Image compression methods and standards

by Gleb V. Tcheslavski: gleb@ee.lamar.edu
http://ee.lamar.edu/gleb/dip/index.htm

Spring 2008 ELEN 4304/5365 DIP 1

JPEG

JPEG – while being one of the most popular continuous tone image compression standards – defines three basic coding schemes:

1) A lossy baseline coding system based on DCT;
2) An extended coding system for greater compression, higher precision, or progressive reconstruction applications;
3) A lossless independent coding system for reversible compression.

In a baseline format, the image is subdivided into 8x8 pixel blocks, which are processed left to right, top to bottom. For each block, its 64 pixels are level-shifted by subtracting $2^{k-1}$, where $2^k$ is the maximum number of intensity levels. Next, a 2D DCT of the block is computed, quantized, and reordered using the zigzag pattern to form a 1D sequence of quantized coefficients. Next, the nonzero AC coefficients are coded using a variable-length code. The DC coefficient is difference coded relative to the DC coefficient of the previous block.
The JPEG recommended luminance quantization array can be scaled to provide a variety of compression levels (select the quality of JPEG compression).

Consider compression and reconstruction of the following 8x8 subimage:

<table>
<thead>
<tr>
<th>52</th>
<th>55</th>
<th>61</th>
<th>66</th>
<th>70</th>
<th>61</th>
<th>64</th>
<th>73</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>59</td>
<td>66</td>
<td>90</td>
<td>85</td>
<td>69</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>59</td>
<td>68</td>
<td>113</td>
<td>144</td>
<td>104</td>
<td>66</td>
<td>73</td>
</tr>
<tr>
<td>63</td>
<td>58</td>
<td>71</td>
<td>122</td>
<td>154</td>
<td>106</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td>67</td>
<td>61</td>
<td>68</td>
<td>104</td>
<td>126</td>
<td>88</td>
<td>68</td>
<td>70</td>
</tr>
<tr>
<td>79</td>
<td>65</td>
<td>60</td>
<td>70</td>
<td>77</td>
<td>63</td>
<td>58</td>
<td>75</td>
</tr>
<tr>
<td>85</td>
<td>71</td>
<td>64</td>
<td>59</td>
<td>55</td>
<td>61</td>
<td>65</td>
<td>83</td>
</tr>
<tr>
<td>87</td>
<td>79</td>
<td>68</td>
<td>65</td>
<td>76</td>
<td>78</td>
<td>94</td>
<td></td>
</tr>
</tbody>
</table>

The original 256 = 2^8 levels image

Scale by \(2^7 = 128\)

<table>
<thead>
<tr>
<th>52</th>
<th>55</th>
<th>61</th>
<th>66</th>
<th>70</th>
<th>61</th>
<th>64</th>
<th>73</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>59</td>
<td>66</td>
<td>90</td>
<td>85</td>
<td>69</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>59</td>
<td>68</td>
<td>113</td>
<td>144</td>
<td>104</td>
<td>66</td>
<td>73</td>
</tr>
<tr>
<td>63</td>
<td>58</td>
<td>71</td>
<td>122</td>
<td>154</td>
<td>106</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td>67</td>
<td>61</td>
<td>68</td>
<td>104</td>
<td>126</td>
<td>88</td>
<td>68</td>
<td>70</td>
</tr>
<tr>
<td>79</td>
<td>65</td>
<td>60</td>
<td>70</td>
<td>77</td>
<td>63</td>
<td>58</td>
<td>75</td>
</tr>
<tr>
<td>85</td>
<td>71</td>
<td>64</td>
<td>59</td>
<td>55</td>
<td>61</td>
<td>65</td>
<td>83</td>
</tr>
<tr>
<td>87</td>
<td>79</td>
<td>68</td>
<td>65</td>
<td>76</td>
<td>78</td>
<td>94</td>
<td></td>
</tr>
</tbody>
</table>

DCT of the scaled image

### Quantized transformed array

\([-26 -3 -3 -2 -6 2 -4 1 -1 5 0 2 0 0 \text{EOB}]\)

Where EOB is a special end-of-block symbol.

Next, the difference between the current block’s DC symbol and the DC symbol from the previous block is computed and coded. The nonzero AC coefficients are coded according to another code table.
JPEG decompression begins from decoding DC and AC coefficients and recovering an array of quantized coefficients from a 1D zigzag.

De-normalized DCT coefficients

| −70 | −64 | −61 | −64 | −69 | −66 | −58 | −50 | −70 | −64 | −61 | −64 | −69 | −66 | −58 | −50 |
| −72 | −73 | −61 | −39 | −30 | −40 | −54 | −59 | −72 | −73 | −61 | −39 | −30 | −40 | −54 | −59 |
| −68 | −78 | −9 | 13 | −12 | −48 | −64 | −68 | −78 | −9 | 13 | −12 | −48 | −64 |
| −59 | −77 | −57 | 0 | 22 | −13 | −51 | −60 | −59 | −77 | −57 | 0 | 22 | −13 | −51 | −60 |
| −54 | −75 | −64 | −23 | −13 | −44 | −63 | −56 | −54 | −75 | −64 | −23 | −13 | −44 | −63 | −56 |
| −52 | −71 | −72 | −54 | −54 | −71 | −71 | −54 | −52 | −71 | −72 | −54 | −54 | −71 | −71 | −54 |
| −45 | −59 | −70 | −68 | −67 | −67 | −61 | −50 | −45 | −59 | −70 | −68 | −67 | −67 | −61 | −50 |
| −35 | −47 | −61 | −66 | −60 | −48 | −44 | −44 | −35 | −47 | −61 | −66 | −60 | −48 | −44 | −44 |

IDCT array

Upscaled IDCT – reconstructed image

The error between the original and reconstructed images is due to the lossy nature of the JPEG compression. The rms error is approximately 5.8 intensity levels.
Predictive coding

Predictive coding is based on eliminating the redundancies of closely spaced pixels – in space and/or in time – by extracting and coding only the *new information* in each pixel. The new information is defined as the difference between the actual and predicted value of the pixel.

**Lossless predictive coding**

Lossless predictive coding system

Predictor generates the expected value of each sample based on a specified number of past samples.
Predictive coding

Predictor’s output is rounded to the nearest integer and is used to compute prediction error

\[ e(n) = f(n) - \hat{f}(n) \]

Prediction error is encoded by a variable-length code to generate the next element of the encoded data stream. The decoder reconstructs \( e(n) \) from the encoded data and performs the inverse operation

\[ f(n) = e(n) + \hat{f}(n) \]

Various local, global, and adaptive methods can be used to generate \( \hat{f}(n) \).

Often, the prediction is a linear combination of \( m \) previous samples:

\[ \hat{f}(n) = \text{round} \left[ \sum_{i=1}^{m} \alpha_i f(n - i) \right] \]

Predictive coding

Where \( m \) is the order of the linear predictor and \( \alpha_i, i = 1, \ldots m \) are prediction coefficients, \( f(n) \) are the input pixels. The \( m \) samples used for prediction can be taken from the current scan line (1D linear predictive coding – LPC), from the current and previous line (2D LPC), or from the current image and the previous images in an image sequence (3D LPC). The 1D LPC:

\[ \hat{f}(n) = \text{round} \left[ \sum_{i=1}^{m} \alpha_i f(x, y - i) \right] \]

which is a function of the previous pixels in the current line. Note that the prediction cannot be formed for the first \( m \) pixels. These pixels are coded by other means (f.e. Huffman code).
Predictive coding

For the image shown, form a first-order \((m = 1)\) LPC in form:

\[
\hat{f}(x, y) = \text{round} \left[ \alpha f(x, y - 1) \right]
\]

A predictor is called a \textit{previous pixel} predictor, and the coding procedure is \textit{differential} (\textit{previous pixel}) coding.

Prediction error image up-scaled by 128.

The average prediction error 0.26

The entropy reduction is due to removal of spatial redundancy.

Predictive coding

The compression achieved in predictive coding is related to the entropy reduction resulting from mapping an input image into a prediction error sequence. Therefore, the pdf of the prediction error is (in general) highly peaked at 0 and has relatively small (compared to the input image) variance. It is often modeled by a zero-mean uncorrelated Laplacian pdf:

\[
p_e(e) = \frac{1}{\sqrt{2\sigma_e}} e^{-\frac{\sqrt{2}|e|}{\sigma_e}}
\]

where \(\sigma_e\) is the standard deviation of \(e\).
Predictive coding

Two successive frames of Earth taken by NASA spacecraft.

Using the first-order \((m = 1)\) LPC:

\[
\hat{f}(x, y, t) = \text{round}\left[\alpha f(x, y, t-1)\right]
\]

with \(\alpha = 1\), the pixel intensities in the second frame can be predicted from the intensities in the first frame; the residual image

Considerable decrease in standard deviation and in the entropy indicates significant compression that can be achieved.

Predictive coding

Motion compensated prediction residuals

Since successive frames in a video sequence are often quite similar, coding their differences can reduce temporal redundancy and provide significant compression. On the other hand, when a frame sequence contains rapidly moving objects, the similarity between neighboring frames is reduced. The attempt to use LPC on images with little temporal redundancy may lead to data expansion. Video compression systems avoid the problem of data expansion by:

1. Tracking object movement and compensating for it during the prediction and differencing process;
2. Switching to an alternative coding method when there is insufficient inter-frame correlation (similarity between frames) to make predictive coding advantageous.
Predictive coding

Basics of motion compensation:

Each video frame is divided into non-overlapping rectangular regions – typically of size 4x4 to 16x16 – macroblocks. The “movement” of each macroblock with respect to the previous frame (reference frame) is encoded in a motion vector that describes the motion by defining the vertical and horizontal displacement from the “most likely” position. This displacement is usually specified to the nearest pixel, ½ pixel, or ¼ pixel precision. If sub-pixel precision is used, prediction must be interpolated from a combination of pixels in the frame.

Predictive coding

An encoded frame that is based on the previous frame (forward prediction) is called a predictive frame (P-frame); the frame that is also based on the subsequent frame (backward prediction) is called a bidirectional frame (B-frame). B-frames require the compressed code stream to be reordered. Finally, some frames are encoded without referencing to any of the neighboring frames (like JPEG) and are encoded independently. Such frames are called intraframes or independent frames (I-frames) and are ideal starting points for the generation of prediction residuals. Also, I-frames can be easily accessed without decoding the stream.
Predictive coding

Motion estimation is the key concept of motion compensation. During it, the motion of objects is measured and encoded into motion vectors. The search for the “best” motion vector requires specification of optimality criterion. For instance, motion vectors may be selected on the basis of maximum correlation or minimum error between macroblock pixels and the predicted (or interpolated) pixels for the chosen reference frame. One of the most frequently used error measures is the mean absolute distortion (MAD):

$$MAD(x, y) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} |f(x+i, y+j) - p(x+i+dx, y+j+dy)|$$

where \(x\) and \(y\) are the coordinates of the upper-left pixel of the \(mxn\) macroblock being coded, \(dx\) and \(dy\) are displacements from the reference frame, and \(p\) is an array of predicted macroblock pixels.

Typically, \(dx\) and \(dy\) must fall within a limited search region around each macroblock. Values from ±8 to ±64 pixels are common, and the horizontal search area often is significantly larger than the vertical search area. Another, more computationally efficient measure is the sum of absolute distortions (SAD) that omits the \(1/mn\) factor.

For the specified selection criterion (say, MAD), motion estimation is performed by searching for the \(dx\) and \(dy\) minimizing \(MAD(x,y)\) over the allowed range of motion vector displacements – block matching. An exhaustive search is efficient but expensive; there are fast algorithms that are inexpensive but don’t guarantee optimum.
Predictive coding

Two images differing by 13 frames.

Difference image: std dev of error = 12.73; entropy = 4.2

Motion-compens. difference image

Motion vectors: highly correlated – variable-length code

Predictive coding

Motion compensated prediction residual was computed by dividing the latest frame into 16x16 macroblocks and comparing each macroblock to all possible 16x16 macroblock in the earlier frame within ±16 pixels position. The MAD criterion was used. The resulting standard deviation was 5.62 and the entropy was 3.04 bits/pixel.

We observe that there is no motion in the lower portion of the image corresponding to the space shuttle. Therefore, no motion vectors are shown. The macroblocks in this area are predicted from similarly located macroblocks in the reference frame.
**Predictive coding**

Prediction accuracy can be increased using sub-pixel motion compensation.

**Prediction residual with no motion compensation:**
- Stdev = 12.7;
- Entropy = 4.17

**Residual with 1/2 pixel motion compensation:**
- Stdev = 4;
- Entropy = 3.35

**Residual with 1 pixel motion compensation:**
- Stdev = 4.4;
- Entropy = 3.34

**Residual with 1/4 pixel motion compensation:**
- Stdev = 3.8;
- Entropy = 3.34

---

**Predictive coding**

Motion estimation is a computationally intensive process. Fortunately, only the encoder must estimate the macroblock motion. The decoder – for the known motion vectors of the macroblocks – accesses the areas of the reference frames that were used in the encoder to form the prediction residuals.

For this reason, most video compression standards do not include motion estimation. Instead, compression standards focus on the decoder: place constraints on macroblock dimensions, motion vector precision, horizontal and vertical displacement ranges, etc.
Predictive coding

Most of the video compression standards use an 8x8 DCT for I-frame encoding but specify a larger area (16x16 macroblocks) for motion compensation. Additionally, even the P- and B-frame prediction residuals are transform coded due to effectiveness of DCT coefficient quantization. The H.264 and MPEG-4 AVC support intraframe predictive coding (in I-frames) to reduce spatial redundancy…
Predictive coding

A typical motion-compensated video encoder

An encoder exploits redundancies within and between adjacent video frames, and the psychovisual properties of the human visual system.

Predictive coding

Encoder’s input is a sequence of macroblocks. For color video, each macroblock consists of a luminance block and 2 chrominance blocks. The human eye has less spatial acuity for color than for luminance, therefore, the chrominance blocks are often sampled at half the horizontal and vertical resolution the luminance block.

Grayed elements of the encoder are JPEG encoder that may operate on conventional macroblock (I-frames) or their differences (P- and B-blocks). Inverse mapper performs IDCT.
Predictive coding

1 minute HD (1280x720) full-color video containing 150 frames fade-in from black (frames 21, 44), to black (frames 1595, 1609, 1652), abrupt changes (frames 1303, 1304). H.264 compression requires 44.56 MB of storage as compared to about 5 GB uncompressed.

MPEG-2

MPEG-2 Profiles

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Name</th>
<th>Picture Coding Types</th>
<th>Chroma Format</th>
<th>Aspect Ratios</th>
<th>Scalable modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>Simple profile</td>
<td>I, P</td>
<td>4:2:0</td>
<td>square pixels, 4:3, or 16:9</td>
<td>none</td>
</tr>
<tr>
<td>MP</td>
<td>Main profile</td>
<td>I, P, B</td>
<td>4:2:0</td>
<td>square pixels, 4:3, or 16:9</td>
<td>none</td>
</tr>
<tr>
<td>SNR</td>
<td>SNR Scalable profile</td>
<td>I, P, B</td>
<td>4:2:0</td>
<td>square pixels, 4:3, or 16:9</td>
<td>SNR (signal-to-noise ratio) scalable</td>
</tr>
<tr>
<td>Spatial</td>
<td>Spatially Scalable profile</td>
<td>I, P, B</td>
<td>4:2:0</td>
<td>square pixels, 4:3, or 16:9</td>
<td>SNR- or spatial-scalable</td>
</tr>
<tr>
<td>HP</td>
<td>High profile</td>
<td>I, P, B</td>
<td>4:2:2 or 4:2:0</td>
<td>square pixels, 4:3, or 16:9</td>
<td>SNR- or spatial-scalable</td>
</tr>
</tbody>
</table>
# MPEG-2

## MPEG-2 Levels

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Name</th>
<th>Frame rates (Hz)</th>
<th>Max h. res.</th>
<th>Max v. res.</th>
<th>Max luminance samples/s</th>
<th>Max bit rate (Mbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL</td>
<td>Low Level</td>
<td>23.976, 24, 25, 29.97, 30</td>
<td>352</td>
<td>288</td>
<td>3,041,280</td>
<td>4</td>
</tr>
<tr>
<td>ML</td>
<td>Main Level</td>
<td>23.976, 24, 25, 29.97, 30</td>
<td>720</td>
<td>576</td>
<td>10,368,000; Hp: 14,475,600 for 4:2:0 and 11,059,200 for 4:2:2</td>
<td>15</td>
</tr>
<tr>
<td>H-14</td>
<td>High 1440</td>
<td>23.976, 24, 25, 29.97, 30, 50, 59.94, 60</td>
<td>1440</td>
<td>1152</td>
<td>47,001,600 Hp with 4:2:0: 62,668,800</td>
<td>60</td>
</tr>
<tr>
<td>HL</td>
<td>High Level</td>
<td>23.976, 24, 25, 29.97, 30, 50, 59.94, 60</td>
<td>1920</td>
<td>1152</td>
<td>62,668,800 Hp with 4:2:0: 83,558,400</td>
<td>80</td>
</tr>
</tbody>
</table>

## MPEG-2

### Allowed Resolutions
- 720 × 480, 704 × 480, 352 × 480, 352 × 240 pixel (NTSC)
- 720 × 576, 704 × 576, 352 × 576, 352 × 288 pixel (PAL)

### Allowed Aspect ratios (Display AR)
- 4:3
- 16:9
  - (1.85:1 and 2.35:1, among others, are often listed as valid DVD aspect ratios, but are actually just a 16:9 image with the top and bottom of the frame masked in black)

### Allowed Frame rates
- 29.97 frame/s (NTSC)
- 25 frame/s (PAL)

### Note: By using a pattern of REPEAT_FIRST_FIELD flags on the headers of encoded pictures, pictures can be displayed for either two or three fields and almost any picture display rate (minimum ⅔ of the frame rate) can be achieved. This is most often used to display 23.976 (approximately film rate) video on NTSC. Audio+video bitrate
- Video peak 9.8 Mbit/s
- Total peak 10.08 Mbit/s
- Minimum 300 kbit/s
Lossy predictive coding

Lossy predictive coding is achieved by including a quantizer that replaces the error-free output by the nearest integer.

The decompressed sequence
\[ \hat{f}(n) = \hat{e}(n) + \hat{f}(n) \]

is also a predictor's input.

Delta modulation

Delta modulation is a simple form of lossy predictive coding, where the predictor and quantizer are defined as
\[ \hat{f}(n) = \alpha \hat{f}(n-1) \]
\[ \hat{e}(n) = \begin{cases} +\xi & \text{for } e(n) > 0 \\ -\xi & \text{otherwise} \end{cases} \]

where \( \alpha \) is a prediction coefficient (normally < 1) and \( \xi \) is a positive constant. The quantizer output can be represented by a single bit.
Lossy predictive coding

Calculations needed to compress and reconstruct an input sequence \{14 15 14 15 13 15 14 20 26 27 28 27 29 37 62 75 77 78 79 80 81 81 82 82\} with \(\alpha = 1\), \(\xi = 6.5\).

The process begins with the error-free transfer of the first input sample 14 to the decoder. The remaining outputs are computed according to the equations.

<table>
<thead>
<tr>
<th>(\xi)</th>
<th>(f(x))</th>
<th>(\tilde{f}(x))</th>
<th>(\epsilon(x))</th>
<th>(\tilde{\epsilon}(x))</th>
<th>(f(x) - \tilde{f}(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>14</td>
<td>-1</td>
<td>14</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>20</td>
<td>-6</td>
<td>14</td>
<td>-6</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>20</td>
<td>-6</td>
<td>14</td>
<td>-6</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>20</td>
<td>-1</td>
<td>14</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>27</td>
<td>27</td>
<td>0</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>37</td>
<td>37</td>
<td>0</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>52</td>
<td>46</td>
<td>-6</td>
<td>46</td>
<td>-6</td>
</tr>
<tr>
<td>8</td>
<td>75</td>
<td>66</td>
<td>-9</td>
<td>66</td>
<td>-9</td>
</tr>
<tr>
<td>9</td>
<td>97</td>
<td>84</td>
<td>-13</td>
<td>84</td>
<td>-13</td>
</tr>
</tbody>
</table>

Lossy predictive coding

When \(\xi\) is too small to represent the input’s change, a distortion called \textit{slope overload} occurs. On the other hand, if \(\xi\) is too large to represent the input’s smallest change, \textit{granular noise} appears. In images, these distortions show up as blurred edges and grainy (noisy) surfaces (distorted smooth areas).

These distortions are common to all lossy predictive coding forms and arise from a combination of quantization and prediction methods used. Predictors are normally designed with the assumption of no quantization error, and quantizers are designed to minimize their own errors.
Optimal predictors

Usually, the predictor is chosen to minimize the encoder’s mean-square prediction error:

\[
E \{ e^2(n) \} = E \left\{ \left( f(n) - \hat{f}(n) \right)^2 \right\}
\]

with the constrain that

\[
\hat{f}(n) = \dot{e}(n) + \hat{f}(n) \approx e(n) + \hat{f}(n) = f(n)
\]

and

\[
\hat{f}(n) = \sum_{i=1}^{m} \alpha_i f(n-1)
\]

Therefore, the optimization criterion is minimal mean-square prediction error, the quantization error is assumed to be negligible (\(\dot{e}(n) \approx e(n)\)), and the prediction is a linear combination of \(m\) previous samples.

Optimal predictors

The resulting predictive coding approach is called a differential pulse code modulation (DPCM). Therefore, we need to select the \(m\) prediction coefficients that minimize

\[
E \{ e^2(n) \} = E \left\{ \left( f(n) - \sum_{i=1}^{m} \alpha_i f(n-i) \right)^2 \right\}
\]

The minimization can be done by differentiating the above equation with respect to each coefficient, equating the derivatives to zero, and solving them:

\[
\alpha = R^{-1} r
\]
Optimal predictors

where $R^{-1}$ is the inverse of the $m \times m$ autocorrelation matrix

$$R = \begin{bmatrix}
E\{f(n-1)f(n-1)\} & E\{f(n-1)f(n-2)\} & \cdots & E\{f(n-1)f(n-m)\} \\
E\{f(n-2)f(n-1)\} & E\{f(n-2)f(n-2)\} & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
E\{f(n-m)f(n-1)\} & E\{f(n-m)f(n-2)\} & \cdots & E\{f(n-m)f(n-m)\}
\end{bmatrix}$$

And $r$ and $\alpha$ are the $m$-element vectors:

$$r = \begin{bmatrix}
E\{f(n)f(n-1)\} \\
E\{f(n)f(n-2)\} \\
\vdots \\
E\{f(n)f(n-m)\}
\end{bmatrix}, \quad \alpha = \begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_m
\end{bmatrix}$$

The coefficients depend only on the autocorrelations of the samples in the original sequence and can be determined by matrix operations.

Optimal predictors

The variance of the prediction error resulting from using these optimal coefficients would be

$$\sigma_e^2 = \sigma^2 - \alpha^T r = \sigma^2 - \sum_{i=1}^{m} E\{f(n)f(n-i)\} \alpha_i$$

However, computations of autocorrelations is very difficult in practice. Usually, a set of global coefficients is computed assuming a simple input model and substituting the corresponding autocorrelations. For instance, when a 2D Markov image source with the separable autocorrelation function

$$E\{f(x, y)f(x-i, y-j)\} = \sigma^2 \rho_x^i \rho_y^j$$

and a generalized 4th order linear predictor

$$\hat{f}(x, y) = \alpha_1 f(x, y-1) + \alpha_2 f(x-1, y-1) + \alpha_3 f(x-l, y) + \alpha_4 f(x-1, y+1)$$
Optimal predictors

are assumed, the resulting optimal coefficients are

$$\alpha_1 = \rho_h, \quad \alpha_2 = -\rho_h, \quad \alpha_3 = \rho_v, \quad \alpha_4 = 0$$

where $\rho_h$ and $\rho_v$ are horizontal and vertical correlation coefficients of the considered image. The sum of prediction coefficients usually is

$$\sum_{i=1}^{m} \alpha_i \leq 1$$

which ensures that the predictor’s output is within the allowed range and reduces the impact of transmission noise (usually seen as horizontal streaks). Reducing the decoder’s susceptibility to the input noise is important since a single error may propagate to all future outputs! That is, the decoder’s output may become unstable. If the sum is strictly less than one, an input error will affect only a small number of outputs.

Optimal predictors

Considering the prediction error resulting from DPMC coding the monochrome “Lena” image assuming a zero quantization error and using each of the 4 predictors:

$$\hat{f}(x, y) = 0.97 f(x, y - 1)$$

$$\hat{f}(x, y) = 0.5 f(x, y - 1) + 0.5 f(x - 1, y)$$

$$\hat{f}(x, y) = 0.75 f(x, y - 1) + 0.75 f(x - 1, y) - 0.5 f(x - 1, y - 1)$$

$$\hat{f}(x, y) = \begin{cases} 0.97 f(x, y - 1), & \text{if } \Delta h \leq \Delta v \\ 0.97 f(x - 1, y), & \text{otherwise} \end{cases}$$

where

$$\Delta h = |f(x - 1, y) - f(x - 1, y - 1)|$$

$$\Delta v = |f(x, y - 1) - f(x - 1, y - 1)|$$

are the horizontal and vertical gradients at point $(x, y)$. 
Optimal predictors

Observe that the 4th adaptive predictor is designed to improve edge rendition by computing a local measure of the directional properties.

Prediction error images computed for the 4 predictors.

The visually perceptive error decreases as the predictor’s order increases.

The standard deviations are 11.1, 9.8, 9.1, and 9.7 intensity levels.

Optimal quantization

The staircase quantization function \( t = q(s) \) is an odd function of \( s \) (i.e. \( q(-s) = -q(s) \)) and can be described completely by the \( L/2 \) values of \( s_i \) and \( t_i \) shown in the first quadrant of the graph. These break points define function discontinuities and are called the quantizer’s decision and reconstruction levels.

By the convention, \( s \) is mapped to \( t_i \) if it lies in \( (s_{i-1}, s_i] \).

The quantizer design problem is to select the best \( s_i \) and \( t_i \) for a particular optimization criterion and input pdf \( p(s) \).
Optimal quantization

If the optimization criterion (that can be either statistical or psycho-visual measure) is the minimization of the mean-square quantization error $E\{(s_i - t_i)^2\}$ and $p(s)$ is an even function, the conditions for minimal error are

$$
\int_{s_{i-1}}^{s_i} (s-t_i)p(s)ds = \begin{cases} 
0 & i = 0 \\
\frac{t_i + t_{i+1}}{2} & i = 1, 2, \ldots L/2 - 1 \\
\infty & i = L/2 
\end{cases}
$$

Therefore, the reconstruction levels are the centroids of the area under $p(s)$ over the specified decision intervals; the decision levels are halfway between the reconstruction levels; $q$ is an odd function.

The quantizer satisfying the above conditions is optimal in the mean-square error sense and is called the $L$-level Lloyd-Max quantizer.

For a unit variance Laplacian pdf, the 2-, 4-, and 8-level Lloyd-Max decision and reconstruction levels (computed numerically) are:

<table>
<thead>
<tr>
<th>Levels</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$s_i$</td>
<td>$t_i$</td>
<td>$s_i$</td>
</tr>
<tr>
<td>1</td>
<td>$\infty$</td>
<td>0.707</td>
<td>1.102</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$</td>
<td>1.102</td>
<td>1.810</td>
</tr>
<tr>
<td>3</td>
<td>2.285</td>
<td>1.576</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$\infty$</td>
<td>2.994</td>
<td></td>
</tr>
<tr>
<td>$\theta$</td>
<td>1.414</td>
<td>1.087</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>
Optimal quantization

The 3 quantizers shown provide fixed output rates of 1, 2, and 3 bits/pixel. If a non-unit variance pdf is considered, the reconstruction and decision levels are obtained by multiplying the shown values by the standard deviation of the considered pdf. The last row in the table shows the step size satisfying

\[ \theta = t_i - t_{i-1} = s_i - s_{i-1} \]

The Lloyd-Max quantizer is not adaptive. However, adjusting the quantization levels based on the local behavior of an image can be very beneficial. In theory, slowly changing regions can be finely quantized, while the rapidly changing areas are quantized more coarsely. This approach reduces both the granular noise and the slope overload, while requiring a minimal increase in code rate and increased quantizer complexity.

Wavelet compression

The idea of wavelet-based compression is similar to the idea of DCT compression: the transform coefficients can be stored more efficiently than the pixels themselves since a transform decorrelates the information stored in pixels. If the transform basis functions (in this case wavelets) pack most of the visual information in few coefficients, the remaining coefficients can be quantized coarsely or zeroed with little image distortion.
Wavelet compression

To encode a $2^J \times 2^J$ image, an analyzing wavelet $\psi$ and a minimum decomposition level $J-P$ are selected and used to compute DWT of the image. If the wavelet has the complementary scaling function $\phi$, the fast wavelet transform can be used. The transform converts a large portion of the original image to horizontal, vertical, and diagonal coefficients with zero mean and Laplacian-like distribution. Many of these coefficients carry little visual information and can be quantized and coded to reduce inter-coefficient and coding redundancy.

Since the wavelet transform is both computationally efficient and inherently local (since basis functions have a limited duration), image subdivision into block is not needed, which eliminate the blocking artifact and is the major difference compared to the transform coding systems.

Wavelet compression

Wavelet selection

The selection of a wavelet affects the computational complexity of the transforms and the system’s ability to compress and reconstruct images of acceptable error.

When a wavelet has a companion scaling function, transformations can be implemented as a sequence of filtering operations. The ability of the wavelet to pack information into a small number of transform coefficients determines its compression and reconstruction performance.

The most frequently used are Daubechies wavelets and biorthogonal wavelets.
Wavelet compression

Symlets are an extension of Daubechies wavelets with increased symmetry. As seen in the table, computational intensity increases (from 4 to 28 multiplications and additions per coefficient) when moving to more complex wavelets, as well as the information packing performance.

<table>
<thead>
<tr>
<th>Wavelet</th>
<th>Filter Taps (Scaling + Wavelet)</th>
<th>Zeroed Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haar (see Ex. 7.10)</td>
<td>2 + 2</td>
<td>33.8%</td>
</tr>
<tr>
<td>Daubechies (see Fig. 7.8)</td>
<td>8 + 8</td>
<td>40.9%</td>
</tr>
<tr>
<td>Symlet (see Fig. 7.26)</td>
<td>8 + 8</td>
<td>41.2%</td>
</tr>
<tr>
<td>Biorthogonal (see Fig. /59)</td>
<td>11 + 11</td>
<td>42.1%</td>
</tr>
</tbody>
</table>

The coefficients below 1.5 were set to zero. The potential compression ability of biorthogonal wavelet is almost 10% higher than the one of Haar wavelet.
Wavelet compression

Decomposition level selection

Since a $P$-scale fast wavelet transform involves $P$ filter banks, the number of iterations in the computation of the forward and inverse transforms increases with the number of decomposition levels. In many applications (like searching image database or transmitting images for progressive reconstruction), the resolution of the stored or transmitted images and the scale of the lowest useful approximation normally determine the number of transform levels.

For a biorthogonal wavelet and a global threshold 25

<table>
<thead>
<tr>
<th>Decomposition Level (Scales or Filter Bank Iterations)</th>
<th>Approximation Coefficient Image</th>
<th>Truncated Coefficients (%)</th>
<th>Reconstruction Error (rms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$256 \times 256$</td>
<td>74.7%</td>
<td>3.27</td>
</tr>
<tr>
<td>2</td>
<td>$128 \times 128$</td>
<td>91.7%</td>
<td>4.23</td>
</tr>
<tr>
<td>3</td>
<td>$64 \times 64$</td>
<td>95.1%</td>
<td>4.54</td>
</tr>
<tr>
<td>4</td>
<td>$32 \times 32$</td>
<td>95.6%</td>
<td>4.61</td>
</tr>
<tr>
<td>5</td>
<td>$16 \times 16$</td>
<td>95.5%</td>
<td>4.63</td>
</tr>
</tbody>
</table>

Wavelet compression

Quantizer design

The most important factor affecting wavelet coding compression and reconstruction error is coefficient quantization. The most widely used compressors are uniform. However, the effectiveness of the quantization can be improved significantly by

1) Introducing a larger quantization interval around zero (a **dead zone**);

2) Adapting the size of the quantization interval from scale to scale.

In either case, the selected quantization intervals must be transmitted to the decoder with the encoded image bit stream. The intervals themselves may be determined heuristically or automatically computed based on the image being decoded.
Wavelet compression

The impact of dead zone interval size on the percentage of truncated detail coefficients for a 3 scale biorthogonal wavelet-based encoding. As the size of the dead zone increases, the number of truncated coefficients increases too. There is almost no gain beyond 5.

The rms reconstruction error due to the dead zone thresholding increases from 0 to 1.94 (at threshold 5). If every detail coefficient were eliminated, approximately 97.92% of coefficients would be truncated and the reconstruction error would reach 12.3 levels.

Wavelet compression: JPEG-2000

JPEG-2000 is based on the wavelet coding technique and provides an increased flexibility in both the compression of continuous-tone still images and access to the compressed data. Portions of the compressed image can be extracted for retranslation, storage, display, or editing. Coefficient quantization is adapted to individual scales and subbands and the quantized coefficients are arithmetically coded on a bit-plane basis.

The first step of the encoding process is to shift the image intensity (or the three component images in the case of color images) by subtracting $2^{S\text{size}-1}$. If there are exactly three components, they may be optionally decorrelated using a reversible or nonreversible linear combination of the components.
Wavelet compression : JPEG-2000

For instance, the irreversible component transform of JPEG-200 is:

\[
\begin{align*}
Y_0(x, y) &= 0.299I_0(x, y) + 0.587I_1(x, y) + 0.114I_2(x, y) \\
Y_1(x, y) &= -0.16875I_0(x, y) - 0.33126I_1(x, y) + 0.5I_2(x, y) \\
Y_2(x, y) &= 0.5I_0(x, y) - 0.41869I_1(x, y) - 0.08131I_2(x, y)
\end{align*}
\]

where \(I_0, I_1,\) and \(I_2\) are the level-shifted input components and \(Y_0, Y_1,\) and \(Y_2\) are the corresponding decorrelated components. If the input components are the red, green, and blue planes, the equation approximates the \(R'G'B'\) to \(Y'CbCr\) color video. The coal of the transformation is to improve compression efficiency since the transformed components \(Y_1\) and \(Y_2\) are difference images whose histograms are highly peaked around zero.

Wavelet compression : JPEG-2000

After the image has been level shifted and optionally decorrelated, its components can be divided into tiles – rectangular arrays of pixels that are processed independently providing a simple mechanism to access and/or manipulate a limited region of a coded image.

For instance, an image with a 16:9 aspect ratio can be subdivided into tiles such that one of its tiles is a subimage with a 4:3 aspect ratio. That tile could be reconstructed without accessing the other tiles in the compressed image.

If the image is not subdivided into tiles, it is called a single tile.
Wavelet compression : JPEG-2000

Next, the 1D DWT of the rows and columns of each tile component is computed. For error-free compression, the transform uses a biorthogonal 5-3 coefficient scaling and wavelet vector. A rounding procedure is used for the transform coefficients having non-integer values.

In lossy applications, a 9-7 coefficient scaling-wavelet vector is used. In either case, the transform is computed using either the fast wavelet transform or a complementary lifting-based approach.

<table>
<thead>
<tr>
<th>Filter Tap</th>
<th>Highpass Wavelet Coefficient</th>
<th>Lowpass Scaling Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.115087052456994</td>
<td>0.6029490182363579</td>
</tr>
<tr>
<td>±1</td>
<td>0.5912717631142470</td>
<td>0.2688461144428723</td>
</tr>
<tr>
<td>±2</td>
<td>0.05754352622849957</td>
<td>-0.0782326652898785</td>
</tr>
<tr>
<td>±3</td>
<td>-0.09127176311424948</td>
<td>-0.0168641144428749</td>
</tr>
<tr>
<td>±4</td>
<td>0</td>
<td>0.02674875741080976</td>
</tr>
</tbody>
</table>

Here $X$ is the component being transformed, $Y$ is the resulting transform, $i_0$ and $i_1$ define the position of the tile component.

$$\alpha = -1.586134342; \quad \beta = -0.052980118; \quad \gamma = 0.882911075; \quad \delta = 0.433506852$$

$$K = 1.230174105$$
Wavelet compression: JPEG-2000

The described transformation produces 4 subbands: a low-resolution approximation of the tile component and the component’s horizontal, vertical, and diagonal detail images. Repeating the transformation $N_L$ times produces an $N_L$-scale wavelet transform. Adjacent scales are related spatially by power of 2.

JPEG-2000 two-scale ($N_L = 2$) wavelet transform tile-component notation.

When each of the tile components has been processed, the total number of transform coefficients is equal to the number of samples (pixels) in the original image. However, the important visual information is concentrated in a few coefficients.

Wavelet compression: JPEG-2000

To reduce the number of bits needed to store the information, coefficient $a_b(u,v)$ of subband $b$ is quantized to the value $q_b(u,v)$ by

$$q_b(u,v) = \text{sign}[a_b(u,v)] \cdot \text{floor} \left( \frac{|a_b(u,v)|}{\Delta_b} \right)$$

where the quantization step size is

$$\Delta_b = 2^{R_b - \varepsilon_b} \left( 1 + \frac{\mu_b}{2^{\varepsilon_b}} \right)$$

$R_b$ is the nominal dynamic range of subband $b$ (the sum of the number of bits used to represent the original image and the analysis gain bits for subband $b$), and $\varepsilon_b$ and $\mu_b$ are the number of bits allotted to the exponent and mantissa of the subband’s coefficients.
Wavelet compression : JPEG-2000

For error-free compression, $\mu_b = 0$, $R_b = \varepsilon_b$, and $\Delta_b = 1$. For irreversible compression, no particular quantization step size is specified in the standard. Instead, the number of exponent and mantissa bits must be provided to the decoder on a subband basis, called a subband quantization, or for the $N_{L,LL}$ subband only, called derived quantization. In the latter case, the remaining subbands are quantized using extrapolated $N_{L,LL}$ subband parameters. If $\varepsilon_0$ and $\mu_0$ are the number of bits allocated to the $N_{L,LL}$ subband, the extrapolated parameters for subband $b$ are

$$\mu_b = \mu_0$$

$$\varepsilon_b = \varepsilon_0 + n_b - N_L$$

where $n_b$ is the number of subband decomposition levels from the original image tile component to subband $b$.

Wavelet compression : JPEG-2000

In the final steps of the encoding, the coefficients of each transformed tile-component’s subbands are arranged into rectangular code blocks, which are coded individually, one bit at a time. Starting from the most significant bit plane with a nonzero element, each bit plane is processed in three passes. Each bit is coded in only one of the three passes that are called significance propagation, magnitude refinement, and cleanup. The outputs are then arithmetically coded and grouped with similar passes from other code blocks to form layers – arbitrary numbers of groupings of coding passes from each code block. The resulting layers are finally partitioned into packets that are fundamental units of the encoded code stream providing an additional method of extracting a spatial region of interest from the code stream.
Wavelet compression : JPEG-2000

JPEG-2000 decoders invert the previously described operations. After reconstructing the subbands of the tile-components from the arithmetically coded packets, a user-selected number of subbands is decoded. Any nondecoded bits are set to zero and the resulting coefficients $\bar{q}_b(u,v)$ are inverse quantized using the inverse-quantized transform coefficient

$$R_{q_b}(u,v) = \begin{cases} 
\left(\bar{q}_b(u,v) + r \cdot 2^{M_b-N_b(u,v)}\right) \cdot \Delta_b & \bar{q}_b(u,v) > 0 \\
\left(\bar{q}_b(u,v) - r \cdot 2^{M_b-N_b(u,v)}\right) \cdot \Delta_b & \bar{q}_b(u,v) < 0 \\
0 & \bar{q}_b(u,v) = 0
\end{cases}$$

where $M_b$ is the number of bit planes for a particular subband and $N_b$ is the number of decoded bit planes. The reconstruction parameter $r$ is chosen by the decoder to produce the best visual or objective quality of reconstruction: $0 \leq r \leq 1$ (commonly $r = \frac{1}{2}$).

Wavelet compression : JPEG-2000

The inverse quantized coefficients are then inverse transformed by column and by row using IFWT filter bank whose coefficients are obtained from the table or via the lifting-based operations:

$$X(2n) = K \cdot Y(2n) \quad i_0 - 3 \leq 2n < i_i + 3$$
$$X(2n+1) = (-1/K) \cdot Y(2n+1) \quad i_0 - 2 \leq 2n - 1 < i_i + 2$$
$$X(2n) = X(2n) - \delta [X(2n-1) + X(2n+1)] \quad i_0 - 3 \leq 2n < i_i + 3$$
$$X(2n+1) = X(2n+1) - \gamma [X(2n) + X(2n+2)] \quad i_0 - 2 \leq 2n + 1 < i_i + 2$$
$$X(2n) = X(2n) - \beta [X(2n-1) + X(2n+1)] \quad i_0 - 1 \leq 2n < i_i + 1$$
$$X(2n+1) = X(2n+1) - \alpha [X(2n) + X(2n+2)] \quad i_0 \leq 2n + 1 < i_i$$

The parameters $\alpha$, $\beta$, $\gamma$, $\delta$, and $K$ are defined as previously and the inverse-quantized coefficient row or column element $Y(n)$ is symmetrically extended if necessary.
Wavelet compression: JPEG-2000

Finally, the component tiles are assembled, inverse component transformed (if needed), and DC-level shifted. For irreversible coding, the inverse component transformation is

\[
I_0(x, y) = Y_0(x, y) + 1.402Y_2(x, y)
\]

\[
I_1(x, y) = Y_0(x, y) - 0.34413Y_1(x, y) + 0.71414Y_2(x, y)
\]

\[
I_2(x, y) = Y_0(x, y) + 1.772Y_1(x, y)
\]

and the transformed pixels are shifted by \(2^{\text{Size}-1}\).

---

**JPEG-2000**

- \(C = 25\). rms error = 3.86
- \(C = 52\). rms error = 5.77
  Much better visual quality and smaller error than with JPEG.
  \(C = 75\).
  C = 105.
  JPEG-2000 provides usable images compressed by more than 100:1.