9

Introduction to digital slow-scan TV

The development of computers and new opportunities, which gives us the use of powerful processors and sound cards modems have resulted in the design of new communication modes. One of these modes is the digital slow-scan television (DSSTV), which allows transmission of images without any loss of quality.

We have two choices for digital image transmission. The first system is *RDFT – Redundant Data File Transfer*, which is the result of several years of creative efforts of Barry Sanderson, KB9VAK and a group of ham radio enthusiasts.

The second system is called *Digital Radio Mondiale (DRM)*. It's open standard for digital broadcast on short waves. It was developed by DRM Consorcium and was standardized by organizations ITU, IEC and ETSI. The DRM system is used mainly by short-wave broadcast stations and its modification for ham radio purposes is called *HamDRM*.

It is possible to use these modes also for transfer of any types of files (text, sounds, software,...) instead of images.

The difference between analog and digital SSTV is huge. There are used entirely different modulation principles and essentially some files in JPEG, JPEG200, PNG, etc. formats are sent. Also the error correction and detection is implemented using Reed-Solomon code.

The result is, that image is transfered without any distortion (means transfer distortion, the images should be distorted due to loss compression techniques, but you can control the degree of it). There is no constrain in image resolution, it is given by image file parameters. Only constrain is the bandwidth of SSB channel and the resulting maximum data rate and the time required for transmission.

There are also bigger requirements for stations equipment in comparison with classic SSTV. You need better computer, at least 1GHz CPU and 256 MB RAM. In your ham-shack you can use without any problems an oldish computer (e.g. Pentium 150 MHz) for digital modes like RTTY or PSK, packet radio and analog

SSTV, but algorithms of signal processing for DSSTV is so complex that a slow 150MHz Pentium would not be able to process signals in real-time.

Also, there are increased requirements on the transceiver used. Used modulation techniques use the maximal width of the communication channel and SSB transceiver without the linear range of SSB channel isn't usable for DSSTV. Of course, there must be switched off any additional signal and modulation filters (speech processor, equalizer,...).

Sound card interfacing is same as for classic SSTV, the connection between TRX and sound card is enough. Output level of sound card should be about $\frac{1}{3}$ of maximum (switch off software AGC). When signal is overexcited, the intermodulation causes disproportionately large signal and distortion and the signal cannot be decoded. A peak power with 100W transceiver is about 20–25 W.

9.1 Digital communication basics

Before I describe the transmission systems we look at some important concepts of data communication. What interests us most is the speed which is possible to transmit information – we distinguish between the speed of transmission and modulation:

Symbol rate v_m – express the number of changes a of carrier signal per second. It is measured in unit *Baud (Bd)* or *Symbols per second (S/s)*. Symbol rate does not say anything about how much information transmitted on signal carrier.

$$v_m = 1/a \quad [\text{Bd}]$$

Bit rate v_p – indicates the amount of information transferred per second. It is expressed in *bits per second (bps)*. Bit rate says nothing about how fast the signal carrier changes.

$$v_p = v_m \cdot \log_2 m \quad [\text{bps}],$$

where m is the number of modulation states.

We know from previous chapters that an important feature of communication channel is a limited bandwidth B. Relation between symbol rate and bandwidth shows *Nyquist rate:*

$$v_m = 2 \cdot B.$$

Ideally, the symbol rate should be twice the bandwidth. Substituting the formula for the symbol rate, we get:

$$v_p = 2 \cdot B \cdot \log_2 m.$$

Let's look on the relationship between symbol rate and bit rate, because these two term are often use interchangeably. E.g. packet radio on VHF has bit rate 1200 bps

and the used modulation is AFSK (Audio Frequency-Shift Keying). Frequencies carrying information are two – 2200 Hz for mark (log. 1) and 1200 Hz for space (log. 0). We know $v_p = 1200$ bps, $m = 2$, so symbol rate is equal to bit rate:

$$v_m = \frac{v_p}{\log_2 m} = \frac{1\,200}{\log_2 2}\,\mathrm{Bd} = 1\,200\,\mathrm{Bd}.$$

A packet radio is based on ITU-T V.23 specification for telephone modems, where bandwidth is limited to about 4 kHz. Modern dial-up modems, but have a much higher bit rates, up to 56 kbps and the bandwidth remains 4 kHz. How is that possible?

It's possible through the used advanced modulation, which has more modulation states m then two. For example, modems based on V.32 specification can use bit rate up to 9,600 bps. There is used *QAM (Quadrature Amplitude Modulation)*, which in case of QAM-16 has 16 states per one modulation symbol. The symbol rate in this case is:

$$v_m = \frac{v_p}{\log_2 m} = \frac{9\,600}{\log_2 16}\,\mathrm{Bd} = 2\,400\,\mathrm{Bd}.$$

One could think that it's possible to reach any speed because of improved modulation and more states. Unfortunately not, because there are stark physical limits. Maximal channel capacity C (bit rate) in bps is given by *Shannon's law*, which depends on bandwidth B (Hz) and channel parameters signal/noise ratio S/N (dB):

$$C = B \cdot \log_2\left(1 + \frac{S}{N}\right).$$

As we can see the maximum bit rate speed is not affected by the used technology, but the bandwidth B and signal/noise ratio (SNR), which cannot be changed. SNR is given in decibels (dB) and describes the ratio of a signal power to a noise power of a processed bandwidth.

9.2 Error detection and correction

The error detection codes are used as a check of error-free transmission. The idea is based on some extra data (redundancy) added to a message. The redundancy is generated from some input data on a transmission side *(FEC – Forward Error Correction)* and on a reception side it is possible to check if the data was transferred without error. An used code may have also ability for an error correction, so data affected during transmission can be repaired on reception side without retransmission. There are several error detection codes, e.g. even parity described in chapter ??.

The codes have several parameters. First is the bit length of information k, which we want to encode and the length of codeword n. The difference $r = n - k$ is the length of redundancy data. Redundancy does not transfer any information,

but it is only used for error detection and possibly correction. The ratio of the number of information symbols to the number of all symbols

$$R = \frac{k}{n} = \frac{n-r}{n}$$

expresses *information ratio*. In practice, we require that redundancy is minimized.

The ability of the code, how many errors should be detected or corrected is given by *Hamming distance*. It is determined as the number of different symbols of two different codewords. The most important is minimal Hamming distance d of all arbitrary codewords. E.g. Hamming distance of 01**0**100**0** and 01**1**100**1** is $d = 2$. The errors during transmission cause replacement of one symbol to another and Hamming distance indicates how many replacements may occur to change the codeword to another valid codeword. Is is advantageous to have a Hamming distance of codewords larger as possible. So if you want code that reveals just one error bit, the minimum distance must be $d = 2$. Block code with minimal distance d *detects* all t-multiple errors for $t < d$. If there is too much errors that $t = d$ or $t > d$ there should be created a new valid word, so the error cannot be detected. The code can correct errors for larger d, if for error word is found a valid codeword with smaller Hamming distance between error and valid word. The block code *corrects* all t-multiple error when

$$t < \frac{d}{2}.$$

These findings can be demonstrated on a simple case of 2-bit code secured with even parity. Two-bit code can have a total of 4 words of information, and a redundant bit will be added, so that the number of log. ones in the codeword will be even.

Information word	**Parity**	**Codeword**
00	0	000
01	1	011
10	1	101
11	0	110

Table 9.1: none

The resulting code words have 3 bits and there are $2^3 = 8$ different bit words (code words are bold):

000	001	010	**011**
100	**101**	**110**	111

Table 9.2: none

The minimum distance d of our parity code is equal to 2, so the code is able to detect just one error. When word 011 is sent and 010 is received we know that there is an error. If there are two errors and 011 changes to 000, then there is a word that belongs to a set of codewords and error isn't detected.

In the following sections are described some commonly used error-detection and correction codes.

9.2.1 Cyclic redundancy check

The *CRC* is commonly used code. The *systematic cyclic code*, adds a fixed-length check *(checksum)* value to message. The checksum is used for error detection.

CRC calculation is performed on block or section of data is stored in memory, the k-bit sequence is represented as a polynomial $G(x)$. This is polynomial is divided by generating polynomial $P(x)$ in arithmetic modulo 2. The result is polynomial $Q(x)$ and the remainder after dividing $R(x)$. The remainder $R(x)$ is added to input data and transmitted in message.

On the reception side the division with $P(x)$ is computed again and new remainder $R'(x)$ is compared with transferred remainder $R(x)$. If both values are the same transfer went without error, if not at least one bit was transferred incorrectly.

9.2.2 Hamming code

In the area of data communications (e.g. TV teletext) is sometimes used *Hamming code*, which can detect up to two errors and in the case of a one error it is able to determine at what point of codeword error occurred and it can fix received bits.

Basically, it uses for its purposes even parity. While the parity bits are in the final codeword positioned at the serial number is equal to the square of 2 (1., 2., 4., 8.,...). Under the control bit position is then selected certain sequence of information words, which is used to determine the value of control bit.

9.2.3 Reed-Solomon code

Hamming code works well in environments where errors occur randomly and their incidence is low (e.g. in computer memory, which can detect a erroneous bit to 100 million). But even if that failure causes a greater number of adjacent bits are corrupted *(burst error)*, the Hamming code is useless. In the field of radio transmission, where the signal is often affected by atmospheric disturbances, fade outs and interference, then errors occur in clusters. This means that close to the incorrect symbol are other symbols incorrect too. For burst error correction is applied *Reed-Solomon code (RS)*.

RS codes are the most widely used codes for detection and error correction. They are characterized by having the largest possible minimum distance, and compared to the previous code will not correct the individual bits, but all symbols. RS code have found application in the number of areas – is used by NASA for space communications, protects data on CD-ROMs and DVDs and is also used for

terrestrial transmission of HDTV or in the data modems for the cable television networks.

Like the CRC the RS code is systematic. For its generation are used the algebraic calculations of Galois field.

Parameters of the RS(n, k) are defined as follows:

- ▷ s – number of bits in one informational symbol,
- ▷ k – number of s-bit symbols in data block,
- ▷ n – number of bits in codeword.

RS(n, k) is able to correct $\frac{n-k}{2}$ errors in k information symbols. Often used code is RS$(255, 223)$, it uses 223 8-bit symbols for creation of 255 symbols of codeword. There is 32 symbols dedicated for error correction. RS$(255, 223)$ is able to repair up to 16 erroneous 8-bit symbols.

9.3 Data compression

The image with resolution 320×240 with a color depth of 16 million colors (256^3) takes 230,400 Bytes (320×240×3) without compression. This file would be transmitted fortyone minutes by RDFT with speed 92 Bytes per second! This time is really scary in comparison with analog SSTV. It is really necessary to reduce the file size and reduce the time required for transmission.

The *data compression* is widely used in such cases, where the data capacity of communication channels or storage media and memory is limited.

The compression is the process where the physical data block size is reduces to a certain level. Input data is compressed using the compression algorithm and then stored on media or transmitted via communication channel. The data are decompressed in its original form, when a media is read or a signal received.

One of the important parameters of compression algorithms is *lossy*. While the programs or text must by stored in perfect form, but in case of sound, images or animations we can settle with the omission of certain details, then we're talking about lossy compression method.

9.3.1 Information entropy

When Clause E. Shannon was engaged in applied mathematic of communication theory during 1940s, he started with definition of informational value of message content. The message which is repeated often is less informative than the message, which occurs sporadically. So, the often repeated message is more likely than the unique. The probability in mathematic is expressed by real numbers in range from 0 (for a completely unlikely events) to 1 (for the phenomena that occur surely). Shannon defined the amount of information $I(x_i)$ for the message x_i with the probability of occurrence $p(x_i)$ as follows:

$$I(x_i) = -\log_2 p(x_i)\,.$$

The graph of negative logarithm see on **fig. 9.1** – if the message content is less likely that its information value is higher.

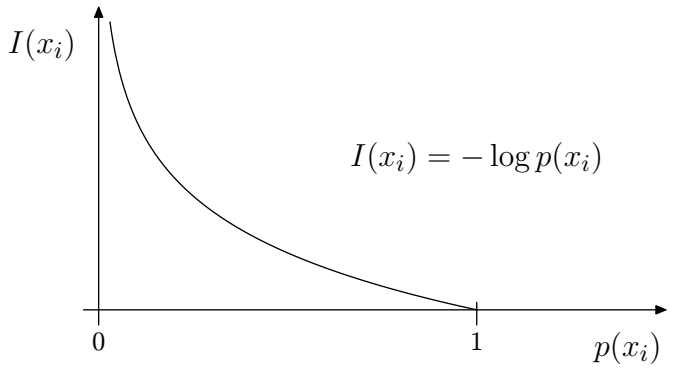


Figure 9.1: The relation between information content $I(x_i)$ and its probability $p(x_i)$.

Information entropy H is defined as *average rate of information value* $I(x_i)$:

$$H = \sum_{i=1}^{N} p(x_i)\, I(x_i) = -\sum_{i=1}^{N} p(x_i) \log_2 p(x_i) \qquad [\text{bit}]$$

We show the entropy meaning in example. We need to transfer messages $a_1, a_2, \ldots a_8$ and probability of their occurrence is same: $p_i = 1/8 = 0,125$. The entropy of source is

$$H = -\sum_{i=1}^{8} p(a_i) \log_2 p(x_i) = -\left(8 \cdot \frac{1}{8} \log_2 \frac{1}{8}\right) = 3\,\text{bits}.$$

The observed entropy determines how the message content can be encoded for data transmission. The length of message in bits is greater then or equal to the entropy, without loss of information. So the message can be encoded as word of 3bit length: 000, 001, 010,...Maximum entropy is reached when the probability of occurrence of each message is the same.

But the messages have often different probabilities in many cases. In this example we need to transfer messages $a_1, a_2, \ldots a_7$. Their probabilities are $p(a_1) = 0.235$, $p(a_2) = 0.206$, $p(a_3) = 0.176$, $p(a_4) = 0.147$, $p(a_5) = 0.118$, $p(a_6) = 0.059$, $p(a_7) = 0.029$, $p(a_8) = 0.029$. Entropy of source is

$$\begin{aligned} H &= -\sum_{i=1}^{7} p(a_i) \log_2 p(a_i) = \\ &\sim = -\big(0.235 \cdot (-2.09) + 0.206 \cdot (-2.28) + 0.176 \cdot (-2.50) + 0.147 \cdot (-2.76) + \\ &\quad 0.118 \cdot (-3.08) + 0.059 \cdot (-4.08) + 0.029 \cdot (-5.08) + 0.029 \cdot (-5.08)\big)\,\text{bits} \\ H &\approx -2.712\,\text{bits} \end{aligned}$$

We see, that the entropy of source is lower and because data bits are not divisible, it is necessary to encode the message again to the words of length 3. But suspect that such an encoding is no longer optimal. There is the idea to encode frequently

occurring words as the message of the shorter length. This idea was well-counseled by David A. Huffman, the Shannon's student.

9.3.2 Huffman coding

We can show an example of Huffman coding construction. The message we are going to encode if following:

```
THE SHELLS SHE SELLS ARE SEASHELLS
```

This message contains 8 symbols (S, E, L, “ ”, H, A, T, R). The message can be expressed with code words of 3bit length. Its whole length is $3 \cdot 34 = 102$ bits.

For Huffman coding we need to determine number of each symbol and their probabilities.

S	E	L		H	A	T	R
8×	7×	6×	5×	4×	2×	1×	1×
0.235	0.206	0.176	0.147	0.118	0.059	0.029	0.029

Table 9.3: none

There is used *binary tree*, it is a data structure often used in programming. The symbols are sorted by their frequency and then each symbol represents a tree leaf, and its weight is given by symbol occurrence. In first step join two leafs with the lowest weight, in our case T and R and create a node. The node weight is sum of weights T and R. In the next step join leafs or nodes with the lowest weight and proceed as long as there is only one node (the root of binary tree).

Now, go from the root toward leafs by the edges and each edge label by 0 or 1, if the edge goes up or down (in tree terminology to left or right subtree). The constructed tree with labeled edges see on **fig 9.2**. To find the code of each symbol pass all ways from the root towards the leaves. The path going along the edges of 0, 0 ends in S, the path going along 1, 1, 1, 0 ends in A.

Symbol	Code
S	00
E	10
L	010
	011
H	110
A	1110
T	11110
R	11111

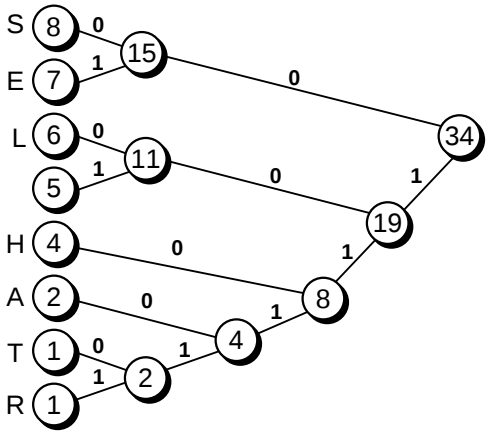


Figure 9.2: The results of Huffman encoding.

We see, that more frequent symbols with high probability of occurrence have shorter code than sporadic symbols. Our message after encoding:

```
11110110011010011001100100001 01
00110011010001100100001011110 11
11101101000111100011001100100 00
```

The message length was reduced from 102 to 93 bits. For decoding the binary tree **on 9.2** can be used again. We will start in the root and go along edges 1, 1, 1, 1, 0 until we arrive to lead, here symbol T, then we return to the root and go along 1, 1, 0 and we arrive to leaf H. By this way we continue until the whole message is decoded. Because Huffman coding is has unique *prefixes* for each code, and this prefixes is not start of another codeword the decoding can not do mistake.

Other compression algorithms using dictionary methods. These methods are based on fact that some words in the input file occur more frequently. Repeating words are stored in the dictionary. These words are replaced with their corresponding code words in output file. Among the representatives of this type of compression belongs *LZW (Lempel-Ziv-Welch)* as used in the ZIP compression or GIF or a variant of the TIFF formats.

9.3.3 Lossless data compression

Many applications needs for their requirements that data aren't impaired if they are compressed. E.g. for binary programs and data. Lossless compression has its justification in the field of computer graphics and image storage too. Lossy compression fits on "nature images" and photographs, but when it is used on a computer-generated graphics such as diagrams and charts, the image distortion is more noticeable on sharp edges and color gradients, even at low compress ratio (see **section 9.3.4.3**.).

Many compression algorithms were developed for lossless compression. A simple algorithm is for example *Run Length Encoding (RLE)*. This algorithm stores repeated bytes as their value and number. E.g. `AB AB AB CD EF EF EF EF EF` is stored as `03 AB 01 CD 05 EF`, so instead of 9 bytes should be only 6 stored.

Other types of algorithms are based on statistical methods. Before or during the compression process the algorithm determines the relative representations of elements of the file, and those repeated frequently are expressed as a short code word. Such algorithm is the Huffman coding described above. Also, Morse code is one of those codes, frequently recurring characters such as E (.) A (.–), I (..) have assigned shorter codes and the less frequent, such as H (....), J (.– – –), F (..–.) longer codes.

9.3.3.1 Portable Network Graphics

The PNG is appropriate graphics format with lossless compression. PNG was created to replace the outdated GIF format. PNG is not limited to a palette of

256 colors like GIF and allows to set a continuous level of transparency *(alpha-channel)* compared to GIF, which has the option to choose only two levels (yes or no transparency). If you want to save the lossless image just choose PNG.

The algorithm used in PNG is called *deflate*. This method is enhanced in some ways, the image lines are firstly processed by filter, which tries to find a similar neighborhood for each pixel. After processing there is a large number of data with zero value or a value close to zero (for same or similar values), so compression algorithms finds in data areas with same value so it can shrinks the length of the resulting file.

9.3.4 Lossy compression

The principle of lossy compression takes advantage of the processing equipment, in the case of the human eye it is unable to process certain information, so it actually would be an extra piece of information omitted.

A widely used method for lossy image compression format is *JPEG (Joint Photographic Experts Group)*. The JPEG is the standard established by the ISO and ITU, released in 1992 (later upgraded in 1997). A successor is upgraded format *JPEG2000*. It was developed by JPEG committee since 1995, was released in December 2000 and further revised in 2003, but it is not so widespread as its predecessor.

9.3.4.1 JPEG compression

JPEG usually does not use RGB color coding but use YCrCb, see chapter ??, because th human eye perceives brightness and colors with different sensitivity. The storage of YCrCb colors, mostly in the ration 4 : 2 : 0 reduces size of file, but itself is not enough. The image is further transformed, see schema in **fig. 9.3**.

In first step the image is divided on square block of 8×8 pixels and these 64 points is transformed from spatial domain $(x,\ y)$ to frequency $(i,\ j)$ by discrete cosine transform. Just for completeness, as follows:

$$\mathrm{DCT}(i,j) = \frac{1}{4}\,\mathrm{C}(i)\cdot\mathrm{C}(j)\sum_{x=0}^{7}\sum_{y=0}^{7}\mathrm{pixel}(x,y)\cdot\cos\left[\frac{(2x+1)\,i\pi}{16}\right]\cdot\cos\left[\frac{(2y+1)\,j\pi}{16}\right],$$

$$\text{where} \sim \mathrm{C}(a) = \begin{cases} \dfrac{1}{\sqrt{2}} & \text{for} \sim a = 0; \\ 1 & \text{in} \sim \text{other} \sim \text{cases}. \end{cases}$$

The first position $i = 0$, $j = 0$ holds *DC coefficient*, the mean value of the waveform for 64 values of block. The other positions contains *AC coefficients* and their value is derived from deviations between each values and DC coefficient. Basically, the DCT trying the block of 8×8 “to fit” a linear combination of shapes given byt the previous formula.

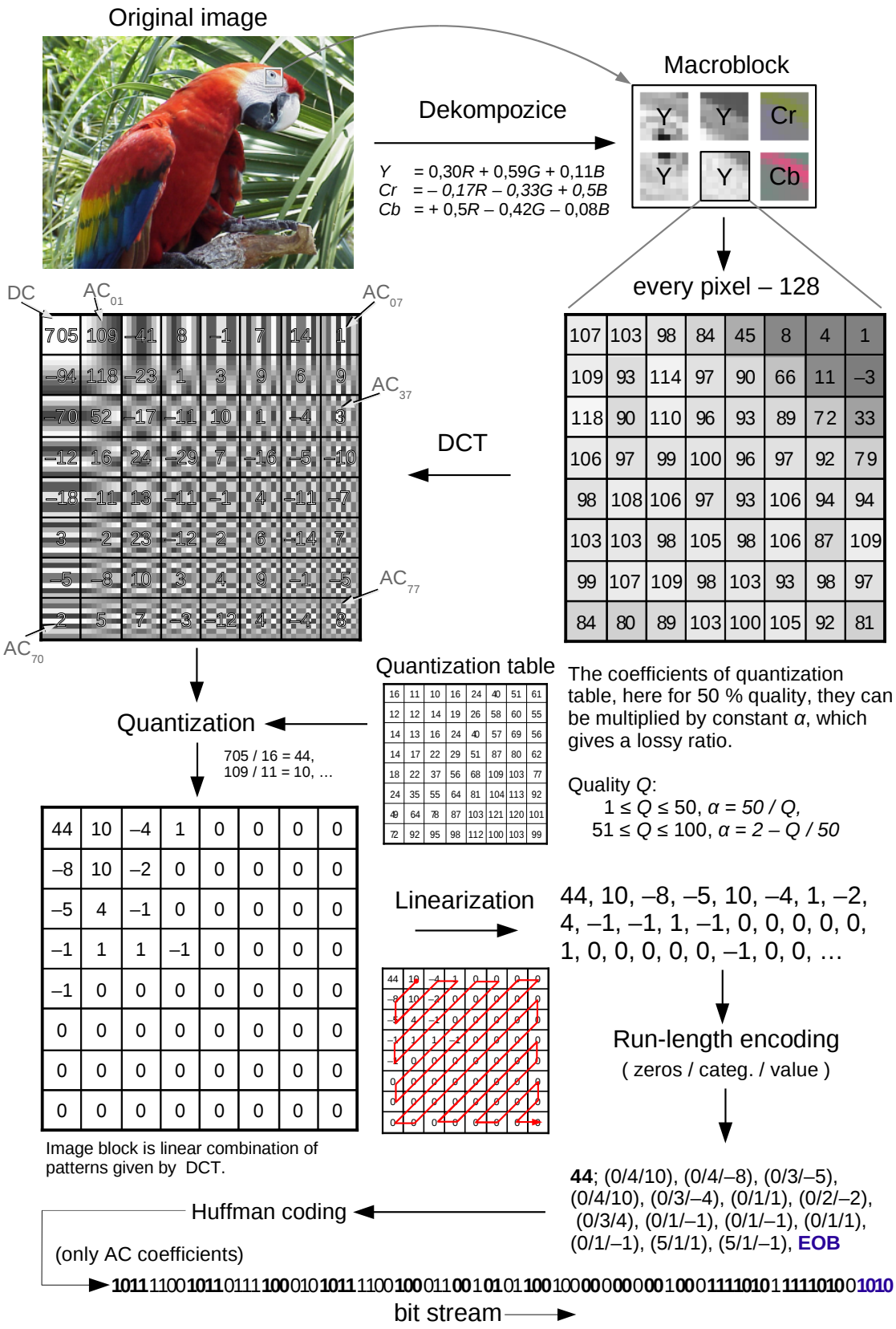


Figure 9.3: The JPEG compression for one 8×8 block of brightness.

Then follows a step that most affects the resulting image and a perception of the lossy compression level. The quantization is carried out by individual members of a predefined luminance quantization table (chrominance component has a different predefined table). A member of the block at position 00 is divided by a member 00 of the quantization table and the position of the integer part of number is stored at position 00, continues 01/01, 02/02,...up to each value is divided by its corresponding coefficient. The result of this process is a square matrix, where most information is stored in the upper left corner and around the lower right corner are just zeros.

This matrix is linearized into a sequence. Thanks to "zig-zag" reading the non-zero values appear in front of the sequence and remain part is filled by unnecessary zeroes.

Then the sequence is divided into categories, the first is DC coefficient and then other values continue and for each is determined following values: (number of preceding zeros / category / intrinsic value). The redundant zeros are reduces by RLE coding and from some place are presented only zeros. The all zeros are omitted and replaced by EOB (end of block) mark. DC coefficient, brightness and chrominance values have their codings.

For AC coefficients are zeroes labeled as category 0, for other integer values their categories is given by bit length of value. For most common AC coefficients $\{-1, +1\}$ it is 1, these two values can be represented by value 0 or 1, for $\{-3, -2, +2, +3\}$ is length 2, and it is represented by $\{00, 01, 10, 11\}$, for $\{-7, \ldots, -4, +4, \ldots, +7\}$ is length 3, etc. The result code depends on number of preceding zeros and bit length, so 0/1 (no zero/ length 1) has 00, 0/2 01, 0/3 100, 1/1 (1 zero/length 1) 1100, 5/1 1111010, etc. The results of Huffman coding for one block se in **fig. 9.3**.

We have an option to choose the image quality for JPEG files. For quality of 75 % the distortion is not noticeable in most cases and compress ratio can be around 20 : 1 to 25 : 1. The results of different quality for image with 256×192 seen in **fig. 9.4**. You can notice little distortion for quality about 50 %, mainly in areas with sharp color gradients.

Lossy compression of JPEG is not suitable for all types of images. It is good for "natural" images, but it is problematic for computer generated graphics like schematic diagrams, 3D renders, etc., where there are many sharp color gradients. The example of bad chose of compression see in **fig. 9.6**. The file size of both images is almost same. While for lossless PNG we cannot see any distortion, in the right image stored in JPEG format with compression set to the closest file size to PNG, we see that a DCT transform cannot handle sharp edges and the bias around them makes image heavily distorted.

There is also an option for storage data in *progressive mode*. In progressive mode, in first step the DC coefficients of all image blocks are transferred, then first AC coefficients, second AC coef., etc. This allows a low detail preview after receiving only a portion of the data and during a reception more and more details are displayed. The progressive mode is very useful for slow DSSTV transfer.

Quality 75%
15,970 B

Quality 40%
9,564 B

Quality 10%
4,277 B

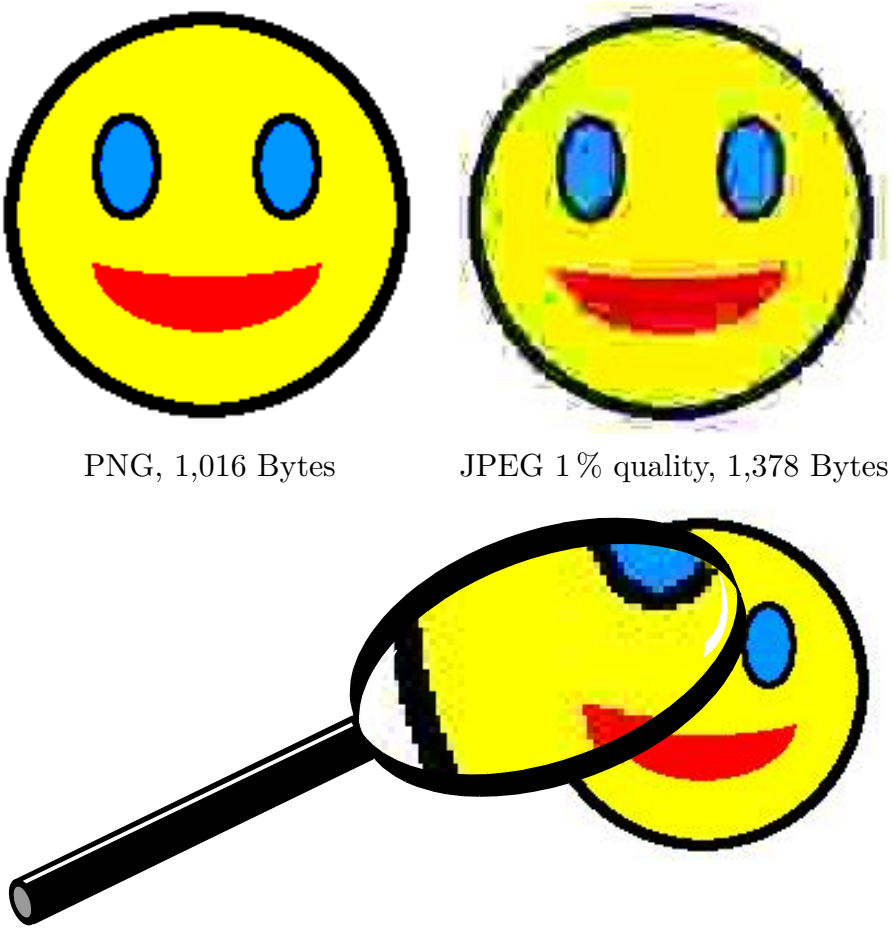
Quality 3%
2,474 B

Figure 9.4: The file size as result of JPEG compression loss degree.

Figure 9.5: The detail of image saved in 10% quality.

9.3.4.2 JPEG2000

When compared with original JPEG standard the new JPEG2000 has many improvements. There are used more sophisticate mathematical methods. The DCT is not used, but *discrete wavelet transformation (DWT)*. Wavelet transformation is one of methods for frequency domain representation of signal and has some



PNG, 1,016 Bytes JPEG 1 % quality, 1,378 Bytes

Figure 9.6: The detail of image with unproper compression.

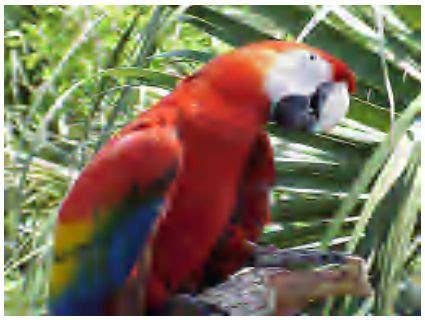
advantages over DCT. A functions with defined wave shape are used instead of sinuses and cosinuses.

Thanks to new transform method the compress ratio is better about 20 to 30 %. The images with sharp edges and color gradients have lower distortion.

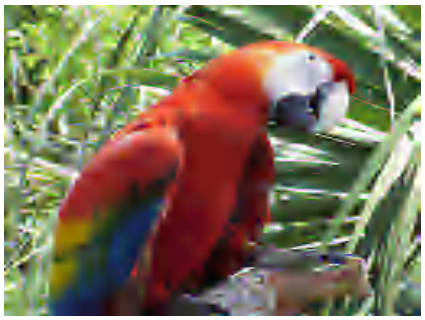
Users of new format appreciate the most a better compression ratio and higher image quality when using the lossy compression. DCT in JPEG format requires the division of the image into small 8×8 blocks, while JPEG2000 uses the whole image. The RGB color coding is used. And in addition the user has the choice to mark "area of interest". These areas are part of the image, where is required to set lower or higher compression ratio. For use in DSSTV is advantageous fault tolerance of the data stream. Only a small portion of the image displays poorly in the case of faulty transmission, other sections carried well are not affected. For older JPEG, the image part following the fault data of stream used to be completely discarded.

The new JPEG2000 has also progressive mode like an old JPEG. So the received image can be viewed during reception. You can see phases of reception in **fig. 9.8**.

Ratio 10 : 1
14,628 B



Ratio 20 : 1
7,310 B



Ratio 29 : 1
4,779 B

Ratio 50 : 1
2,909 B

Figure 9.7: File size depends on the compression ratio of JPEG 2000.

9.3.4.3 Lossy versus lossless image compression — conclusion

In JPEG section is described that lossy compression is not suitable for all types of images. Charts, diagrams and other images featuring sharp color gradients get significant loss, see **fig. 9.6**. Despite the significant quality loss the file size is not considerable reduced. inTable[tab:comparison] contains a comparison of file size for various formats. As the input file was used “smiling face” from **fig. 9.6** stored at a resolution of 256 × 192 in 16 colors.

Even relatively dumb RLE algorithm for lossless compression, but maintaining a 100% quality beats JPEG. It is the user's choice how to deal with the right choice of format and select a suitable compromise between resolution, number of colors and image quality.

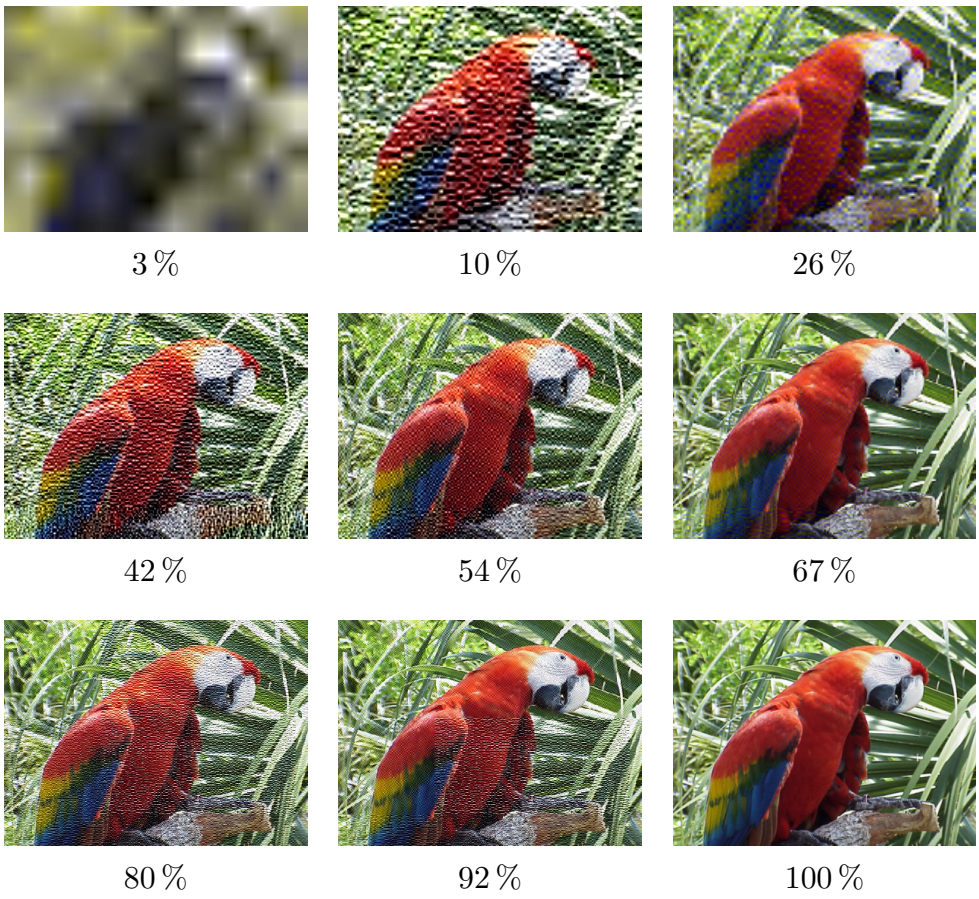


Figure 9.8: Progressive display of JPEG2000 image depends on amount of transfered data, original image has 400×298 resolution.

Format	Quality	File size
Windows Bitmap	100 %	24,654 B
JPEG	100 %	17,740 B
JPEG	75 %	7,300 B
JPEG	50 %	5,298 B
TIFF, PackBits compression	100 %	4,352 B
Windows Bitmap RLE	100 %	3,984 B
TIFF, komprese LZW	100 %	3,850 B
JPEG	25 %	3,766 B
GIF	100 %	1,569 B
JPEG	1 %	1,378 B
Portable Network Graphics	100 %	1,111 B

Table 9.4: Comparison of file sizes for different graphic formats.