Digital Image Processing

- Color Image Processing
- Image Compression
Topics

- Color Image Processing
  - Color Models
  - RGB
  - YIQ
  - HSI
- Image Compression
  - Lossy and lossless methods
  - Run length coding
  - Entropy coding
  - Arithmetic coding
  - Jpeg coding
Color Image Processing: Motivation

- Color is an important feature for detecting and identifying objects
  - Human face detection
  - License plate detection
  - Content based image retrieval systems
- Image segmentation can be performed base on color values
- Tracking moving objects in video (surveillance)
- Etc.
Color Spectrum

![Diagram of the color spectrum showing white light, optical prism, infrared, and ultraviolet light.]
Visible Wavelength
Absorption of light by human eye
Red, Green, Blue Color Cube
RGB Color Cube
YIQ Color Model

- YIQ is the color space used by the NTSC color TV system
- The Y component represents the luma information, and is the only component used by black-and-white television receivers
- I and Q represent the chrominance information
- For example, applying a histogram equalization to a color image is done by Y component only
YIQ Color Model Conversion

\[
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix} =
\begin{bmatrix}
1 & 0.9563 & 0.6210 \\
1 & -0.2721 & -0.6474 \\
1 & -1.1070 & +1.7046
\end{bmatrix}
\begin{bmatrix}
Y \\
I \\
Q
\end{bmatrix}
\]

\[
\begin{bmatrix}
Y \\
I \\
Q
\end{bmatrix} =
\begin{bmatrix}
0.299 & 0.587 & 0.114 \\
0.595716 & -0.274453 & -0.321263 \\
0.211456 & -0.522591 & 0.311135
\end{bmatrix}
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]
HSI (or HSV) Color Model
Conversion from RGB to HSI

\[ H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \]

with

\[ \theta = \cos^{-1} \left\{ \frac{\frac{1}{2} [(R - G) + (R - B)]}{\left[ (R - G)^2 + (R - B)(G - B) \right]^{1/2}} \right\} \]

\[ S = 1 - \frac{3}{(R + G + B)} \min(R, G, B) \]

\[ I = \frac{1}{3} (R + G + B) \]
Enhancement using HSI Model
Color Space Conversions in MATLAB (1)

- To convert from RGB to HSV use `rgb2hsv`
- For HSV to RGB conversion use `hsv2rgb`
- E.g.
  - `I = imread('test.bmp');`
  - `H = rgb2hsv(I);`
Color Space Conversions in MATLAB (2)

- To convert from RGB color space into YIQ color space use rgb2ntsc
  - RGB = imread('sample.png');
  - YIQ = rgb2ntsc(RGB);
- ntsct rgb convert from YIQ to RGB
- Images can be displayed in RGB space only
Color Space Conversions in MATLAB (3)

- I component in YIQ (the first component) is equivalent to a gray scale conversion. However, you may also use rgb2gray either.
  - I=imread(‘test.bmp’);
  - YIQ = rgb2ntsc(I);
  - imshow( YIQ(:,:,1))
Image Compression
Image Compression

- Importance of Image Compression
  - Typical image resolution/depth
    - 1024 x 1024 x 24
    - Typical image size in bytes: 3 MB
  - Typical video rate/resolution/depth
    - 30 fps, 720×576, 24 bits
    - Typical video size for 1 minute: More than 2 GB
Data Redundancy

- Redundancy in information theory is the number of bits used to transmit a message minus the number of bits of actual information in the message.

- Informally, it is the amount of wasted "space" used to transmit certain data
Data Compression

- Data compression or source coding is the process of encoding information using fewer bits (or other information-bearing units) than an un-encoded representation would use through use of specific encoding schemes.
Compression Methods

- Lossless Methods
- Lossy Methods
Run Length Coding

- In this method runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count.
- Useful in relatively simple graphic images such as icons, line drawings, and animations.
- Example,

```
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWBBBWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWBWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
```

is stored as:

```
12W1B12W3B24W1B14W
```
Entropy Coding

- Entropy coding is a lossless coding method.
- Entropy coding creates variable length codeword. The length of each codeword is approximately proportional to the negative logarithm of its probability.
- Huffman Coding
  Huffman coding is an entropy encoding algorithm used for lossless data compression.
- The term *Entropy Coding* refers to the use of a variable-length code table for encoding a source symbol.
Example of Variable Length Coding

<table>
<thead>
<tr>
<th>$r_k$</th>
<th>$p_r(r_k)$</th>
<th>Code 1</th>
<th>$l_1(r_k)$</th>
<th>Code 2</th>
<th>$l_2(r_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{87} = 87$</td>
<td>0.25</td>
<td>01010111</td>
<td>8</td>
<td>01</td>
<td>2</td>
</tr>
<tr>
<td>$r_{128} = 128$</td>
<td>0.47</td>
<td>100000000</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$r_{186} = 186$</td>
<td>0.25</td>
<td>11000100</td>
<td>8</td>
<td>000</td>
<td>3</td>
</tr>
<tr>
<td>$r_{255} = 255$</td>
<td>0.03</td>
<td>11111111</td>
<td>8</td>
<td>001</td>
<td>3</td>
</tr>
<tr>
<td>$r_k$ for $k \neq 87, 128, 186, 255$</td>
<td>0</td>
<td>—</td>
<td>8</td>
<td>—</td>
<td>0</td>
</tr>
</tbody>
</table>
Huffman Coding Example

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Original source</th>
<th>Source reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_2$</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>$a_6$</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$a_1$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.06</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>$a_5$</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Huffman Coding Example

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Code</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_2$</td>
<td>0.4</td>
<td>1</td>
<td>0.4</td>
<td>1</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>$a_6$</td>
<td>0.3</td>
<td>00</td>
<td>0.3</td>
<td>00</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>$a_1$</td>
<td>0.1</td>
<td>011</td>
<td>0.1</td>
<td>011</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.1</td>
<td>0100</td>
<td>0.1</td>
<td>0100</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.06</td>
<td>01010</td>
<td>0.1</td>
<td>0101</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>$a_5$</td>
<td>0.04</td>
<td>01011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source reduction:

- $0.6 \to 0$
- $0.4 \to 1$
Arithmetic Coding

- The output from an arithmetic coding process is a single number less than 1 and greater than or equal to 0.
- This single number can be uniquely decoded to create the exact stream of symbols that went into its construction.
Example: $a_1 a_2 a_3 a_3 a_4$

<table>
<thead>
<tr>
<th>Source Symbol</th>
<th>Probability</th>
<th>Initial Subinterval</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>0.2</td>
<td>[0.0, 0.2)</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0.2</td>
<td>[0.2, 0.4)</td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.4</td>
<td>[0.4, 0.8)</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.2</td>
<td>[0.8, 1.0)</td>
</tr>
</tbody>
</table>
Example
Lossy Compressions

- A lossy compression is a data compression method which discards (loses) some of the data, in order to achieve a high compression rate, with the result that decompressing the data yields content that is different from the original one.
JPEG Standard

- Steps followed in JPEG
  - Color space transformation
  - Down-sampling
  - Block splitting
  - Discrete cosine transform
  - Quantization
  - Run length and Entropy coding
Color space transformation

- First, the image should be converted from RGB into YCbCr color space.
- The compression is more efficient as the brightness information, which is more important to the eventual perceptual quality of the image, is confined to a single channel, more closely representing the human visual system.
Down Sampling

- Due to the densities of color- and brightness sensitive receptors in the human eye, humans can see considerably more fine detail in the brightness of an image (the Y component) than in the color of an image (the Cb and Cr components).
- Down-sample Cb and Cr components only 4:2:0
Splitting the Frame into Blocks

- After sub-sampling, each channel must be split into 8×8 blocks of pixels.
Discrete cosine transform

\[
\begin{bmatrix}
52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\
63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\
62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\
63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\
67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\
79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\
85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\
87 & 79 & 69 & 68 & 65 & 76 & 78 & 94
\end{bmatrix}
\]
Discrete Cosine Transform

\[ G_{u,v} = \alpha(u) \alpha(v) \sum_{x=0}^{7} \sum_{y=0}^{7} g_{x,y} \cos \left[ \frac{\pi}{8} \left( x + \frac{1}{2} \right) u \right] \cos \left[ \frac{\pi}{8} \left( y + \frac{1}{2} \right) v \right] \]

\[ \alpha_p(n) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } n = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases} \]
Result of DCT
Quantization

- Quantization is done by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer.
Sample Quantization Table

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

\[
\begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Zig-Zag Scanning and Run Length Coding
Zig-Zag Scanning

\[
\begin{array}{cccc}
-26 & 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 & 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 \\
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 \\
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 \\
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 \\
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 \\
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 \\
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 \\
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 \\
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 \\
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 \\
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 \\
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 \\
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 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \</array>\]
Variable length codes

- Variable length codes are generated for the code words of the zig-zag scan result.
- Trailing zeros are truncated by inserting an EOB symbol after the last non-zero value.
Typical compression ratios

- The compression ration depends on the quantization factors used.
- Typical ratio is 4:1
Questions?