CS 260: Seminar in Computer Science: Multimedia Networking

Jiasi Chen
Lectures: MWF 4:10-5pm in CHASS

http://www.cs.ucr.edu/~jiasi/teaching/cs260_spring17/
Multimedia is...

User perception

Applications

On-demand video

Live video

Virtual/augmented reality

Internet

Content creation

Compression

Storage

Distribution
Encoding Images

1. Pre-processing
2. Discrete cosine transform
3. Quantization
4. Entropy encoding
Encoding Images: Pre-processing

• Convert from color to luma and chroma components

• Divide image into blocks (e.g. 8x8 pixels)
Encoding Images: Discrete Cosine Transform

• Transform from spatial domain to frequency domain


\[ G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ -12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{bmatrix}

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

Example: https://upload.wikimedia.org/wikipedia/commons/5/5e/Idct-animation.gif
Encoding Images: Quantization

- Lossy compression by division and rounding

\[
G = \begin{bmatrix}
-415.38 & -30.19 & -61.20 \\
4.47 & -21.86 & -60.76 \\
-46.83 & 7.37 & 77.13 \\
-48.53 & 12.07 & 34.10 \\
12.12 & -6.55 & -13.20 \\
-7.73 & 2.91 & 2.38 \\
-1.03 & 0.18 & 0.42 \\
-0.17 & 0.14 & -1.07 \\
\end{bmatrix}
\]

\[
\begin{array}{cccccccc}
v \\
\hline
u \\
\end{array}
\]

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

By dividing by

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

and then rounding.
Encoding Images: Entropy Encoding

- Lossless compression to get close to optimal code rate of $-\log_{\# \text{ symbols}}(\text{probability of the symbol})$

This is an example of a Huffman tree:

Using the codebook:

- 26 characters in the alphabet $\rightarrow$ 5 bits/character
- 5 bits/character $\times$ 36 characters in the sentence = 180 bits
Encoding Images: Quality Examples

<table>
<thead>
<tr>
<th>Quality</th>
<th>100</th>
<th>25</th>
<th>10</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>83 bytes</td>
<td>10 bytes</td>
<td>5 bytes</td>
<td>1.5 bytes</td>
</tr>
</tbody>
</table>
Aside: Lena
Video Encoding

1. Motion estimation
2. I-frame encoding
Video Encoding: I-frame encoding

• Naïve solution: encode every frame as a JPEG

• Leverage temporal redundancy by encoding the difference between frames
  • I-frame: inter frame
  • P-frame: predictive inter frame
  • B-frame: bi-predictive inter frame

• GOP = “group of pictures” frame pattern
  • E.g., IPPBPPBPP
Video Encoding: Motion Estimation

• How to look for similarity in time?
• Computationally complex

Input: macroblock (16x16 pixels)

Is this block very similar to the previous block in time?

Search threshold

How close in time should we search?
How far in space should we look?

No

Output: motion vector

Yes

Output: same as input macroblock
Video Encoding: Block Matching

Source: T. Wiegand / B. Girod: EE398A Image and Video Compression
Video Encoding: Block Matching

- Mean squared error
  \[
  SSD(d_x, d_y) = \sum_{y=1}^{By} \sum_{x=1}^{Bx} [s(x, y, t) - s'(x - d_x, y - d_y, t - \Delta t)]^2
  \]

- Sum of absolute differences
  \[
  SAD(d_x, d_y) = \sum_{y=1}^{By} \sum_{x=1}^{Bx} |s(x, y, t) - s'(x - d_x, y - d_y, t - \Delta t)|
  \]

Source: T. Wiegand / B. Girod: EE398A Image and Video Compression
Video Encoding: Search Strategies

General algorithm:
1. Start with an initial step size $S$
2. Search $N$ locations within $S$ distance
3. If the center is best
   a) $S = S/2$
   b) Go to 2
4. If an edge location is best
   a) Re-center the origin
   b) Go to 2

Source: T. Wiegand / B. Girod: EE398A Image and Video Compression
Content Type and Compression

Example: https://www.youtube.com/watch?v=YyRgdWNq-aQ
Video Metrics

• Resolution = (# pixels) x (# pixels)
  • 720p = 1280 x 720
  • 1080p = 1920 x 1080
  • 4K = 3840 x 2160

• Frames per second
  • 30 fps
  • 60 fps

• Bitrate
  • Wireless: ~1 Mbps
  • Desktop: ~3-5 Mbps
  • High-resolution: 10+ Mbps

• Codec = encoding type
  • H.264
  • VP8

• Container = holds video + audio
  • webm
  • MPEG4

• Decoder
• Encoder
Image Quality: Quantitative Metrics

- How to measure video quality quantitatively?
- **PSNR**

\[
MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2
\]

\[
PSNR = 10 \cdot \log_{10} \left( \frac{MAX_I^2}{MSE} \right) = 20 \cdot \log_{10} \left( \frac{MAX_I}{\sqrt{MSE}} \right) = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE)
\]

I: original image
K: compressed image
i,j: directions
MAX = max value of pixel
PSNR Example

Original uncompressed image  
PSNR = 45.53 dB

PSNR = 36.81 dB

PSNR = 31.45 dB
Image Quality: Quantitative Metrics

All of these images have the same MSE

→ Not all errors are created equal

Video Quality: SSIM

• Key idea: humans are responsive to changes in *structure*
  • E.g., increase contrast or average brightness doesn’t matter too much
  • More closely approximate human visual system
  • Operate on luma component only (not color or chrominance)

• Three components
  • Luminance: based on mean
    \[ \mu_x = \frac{1}{N} \sum_{i=1}^{N} x_i. \]
  • Contrast: based on variance, with mean subtracted
    \[ \sigma_x = \left( \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)^2 \right)^{\frac{1}{2}} \]
  • Structure: based on correlation, with mean subtracted and variance normalized
Video Quality: SSIM

- Luminance
  \[ l(x, y) = \frac{2\mu_x \mu_y + c_1}{\mu^2_x + \mu^2_y + c_1} \]
- Contrast
  \[ c(x, y) = \frac{2\sigma_x \sigma_y + c_2}{\sigma^2_x + \sigma^2_y + c_2} \]
- Structure
  \[ s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x \sigma_y + c_3} \]

\[ SSIM(x, y) = [l(x, y)^\alpha \cdot c(x, y)^\beta \cdot s(x, y)^\gamma] \]

\[ \alpha, \beta, \gamma = 1, c_3 = c_2/2 \]

\[ SSIM(x, y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{\left(\mu^2_x + \mu^2_y + c_1\right)\left(\sigma^2_x + \sigma^2_y + c_2\right)} \]
Image Quality: Quantitative Metrics

All of these images have the same MSE = 210

→ Not all errors are created equal

SSIM = 0.9168
SSIM = 0.9900
SSIM = 0.9900
SSIM = 0.6949
SSIM = 0.7052
SSIM = 0.7748

Image Quality: Qualitative Metrics

• Mean Opinion Score
  • 5: Excellent
  • 4: Good
  • 3: Fair
  • 2: Poor
  • 1: Bad

• ITU recommendations for how to set up the experiment
  • Distance from viewers, number of views visible, etc.

• User studies can be time-consuming and expensive
Image Quality Metric Comparison

![Graphs showing MOS vs PSNR and MOS vs MSSIM for JPEG and JPEG2000 images. The graphs include fitting with Logistic Function.](image-url)
Video Quality

• User quality of experience (QoE)
  • Average PSNR or SSIM across all frames
  • MOS
  • Watch time = how long the user watches the video

• Video metrics
  • Stalls = # of times the buffer is empty
  • Buffering ratio = # the fraction of time the buffer is empty
  • Bitrate switches = # times the video changes quality
  • Startup time = time from when the user requests the video to when it starts playing
Metrics

Network metrics
- CDN choice
- Throughput
- Latency
- Packet loss

Video metrics
- Stalls
- Buffering ratio
- Bitrate switches
- Startup time

User QoE
- MOS
- PSNR/SSIM

Applications
- On-demand video
- Live video
- Virtual/augmented reality

Content creation → Compression → Storage → Distribution → Internet → Applications

Video metrics
• Stalls
• Buffering ratio
• Bitrate switches
• Startup time

Network metrics
• CDN choice
• Throughput
• Latency
• Packet loss
Developing a Predictive Model of Quality of Experience for Internet Video

A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, H. Zhang

ACM Sigcomm 2013
# Relationship between Metrics

<table>
<thead>
<tr>
<th></th>
<th>Engagement-centric</th>
<th>Actionable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR-like (e.g., [17])</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Opinion Scores (e.g., [6])</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Network-level (e.g., bandwidth, latency [35])</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Single metric (e.g., bitrate, buffering)</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Naive learning</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Our approach</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Network metrics**
- CDN choice
- Throughput
- Latency
- Packet loss

**Video metrics**
- Stalls
- Buffering ratio
- Bitrate switches
- Startup time

**User QoE**
- MOS
- PSNR/SSIM
Method

• Data from Conviva, a video delivery platform
  • 40 million sessions over 3 months in the US
  • VoD and live sports
  • Metrics collected by client

• Decision trees
  • Input: Video metrics
  • Output: Engagement metric
  • Bin these metrics
Confounding Factors?

- Type of video
  - Live
  - Video-on-demand
- User attributes
  - Location
  - Device (smartphones, tablets, laptop)
  - Connectivity (wireless, Ethernet)
- Temporal attributes
  - Time of day/week
  - Freshness
Detecting Confounding Factors

• **Information gain metric**
  - Entropy \[ H(Y) = -\sum_i P(Y=y_i) \log(P(Y=y_i)) \]
  - Conditional entropy \[ H(Y|X) = \sum_i P(X=x_i) H(Y|X=x_i) \]
  - Information gain \[ H(Y) - H(Y|X) \]

• **Determine which confounding factors have max information gain**

• **Create a new decision tree for each confounding factor**

<table>
<thead>
<tr>
<th>Confounding Factor</th>
<th>Engagement</th>
<th>Join Time</th>
<th>Buff. Ratio</th>
<th>Rate of buff.</th>
<th>Avg. bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of video (live or VOD)</td>
<td>8.8</td>
<td>15.2</td>
<td>0.7</td>
<td>0.3</td>
<td>6.9</td>
</tr>
<tr>
<td>Overall popularity (live)</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>Overall popularity (VOD)</td>
<td>0.1</td>
<td>0.2</td>
<td>0.4</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>Time since release (VOD)</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
<td>0.2</td>
</tr>
<tr>
<td>Time of day (VOD)</td>
<td>0.2</td>
<td>0.6</td>
<td>2.2</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>Day of week (VOD)</td>
<td>0.1</td>
<td>0.2</td>
<td>1.1</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>Device (live)</td>
<td>1.3</td>
<td>1.3</td>
<td>1.1</td>
<td>1.2</td>
<td>2.7</td>
</tr>
<tr>
<td>Device (VOD)</td>
<td>0.5</td>
<td>11.8</td>
<td>1.5</td>
<td>1.5</td>
<td>10.3</td>
</tr>
<tr>
<td>Region (live)</td>
<td>0.6</td>
<td>0.7</td>
<td>1.3</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>Region (VOD)</td>
<td>0.1</td>
<td>0.3</td>
<td>1.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Connectivity (live)</td>
<td>0.7</td>
<td>1.1</td>
<td>1.4</td>
<td>1.1</td>
<td>1.5</td>
</tr>
<tr>
<td>Connectivity (VOD)</td>
<td>0.1</td>
<td>0.4</td>
<td>1.1</td>
<td>1.4</td>
<td>1.3</td>
</tr>
</tbody>
</table>

Y: the factor we are considering
X: the factor we could split along
Using the Model

• Output a decision tree that can predict the user QoE
• Use this to select CDN server

origin server in North America

CDN distribution node

CDN server in S. America

CDN server in Europe

CDN server in Asia

Video metrics

Video metrics

Video metrics

???