Image Compression Basis

Sebastiano Battiato, Ph.D.
battiato@dmi.unict.it
Compression and Image Processing

- Fundamentals;
- Overview of Main related techniques;
- JPEG tutorial;
- Jpeg vs Jpeg2000;
- SVG
The following terminology is often used when referring to digital image sizes (bit is abbreviation for *binary digit*):

<table>
<thead>
<tr>
<th>Unit</th>
<th>Byte Equivalent</th>
<th>Decimal Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit (b)</td>
<td>1 bit</td>
<td>1 bit</td>
</tr>
<tr>
<td>byte (B)</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>kilobyte (KB)</td>
<td>$2^{10}$ bytes</td>
<td>$10^3 + 2.4%$</td>
</tr>
<tr>
<td>megabyte (MB)</td>
<td>$2^{20}$ bytes</td>
<td>$10^6 + 4.9%$</td>
</tr>
<tr>
<td>gigabyte (GB)</td>
<td>$2^{30}$ bytes</td>
<td>$10^9 + 7.4%$</td>
</tr>
<tr>
<td>terabyte (TB)</td>
<td>$2^{40}$ bytes</td>
<td>$10^{12} + 10%$</td>
</tr>
</tbody>
</table>
An image with \( n \times m \) pixels each with a bit-depth of \( N \) bits can be represented by \( N \) binary bit planes of size \( n \times m \) ranging from the Most Significant Bit (MSB) to the Least Significant Bit (LSB).

\[
x(k,l) = 113 = 01110001
\]
Lena – Bit Planes

Most Significant bit (MSB)  Least Significant bit (LSB)
Bit Plane coding
Num True Color (RGB – 24 bit) images $MxN$ require:

$$3 \times \text{Num} \times MxN \text{ bytes}$$

Example:

$\text{Num} = 10$

$MxN = 1712 \times 1368 \approx 2.2 \text{ Mpixel}$

$$3 \times 10 \times 2.2 \text{Mb} \approx 67 \text{ Mbyte} !!!$$
<table>
<thead>
<tr>
<th>Application</th>
<th>Data Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uncompressed</td>
</tr>
<tr>
<td>Voice</td>
<td>64 kbps</td>
</tr>
<tr>
<td>8 ksample/s, 8 bits/sample</td>
<td></td>
</tr>
<tr>
<td>Slow-motion video (10fps) framesize 176x120, 24bit/pixel</td>
<td>5,07 Mbps</td>
</tr>
<tr>
<td>Audio conference</td>
<td>128 kbps</td>
</tr>
<tr>
<td>8 ksample/s, 16 bits/sample</td>
<td></td>
</tr>
<tr>
<td>Video conference (15fps) framesize 352x240, 24bit/pixel</td>
<td>30.41 Mbps</td>
</tr>
<tr>
<td>Digital audio</td>
<td>1,5 Mbps</td>
</tr>
<tr>
<td>44,1 ksample/s, 16 bits/sample</td>
<td></td>
</tr>
<tr>
<td>Digital video on CD-ROM (30 fps) framesize 352x240, 24bit/pixel</td>
<td>60,83 Mbps</td>
</tr>
<tr>
<td>Broadcast video (30fps) framesize 720x480, 24bit/pixel</td>
<td>1,33 Gbps</td>
</tr>
</tbody>
</table>
Fresh-Squeezed!

H₂O
Water is the redundant element.

Concentrate: Shipped, Stored and Sold

Tastes Like Fresh-Squeezed!
Aims at finding methods for reducing the number of bits needed to represent a digital image without compromising the required image quality for a given application.

Image compression (e.g. art/science of finding efficient representation for digital images) is used to:

- Reduce memory for image storage.
  - Reduce the bandwidth/time required for image transmission.
  - Increase effective data transfer rate.
Several Classifications of Compression Methods are possible:

- Based on **Data Type** (Generic, Audio, Image, Video);

- Based on **Compression Type**:
  - **Lossless**: The decoded (uncompressed) data will be exactly equal to the original (*modest compression ratios*).
  - **Lossy**: The decoded data will be a replica of the original, but not necessarily the same (*higher compression ratios*).
Information Theory (Shannon 1948) was developed to provide a mathematical tool to better design data compression algorithms and introduced a revolutionary new probabilistic way of thinking about communication. Simultaneously it was created the first truly mathematical theory of entropy (Measure of randomness).

Entropy of the source generating the data: it is impossible to compress data in a lossless way with a bitrate less than the entropy of the source that generated it.

The entropy $H$ the source generating a data is in general impossible to measure in practice, due to the larger amount of interdependencies (of infinite order) and the non-stationarities.
Entropy: measure of uncertainty

Given a random variable $X$ and probability distribution $p(X)$ the following assumptions must be granted:

1 - $H(X)$ exists;
2 - $H(X)$ has a continuous functional dependence on the probability distribution $p(X)$;
3 - If $p(x_i) = 1/n$ $i=1, 2, \ldots, n$ then $H(X)$ should be a monotonic increasing function of $n$;
4 - $H(X)$ must be obeys to the composition or additivity law
**Theorem:**

The only function $H$ that satisfies the properties 1-4 above is of the form:

$$H_0 = -k \sum_{i \in S} p_i \log_2 (p_i)$$

where $k$ is a positive constant. It is common to set $k$ to 1. $H$ is called *Entropy* or *Information Entropy*. Usually, a zero-order entropy measure is used to estimate the entropy of the source.
The Entropy function of two events with probability $p$ and $(1-p)$
For lossy coding, the rate-distortion theory was developed. Its main goal is summarized by *Rate-Distortion* optimization criterion:

**Find the lowest bitrate for a certain distortion, or the lowest distortion for a given bitrate.**
The most popular distortion measure is the mean square error (MSE):

\[ MSE = \frac{1}{N} \sum_{i=1}^{N} [x(i) - x^*(i)]^2 \]

The MSE (Mean Squared Error) does not always reflect the real distortion perceived by human visual system. For practical purposes the PSNR (Peak Signal to Noise Ratio) is used:

\[ PSNR = k \log_k (255^2 / MSE) \]
Rate-Distortion

Distortion

best solution

Optimal R-D curve

target rate

Rate
The performance of an image compression technique must be evaluated considering three different aspects:

- Compression efficiency (Compression Ratio/Factor, bit per pixel $bpp$ or bit rate);
- Image quality (Distortion Measurement);
- Computational cost.
Interpixel Redundancy relates to the statistical properties of an image (e.g., pixel-to-pixel correlation, spectral (RGB) correlation, frame-to-frame similarity, etc.) and is a function of resolution, bit-depth, image noise, and image detail. It could be Spatial/Temporal Typical techniques used in this case: Huffman/Run Length

Coding Redundancy

Coding efficiency (DCT)

Psychovisual Redundancy relates to an observer viewing an image (HVS spatial and temporal CSF, visual masking, etc.) and is a function of image resolution, noise, detail, and viewing conditions. Discards Less important information, Quantization, ....
Spatial Redundancy

<table>
<thead>
<tr>
<th>46</th>
<th>44</th>
<th>46</th>
<th>47</th>
<th>44</th>
<th>50</th>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>40</td>
<td>46</td>
<td>48</td>
<td>76</td>
<td>62</td>
<td>69</td>
</tr>
<tr>
<td>45</td>
<td>39</td>
<td>36</td>
<td>40</td>
<td>88</td>
<td>87</td>
<td>65</td>
</tr>
<tr>
<td>48</td>
<td>38</td>
<td>36</td>
<td>？</td>
<td>56</td>
<td>90</td>
<td>65</td>
</tr>
<tr>
<td>76</td>
<td>48</td>
<td>41</td>
<td>43</td>
<td>69</td>
<td>112</td>
<td>77</td>
</tr>
<tr>
<td>90</td>
<td>72</td>
<td>66</td>
<td>66</td>
<td>90</td>
<td>108</td>
<td>74</td>
</tr>
<tr>
<td>98</td>
<td>86</td>
<td>86</td>
<td>91</td>
<td>83</td>
<td>72</td>
<td>57</td>
</tr>
</tbody>
</table>
Spatial Redundancy

Original

Difference (Previous Pixel)
The frequency-dependent behavior of the human visual system (HVS) can be characterized by its response to harmonic (sinusoidal) functions.

For each sinusoid with a given frequency, the amount of contrast needed to elicit a criterion level of response from a neuron is called the contrast threshold.

The inverse of the contrast threshold is called the contrast sensitivity and when plotted as a function of frequency is referred to as the contrast sensitivity function (CSF).

The luminance CSF peaks at around 5 cycles/degree, and rapidly drops off to almost zero at 50 cycles/degree. The chrominance CSF drops even faster.
Visual Masking
The local surround of the feather is very active and the HVS needs more time to pick up the distortion.
The image after compression and decompression is identical to the original.

Only the statistical redundancy is exploited to achieve compression.

Data compression techniques such as **LZW** or **LZ77** are used in **GIF**, **PNG**, and **TIFF** file formats and the Unix “**Compress**” command.

Image compression techniques such as **lossless JPEG** or **JPEG-LS** perform slightly better.

Compression ratios are typically ~2:1 for natural imagery but can be much larger for document images.
Lossy (Irreversible) Compression

The reconstructed image contains degradations with respect to the original image.

Both the statistical redundancy and the perceptual irrelevancy of image data are exploited.

Much higher compression ratios compared to lossless.

Image quality can be traded for compression ratio.

The term **visually lossless** is often used to characterize lossy compression schemes that result in no visible degradation under a set of designated viewing conditions.
Lossy/Lossless data compression in Image Processing are mainly based on the following coding approaches:

- Differential Coding: Pulse Code Modulation (PCM), Palette based;
- Quad Trees
- Huffman/Run-length;
- Arithmetic;
- Transform Domain (DCT, DWT,…).
Compression-Decompression process

Encoder

Data → Source Coder → Channel Coder

Decoder

Data* ← Source Decoder ← Channel Decoder
S source that generates random $s_i$

self-information of $s_i$:

$$I(s_i) = \log(1/p_i) = -\log(p_i)$$

Entropy (~ randomness) of source S:

$$H(S) = \sum p_i \log_2 \left( \frac{1}{p_i} \right)$$

$H(S)$ bounds the coding efficiency
Differential Coding

\[ y_i = (x_i - x_{i-1}) \]

- Based on Interpixel correlation

- \( y_i \) are prediction residual
In DPCM, a combination of previously encoded pixels (A, B, C) is used as a prediction (\( p \)) for the current pixel (X).

The difference between the actual value and the prediction (\( p - X \)) is encoded.

\[
\begin{align*}
\rho_1 &= A \\
\rho_2 &= (A+C)/2 \\
\rho_3 &= (A+C-B)/3 \\
\rho_4 &= w_1 A + w_2 B + w_3 C \\
\text{with} & \quad w_1+w_2+w_3=1
\end{align*}
\]

- Low complexity
- High quality (limited compression)
- Low memory requirements
Differential coding

Original

DPCM Output

![Graphs showing pixel values and distributions for Original and DPCM Output images.](image-url)
\[ P_{ij} = \alpha \hat{y}_{ij-1} \]
\[
q_{ij} = \begin{cases} 
+\xi & e_{ij} > 0 \\
-\xi & \text{otherwise}
\end{cases}
\]

<table>
<thead>
<tr>
<th>Input</th>
<th>Encoder</th>
<th>Decoder</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>( f )</td>
<td>( \hat{f} )</td>
<td>( e )</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>14.0</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>20.5</td>
<td>-6.5</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>14.0</td>
<td>1.0</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>14</td>
<td>29</td>
<td>20.5</td>
<td>8.5</td>
</tr>
<tr>
<td>15</td>
<td>37</td>
<td>27.0</td>
<td>10.0</td>
</tr>
<tr>
<td>16</td>
<td>47</td>
<td>33.5</td>
<td>13.5</td>
</tr>
<tr>
<td>17</td>
<td>62</td>
<td>40.0</td>
<td>22.0</td>
</tr>
<tr>
<td>18</td>
<td>75</td>
<td>46.5</td>
<td>28.5</td>
</tr>
<tr>
<td>19</td>
<td>77</td>
<td>53.0</td>
<td>24.0</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>
To find an indexing that optimizes entropy of differences

Original

Gray-level

Random

Heuristic
Histogram of the differences in the original image

Entropy = 5.25
Histogram of the differences in the random image

Entropy = 5.85
Histogram of the differences in the “heuristic” image

Entropy = 4.65
Entropy of differences

Gray-level: 5.25 db
Random: 5.85 db
Heuristic: 4.65 db
Quad Trees

Matlab Demo...
Variable Length coding

- Goal: exploit non-uniformity of probability distribution of quantization indices
- Simple example:

<table>
<thead>
<tr>
<th>Probabilities</th>
<th>Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_q(q) = \begin{cases} 1/2 &amp; \text{if } q = 0 \ 1/4 &amp; \text{if } q = 1 \ 1/8 &amp; \text{if } q = 2 \ 1/8 &amp; \text{if } q = 3 \end{cases} )</td>
<td>1 01 001 000</td>
</tr>
</tbody>
</table>

- Average code word length: \( \frac{1}{2} \cdot 1 \text{ bits} + \frac{1}{4} \cdot 2 \text{ bits} + 2 \cdot \frac{1}{8} \cdot 3 \text{ bits} = 1.75 \text{ bits} \)
- Often much bigger gains for image compression
Symbol modeling and encoding involves the process of defining a statistical model for the symbols to be encoded (e.g., quantizer output levels or indices) and assigning a binary codeword to each possible output symbol based on its statistics.

• The resulting code should be **uniquely decodable**, i.e., each string of input symbols should be mapped into a unique string of output binary symbols.

• Examples are fixed-length coding, **Huffman** coding, **Arithmetic** coding, Golomb-Rice coding, Lempel-Ziv-Welch (LZW) coding.
Huffman Coding

Probability model and symbol-to-codeword are combined
Input: sequence of symbols.

- Order the symbols according to their probabilities.
- Apply a contraction to the two symbols with the smaller probabilities.
- Repeat the previous step until the final set has only one member.

Construction of a binary tree:
The codeword for each symbol is obtained traversing the binary tree from its root to the leaf corresponding to the symbol.
Huffman Process

\[ l_{avg} = \sum l_i p_i \]

\[ H(S) \leq l_{avg} \leq H(S) + 1 \]
Run-Length coding

(pixel, value)

The combination of a run-length coding scheme followed by a Huffman coder forms the basis of image compression standards.

These compression standards yield good compression (20:1 to 50:1)
An arithmetic coder accepts at its input the symbols in a source sequence along with their corresponding probability estimates, and produces at its output a code stream with a length equal to the combined ideal code-lengths of the input symbols.

Some implementations of arithmetic coding adaptively update the symbol probability estimate in each context as the symbols get encoded.

Practical implementations of AC, such as the JBIG/JPEG QM-Coder or MQ-Coder
Codifica Aritmetica: MQ-Coder

coding “H”

0.35 0.65

AEGHLO#

0.365 0.44

0.41375 0.42125

0.418625 0.419375

0.4191875 0.4193375

0.41933

Coding “E”

0.35

AEGHLO#

AEGHLO#

coding “L”

0.41375

AEGHLO#

AEGHLO#

coding “L”

0.418625

AEGHLO#

AEGHLO#

coding “O”

0.4191875

AEGHLO#

AEGHLO#

coding “#”

AEGHLO#

AEGHLO#

Code Point : 0.41933
A many-to-one mapping that reduces the number of possible signal values at the cost of introducing errors.

- The simplest form of quantization (also used in all the compression standards) is **scalar quantization** (SQ), where each signal value is individually quantized.
Not-Uniform Scalar quantization

Sometimes, this convention is used:

\[
\begin{align*}
Q & : x \rightarrow q \\
Q^{-1} & : q \rightarrow \hat{x}
\end{align*}
\]
Vector Quantization
The joint quantization of a block of signal values is called **vector quantization** (VQ). It has been theoretically shown that the performance of VQ can get arbitrarily close to the rate-distortion (R-D) bound by increasing the block size.
Most of the compression occurs in the quantization stage.

Lossless compression/entropy coding, typically involves run-length coding combined with Huffman codes, further save bits in an invertible fashion.
\[ F(u, v) = \frac{1}{4} C(u)C(v) \left[ \sum_{x=0}^{7} \sum_{y=0}^{7} f(x, y) \cos \left( \frac{(2x+1)u\pi}{16} \right) \cos \left( \frac{(2y+1)v\pi}{16} \right) \right] \]

\[ f(x, y) = \frac{1}{4} \left[ \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u, v) \cos \left( \frac{(2x+1)u\pi}{16} \right) \cos \left( \frac{(2y+1)v\pi}{16} \right) \right] \]

where:

\[ C(u), C(v) = \frac{1}{\sqrt{2}} \quad \text{for } u, v = 0; \]

\[ C(u), C(v) = 1 \quad \text{otherwise} \]
The 64 (8 x 8) DCT basis functions:
DCT coefficients can be viewed as weighting functions that, when applied to the 64 cosine basis functions of various spatial frequencies (8 x 8 templates), will reconstruct the original block.

\[
y = y(0,0) x + y(1,0) x + \ldots + y(7,7) x
\]

Original image block  **DC (flat) basis function**  **AC basis functions**
The DCT coefficients are quantized to limited number of possible levels.

The Quantization is needed to reduce the number of bits per sample.

Example:

101000 = 40 (6 bits precision) → Truncates to 4 bits = 1000 = 8 (4 bits precision).

i.e. 40/5 = 8, there is a constant N=5, or the quantization or quality factor.

Formula:

\[ F( u, v) = \text{round}\left( \frac{F( u, v)}{Q( u, v)} \right) \]

- \( Q( u, v) = \text{constant} \Rightarrow \text{Uniform Quantization.} \)
- \( Q( u, v) = \text{variable} \Rightarrow \text{Non-uniform Quantization.} \)
DCT example
The **forward discrete wavelet transform** (DWT) decomposes a one-dimensional (1-D) sequence (e.g., line of an image) into two sequences (called **subbands**), each with half the number of samples, according to the following procedure:

- The 1-D sequence is separately **low-pass** and **high-pass** filtered.
- The filtered signals are downsampled by a factor of two to form the low-pass and high-pass subbands.
- The two filters are called the **analysis filter bank**.
The 1-D Two-Band DWT

Analysis filter bank

Synthesis filter bank
2-D Wavelet decomposition
• Multiple resolution representation

• Lossless representation with integer filters

• Better decorrelation than DCT, resulting in higher compression efficiency

• Use of visual models: DWT provides a frequency band decomposition of the image where each subband can be quantized according to its visual importance (similar to the quantization table specification in JPEG-DCT)
Scalable Vector Graphics
An Overview
• SVG is a language for describing two-dimensional graphics in XML.

• SVG allows for three types of graphic objects:
  - Vector graphic shapes (e.g., paths consisting of straight lines and curves),
  - Images;
  - Text (also called glyphs).
• **SVG 1.0** is a Web standard (a W3C Recommendation).

• **Work continues on the modular SVG 1.1/1.2** and on the **Mobile SVG profiles**.

• **Powerful portability** *(display resolution independence)*
An SVG document is an **XML** document. That means that SVG documents have certain basic attributes:

- All tags must have a start and end tag,

  `<rect>...</rect>;  <text>...</text>;  <g>...</g>

  or must be noted as an empty tag. Empty tags are closed with a backslash, as in

  `<rect ... />;  <circle ... />;  <path ... />`

- Tags must be nested properly. If a tag is opened within another tag, it must be closed within that same tag. For example,

  `<g><text>Hello there!</text></g>` **is correct,**

  but `<g><text>Hello there!</text></g>` **is not.**
● The document must have a **single root**. Just as a single `<html>` element contains all content for an HTML page, singular `<svg>` element contains all content for an SVG document.

● The document should start with the **XML** declaration, `<?xml version="1.0"?>`

● The document should contain a DOCTYPE declaration, which points to a list of allowed elements:

```xml
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <desc>All SVG documents should have a description</desc>
  <defs>
    <!-- Items can be defined for later use -->
  </defs>
  <g>
    <circle cx="100" cy="100" r="75" fill="green"/>
  </g>
</svg>
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="400" height="300" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <desc>Text on a path</desc>
  <defs>
    <path id="wavyPath" d="M75,100 c25,-75 50,50 100,0 s50,-50 150,50"/>
  </defs>
  <rect x="1" y="1" width="398" height="200" fill="none" stroke="blue" />
  <text x="50" y="50" font-size="14">
    <textPath xlink:href="#wavyPath">
      Text travels along any path that you define for it.
    </textPath>
  </text>
</svg>
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="400" height="200" xmlns="http://www.w3.org/2000/svg">
<desc>Colors</desc>
<defs>
  <linearGradient id="linear1">
    <stop offset="0%" stop-color="#FF0000"/>
    <stop offset="50%" stop-color="#FFFFFF"/>
    <stop offset="100%" stop-color="#0000FF"/>
  </linearGradient>
  <linearGradient id="linear2" x1="100%" y1="0%" x2="0%" y2="100%"/>
  <linearGradient id="linear3" gradientTransform="rotate(90)"/>
  <radialGradient id="radial1"/>
  <radialGradient id="radial2" fx="225" fy="225"/>
  <radialGradient id="radial3" cx="25%" cy="25%" r="75%"/>
</defs>
<g>
  <!-- First row -->
  <rect x="10" y="10" height="100" width="100" stroke="#000000" fill="url(#linear1)"/>
  <rect x="125" y="10" height="100" width="100" stroke="#000000" fill="url(#linear2)"/>
  <rect x="240" y="10" height="100" width="100" stroke="#000000" fill="url(#linear3)"/>
  <!-- Second row -->
  <rect x="10" y="125" height="100" width="100" stroke="#000000" fill="url(#radial1)"/>
  <rect x="125" y="125" height="100" width="100" stroke="#000000" fill="url(#radial2)"/>
  <rect x="240" y="125" height="100" width="100" stroke="#000000" fill="url(#radial3)"/>
</g>
</svg>
Comparison between raster image and vectorial image.
Comparison between raster image and vectorial image.
Possible Optimization

- Use of pattern gradient detection instead of multiple color surfaces;
- Better edge detection;
- Preprocessing filtering (Noise Reduction, Color Enhancement, Sharpness);
- Better use of “style” and “transform” tags.