Appendix 2. JPEG

JPEG is an international image compression standard developed by Joint Pictures Expert Group. It can be applied to images represented in the BMP format. The standard coder contains the following main blocks:

- Color format conversion
- Transform
- Quantizer
- Lossless (entropy) encoder

The JPEG standard is based on the two-dimensional discrete cosine transform (2-D DCT) coding technique. The component Y and decimated components Cb and Cr (U and V) (see Home assignment 1) are processed by blocks of size 8×8 pixels. The 2-D DCT is applied to each 8×8 block of pixels

\[
X(k, l) = \frac{c(k)}{2} \sum_{i=0}^{7} \left[ \frac{c(l)}{2} \sum_{j=0}^{7} x(i, j) \cos \left( \frac{(2j + 1)l \pi}{16} \right) \right] \cos \left( \frac{(2i + 1)k \pi}{16} \right)
\]

where \(x(i, j)\) denotes the pixel of the image component (Y, U or V) and \(X(k, l)\) is the transform coefficient. Since DCT is a separable transform, the two-dimensional DCT for blocks 8×8 can be performed by first applying 1-D transform to rows of each 8×8 block and then applying 1-D transform to the columns of the resulting block. The one-dimensional DCT for sequence of length 8 is given by the formula

\[
X(k) = \frac{c(k)}{2} \sum_{n=0}^{7} x(n) \cos \left( \frac{(2n + 1)k \pi}{16} \right), k = 0, 1, ..., 7
\]
\[ c(k) = \begin{cases} 
\frac{1}{\sqrt{2}}, & k = 0 \\
1, & k \neq 0 
\end{cases} \]

\(x(n)\) is an input pixel and \(X(k)\) is the transform coefficient. The inverse transform can be written as

\[ x(n) = \sum_{k=0}^{7} X(k) \frac{c(k)}{2} \cos \left( \frac{(2n+1)k\pi}{16} \right). \]

When performing 1-D DCT we decompose the 1×8 or 8×1 block over a set of eight different cosine waveforms sampled at eight points. The transform coefficients are amplitudes of the corresponding basis functions. The coefficient that scales the constant basis function \((k = 0)\) is called the DC coefficient. The other coefficients are called AC coefficients.

When performing 2-D DCT we decompose the 8×8 image block using a set of 64 2-D cosine basis functions. These functions are created by multiplying a horizontally oriented set of 1-D 8-point basis functions by a vertically oriented set of the same functions. By convention, the DC term of the horizontal basis functions is to the left, and the DC term for vertical functions is at the top. Because the 2-D basis functions are products of two 1-D DCT basis functions, the only constant basis function is in the upper left corner of the array. The coefficient for this basis function is called the DC coefficient, whereas the rest of the coefficients are called AC coefficients. The horizontal DCT frequency of the basis function increases from left to right and the vertical DCT frequency of the basis function increases from top to bottom.

The obtained transform coefficients are quantized by the uniform scalar quantizer. The quantization is implemented as rounding-off of the DCT coefficients divided by a quantization step. The values of steps are set individually for each DCT coefficient, using criteria based on visibility of the basis functions. Thus, the quantized coefficient is

\[ Z(k,l) = \text{round} \left( \frac{Y(k,l)}{Q(k,l)} \right) = \left\lfloor \frac{Y(k,l) \pm Q(k,l)/2}{Q(k,l)} \right\rfloor, \quad k,l = 0,1,...,7 \]

where \(Q(k,l)\) is the \((k,l)\)th entry of the quantization matrix \(Q\) of size 8×8.

The standard JPEG uses two different quantization matrices. The first matrix is used to quantize the luminance component \((Y)\) and has the form
The second matrix is used to quantize the chrominance components and looks like

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

!!! Notice that more modern implementations of the JPEG standard allow to quantize all AC coefficients by the same step \( \Delta \), say, and recommend to use quantization step equal to 8 for the DC coefficient.

The quantized DCT coefficients are coded by a variable-length coder. The coding procedure is performed in two steps. At the first step the DC coefficient is DPCM coded, using the first order predictor that is the DC value of the previous 8×8 coded block is subtracted from the current DC coefficient. The AC coefficients are coded by a so-called run-length coder which will be explained below. At the second step the obtained values are coded by the Huffman code.

1 Coding of DC coefficients

Let \( DC_i \) and \( DC_{i-1} \) denote the DC coefficients of the \( i \)th and \( (i - 1) \)th blocks, respectively. Due to high correlation between DC coefficients they
are DPCM coded, that is, their difference is computed and then coded. For gray scale images (or one of the Y, U, V components of a color image) a pixel is represented by 8 bits. Thus, the difference $DC_i - DC_{i-1}$ takes values from the range [-2047, 2047]. This range is split into 12 categories, where the $i$th category includes the differences with the length of their binary representation equal to $i$ bits. These categories are the first 12 categories shown in Table 1.

Each DC coefficient is described by a pair (category, amplitude). If the value $DC_i - DC_{i-1} \geq 0$, then the amplitude is the binary representation of this value of length equal to the category. If $DC_i - DC_{i-1} < 0$, then the amplitude is the codeword of the complement binary code for the absolute value of $DC_i - DC_{i-1}$, which also has length equal to category. The category value is then coded by the Huffman code.

Example. Let $DC_{i-1} = 191$ and $DC_i = 180$. Then the difference $DC_i - DC_{i-1} = -11$. It follows from Table 1 that the value $-11$ belongs to the category 4. The binary representation of value 11 is 1011 and the codeword of the complementary code is 0100. Thus, the value $-11$ is represented as (4,0100). If the codeword of the Huffman code for 4 is 110 then $-11$ is coded by the codeword 1100100 of length 7. The decoder first processes the category value (in our case it is 4) then the next 4 bits correspond to the value of $DC_i - DC_{i-1}$. Since the most significant bit is equal to 0 then the value is negative. Inverting bits we obtain the binary representation of 11. Notice that using the categories simplifies the Huffman code. Without using categories we would need the Huffman code for an alphabet of a much larger size, that is, coding and decoding would be much more complicated.

2 Coding of AC coefficients

For gray scale images or (Y, U or V components) the AC coefficients can take values from the range [-1023,1023]. After quantization many of these coefficients become zeros. In other words, it is necessary to code only small number of non-zero coefficients simply indicating before their positions. To do this efficiently, the 2-D array of the DCT coefficients is rearranged into a 1-D linear array by scanning in zigzag order as shown in Fig.1. This zigzag index sequence creates a 1-D vector of coefficients, where the DCT coefficients, which are amplitudes of the basis functions with lower frequencies, tend to be at lower indices. This zigzag sequence is an important part of the
### Table 1: Categories of integer numbers

<table>
<thead>
<tr>
<th>Category</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1,1</td>
</tr>
<tr>
<td>2</td>
<td>-3,-2,2,3</td>
</tr>
<tr>
<td>3</td>
<td>-7,...,-4,4,...,7</td>
</tr>
<tr>
<td>4</td>
<td>-15,...,-8,8,...,15</td>
</tr>
<tr>
<td>5</td>
<td>-31,...,-16,16,...,31</td>
</tr>
<tr>
<td>6</td>
<td>-63,...,-32,32,...,63</td>
</tr>
<tr>
<td>7</td>
<td>-127,...,-64,64,...,127</td>
</tr>
<tr>
<td>8</td>
<td>-255,...,-128,128,...,255</td>
</tr>
<tr>
<td>9</td>
<td>-511,...,-256,256,...,511</td>
</tr>
<tr>
<td>10</td>
<td>-1023,...,-512,512,...,1023</td>
</tr>
<tr>
<td>11</td>
<td>-2047,...,-1024,1024,...,2047</td>
</tr>
<tr>
<td>12</td>
<td>-4095,...,-2048,2048,...,4095</td>
</tr>
<tr>
<td>13</td>
<td>-8191,...,-4096,4096,...,8191</td>
</tr>
<tr>
<td>14</td>
<td>-16383,...,-8192,8192,...,16383</td>
</tr>
<tr>
<td>15</td>
<td>-32767,...,-16384,16384,...,32767</td>
</tr>
<tr>
<td>16</td>
<td>32768</td>
</tr>
</tbody>
</table>
coding model, as it affects the statistics of the symbols. When the coefficients are ordered in this fashion the probability of coefficients being zero is an approximately monotonic increasing function of the index.

The run-length coder generates a codeword \(((\text{run-length},\text{category}),\text{amplitude})\), where run-length is the length of zero run followed by the given non-zero coefficient, amplitude is the value of this non-zero coefficient and category is the number of bits needed to represent the amplitude. The pair \((\text{run-length, category})\) is coded by the 2-D Huffman code and the amplitude is coded as in the case of DC coefficients and is added to the codeword.

**Example.** Let the nonzero coefficient preceded by 6 zeros be equal to -18. It follows from Table 1 that -18 belongs to the category 5. The codeword of the complement code is 01101. Thus, the coefficient is represented by \(((6,5),01101)\). The pair \((6,5)\) is coded by the Huffman code and the value 01101 is added to the codeword. If the codeword of the Huffman code for \((6,5)\) is 1101, then the codeword for -18 is 110101101.

There are two special cases when we encode the AC coefficients.

After a non-zero coefficient all other AC coefficients are zero. In this case the special symbol (EOB) is transmitted which codes end-of-block condition.
A pair \((\text{run-length}, \text{category})\) appeared which is not included in the table of the Huffman code. In this case a special codeword called escape-code followed by the uniform codes for run-length and non-zero value are transmitted.

In order to improve coding efficiency more modern implementations of the JPEG standard use a map of zero/nonzero blocks. We say that a block of size \(8 \times 8\) is zero if it contains only nonzero DC coefficient and all AC coefficients of this block are zeros. The size of the map is equal to \(M/8 \times N/8\), that is, the number of \(8 \times 8\) blocks in an image component. We mark zero block by 0 and nonzero block by 1.

The quality of the synthesized image is characterized by the signal-to-noise ratio (SNR) at the output of the decoder

\[ \text{SNR} = 10 \log_{10}(E_{\text{inp}}/E_n) \text{(dB)} \]

where \(E_{\text{inp}}\) denotes the energy of the original image component, \(E_n\) is the energy of the quantization noise. \(E_n\) represents the energy of the difference between the original and the reconstructed image component. More often to characterize the synthesized image quality the peak signal-to-noise ratio (PSNR) is used. It is defined as follows

\[ \text{PSNR} = 10 \log_{10}((255)^2/E_{na}), \]

where 255 is the maximal pixel value, \(E_{na}\) is the average energy of the quantization noise. For the image of size \(M \times N\), \(E_{na}\) is computed as

\[ E_{na} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (p_{i,j} - \hat{p}_{i,j})^2, \]

where \(p_{i,j}\) is the \((i,j)\)th pixel value of an image component and \(\hat{p}_{i,j}\) is the \((i,j)\)th pixel value of the reconstructed image component.

Coding efficiency is usually evaluated as a compression ratio which is the ratio of the original file size in bits divided by the size of the compressed file in bits.