CPSC 4040/6040
Computer Graphics
Images

Joshua Levine
levinej@clemson.edu
Lecture 24
Image Compression
Dec. 1, 2015

Slide Credits:
Kenny A. Hunt
Justin Solomon
Agenda

- Grades / Final Exam Exemption
- Final Project Schedule
Refresher from Lec23
Each Coefficient Corresponds to a Basis “Sinusoid”
The Convolution Theorem

- Convolution in the spatial domain corresponds to **multiplication** in the Fourier domain.

\[ f \otimes h \Leftrightarrow F \cdot H \]

- This has strong computational implications

- Recall that convolution of an NxN image with MxM kernel is \( O(M^2N^2) \) in the spatial domain.

- What is the performance in the Fourier domain? \( O(N^2) \)!!
Convolution Theorem Example

\[ f \ast h \]

\[ F \ast H \]
Match the spatial domain image to the Fourier amplitude image.
Frequency Domain and Perception of Images
Recall: Contrast Sensitivity Function

Where the bands can be distinguished depends on both the person and distance.
Da Vinci and Peripheral Vision
Our Visual System Detects Various Frequencies Depending on Which Part of the Eye Receives Them

http://www.mindsmachine.com/asf07.04.html
Image Compression
What is Image Compression and Why Do We Need/Want It?

• Goal: Minimize the memory footprint of image data

• Storage capacity can be limited, as is the case with digital cameras

• Storage can be costly, as is the case when creating large warehouses of image data.

• Transmission of image data is the more pressing bottleneck:
  
  • Recent studies of web use, for example, have estimated that images and video account for approximately 85% of all Internet traffic.

  • We need to display, transfer, move, and send images.
Data Rates for Image and Video

• Full screen Image
  • 1024x768x24b = ~2.5MB

• DVD
  • 720x480x24bx30f/s = ~30 MB/s

• High Definition DVD
  • 1920x1080x24bx30f/s = ~178MB/s

• Film
  • 4000x3000x36bx30f/s = ~1.5GB/s
  • 8 TB for one 90 minute movie!
Recall: Which image has the most data?

A

B

C

Recall: Two Concepts for Data Encoding

1. **Coherency**: the tendency for one portion of the image to be similar to another.
   - Could be spatial (nearby in (x,y)-space) sequential (nearby in linearized array), temporal (for video)

2. **Redundancy**: the amount of irrelevant or repeated information
   - Differences that the human eye cannot discern
   - E.g., a far away checkerboard looks grey
Image Compression (according to Hunt)

- Image compression works by identifying and eliminating redundant, or duplicated, data from a source image.

- There are three main sources of redundancy in image compression.
  1. **Interpixel redundancy**: Pixels that are in close proximity within an image are generally related to each other.
  2. **Psycho-visual redundancy**: Importance of each pixel related to how the human visual system perceives visible information.
  3. **Coding redundancy**: Required storage of each pixel is directly proportional to the number of bits required to encode them in a file.
Compression Ratio

- **Compression ratio** measures the number of bits that can be eliminated from an uncompressed representation of a source image.

- Let $N_1$ be the total number of bits required to store an uncompressed (raw) source image and let $N_2$ be the total number of bits required to store the compressed data. Define the compression ratio $C_r$ as

\[
C_r = \frac{N_1}{N_2}
\]

- Larger compression ratios indicate more effective compression.

- Smaller compression ratios indicate less effective compression.

- Compression ratios less than one indicate that the compressed representation is actually larger than the uncompressed representation.
Fidelity

• Root mean squared (RMS) error is a generally accepted way of measuring the quality of a compressed image as compared with the uncompressed original.

• RMS error is a measure of the difference between two same-sized images and is not related to the memory footprint of an image.

• Assume that a WxH image $I$ having B channels is compressed into image $I'$. The root mean square error is then given by

$$e_{rms} = \sqrt{\frac{\sum_{x=0}^{W-1} \sum_{y=0}^{H-1} \sum_{b=0}^{B-1} (I'(x, y)_b - I(x, y)_b)^2}{W \cdot H \cdot B}}.$$  

(10.3)
Figure 10.2. Compression and bandwidth.
Some Techniques We Know a Bit About
Techniques We’ve Already Seen #1: Run-Length Encoding

- Encode repetitions in the sequence

*Figure 10.3. Run length encoding.*
RLE and Bit Planes

- Images can be decomposed into separate monochrome images by bit plane slicing.

- A 3x2 4-bit grayscale image is shown in part (b) where the samples are displayed in binary notation.

- The least significant bit of each sample forms the 0th bit plane: \( P_0 \)

- The most significant bit of each sample forms the 3rd bit plane: \( P_3 \)

- The bit planes are shown as the four binary images of parts (b) through (e).

<table>
<thead>
<tr>
<th>1101</th>
<th>0111</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100</td>
<td>1011</td>
<td>1001</td>
</tr>
</tbody>
</table>

(a) Source image.  
(b) \( P_0 \)  
(c) \( P_1 \)  
(d) \( P_2 \)  
(e) \( P_3 \)

**Figure 10.4.** The bit planes of a 4 bpp source image.
Gray Coding

- Compression ratio improves if there are more long runs than short runs.

- Consider two adjacent samples that have similar grayscale values of 127 and 128. In binary, these are given as 01111111 and 10000000. Each run on each bit plane is terminated by this small change in grayscale value.

- Can usually lengthen runs by using gray coding rather than binary-coded-decimal encoding. Gray codes ensure that two adjacent values differ by a single binary bit.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Gray</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>111</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>100</td>
</tr>
</tbody>
</table>
Figure 10.5. Bit planes as binary images.

Figure 10.6. Gray coded bit planes.
Techniques We’ve Already Seen #2: Color Indexed Images

- One alternative for true-color images having a small number of colors in the image. Instead, use color indexing.
  - Each pixel is a single byte.
  - This byte is an index into a color table (or palette).
  - The color is indirectly given by the pixel value.

- How many possible colors? Relationship between color depth and table size.

(a) Source image. (b) Indexed representation.
Color Space Reduction: Same Image w/ Less Colors

GIF:
Stores 256 colors in a palette; colors are 8 bits per primary ($2^{24}$)

Also, use LZW encoding to compress sequences of color -- See House 5.3
Color Dithering

- Approximate a wider range of colors with a small palette using pixels of two or more colors to approximate in-between colors.
Lossy vs. Lossless
Lossless Compression

• No information is lost during compression; the original can be reconstructed exactly.

• A lossless technique is one which always produces a compressed image with an RMS error of 0 relative to the source.
Lossy Compression

- Some information is lost during compression to achieve a smaller size.
- A lossy technique generates a compressed image that is not identical to the source.
- Lossy techniques are typically able to achieve greater compression ratios by sacrificing the quality of the result.
Amount of information in an image isn’t necessarily the same as the number of pixels!
Entropy \textit{[en-truh-pee]}: Average missing information content when one does not know the value of a random variable.
Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

Gettysburg Address, Abraham Lincoln (1863)
Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

Gettysburg Address, Abraham Lincoln (1863)
Image Entropy

Increasing entropy

http://media.photobucket.com/image/recent/phoenixeous/SmurfIceCream.jpg
http://cdn.smosh.com/sites/default/files/bloguploads/rebecca-black-hipster.png
http://www.quasimondo.com/hydra/sineNoise1.jpg
Another View of Lossless Compression

- Exchange more entropy (information per bit) for less space.

- Also known as:
  - Source coding
  - Bit-rate reduction

- Goal: Eliminate statistical redundancy
Huffman Coding
Huffman Coding

• A lossless compression technique

• Starts with building a table of how frequent each color is used in the image

<table>
<thead>
<tr>
<th>Colors</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.1</td>
</tr>
<tr>
<td>B</td>
<td>0.15</td>
</tr>
<tr>
<td>C</td>
<td>0.3</td>
</tr>
<tr>
<td>D</td>
<td>0.16</td>
</tr>
<tr>
<td>E</td>
<td>0.29</td>
</tr>
</tbody>
</table>
Huffman Coding

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.1</td>
</tr>
<tr>
<td>B</td>
<td>0.15</td>
</tr>
<tr>
<td>C</td>
<td>0.3</td>
</tr>
<tr>
<td>D</td>
<td>0.16</td>
</tr>
<tr>
<td>E</td>
<td>0.29</td>
</tr>
</tbody>
</table>
Huffman Coding

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.1</td>
</tr>
<tr>
<td>B</td>
<td>0.15</td>
</tr>
<tr>
<td>C</td>
<td>0.3</td>
</tr>
<tr>
<td>D</td>
<td>0.16</td>
</tr>
<tr>
<td>E</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Smallest values: A, B
Huffman Coding

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>0.25</td>
</tr>
<tr>
<td>C</td>
<td>0.3</td>
</tr>
<tr>
<td>D</td>
<td>0.16</td>
</tr>
<tr>
<td>E</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Diagram:
- AB (.25)
- A (.10)
- B (.15)
- C (.3)
- D (.16)
- E (.29)
Huffman Coding

Letter Frequency

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>0.25</td>
</tr>
<tr>
<td>C</td>
<td>0.3</td>
</tr>
<tr>
<td>D</td>
<td>0.16</td>
</tr>
<tr>
<td>E</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Diagram:
- AB (0.25) at the top
- A (0.1) and B (0.15) at the bottom left
- C (0.3) and D (0.16) at the bottom right
- E (0.29) at the bottom center
Huffman Coding

Letter | Frequency
--- | ---
ABD | 0.41
C | 0.3
E | 0.29
Huffman Coding

Letter | Frequency
--- | ---
ABD | 0.41
C | 0.3
E | 0.29
Huffman Coding

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABD</td>
<td>0.41</td>
</tr>
<tr>
<td>CE</td>
<td>0.59</td>
</tr>
</tbody>
</table>

A: 0.1
B: 0.15
C: 0.3
D: 0.16
E: 0.29

ABD: 0.41
CE: 0.59
Huffman Coding

Letter | Frequency
--- | ---
ABCDE | 1

ABD | .41
AB | .25
A | .1
B | .15
D | .16
CE | .59
C | .3
E | .29
Huffman Coding
Huffman Coding

A = 000
Huffman Coding

A = 000
D = 01

ABD

AB

A

B

C

D

CE

DE

E

ABC

1

0

1

0

1

0

1

1

1

.59

.25

.15

.1

.16

.3

.29

.41

1

A

D
Huffman Coding

A = 000
D = 01
C = 10
Huffman Coding

Encoding

CAD

= 10000001
Huffman Coding

Decoding

0011101 =
Decoding

0011101  
=  

Huffman Coding
Huffman Coding

Decoding

0011101

=
Decoding

001111011 = B
Decoding

0011101 = B
Decoding

00111011

= B
Decoding

0011101

= BE
Decoding

00111101

= BE

Huffman Coding
Decoding

00111101 = BE
Decoding

00111101

= 

BED
Decoding

00111011 = BED
Huffman Coding

Decoding

00111101 = BED

Entropy encoding
JPEG Overview

• JPEG is a file format developed by the Joint Photographic Experts Group.

• JPEG is a broad standard

  • Supports both lossy and lossless compression

• The most common implementation is JFIF (Joint File Interchange Format). Most often people mean JFIF when they use the term JPEG.

• JFIF is based on the DCT coefficients of an image
JPEG/JFIF Pipeline

1. Optionally convert color spaces

2. Reduce chroma data (usually by 2x)

3. Apply discrete cosine transform to 8x8 blocks

4. Quantize DCT values

5. Store result using lossless encoding (RLE, Huffman)

Steps 2 & 4 = Lossy Portions
JPEG/JFIF Pipeline

1. **Color Space Conversion**: R, G, B → YCbCr
2. **Subsampling**: YCbCr → Y
3. **Quantization**: Y → Quantized Y
4. **DCT**: Quantized Y → DCT
5. **8x8 Tiles**: DCT → 8x8 Tiles
6. **ZigZag Scan**: 8x8 Tiles → ZigZag Scan
7. **Delta Coding**: ZigZag Scan → Delta Coding
8. **Huffman Coding**: Delta Coding + Huffman Coding → 10010100...
9. **Run-Length Encoding (RLE)**: AC (Adjacent Components) → RLE
JPEG Quantization

• After color space conversion, tile into 8x8 blocks, offset by -128 and take the DCT

• Quantize the coefficients. Divide by small numbers in the upper-left of each tile and large numbers in the lower-right.

• More bits for lower frequencies
  • ...sometimes throw out high frequencies altogether!

http://en.wikipedia.org/wiki/JPEG
JPEG Example (Quality Increases Left to Right)
Blockiness Artifacts

http://en.wikipedia.org/wiki/JPEG
JPEG2000

• Similar to JPEG, but replace DCT with Haar Wavelet basis.