Sampling and Quantization
Last Time: Sampling

Sampling allows the use of modern digital electronics to process, record, transmit, store, and retrieve CT signals.

- audio: MP3, CD, cell phone
- pictures: digital camera, printer
- video: DVD
- everything on the web
**Last Time: Sampling Theory**

**Sampling**

\[ x(t) \rightarrow x[n] = x(nT) \]

**Impulse Reconstruction**

\[ x[n] \xrightarrow{\text{Impulse Reconstruction}} x_p(t) = \sum x[n] \delta(t - nT) \]

**Bandlimited Reconstruction**

\[ (\omega_s = \frac{2\pi}{T}) \]

\[ x[n] \xrightarrow{\text{Impulse Reconstruction}} x_p(t) = \sum x[n] \delta(t - nT) \]

**Sampling Theorem:** If \( X(j\omega) = 0 \ \forall \ |\omega| > \frac{\omega_s}{2} \) then \( x_r(t) = x(t) \).
Aliasing

Frequencies outside the range $\frac{-\omega_s}{2} < \omega < \frac{\omega_s}{2}$ alias.

$$X_p(j\omega) = \frac{1}{2\pi} (X(j\cdot) \ast P(j\cdot))(\omega)$$
Anti-Aliasing Filter

To avoid aliasing, remove frequency components that alias before sampling.
Anti-Aliasing

Remove frequencies outside the range \(-\frac{\omega_s}{2} < \omega < \frac{\omega_s}{2}\) before sampling to avoid aliasing.

\[
X_p(j\omega) = \frac{1}{2\pi} \left( X(j\cdot) \ast P(j\cdot) \right)(\omega)
\]
Today

Digital recording, transmission, storage, and retrieval requires discrete representations of both time (e.g., sampling) and amplitude.

- audio: MP3, CD, cell phone
- pictures: digital camera, printer
- video: DVD
- everything on the web

Quantization: discrete representations for amplitudes
Quantization

We measure discrete amplitudes in bits.

![Graphs showing quantization levels for 2, 3, and 4 bits with corresponding input voltage and output voltage plots.]

Bit rate = (# bits/sample) × (# samples/sec)
Check Yourself

We hear sounds that range in amplitude from 1,000,000 to 1.

How many bits are needed to represent this range?

1. 5 bits
2. 10 bits
3. 20 bits
4. 30 bits
5. 40 bits
Check Yourself

How many bits are needed to represent 1,000,000:1?

<table>
<thead>
<tr>
<th>bits</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
</tr>
<tr>
<td>10</td>
<td>1,024</td>
</tr>
<tr>
<td>11</td>
<td>2,048</td>
</tr>
<tr>
<td>12</td>
<td>4,096</td>
</tr>
<tr>
<td>13</td>
<td>8,192</td>
</tr>
<tr>
<td>14</td>
<td>16,384</td>
</tr>
<tr>
<td>15</td>
<td>32,768</td>
</tr>
<tr>
<td>16</td>
<td>65,536</td>
</tr>
<tr>
<td>17</td>
<td>131,072</td>
</tr>
<tr>
<td>18</td>
<td>262,144</td>
</tr>
<tr>
<td>19</td>
<td>524,288</td>
</tr>
<tr>
<td>20</td>
<td>1,048,576</td>
</tr>
</tbody>
</table>
We hear sounds that range in amplitude from 1,000,000 to 1.

How many bits are needed to represent this range? 3

1. 5 bits
2. 10 bits
3. 20 bits
4. 30 bits
5. 40 bits
Quantization Demonstration

Quantizing Music

• 16 bits/sample
• 8 bits/sample
• 6 bits/sample
• 4 bits/sample
• 3 bits/sample
• 2 bit/sample

J.S. Bach, Sonata No. 1 in G minor Mvmt. IV. Presto
Nathan Milstein, violin
Quantization

We measure discrete amplitudes in bits.

Example: audio CD

$$2 \text{ channels} \times 16 \frac{\text{bits}}{\text{sample}} \times 44,100 \frac{\text{samples}}{\text{sec}} \times 60 \frac{\text{sec}}{\text{min}} \times 74 \text{ min} \approx 6.3 \text{ G bits}$$

$$\approx 0.78 \text{ G bytes}$$
Quantizing Images

Converting an image from a continuous representation to a discrete representation involves the same sort of issues.

This image has $280 \times 280$ pixels, with brightness quantized to 8 bits.
Quantizing Images

8 bit image

7 bit image
Quantizing Images

8 bit image

6 bit image
Quantizing Images

8 bit image

5 bit image
Quantizing Images

8 bit image

4 bit image
Quantizing Images

8 bit image

3 bit image
Quantizing Images

8 bit image

2 bit image
Quantizing Images

8 bit image

1 bit image
Check Yourself

What is the most objectionable artifact of coarse quantization?

8 bit image  
4 bit image
Dithering

One very annoying artifact is banding caused by clustering of pixels that quantize to the same level.

Banding can be reduced by dithering.

Dithering: adding a small amount ($\pm \frac{1}{2}$ quantum) of random noise to the image before quantizing.

Since the noise is different for each pixel in the band, the noise causes some of the pixels to quantize to a higher value and some to a lower. But the average value of the brightness is preserved.
Quantizing Images with Dither

7 bit image

7 bits with dither
Quantizing Images with Dither

6 bit image

6 bits with dither
Quantizing Images with Dither

5 bit image

5 bits with dither
Quantizing Images with Dither

4 bit image

4 bits with dither
Quantizing Images with Dither

3 bit image

3 bits with dither
Quantizing Images with Dither

2 bit image

2 bits with dither
Quantizing Images with Dither

1 bit image

1 bit with dither
Check Yourself

What is the most objectionable artifact of dithering?

3 bit image

3 bit dithered image
Check Yourself

What is the most objectionable artifact of dithering?

One annoying feature of dithering is that it adds noise.
Quantization Schemes

Example: slowly changing backgrounds.

Quantization: \( y = Q(x) \)

Quantization with dither: \( y = Q(x + n) \)
Check Yourself

What is the most objectionable artifact of dithering?

One annoying feature of dithering is that it adds noise.

Robert’s technique: add a small amount \((\pm \frac{1}{2} \text{ quantum})\) of random noise before quantizing, then subtract that same amount of random noise.
Quantization Schemes

Example: slowly changing backgrounds.

Quantization: \( y = Q(x) \)

Quantization with dither: \( y = Q(x + n) \)

Quantization with Robert’s technique: \( y = Q(x + n) - n \)
Quantizing Images with Robert’s Method

7 bits with dither

7 bits with Robert’s method
Quantizing Images with Robert’s Method

6 bits with dither

6 bits with Robert’s method
Quantizing Images with Robert’s Method

5 bits with dither

5 bits with Robert’s method
Quantizing Images with Robert’s Method

4 bits with dither

4 bits with Robert’s method
Quantizing Images with Robert’s Method

3 bits with dither

3 bits with Robert’s method
Quantizing Images with Robert’s Method

2 bits with dither

2 bits with Robert’s method
Quantizing Images with Robert’s Method

1 bits with dither

1 bit with Robert’s method
Quantizing Images: 3 bits

8 bits

3 bits

dither

Robert’s
Quantizing Images: 2 bits

8 bits

2 bits

dither

Robert’s
Quantizing Images: 1 bit

8 bits 1 bit
dither Robert’s
Progressive Refinement

Trading precision for speed.

Start by sending a crude representation, then progressively update with increasing higher fidelity versions.
Discrete-Time Sampling (Resampling)

DT sampling is much like CT sampling.

\[ x[n] \rightarrow \times \rightarrow x_p[n] \]

\[ p[n] = \sum_k \delta[n - kN] \]

\[ x[n] \]

\[ p[n] \]

\[ x_p[n] \]
Discrete-Time Sampling

As in CT, sampling introduces additional copies of $X(e^{j\Omega})$.

$$x[n] \times x_p[n]$$

$$p[n] = \sum_k \delta[n - kN]$$

$X(e^{j\Omega})$

$P(e^{j\Omega})$

$X_p(e^{j\Omega})$

$\Omega$
Discrete-Time Sampling

Sampling a finite sequence gives rise to a shorter sequence.

\[
X_b(e^{j\Omega}) = \sum_n x_b[n]e^{-j\Omega n} = \sum_n x_p[3n]e^{-j\Omega n} = \sum_k x_p[k]e^{-j\Omega k/3} = X_p(e^{j\Omega/3})
\]
Discrete-Time Sampling

But the shorter sequence has a wider frequency representation.

\[ X(e^{j\Omega}) \]

\[ X_p(e^{j\Omega}) \]

\[ X_b(e^{j\Omega}) = X_p(e^{j\Omega}/3) \]
Discrete-Time Sampling
Discrete-Time Sampling
Discrete-Time Sampling
Discrete-Time Sampling
Discrete-Time Sampling

High Pass

Down-Sample

Low Pass

High Pass

DS

Low Pass
Discrete-Time Sampling
Discrete-Time Sampling
Discrete-Time Sampling

Insert zeros between samples to upsample the images.

\[ x[n] \]

\[ 0 \quad n \]

\[ x_p[n] \]

\[ 0 \quad n \]

\[ x_b[n] \]

\[ 0 \quad n \]

\[
X_b(e^{j\Omega}) = \sum_n x_b[n]e^{-j\Omega n} = \sum_n x_p[3n]e^{-j\Omega n} = \sum_k x_p[k]e^{-j\Omega k/3} = X_p(e^{j\Omega/3})
\]
Discrete-Time Sampling

Then filter out the additional copies in frequency.

\[ X(e^{j\Omega}) \]

\[ X_p(e^{j\Omega}) \]

\[ X_b(e^{j\Omega}) = X_p(e^{j\Omega/3}) \]
Discrete-Time Sampling: Progressive Refinement
Discrete-Time Sampling: Progressive Refinement
Discrete-Time Sampling: Progressive Refinement
Discrete-Time Sampling: Progressive Refinement
Discrete-Time Sampling: Progressive Refinement
Discrete-Time Sampling
Perceptual Coding

Quantizing in the Fourier domain: JPEG.
Example: JPEG ("Joint Photographic Experts Group") encodes images by a sequence of transformations:

- color encoding
- DCT (discrete cosine transform): a kind of Fourier series
- quantization to achieve perceptual compression (lossy)
- Huffman encoding: lossless information theoretic coding

We will focus on the DCT and quantization of its components.

- the image is broken into $8 \times 8$ pixel blocks
- each block is represented by its $8 \times 8$ DCT coefficients
- each DCT coefficient is quantized, using higher resolutions for coefficients with greater perceptual importance
Discrete cosine transform (DCT) is similar to a Fourier series, but high-frequency artifacts are typically smaller.

Example: imagine coding the following $8 \times 8$ block.

For a two-dimensional transform, take the transforms of all of the rows, assemble those results into an image and then take the transforms of all of the columns of that image.
Periodically extend a row and represent it with a Fourier series.

\[ x[n] = x[n + 8] \]

There are 8 distinct Fourier series coefficients.

\[ a_k = \frac{1}{8} \sum_{n=\langle 8 \rangle} x[n] e^{-jk\Omega_0 n}; \quad \Omega_0 = \frac{2\pi}{8} \]
DCT is based on a different periodic representation, shown below.

\[ y[n] = y[n + 16] \]
Check Yourself

Which signal has greater high frequency content?

\[ x[n] = x[n + 8] \]

\[ y[n] = y[n + 16] \]
Check Yourself

The first signal, $x[n]$, has discontinuous amplitude. The second signal, $y[n]$ is not discontinuous, but has discontinuous slope.

$x[n] = x[n + 8]$

$y[n] = y[n + 16]$

The magnitude of its Fourier series coefficients decreases faster with $k$ for the second than for the first.
Check Yourself

Which signal has greater high frequency content? $x[n]$
Periodic extension of an $8 \times 8$ pixel block can lead to a discontinuous function even when the “block” was taken from a smooth image.

$$x[n] = x[n + 8]$$
Periodic extension of the type done for JPEG generates a continuous function from a smoothly varying image.

\[
y[n] = y[n + 16]
\]
Although periodic in $N = 16$, $y[n]$ can be represented by just 8 distinct DCT coefficients.

$$y[n] = y[n + 16]$$

$$b_k = \sum_{n=0}^{7} y[n] \cos \left( \frac{\pi k}{N} \left( n + \frac{1}{2} \right) \right)$$

This results because $y[n]$ is symmetric about $n = -\frac{1}{2}$, and this symmetry introduces redundancy in the Fourier series representation.

Notice also that the DCT of a real-valued signal is real-valued.
The magnitudes of the higher order DCT coefficients are smaller than those of the Fourier series.
Humans are less sensitive to small deviations in high frequency components of an image than they are to small deviations at low frequencies. Therefore, the DCT coefficients are quantized more coarsely at high frequencies.

Divide coefficient $b[m,n]$ by $q[m,n]$ and round to nearest integer.

$$q[m,n] \quad m \quad \rightarrow$$

<table>
<thead>
<tr>
<th>$m$</th>
<th>16</th>
<th>11</th>
<th>10</th>
<th>16</th>
<th>24</th>
<th>40</th>
<th>51</th>
<th>61</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>26</td>
<td>58</td>
<td>60</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>13</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>57</td>
<td>69</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>17</td>
<td>22</td>
<td>29</td>
<td>51</td>
<td>87</td>
<td>80</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>22</td>
<td>37</td>
<td>56</td>
<td>68</td>
<td>109</td>
<td>103</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>35</td>
<td>55</td>
<td>64</td>
<td>81</td>
<td>104</td>
<td>113</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>64</td>
<td>78</td>
<td>87</td>
<td>103</td>
<td>121</td>
<td>120</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>112</td>
<td>100</td>
<td>103</td>
<td>99</td>
</tr>
</tbody>
</table>
Check Yourself

Which of the following tables of $q[m, n]$ (top or bottom) will result in higher “quality” images?

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

$$
\begin{array}{cccccccc}
q[m, n] & 32 & 22 & 20 & 32 & 48 & 80 & 102 & 122 \\
    & 24 & 24 & 28 & 38 & 52 & 116 & 120 & 110 \\
    & 28 & 26 & 32 & 48 & 80 & 114 & 139 & 112 \\
    & 28 & 34 & 44 & 58 & 102 & 174 & 160 & 124 \\
n & 36 & 44 & 74 & 112 & 136 & 218 & 206 & 154 \\
\downarrow & 48 & 70 & 110 & 128 & 162 & 208 & 226 & 194 \\
    & 98 & 128 & 156 & 174 & 206 & 256 & 240 & 202 \\
    & 144 & 184 & 190 & 196 & 224 & 200 & 206 & 198 \\
\end{array}
$$
Check Yourself

Which of the following tables of $q[m, n]$ (top or bottom) will result in higher “quality” images?  **top**

<table>
<thead>
<tr>
<th>$q[m, n]$</th>
<th>$m$</th>
<th>$n$</th>
<th>$q[m, n]$</th>
<th>$m$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16</td>
<td>11</td>
<td>10</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>13</td>
<td>16</td>
<td>24</td>
<td>40</td>
</tr>
<tr>
<td>↓</td>
<td>14</td>
<td>17</td>
<td>22</td>
<td>29</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>22</td>
<td>37</td>
<td>56</td>
<td>68</td>
</tr>
<tr>
<td>↓</td>
<td>24</td>
<td>35</td>
<td>55</td>
<td>64</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>64</td>
<td>78</td>
<td>87</td>
<td>103</td>
</tr>
<tr>
<td>↓</td>
<td>72</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>112</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$q[m, n]$</th>
<th>$m$</th>
<th>$n$</th>
<th>$q[m, n]$</th>
<th>$m$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
<td>22</td>
<td>20</td>
<td>32</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>24</td>
<td>28</td>
<td>38</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>26</td>
<td>32</td>
<td>48</td>
<td>80</td>
</tr>
<tr>
<td>↓</td>
<td>28</td>
<td>34</td>
<td>44</td>
<td>58</td>
<td>102</td>
</tr>
<tr>
<td>↓</td>
<td>36</td>
<td>44</td>
<td>74</td>
<td>112</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>70</td>
<td>110</td>
<td>128</td>
<td>162</td>
</tr>
<tr>
<td>↓</td>
<td>98</td>
<td>128</td>
<td>156</td>
<td>174</td>
<td>206</td>
</tr>
<tr>
<td>↓</td>
<td>144</td>
<td>184</td>
<td>190</td>
<td>196</td>
<td>224</td>
</tr>
</tbody>
</table>
Finally, encode the DCT coefficients for each block using “run-length” encoding followed by an information theoretic (lossless) “Huffman” scheme, in which frequently occurring patterns are represented by short codes.

The “quality” of the image can be adjusted by changing the values of $q[m,n]$. Large values of $q[m,n]$ result in large “runs” of zeros, which compress well.
JPEG: Results

1%: 1666 bytes
10%: 2550 bytes
20%: 3595 bytes

40%: 5318 bytes
80%: 10994 bytes
100%: 47k bytes
6.003 Signals and Systems
Fall 2011

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.