

Lecture 16: Image and Video Compression

Yuhao Zhu

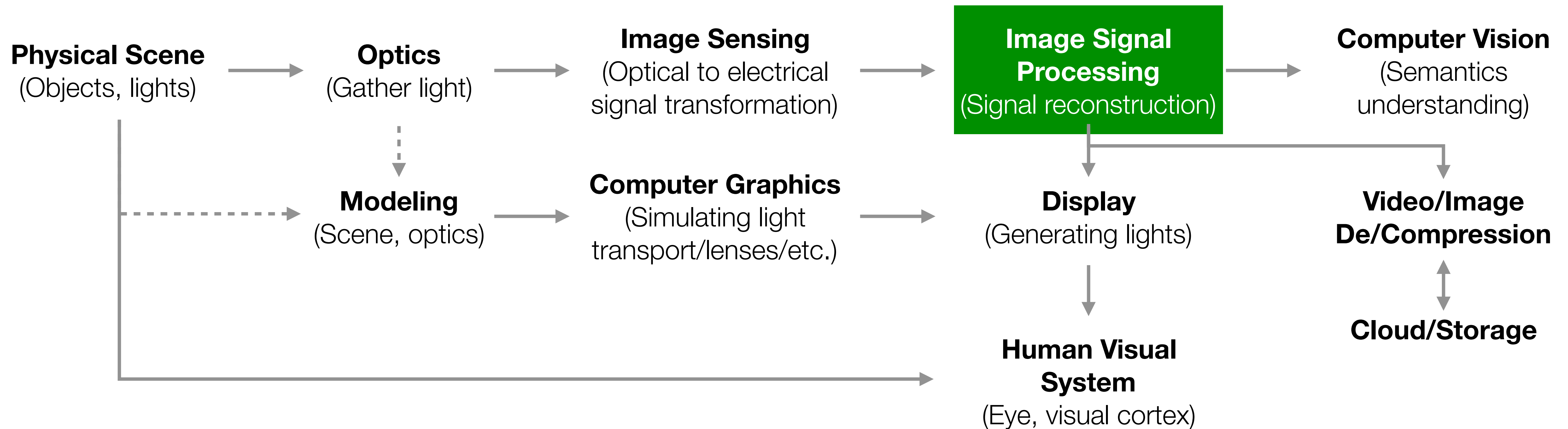
<http://yuhaozhu.com>

CSC 292/572, Fall 2022
Mobile Visual Computing

Logistics

PA2 due 11/16, 11:30 AM.

Where Are We



The Roadmap

Theoretical Preliminaries

Human Visual Systems

Color in Nature, Arts, Tech
(a.k.a., the birth, life, and death of light)

Digital Camera Imaging

Modeling and Rendering

Applications



Optics in Camera

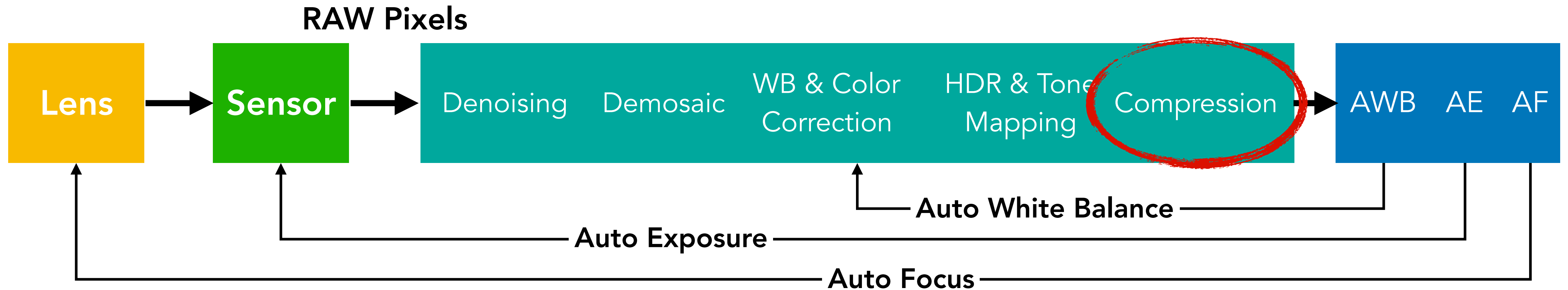
Image Sensor

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).

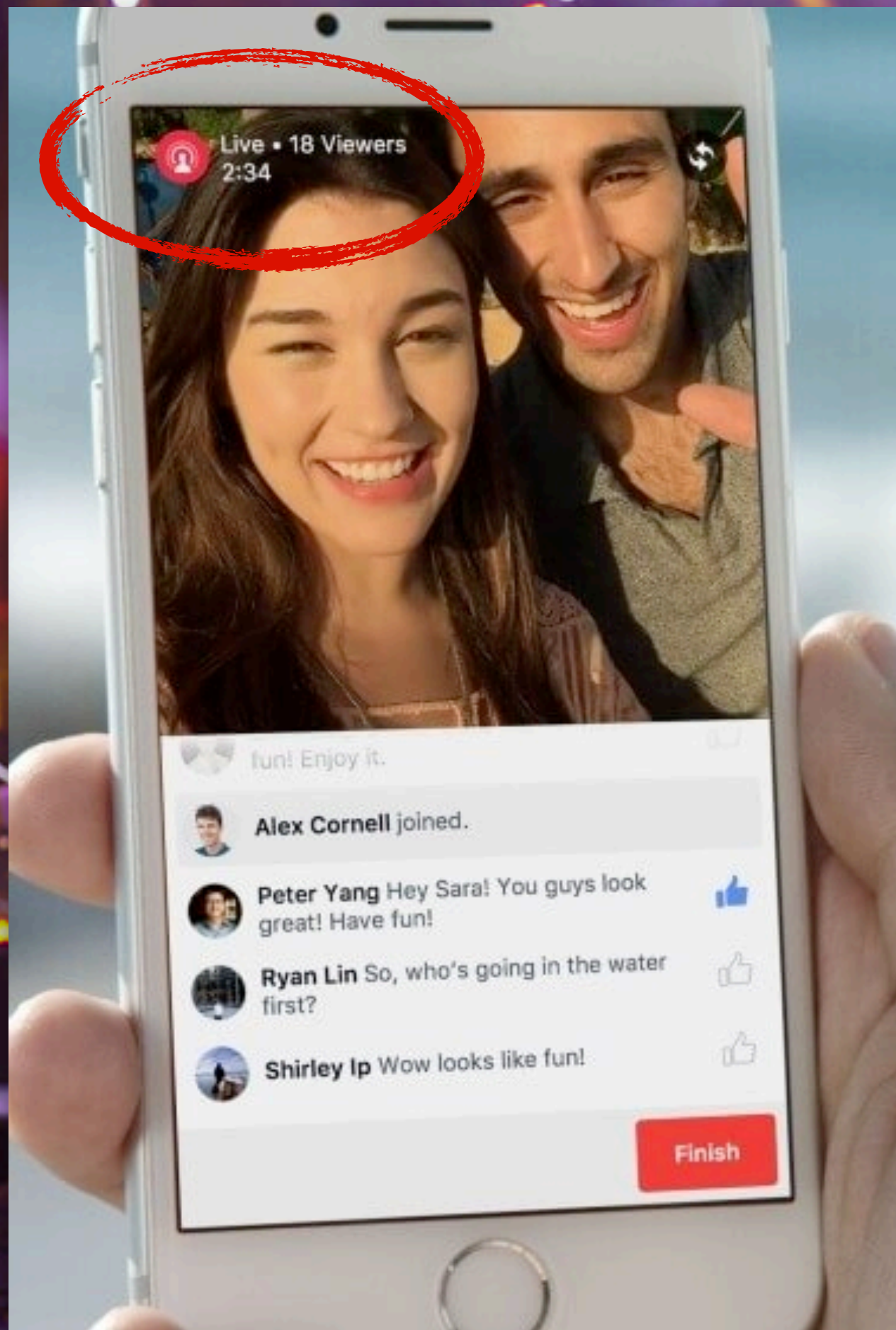
Otherwise, images/videos you get are compressed in the color space of the output device, mostly sRGB.





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!





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.

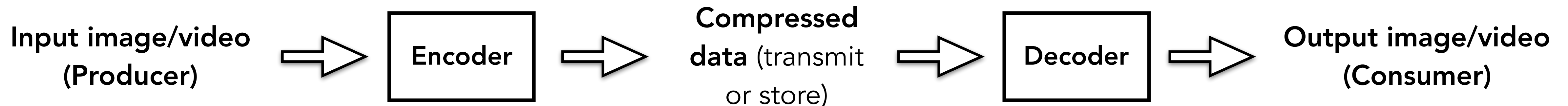


Image and Video Compression in Hardware

Nvidia
Xavier SoC

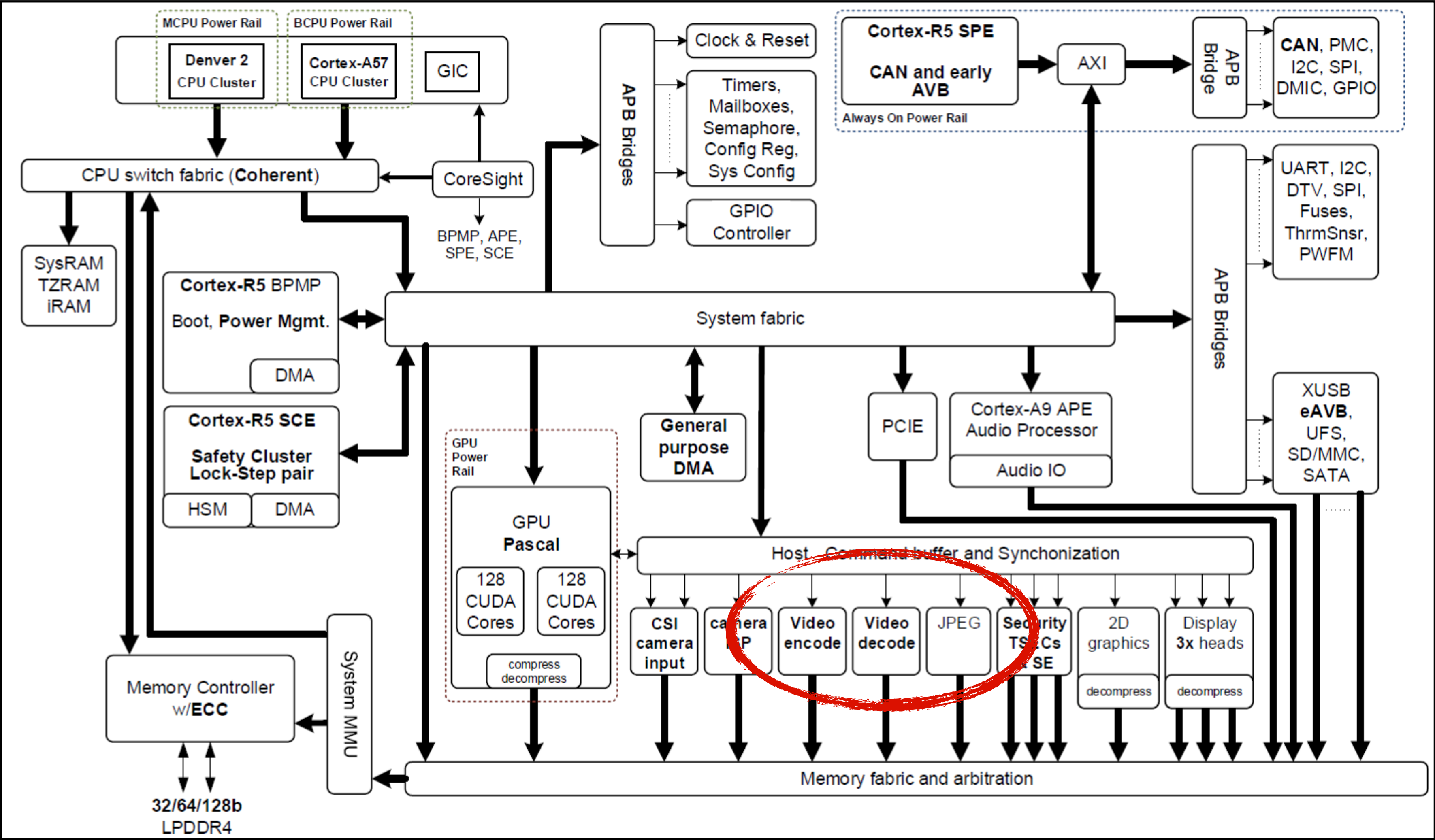
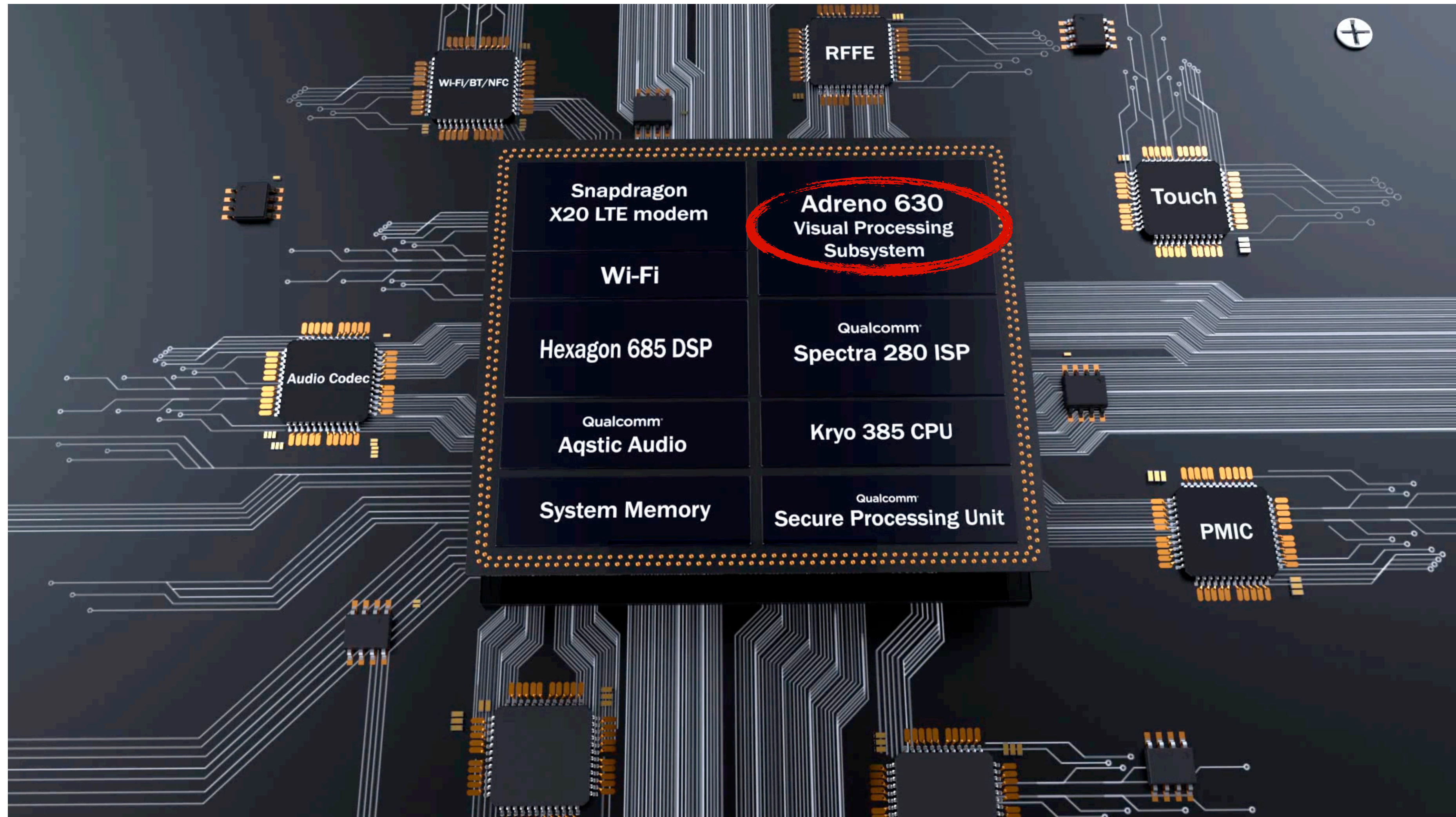


Image and Video Compression in Hardware

Qualcomm
Snapdragon
845



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 chrominance.

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

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

- High-frequency information looks “busy”/“noisy” anyway, so a bit of inaccuracy/corruption in pixel values isn’t very noticeable.

In What Color Space We Compress?

We are more sensitive to brightness differences than to chromatic differences; we can afford to lose data in chrominance, but not so much in luminance.

First transform RGB into a color space that separates **luminance** and **chrominance**, and then subsample the chrominance.

There are many such color spaces, but they all consist of a **luma dimension** and **two chroma dimensions**.

- Similar to HSV, but better justified by human visual perception
- They (mostly) all start from a gamma-encoded sRGB color space.

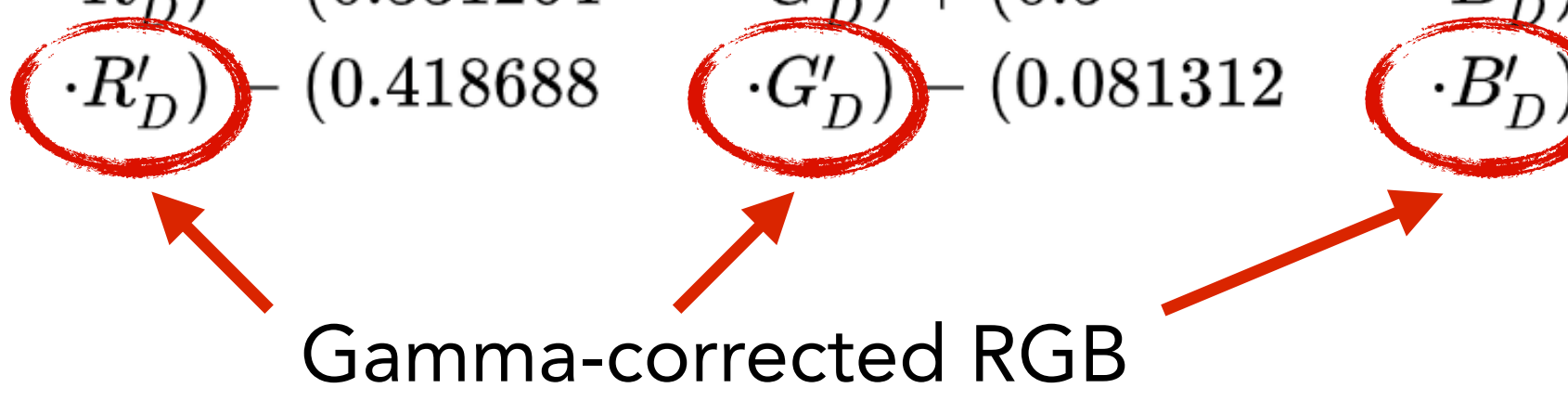
JPEG's Y'CbCr Color Space

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

JPEG's Y'CbCr Color Space

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

Gamma-corrected RGB
values within [0, 255]



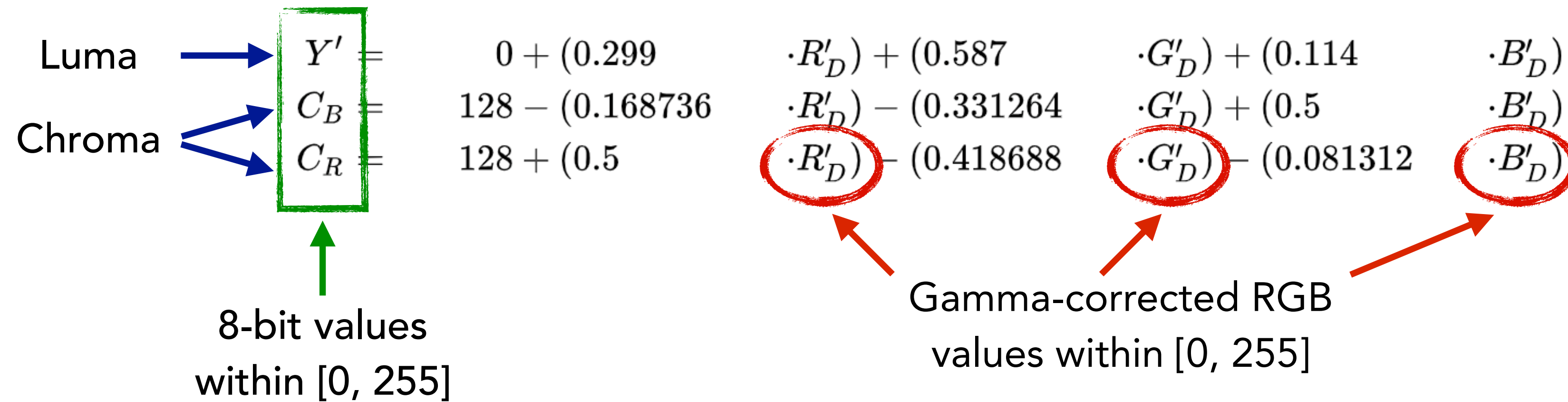
JPEG's Y'CbCr Color Space

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

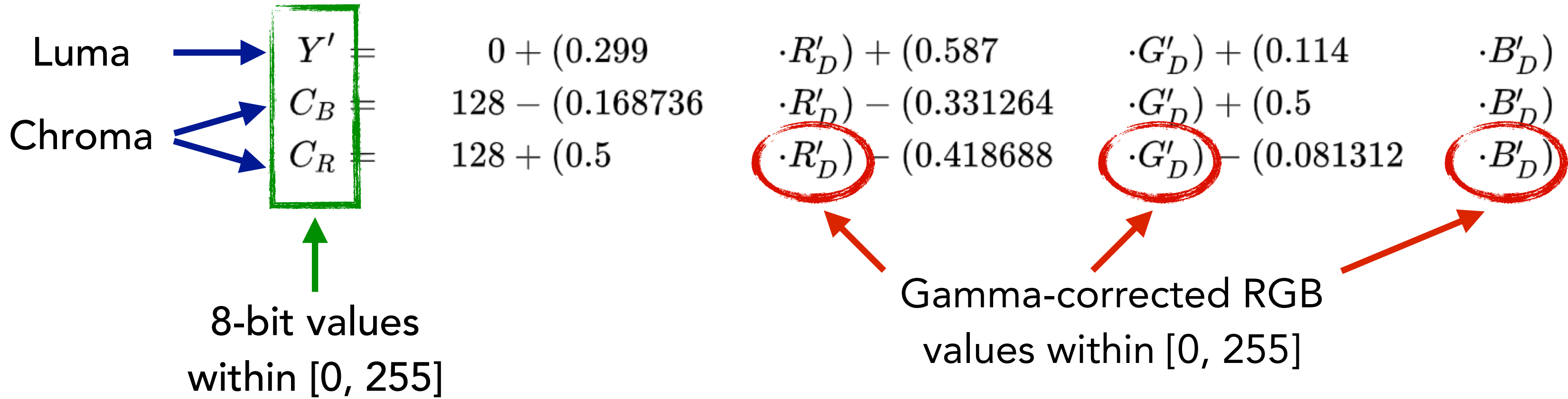
8-bit values within [0, 255]

Gamma-corrected RGB values within [0, 255]

JPEG's Y'CbCr Color Space



JPEG's Y'CbCr Color Space



RGB



Y'



Cb



Cr



Chroma Subsampling (Lossy)

Downsample the two chroma components, since human eyes are less sensitive to loss of information to chrominance than to luminance.

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

Y'	Y'	Y'	Y'
Cb	Cb	Cb	Cb
Cr	Cr	Cr	Cr
Y'	Y'	Y'	Y'
Cb	Cb	Cb	Cb
Cr	Cr	Cr	Cr

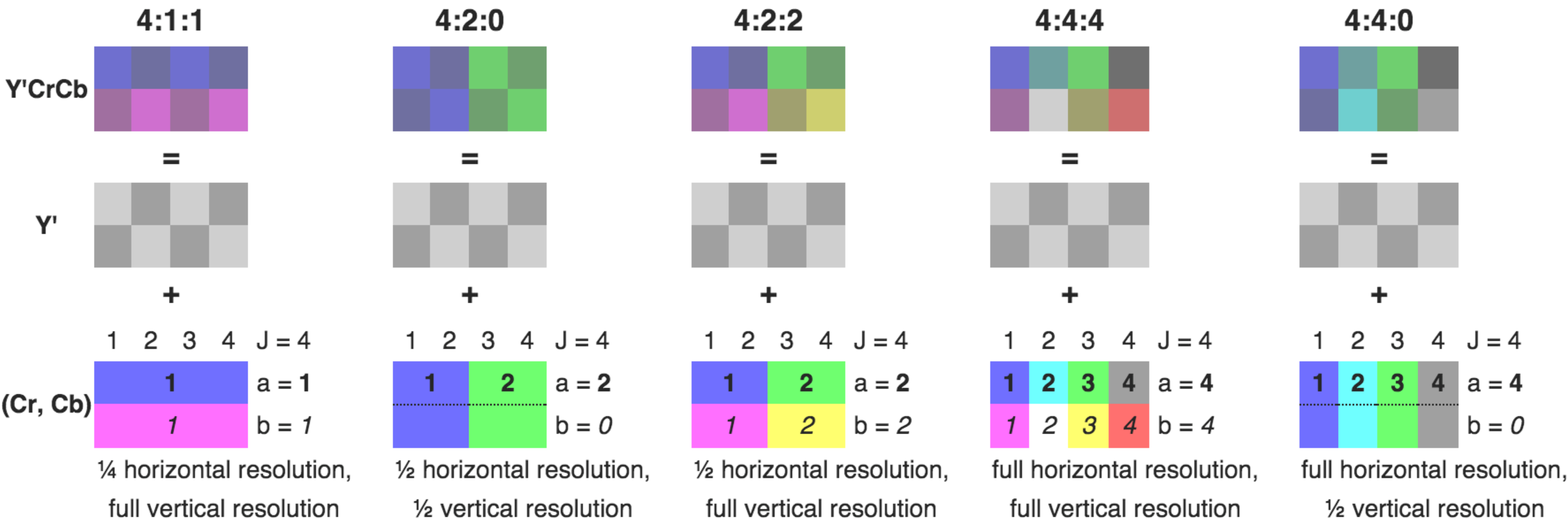
4:2:2 representation (Cb and Cr are subsampled 2X)

Y'	Y'	Y'	Y'
Cb		Cb	
Cr		Cr	
Y'	Y'	Y'	Y'
Cb		Cb	
Cr		Cr	

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

Y'	Y'	Y'	Y'
Cb		Cb	
Cr		Cr	
Y'	Y'	Y'	Y'

Chroma Subsampling (Lossy)



Examples



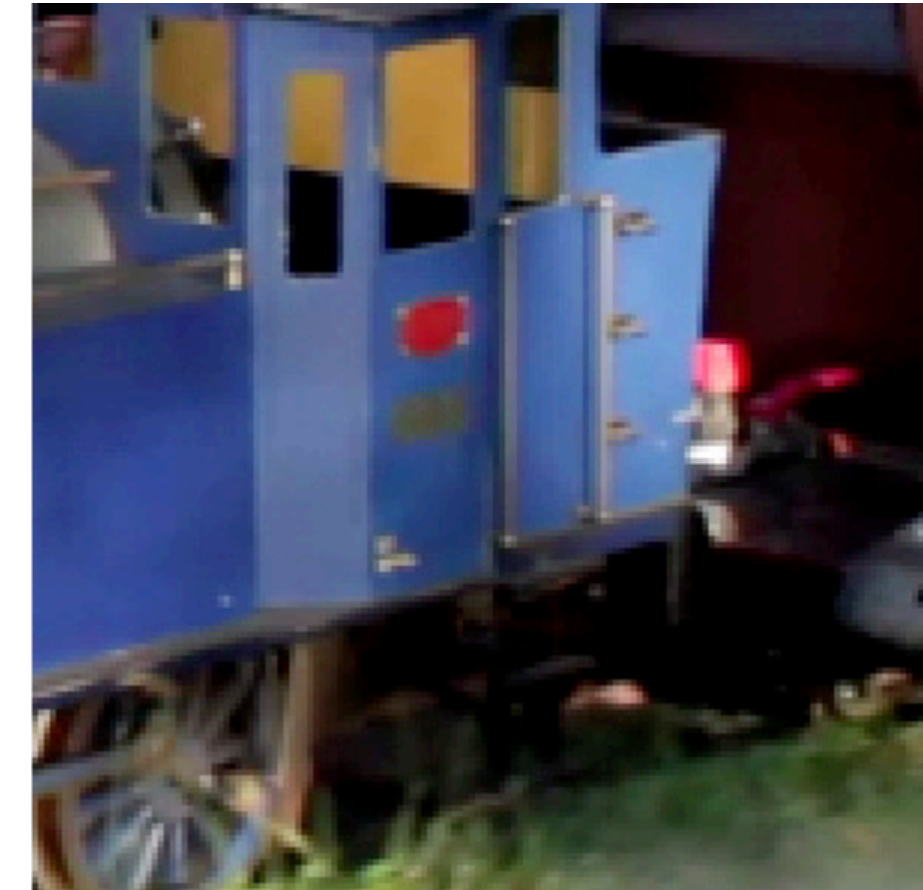
4:1:1



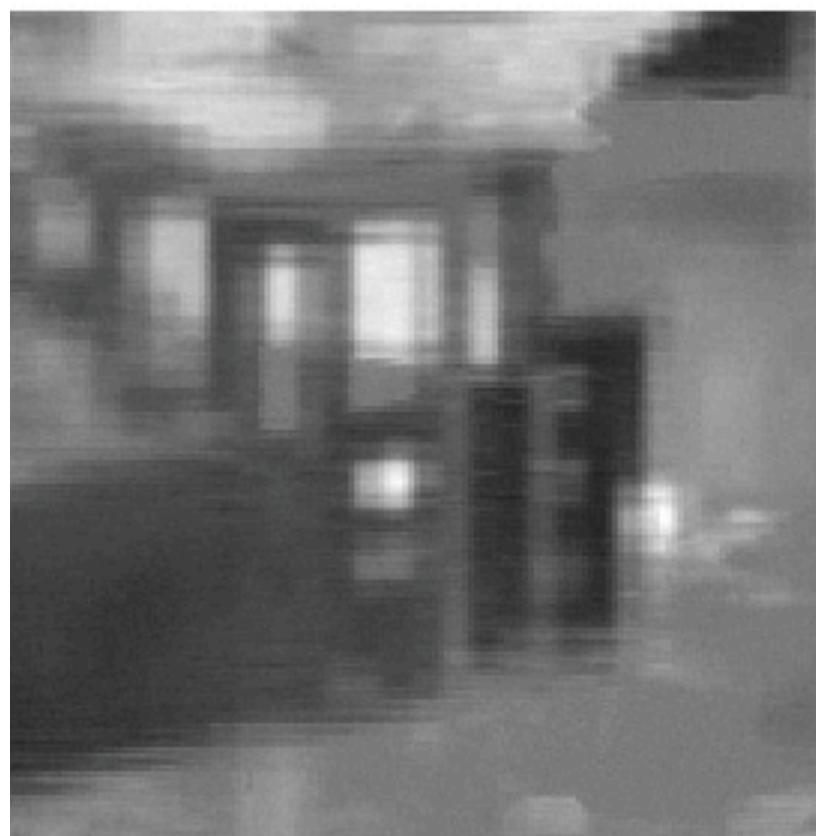
4:2:0



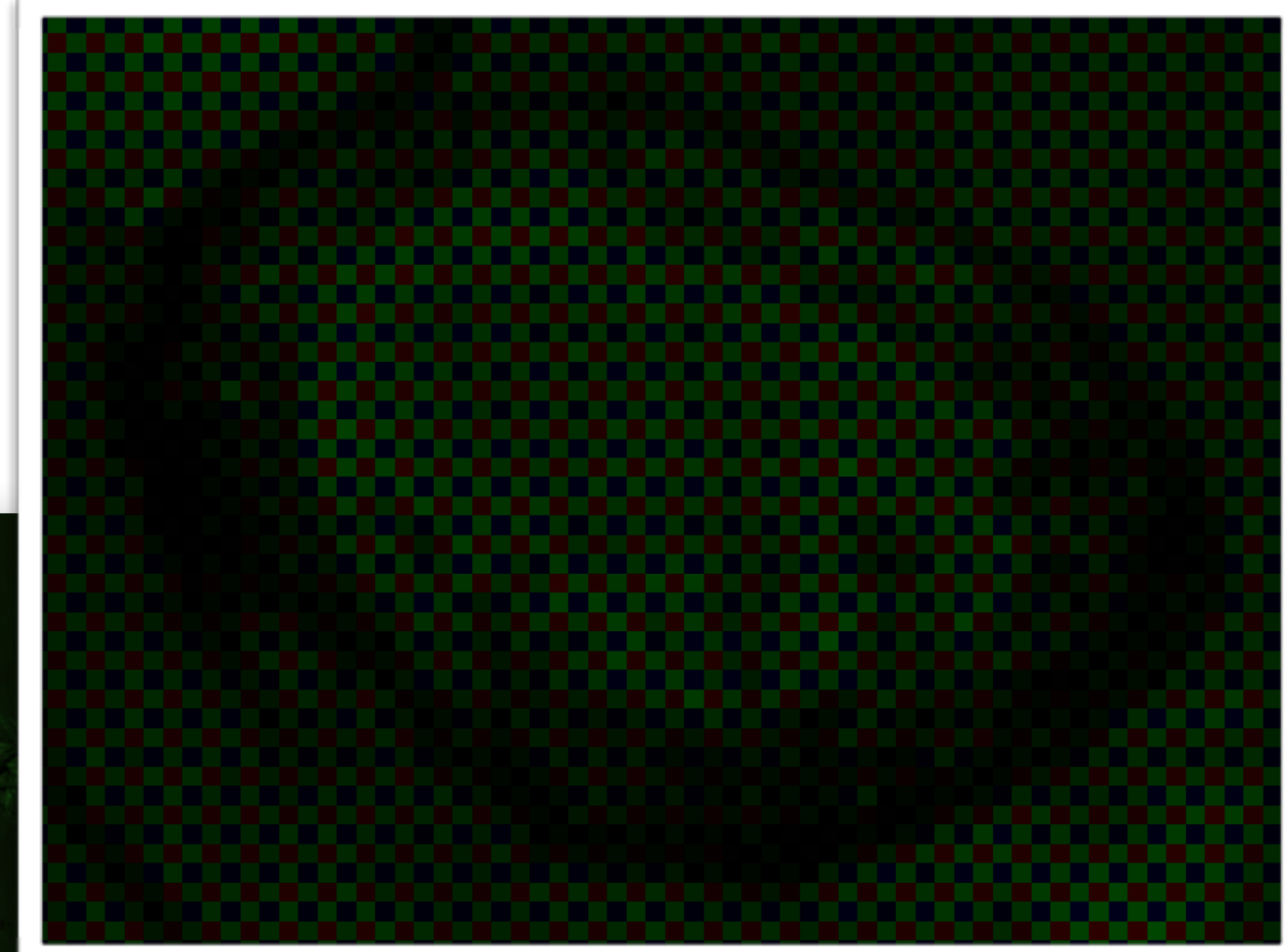
4:2:2



4:4:4



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!

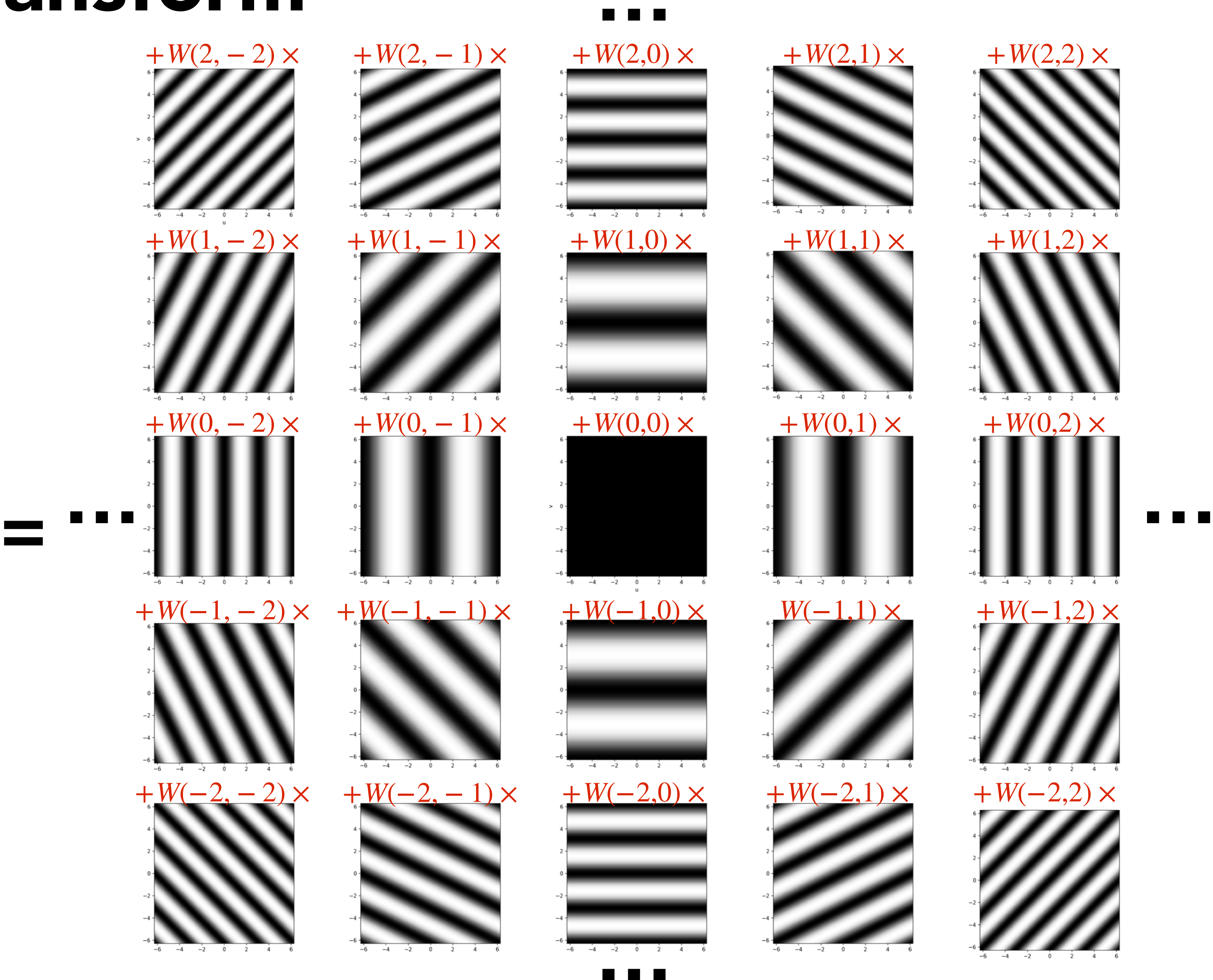
High-Frequency Information Compression

High-frequency information looks “busy”/“noisy”, so a bit of inaccuracy/corruption in pixel values isn’t very noticeable.

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.

2D Discrete Fourier Transform

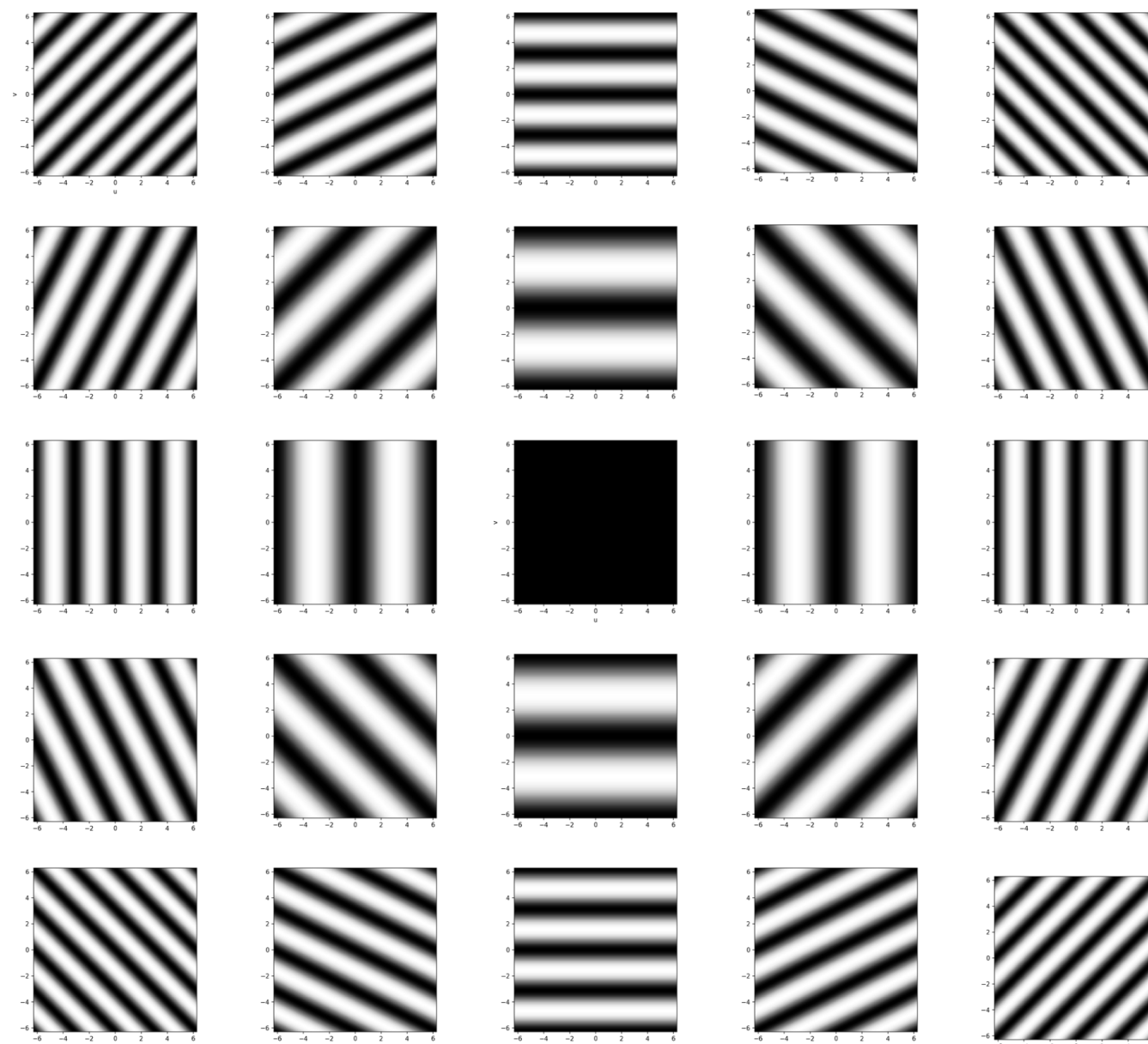


High-Frequency Information Compression

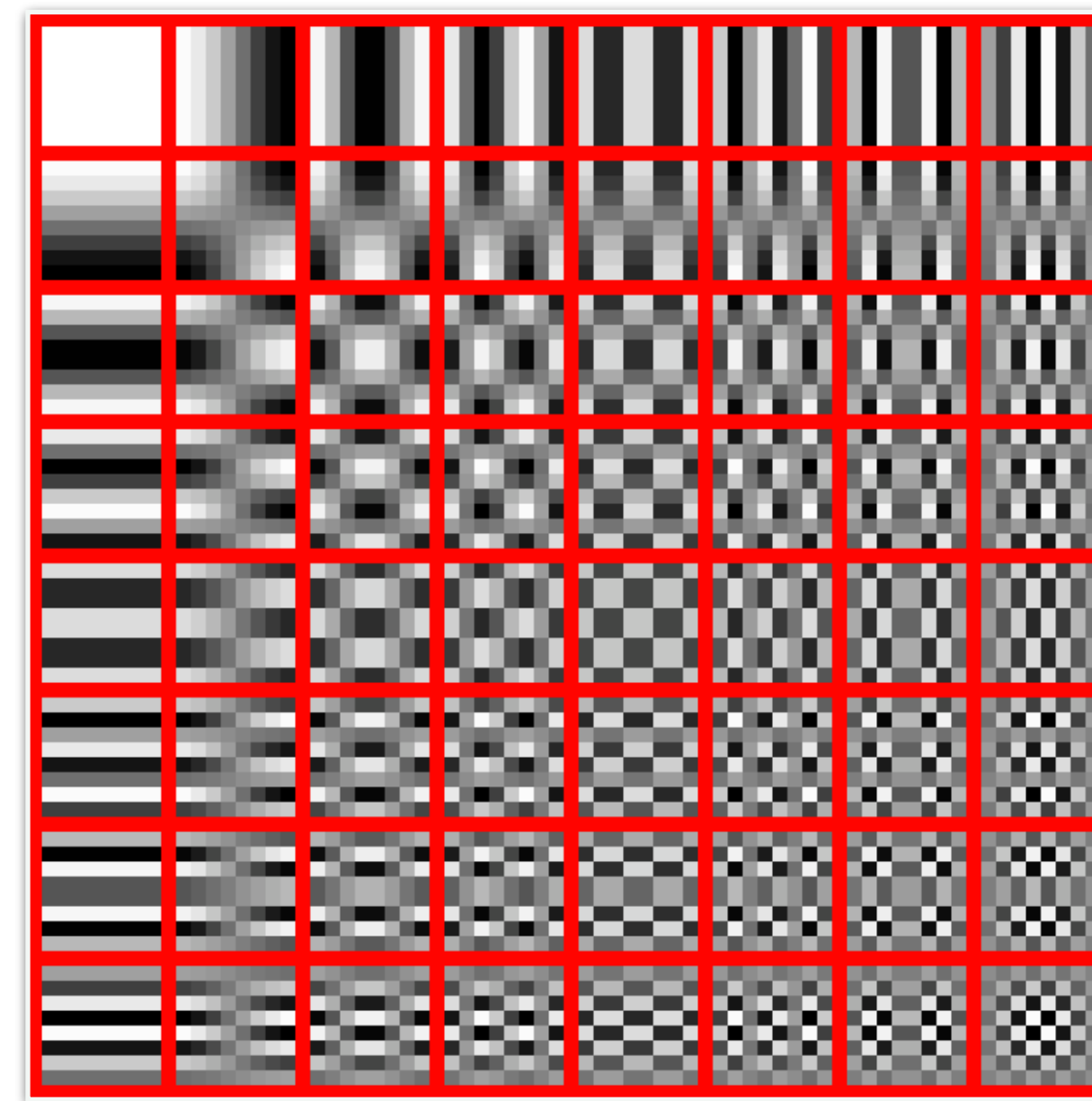
JPEG uses **Discrete Cosine Transformation** (DCT), just a different set of basis functions than what Discrete Fourier Transform uses.

- 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)

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


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



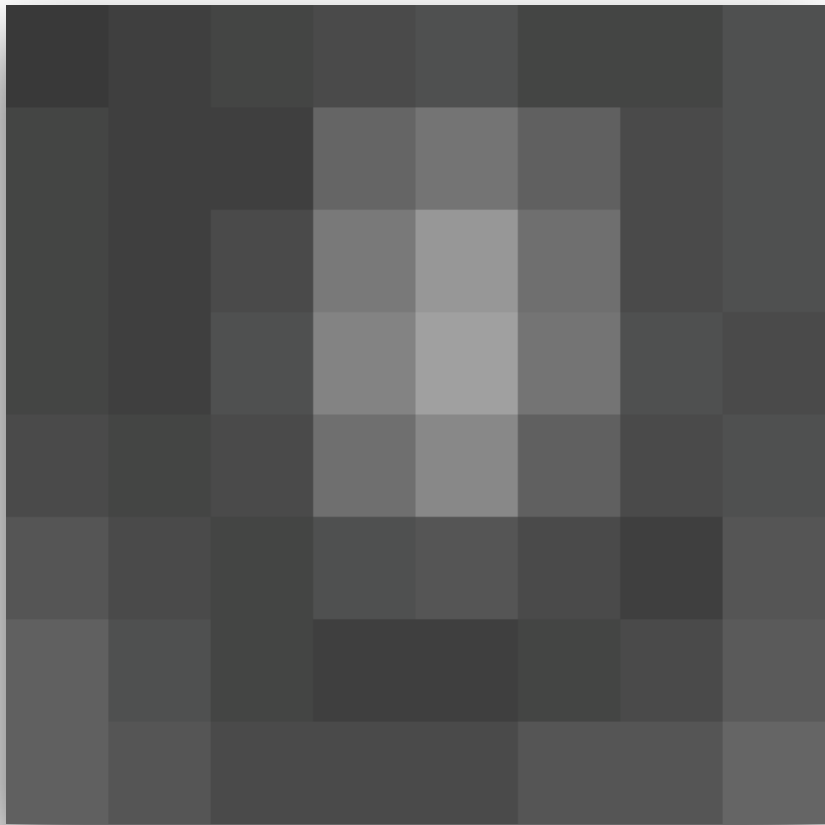
DCT (Lossless)

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

Zero-centered


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

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



DCT (Lossless)

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

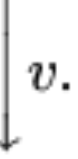
Zero-centered



x							
\longrightarrow							
-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

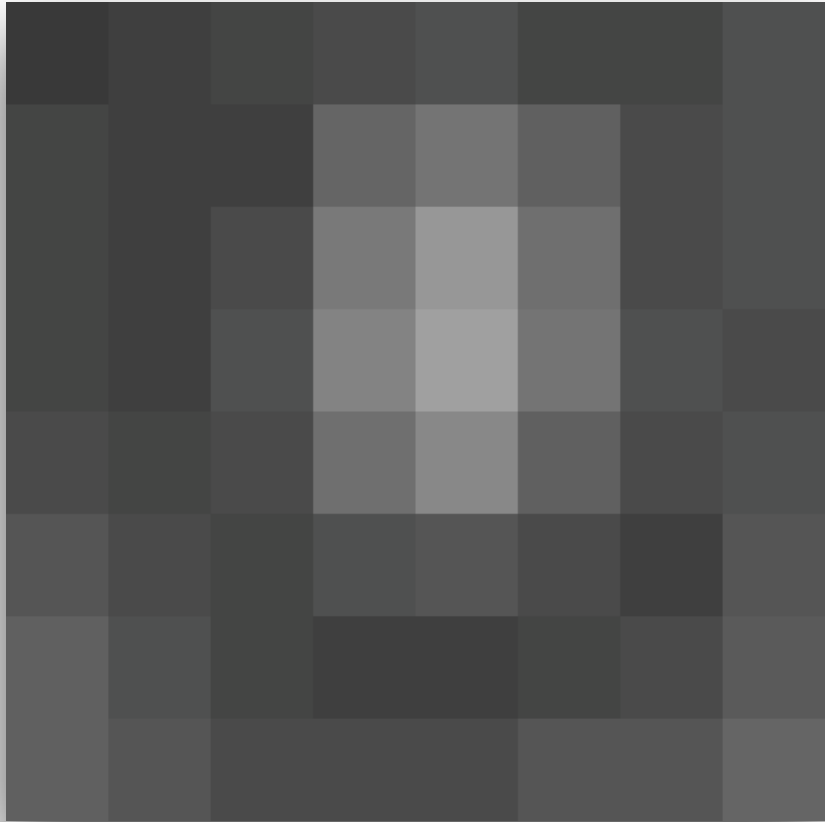


u							
\longrightarrow							
-415.38	-30.19	-61.20	27.24	56.12	-20.10	-2.39	0.46
4.47	-21.86	-60.76	10.25	13.15	-7.09	-8.54	4.88
-46.83	7.37	77.13	-24.56	-28.91	9.93	5.42	-5.65
-48.53	12.07	34.10	-14.76	-10.24	6.30	1.83	1.95
12.12	-6.55	-13.20	-3.95	-1.87	1.75	-2.79	3.14
-7.73	2.91	2.38	-5.94	-2.38	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



Weights (coefficients)

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



DCT (Lossless)

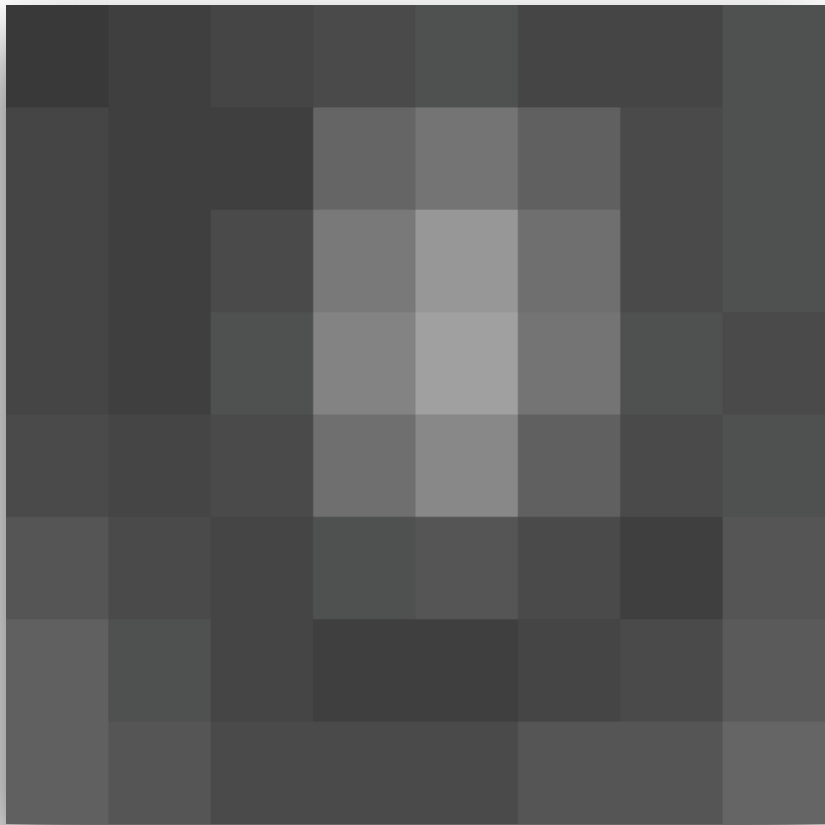
$$G_{u,v} = \frac{1}{4} \alpha(u) \alpha(v) \sum_{x=0}^7 \sum_{y=0}^7 g_{x,y} \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

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

Zero-centered

x							
\longrightarrow							
-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
y							
\downarrow							

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



u							
\longrightarrow							
-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
v							
\downarrow							

Weights (coefficients)

DCT (Lossless)

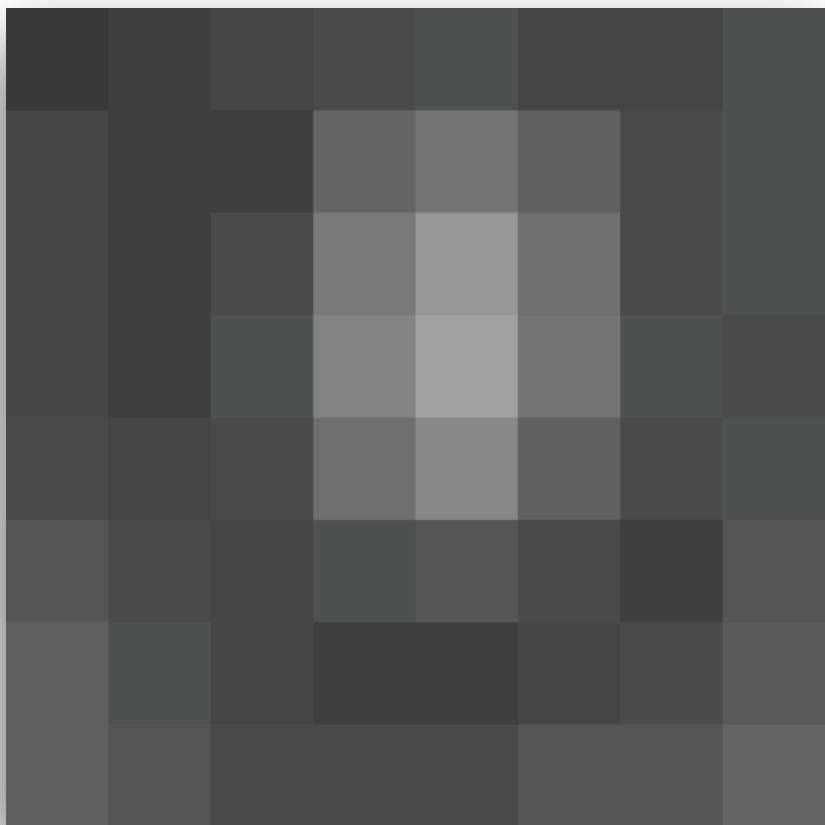
$$G_{u,v} = \frac{1}{4} \alpha(u) \alpha(v) \sum_{x=0}^7 \sum_{y=0}^7 g_{x,y} \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

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

Zero-centered

x							
\longrightarrow							
-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
				y			
				\downarrow			

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



u							
\longrightarrow							
-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
				v			
				\downarrow			

Weights (coefficients)

DCT (Lossless)

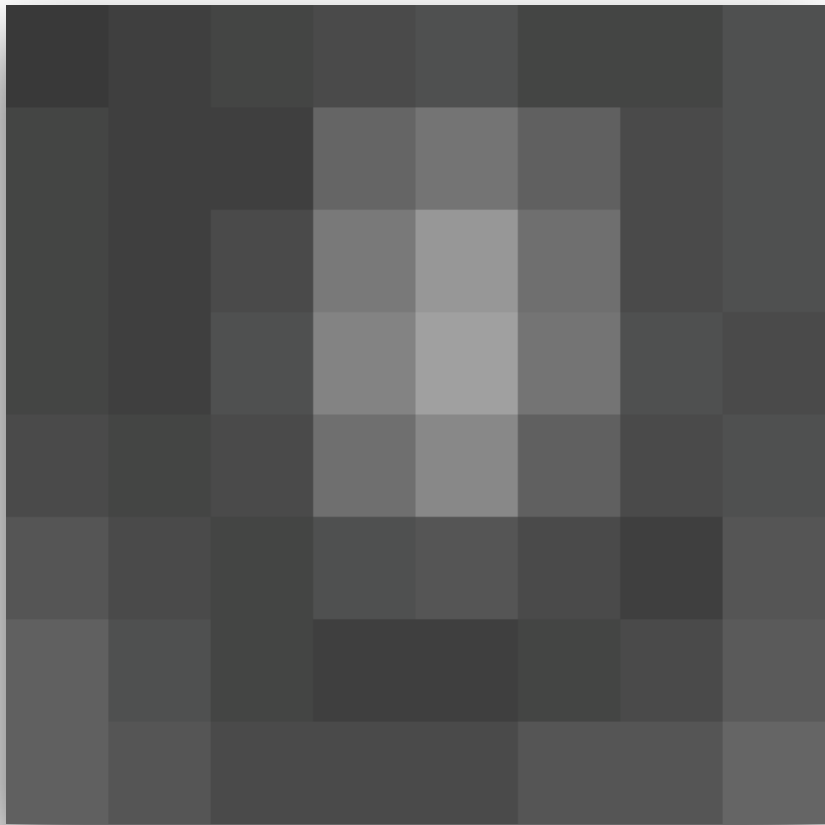
$$G_{u,v} = \frac{1}{4} \alpha(u) \alpha(v) \sum_{x=0}^7 \sum_{y=0}^7 g_{x,y} \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

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

Zero-centered

x							
\rightarrow							
-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
				y			
				\downarrow			

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



u							
\rightarrow							
$G_{0,0}$ -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	$G_{7,7}$ 1.68
				v			
				\downarrow			

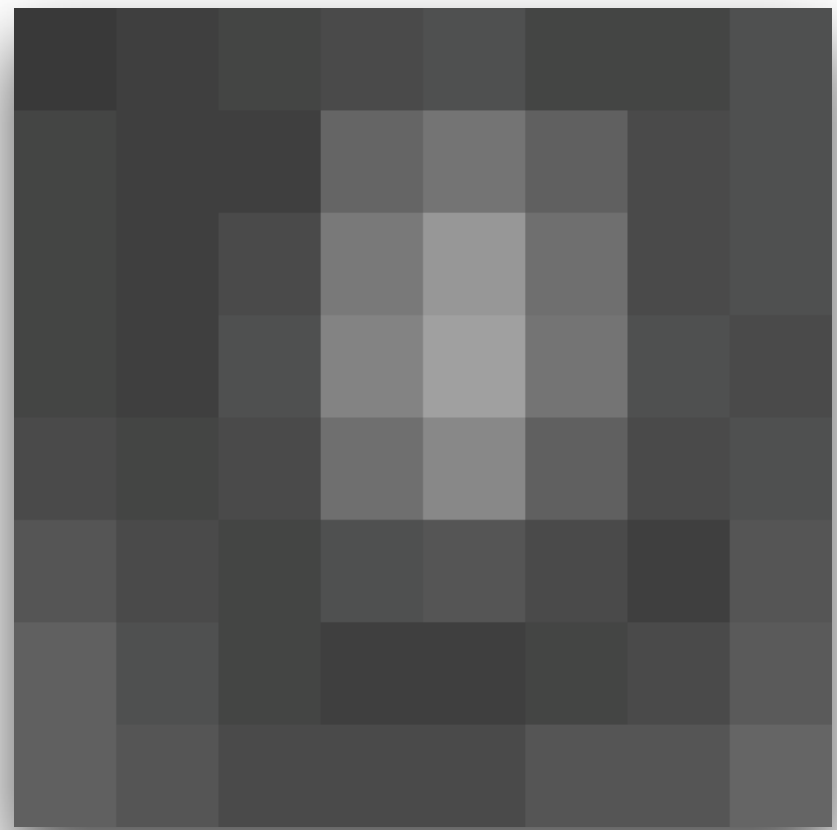
Weights (coefficients)

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.

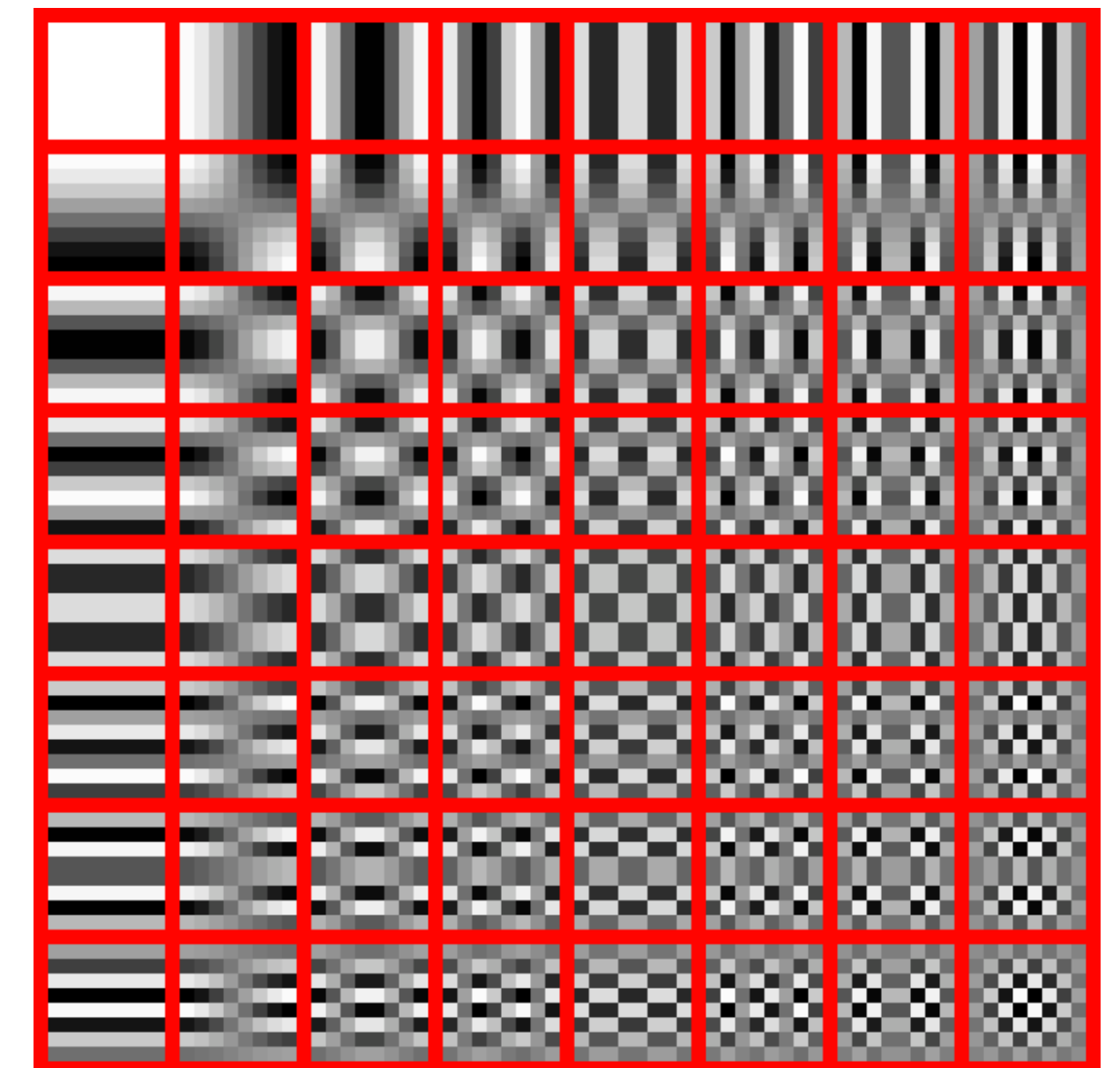
An 8x8 block.



Weights (coefficients)

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

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



Coefficient Quantization (Lossy)

Weights (coefficients)

$$G = \begin{matrix} & \begin{matrix} u \\ \longrightarrow \end{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{matrix} \\ \begin{matrix} \downarrow v. \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \end{matrix}$$

Coefficient Quantization (Lossy)

Weights (coefficients)

u
→

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

Quantization matrix (fixed, no need to encode)

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

Coefficient Quantization (Lossy)

Weights (coefficients)

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

Quantization matrix (fixed, no need to encode)

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

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

Quantized coefficients

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

Coefficient Quantization (Lossy)

Weights (coefficients)

$G =$

	$u \rightarrow$	

$\downarrow v.$

Quantization matrix (fixed, no need to encode)

$Q =$

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

Quantized coefficients

$B =$

Coefficient Quantization (Lossy)

Weights (coefficients)

$G =$

	$u \rightarrow$	

$\downarrow v.$

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)

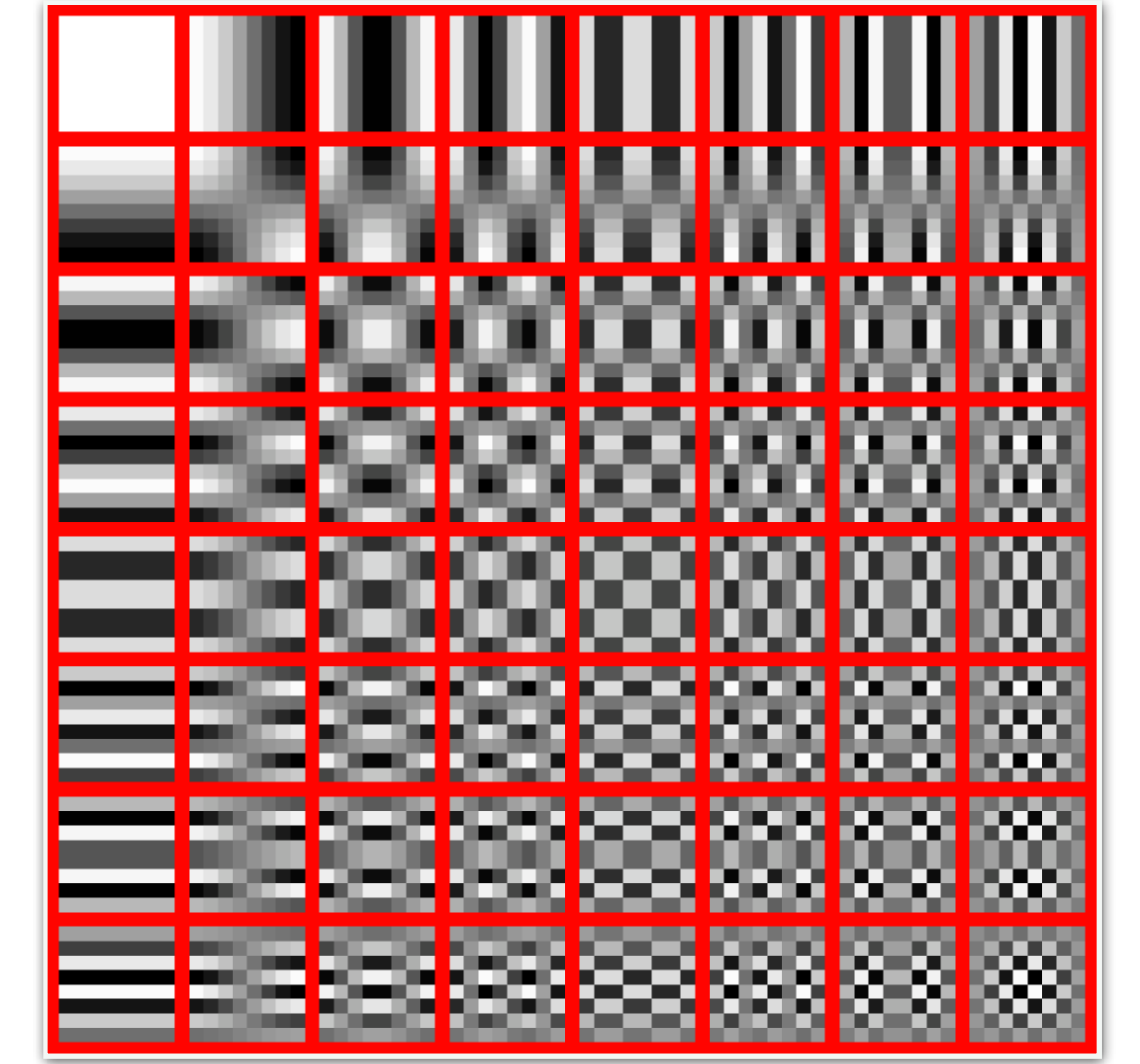
$Q =$

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

Quantized coefficients

$B =$

Quantization Matrix

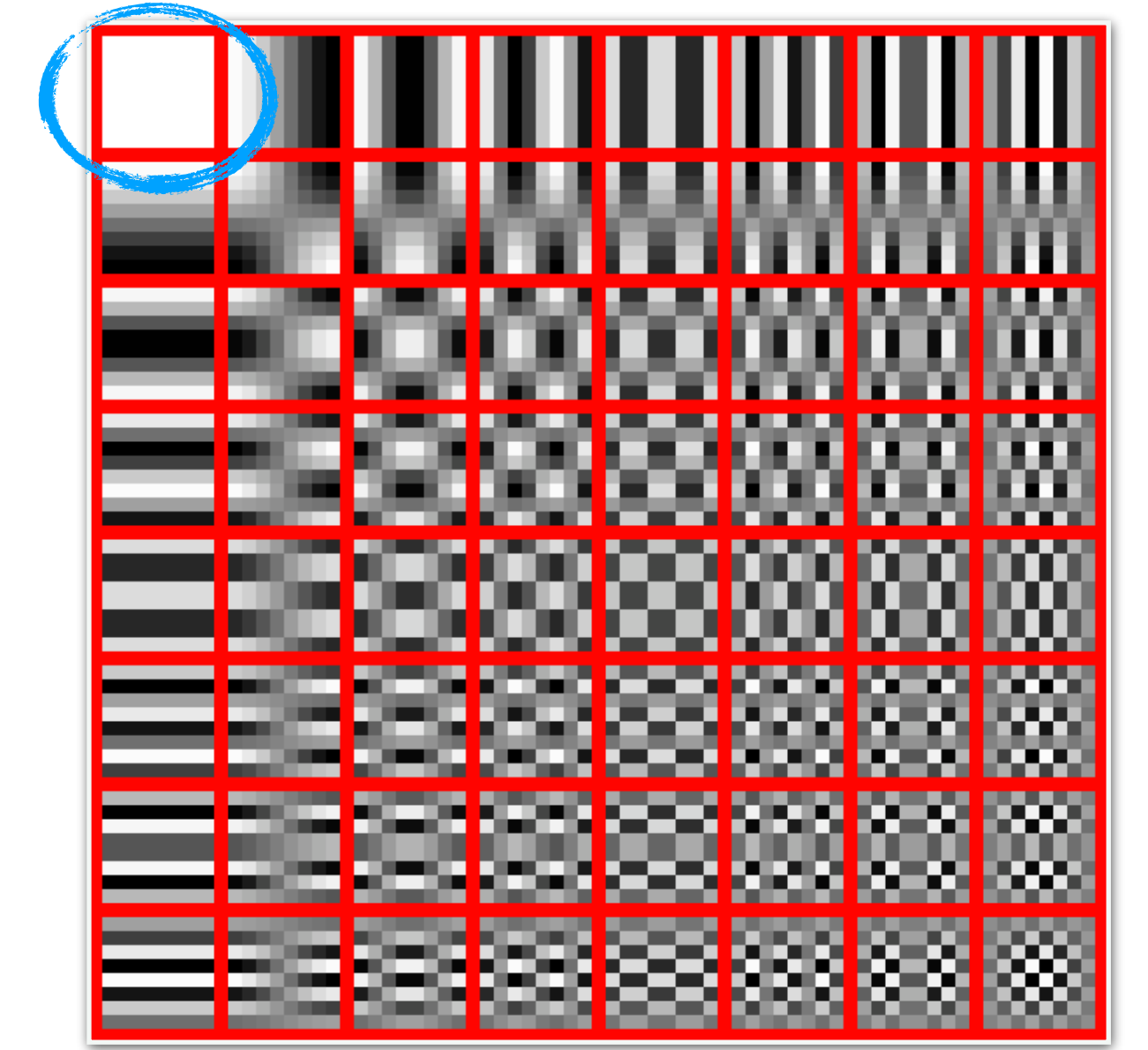


Quantization matrix

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

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

Quantization Matrix

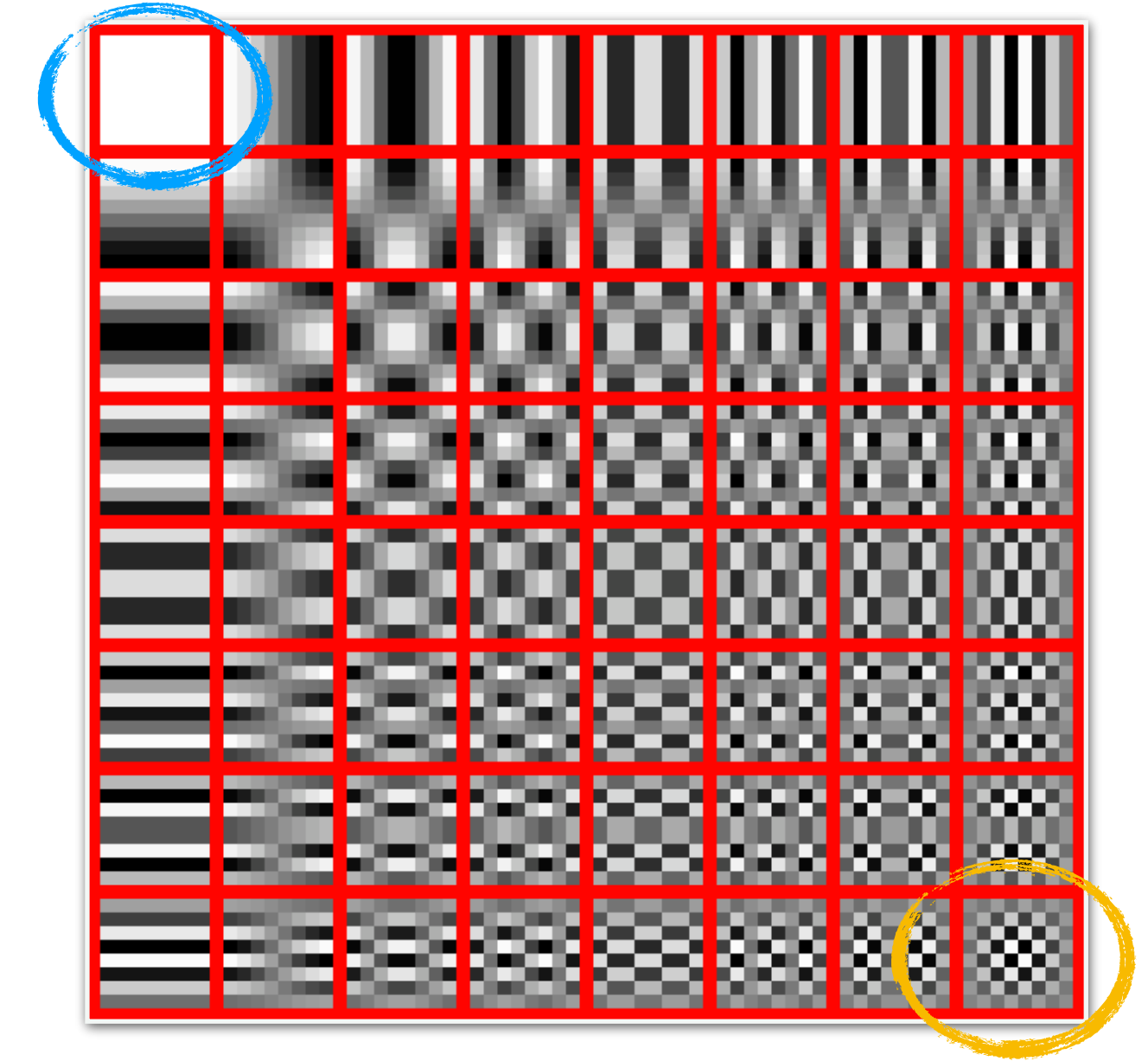


Quantization matrix

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

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

Quantization Matrix

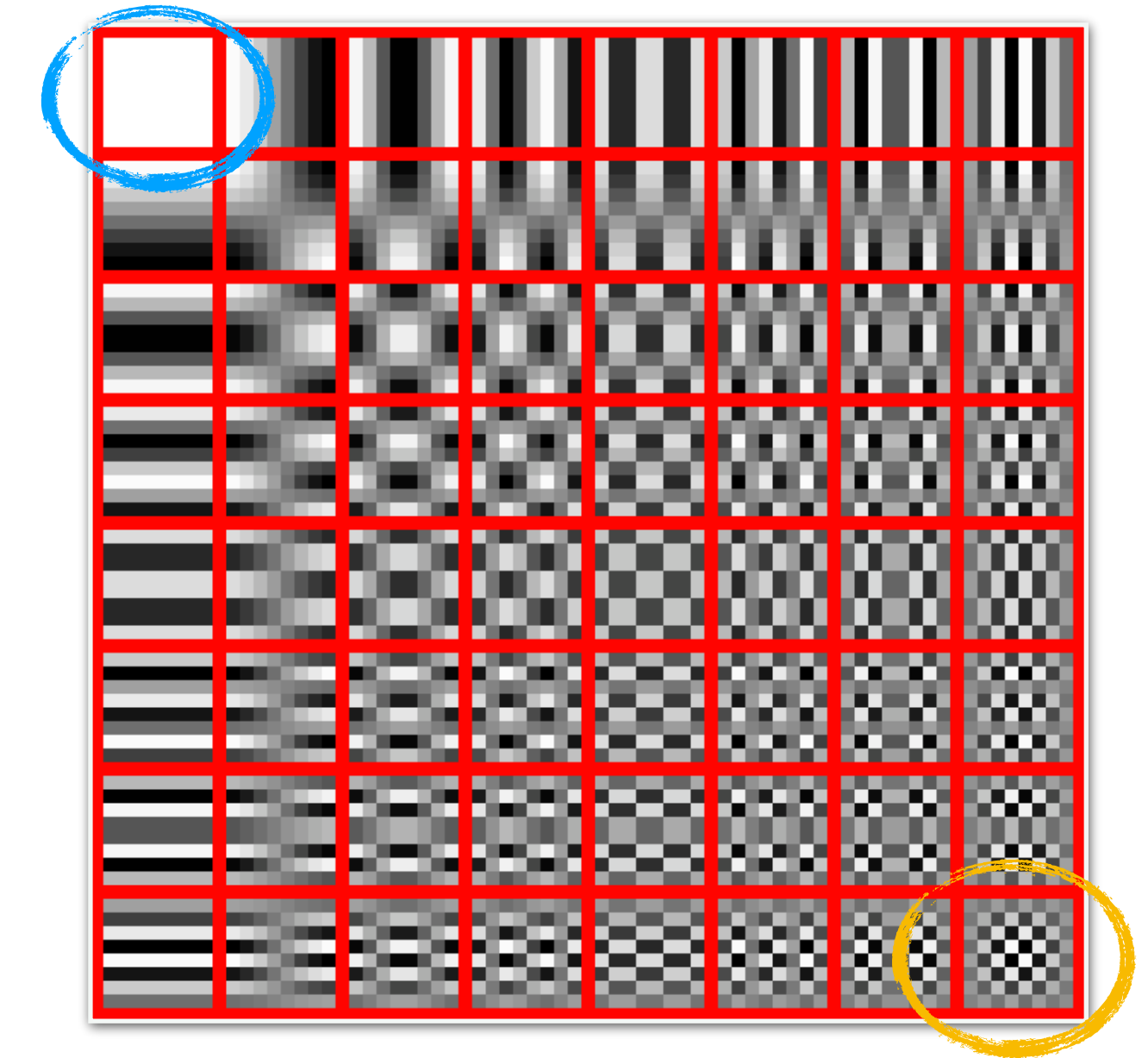


Quantization matrix

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

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

Quantization Matrix



Quantization matrix

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

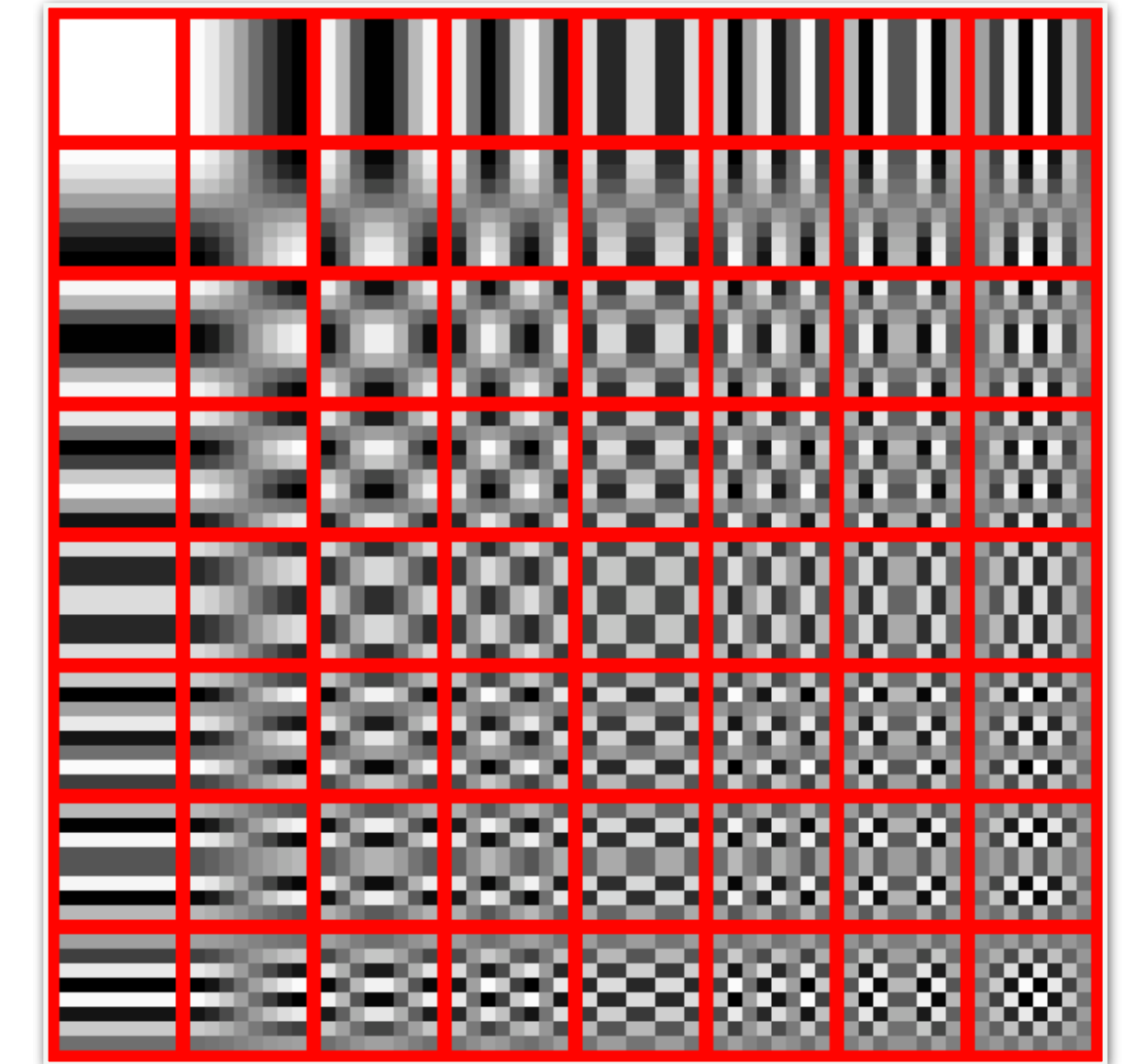
Low-frequency weights are smaller, and high-frequency weights are larger. As a result, quantization zeros out most of the high-frequency coefficients — a good thing!

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

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

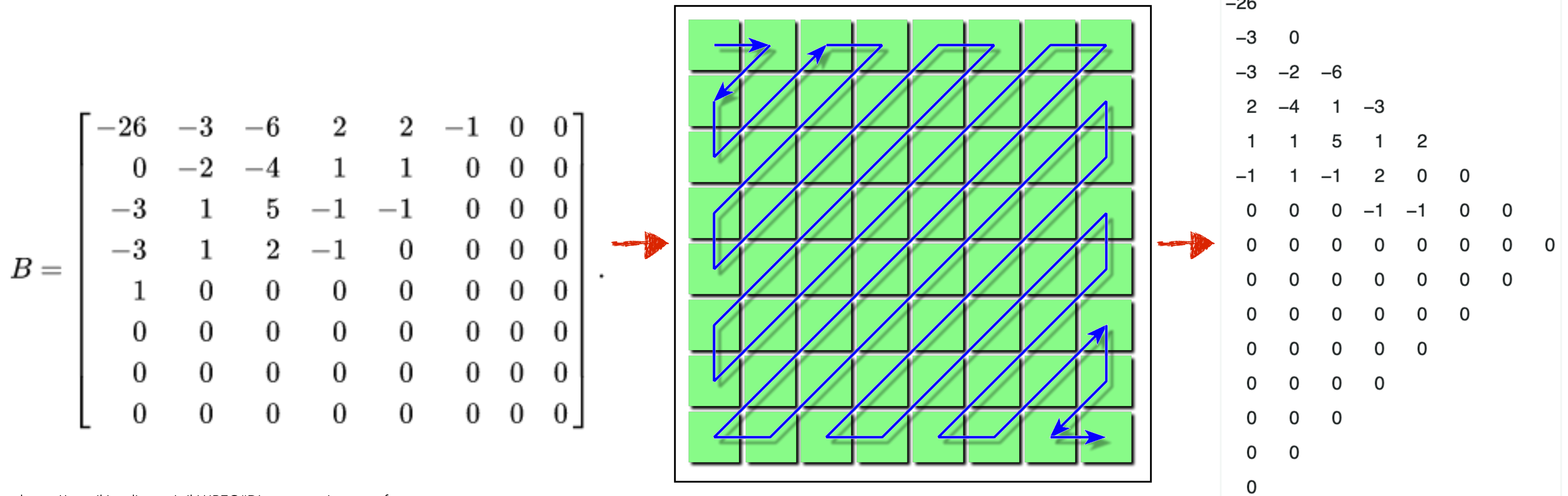
$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 25 & 36 & 49 & 62 \\ 12 & 12 & 14 & 20 & 33 & 44 & 58 & 72 \\ 14 & 13 & 16 & 22 & 36 & 49 & 64 & 80 \\ 14 & 17 & 22 & 29 & 44 & 60 & 77 & 94 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

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!

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

Entropy Encoding (Lossless)

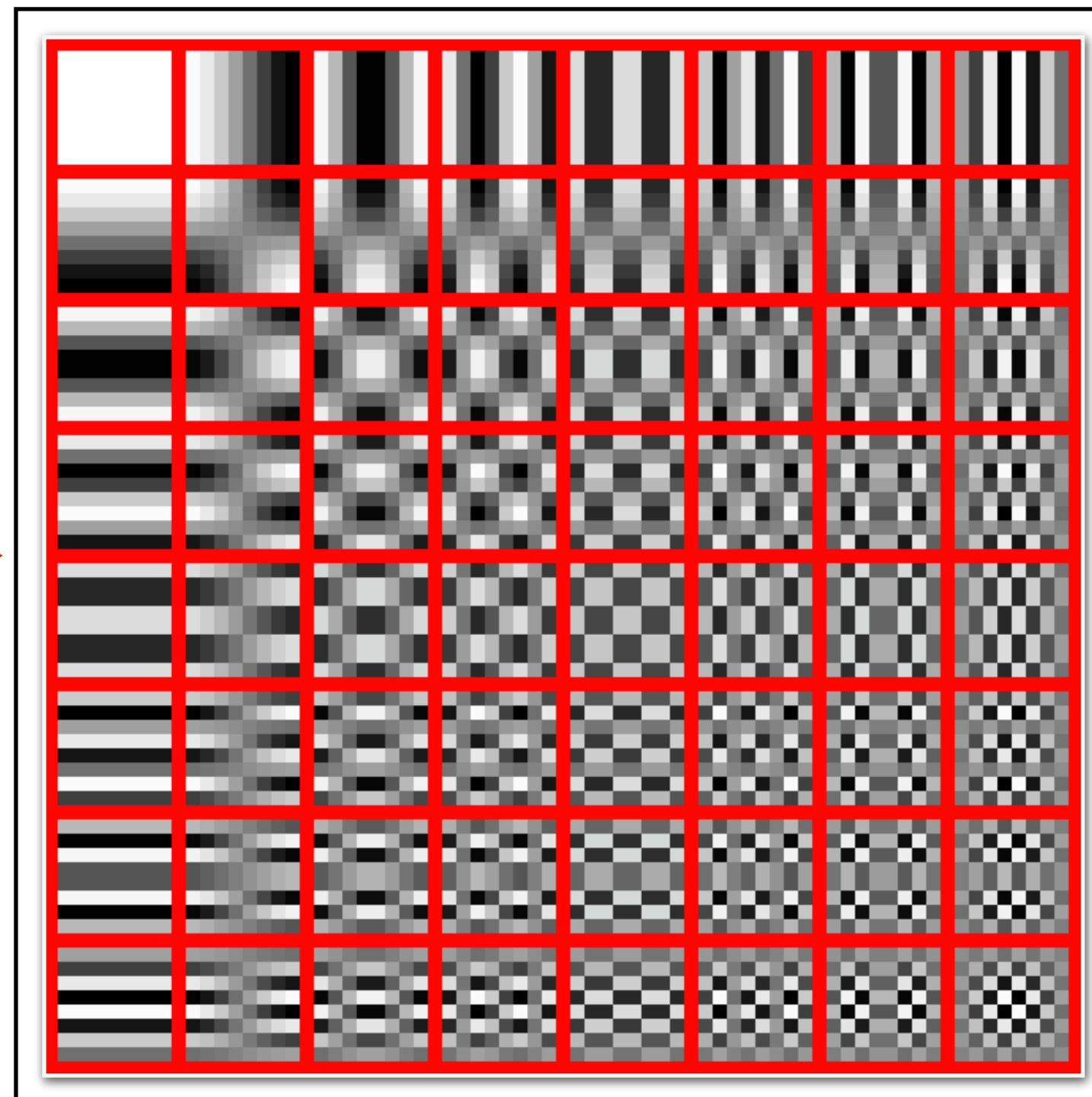
1. Reorder quantized coefficients in zig-zag order, which orders the frequencies in the ascending order.
 - Why zig-zag? Look at the pattern in the basis function.



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	0	0	0						
0	0	0							
0	0								
0									
0									




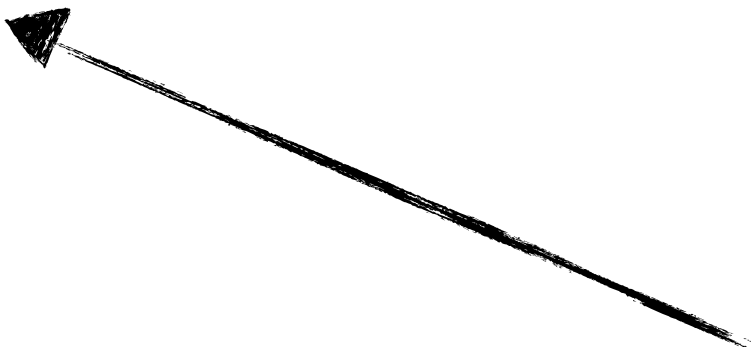
Entropy Encoding (Lossless)

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.

[illegible]

JPEG Algorithm Recap

A combination of **lossless** and **lossy** techniques.

1. **Gamma encode RGB values.**  Evenly encode perceived brightness, not luminance. Lossy due to quantization errors, but it's not unique to JPEG compression.
2. **Convert image to Y'CbCr color space.** 
3. **Chroma subsampling.**  Human eyes are sensitive to luminance difference but not chrominance.
4. For each channel (Y', Cb, Cr) 
 - A. For each 8x8 pixel block
 1. **Compute DCT coefficients**
 2. **Quantize DCT coefficients**
 3. **Entropy encoding**

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

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

Compression Quality

Start seeing quantization
artifact (blocking)



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

Compression Quality

Start seeing quantization
artifact (blocking)



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

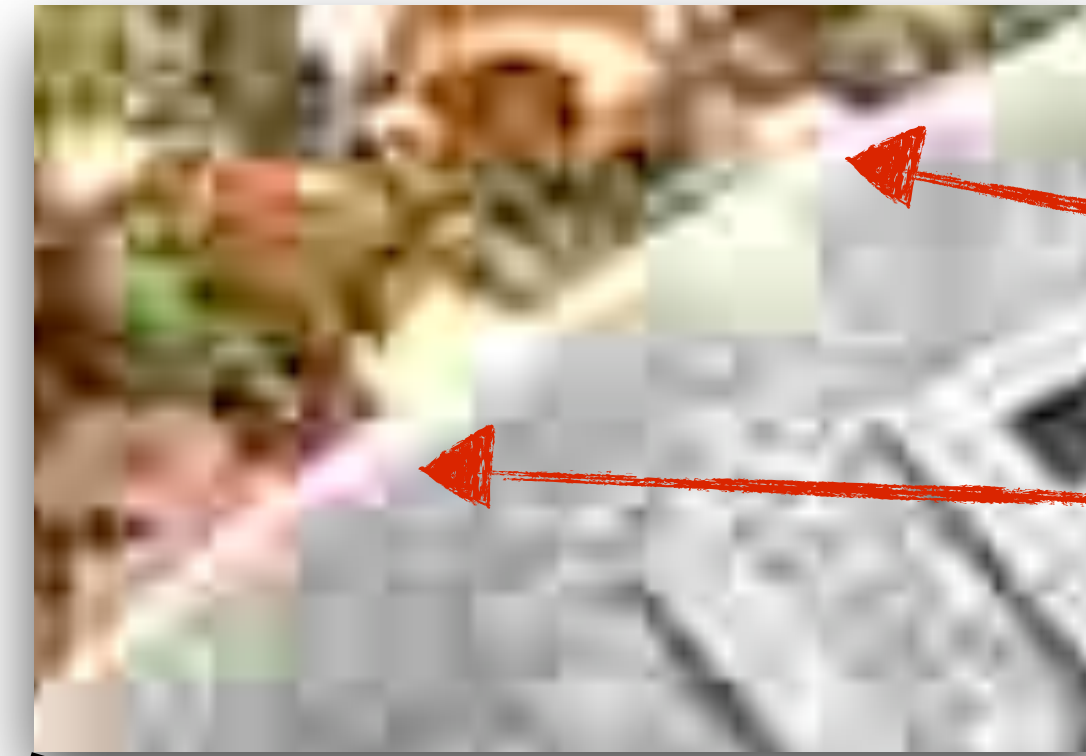
Compression Quality

Start seeing quantization
artifact (blocking)



Color is off (chroma
subsampling artifacts)

Severe blocking
artifacts



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

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.

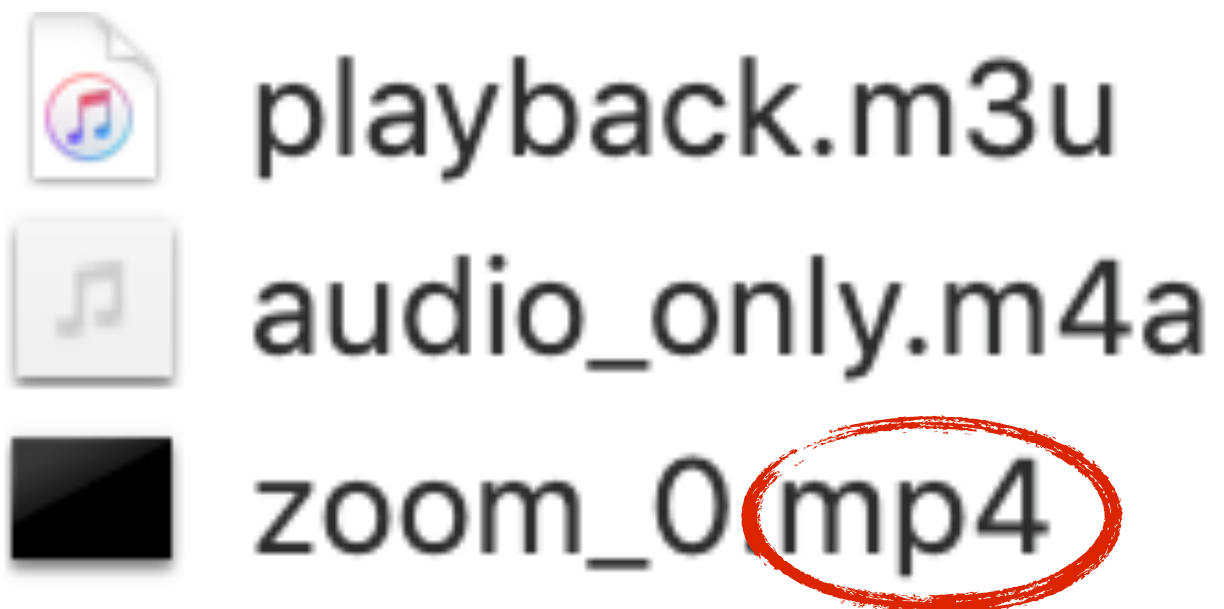
- Supports H.264 and H.265 video encoding and MPEG-4 Part 3 audio encoding.

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.

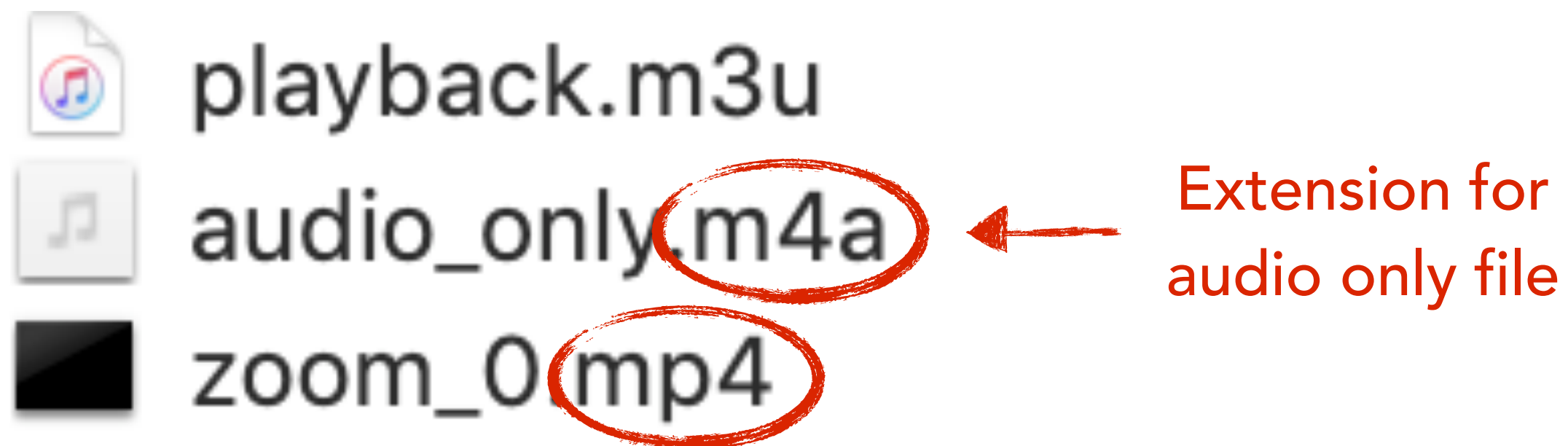


Container Format

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.

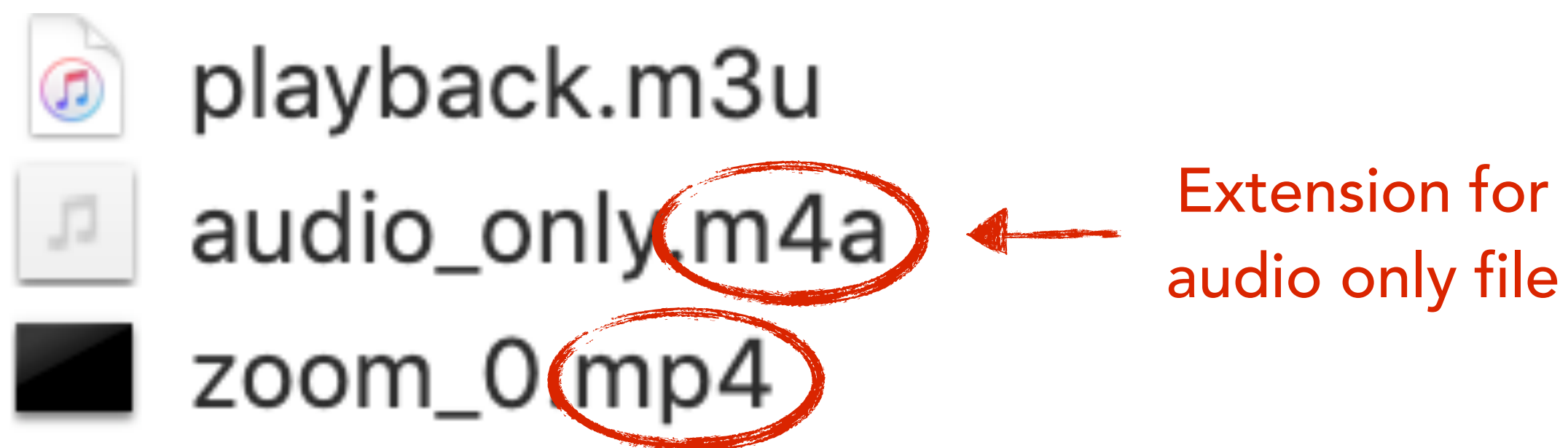


Container Format

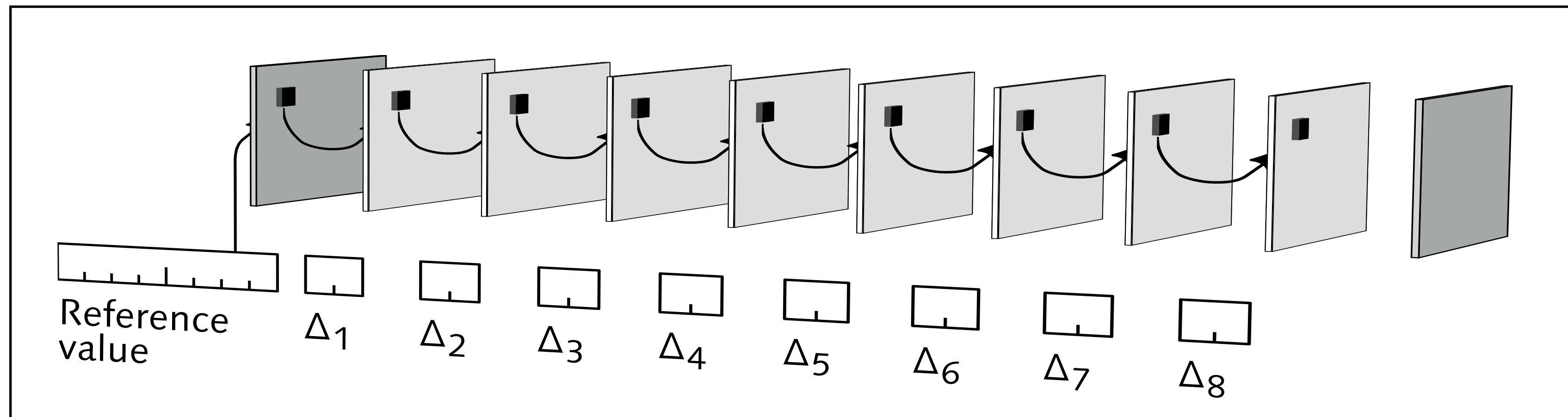
A container format bundles different data types, e.g., video, audio, subtitles.

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

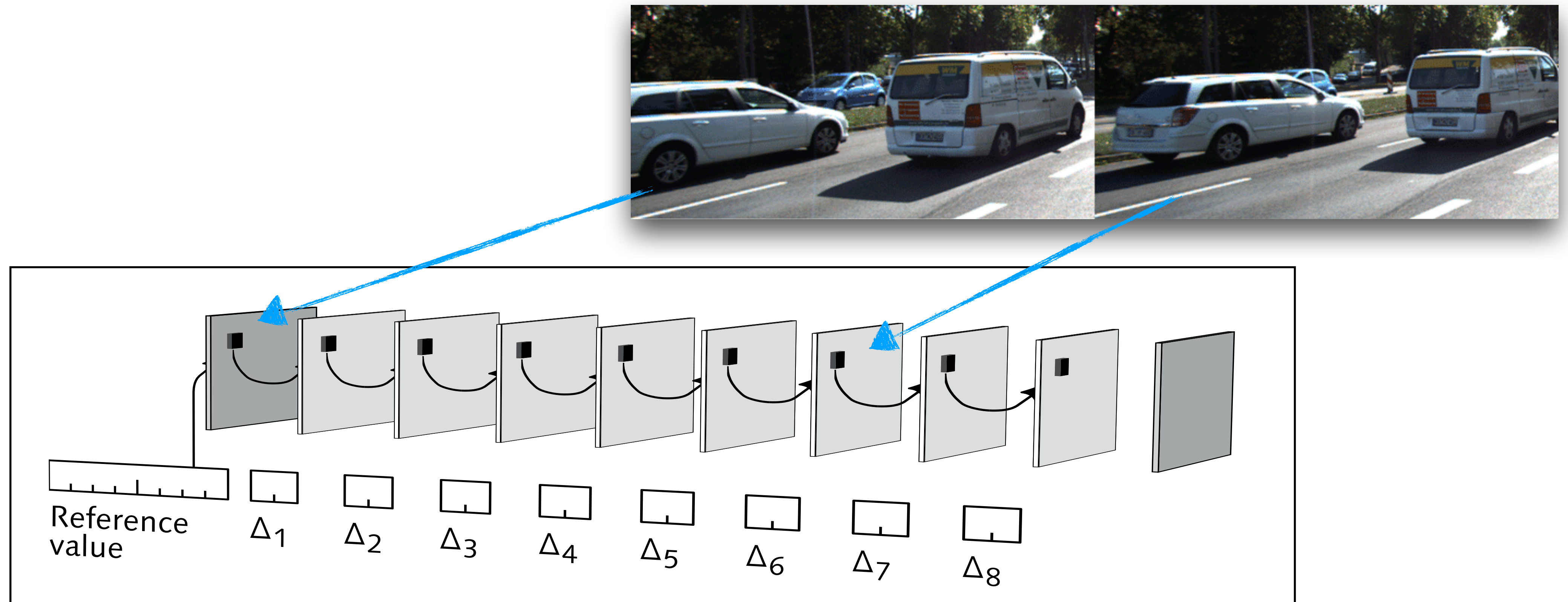
- Supports H.264 and H.265 video encoding and MPEG-4 Part 3 audio encoding.
- How many of you owned a “MP4 player”?! Can you imagine we use to buy devices that do nothing but playing .mp4 files?



Video Compression: The Big Ideas

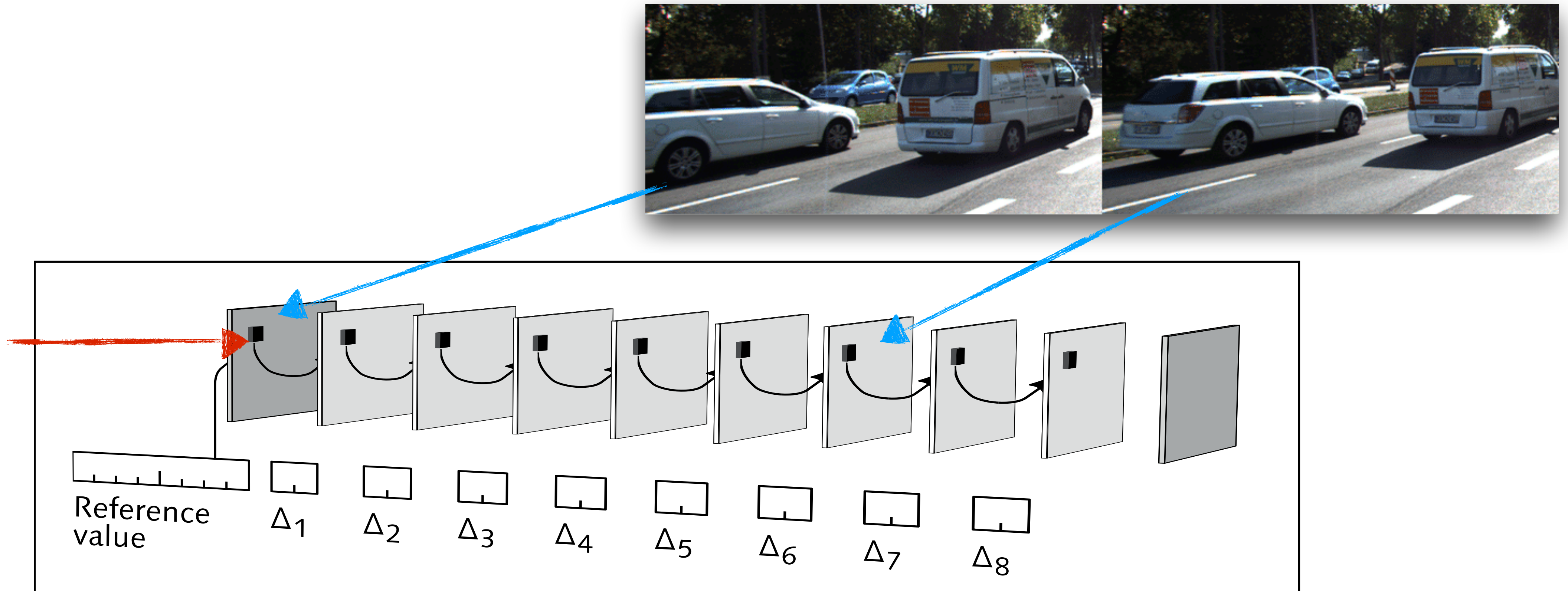


Video Compression: The Big Ideas



Video Compression: The Big Ideas

**Intra-frame
compression**
(Compressed
individually)

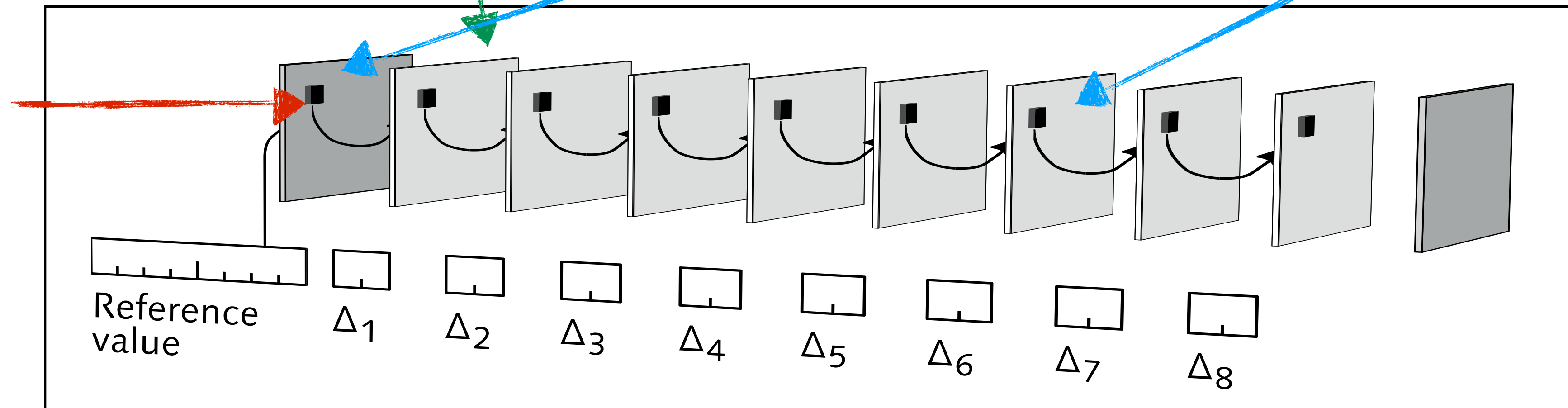


Video Compression: The Big Ideas

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



Intra-frame compression
(Compressed individually)

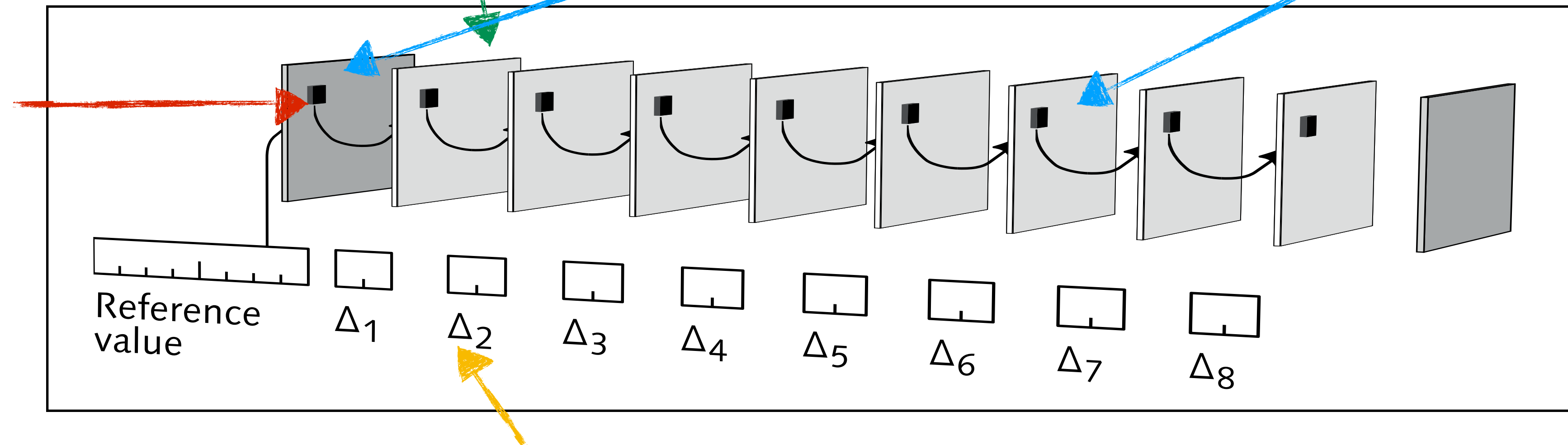


Video Compression: The Big Ideas

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



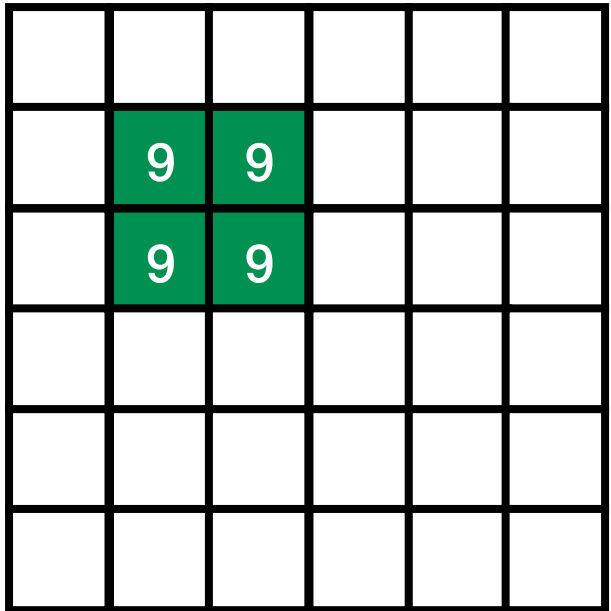
Intra-frame compression
(Compressed individually)



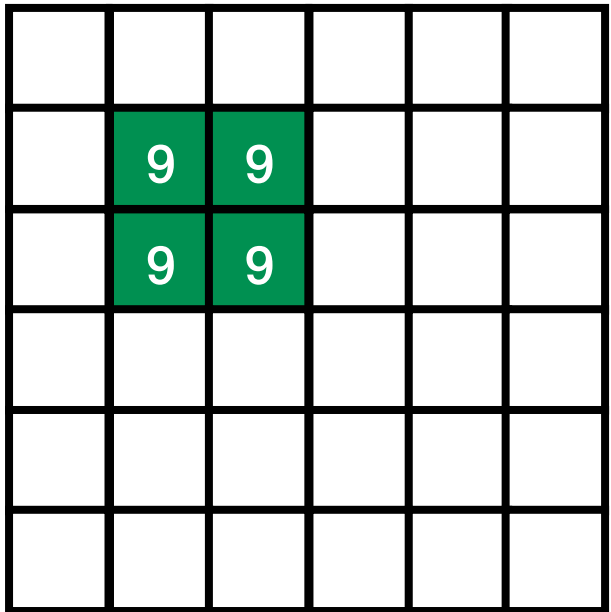
Provided that residuals can be encoded
more compactly than the image itself.

Residuals

Frame t

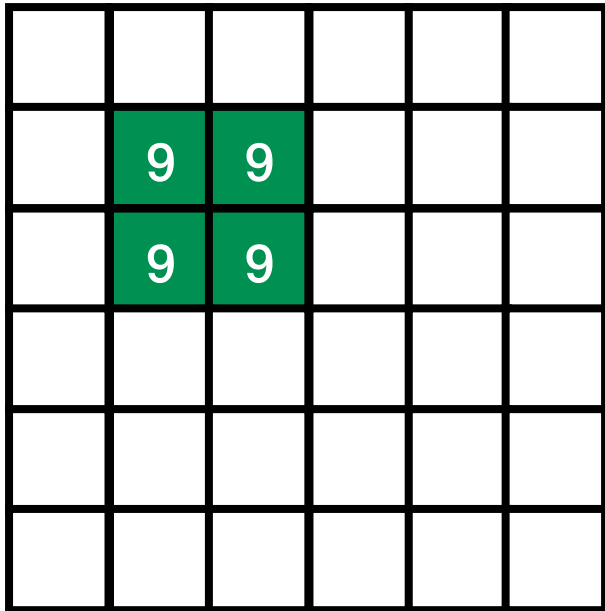


Frame t + 1

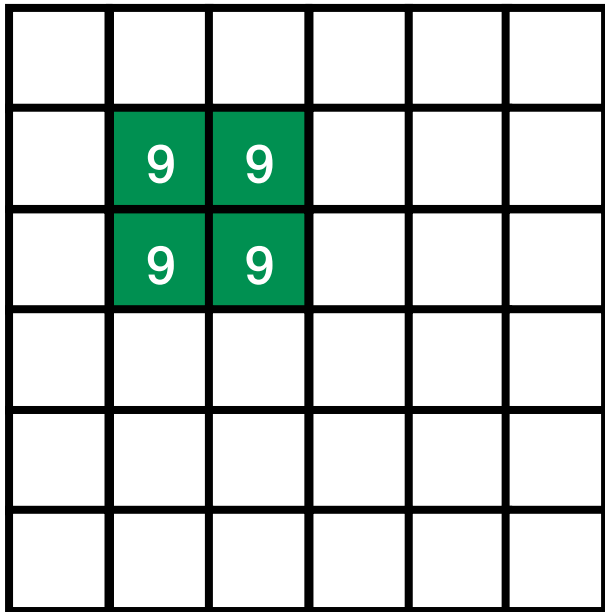


Residuals

Frame t

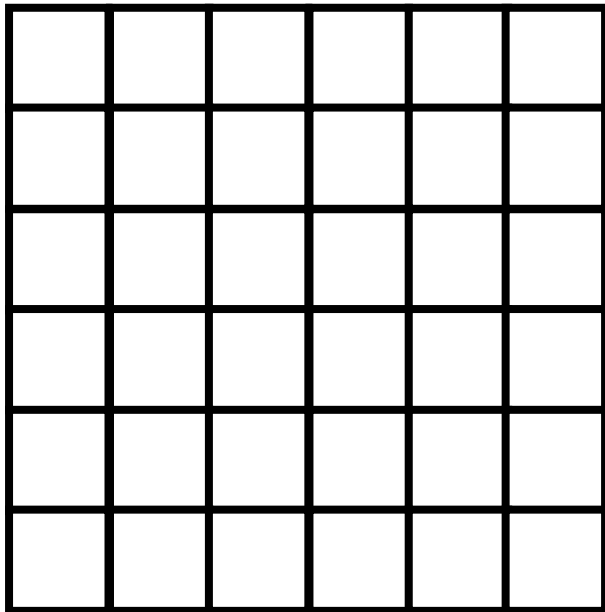


Frame t + 1



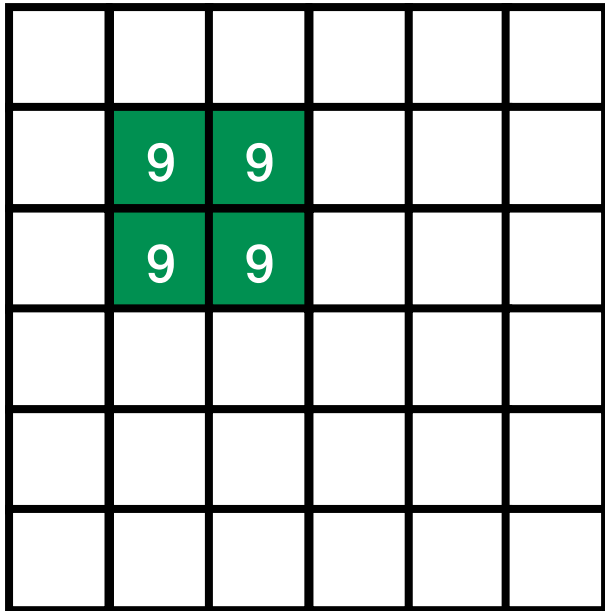
Residual

Ideal!

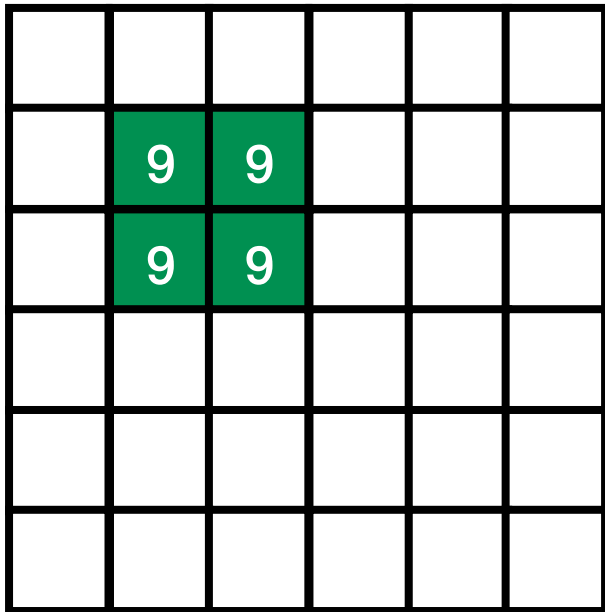


Residuals

Frame t

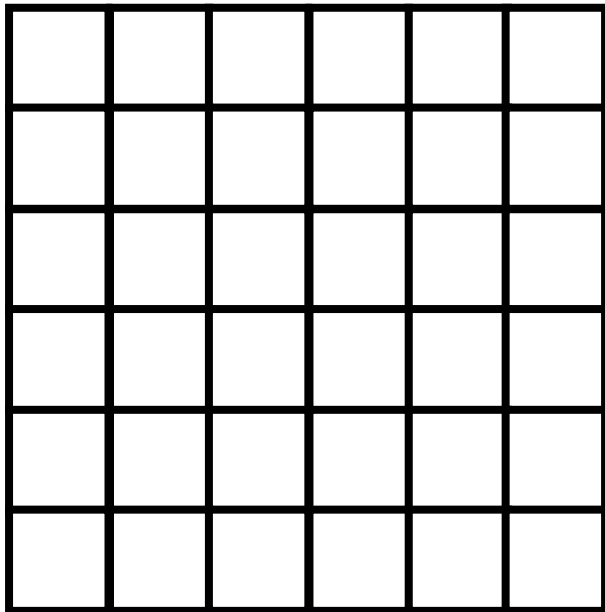


Frame t + 1

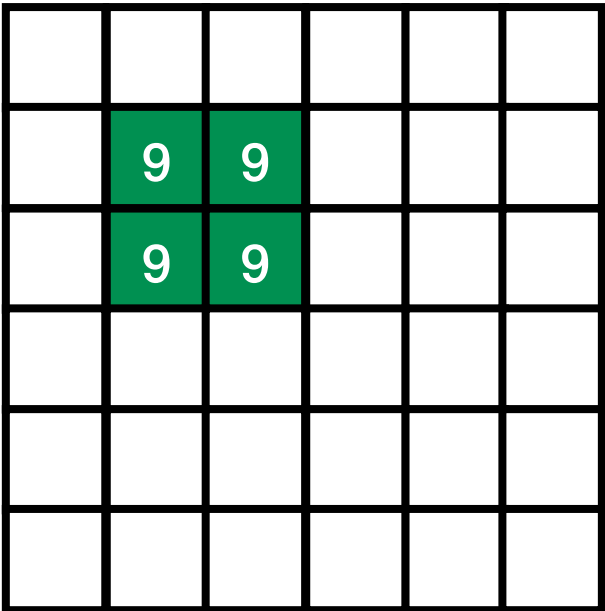


Residual

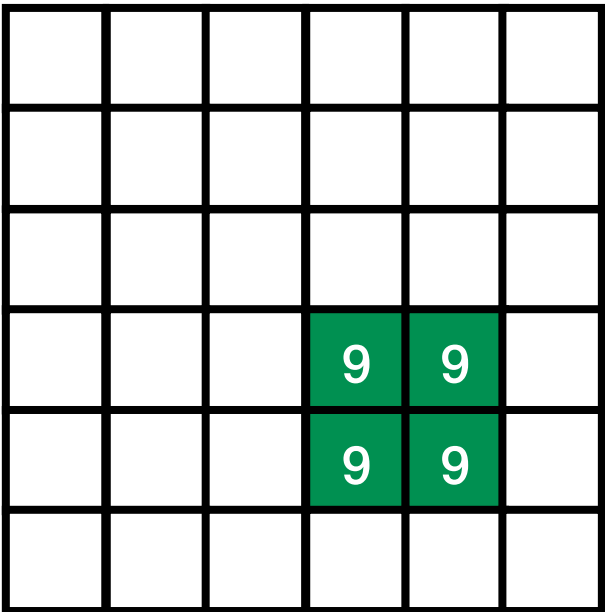
Ideal!



Frame t

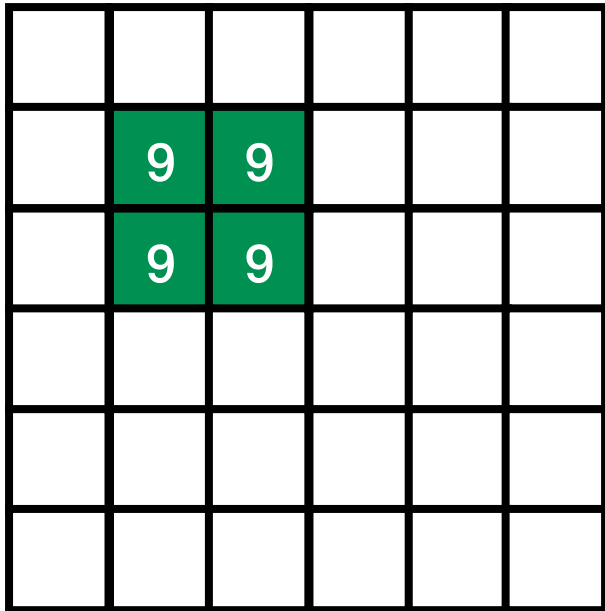


Frame t + 1

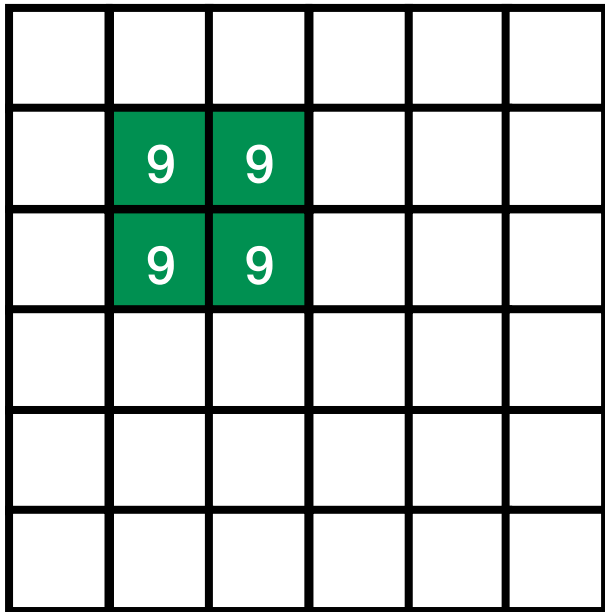


Residuals

Frame t

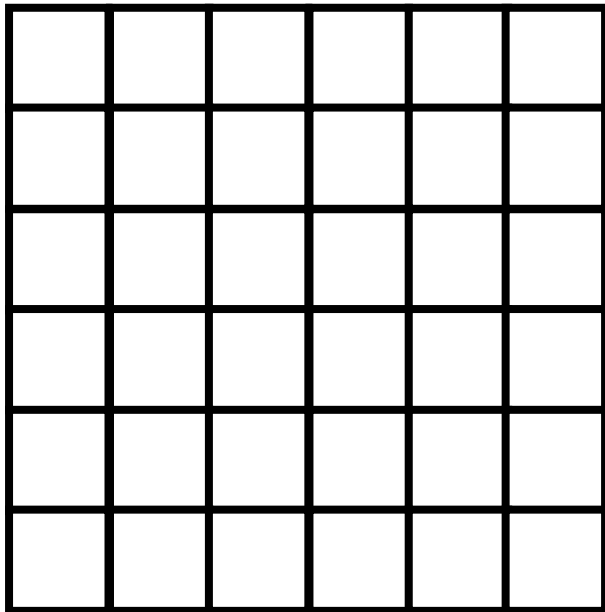


Frame t + 1

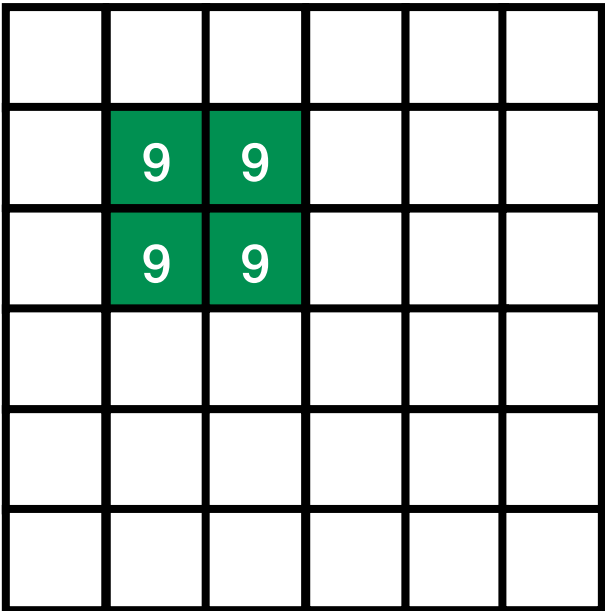


Residual

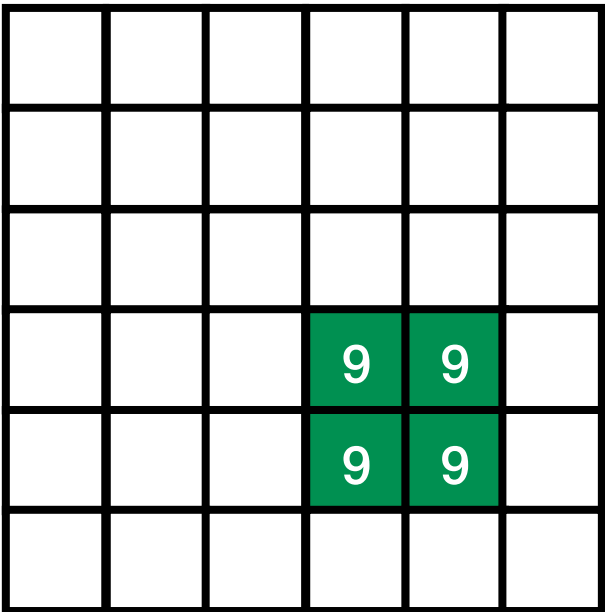
Ideal!



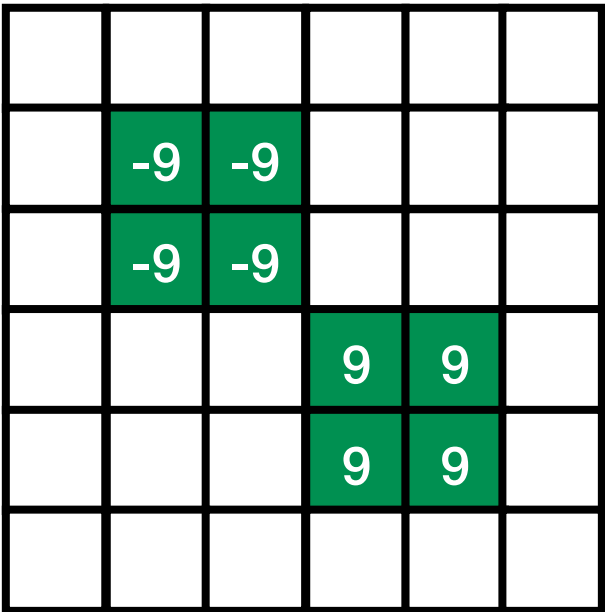
Frame t



Frame t + 1

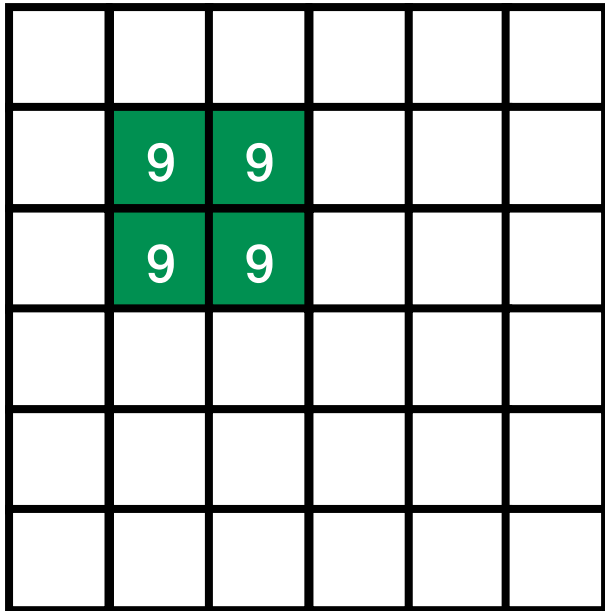


Residual

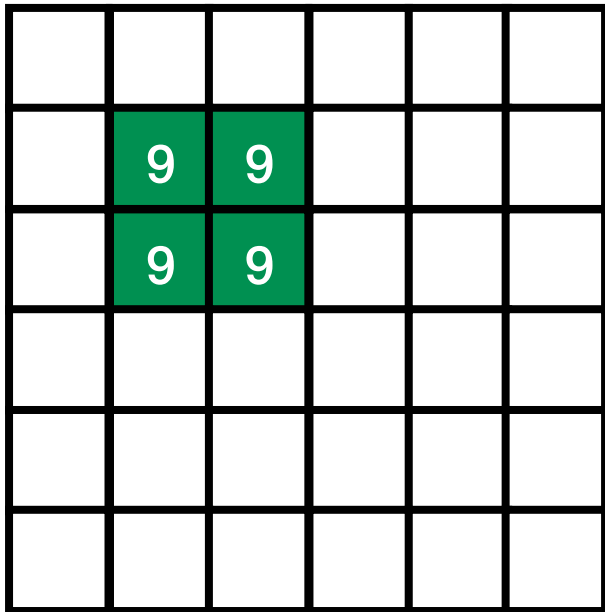


Residuals

Frame t

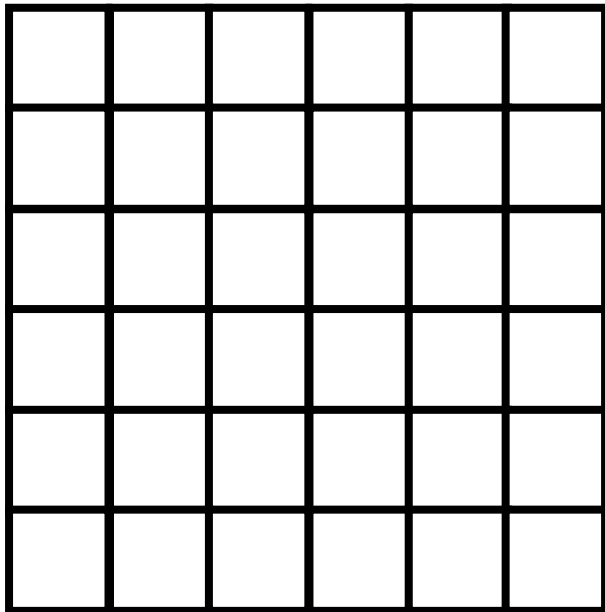


Frame t + 1

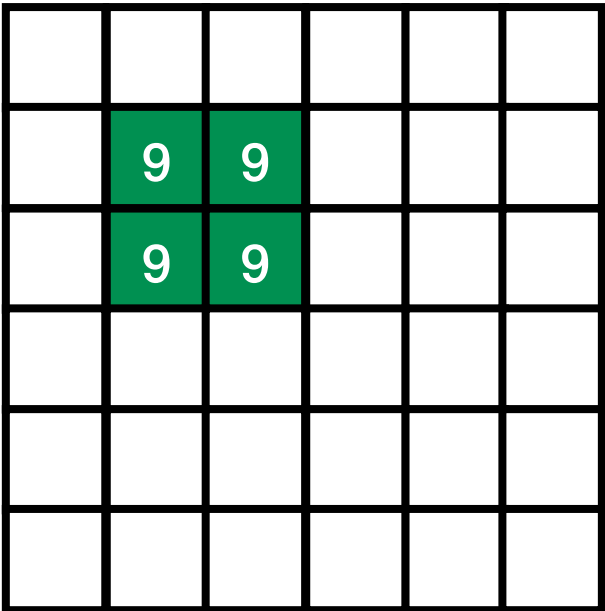


Residual

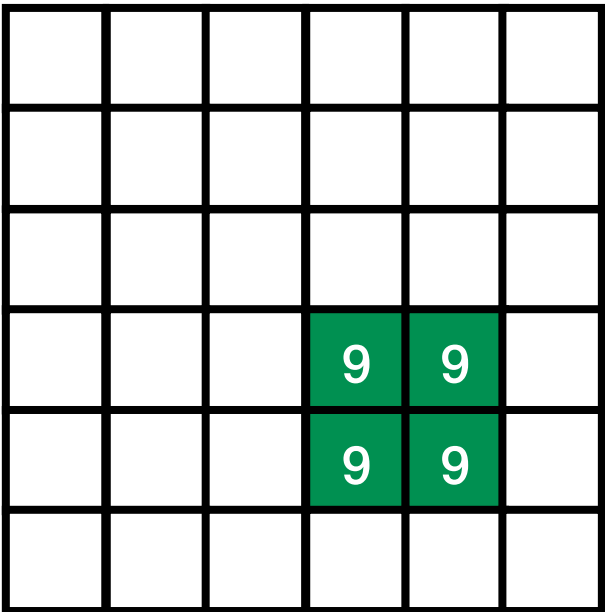
Ideal!



Frame t

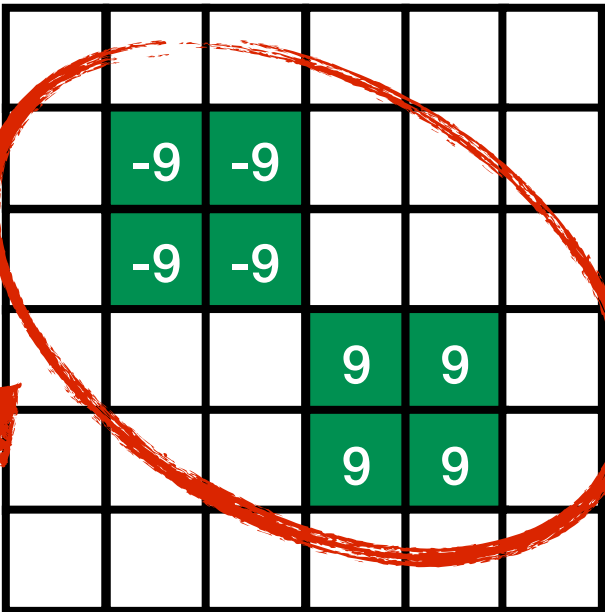


Frame t + 1



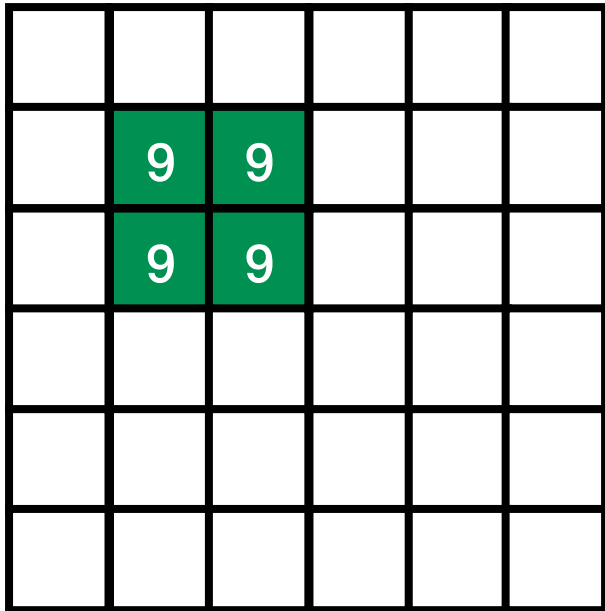
Residual

Worse!

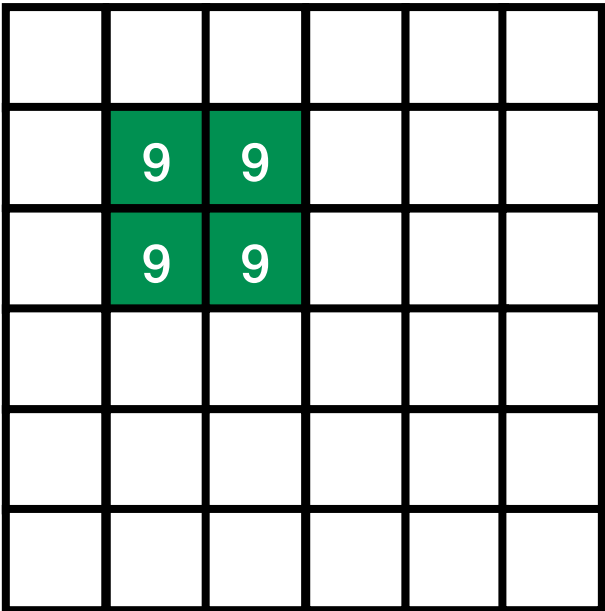


Residuals

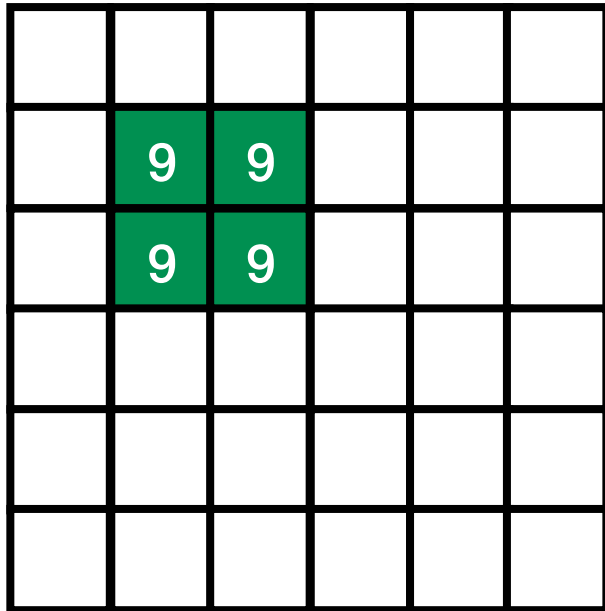
Frame t



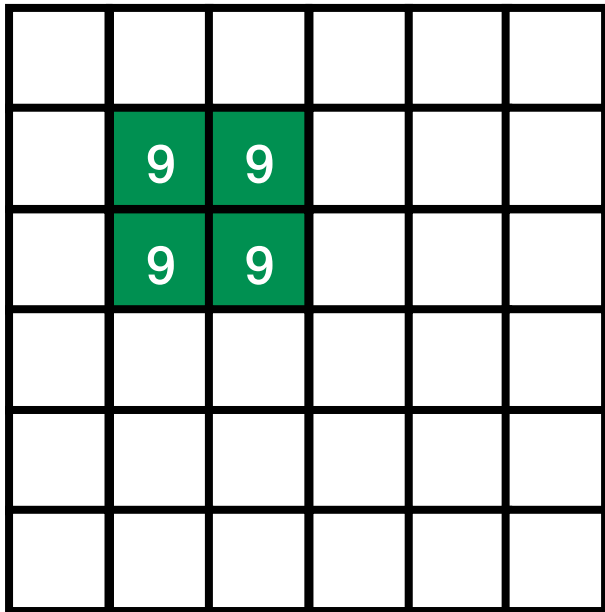
Frame t



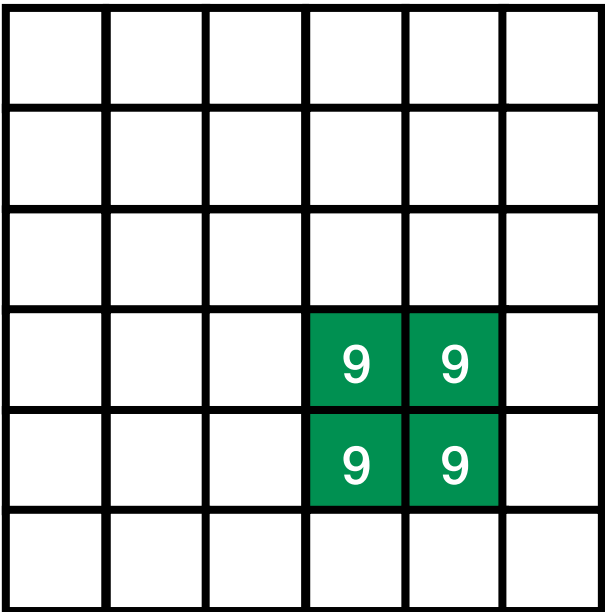
Frame t



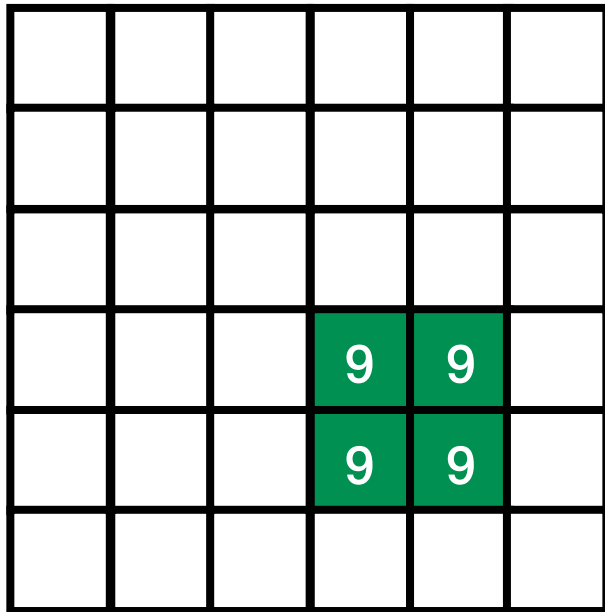
Frame t + 1



Frame t + 1

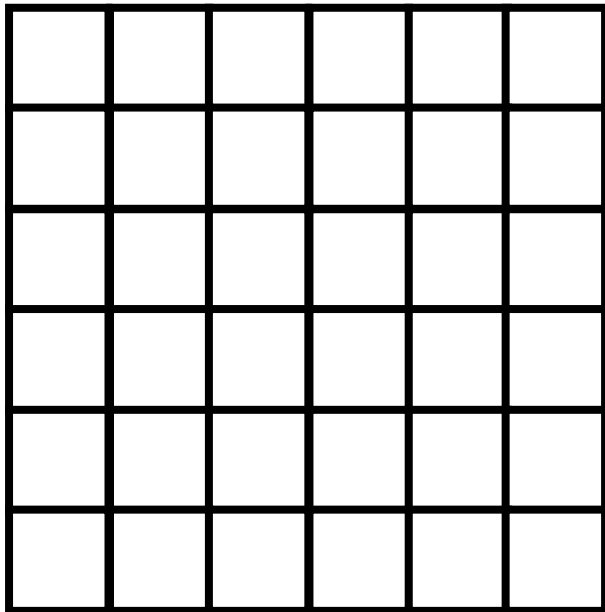


Frame t + 1



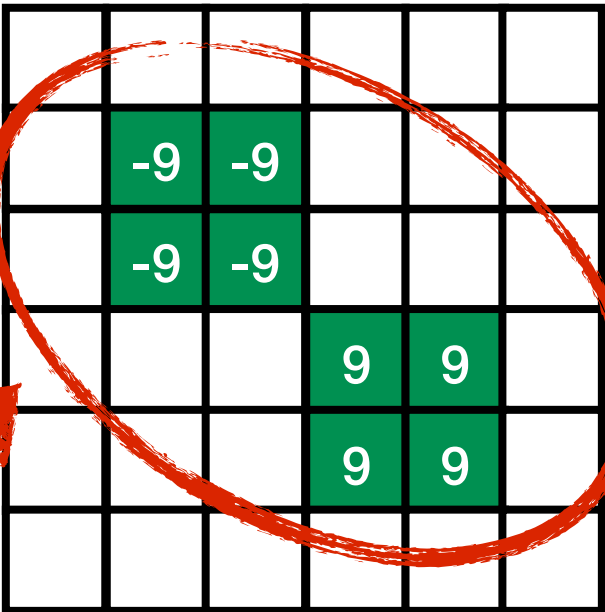
Residual

Ideal!

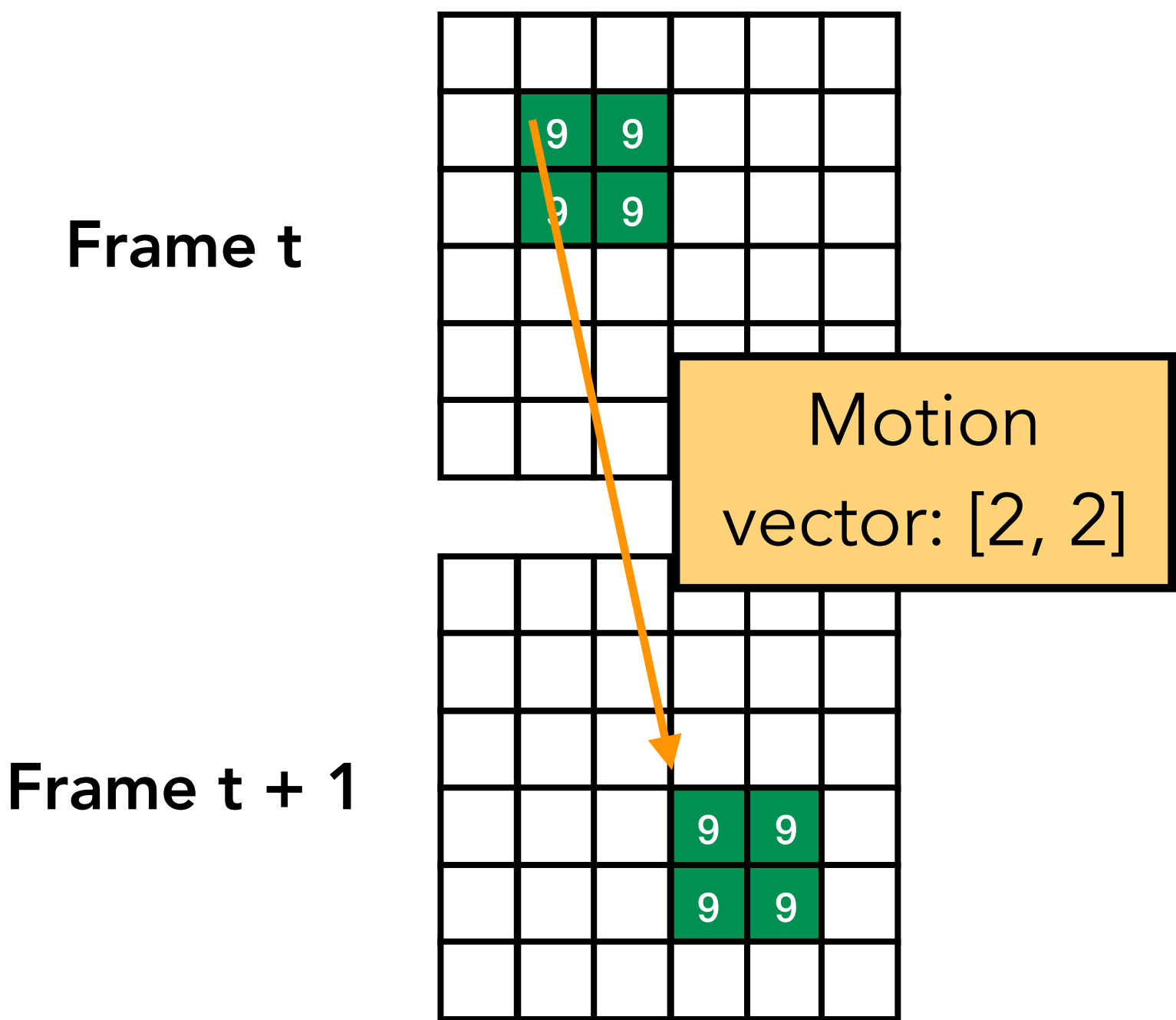
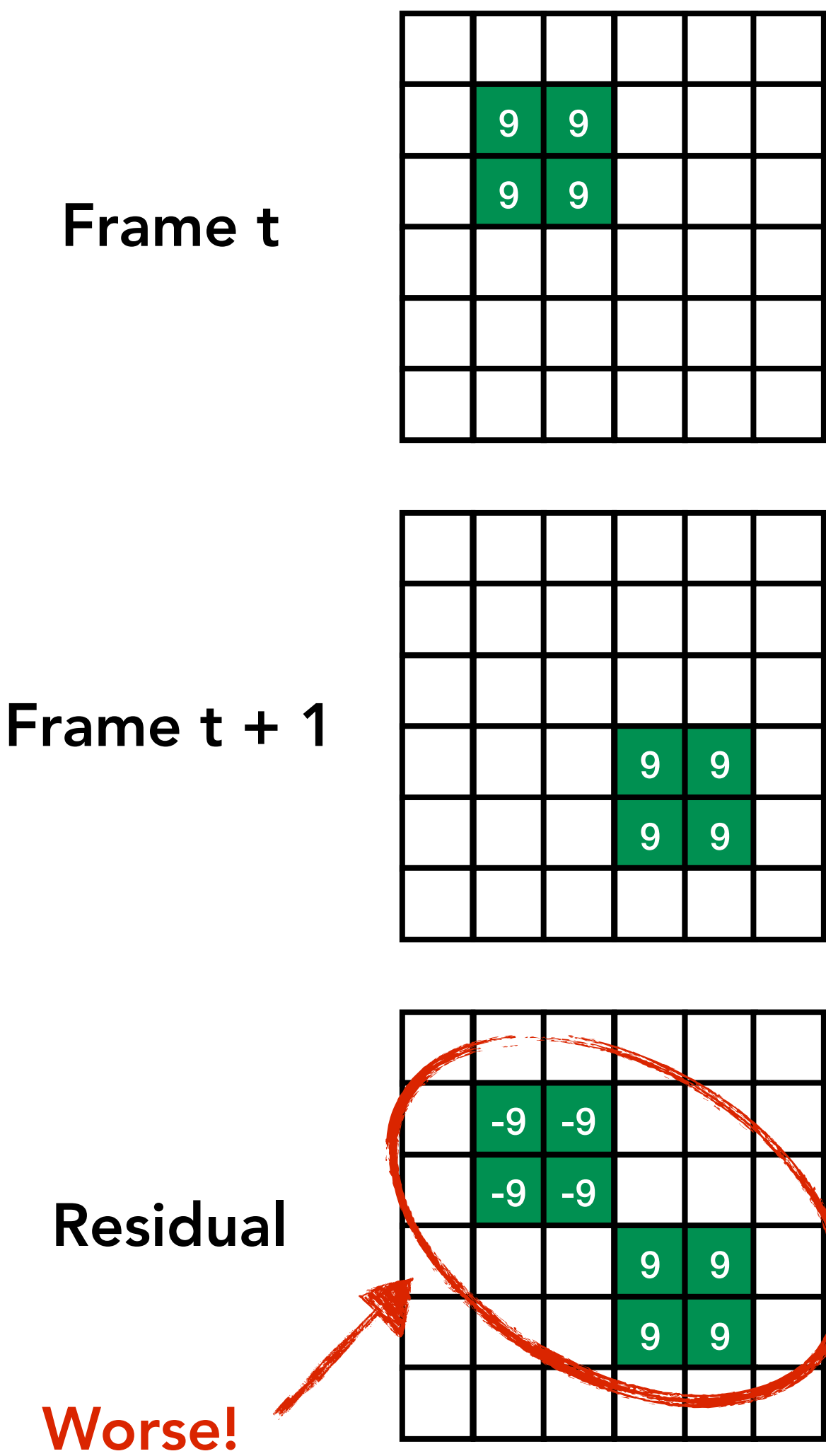
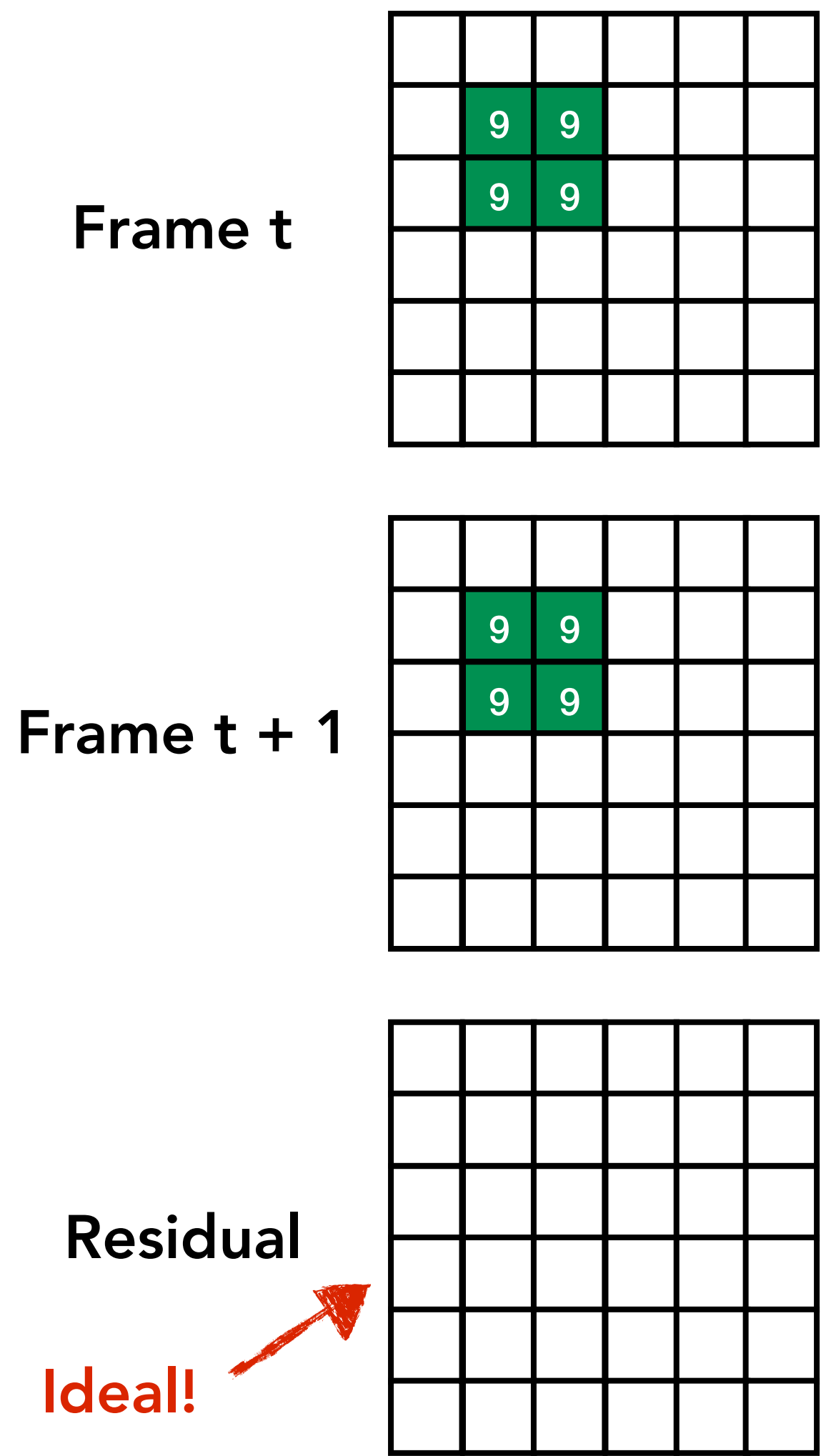


Residual

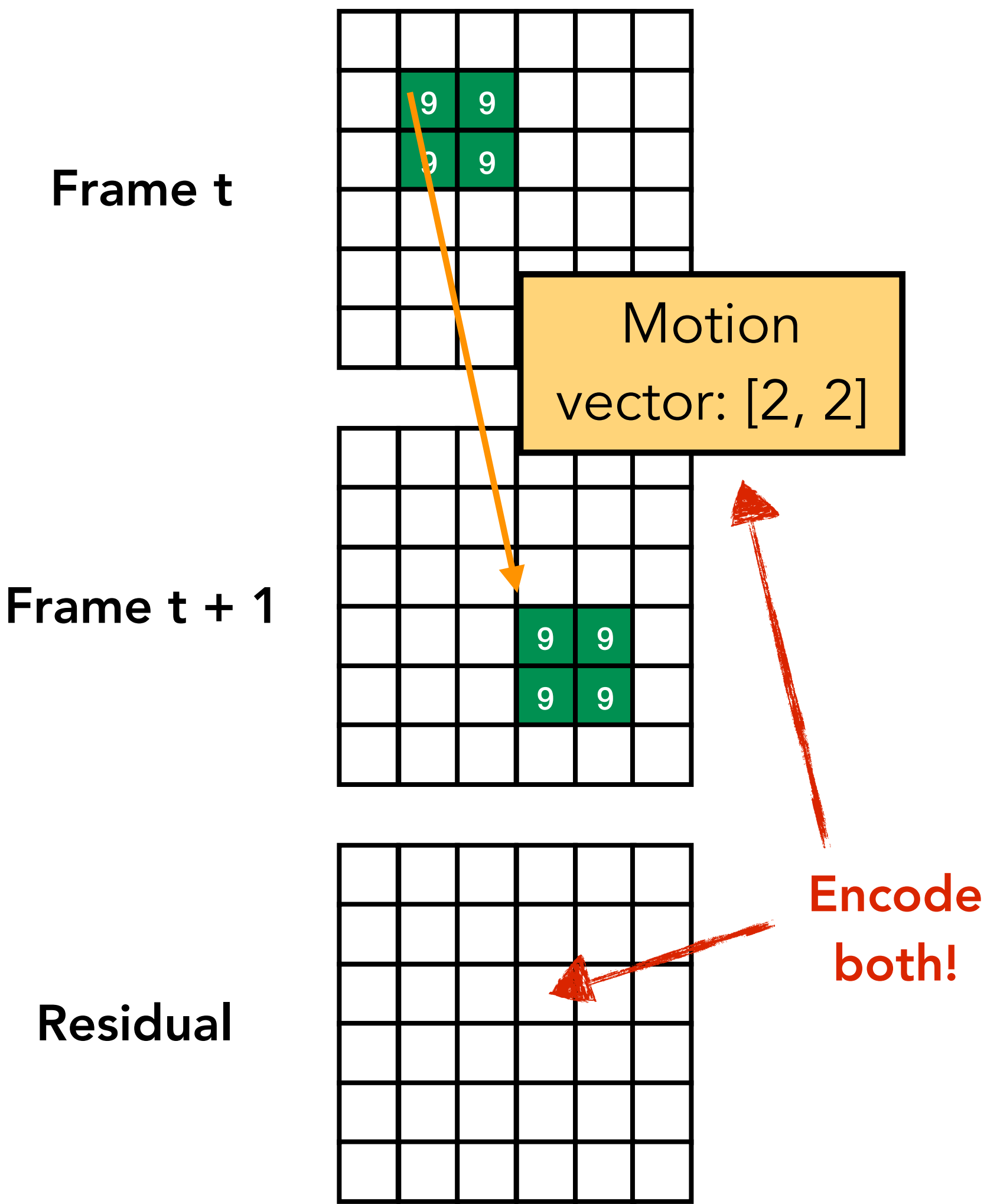
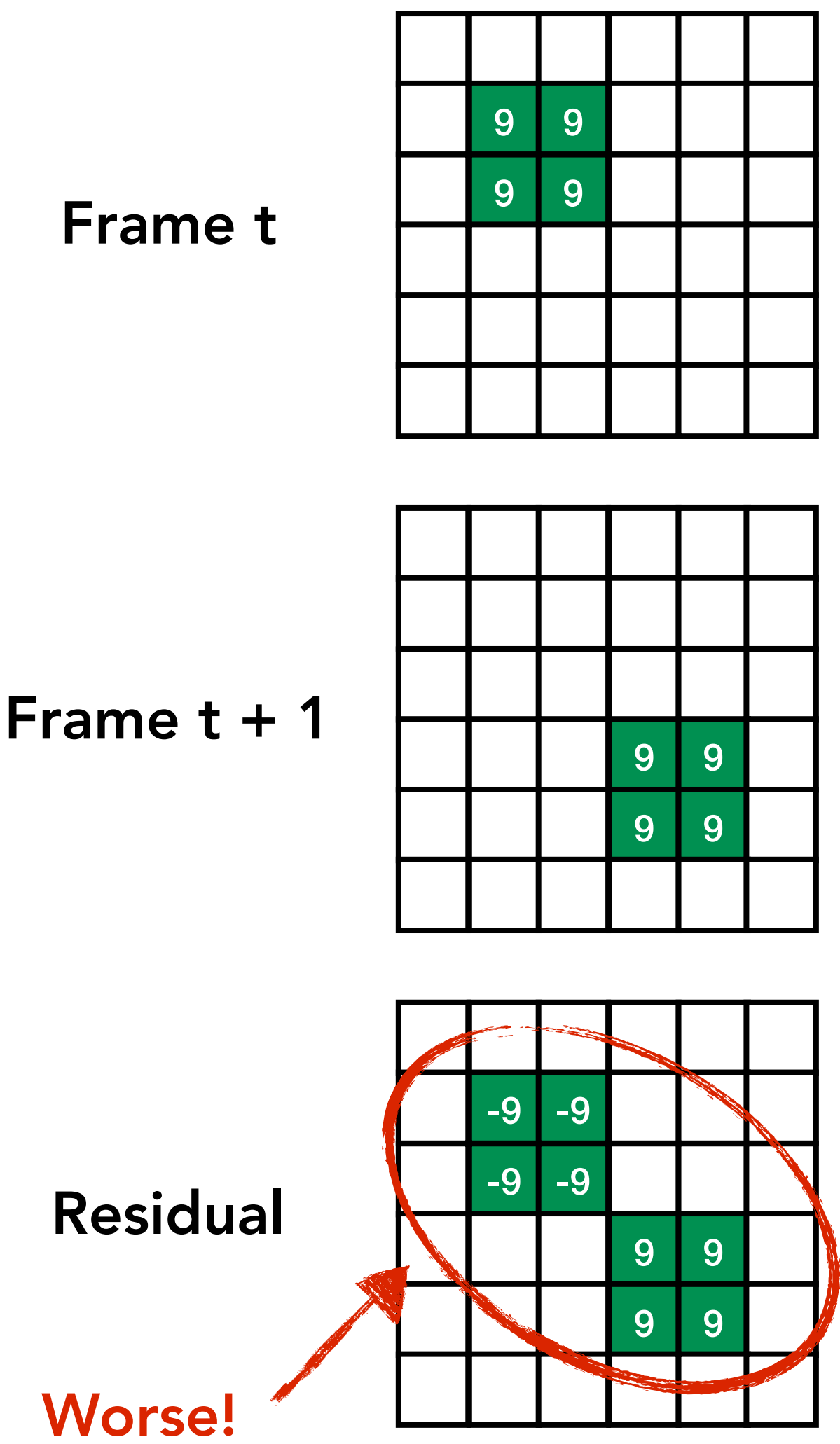
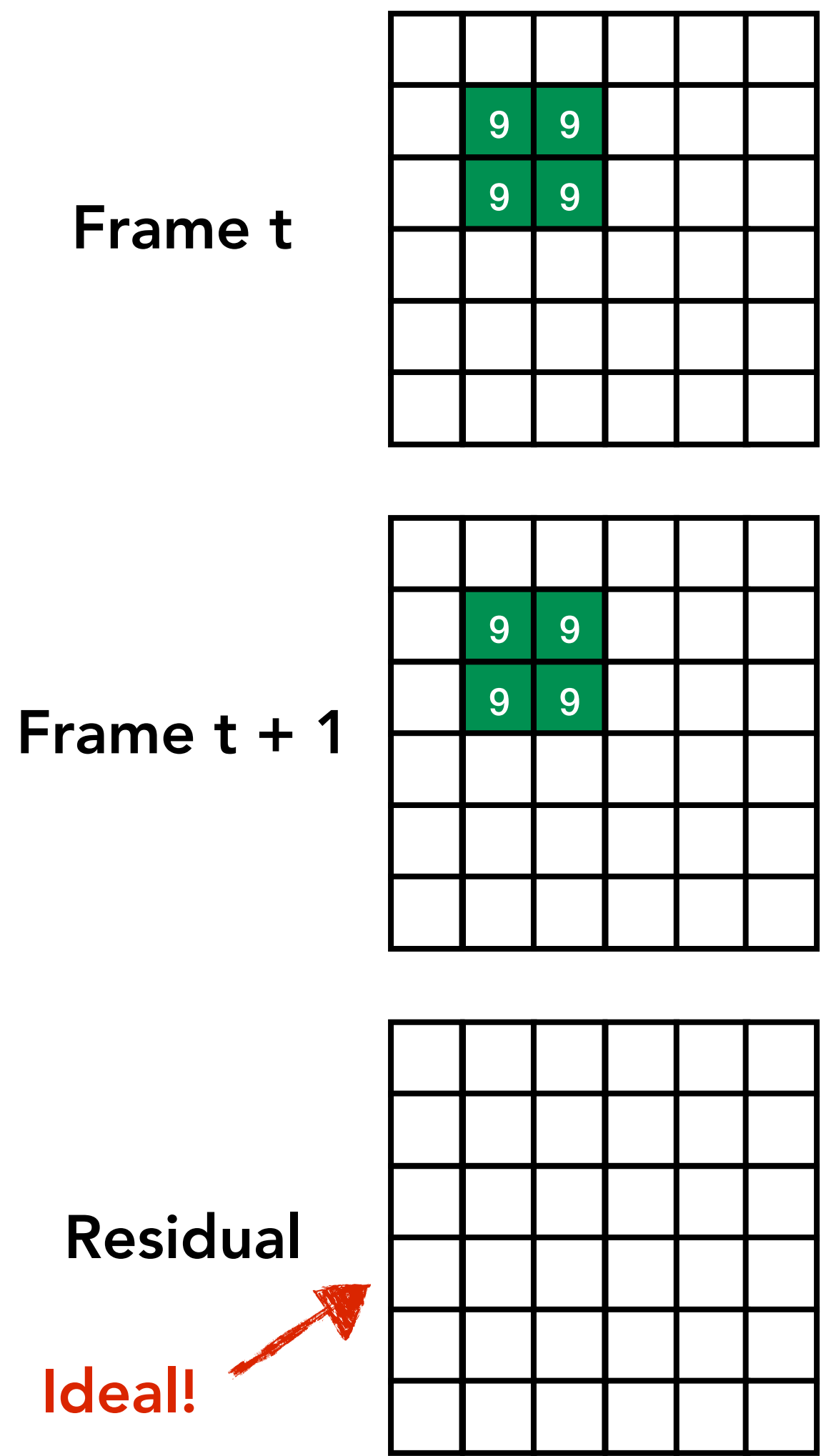
Worse!



Residuals

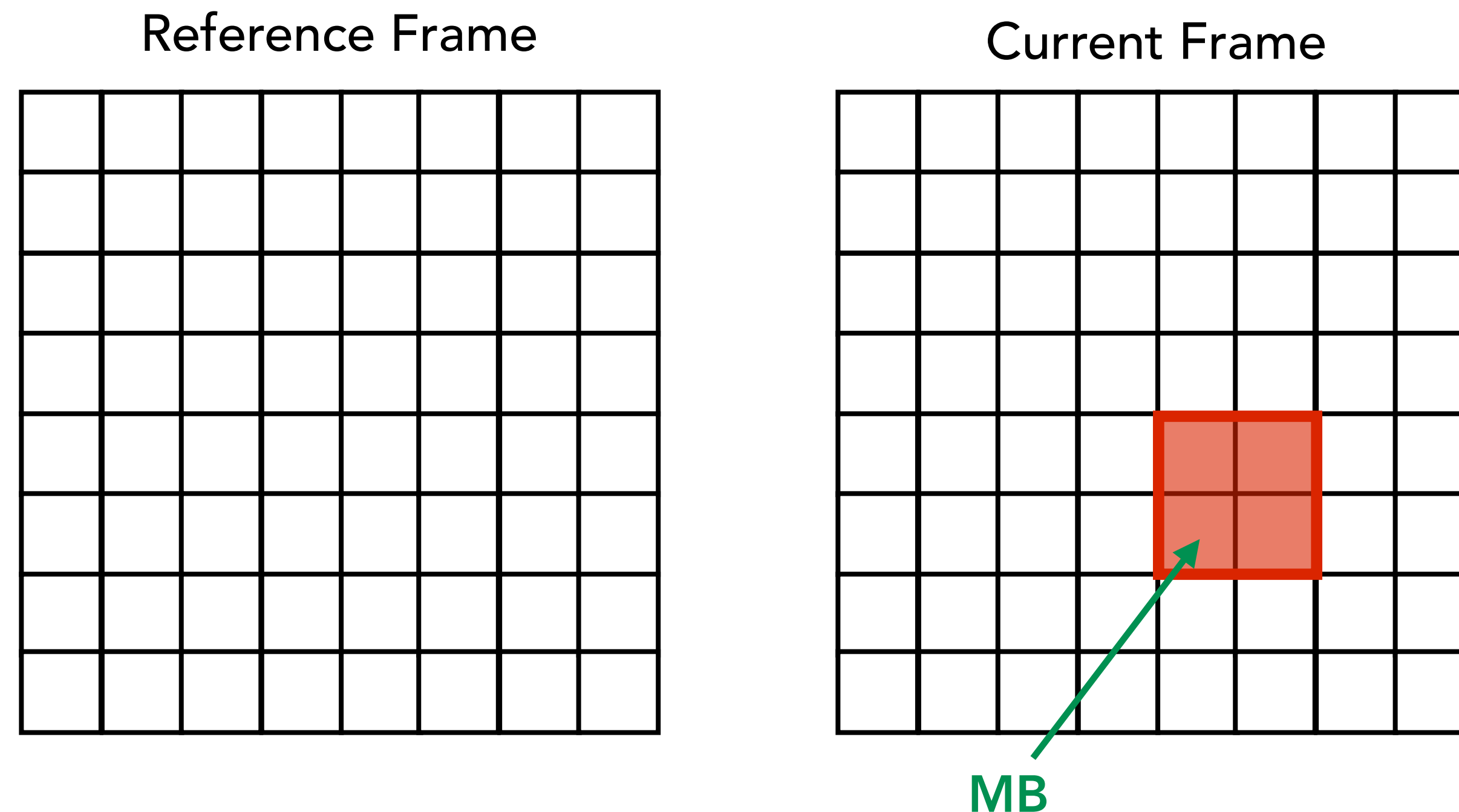


Residuals



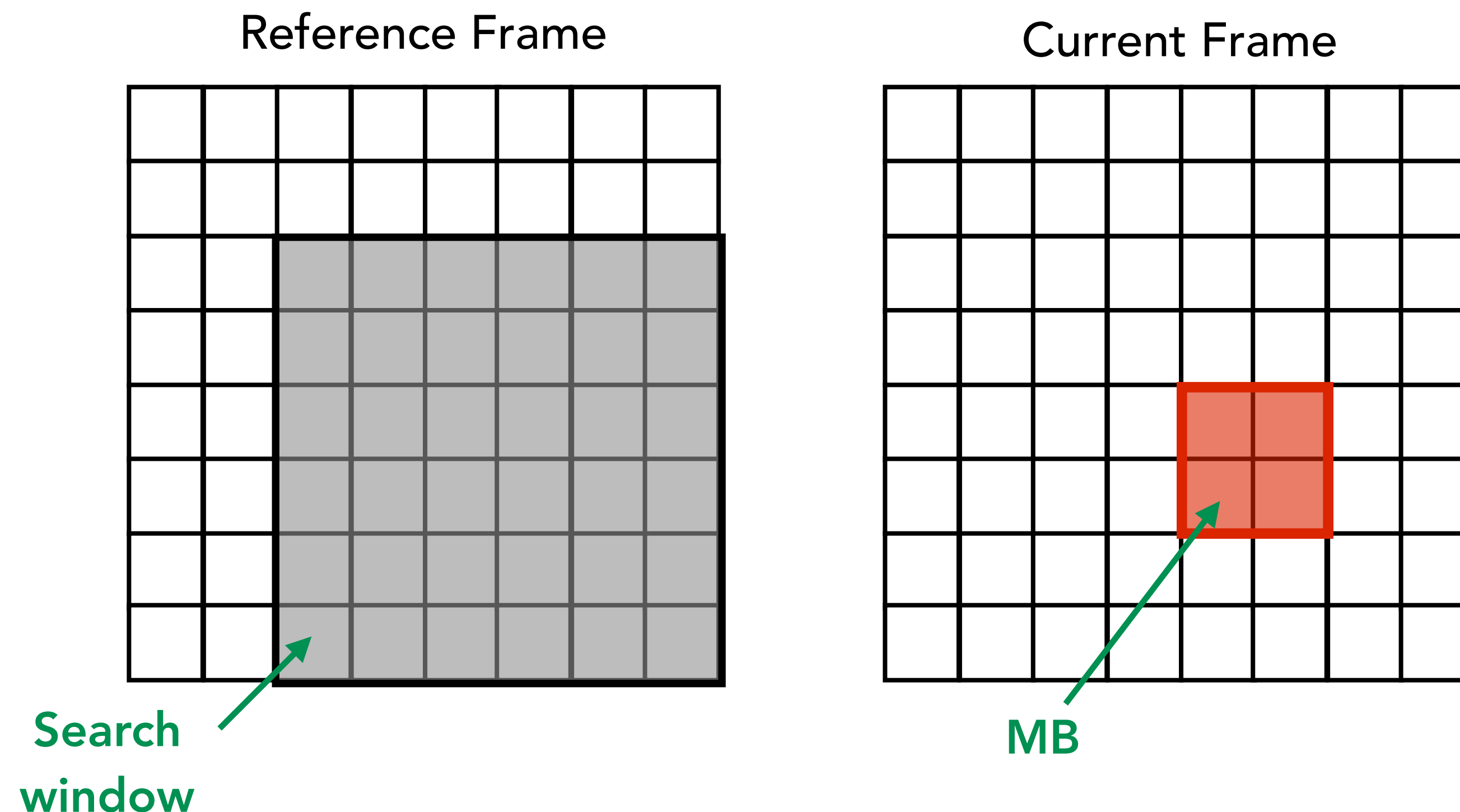
Motion Estimation

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.



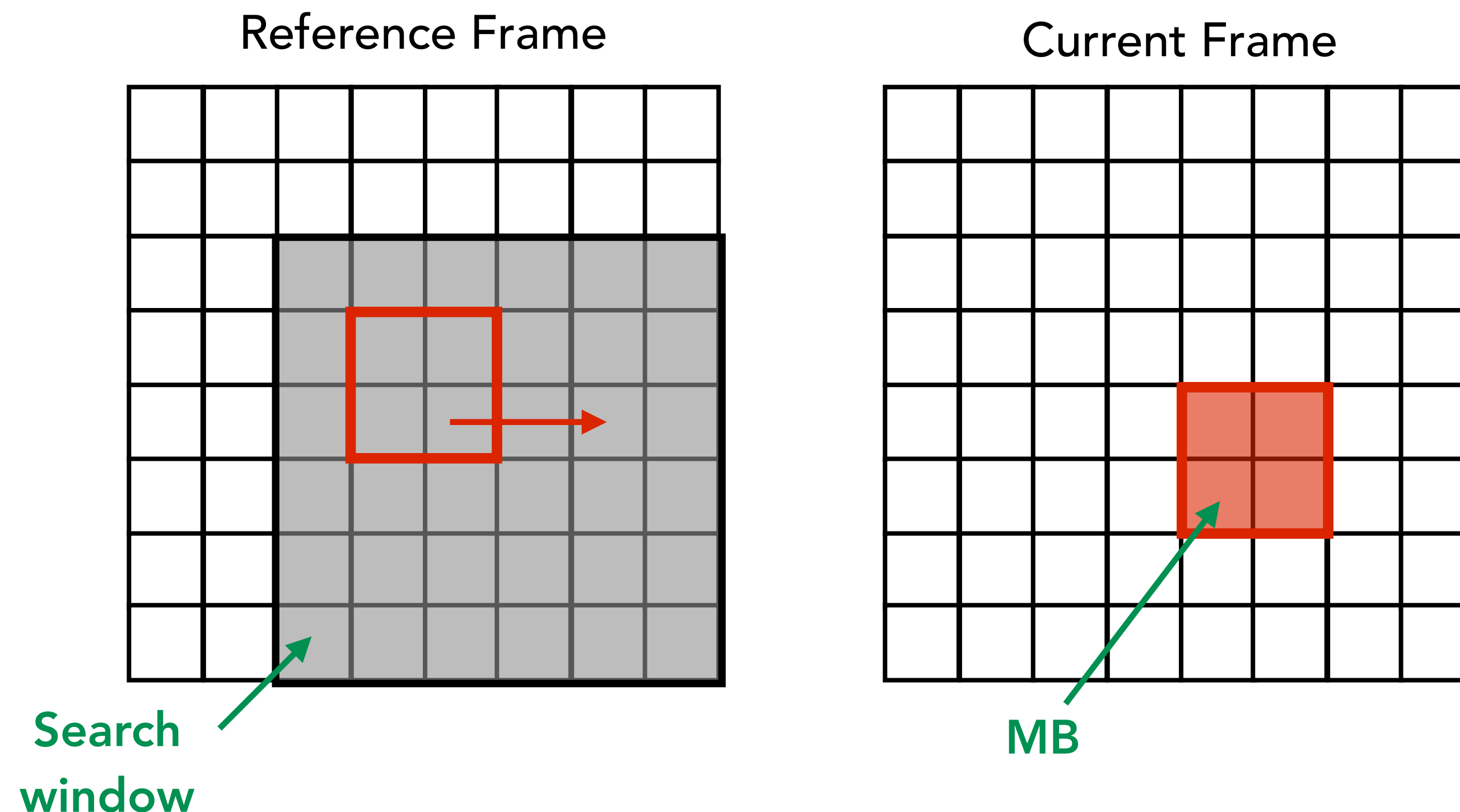
Motion Estimation

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.



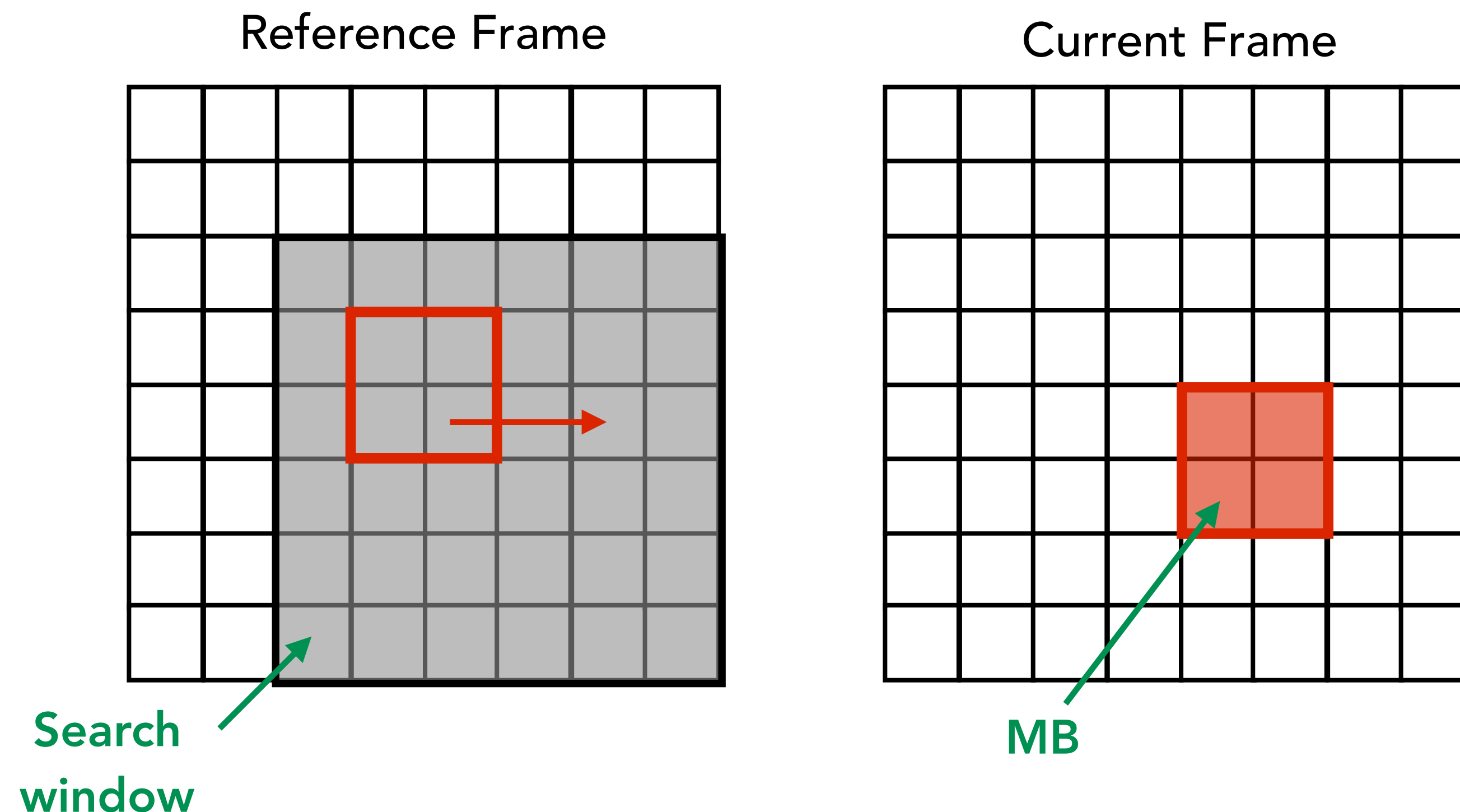
Motion Estimation

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 Estimation

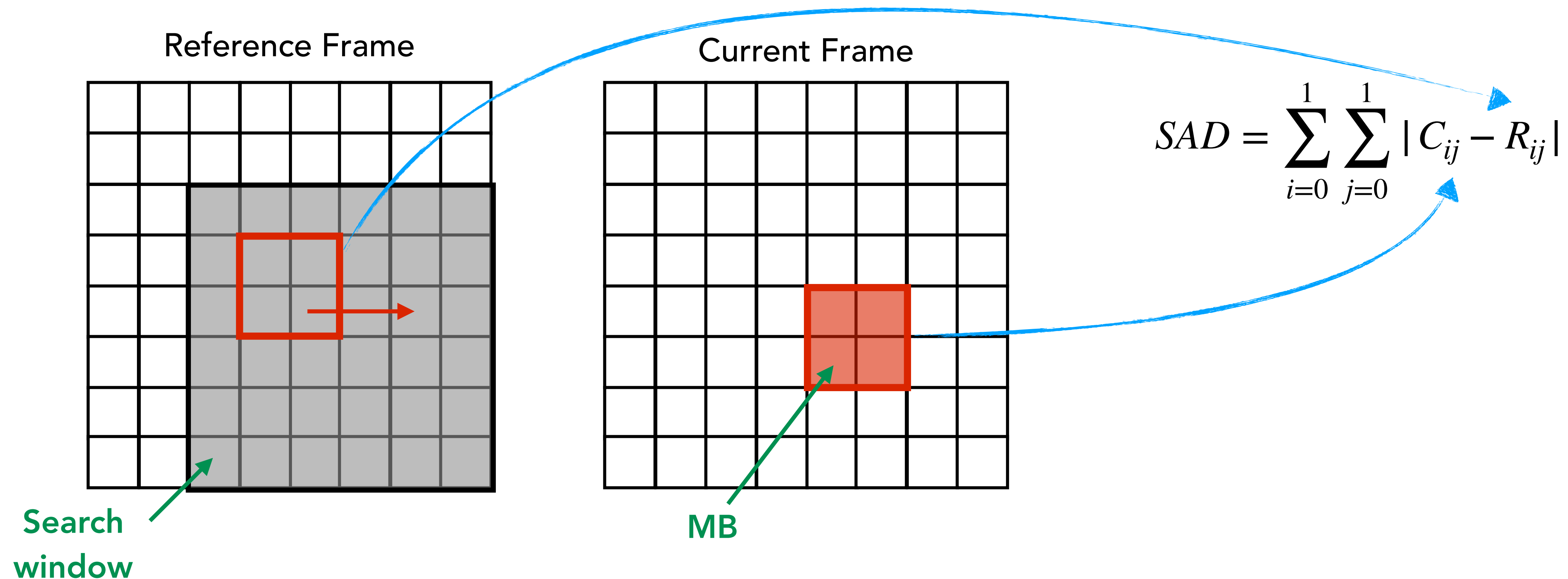
In MPEG video compression, motion estimation is done using the **block matching** algorithm, which operates at the granularity of **macroblocks** (MB), and uses some form of **absolute difference** as the cost function.



$$SAD = \sum_{i=0}^1 \sum_{j=0}^1 |C_{ij} - R_{ij}|$$

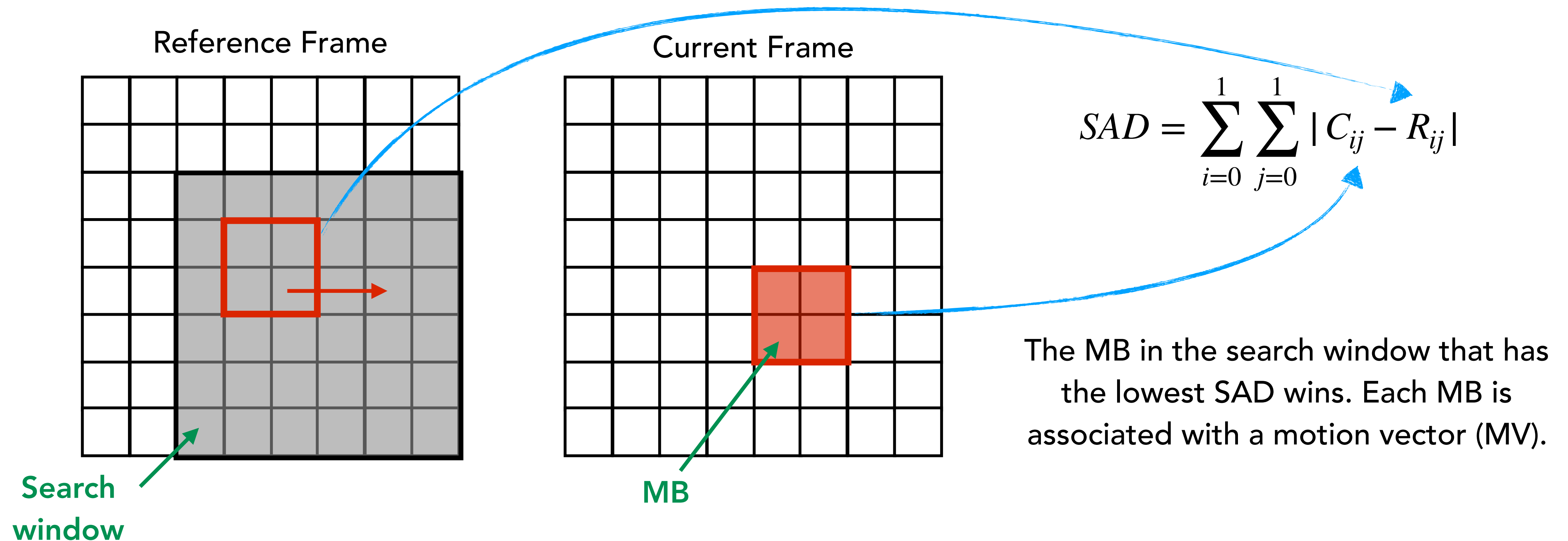
Motion Estimation

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.



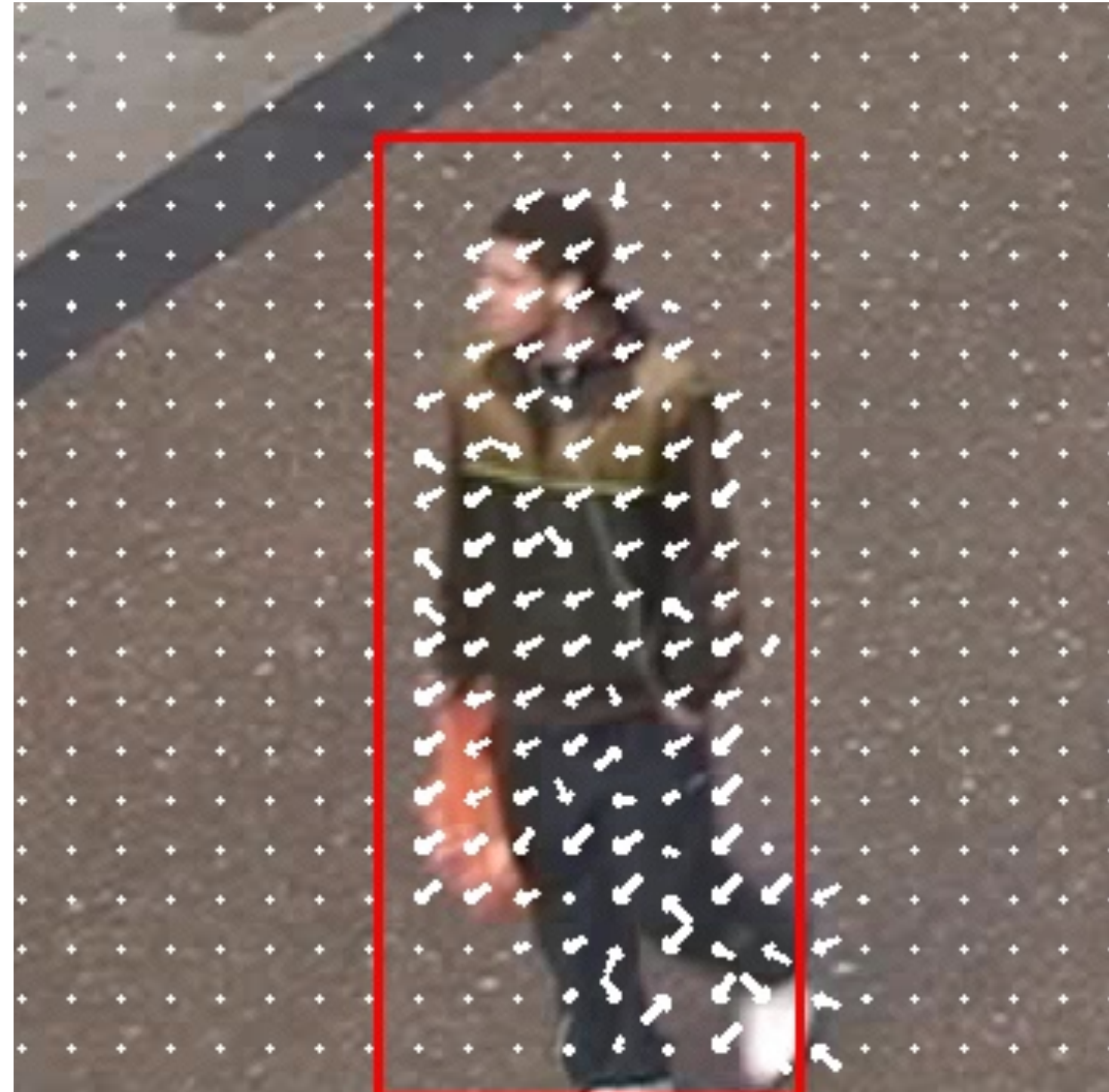
Motion Estimation

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.



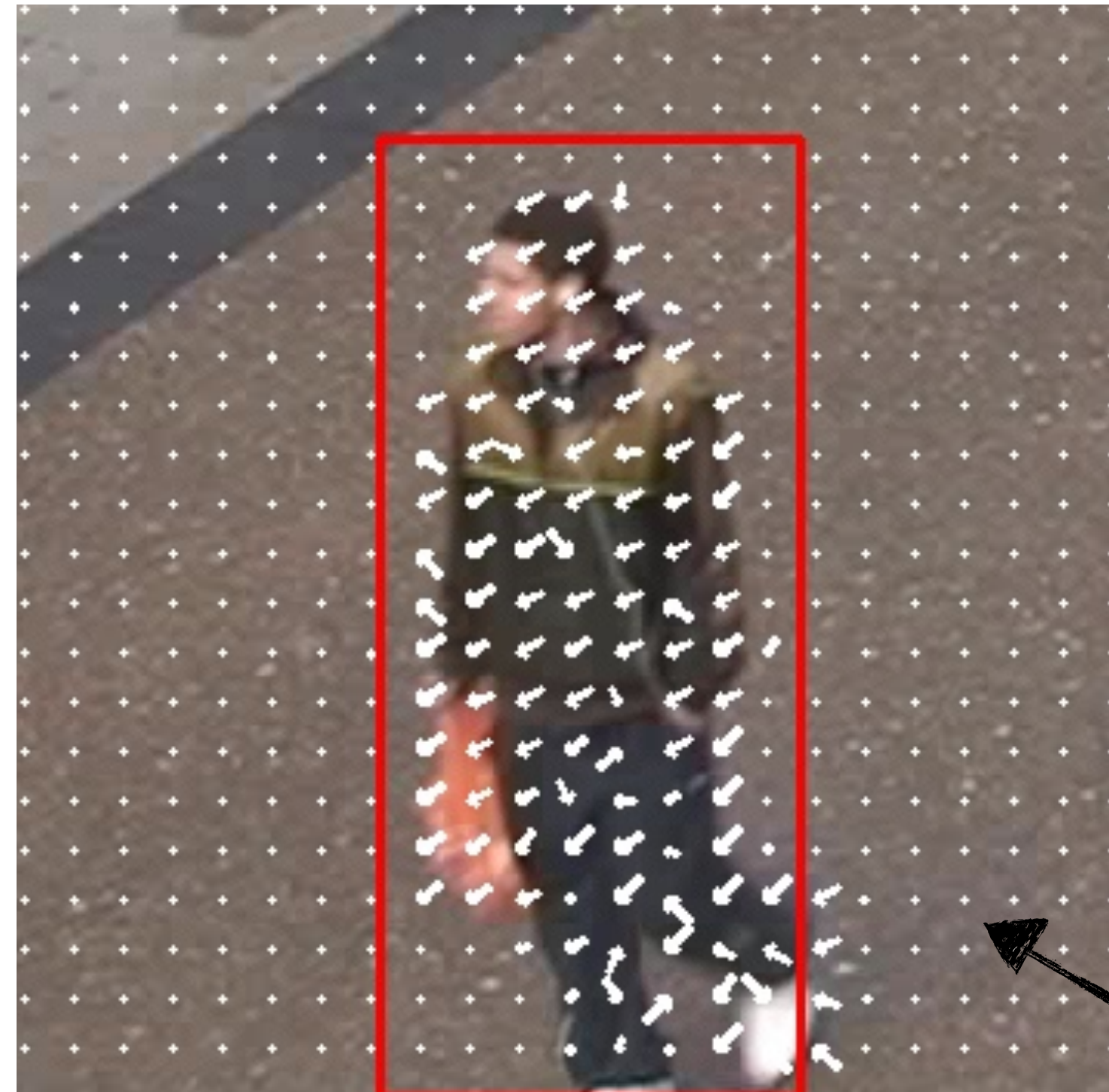
Motion Vectors

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



Motion Vectors

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

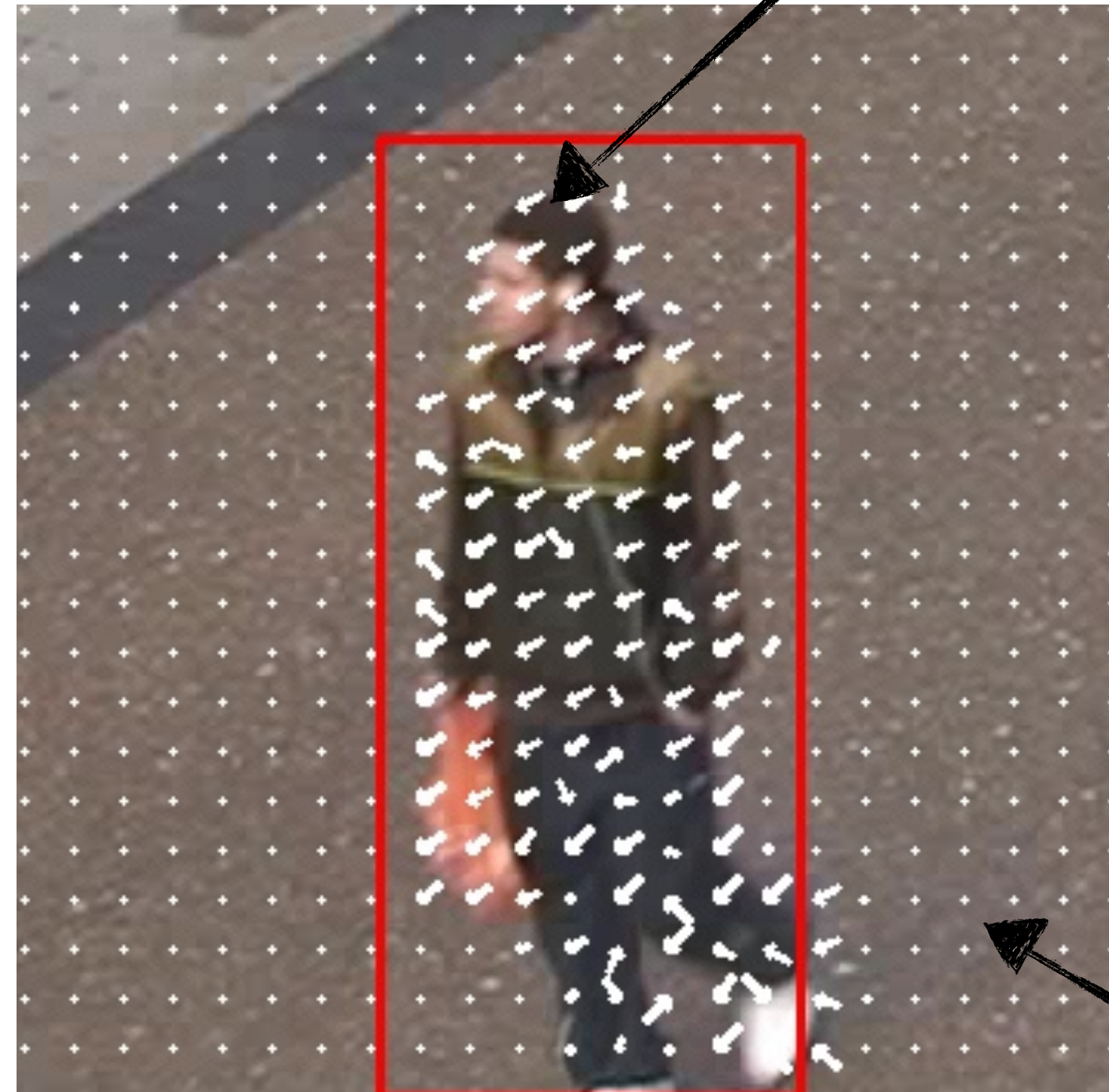


Why do these pixels have zero motion?

Motion Vectors

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

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



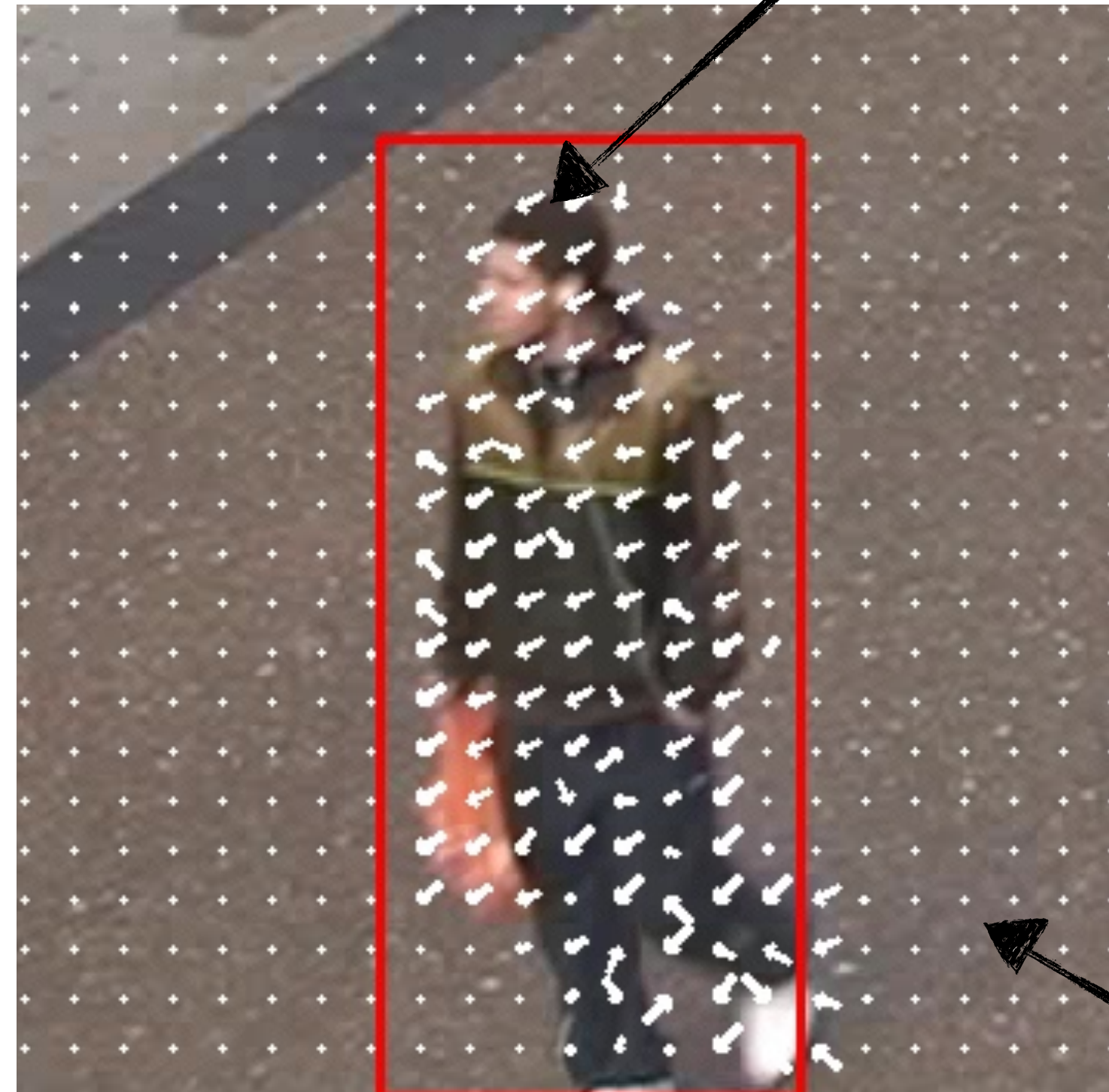
Why do these pixels have zero motion?

Motion 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 infer the scene: object detection, tracking, etc.

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

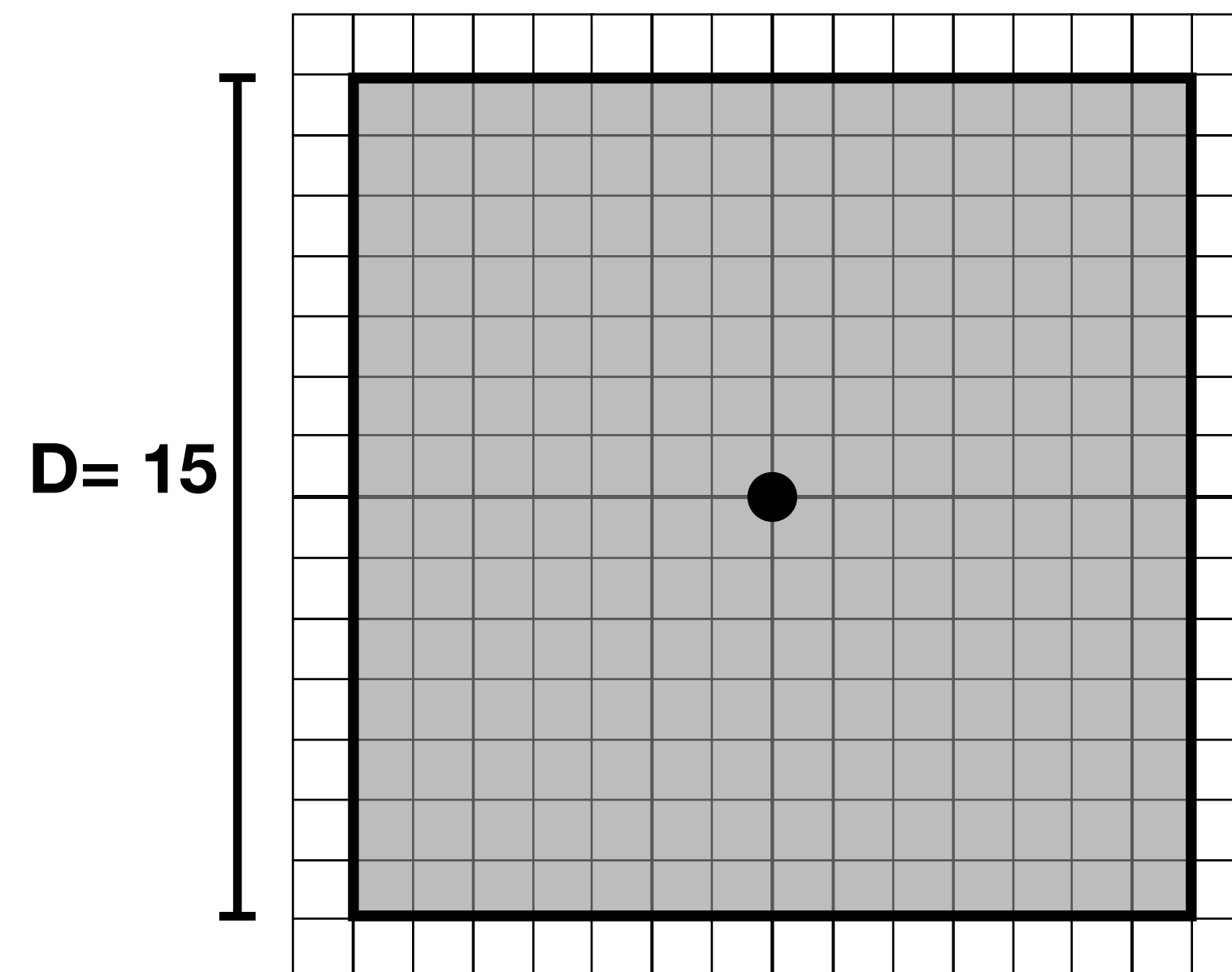


Why do these pixels have zero motion?

Motion Estimation Heuristics

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

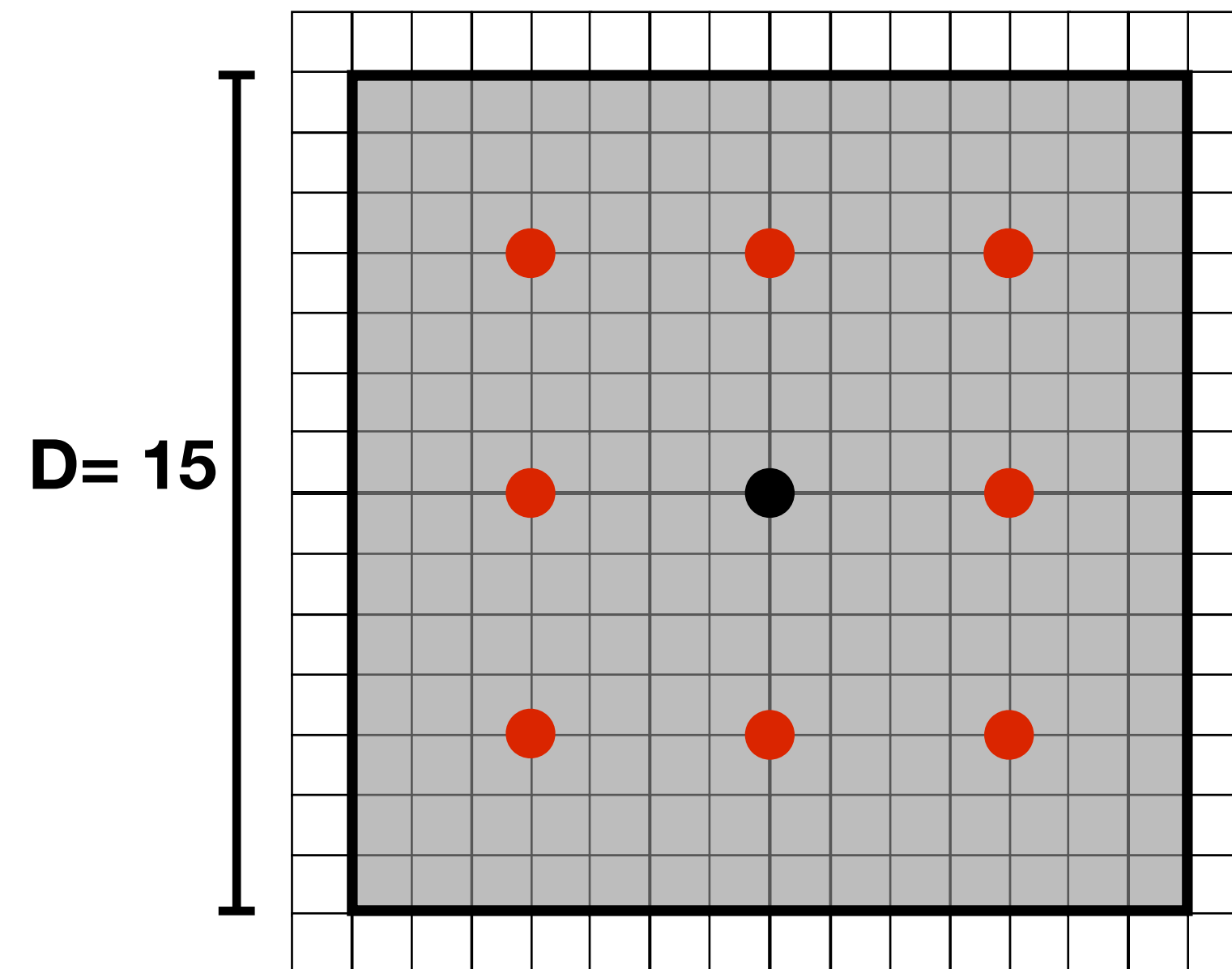
Three step search (TSS) is a classic heuristics that adaptively narrows the search range (coarse-to-fine search).



Motion Estimation Heuristics

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

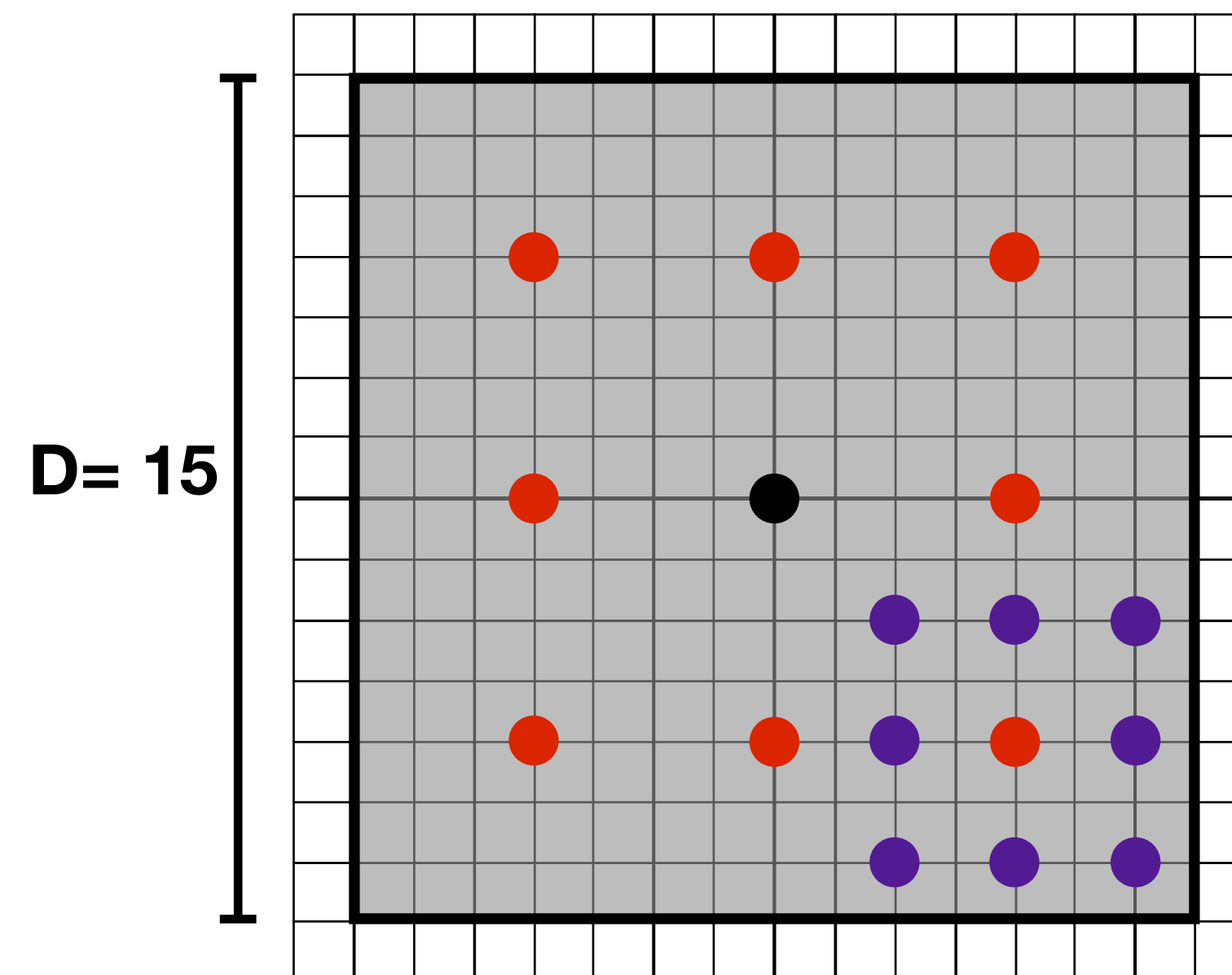
Three step search (TSS) is a classic heuristics that adaptively narrows the search range (coarse-to-fine search).



Motion Estimation Heuristics

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

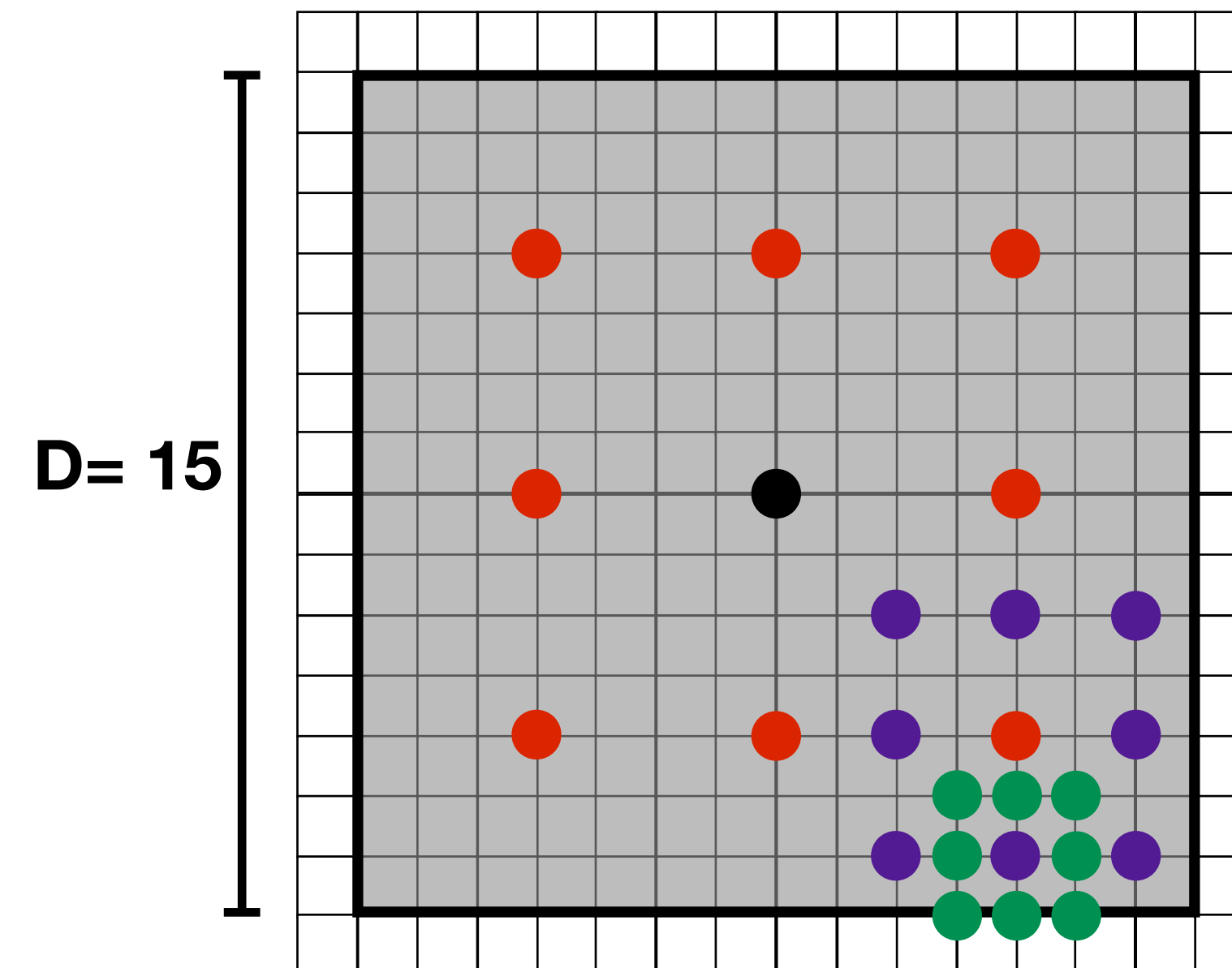
Three step search (TSS) is a classic heuristics that adaptively narrows the search range (coarse-to-fine search).



Motion Estimation Heuristics

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

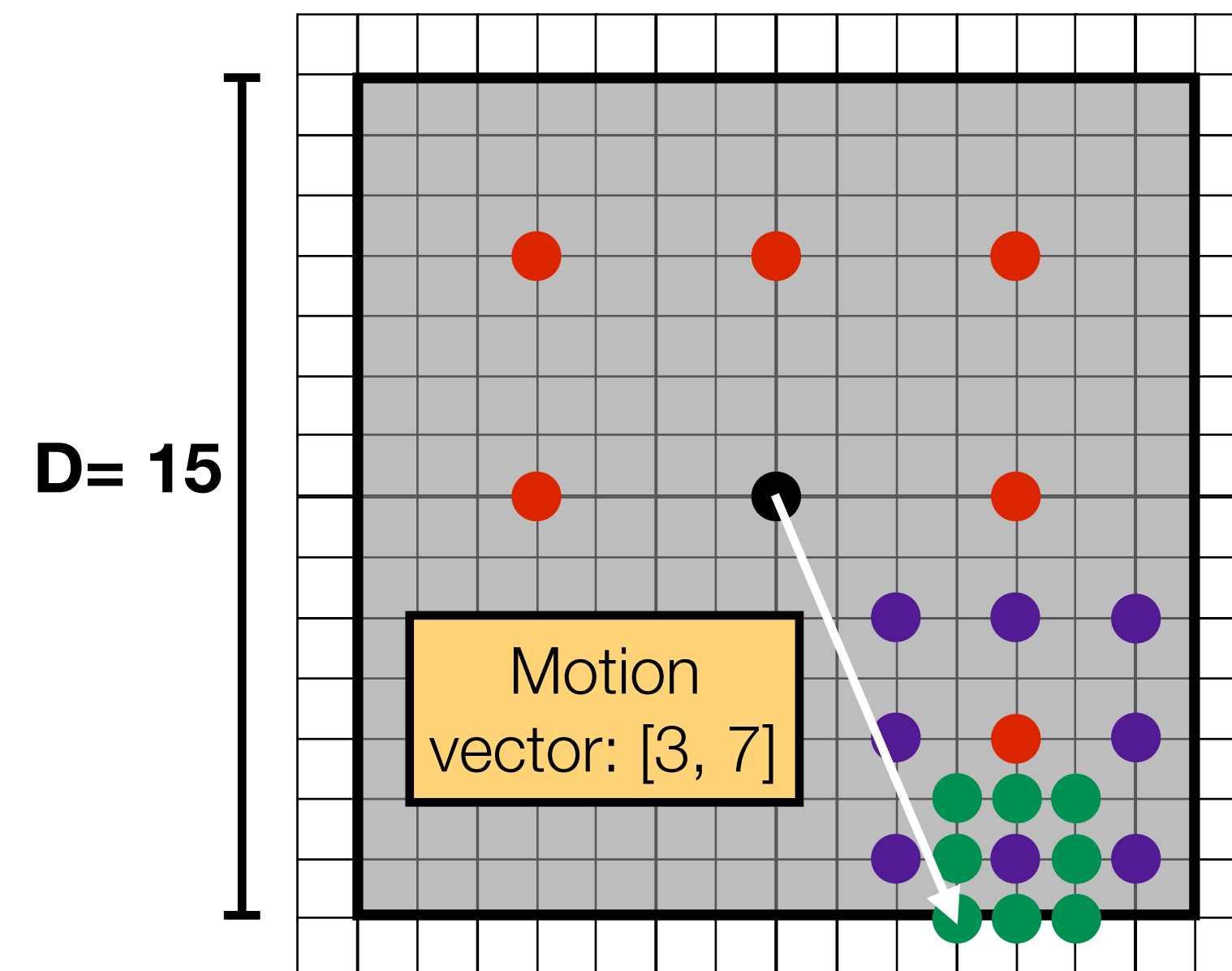
Three step search (TSS) is a classic heuristics that adaptively narrows the search range (coarse-to-fine search).



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).

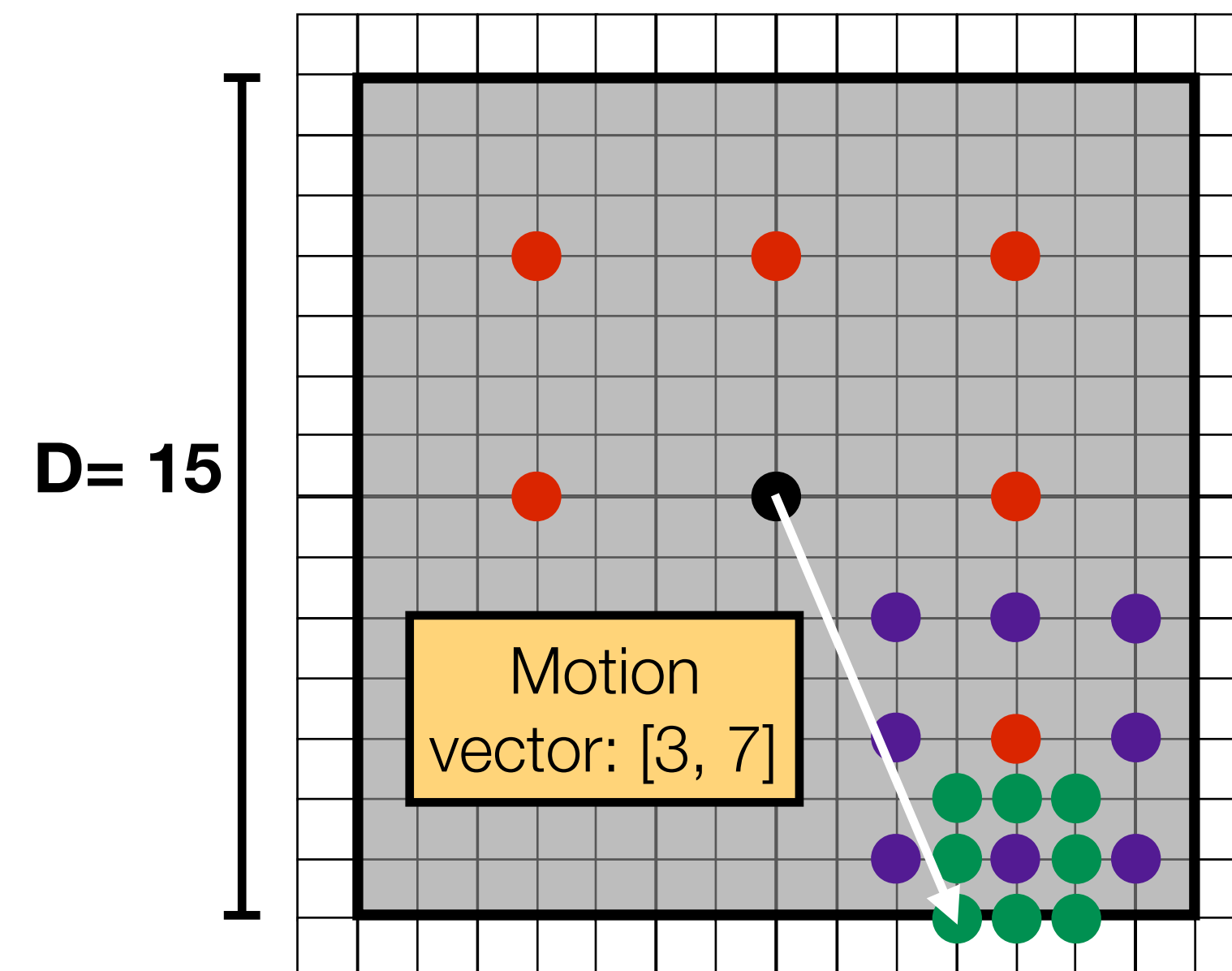


Motion Estimation Heuristics

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

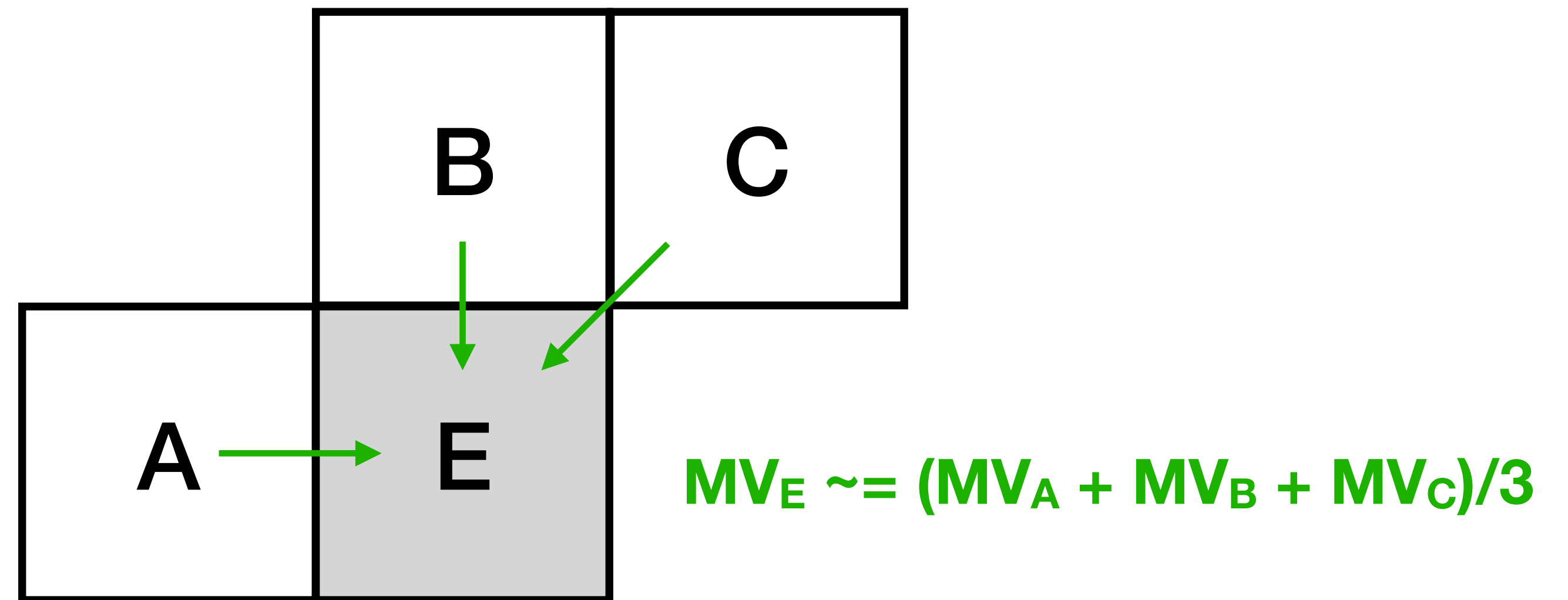
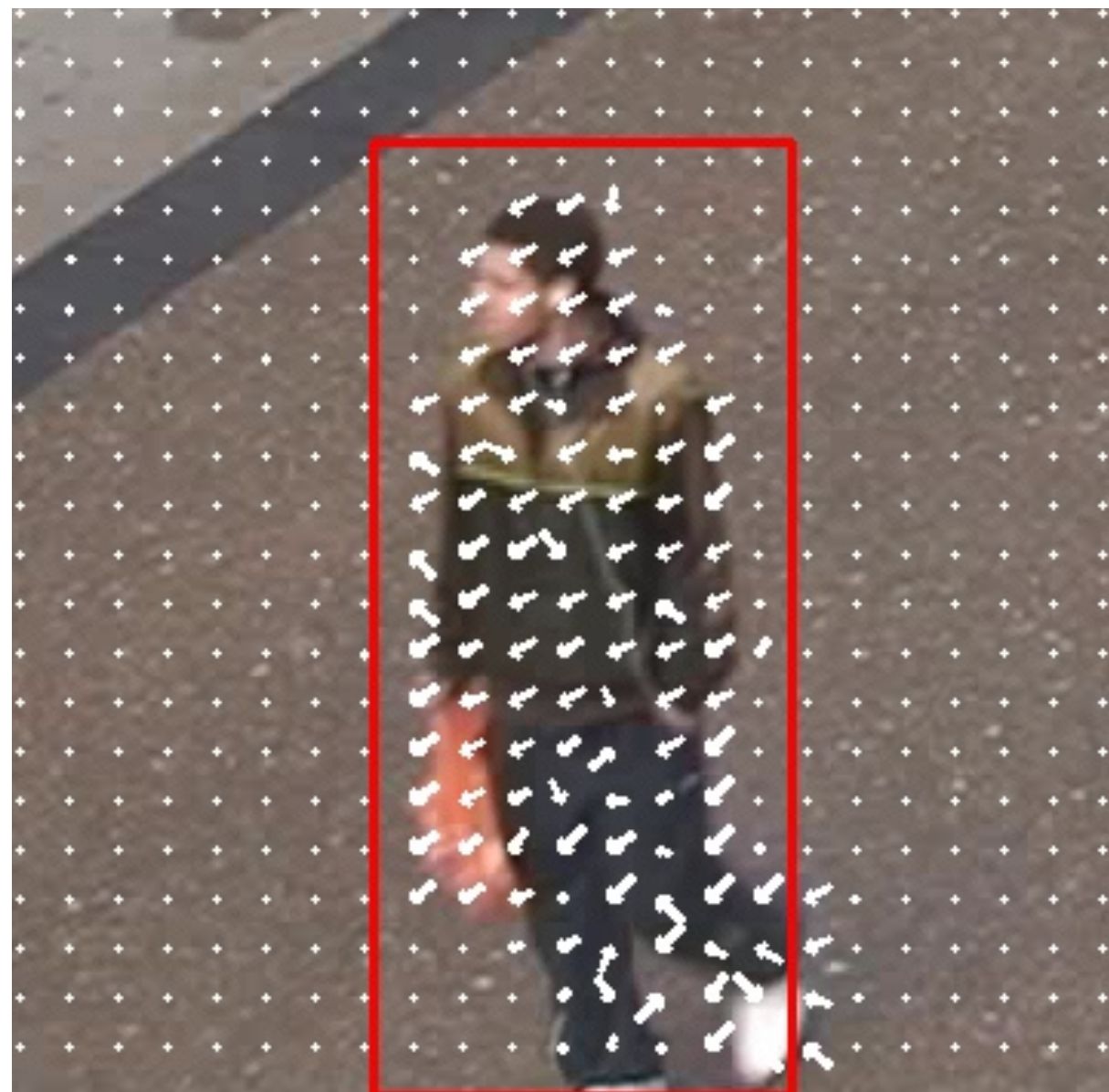
0, 1	1, 0
0, 1	1, 0

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

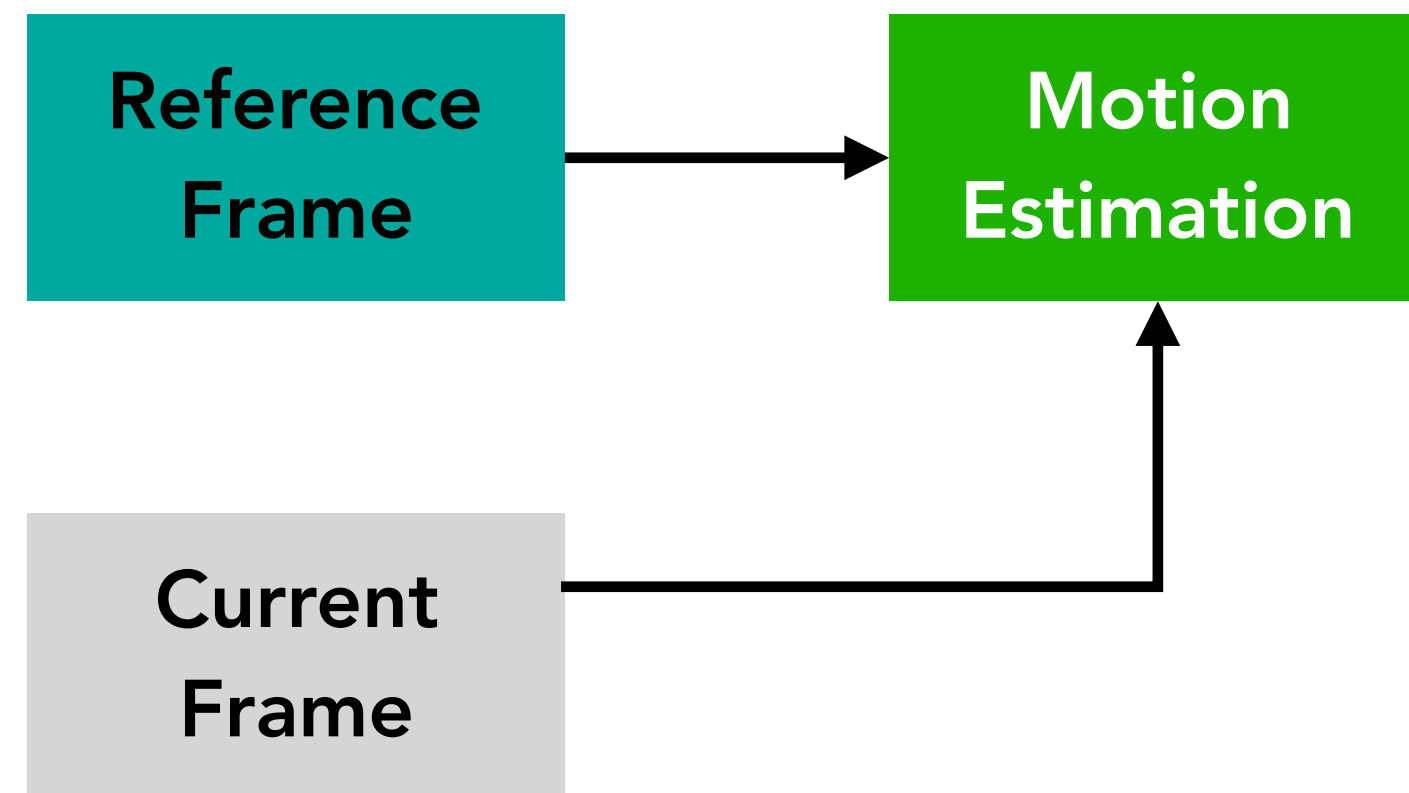
Data that needs
to be encoded

Reference
Frame

Current
Frame

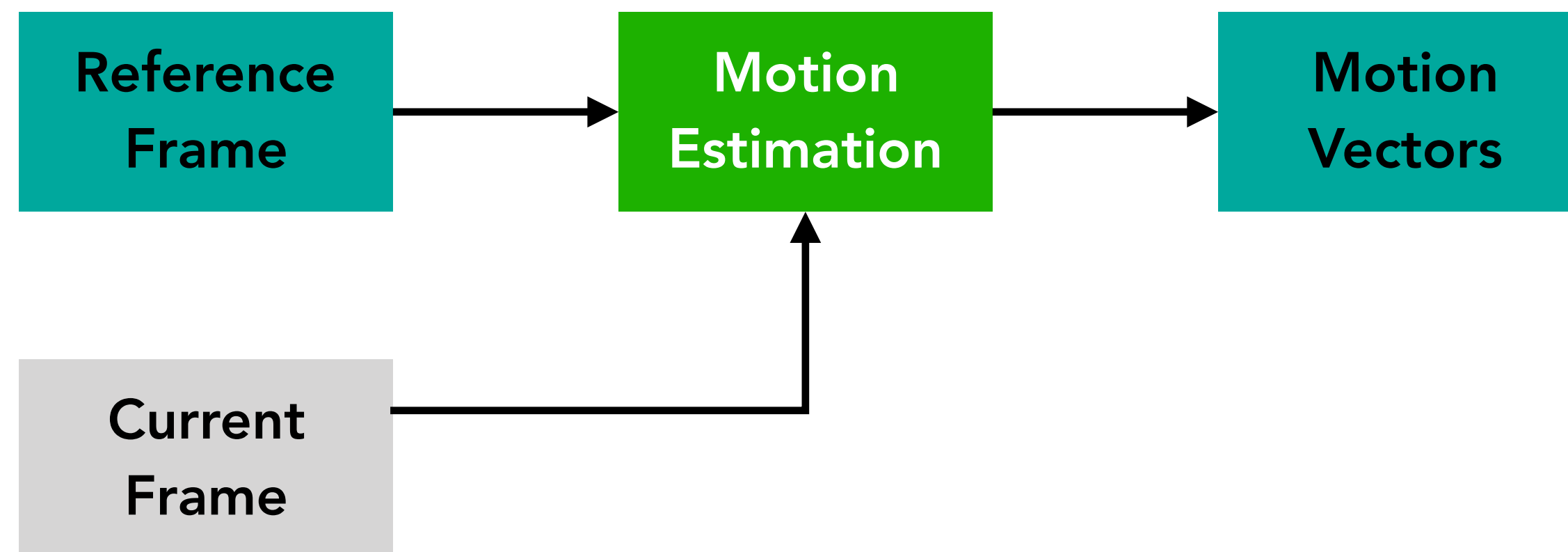
Inter-Frame Encoding Flow

Data that needs
to be encoded



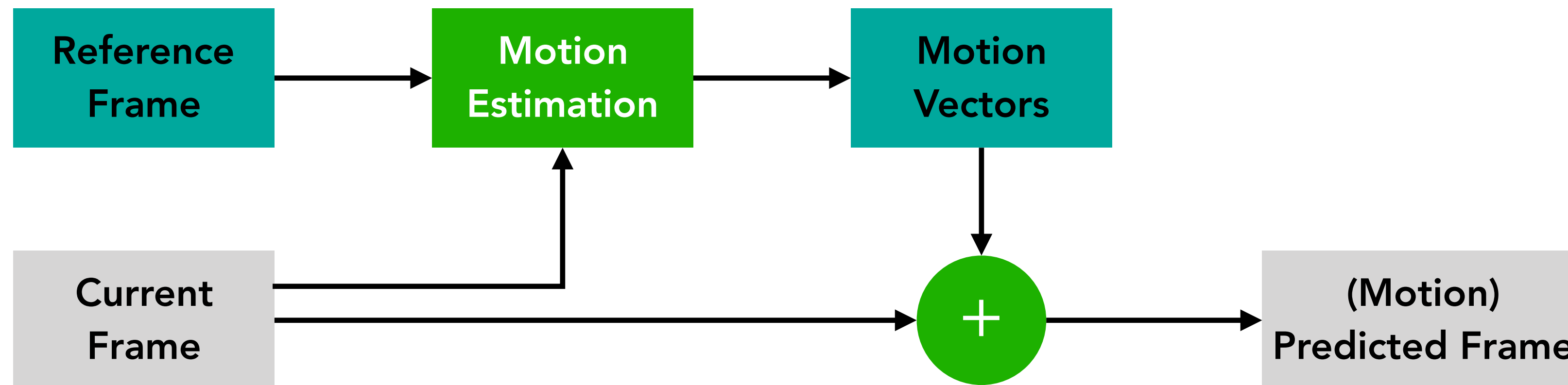
Inter-Frame Encoding Flow

Data that needs
to be encoded



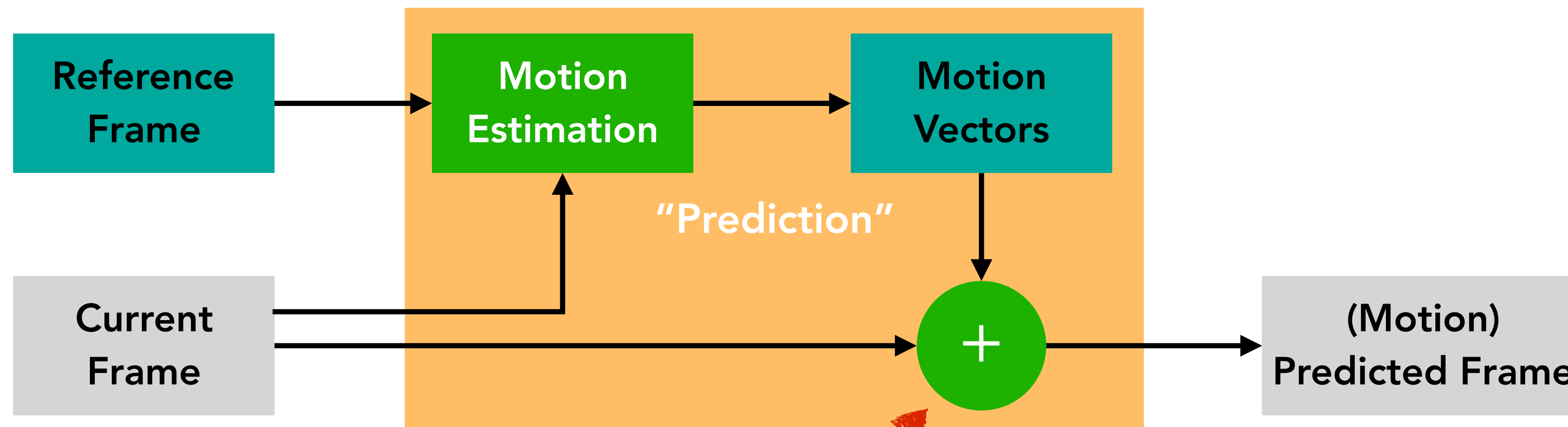
Inter-Frame Encoding Flow

Data that needs
to be encoded



Inter-Frame Encoding Flow

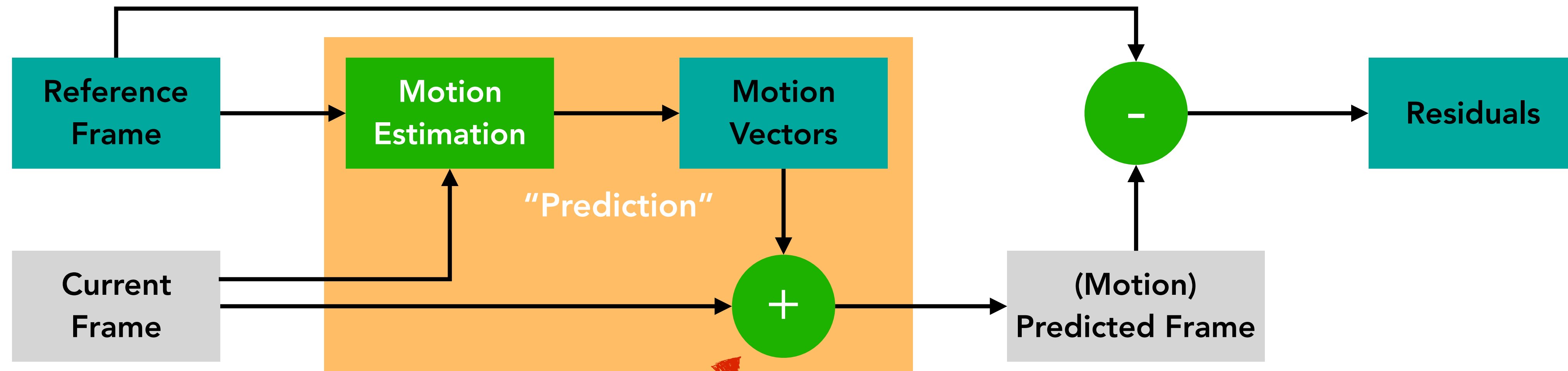
Data that needs
to be encoded



A fancy name is "motion
compensated interpolation"

Inter-Frame Encoding Flow

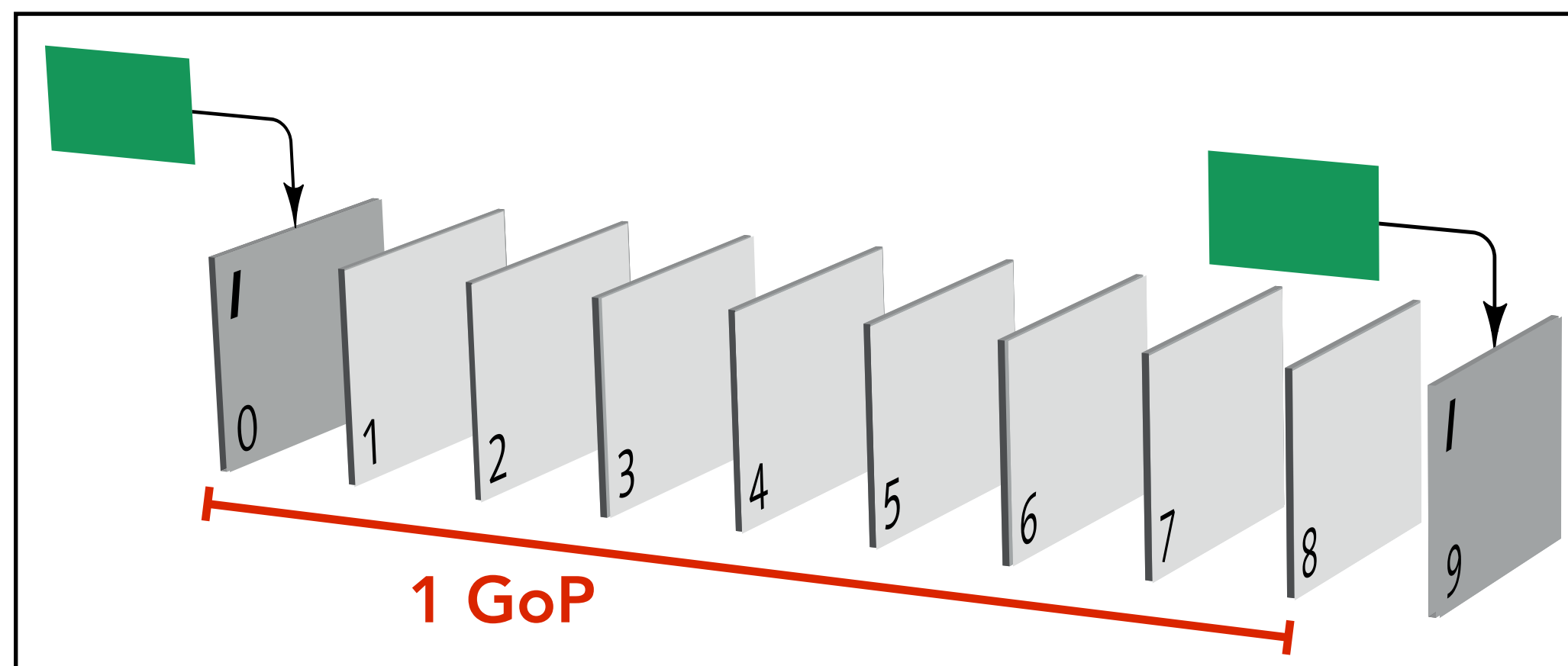
Data that needs
to be encoded



A fancy name is "motion compensated interpolation"

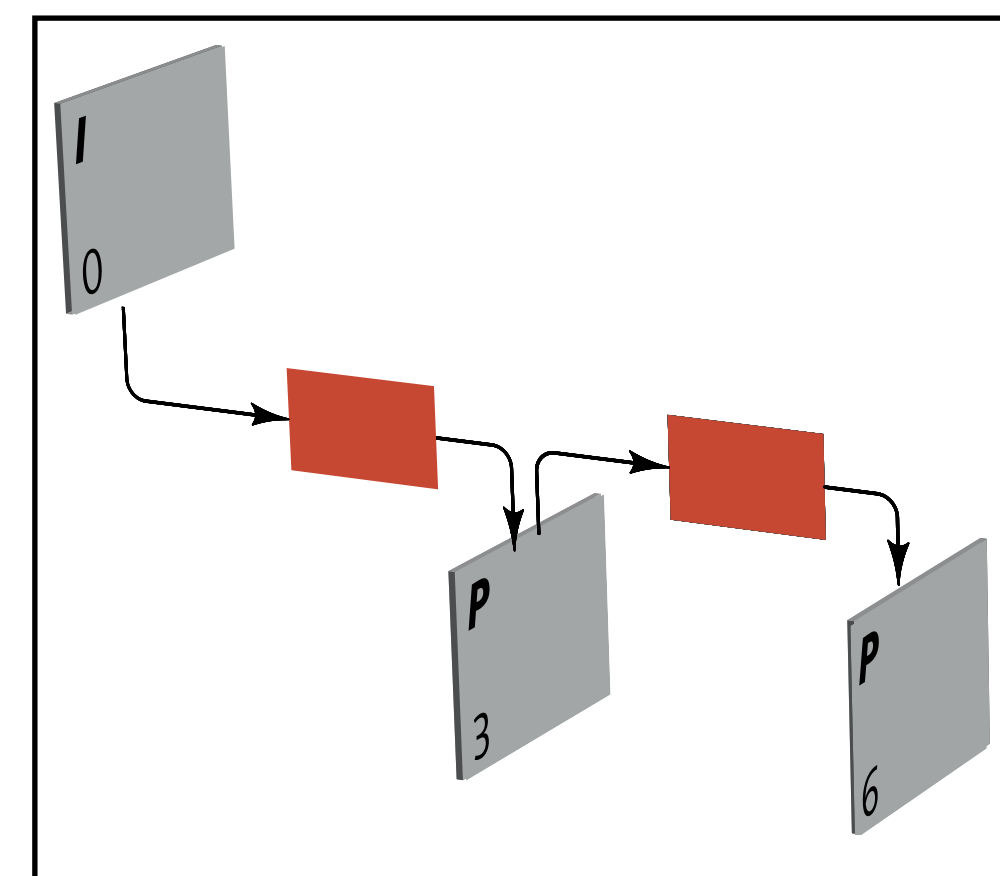
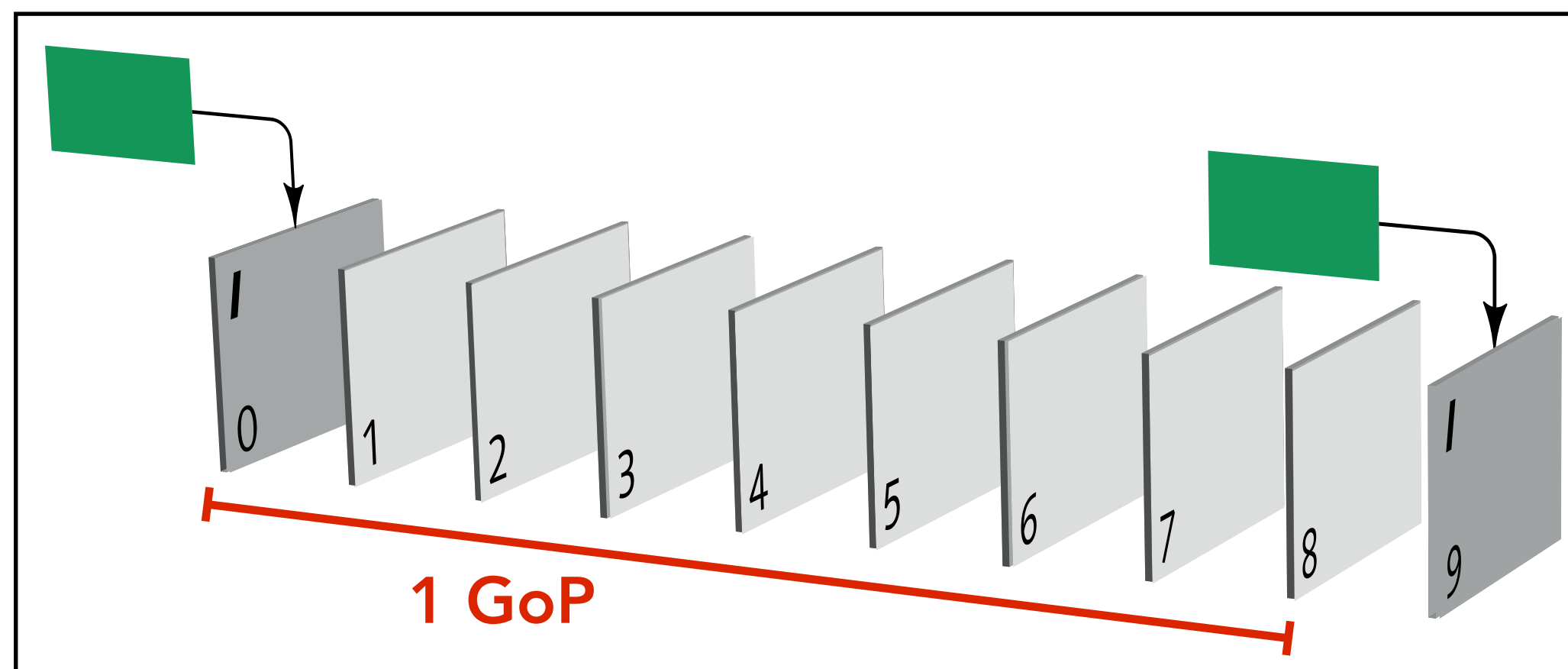
Reference Frame Choice

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



Reference Frame Choice

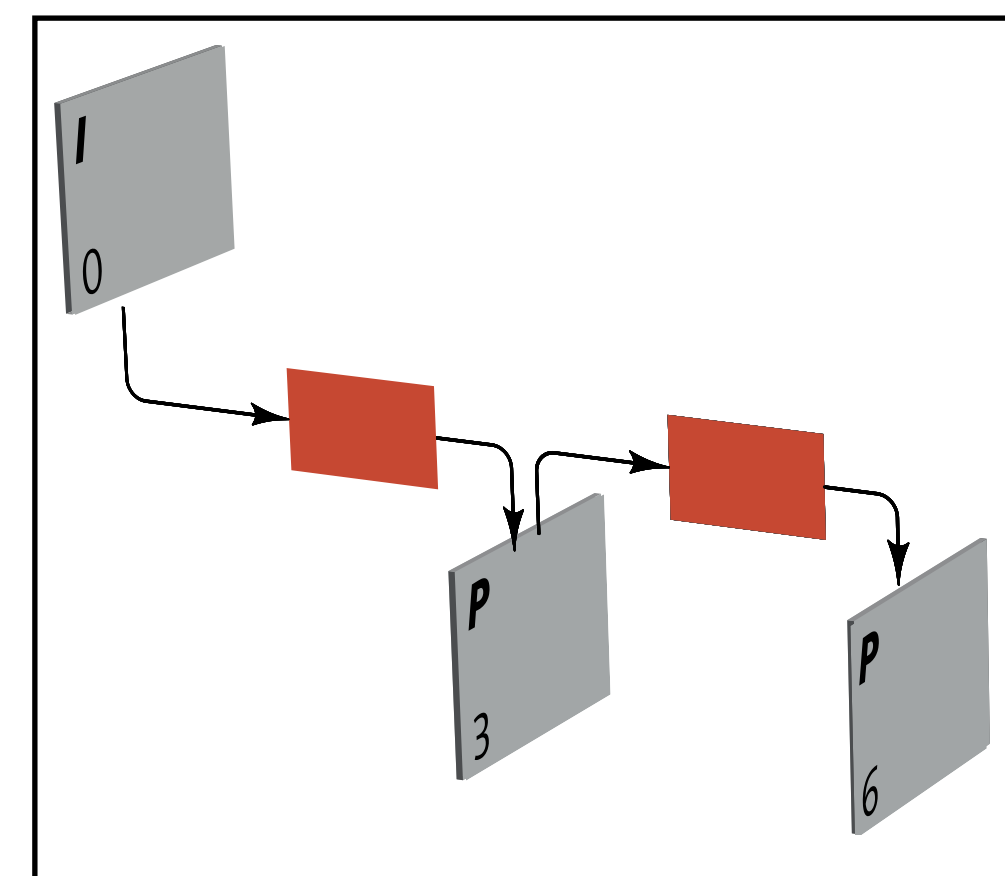
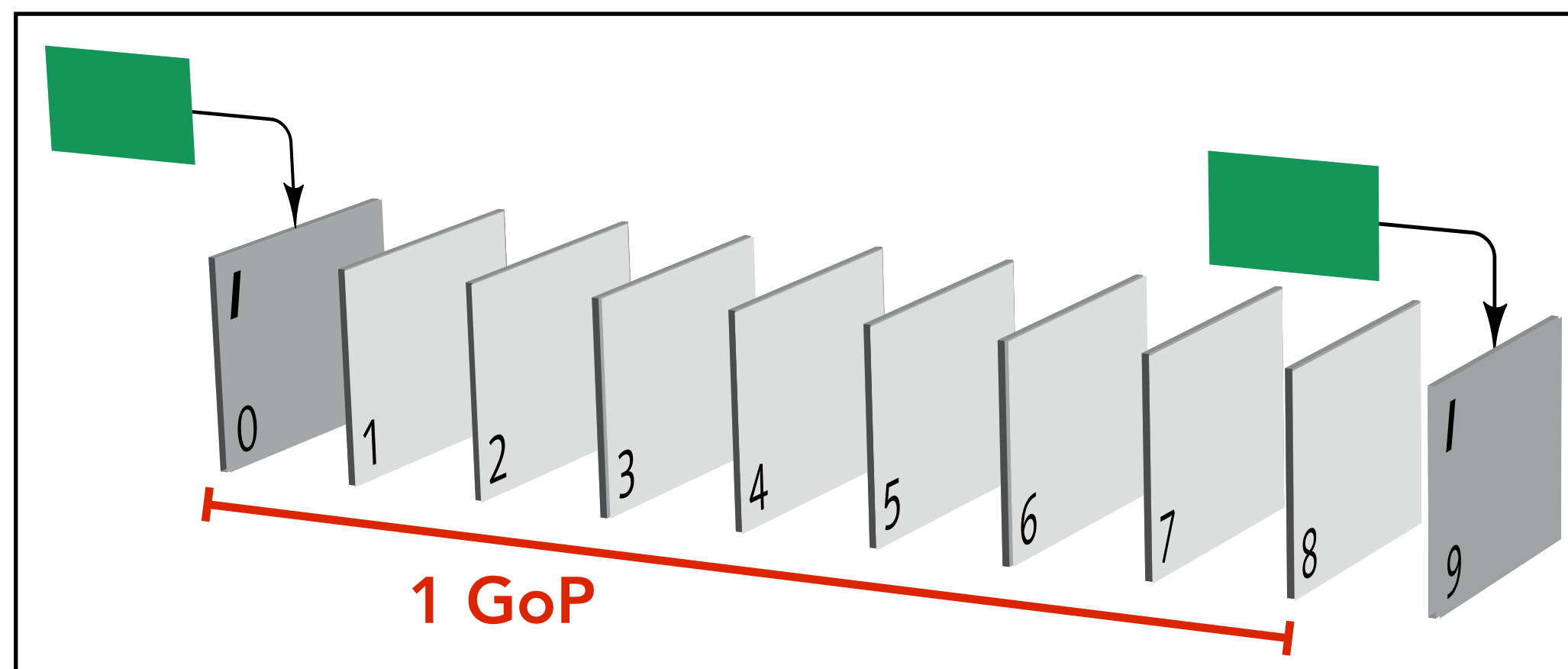
Each GoP has three (mutually exclusive) kinds of frames:



Reference Frame Choice

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

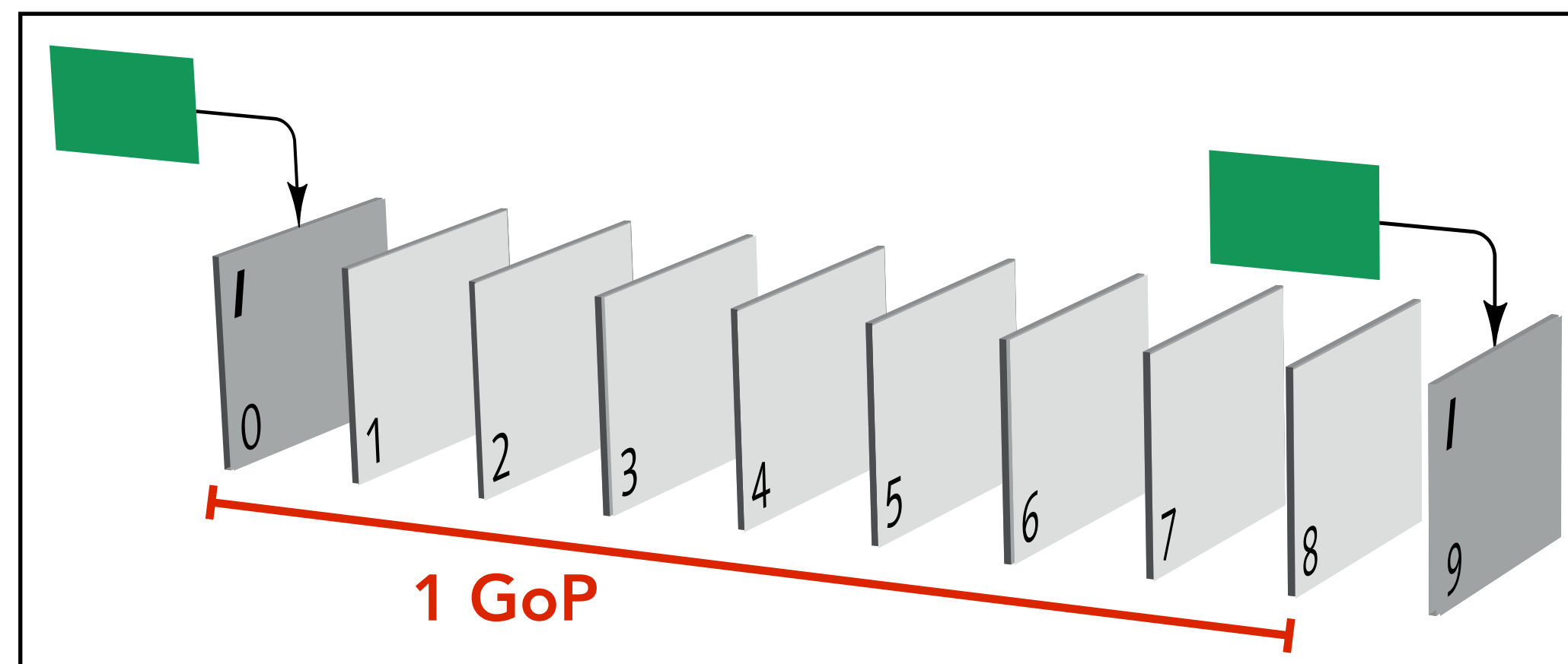
- An **I-frame** is intra-compressed and is a reference frame.



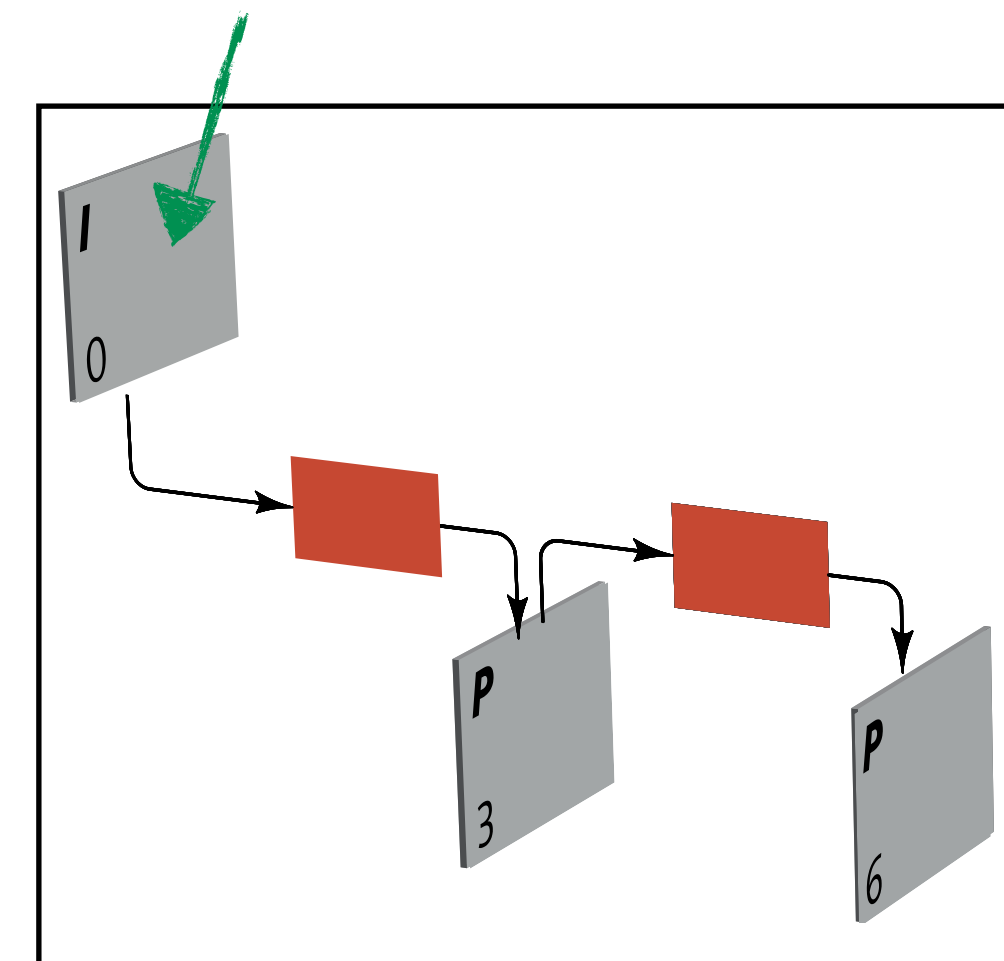
Reference Frame Choice

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

- An **I-frame** is intra-compressed and is a reference frame.



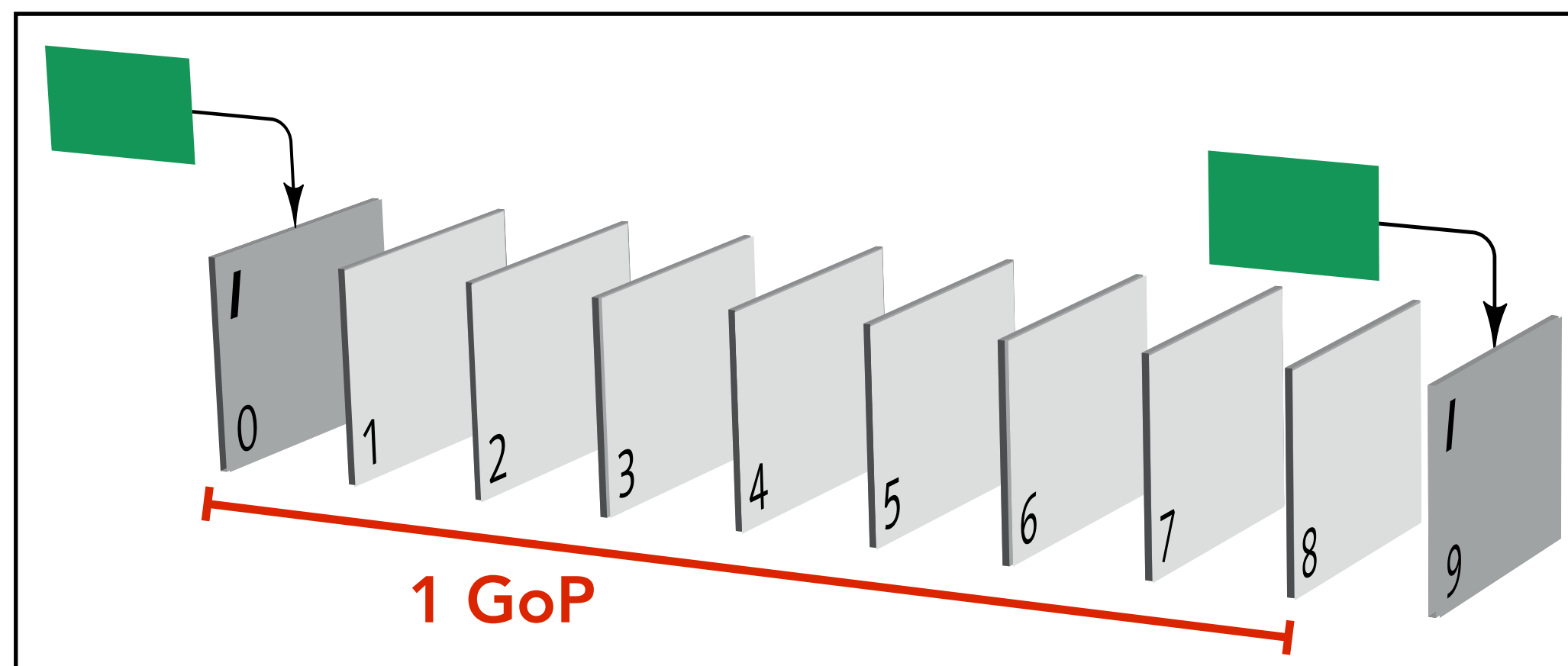
The I-frame



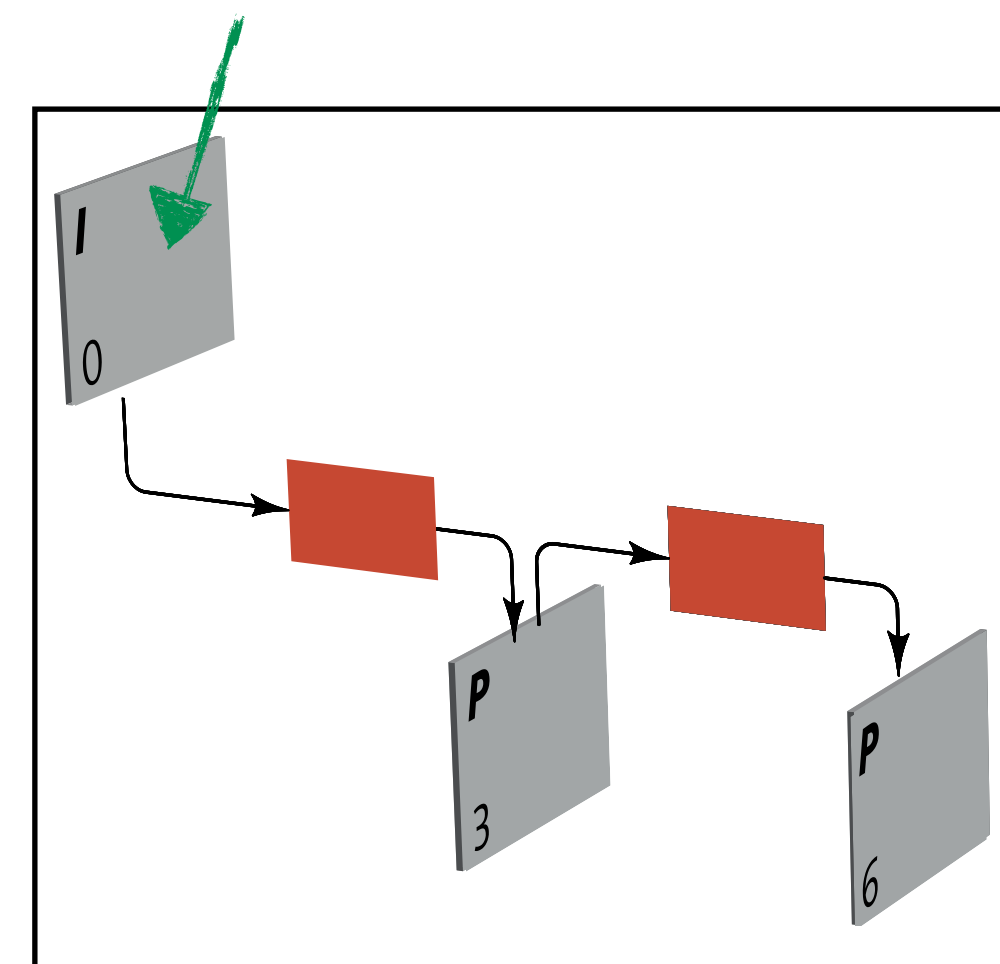
Reference Frame Choice

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.



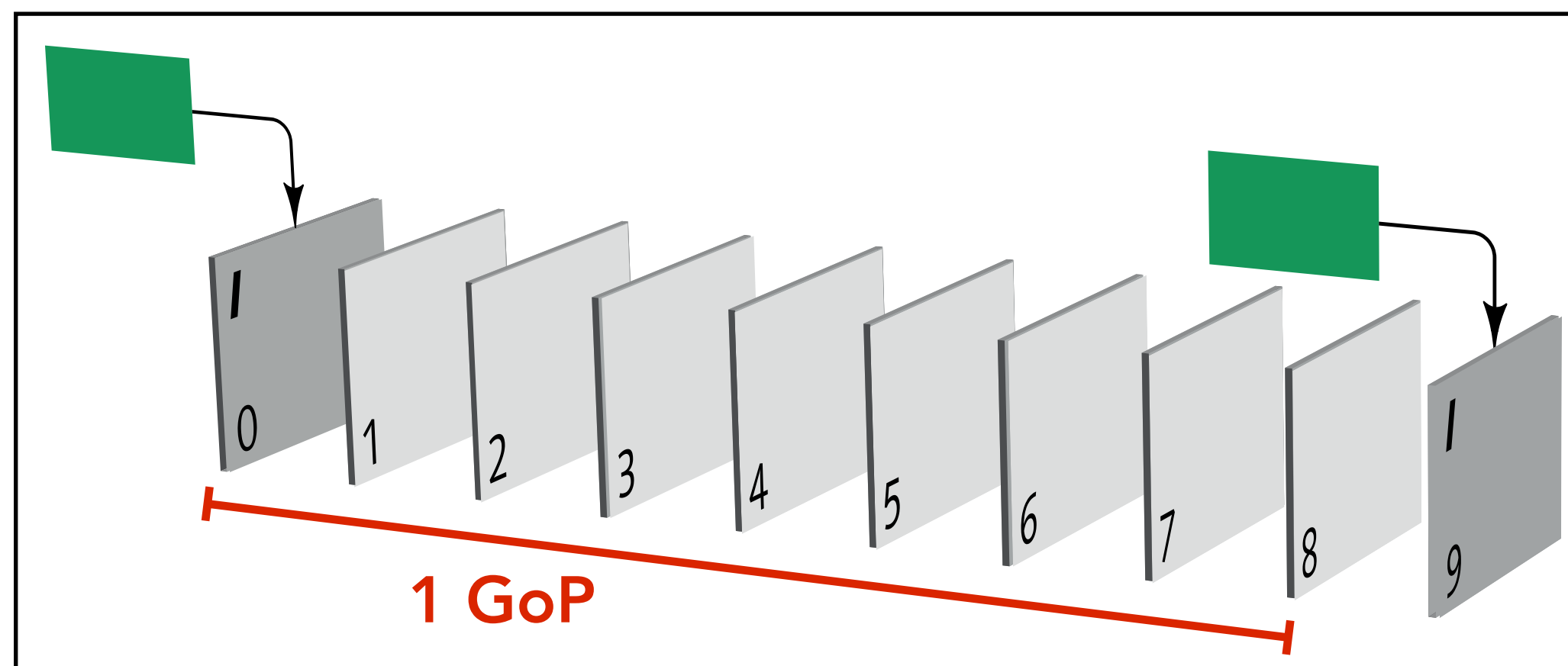
The I-frame



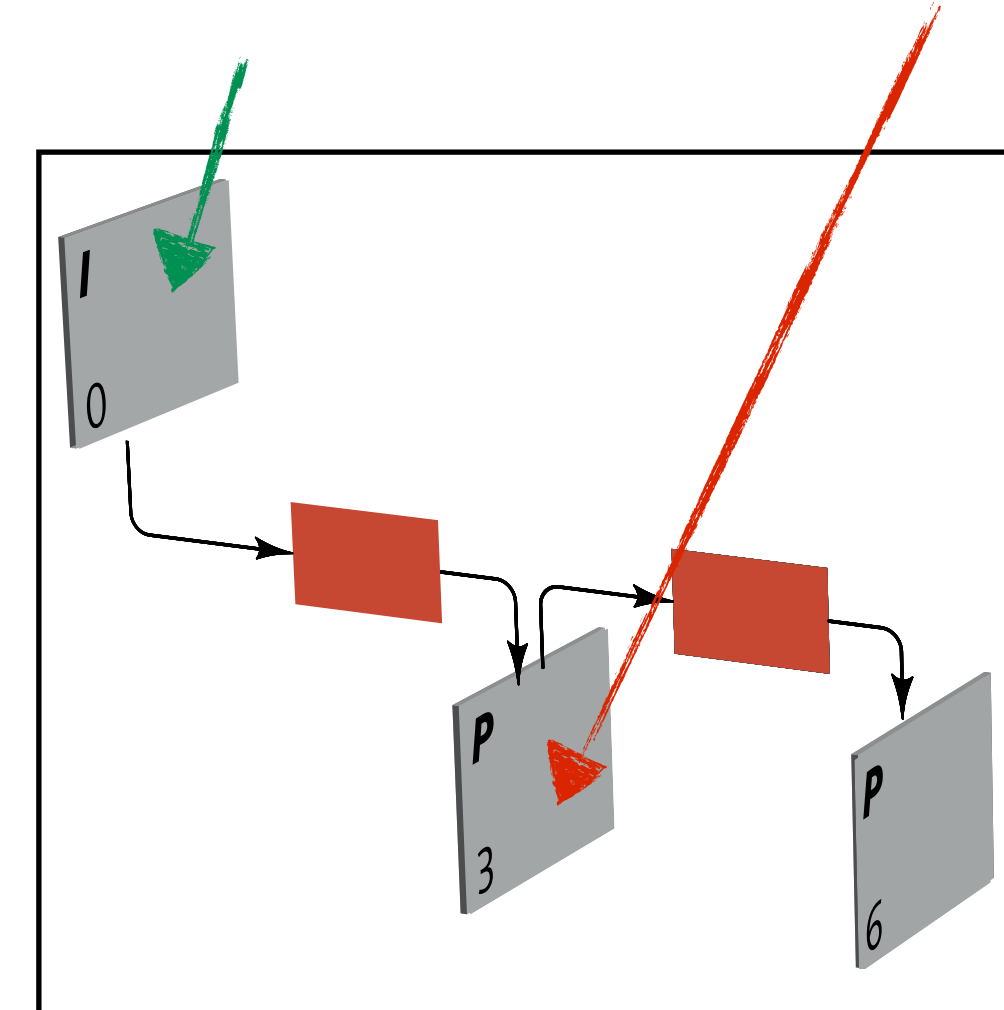
Reference Frame Choice

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

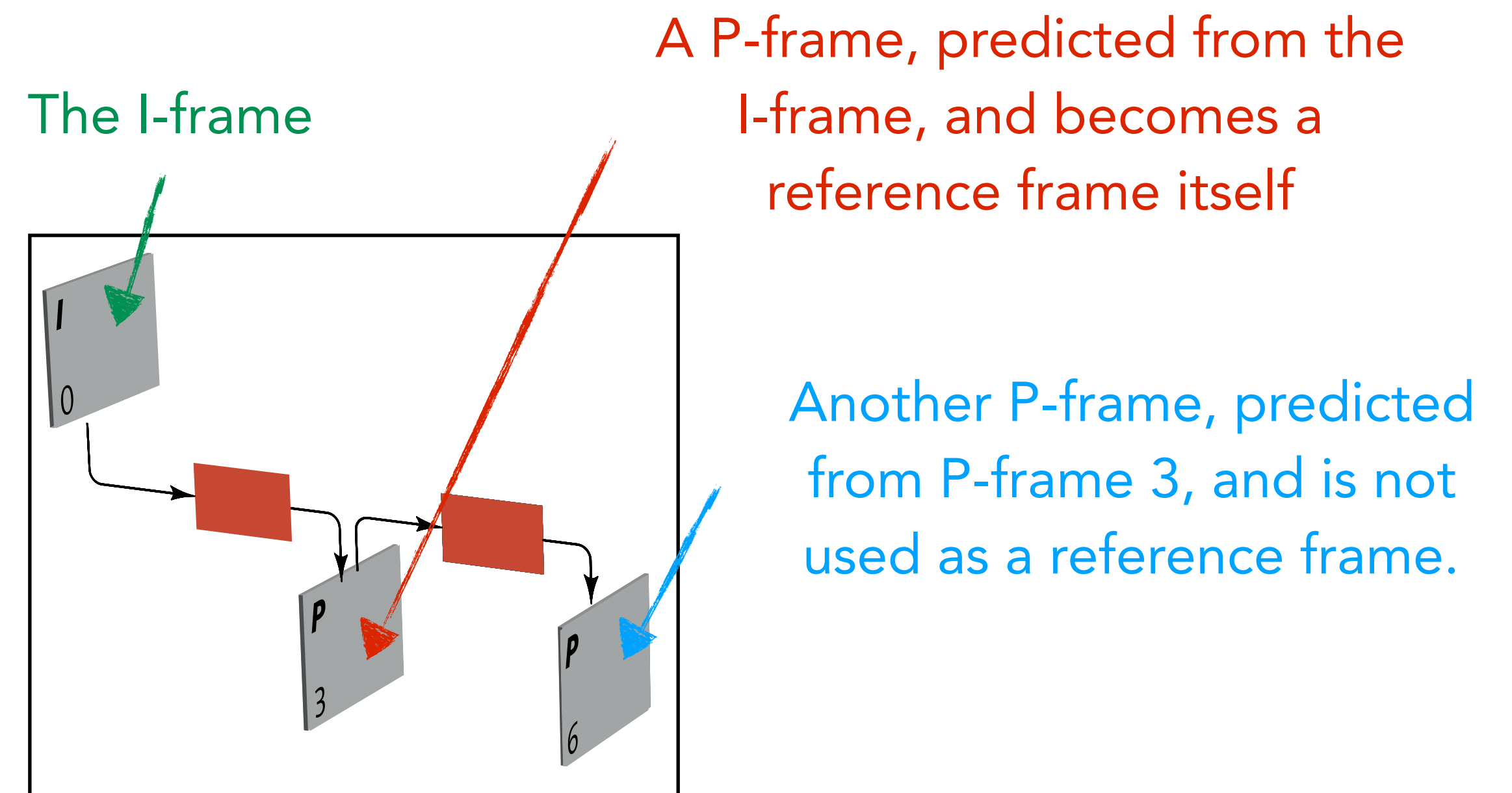
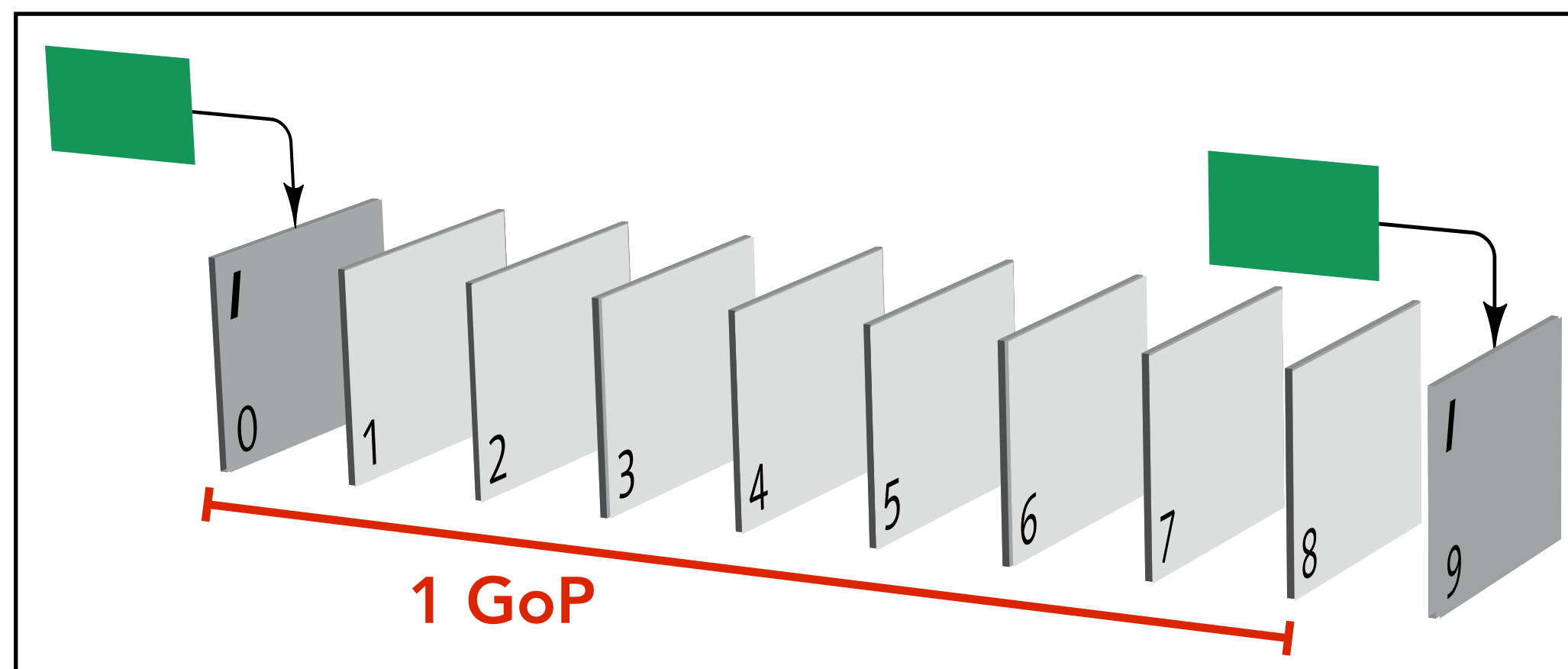


A P-frame, predicted from the I-frame, and becomes a reference frame itself

Reference Frame Choice

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

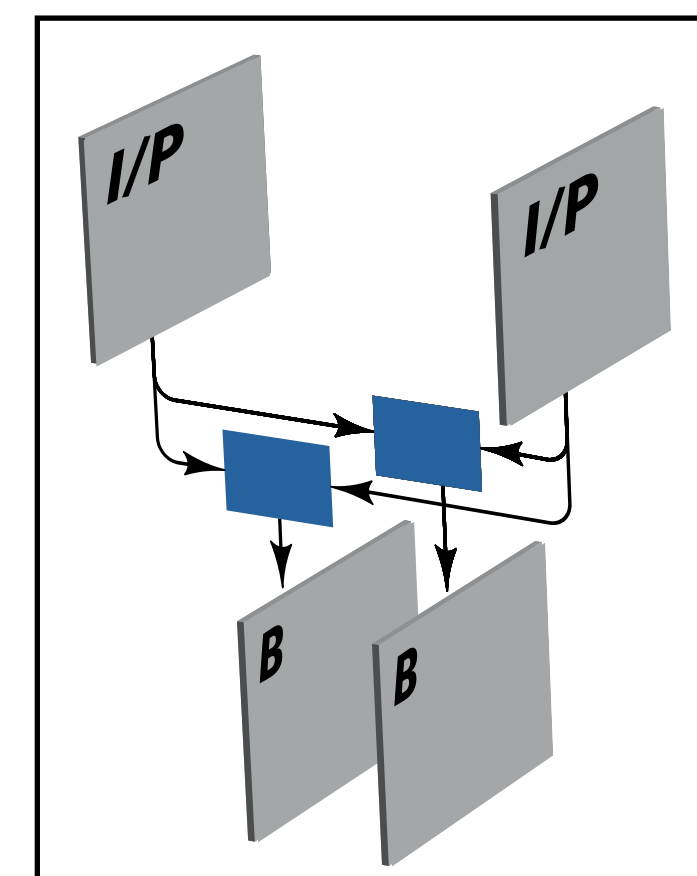
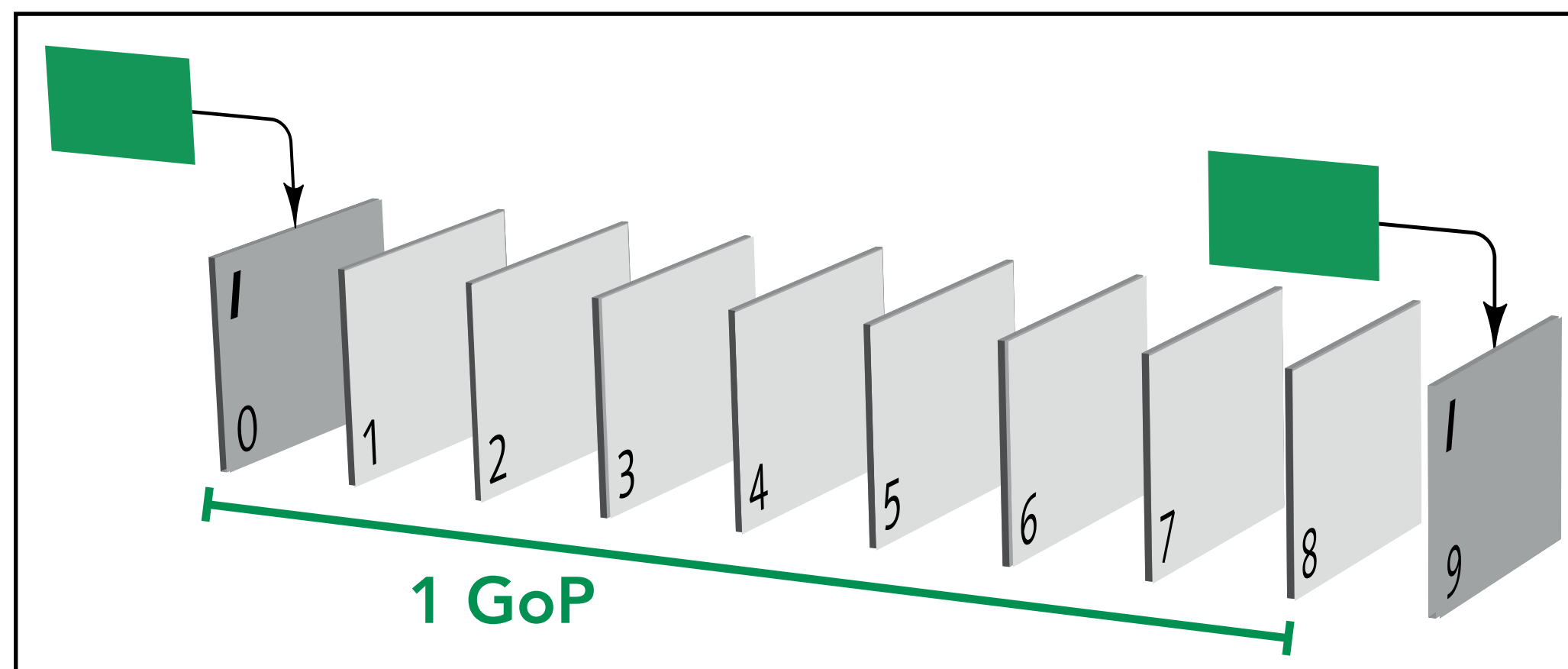
- An **I-frame** is intra-compressed and is a reference frame.
- A **P-frame** is a frame that is predicted from a reference frame, and itself can be used as a reference frame.



Reference Frame Choice

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

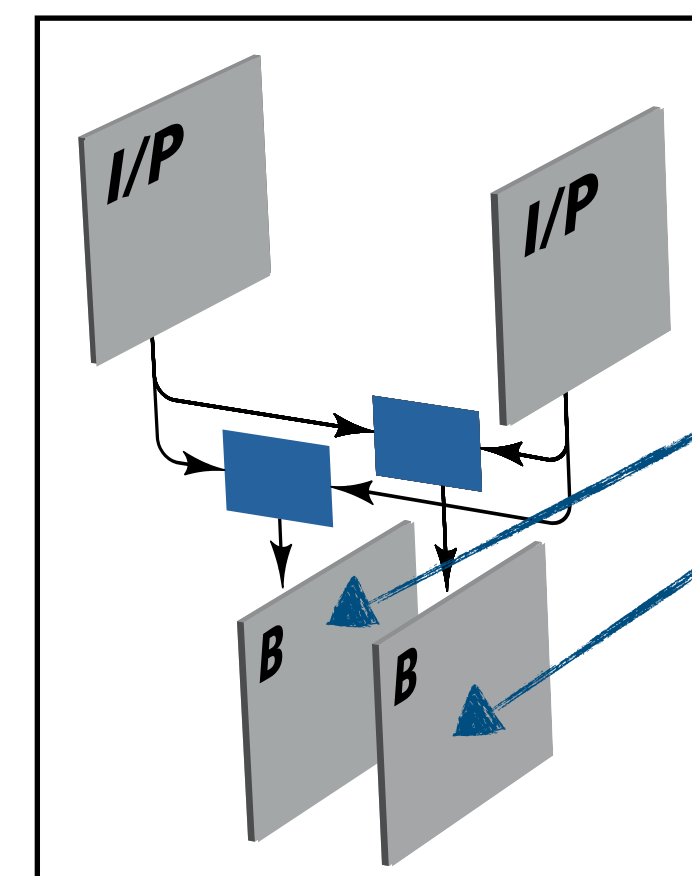
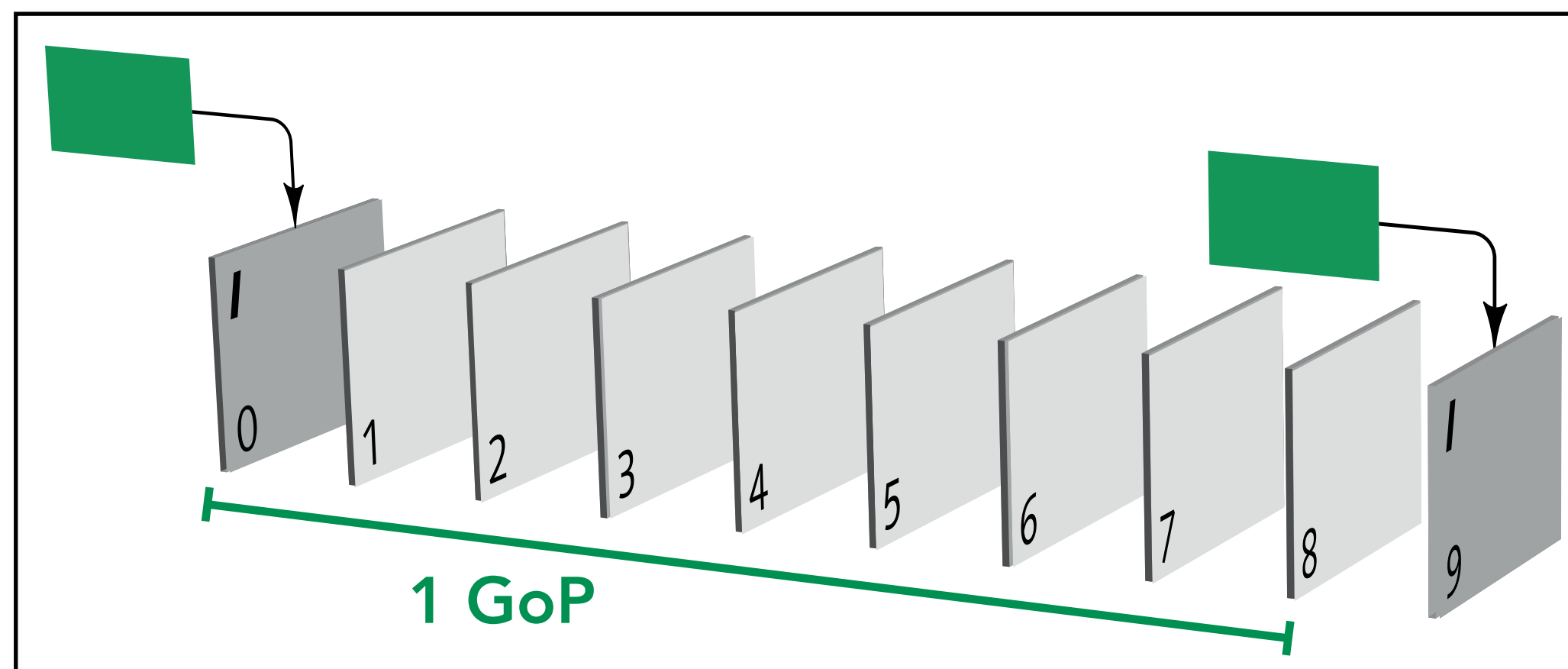
- An **I-frame** is intra-compressed and is a reference frame.
- A **P-frame** is a frame that is predicted from a reference frame, and itself can be used as a reference frame.
- A **B-frame** is a frame that is predicted from two reference frames (because the residuals are smaller that way), but itself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



Reference Frame Choice

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

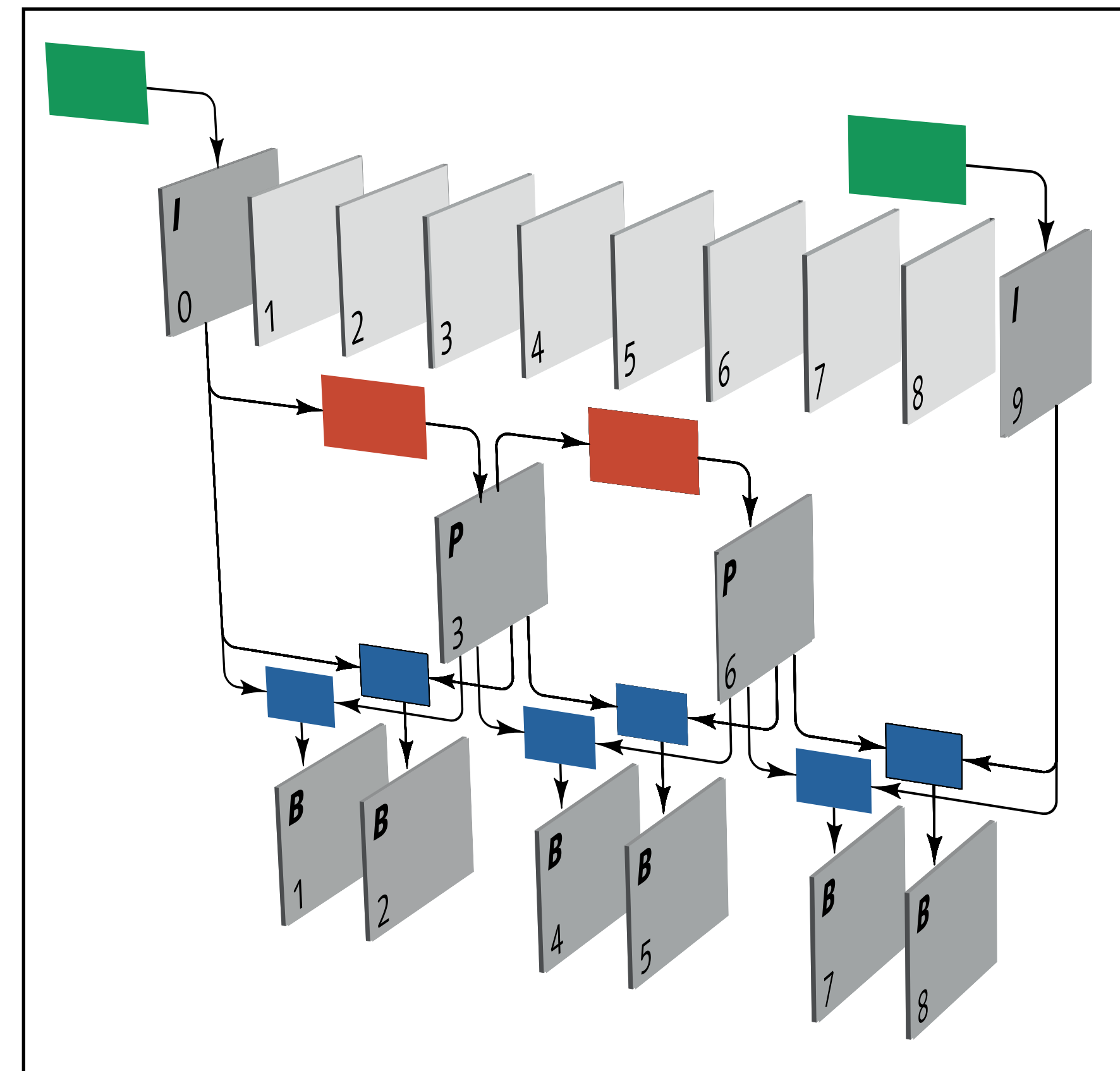
- An **I-frame** is intra-compressed and is a reference frame.
- A **P-frame** is a frame that is predicted from a reference frame, and itself can be used as a reference frame.
- A **B-frame** is a frame that is predicted from two reference frames (because the residuals are smaller that way), but itself 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).

Reordering Frames in a GoP

$I_0 B_1 B_2 P_3 B_4 B_5 P_6 B_7 B_8$

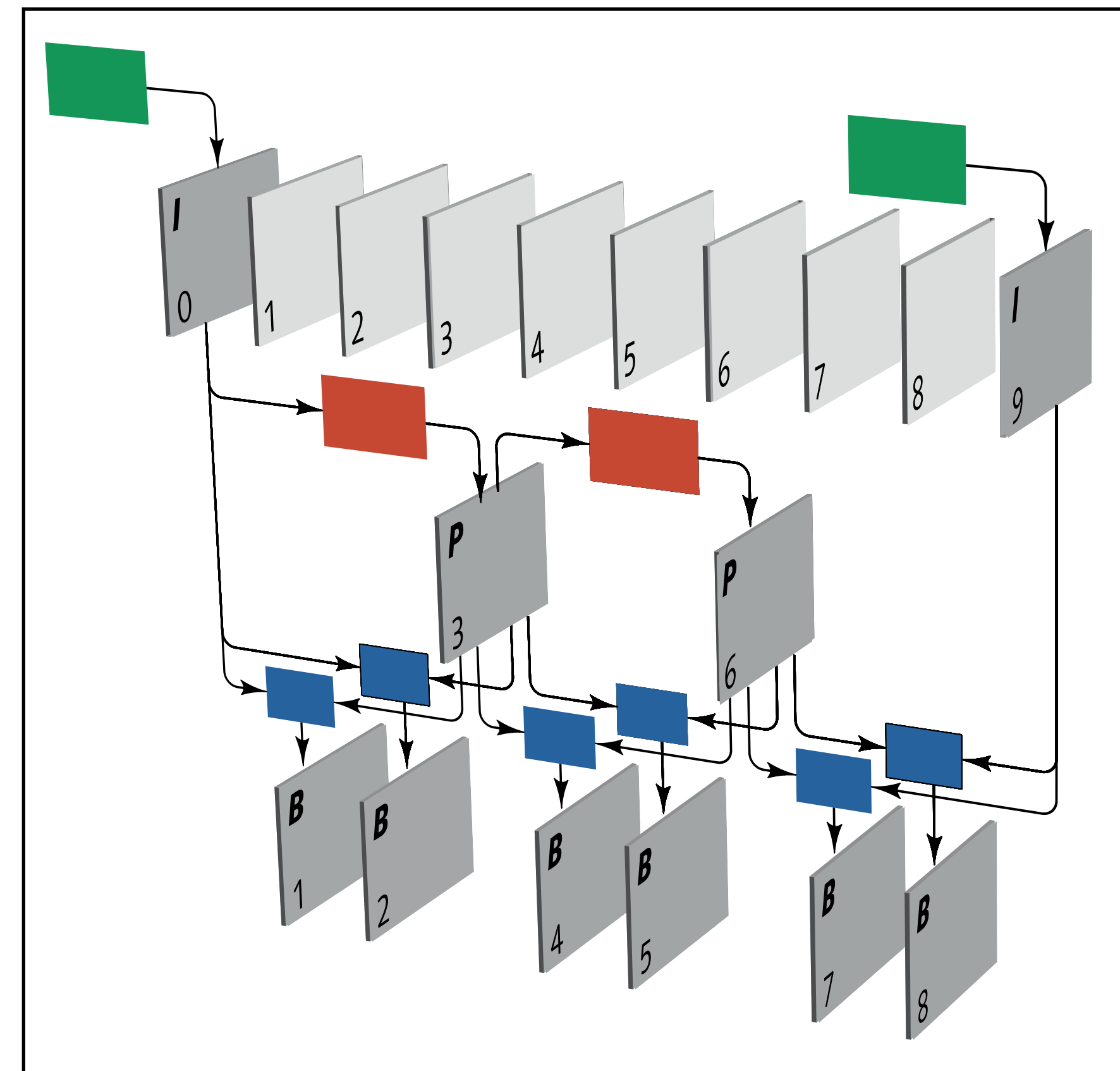


Reordering Frames in a GoP

$I_0 B_1 B_2 P_3 B_4 B_5 P_6 B_7 B_8$

Bad for live streaming/teleconferencing!!!

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



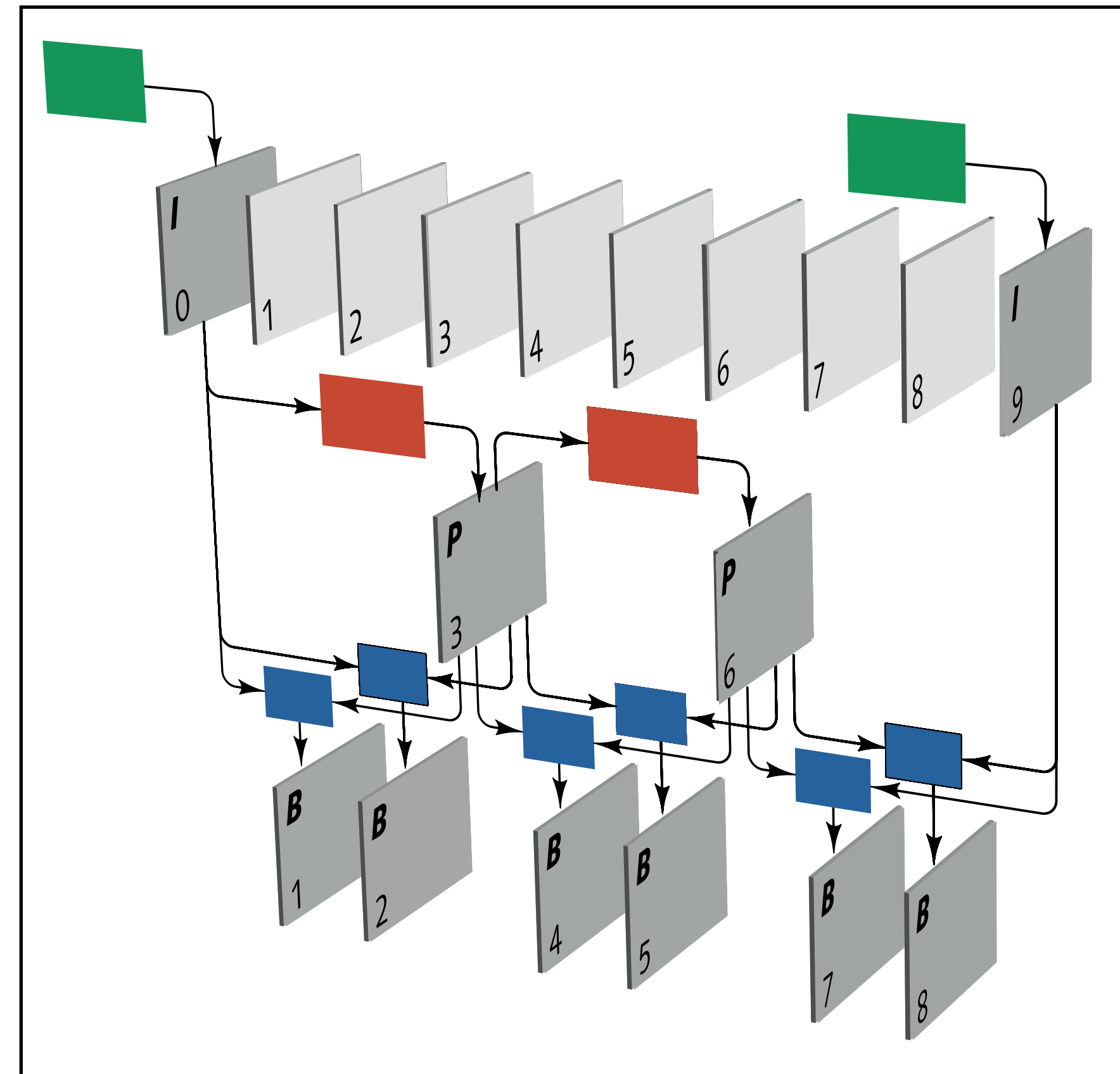
Reordering Frames in a GoP

$I_0 B_1 B_2 P_3 B_4 B_5 P_6 B_7 B_8$

Bad for live streaming/teleconferencing!!!

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

$I_0 P_3 B_1 B_2 P_6 B_4 B_5 (I_9) B_7 B_8$



Reordering Frames in a GoP

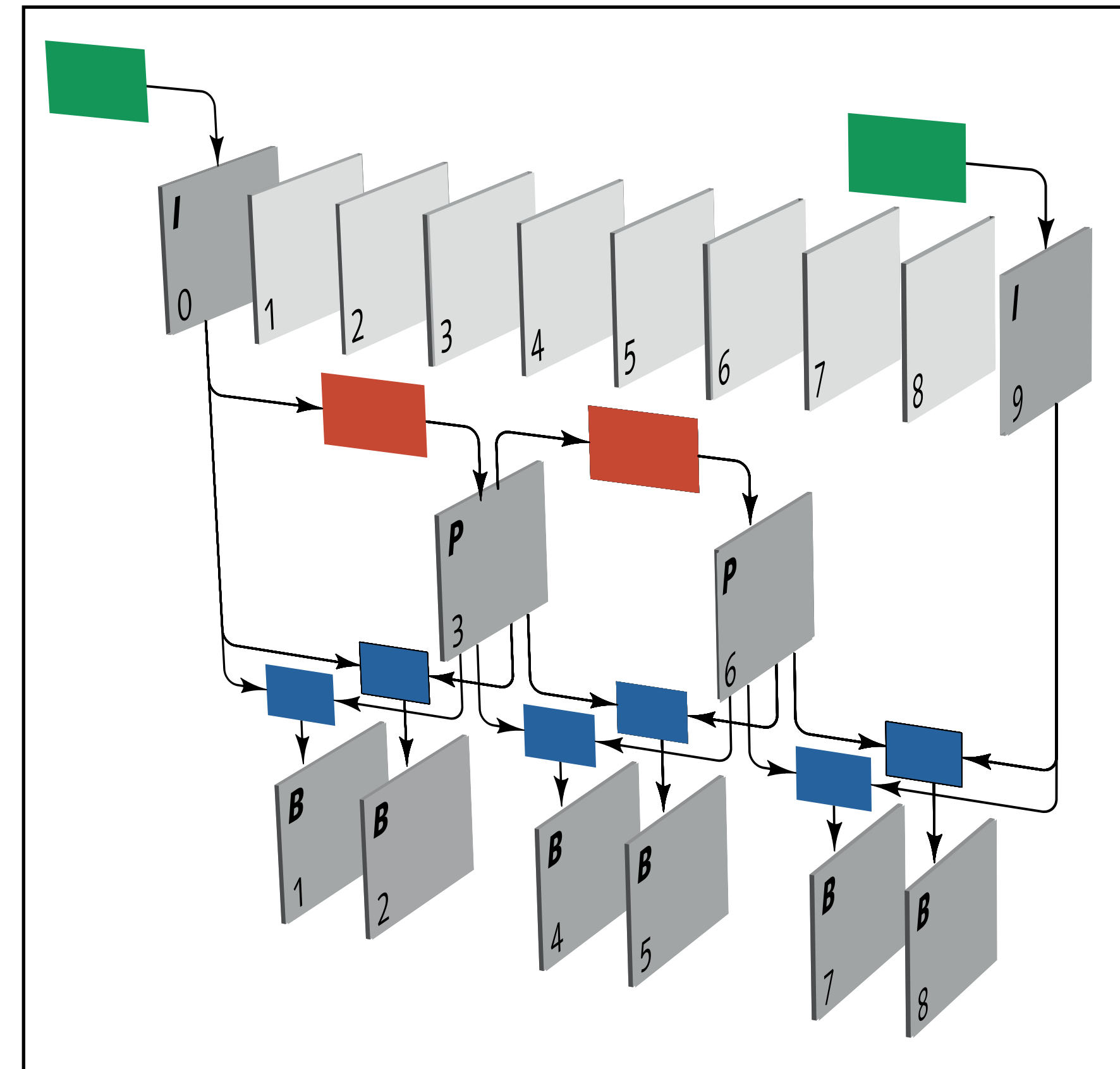
$I_0 B_1 B_2 P_3 B_4 B_5 P_6 B_7 B_8$

Bad for live streaming/teleconferencing!!!

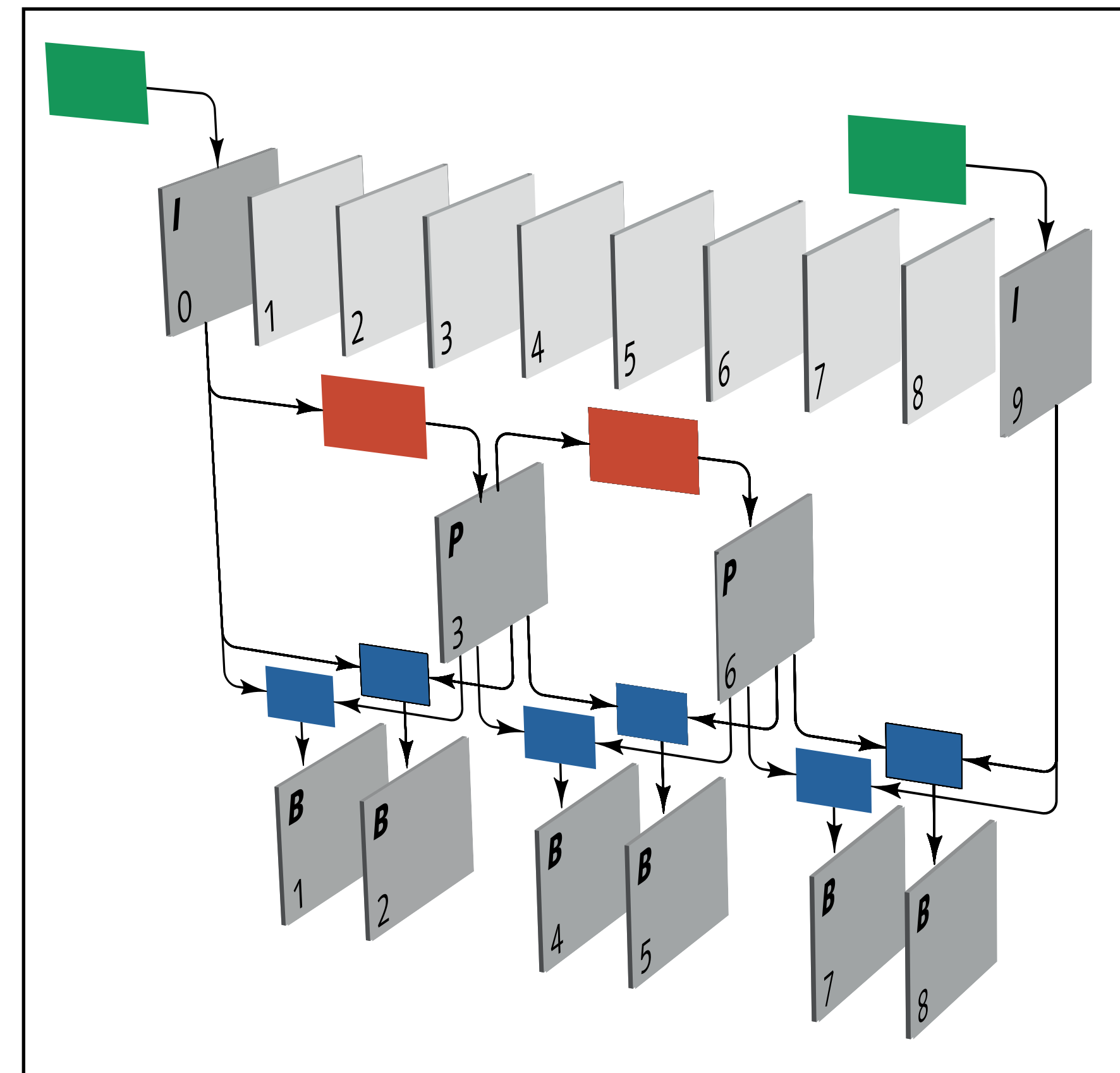
1. When encoding, need to buffer B_1 and B_2 before P_3 arrives (e.g., from camera).
2. When decoding, need to buffer B_1 and B_2 before P_3 arrives (e.g., from Internet).

$I_0 P_3 B_1 B_2 P_6 B_4 B_5 (I_9) B_7 B_8$

Solution: reorder frames during transmission!

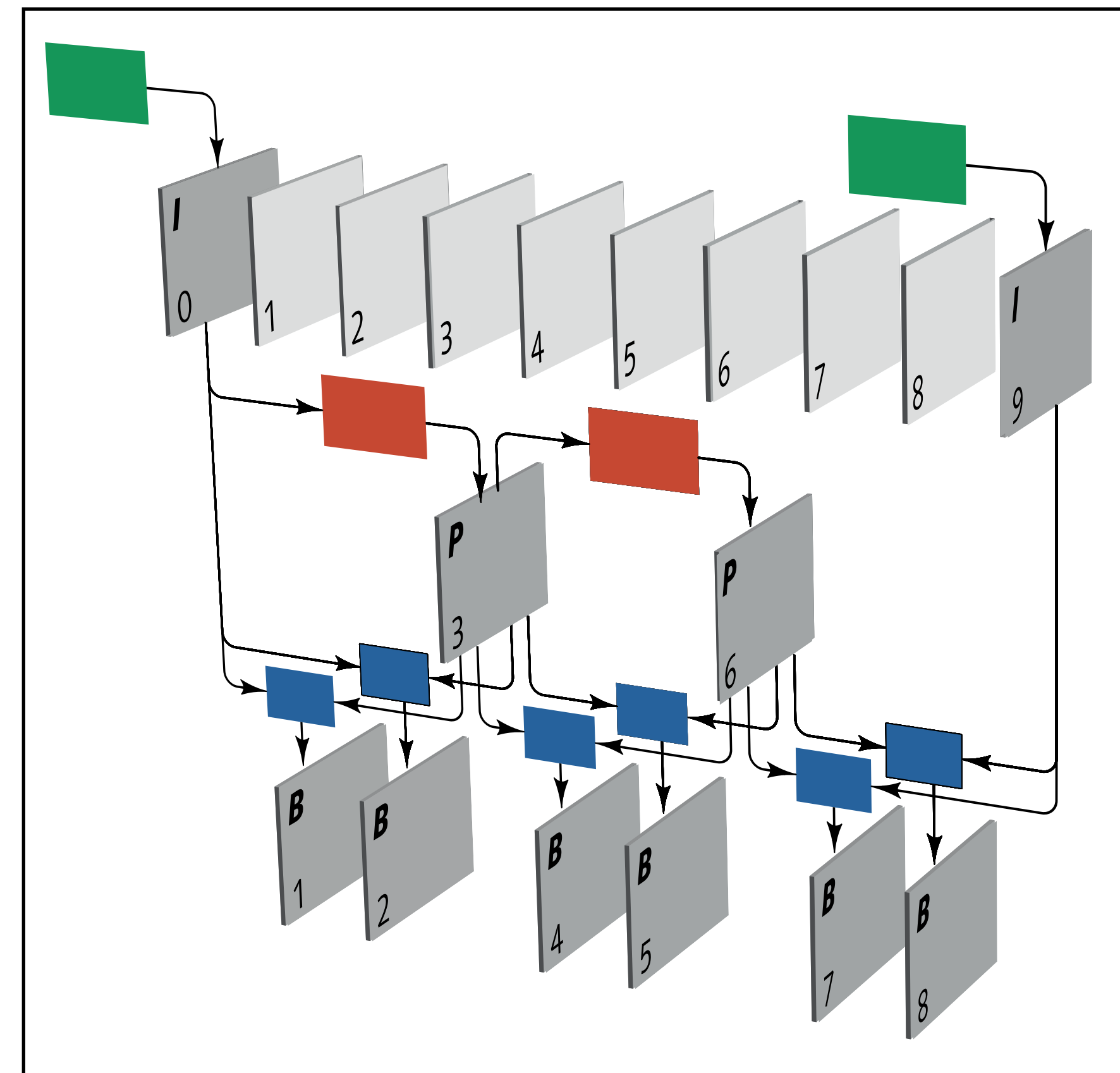


Incorporating the Decoder in the Encoder



Incorporating the Decoder in the Encoder

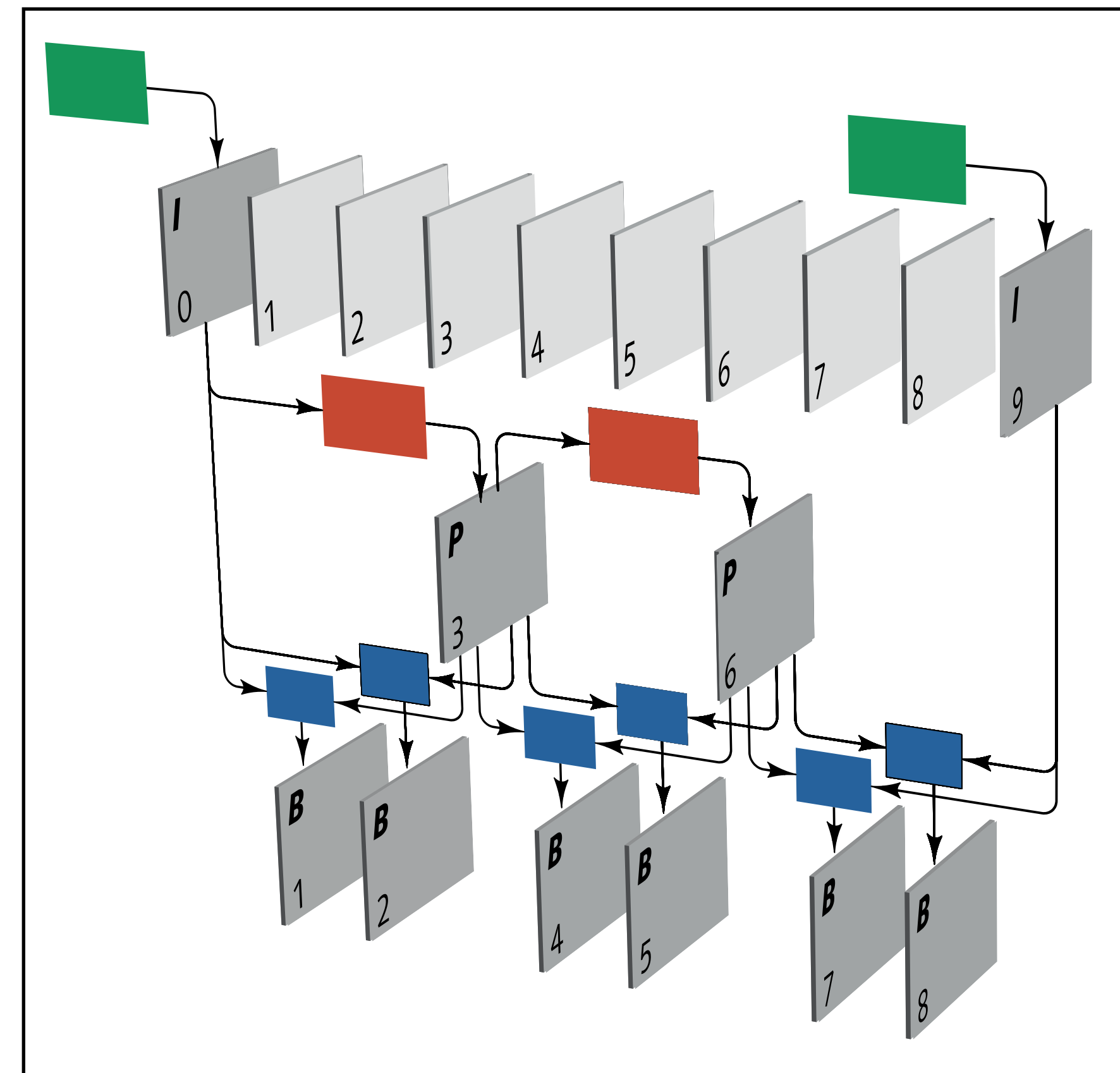
B_1 is generated from I_0 and P_3 , but both I_0 and P_3 themselves are compressed in a lossy fashion. What the decoder sees are the lossy versions I_0' and P_3' .



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' .

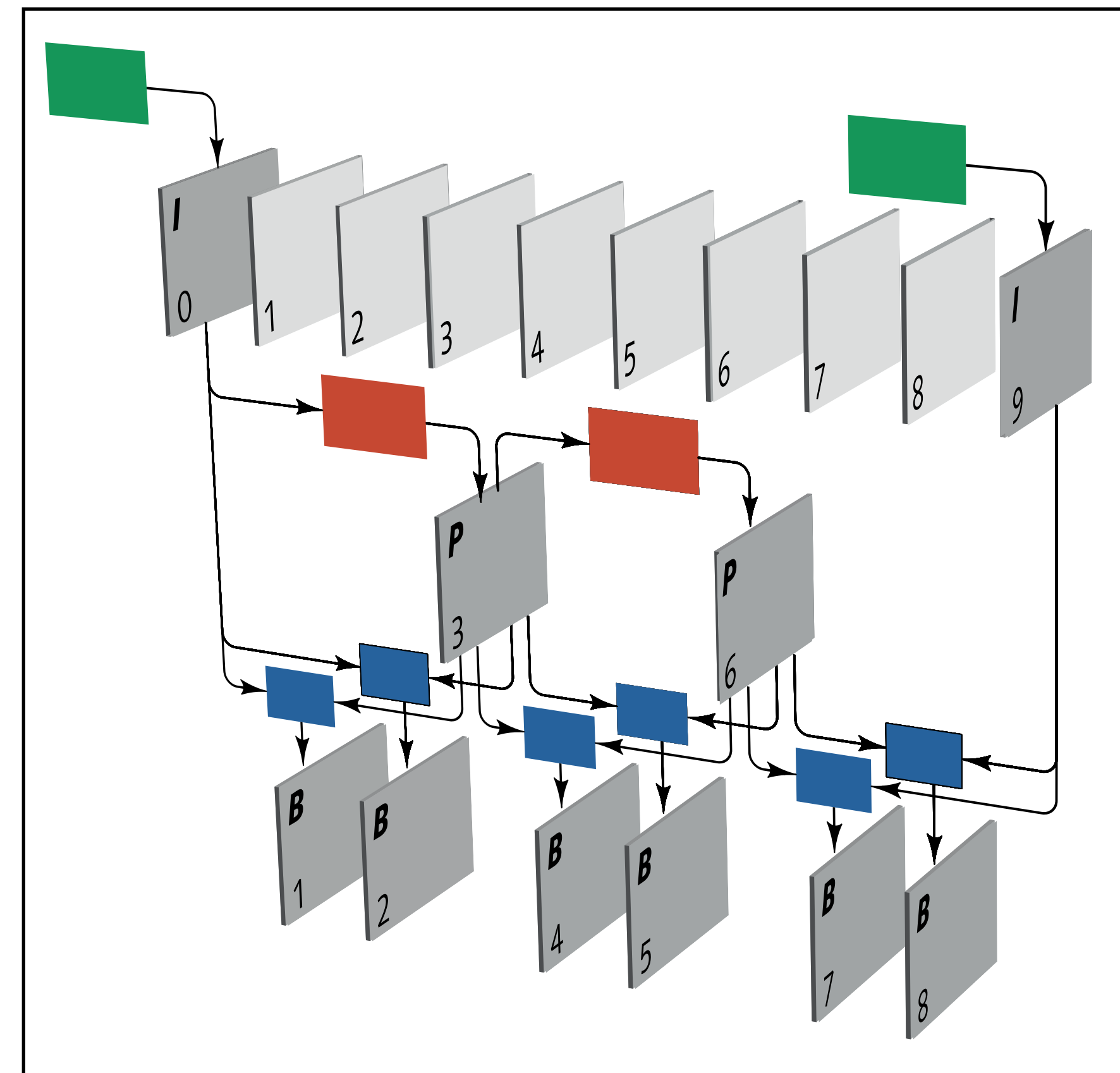


Incorporating the Decoder in the Encoder

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.

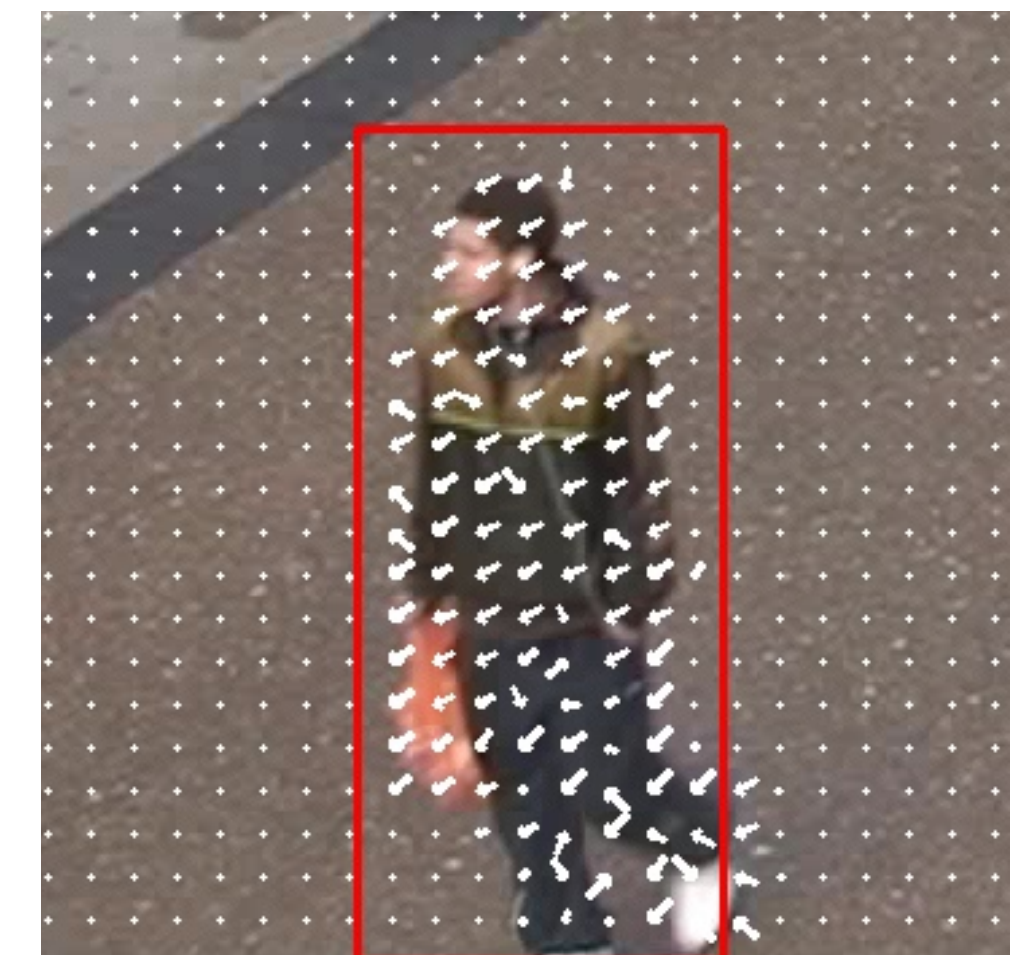
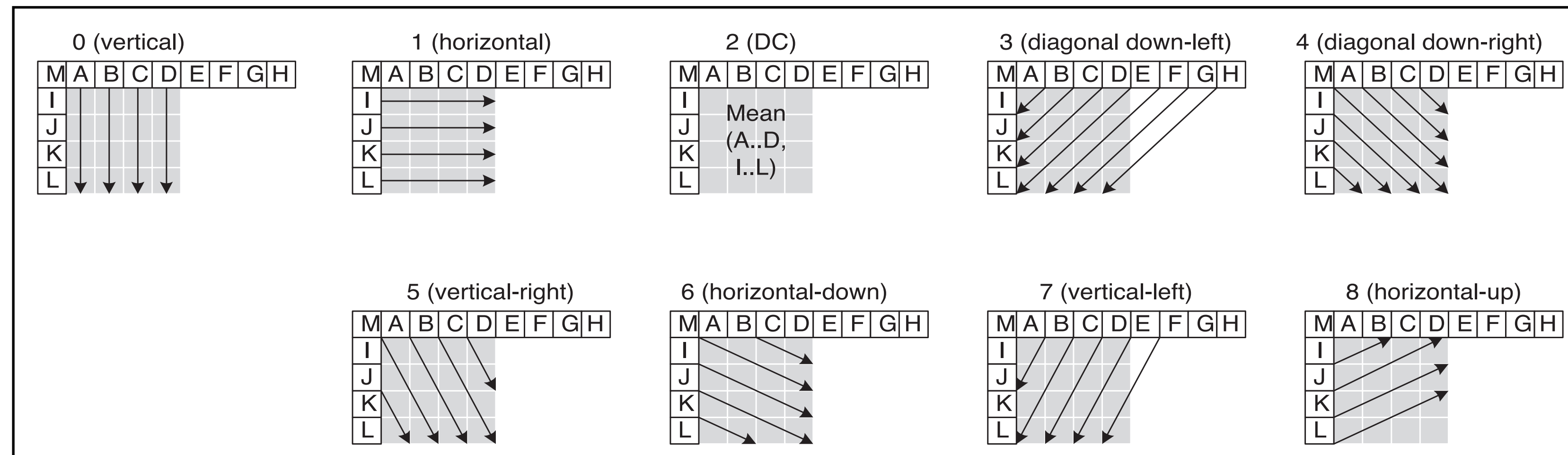


Intra-Frame Encoding (Spatial Prediction)

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

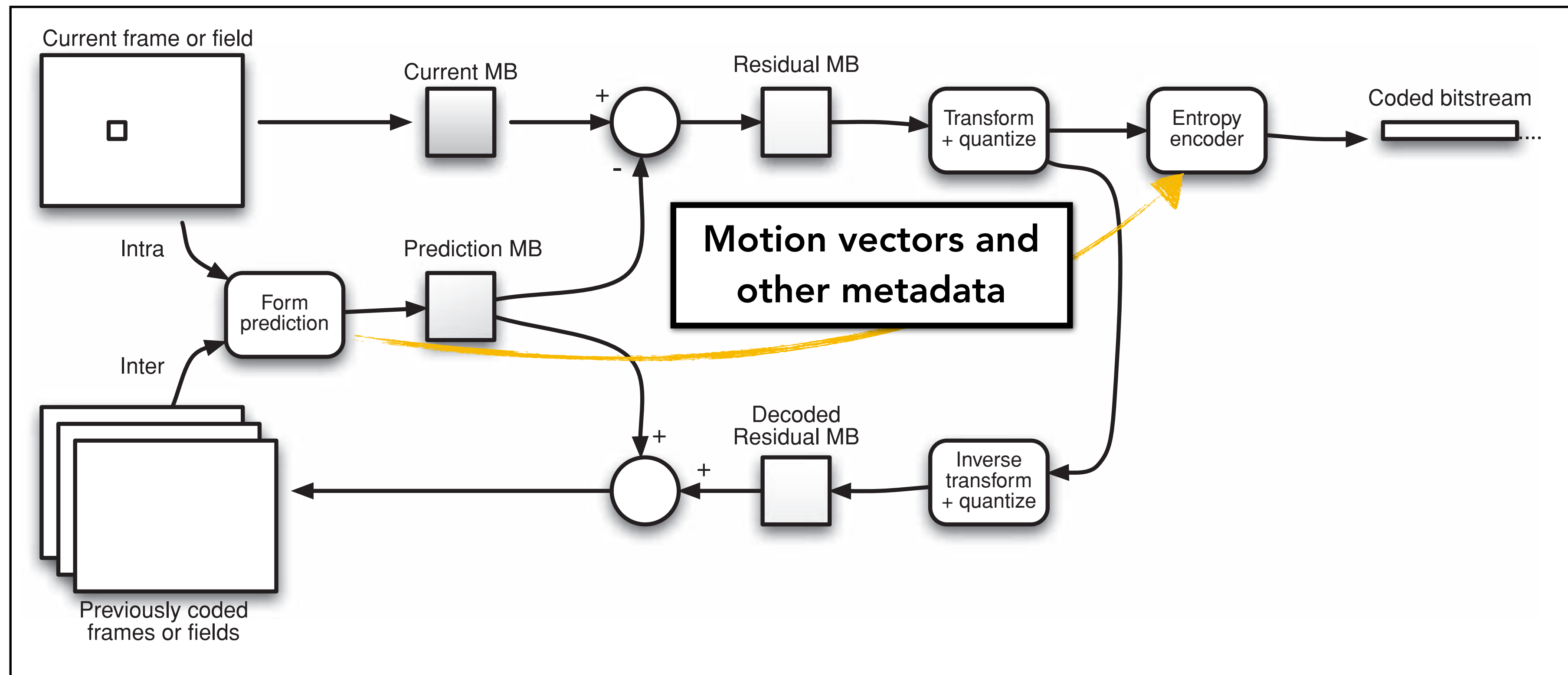
Starting from H.264, I-frames macroblocks are spatially predicted, exploring the observations that spatially adjacent pixels are strongly correlated.

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

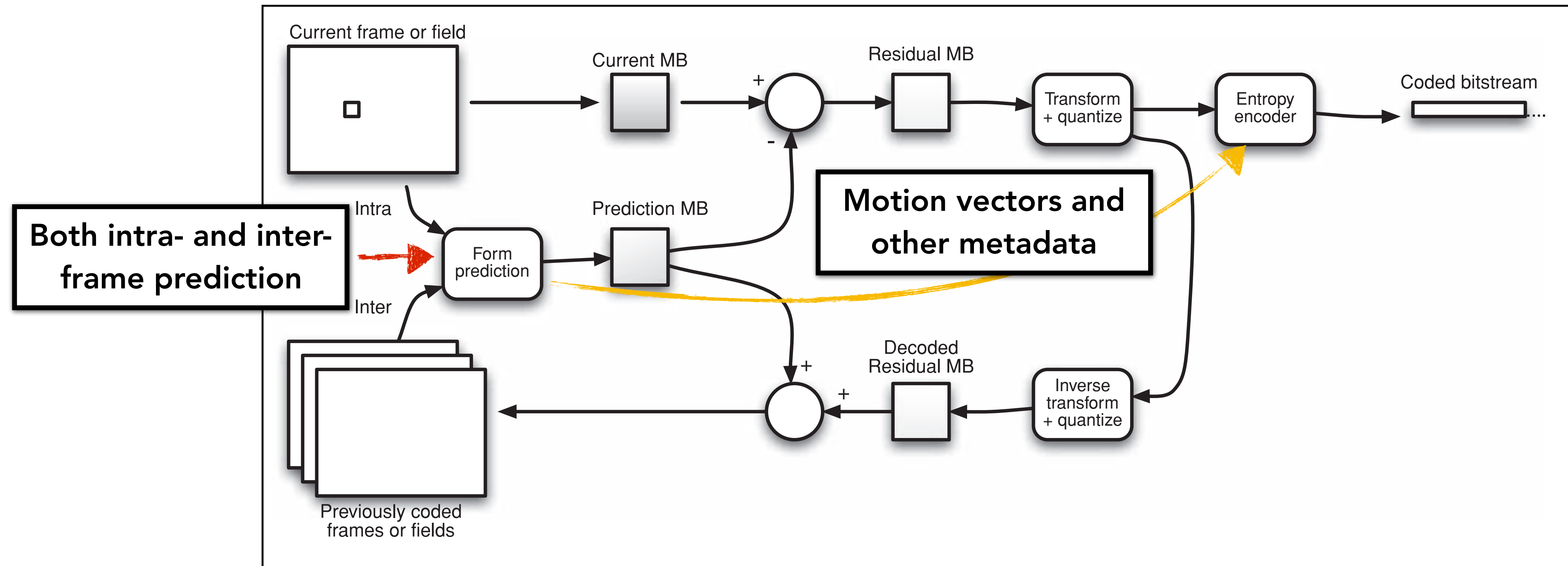


Modes for spatially-predicting a 4x4 tile

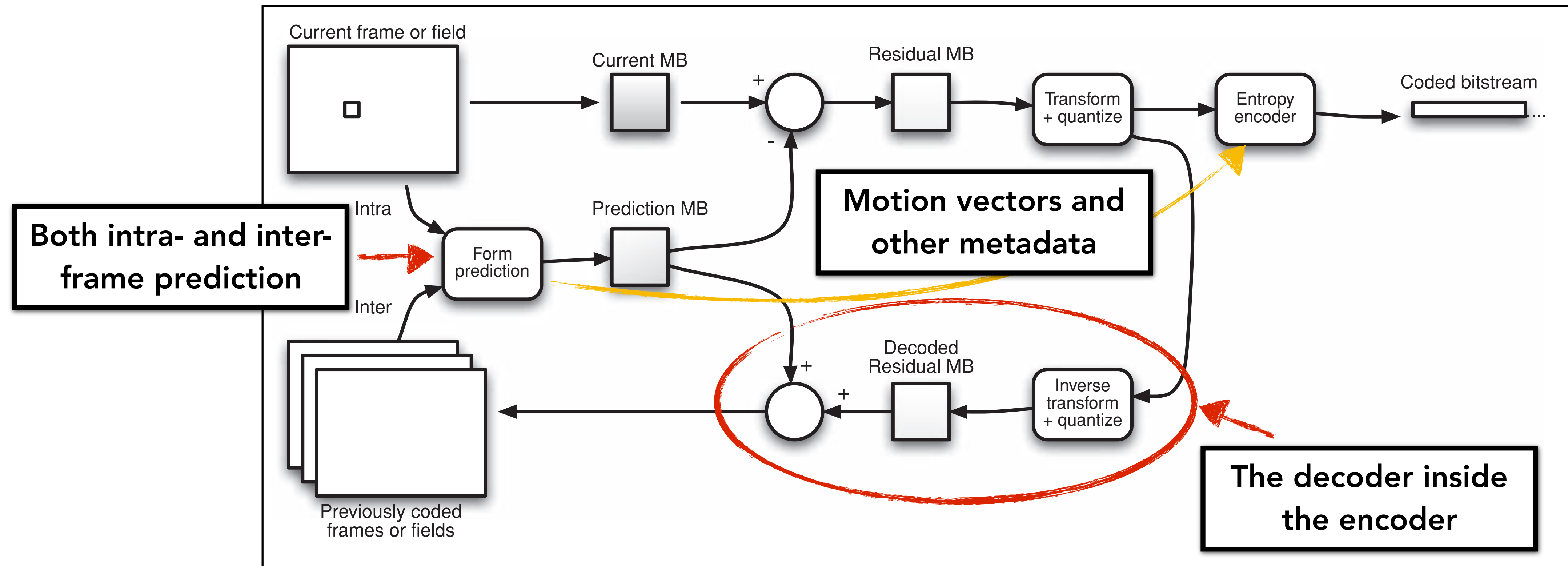
H.264 Encoding Architecture



H.264 Encoding Architecture



H.264 Encoding Architecture



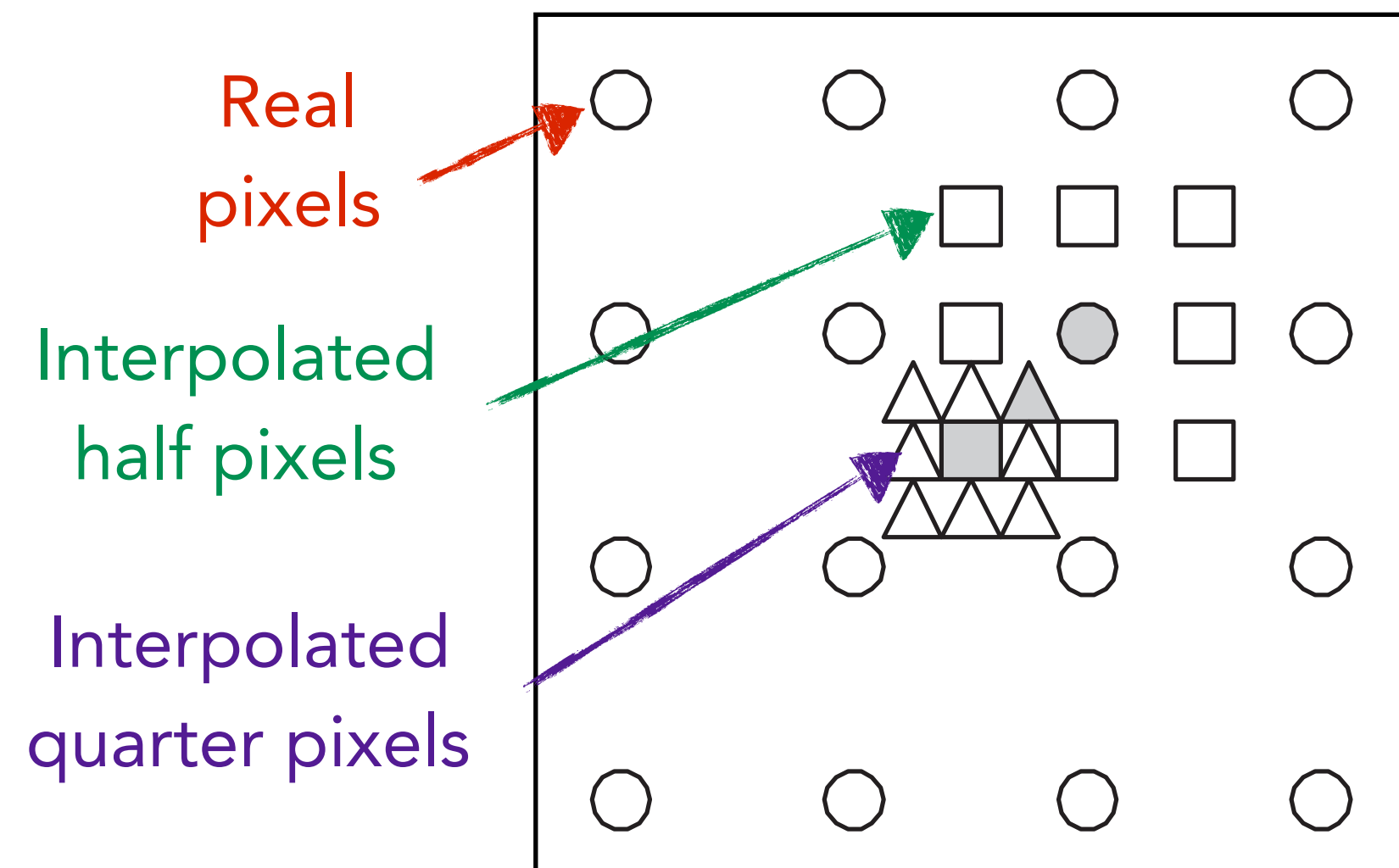
Other Details/Optimizations in Video Codecs

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

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).

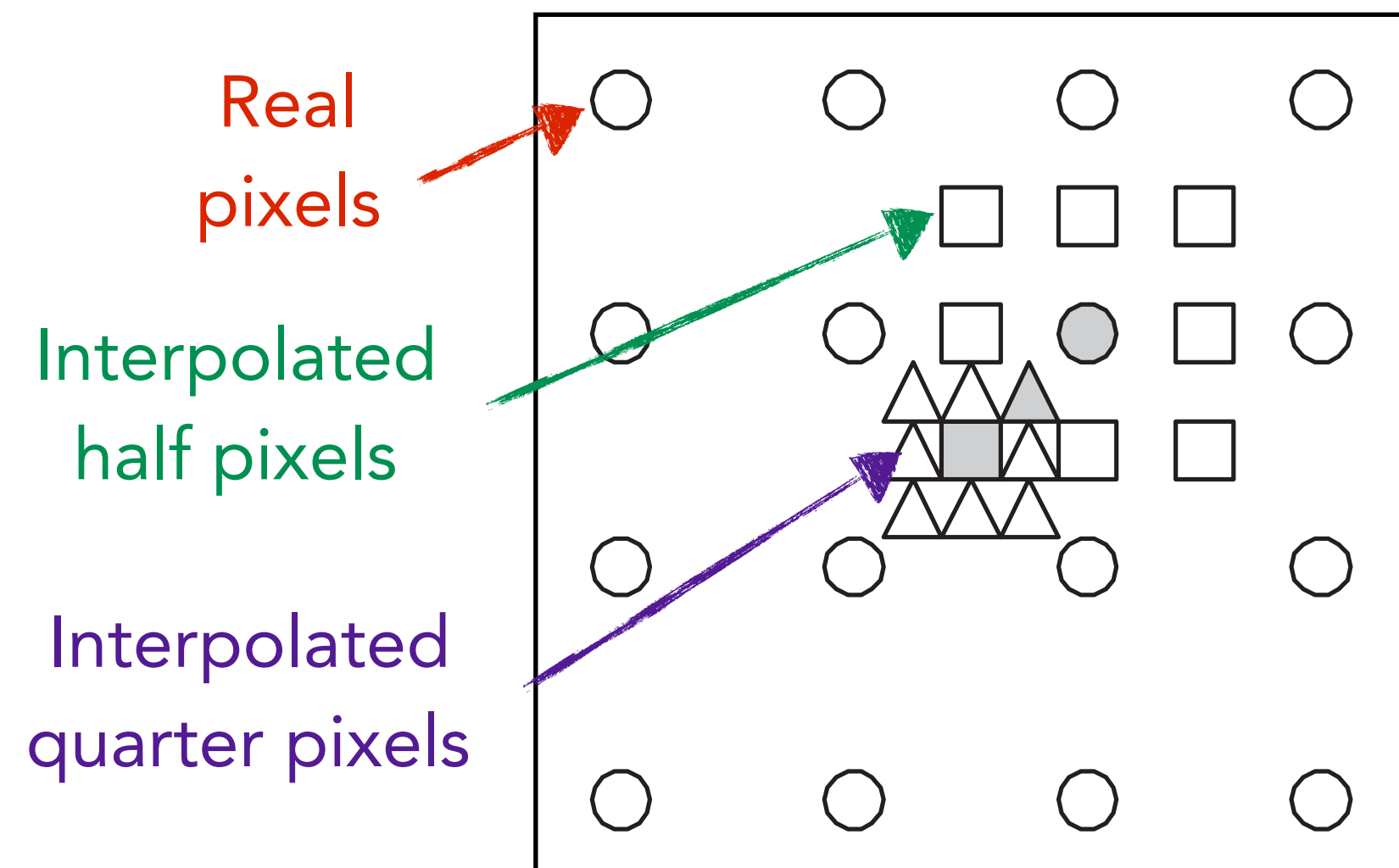


Other Details/Optimizations in Video Codecs

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

Allow sub-pixel motion (smaller residuals).

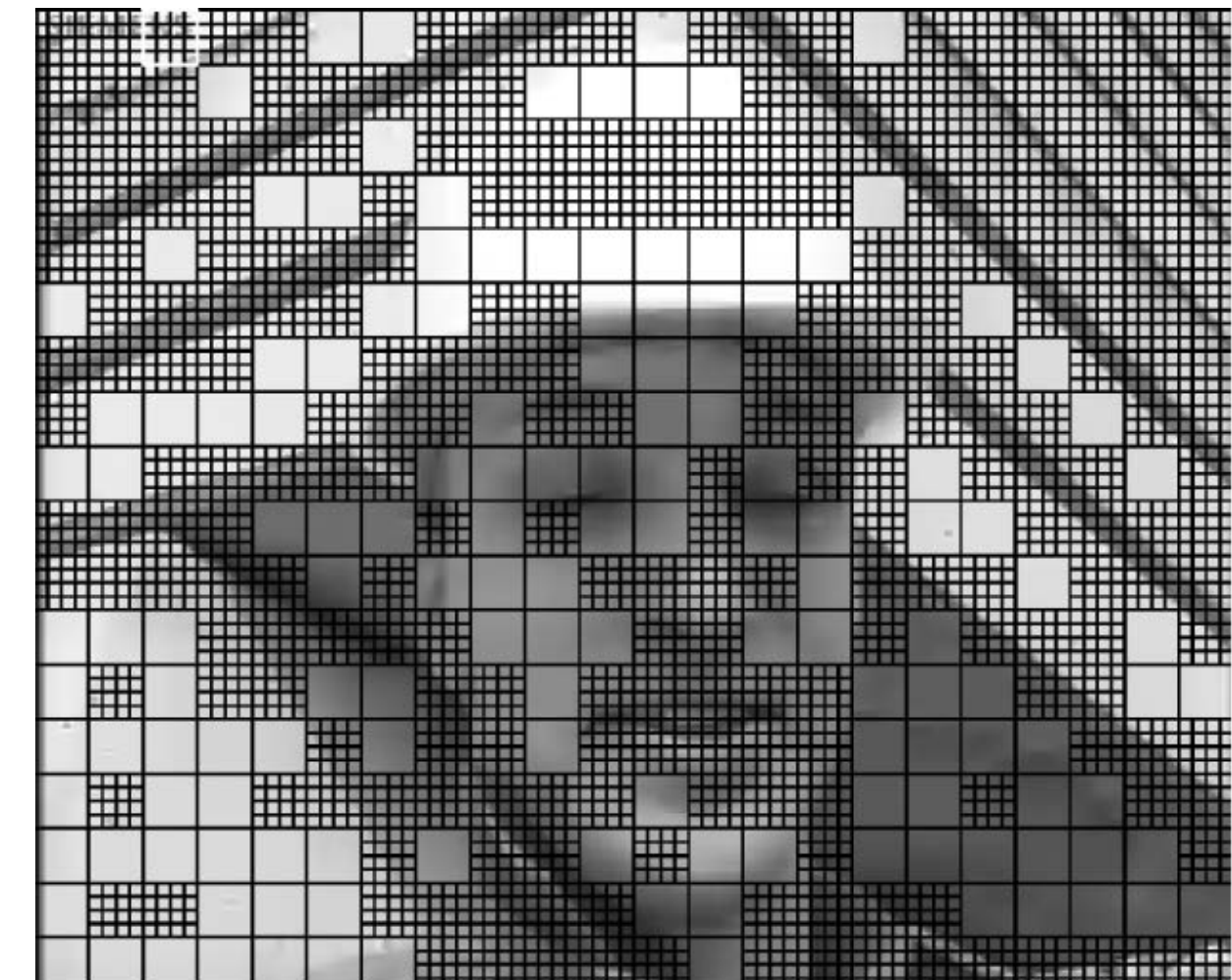
Allow different MB sizes.



Different MB sizes in a P-frame.



Different MB sizes in an I-frame.

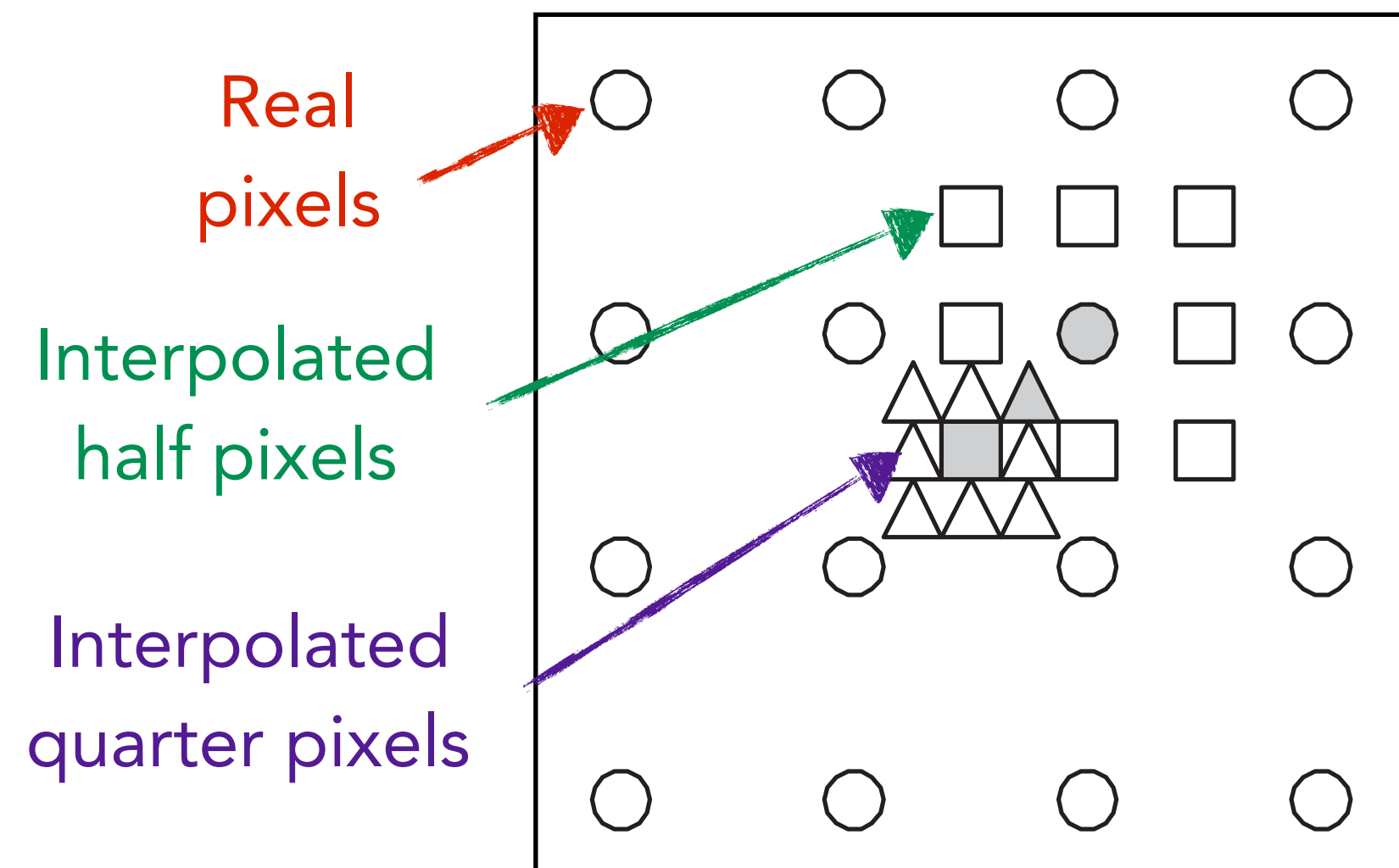


Other Details/Optimizations in Video Codecs

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

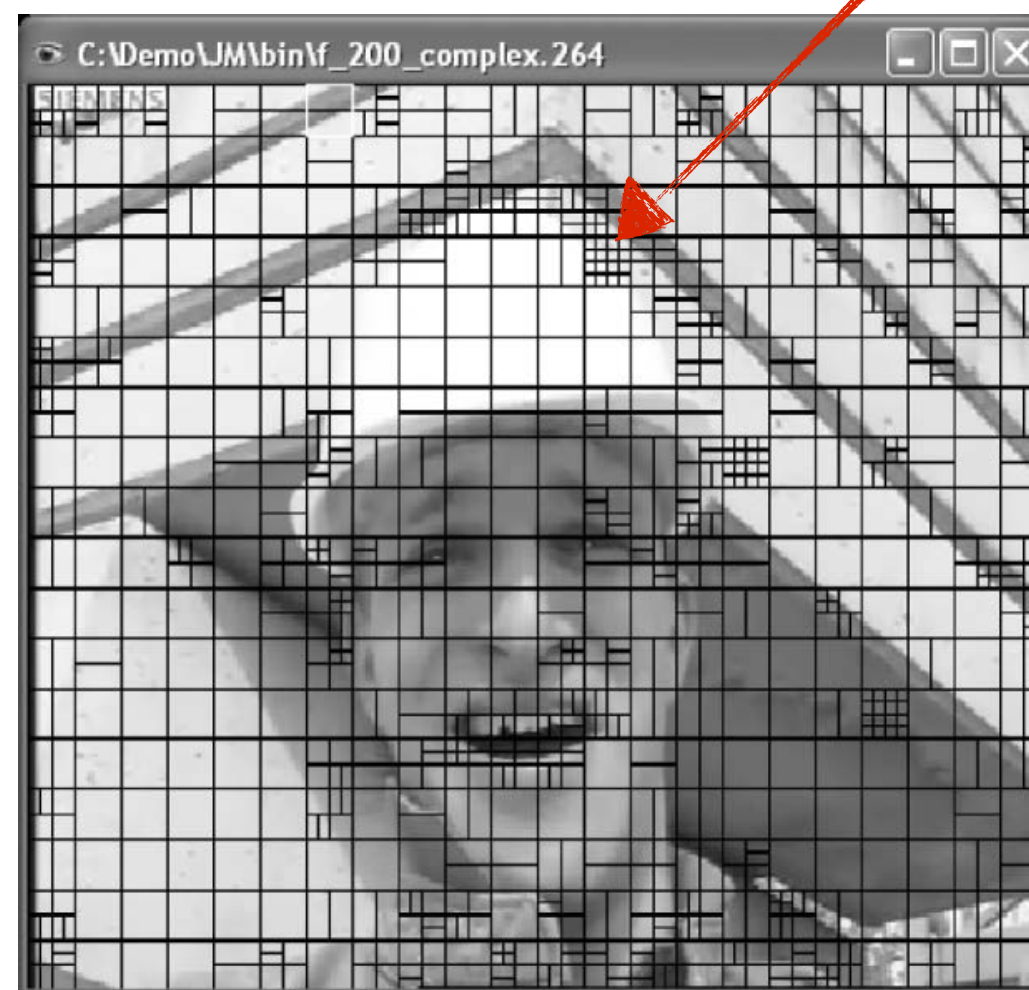
Allow sub-pixel motion (smaller residuals).

Allow different MB sizes.

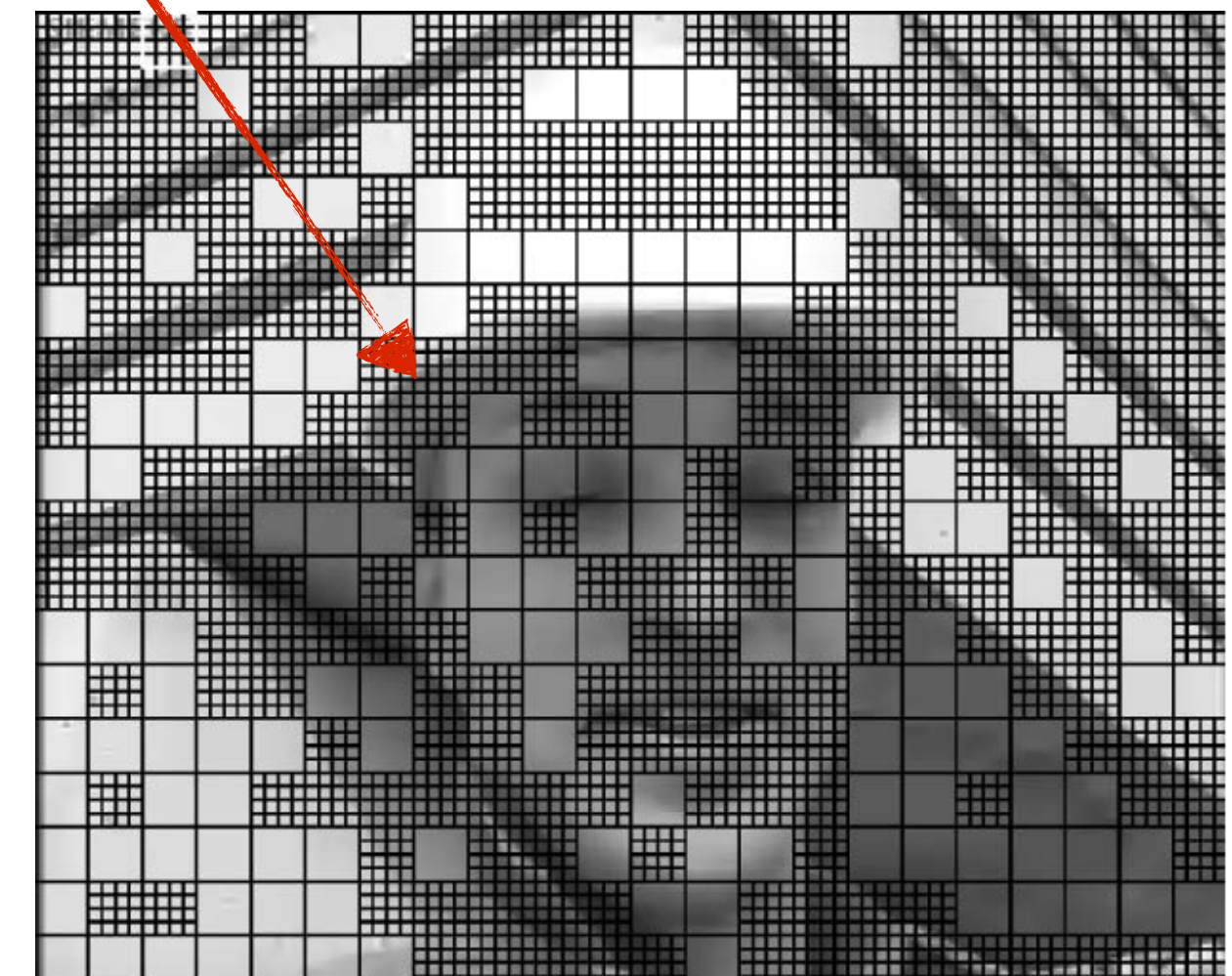


"Busy" areas in a frames are better predicted using smaller MBs.

Different MB sizes in a P-frame.



Different MB sizes in an I-frame.



Video Compression Recap

Spatial redundancy

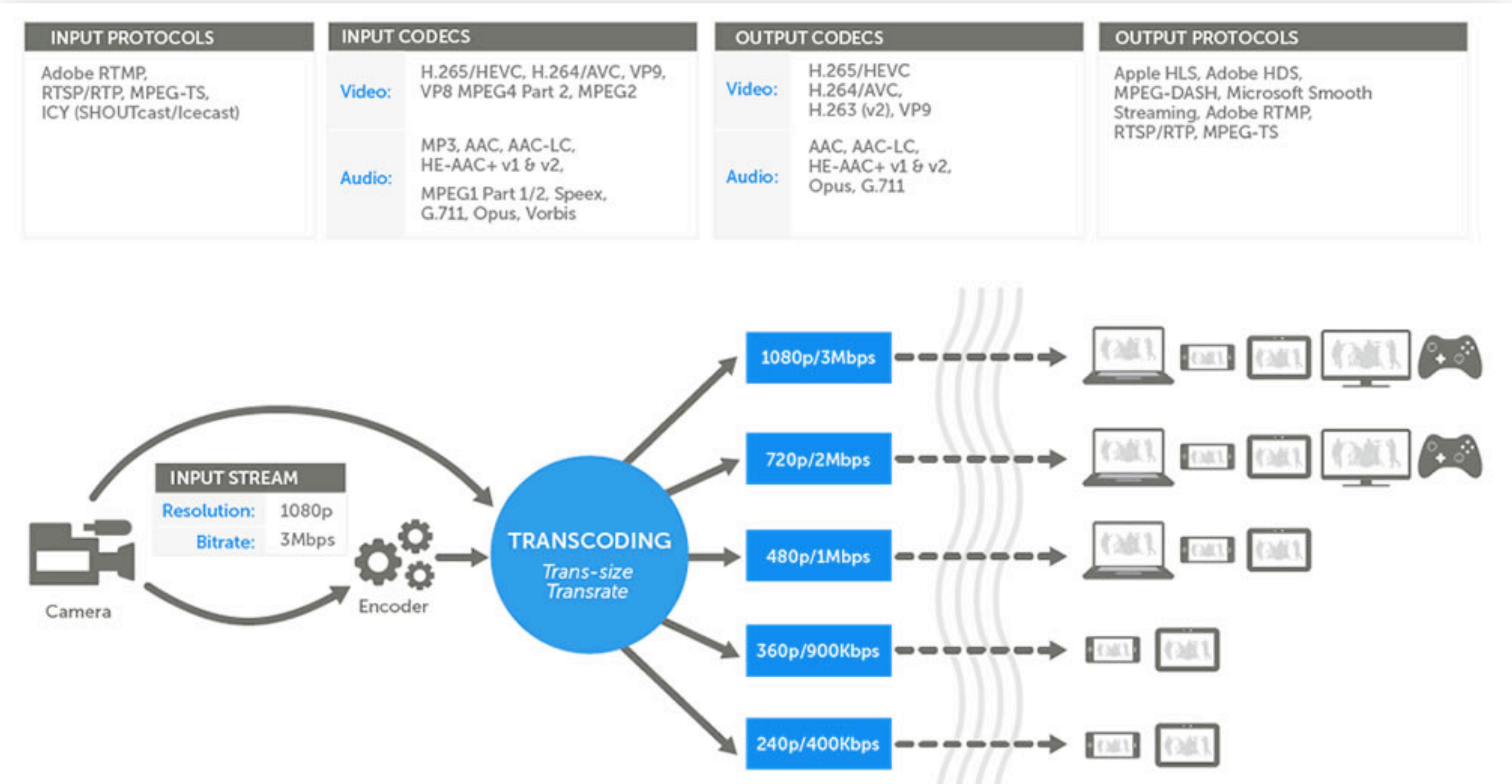
- Pixels in a small neighborhood have strong correlations. I-frames are spatially predicted.

Temporal redundancy

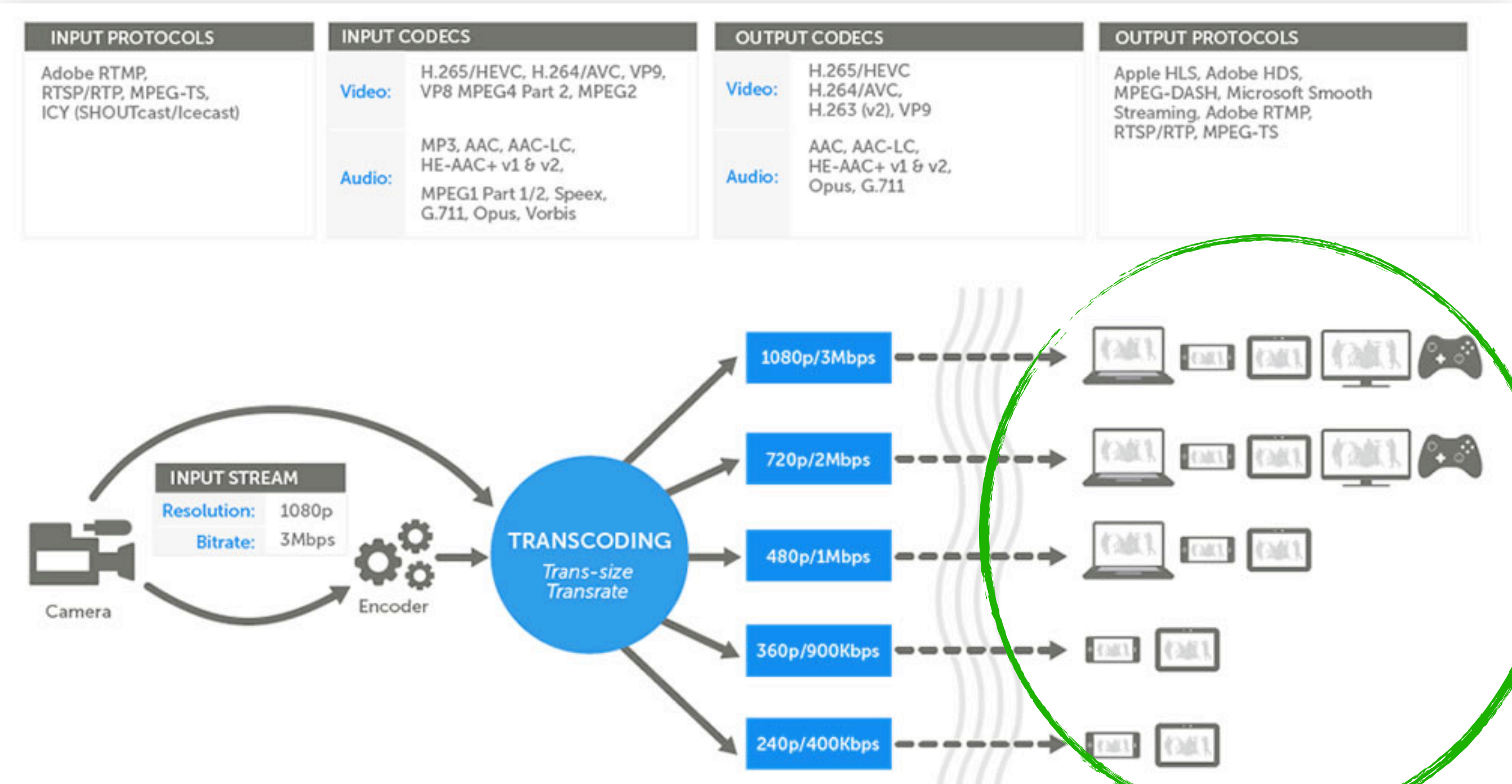
- Consecutive frames have strong correlation. P/B-frames are temporally predicted.

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

Video Transcoding

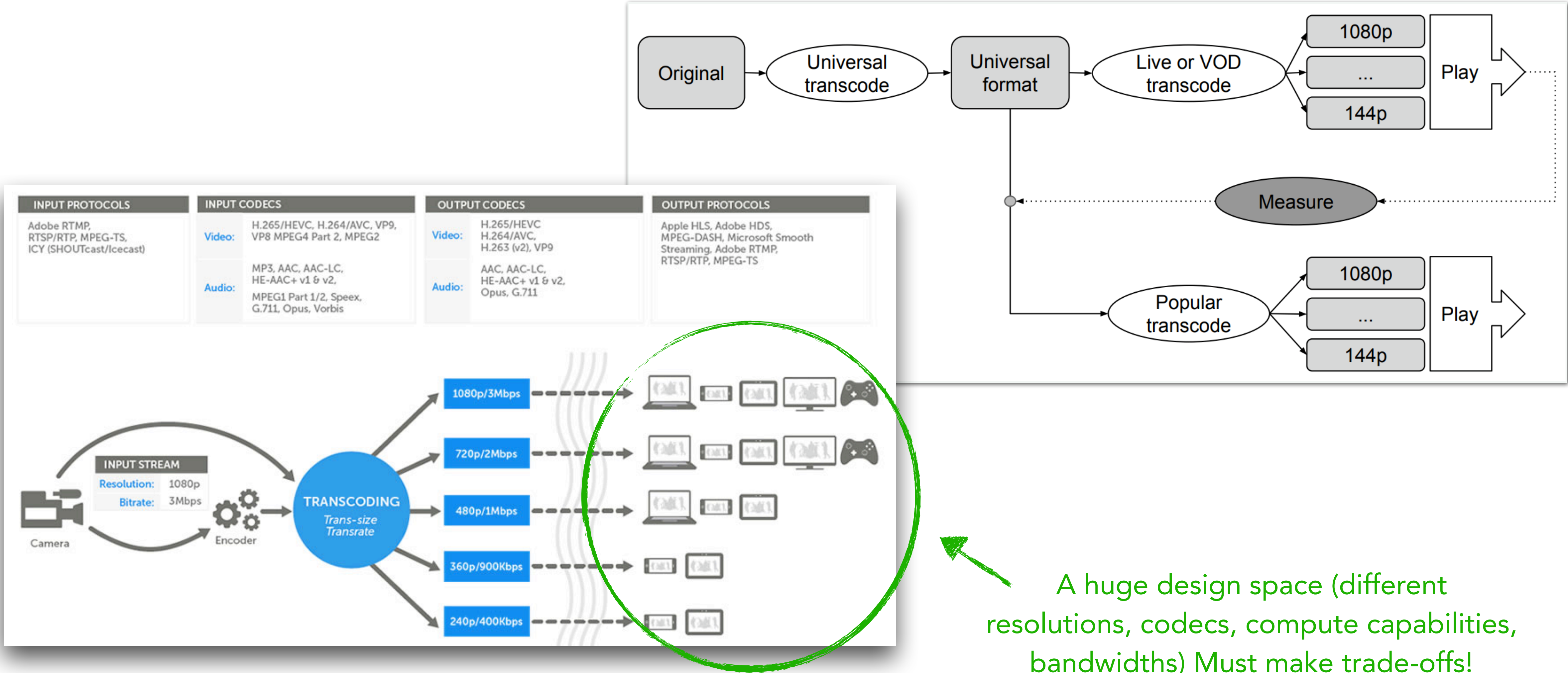


Video Transcoding



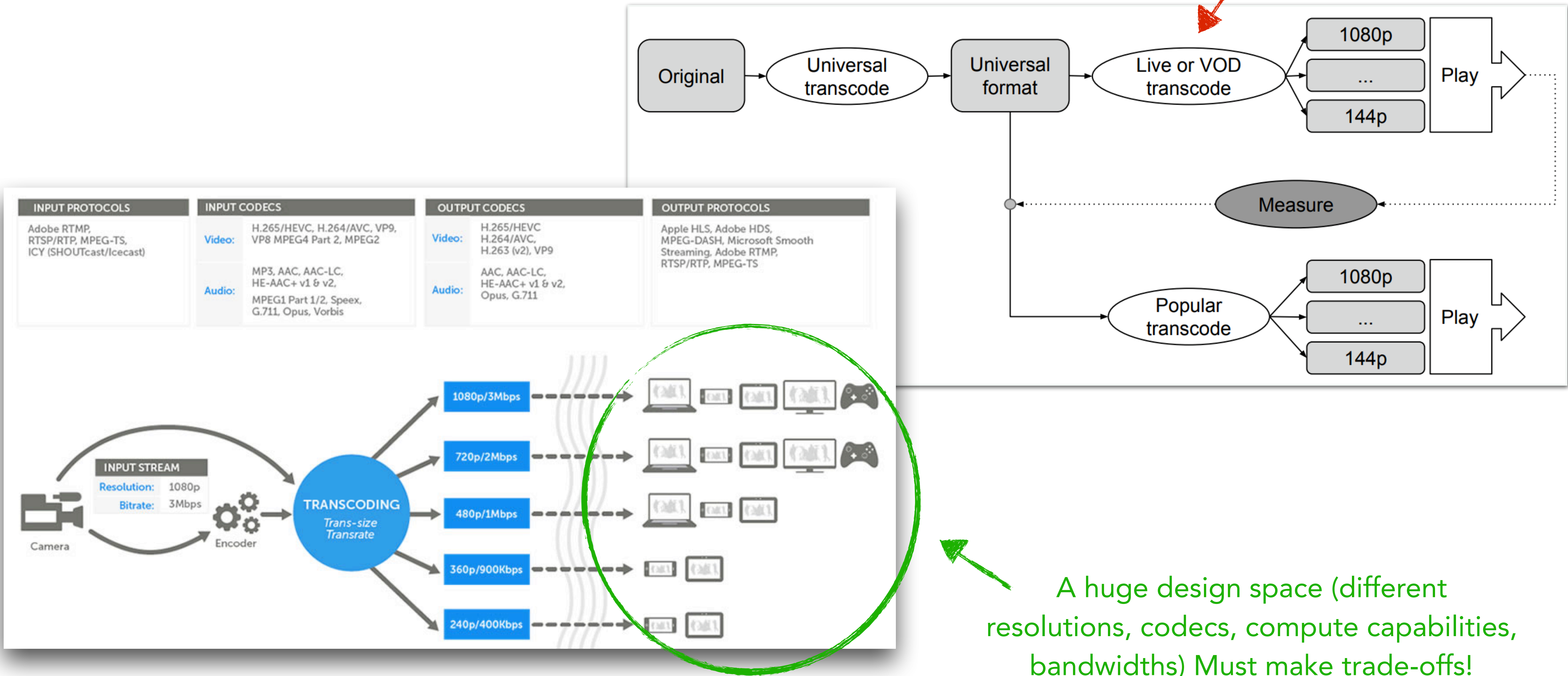
A huge design space (different resolutions, codecs, compute capabilities, bandwidths) Must make trade-offs!

Video Transcoding

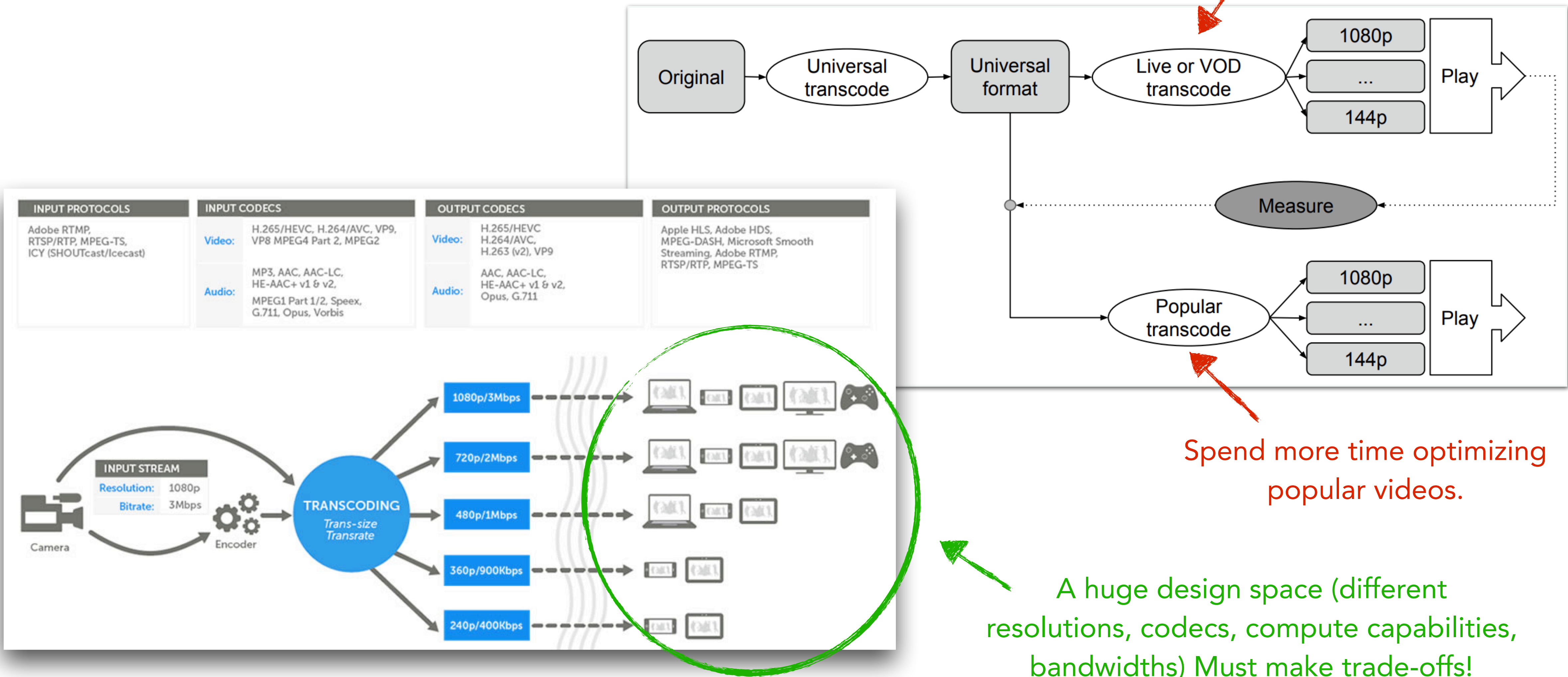


Video Transcoding

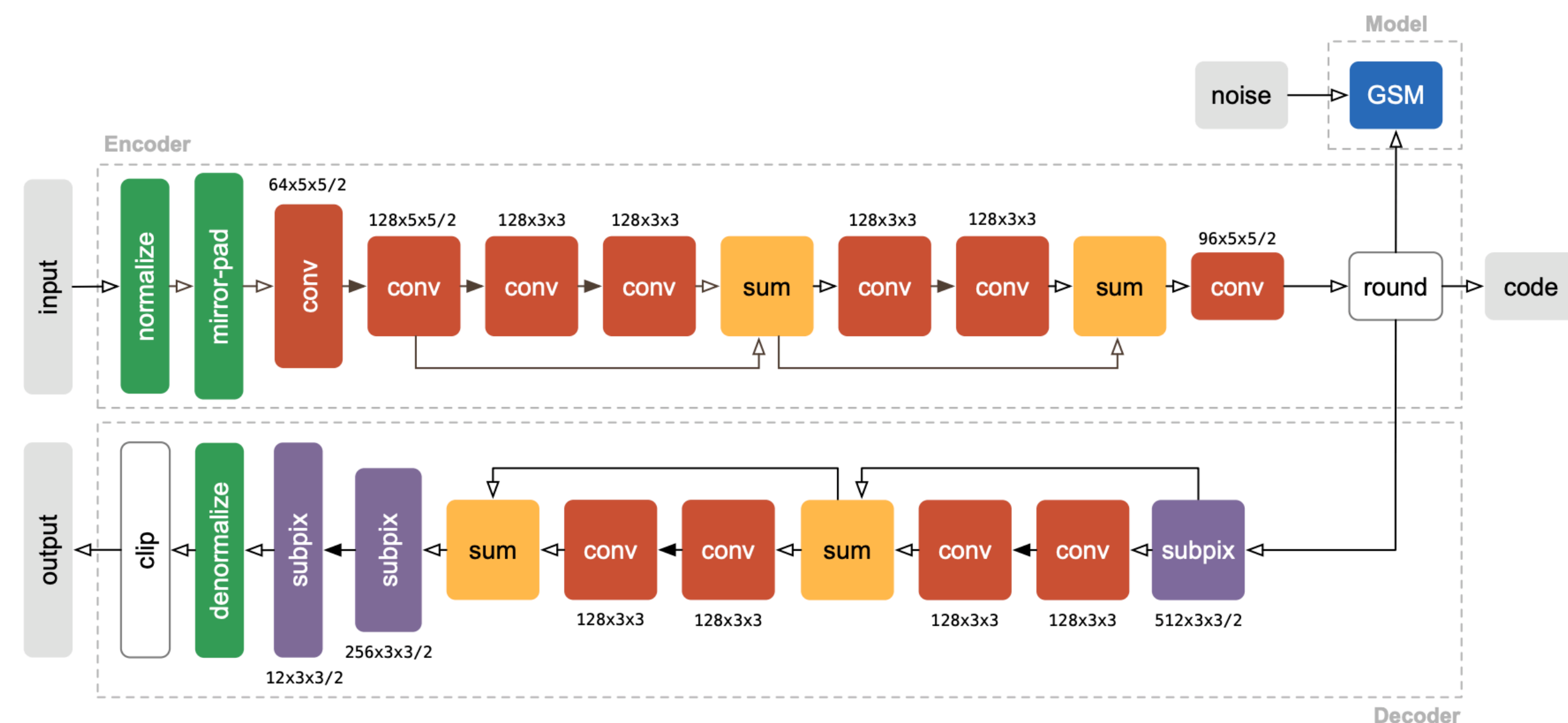
Video-on-demand (VOD). Speed is key.



Video Transcoding



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