JPEG and H.26x Standards

- Video Data Size and Bit Rate
- DCT Transform and Quantization
- JPEG Standard for Still Image
- Intra-frame and Inter-frame Compression
- Block-based Motion Compensation
- H.261 Standard for Video Compression
- H.263, H.263+, H.263++, H.26L, H.264
Video Bit Rate Calculation

- **width** ~ pixels (160, 320, 640, 720, 1280, 1920, …)
- **height** ~ pixels (120, 240, 480, 485, 720, 1080, …)
- **depth** ~ bits per pixel (1, 4, 8, 15, 16, 24, …)
- **fps** ~ frames per second (5, 15, 20, 24, 30, …)
- **compression factor** (1 ~ 100 ~ )

\[
\frac{\text{width} \times \text{height} \times \text{depth} \times \text{fps}}{\text{compression factor}} = \text{bits/sec (bps)}
\]

One Frame = 3 pictures (YCrCb)
### Uncompressed Video Data Size

**compression factor = 1**

*Size of uncompressed video in gigabytes*

<table>
<thead>
<tr>
<th>Video Resolution</th>
<th>1 sec</th>
<th>1 min</th>
<th>1 hour</th>
<th>1000 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920x1080</td>
<td>0.19</td>
<td>11.20</td>
<td>671.85</td>
<td>671,846.40</td>
</tr>
<tr>
<td>1280x720</td>
<td>0.08</td>
<td>4.98</td>
<td>298.60</td>
<td>298,598.40</td>
</tr>
<tr>
<td>640x480</td>
<td>0.03</td>
<td>1.66</td>
<td>99.53</td>
<td>99,532.80</td>
</tr>
<tr>
<td>320x240</td>
<td>0.01</td>
<td>0.41</td>
<td>24.88</td>
<td>24,883.20</td>
</tr>
<tr>
<td>160x120</td>
<td>0.00</td>
<td>0.10</td>
<td>6.22</td>
<td>6,220.80</td>
</tr>
</tbody>
</table>

### Image size of video

- **1280x720 (1.77)**
- **640x480 (1.33)**
- **320x240**
- **160x120**
Effects of Compression

Storage for 1 hour of compressed video in megabytes

Compression ratio

<table>
<thead>
<tr>
<th></th>
<th>1920x1080</th>
<th>1280x720</th>
<th>640x480</th>
<th>320x240</th>
<th>160x120</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1</td>
<td>671,846</td>
<td>298,598</td>
<td>99,533</td>
<td>24,883</td>
<td>6,221</td>
</tr>
<tr>
<td>3:1</td>
<td>223,949</td>
<td>99,533</td>
<td>33,178</td>
<td>8,294</td>
<td>2,074</td>
</tr>
<tr>
<td>6:1</td>
<td>111,974</td>
<td>49,766</td>
<td>16,589</td>
<td>4,147</td>
<td>1,037</td>
</tr>
<tr>
<td>25:1</td>
<td>26,874</td>
<td>11,944</td>
<td>3,981</td>
<td>995</td>
<td>249</td>
</tr>
<tr>
<td>100:1</td>
<td>6,718</td>
<td>2,986</td>
<td>995</td>
<td>249</td>
<td>62</td>
</tr>
</tbody>
</table>

3 bytes/pixel, 30 frames/sec
Coding Overview

• Digitize
  – Subsample to reduce data
• Compression algorithms exploit:
  – Spatial redundancy - correlation between neighboring pixels
    • **Intra-frame** compression
    • remove redundancy within frame
  – Temporal redundancy - correlation betw. frames
    • **Inter-frame** compression
    • Remove redundancy between frames
• Symbol Coding
  – Efficient coding of sequence of symbols
    • RLC (Run Length Coding)
    • Huffman coding
Transform Coding

- An image conversion process that transforms an image from the spatial domain to the frequency domain.
- Subdivide an individual $N \times M$ image into small $n \times n$ blocks.
- Each $n \times n$ block undergoes a reversible transformation.
- Basic approach:
  - De-correlate the original block - radiant energy is redistributed amongst only a small number of transform coefficients.
  - Discard many of the low energy coefficients (through quantization).

```
\begin{align*}
f(i,j) & \quad \xrightarrow{\text{Transform Function}} \quad F(u,v) \quad \xrightarrow{\text{Quantizer}} \quad Fq(u,v)
\end{align*}
```
DCT – $n \times n$ Discrete Cosine Transform

\[
F = D \times f
\]

$F, D, f$ are $n$-by-$n$ matrixes

\[
F[u,v] = \frac{4C(u)C(v)}{n^2} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos \left( \frac{(2j+1)u\pi}{2n} \right) \cos \left( \frac{(2k+1)v\pi}{2n} \right)
\]

where

\[
C(w) = \begin{cases} 
\frac{1}{\sqrt{2}} & \text{for } w=0 \\
1 & \text{for } w=1,2,\ldots,n-1
\end{cases}
\]

- IDCT is very similar
- 8x8 DCT coefficients

<table>
<thead>
<tr>
<th></th>
<th>0.7</th>
<th>0.7</th>
<th>0.7</th>
<th>0.7</th>
<th>0.7</th>
<th>0.7</th>
<th>0.7</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.8</td>
<td>0.6</td>
<td>0.2</td>
<td>-0.2</td>
<td>-0.6</td>
<td>0.8</td>
<td>-1</td>
</tr>
<tr>
<td>0.9</td>
<td>0.9</td>
<td>0.4</td>
<td>-0.4</td>
<td>-0.9</td>
<td>-0.9</td>
<td>-0.4</td>
<td>0.4</td>
<td>0.9</td>
</tr>
<tr>
<td>0.8</td>
<td>0.8</td>
<td>-0.2</td>
<td>-1</td>
<td>-0.6</td>
<td>0.6</td>
<td>1</td>
<td>0.2</td>
<td>-0.8</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7</td>
<td>-0.7</td>
<td>-0.7</td>
<td>0.7</td>
<td>0.7</td>
<td>-0.7</td>
<td>-0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6</td>
<td>-1</td>
<td>0.2</td>
<td>0.8</td>
<td>-0.8</td>
<td>-0.2</td>
<td>1</td>
<td>-0.6</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4</td>
<td>-0.9</td>
<td>0.9</td>
<td>-0.4</td>
<td>-0.4</td>
<td>0.9</td>
<td>-0.9</td>
<td>0.4</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
<td>-0.6</td>
<td>0.8</td>
<td>-1</td>
<td>1</td>
<td>-0.8</td>
<td>0.6</td>
<td>-0.2</td>
</tr>
</tbody>
</table>
Quantization

- **Purpose of quantization**
  - Achieve high compression by representing DCT coefficients with no greater precision than necessary
  - Discard information which is not visually significant
- **After output from the FDCT, each of the 64 DCT coefficients is quantized**
  - Many-to-one-mapping => fundamentally lossy process
  - \( F_{q}[u,v] = \text{Round} \left( \frac{F[u,v]}{q[u,v]} \right) \)
  - Example: \( F[u,v] = 101101 = 45 \) (6 bits).
    If \( q[u,v] = 4 \), truncate to 4 bits, \( F_{q}[u,v] = 1011 \)

\[
\begin{align*}
\text{Example:} & \quad F[u,v] = \begin{bmatrix} 45 & 12 \\ 8 & 3 \end{bmatrix} \\
& \quad Q[u,v] = \begin{bmatrix} 4 & 6 \\ 6 & 8 \end{bmatrix} \\
& \quad F_{q}[u,v] = \begin{bmatrix} 11 & 2 \\ 1 & 0 \end{bmatrix}
\end{align*}
\]

- Quantization is the principal source of lossiness in DCT-based encoders
- Uniform quantization: each \( F[u,v] \) is divided by the same constant \( N \)
- Non-uniform quantization: use quantization tables from psycovisual experiments to exploit the limit of human visual system
### DCT and Quantization Example

DC component, others called AC

<table>
<thead>
<tr>
<th></th>
<th>139</th>
<th>144</th>
<th>149</th>
<th>153</th>
<th>155</th>
<th>155</th>
<th>155</th>
<th>155</th>
</tr>
</thead>
<tbody>
<tr>
<td>144</td>
<td>151</td>
<td>153</td>
<td>156</td>
<td>159</td>
<td>156</td>
<td>156</td>
<td>156</td>
<td>156</td>
</tr>
<tr>
<td>150</td>
<td>155</td>
<td>160</td>
<td>163</td>
<td>158</td>
<td>156</td>
<td>156</td>
<td>156</td>
<td>156</td>
</tr>
<tr>
<td>159</td>
<td>161</td>
<td>162</td>
<td>160</td>
<td>160</td>
<td>159</td>
<td>159</td>
<td>159</td>
<td>159</td>
</tr>
<tr>
<td>159</td>
<td>160</td>
<td>161</td>
<td>162</td>
<td>162</td>
<td>155</td>
<td>155</td>
<td>155</td>
<td>155</td>
</tr>
<tr>
<td>161</td>
<td>161</td>
<td>161</td>
<td>161</td>
<td>160</td>
<td>157</td>
<td>157</td>
<td>157</td>
<td>157</td>
</tr>
<tr>
<td>162</td>
<td>161</td>
<td>161</td>
<td>163</td>
<td>162</td>
<td>157</td>
<td>157</td>
<td>157</td>
<td>157</td>
</tr>
<tr>
<td>162</td>
<td>161</td>
<td>161</td>
<td>161</td>
<td>163</td>
<td>158</td>
<td>158</td>
<td>158</td>
<td>158</td>
</tr>
</tbody>
</table>

(a) source image samples $\mathbf{f}$

<table>
<thead>
<tr>
<th></th>
<th>16</th>
<th>11</th>
<th>10</th>
<th>16</th>
<th>24</th>
<th>40</th>
<th>51</th>
<th>61</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>26</td>
<td>58</td>
<td>60</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>57</td>
<td>69</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>22</td>
<td>29</td>
<td>51</td>
<td>87</td>
<td>80</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>22</td>
<td>37</td>
<td>56</td>
<td>68</td>
<td>109</td>
<td>103</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>35</td>
<td>55</td>
<td>64</td>
<td>81</td>
<td>104</td>
<td>113</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>64</td>
<td>78</td>
<td>87</td>
<td>103</td>
<td>121</td>
<td>120</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>112</td>
<td>100</td>
<td>103</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>

(b) forward DCT coefficients $\mathbf{F}$

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>0</th>
<th>-1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>240</th>
<th>0</th>
<th>-10</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-24</td>
<td>-12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-14</td>
<td>-13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(c) quantization table $\mathbf{Q}$

<table>
<thead>
<tr>
<th></th>
<th>144</th>
<th>146</th>
<th>149</th>
<th>152</th>
<th>154</th>
<th>156</th>
<th>156</th>
<th>156</th>
</tr>
</thead>
<tbody>
<tr>
<td>148</td>
<td>150</td>
<td>152</td>
<td>154</td>
<td>156</td>
<td>156</td>
<td>156</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>156</td>
<td>157</td>
<td>158</td>
<td>158</td>
<td>157</td>
<td>156</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>161</td>
<td>161</td>
<td>162</td>
<td>161</td>
<td>159</td>
<td>157</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>163</td>
<td>164</td>
<td>163</td>
<td>162</td>
<td>160</td>
<td>158</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>164</td>
<td>164</td>
<td>164</td>
<td>162</td>
<td>160</td>
<td>158</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>161</td>
<td>162</td>
<td>162</td>
<td>162</td>
<td>161</td>
<td>159</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>159</td>
<td>161</td>
<td>161</td>
<td>162</td>
<td>161</td>
<td>159</td>
<td>158</td>
<td></td>
</tr>
</tbody>
</table>

(d) normalized quantized $\mathbf{Fq}$

### Denormalized Quantized Coefficients $\mathbf{F}^{-1}$

<table>
<thead>
<tr>
<th></th>
<th>235.6</th>
<th>-1.0</th>
<th>12.1</th>
<th>-5.2</th>
<th>2.1</th>
<th>-1.7</th>
<th>-2.7</th>
<th>1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-22.6</td>
<td>-17.5</td>
<td>-6.2</td>
<td>-3.2</td>
<td>-2.9</td>
<td>-0.1</td>
<td>0.4</td>
<td>-1.2</td>
<td></td>
</tr>
<tr>
<td>-10.9</td>
<td>-9.3</td>
<td>-1.6</td>
<td>1.5</td>
<td>0.2</td>
<td>-0.9</td>
<td>-0.6</td>
<td>-0.1</td>
<td></td>
</tr>
<tr>
<td>-7.1</td>
<td>-1.9</td>
<td>0.2</td>
<td>1.5</td>
<td>0.9</td>
<td>-0.1</td>
<td>0.0</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>-0.6</td>
<td>-0.8</td>
<td>1.5</td>
<td>1.6</td>
<td>-0.1</td>
<td>-0.7</td>
<td>0.6</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>-0.2</td>
<td>1.6</td>
<td>-0.3</td>
<td>-0.8</td>
<td>1.5</td>
<td>1.0</td>
<td>-1.0</td>
<td></td>
</tr>
<tr>
<td>-1.3</td>
<td>-0.4</td>
<td>-0.3</td>
<td>-1.5</td>
<td>-0.5</td>
<td>1.7</td>
<td>1.1</td>
<td>-0.8</td>
<td></td>
</tr>
<tr>
<td>-2.6</td>
<td>1.6</td>
<td>-3.8</td>
<td>-1.8</td>
<td>1.9</td>
<td>1.2</td>
<td>-0.6</td>
<td>-0.4</td>
<td></td>
</tr>
</tbody>
</table>

(e) denormalized quantized $\mathbf{F}^{-1}$

<table>
<thead>
<tr>
<th></th>
<th>144</th>
<th>146</th>
<th>149</th>
<th>152</th>
<th>154</th>
<th>156</th>
<th>156</th>
<th>156</th>
</tr>
</thead>
<tbody>
<tr>
<td>148</td>
<td>150</td>
<td>152</td>
<td>154</td>
<td>156</td>
<td>156</td>
<td>156</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>156</td>
<td>157</td>
<td>158</td>
<td>158</td>
<td>157</td>
<td>156</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>161</td>
<td>161</td>
<td>162</td>
<td>161</td>
<td>159</td>
<td>157</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>163</td>
<td>164</td>
<td>163</td>
<td>162</td>
<td>160</td>
<td>158</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>164</td>
<td>164</td>
<td>164</td>
<td>162</td>
<td>160</td>
<td>158</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>161</td>
<td>162</td>
<td>162</td>
<td>162</td>
<td>161</td>
<td>159</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>159</td>
<td>161</td>
<td>161</td>
<td>162</td>
<td>161</td>
<td>159</td>
<td>158</td>
<td></td>
</tr>
</tbody>
</table>

(f) reconstructed image samples $\mathbf{f}^{-1}$
JPEG Image Compression Standard

- Mainly for still image (gray and color)
- Four Modes:
  - Lossless JPEG
  - Sequential (Baseline) JPEG
  - Progressive JPEG
  - Hierarchical JPEG
- Hybrid Coding Techniques:
  - DCT Coding
  - Run Length Encoding (RLE)
  - Huffman Coding
  - Linear Prediction (only in lossless mode)
- New Standard: JPEG2000
- Motion JPEG for video
Overview of Baseline JPEG

YCrCb

8 x 8

DCT

Quantization

Quantiz. Tables

Coding Tables

Entropy Coding

DPCM

RLC

Header Tables

Data

.fg file

F(u, v)

Fq(u, v)
Block Transform Encoding

- Block Transform
- DCT (Discrete Cosine Transform)
- Zig-zag ordering
- Quantize
- Run-length Code
- Huffman Code

encoded sequence: 011010001011101...
Example of Block Encoding

original image

\[
\begin{bmatrix}
139 & 144 & 149 & 153 \\
144 & 151 & 153 & 156 \\
150 & 155 & 160 & 163 \\
159 & 161 & 162 & 160 \\
\end{bmatrix}
\]

DCT

\[
\begin{bmatrix}
1260 & -1 & -12 & -5 \\
-23 & -17 & -6 & -3 \\
-11 & -9 & -2 & 2 \\
-7 & -2 & 0 & 1 \\
\end{bmatrix}
\]

Quantize Table

\[
\begin{bmatrix}
15 & 10 & 10 & 16 \\
10 & 16 & 12 & 16 \\
10 & 8 & 16 & 16 \\
16 & 12 & 16 & 32 \\
\end{bmatrix}
\]

AC components

\[
\begin{bmatrix}
79 & 0 & -2 & -1 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

zigzag

\[
\begin{bmatrix}
79 & 0 & -1 & 0 \\
-2 & -1 & 0 & 0 \\
-1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

run-length code

\[
\begin{bmatrix}
0 & 79 \\
1 & -2 \\
0 & -1 \\
0 & -1 \\
2 & -1 \\
0 & 0 \\
\end{bmatrix}
\]

Huffman code

\[100110111000011...\]

coded bitstream < 10 bits (0.55 bits/pixel)
Result of Coding/Decoding

\[
\begin{bmatrix}
139 & 144 & 149 & 153 \\
144 & 151 & 153 & 156 \\
150 & 155 & 160 & 163 \\
159 & 161 & 162 & 160
\end{bmatrix}
\quad\text{original block}
\]

\[
\begin{bmatrix}
144 & 146 & 149 & 152 \\
156 & 150 & 152 & 154 \\
155 & 156 & 157 & 158 \\
160 & 161 & 161 & 162
\end{bmatrix}
\quad\text{reconstructed block}
\]

\[
\begin{bmatrix}
-5 & -2 & 0 & 1 \\
-4 & 1 & 1 & 2 \\
-5 & -1 & 3 & 5 \\
-1 & 0 & 1 & -2
\end{bmatrix}
\quad\text{errors}
\]

Small Loss
Neglect-able
Examples

Uncompressed
(262 KB)
8 bits/pixel

Compressed (50)
(22 KB, 12:1)
0.67 bit/pixel

Compressed (1)
(6 KB, 43:1)
0.17 bit/pixel
JPEG vs. GIF

- **JPEG Advantages**
  - more colors (GIF limited to 256)
  - lossless option
  - best for scanned photographs
  - progressive JPEG downloads rough image before whole image arrives

- **GIF Advantages**
  - transparent color setting
  - animated GIFs
  - better for flat color fields: clip art, cartoons, etc.
  - interlaced delivery downloads low resolution image before whole image arrives
Intra- vs. Inter-frame Compression

- **Intra-frame compression**
  - For still image like JPEG
  - Exploit the redundancy in image (*spatial redundancy*)
  - Can be applied to individual frames in a video sequence

- **Techniques**
  - Subsampling (small size)
  - Block transform coding
  - Coarse quantization

- **Intra + inter-frame compression**
  - For video like H.26x & MPEG
  - Exploit the similarities between successive frames (*temporal redundancy*)

- **Techniques**
  - Subsampling (small frame rate)
  - Difference coding
  - Block-based difference coding
  - Block-based motion compensation
Difference Coding

• Compare pixels with previous frame
  – Only pixels that have been changed are updated
  – A fraction of the number of pixel values will be recorded

• Overhead associated with which pixels are updated: what if a large number of pixels are changed?
• Pixels values are slightly different even with no movement of objects: ignore small changes (lossy)
Block-based Difference Coding

• Difference coding *at the block level*
  – Send sequence of blocks rather than frames
  – If previous block similar, skip it or send difference
  – Update a whole block of pixels at once
  – 160 x 120 pixels (19200 pixels) => 8x8 blocks (300 blocks)
  – Possible artifact at the border of blocks

• Limitations of difference coding
  – Useless where there is a lot of motion (few pixels unchanged)
  – What if a camera itself is moving ?

• Need to compensate for object motion
Block-based Motion Compensation

• **Motion compensation** assumes that current frame can be modeled as a translation of a previous frame
• Search around block in previous frame for a better matching block and encode position and error difference
Block-based Motion Compensation

- Current frame is divided into *uniform non-overlapping blocks*
- Each block in the current frame is compared to areas of similar size from the preceding frame in order to find an area that is similar
- The relative difference in locations is known as the *motion vector*
- Because fewer bits are required to code a motion vector than to code actual blocks, compression is achieved.
Bidirectional Motion Compensation

- **Bidirectional motion compensation**
  - Areas just uncovered are not predictable from the past, but can be predicted from the future
  - Search in *both past and future frames*

- Effect of noise and errors can be reduced by averaging between previous and future frames

- Bi-directional interpolation provides a high degree of compression
  - Requires that frames be encoded and transmitted in a different order from which they will be displayed.

- In reality, exact matching is not possible, thus lossy compression
Overview of H.261

• Developed by CCITT (Consultative Committee for International Telephone and Telegraph) in 1988-1990
• Designed for videoconferencing, video-telephone applications over ISDN telephone lines.
  – Bit-rate is $p \times 64$ Kbps, where $p$ ranges from 1 to 30 (2048 kbps)
• Supports CCIR 601 CIF (352 x 288) and QCIF (176 x 144) images with 4:2:0 subsampling.
• Significant influence on H.263, MPEG 1-4, etc.
Frame Sequence of H.261

- Two frame types: Intra-frames (*I-frames*) and Inter-frames (*P-frames*): I-frame provides an accessing point, it uses basically JPEG.
- P-frames use "pseudo-differences" from previous frame ("predicted"), so frames depend on each other.
**Intra-frame Coding**

- **Macroblock:**
  - 16 x 16 pixel areas on Y plane of original image.
  - Usually consists of 4 Y blocks, 1 Cr block, and 1 Cb block (4:2:0 or 4:1:1)

- **Quantization is by constant value for all DCT coefficients** (i.e., no quantization table as in JPEG).
Inter-frame Coding

Diagram showing the process of inter-frame coding, including steps like matching, motion vector, difference, DCT, quantization, run-length encoding (RLE), and Huffman coding.
Motion Vector Searches

C(x+k, y+l): macro block pixels in the target
R(x+i+k, y+j+l): macro block pixels in the reference

\[
\text{MAE}(i, j) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} |C(x+k, y+l) - R(x+i+k, y+j+l)|
\]

The goal is to find a vector \((u, v)\) such that the mean Absolute Error, \(\text{MAE}(u, v)\) is minimum:
1. Full Search Method
2. Two-dimensional Logarithmic Search
3. Hierarchical Motion Estimation
Encoder

I-Frames:
- Video In
- DCT
- Quant
- Control
- MV search & memory
- IDCT
- Quant
- Control
- MV search & memory
- IDCT
- Huffman

P-Frames:
- Video In
- DCT
- Quant
- Control
- MV search & memory
- IDCT
- Huffman
- SI034 format
- Motion vectors
H.262, H.263 and H.264

- H.262 = MPEG-2 jointly by ITU and ISO/IEC
  - Current best standard for practical video telecommunication
  - Has overtaken H.261 as videoconferencing codec
  - Superior to H.261 at all bit rates (1/2)
  - Video size: Sub-QCIF (128x96), QCIF (176x144), CIF (352x288), 4CIF (704x576), 16CIF (1408x1152)
  - PB frames mode (bidirectional prediction)
  - 4 motion vector for each block, $\frac{1}{2}$ pixel accuracy
  - Arithmetic coding efficient than Huffman coding in H.261
- H.263 v2 (H.263+, 1997)
- H.264/AVC (now)
Demos of Image GIF and JPEG Coding