ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
UDC/UDK 621.397.74:316.776.22

CODED IMAGES SENSITIVITY ON THE ERRORS IN THE COMMUNICATION CHANNEL TRANSMISSION

Časlav Livada, Irena Galić, Josip Job

Preliminary notes
This research attempts to analyze the effects of packet loss in the transmission through communication channel in the form of picture quality degradation. Errors are simulated using the three most common types of errors in the communication channel with the use of two types of entropy coders. On obtained, damaged pictures an objective image analysis is performed and the results are presented in tabular form.

Keywords: *communication channel, compression, entropy coders, errors, image analysis, picture quality*

Osjetljivost kodirane slike na pogreške u prijenosu komunikacijskim kanalom

Prethodno priopćenje
Ovo istraživanje pokušava analizirati posljedice gubitka paketa u prijenosu komunikacijskim kanalom u vidu pogrešaka na slici. Greške se simuliraju korištenjem tri najčešća tipa grešaka u komunikacijskom kanalu uz korištenje dva tipa entropijskog kodiranja. Na dobivenim, oštećenim, slikama provodi se objektivna analiza slike te se rezultati prikazuju u tabličnom obliku.

Ključne riječi: *analiza slike, entropijski koder, greške, kvaliteta slike, kompresija, komunikacijski kanal*

1 Introduction

This article investigates the problem of the impact of transmission errors in the communication channel on the quality of encoded images. Errors that can occur in the communication channel are applied on the coded image, i.e. a sequence of bits.

Errors that are analyzed in this paper are single bit errors, burst errors with and without interleaving and error in the AWGN channel (white noise).

For image coding and communication error simulation, MATLAB software package will be used with two encryption algorithms: Huffman and arithmetic coding.

2 Related Work

In [7] Li et al. presented a new paradigm for image transmission through analog error correction codes. They presented analog coding as a new and potentially very rewarding technology for transmitting images.

One approach to protect data from communication transmission errors is to start with a high-performance source coder, and protect its output from errors by adding redundancy [8, 9, 10]. Another method is to design resilience into the source coder so the effect of channel errors is reduced and less channel coding is necessary [11, 12, 13].

The excellent compression performance of the Set Partitioning In Hierarchical Trees (SPIHT) and Packetized Zerotree Wavelet (PZW) coder are explained in [11] and so is the Hybrid coder that shows better results than the latter.

3 Compression

According to [1], compression is used to reduce needed physical space for storing data using certain methods of data writing. The main unit for data storage is a file and depending on the data type within the file, some repetition is noticed. The data is stored once and after that, only the position of repeated data is stored. In this way, it is possible to decrease needed space for data storage, which depends on data type and structure.

There are two kinds of compression: Lossless and lossy compression (compression without and with certain data loss). Lossless compression is carried out by data compression algorithms which ensure that reconstructed data from compressed files corresponds with starting data. Lossless compression is used when it is needed to save the original data and typical examples on which the lossless compression must be used are executable files, source codes, textual data, etc. [2, 4].

Majority of programs that perform lossless compression divide compression into two steps and these steps are:

- Input data statistical model generation
- Input data can be transformed into sequences of bits by using the statistical model so the less occurring data give longer output sequences [5].

By using lossy compression, reconstructed data differs from input data, but those differences are so small and reconstructed data is still usable. According to [1] and [2] lossy compression methods are used for multimedia compression (audio, video and pictures).

Two main lossy compression schemes are:

- Lossy transformation codecs (coder - decoder) – Picture or audio samples are taken, divided into smaller parts, quantized and compressed using entropy coding [1].
- Predictive lossy codecs – preceding and/or following decoded data is used to assume current picture or audio sample. Difference between assumed and real sample with additional data is quantized and coded.

3.1 Used compression methods

This paper will consider lossless methods of compression because we wanted to focus on the communication errors analysis on images that are unchanged in compression process.

Lossless compression methods that were used are: Huffman coding and arithmetic coding.

3.1.1 Huffman coding

As per [2], Huffman coding codes symbols with variable length code words, depending on their probability of incidence.

Huffman coding is based on two simple facts:

1. In optimal code, symbols with greater probabilities of incidence have smaller code words than symbols with lesser probabilities of incidence.
2. In optimal code, two symbols with smallest probabilities have code words of same length.

Huffman coding algorithm is as follows alike [2]:

1. Sort symbols by falling probabilities.
2. Link two symbols with least probabilities into one new symbol and on branching assign "0" to one branch and "1" to second branch.
3. Sort obtained symbols by probabilities.
4. Repeat until there is only one symbol.
5. Going back through tree read codes.

Algorithm data structure is binary tree. Decoding algorithm uses the same procedure as the coder for building a tree and before sending data, coder must first send the probabilities of symbol incidence.

Let us take for example a five-character set {A, B, C, D, E} with known probability of occurrence p(A) = 0,16; p(B) = 0,51; p(C) = 0,09; p(D) = 0,13; p(E) = 0,11 (Σ = 1). For a uniform code, the length would be 3 bit/symbol and the optimal code length would be 1,96 bit/symbol. This example can be seen in Fig. 1.

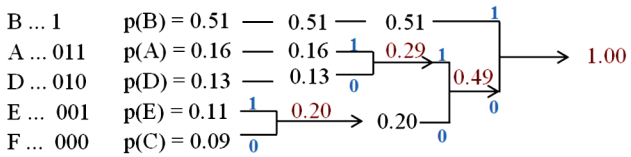
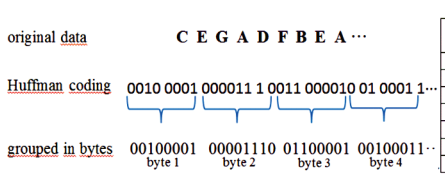


Figure 1 Huffman coding example

Coding is optimal if the probability values are ½, ¼, $1/2^n$. In real cases a near optimal encoding can be achieved but the optimality of the code depends on the probability of occurrence of symbols. Another example shows Huffman coding output and is shown in Fig. 2.



code table:

sign	probability	Huffman code
A	0.154	1
B	0.110	01
C	0.072	0010
D	0.063	0011
E	0.059	0001
F	0.015	000010
G	0.011	000011

Figure 2 Huffman coding output

3.1.2 Arithmetic coding

Arithmetic coding is a technique of conversion of sequences to a single integer. Algorithm takes queues of symbols as input data and transforms them into floating point number.

Arithmetic coding procedure as [2]:

1. Divide interval [0, 1] into M intervals that correspond to symbols where symbol length is proportional to symbols probability.
2. Select the interval of the first symbol in queue.
3. Subdivide current symbol interval into new M subintervals, proportional to their probabilities.
4. From these subintervals, select one that matches next symbol in queue.
5. Repeat steps 3 and 4 until all symbols are coded.
6. Output the interval value in binary form

The process of dividing the interval can be seen in Fig. 3.

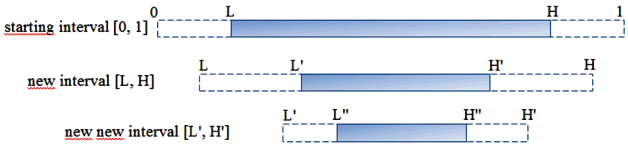


Figure 3 Interval division process

Example: Picture has 10000 elements. Statistical analysis gave the following element frequencies:
A : 200, B : 300, C : 100, D : 300,
E : 300, F : 500, G : 600, H : 200

Probabilities are as follows:
P(A) = 0,08; P(B) : 0,12; P(C) : 0,04; P(D) : 0,12;
P(E) : 0,12; P(F) : 0,2; P(G) : 0,24; P(H) : 0,08.

Fig. 4 shows arithmetic coding process.

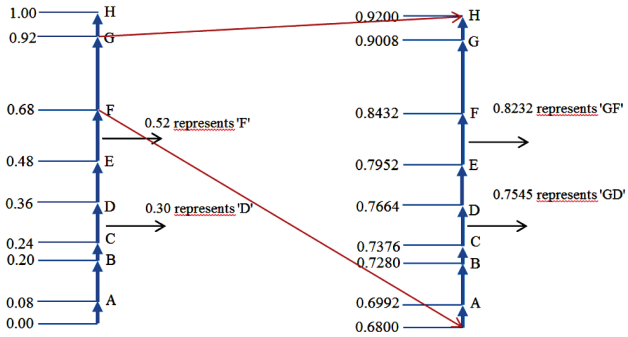


Figure 4 Arithmetic coding process example

4 Errors in communication channel

When data is sent through communication channel in the form of packets, packet loss and thus errors in received data can occur. On the way from source to destination packages can be affected by external interference [4]. This paper will analyze the following errors:

- Single bit errors
- Burst errors without interleaving
- Burst errors with interleaving
- White noise – errors in AWGN channel

4.1 Single bit errors

Single bit errors include changing bits from one to another since this is a transfer of bit sequences. If this error affects the bit "0", it turns into "1" and vice versa. In this paper we analyze the single bit errors in three different places in the random sequence of bits (at the beginning, i.e. the first third, at the middle and at the end), Fig. 5.

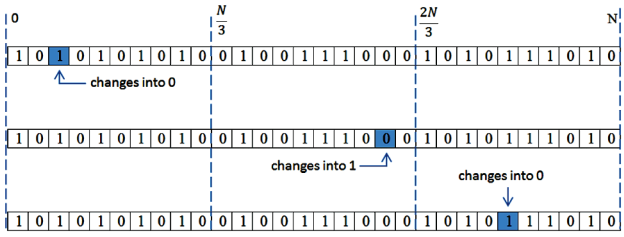


Figure 5 Single bit error occurrence possibilities

4.2 Burst errors

Burst errors are errors where the first and the last bit are corrupted and bits in between may or may not be corrupted. Bit changing rules apply as for single bit errors. This paper analyzes burst errors in three different places in the random sequence of bits (at the beginning, i.e. the first third, at the middle and at the end), shown in Fig. 6.

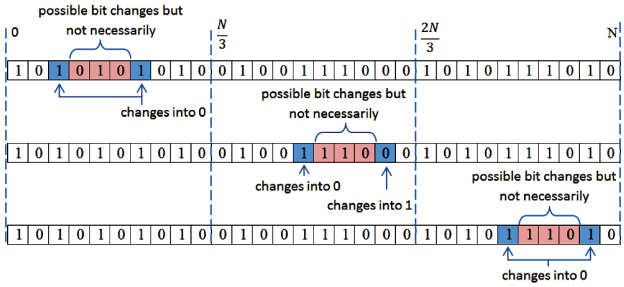


Figure 6 Burst error occurrence possibilities

Interleaving process can be applied on bit sequence affected by burst errors to break burst errors on several single bit errors. This procedure attempts to reduce the size of error made by burst errors in transfer and can be seen in Fig. 7.

original sequence	a	a	a	a	b	b	b	b	c	c	c	c	d	d	d	d
interleaving	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
transfer with burst error	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
received sequence after deinterleaving	a	a	a	a	b	b	b	b	c	c	c	c	d	d	d	d

Figure 7 Burst error with interleaving

4.3 White noise – errors in AWGN channel

The simplest model of the channel is AWGN (Additive White Gaussian Noise), i.e. a channel where only the white noise of constant power density spectrum with Gaussian distribution of amplitude is added to the input signal. This model does not include fading, interference, nonlinearity or dispersion of signal. White noise is added as shown in Fig. 8 which shows the Simulink model of AWGN channel. Other noise types, such as Poisson, Laplace and Lorentz have quite similar effects as the Gaussian noise so their results will not be shown.

First, it is necessary to modulate the input signal and for this study, BPSK modulation is used. BPSK (Binary Phase Shift Keying) [5] or 2-PSK modulation is two-phase discrete phase modulation in which the modulated signal can take two discrete phases. Phase of the modulated signal depends on the binary state of the digital modulation signals. By convention, phase of the modulated signal remains the same if the modulating signal is a binary one and it is changed by 180° if the modulating signal is a binary zero.

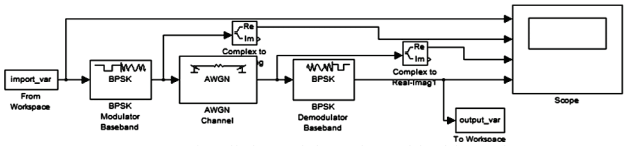


Figure 8 Simulink model used to add white noise

The advantage of BPSK is the least sensitivity to noise compared to the "M-phase" PSK modulations, where $M = \{4, 8, 16, ...\}$. Reason is that the modulated signal amplitude is constant and phase takes only two discrete phases which differ by 180°. In this way the so-called noise circles are relatively large around the vectors of the individual amplitudes, compared to M-phase PSK.

Lack of BPSK modulation is small spectral efficiency, whose theoretical maximum value is 1 bit/s/Hz. In practice, spectral efficiency is around 0,67 bit/s/Hz, so QPSK and 8-PSK are generally used.

5 Analysis of transmission errors impact on image quality

Image compression, application of the described errors on an encoded sequence of bits and image reconstruction will be performed in MATLAB (MATrix LABoratory). MATLAB allows matrices manipulations, data plotting, various algorithm applications, data reading from an external file and writing in the same. MATLAB contains an environment called Simulink that enables modeling, simulating and analyzing of dynamic systems. In this paper, Simulink is used to simulate the AWGN channel, while for other things MATLAB is used.

In this study, the image in BMP format is encoded in a sequence of bits that is sent over communication channel with the addition of errors. That changed sequence of bits is decoded and image with errors arises. This procedure is shown in Fig. 9.

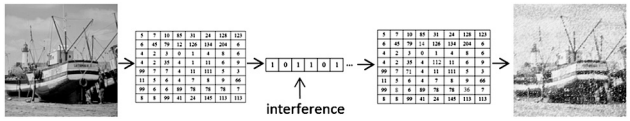


Figure 9 Process of introducing errors in coded sequence

The analysis will be conducted on two images with different sizes:

- Lena – 64 × 64 pixels
- Lighthouse – 512 × 512 pixels

In addition to subjective visual quality measurement, some statistical values will be measured as image quality indicators and these are [6]:

- Mean Square Error - MSE
- Peak Signal to Noise Ratio – PSNR
- Normalized Cross-Correlation – NCC
- Structural Content – SC
- Average Difference – AD
- Maximum Difference – MD
- Normalized Absolute Error – NAE

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j) - f'(i,j)]^2 \quad (1)$$

$$PSNR = 10\lg\frac{\left(2^n - 1\right)^2}{MSE} \quad (2)$$

$$NCC = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j)\cdot f'(i,j)]}{\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j)]^2} \quad (3)$$

$$SC = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j)]^2}{\sum_{i=1}^{M}\sum_{j=1}^{N}[f'(i,j)]^2} \quad (4)$$

$$AD = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j) - f'(i,j)]^2 \quad (7)$$

$$MD = \mathrm{Max}\left(\left|f(i,j) - \right|f'(i,j)\right) \quad (8)$$

$$NAE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j) - f'(i,j)]}{\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j)]^2} \quad (9)$$

where $f(i,j)$ is the original image and $f'(i,j)$ is the degraded image of $M \times N$ pixels.

The following table displays the values of metric for the image without degradation, i.e. $f(i,j) = f'(i,j)$.

Table 1 Metrics values of image without degradation

Metric	Value
Mean Square Error	0
Peak Signal to Noise Ratio	99
Normalized Cross-Correlation	1
Structural Content	1
Average Difference	0
Maximum Difference	0
Normalized Absolute Error	0

Analysis of errors in above mentioned images with Huffman and arithmetic coding follows.

5.1 Single bit errors

Fig. 10 shows the original images that are used in this experiment.

Fig. 11 shows reconstructed images of Lena from a bit sequence coded with Huffman coding and contaminated with single bit error, in Fig. 11a reveals an error on the hat, in Fig. 11c we can see an error right of Lena's face and in Fig. 11b an error that occurs in the middle of the sequence cannot be seen with naked eye on the reconstructed image. Images reconstructed from bit sequence made using arithmetic coding with added single bit errors can be seen in Fig. 11. The image is unrecognizable from the position of the bit change.

a) Lena b) Lighthouse

Figure 10 Original images

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 11 Lena – Huffman coding – single bit error

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 12 Lena – Arithmetic coding – single bit error

Table 2 Metrics for the image Lena and single bit errors

	Huffman coding			Arithmetic coding		
Metric	Beginning	Middle	End	Beginning	Middle	End
MSE	2,692	0,024	6,711	3523,287	2034,355	1282,9
PSNR	43,831	64,254	39,863	12,661	15,047	17,049
NK	1,000	1,000	1,000	0,887	0,901	0,953
SC	1,000	1,000	0,999	1,018	1,085	1,019
AD	0,026	0,002	-0,086	2,472	5,315	1,441
MD	105,000	10,000	40,000	164,000	164,000	164,000
NAE	2×10^{-4}	$1{,}9\times10^{-5}$	0,0009	0,332	0,189	0,118

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 13 Lighthouse – Huffman coding – single bit error

Figs. 12 and 13 show reconstructed Lighthouse images. It is difficult to spot errors on reconstructed images coded with Huffman coding at this image resolution.

Large scale reconstructed images (512 × 512 pixels) coded with arithmetic coding show the same error pattern as reconstructed Lena images, i.e. depending on the error occurrence, the next part of the picture is unrecognizable.

Table 3 Metrics for the image Lighthouse and single bit errors

	Huffman coding			Arithmetic coding		
Metric	Beginning	Middle	End	Beginning	Middle	End
MSE	0,049	0,002	0,170	4222,437	2048,327	1608,21
PSNR	61,258	75,872	55,836	11,875	15,017	16,067
NK	1,000	1,000	1,000	0,900	0,956	0,985
SC	1,000	1,000	1,000	0,943	0,964	0,935
AD	-38×10^{-5}	8×10^{-5}	3×10^{-4}	−2,728	−2,271	−3,793
MD	30,000	21,000	62,000	230,000	238,000	234,000
NAE	$7{,}2\times10^{-6}$	$6{,}9\times10^{-7}$	2×10^{-5}	0,386	0,186	0,149

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 14 Lighthouse – Arithmetic coding – single bit error

Numbers in Tabs. 2 and 3 are rounded to three decimal places. Tabs. 2 and 3 represent the objective image analysis for applied single bit errors. By observing the results shown in these two tables and reference values shown in Tab. 1 it can be concluded that the Huffman coding reacts much better to single bit errors than the arithmetic coding. Its results depend only on spatial coordinates of single bit error.

5.2 Burst errors

Due to easier comparison, in this section images with applied burst errors with and without interleaving will be shown. Fig. 15 shows images with errors due to the occurrence of burst errors without interleaving on coded sequence. Errors can occur at three random places, i.e. at the beginning, the middle and end. This study uses a 6 bits burst error (length of burst error is 6 bits).

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

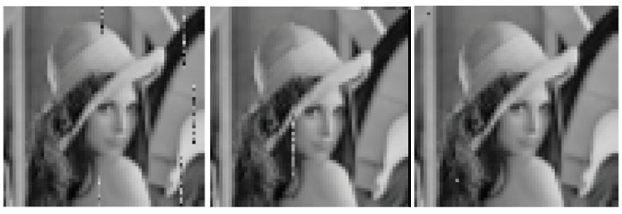
Figure 15 Lena – Huffman coding – burst error w/o interleaving

Observed errors are similar to single bit errors but, of course, a lot more pixels are damaged due to a larger number of error affected bits in coded sequence. In Fig. 15 errors are noticeable in Fig. 15a where burst error has affected the first part of the picture. Fig. 15b shows the effects when burst errors affect middle of the sequence. Errors in Fig. 15c are hard to notice, i.e. errors can be found around the mirror frame.

Fig. 16 shows the effect of burst errors on the image coded with arithmetic coding. After the occurrence of burst error, image is highly degraded - only black and white randomly positioned pixels can be seen. Error reduction by implementing interleaving has the exactly opposite effect, i.e. the image has more errors which can be seen by comparing Fig. 14 and Fig. 16 which show Huffman coding, but also by comparing Fig. 15 and Fig. 17 which show arithmetic coding. This is evidenced by Tabs. 4 and 5.

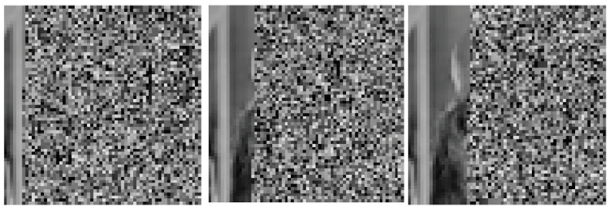
a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 16 Lena – Arithmetic coding – burst error w/o interleaving



a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 17 Lena – Huffman coding – burst error with interleaving



a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 18 Lena – Arithmetic coding – burst error with interleaving

Table 4 Metrics for the image Lena and burst errors w/o interleaving

	Huffman coding			Arithmetic coding		
Metric	Beginning	Middle	End	Beginning	Middle	End
MSE	37,619	442,894	2,437	3853,66	2169,37	1126,95
PSNR	32,376	21,667	44,260	12,272	14,767	17,611
NK	0,999	0,977	1,000	0,896	0,900	0,955
SC	0,999	1,019	0,999	0,979	1,075	1,022
AD	0,014	2,029	-0,031	-0,160	4,744	1,581
MD	124	211	0	164	164	164
NAE	$4{,}3\times10^{-3}$	54×10^{-3}	3×10^{-4}	0,372	0,203	0,107

Table 5 Metrics for the image Lena and burst errors with interleaving

	Huffman coding			Arithmetic coding		
Metric	Beginning	Middle	End	Beginning	Middle	End
MSE	70,148	474,472	3,515	3946,41	3296,33	3168,70
PSNR	29,671	21,368	42,670	12,168	12,950	13,121
NK	0,996	0,978	0,999	0,904	0,884	0,881
SC	1,002	1,014	0,999	0,960	1,038	1,053
AD	0,110	1,689	-0,006	-1,221	3,303	4,094
MD	158	211	72	164	164	164
NAE	$7{,}2*10^{-3}$	0,062	$3*10^{-4}$	0,387	0,320	0,302

Results in Tabs. 4 and 5 clearly show that the burst errors without interleaving achieve much better results in all objective image analysis parameters.

Figs. 19 ÷ 22 show the impact of burst errors with and without interleaving on an image of dimensions 512 × 512 pixels coded with Huffman and Arithmetic coding. Tabs. 6 and 7 give numerical representation of image quality degradation in which the image Lighthouse and burst errors with interleaving show somewhat better results.

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 19 Lighthouse – Huffman coding – burst error w/o interleaving

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 20 Lighthouse – Arithmetic coding – burst error w/o interleaving

Table 6 Metrics for the image Lighthouse and burst errors w/o interleaving

	Huffman coding			Arithmetic coding		
Metric	Beginning	Middle	End	Beginning	Middle	End
MSE	$9{,}7\times10^{-4}$	157,691	0,116	3995,1	2600,14	1694,2
PSNR	78,233	25,153	57,483	12,115	13,980	15,841
NK	1,000	0,995	1,000	0,906	0,927	0,979
SC	1,000	0,999	0,999	0,943	0,986	0,940
AD	$6{,}1\times10^{-5}$	$-2{,}6\times10^{-3}$	9×10^{-4}	-2,811	-0,921	-3,528
MD	16	220	59	232	236	230
NAE	$5{,}29\times10^{-7}$	0,039	2×10^{-5}	0,363	0,237	0,156

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 21 Lighthouse – Huffman coding – burst error with interleaving

It can be concluded that the reconstructed images on which interleaving was not applied are much better than those on which interleaving was applied. It is fair to conclude that the method of interleaving applied to coded sequence with corrupted bits is not suitable for use with lossless compression algorithms.

a) Error at the beginning of sequence b) Error in the middle of sequence c) Error at the end of sequence

Figure 22 Lighthouse – Arithmetic coding – burst error with interleaving

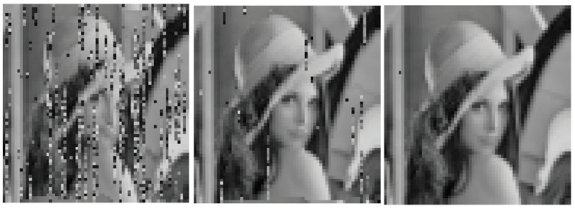
Table 7 Metrics for the image Lighthouse and burst errors with interleaving

	Huffman coding			Arithmetic coding		
Metric	Beginning	Middle	End	Beginning	Middle	End
MSE	70,957	21,591	151,94	5402,28	3626,85	3764,31
PSNR	29,621	34,788	26,32	10,805	12,535	12,374
NK	0,998	0,999	0,995	0,849	0,910	0,909
SC	0,999	0,999	0,999	0,968	0,958	0,951
AD	$-2{,}5\times10^{-3}$	$3{,}1\times10^{-4}$	3×10^{-4}	-1,601	-2,179	-2,497
MD	142	142	146	240	232	244
NAE	0,0261	0,008	0,055	0,492	0,331	0,343

5.3 White noise – errors in AWGN channel

The following figures show the images obtained by reconstructing the encoded images that passed through the AWGN channel with different SNR values.

Fig. 23 shows that the number of errors reduces by increasing SNR values. Fig. 24 shows that for the SNR values of 5 and 7, entire picture is changed depending on where the noise damaged the coded sequence. The damage was caused at the beginning and again the whole picture is changed from the first place of noise.

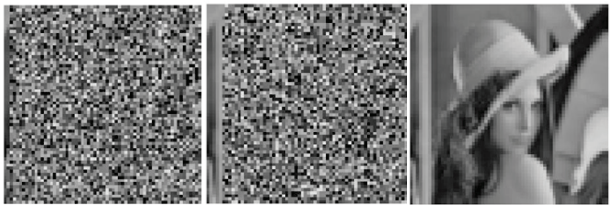


a) SNR = 5 b) SNR = 7 c) SNR = 9

Figure 23 Lena – Huffman coding – AWGN channel

Table 8 Metrics for the image Lena and white noise

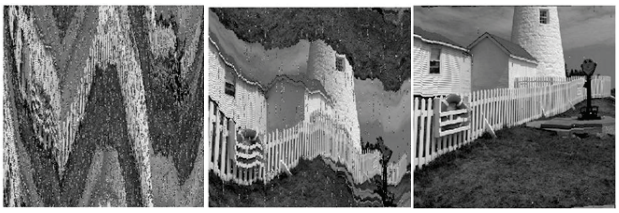
	Huffman coding			Arithmetic coding		
Metric	SNR = 5	SNR = 7	SNR = 9	SNR = 5	SNR = 7	SNR = 9
MSE	1298,453	444,553	2,344	4117,14	3838,47	0
PSNR	16,996	21,651	44,429	11,985	12,289	99
NK	0,961	0,989	0,999	0,884	0,899	1
SC	1,002	0,995	1,000	0,987	0,975	1
AD	0,154	−0,324	0,024	0,700	−0,350	0
MD	158	156	98	164	164	0
NAE	0,164	0,087	$1{,}9\times10^{-4}$	0,403	0,379	0



a) SNR = 5 b) SNR = 7 c) SNR = 9

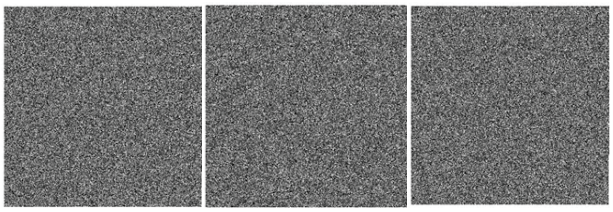
Figure 24 Lena – Arithmetic coding – AWGN channel

Fig. 25 shows the influence of white noise at the higher resolution image coded with Huffman coding. Large distortions can be seen on images with SNR values of 5 and 7, while image with SNR value of 9 has lesser distortions.



a) SNR = 5 b) SNR = 7 c) SNR = 9

Figure 25 Lighthouse – Huffman coding – AWGN channel



a) SNR = 5 b) SNR = 7 c) SNR = 9

Figure 26 Lighthouse – Arithmetic coding – AWGN channel

Table 9 Metrics for the image Lighthouse and white noise

	Huffman coding			Arithmetic coding		
Metric	SNR = 5	SNR = 7	SNR = 9	SNR = 5	SNR = 7	SNR = 9
MSE	5223,79	5071,64	748,45	5772,87	5773,57	5776,63
PSNR	10,950	11,079	19,389	10,517	10,517	10,514
NK	0,832	0,842	0,977	0,823	0,822	0,823
SC	1,014	1,003	0,999	0,997	0,999	0,997
AD	0,828	0,157	0,002	-0,173	-0,033	-0,145
MD	233	218	234	240	238	234
NAE	0,480	0,484	0,123	0,522	0,522	0,522

Tabs. 9 and 10 show that images coded with Huffman coding yield much better results in objective image analysis parameters as the SNR ratio increases whereas this rule does not apply to the images coded with arithmetic coding.

In addition to metrics for measuring image quality, coding and decoding execution time was also measured. The results are shown in Tab. 10. For image Lena, arithmetic coding is five times faster than Huffman coding and for image Lighthouse, it is seven times faster.

Table 10 Coding and decoding execution time (s)

	Coding method	
Image, pixels	Huffman	Arithmetic
Lena, 64 × 64	10	2
Lighthouse, 512 × 512	420	60

6 Conclusion

Huffman coding method proved to be better than arithmetic coding method in terms of sensitivity to errors in the coded sequence of bits. Reconstructed images that were coded with arithmetic coding were unrecognizable, i.e. from the location of the error (any type of error), only randomly distributed black and white pixels are visible. The following comments on the sensitivity analysis are related to Huffman coding.

In single bit errors, an error is more noticeable in smaller resolution image. On the image Lena it was easy to spot a mistake because the image resolution is 64 × 64 while on the image Lighthouse naked eye virtually cannot see the mistake.

Since the burst errors are approximately equal to the series of single bit errors, it is not surprising that the smaller image is more sensitive to this type of error. On the other hand, image Lighthouse is less sensitive to burst errors. Attempt to reduce the image degradation by implementing interleaving did not go so well, i.e. the reconstructed images were of lower quality than the reconstructed images on whose coded sequences interleaving was not applied.

As far as influence of the white noise in the AWGN channel is concerned, it can be concluded that its impact was smaller on the image Lena. In the most severe "pollution" of the sequence of bits at SNR of value 5, the image Lena, although much altered, is still recognizable while the image Lighthouse is not recognizable. By increasing the SNR parameter, images are less "contaminated", i.e. numerical values are closer to the ideal values.

Only parameter in which the arithmetic coding betters the Huffman coding is the execution time of the compression.

7 References

[1] Mengyi Pu, I. Fundamental Data Compression, Butterworth - Heinemann, 2006.

[2] Sayood, K. Introduction to Data Compression, Elsevier, 2006.

[3] Miano, J. Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP, Addison Wesley, 1999.

[4] Proakis, J. G. Digital Communications, McGraw Hill, 1994.

[5] Stallings, W. Data and Computer Communications, Prentice Hall, 2010.

[6] Kratochvil, T.; Šimiček, P. Utilization of MATLAB for Picture Quality Evaluation, Brno University of Technology, 2008.

[7] Liu, Y.; Li, J.; Xie, K. Efficient image transmission through analog error correction. // 13th IEEE Workshop on Multimedia Signal Procession, 2011, pp. 1-6.

[8] Kim, J.; Mersereau, R. M.; Altunbasak, Y. Error-Resilient Image and Video Transmission Over the Internet Using Unequal Error Protection. // IEEE Transactions on Image Processing, 12, 2(2013), pp. 121-131.

[9] Sherwood, P. G.; Zeger, K. Progressive image coding on noisy channels. // Proc. Data Compression Conf. '97, 1998, pp. 72-81.

[10] Chande, V.; Farvardin, N. Progressive transmission of images over memoryless noisy channels. // IEEE J. Select. Areas Communication, 18, (2000), pp. 850-860.

[11] Cosman, P. C.; Rogers, J. K.; Sherwood, P. G.; Zeger, K. Image transmission over channels with bit errors and packet erasures. // Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers. 2, (1998), pp. 1621-1625.

[12] Kozintsev, I.; Chou, J.; Ramchandran, K. Image transmission using arithmetic coding based continuous error detection. In Proc. DCC 1998, pp. 339-348.

[13] Penrose, A. J.; Dodgson, N. A. Error Resilient Lossless Image Coding, in Proc. ICIP 1999

Authors' addresses

Časlav Livada, dr. sc., dipl. ing.
J. J. Strossmayer University of Osijek
Faculty of Electrical Engineering
Kneza Trpimira 2B, 31000 Osijek, Croatia
E-mail: caslav.livada@etfos.hr

Irena Galić, doc. dr. sc., prof.
J. J. Strossmayer University of Osijek
Faculty of Electrical Engineering
Kneza Trpimira 2B, 31000 Osijek, Croatia
E-mail: irena.galic@etfos.hr

Josip Job, doc. dr. sc., dipl. ing.
J. J. Strossmayer University of Osijek
Faculty of Electrical Engineering
Kneza Trpimira 2B, 31000 Osijek, Croatia
E-mail: josip.job@etfos.hr