Lecture 19: Image and Video Compression

Yuhao Zhu

http://yuhaozhu.com
yzhu@rochester.edu

CSC 259/459, Fall 2024
Computer Imaging & Graphics

The Roadmap

Theoretical Preliminaries

Human Visual Systems

Digital Camera Imaging

Modeling and Rendering

Photographic Optics

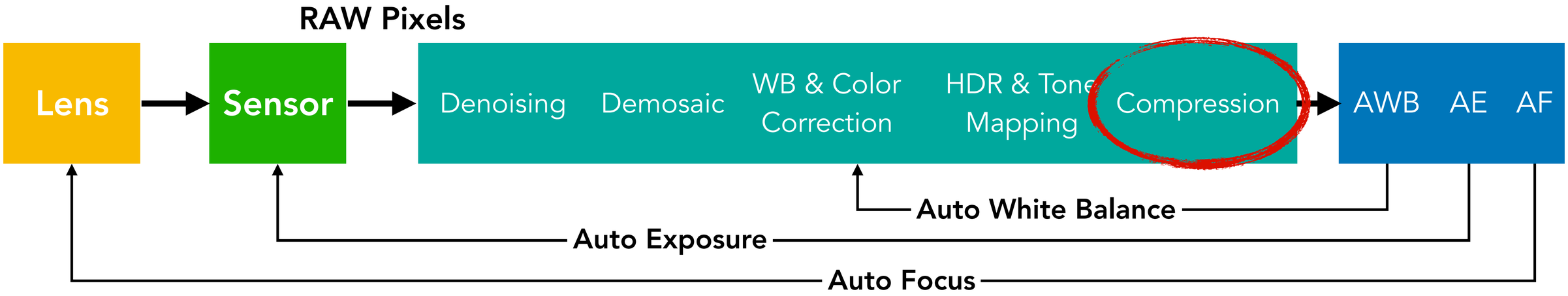
Image Sensor

Image Signal Processing

Image/Video Compression

Immersive Content

Compression

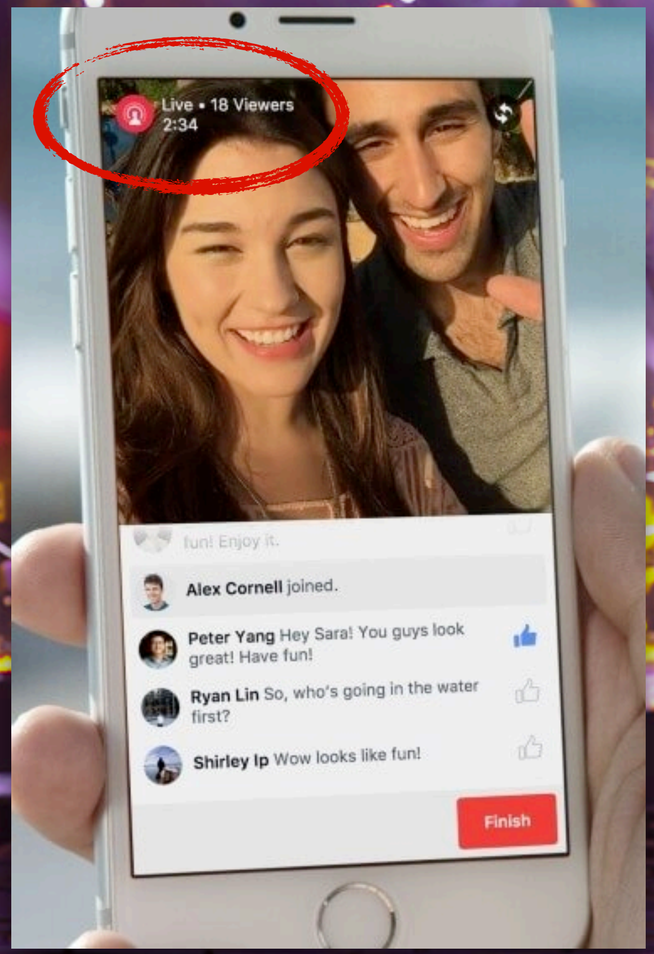


Most of cameras and smartphone allow you to shoot in RAW, i.e., exporting RAW images without any post-processing (in camera-native RGB space).

Otherwise, images/videos you get are compressed in the color space of the output device, mostly sRGB.

30-second video @ 1080p resolution (1920 x 1080 pixels per frame) @ 30 frames per second (FPS)
3 colors per pixel + 1 byte per color → 6.2 MB/frame → 6.2 MB x 30 s x 30 FPS = 5.2 GB total size
Actual H.264 video file size: 65.4 MB (80-to-1 compression ratio).
Compression/encoding done in real-time without you even realizing it!

Numbers credit: Kayvon Fatahalian



https://www.newscaststudio.com/2017/12/27/live-video-streaming-low-latency-codec/ https://inplayer.com/10-best-mobile-live-streaming-apps-2020/

Basic Concepts and Ideas of (Any) Compression

Goal: reduce data size while maintaining high (visual) quality

Lossless compression vs. lossy compression

Two main techniques:

- Lossless: removing redundancies.
- Lossy: sacrificing “unimportant” details. In the context of image/video compression, “importance” is usually dictated by human perception.

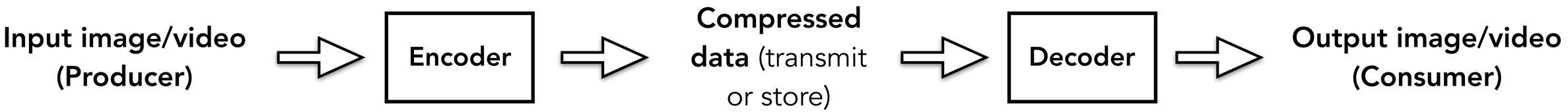
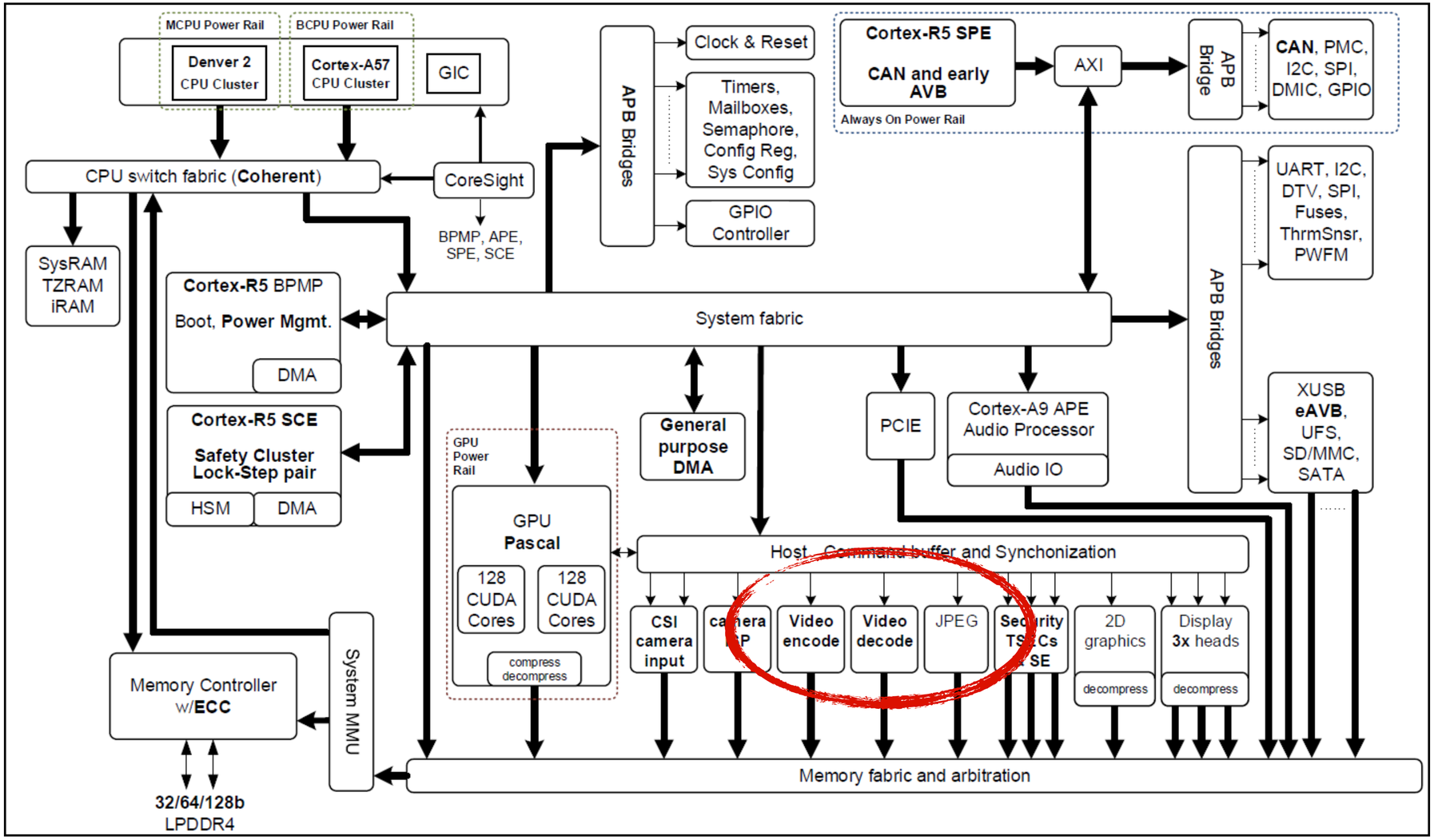


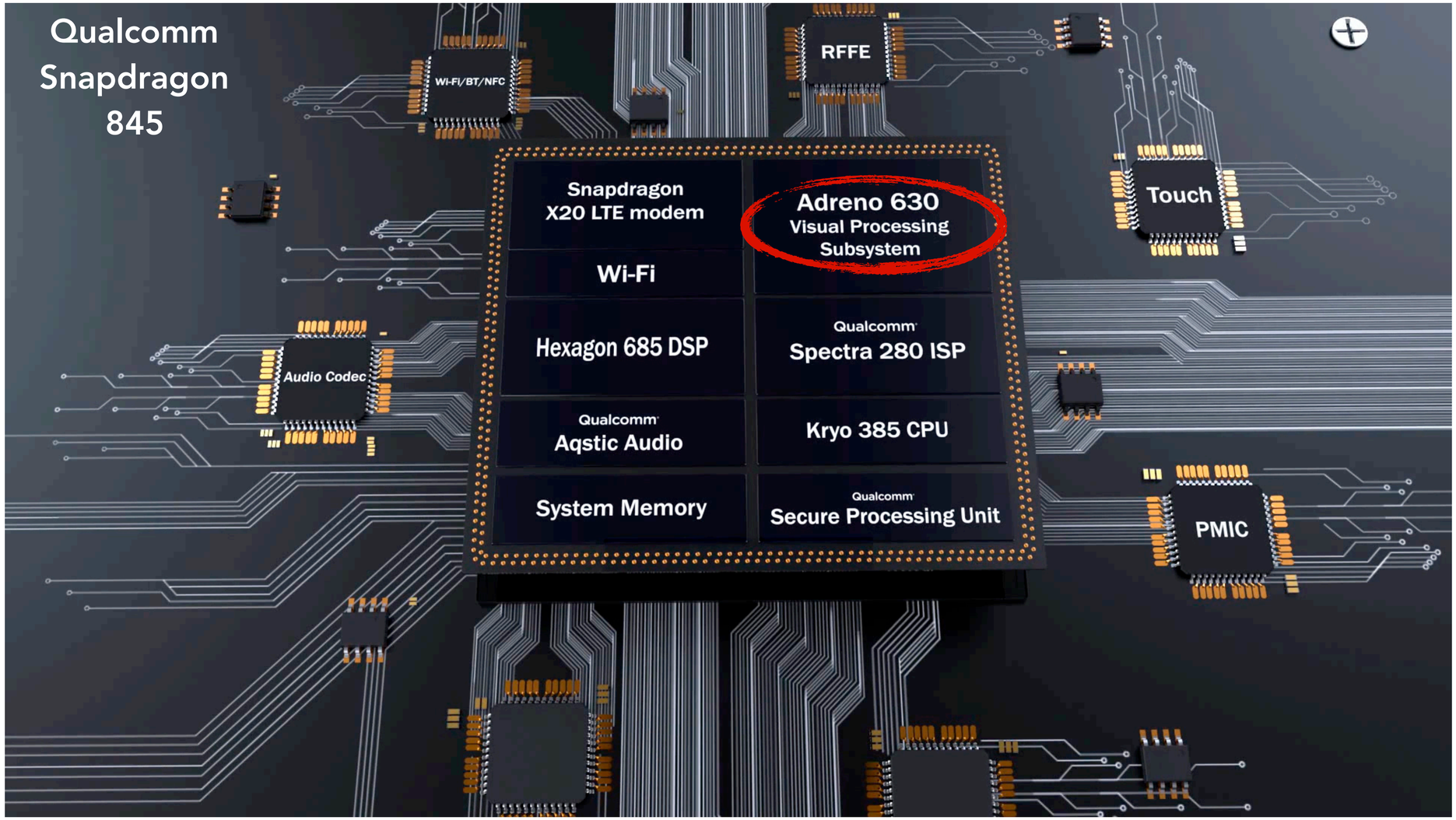
Image and Video Compression in Hardware

Nvidia
Xavier SoC



https://forums.developer.nvidia.com/t/supported-recommended-rtos-for-jetson-xavier/76623

Image and Video Compression in Hardware



https://techcrunch.com/2017/12/06/qualcomms-snapdragon-845-focuses-on-ai-vr-and-battery-life/

JPEG Image Compression

Two Big Ideas in JPEG Image Compression

JPEG is lossy, so must choose wisely what information to sacrifice.

Idea 1: retain luminance; subsample "colors".

- Luminance (brightness) encodes visual details.
- We are more sensitive to brightness differences than to chromatic differences; we can afford to lose information in colors, but not so much in luminance.

Idea 2: retain low frequency information; sacrifice high frequency information.

- Human visual system has a frequency threshold (photoreceptors are doing spatial sampling), and high-frequency information looks "busy"/"noisy" anyway, so a bit of inaccuracy/corruption in pixel values isn't very noticeable.

How Do We Separate Luminance and "Colors"

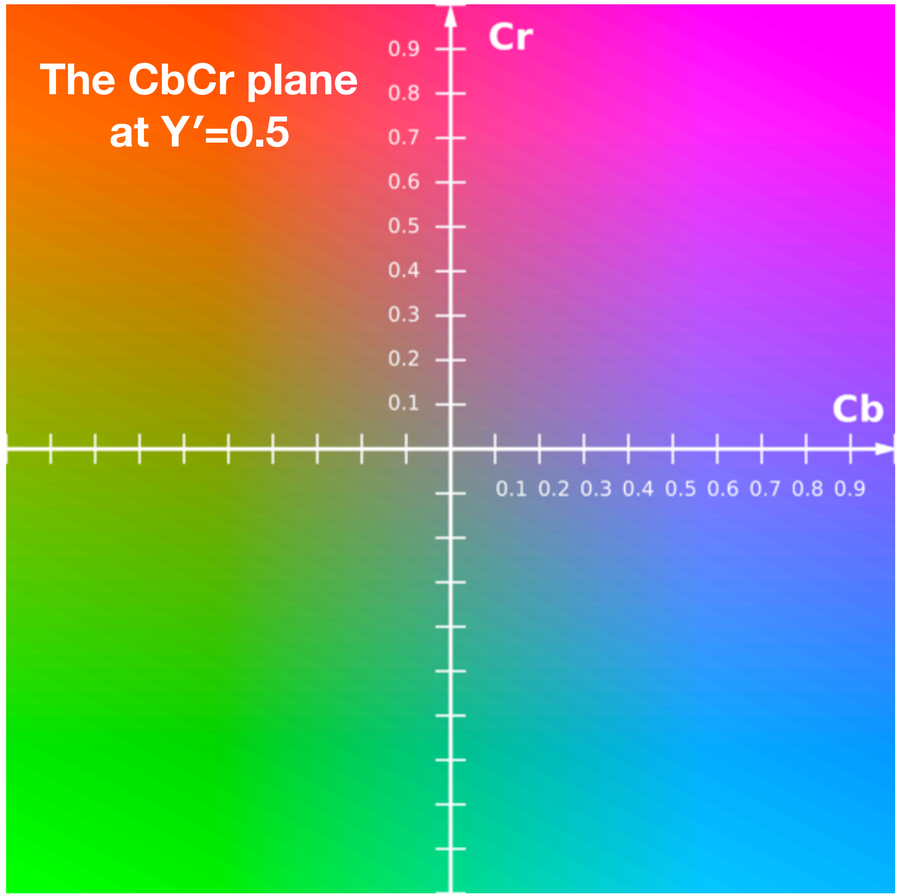
Recall Hering's Opponent Processes:

- Red-Green, Blue-Yellow, Light-Dark

The perceptual opponent space is ***not*** a linear transformation from, say, cone space

- There are neural opponent spaces that are linear w.r.t. cone space (e.g., the DKL space)

JPEG uses an empirical perceptual opponent space that is a linear transformation from sRGB



https://en.wikipedia.org/wiki/YCbCr

JPEG's Y'CbCr Color Space

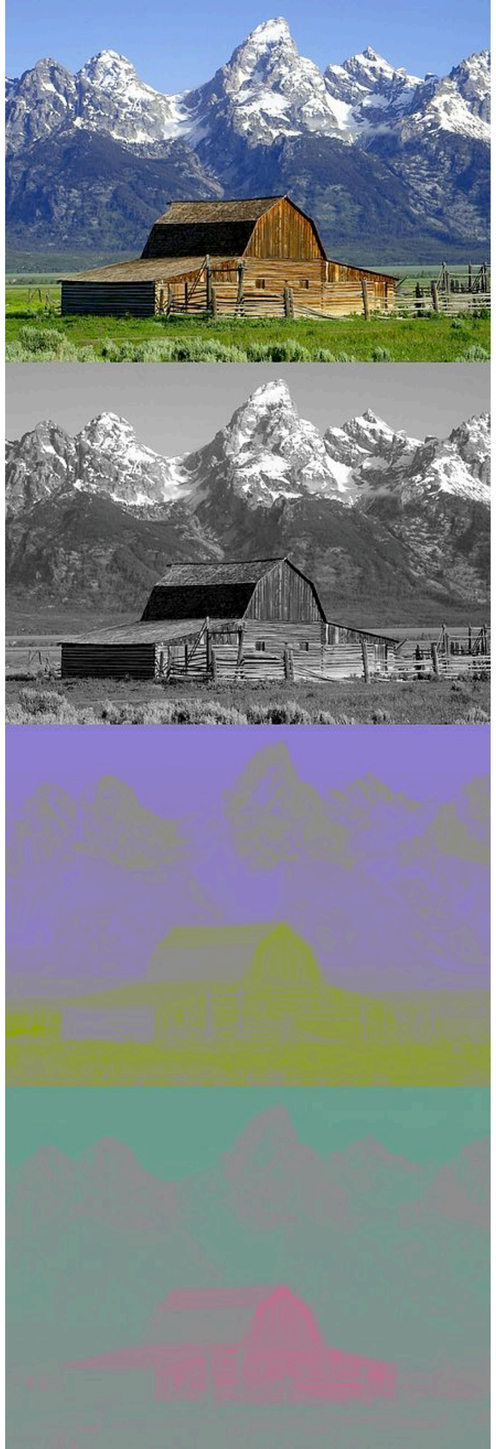
"Luma" → Y'
"Chroma" → C_B, C_R

$$\begin{aligned} Y' &= 0 + (0.299 \cdot R'_D) + (0.587 \cdot G'_D) + (0.114 \cdot B'_D) \\ C_B &= 128 - (0.168736 \cdot R'_D) - (0.331264 \cdot G'_D) + (0.5 \cdot B'_D) \\ C_R &= 128 + (0.5 \cdot R'_D) - (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D) \end{aligned}$$

8-bit values within [0, 255]

Gamma-corrected RGB values within [0, 255]

* Gamma correction here means the RGB values are not luminance-linear



Chroma Subsampling (Lossy)

Downsample the two chroma components but keep luma unchanged.

4:4:4 representation (no chroma subsampling, lossless)

Y’ Cb Cr	Y’ Cb Cr	Y’ Cb Cr	Y’ Cb Cr
Y’ Cb Cr	Y’ Cb Cr	Y’ Cb Cr	Y’ Cb Cr

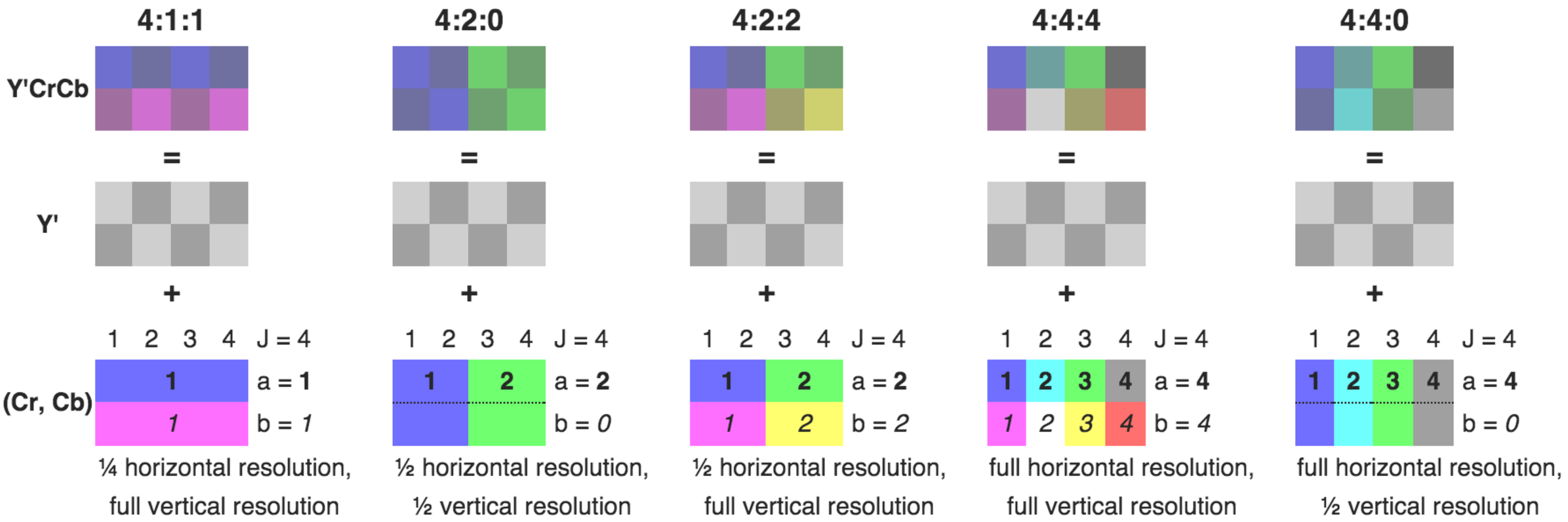
4:2:2 representation (Cb and Cr are subsampled 2X)

Y’ Cb Cr	Y’	Y’ Cb Cr	Y’
Y’ Cb Cr	Y’	Y’ Cb Cr	Y’

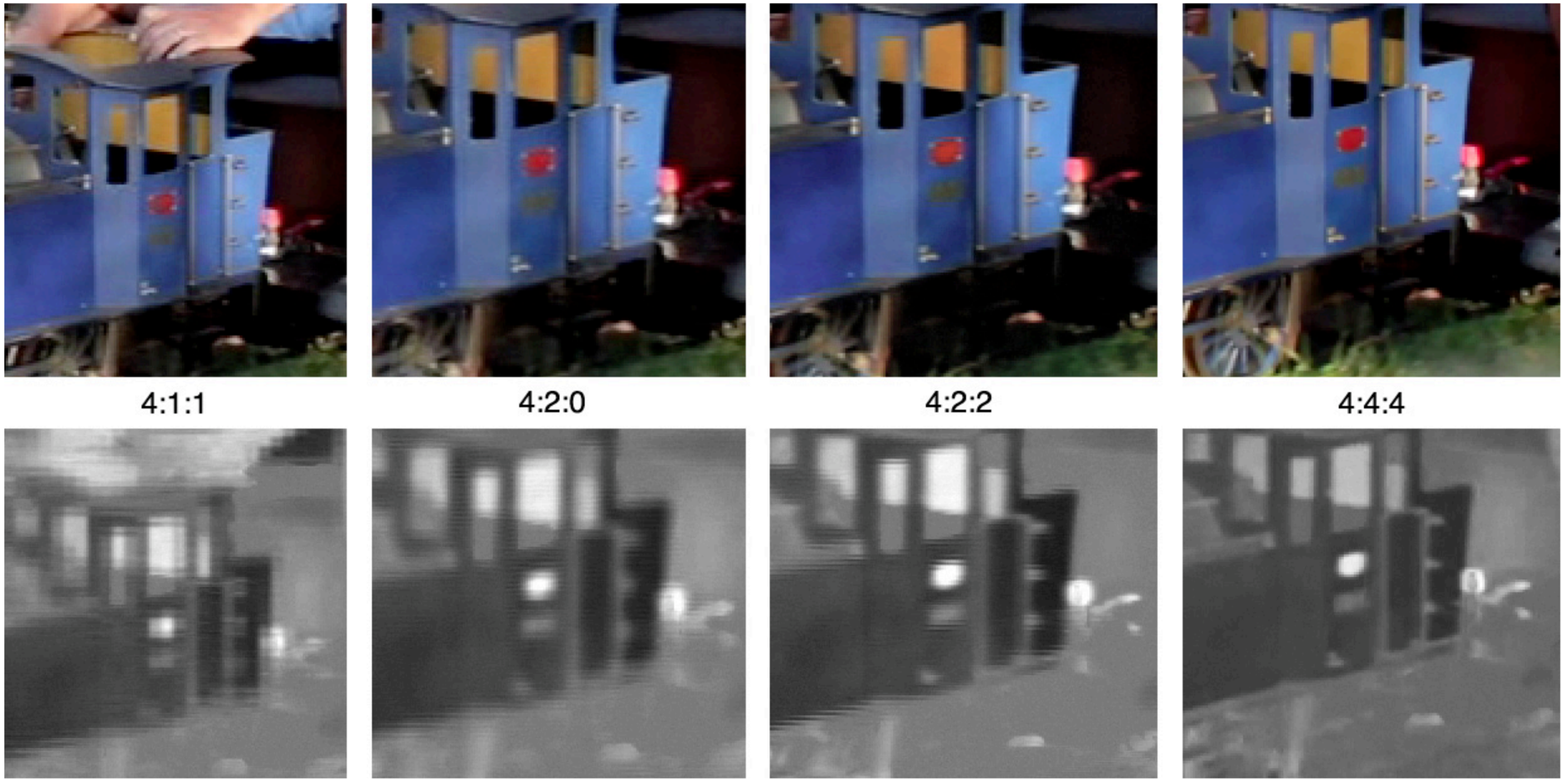
4:2:0 representation (Cb and Cr are subsampled 4X)

Y’ Cb Cr	Y’	Y’ Cb Cr	Y’
Y’	Y’	Y’	Y’

Chroma Subsampling (Lossy)



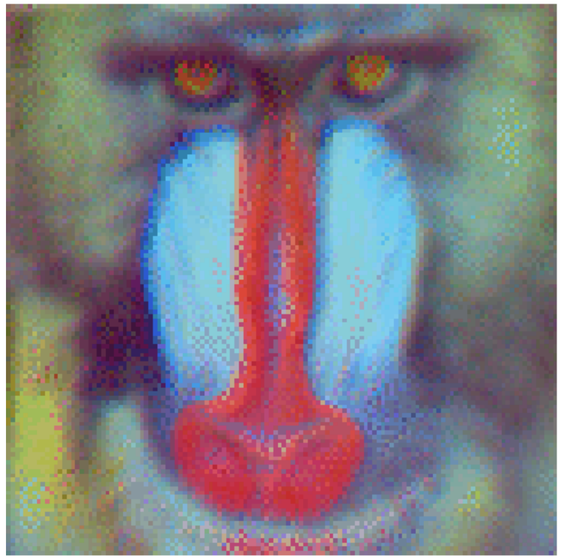
Examples



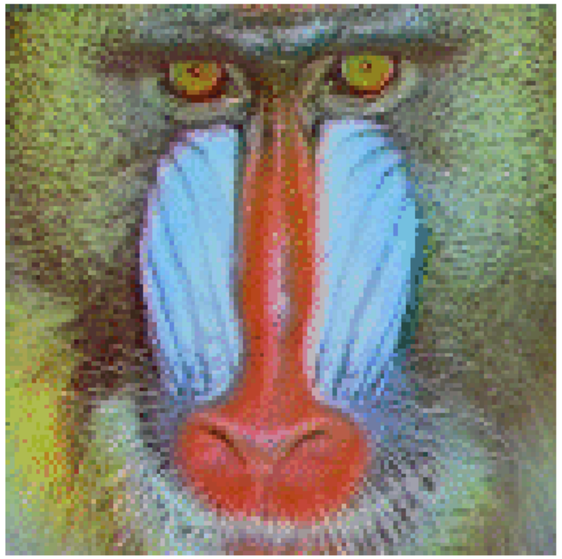
https://en.wikipedia.org/wiki/Chroma_subsampling

HVS is sensitive to blurring in dark-light channel

Blur dark-light channel



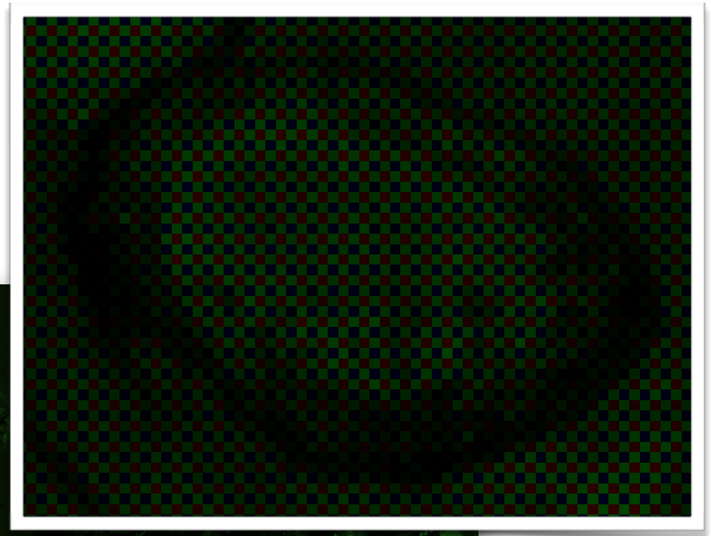
Blur red-green channel



Blur yellow-blue channel

Wandell, 1995

Subsampling in RAW RGB Space?



A bayer filter is a simple form of directly subsampling in the RGB color space. Details are visibly lost — that's why we need demosaicking!

https://storage.googleapis.com/hdrplusdata/20171106_subset/gallery_20171023/0006_20160722_115157_431.jpg

High-Frequency Information Compression

There is a highest frequency threshold beyond which human visual system can’t see.

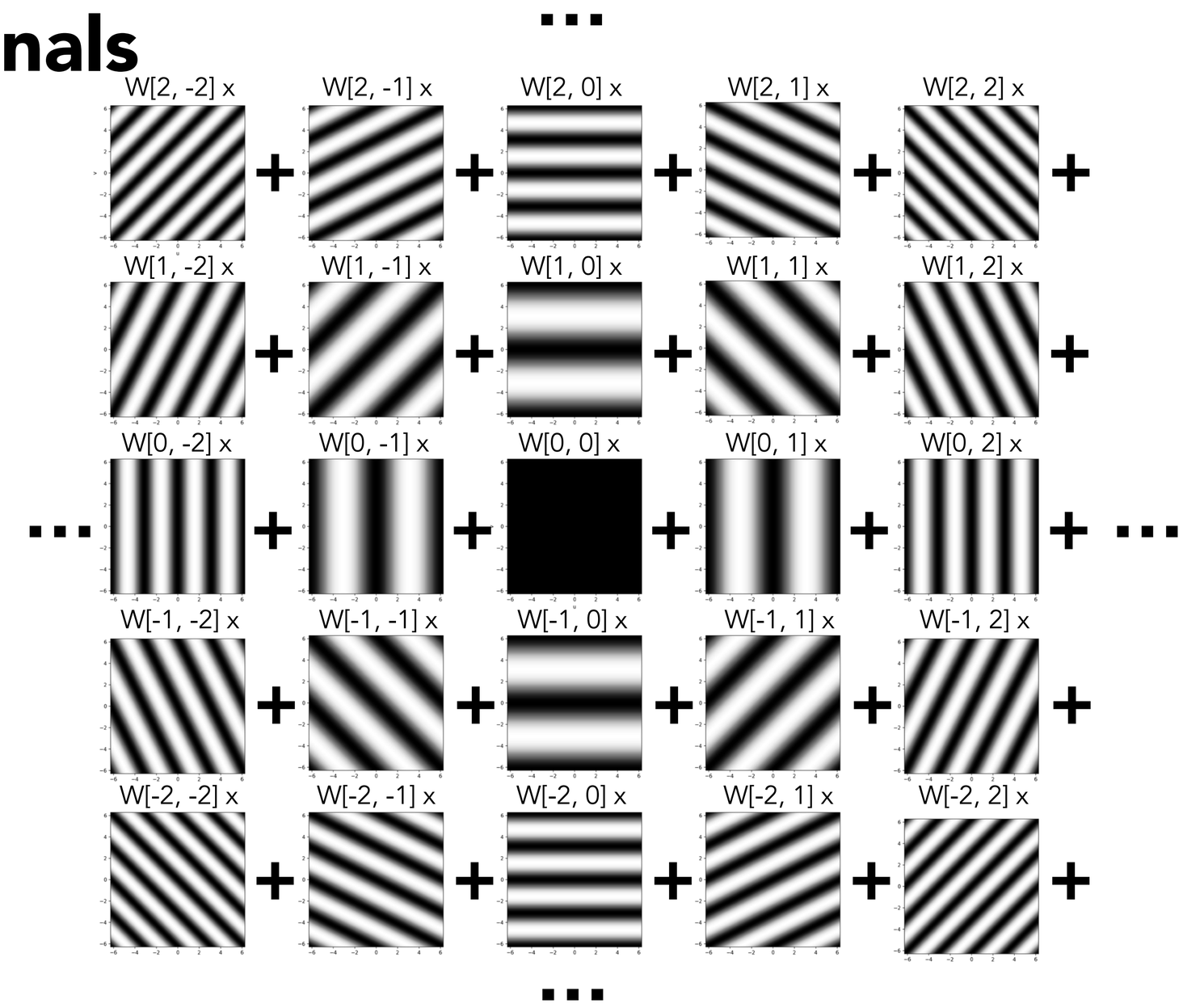
- Because photoreceptors are performing a spatial sampling

RGB/Y’CbCr represent spatial domain information. Convert to frequency domain for compression.

Recall Fourier transform: any periodic function can be exactly represented as a weighted sum of simple sinusoids.

Decomposing 2D Signals

=



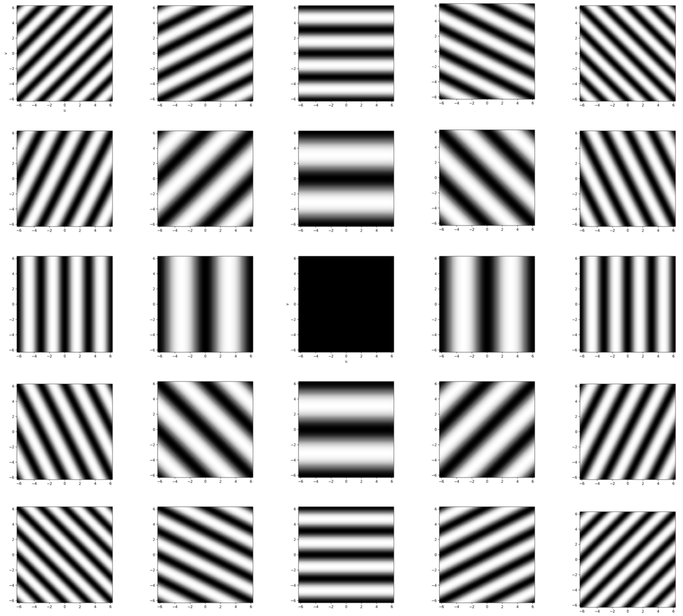
https://www.theverge.com/2016/8/2/12351210/ferrari-488-spider-supercar-test-drive

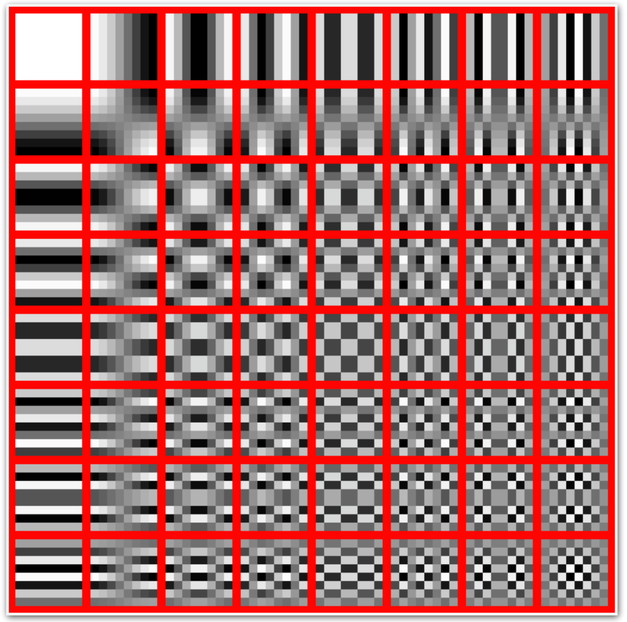
High-Frequency Information Compression

JPEG uses **Discrete Cosine Transformation** (DCT), just a different set of basis functions than what Discrete Fourier Transform uses.

- DCT (in image compression): any 8 ×8 zero-centered image can be represented by a weighted sum of the 64 8×8 images (basis functions).

Basis Functions used in DFT





Basis Functions used in DCT

DCT (Lossless)

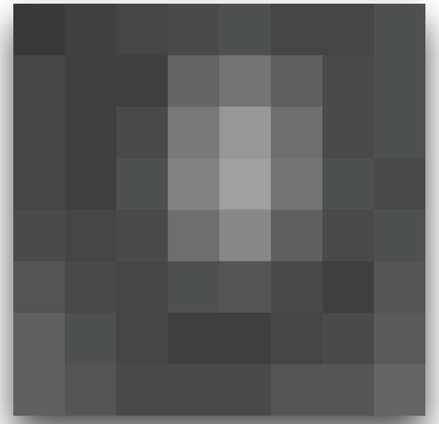
$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}.$$

Zero-centered

$$g = \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \downarrow y. \end{matrix} \quad (x \rightarrow)$$

8x8 block. Done on Y', Cb, Cr separately.



$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \downarrow v. \end{matrix} \quad (u \rightarrow)$$

$G_{0,0}$

$G_{7,7}$

Weights (coefficients)

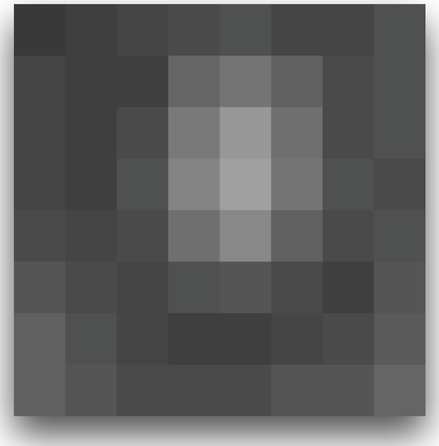
https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform

DCT (Lossless)

Calculating the DCT coefficients is mathematically lossless, but could be lossy if the computation isn't done using enough precision.

- Often 16-bit computation is needed even for 8-bit images.

An 8x8 block.



=

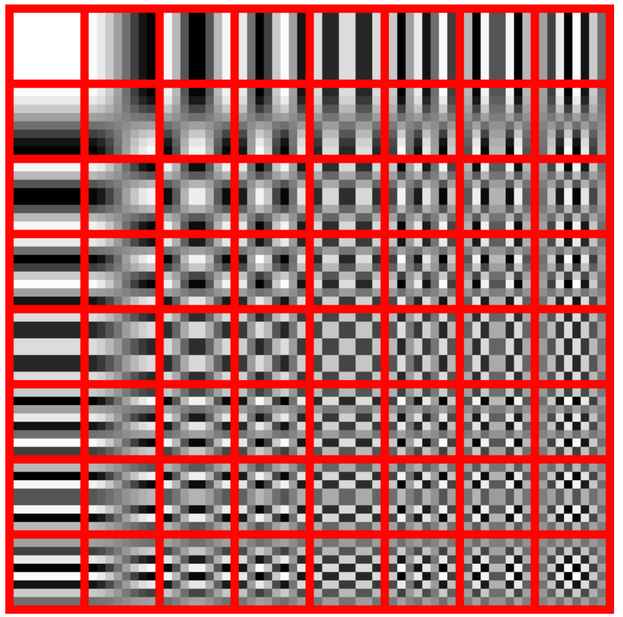
Weights (coefficients)

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \downarrow v. \end{matrix}$$

$u \longrightarrow$

X

The 8x8 basis functions. Fixed, so no need to encode.



https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform

Coefficient Quantization (Lossy)

Weights (coefficients)

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

$u \longrightarrow$ (horizontal) $\quad v \downarrow$ (vertical)

B and G aren't equivalent (lossy), but B is small fix-point numbers and thus requires fewer bits to encode.

Quantization matrix (fixed, no need to encode)

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

$$B_{u,v} = round(\frac{G_{u,v}}{Q_{u,v}})$$

Quantized coefficients

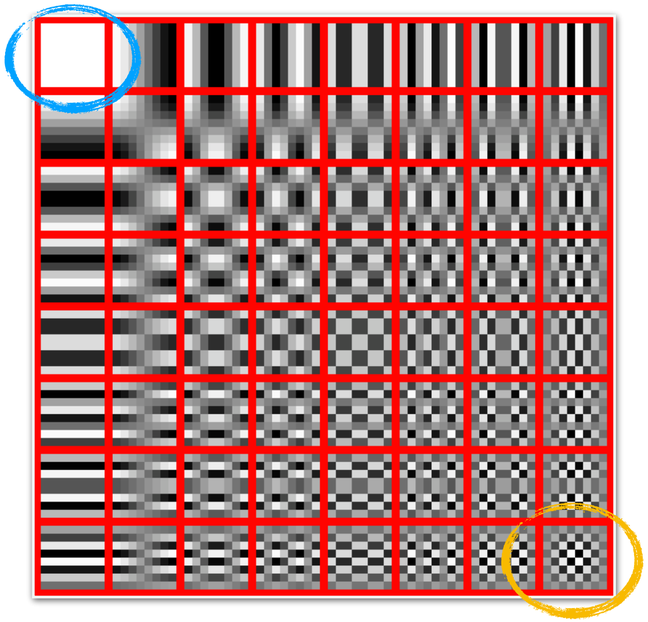
$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform

Quantization Matrix

JPEG standard specifies different quantization matrices at different quality levels. Lower quality means larger magnitudes in the matrix (quantize more).

- The exact values in the matrix are derived from modeling human perception to different frequencies.



Quantization ma

$$Q = \begin{bmatrix} 16 & 11 & 10 & & & & & \\ 12 & 12 & 14 & & & & & \\ 14 & 13 & 16 & & & & & \\ 14 & 17 & 22 & & & & & \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

Low-frequency weights are smaller, and high-frequency weights are larger. As a result, quantization zeros out most of the high-frequency coefficients — a good thing!

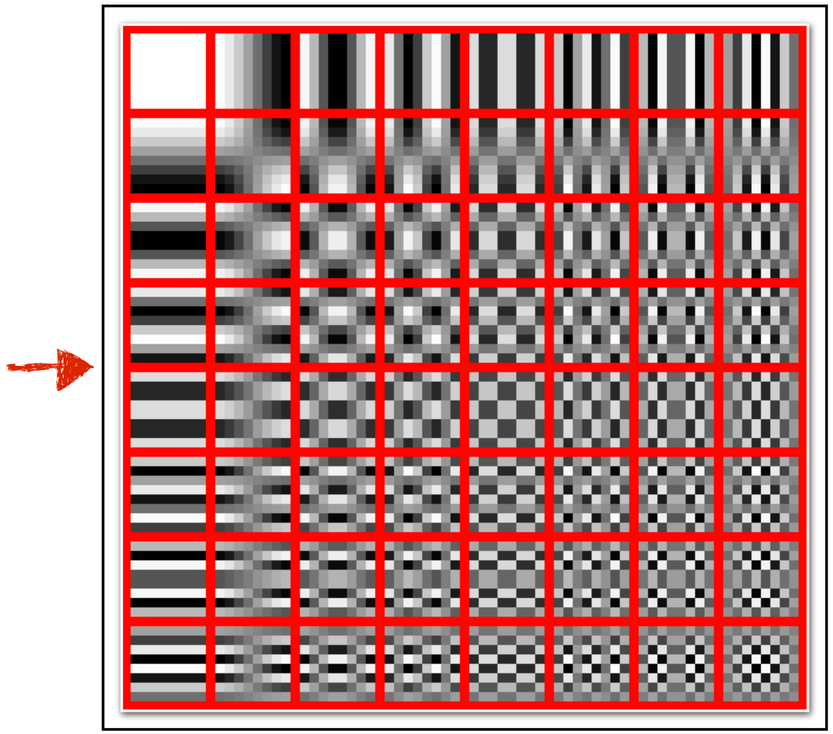
$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform

Entropy Encoding (Lossless)

1. Reorder quantized coefficients in zig-zag order, which orders the frequencies in the ascending order.
   - Why zig-zag? Look at the pattern in the basis function.

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



−26
−3 0
−3 −2 −6
2 −4 1 −3
1 1 5 1 2
−1 1 −1 2 0 0
0 0 0 −1 −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 0
0 0 0
0 0
0

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform

Entropy Encoding (Lossless)

1. Reorder quantized coefficients in zig-zag order, which orders the frequencies in the ascending order.
2. Apply run-length encoding. Compresses zeros.
3. Apply Huffman coding (or arithmetic coding) to compress the rest.

RLE

`WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW` → `12W1B12W3B24W1B14W`

https://en.wikipedia.org/wiki/Run-length_encoding

JPEG Algorithm Recap

A combination of lossless and lossy techniques.

1. Gamma encode RGB values.
2. Convert image to Y'CbCr color space.
3. Chroma subsampling.
4. For each channel (Y', Cb, Cr)
   A. For each 8x8 pixel block
      1. Compute DCT coefficients
      2. Quantize DCT coefficients
      3. Entropy encoding

Evenly encode perceived brightness, not luminance. Lossy due to quantization errors, but it's not unique to JPEG compression.

Human eyes are sensitive to luminance difference but not chrominance.

Human eyes are insensitive to defects in high-frequency information, but not low-frequency information.

Compression Quality

Quality level (Q) = 100
Compression ratio: 2.7 : 1

Quality level (Q) = 25
Compression ratio: 23 : 1

Quality level (Q) = 1
Compression ratio: 144 : 1

https://en.wikipedia.org/wiki/JPEG

Image Compression Summary

JPEG is lossy (chroma subsampling and coefficient quantization).

- PNG and TIFF are lossless compression standards.

Quantization artifacts are most visible across edges and sharp corners (high frequency areas), where information loss is the greatest.

- For this reason, JPEG isn't good for compressing drawings and graphics. Co-designing graphics algorithms with compression algorithms?

The compressing (encoding) and uncompressing (decoding) system (software and hardware) is called the "*codec*", which are most definitely done specialized hardware. Your smartphone has one.

Video Compression

Video Compression

Compressing a sequence of continuous frames together.

What about compressing each image using JPEG?

That’s what Motion-JPEG (M-JPEG) does.

- Support by most video codecs, but not widely used.
- Not well-standardized.
- Not very efficient.
- Fundamentally, compressing image individually ignores the temporal dimension in a video.

Common Video Compression Standards

MPEG-2 part 2, a.k.a., **H.262**.

- Old but still widely used.

MPEG-4 part 10, a.k.a., **H.264** or Advanced Video Coding (AVC).

- Most widely used; 91% of video industry developers (9/2019); mandated by blue-ray.

MPEG-H part 2, a.k.a., **H.265** or High Efficiency Video Coding (HEVC).

- Gradually taking over

Common Video Compression Standards

There are usually different “profiles” and “levels” within each MPEG standard.

- Profiles refer to different algorithmic choices; levels refer to different parameter choices

MPEG standards don’t define the encoding process, but defines the format of a coded stream and the decoding process.

- You can implement your own H264 encoder/compression algorithm as long as the compressed format complies with the standard.

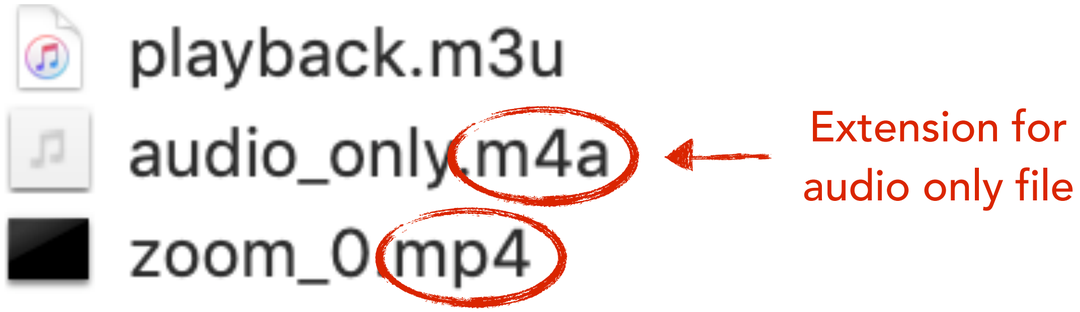
VP8/VP9, from Google, open-sourced; compete with H.264 and H.265.

Container Format

A container format bundles different data types, e.g., video, audio, subtitles.

Perhaps the most commonly used container format is .mp4, which is defined in MPEG-4 Part 14.

- Supports H.264 and H.265 video encoding and MPEG-4 Part 3 audio encoding.
- How many of you owned a “MP4 player”?! Can you imagine we use to buy devices that do nothing but playing .mp4 files?

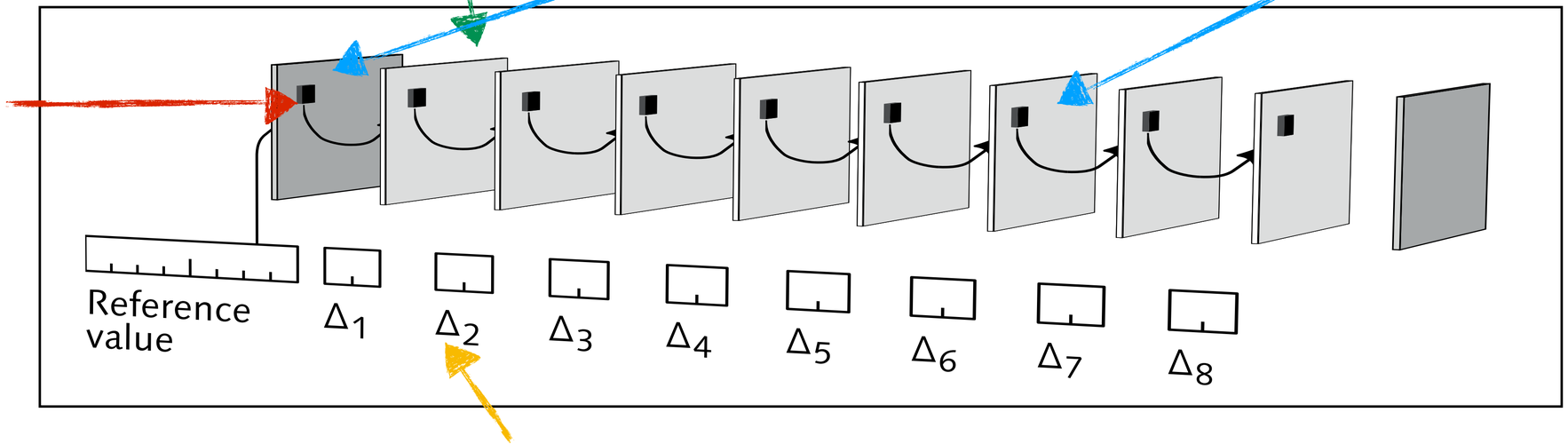


https://www.amazon.com/Original-Appleipod-Compatible-Gigabyte-Classic/dp/B07Z64J2T1

Video Compression: The Big Ideas

Inter-frame compression
(Compressed as deltas/residuals w.r.t., to the reference frame)

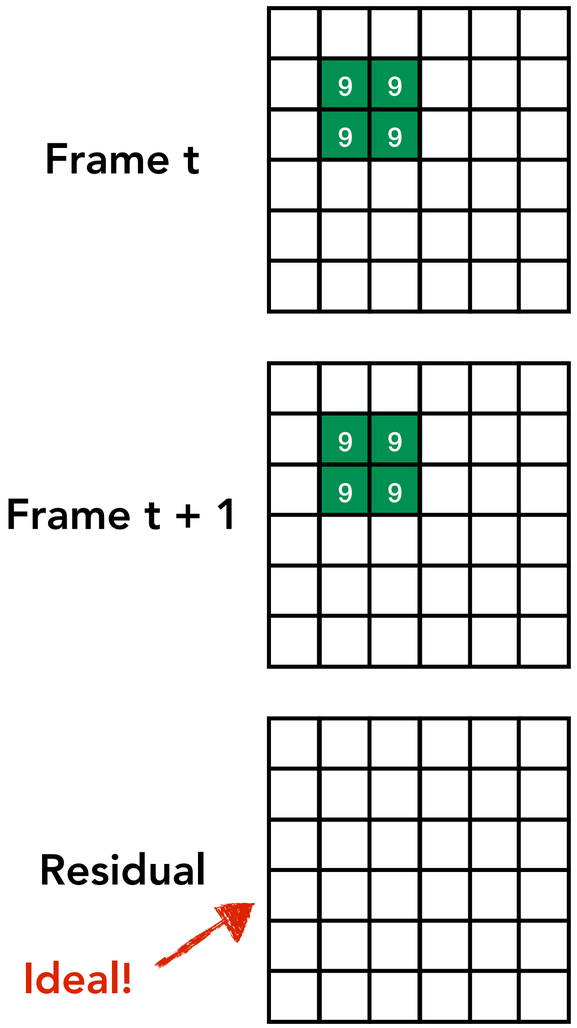
Intra-frame compression
(Compressed individually)

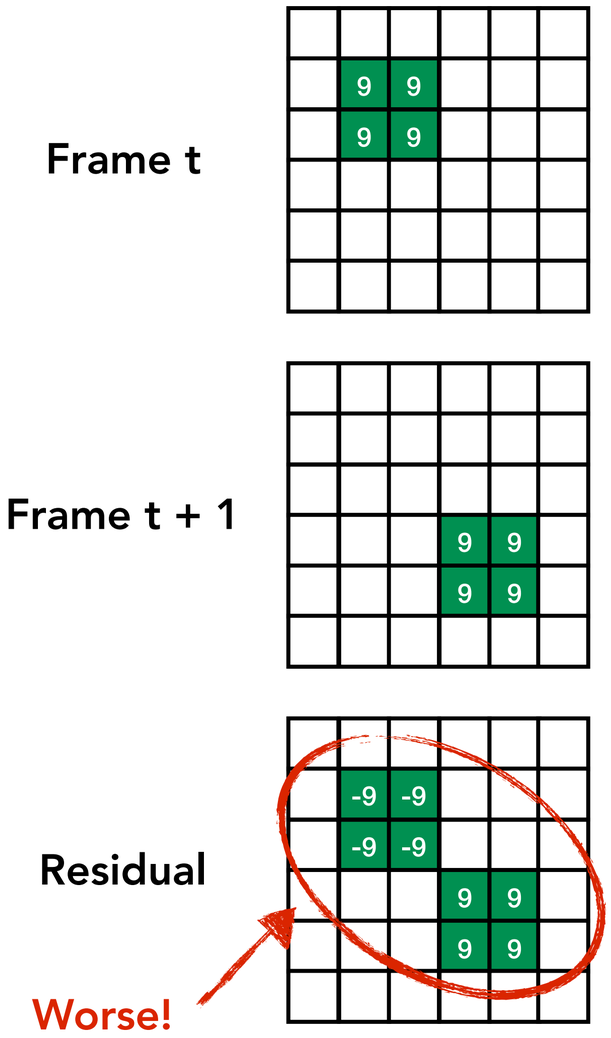


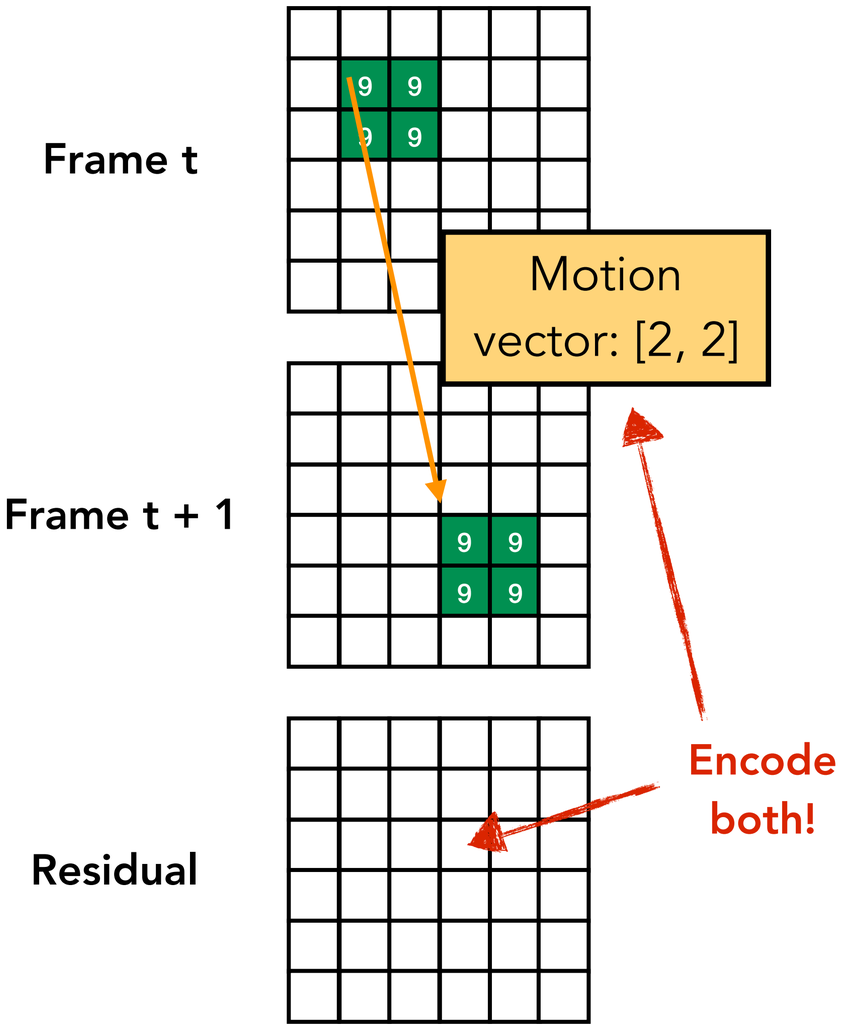
Provided that residuals can be encoded more compactly than the image itself.

https://www.researchgate.net/figure/Comparison-of-Optical-Flow-and-Scene-Flow-For-four-stereo-images-of-a-traffic-scene-top_fig1_322500134 Figure 16.2, Digital Video and HDTV Algorithms and Interfaces, Charles Poynton

Residuals

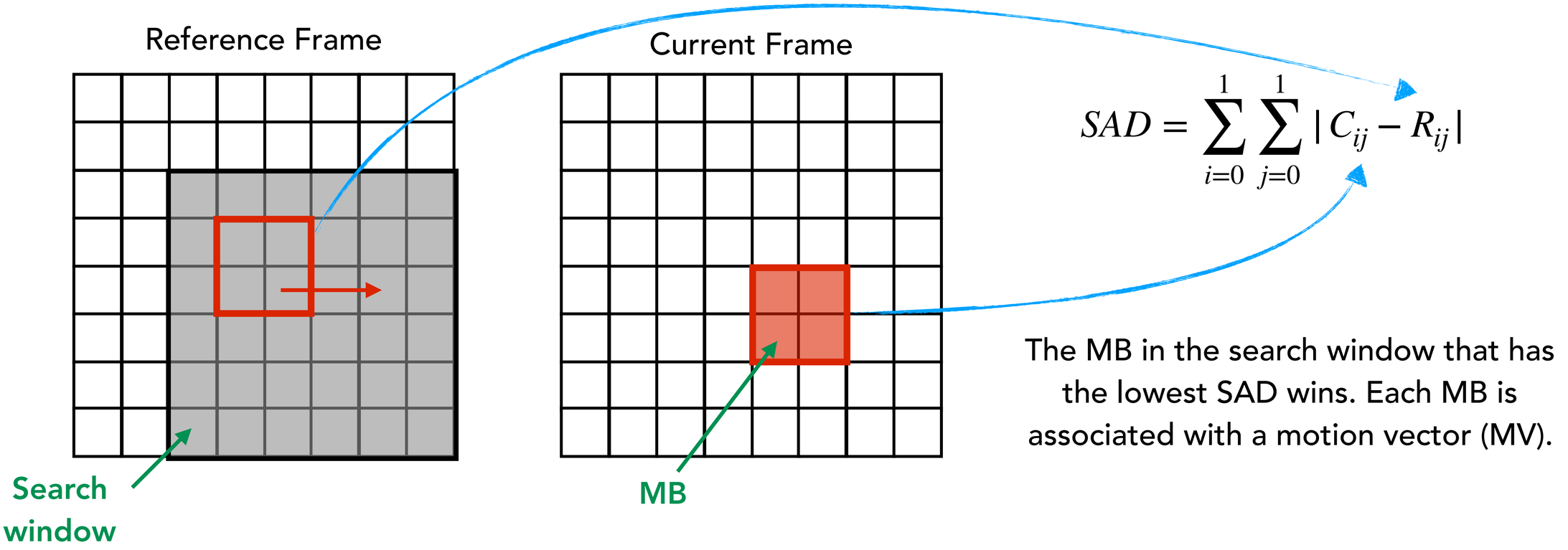

Frame t
9 9
9 9
Frame t + 1
9 9
9 9
Residual
Ideal!


Frame t
9 9
9 9
Frame t + 1
9 9
9 9
Residual
-9 -9
-9 -9
9 9
9 9
Worse!


Frame t
9 9
9 9
Motion vector: [2, 2]
Frame t + 1
9 9
9 9
Residual
Encode both!

Motion Estimation

In MPEG video compression, motion estimation is done using the **block matching** algorithm, which operates at the granularity of **macroblocks** (MB), and uses some form of **absolute difference** as the cost function.

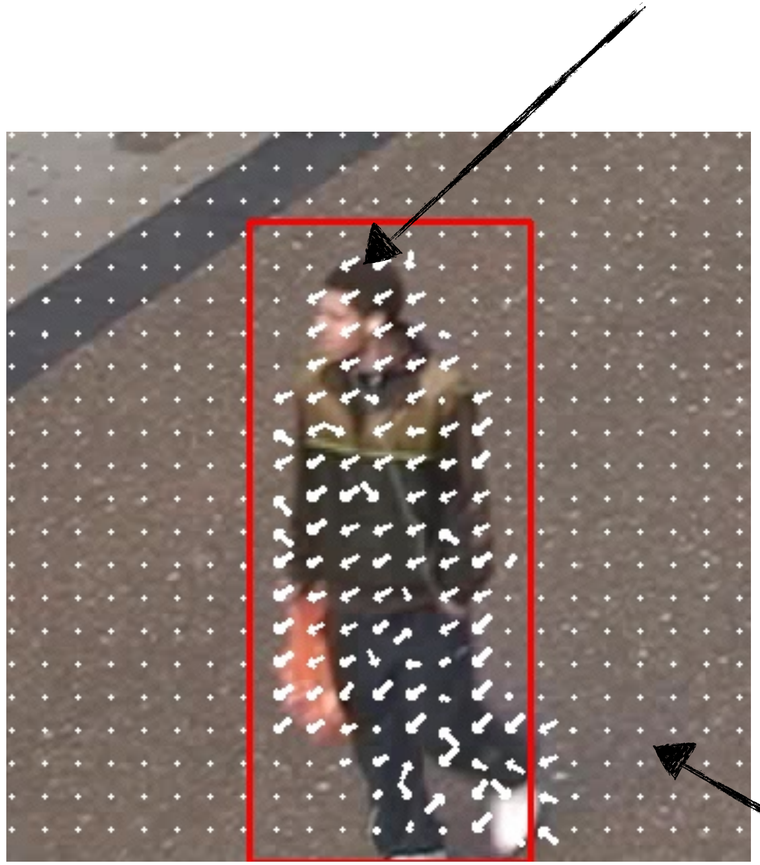


Motion Vectors

A frame from a still camera capturing a moving person. Each arrow represents the MV of a 8x8 MB.

Motion vectors can be used to inferred the scene: object detection, tracking, etc.

Looking at just the MVs (not the image itself), can you guess where the object is moving?



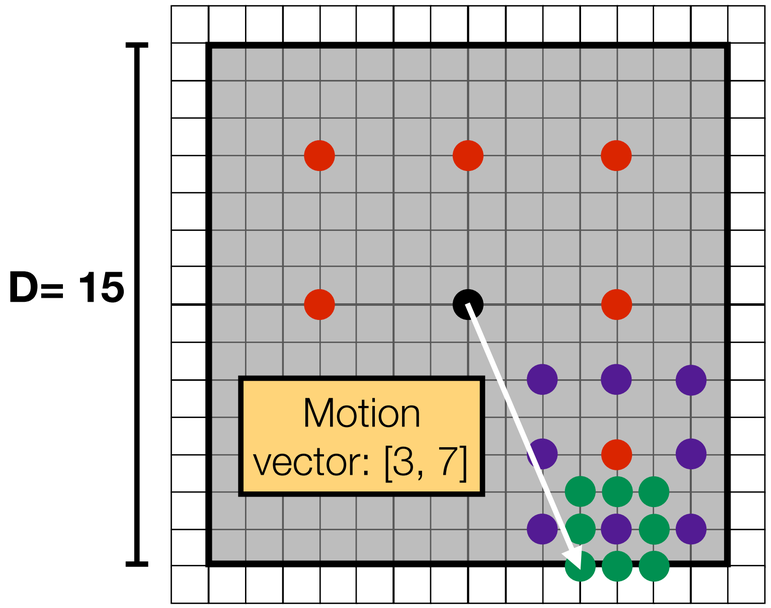
Why do these pixels have zero motion?

Motion Estimation Heuristics

Exhaustive search (ES) is costly. Lots of heuristics are proposed. Always a trade-off between accuracy vs. compute efficiency.

Three step search (TSS) is a classic heuristics that adaptively narrows the search range (coarse-to-fine search).

In ES, 255 MBs are to be searched if the search window size is 15x15. With TSS, only 25 MBs are searched. 25X speed-up.



Encoding Residuals

Motion estimation will most likely not find a perfect match, i.e., the cost function will not be 0. Therefore, the residuals must still be encoded.

The residuals form a residual image, which is encoded in a JPEG-like fashion.

- The quantization matrices are usually different from the ones used in encoding images.

9	8	1	3
7	6	9	0
1	1	9	0
1	9	1	2

Frame t

1	3	8	8
8	1	6	6
9	2	1	1
2	2	1	9

Frame t+1

0	0	-1	0
-1	1	-1	0
0	2	0	0
1	0	0	0

Residual

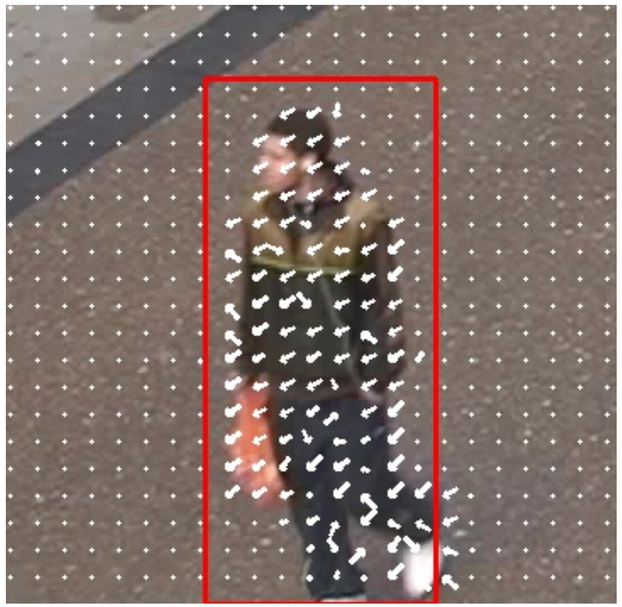
0, 1	1, 0
0, 1	1, 0

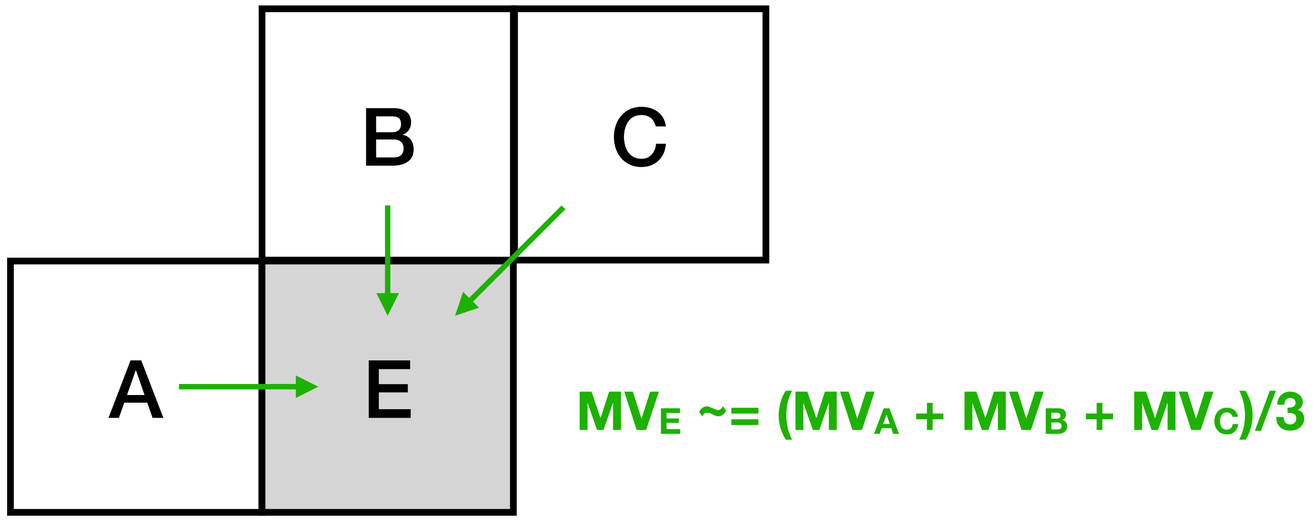
Motion vectors

Encoding Motion Vectors

Motion vectors of nearby MBs are often correlated (rigid objects).

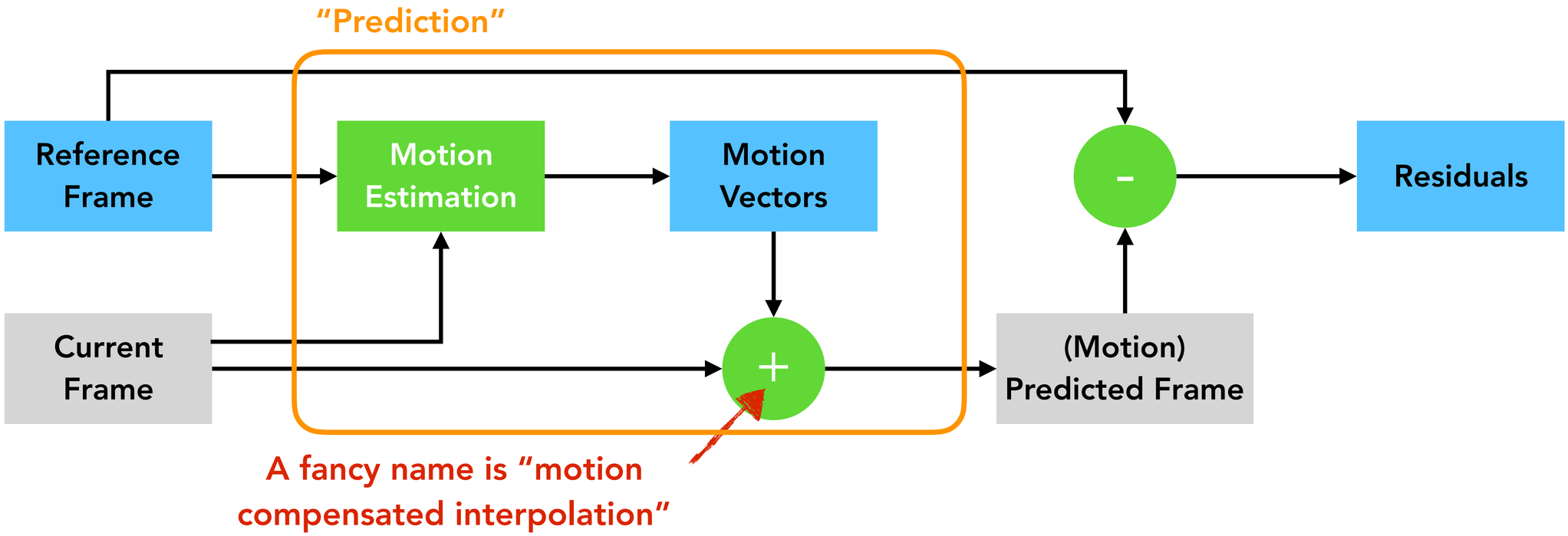
Idea: use nearby MVs to estimate/predict the current MV, compute the residuals, and encode the MV residuals.





Inter-Frame Encoding Flow

Data that needs to be encoded



Reference Frame Choice

In MPEG, a video is partitioned into successive groups of pictures (GoPs), each of which contains a sequence of successive frames.

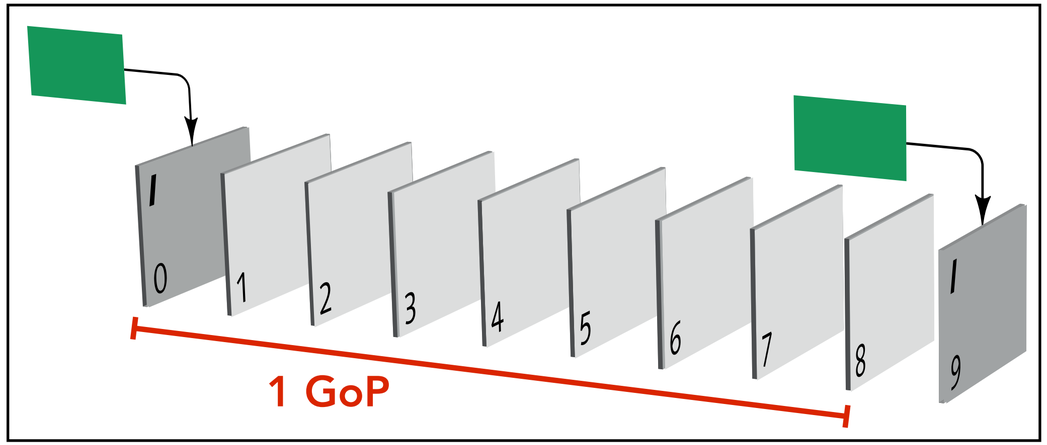
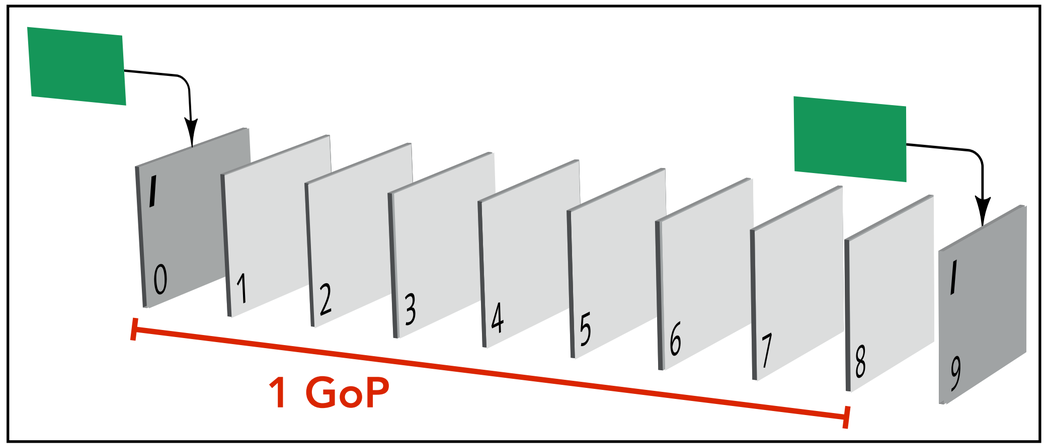


Figure 16.2, Digital Video and HDTV Algorithms and Interfaces, Charles Poynton

Reference Frame Choice

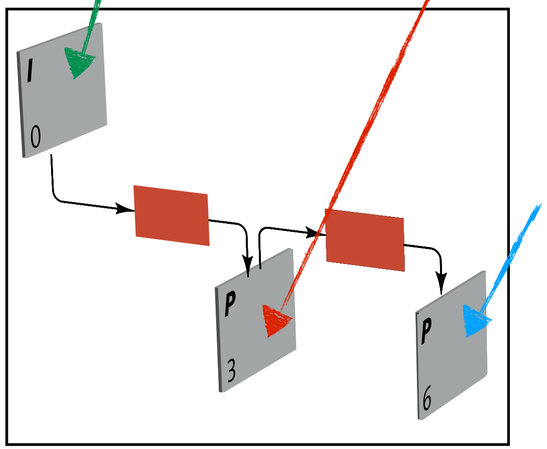
Each GoP has three (mutually exclusive) kinds of frames:

- An **I-frame** is intra-compressed and is a reference frame.
- A **P-frame** is a frame that is predicted from a reference frame, and itself can be used as a reference frame.



The I-frame

A P-frame, predicted from the I-frame, and becomes a reference frame itself



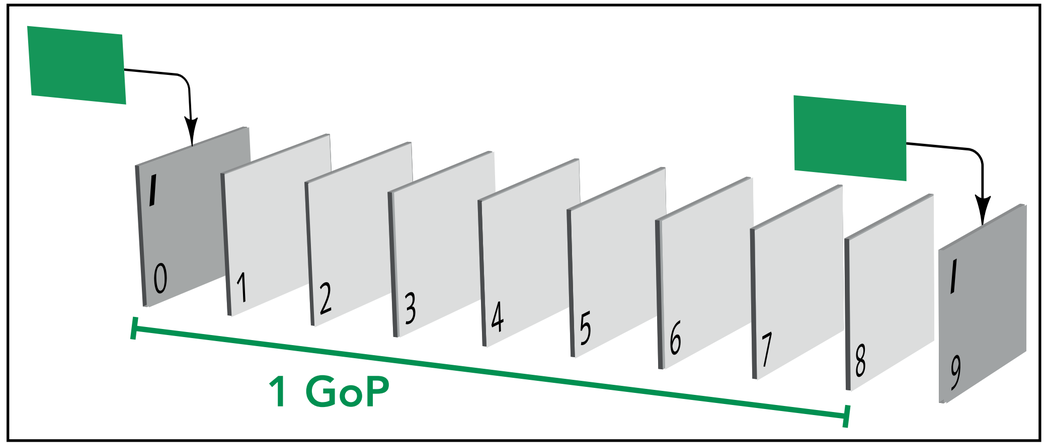
Another P-frame, predicted from P-frame 3, and is not used as a reference frame.

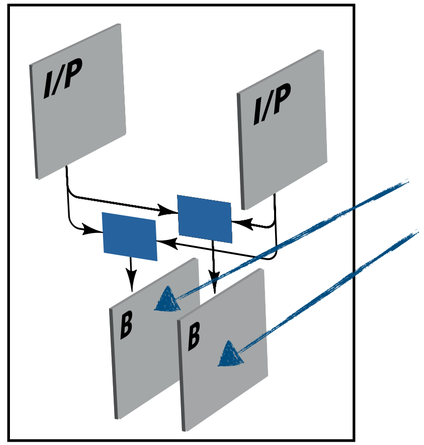
Figure 16.2/16.3, Digital Video and HDTV Algorithms and Interfaces, Charles Poynton

Reference Frame Choice

Each GoP has three (mutually exclusive) kinds of frames:

- An **I-frame** is intra-compressed and is a reference frame.
- A **P-frame** is a frame that is predicted from a reference frame, and itself can be used as a reference frame.
- A **B-frame** is a frame that is predicted from two reference frames (because the residuals are smaller that way), but itself can't be used as a reference frame.





Two B-frames. Usually a B-frame is predicted from the average of the two reference frames (but could also take a weighted average).

Figure 16.2/16.4, Digital Video and HDTV Algorithms and Interfaces, Charles Poynton

Reordering Frames in a GoP

$I_0B_1B_2P_3B_4B_5P_6B_7B_8$

Bad for live streaming/teleconferencing!!!

1. When encoding, need to buffer $\mathbf{B_1}$ and $\mathbf{B_2}$ before $\mathbf{P_3}$ arrives (e.g., from camera).
2. When decoding, need to buffer $\mathbf{B_1}$ and $\mathbf{B_2}$ before $\mathbf{P_3}$ arrives (e.g., from Internet).

$I_0P_3B_1B_2P_6B_4B_5(I_9)B_7B_8$

Solution: reorder frames during transmission!

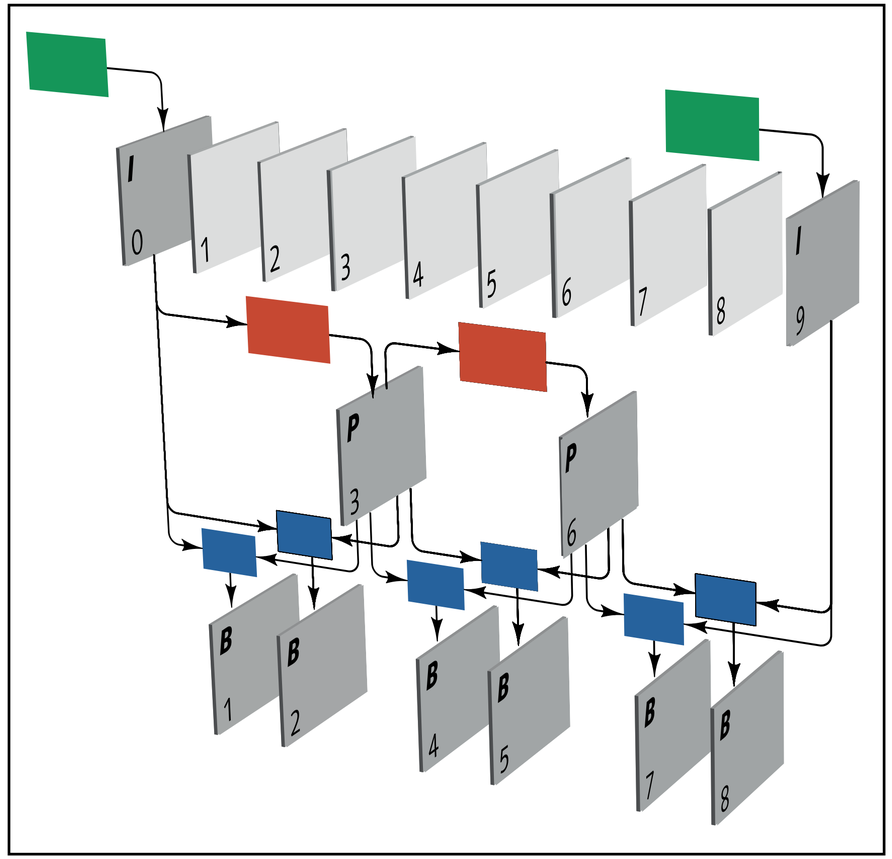


Figure 16.5, Digital Video and HDTV Algorithms and Interfaces, Charles Poynton

Incorporating the Decoder in the Encoder

B_1 is generated from I_0 and P_3, but both I_0 and P_3 themselves are compressed in a lossy fashion. What the decoder sees are the lossy versions I_0' and P_3'.

This means we should really predict/ motion-estimate B_1 from I_0' and P_3'.

How? Incorporate the decoder in the encoder so that we use I_0' and P_3' exactly as how decoder will see them.

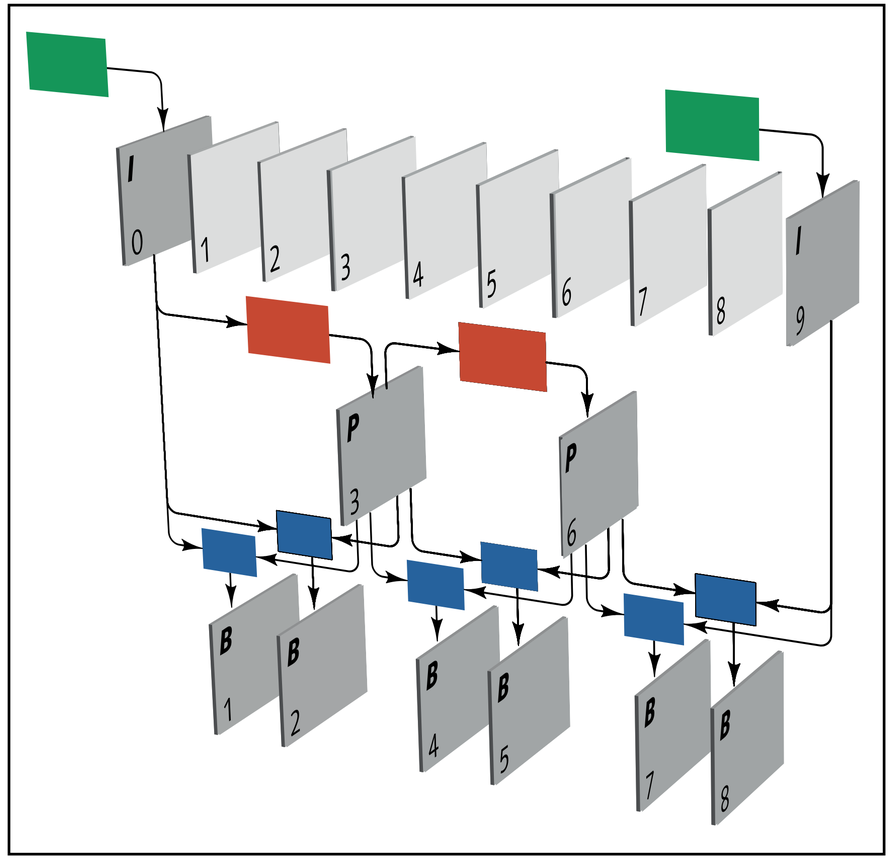


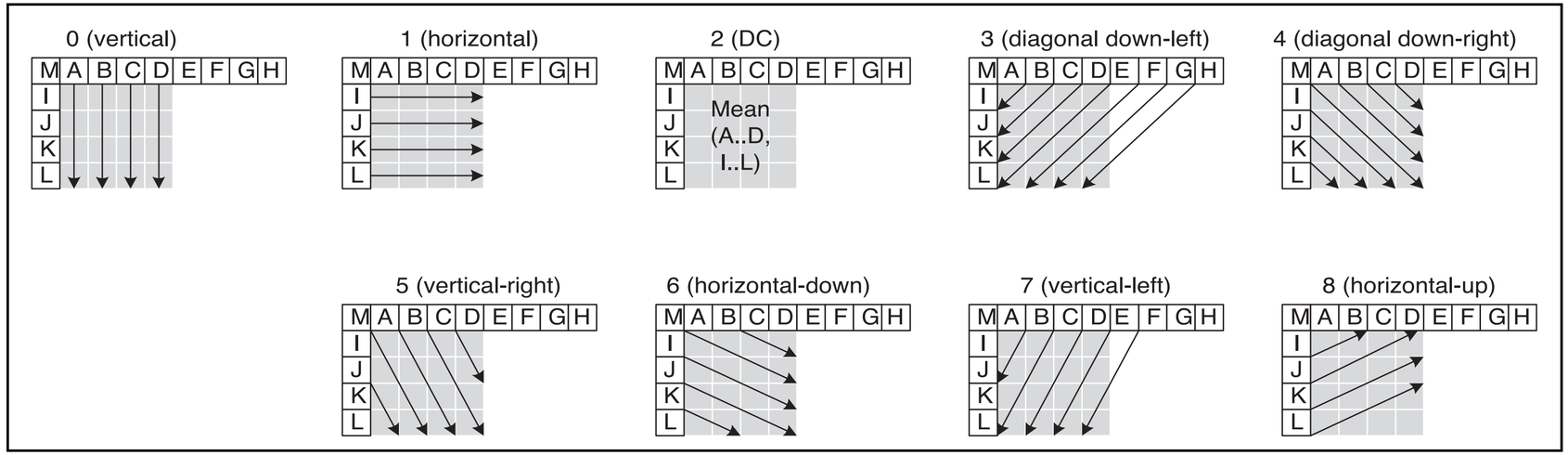
Figure 16.5, Digital Video and HDTV Algorithms and Interfaces, Charles Poynton

Intra-Frame Encoding (Spatial Prediction)

Prior to H.264, I-frames are simply compressed using JPEG-like techniques.

Starting from H.264, I-frames macroblocks are spatially predicted, exploring the observations that spatially adjacent pixels are strongly correlated.

- Again residuals are then encoded (same as in predicted frames).



Modes for spatially-predicting a 4x4 tile

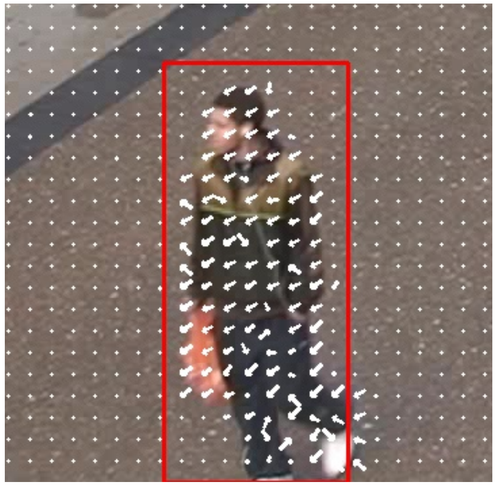


Figure 6.11, The H.264 advanced video compression standard, Iain E. Richardson

H.264 Encoding Architecture

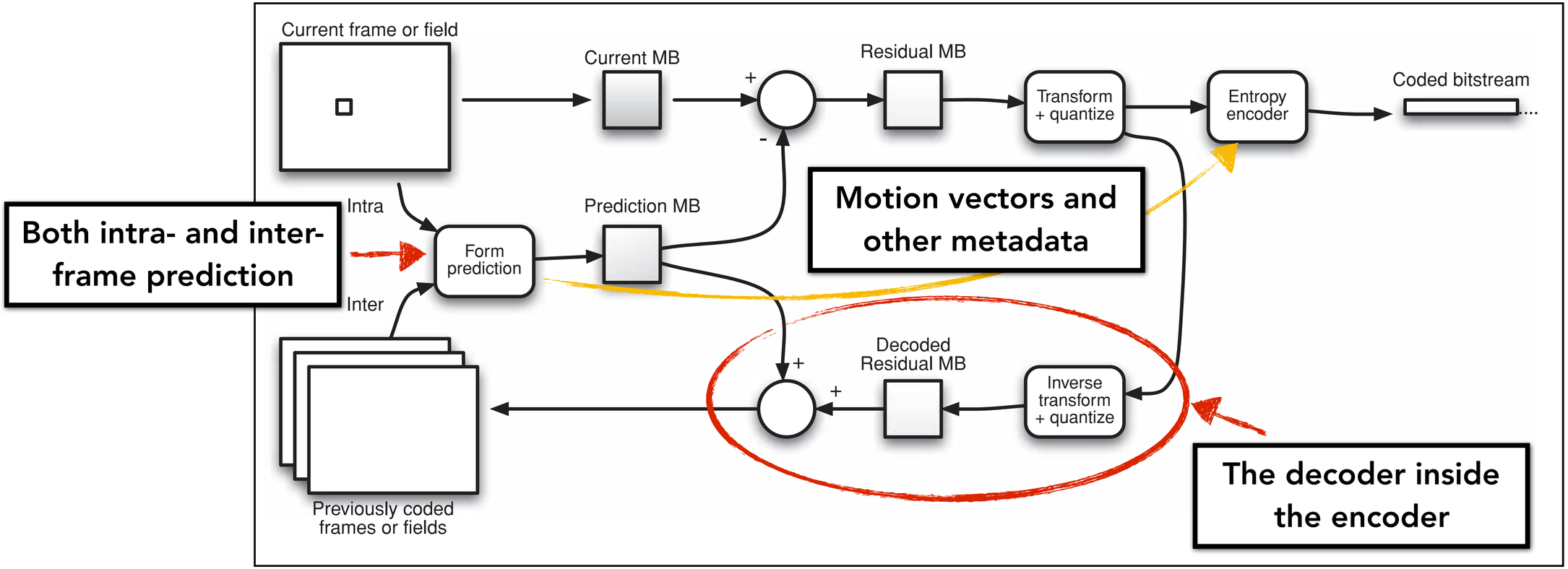


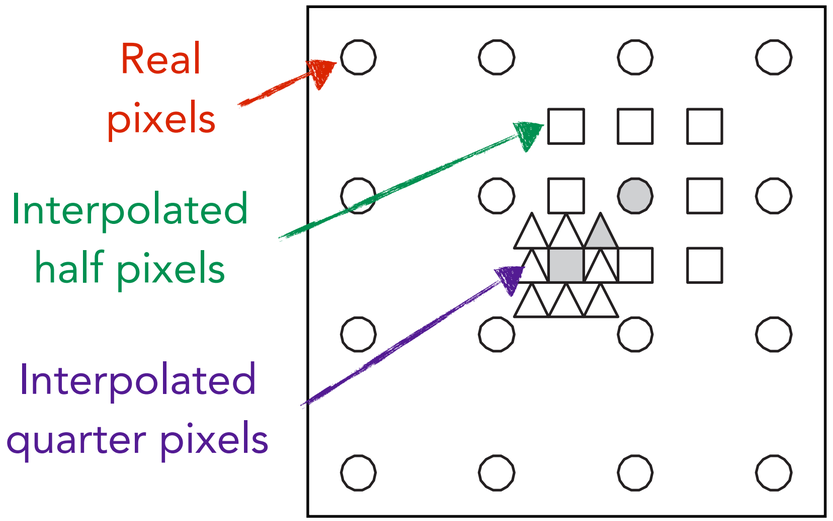
Figure 4.4, The H.264 advanced video compression standard, Iain E. Richardson

Other Details/Optimizations in Video Codecs

Luma and chroma components are dealt with separately, both in intra-compressed and inter-compressed schemes.

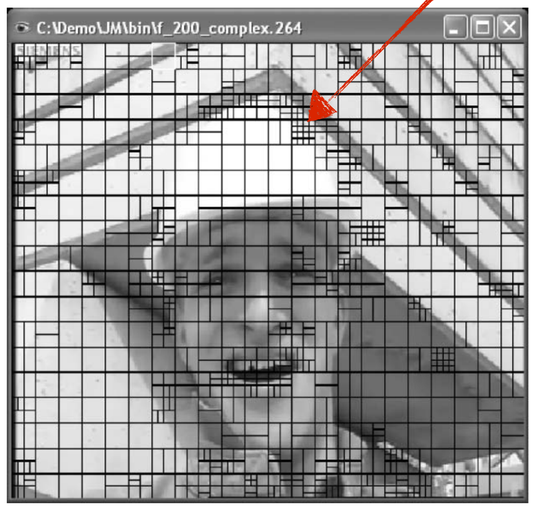
Allow sub-pixel motion (smaller residuals).

Allow different MB sizes.



"Busy" areas in a frames are better predicted using smaller MBs.

Different MB sizes in a P-frame.



Different MB sizes in an I-frame.

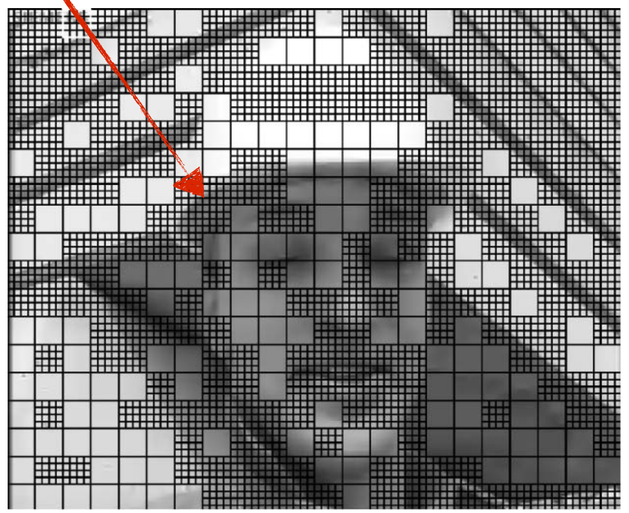


Figure 3.18/6.5/6.37, The H.264 advanced video compression standard, Iain E. Richardson

Video Compression Recap

Spatial redundancy

- Pixels in a small neighborhood have strong correlations. I-frames are spatially predicted.

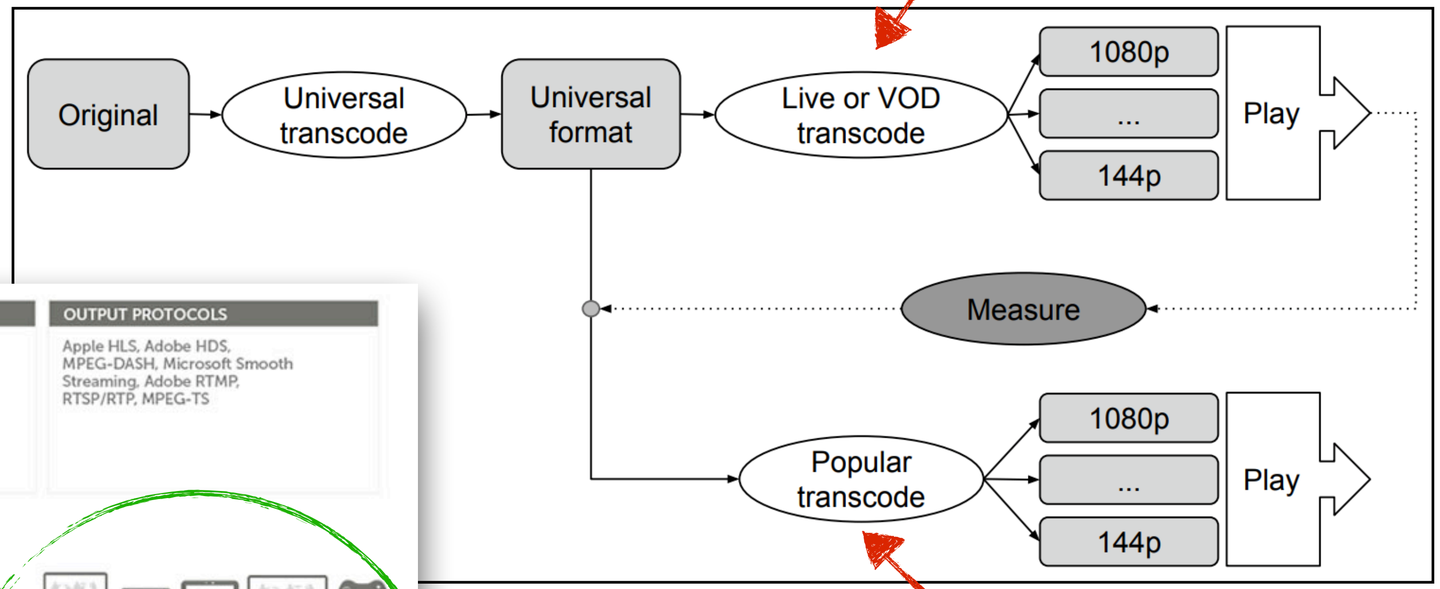
Temporal redundancy

- Consecutive frames have strong correlation. P/B-frames are temporally predicted.

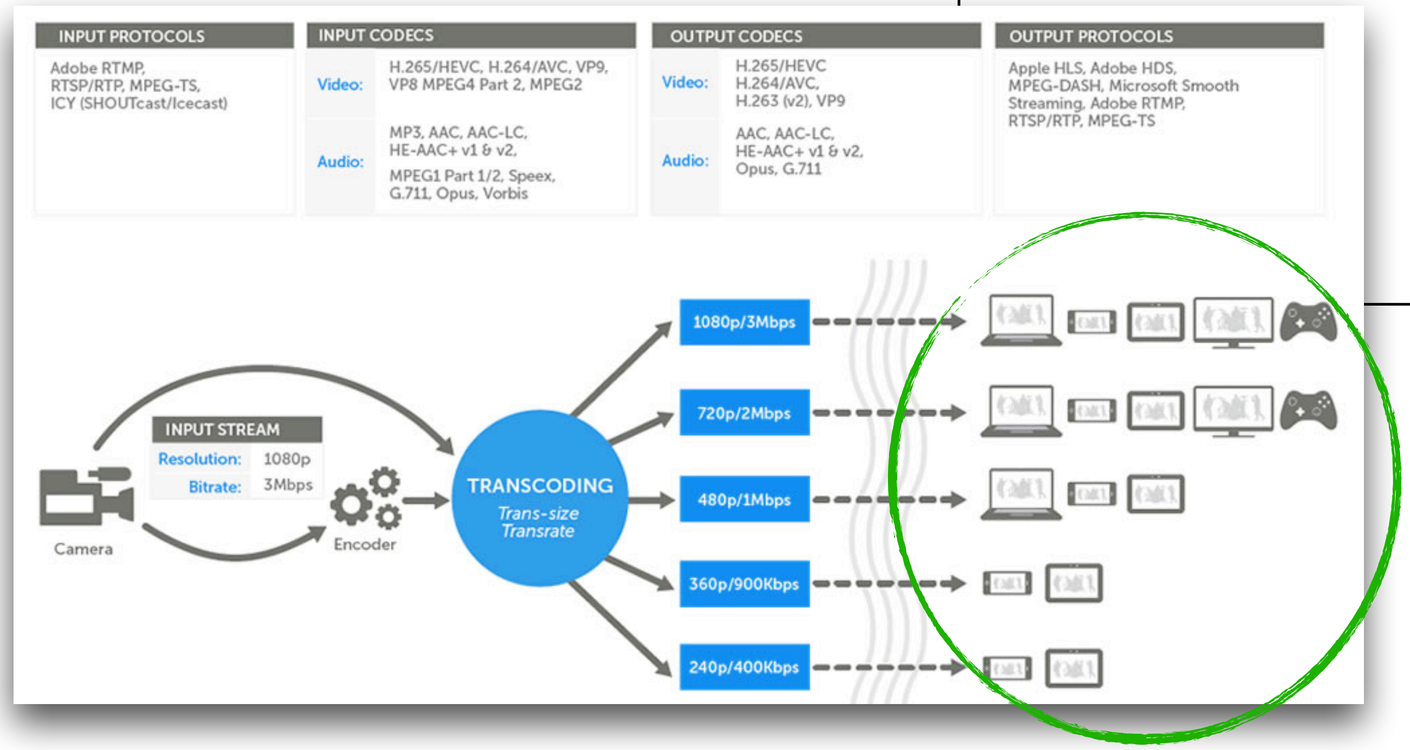
Encoding time is dominated by motion estimation. Encoding and decoding are widely support by fixed-function hardware (ASICs).

Video Transcoding

Video-on-demand (VOD). Speed is key.



Spend more time optimizing popular videos.

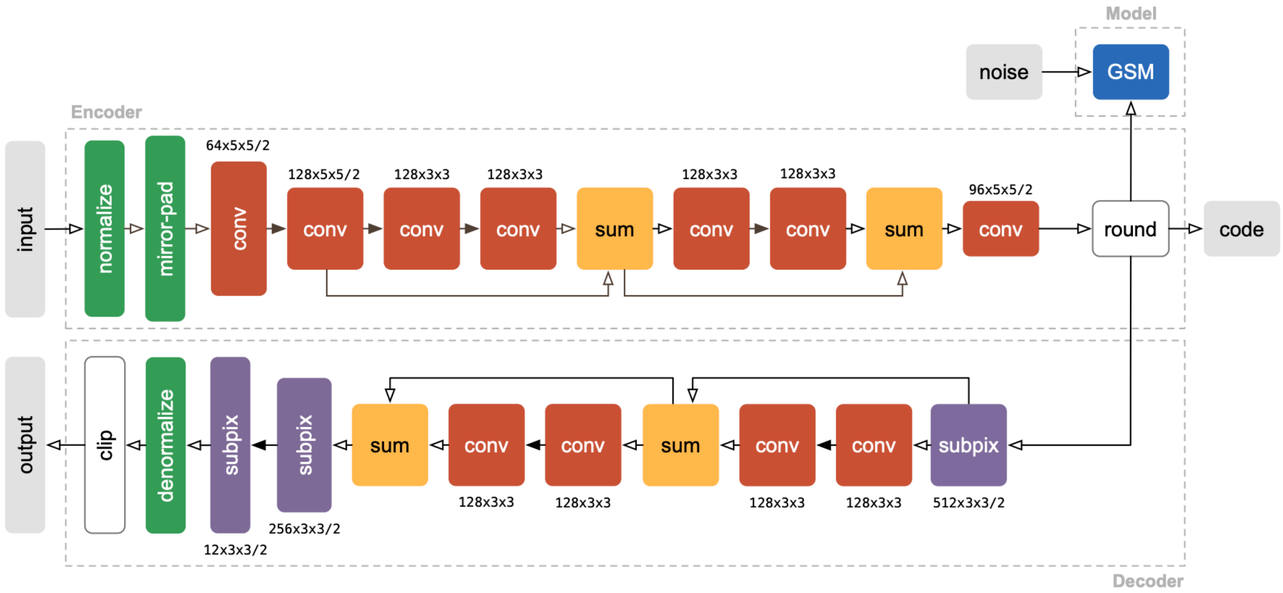


A huge design space (different resolutions, codecs, compute capabilities, bandwidths) Must make trade-offs!

https://streamingpulse.com/main/live-audio-video-transcoding-ndvr/

vbench: Benchmarking Video Transcoding in the Cloud, ASPLOS 2018

AI For Compression and Compressing for AI



Immediate goal: remove empirical decisions and heuristics from compression. Learn the best compressed representation automatically.

A better question: how to design the compression algorithm (network) **for computer vision algorithms**? After all, many videos will be consumed by not humans, but computers.

1. Vision algorithms might require a different quality measure from human
2. Different vision algorithms might require a different compression scheme

Lossy Image Compression with Compressive Autoencoders, ICLR 2017