Lecture 2:
Multimedia Compression Techniques

Prof. Shih-Fu Chang

EE 6850, Fall 2002

Sept. 11, 2002

Course URL: http://www.ee.columbia.edu/~slchang/course/vis/
References


Why compression?

- Large data size
  - Digital video 720x480x24x30 →
    249 Mbps, 112 GB per hour
  - v.s.
    ADSL, wireless/mobile

- Network games →
  thousands of objects in a scene,
  thousands of elements in an object
Standards

- MPEG-1
  - 1.5 Mbps, CD-ROM interactive applications
- MPEG-2
  - 2-10 Mbps, Digital video, e.g., DTV, DVD
- MPEG-4
  - Wide range of bitrates
  - Low-bit-rate video, e.g., wireless
  - Natural-synthetic hybrid video
  - High-quality audio
- H-263(+)
  - Video conferencing
  - Internet video
Important Factors

- Bit rate
- Format and Quality
- Complexity
- Error resilience
- Random access
- Interoperability
Digitization of Video Signal

Color Component → Filter → Sample → Quantization → PCM Coding

Color Representation: R, G, B (Trichromatic Theory)

Human Color Perceptual Attributes:
- Hue — dominant light wavelength
- Saturation — purity
- Intensity —
Luminance/Chrominance
— Monochrome Display vs. Color

PAL system: \( (625 \text{ lines}) \)
\[
Y = 0.299 \, R + 0.587 \, G + 0.114 \, B
\]
\[
U = 0.492 \, (B - Y)
\]
\[
V = 0.877 \, (R - Y)
\]

NTSC system: \( (525 \text{ lines}) \)
\[
I = - (\sin 33^\circ) \, U + (\cos 33^\circ) \, V
\]
\[
Q = (\cos 33^\circ) \, U + (\sin 33^\circ) \, V
\]

Digital Video: \( (\text{CCIR 601}) \, (720 \times 480 \text{ pixels}) \)
\[
Y = 0.297 \, R + 0.587 \, G + 0.114 \, B + 16
\]
\[
Cr = 0.439 \, R - 0.368 \, G - 0.071 \, B + 128
\]
\[
Cb = -0.148 \, R - 0.291 \, G + 0.439 \, B + 128
\]
(scaled and shifted version of YUV)
A/D conversion of TV signals

\[ Y \rightarrow \text{Filter} \rightarrow \text{Sample} \rightarrow \text{Recon} \rightarrow \text{PCM Coding} \]

**Sampling**: 4:2:0

\[ x \ x \ x \ x \ x \ x \ x \ x \ x \]

**Sampling**: 4:2:2

**Examples of frames**: SIF, CCIR601 4:2:0
Progressive

Interlace

Temporal Resolution:

<table>
<thead>
<tr>
<th>Format</th>
<th>Frames/Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Film</td>
<td>24</td>
</tr>
<tr>
<td>TV</td>
<td>60 fields/sec</td>
</tr>
<tr>
<td></td>
<td>50 fields/sec</td>
</tr>
</tbody>
</table>
Temporal Conversion

- De-Interlacing

Interlaced Frame 1  Non-Interlaced Frame 1

\[
\begin{align*}
I' & \quad J' \\
K' & \quad L' \\
M' & \quad N'
\end{align*}
\]

For Non-Interlaced Frame 1

\[
K' = k
\]

\[
L' = \frac{1}{2} (k+M) + \frac{1}{4} \left( L - \frac{1}{2} J - \frac{1}{2} N \right)
\]

For Non-Interlaced Frame 2

\[
L' = L
\]

\[
K' = \frac{1}{2} (J + L) + \frac{1}{4} \left( k - \frac{1}{2} I - \frac{1}{2} M \right)
\]

- Reverse

\[
\begin{align*}
J' & \quad L' \\
K' & \quad M' \\
I' & \quad N'
\end{align*}
\]

\[
k' = \frac{1}{2} J + \frac{1}{2} K + \frac{1}{2} L \\
L' = \frac{1}{2} k + \frac{1}{2} L + \frac{1}{2} M
\]
Film to Video Conversion
* 3:2 pull down

Diagram showing the process of film to video conversion with labeled fields and frames.
Video Compression Basics

- Predictive Coding
- Transform Coding
- Vector Quantization
- Entropy Coding
- Perceptual Factors
Motion Compensated Prediction:

Block-based

Ref. Frame
Current Frame

Logarithmic Search

2 factors:
- block size
- search range

Complexity:

Exhaustive search
$(6x2+1)^2 = 169$

Log. search
21 pts.

O" Denotes a Search Point of Step n
MC can improve compression by a factor of 2-3 depending on scene, type of motion, and object in motion.
Group of Pictures with an I-Picture every N Pictures and a P-Picture every M Pictures
Transform Coding

\[ \begin{pmatrix} \text{pixels} \end{pmatrix}_N \rightarrow \begin{pmatrix} \text{Transform Coeff.} \end{pmatrix}_N \]

- remove spatial redundancy
- some coeff. can be more coarsely quantized
- Non-uniform probability distribution of coeff.

1-D case

\[ \tilde{e} = T \tilde{b} \]
\[ \tilde{b} \in \mathbb{R}^n \] column vector

\[ T = \begin{bmatrix} \tilde{e}_1 & \tilde{e}_2 & \cdots & \tilde{e}_n \end{bmatrix} \]

\[ \tilde{e}_i \text{ : orthogonal basis vector} \]

\[ \tilde{b} = T^t \tilde{e} = a_1 \tilde{e}_1 + a_2 \tilde{e}_2 + \cdots + a_n \tilde{e}_n \]
DCT: Discrete Cosine Transform

- Energy compaction
- Fast algorithm

\[ a_m = \sqrt{\frac{2}{N}} \quad a_0 = \frac{1}{\sqrt{N}} \]

\[ f_m(i) = a_m \cdot \cos \left( \frac{\pi}{N} (i - \frac{1}{2}) (m - 1) \right) \quad m = 1 \ldots N \]

\[ i = 1 \ldots N \]

Fig. 1-3: Demonstration of DCT with \( N = 16 \).
Entropy Coding

- symbols w/ higher probability → longer codewords
Putting Together: Hybrid Coders

![Diagram of a hybrid coder]

- **Image**: Input image
- **DCT**: Discrete Cosine Transform
- **Q**: Quantization
- **VLC**: Variable Length Coding
- **BUF**: Buffer
- **CODES**: Output codes
- **IDCT**: Inverse Discrete Cosine Transform
- **Motion Compensated Predictor**: Predicted motion
- **Motion Estimator**: Estimate motion
- **MOTION VECTORS**: Motion vectors
Quality Measurement

- subjective
- objective

**Error** = \( U(m,n) - U'(m,n) \)

\[ \text{mse} = \sigma_e^2 = \frac{\sum_{m,n} |e(m,n)|^2}{M \times N} \]

**Signal energy**

\[ \sigma^2 = \frac{\sum_{m,n} (U(m,n) - \text{mean})^2}{N \times N} \]

**SNR**: Signal to noise ratio

\[ = 10 \log_{10} \left( \frac{\sigma^2}{\sigma_e^2} \right) \text{ dB} \]

- Eq. 6/60 = 100 \( \Rightarrow \) 40 dB
- Eq. 6/60 = 10 \( \Rightarrow \) 20 dB
H.263 ITUT Low Bit Rate Video Coding Standard

- Started in 1994
- Goal: refine H.261 for videophone applications over PSTN (10-24 Kbps)
- Use block motion-compensated DCT transform coding
- Can use subQCIF, QCIF, CIF, 4CIF, 16CIF
- Include 4 new negotiable options
  1. Unrestricted Motion Vector (UMV)
  2. Syntax-based Arithmetic Coding (SAC)
  3. Advanced Prediction Mode (AP)
  4. PB Frames (PB)

- H.263 becomes basis of MPEG-4 video coding
- H.263v2 (263+) has more features, e.g. error resilience, dynamic range, format

CIF (SIF)
Source Input Format
852 x 240 x 30
or
852 x 288 x 25

Corresponding to NTSC or PAL
UMV Mode:

Motion vectors (mv) are allowed to point outside of the picture. If the reference pixel is outside of the picture use the boundary pixel.

default mode— mv in [-16, 15.5] (half-pixel resolution)
UMV mode— mv in [-31.5, 31.5]

AP Mode: \(\text{most useful}\)

four mv's for a macroblock or overlapped block motion compensation

\[
p(i, j) = (q(i, j) \times H_0(i, j) + r(i, j) \times H_1(i, j) + s(i, j) \times H_2(i, j)) / 8
\]

\[
H_0 = \begin{bmatrix}
4 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
4 & 5 & 5 & 5 & 5 & 5 & 5 & 5
\end{bmatrix}
\]

\[
H_1 = \begin{bmatrix}
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2
\end{bmatrix}
\]

\[
H_2 = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]
**PB Frame:**

\[ MV_F = \frac{TR_B \cdot MV}{TR_D} + MV_D \]  
\[ MV_B = MV_F - MV = \frac{TR_B \cdot MV}{TR_D} - MV \]  

\( MV_D \): refinement  
\( if \ MV_D = 0 \)

The block in the B frame (within the PB frame) is approximated by the average of the forward reference and the backward reference.

No prediction errors are coded.

*Good for low-bit-rate simple sequences*

*Columbia University*
**Arithmetic Coding**

Huffman coding:
- one symbol → a codeword of an integral no. of bits

Arithmetic coding:
- variable no. of symbols → variable no. of bits

Example:
- Case 1: \( \text{Prob} = \frac{1}{2} \)
- Case 2: \( \text{Prob} = \frac{1}{4} \)

Entropy: \( \log \left( \frac{1}{\text{Prob}} \right) \)

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Prob</th>
<th>( \log )</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>0.5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>0.25</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>0.125</td>
<td>3.32193</td>
<td>3</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>0.125</td>
<td>3.32193</td>
<td>3</td>
</tr>
</tbody>
</table>

Avg. Length = 1.75

Theoretical limit: \( \sum \text{Prob} \cdot \log \left( \frac{1}{\text{Prob}} \right) \)

\[-(0.5 \cdot \log(2) + 0.25 \cdot \log(4) + 0.125 \cdot \log(8)) = 1.75\]
Example

<table>
<thead>
<tr>
<th>Alphabet</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>0.35</td>
</tr>
<tr>
<td>C</td>
<td>0.15</td>
</tr>
<tr>
<td>D</td>
<td>0.15</td>
</tr>
</tbody>
</table>

- Map probability to the interval [0, 1]
- Represent boundary points of an interval in binary form.

\[ 0.25 \text{ or } 1/4 \]

\[ 0.20(2) = 0.01(2) \]

\[ C = \frac{1}{\text{len}} \sum_{k=1}^{\text{len}} a_k \]

- symbols \[ \overline{ABC} \rightarrow 1101 \]

Each sequence of symbols has an associated probability which can be mapped to a subinterval of [0, 1).

- Decode
  - Received Bits \( \rightarrow \) a segment in [0, 1]
  - Need reserved symbol for end of message
Encoding (Output common bits but a boundaries)

\[ a_2 a_1 a_0 \Rightarrow 1001 \] cp to VLC (100 110)

Decoding (identify unique segment)

- high prob \Rightarrow long segment \Rightarrow less bits
- Approach The more optimal
Ziv-Lempel Coding (1977)

- Variable symbol block length
  → fixed codeword length

Example:

<table>
<thead>
<tr>
<th>Input</th>
<th>Dictionary</th>
<th>Index</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>01</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>010</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0110</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Look at the block length equal to the largest C0
2. If not exist try a shorter one
3. After counting add a raw C0 = [symbol, block] + 1 symbol

- Used in UNIX "compress" & PC "arc"
- Does not need knowledge of probability
- If unlimited dictionary size is optimal
- May be changed to adapt to varying input characteristics