Lecture 12: Lossless and Lossy Image and Video Compression
Logistics

- Moved my office hours to Friday 3:30 - 4:30, effective this week.
- Project proposal deadline 10/15. Submit a one page write-up. Talk to me if you still haven’t!
- Grades for previous assignments are released.
- Solution for WA2 is also on BB.
- WA3 is out. Due 10/2, 11:30 AM.
Where Are We

- Image compression and video compression happen in lots of places without us realizing it, because they are fast and (usually) preserve the visual quality.
- How do they work, why do they work that way, and how can we improve it?
Most of cameras and smartphones 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.
Video Streaming/Live Broadcast


Basic Concepts and Ideas of (Any) Compression

- The goal: reduce data size while maintaining high (visual) quality
- Lossless compression vs. lossy compression
- Two main techniques:
  - Removing redundancies.
  - Sacrificing “unimportant” details. In the context of image/video compression, “importance” is usually dictated by human perception.
Image and Video Compression in Hardware

https://forums.developer.nvidia.com/t/supported-recommended-rtos-for-jetson-xavier/76623
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.
  - Informally, 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**.
- They (mostly) all start from a gamma-encoded sRGB color space.
Y’CbCr Color Space

\[ Y' = K_R R' + K_G G' + K_B B' \]

\[ P_B = \frac{1}{2} \frac{B' - Y'}{1 - K_B} \]

\[ P_R = \frac{1}{2} \frac{R' - Y'}{1 - K_R} \]
Y’CbCr Color Space

Gamma-corrected sRGB values in [0, 1] range

\[ Y' = K_R R' + K_G G' + K_B B' \]

\[ P_B = \frac{1}{2} \frac{B' - Y'}{1 - K_B} \]

\[ P_R = \frac{1}{2} \frac{R' - Y'}{1 - K_R} \]
Y’CbCr Color Space

Gamma-corrected sRGB values in [0, 1] range

\[ Y' = K_R R' + K_G G' + K_B B' \]

\[ P_B = \frac{1}{2} \frac{B' - Y'}{1 - K_B} \]

\[ P_R = \frac{1}{2} \frac{R' - Y'}{1 - K_R} \]

\( K_R + K_G + K_B = 1 \)
Y’CbCr Color Space

**luma** (think of it as gamma corrected luminance)

\[ Y' = K_R R' + K_G G' + K_B B' \]

\[ P_B = \frac{1}{2} \frac{B' - Y'}{1 - K_B} \]

\[ P_R = \frac{1}{2} \frac{R' - Y'}{1 - K_R} \]

Gamma-corrected sRGB values in [0, 1] range

\[ K_R + K_G + K_B = 1 \]
Y’CbCr Color Space

Gamma-corrected sRGB values in [0, 1] range

\[ Y' = K_R R' + K_G G' + K_B B' \]

\[ P_B = \frac{1}{2} \frac{B' - Y'}{1 - K_B} \]

\[ P_R = \frac{1}{2} \frac{R' - Y'}{1 - K_R} \]

\[ K_R + K_G + K_B = 1 \]

**luma** (think of it as gamma corrected luminance)

Difference between blue and luma (with some scaling)

Difference between red and luma (with some scaling)
Y’CbCr Color Space

Gamma-corrected sRGB values in [0, 1] range

\[ Y' = K_R R' + K_G G' + K_B B' \]

\[ P_B = \frac{1}{2} \frac{B' - Y'}{1 - K_B} \]

\[ P_R = \frac{1}{2} \frac{R' - Y'}{1 - K_R} \]

**luma** (think of it as gamma corrected luminance)

Collectively called **chroma**

\[ K_R + K_G + K_B = 1 \]

Difference between blue and luma (with some scaling)

Difference between red and luma (with some scaling)
Y’CbCr Color Space

Gamma-corrected sRGB values in [0, 1] range

\[ Y' = K_R R' + K_G G' + K_B B' \]

\[ P_B = \frac{1}{2} \frac{B' - Y'}{1 - K_B} \]

\[ P_R = \frac{1}{2} \frac{R' - Y'}{1 - K_R} \]

luma (think of it as gamma corrected luminance)

Collectively called chroma

The CbCr plane at Y’=0.5
Y’CbCr Color Space

- Luma isn’t exactly luminance because:
  - Luminance is linear while luma is gamma-corrected.
  - The $K_R$, $K_G$, and $K_B$ coefficients are specific to a specific compression standard; they most often are not the same coefficients that transform linear sRGB to Y (luminance) in CIE XYZ.

luma (think of it as gamma corrected luminance)

Collectively called chroma

\[ Y' = K_R R' + K_G G' \]

\[ P_B = \frac{1}{2} \frac{B' - Y'}{1 - K_B} \]

\[ P_R = \frac{1}{2} \frac{R' - Y'}{1 - K_R} \]
JPEG’s Y’CbCr Color Space

- Different standards differ in: 1) how the $K_R$, $K_G$, $K_B$ values are set, and 2) how to scale from $[0, 1]$ to $[0, 255]$. JPEG is based on Rec. 601.

\[
Y' = 0 + (0.299 \cdot R'_D) + (0.587 \cdot G'_D) + (0.114 \cdot B'_D)
\]

\[
C_B = 128 - (0.168736 \cdot R'_D) - (0.331264 \cdot G'_D) - (0.5 \cdot B'_D)
\]

\[
C_R = 128 + (0.5 \cdot R'_D) - (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D)
\]
JPEG’s Y’CbCr Color Space

- Different standards differ in: 1) how the $K_R$, $K_G$, $K_B$ values are set, and 2) how to scale from $[0, 1]$ to $[0, 255]$. JPEG is based on Rec. 601.

<table>
<thead>
<tr>
<th></th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y'$</td>
<td>$0 + (0.299 \cdot R'_D) + (0.587 \cdot G'_D) + (0.114 \cdot B'_D)$</td>
</tr>
<tr>
<td>$C_B$</td>
<td>$128 - (0.168736 \cdot R'_D) - (0.331264 \cdot G'_D) + (0.5 \cdot B'_D)$</td>
</tr>
<tr>
<td>$C_R$</td>
<td>$128 + (0.5 \cdot R'_D) - (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D)$</td>
</tr>
</tbody>
</table>

Gamma-corrected RGB values within $[0, 255]$
### JPEG’s Y’CbCr Color Space

- Different standards differ in: 1) how the $K_R$, $K_G$, $K_B$ values are set, and 2) how to scale from [0, 1] to [0, 255]. JPEG is based on Rec. 601.

\[
\begin{align*}
Y' &= 0 + (0.299 \cdot R_D' + 0.587 \cdot G_D' + 0.114 \cdot B_D') \\
C_B &= 128 - (0.168736 \cdot R_D' - (0.331264 \cdot G_D' + 0.5 \cdot B_D') \\
C_R &= 128 + (0.5 \cdot R_D' - (0.418688 \cdot G_D' - (0.081312 \cdot B_D'))
\end{align*}
\]

8-bit values within [0, 255]

Gamma-corrected RGB values within [0, 255]
## JPEG’s Y’CbCr Color Space

- Different standards differ in: 1) how the $K_R$, $K_G$, $K_B$ values are set, and 2) how to scale from $[0, 1]$ to $[0, 255]$. JPEG is based on Rec. 601.

<table>
<thead>
<tr>
<th>Component</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luma</td>
<td>$Y' = 0 + (0.299 \cdot R') + (0.587 \cdot G') + (0.114 \cdot B')$</td>
</tr>
<tr>
<td>Chroma</td>
<td>$C_B = 128 - (0.168736 \cdot R') - (0.331264 \cdot G') + (0.5 \cdot B')$</td>
</tr>
<tr>
<td></td>
<td>$C_R = 128 + (0.5 \cdot R') - (0.418688 \cdot G') - (0.081312 \cdot B')$</td>
</tr>
</tbody>
</table>

8-bit values within $[0, 255]$ | Gamma-corrected RGB values within $[0, 255]$
JPEG’s Y’CbCr Color Space

- Different standards differ in: 1) how the $K_R$, $K_G$, $K_B$ values are set, and 2) how to scale from $[0, 1]$ to $[0, 255]$. JPEG is based on Rec. 601.

\[
\begin{align*}
Y' &= 0 + (0.299 \cdot R_D') + (0.587 \cdot G_D') + (0.114 \cdot B_D') \\
C_B &= 128 - (0.168736 \cdot R_D') - (0.331264 \cdot G_D') - (0.5 \cdot B_D') \\
C_R &= 128 + (0.5 \cdot R_D') - (0.418688 \cdot G_D') - (0.081312 \cdot B_D')
\end{align*}
\]

8-bit values within $[0, 255]$  
Gamma-corrected RGB values within $[0, 255]$
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)

<table>
<thead>
<tr>
<th>Y’</th>
<th>Y’</th>
<th>Y’</th>
<th>Y’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cb</td>
<td>Cb</td>
<td>Cb</td>
<td>Cb</td>
</tr>
<tr>
<td>Cr</td>
<td>Cr</td>
<td>Cr</td>
<td>Cr</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Y’</th>
<th>Y’</th>
<th>Y’</th>
<th>Y’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cb</td>
<td>Cb</td>
<td>Cb</td>
<td>Cb</td>
</tr>
<tr>
<td>Cr</td>
<td>Cr</td>
<td>Cr</td>
<td>Cr</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Y’</th>
<th>Y’</th>
<th>Y’</th>
<th>Y’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cb</td>
<td>Cb</td>
<td>Cb</td>
<td>Cb</td>
</tr>
<tr>
<td>Cr</td>
<td>Cr</td>
<td>Cr</td>
<td>Cr</td>
</tr>
</tbody>
</table>
Chroma Subsampling (Lossy)

https://en.wikipedia.org/wiki/Chroma_subsampling
Subsampling in RGB?

› 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://www.red.com/red-101/bayer-sensor-strategy
https://codegolf.stackexchange.com/questions/86410/reverse-bayer-filter-of-an-image
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.
- JPEG uses **Discrete Cosine Transformation** (DCT) for this conversion.
- Recall Fourier transform: any periodic function, however complex it is along the period, can be exactly represented as a weighted sum of simple sinusoids.
- 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).
  - Each basis function is a 8x8 image.
DCT (Lossless)

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

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform
DCT (Lossless)

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

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform
DCT (Lossless)

Weights (coefficients)

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

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform
DCT (Lossless)

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

Zero-centered

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

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform
DCT (Lossless)

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

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

Zero-centered weights (coefficients)

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform
Weights (coefficients)

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

DCT (Lossless)

\[
G = \begin{bmatrix}
G_{0,0} & G_{0,1} & G_{0,2} & \cdots & G_{0,15} \\
G_{1,0} & G_{1,1} & G_{1,2} & \cdots & G_{1,15} \\
G_{2,0} & G_{2,1} & G_{2,2} & \cdots & G_{2,15} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
G_{15,0} & G_{15,1} & G_{15,2} & \cdots & G_{15,15}
\end{bmatrix}
\]

\[
\sum_{x=0}^{7} \sum_{y=0}^{7} g(x, y) \cos \left( \frac{2x+1}{16} \pi \right) \cos \left( \frac{2y+1}{16} \pi \right)
\]

Zero-centered

https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform
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.
# Coefficient Quantization (Lossy)

Weights (coefficients)

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

Weights (coefficients)

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

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)

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

Quantization matrix (fixed, no need to encode)

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

Quantized coefficients

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

\[
B = \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
Coefficient Quantization (Lossy)

Weights (coefficients)

\[
G = \begin{bmatrix}
-415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\
-46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\
-48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\
12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\
-7.73 & -2.91 & -23.8 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\
-1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\
-0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68
\end{bmatrix}
\]

Quantization matrix (fixed, no need to encode)

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

Quantized coefficients

\[
B_{u,v} = \text{round}(G_{u,v}Q_{u,v})
\]

\[
B = \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
B and G aren’t equivalent (lossy), but B’s coefficients are small in magnitude, i.e., requiring fewer bits to encode.
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

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

High frequency

Low frequency

\[ B = \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \]
Quantization Matrix

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

Low frequency

High frequency

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

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!

| Q = | 16 11 10  |
|     | 12 12 14  |
|     | 14 13 16  |
|     | 14 17 22  |
|     | 18 22 37  |
|     | 24 35 55  |
|     | 49 64 78  |
|     | 72 92 95  |

| B = | -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  |
|     | 0 0 0 0 0 0 0 0  |
Entropy Encoding (Lossless)

1. Reorder quantized coefficients in zig-zag order to increase the repetitiveness in the sequence. Why zig-zag? Look at the pattern in the basis function.

\[
B = \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}
\]
1. Reorder quantized coefficients in zig-zag order to increase the repetitiveness in the sequence. Why zig-zag? Look at the pattern in the basis function.
Entropy Encoding (Lossless)

1. Reorder quantized coefficients in zig-zag order to increase the repetitiveness in the sequence. Why zig-zag? Look at the pattern in the basis function.

2. Apply run-length encoding. Compresses zeros.

3. Apply Huffman coding (or arithmetic coding) to compress the rest.

RLE

![Example RLE](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
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
Pressure

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

Start seeing quantization artifact (blocking)

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

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

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

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

Quality level (Q) = 100
Compression ratio: 2.7 : 1
Start seeing quantization artifact (blocking)

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

Quality level (Q) = 1
Compression ratio: 144 : 1
Color is off (chroma subsampling artifacts)
Severe blocking artifacts

https://en.wikipedia.org/wiki/JPEG
Image Compression Summary

- JPEG is lossy compression (chroma subsampling and coefficient quantization).
- 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?
- PNG and TIFF are lossless compression standards.
- The compressing (encoding) and uncompressing (decoding) system (software and hardware) is called the “codec”.
- Image (and video) codes are well-optimized by specialized hardware. Your smartphone most likely has one.
Video Compression

- Compressing a sequence of continuous frames together.
- What about compressing each image using JPEG?
- That gets you Motion-JPEG (M-JPEG).
  - 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
- 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.
- 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.

https://www.amazon.com/Original-Appleipod-Compatible-Gigabyte-Classic/dp/B07Z64J7T1
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.

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

https://www.amazon.com/Original-Appleipod-Compatible-Gigabyte-Classic/dp/B07Z64J2T1
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/B07Z64JZT1
Video Compression: The Big Ideas

Apart from scene changes, there is a statistical likelihood that successive pictures in a video sequence are very similar. In fact, it is necessary that successive pictures are mostly similar: If this were not the case, human vision could make no sense of the sequence!

The M in MPEG stands for moving, not motion! If you read a document that errs in this detail, the accuracy of the document is suspect!

The efficiency of transform coding can be increased by a factor of 10 or more by exploiting the inherent temporal redundancy of video. The MPEG standard was developed by the Moving Picture Experts Group within ISO and IEC. In MPEG, intra-frame compression is used to provide an initial, self-contained picture – a reference picture, upon which predictions of succeeding pictures can be based. The encoder then transmits pixel differences, that is, prediction errors, or residuals – from the reference, as sketched in Figure 16.1. The method is termed interframe coding. (In interlaced video, differences between fields may be used, so the method is more accurately described as interpicture coding.)

Once a reference picture has been received by the decoder, it provides a basis for predicting succeeding pictures. This estimate is improved when the decoder receives the residuals. The scheme is effective provided that the residuals can be coded more compactly than the raw picture information.

Motion in a video sequence causes displacement of scene elements with respect to the image array. A fast-moving image element may easily move 20 pixels in one frame time. In the presence of motion, a pixel at a certain location may take quite different values in successive pictures. Motion is liable to cause prediction


Digital Video and HDTV Algorithms and Interfaces, Figure 16.2
Video Compression: The Big Ideas

Apart from scene changes, there is a statistical likelihood that successive pictures in a video sequence are very similar. In fact, it is necessary that successive pictures are mostly similar: If this were not the case, human vision could make no sense of the sequence!

The M in MPEG stands for moving, not motion! If you read a document that errs in this detail, the accuracy of the document is suspect!

The efficiency of transform coding can be increased by a factor of 10 or more by exploiting the inherent temporal redundancy of video. The MPEG standard was developed by the Moving Picture Experts Group within ISO and IEC. In MPEG, intra-frame compression is used to provide an initial, self-contained picture—a reference picture, upon which predictions of succeeding pictures can be based. The encoder then transmits pixel differences, that is, prediction errors, or residuals— from the reference, as sketched in Figure 16.1. The method is termed interframe coding. (In interlaced video, differences between fields may be used, so the method is more accurately described as interpicture coding.)

Once a reference picture has been received by the decoder, it provides a basis for predicting succeeding pictures. This estimate is improved when the decoder receives the residuals. The scheme is effective provided that the residuals can be coded more compactly than the raw picture information.

Motion in a video sequence causes displacement of scene elements with respect to the image array. A fast-moving image element may easily move 20 pixels in one frame time. In the presence of motion, a pixel at a certain location may take quite different values in successive pictures. Motion is liable to cause prediction errors.

Apart from scene changes, there is a statistical likelihood that successive pictures in a video sequence are very similar. In fact, it is necessary that successive pictures are mostly similar: If this were not the case, human vision could make no sense of the sequence!

The M in MPEG stands for moving, not motion! If you read a document that errs in this detail, the accuracy of the document is suspect!

The efficiency of transform coding can be increased by a factor of 10 or more by exploiting the inherent temporal redundancy of video. The MPEG standard was developed by the Moving Picture Experts Group within ISO and IEC. In MPEG, intra-frame compression is used to provide an initial, self-contained picture – a reference picture, upon which predictions of succeeding pictures can be based. The encoder then transmits pixel differences, that is, prediction errors, or residuals – from the reference, as sketched in Figure 16.1. The method is termed inter-frame coding.

Once a reference picture has been received by the decoder, it provides a basis for predicting succeeding pictures. This estimate is improved when the decoder receives the residuals. The scheme is effective provided that the residuals can be coded more compactly than the raw picture information.

Motion in a video sequence causes displacement of scene elements with respect to the image array. A fast-moving image element may easily move 20 pixels in one frame time. In the presence of motion, a pixel at a certain location may take quite different values in successive pictures. Motion is liable to cause prediction errors.
MPEG

Apart from scene changes, there is a statistical likelihood that successive pictures in a video sequence are very similar. In fact, it is necessary that successive pictures are mostly similar: If this were not the case, human vision could make no sense of the sequence!

The M in MPEG stands for moving, not motion! If you read a document that errs in this detail, the accuracy of the document is suspect!

The efficiency of transform coding can be increased by a factor of 10 or more by exploiting the inherent temporal redundancy of video. The MPEG standard was developed by the Moving Picture Experts Group within ISO and IEC. In MPEG, intra-frame compression is used to provide an initial, self-contained picture – a reference picture, upon which predictions of succeeding pictures can be based. The encoder then transmits pixel differences, that is, prediction errors, or residuals – from the reference, as sketched in Figure 16.1. The method is termed interframe coding.

(In interlaced video, differences between fields may be used, so the method is more accurately described as interpicture coding.)

Once a reference picture has been received by the decoder, it provides a basis for predicting succeeding pictures. This estimate is improved when the decoder receives the residuals. The scheme is effective provided that the residuals can be coded more compactly than the raw picture information.

Motion in a video sequence causes displacement of scene elements with respect to the image array. A fast-moving image element may easily move 20 pixels in one frame time. In the presence of motion, a pixel at a certain location may take quite different values in successive pictures. Motion is liable to cause prediction

\[ \Delta_1 \Delta_2 \Delta_3 \Delta_4 \Delta_5 \Delta_6 \Delta_7 \Delta_8 \]

Reference value

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.
Apart from scene changes, there is a statistical likelihood that successive pictures in a video sequence are very similar. In fact, it is necessary that successive pictures are mostly similar: If this were not the case, human vision could make no sense of the sequence!

The M in MPEG stands for moving, not motion! If you read a document that errs in this detail, the accuracy of the document is suspect!

The efficiency of transform coding can be increased by a factor of 10 or more by exploiting the inherent temporal redundancy of video. The MPEG standard was developed by the Moving Picture Experts Group within ISO and IEC. In MPEG, intra-frame compression is used to provide an initial, self-contained picture – a reference picture, upon which predictions of succeeding pictures can be based. The encoder then transmits pixel differences, that is, prediction errors, or residuals – from the reference, as sketched in Figure 16.1. The method is termed interframe coding.

Once a reference picture has been received by the decoder, it provides a basis for predicting succeeding pictures. This estimate is improved when the decoder receives the residuals. The scheme is effective provided that the residuals can be coded more compactly than the raw picture information.

Motion in a video sequence causes displacement of scene elements with respect to the image array. A fast-moving image element may easily move 20 pixels in one frame time. In the presence of motion, a pixel at a certain location may take quite different values in successive pictures. Motion is liable to cause prediction errors.

Reference value

\[ \Delta_1 \Delta_2 \Delta_3 \Delta_4 \Delta_5 \Delta_6 \Delta_7 \Delta_8 \]

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

Frame $t$ vs. Frame $t + 1$
Residuals

Frame $t$

Frame $t+1$

Residual

Ideal!
Residuals

Frame $t$

Frame $t + 1$

Residual

Ideal!
Residuals

Frame $t$

Residual

Frame $t+1$

Ideal!
Residuals

Frame t

Frame t + 1

Residual

Ideal!

Worse!
Residuals

Frame $t$

Frame $t + 1$

Residual

Ideal!

Worse!
Residuals

Frame $t$

9 9 9
9 9

Frame $t + 1$

9 9 9
9 9

Residual

Ideal!

Frame $t$

9 9 9
9 9

Frame $t + 1$

9 9 9
9 9

Residual

Worse!

Frame $t$

9 9

Frame $t + 1$

9 9

Motion vector: [2, 2]
Residuals

Frame $t$

Frame $t + 1$

Residual

Ideal!

Worse!

Motion vector: [2, 2]

Encode both!
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.
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.

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

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

The MB in the search window that has the lowest SAD wins. Each MB is associated with a motion vector (MV).
Motion Vectors

\[ MV = \langle u, v \rangle \]

Search window

Best match in frame \( N-1 \)

Macroblock in frame \( N \)

\[ 2d + 1 \]
Motion Vectors

A frame from a still camera capturing a moving person. Each arrow represents the MV of a 8x8 MB.
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?
Lots of heuristics are proposed. Always a trade-off between accuracy vs. compute efficiency. Exhaustive search (ES) represents one extreme.

Three step search (TSS) is a classic heuristics that adaptively narrows the search range (coarse-to-fine search).
Lots of heuristics are proposed. Always a trade-off between accuracy vs. compute efficiency. Exhaustive search (ES) represents one extreme.

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

- Lots of heuristics are proposed. Always a trade-off between accuracy vs. compute efficiency. Exhaustive search (ES) represents one extreme.
- Three step search (TSS) is a classic heuristics that adaptively narrows the search range (coarse-to-fine search).
Motion Estimation Heuristics

- Lots of heuristics are proposed. Always a trade-off between accuracy vs. compute efficiency. Exhaustive search (ES) represents one extreme.
- Three step search (TSS) is a classic heuristics that adaptively narrows the search range (coarse-to-fine search).
Lots of heuristics are proposed. Always a trade-off between accuracy vs. compute efficiency. Exhaustive search (ES) represents one extreme.

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

- Lots of heuristics are proposed. Always a trade-off between accuracy vs. compute efficiency. Exhaustive search (ES) represents one extreme.
- Three step search (TSS) is a classic heuristic 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.
Design Decisions

‣ **How large should the search window be?**
  ▶ Large search window can potentially find a more accurate motion vector.
  ▶ But large search window increases the search time and the storage for the motion vector.
  ▶ If the search window is 15x15, the motion vector (on each dimension) is between -7 and 7, which can be encoded in 4 bits.

‣ **How large should the macroblock size be?**
  ▶ Small MBs should allow more accurate motion estimation, especially for non-rigid objects.
  ▶ Small MBs also reduces compute cost of motion estimation (keeping everything else).
  ▶ Small MBs increase the storage overhead for MB metadata.
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.

Fixed after the lecture. The residuals and MVs should be calculated for Frame (t+1). I did it for Frame t in the lecture.
Encoding Motion Vectors

- Motion vectors of nearby MBs are often correlated, especially for rigid objects.
- Idea: use nearby MVs to estimate/predict the current MV, compute the residuals, and encode the MV residuals, e.g., entropy encoding (lossless).

$\text{MV}_E \approx (\text{MV}_A + \text{MV}_B + \text{MV}_C)/3$
Inter-Frame Encoding Flow

Reference Frame

Current Frame

Data that needs to be encoded
Inter-Frame Encoding Flow

Reference Frame → Motion Estimation → Current Frame

Data that needs to be encoded
Inter-Frame Encoding Flow

Reference Frame → Motion Estimation → Motion Vectors

Data that needs to be encoded

Current Frame
Inter-Frame Encoding Flow

Data that needs to be encoded

Reference Frame → Motion Estimation → Motion Vectors → (Motion) Predicted Frame

Current Frame ⏯️
Inter-Frame Encoding Flow

Data that needs to be encoded

Reference Frame → Motion Estimation → Motion Vectors → + → (Motion) Predicted Frame

A fancy name is “motion compensated interpolation”
Inter-Frame Encoding Flow

Data that needs to be encoded

Reference Frame → Motion Estimation → Motion Vectors → (Motion) Predicted Frame

“Prediction”

Current Frame

A fancy name is “motion compensated interpolation”
Inter-Frame Encoding Flow

Data that needs to be encoded

Reference Frame → Motion Estimation → Motion Vectors → Residuals

Current Frame

“Prediction”

(Motion) Predicted Frame

A fancy name is “motion compensated interpolation”

("Prediction")
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

- In MPEG, a video is partitioned into successive groups of pictures (GoPs), each of which contains a sequence of successive frames.
- There are two (not mutually exclusive) kinds of frames in a GoP:

![Reference Frame Choice Diagram](image-url)
Reference Frame Choice

- In MPEG, a video is partitioned into successive groups of pictures (GoPs), each of which contains a sequence of successive frames.
- There are two (not mutually exclusive) kinds of frames in a GoP:
  - **Reference** frame: frames that could be used to motion-estimate (predict) other frames.

![1 GoP](image)
Reference Frame Choice

- In MPEG, a video is partitioned into successive groups of pictures (GoPs), each of which contains a sequence of successive frames.
- There are two (not mutually exclusive) kinds of frames in a GoP:
  - **Reference** frame: frames that could be used to motion-estimate (predict) other frames.
  - **Predicted** frame: frames that are predicted, i.e., motion-estimated, from reference frames. A predicted frame can be a reference frame.
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.
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 predicted from a reference frame, and itself can be used as a reference frame.

![Diagram of GoP and frames](image-url)
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, 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, but itself can’t be used as a reference frame.

[Diagram of 1 GoP with I-frame 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and B-frames]

Two B-frames. Usually a B-frame is predicted as 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

$I_0B_1B_2P_3B_4B_5P_6B_7B_8$

Two downsides:
1. When encoding, need to buffer $B_1$ and $B_2$ before $P_3$ arrives.
2. When decoding, need to buffer $B_1$ and $B_2$ before $P_3$ arrives.

Bad for live streaming/teleconferencing!!!
Reordering Frames in a GoP

$I_0B_1B_2P_3B_4B_5P_6B_7B_8$

Two downsides:
1. When encoding, need to buffer $B_1$ and $B_2$ before $P_3$ arrives.
2. When decoding, need to buffer $B_1$ and $B_2$ before $P_3$ arrives.
Bad for live streaming/teleconferencing!!

$I_0P_3B_1B_2P_6B_4B_5(I_9)B_7B_8$
Reordering Frames in a GoP

$I_0B_1B_2P_3B_4B_5P_6B_7B_8$

Two downsides:
1. When encoding, need to buffer $B_1$ and $B_2$ before $P_3$ arrives.
2. When decoding, need to buffer $B_1$ and $B_2$ before $P_3$ arrives.

Bad for live streaming/teleconferencing!!

$I_0P_3B_1B_2P_6B_4B_5(I_9)B_7B_8$

Solution: reorder frames during transmission!
INTRODUCTION TO VIDEO COMPRESSION
CHAPTER 16

The three-level MPEG picture hierarchy is summarized in Figure 16.5 above; this example has the structure IBBPBBPBB.

A 15-frame “long” GoP structured IBBPBBPBBPBB is common for broadcasting. A sophisticated encoder may produce irregular GoP patterns.

A simple encoder typically produces a bitstream having a fixed schedule of I-, P-, and B-pictures. A typical GoP structure is denoted IBBPBBPBB. At 30 pictures per second, there are two such GoPs per second. Periodic GoP structure can be described by a pair of integers \( n \) and \( m \); \( n \) is the number of pictures from one I-picture (inclusive) to the next (exclusive), and \( m \) is the number of pictures from one anchor picture (inclusive) to the next (exclusive). If \( m = 1 \), there are no B-pictures. Figure 16.5 shows a regular GoP structure with an I-picture interval of \( n = 9 \) and an anchor-picture interval of \( m = 3 \). The \( m = 3 \) component indicates two B-pictures between anchor pictures.

Rarely do more than 2 B-pictures intervene between reference pictures.

Coded B-pictures in a GoP depend upon P- and I-pictures; coded P-pictures depend upon earlier P-pictures and I-pictures. Owing to these interdependencies, an MPEG sequence cannot be edited, except at GoP boundaries, unless the sequence is decoded, edited, and subsequently reencoded. MPEG is very suitable...
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 $P_3'$. 

Digital Video and HDTV Algorithms and Interfaces, Figure 16.5
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 $P_3'$. 

![Diagram showing the three-level MPEG picture hierarchy](image)
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

The H.264 advanced video compression standard, Figure 4.4
The H.264 advanced video compression standard, Figure 4.4

Both intra- and inter-frame prediction
The H.264 advanced video compression standard, Figure 4.4

Both intra- and inter-frame prediction

The decoder inside the encoder
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.
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.
- Allow different MB sizes.

![Real pixels](Image)

Interpolated half pixels

Interpolated quarter pixels

**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.
- Allow different MB sizes.

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

Different MB sizes in an I-frame.

Different MB sizes in a P-frame.
Video Compression Recap

- M-JPEG (and the I-frame only mode in MPEG-2) compresses images individually. Not very effective as it ignores the nature of videos.
- H.264 and H.265 exploit two additional dimensions.

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

- **Temporal redundancy**
  - Pixels from continuous frames have strong correlation. P/B-frames are temporally predicted.

- Encoding time is dominated by motion estimation. Encoding and decoding are widely support by fix-functioned hardware (ASICs).
Video Transcoding

Video Transcoding

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


vbench: Benchmarking Video Transcoding in the Cloud, ASPLOS 2018
Video Transcoding

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

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

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


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
Reference Material

- Digital Video and HD: Algorithms and Interfaces, Chapter 16 & 45
- The H.264 Advanced Video Compression Standard, Chapter 3
- vbench: Benchmarking Video Transcoding in the Cloud, ASPLOS 2018
- Lossy Image Compression with Compressive Autoencoders, ICLR 2017
- Euphrates: Algorithm-SoC Co-Design for Low-Power Mobile Continuous Vision, ISCA 2018