JPEG AND JPEG2000

Catherine Bénéteau, Caroline Haddad, David Ruch and Patrick J. Van Fleet

Center for Applied Mathematics
University of St. Thomas
St. Paul, MN USA

PREP - Wavelet Workshop, 2008
Today’s Schedule

9:00-10:15  Lecture Nine: The Lifting Method
10:15-10:30  Coffee Break (OSS 237)
10:30-11:45  Lecture Ten: JPEG and JPEG2000
11:45-12:00  Evaluations/Wrap Up
12:00-1:00  Lunch (Cafeteria)
OUTLINE

TODAY’S SCHEDULE

BASIC JPEG
- Naive Algorithm
- An Example

JPEG2000
- Features and Enhancements
- Basic Algorithm
- The Cohen/Daubechies/Feauveau (CDF97) Filter
- The Quantization Process
- An Example (Reprise)
Given an \( N \times N \) image \( A \) (\( A \) square and \( N \) is divisible by 8 for simplicity):

- Subtract 127 from each element in \( A \).
- Partition \( A \) into \( 8 \times 8 \) blocks.
- Transform each block using the Discrete Cosine Transform (DCT).
- Quantize the elements in each block.
- Encode using Huffman encoding (more sophisticated version).

Note that the quantization step makes JPEG an example of lossy compression.
Given an $N \times N$ image $A$ ($A$ square and $N$ is divisible by 8 for simplicity):

- Subtract 127 from each element in $A$.
- Partition $A$ into $8 \times 8$ blocks.
- Transform each block using the Discrete Cosine Transform (DCT).
- Quantize the elements in each block.
- Encode using Huffman encoding (more sophisticated version).

Note that the quantization step makes JPEG an example of lossy compression.
Given an $N \times N$ image $A$ ($A$ square and $N$ is divisible by 8 for simplicity):

- **Subtract** 127 from each element in $A$.
- **Partition** $A$ into $8 \times 8$ blocks.
- **Transform** each block using the *Discrete Cosine Transform (DCT)*.
- **Quantize** the elements in each block.
- **Encode** using Huffman encoding (more sophisticated version).

Note that the quantization step makes JPEG an example of lossy compression.
Given an $N \times N$ image $A$ ($A$ square and $N$ is divisible by 8 for simplicity):

- Subtract 127 from each element in $A$.
- Partition $A$ into $8 \times 8$ blocks.
- Transform each block using the Discrete Cosine Transform (DCT).
- Quantize the elements in each block.
- Encode using Huffman encoding (more sophisticated version).

Note that the quantization step makes JPEG an example of lossy compression.
Given an $N \times N$ image $A$ ($A$ square and $N$ is divisible by 8 for simplicity):

- Subtract 127 from each element in $A$.
- Partition $A$ into $8 \times 8$ blocks.
- Transform each block using the Discrete Cosine Transform (DCT).
- Quantize the elements in each block.
- Encode using Huffman encoding (more sophisticated version).

Note that the quantization step makes JPEG an example of lossy compression.
Given an $N \times N$ image $A$ (a square and $N$ is divisible by 8 for simplicity):

- Subtract 127 from each element in $A$.
- Partition $A$ into $8 \times 8$ blocks.
- Transform each block using the Discrete Cosine Transform (DCT).
- Quantize the elements in each block.
- Encode using Huffman encoding (more sophisticated version).

Note that the quantization step makes JPEG an example of lossy compression.
Given an $N \times N$ image $A$ ($A$ square and $N$ is divisible by 8 for simplicity):

- Subtract 127 from each element in $A$.
- Partition $A$ into $8 \times 8$ blocks.
- Transform each block using the Discrete Cosine Transform (DCT).
- Quantize the elements in each block.
- Encode using Huffman encoding (more sophisticated version).

Note that the quantization step makes JPEG an example of lossy compression.
Consider the moon surface image stored in A. The dimensions are $200 \times 200$. 

![Image of moon surface]
We partition $A$ into $8 \times 8$ blocks. There are $25 \times 25$ blocks. We have highlighted one block (call it $A_{23}$) for use as a running example:
The elements in $A_{23}$ are:

$$
A_{23} = \begin{bmatrix}
140 & 142 & 127 & 126 & 124 & 129 & 135 & 129 \\
142 & 142 & 132 & 125 & 125 & 124 & 130 & 133 \\
143 & 135 & 122 & 134 & 140 & 115 & 124 & 139 \\
121 & 93 & 104 & 159 & 209 & 166 & 117 & 130 \\
96 & 58 & 83 & 150 & 220 & 224 & 142 & 123 \\
81 & 37 & 68 & 124 & 203 & 231 & 161 & 113 \\
88 & 29 & 56 & 101 & 166 & 201 & 141 & 118 \\
126 & 72 & 57 & 92 & 131 & 135 & 135 & 133
\end{bmatrix}
$$
Shifting by 127 gives

$\tilde{A}_{23} = \begin{bmatrix}
13 & 15 & 0 & -1 & -3 & 2 & 8 & 2 \\
15 & 15 & 5 & -2 & -2 & -3 & 3 & 6 \\
16 & 8 & -5 & 7 & 13 & -12 & -3 & 12 \\
-6 & -34 & -23 & 32 & 82 & 39 & -10 & 3 \\
-31 & -69 & -44 & 23 & 93 & 97 & 15 & -4 \\
-46 & -90 & -59 & -3 & 76 & 104 & 34 & -14 \\
-39 & -98 & -71 & -26 & 39 & 74 & 14 & -9 \\
-1 & -55 & -70 & -35 & 4 & 8 & 8 & 6
\end{bmatrix}$
The DCT is the $8 \times 8$ matrix

$$U = \frac{1}{2} \begin{bmatrix}
\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\
\cos(\frac{\pi}{16}) & \cos(\frac{3\pi}{16}) & \cos(\frac{5\pi}{16}) & \cos(\frac{7\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{11\pi}{16}) & \cos(\frac{13\pi}{16}) & \cos(\frac{15\pi}{16}) \\
\cos(\frac{\pi}{8}) & \cos(\frac{3\pi}{8}) & \cos(\frac{5\pi}{8}) & \cos(\frac{7\pi}{8}) & \cos(\frac{9\pi}{8}) & \cos(\frac{11\pi}{8}) & \cos(\frac{13\pi}{8}) & \cos(\frac{15\pi}{8}) \\
\cos(\frac{3\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{27\pi}{16}) & \cos(\frac{33\pi}{16}) & \cos(\frac{39\pi}{16}) & \cos(\frac{45\pi}{16}) \\
\cos(\frac{\pi}{4}) & \cos(\frac{3\pi}{4}) & \cos(\frac{5\pi}{4}) & \cos(\frac{7\pi}{4}) & \cos(\frac{9\pi}{4}) & \cos(\frac{11\pi}{4}) & \cos(\frac{13\pi}{4}) & \cos(\frac{15\pi}{4}) \\
\cos(\frac{5\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{25\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{45\pi}{16}) & \cos(\frac{55\pi}{16}) & \cos(\frac{65\pi}{16}) & \cos(\frac{75\pi}{16}) \\
\cos(\frac{3\pi}{8}) & \cos(\frac{9\pi}{8}) & \cos(\frac{15\pi}{8}) & \cos(\frac{21\pi}{8}) & \cos(\frac{27\pi}{8}) & \cos(\frac{33\pi}{8}) & \cos(\frac{39\pi}{8}) & \cos(\frac{45\pi}{8}) \\
\cos(\frac{7\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{49\pi}{16}) & \cos(\frac{63\pi}{16}) & \cos(\frac{77\pi}{16}) & \cos(\frac{91\pi}{16}) & \cos(\frac{105\pi}{16})
\end{bmatrix}$$
Row $j$ of $U$, $j = 2, \ldots, 7$ is formed by evaluating $\cos t$ at 8 uniformly spaced points on the interval $[0, j\pi)$.

- We won’t talk much about $U$ today, but we will note:
  - $U$ is orthogonal so $U^{-1} = U^T$.
  - $U$ maps the vector $(1, 1, 1, 1, 1, 1, 1, 1)^T$ to $(2\sqrt{2}, 0, 0, 0, 0, 0, 0, 0)^T$.
- To apply the DCT to $A_{23}$, we compute $U A U^T$. 
Row $j$ of $U$, $j = 2, \ldots, 7$ is formed by evaluating $\cos t$ at $8$ uniformly spaced points on the interval $[0, j\pi)$.

We won't talk much about $U$ today, but we will note:

- $U$ is orthogonal so $U^{-1} = U^T$.
- $U$ maps the vector $(1, 1, 1, 1, 1, 1, 1, 1)^T$ to $(2\sqrt{2}, 0, 0, 0, 0, 0, 0, 0)^T$.
- To apply the DCT to $A_{23}$, we compute $U A U^T$. 

Row $j$ of $U$, $j = 2, \ldots, 7$ is formed by evaluating $\cos t$ at 8 uniformly spaced points on the interval $[0, j\pi)$.

We won’t talk much about $U$ today, but we will note:

- $U$ is *orthogonal* so $U^{-1} = U^T$.
- $U$ maps the vector $(1, 1, 1, 1, 1, 1, 1, 1)^T$ to $(2\sqrt{2}, 0, 0, 0, 0, 0, 0, 0)^T$.

To apply the DCT to $A_{23}$, we compute $UAU^T$. 
Row $j$ of $U$, $j = 2, \ldots, 7$ is formed by evaluating $\cos t$ at 8 uniformly spaced points on the interval $[0, j\pi)$.

We won’t talk much about $U$ today, but we will note:

- $U$ is orthogonal so $U^{-1} = U^T$.
- $U$ maps the vector $(1, 1, 1, 1, 1, 1, 1, 1)^T$ to $(2\sqrt{2}, 0, 0, 0, 0, 0, 0, 0)^T$.

To apply the DCT to $A_{23}$, we compute $UAU^T$. 
Row $j$ of $U$, $j = 2, \ldots, 7$ is formed by evaluating $\cos t$ at 8 uniformly spaced points on the interval $[0, j\pi)$.

We won’t talk much about $U$ today, but we will note:

- $U$ is *orthogonal* so $U^{-1} = U^T$.
- $U$ maps the vector $(1, 1, 1, 1, 1, 1, 1, 1)^T$ to $(2\sqrt{2}, 0, 0, 0, 0, 0, 0, 0)^T$

To apply the DCT to $A_{23}$, we compute $UAU^T$. 
We have

\[ T_{23} = U\tilde{A}_{23}U^T \]

\[
= \begin{bmatrix}
-50.705 & 41.645 & 95.027 & -35.707 & -27.598 & 15.933 & -12.217 & -0.976 \\
\end{bmatrix}
\]
Perhaps a plot is more informative:

The DCT tends to store information about all 64 values into a few values and “shove” them to the upper left of the output. The remaining values are either 0 or approximately 0.
Here is a picture of the DCT of the entire image Ā:
The quantization step involves dividing element-wise the individual $8 \times 8$ blocks by the matrix

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

and then rounding the results to the nearest integer. Note that the rounding step is irreversible.
For our $8 \times 8$ block $T_{23}$ we have

\[
\begin{bmatrix}
0 & -12 & -8 & 8 & 2 & 0 & 0 & 0 \\
5 & 10 & 3 & -5 & -1 & 0 & 0 & 0 \\
-4 & 3 & 6 & -1 & -1 & 0 & 0 & 0 \\
1 & -4 & -3 & 2 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
We then add 127 to each element of each block, and then apply Huffman encoding to the result.

The compressed image can be represented using 57231 bits or 1.43078 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the image.

There are some drawbacks:

- The compression is lossy
- The uncompressed image is “blocky”
- The transform doesn’t handle edge effects
We then add 127 to each element of each block, and then apply Huffman encoding to the result.

The compressed image can be represented using 57231 bits or 1.43078 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the image.

There are some drawbacks:

- the compression is lossy
- the uncompressed image is “blocky”
- the transform doesn’t handle edge effects
We then add 127 to each element of each block, and then apply Huffman encoding to the result.

The compressed image can be represented using 57231 bits or 1.43078 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the image.

There are some drawbacks:
- the compression is lossy
- the uncompressed image is “blocky”
- the transform doesn’t handle edge effects
We then add 127 to each element of each block, and then apply Huffman encoding to the result.

The compressed image can be represented using 57231 bits or 1.43078 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the image.

There are some drawbacks:

- the compression is lossy
- the uncompressed image is “blocky”
- the transform doesn’t handle edge effects
We then add 127 to each element of each block, and then apply Huffman encoding to the result.

The compressed image can be represented using 57231 bits or 1.43078 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the image.

There are some drawbacks:
- the compression is lossy
- the uncompressed image is “blocky”
- the transform doesn’t handle edge effects
We then add 127 to each element of each block, and then apply Huffman encoding to the result.

The compressed image can be represented using 57231 bits or 1.43078 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the image.

There are some drawbacks:
- the compression is lossy
- the uncompressed image is “blocky”
- the transform doesn’t handle edge effects
We then add 127 to each element of each block, and then apply Huffman encoding to the result.

The compressed image can be represented using 57231 bits or 1.43078 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the image.

There are some drawbacks:

- the compression is lossy
- the uncompressed image is “blocky”
- the transform doesn’t handle edge effects
Compressed:
Original:
Here is a blow-up of the original:
Here is a blow-up of the compressed image:
Work on the new standard began in the late 1990’s (1997)

Here are some features of the new standards:

- Lossy and lossless compression The new standard has both options.
- Better compression rates than JPEG.
- Progressive Signal Transmission - JPEG2000 can reconstruct the digital image as it is received via the internet.
- Tiling - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
- Regions of interest - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
- Larger image size - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
- Multiple channels - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
Work on the new standard began in the late 1990’s (1997)

Here are some features of the new standards:

- **Lossy and lossless compression** The new standard has both options.
- **Better compression rates than JPEG.**
- **Progressive Signal Transmission** - JPEG2000 can reconstruct the digital image as it is received via the internet.
- **Tiling** - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
- **Regions of interest** - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
- **Larger image size** - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
- **Multiple channels** - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
Work on the new standard began in the late 1990’s (1997)
Here are some features of the new standards:

- **Lossy and lossless compression** The new standard has both options.
- Better compression rates than JPEG.
- **Progressive Signal Transmission** - JPEG2000 can reconstruct the digital image as it is received via the internet.
- **Tiling** - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
- **Regions of interest** - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
- **Larger image size** - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
- **Multiple channels** - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
Work on the new standard began in the late 1990’s (1997)

Here are some features of the new standards:

- **Lossy and lossless compression** The new standard has both options.
- **Better compression rates** than JPEG.
- **Progressive Signal Transmission** - JPEG2000 can reconstruct the digital image as it is received via the internet.
- **Tiling** - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
- **Regions of interest** - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
- **Larger image size** - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
- **Multiple channels** - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
Work on the new standard began in the late 1990’s (1997)
Here are some features of the new standards:

- **Lossy and lossless compression** The new standard has both options.
- **Better compression rates** than JPEG.
- **Progressive Signal Transmission** - JPEG2000 can reconstruct the digital image as it is received via the internet.
- **Tiling** - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
- **Regions of interest** - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
- **Larger image size** - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
- **Multiple channels** - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
Work on the new standard began in the late 1990’s (1997)

Here are some features of the new standards:

- **Lossy and lossless compression** The new standard has both options.
- **Better compression rates** than JPEG.
- **Progressive Signal Transmission** - JPEG2000 can reconstruct the digital image as it is received via the internet.
- **Tiling** - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
- **Regions of interest** - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
- **Larger image size** - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
- **Multiple channels** - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
Work on the new standard began in the late 1990’s (1997)

Here are some features of the new standards:

- **Lossy and lossless compression** The new standard has both options.
- **Better compression rates** than JPEG.
- **Progressive Signal Transmission** - JPEG2000 can reconstruct the digital image as it is received via the internet.
- **Tiling** - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
- **Regions of interest** - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
- **Larger image size** - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
- **Multiple channels** - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
▶ Work on the new standard began in the late 1990’s (1997)
▶ Here are some features of the new standards:
  ▶ **Lossy and lossless compression** The new standard has both options.
  ▶ **Better compression rates** than JPEG.
  ▶ **Progressive Signal Transmission** - JPEG2000 can reconstruct the digital image as it is received via the internet.
  ▶ **Tiling** - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
  ▶ **Regions of interest** - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
  ▶ **Larger image size** - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
  ▶ **Multiple channels** - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
Work on the new standard began in the late 1990’s (1997)

Here are some features of the new standards:

- **Lossy and lossless compression** The new standard has both options.
- **Better compression rates** than JPEG.
- **Progressive Signal Transmission** - JPEG2000 can reconstruct the digital image as it is received via the internet.
- **Tiling** - a generalization of the JPEG practice of partitioning images into $8 \times 8$ blocks.
- **Regions of interest** - JPEG2000 allows the user to identify ROIs and compress them at higher rates than other parts of the images.
- **Larger image size** - JPEG could handle images of size $64,000 \times 64,000$ or smaller while the maximum image size for JPEG2000 is $2^{32} - 1 \times 2^{32} - 1$.
- **Multiple channels** - JPEG could handle color images (3 channels) while JPEG2000 can handle up to 256 channels.
- Subtract 127 from each element of input grayscale image A.
- Apply $i$ iterations of the biorthogonal wavelet transform using the LeGall filter for lossless compression and the Cohen/Daubechies/Feauveau (9, 7) filter (more later) for lossy compression.
- For lossy compression, apply a quantizer (more later) much like that used for the wavelet shrinkage algorithm in denoising applications.
- Use an arithmetic coding system called Embedded Block Coding with Optimized Truncation (EBCOT) to the quantized transform.
Subtract 127 from each element of input grayscale image $A$.

Apply $i$ iterations of the biorthogonal wavelet transform using the LeGall filter for lossless compression and the Cohen/Daubechies/Feauveau (9, 7) filter (more later) for lossy compression.

For lossy compression, apply a quantizer (more later) much like that used for the wavelet shrinkage algorithm in denoising applications.

Use an arithmetic coding system called Embedded Block Coding with Optimized Truncation (EBCOT) to the quantized transform.
▶ Subtract 127 from each element of input grayscale image $A$.
▶ Apply $i$ iterations of the biorthogonal wavelet transform using the LeGall filter for lossless compression and the Cohen/Daubechies/Feauveau (9, 7) filter (more later) for lossy compression.
▶ For lossy compression, apply a quantizer (more later) much like that used for the wavelet shrinkage algorithm in denoising applications.
▶ Use an arithmetic coding system called Embedded Block Coding with Optimized Truncation (EBCOT) to the quantized transform.
▶ Subtract 127 from each element of input grayscale image $A$.
▶ Apply $i$ iterations of the biorthogonal wavelet transform using the LeGall filter for lossless compression and the Cohen/Daubechies/Feauveau $(9, 7)$ filter (more later) for lossy compression.
▶ For lossy compression, apply a quantizer (more later) much like that used for the wavelet shrinkage algorithm in denoising applications.
▶ Use an arithmetic coding system called **Embedded Block Coding with Optimized Truncation (EBCOT)** to the quantized transform.
The JPEG2000 committee decided they wanted to use wavelets instead of the DCT.

Application of the wavelet transform to the entire image reduces the block effects caused by applying the DCT to $8 \times 8$ blocks of the partitioned image.

Insisting the filters be symmetric helps remove boundary effects.

Committee members wanted long filters but with minimal difference in filter lengths.

Decided on lengths 9 and 7 but rejected the $(9, 7)$ biorthogonal spline filter.

The Fourier series for the filters are unbalanced at $\omega = \pi$. 
The JPEG2000 committee decided they wanted to use wavelets instead of the DCT.

Application of the wavelet transform to the entire image reduces the block effects caused by applying the DCT to $8 \times 8$ blocks of the partitioned image.

Insisting the filters be symmetric helps remove boundary effects.

Committee members wanted long filters but with minimal difference in filter lengths.

Decided on lengths 9 and 7 but rejected the (9, 7) biorthogonal spline filter.

The Fourier series for the filters are unbalanced at $\omega = \pi$. 
The JPEG2000 committee decided they wanted to use wavelets instead of the DCT.

Application of the wavelet transform to the entire image reduces the block effects caused by applying the DCT to $8 \times 8$ blocks of the partitioned image.

Insisting the filters be symmetric helps remove boundary effects.

Committee members wanted long filters but with minimal difference in filter lengths.

Decided on lengths 9 and 7 but rejected the $(9, 7)$ biorthogonal spline filter.

The Fourier series for the filters are unbalanced at $\omega = \pi$. 

---

Saturday, 7 June, 2008 (Lecture 10)
The JPEG2000 committee decided they wanted to use wavelets instead of the DCT.

Application of the wavelet transform to the entire image reduces the block effects caused by applying the DCT to $8 \times 8$ blocks of the partitioned image.

Insisting the filters be symmetric helps remove boundary effects.

Committee members wanted long filters but with minimal difference in filter lengths.

Decided on lengths 9 and 7 but rejected the $(9, 7)$ biorthogonal spline filter.

The Fourier series for the filters are unbalanced at $\omega = \pi$. 
The JPEG2000 committee decided they wanted to use wavelets instead of the DCT.

Application of the wavelet transform to the entire image reduces the block effects caused by applying the DCT to $8 \times 8$ blocks of the partitioned image.

Insisting the filters be symmetric helps remove boundary effects.

Committee members wanted long filters but with minimal difference in filter lengths.

Decided on lengths 9 and 7 but rejected the (9, 7) biorthogonal spline filter.

The Fourier series for the filters are unbalanced at $\omega = \pi$. 
The JPEG2000 committee decided they wanted to use wavelets instead of the DCT.

Application of the wavelet transform to the entire image reduces the block effects caused by applying the DCT to $8 \times 8$ blocks of the partitioned image.

Insisting the filters be symmetric helps remove boundary effects.

Committee members wanted long filters but with minimal difference in filter lengths.

Decided on lengths 9 and 7 but rejected the $(9, 7)$ biorthogonal spline filter.

The Fourier series for the filters are unbalanced at $\omega = \pi$. 
The Fourier series $\tilde{H}(\omega)$ for the 7-term filter (built from the binomial coefficients) $\tilde{h}$ satisfies $\tilde{H}^{(j)}(\pi) = 0$ for $j = 0, \ldots, 5$:
But Fourier series $H(\omega)$ for the 9-term filter $h$ satisfies $H^{(j)}(\pi) = 0$ for only $j = 0, 1$:
The CDF97 filter comes from the following results:

**Theorem**

If $\tilde{h}$ and $h$ are symmetric, odd-length biorthogonal filters, then their Fourier series $\tilde{H}(\omega)$ and $H(\omega)$ can be written as

$$
\tilde{H}(\omega) = \sqrt{2} \cos^2 \frac{\omega}{2} \tilde{p}(\cos(\omega))
$$

and

$$
H(\omega) = \sqrt{2} \cos^2 \frac{\omega}{2} p(\cos(\omega))
$$

where $\tilde{\ell}$ and $\ell$ are nonnegative integers and $\tilde{p}$ and $p$ are polynomials with $\tilde{p}(-1) \neq 0$, $p(-1) \neq 0$ and $\tilde{p}(1) = p(1) = 1$. 
and

**Theorem**

The polynomials \( \tilde{p} \) and \( p \) satisfy

\[
\tilde{p}(\cos(\omega))p(\omega) = \sum_{j=0}^{K-1} \left( K - 1 + j \right) \sin^2(j\omega) \]

where \( K = \tilde{\ell} + \ell \).

For the biorthogonal spline filters, \( \tilde{p}(t) = 1 \).
\[
\tilde{H}(\omega) = \sqrt{2} \cos^{\tilde{N}}(\frac{\omega}{2})
\]

and

\[
H(\omega) = \sqrt{2} \cos^{N}(\frac{\omega}{2}) \sum_{j=0}^{K-1} \left( K - 1 + j \right) \sin^{2j}(\frac{\omega}{2})
\]
The CDF97 filter is constructed by splitting the factors of the polynomial

\[ P(t) = \sum_{j=0}^{K-1} \binom{K - 1}{j} t^j \]

We use \( \tilde{\ell} = \ell = 2 \) so that \( K = \tilde{\ell} + \ell = 4 \) and consider the cubic polynomial

\[ P(t) = \sum_{j=0}^{3} \binom{3}{j} t^j = 1 + 4t + 10t^2 + 20t^3 \]

This polynomial has one real root \( r_1 = -0.342484 \) and two complex roots \( c_1 = -0.078808 - .373391i \), \( c_2 = -0.078808 + .373391i \).
The CDF97 filter is constructed by splitting the factors of the polynomial

\[ P(t) = \sum_{j=0}^{K-1} \binom{K-1}{j} t^j \]

We use \( \tilde{\ell} = \ell = 2 \) so that \( K = \tilde{\ell} + \ell = 4 \) and consider the cubic polynomial

\[ P(t) = \sum_{j=0}^{3} \binom{3}{j} t^j = 1 + 4t + 10t^2 + 20t^3 \]

This polynomial has one real root \( r_1 = -0.342484 \) and two complex roots \( c_1 = -0.078808 - .373391i \), \( c_2 = -0.078808 + .373391i \).
The CDF97 filter is constructed by splitting the factors of the polynomial

\[ P(t) = \sum_{j=0}^{K-1} \binom{K-1+j}{j} t^j \]

We use \( \tilde{\ell} = \ell = 2 \) so that \( K = \tilde{\ell} + \ell = 4 \) and consider the cubic polynomial

\[ P(t) = \sum_{j=0}^{3} \binom{3+j}{j} t^j = 1 + 4t + 10t^2 + 20t^3 \]

This polynomial has one real root \( r_1 = -0.342484 \) and two complex roots \( c_1 = -0.078808 - .373391i, \ c_2 = -0.078808 + .373391i. \)
Thus we can rewrite $P(t) = 1 + 4t + 10t^2 + 20t^4$ as

$$P(t) = 20(t - r_1)(t - c_1)(t - c_2)$$

We set

$$\tilde{p}(t) = a(t - r_1) \quad \text{and} \quad p(t) = \frac{20}{a}(t - c_1)(t - c_2)$$

To make the Fourier series $\tilde{H}(\omega)$ and $H(\omega)$ satisfy $\tilde{H}(0) = H(0) = \sqrt{2}$, we must have $a = 2.920696$. 
Thus we can rewrite \( P(t) = 1 + 4t + 10t^2 + 20t^4 \) as

\[
P(t) = 20(t - r_1)(t - c_1)(t - c_2)
\]

We set

\[
\tilde{p}(t) = a(t - r_1) \quad \text{and} \quad p(t) = \frac{20}{a}(t - c_1)(t - c_2)
\]

To make the Fourier series \( \tilde{H}(\omega) \) and \( H(\omega) \) satisfy \( \tilde{H}(0) = H(0) = \sqrt{2} \), we must have \( a = 2.920696 \).
Thus we can rewrite $P(t) = 1 + 4t + 10t^2 + 20t^4$ as

$$P(t) = 20(t - r_1)(t - c_1)(t - c_2)$$

We set

$$\tilde{p}(t) = a(t - r_1) \quad \text{and} \quad p(t) = \frac{20}{a}(t - c_1)(t - c_2)$$

To make the Fourier series $\tilde{H}(\omega)$ and $H(\omega)$ satisfy $\tilde{H}(0) = H(0) = \sqrt{2}$, we must have $a = 2.920696$. 
Substituting $t = \sin^2\left(\frac{\omega}{2}\right)$, $a = 2.020696$ and expanding into Fourier series gives

$$\tilde{H}(\omega) = \sum_{k=-3}^{3} \tilde{h}_k e^{ik\omega}$$

where (to 6 digits)

$$\begin{align*}
\tilde{h}_0 &= 0.788486 \\
\tilde{h}_{-1} &= \tilde{h}_1 = 0.418092 \\
\tilde{h}_{-2} &= \tilde{h}_2 = -0.040689 \\
\tilde{h}_{-3} &= \tilde{h}_3 = -0.064539
\end{align*}$$
Substituting \( t = \sin^2\left(\frac{\omega}{2}\right) \), \( a = 2.020696 \) and expanding into Fourier series gives

\[
H(\omega) = \sum_{k=-4}^{4} h_k e^{ik\omega}
\]

where (to 6 digits)

\[
\begin{align*}
h_0 &= 0.852699 \\
h_{-1} &= h_1 = 0.377403 \\
h_{-2} &= h_2 = -0.110624 \\
h_{-3} &= h_3 = -0.023850 \\
h_{-4} &= h_4 = 0.037829
\end{align*}
\]
The plots of $|\tilde{H}(\omega)|$ and $|H(\omega)|$ look more like those representing lowpass filters.

$|\tilde{H}(\omega)|$ for the seven-term filter with $\tilde{H}(j)(\pi) = 0$, for $j = 0, 1, 2, 3$. 
The plots of $|\tilde{H}(\omega)|$ and $|H(\omega)|$ look more like those representing lowpass filters:

$|H(\omega)|$ for the nine-term filter with $H^{(j)}(\pi) = 0$, for $j = 0, 1, 2, 3$. 
For lossy compression, a quantization function is applied to each portion of the biorthogonal wavelet transformation.

The quantization function is

\[ q(t) = \text{sgn}(t) \lfloor |t|/d \rfloor \]

The value \( d \) is called the quantization step size and this value changes depending on (1) the iteration and (2) the portion of the transform being quantized.

Other factors that influence the choice of \( d \):
- The number of bits needed to represent the values of the original image.
- The number of bits needed to represent the exponent and mantissa of values in the blur portion of the transform.
- The analysis gain bits for each portion of the transformation.
For lossy compression, a quantization function is applied to each portion of the biorthogonal wavelet transformation.

The quantization function is

\[ q(t) = \text{sgn}(t) \lfloor |t|/d \rfloor \]

The value \( d \) is called the quantization step size and this value changes depending on (1) the iteration and (2) the portion of the transform being quantized.

Other factors that influence the choice of \( d \):

- The number of bits needed to represent the values of the original image.
- The number of bits needed to represent the exponent and mantissa of values in the blur portion of the transform.
- The analysis gain bits for each portion of the transformation.
For lossy compression, a quantization function is applied to each portion of the biorthogonal wavelet transformation.

The quantization function is

\[ q(t) = \text{sgn}(t) \lfloor |t|/d \rfloor \]

The value \( d \) is called the quantization step size and this value changes depending on (1) the iteration and (2) the portion of the transform being quantized.

Other factors that influence the choice of \( d \):

- The number of bits needed to represent the values of the original image.
- The number of bits needed to represent the exponent and mantissa of values in the blur portion of the transform.
- The analysis gain bits for each portion of the transformation.
For lossy compression, a quantization function is applied to each portion of the biorthogonal wavelet transformation.

The quantization function is

\[ q(t) = \text{sgn}(t) \left\lfloor \frac{|t|}{d} \right\rfloor \]

The value \( d \) is called the quantization step size and this value changes depending on (1) the iteration and (2) the portion of the transform being quantized.

Other factors that influence the choice of \( d \):

- The number of bits needed to represent the values of the original image.
- The number of bits needed to represent the exponent and mantissa of values in the blur portion of the transform.
- The analysis gain bits for each portion of the transformation.
For lossy compression, a quantization function is applied to each portion of the biorthogonal wavelet transformation.

The quantization function is

\[ q(t) = \text{sgn}(t) \lfloor |t|/d \rfloor \]

The value \( d \) is called the **quantization step size** and this value changes depending on (1) the iteration and (2) the portion of the transform being quantized.

Other factors that influence the choice of \( d \):

- The number of bits needed to represent the values of the original image.
- The number of bits needed to represent the exponent and mantissa of values in the blur portion of the transform.
- The analysis gain bits for each portion of the transformation.
For lossy compression, a quantization function is applied to each portion of the biorthogonal wavelet transformation.

The quantization function is

\[ q(t) = \text{sgn}(t) \left\lfloor \frac{|t|}{d} \right\rfloor \]

The value \( d \) is called the **quantization step size** and this value changes depending on (1) the iteration and (2) the portion of the transform being quantized.

Other factors that influence the choice of \( d \):
- The number of bits needed to represent the values of the original image.
- The number of bits needed to represent the exponent and mantissa of values in the blur portion of the transform.
- The analysis gain bits for each portion of the transformation.
For lossy compression, a quantization function is applied to each portion of the biorthogonal wavelet transformation.

The quantization function is

\[ q(t) = \text{sgn}(t) \lfloor |t|/d \rfloor \]

The value \( d \) is called the **quantization step size** and this value changes depending on (1) the iteration and (2) the portion of the transform being quantized.

Other factors that influence the choice of \( d \):

- The number of bits needed to represent the values of the original image.
- The number of bits needed to represent the exponent and mantissa of values in the blur portion of the transform.
- The **analysis gain bits** for each portion of the transformation.
We return to the $200 \times 200$ image

and perform lossy compression using JPEG2000.
We perform 3 iterations of the biorthogonal wavelet transformation using the CDF97 filter and exploiting symmetry:
Here is the modified biorthogonal wavelet transformation after the quantization step:
We then add 127 to each element of each block, and then apply EBCOT encoding to the result.

The compressed image can be represented using 24576 bits or 0.6144 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the original image and better than a 50% improvement over JPEG.

Recall that JPEG needed 57231 bits or 1.43078 bits per pixel.

Let’s look at the compressed images:
- We then add 127 to each element of each block, and then apply EBCOT encoding to the result.
- The compressed image can be represented using 24576 bits or 0.6144 bits per pixel.
- That is quite a substantial savings over the original 320000 bits needed to represent the original image and better than a 50% improvement over JPEG.
- Recall that JPEG needed 57231 bits or 1.43078 bits per pixel.
- Let’s look at the compressed images:
We then add 127 to each element of each block, and then apply EBCOT encoding to the result.

The compressed image can be represented using 24576 bits or 0.6144 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the original image and better than a 50% improvement over JPEG.

Recall that JPEG needed 57231 bits or 1.43078 bits per pixel.

Let’s look at the compressed images:
We then add 127 to each element of each block, and then apply EBCOT encoding to the result.

The compressed image can be represented using 24576 bits or 0.6144 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the original image and better than a 50% improvement over JPEG.

Recall that JPEG needed 57231 bits or 1.43078 bits per pixel.

Let’s look at the compressed images:
We then add 127 to each element of each block, and then apply EBCOT encoding to the result.

The compressed image can be represented using 24576 bits or 0.6144 bits per pixel.

That is quite a substantial savings over the original 320000 bits needed to represent the original image and better than a 50% improvement over JPEG.

Recall that JPEG needed 57231 bits or 1.43078 bits per pixel.

Let's look at the compressed images:
Original:
JPEG:

PSNR = 34.68773088.
JPEG2000:

PSNR = 39.54711669.
Corner enlargement - original:
Corner enlargement - JPEG:
Corner enlargement - original:
Corner enlargement - JPEG2000:
Today’s Schedule

9:00-10:15  Lecture Nine: The Lifting Method
10:15-10:30  Coffee Break (OSS 237)
10:30-11:45  Lecture Ten: JPEG and JPEG2000
11:45-12:00  ⇒ Evaluations/Wrap Up
12:00-1:00  Lunch (Cafeteria)