The DCT domain and JPEG
CSM25 Secure Information Hiding

Dr Hans Georg Schaathun
University of Surrey
Spring 2007

Overview

The elements of JPEG

- Operates on luminence and chrominance (YCbCr) (not on RGB)
  - Grayscale images have luminence component only.
- Downsampling
  - Works in the DCT domain (not the spatial domain)
- Quantisation
- Entropy coding (lossless compression)

Overview

JPEG is not a file format

- JPEG is a compression system
  - The system employs three different compression techniques
- JPEG is not a file format.
- A common file format using JPEG is JFIF.
- Files with extension .jpeg are often JFIF.

Reading

Core Reading

*Digital Image Processing Using MATLAB.*

- Chapter 6: colour images
  - Representation
  - Processing
  - Conversion
- Chapter 8.5: JPEG compression
- Chapter 4: Frequency domain processing
The RGB colour representations

- RGB: A colour is a vector \((R, G, B)\)
  - \(R\) is amount of red light.
  - \(G\) is amount of green light.
  - \(B\) is amount of blue light.
- Each pixel can be either
  - A colour vector \((R, G, B)\); or
  - a reference to an array of colour vectors (the palette)
- Each coefficient can be
  - \(\in [0, 1]\); floating point (\texttt{double} in matlab)
  - \(\in \{0, 1, \ldots, 255\}\); 8-bit integer (\texttt{uint8} in matlab)
  - \(\in \{0, 1, \ldots, 2^{16} - 1\}\); 16-bit integer (\texttt{uint16} in matlab)

Dr Hans Georg Schaathun

Alternatives to RGB

- NTSC: \((Y, I, Q)\)
  \[
  \begin{bmatrix}
  Y \\
  I \\
  Q
  \end{bmatrix}
  =
  \begin{bmatrix}
  0.299 & 0.587 & 0.114 \\
  0.596 & -0.274 & -0.322 \\
  0.211 & -0.523 & 0.312
  \end{bmatrix}
  \begin{bmatrix}
  R \\
  G \\
  B
  \end{bmatrix}
  \]
  where \(R, G, B \in [0, 1]\).
- YCbCr: \((Y, Cb, Cr)\)
  \[
  \begin{bmatrix}
  Y \\
  Cb \\
  Cr
  \end{bmatrix}
  =
  \begin{bmatrix}
  16 \\
  128 \\
  128
  \end{bmatrix}
  +
  \begin{bmatrix}
  65.481 & 128.553 & 24.966 \\
  -37.797 & -74.203 & 112.000 \\
  112.000 & -93.786 & -18.214
  \end{bmatrix}
  \begin{bmatrix}
  R \\
  G \\
  B
  \end{bmatrix}
  \]
  where \(R, G, B \in [0, 1]\) and \(Y, Cb, Cr \in [0, 255]\).

Dr Hans Georg Schaathun

The DCT domain and JPEG

Block-wise

- Each colour-channel (Y,Cb,Cr) considered separately
- \(M \times N\) matrix divided into \(8 \times 8\) blocks
- Each block is handled separately

Dr Hans Georg Schaathun

The DCT transform

\[
T_f(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \sqrt{\frac{\alpha(u)\alpha(v)}{MN}} \cos \frac{(2x+1)u\pi}{2M} \cos \frac{(2y+1)v\pi}{2N}
\]

where

\[
\alpha(a) = \begin{cases} 
1, & \text{if } a = 0, \\
2, & \text{otherwise}.
\end{cases}
\]

\(M = N = 8\).
The DCT transform

- The inverse is similar

\[ f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T_i(u, v) \sqrt{\frac{\alpha(u)\alpha(v)}{MN}} \cos \left( \frac{2x + 1}{2M} \pi u \right) \cos \left( \frac{2y + 1}{2N} \pi v \right) \]

where

\[ \alpha(a) = \begin{cases} 1, & \text{if } a = 0, \\ 2, & \text{otherwise}. \end{cases} \]

\[ M = N = 8, \]

Matlab

- Matlab functions
  - `dct2` (2D DCT transform)
  - `idct2` (Inverse)
  - `blkproc (X, [M N], FUN )`

- For instance
  - `blkproc (X, [8 8], @dct2 )`

- Use help system for details
- Unfortunately, we do not have the JPEG toolbox.
  - Loading JPEG images converts them to the spatial domain.

Exercise

- Take one of your previous cover images (grayscale to keep it simple).
- Create the DCT transform of the image, and show it.
- Create the blockwise DCT transform of the image, and show it.
- Check that you can recover the original from the transform using `idct2`.

Hint: Use `dct2`, `idct2`, `blkproc`.

Transform image

- Linear combination of patterns (see right)
- DC (upper left) gives average colour intensity
- Low frequency: coarse structure
- High frequency: fine details
**What is sampling?**

**Fact**

*The human eye is more sensitive to changes in luminance than in chrominance.*

- To sample is to collect measurements.
  - Each pixel is a sample (measuring the colour of the image).
  - Lower resolution means fewer samples.
- Reducing resolution = downsampling
- Basic $M \times N$ image: $N \cdot M$ samples per component ($Y$, $Cb$, $Cr$).
- $Y$ is more useful than $Cb$ and $Cr$.
- Therefore we can downsample $Cb$ and $Cr$
  - $M/2 \times N/2$ is common for $Cb$ and $Cr$
  - Still use $M \times M$ for $Y$

---

**What do we save?**

- Original: $M \times N$ pixels $\times 3$ components.
- Compressed:
  
  $$2 \times \frac{M}{2} \times \frac{N}{2} + M \times N = \frac{1}{2} M \times N$$

- Ratio
  
  $$\frac{\text{Compressed}}{\text{Original}} = \frac{\frac{1}{2} MN}{3MN} = \frac{1}{2}$$

- We just saved 50%

---

**Chrominance versus Luminence**

**Fact**

*The human eye is more sensitive to changes in luminance than in chrominance.*

- Watermarking tend to embed in $Y$ (luminence)
- Embedding in $Cb$ and $Cr$ would more easily be destroyed by JPEG

---

**Downsampling in JPEG**

- Translation to YCbCr.
- Downsampling
- DCT transform
  - Each downsampled component matrix $Y$, $Cb$, $Cr$ is
    - Divided into $8 \times 8$ blocks
    - DCT transformed blockwise
  - An $8 \times 8$ block in $Cb$ can be associated with 1,2 or 4 $Y$ blocks depending on downsampling.
What is quantisation?

Rounding in general

- Rounding numbers is quantisation.
- Measuring gives continuous numbers
  - Whether you measure pixel luminence, or the length of your garage.
  - No matter how close to points are, there is a point in between.
- However, our precision is limited.
  - We give lengths to the nearest unit.
  - Luminence is categorised into 256 intervals (8bit integers).
- Computer memory is finite, 256 different possibilities for a byte

Quantisation in JPEG

- Quantisation in the DCT domain
- Each coefficient is divided by the quantisation constant.
- The result is rounded to nearest integer.
- Different quantisation constants for each coefficient in the block.

Example

Quantisation in JPEG

```
DCT matrix
-415 -30 -61 27 56 -20 -2 2
-47 7 77 -25 -29 10 5 -6
-49 12 34 -15 -10 6 2 2
12 -7 -13 -4 -2 2 -3 -2
-8 3 2 -6 -2 1 4 2
-1 0 0 -2 -1 -3 4 1
0 0 -1 -4 -1 0 1 2
```

Quantisation matrix

```
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
49 64 79 87 103 121 120 92
72 92 95 98 112 100 103 99
```

Observe

0 is extremely common
±1 is common
Two-digit numbers are very rare

This is typical
In order to compress the data
- Use few bits (short codewords) for frequent symbols
- Many bits (long codewords) only for rare symbols

Usually, JPEG uses a simple Huffman code.
- It can use other codes (saving space, but computationally costly)

For instance, a single short codeword to say
- ‘the rest of the block is zero’