ERRORS OF IMAGE COMPRESSION BY THE UAV COMPUTER BY DIFFERENT METHODS IN REAL TIME

Andrii Levchenko, Ilnara Sharipova, Yurii Shugailo,
Yurii Bercov, Hanna Korenkova, Oksana Zui

Abstract. The paper presents a comparative analysis of the effectiveness of image compression methods. A preliminary review of possible problems of representing numbers in binary code in the implementation of software in high-level programming languages. The paper summarizes the situations when the features of the representation of numbers in the computer significantly affects the results of data processing in general and images in particular. Test images of various degrees of saturation tested in known works were used as test data for the analysis of the efficiency of compression methods. Arrays of numerical data of test digital images were obtained with the help of on-board cameras of modern small-sized non-commercial drones. It is concluded that in the case of describing the original numerical data as well as Double the implementation of compression procedures in the C++ programming language. This choice is the best for further practical implementation image compression by the UAV computer for different methods. Image compression by the UAV computer in real time can be used by method with zonal-threshold selection of coefficients of fast two-dimensional Haar transformation.

Keywords: errors of image compression, programming languages, numerical data, compression method, image transformation.

Introduction and statement of the problem

The aim of image compression is to solve the problem of reducing the amount of data required for digital presentation. The basis of this process is the removal of redundant data from the original array, which reproduces the image in digital view. From a mathematical point of view, this is equivalent to converting a two-dimensional data array into a statistically uncorrelated array. The compression conversion is applied to the original image before it is saved or transmitted. Subsequently, the compressed data format is unpacked and the original image or some approximation of it is restored.

The main reason for the problems of image compression in digital data processing systems is the victory of the computing power of modern processors over the capacity limitations of storage and data transmission systems. Compression can reduce the load on data channels by converting them, which creates a load on the computer processor, because data compression is associated with large amounts of mathematical calculations.

It seemed that by increasing the bandwidth of network equipment and creating high-power computer data processing systems, the problem of limited computer resources will be a thing of the past, but social network users and technological progress are constantly evolving, leading to increased computing and restoring the problem of computer resource constraints. various conditions. For example, in the case of computer image processing by a drone, there are weight limitations. This does not allow engineering to use modern high-performance systems with high-performance processors as an on-board drone computer.

An overview of the errors in the presentation of the original data by the on-board computer

Preliminarily consider a simplified scheme for presenting graphic information in a computer (Fig. 1) image field is covered with an 8x8 grid. Compare bit 0 to those fields that are free from the image and 1 to those that are covered by image elements. If each line is interpreted as a binary number, the image can be written (encoded) in a sequence of eight decimal numbers: 0, 4, 132. 254, 254, 4, 4, 0, or in hexadecimal: 00, 04, 84, FE, FE, 04, 04, 00 and store in computer memory as bytes. Representation of numerical values in the form of bytes is the most rational in connection with features of architecture of memory chips.

The main format for presenting data in modern computers is the format for presenting binary numbers in a floating point. The format for representing binary numbers in a floating point is introduced by the IEEE754 standard [1, 2].

In engineering and programming, as a rule, 32 and 64 bit formats are used. In VB, numerical data types are represented as single (32 bits) and double (64 bits), and in C, C ++ – float (32 bits) and double (64 bits). The rest of the IEEE754 number formats are an extended version of single-precision.

0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0

Figure 1. A fragment of a digital image in binary form

To represent a number in the single-precision IEEE754 format, you must convert it to binary normalized form. To describe the procedure for converting numbers to 32-bit IEEE754 format, it makes sense to consider the appropriate steps:

– 1 bit is reserved for the sign of a number: 0-positive, 1-negative. This is the most significant bit in a 32-bit sequence;

– a bits of exponential representation follow, 1 byte (8 bits) are allocated for this. The exponent can be, like a number, with a + or – sign, although the meaning of these signs for the number, exponent and mantisa is different. To determine the sign of the exponent, so as not to introduce another sign bit, add a half-byte offset to the exponent. The offset exponent is written to the allocated 8 bits;

– the remaining 23 bits are reserved for the mantissa. But, in a normalized binary mantissa, the first bit is always 1, since the number lies in the range $1 <= M < 2$. There is no point in writing one to the allocated 23 bits, therefore, the remainder of the mantissa is written to the allocated 23 bits.

The numbers presented in the IEEE754 format represent a finite set on which an infinite set of real numbers is displayed. Therefore, the original number can be represented in IEEE754 format with an error. The graphical view of the number representation accuracy error function is presented in [2].

The absolute maximum error for a number in IEEE754 format is within half a step of the numbers. The step of the numbers doubles as the exponent of the binary number increases by one. That is, the farther from zero, the wider the step of numbers in the format IEEE754 on the numerical axis. The step of the denormalized numbers is $2^{(E-149)}$ (*Single*) and $2^{(E-1074)}$ (*Double*). Accordingly, the maximum absolute error limit for representing the original decimal number will be 1/2 of the number step: $2^{(E-150)}$ (*Single*) and $2^{(E-1075)}$ (*Double*).

The relative error expressed as a percentage will be: $(2^{(E-150)}/F)*100\%$ (*Single*) and $(2^{(E-1075)}/F)*100\%$ (*Double*).

The step of the normalized numbers is $2^{(E-150)}$ (*Single*) and $2^{(E-1075)}$ (*Double*). Accordingly, the limit of max. the absolute error will be equal to 1/2 of the number step: $2^{(E-151)}$ (*Single*) and $2^{(E-1076)}$ (*Double*). The relative error in% will be: $(2^{(E-151)}/F)*100\%$ (*Single*) and $(2^{(E-1076)}/F)*100\%$ (*Double*).

Thus, the main set of numbers in the IEEE754 format has a stable small relative error.

The maximum possible relative error for the number Single is $2^{-23}*100\%=11{,}920928955078125e^{-6}\%$. The maximum possible relative error for the number *Double* is 2^{-52}

$$*100\%=2{,}220446049250313080847263336181 6e^{-14}\%.$$

For the case of image compression by the UAV computer by different methods in real time, this means that there is a real threat of replacing the original matrices of numerical data with close to them with completely different conditionality numbers. Accordingly, small details of the original image may be lost. The task of data processing itself acquires a different nature and passes into the area, which is called incorrect tasks.

It should be noted that the IEEE754 standard provides four ways to round numbers:

- rounding tending to the nearest whole;
- rounding tending to zero;
- rounding tending to + ∞;
- rounding tending to – ∞.

When converting numbers, you must select one of the rounding methods. By default for software products, this is the first method – rounding to the nearest whole. Often in various devices use the second method – rounding to zero. When rounding to zero, you just need to discard insignificant digits of the number, so this method is the easiest in the hardware implementation [2, 3, 4].

The IEEE754 standard is widely used in engineering and programming. Most modern microprocessors are manufactured with a hardware implementation of the representation of real variables in the IEEE754 format. The programming language and the programmer cannot change this situation, there is no other representation of a real number in the microprocessor. When the IEEE754–1985 standard was created, the representation of a real variable in the form of 4 or 8 bytes seemed very large, as the amount of MS-DOS RAM was 1 MB. The program in this system could

use only 0.64 MB. For modern operating systems, the size of 8 bytes is negligible, however, the variables in most microprocessors continue to be represented in the format IEEE754–1985 [2, 4].

Currently, educational resources on the Internet present a lot of material on the features of the representation of numerical data in computers. However, in the opinion of the authors, the most complete and demonstrative from the point of view of estimating the values of transformation errors of numerical data since 2012 remains [2, 4]. As part of the solution of a number of applied problems, the authors considered some features of the representation of numbers in computer systems of technical diagnostics, navigation problems. An example of such work is [5, 6, 7].

To achieve the objectives of this work, it is advisable to summarize the known facts of computer computational errors caused by the use of numbers in the IEEE754 format.

Errors related to the accuracy of representation of real numbers in IEEE754 format. "Dangerous reduction". This error is always present in computer calculations. The reason for its occurrence is due to the lack of significant discharges. The relative error value associated with the accuracy of the representation has a dimension for single 10^{-6} for double 10^{-14}. The values of the absolute errors can be significant, for a maximum of single 1031 and double 10292, which can cause certain computational problems.

Interesting examples of specialized error calculations related to the application of the IEEE754 standard are translated in [2, 4]. Thus, for the case of errors related to the accuracy of the representation of real numbers, an example of computer and mathematical calculation of the representation of numbers is given.

For test examples, the relative error of the original numbers is approximately $3.24e^{-6}$%. As a result of one operation, the relative error of the result became 800%, ie. increased by $2.5e^{+8}$ times. The author of this work called such errors: "Dangerous reduction". The term refers to a catastrophic decrease in the accuracy of calculations in an operation where the absolute value of the result is much less than any of the input values of variables in computer calculations.

As a rule, modern programmers do not take into account the possibility of these errors when using high-level programming languages. However, they can significantly affect the final results of the calculations, and the user of the software product will sincerely believe in the correctness of the results [5].

Errors related to incorrect data type casting. "Wild mistakes". The errors are caused by the fact that the original number presented in single and double format is usually not equal to each other. For example [2]: source number 123456789,123456789, with: *Single*: 4CEB79A3=+123456792,0 (*dec*), *Double*: 419D6F34547E6B75=+123456789,123456791043281555175781 25. As it is not difficult to determine the difference between *Single* and *Double* will be: 2,87654320895671844482421875.

Therefore, it is fair to conclude that the variables and intermediate results of computer calculations should be reduced to one type of data, regardless of the expediency of different descriptions of variables in terms of saving memory in computations. Interestingly, similar requirements are imposed on software products and programming language developers (eg: C: ISO/EC9899: 1999). It should be noted that it is not enough to simply reduce all source data to one type. It is also necessary to reduce the results of intermediate operations to one type [2, 4].

The same source points to the fact that the result of the calculations may depend on the type and version of the compiler of a particular language. For test examples, the conclusion is made that the VC ++ compiler automatically lists the types of variables, and an attempt to force them to one type ends in a compilation error.

Calculation errors caused by failure to specify data types in [2] are proposed to be called "wild mistakes". These errors in the implementation of software products in high-level programming languages are due to ignorance of standards and programming theory (ie, poor basic education). That is why we support the use of the word "mistakes" when referring to such errors in the implementation of software products.

Mantiss shift errors. "Cyclic holes". These errors are due to the loss of accuracy of the result when the mantissa numbers on the numerical axis do not intersect. If the mantises of numbers do not intersect on the numerical axis, then the operations of addition and subtraction between these numbers are impossible.

According to the results of test examples, the shift error occurs if the original normalized numbers have different exponents. If the numbers differ by more than 2^{23} (for single) and 2^{52} (for double), then the operations of addition and subtraction between these numbers are not possible.

The maximum relative error of the operation result is approximately 5.96^{e-6}%, which does not exceed the relative error of the number representation.

The value of the relative error is small, but there are problems with programming. First, at the specified numerical values of the initial data it is possible to work with numbers only in a narrow range of a numerical axis where mantises intersect. Second, for each initial number, there is a cycle execution limit called a "cyclic hole". That is, if there is a cycle in which the original number is summed to the sum, then there is a numerical limit of the sum for this number. The sum, having reached a certain value, ceases to increase from its addition to the original number.

An example of a "cyclic hole" in an automatic control system is given in [2].

Rounding errors. "Dirty zero". In computer calculations, two types of rounding are always implemented. Either the numerical value of the performed arithmetic operation is rounded as a result of calculations and features of data representation or rounding occurs with the output and input of a real number in the window of operating system.

In the first case, the variable is rounded by one of the 4 types of rounding IEEE754, by default rounding occurs to the nearest whole. In this case, the variable takes a new rounded value different from the actual value.

In the second case, Windows has the following problems [4, 8]; there is no technical possibility to display or enter exact values of variables in windows of the graphic interface of the user, there are serious errors like "dirty zero".

Source [8] points to the fact that this kind of error often occurs in the "operator-machine" interface for various commercial applications in high-level programming languages. As a confirmation, this source indicates a fragment of the interface of the "logistic weight program" when resetting the container.

"Dirty zero" errors are situations where the numeric value of a variable is not actually zero in the program code. Whereas when creating programs, a variable equal to zero is considered according to mathematical relations. The relative error of the result is infinity.

In logical comparison operations, this non-zero can lead program execution to another branch of the algorithm. If for cases of presence of the interface of the operator there is a possibility to react to emergence of an error, for cases of automated processing of digital data such possibility is absent. This case includes image compression by the UAV computer by different methods in real time.

Errors at the norm/denor limit of the number. "Numbers –net-killer". These errors occur when working with numbers that are on the border

of the normalized representation of numbers. They are related to the difference in the representation of numbers in the IEEE754 format and in the difference in the formulas for translating the IEEE754 format into real numbers. That is, devices (or programs) must use different algorithms depending on the position of the real number on the numerical axis of the format. In addition to complicating devices and algorithms, there is also uncertainty in the transition zone. The uncertainty of the transition zone is that the standard does not specify a specific value of the transition boundary. In fact, the transition boundary is between two real numbers.

Since the boundary is a real number, its accuracy can be set indefinitely and the digital device or program does not have enough bit to decide to which range to assign this number.

In [2], as an example is the bug № 53632 for PHP, which caused panic in early 2011.

Entering the number $2.2250738585072011e^{-308}$ caused the process to hang with almost 100% CPU usage. Other numbers in this range did not cause problems. The same source indicates that the message about the bug was received on 30.12.2010, the message about its correction by the developer was received on 10.01.2011. Since the PHP preprocessor is used by most servers, any network user had the opportunity to "cut down" any server within 10 days. According to the developers, this bug only works in 32-bit systems. It is suggested that if you increase the accuracy of the limit number, modern 64-bit systems will also hang. Although for commercial reasons, the developers deny this possibility. The reason for the denial is clear: any user of the network, with a certain level of persistence, has the potential to "cut down" most of the information resources of the world community within ten days.

Comparison of image compression methods

Errors of image compression by the UAV computer by different methods in real time. The description of the problem of presenting numerical data led to restrictions on further comparative analysis errors of image compression by the uav computer by different methods in real time.

As models of image compression and recovery procedures, their software implementations in the C++ programming language were used. The elements of the test images were represented by 32-bit code. Both original

and restored images were recorded. The evaluation of compression methods for efficiency was carried out on the basis of private indicators, which include:

– σ – is the standard deviation of the restored image;

– $K_{+/-}$ – is the number of addition/subtraction operations when performing compression-recovery operations;

– $K_{m/d}$ – is the number of multiplication / division operations when performing compression-recovery operations;

– t_t – is the time of transformation of the original image (from the moment of reading the original image to writing to the file of the obtained compressed sequence);

– K_t – is the compression ratio of the original image.

The modified method of arithmetic coding with a fixed model of image compression (szsdv&iqrp) [10, 11] was tested in the following variants: with (szsdv&iqrp+) and without (szsdv&iqrp-) subquantization and using zonal-threshold selection at different values of the image quality restoration parameter (szsdv & iqrp).

The original images were used in the standard 4: 3 format with a dimension of 640 × 480 image samples, which allowed when dividing the image into 8 × 8 blocks to obtain an integer number of blocks and avoid the recovery error introduced by the addition of samples of the original image.

When performing research, a comparative evaluation of image compression efficiency [12, 13, 14, 15] for varieties of the method with zonal threshold selection and compression methods widely used in practice in image processing, such as:

– JPEG [16, 17], which includes two-dimensional discrete-cosine conversion of samples of the original image;

– JPEG-2 [15], which includes wavelet transform and statistical coding when changing the parameter "compression" in the range from 0 to 100;

– WI, which includes performing wavelet transform and statistical coding when changing the parameter "compression" in the range from 0 to 100.

In the general case, the transformation of the image by the algorithms of the JPEG method involves the following steps:

– change of color space of representation of RGB samples on YCrCb;

– two-dimensional discrete-cosine transformation of samples of the original image;

– quantization of the transformant in accordance with the selected parameter "compression" in the range from 0 to 100;

– coding of series lengths (RLE) of the obtained sequences;
– Huffman statistical coding;

In the methods JPEG, JPEG-2, the "compression" (Z) parameter is used to control the compression ratio.

A separate comparison is made with image compression methods, which implement procedures similar to those used in the compression method with zonal-threshold selection. Such methods include compression methods based on the zonal selection of the coefficients of two-dimensional discrete-cosine transformation (2-ddcost) [19] and the zonal selection of the coefficients of fast integer two-dimensional Haar transformation (zscfastint2-dtHaar).

The analysis of the values K_t and σ obtained when processing images with different degrees of saturation, depending on the values of Z, showed that the analyzed compression methods provide σ of less than 3% with Z values less than 70, 60 and 50 when processing low-, medium- and highly saturated images respectively.

Compression ratios for this value σ range from 3 to 25 times for the JPEG method, from 5 to 35 times for JPEG-2000, and from 5 to 30 times for WI. Note that when using the WI compression method at compression values of 50 and 70, significant compression ratios were obtained (of the order of 50–80 times), but the σ value in this case increases 2–3 times in relation to the σ values of the JPEG methods. The analysis of the results of image compression shows that the JPEG-2000 method was the most effective in terms of σ and K_t.

Analysis of indicators for compression methods 2-ddcost, zscfastint2-dtHaar showed that the methods provide σ less than 3% at values of $W_S \geq 5$ when processing highly saturated images and at values $W_S \geq 3$ for low- and medium-saturated images. Compression coefficients for such an σ value lie in the intervals: from 1 to 12 times for the 2-ddcost method, from 1.8 to 18 times for zscfastint2-dtHaar, from 1.8 to 23 times for the szsdv&iqrp method.

With $WS = 1$ and 2, the RMS value increases sharply, so it is advisable to use such values of the quality parameter only to obtain large compression ratios. This conclusion confirms the results for the known test data [11], however, it requires verification for other images, where the numerical values of the intensity of the color components have different sizes.

Analysis of the nature of the change in compression ratios shows that when processing "low-saturated" images, the szsdv&iqrp method [10]

provides less compression at WS = 5–10 than the 2-ddcost compression method with zonal selection. However, when using the 2-ddcost method, much worse σ values were obtained for szsdv & iqrp method.

In [11], this is due to the presence of a large number of "natural" zeros, i.e. zero values of the conversion coefficients before selection. We can conclude that the selected test images are inadequate for the totality of all possible situations. When processing medium and highly saturated images, the szsdv & iqrp method provides high values of K_t.

In a comparative analysis of compression methods, the ratio of the obtained K_t and σ is of greatest interest. The main tasks of image processing in the interests of object recognition, therefore, will analyze those settings of compression methods at which the σ value takes on a value of no more than 2%. The analysis shows that the JPEG-2 compression method is characterized by the best K_t / σ ratio. In addition, it should be noted that the maximum values of the compression ratios for such methods as JPEG, JPEG-2, WI are significantly superior to other compression methods.

Since the evaluation of the compression parameters – K_t and σ for images with different degrees of saturation is carried out using a limited number of experiments (n = 900), then the error of the method itself is introduced into the determination of these parameters.

The values of the probability that the error will not exceed in the assessment of K_t and σ – P are presented in table 1.

Table 1. – Values of the probability P for the investigated compression efficiency indicators for the test case

K_t			σ		
p = 0,95	q = 0,05	n = 900	p = 0,9	q = 0,1	n = 900
P = 0,99			P = 0,95		

The values given in the table based on the results [11] are estimates, since one value of the number of tests is taken into account. The dependence of the studied parameters of image compression on the number of results is subject to further study.

Computational complexity is an important parameter for comparing image compression methods that characterize the time of their implementation. To determine its value, two approaches are used: an estimation of the number of elementary operations performed (addition, subtraction, multiplication, division, etc.) or the execution time of the method procedures.

When using the first approach, addition/subtraction and multiplication/division operations are considered separately. This is due to the fact that to perform multiplication/division operations, more clock cycles of computational elements are required and their technical feasibility is more complicated. So, when processing images by the analyzed methods, it is necessary to perform:

When using the first approach, addition/subtraction $K_{+/-}$ and multiplication/division $K_{m/d}$ operations are considered separately. This is due to the fact that to perform multiplication/division operations, more clock cycles of computational elements are required and their technical feasibility is more complicated. So, when processing images by the analyzed methods, it is necessary to perform:

14680064 operations addition/subtraction and 16777216 operations multiplication/division when using the method based on the 2-ddcost transformation;

3670016 operations addition/subtraction and 2097152 operations multiplication/division when using the method based on the zscfastint2-dtHaar transform method;

5155016 addition/subtraction operations and 3582152 multiplication/division operations when using the szsdv & iqrp transform method.

Thus, the computational complexity of the image compression method szsdv&iqrp is 5 times less than that of the method based on 2-ddcost transformation, but about 1.5 times more than with zscfastint2-dtHaar transform. This is due to the need to carry out additional calculations to find the threshold of the breeding. However, the proposed method provides higher compression ratios with lower σ compared to the zscfastint2-dtHaar transform method.

An analysis of the computational complexity of image compression methods from a practical point of view, in terms of the number of operations performed, is not advisable and is of interest in studying the computer implementation of mathematical methods.

Evaluation of the effectiveness of compression methods from the point of view of the expediency of using image compression by the computer UAV in real time procedures will be carried out according to the value of the time for transforming an image into a file. By the time indicator of compression methods we mean the time spent on the execution of compression procedures [11].

The time depends on the computational complexity of the compression procedures used, on the performance of computational elements and

storage devices, as well as on the efficiency of the program code that implements the algorithm in the software implementation of the compression method [19]. In (Fig. 2) shows the results of a comparative analysis of the compression time of images by the known algorithms JPEG, JPEG-2, WI and software implementation szsdv&iqrp transform method.

It should be noted that the szsdv&iqrp procedures for arithmetic operations are integer. In terms of the study errors in the presentation of the original data by the on-board computer, this is a significant advantage.

At the same time, according to the sources [3, 9, 10, 11], allows obtaining at $\sigma < 3\%$:

- for a highly saturated image $K_t = 3.5$ times;
- for the medium-saturated image $K_t = 7$ times;
- for the low-saturated image $K_t = 23$ times.

The use of subquantization of coefficients in direct transformation according to the Haar basis makes it possible to increase the compression ratio by reducing the dynamic range of the conversion coefficients in relation to the range of samples of the original image. However, this leads to a significant increase in the values of the standard deviation. At sub-quantization levels $V = 1$ or 2, the compression ratio increases by 10–25% due to a decrease in the alphabet of the encoded source.

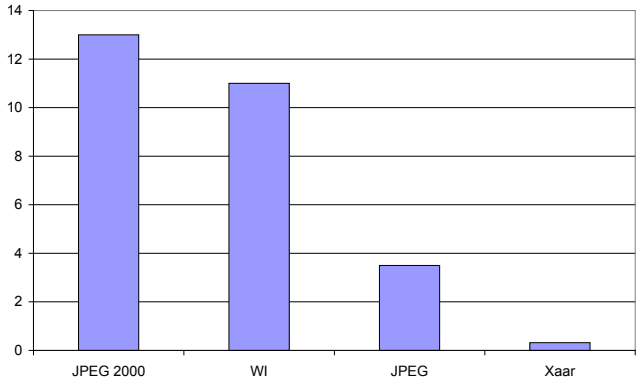


Figure 2. Time of images compression for different methods

It should be noted that the proposed recommendations for choosing the values of the parameters of the developed compression method make it possible to minimize the visibility of introduced distortions in the processed images when reaching relatively low compression ratios 5.5–10 times.

Conclusions

1. It is false to believe that floating-point calculations are within the relative error of representing the largest number. The magnitude of errors is cumulative. Numerical values are added to each other. The postulate of metrology on mutual error compensation during data averaging does not work in the case of computer processing of decimal data.

2. When processing data by programs written in high-level programming languages with converting values from decimal to binary number system, the tendency of the averaged value of the measurement results to the true value of the physical quantity does not occur. Errors such as dirty zero and dangerous reduction make the amount of computational error unacceptable. When programming computer calculations, special attention should be paid to the transformation of values close.

3. The main job of programmers is to deal with errors, and not in mathematical solutions to the problem. The programmer needs to calculate the error of the result in each cycle in order to take it into account in the next cycle. The programmer must be absolutely prepared for the fact that some basic concepts of mathematics are not carried out in calculations in the IEEE754 format. For example, the rules of algebraic commutativity $(a + b) + c = (a + c) + b$ are usually not met in such calculations. Unfortunately, in modern fundamental mathematics education, little attention is paid to this issue, leaving the solution of the toolbox problems to the applied programmer.

4. The results of various independent studies presented in the work allow us to conclude that the selected test images are inadequate for the totality of all possible situations.

5. The values given according to the results of [11] are estimates, since one value of the number of tests is taken into account. The dependence of the studied parameters of image compression on the number of results is subject to further study.

References:

1. EEE Standard for Binary Floating-Point Arithmetic. Copyright 1985 by The Institute of Electrical and Electronics Engineers, Inc 345 East 47th Street, – New York, NY10017, USA.
2. *Yashkardin V.* IEEE754 – binary floating point arithmetic standard/ IEEE754 – стандарт двоичной арифметики с плавающей точкой: [Online].– Access mode: URL: https: //i www.softelectro.ru, 02.11.2020.

3. *Salnik Yu. P.* Modified method of arithmetic coding with a fixed model/Модифицированный метод арифметического кодирования с фиксированной моделью in Radioelectronic and computer systems, 2005. – Issue 5. – P. 50–54.
4. The IEEE754 program is a converter of numbers in the IEEE754 format with an absolute precision of the result representation: [Online].– Access mode: URL: // http://www.softelectro.ru/pr_ieee754.htm.
5. *Levchenko A., Voitenkov R.* Origins of disability of diagnostic systems of weapons of the second kind with representation of numbers with a floating point / Витоки втрати працездатності систем діагностики ОВТ другого роду з представленням чисел з плаваючою комою, SWorld scientific works, – Vol. 1(38). – Part 3. – P. 108–115. – March, 2015.
6. *Levchenko A., Voitenkov R.* Method of representation of numbers for software tools of guaranteeing information technologies of decision support systems for weapons management / Метод представлення чисел для програмних засобів гарантоздатних інформаційних технологій систем підтримки прийняття рішень для керування станом ОВТ, in Proc. 19th conference "Problems of creation, development and application of special purpose information systems", Zhytomyr, ZMI NAU, 2012. – P. 142–143.
7. *Semchac O., Levchenko A.* Disadvanteges of computer implementation of SLAM-metods of local navigation autonovus mobile objects, SWorld journal, – Vol. 2. – Part 2. – P. 108–115. – October, 2019.
8. *Yurovitsky V. M.* IEEE754-tick threatens humanity/IEEE754-тика угрожает человечеству: [Online].– Access mode: URL: https: // IEEE754.htm, 14.10.2020.
9. *Korolev A. V., Bokhan K. A., Salnik Yu. P.* Subquantization of the transformer of a two-dimensional integer Haar transformation in Modeling and information technologies, – Kyiv: IPME, NASU, 2004. – Issue 26. – Vol. 1. – P. 3–7.
10. *Korolev A. V., Bokhan K. A., Salnik Yu. P.* Zonal-threshold selection of coefficients of fast two-dimensional Haar transformation in Information processing systems, NASU, 2004. – Issue 4. – Vol. 1. – P. 109–113.
11. *Bokhan K. A., Koroleva N. A., Salnik Yu. P.* Evaluation of the effectiveness of the zonal-threshold selection for the Haar transformation coefficients / Оценка эффективности зонально-пороговой селекции коэффициентов преобразования Хаара in Systems of processing information, NASU, 2004. – Issue 6. – P. 13–17.

12. *Bondarev V. N., Trester G., Chernega V. S.* Digital signal processing: methods and tools: 2nd ed., H, Cosinus, 2001. – 398 c.
13. *Klimenko L. A., Malakhov S. V.* Video encoding methods / Методы кодирования видеоданных in Information and control systems in railway transport, 1998. – Issue 2. – P. 59–62.
14. *Pratt W. C.* Image transfer methods. Reduction of redundancy / Методы передачи изображений. Сокращение избыточности, – M., Radio and communication, 1983. – 264 p.
15. *Vatolin D., Ratushnyak A., Smirnov M., Yukin V. M.* Methods of data compression: The device of archivers, compression of images and video / Методы сжатия данных: Устройство архиваторов, сжатие изображений и видео, Dialog–MIFI, 2002. – 384 c.
16. *Miano J.* Image compression formats in action, – M., Triumf, 2003. – 354 c.
17. *Wallace G. K.* The JPEG Still Picture Compression Standard // Communication of the ACM, 1991. – Vol. 34. – Issue 4.
18. *Korolev A. V., Malakhov S. V., Linnik N. F.* Modified zonal compression of images with partial elimination of phase components of the spectrum / Модифицированное зональное сжатие изображений при частичном устранении фазовых составляющих спектра in Systems of processing information, HMU, 2001. – Issue 5(15). – P. 176–180.
19. *Levchenko A.* The structure of the information technology of the guaranteed development of the tasks of forecasting the state of the defense system / Структура інформаційної технології гарантованого розв'язання задачі прогнозу стану системи озброєння in Problems of opening, testing, storage and operation of folding information systems, ZMI NAU, 2010. – Issue 3. – P. 96–110.

Andrii Levchenko, C.t.s., as.prof, assistant professor of the Computer science and technologies department, Odesa National University, 2 Dvoryanska str., Odesa, Ukraine
E-mail: katyaandreylev@gmail.com, a.levchenko@onu.edu.ua.
ORCID: https://orcid.org/0000-0001-5550-0027,

Yurii Shugailo, C.f-m.s., assistant professor of the Computer science and technologies department, Odesa National University, 2 Dvoryanska str., Odesa, Ukraine
E-mail: y.shugailo@gmail.
ORCID: https://orcid.org/0000-0003-2144-0930

Hanna Korenkova, C.f-m.s., assistant professor of the Computer science and technologies department, Odesa National University, 2 Dvoryanska str., Odesa, Ukrain
E-mail: nitsuk@onu.edu.ua
ORCID: https://orcid.org/0000-0001-7207-3688

Ilnara Sharipova, Master of metrology, Postgraduate student of the Computer science and technologies department, Odesa National University, 2 Dvoryanska str., Odesa, Ukraine
E-mail: iln.sharipova@ukr.net.
ORCID: https://orcid.org/0000-0003-0521-1299

Yurii Bercov, Master of computer science, senior master of the Computer science and technologies department, Odesa National University, 2 Dvoryanska str., Odesa, Ukraine
E-mail: bercov@gmail.com.
ORCID: https://orcid.org/0000-0001-9704-1121

Oksana Zui, Odesa National University, 2 Dvoryanska str., Odesa, Ukraine
E-mail: oks.zuj@gmail.com
ORCID: https://orcid.org/0000-0001-9520-4441