ANALOG AND DIGITAL VIDEO

Henning Schulzrinne
Columbia University
COMS 6181 - Spring 2015

with material from Mark Handley
Objectives

• Understand the concept of display gamma
• How are video pixels represented?
• What is lossless coding?
• How do JPEG, PNG and GIF work?
• How does MPEG reduce the bit rate?
Gamma correction

- non-linear transformation between value and brightness
  \[ V_{\text{out}} = AV_{\text{in}}^\gamma \]
- similar to \( \mu \)-law in audio
  - brightness sensitivity differs non-linearly

\[
\gamma = \frac{d \log(V_{\text{out}})}{d \log(V_{\text{in}})}
\]
Video types

- Bi-level images: black and white
  - fax, printed output (at pixel level)
- Gray level (monochrome) images
- Color (continuous tone)

<table>
<thead>
<tr>
<th>Image type</th>
<th>pixels per frame</th>
<th>bits/pixel</th>
<th>uncompressed size</th>
</tr>
</thead>
<tbody>
<tr>
<td>fax (200 dpi)</td>
<td>1700x2200</td>
<td>1</td>
<td>3.75 Mb</td>
</tr>
<tr>
<td>VGA</td>
<td>640x480</td>
<td>8</td>
<td>2.46 Mb</td>
</tr>
<tr>
<td>XVGA</td>
<td>1024x768</td>
<td>24</td>
<td>18.87 Mb</td>
</tr>
</tbody>
</table>
Video formats

• SD (standard def. NTSC) = 646 x 486
• HDTV
  • progressive ("p") vs. interlaced ("i")
  • 480p = 852 x 480 pixels
  • 720p = 1280 x 720
  • 1080p = 1920 x 1080
• Aspect ratio:
  • TV: 4:3 (classical TV)
  • widescreen: 16:9 (HDTV, DVD)
Chroma subsampling

• Human eye more sensitive to luminance than chrominance details
• $J:a:b = \text{Pattern size (4)} : \text{chrominance first row} : \text{second row}$
• Should average, rather than just replicate
YUV Formats

- **YUV 4:4:4**
  - 8 bits per Y,U,V channel (no chroma subsampling)
- **YUV 4:2:2**
  - 4 Y pixels sample for every 2 U and 2V
  - 2:1 horizontal downsampling, no vertical downsampling
- **YUV 4:2:0**
  - 2:1 horizontal downsampling
  - 2:1 vertical downsampling
- **YUV 4:1:1**
  - 4 Y pixels sample for every 1 U and 1V
  - 4:1 horizontal downsampling, no vertical downsampling
YUV 4:2:0

YUV 4:2:0 (MPEG1/H.261/H.263)

Average from two lines
Video stream format

Video Stream Format

- **YUV 4:2:2 formats:**
  - YUV2: \[Y_0 U_0 V_0, Y_2 U_1 Y_3 V_1, Y_4 U_2 Y_5 V_2\]
  - UYVY: \[U_0 Y_0 V_0 Y_1, U_1 Y_2 V_1 Y_3, U_2 Y_4 V_2 Y_5\]

- **YUV 4:2:0 formats (12 bits per pixel packed format)**
  - YV12

  All the Y samples precede all the U samples, then all the V samples
Uncompressed video rates

<table>
<thead>
<tr>
<th>Format</th>
<th>resolution</th>
<th>sampling</th>
<th>bits/pixel</th>
<th>fps</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL</td>
<td>684x625</td>
<td>4:2:2</td>
<td>20</td>
<td>25</td>
<td>270 Mb/s</td>
</tr>
<tr>
<td>PAL</td>
<td>684x625</td>
<td>4:2:2</td>
<td>16</td>
<td>25</td>
<td>216 Mb/s</td>
</tr>
<tr>
<td>PAL</td>
<td>720x576</td>
<td>4:2:2</td>
<td>16</td>
<td>25</td>
<td>166 Mb/s</td>
</tr>
<tr>
<td>720p</td>
<td>1280x720</td>
<td>4:2:0</td>
<td>24</td>
<td>60</td>
<td>663 Mb/s</td>
</tr>
<tr>
<td>1080p</td>
<td>1920x1080</td>
<td>4:2:0</td>
<td>24</td>
<td>60</td>
<td>1.49 Gb/s</td>
</tr>
</tbody>
</table>

Thunderbolt: 20 Gb/s PCIe
USB: < 4 Gb/s
Image & video compression – in brief

• unlike audio, no physiological model (masking)
  • except lower color resolution than luminance

• statistical redundancy
  • background correlation
  • correlations across an image
  • nearby pixel correlation
  • frame correlation (motion compensation)

• subjective redundancy
  • impact of different impairments
  • block artifacts, noise, stair step (“jaggies”), …
Image compression

- TIFF (tagged image file format) – container file
- XBM, BMP (bitmap image format) - uncompressed
- GIF (Graphics Interchange Format)
  - including “animated GIF”
- PNG (Portable Network Graphics)
- MNG (Multiple-image Network Graphics)
- JPEG (Joint Picture Expert Group)
- JPEG-2000
GIF (Graphics Interchange Format)

- Lossless compression for computer-generated images
- CompuServ 1987 (GIF87a)
- GIF89a: metadata, multiple images ("animated")
- Indexed image format:
  - 256 colors from palette → not suitable for photography
  - one color index may indicate transparency
  - lossless LZW compression
  - interlacing optional
- First image format for NCSA Mosaic
- Good for diagrams, logos, icons, ...
  - avoids speckling of sharp edges (writing)
GIF patent issues

- 1984: algorithm published in *IEEE Computer* magazine
- 1985: LZW patent US 4558302 issued to Unisys
- 1987: CompuServ develops GIF
- 1994: license agreement, controversy
- 1995: PNG developed in response
- 2003/2004: patent expires
LZW compression

- dictionary contains longer and longer strings
- send dictionary index
  - possibly entropy-encoded

```php
dictionary = one entry per byte
string = ''
foreach ($input as $ch) {
    if (input + char in dictionary) {
        string += char
    } else {
        emit dictionary code for string
        add string + char to dictionary
        string = char
    }
}
output code for string
```
PNG (Portable Network Graphics)

- Lossless image format:
  - Palette-based (24 bit RGB)
  - RGB
  - Grayscale
- Does not support other color spaces (e.g., CMYK)
- RFC 1951
- Compression:
  - line-by-line filter (predictor) → see DPCM
    - byte to left, byte above, average of left & above, Paeth filter
  - DEFLATE (zlib, LZ77 + Huffman)
LZ77

• Abraham Lempel and Jacob Ziv in 1977
• dictionary code
• sliding window compression
• “each of the next length characters is equal to the characters exactly distance characters behind it in the uncompressed stream” (Wikipedia)
Huffman coding

- Goal: get close to entropy $H(x) = \sum p(x) \log(1/p(x))$
- Source coding theorem: exists coding $[H(x), H(x)+1)$
- Uniquely decodable
- Easy to decode $\rightarrow$ prefix code (“self-punctuating”)
  - no code word is a prefix of another code word
  - otherwise, would need delimiters
- Huffman: 1951 student paper
Huffman algorithm

- Take the two least probable symbols in the alphabet
  - become longest code words, differing in last bit
- Combine into single symbol
- Repeat
Example

- $A_x = \{a, b, c, d, e\}$
- $P_x = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

$$
\begin{array}{|c|c|c|c|c|}
\hline
a & 0.25 & 2.0 & 2 & 00 \\
\hline
b & 0.25 & 2.0 & 2 & 10 \\
\hline
c & 0.2 & 2.3 & 2 & 11 \\
\hline
d & 0.15 & 2.7 & 3 & 010 \\
\hline
e & 0.15 & 2.7 & 3 & 011 \\
\hline
\end{array}
$$
Huffman limitations

- Optimal only for independent symbols
  - but most sources have correlated symbols (e.g., within word)
- Changing ensemble
Run-length encoding (RLE)

- Value (repeat)
  - $1110011111 \rightarrow 1 \ 3 \ 0 \ 2 \ 1 \ 5$
- Common for images (e.g., line)
  - horizontal and vertical
- JPEG DCT output
- easily reversible, lossless
GIF: 30,000 bytes

PNG: 83,257 bytes

JPEG: 53,401 bytes
JPEG (Joint Photographic Experts Group)

- Good for compressing photographic images
  - gradual changes in pixel chrominance & luminance
- not good for line-style graphics
  - edges in image (text, sharp lines)
- compression ratio of 10:1 achievable without visible loss.
- uses JFIF or EXIF file format for meta information:
  - Application Segment #0
  - include photographic, author and geo data
EXIF example

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aperture Value</td>
<td>4.594</td>
</tr>
<tr>
<td>Color Space</td>
<td>sRGB</td>
</tr>
<tr>
<td>Components Configuration</td>
<td>1, 2, 3, 0</td>
</tr>
<tr>
<td>Compressed Bits Per Pixel</td>
<td>3</td>
</tr>
<tr>
<td>Custom Rendered</td>
<td>Normal process</td>
</tr>
<tr>
<td>Date Time Digitized</td>
<td>2011:11:12 15:28:36</td>
</tr>
<tr>
<td>Date Time Original</td>
<td>2011:11:12 15:28:36</td>
</tr>
<tr>
<td>Digital Zoom Ratio</td>
<td>2.04</td>
</tr>
<tr>
<td>Exif Version</td>
<td>2.2</td>
</tr>
<tr>
<td>Exposure Bias Value</td>
<td>0</td>
</tr>
<tr>
<td>Exposure Mode</td>
<td>Auto exposure</td>
</tr>
<tr>
<td>Exposure Time</td>
<td>1 / 50</td>
</tr>
<tr>
<td>File Source</td>
<td>Flash did not fire, compulsory flash mode</td>
</tr>
<tr>
<td>FlashPix Version</td>
<td>1.0</td>
</tr>
<tr>
<td>FNumber</td>
<td>4.9</td>
</tr>
<tr>
<td>Focal Length</td>
<td>18.6</td>
</tr>
<tr>
<td>Focal Plane Resolution Un</td>
<td>inches</td>
</tr>
<tr>
<td>Focal Plane X Resolution</td>
<td>23,500.801</td>
</tr>
<tr>
<td>Focal Plane Y Resolution</td>
<td>23,466.035</td>
</tr>
<tr>
<td>ISO Speed Ratings</td>
<td>200</td>
</tr>
<tr>
<td>Max Aperture Value</td>
<td>4.594</td>
</tr>
<tr>
<td>Metering Mode</td>
<td>Pattern</td>
</tr>
<tr>
<td>Pixel X Dimension</td>
<td>2,592</td>
</tr>
<tr>
<td>Pixel Y Dimension</td>
<td>1,944</td>
</tr>
<tr>
<td>Scene Capture Type</td>
<td>Standard</td>
</tr>
<tr>
<td>Sensing Method</td>
<td>One-chip color area sensor</td>
</tr>
<tr>
<td>Shutter Speed Value</td>
<td>5.656</td>
</tr>
<tr>
<td>White Balance</td>
<td>Auto white balance</td>
</tr>
<tr>
<td>White Balance Info</td>
<td>0.41, 0.41, 0.18, 0.18, f</td>
</tr>
<tr>
<td>Firmware</td>
<td>Firmware Version 1.01</td>
</tr>
<tr>
<td>Flash Compensation</td>
<td>0</td>
</tr>
<tr>
<td>Focus Mode</td>
<td>1</td>
</tr>
<tr>
<td>ImageStabilization</td>
<td>3</td>
</tr>
<tr>
<td>Lens Info</td>
<td>6.2, 18.6, 0,0</td>
</tr>
<tr>
<td>Lens Model</td>
<td>6.2–18.6 mm</td>
</tr>
</tbody>
</table>
JPEG

- Convert RGB (24 bit) data to YUV
  - typically, 4:2:0
  - → three sub-images: Y, Cb, Cr
  - Cb, Cr half the width & height of Y image
- Divide each image into 8x8 tiles
- Convert into frequency space: two-dimensional DCT
- Quantize in frequency domain
  - lower frequencies → more bits/value
- Encode quantized values using Huffman and RLE zig-zag manner
JPEG Diagram

Encoder

RGB->YUV → FDCT → Quantizer → Entropy Encoder

Decoder

YUV->RGB → IDCT → Quantizer → Entropy Decoder

Raster Image

8x8 block

Quantization Tables

Huffman Tables

JPEG Compressed Bitstream
JPEG example

![original 8x8 luminance block](image)

<table>
<thead>
<tr>
<th>Sample values</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
</tr>
<tr>
<td>63</td>
</tr>
<tr>
<td>62</td>
</tr>
<tr>
<td>63</td>
</tr>
<tr>
<td>67</td>
</tr>
<tr>
<td>79</td>
</tr>
<tr>
<td>85</td>
</tr>
<tr>
<td>87</td>
</tr>
</tbody>
</table>
Subtract 128 from each value to convert to signed

Then apply FDCT:

\[
T(i, j) = c_i c_j \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} V(y, x) \cos \left( \frac{(2y + 1)i\pi}{2N} \right) \cos \left( \frac{(2x + 1)j\pi}{2N} \right)
\]

\[
c_i = \sqrt{1/N} \quad \text{if } i = 0, \quad c_i = \sqrt{2/N} \quad \text{otherwise. Similarly } c_j
\]

Giving:

\[
\begin{array}{cccccccc}
-415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\
5 & -22 & -61 & 10 & 13 & -7 & -8 & 5 \\
-47 & 7 & 77 & -24 & -29 & 10 & 5 & -6 \\
-49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\
12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\
-8 & 3 & 2 & -6 & -3 & 1 & 4 & 2 \\
-1 & 0 & 0 & -3 & -1 & -3 & 4 & -1 \\
0 & 0 & -1 & -4 & -1 & 0 & 0 & 2
\end{array}
\]

Note DC Coefficient has lots of power

Very little power in high frequencies
DCT basis functions

DC coefficient

highest frequency
Quantize using a quantization matrix such as:

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

Better quantization at low frequencies

Coarse quantization at high frequencies

Giving:

\[
\begin{pmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

Eg round(-415/16) = -26

High frequencies often quantize to zero
Quantized DCT coefficients:

\[
\begin{pmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

Scaled DCT basis functions that make up the (quantized) image

Original Image:
Order the coefficients in zig-zag order:

\[
\begin{pmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

\[
-26, -3, 0, -3, -2, -6, 2, -4, 1, -4, 1, 1, 5, 1, 2, -1, 1, -1, 2, 0, 0, 0, \\
0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
\]

Run-length encode:

\[
-26, -3, 0, -3, -2, -6, 2, -4, 1, -4, \{2 x 1\}, 5, 1, 2, -1, 1, -1, 2, \{5 x 0\}, -1, -1, EOB
\]

Huffman code what remains. Encoding is complete.
JPEG Decoding

- Decoding is simply the reverse of encoding.
- Reverse the huffman, RLE encodings.
- Dequantize.
- Apply inverse DCT (IDCT):

\[
V(x, y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c_i c_j T(i, j) \cos \left(\frac{(2y + 1)i\pi}{2N}\right) \cos \left(\frac{(2x + 1)j\pi}{2N}\right)
\]

- Add 128 to convert back to unsigned.
Original & decompressed

original image

decompressed image
JPEG compression ratio

- compression ratio depends on quantization matrix
- effect depends on rendering size and image content
- 10:1 typical
- 100:1 with artifacts (blockiness)
Lenna

PNG 473,481 bytes (512x512)

JPEG, lowest quality, 19,100 bytes
H.261 VIDEO
H.261 Video

- H. 261 Compression was designed for videotelephony and videoconferencing applications.
  - Developed by CCITT (now ITU-T) in 1988-1990
  - Intended for use over ISDN telephone lines, as part of the H.320 protocol suite.
  - Datarate was specified as multiples of 64Kb/s (“p x 64”)

- Goals for ISDN videotelephony:
  - Low end-to-end delay.
  - Constant bit rate.
H.261 structure

Each CIF frame composed of 12 Groups of Blocks (GOBs)

Video composed of frames

Each GOB is composed of 11x3 MacroBlocks

Each MB is 16x16 pixels
CIF and QCIF Frame Formats

Each CIF frame (352x288 pixels) is composed of 12 Groups of Blocks (GOBs)

Each QCIF frame (176x144 pixels) is composed of 3 Groups of Blocks (GOBs)

GOB and MacroBlock format is identical in both frame formats.
GOB and Resynchronization

- Purpose of Group of Blocks is resynchronization.
- GOB starts with a sync code (binary: 00000000 00000001)
- Within a GOB, encoded MBs don’t even start on byte boundaries.
  - If there’s a bit error and you lose sync, or you join in the middle, you can’t decode the next bits (you don’t know where you are in the bitstream).
  - Scan for the next GOB sync code, and then you can start decoding.
Macroblocks

- Macroblock is basic unit for compression.
- Each macroblock is 16x16 pixels.
  - Represent as YUV 4:2:0 data.
  - 16x16 Luminance (Y) and subsampled 8x8 C_r, 8x8 C_b
- Represent this as 6 Blocks of 8x8 pixels:

```
Y
U
V
4 8x8 Y blocks
1 8x8 U block
1 8x8 V block
```
Macrobloc coding

Three ways to code a Macrobloc:

1. Don’t.
   - If it hasn’t changed since last frame, don’t send it.

2. Intra-frame compression
   - Do DCT, Quantize, Zig-zag, Run-length encoding, and Huffman coding. Just like JPEG.

3. Inter-frame compression
   - Calculate difference from previous version of same block.
   - Can use motion estimation to indicate block being differenced can from a slightly different place in previous frame.
   - Same DCT/quant/huffman coding as Intra, but data is differences rather than absolute values.
H.261 intra-frame compression

Intra-coding of blocks is very similar to JPEG:

- DCT.
- Quantize DCT.
  - Unlike JPEG, H.261 uses the same quantizer value for all coefficients.
  - Feedback loop changes quantizer to achieve target bitrate.
- Order coefficients in zig-zag order.
- Run-length encode.
- Huffman code what remains.
H.261 inter-frame compression

- Basic compression process is the same as intra-frame compression, but the data is the differences from the immediately preceding frame rather than the raw samples themselves.
Frame Differencing

Often the amount of information in the difference between two frames is a lot less than in the second frame itself.
Motion

- Motion in the scene will increase the differences.
- If you can figure out the motion (where each block came from in the previous frame):
  - Encode the motion as a motion vector (two small integers indicating motion in x and y directions)
  - Encode the differences from the *moved* block using DCT + quantization + RLE + Huffman encoding.
Motion

Coding from moved part of previous image can reduce the differences

Frame 1

Frame 2

Frame 2 - 1
(lots of motion)
Motion Compensation in H.261

- Each inter-coded 16x16 pixel macroblock has its own motion vector.
  - Applies to all six 8x8 blocks in the macroblock.

- Encoder must search the image surrounding the MB to discover where it came from.
  - Don’t care whether it’s really motion or not - only that differencing reduces the data to send.
  - Motion Vector search can be the most CPU-intensive part of H.261.
  - Standard doesn’t say how to do this - only how to decode the results. Plenty of room for innovation.
Motion Vector Search

Where did this Macroblock come from in the previous frame?
Motion Vector Search

Where did this Macroblock come from in the previous frame?

Each motion vector can encode motions of \( \pm 15 \) pixels in both x and y direction.

\[ 30^2 = 900 \] possible vectors for each Macroblock.

Calculate mean difference for each possible vector. Choose vector with least mean difference.

\[ \Rightarrow 256 \text{ subtractions and } 256 \text{ additions per possible vector} \]

\[ \Rightarrow 460K \text{ calculations per MB, } \]

\[ \Rightarrow 182M \text{ calculations per frame (CIF), } \]

\[ \Rightarrow 5.5 \text{ billion calculations per second (30fps NTSC video). } \]

Not possible on today's CPUs.
Motion Vector Search: Brute Force

- Each motion vector can encode motions of \( \pm 15 \) pixels in both x and y direction.
- \( 30^2 = 900 \) possible vectors for each Macroblock.

- Calculate mean difference for each possible vector. Choose vector with least mean difference.
  - \( \Rightarrow 256 \) subtractions and 256 additions per possible vector
  - \( \Rightarrow 460K \) calculations per MB,
  - \( \Rightarrow 182M \) calculations per frame (CIF),
  - \( \Rightarrow 5.5 \) billion calculations per second (30fps NTSC video).
- Not possible on today’s CPUs.
Hierarchical Search

Motion Vector

Motion vectors: ±1
around level 2 result
⇒ 9 positions checked
⇒ 16\(^2\) \times 2 \text{ ops per check}
⇒ 4650 \text{ ops}

Motion vectors: ±1
around level 2 result
⇒ 9 positions checked
⇒ 8\(^2\) \times 2 \text{ ops per check}
⇒ 1152 \text{ ops}

Motion vectors: ±4
⇒ 64 positions checked
⇒ 4\(^2\) \times 2 \text{ ops per check}
⇒ 2048 \text{ ops}

Total: 90M \text{ ops/sec for 30fps}
Intra-Block Encoding

Source Block → DCT → Quantize → Run Length + Huffman → Encoded Block

Decoded Frame

Frame Store -> De-Quantize -> IDCT

Diagram showing the process of Intra-Block Encoding: DCT, Quantize, Run Length + Huffman, Encoded Block, Decoded Frame, Frame Store, De-Quantize, IDCT.
Inter-Block Encoding

Source Block

DCT

Quantize

Run Length + Huffman

Encoded Block

Estimated Block

Motion Compensation

Decoded frame

Previous frame

Motion Estimation

Motion Vector

Frame Store

IDCT

De-Quantize

Quantized DCT Coefficients

Motion Vector
Bitstream Structure

- Frame
- Start Code
- Time Reference
- CIF/QCIF
- GOB
- GOB Start
- GOB Number
- Quantizer
- Macro Block
- MB Addr
- Intra/Inter
- Quant
- Motion Vector
- Which Blocks?
- Block0
- Block1
- Block5
- DC Coefficient
- Skip+ Coefficient
- ... Skip+ Coefficient
- EOB
H.261 Design Goals

Intended for videotelephony.

- Low delay.
  - Each frame coded as it arrives.
  - Only need a small bitstream buffer on output to smooth to CBR (adds a little delay)
- Constant Bit Rate (CBR)
  - Only send a small number of intra-coded blocks in each frame, so data rate variation is only a function of video content.
  - Adjust the quantization based on occupancy of the bitstream buffer.
H.261 Non-design Goals

- Not intended for recording and playback.
- No way to seek backwards or forwards because you don’t normally encode any frames with entirely intra-coded blocks.
  - Could do this, but wouldn’t give CBR flow needed for ISDN usage.
- Limited robustness to bit errors.
  - Errors cause corruption (incorrect huffman decoding of rest of GOB). Possibly detected by hitting a illegal state in decoder.
  - Stop decoding, search for next GOB. Start decoding again.
  - Intra blocks recover damage slowly over next few seconds.
H.263

- Son of H.261.
  - Standardized in 1996.
  - Replacing H.261 in many applications.
- Basic design is very similar to H.261 (DCT/Quantization based, using intra or inter frame coding).
  - Numerous optional improvements to improve compression, robustness, and flexibility of use.
H.263 Improvements

- New options:
  - Unrestricted Motion Vectors,
  - Syntax-based arithmetic coding (replace RLE/Huffman)
  - Advance prediction (uses 4 8*8 blocks instead of 1 16*16: gives better detail.)
  - Forward and backward frame prediction similar to MPEG
- Five resolutions (H.261 only does QCIF and CIF):

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQCIF</td>
<td>128x96</td>
</tr>
<tr>
<td>QCIF</td>
<td>176x144</td>
</tr>
<tr>
<td>CIF</td>
<td>352x288</td>
</tr>
<tr>
<td>4CIF</td>
<td>704x576</td>
</tr>
<tr>
<td>16CIF</td>
<td>1408x1152</td>
</tr>
</tbody>
</table>
MPEG
MPEG Family

- MPEG-1
  - Similar to H.263 CIF in quality
- MPEG-2
  - Higher quality: DVD, Digital TV, HDTV
- MPEG-4/H.264
  - More modern codec.
  - Aimed at lower bitrates.
  - Works well for HDTV too.
MPEG-1 Compression

- MPEG: Motion Pictures Expert Group
- Finalized in 1991
- Optimized for video resolutions:
  - 352x240 pixels at 30 fps (NTSC)
  - 352x288 pixels at 25 fps (PAL/SECAM)
- Optimized for bit rates around 1-1.5Mb/s.
- Syntax allows up to 4095x4095 at 60fps, but not commonly used.
- Progressive scan only (not interlaced)
**MPEG Frame Types**

- Unlike H.261, each frame must be of one type.
  - H.261 can mix intra and inter-coded MBs in one frame.

- Three types in MPEG:
  - I-frames (like H.261 intra-coded frames)
  - P-frames (“predictive”, like H.261 inter-coded frames)
  - B-frames (“bidirectional predictive”)
MPEG I-frames

- Similar to JPEG, except:
  - Luminance and chrominance share quantization tables.
  - Quantization is adaptive (table can change) for each macroblock.

- Unlike H.261, every $n$ frames, a full *intra-coded frame* is included.
  - Permits skipping. Start decoding at first I-frame following the point you skip to.
  - Permits fast scan. Just play I-frames.
  - Permits playing backwards (decode previous I-frame, decode frames that depend on it, play decoded frames in reverse order)

- An I frame and the successive frames to the next I frame ($n$ frames) is known as a Group of Pictures.
MPEG P-Frames

- Similar to an entire frame of H.261 inter-coded blocks.
  - Half-pixel accuracy in motion vectors (pixels are averaged if needed).
- May code from previous I frame or previous P frame.

I P P P I P P P I
**Object occlusion**

- Often an object moves in front of a background.
- P frames code the object fine, but can’t effectively code the revealed background.

1. Do motion vector search separately in past reference frame and future reference frame.
2. Compare:
   - Difference from past frame.
   - Difference from future frame.
   - Difference from average of past and future frame.
3. Encode the version with the least difference.
B-frames

- Bidirectional Predictive Frames.
- Each macroblock contains two sets of motion vectors.
- Coded from one previous frame, one future frame, or a combination of both.

1. Do motion vector search separately in past reference frame and future reference frame.
2. Compare:
   - Difference from past frame.
   - Difference from future frame.
   - Difference from average of past and future frame.
3. Encode the version with the least difference.
B-frames: Macroblock averaging

Past Frame | Current Frame | Future Frame

\[
\begin{align*}
\text{Past Frame} & \quad \text{Current Frame} \quad \text{Future Frame} \\
\text{Difference} & + \quad \text{Motion Vectors}
\end{align*}
\]

\[
\left( \frac{1}{2} \right) \left( \text{Past Frame} + \text{Current Frame} \right) - \text{Future Frame} = \text{Difference}
\]
Frame Ordering

- Up to encoder to choose I, P, B frame ordering.
- Eg IBBPBBIBBPBBPI…
Frames are encoded out of order:

1. Encode I-frame 1
2. Store frame 2
3. Store frame 3
4. Encode P frame 5
5. Encode B frame 2
6. Encode B frame 3
7. Store frame 5
8. Store frame 6
9. Encode I frame 7
10. Encode B frame 5
11. Encode B frame 6
Transmission Order

- Frames are encoded out of order
- Need to be decoded in the order they’re encoded.
  - Common to send out of order.

Eg: $I_1B_2B_3P_4B_5B_6I_7B_8B_9P_{10}B_{11}B_{12}I_{14}$
  sent in the order
  $I_1P_4B_2B_3I_7B_5B_6P_{10}B_8B_9I_{14}B_{11}B_{12}$

- Allows decoder to decode as data arrives, although it still has to hold decoded frames until it has decoded prior B frames before playing them out.
B-frame disadvantages

- Computational complexity.
  - More motion search, need to decide whether or not to average.
- Increase in memory bandwidth.
  - Extra picture buffer needed.
  - Need to store frames and encode or playback out of order.
- Delay
  - Adds several frames delay at encoder waiting for need later frame.
  - Adds several frames delay at decoder holding decoded I/P frame, while decoding and playing prior B-frames that depend on it.

Typical MPEG-1 values.

<table>
<thead>
<tr>
<th>Compression Size</th>
<th>Type</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>27:1</td>
<td></td>
<td>4.8KB</td>
</tr>
<tr>
<td>50:1</td>
<td></td>
<td>2.5KB</td>
</tr>
<tr>
<td>20:1</td>
<td></td>
<td>6KB</td>
</tr>
<tr>
<td>7:1</td>
<td></td>
<td>18KB</td>
</tr>
</tbody>
</table>
B-frame advantage

- B-frames increase compression.
- Typically use twice as many B frames as I+P frames.

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>18KB</td>
<td>7:1</td>
</tr>
<tr>
<td>P</td>
<td>6KB</td>
<td>20:1</td>
</tr>
<tr>
<td>B</td>
<td>2.5KB</td>
<td>50:1</td>
</tr>
<tr>
<td>Average</td>
<td>4.8KB</td>
<td>27:1</td>
</tr>
</tbody>
</table>

Typical MPEG-1 values.
Really depends on video content.
MPEG-2

- ISO/IEC standard in 1995
- Aimed at higher quality video.
- Supports interlaced formats.
- Many features, but has profiles which constrain common subsets of those features:
  - Main profile (MP): 2-15Mb/s over broadcast channels (eg DVB-T) or storage media (eg DVD)
  - PAL quality: 4-6Mb/s, NTSC quality: 3-5Mb/s.
## MPEG-2 Levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Max Resolution</th>
<th>Max FPS</th>
<th>Max Coded Data Rate (Mb/s)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>352x288</td>
<td>30</td>
<td>4</td>
<td>Consumer tape equiv.</td>
</tr>
<tr>
<td>Main</td>
<td>720x576</td>
<td>30</td>
<td>15</td>
<td>Studio TV</td>
</tr>
<tr>
<td>Main 1440</td>
<td>1440x1152</td>
<td>60</td>
<td>60</td>
<td>Consumer HDTV</td>
</tr>
<tr>
<td>High</td>
<td>1920x1152</td>
<td>60</td>
<td>80</td>
<td>Film production</td>
</tr>
</tbody>
</table>
MPEG-2 vs MPEG-1

Sequence layer

- progressive vs interlaced
- More aspect ratios (e.g., 16x9)
- Syntax can now signal frames sizes up to 16383x16383
- Pictures must be a multiple of 16 pixels
MPEG-2 vs MPEG-1

Picture Layer:
- All MPEG-2 motion vections are always half-pixel accuracy
  - MPEG-1 can opt out, and do one-pixel accuracy.
- DC coefficient can be coded as 8, 9, 10, or 11 bits.
  - MPEG-1 always uses 8 bits.
- Optional non-linear macroblock quantization, giving a more dynamic step size range:
  - 0.5 to 56 vs 1 to 32 in MPEG-1.
  - Good for high-rate high-quality video.
Interlacing

- MPEG-2 codes a frame. May include both interlaced fields.
- Fields may differ, so compression suffers.
  - More high frequencies in vertical dimension.
- MPEG-2 can use a modified zig-zag for run-length encoding of the coefficients:
Typical MPEG-2 Frame Sizes

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (KB)</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-frame</td>
<td>50</td>
<td>10:1</td>
</tr>
<tr>
<td>P-frame</td>
<td>25</td>
<td>20:1</td>
</tr>
<tr>
<td>B-frame</td>
<td>10</td>
<td>50:1</td>
</tr>
<tr>
<td>Ave:</td>
<td>18</td>
<td>29:1</td>
</tr>
</tbody>
</table>

Average sizes for ~4Mb/s video, Main Profile at Main Level (MP@ML)

Actual frame sizes will vary a lot depending on content.
MPEG-4

- ISO/IEC designation 'ISO/IEC 14496': 1999
- Aimed at low bitrate (10Kb/s)
- Can scale very high (1Gb/s)
- Based around the concept of the composition of basic video objects into a scene.
Media Objects

- Still images (e.g. as a fixed background);
- Video objects (e.g. a talking person - without the background);
- Audio objects (e.g. the voice associated with that person, background music);
- Text and graphics;
- Talking synthetic heads and associated text used to synthesize the speech and animate the head; animated bodies to go with the faces;
- Synthetic sound.

- Also 3-D objects.
Composition of Media Objects

MPEG-4 provides a standardized way to describe a scene

- Place media objects in a coordinate system;
- Apply transforms to change the geometrical or acoustical appearance of a media object;
- Group primitive media objects to form compound media objects;
- Apply streamed data to media objects
  
  Eg: animation parameters driving a synthetic face

- Can change, interactively, the user’s viewing and listening points anywhere in the scene.

- Builds on concepts from the Virtual Reality Modelling Language (VRML)
MPEG-4 Sprites

If you can segment foreground motion from the background, MPEG-4 allows you to send it separately as a sprite.
H.264 (MPEG-4, Part 10)

- MPEG-4, Part 10 is also known as H.264.
H.264 vs MPEG-2

- Multi-picture motion compensation.
  - Can use up to 32 different frames to predict a single frame.
  - B-frames in MPEG-2 only code from two.
- Variable block-size motion compensation
  - From 4x4 to 16x16 pixels.
  - Allows precise segmentation of edges of moving regions.
- Quarter-pixel precision for motion compensation.
- Weighted prediction (can scale or offset predicted block)
  - Useful in fade-to-black or cross-fade between scenes.
- Spatial prediction from the edges of neighboring blocks for "intra" coding.
- Choice of several more advanced context-aware variable length coding schemes (instead of Huffman).
H.264 performance

- Typically half the data rate of MPEG-2.
- HDTV:
  - MPEG-2: 1920x1080 typically 12-20 Mbps
  - H.264: 1920x1080 content at 7-8 Mbps
H.264 Usage

- Pretty new, but expanding use.
- Included in MacOS 10 (Tiger) for iChat video conferencing.
- Used by Video iPod.
- Adopted by 3GPP for Mobile Video.
- Mandatory in both the HD-DVD and Blu-ray specifications for High Definition DVD.