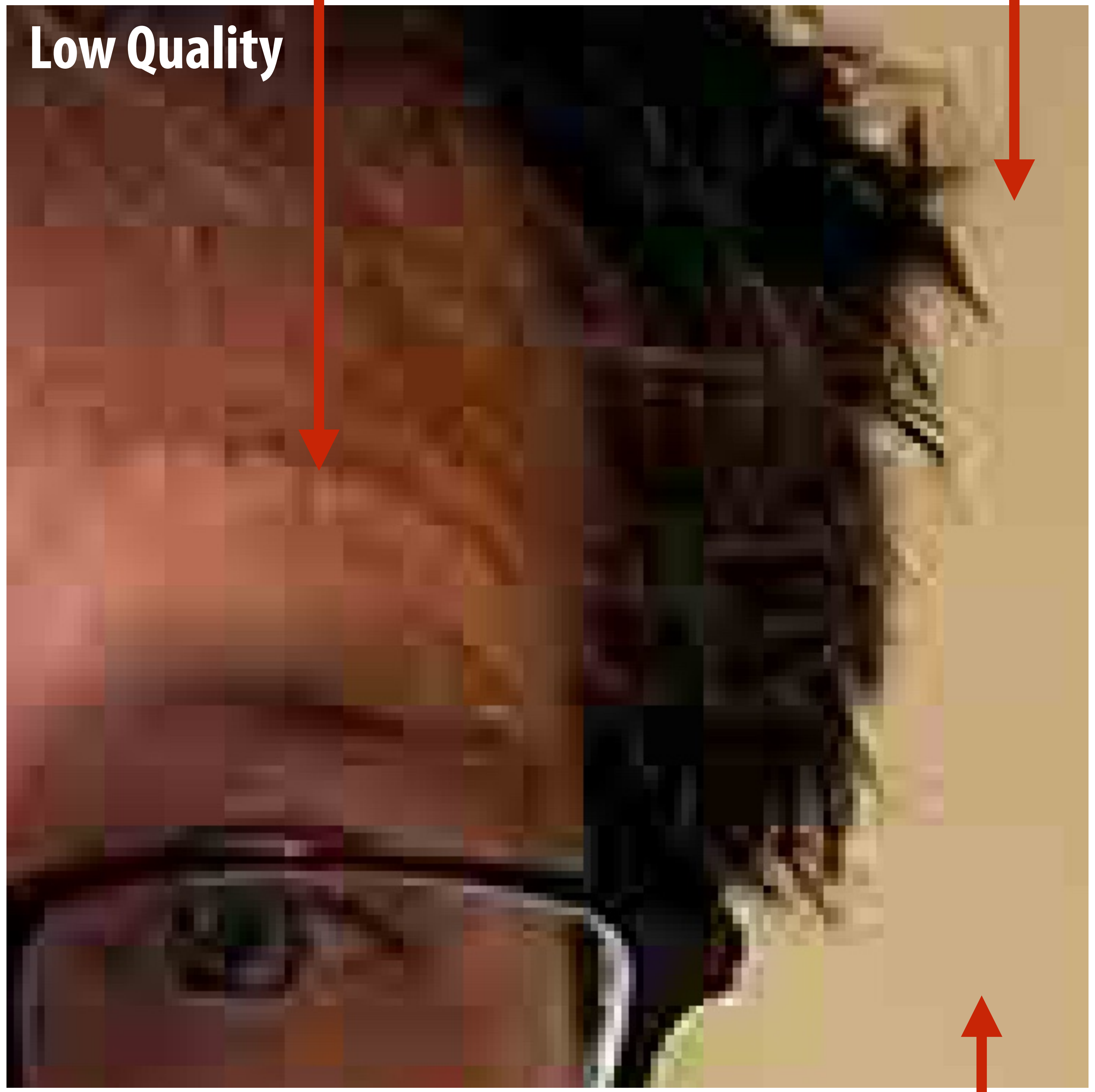
Quantization zeros out many coefficients

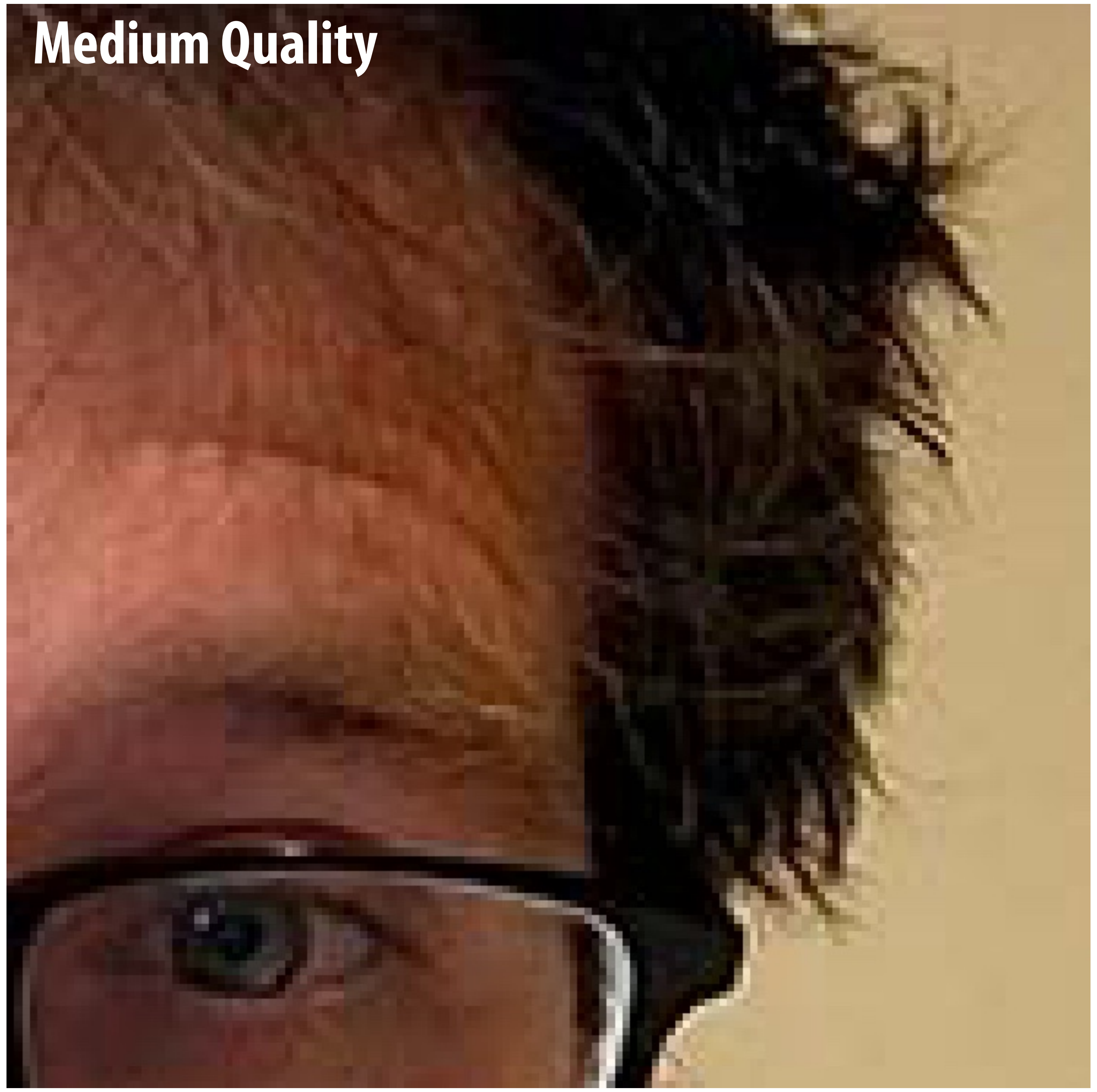
[Credit: Wikipedia, Pat Hanrahan]

JPEG compression artifacts

Noticeable 8x8 pixel block boundaries

Noticeable error near high gradients

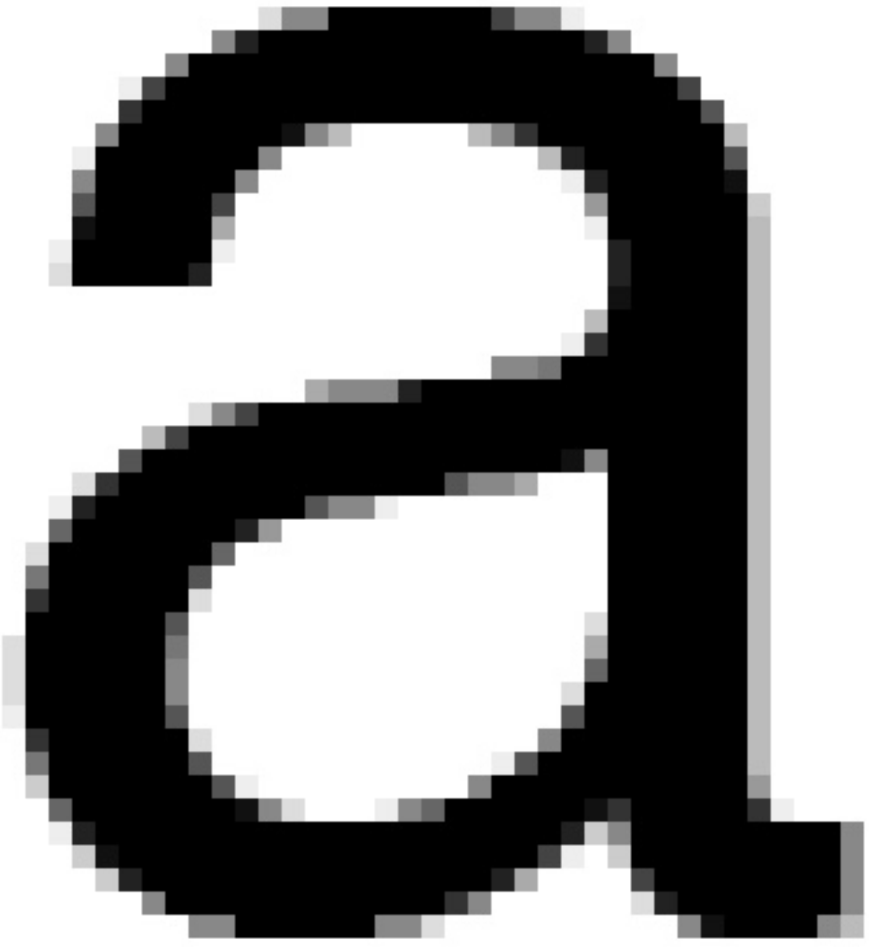




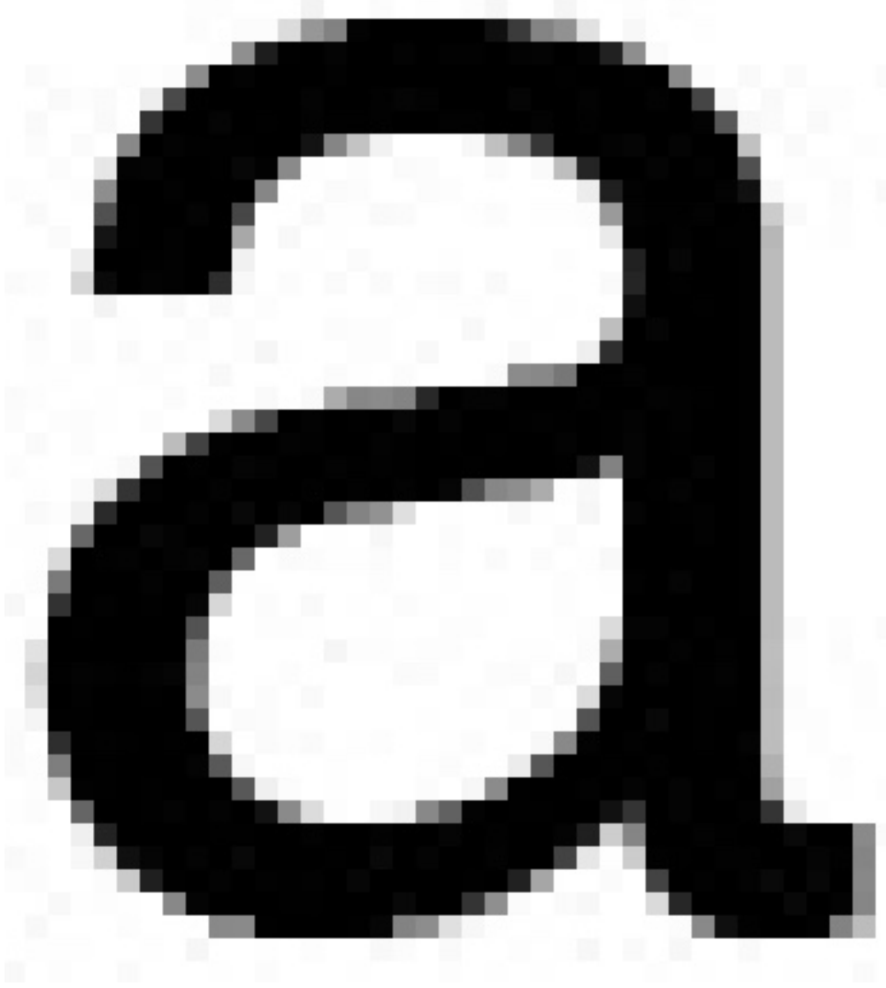
Low-frequency regions of image represented accurately even under high compression

JPEG compression artifacts

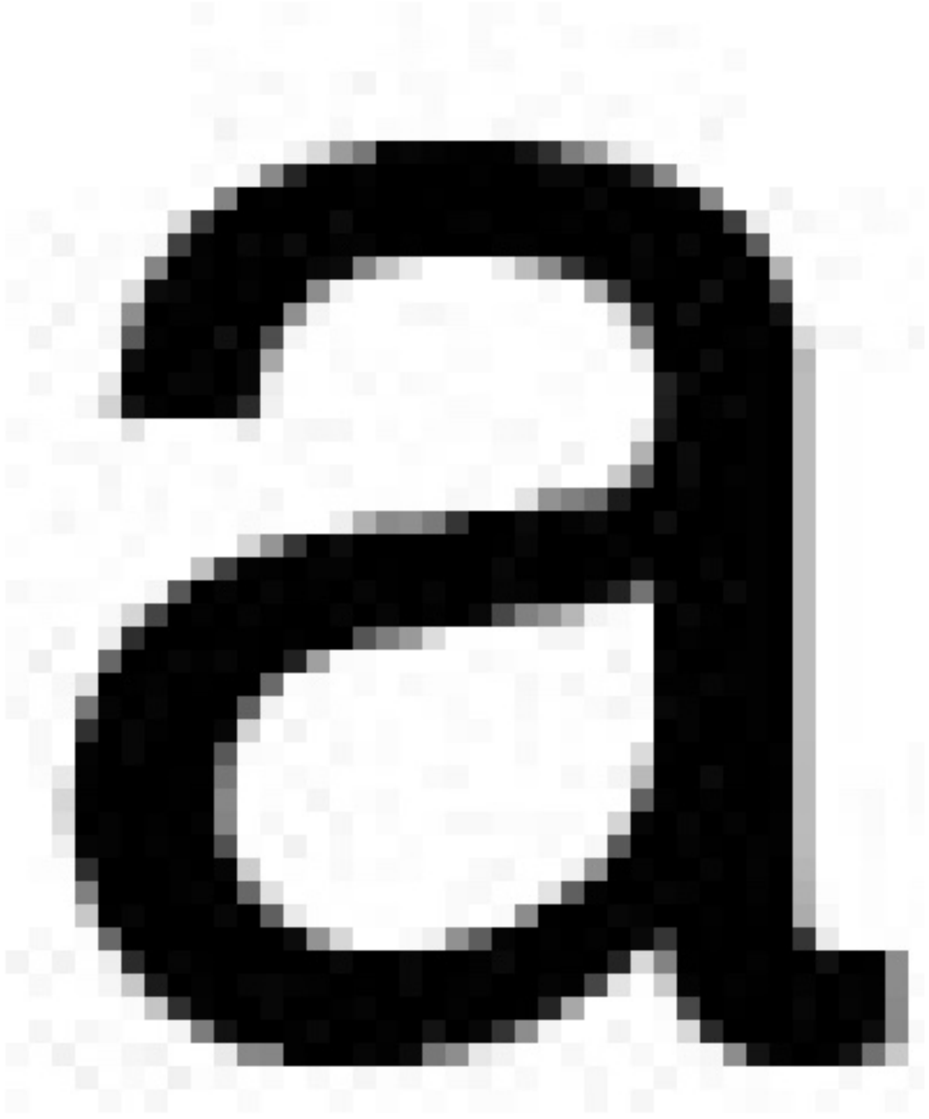
Original Image
(actual size)



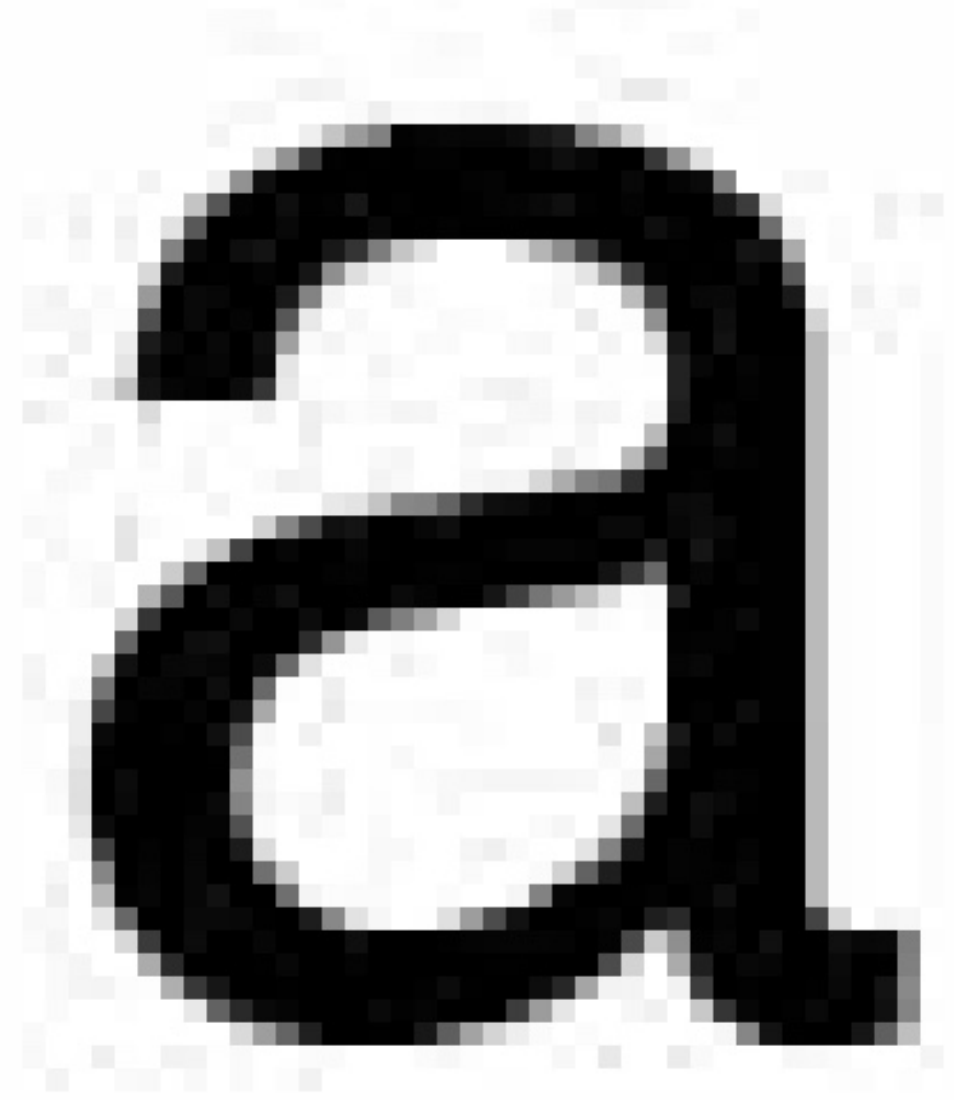
Original Image



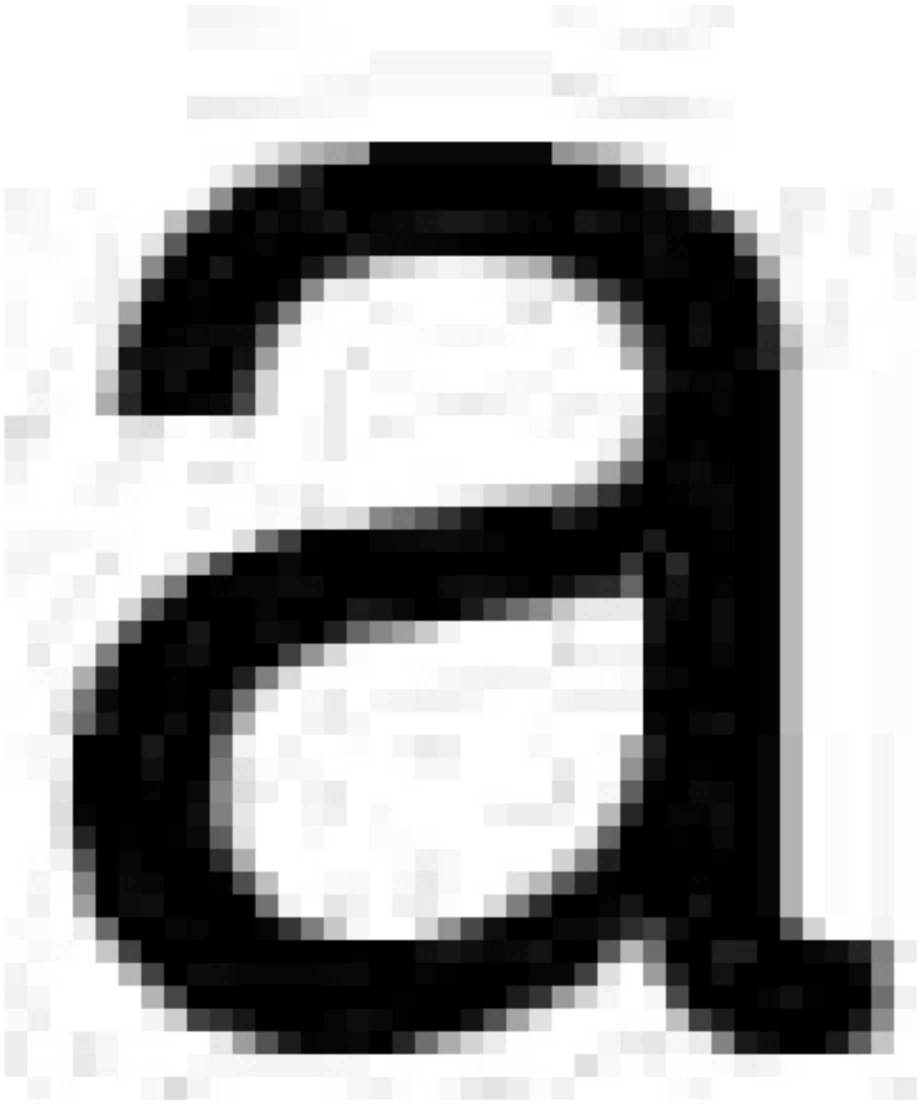
Quality Level 9



Quality Level 6



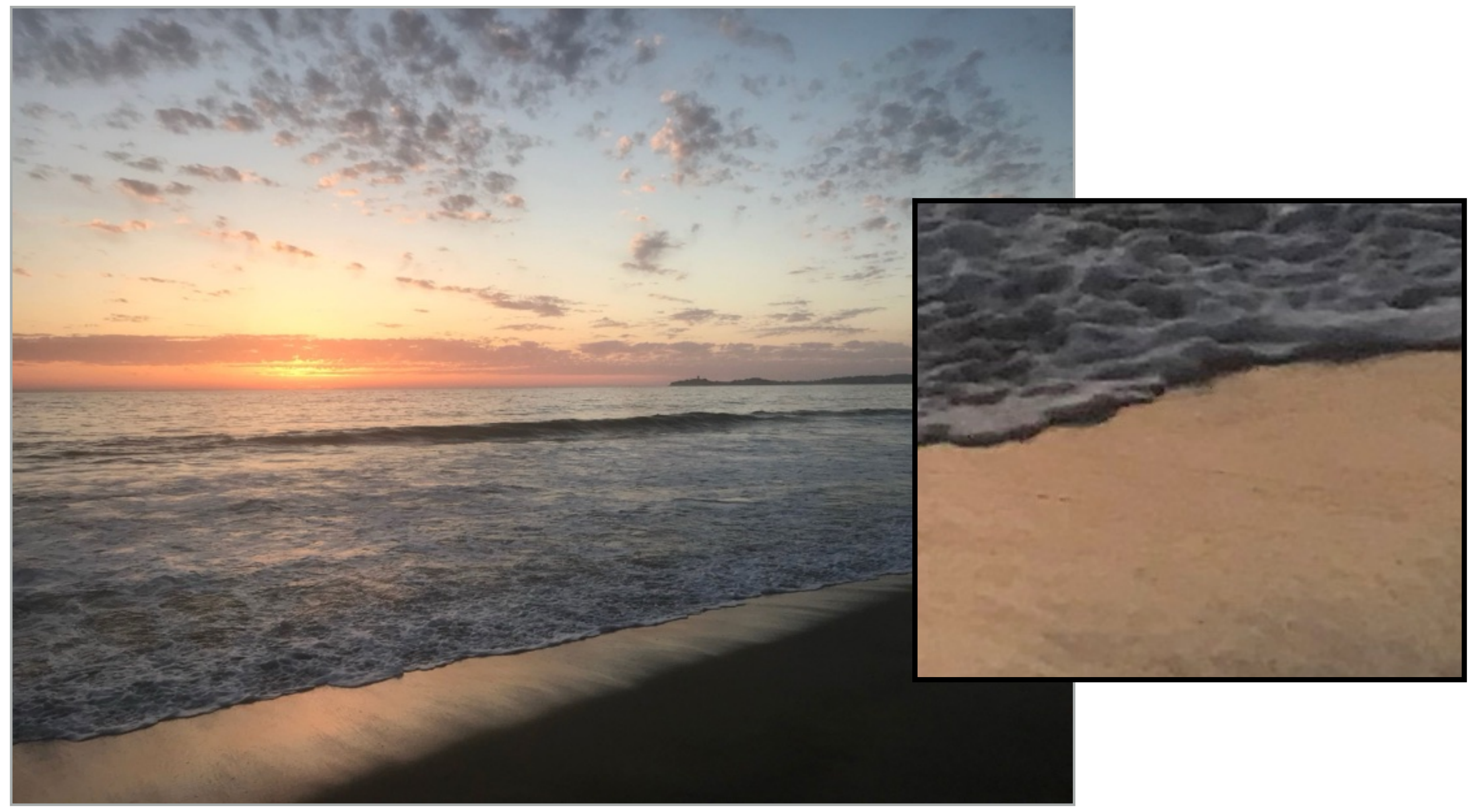
Quality Level 3



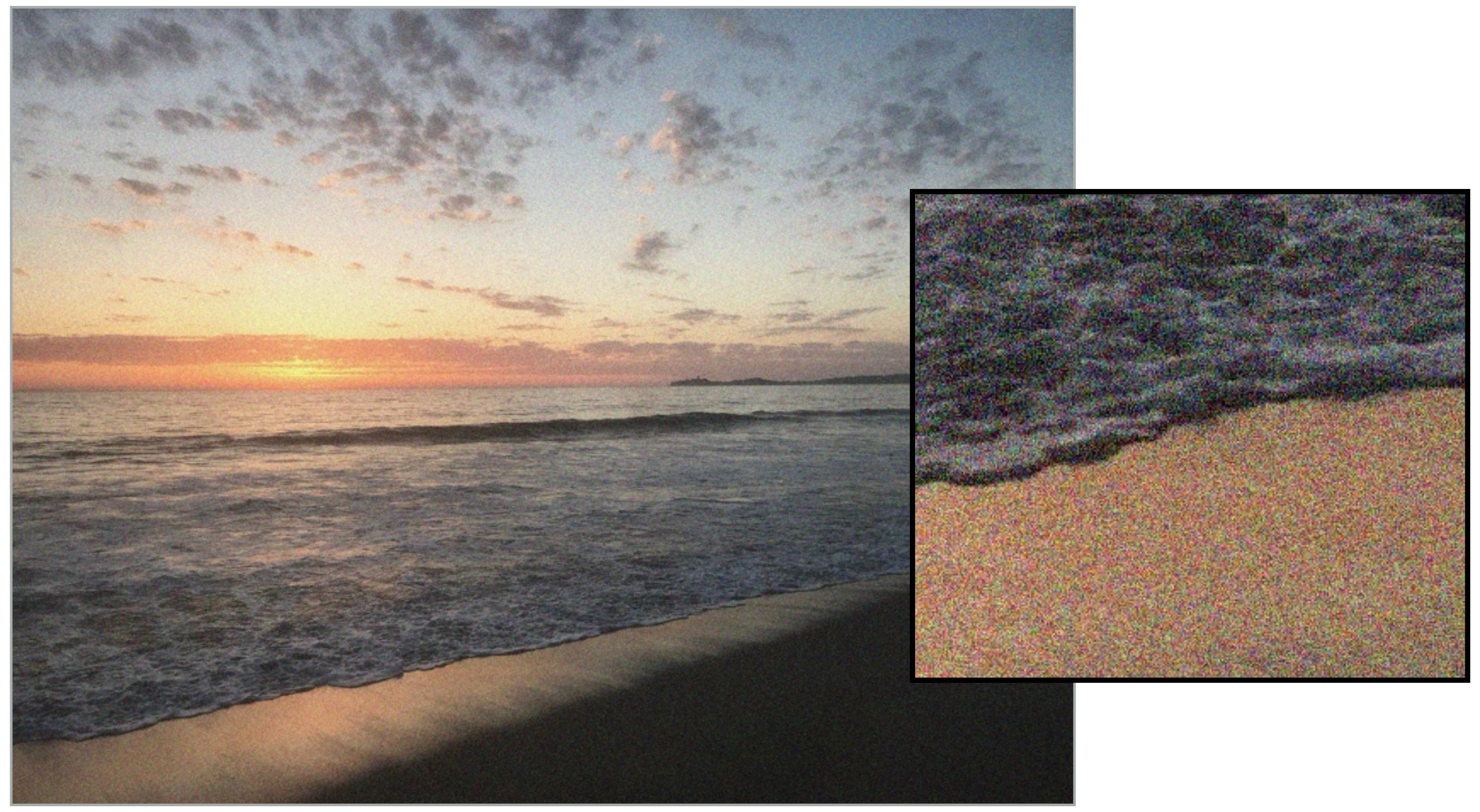
Quality Level 1

Why might JPEG compression not be a good compression scheme for illustrations and rasterized text?

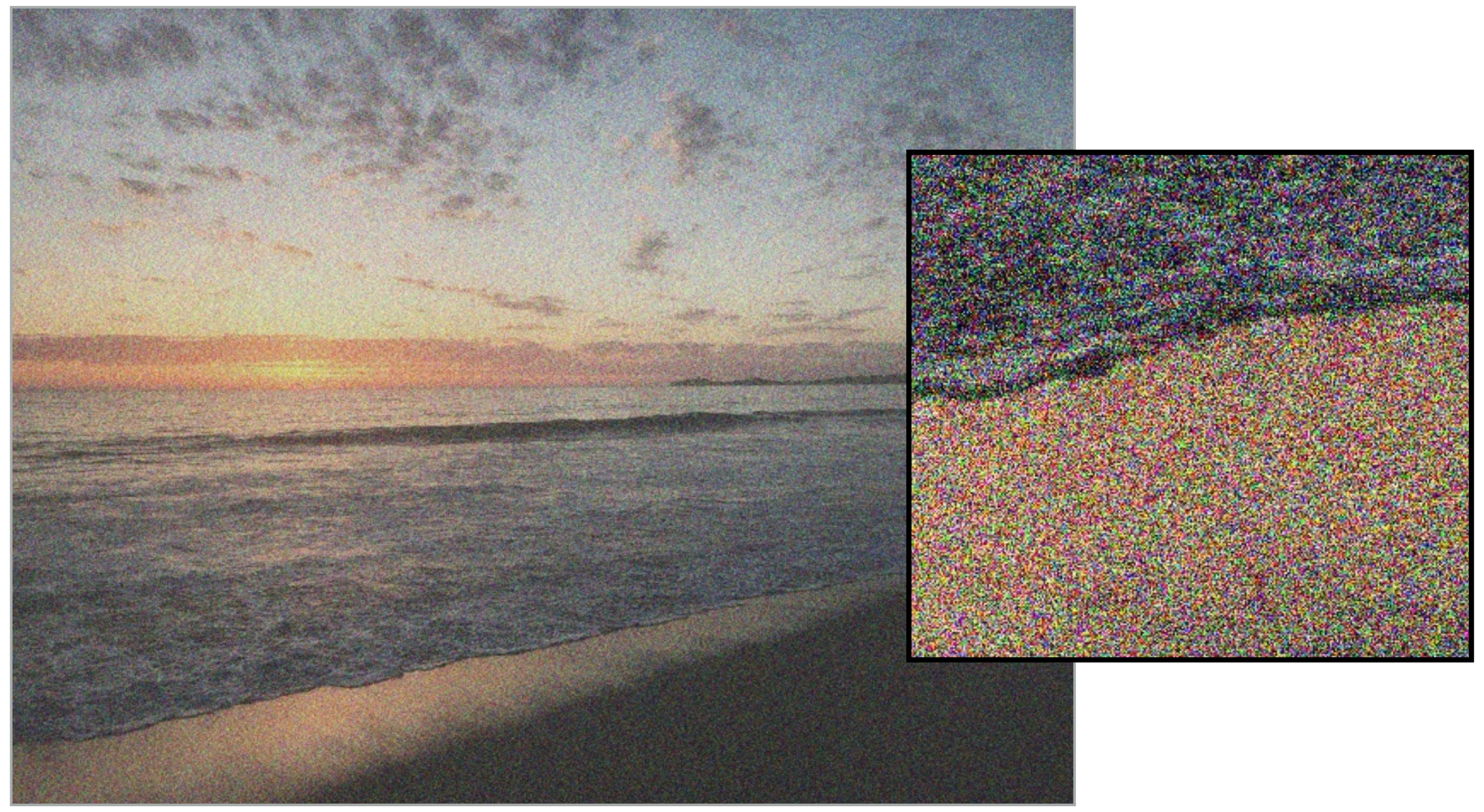
Images with high frequency content do not exhibit as high of compression ratios. Why?



Original image: 2.9MB JPG



Medium noise: 22.6 MB JPG



High noise: 28.9 MB JPG

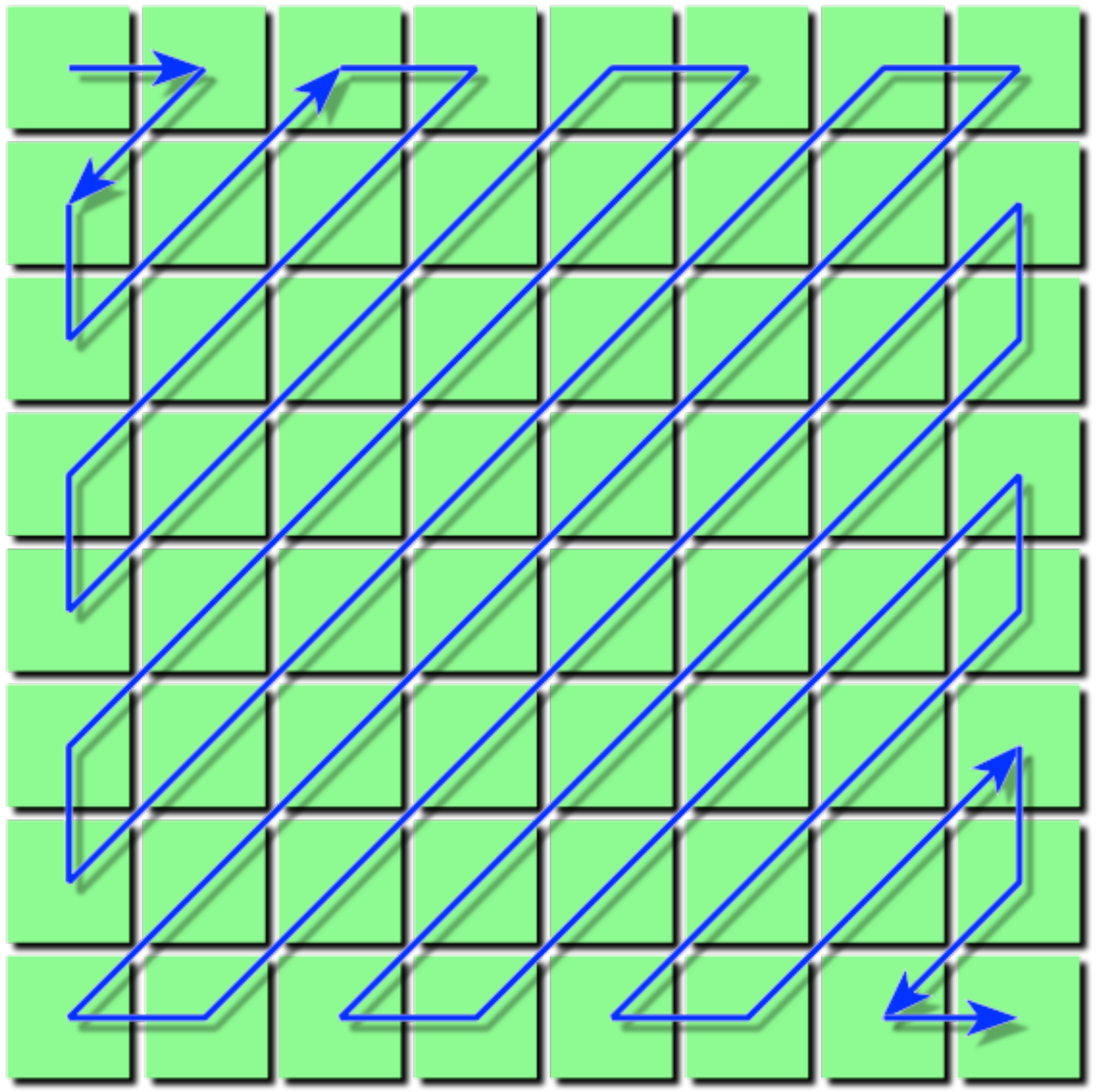
Photoshop JPG compression level = 10
used for all compressed images

Uncompressed image:
4032 x 3024 x 24 bytes/pixel = 36.6 MB

Lossless compression of quantized DCT values

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Quantized DCT Values



Reordering

Entropy encoding: (lossless)

- Reorder values
- Run-length encode (RLE) 0's
- Huffman encode non-zero values

Image credit: Wikipedia

JPEG compression summary

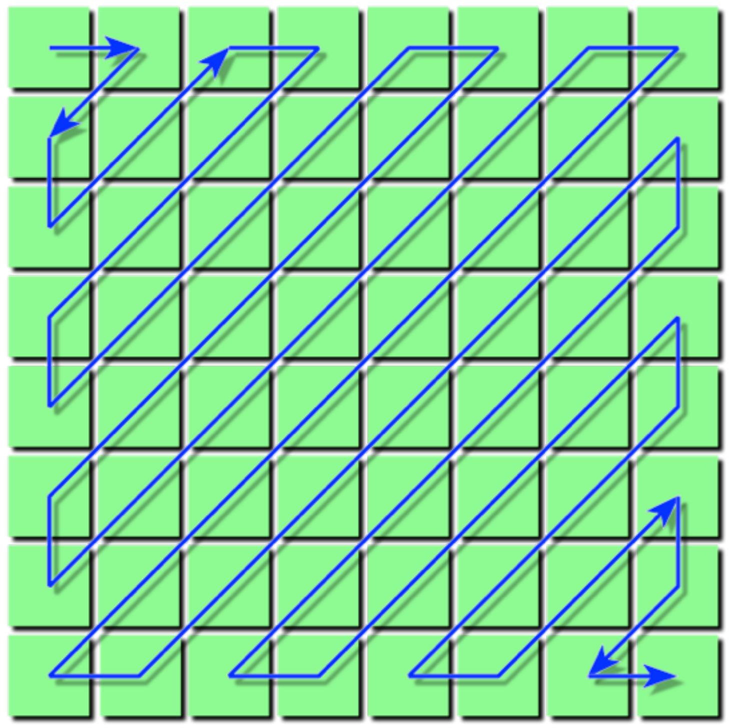
$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \Bigg/ \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

DCT **Quantization Matrix**

$$= \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Quantized DCT

Quantization loses information (lossy compression!)



Coefficient reordering

RLE compression of zeros

Entropy compression of non-zeros

Lossless compression!

Compressed bits

Credit: Pat Hanrahan

JPEG compression summary

```
Convert image to Y’CbCr
Downsample CbCr  (to 4:2:2 or 4:2:0)       (information loss occurs here)
For each color channel (Y’, Cb, Cr):
    For each 8x8 block of values
        Compute DCT
        Quantize results                   (information loss occurs here)
        Reorder values
        Run-length encode 0-spans
        Huffman encode non-zero values
```

Key idea: exploit characteristics of human perception to build efficient image storage and image processing systems

- Separation of luminance from chrominance in color representation (Y'CrCb) allows reduced resolution in chrominance channels (4:2:0)

- Encode pixel values linearly in lightness (perceived brightness), not in luminance (distribute representable values uniformly in perceptual space)

- JPEG compression significantly reduces file size at cost of quantization error in high spatial frequencies
  - Human brain is more tolerant of errors in high frequency image components than in low frequency ones
  - Images of the real world are dominated by low-frequency components

20 second video: 1920 x 1080, @ 30fps

After decode: 8-bits per channel RGB → 24 bits/pixel → 6.2 MB/frame

(6.2 MB/frame x 20 sec x 30 fps = 3.5 GB)

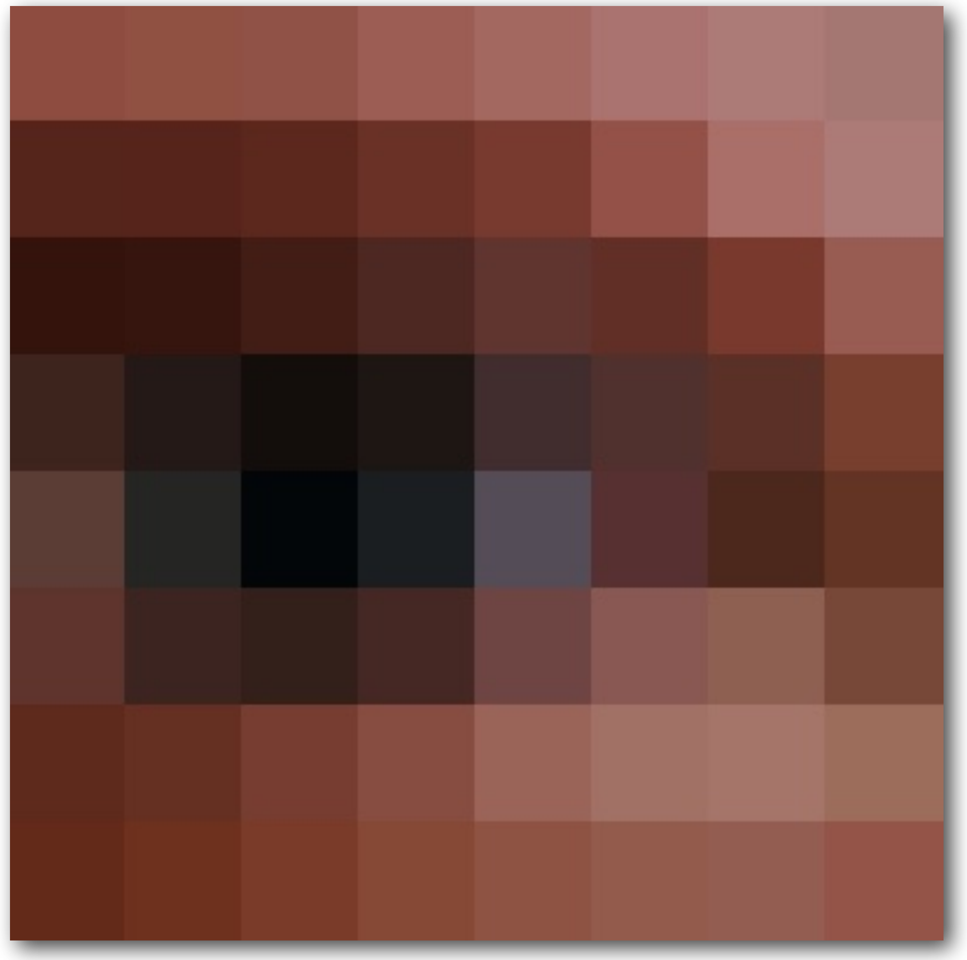
Size of data when each frames stored as JPG: 404 MB

Video file size when compressed using H.264: 26.6 MB (133-to-1 compression ratio compared to uncompressed, 8-to-1 compared to JPG)

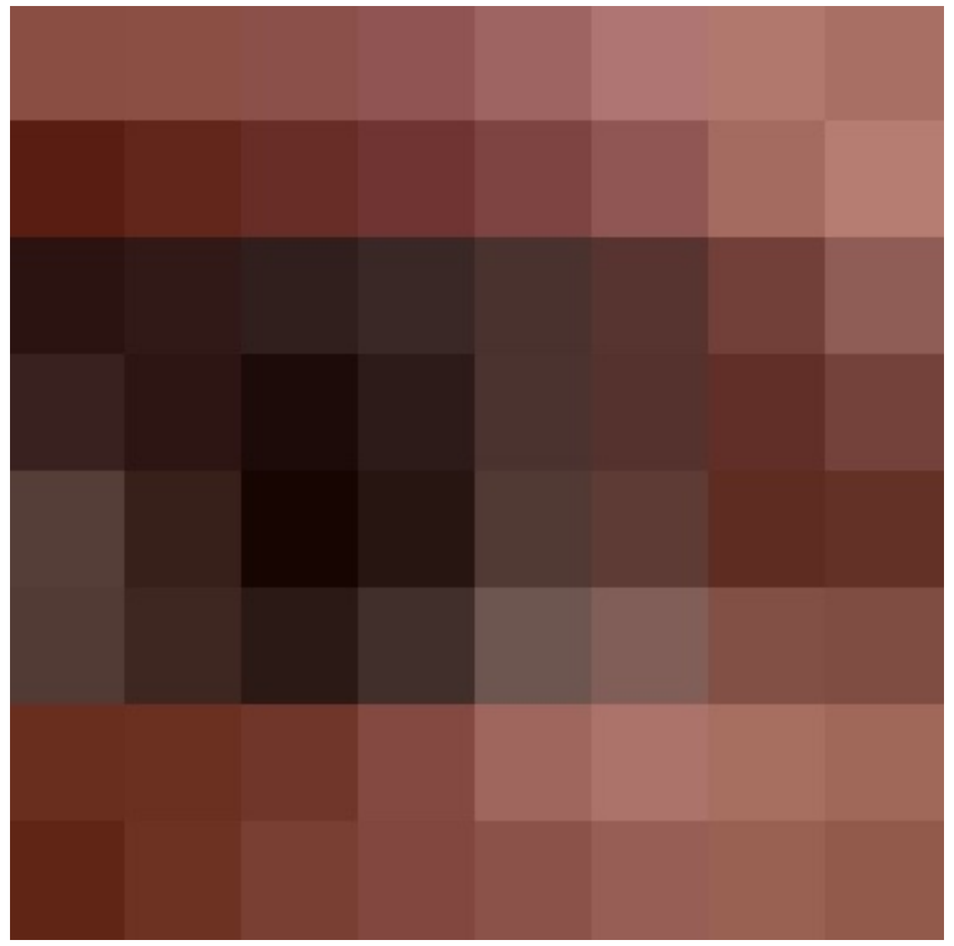
Video compression adds two main ideas

- **Exploiting redundancy:**
  - Intra-frame redundancy: value of pixels in neighboring regions of a frame are good predictor of values for other pixels in the frame (spatial redundancy)
  - Inter-frame redundancy: pixels from nearby frames in time are a good predictor for the current frame's pixels (temporal redundancy)

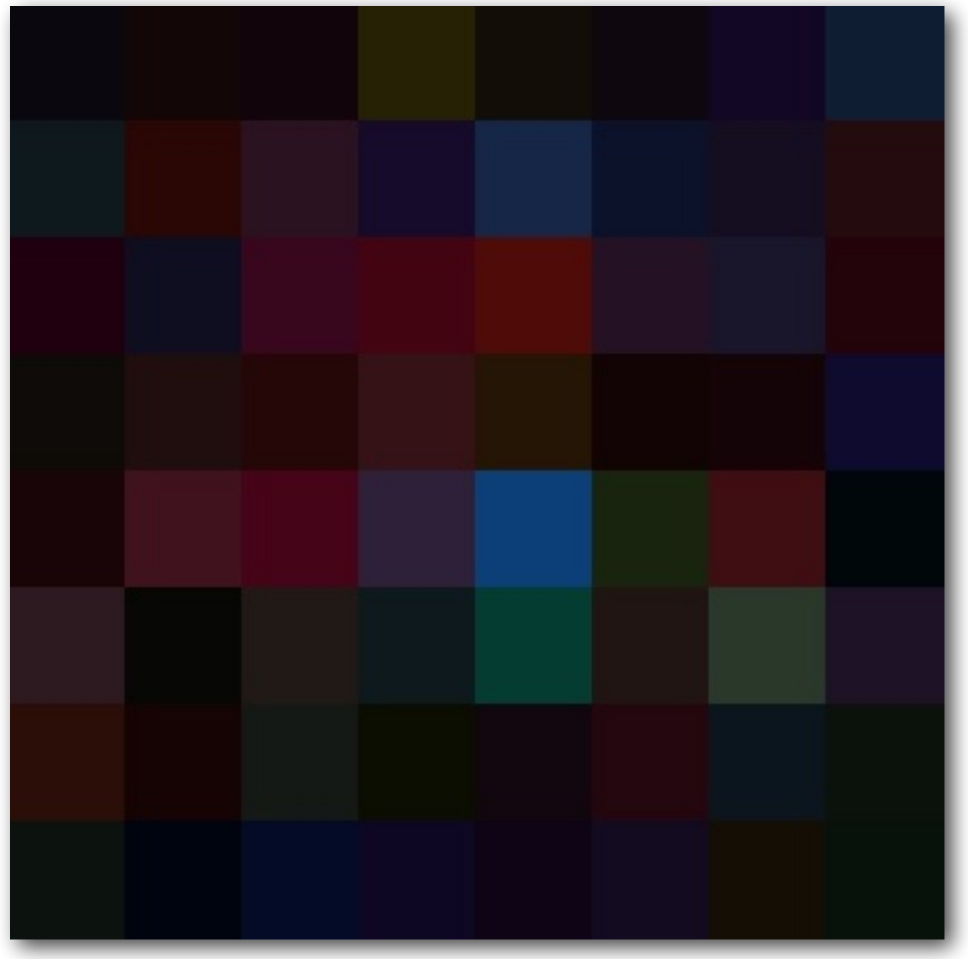
Residual: difference between predicted image and original image



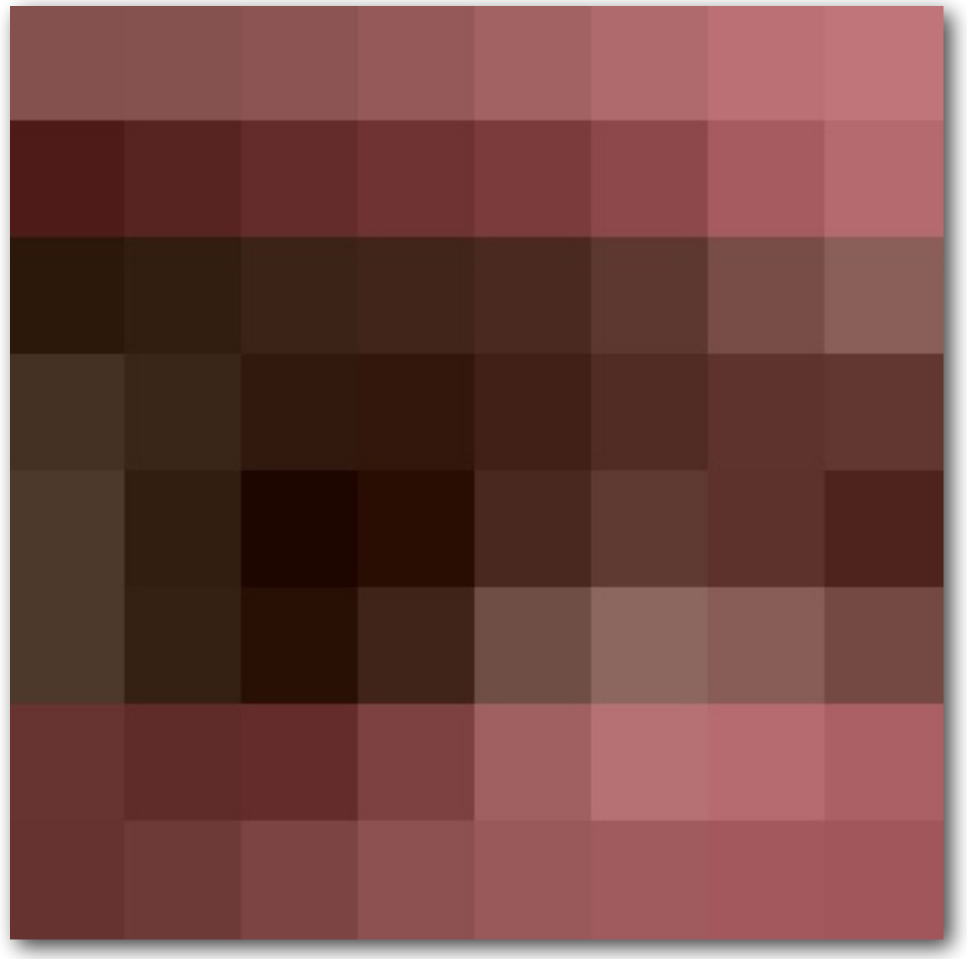
Original pixels



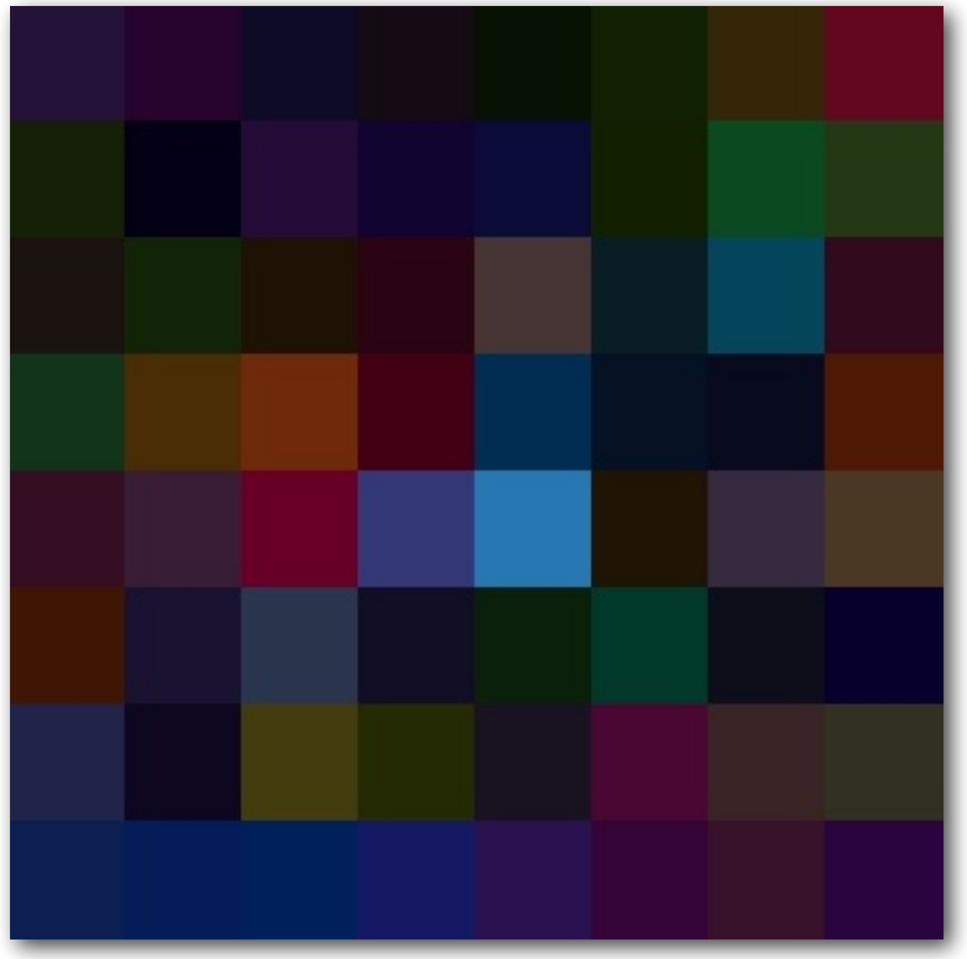
Predicted pixels
(Prediction A)



Residual
(amplified for visualization)



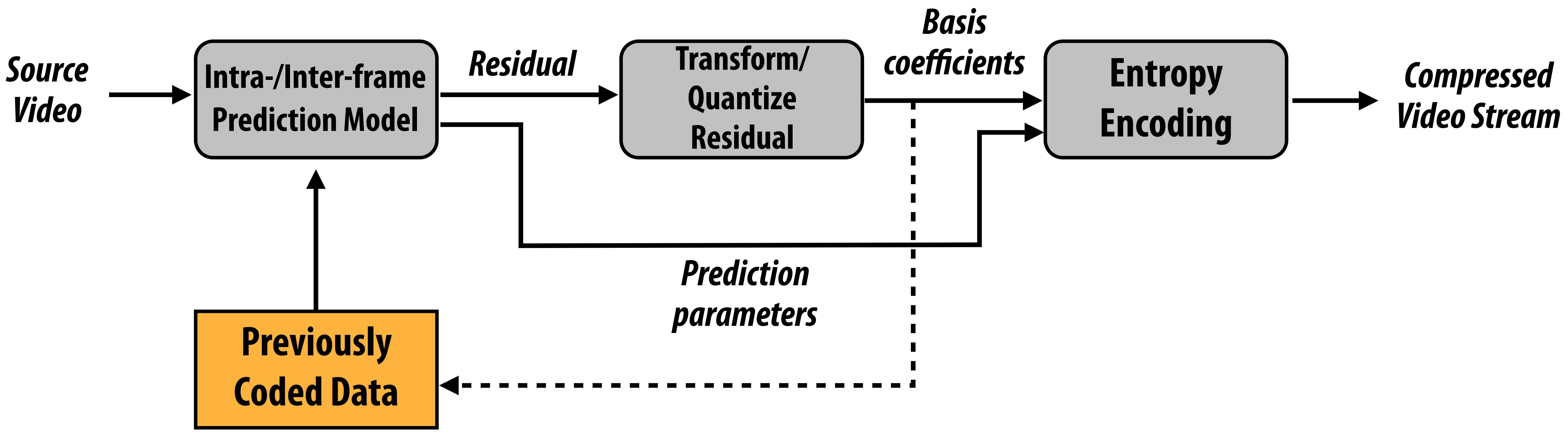
Predicted pixels
(Prediction B)



Residual
(amplified for visualization)

In video compression schemes, the residual image is compressed using lossy compression techniques like those described in the earlier part of this lecture. Better predictions lead to smaller and more compressible residuals!

H.264/AVC video compression overview

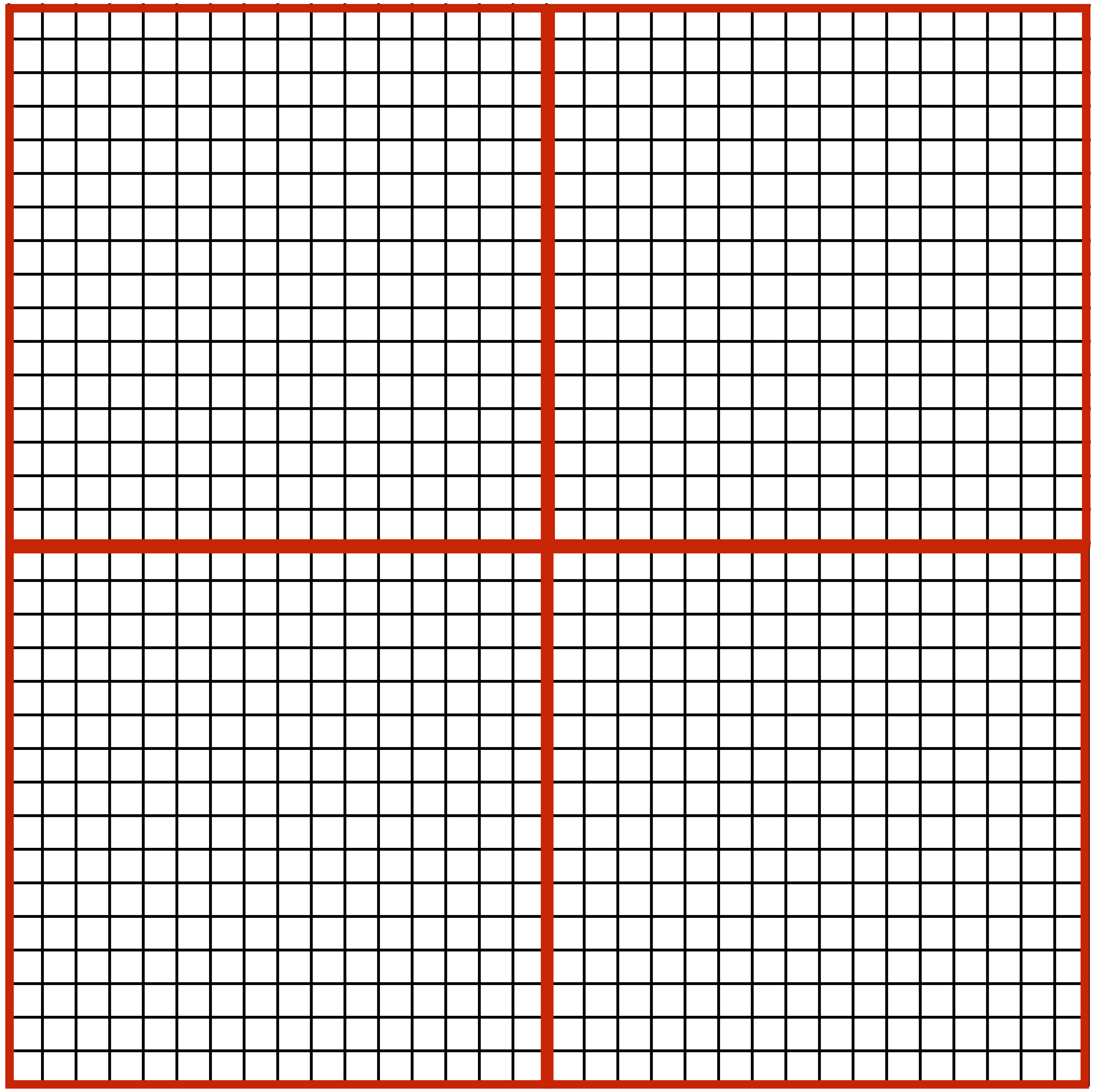


Residual: difference between predicted pixel values and input video pixel values

In other words: use an algorithm to predict what a future pixel should be, then store a description of the algorithm and the residual of the prediction.

Credit: Figure derived from H.264 Advanced Video Compression Standard, I. Richardson, 2010

16 x 16 macroblocks



Video frame is partitioned into 16 x 16 pixel macroblocks

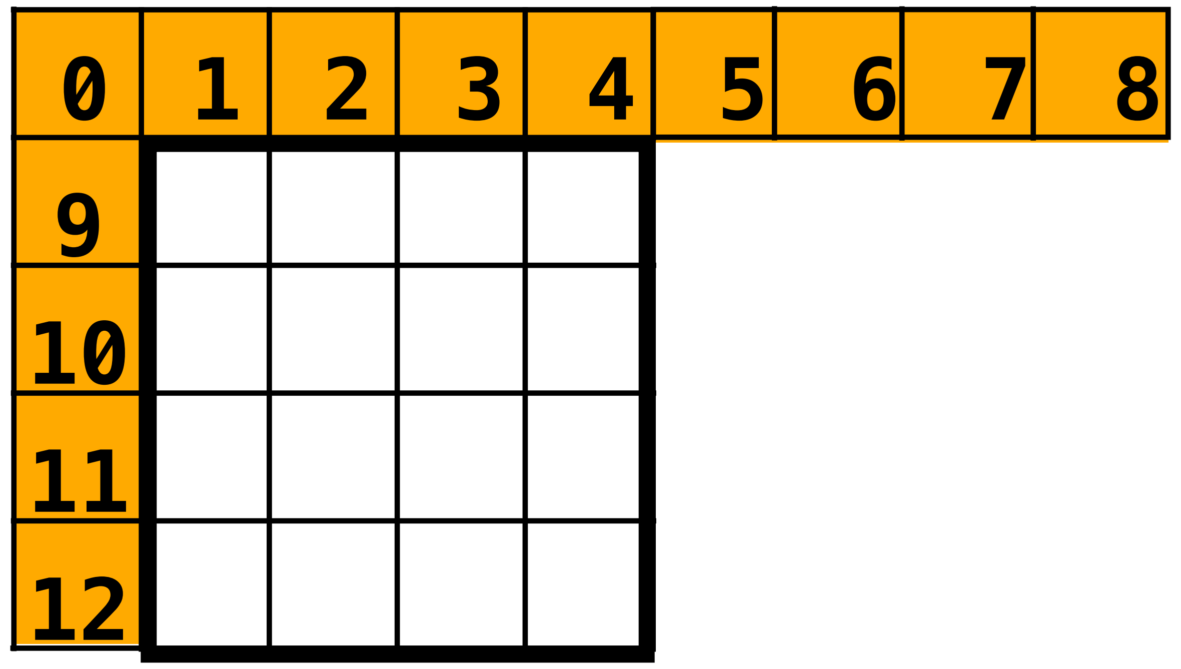
Due to 4:2:0 chroma subsampling, macroblocks correspond to 16 x 16 luma samples and 8 x 8 chroma samples

Decoding via prediction + correction

- **During decode, samples in a macroblock are generated by:**
  1. **Making a prediction based on already decoded samples in macroblocks from the same frame (intra-frame prediction) or from other frames (inter-frame prediction)**
  2. **Correcting the prediction with a residual stored in the video stream**
- **Three forms of prediction:**
  - **I-macroblock: ("intra-picture predictive only") macroblock samples predicted from samples in previous macroblocks in the same slice of the current frame**
  - **P-macroblock: ("predictive") macroblock pixel samples can be predicted from samples from one other frame (one prediction per macroblock)**
  - **B-macroblock: ("bipredictive") macroblock pixel samples can be predicted by a weighted combination of multiple predictions from samples from other frames**

Intra-frame prediction (I-macroblock)

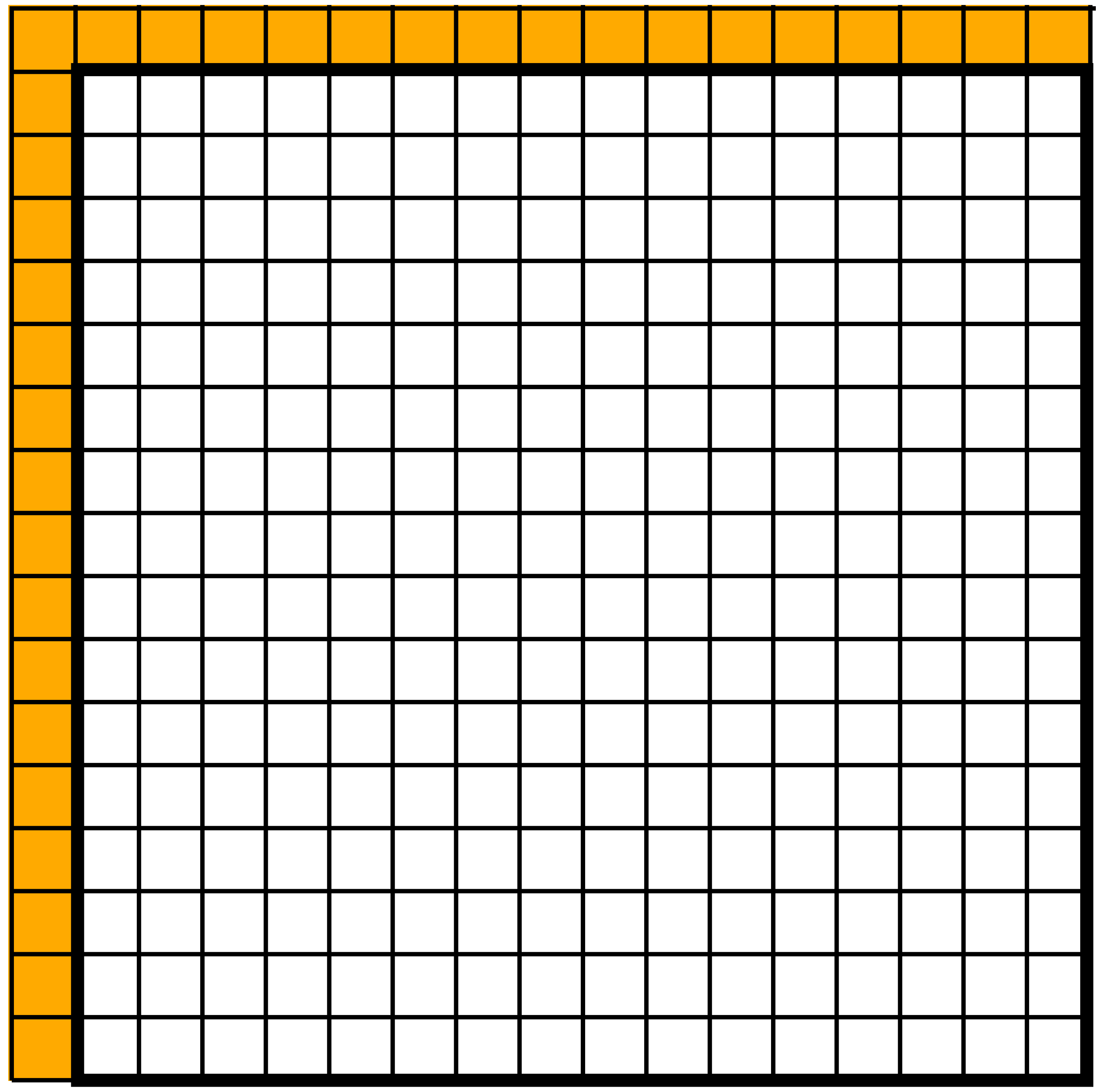
- Prediction of sample values is performed in spatial domain, not transform domain
  - Predict pixel values, not basis coefficients
- Modes for predicting the 16x16 luma (Y) values: *
  - Intra_4x4 mode: predict 4x4 block of samples from adjacent row/col of pixels
  - Intra_16x16 mode: predict entire 16x16 block of pixels from adjacent row/col
  - I_PCM: actual sample values provided



Intra_4X4

Yellow pixels: already reconstructed (values known)

White pixels: 4x4 block to be reconstructed

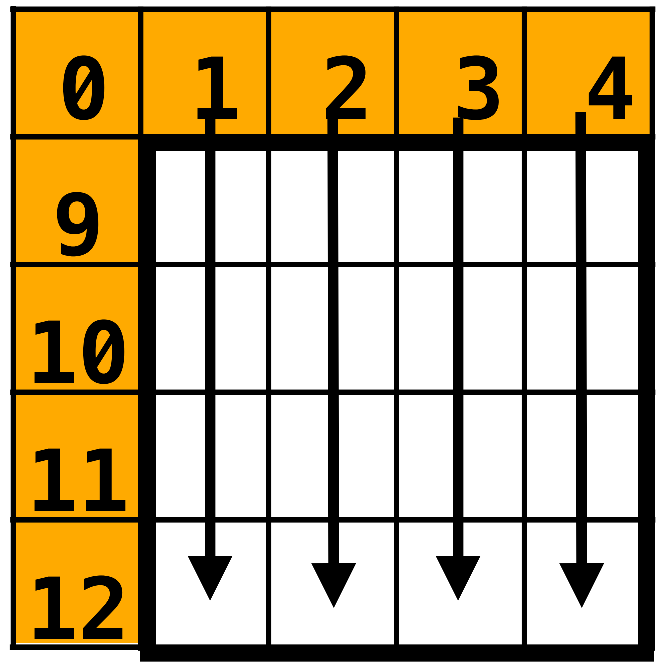


Intra_16x16

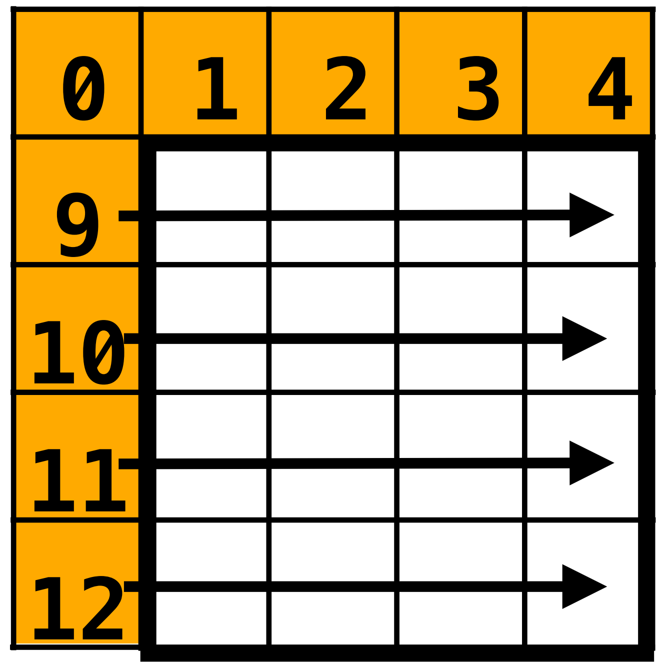
* An additional 8x8 mode exists in the H.264 High Profile

Intra_4x4 prediction modes

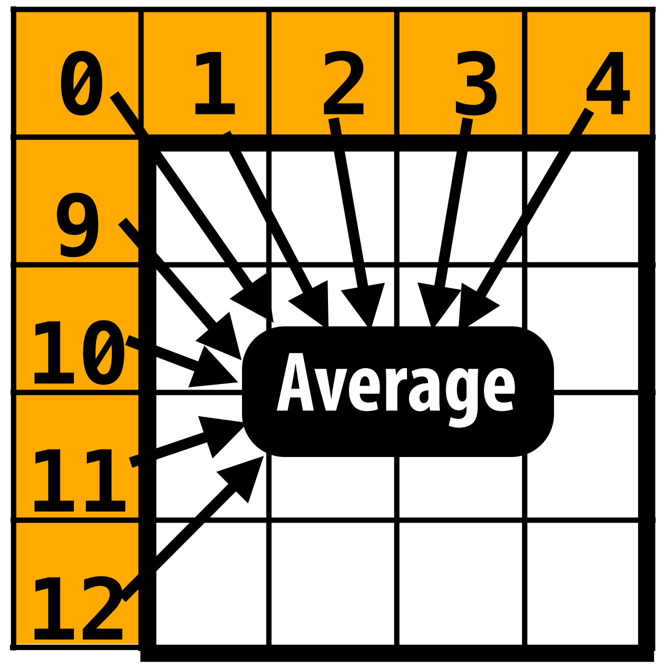
- Nine prediction modes (6 shown below)
  - Other modes: horiz-down, vertical-left, horiz-up



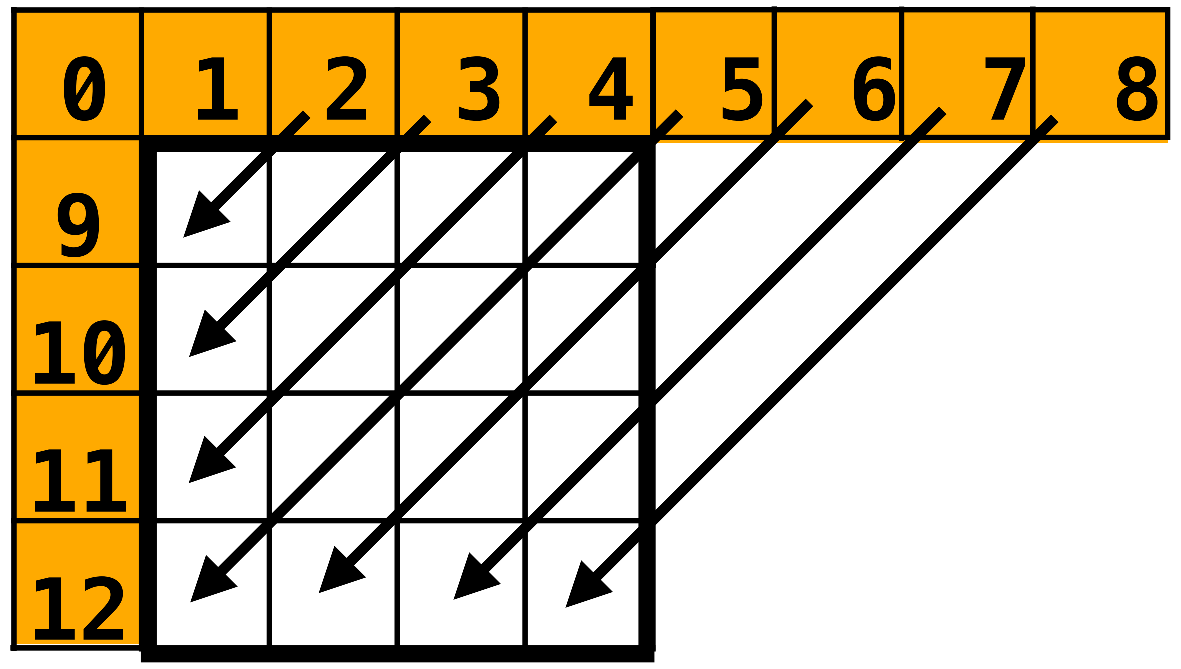
Mode 0: vertical
(4x4 block is copy of
above row of pixels)



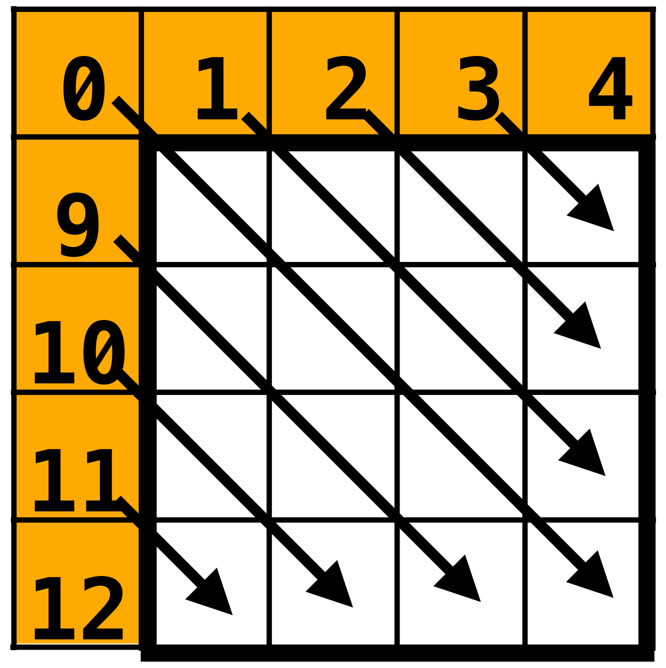
Mode 1: horizontal
(4x4 block is copy of left
col of pixels)



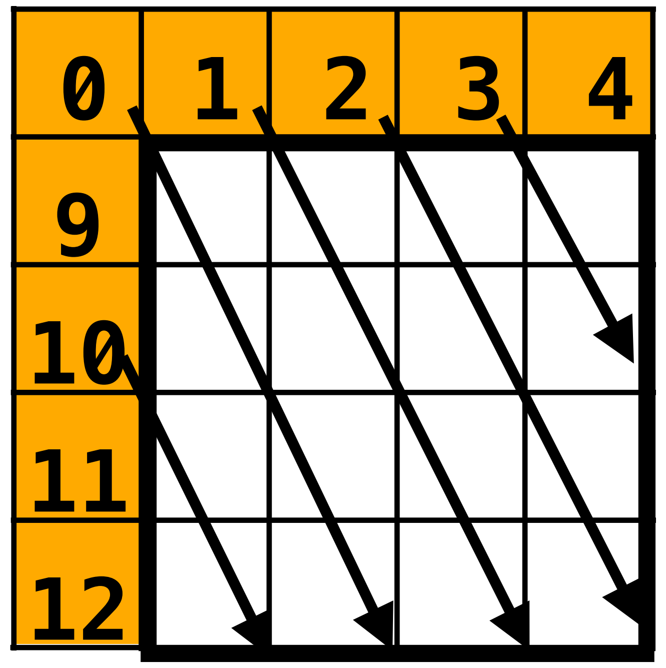
Mode 2: DC
(4x4 block is average of above
row and left col of pixels)



Mode 3: diagonal down-left (45^{o})

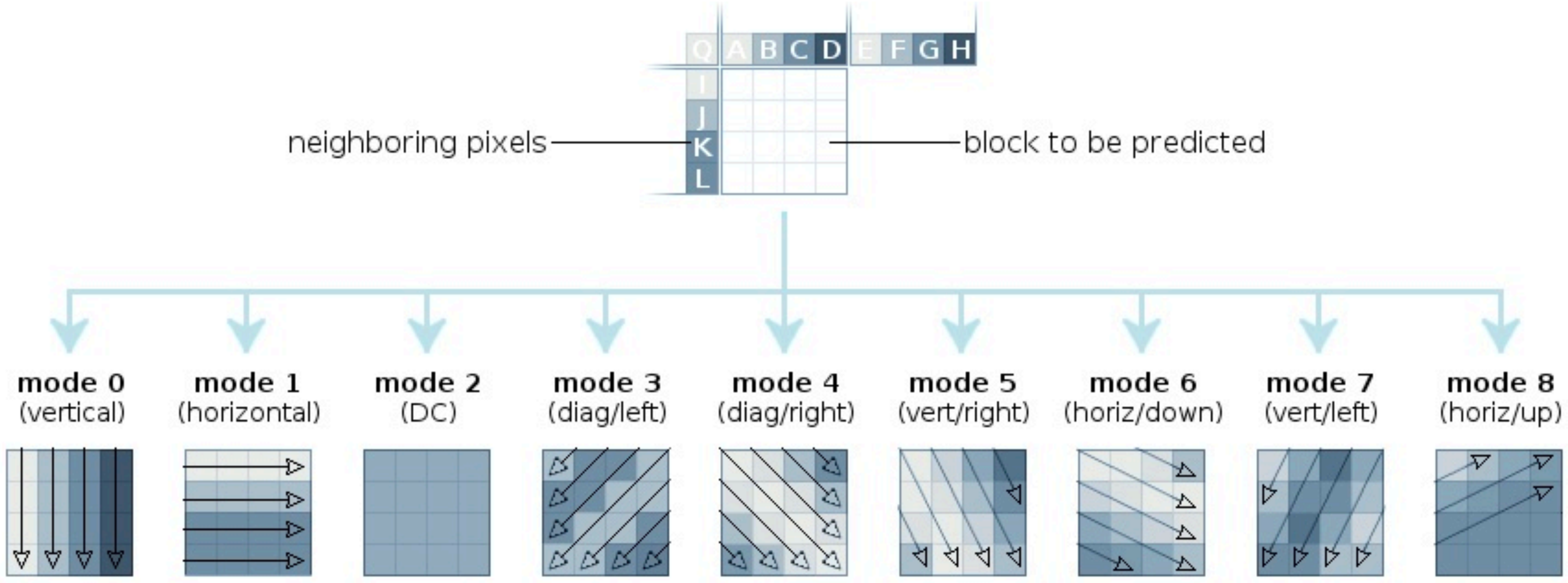


Mode 4: diagonal down-right (45^{o})



Mode 5: vertical-right (26.6^{o})

Intra_4x4 prediction modes (another look)

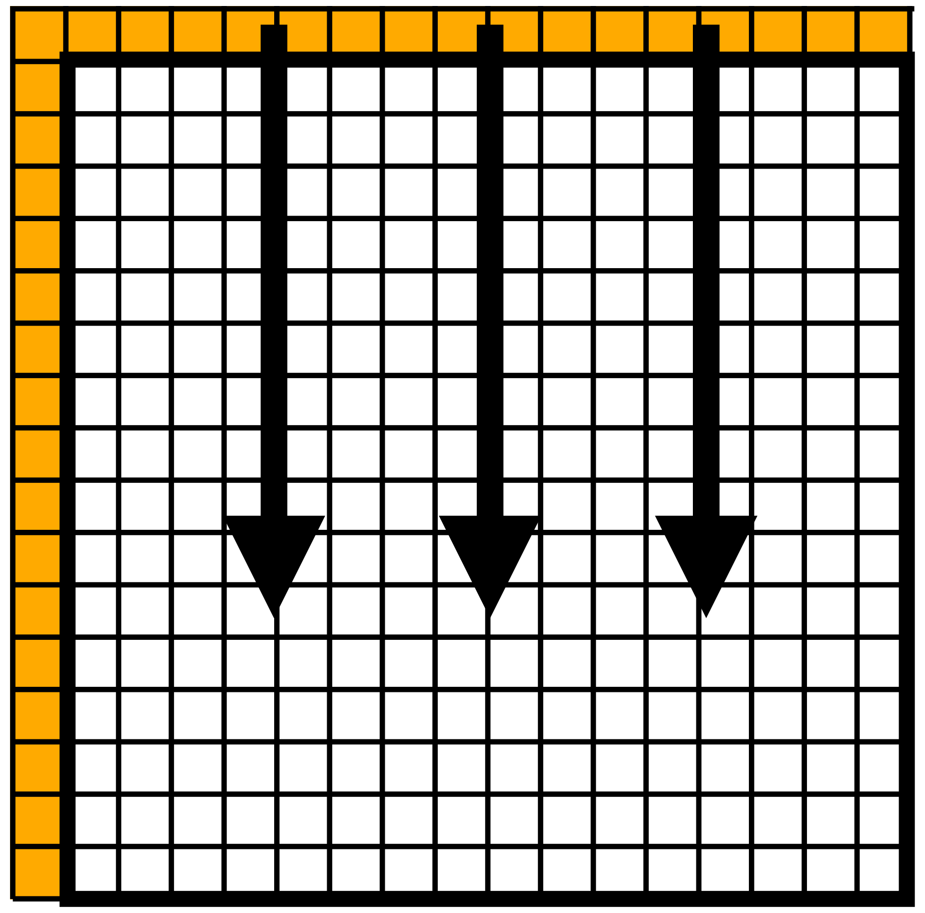


AVC/H.264 intra prediction modes

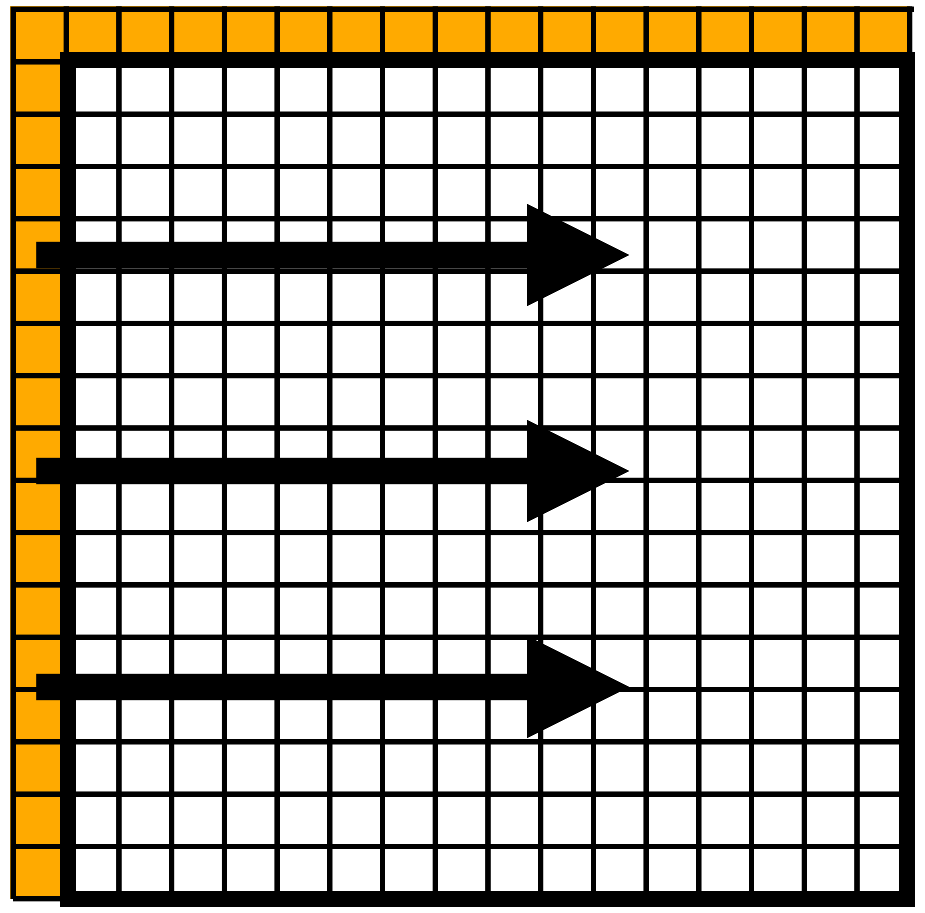
https://hacks.mozilla.org/2018/06/av1-next-generation-video-the-constrained-directional-enhancement-filter/

Intra_16x16 prediction modes

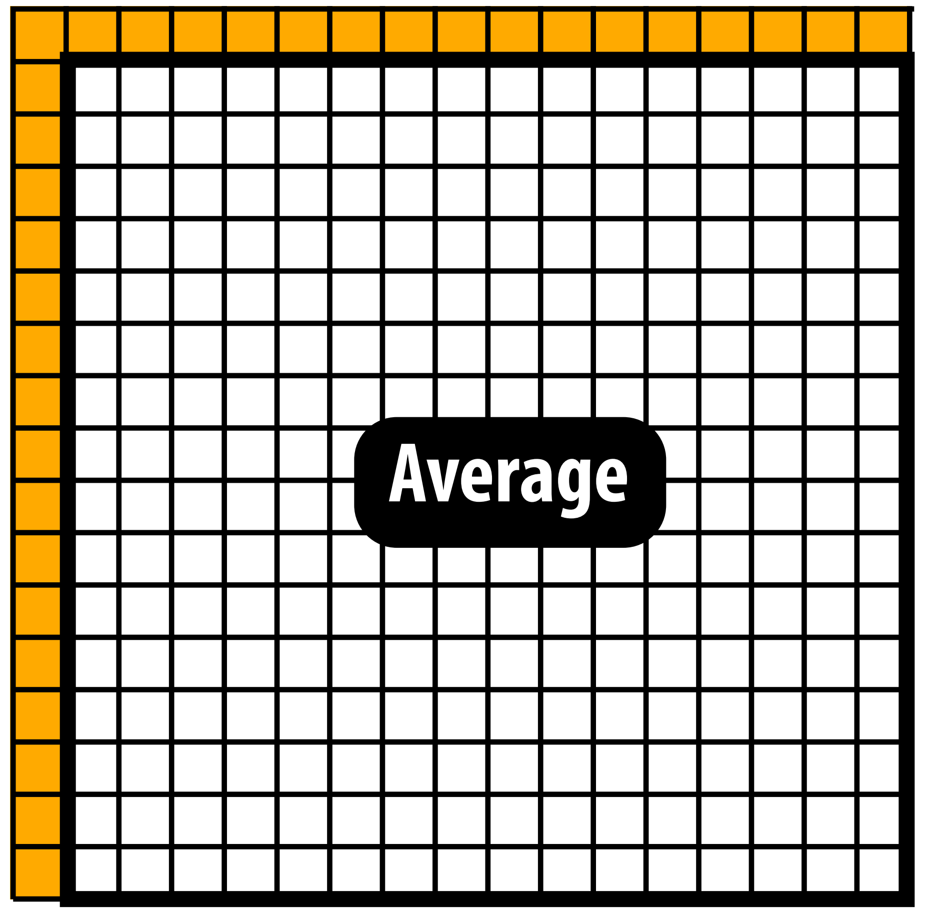
4 prediction modes: vertical, horizontal, DC, plane



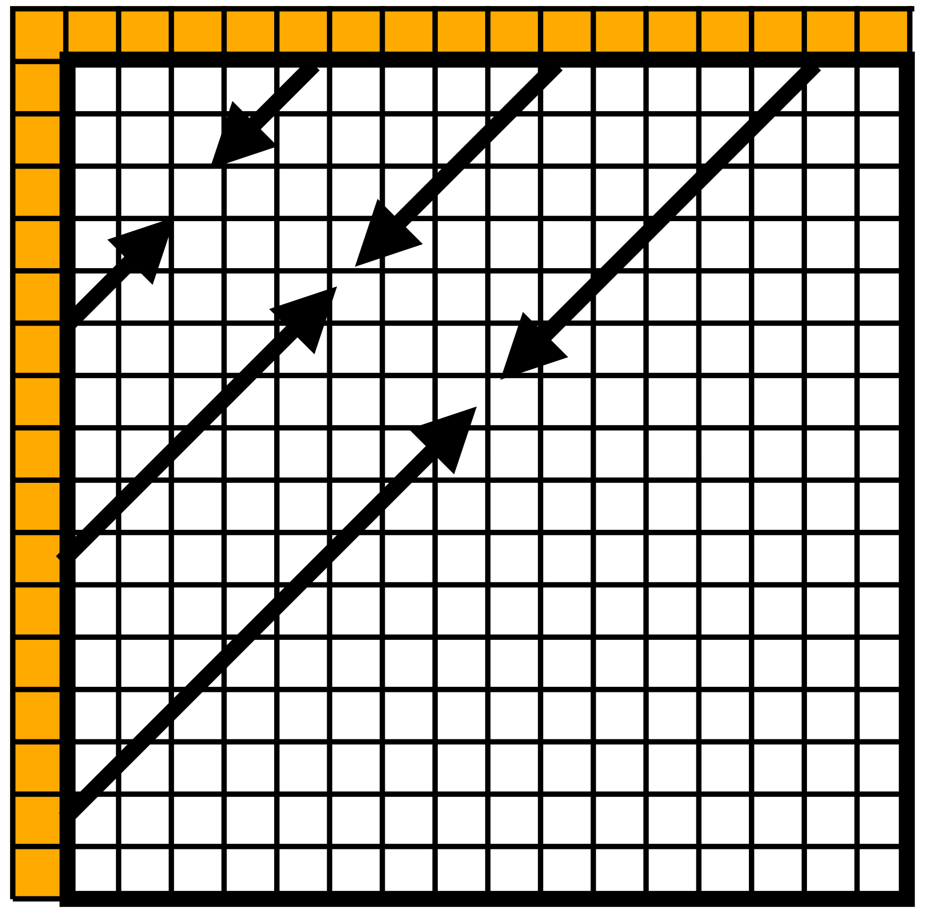
Mode 0: vertical



Mode 1: horizontal



Mode 2: DC



Mode 4: plane

P[i,j] = Ai * Bj + C

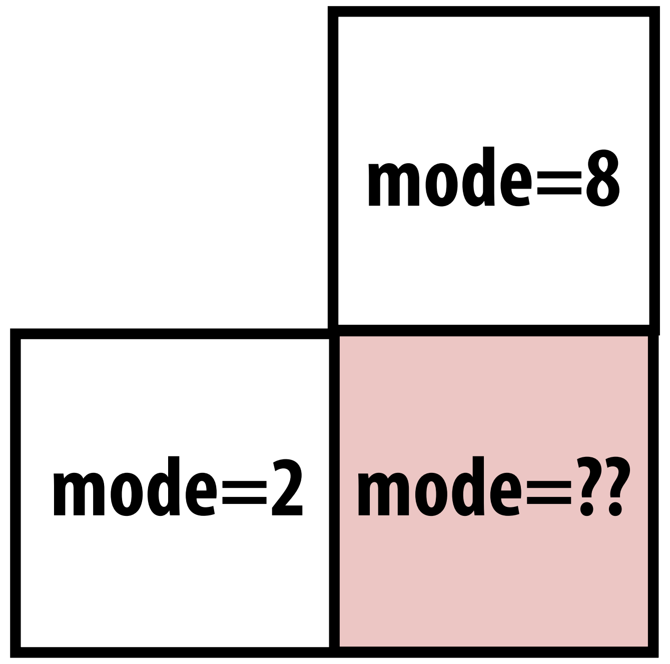
A derived from top row, B derived from left col, C from both

Further details

- Intra-prediction of chroma (8x8 block) is performed using four modes similar to those of intra_16x16 (except they are reordered as: DC, vertical, horizontal, plane)

Each mode is a different prediction algorithm, so we have to store which algorithm we chose in the video stream in order to decode it.

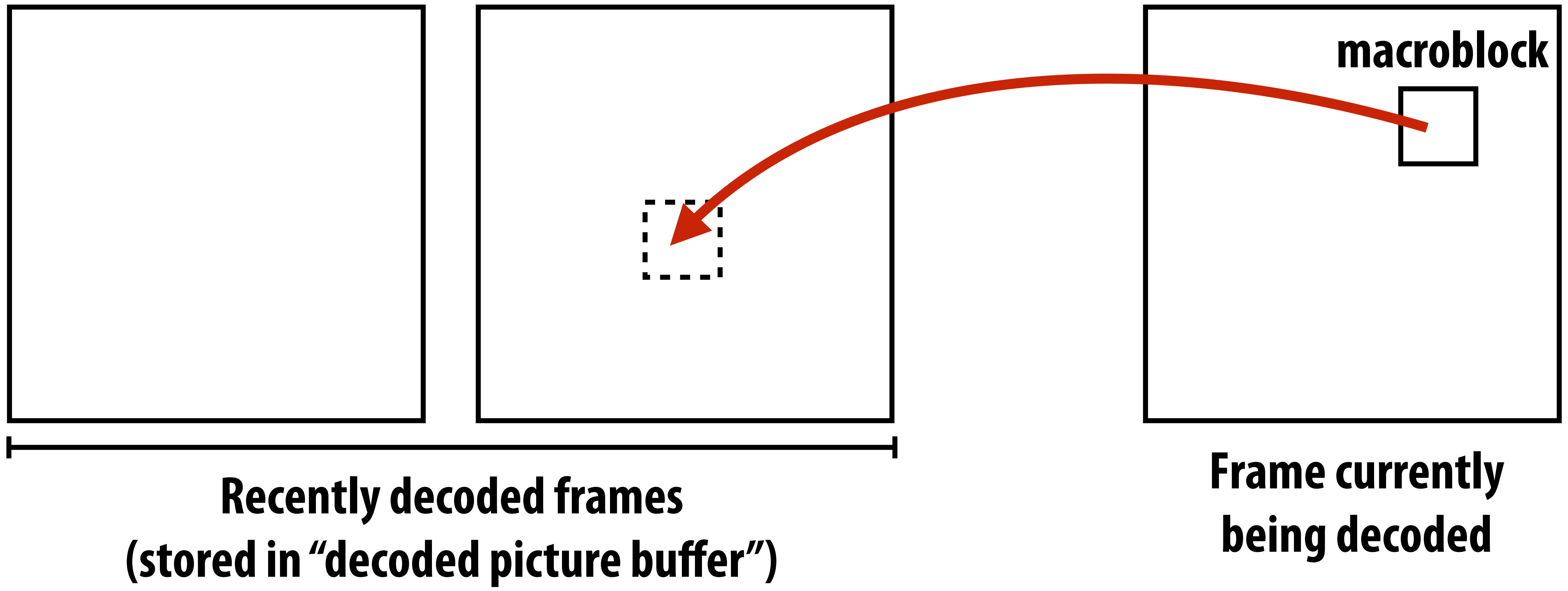
- Intra-prediction scheme for each 4x4 block within macroblock encoded as follows:
  - One bit per 4x4 block:
    - if 1, use most probable mode
      - Most probable = lower of modes used for 4x4 block to left or above current block
    - if 0, use additional 3-bit value `rem_intra4x4_pred_mode` to encode one of nine modes
      - if `intra4x4_pred_mode` is smaller than most probable mode, then actual mode is given by `intra4x4_pred_mode`
      - else, actual mode is `intra4x4_pred_mode + 1`



Inter-frame prediction (P-macroblock)

- Predict sample values using values from a block of a previously decoded frame *

- Basic idea: pixels in current frame are given by some translation of pixels from temporally nearby frames (e.g., consider an object that moved slightly on screen between frames)
  - "Motion compensation": use of spatial displacement to make prediction about pixel values

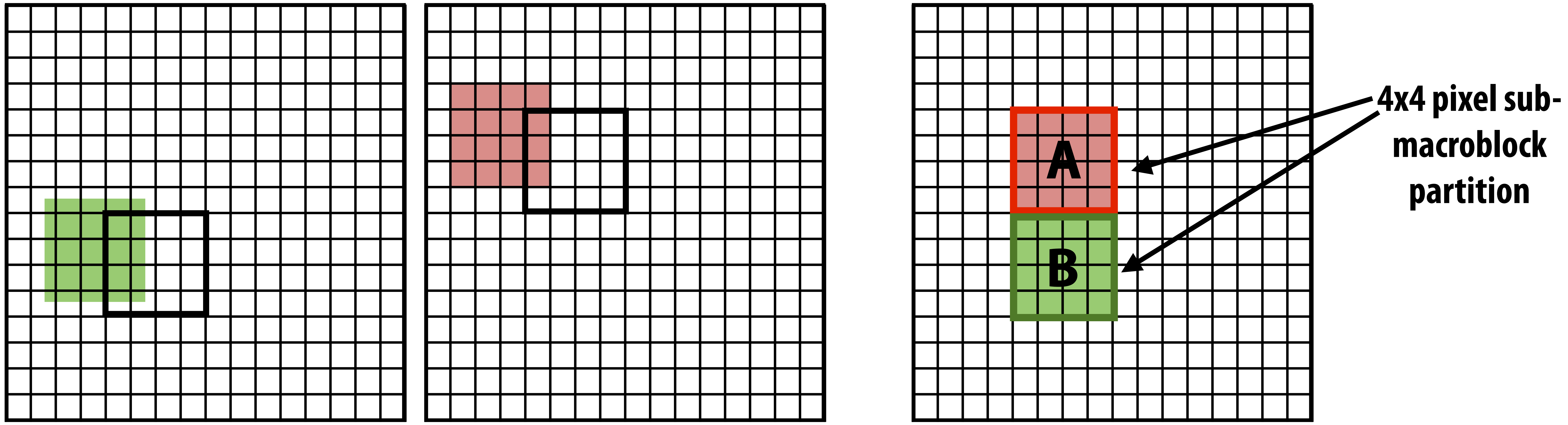


* Note: "previously decoded" does not imply source frame must come before current frame in the video sequence. (H.264 supports decoding out of order.)

P-macroblock prediction

- Prediction can be performed at macroblock or sub-macroblock granularity
  - Macroblock can be divided into 16x16, 8x16, 16x8, 8x8 “partitions”
  - 8x8 partitions can be further subdivided into 4x8, 8x4, 4x4 sub-macroblock partitions

- Each partition predicted by sample values defined by:
  (reference frame id, motion vector)



4x4 pixel sub-macroblock partition

Decoded picture buffer: frame 1

Decoded picture buffer: frame 0

Current frame

Block A: predicted from (frame 0, motion-vector = [-3, -1])

Block B: predicted from (frame 1, motion-vector = [-2.5, -0.5])

Note: non-integer motion vector

Motion vector visualization

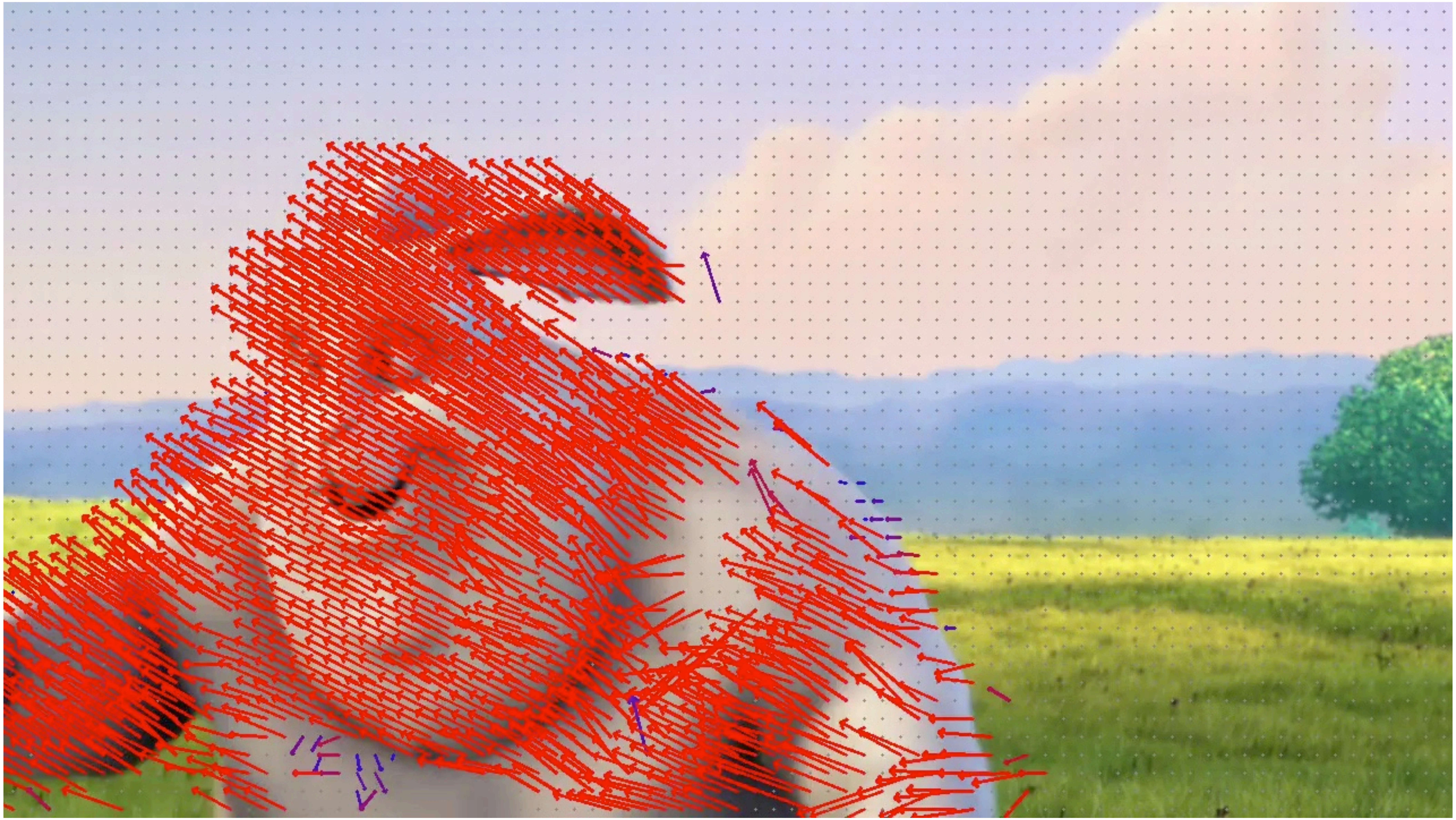
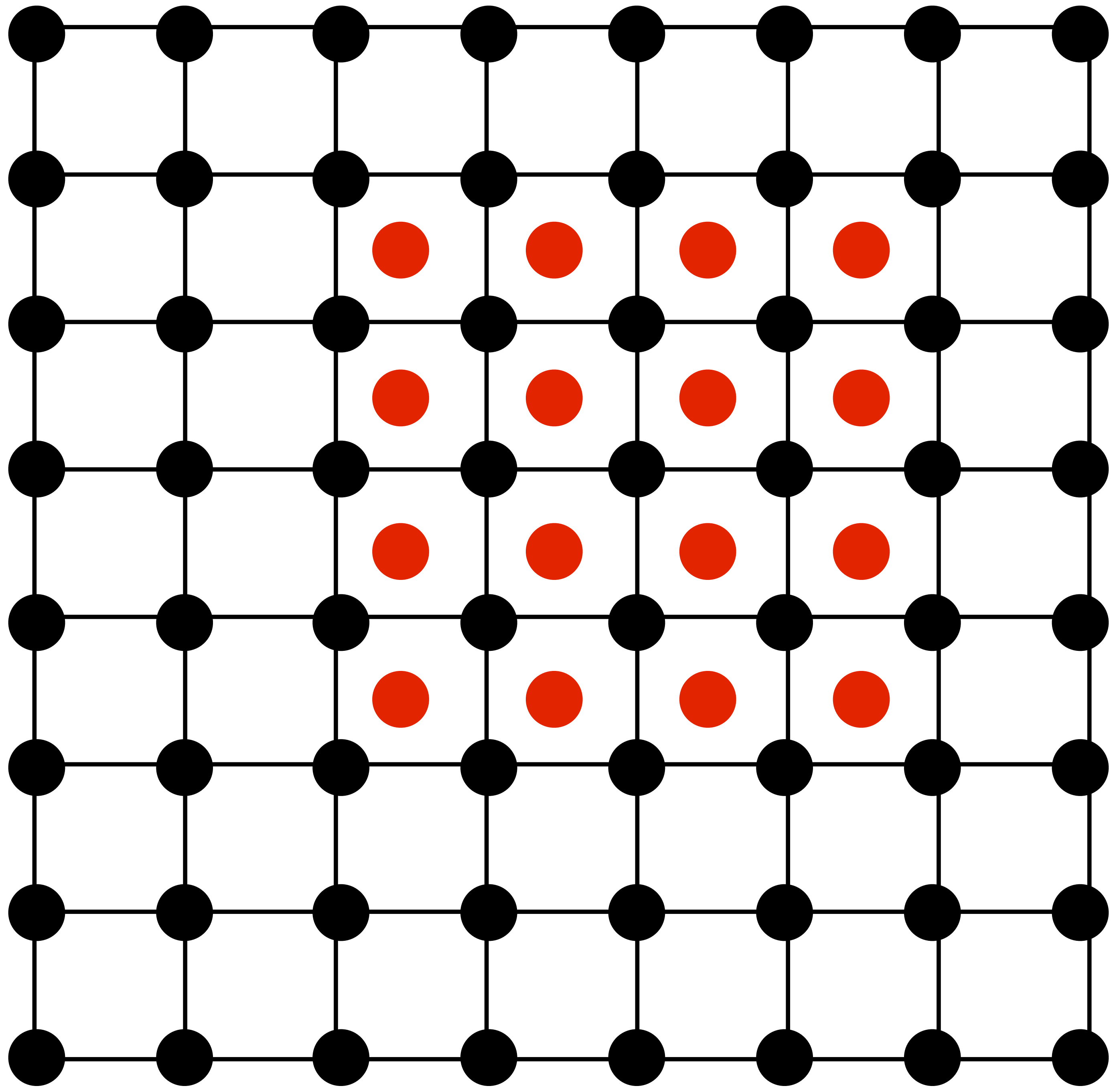
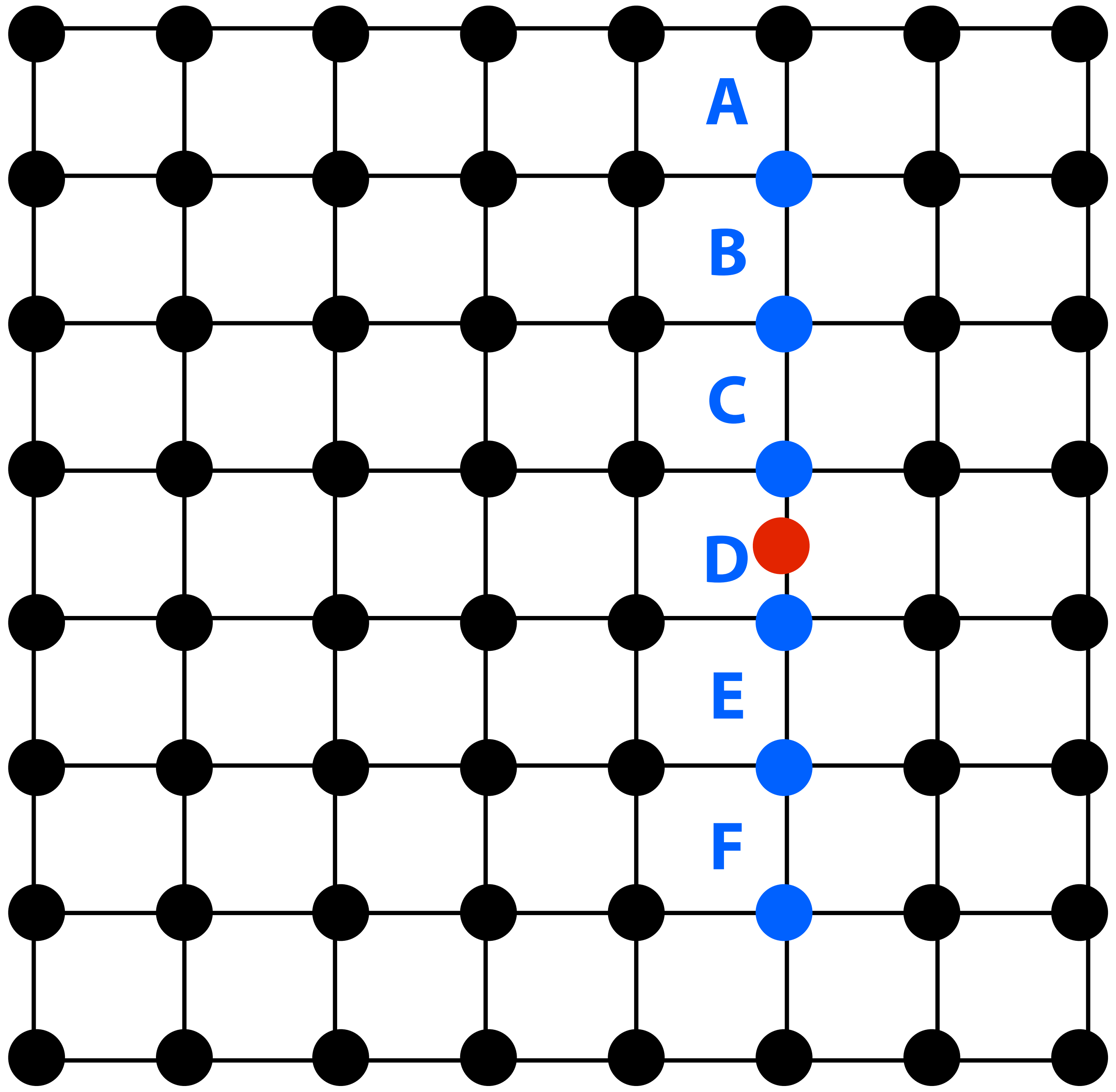


Image credit: Keyi Zhang

Non-integer motion vectors require resampling



Example: motion vector with 1/2 pixel values.
Must resample reference block at positions given by red dots.



Interpolation to 1/2 pixel sample points via 6-tap filter:
half_integer_value = clamp((A - 5B + 20C + 20D - 5E + F) / 32)

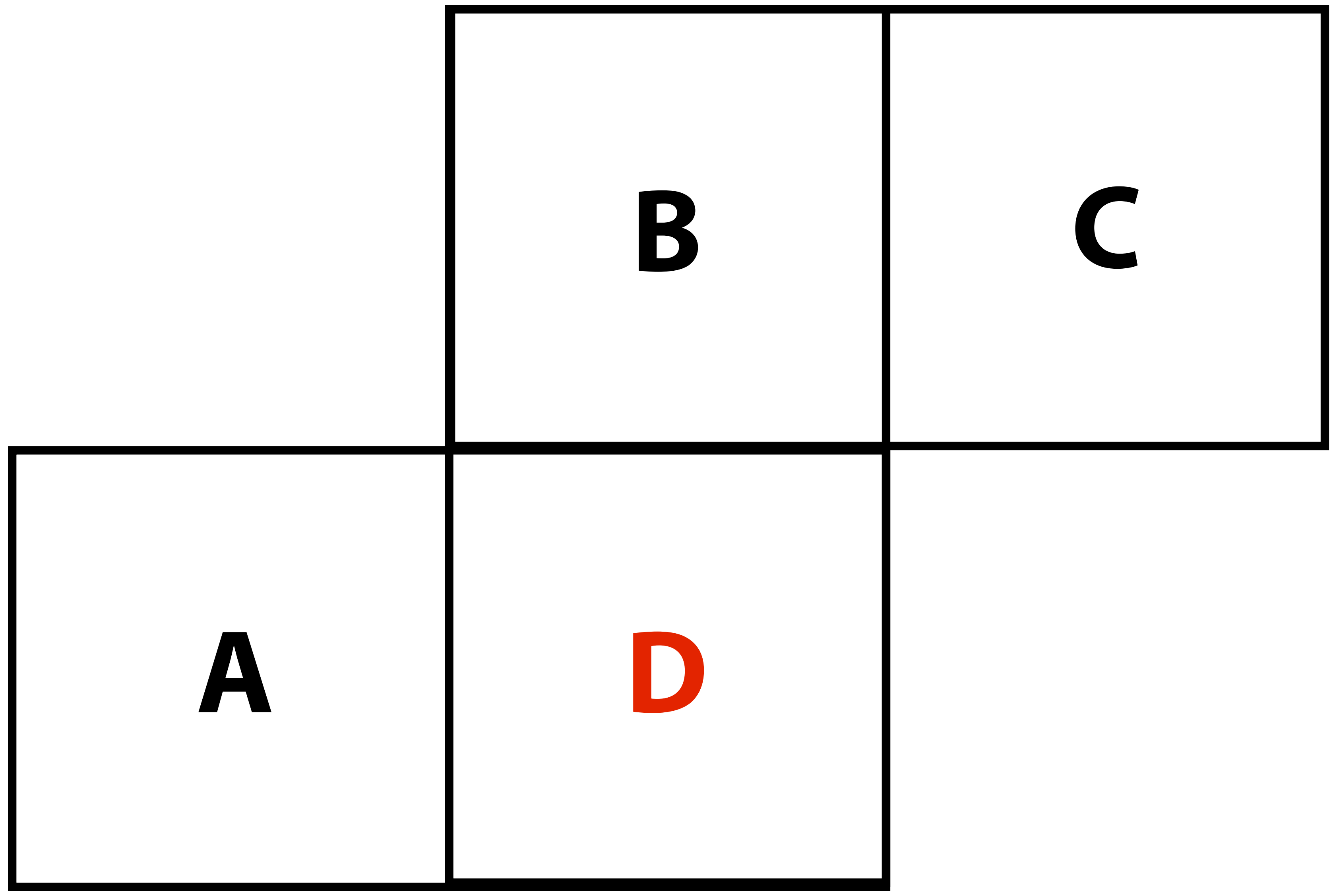
H.264 supports both 1/2 pixel and 1/4 pixel resolution motion vectors

1/4 resolution resampling performed by bilinear interpolation of 1/2 pixel samples

1/8 resolution (chroma only) by bilinear interpolation of 1/4 pixel samples

Motion vector prediction

- Problem: per-partition motion vectors require significant amount of storage
- Solution: predict motion vectors from neighboring partitions and encode residual in compressed video stream
    - Example below: predict block D's motion vector as average of motion vectors from block A, B, C
    - Prediction logic becomes more complex when partitions of neighboring blocks are of different size



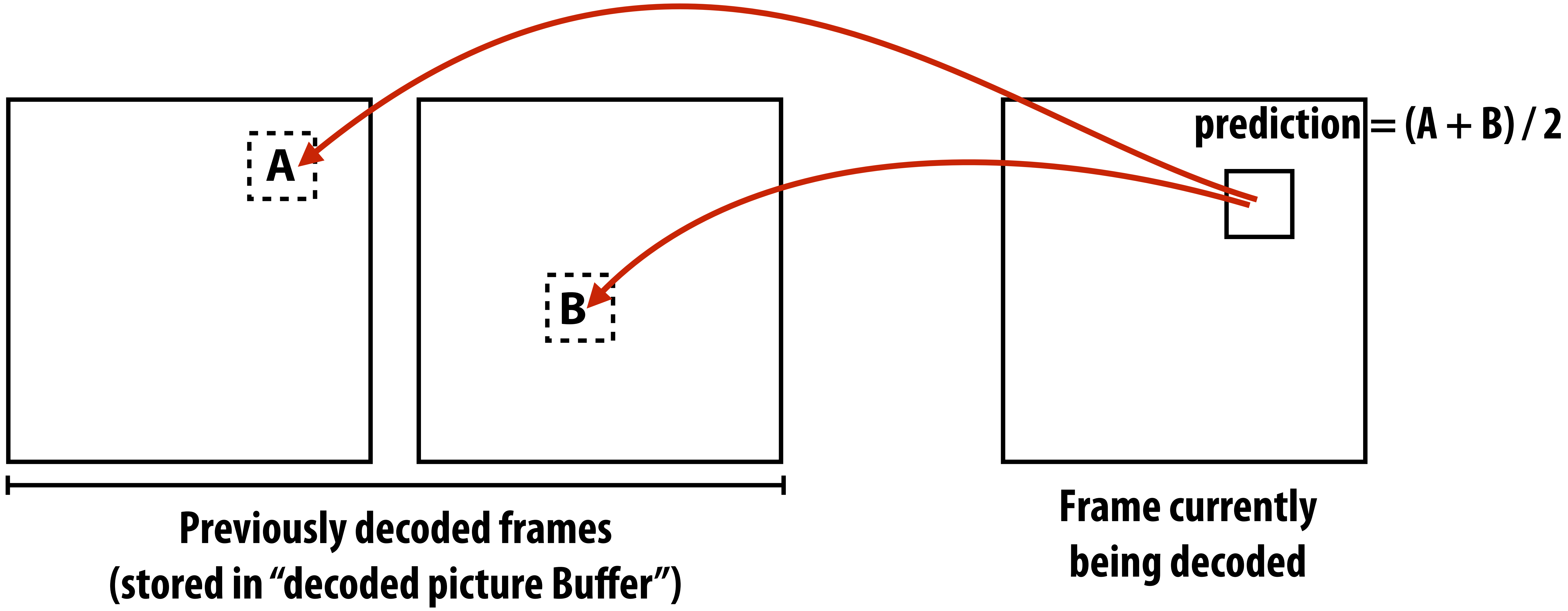
Question: what partition size is best?

- Smaller partitions likely yield more accurate prediction
    - Fewer bits needed for residuals

- Smaller partitions require more bits to store partition information (diminish benefits of prediction)
    - Must store:
        - Source picture id
        - Motion vectors (note that motion vectors are more "coherent" in adjacent blocks with finer sampling, so they likely compress well)

Inter-frame prediction (B-macroblock)

- Each partition predicted by up to two source blocks
  - Prediction is the average of the two reference blocks
  - Each B-macroblock partition stores two frame references and two motion vectors (recall P-macroblock partitions only stored one)


A
B
prediction = (A + B) / 2
Previously decoded frames
(stored in "decoded picture Buffer")
Frame currently
being decoded

Additional prediction details

- **Optional weighting to prediction:**
  - Per-slice explicit weighting (reference samples multiplied by weight)
  - Per-B-slice implicit weights (reference samples weights by temporal distance of reference frame from current frame in video)
    - Idea: weight samples from reference frames nearby in time more

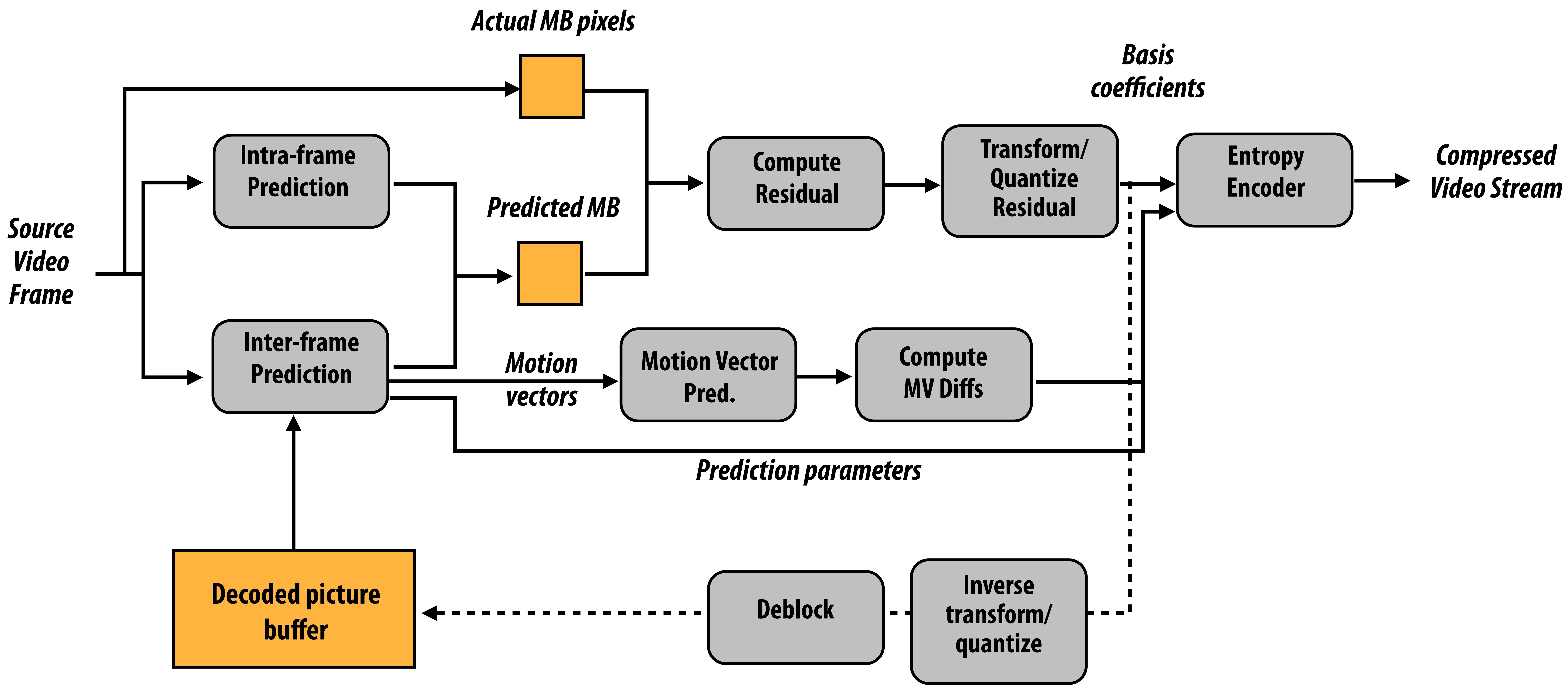
Putting it all together: encoding an inter-predicted macroblock

- Inputs:
  - Current state of decoded picture buffer (state of the video decoder)
  - 16x16 block of input video that the encoder needs to encode
- General steps: (need not be performed in this order)
  - Resample images in decoded picture buffer to obtain 1/2, and 1/4, 1/8 pixel resampling
  - Choose prediction type (P-type or B-type)
  - Choose reference pictures for prediction
  - Choose motion vectors for each partition (or sub-partition) of macroblock
  - Predict motion vectors and compute motion vector difference
  - Encode choice of prediction type, reference pictures, and motion vector differences
  - Encode residual for macroblock prediction
  - Store reconstructed macroblock in decoded picture buffer to use as reference picture for future macroblocks

Coupled decisions

H.264/AVC video encoding

MB = macroblock
MV = motion vector

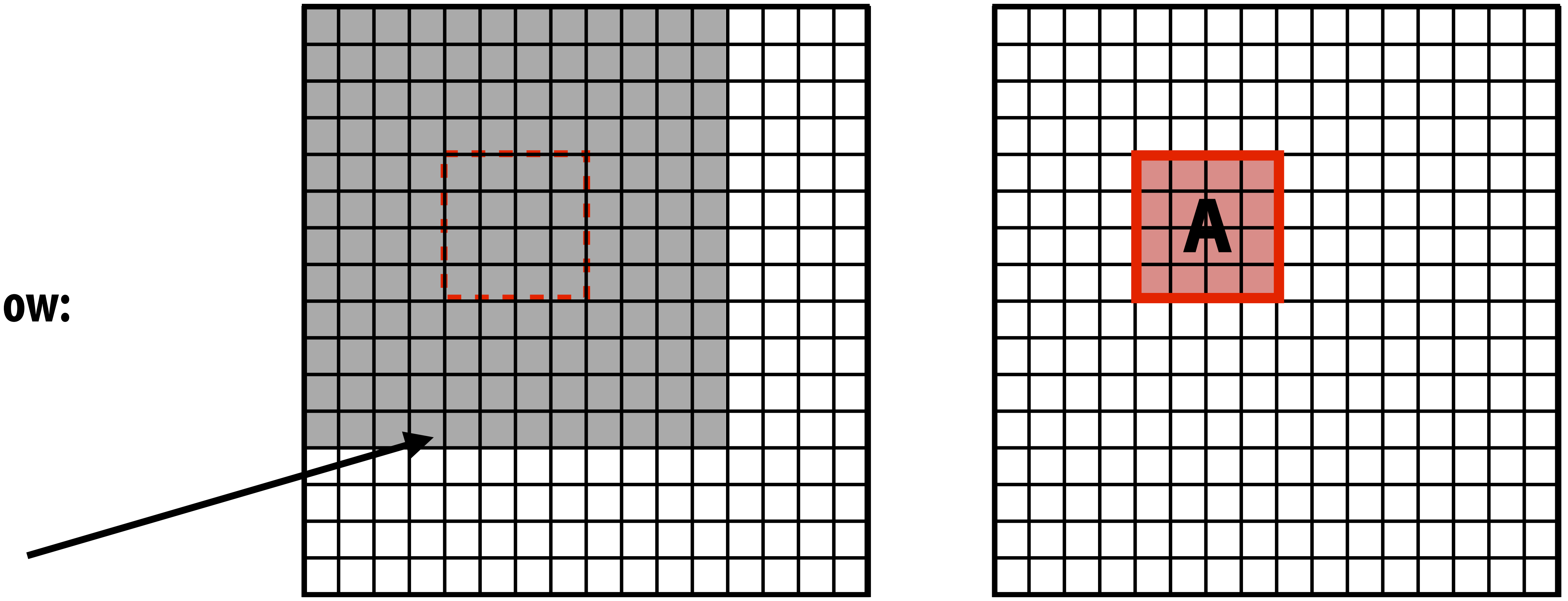


Credit: Figure derived from H.264 Advanced Video Compression Standard, I. Richardson, 2010

Motion estimation algorithms

- Encoder must find reference block that predicts current frame's pixels well.
    - Can search over multiple pictures in decoded picture buffer + motion vectors can be non-integer (huge search space)
    - Must also choose block size (macroblock partition size)
    - And whether to predict using combination of two blocks
    - Literature is full of heuristics to accelerate this process
        - Remember, must execute motion estimation in real-time for HD video (1920x1080) on a low-power smartphone

Limit search window:



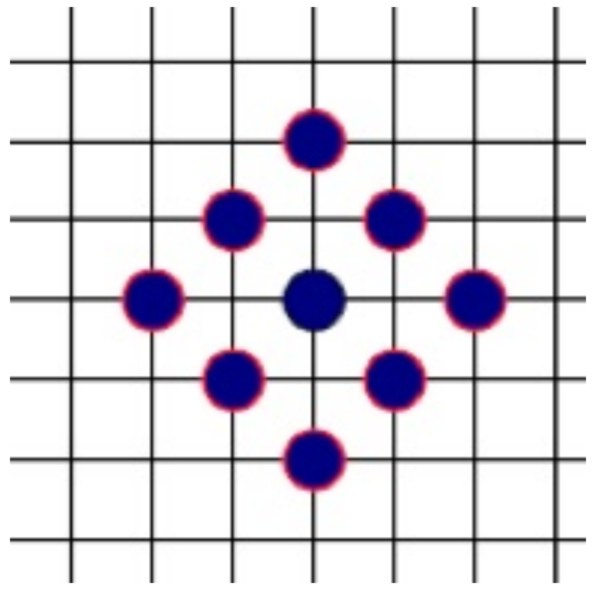
gray area: search region

Decoded picture buffer: frame 0

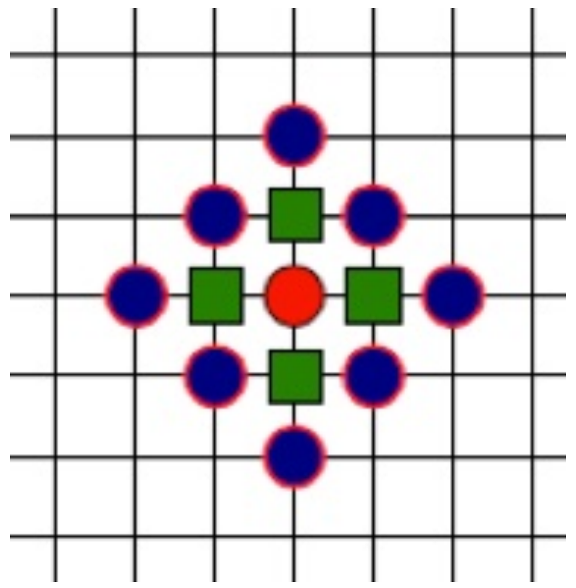
Current frame

Motion estimation algorithm optimizations

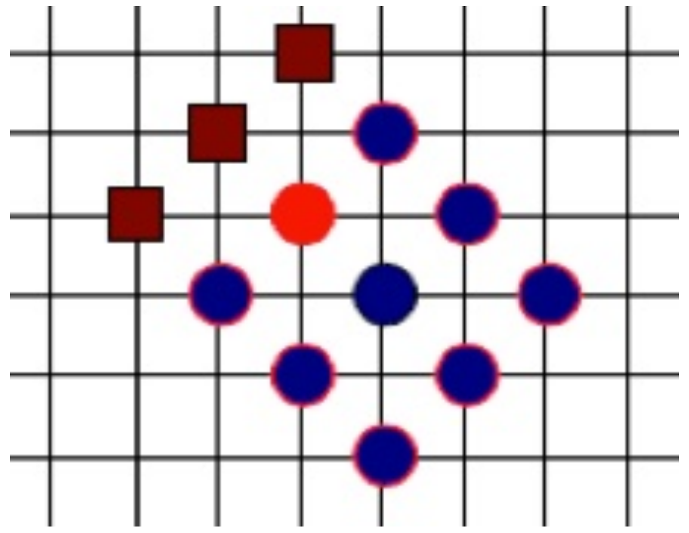
- Coarser search:
  - Limit search window to small region
  - First compute block differences at coarse scale (save partial sums from previous searches)
- Smarter search:
  - Guess motion vectors similar to motion vectors used for neighboring blocks
  - Diamond search: start by test large diamond pattern centered around block
    - If best match is interior, refine to finer scale
    - Else, recenter around best match



Original



Refined



Recentered

- Early termination: don't find optimal reference patch, just find one that's "good enough": e.g., compressed representation is lower than threshold
  - Test zero-motion vector first (optimize for non-moving background)
- Optimizations for subpixel motion vectors:
  - Refinement: find best reference block given only pixel offsets, then try 1/2, 1/4-subpixel offsets around this match

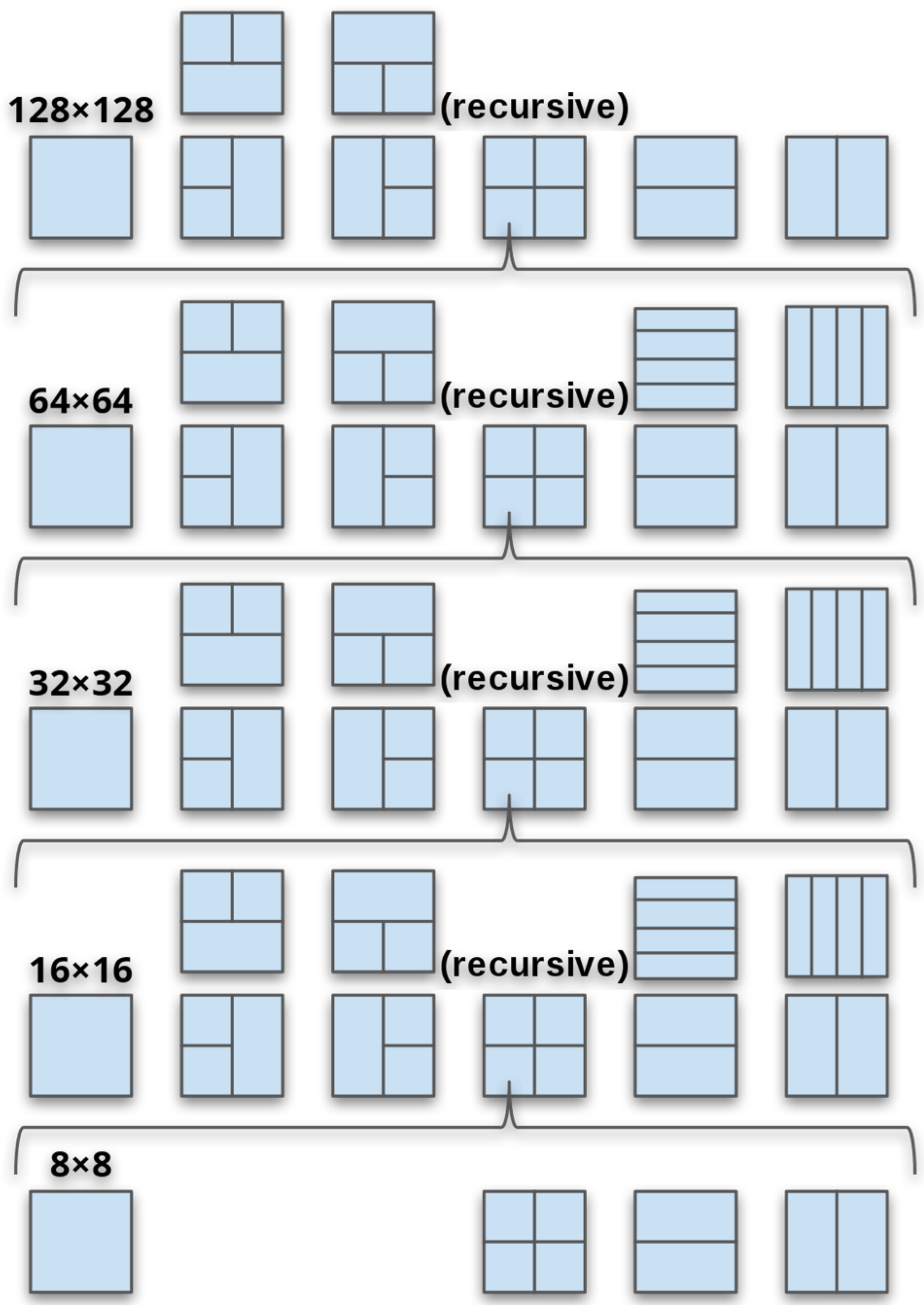
H.265 (HVEC)

- Standard ratified in 2013
- Goal: ~2x better compression than H.264
- Main ideas: (more options, but similar in spirit to what we've discussed so far)
  - Macroblock sizes up to 64x64
  - Prediction block size and residual block sizes can be different
  - 35 intra-frame prediction modes (recall H.264 had 9)
  - …

AV1

Main appeal may not be technical: royalty free codec, but many new options for encoders

AV1 Superblock Partitionings



56 angles for intraframe block prediction!
(recall H.264 had nine!)

Global transforms to geometrically warp previous frames to new frames

Prediction of chroma channels from luma

Synthetic generation of film-grain texture so that high-frequency film grain does not need to be compressed…

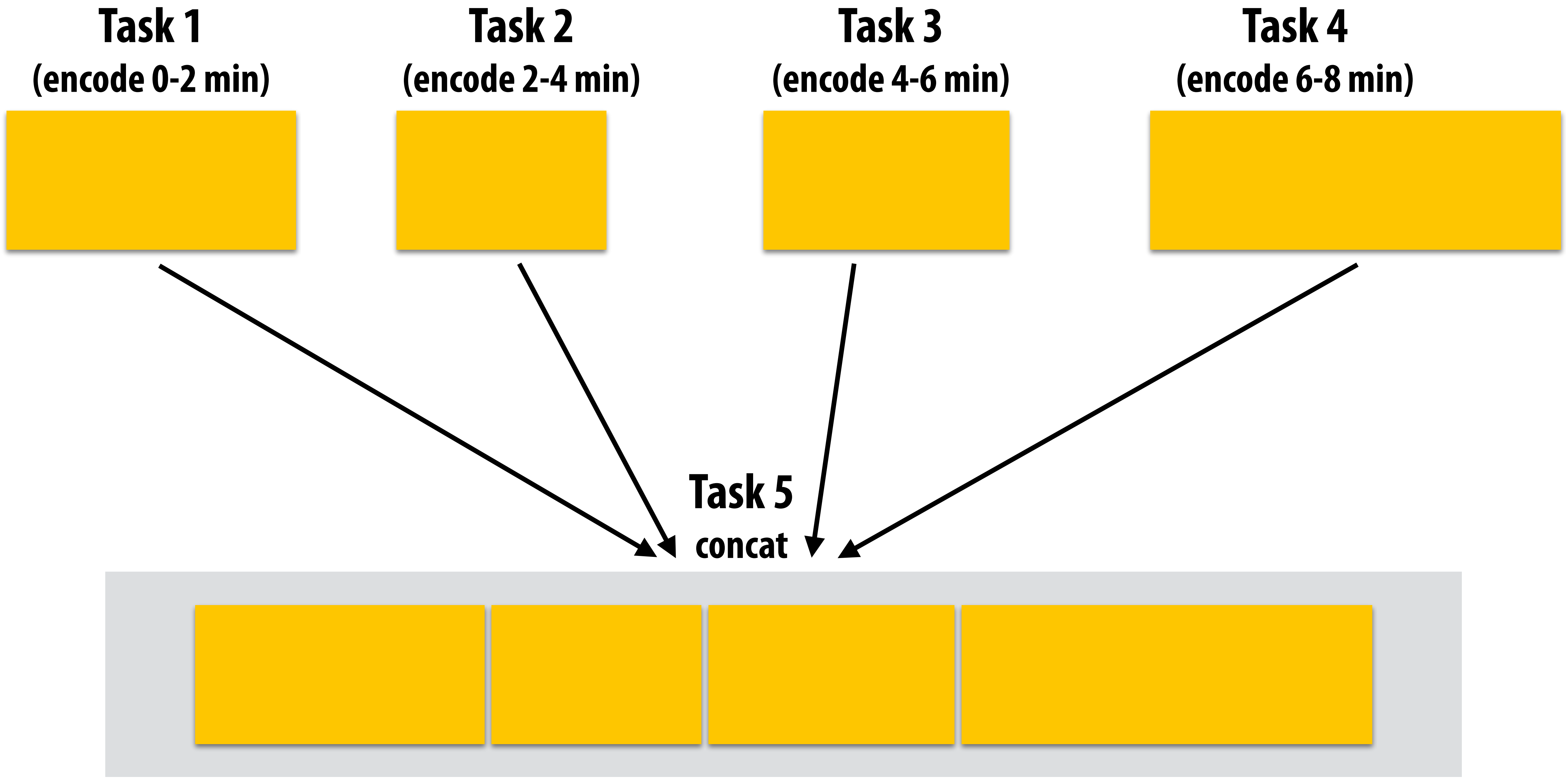
High cost of software encoders

- **Statistic from Google:** [Ranganathan 2021]
    - About 8-10 CPU minutes to compress 150 frames of 2160p H.264 video (4K video)
    - About 1 CPU hour for more expensive VP9 codec

Coarse-grained parallel video encoding

- Parallelized across segments (I-frame inserted at start of segment)
- Concatenate independently encoded bitstreams

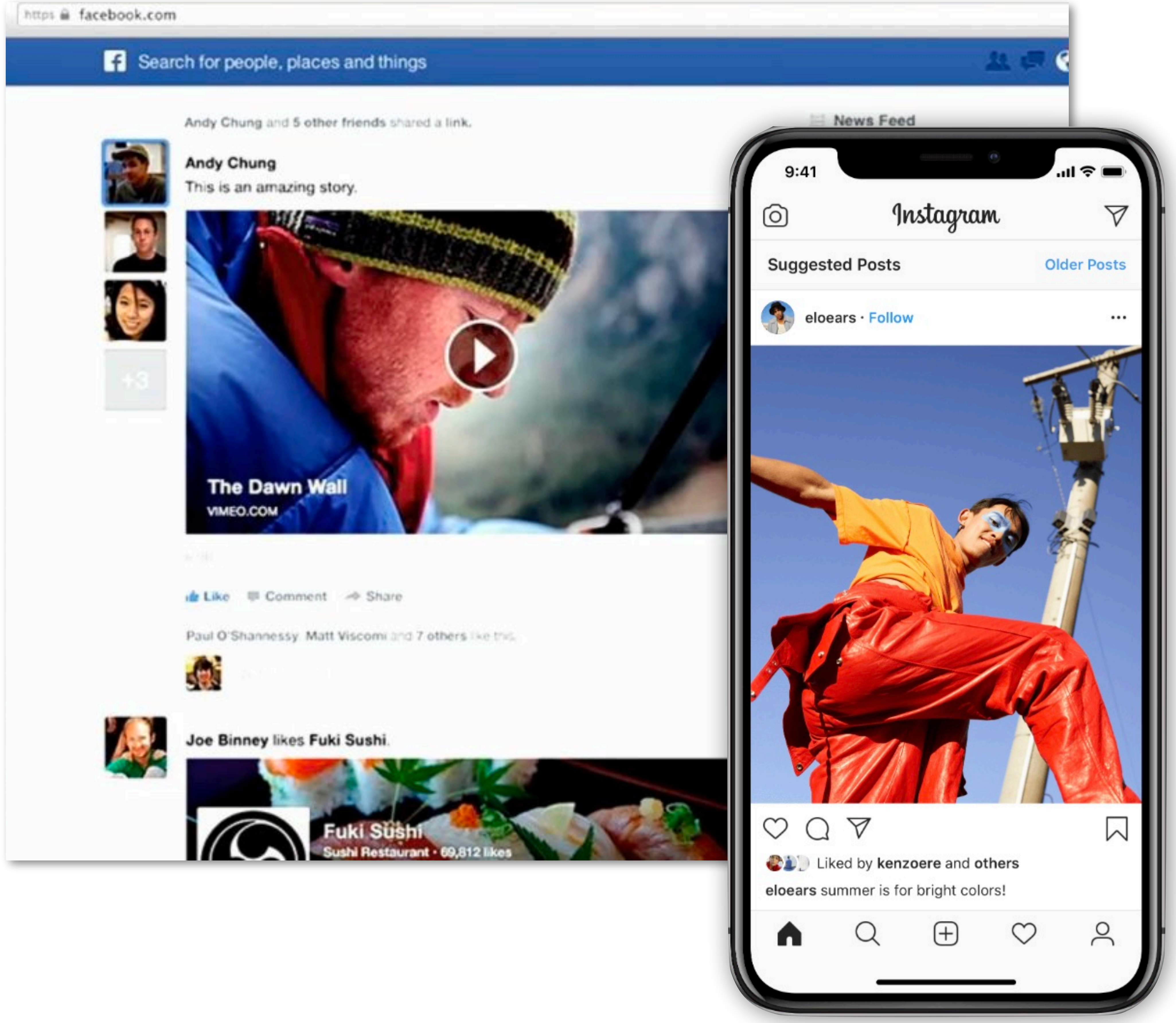
Example: encoding an eight minute video



Smaller segments = more potential parallelism, worse video compression

Three types of encoding from Meta

Facebook / Instagram Posts



Andy Chung and 5 other friends shared a link.

Andy Chung
This is an amazing story.

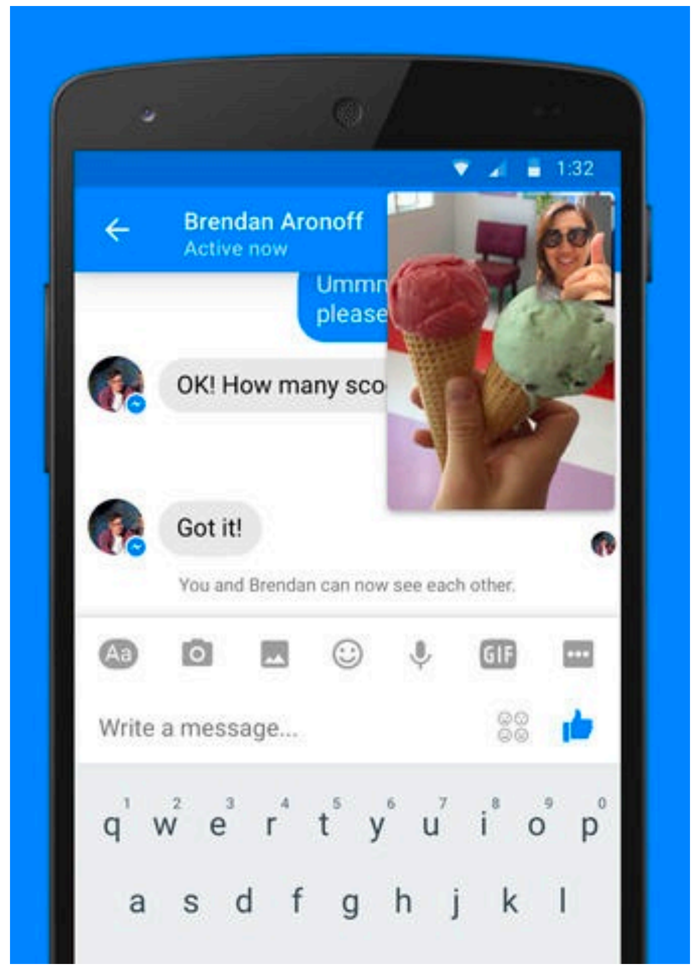
Paul O'Shannessy, Matt Viscomi and 7 others like this.

Joe Binney likes Fuki Sushi.

Liked by kenzoere and others

eloears summer is for bright colors!

Messenger



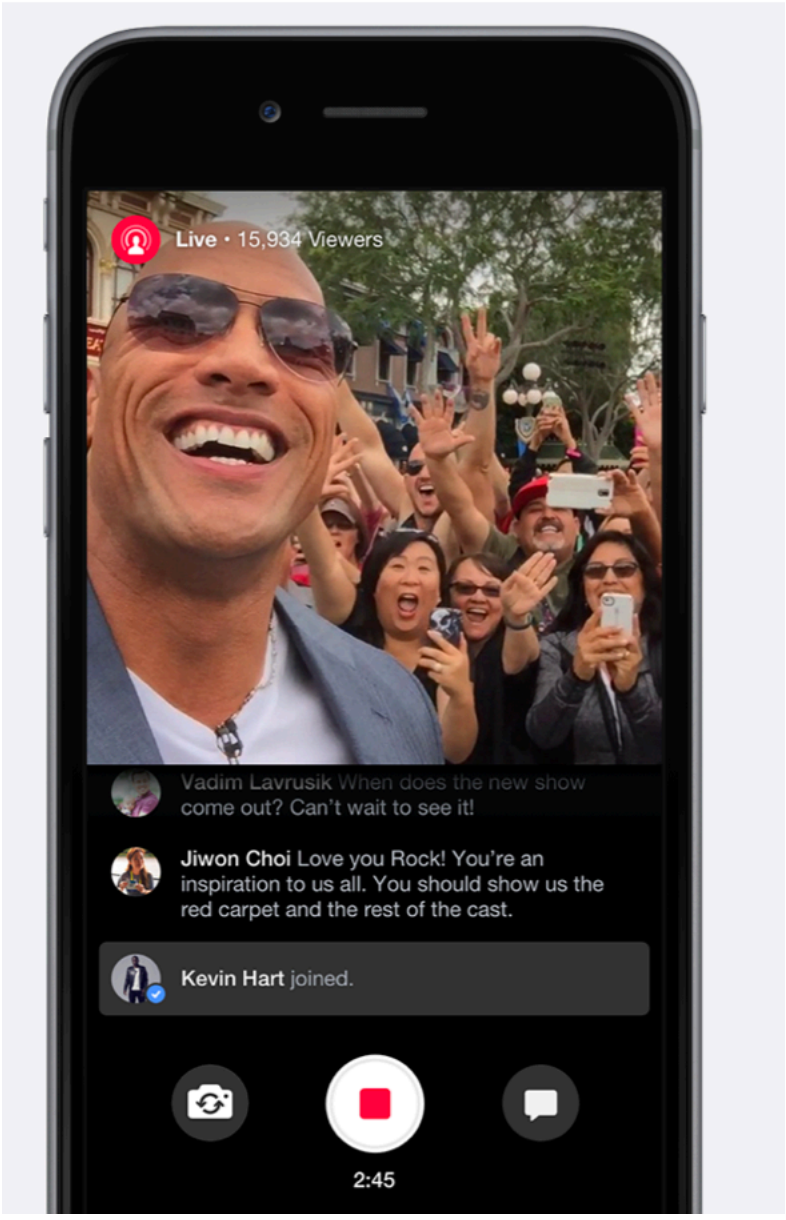
Ummm please

OK! How many sco

Got it!

You and Brendan can now see each other.

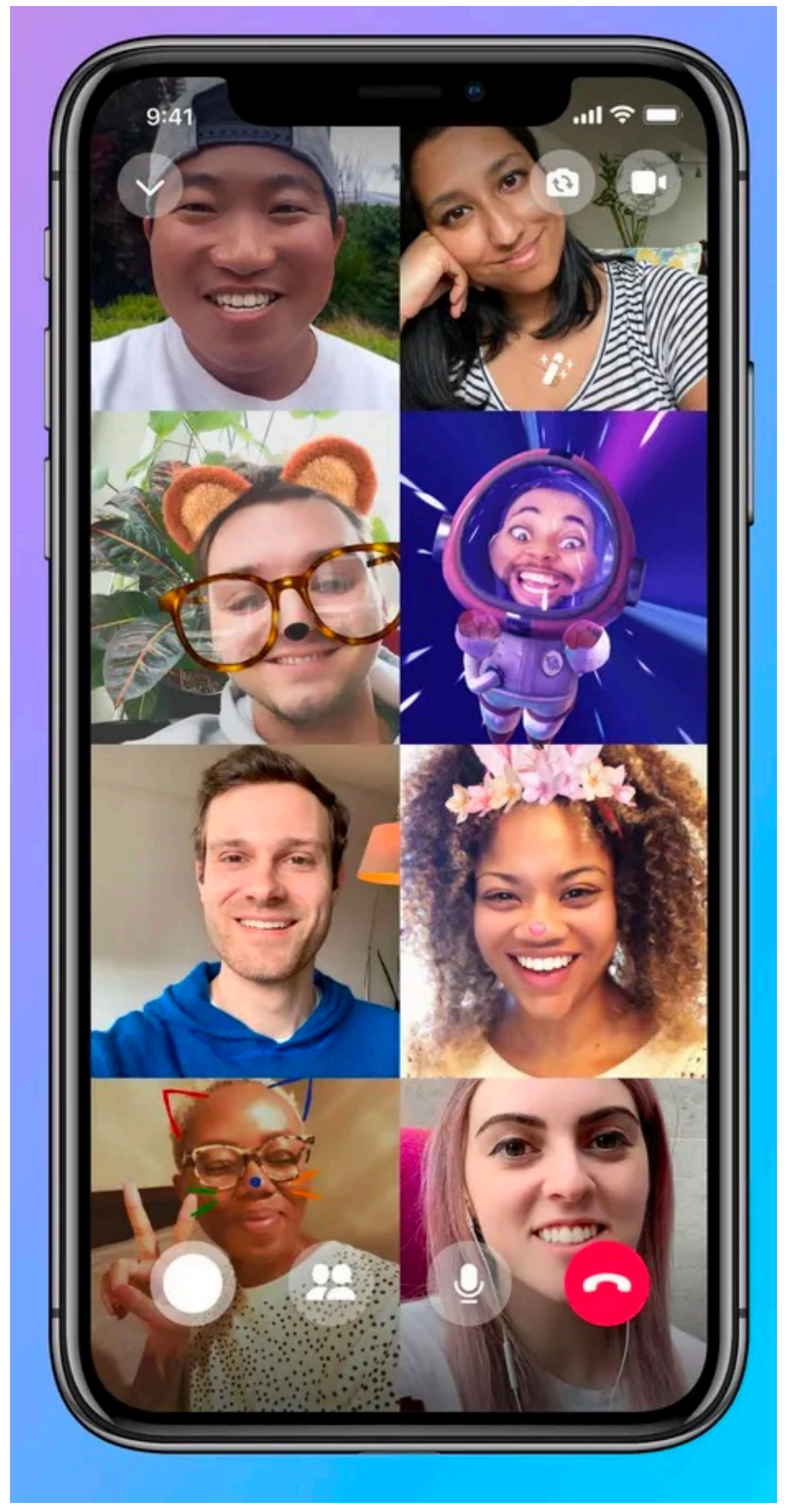
Facebook Live / Messenger Live Video



Vadim Lavrusik When does the new show come out? Can't wait to see it!

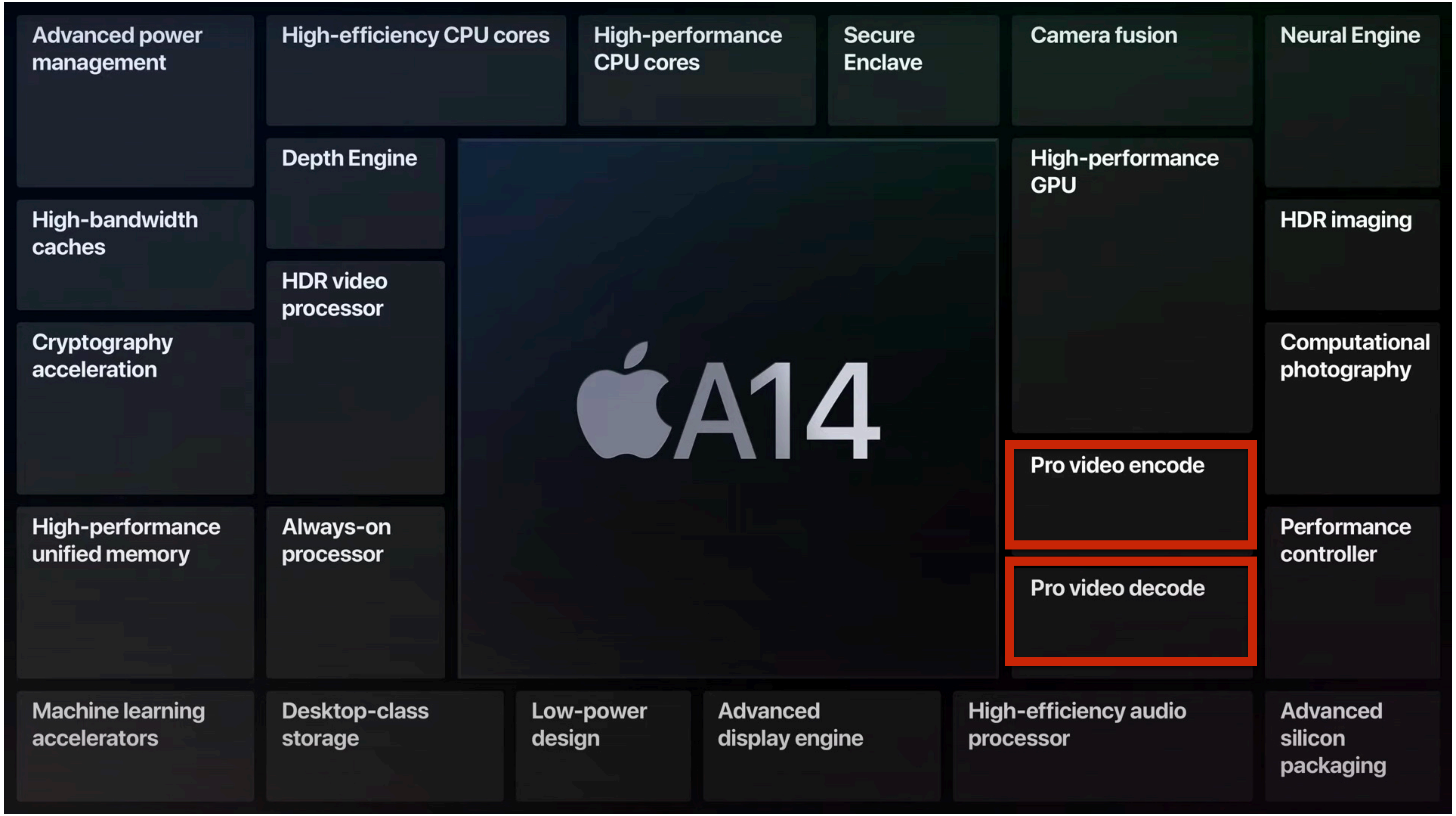
Jiwon Choi Love you Rock! You're an inspiration to us all. You should show us the red carpet and the rest of the cast.

Kevin Hart joined.



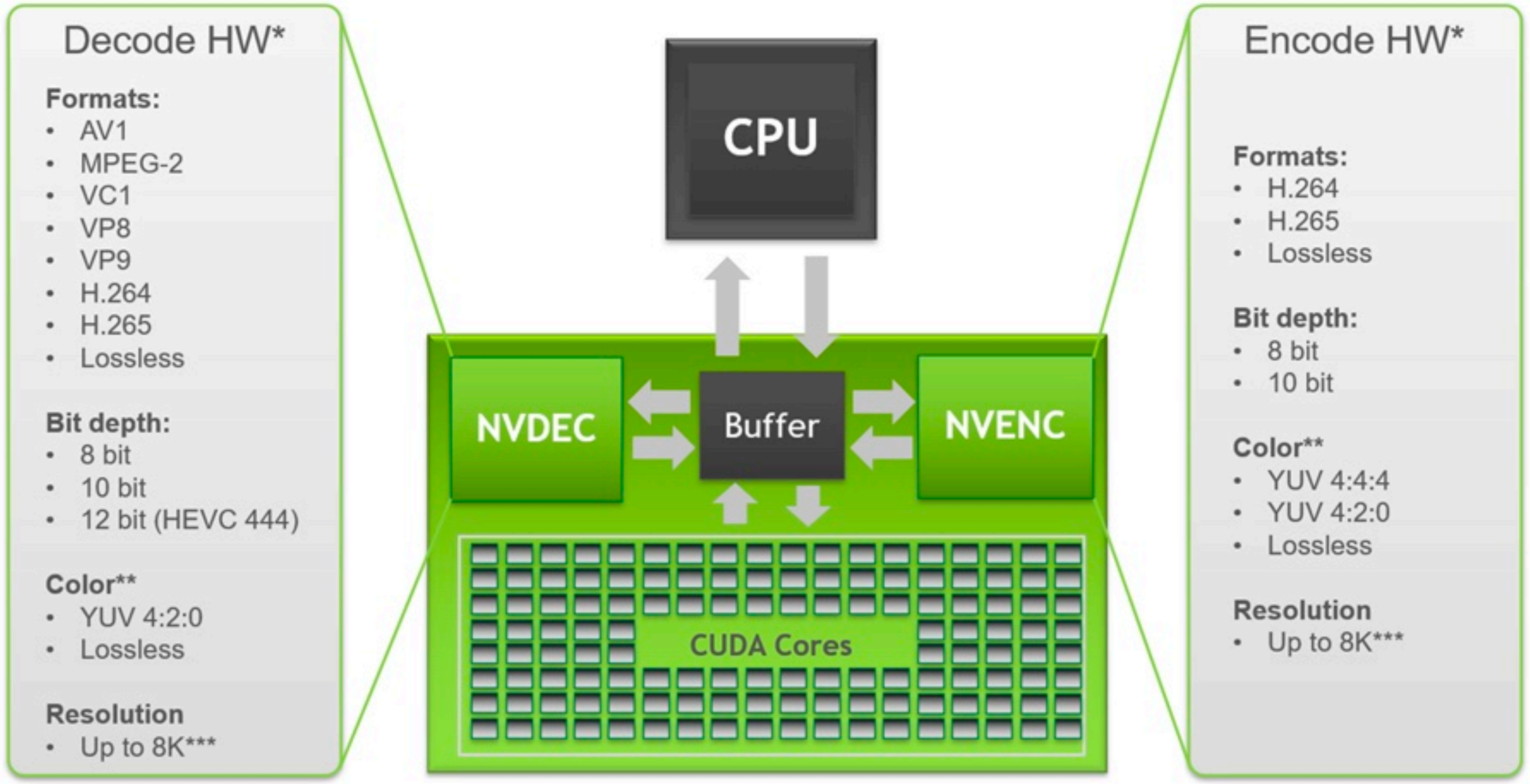
Consider different tradeoffs: compression quality vs. latency in each of these cases

ASIC acceleration of video encode/decode



NVIDIA GPUs have video encode/decode ASICs

- Example: GeForce NOW game streaming service
- Rendered images compressed by GPU and directly streamed over network to remote player



Decode HW*

Formats:
- AV1
- MPEG-2
- VC1
- VP8
- VP9
- H.264
- H.265
- Lossless

Bit depth:
- 8 bit
- 10 bit
- 12 bit (HEVC 444)

Color**
- YUV 4:2:0
- Lossless

Resolution
- Up to 8K***

Encode HW*

Formats:
- H.264
- H.265
- Lossless

Bit depth:
- 8 bit
- 10 bit

Color**
- YUV 4:4:4
- YUV 4:2:0
- Lossless

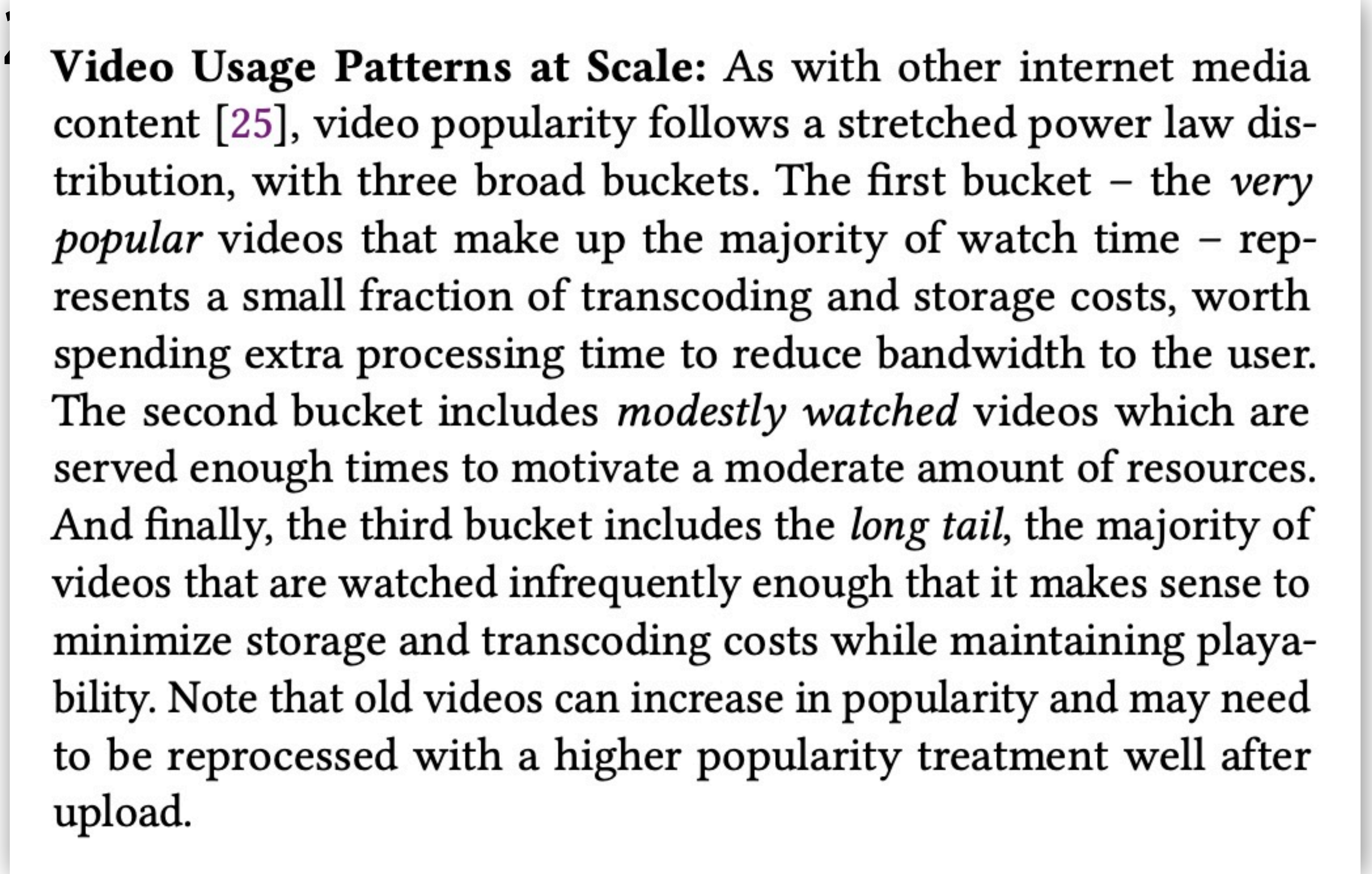
Resolution
- Up to 8K***

- Another example: consumers at home streaming to Twitch
  - Do not want compression to take processing capability away from running the game itself.

Why do you think Google's video sharing services (Youtube, Google photos, etc.) are willing to pay a high compute cost for compression?

- Reminder: statistic from Google:
  - About 8-10 CPU minutes to compress 150 frames of 2160p H.2
  - About 1 CPU hour for more expensive VP9 codec

[Ranganathan 2021]

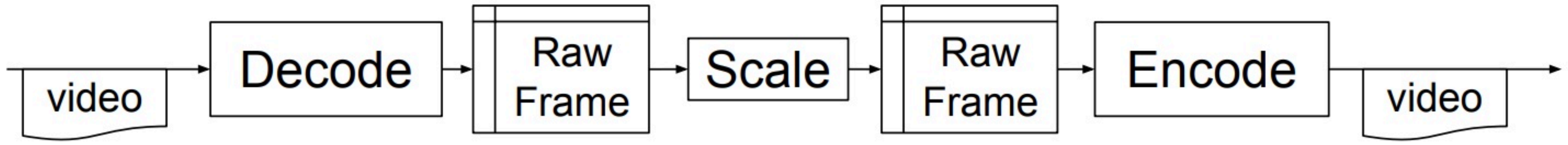


Video Usage Patterns at Scale: As with other internet media content [25], video popularity follows a stretched power law distribution, with three broad buckets. The first bucket – the *very popular* videos that make up the majority of watch time – represents a small fraction of transcoding and storage costs, worth spending extra processing time to reduce bandwidth to the user. The second bucket includes *modestly watched* videos which are served enough times to motivate a moderate amount of resources. And finally, the third bucket includes the *long tail*, the majority of videos that are watched infrequently enough that it makes sense to minimize storage and transcoding costs while maintaining playability. Note that old videos can increase in popularity and may need to be reprocessed with a higher popularity treatment well after upload.

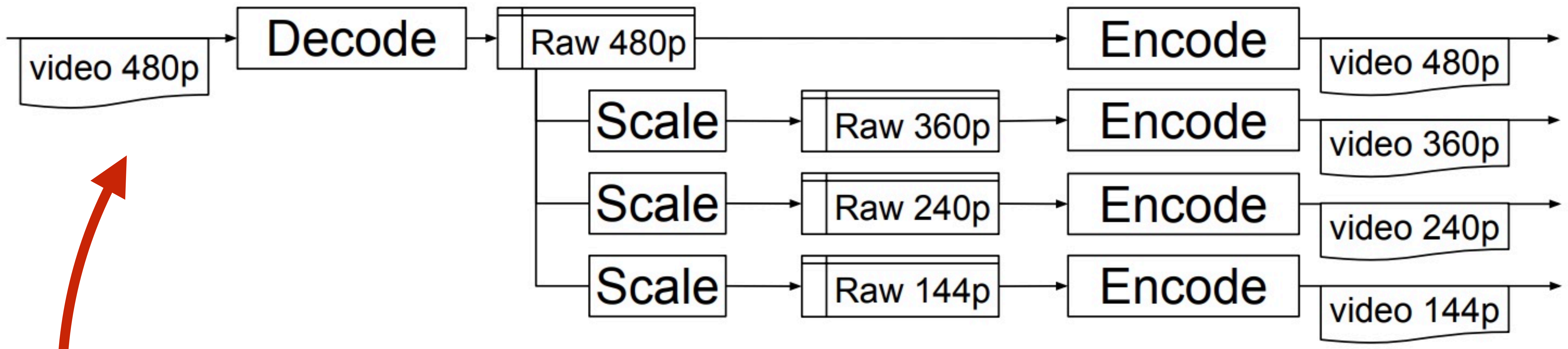
When you upload a video it gets processed into many different output videos for serving

- Different resolutions:
  - For different viewing device types
  - For different network conditions

- Different formats:
  - Different devices might have video decode hardware that supports different formats (older devices might only support H.264, newer devices H.265, AV1 etc)



(a) Single-output transcoding (SOT) pipeline



(b) 480p multiple-output transcoding (MOT) pipeline

Note: it makes sense to amortize data loading (from storage) and data decoding costs over many output resolutions/formats

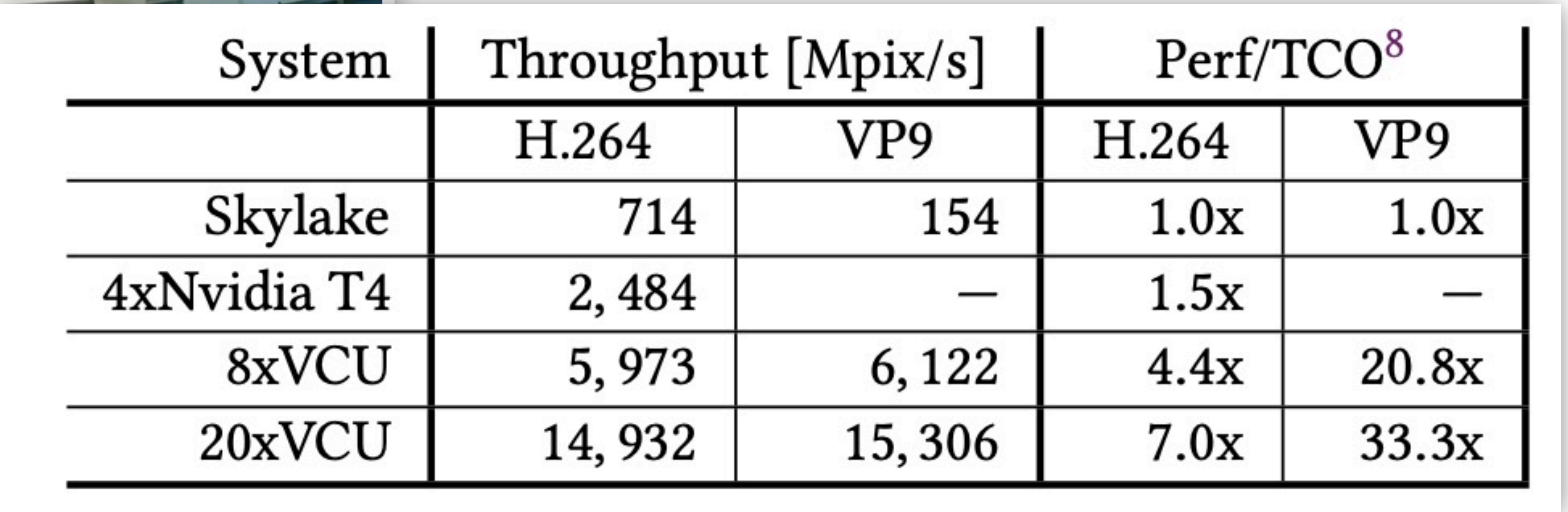
Google's Video (Trans)coding Unit (VCU)

- ASIC hardware for decoding/encoding video in Google datacenter for Youtube/Youtube Live/etc.
- Consider load:
  - 500 hours of video uploaded to Youtube per minute (2019)
  - Must generate encoded versions assets at many resolutions and using different codecs to support streaming to consumers with many different devices and networks

#LuisFonsi #Despacito #Imposible

Luis Fonsi - Despacito ft. Daddy Yankee

7,332,242,498 views • Jan 12, 2017



System	Throughput [Mpix/s]		Perf/TCO[8]	
	H.264	VP9	H.264	VP9
Skylake	714	154	1.0x	1.0x
4xNvidia T4	2, 484	—	1.5x	—
8xVCU	5, 973	6, 122	4.4x	20.8x
20xVCU	14, 932	15, 306	7.0x	33.3x

[Ranganathan 2021]

Learned Compression Schemes

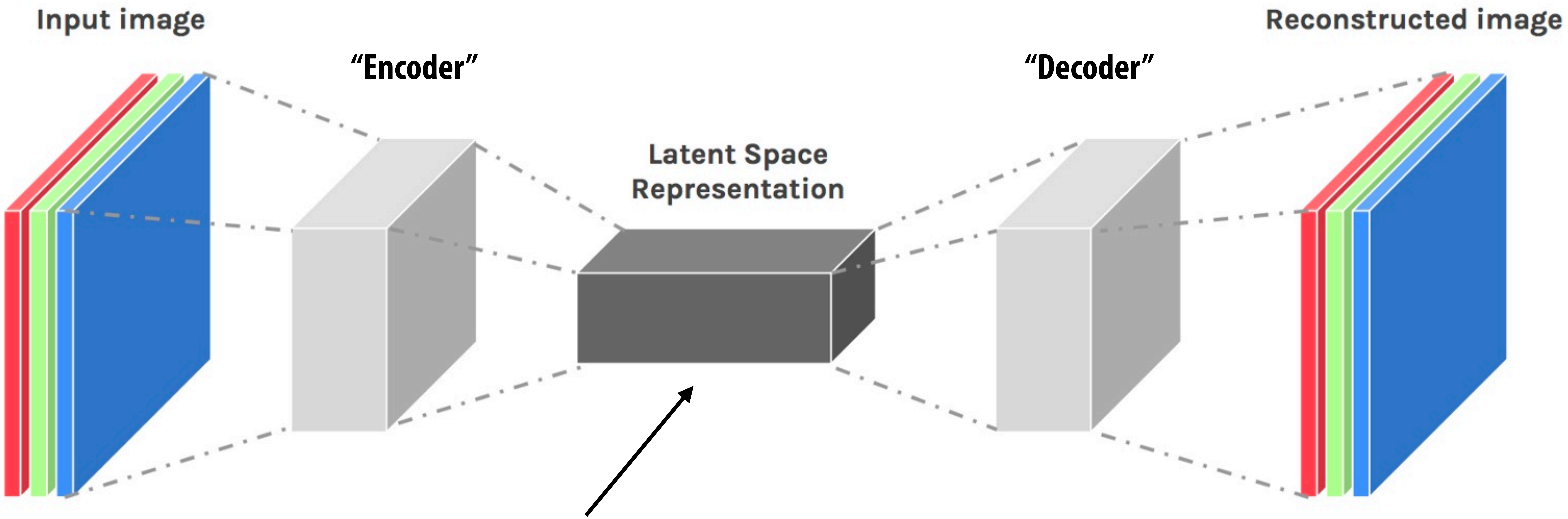
Learned compression schemes

- H.264/265/AV1 video compression are “lossy” compression techniques that discard information is that is present in the visual signal, but less likely to be noticed by the human eye
  - Key principle: “Lossy, but still looks good enough to humans!”

- Compression schemes described in this lecture so far involve manual choice / engineering of good representations (features)

- But machine learning is all about learning good representations from data.
  - Interest in learning highly compressed representations for a specific class of images/videos, or for a *specific task* to perform on images/videos

DNN autoencoder

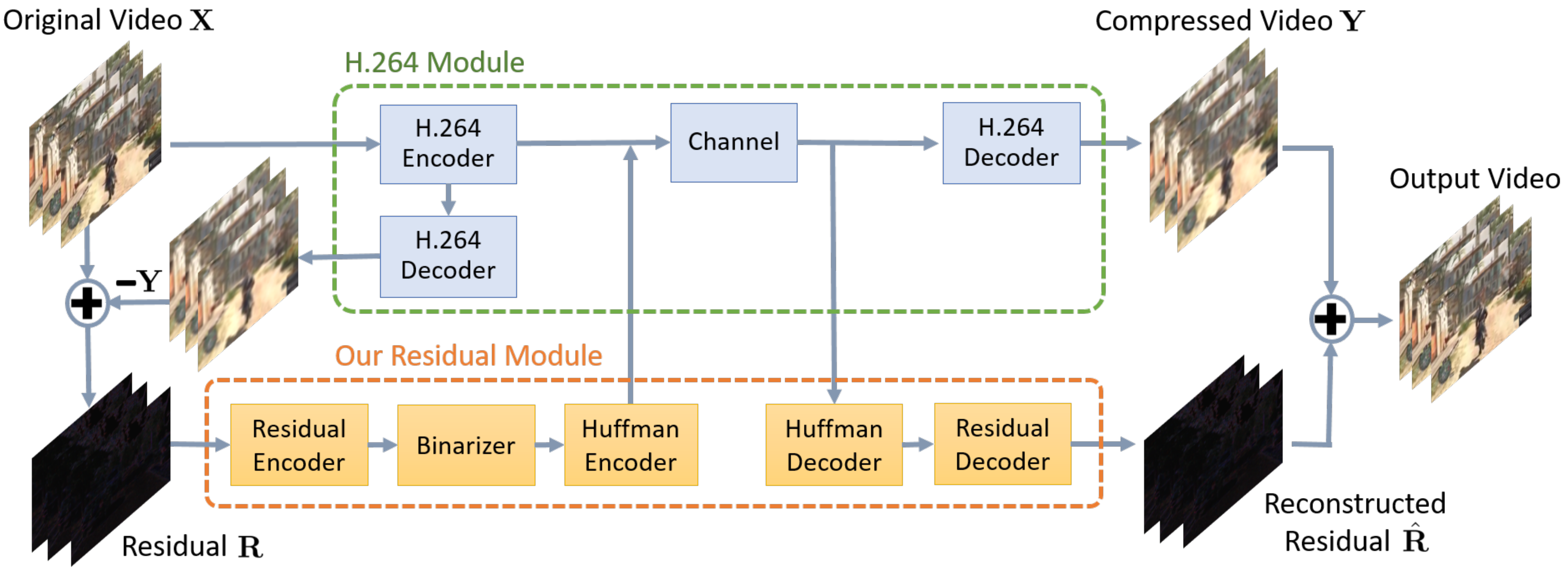


If this latent representation is compact, then it is a compressed representation of the input image

https://medium.com/@birla.deepak26/autoencoders-series-daad78df9350

Learned compression schemes

Many recent DNN-based approaches to compressing video learn to *compress the residual*

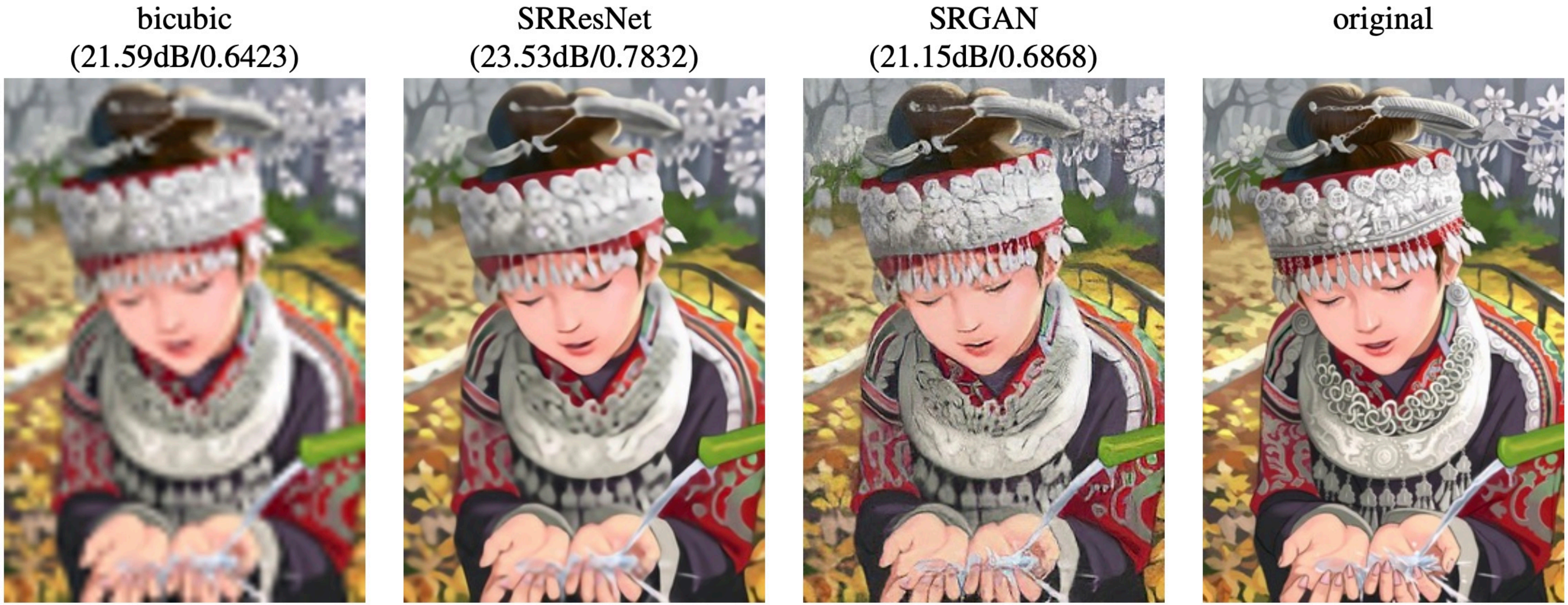


[Tsai et al. 2018]

Use standard video compression at low quality, then use an autoencoder to compress the residual. (Learn to compress the residual)

Super-resolution-based reconstruction

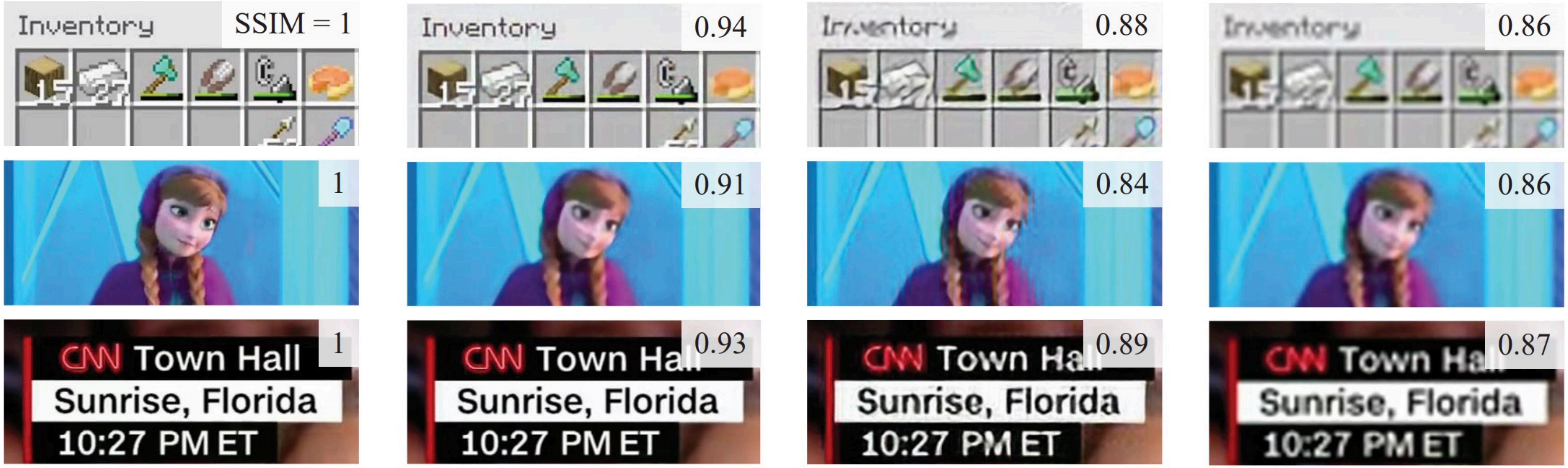
Single image superresolution task: given a low-resolution image, predict the corresponding high-resolution image



[SRGAN, Ledig et al. CVPR 2017]

Super resolution-based reconstruction

- Encode low-resolution video using standard video compression techniques
- Also transfer (as part of the video stream) a video-specific super-resolution DNN to upsample the low resolution video to high res video.
  - Assumption: training costs are amortized over many video downloads

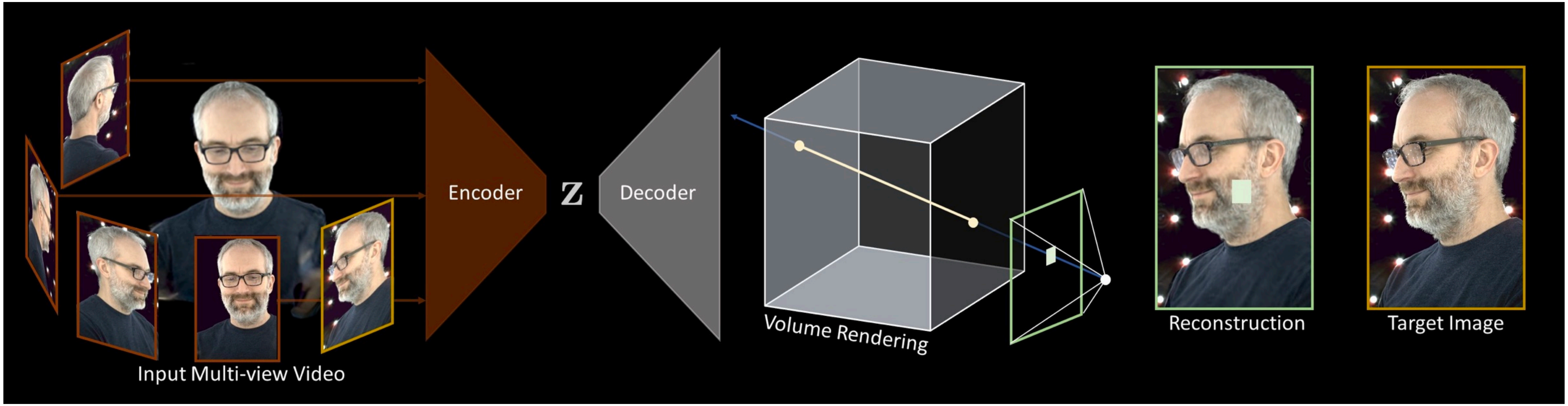


(a) Original (1080p) (b) Content-aware DNN (c) Content-agnostic DNN (d) 240p

[Yeo et al. OSDI 2018]

Neural volumes

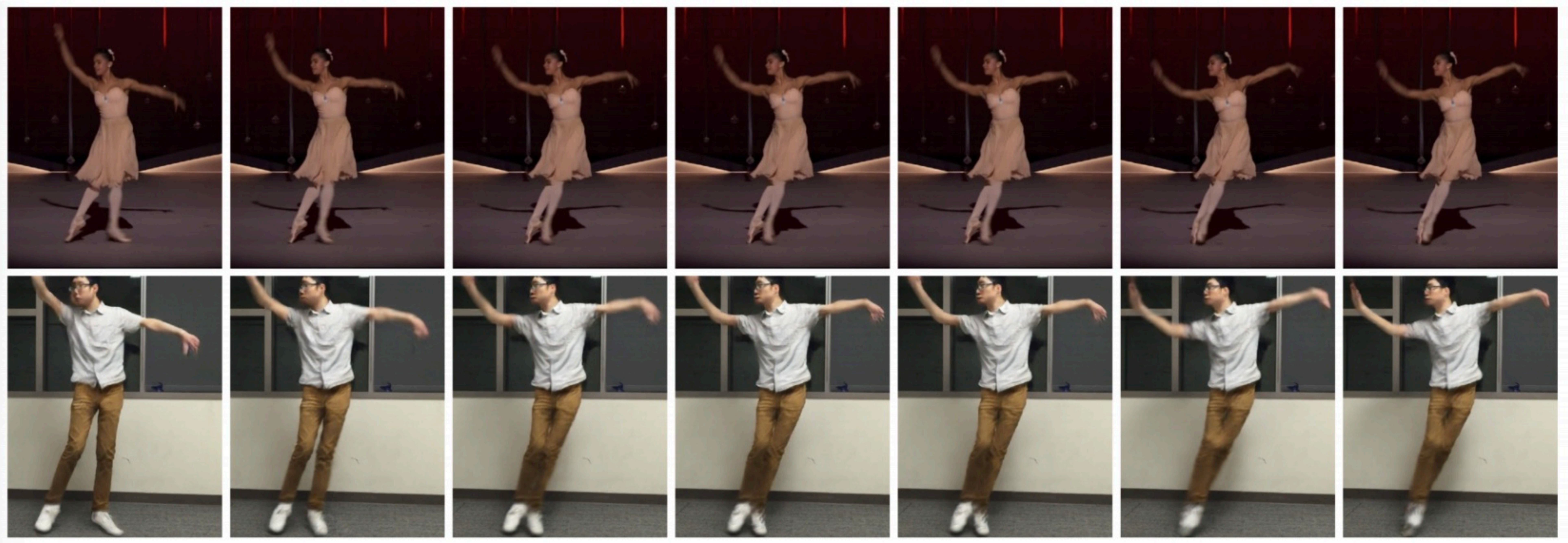
- Learn to encode multiple views of a person into a latent code (z) that is decoded into a volume than can be rendered with conventional graphics techniques *from any viewpoint*



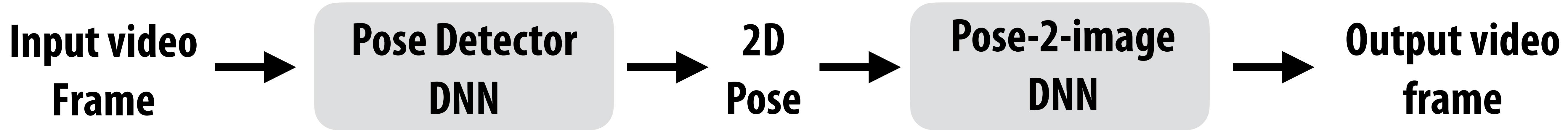
- Motivated by VR applications

Person-specific video compression

Input: video of professional ballerina performing a motion

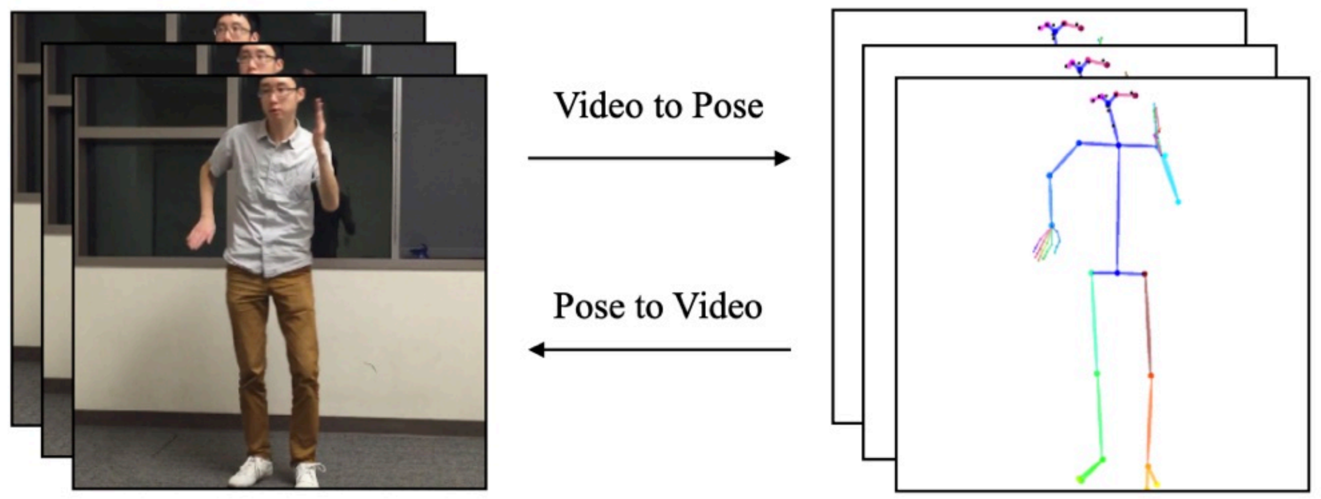


Output: video of graduate student performing the same motion



Input video Frame → Pose Detector DNN → 2D Pose → Pose-2-image DNN → Output video frame

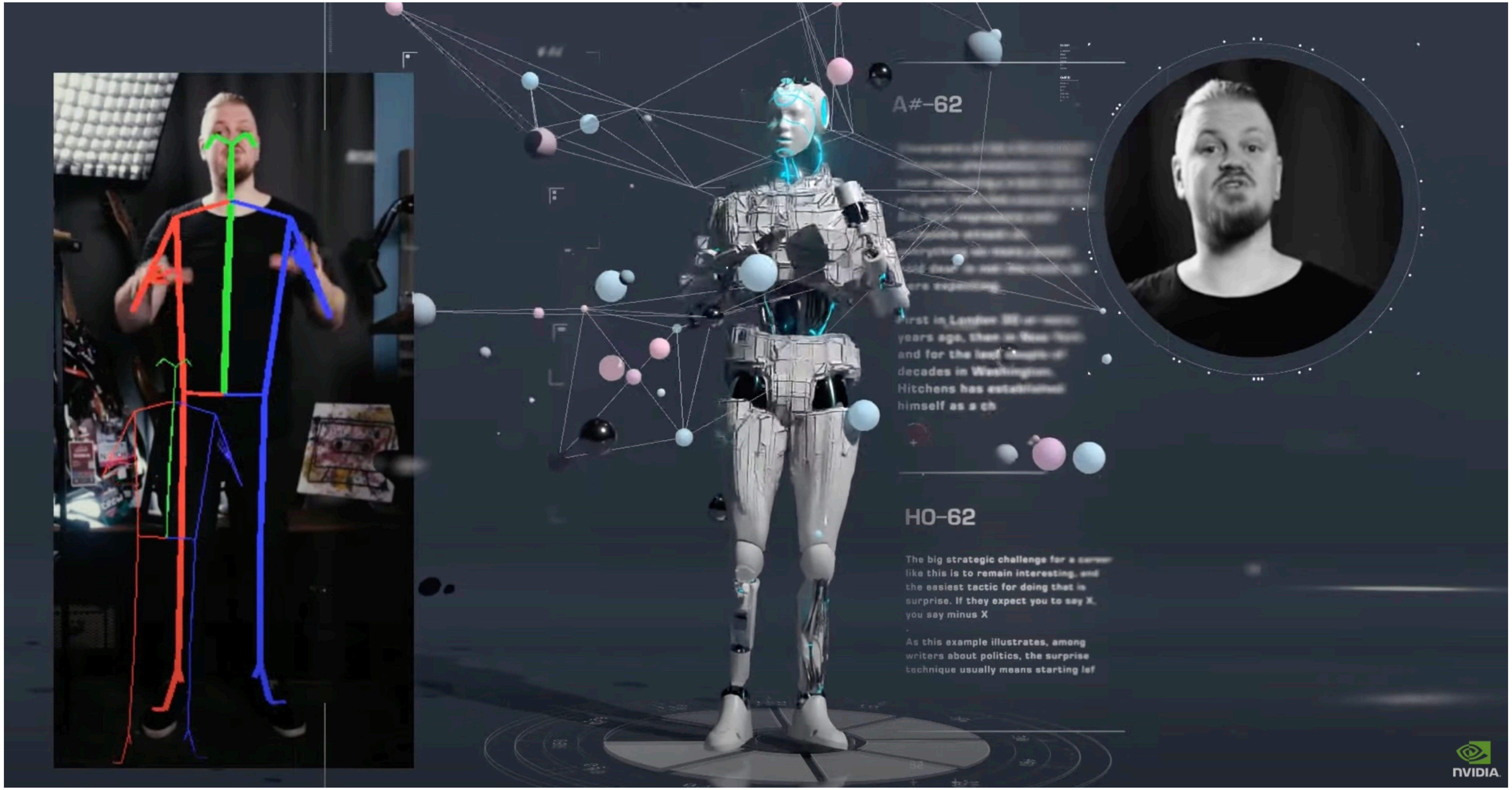
Encode video as just a set of 14 pose joints.



[Chan et al. 2019]

NVIDIA Maxine

GPU-accelerated video processing for video conferencing applications



[illegible]

First in London [illegible] years ago, then in New York and for the last couple of decades in Washington, Hitchens has established himself as a ch

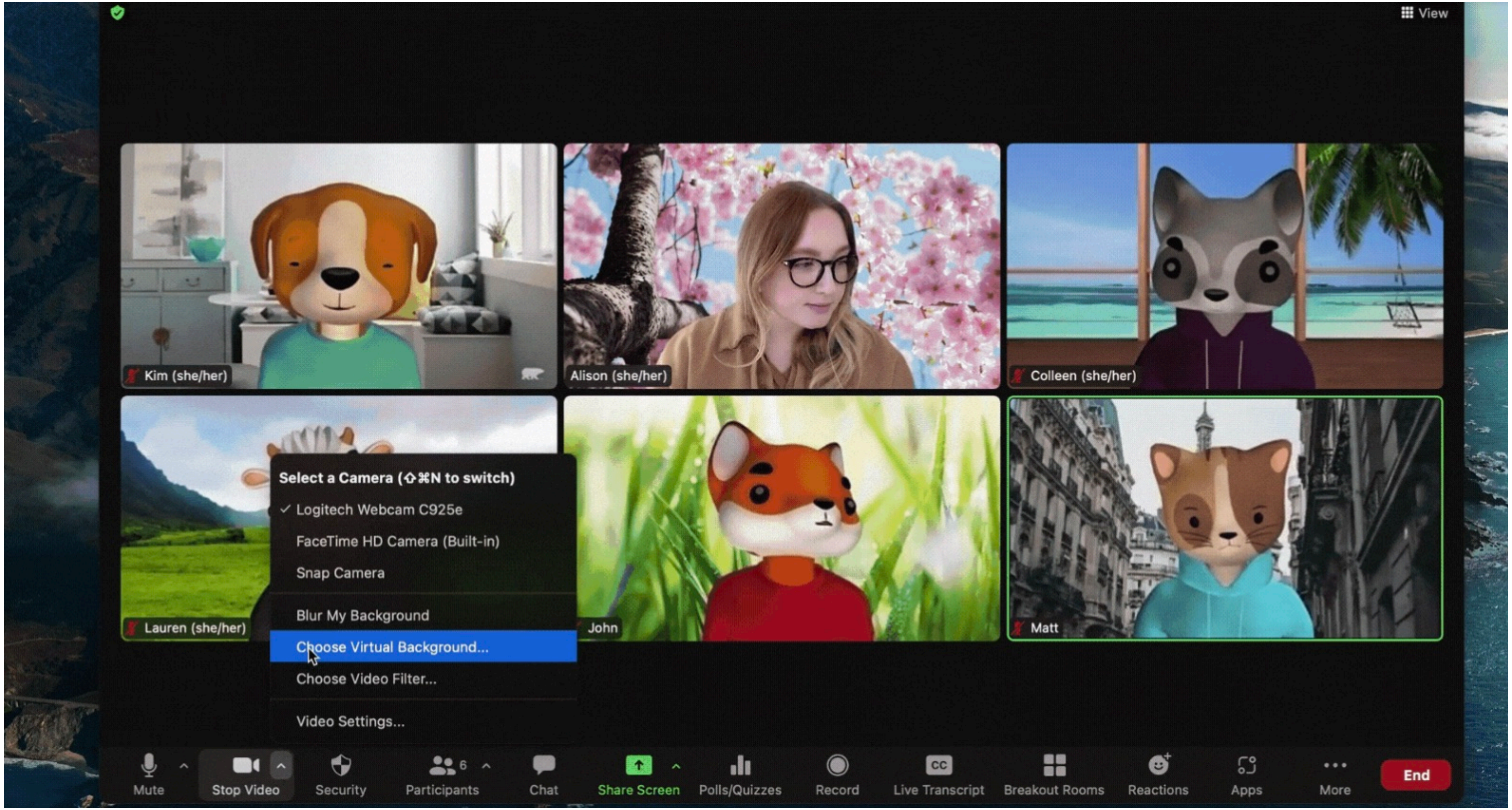
The big strategic challenge for a career like this is to remain interesting, and the easiest tactic for doing that is surprise. If they expect you to say X, you say minus X

.

As this example illustrates, among writers about politics, the surprise technique usually means starting lef

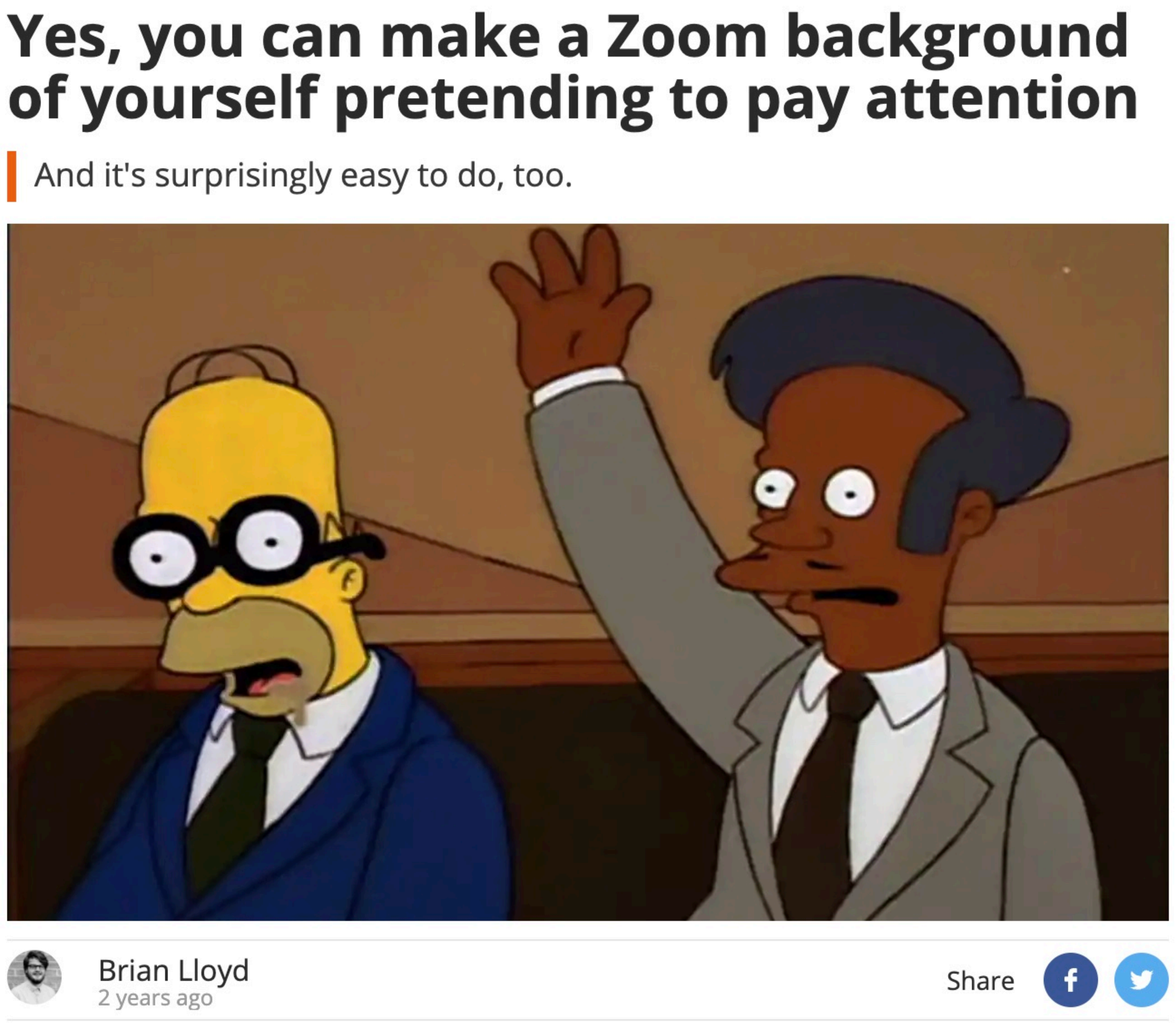
Examples: avatar control, video superresolution, advanced background segmentation

Zoom avatars / Snapcam lenses



Where is the line between transmission of what happened and "making something up"?

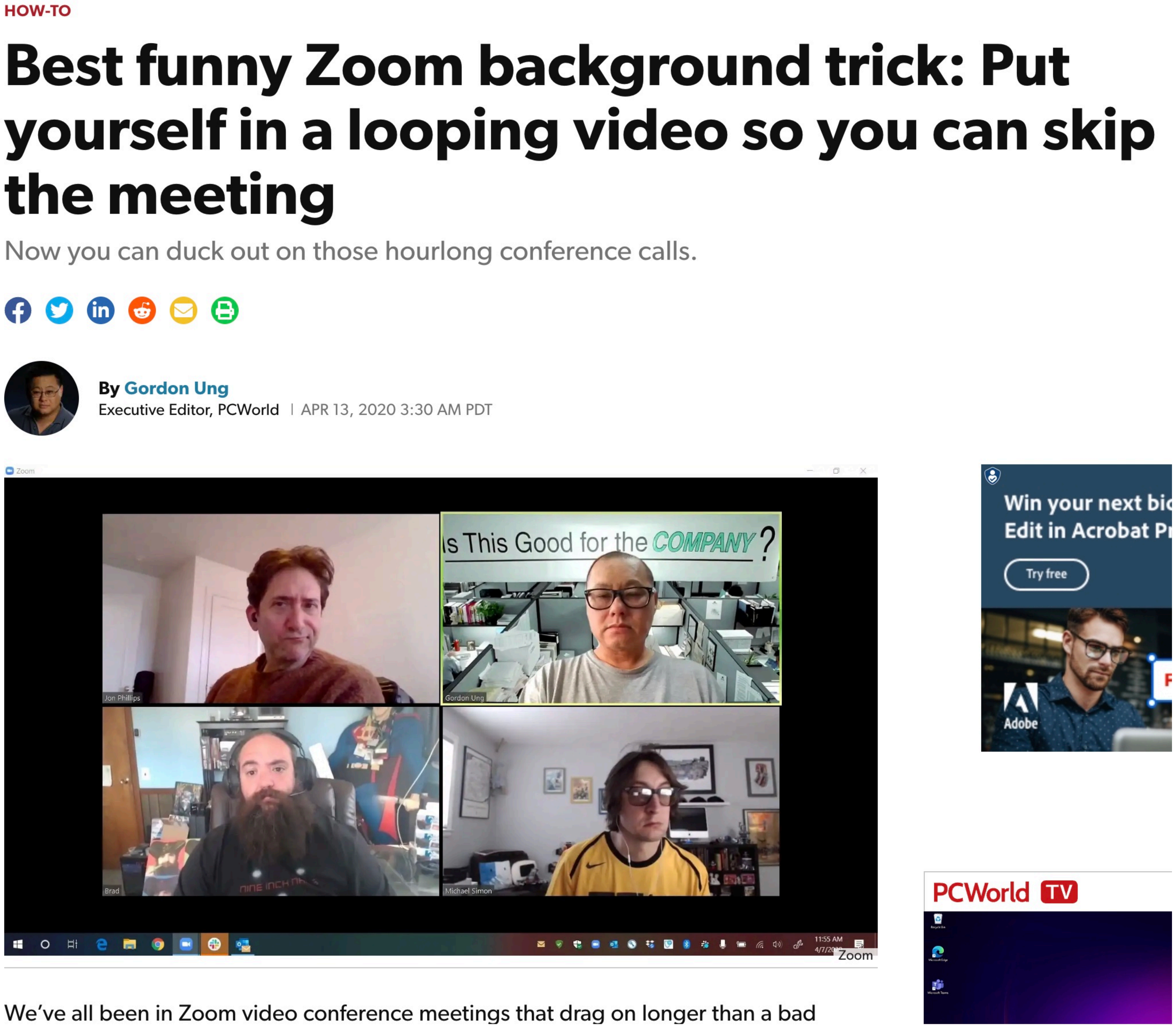
The best camera is the one that's off?



Yes, you can make a Zoom background of yourself pretending to pay attention

And it's surprisingly easy to do, too.

Brian Lloyd
2 years ago



HOW-TO

Best funny Zoom background trick: Put yourself in a looping video so you can skip the meeting

Now you can duck out on those hourlong conference calls.

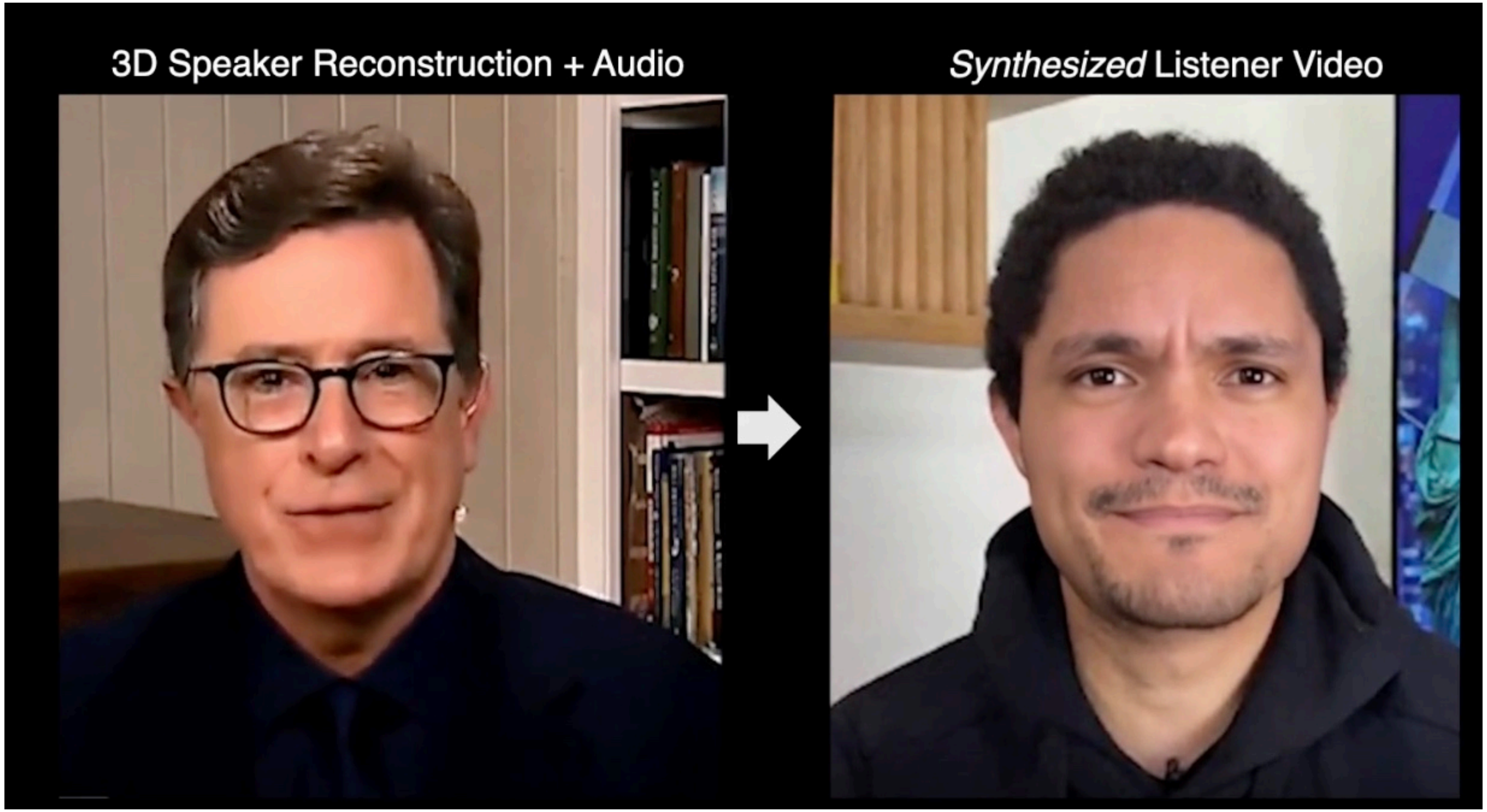
By Gordon Ung
Executive Editor, PCWorld | APR 13, 2020 3:30 AM PDT

We've all been in Zoom video conference meetings that drag on longer than a bad

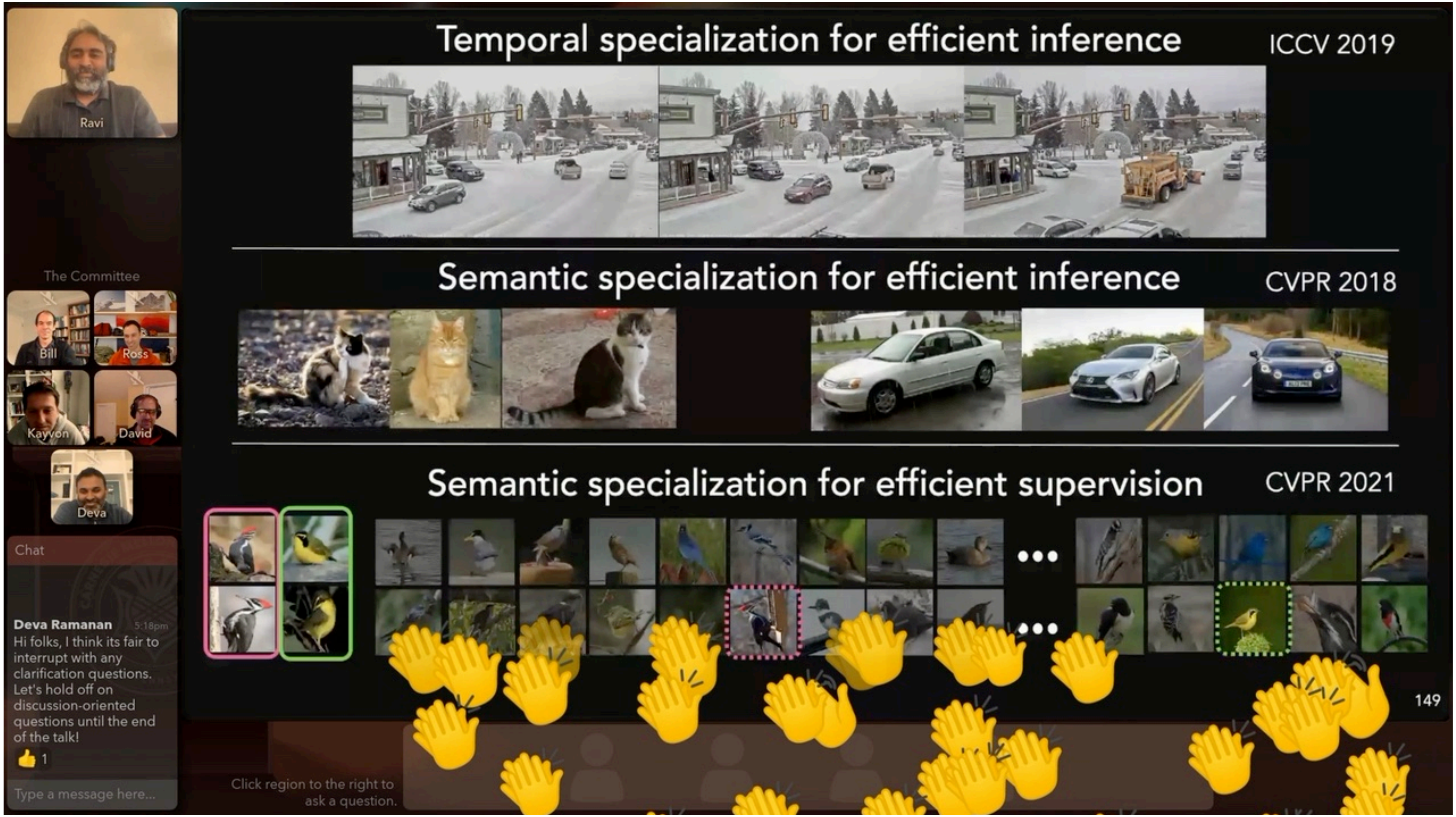
Synthesizing reactions?

Input: audio of speaker

Output: video of listener's reaction


3D Speaker Reconstruction + Audio
Synthesized Listener Video

User-triggered effects (examples: audio clips, "reactions")



Temporal specialization for efficient inference

Semantic specialization for efficient inference

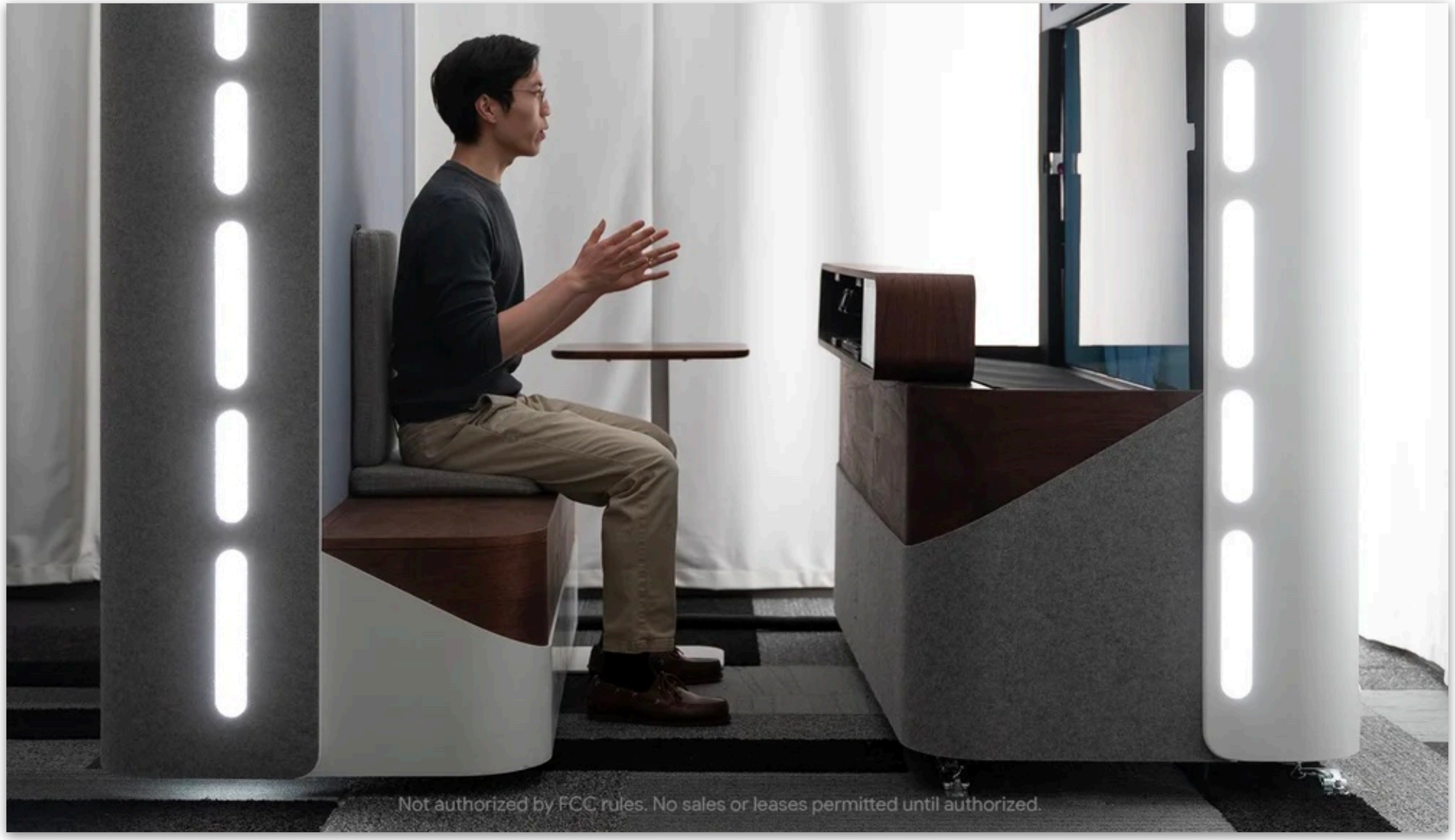
Semantic specialization for efficient supervision

Deva Ramanan 5:18pm
Hi folks, I think its fair to interrupt with any clarification questions. Let's hold off on discussion-oriented questions until the end of the talk!
👍 1

Type a message here...

Click region to the right to ask a question.

Project Starline: pursuit of high fidelity


Not authorized by FCC rules. No sales or leases permitted until authorized.

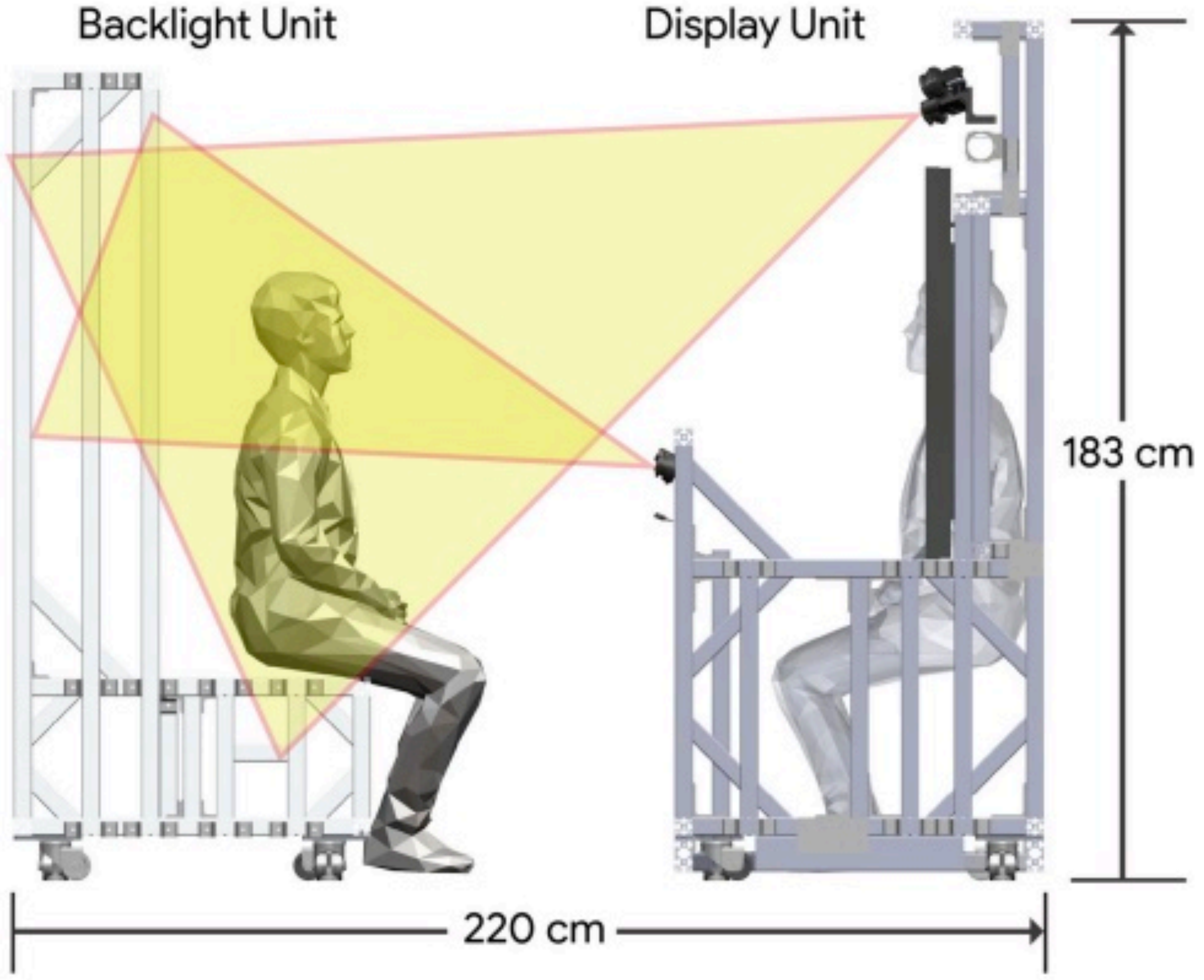
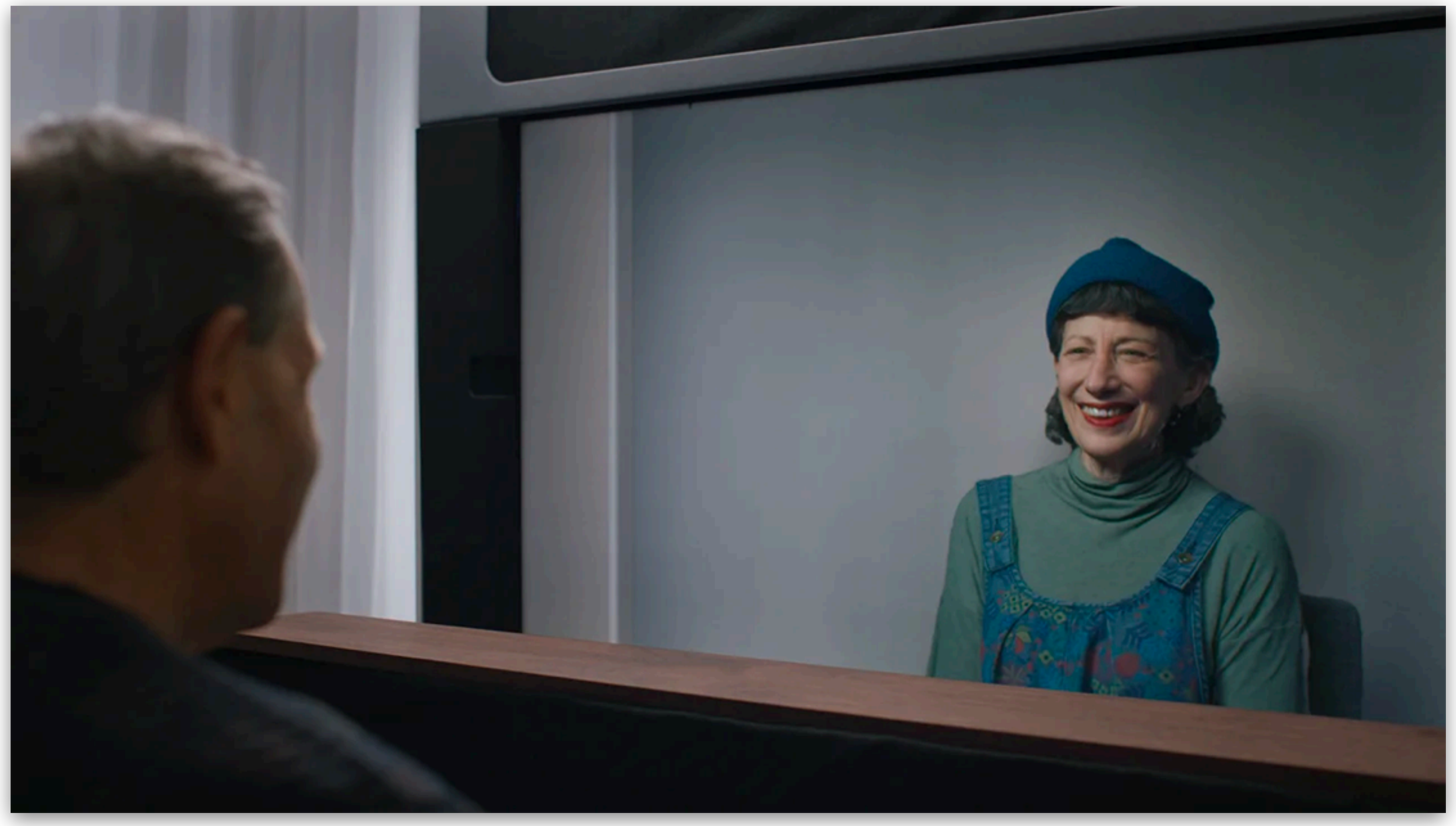
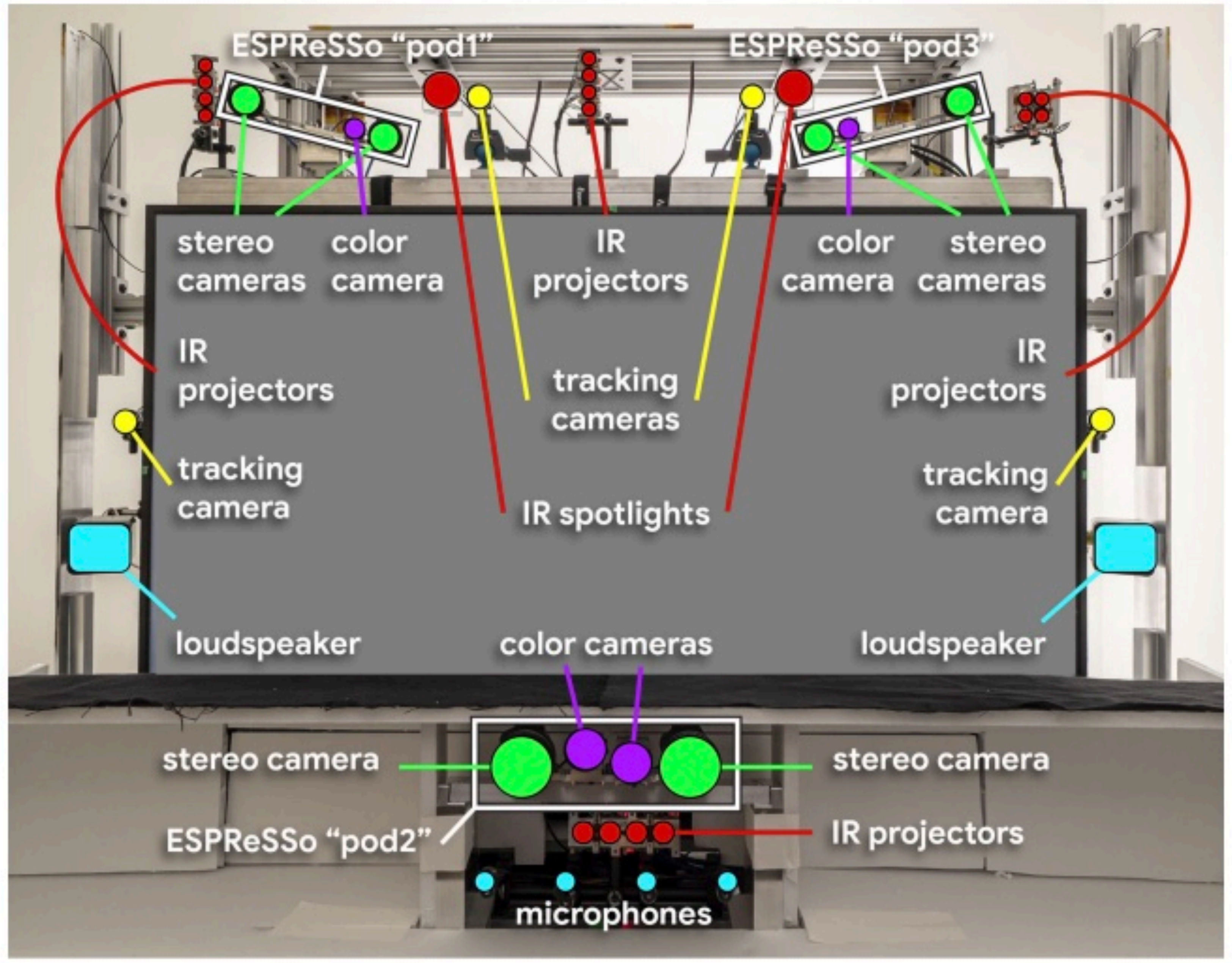


Fig. 4. Side-elevation view of our prototype system, illustrating the relative placement of the user, cameras, display, and virtual remote participant.





[Lawrence et al. 2021]

Summary

- **Last two lectures: representing images accurately and efficiently**
  - Choice of color space (different representations of color)
  - Store values in perceptual space (non-linear in energy)
  - JPEG image compression (tolerate loss due to approximate representation of high frequency components)
- **H.264/265/AV1 video compression are "lossy" compression techniques that discard information is that is present in the visual signal, but less likely to be noticed by the human eye**
- **Key challenge of video encoding is "searching" for a compact encoding of the visual signal in a large space of possibilities**
- **Growing interest in learning these encodings, but it remains hard to beat well-engineered features**
  - But promising if learned features are specialized to video stream contents
  - Or to specific tasks (remember, increasing amount of video is not meant to be consumed by human eyes)