Programming the Formation of Difference Color Models for Lossless Image Compression

Alexander Shportko[1], Andrii Bomba[2] and Veronika Postolatii[3]

[1] *Academician Stepan Demianchuk International University of Economics and Humanities, 4, Acad. S. Demianchuk Str, 33000, Rivne, Ukraine*
[2] *National University of Water and Environmental Engineering, 11, Soborna Str, 33028, Rivne, Ukraine*
[3] *National University "Lviv Polytechnic", 12, St. Bandera Str, 79000, Lviv, Ukraine*

Abstract

The method and corresponding algorithm for selecting adaptive difference color models with integer coefficients for improving the efficiency of progressive lossless image compression was proposed. The necessity was reasoned and the displacement of the median differences of the basic components of the *R*, *G*, *B* color model to the middle of the range of possible values was implemented. Fragments of programs in the C++ language for implementing the algorithm for choosing a difference color model from 49 alternatives and the algorithm for determining the median component in linear time by the counting method are given. On the well-known ACT test set, it is shown that the use of difference color models with integer coefficients makes it possible to reduce the compression coefficients of photorealistic images by an average of 0.58 bpb.

Keywords

Progressive image compression, lossless compression, differential color models with integer coefficients.

1. Introduction

As you know, images significantly facilitate and accelerate the perception of information by a person. That is why today they are an integral part of multimedia information, which is most often transmitted by communication channels or stored on electromagnetic media. Therefore, the problem of increasing the efficiency of image compression is relevant today and will be relevant in the nearest future.

All graphic formats and methods used in them are divided into two main classes based on the principle of image data compression: lossy (for example, JPEG) and lossless (for example, PNG) [1]. And if for the vast majority of lossy image compression algorithms it is possible to provide the required compression ratio (the ratio of compressed to uncompressed image file sizes, expressed in bpb, hereinafter – CR) at the expense of quality degradation, then the level of lossless image compression actually depends only on the differences of the colors of their pixels and the compression algorithm itself. It is not adjustable by software and averages only 30-70% [1]. Therefore, the development of alternative graphic formats, such as HBF-LS [2], is an urgent task today.

2. Related works

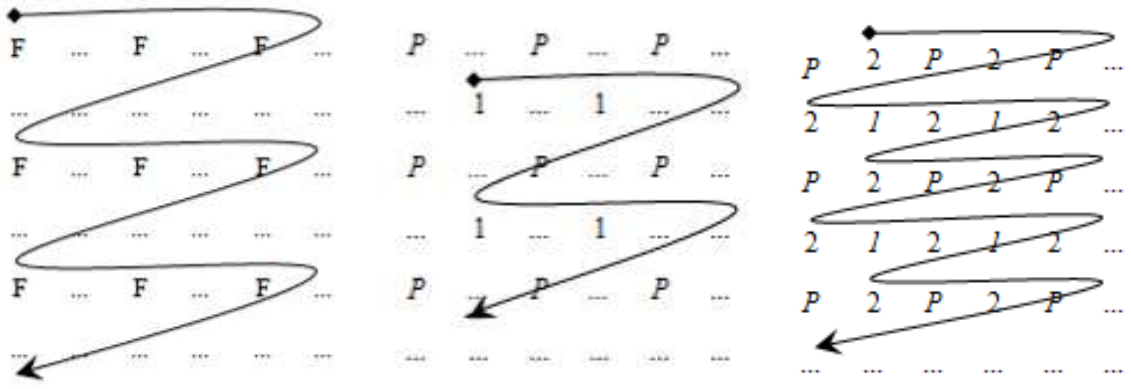
Any data compression is possible due to reduction or elimination of redundancies [3]. Three main types of redundancies are distinguished in images [4]: visual (consisting in the presence of

COLINS-2023: 7th International Conference on Computational Linguistics and Intelligent Systems, April 20–21, 2023, Kharkiv, Ukraine
EMAIL: ITShportko@gmail.com (A. V. Shportko); abomba@ukr.net (A. Ya. Bomba); VeronikaShportko@gmail.com (V. A. Postolatii)
ORCID: 0000-0002-4013-3057 (A. V. Shportko); 0000-0001-5528-4192 (A. Ya. Bomba); 0000-0002-9460-0781 (V. A. Postolatii)

information that is not perceived by the human visual system), inter-element or spatial (manifested in the correlation of brightness's of adjacent pixels) and coded (revealed when using codes of the same length for elements with different probabilities). It is common knowledge that the more types of redundancies of each type are processed by the graphic format, the more effective the compression is. But in the process of lossless compression, information is not lost, that is why the first type of redundancy is not reduced. Therefore, lossless image compression in archivers and graphic formats most often occurs in a maximum of four stages: firstly, context-dependent coding reduces redundancies between the same fragments or fragments with the same structure (reduces inter-element redundancy). At the second stage, the transition to an alternative color model is performed [5]. On the third – the brightness of the pixel components are transformed using predictors [6] (the second and third stages do not compress the image, but increase the unevenness of the brightness distribution and therefore increase the efficiency of the fourth stage). At the fourth stage, context-independent coding forms element codes with lengths dependent on their probabilities (processes code redundancy, for example, with Huffman codes or arithmetic codes [1; 3; 4; 7; 8; 9; 10; 11]). Context-independent coding can even be used instead of context-sensitive codes for individual pixel luminance, if this further reduces CS. For example, in the Deflate format, Huffman codes can be used instead of individual substitutions of the same luminance of the LZ77 dictionary algorithm [7].

Processing of image pixels luminance in popular graphic formats that perform lossless compression is most often carried out sequentially in rows from top to bottom, and in each row – consecutively from left to right. As a result, output of the compressed image in these formats is possible only after decoding is complete. Decompressing pictures or images with millions of pixels with this bypass method can take several seconds regardless of the size of the area or the resolution of the output device. We are developing the HBF-LS [2] format for progressive hierarchical lossless image compression, which will allow you to quickly obtain reduced copies of the image without decoding the entire file.

In the HBF-LS format, a hierarchical scheme is proposed as an alternative to sequential pixel traversal [5; 6], according to which on the first layer the pixels of the image are processed sequentially, starting with the first one in the upper left corner, in rows from top to bottom, and in each row – a sub-row from left to right with a step $h_1 = 2^k$, where k is determined from the condition $k = \left\lfloor \log_2 \left(\frac{\max(\min(image\_row; image\_col); 16) - 1}{15} \right) \right\rfloor$, *image_row* – the number of rows, *image_col* – the number of columns of image pixels (Figure 1a). This step ensures processing on the first layer at least 16 pixels along each of the axes, if the image is at least as large.



a) *b)* *c)*

Figure 1: The sequence of pixel bypass in the process of progressive hierarchical processing: a) pixels of the first layer; b) pixels the first passage regular layer; c) pixels the second passage regular layer

In the following layers ($l = \overline{2, k+1}$), the intermediate pixels of the image are processed in two passes: in the first, those of them that are located at the intersection of the diagonals of the squares with the

vertices in the adjacent pixels of the previous layers are sequentially processed in steps $h_l = 2^{k+2-l}$ both by rows and by columns (see Figure 1b), and on the second, the unprocessed pixels are sequentially bypassed between the adjacent pixels of the previous layers and the pixels of the first pass with the same step in the columns and with a halved step in the rows (see Figure 1c). In Figure 1 the symbol *F* indicates the pixels of the first layer, the symbol *P* indicates the pixels of the previous layers, the number *1* – the pixels of the first pass of the next layer, the number *2* – the pixels of the second pass of the next layer. Pixels that were processed earlier and therefore are not processed on the next pass of the layer are highlighted in italics.

In this article, we propose a method and a corresponding algorithm for increasing code redundancy by reducing inter-element redundancy using adaptive difference color models with integer coefficients in the process of progressive lossless image compression.

3. Usage of non-adaptive difference color models in graphic formats

Difference color models [5], as well as predictors [6] are used to reduce the compression ratio of context-independent coding. The basic principle of such coding can be formulated as follows: **the length of the code of an arbitrary element with a higher probability should not exceed the length of the code of any element with a lower probability**. With regard to images, this principle is based on the fundamental position of information theory, according to which to minimize the length of the sequence code, each value of the element *i* (brightness of a separate component *brightness* (for a separate component of each pixel of images True Color $brightness = \overline{0, 255}$) or the value of a context-dependent code) with the probability of occurrence p_i it is advisable to code with $l_i = -\log_2 p_i$ bits [10], where l_i is *the length of the entropy code element i* (here and everywhere else in the work, the logarithm is taken to the base 2). Therefore, the average code length of a block element after applying any context-independent algorithm, according to the formula of Shannon [4], cannot be less than *the entropy of the source*

$$H = -\sum_i p_i \times \log p_i \ . \tag{1}$$

As it is known, the entropy of the source decreases with increasing unevenness of the distribution of probabilities (frequencies) between elements [6].

Since the average length of a context-independent code is close to entropy (1) [3], the total length of a block of such codes for a sequence of elements is approximately equal to the sum of the lengths of their entropy codes, that is, the *length of the entropy code of the sequence* [11]. Let each of the values *i* occur n_i times in the sequence of length $N = \sum_i n_i$. According to the statistical definition of probability, $p_i = n_i / N$. Therefore, the length of the *entropy code of the element*, to which the length of the arithmetic code is close, is $l_i = -\log p_i = \log \frac{N}{n_i}$, and the total length of the *entropy code of the sequence*, taking into account (1), approaches the value

$$L = N \times H = N \log(N) - \sum_i n_i \log(n_i). \tag{2}$$

We will use this formula to estimate the lengths of the alternative blocks of entropy codes.

To increase the efficiency of context-independent coding in the process of lossless image compression the help of *predictors* is used, which during the round predict the value of the brightness of each component of the next pixel (for the most common 24-bit images, these are the brightness of the red, green and blue components, written as integers in separate bytes), using the brightness of the values of the same components of previously processed adjacent pixels [6], since the brightness data have the highest level of correlation between them. In the process of using predictors, **deviations** Δ_{uv} of the value of the brightness of the next pixel component $brightness_{uv}$ from the value predicted by the selected predictor $predict_{uv}$, are calculated and further coded. So,

$$\Delta_{uv} = brightness_{uv} - predict_{uv} \tag{3}$$

(u and v run through all the rows and columns of the pixel components of the image, respectively). Adjacent pixels of the images often have similar colors (close values of the brightness of the corresponding components), so the forecast value often coincides with the brightness value of the next component. It is often close to this value and rarely differs significantly from it. That is why, most of the values Δ_{uv} are close to zero. Thus, the use of predictors most often increases the unevenness of the probability distribution of brightness values and, as a result, reduces entropy (1).

Different color models are also used to reduce entropy. The fact is that, firstly, the color of each pixel in the three-component color model can be represented in the form of coordinates by three linearly independent vectors of any basic colors [5]. Secondly, different image components display sufficiently similar geometrically spatial structure objects (as, for example, in Figure 2).

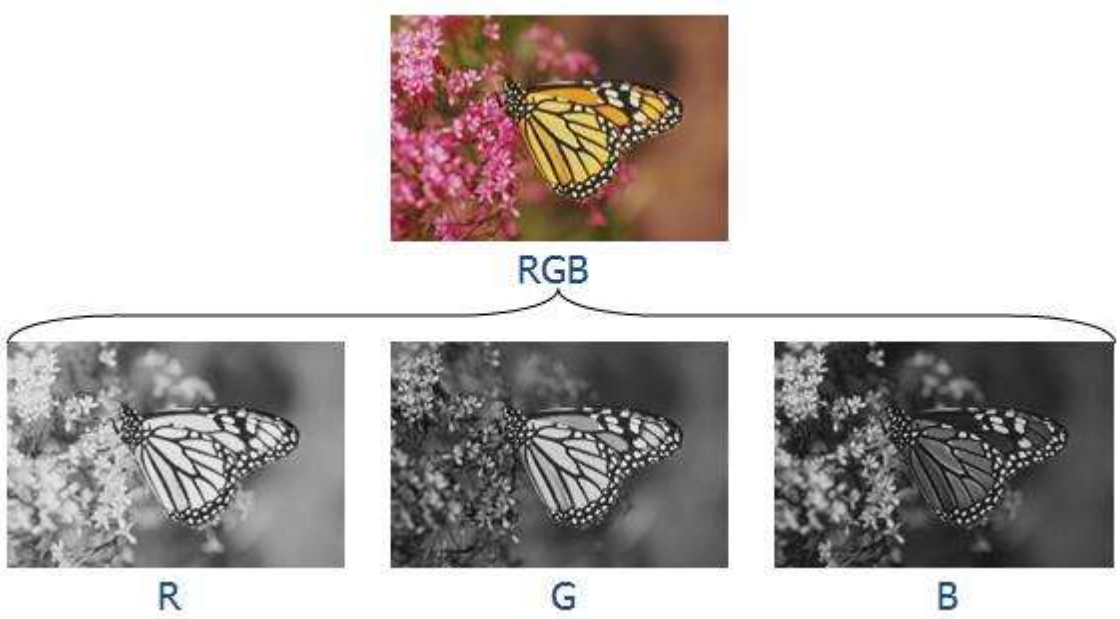


Figure 2: Layout of the Monarch.bmp image from the ACT test set by components of the RGB color model

It is clear that the correlation coefficients between pairs of components in the RGB color model for different images can differ significantly from each other, and one of the components of a such arbitrary pair with a strong correlation can be replaced by the **difference** with another component [5], if this will reduce the entropy (1) in the further process of context-independent coding. Using component differences instead of components luminance of the RGB color model performs inter-component decorrelation, just as predictors (3) implement decorrelation between adjacent pixels luminance. But today, modern archivers and image compression formats process the brightness of pixels mainly in a fixed color model (for example, the PNG format – in the *R, G, B* model; the BMP format – in the *B, G, R* model; the JPEG format [12; 13] – in the *Y, Cb, Cr;* the RAR archiver format is in the model *R – G, G, B – G)* and do not use the ability to choose an effective color model for each image that **reduces entropy the most effectively due to intercomponent decorrelation.** For example, in the popular YCbCr color model, the differences of *R, G,* and *B components* are applied to all images in the two chromatic components *Cb* and *Cr*:

$$\begin{aligned} Y &= \left(\frac{77}{256}\right)R + \left(\frac{150}{256}\right)G + \left(\frac{29}{256}\right)B; \\ Cb &= -\left(\frac{44}{256}\right)R - \left(\frac{87}{256}\right)G + \left(\frac{131}{256}\right)B + 128; \\ Cr &= \left(\frac{131}{256}\right)R - \left(\frac{110}{256}\right)G - \left(\frac{21}{256}\right)B + 128. \end{aligned} \qquad (4)$$

Predictors perform decorrelation of the brightness of individual components of the color model, so the transition to alternative difference color models in the process of lossless image compression is performed before the use of predictors [6]. In addition, graphic file formats for lossless image compression must provide both fast encoding and fast decoding, so it is advisable to use difference color models with integer coefficients [5]. Therefore, the **purpose** of this article is to substantiate the options and algorithm for choosing an adaptive color model for each image to reduce their CR in the process of lossless compression and to provide a software implementation of this algorithm.

4. Formation of adaptive difference color models based on the data of individual components

As it will be shown below, in order to ensure unambiguous decoding in the image, it is possible to perform **a maximum of two** replacements of the values of different components by differences with other components. Therefore, considering this limitation, in the process of coding for each image during preprocessing, the problem of choosing one difference color model among alternatives so as to reduce the CR as much as possible appears. In fact, in the compression process, it is necessary to evaluate the expediency of replacing the *R* component values with one of the *RG, GR, RB* or *BR* differences, the *G* component values with *GR, RG, BG* or *GB* differences, and the *B* component values with *BR, RB, BG* or *GB* differences for each pixel and **among these possible differences, choose a maximum of two that will maximally reduce the predicted length of the entropy code** (2). To solve this problem, in [5] the investigated entropy lengths of individual components were recorded in the form of the analysis matrix *A*:

$$A=\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}=\begin{pmatrix} L(\Delta(R)) & L(\Delta(R-G)) & L(\Delta(R-B)) \\ L(\Delta(G-R)) & L(\Delta(G)) & L(\Delta(G-B)) \\ L(\Delta(B-R)) & L(\Delta(B-G)) & L(\Delta(B)) \end{pmatrix}, \quad (5)$$

where the operator Δ indicates the calculation of the predictor values (3) of the corresponding component of each pixel, and *L* indicates the length of the entropy code (2) of these values. Since for linear static predictors $a_{mn}=a_{nm}$, then to determine the matrix *A* it was enough to calculate six values of the coefficients of its upper triangle, including the main diagonal. Accordingly, the task of choosing a difference color model was reduced to determining at most two non-diagonal elements of different rows of the matrix *A,* which, among the elements smaller than the diagonal elements of their rows, deviate the most from them (ensuring the maximum reduction of the total length of the entropy code). If there are such elements, the row index of each of them defines the reduced component, and the column index defines the component that is subtracted from it (for example, the selection of the element a_{02} indicates that in the alternative color model, for each pixel of the image, the value of the *R* component must be reduced by components *B)*.

The proposed procedure for calculating the coefficients of matrix *A* is, in our opinion, effective in simulating the process of lossless image compression. Firstly, the transition to the difference color model is performed, then predictors are applied, and their results are encoded by a context-independent algorithm. But this approach is focused on the use of linear static predictors. It does not take into account the impact on the compression process of the context-dependent algorithm (which is especially relevant for discrete-tone images) and the cross-correlation of components in the difference color model. It also does not justify the choice of predictor for prediction and does not determine which precisely from two opposite differences (for example, *R – G* or *G – R)*, which have the same entropy code length after applying the linear predictor should be chosen. In addition, the matrix *A* allows you to form a difference color model for the entire image but does not indicate ways to fragment it and determine the difference color models for individual fragments. Therefore, we will deal with the elimination of these shortcomings.

5. Formation of adaptive difference color models considering cross-correlation of components

Difference color models are used to maximize entropy reduction (1) due to inter-component decorrelation. Therefore, they increase the efficiency of the application of the context-independent algorithm, although they do not directly perform data compression. Let's first set the maximum number of differences in the difference color model.

Note that difference color models with integer coefficients can contain **no more than two** component differences.

To justify this fact, we will use the method of proof from the opposite and the requirement of linear independence of vectors of basic colors of any alternative color model. Let there be a difference color model with three component differences. Firstly, each component must enter at least one difference to ensure a fallback to the RGB color model. Secondly, no difference can contain the same component in the decrement and in the subtractor, because then it will turn to zero, which will violate the requirement of linear independence of the components. Thirdly, any of two differences can not contain two same or opposite components, because then they will be linearly dependent. Fourth, none of the components can be included in all three differences, because otherwise at least two differences of the same or opposite sign will be generated, which will contradict the previous statement (for example, if all differences include the component *R,* then the components *G* and *B* must complement two of these differences, and the third difference must again contain *G* or *B* and will contain the same components as one of the other two differences). And, finally, each of the components cannot be included in two differences, because then a linear combination with the sum or subtraction of these differences, which excludes this component, will be linearly dependent with the third difference (for example, if the component *R* is included in the first difference as the minus, and the second as the minus, then the sum of these differences will create a difference $G - B$ or $B - G$, which will be the same or opposite in sign to the third difference, which will contradict the requirement of linear independence of the basic color vectors). It turns out that each of the three components must enter the difference color model only once, which makes it impossible to create three differences. ■

As already mentioned above, in the process of compressing images, a context-dependent algorithm is first performed, and for pixels not processed by this algorithm, a transition to the difference color model is performed, then predictors are applied, and the results of their use and the context-dependent algorithm are coded with a context-independent algorithm. That is, the difference color models increase the efficiency, first of all, of the context-independent algorithm. That is why they are effective primarily for photorealistic images [5]. Among the main symmetric hierarchical predictors for such images, *ProgresPredict1* provides the smallest CR [6], therefore, we use this predictor to select a difference color model. In C++, this predictor is written like this:

```
ubyte ProgresPredict1(ubyte a, ubyte b, ubyte c, ubyte d)
{ubyte pa, pb;
 if (a>=c)   pa=a-c;
 else   pa=c-a;
 if (b>=d)   pb=b-d;
 else   pb=d-b;
 if (pa<pb)   return (a+c)/2;
 if (pb<pa)   return (b+d)/2;
 return (a+b+c+d)/4; }
```

This predictor returns the arithmetic mean of those two opposite elements from the nearest four (*a, b, c, d*) that differ the least. If the deviations of the opposite elements are the same, then *ProgresPredict1* returns the arithmetic mean of all four values.

If each component of the difference color model were coded into separate Deflate blocks [7] (that is, there would be no cross-correlation of the components) and the length of the entropy code (2) of the difference of the components after applying *ProgresPredict1* (the operator Δ from (3)) would be equal to the length of the entropy code from the opposite component differences (that is, the relations $L(\Delta(R-G)) = L(\Delta(G-R))$, $L(\Delta(R-B)) = L(\Delta(B-R))$, $L(\Delta(G-B)) = L(\Delta(B-G))$) would be

fulfilled, then, taking into account the statement proved above, the following differential color models with integer coefficients would be alternative in the process of progressive hierarchical compression:

1'. R, G, B;
2'. R, G, B – R;
3'. R, G, B – G;
4'. R, G – R, B;
5'. R, G – B, B;
6'. R – G, G, B;
7'. R – B, G, B;
8'. R, G – R, B – R;
9'. R, G – R, B – G; (6)
10'. R, G – B, B – R;
11'. R – G, G, B – G;
12'. R – G, G, B – R;
13'. R – B, G, B – G;
14'. R – B, G – R, B;
15'. R – G, G – B, B;
16'. R – B; G – B, B.

This list of alternative integer-difference color models is constructed from variants of all possible models by discarding opposite-difference models and component-permutation models so that the reduced ones retain the input RGB components. We also classified the RGB color model as differential with having zero subtractors.

We also note that, firstly, the difference color models increase the efficiency of using only the context-independent algorithm. Therefore, we will not consider the brightness of the pixels that are completely included in the replacement of the modified LZ77 algorithm by the adjacent previously processed pixels when determining the parameters of these models.

Secondly, the length of the entropy code (2) of the component difference after applying nonlinear hierarchical predictors in general and *ProgresPredict1* in particular is usually not equal to the length of the entropy code from the opposite component difference. Therefore, due to the introduction of opposite differences, the number of difference color models with one difference doubles (from 6 to 12), with two differences – four times (from 9 to 36) and the total number of alternative difference of color models increases from 16 to 49.

Thirdly, when calculating the differences in the luminance of the components in cases where the luminance values of the subtractor insignificantly exceed the luminance values of the reduced one, negative values of the differences are obtained, which, when stored in unsigned 8-bit components of the color model, due to overflow, are transformed into values close to 256. And when the luminance values of the subtractor are slightly smaller than the luminance values of the reduced one, values close to zero are obtained in the differences. This dispersion of color model difference values negatively affects the accuracy of *ProgresPredict1* prediction and, as a consequence, increases entropy (1). To avoid such dispersion of the values of the differences of close luminances, it is customary to shift them to the middle of the range of possible component values. For example, for the most common 8-bit sampling, the difference values are shifted to 128, as in the YCbCr color model (4). We will shift up to 128 **median differences** of the components of the RGB color model, assuming that most of the luminance of these components are centered around their medians. We denote the medians of the *R, G, B* components for pixels that are not processed by the context-dependent algorithm by *medR, medG,* and *medB* respectively.

Taking into account the above remarks, in the process of progressive hierarchical compression of lossless images, the following difference color models with integer coefficients can be alternatives:

1. R, G, B;
2. R, G, (R – medR) – (B – medB) + 128;
3. R, G, (G – medG) – (B – medB) + 128;
4. R, (R – medR) – (G – medG) + 128, B;
5. R, (B – medB) – (G – medG) + 128, B;
6. (G – medG) – (R – medR) + 128, G, B;
7. (B – medB) – (R – medR) + 128, G, B;

8. R, G, (B – medB) – (R – medR) + 128;
9. R, G, (B – medB) – (G – medG) + 128;
10. R, (G – medG) – (R – medR) + 128, B;
11. R, (G – medG) – (B – medB) + 128, B;
12. (R – medR) – (G – medG) + 128, G, B;
13. (R – medR) – (B – medB) + 128, G, B;
14. R, (R – medR) – (G – medG) + 128, (R – medR) – (B – medB) + 128;
15. R, (G – medG) – (R – medR) + 128, (G – medG) – (B – medB) + 128;
16. R, (B – medB) – (G – medG) + 128, (B – medB) – (R – medR) + 128;
17. R, (G – medG) – (R – medR) + 128, (B – medB) – (R – medR) + 128;
18. R, (R – medR) – (G – medG) + 128, (B – medB) – (G – medG) + 128;
19. R, (G – medG) – (B – medB) + 128, (R – medR) – (B – medB) + 128;
20. (R – medR) – (G – medG) + 128, G, (R – medR) – (B – medB) + 128;
21. (G – medG) – (R – medR) + 128, G, (G – medG) – (B – medB) + 128;
22. (B – medB) – (R – medR) + 128, G, (B – medB) – (G – medG) + 128;
23. (G – medG) – (R – medR) + 128, G, (B – medB) – (R – medR) + 128;
24. (R – medR) – (G – medG) + 128, G, (B – medB) – (G – medG) + 128;
25. (R – medR) – (B – medB) + 128, G, (G – medG) – (B – medB) + 128; (7)
26. (R – medR) – (B – medB) + 128, (R – medR) – (G – medG) + 128, B;
27. (G – medG) – (R – medR) + 128, (G – medG) – (B – medB) + 128, B;
28. (B – medB) – (R – medR) + 128, (B – medB) – (G – medG) + 128, B;
29. (B – medB) – (R – medR) + 128, (G – medG) – (R – medR) + 128, B;
30. (R – medR) – (G – medG) + 128, (B – medB) – (G – medG) + 128, B;
31. (R – medR) – (B – medB) + 128, (G – medG) – (B – medB) + 128, B;
32. R, (G – medG) – (R – medR) + 128, (R – medR) – (B – medB) + 128;
33. R, (R – medR) – (G – medG) + 128, (G – medG) – (B – medB) + 128;
34. R, (G – medG) – (B – medB) + 128, (B – medB) – (R – medR) + 128;
35. R, (R – medR) – (G – medG) + 128, (B – medB) – (R – medR) + 128;
36. R, (G – medG) – (R – medR) + 128, (B – medB) – (G – medG) + 128;
37. R, (B – medB) – (G – medG) + 128, (R – medR) – (B – medB) + 128;
38. (G – medG) – (R – medR) + 128, G, (R – medR) – (B – medB) + 128;
39. (R – medR) – (G – medG) + 128, G, (G – medG) – (B – medB) + 128;
40. (R – medR) – (B – medB) + 128, G, (B – medB) – (G – medG) + 128;
41. (R – medR) – (G – medG) + 128, G, (B – medB) – (R – medR) + 128;
42. (G – medG) – (R – medR) + 128, G, (B – medB) – (G – medG) + 128;
43. (B – medB) – (R – medR) + 128, G, (G – medG) – (B – medB) + 128;
44. (B – medB) – (R – medR) + 128, (R – medR) – (G – medG) + 128, B;
45. (R – medR) – (G – medG) + 128, (G – medG) – (B – medB) + 128, B;
46. (R – medR) – (B – medB) + 128, (B – medB) – (G – medG) + 128, B;
47. (R – medR) – (B – medB) + 128, (G – medG) – (R – medR) + 128, B;
48. (G – medG) – (R – medR) + 128, (B – medB) – (G – medG) + 128, B;
49. (B – medB) – (R – medR) + 128, (G – medG) – (B – medB) + 128, B.

We denote these difference color models by DCM_i, $i = \overline{1, 49}$ (abbreviation of different color models).

Firstly, the components of difference color models in which *R, G* or *B* are included with the same signs are better correlated with each other than when these signs are opposite. Therefore, such color models are likely to provide smaller CR. Therefore, with strict limitations on the coding time, the color model can be chosen not among the 49 alternatives, but only among the first 31, and this will slightly affect the CR and reduce the compression time. In addition, as it will be shown later, single-difference color models are effective mainly for discrete-tone images. Therefore, if it is known that photorealistic images are compressed, then alternative difference color models 2-13 from set (7) can also not be analyzed. And, secondly, the use of such difference color models with integer coefficients significantly speeds up decoding compared to difference color models with real coefficients [5] due to performing operations on integers instead of operations on floating-point numbers.

It is also obvious that among these 49 alternative color models, only one should be chosen (determine its *numberDCM)* which will provide the smallest size of the compressed image, that is, the smallest predicted length of the entropy code (2):

$$L\left(\Delta\left(DCM_{numberDCM}\right)\right)=\min_{j=\overline{1,49}} L\left(\Delta\left(DCM_j\right)\right). \quad (8)$$

The length of the entropy code (2) does not depend on individual elements, but on the frequencies of these elements. In turn, the frequencies of the elements obtained as a result of the application of the difference color model and the *ProgresPredict1* predictor consist of the sum of the frequencies of the individual components, since the transformed luminance of the pixel components are included in the compressed Deflate-blocks in sequence. Therefore, if the first components of the difference color models the index 0 are assigned, the second – 1, the third – 2, as was done in [5] (then, for example, from (7) we have that $DCM_8^2=(B-medB)-(R-medR)+128,\ DCM_8^0=R$), then

$$n_i^{\Delta\left(DCM_j\right)}=n_i^{\Delta\left(DCM_j^0\right)}+n_i^{\Delta\left(DCM_j^1\right)}+n_i^{\Delta\left(DCM_j^2\right)},\ j=\overline{1,49},\ i=\overline{0,255}, \quad (9)$$

where n_i is the luminance frequency *i.*

The selection of the difference color model according to formulas (8), (9) requires the determination of 84 component differences (36 models with two differences and 12 models with one difference) and the application of predictors to them on all pixels of the image. But if we take into account that the same component differences are included in difference color models (for example, the difference *(R – medR)* – *(B – medB)* + 128 is included in color models № 2, 13, 14, 19, 20, 25, 26, 31, 32, 37, 38, 40, 46, 47), then to determine the effective color model of the next image, it is enough to calculate the frequencies after applying the predictors for six component differences and for the three input components *R, G* and *B,* and then analyze 49 entropy lengths (8) of combinations of sums of these frequencies (9). To do this, we will write down these 9 sets of frequencies in the form of a matrix of *freqComponent* frequency arrays:

$$freqComponent=\begin{pmatrix} freqComponent_{00} & freqComponent_{01} & freqComponent_{02} \\ freqComponent_{10} & freqComponent_{11} & freqComponent_{12} \\ freqComponent_{20} & freqComponent_{21} & freqComponent_{22} \end{pmatrix}=$$

$$=\begin{pmatrix} \left\{n_i^{\Delta(R)}\right\} & \left\{n_i^{\Delta((R-medR)-(G-medG)+128)}\right\} & \left\{n_i^{\Delta((R-medR)-(B-medB)+128)}\right\} \\ \left\{n_i^{\Delta((G-medG)-(R-medR)+128)}\right\} & \left\{n_i^{\Delta(G)}\right\} & \left\{n_i^{\Delta((G-medG)-(B-medB)+128)}\right\} \\ \left\{n_i^{\Delta((B-medB)-(R-medR)+128)}\right\} & \left\{n_i^{\Delta((B-medB)-(G-medG)+128)}\right\} & \left\{n_i^{\Delta(B)}\right\} \end{pmatrix},\ i=\overline{0,255}. \quad (10)$$

Then each combination of sums of frequencies that specifies one of the color models (7) must include the frequencies of at least one carrier component, that is, at least one array of frequencies from the diagonal of the *freqComponent* matrix*,* and no more than two component differences, that is, arrays of frequencies of off-diagonal elements of this matrix, which must be asymmetrical among themselves. After selecting arrays of component frequencies due to the application of the next color model and *ProgresPredict1,* it is necessary to calculate their element-by-element sums according to (9), determine the predicted length of the entropy code from the received frequencies (2) and choose among all color models the one that provides the best predicted compression according to (8).

Frequency arrays *freqComponent* (10) is similar to the analysis matrix *A* (5), because they are both used to select a color model, the row number of off-diagonal elements in them determines the decreasing component, and the column number is the denominator. But the matrix *A* contains the predicted lengths of possible individual components and does not take into account their cross-correlation, and the matrix *freqComponent* – frequencies of these components and determining the entropy length from combinations of the sums of these frequencies (9) makes it possible to take into account their cross-correlation.

6. Software implementation of the selection of the difference color model

To select the difference color model of the entire image, first let's determine the medians of its components *medR, medG,* and *medB.* Since the median is a value located within a series of sorted

values, then, of course, to determine these medians, it would be possible to sort by increasing values of the pixel components that are not included in the long substitutions of the modified LZ77 algorithm, and then choose the average values from the sorted arrays. But the complexity of such an approach would then be $O(image\_row \times image\_col \times \log(image\_row \times image\_col))$. We implement an algorithm with linear computational complexity relative to the number of pixels, using the idea of the method of sorting by counting [14, p. 223-226]: for each component, we first count the frequencies of each brightness n_i, $i = \overline{0, 255}$, after which, using these frequencies, we determine the brightness relative to which the brightness of at least half of the elements is smaller. If the number of pixels for which the median is determined in this way is odd, then this median will be equal to the average value of the sorted array. If it is even, then it will be equal to the value from which the second half of the sorted array begins (the so-called "upper median").

The frequency array, which is additionally created and processed for such determination of medians, contains only 256 elements, which in general is much less than the number of image pixels and indicates the feasibility of applying the idea of the method of sorting by counting. A fragment of a C++ program for determining median components with frequency counts of individual luminances can look like this:

```
step=1; // calculate the brightness frequencies of the components
for (l=countShar; l>=2; l--) // loop through all layers except the first
 {step*=2;
  // process the pixels of the second pass of the layer
  startColumn=0;
  for (j=0; j<image_row; j+=step/2) // cycle by rows
   {if (startColumn==0) startColumn=step/2; // the first pixel of the first lines
    else startColumn=0; // the first pixel of the second rows
    for (i=startColumn; i<image_col; i+=step) // cycle through the columns
     // if the LZ-decomposition on the nearest processed pixels is not performed
     // or the next pixel is not included in a long replacement (from six elements)
     if (!LZReplaceAdjacent || !longAdjacentReplace[j*image_col+i])
      {// read the components of the next pixel in the BGR color model
       b=image[j][i*3]; g=image[j][i*3+1]; r=image[j][i*3+2];
       // increase the number of processed pixels and accumulate frequencies
       countPixelDCM++;
       freqR[r]++; freqG[g]++; freqB[b]++; }}
  // similarly process the pixels of the first layer pass
  for (j=step/2; j<image_row; j+=step)
   for (i=step/2; i<image_col; i+=step)
    if (!LZReplaceAdjacent || !longAdjacentReplace[j*image_col+i])
     {b=image[j][i*3]; g=image[j][i*3+1]; r=image[j][i*3+2];
      countPixelDCM++;
      freqR[r]++; freqG[g]++; freqB[b]++; }}
UBYTE4 halfPixelCM=countPixelCM/2; // half of processed pixels
UBYTE4 sumFreq=0; medR=0;
// increase the median R until we process half of the frequencies of the elements
while (sumFreq<halfPixelCM)
 sumFreq+=freqR[medR++];
sumFreq=0; medG=0;
while (sumFreq<halfPixelCM)
 sumFreq+=freqG[medG++]; // similarly determine the median of G
sumFreq=0; medB=0;
while (sumFreq<halfPixelCM)
 sumFreq+=freqB[medB++]; // similarly determine the median B
UBYTE1 medComponent[3];
// store the calculated medians for further complex use
```

```
medComponent[0]=medR; medComponent[1]=medG; medComponent[2]=medB;
```

Defined medians of individual components make it possible to accumulate *freqComponent* (10) arrays in the matrix of frequencies due to the application of *ProgresPredict1* to individual components and component differences. To speed up the calculation of the displacement, we enter the constant component differences in the intermediate variable *alfa*. For example, calculating the differences of *R* and *G components* for each pixel that is not included in the long substitutions of the LZ algorithm, we simplify the expression $(R-medR)-(G-medG)+128$ to $R-G+alfa$, where $alfa=-medR+medG+128.$

The function for determining the predicted length of the entropy code (2) based on element frequencies is given as follows:

```
double sizeEntropiCode(UBYTE4 *masFreq, unsigned int countAllFreq=256)
{UBYTE4 count=0, i;
 double size=0;
 for (i=0; i<countAllFreq; i++)
  {count+=masFreq[i];
   if (masFreq[i]>1)
    size+=masFreq[i]*log(masFreq[i]); }
 if (count)   size=(count*log(count)-size)/log(2);
 return size; }
```

To save the differences of the color model, we will use the following variables with the numbers of its components from 0 to 2: *cm1* – number of the component of the result of the first difference, *cm11* – number of the component of the reduced first difference, *cm12* – number of the component of the subtractor of the first difference, *cm2, cm21, cm22* – similar variables for the second color model difference. If the denominator of the first or second difference is equal to its denominator, then this difference is not applied. Having formed the frequency array matrix *freqComponent* and the implementation of the function *sizeEntropiCode,* we now present a fragment of the program for determining the color model that provides the smallest predicted length of the entropy code (8):

```
cm11=0, cm12=0, cm21=0, cm22=0; // parameters of RGB models without differences
for (l=0; l<256; l++) // sum of frequencies of components (9) for RGB models
 freq[l]=freqComponent[0][0][l]+freqComponent[1][1][l]+freqComponent[2][2][l];
// definition of predicted length of the entropy code after application of
// predictors to pixels in the color room RGB models
lenRGB = minLenDCM =(UBYTE4) sizeEntropyCode(freq);
// sort through the possible differences color models
UBYTE1 c11, c12, c21, c22;
for (c11=0; c11<=2; c11++) // index of the first component of the first differences
 for (c12=0; c12<=2; c12++) // index of the second component of the first differences
  if (c11!=c12) // components differences have to differ
• {// sum of frequencies components (9) for models with one difference:
• // from the first component of the first differences  second component is subtracted
• for (l=0; l<256; l++)
• {freq [l]= freqComponent[c11][c12][l]; // difference of the first component with the second
•  if (c11!=0) freq[l]+= freqComponent[0][0][l]; // other two components without changes
•  if (c11!=1) freq[l]+= freqComponent[1][1][l];
• if (c11!=2) freq[l]+= freqComponent[2][2][l]; }
• lenDCM =(UBYTE4) sizeEntropyCode (freq);
• if (lenDCM < minLenDCM) // found more effective color model
• {minLenDCM = lenDCM; // remember her parameters
•  cm1=cm11=c11; // in the model alone the difference in which from the first component
•  cm12=c12; // the second one is subtracted
•  cm21=cm22=0; }
   for (c21=c11+1; c21<=2; c21++) // index the first component the second differences
```

```
 for (c22=0; c22<=2; c22++) // index of the second component of the second differences
  if (c21!=c22 && // components of the second difference are different
      (c21 != c12 || c22!=c11)) // and no symmetrical to the first differences
   {// sum of the frequencies of the components (9) for models with two differences
    // in two differences from the first components are subtracted  from the second
    for (l=0; l<256; l++)
     freq[l]=freqComponent[c11][c12][l]+ // frequency of the first differences
             freqComponent[c21][c22][l]+ // frequency of the second differences
             freqComponent[3-c11-c21][3-c11-c21][l]; // frequency carrier
    lenDCM =(UBYTE4) sizeEntropyCode (freq);
    if (lenDCM < minLenDCM)
     {minLenDCM = lenDCM;
      cm1=cm11=c11; // result and reduced of the first differences
      cm12=c12; // subtractor of the first differences
      cm2=cm21=c21; // result and reduced of the second differences
      cm22=c22; } // subtractor of the second differences
    // in the first difference from the first component the second one is subtracted
    // in the  the second difference from the second component the first one is subtracted
    for (l=0; l<256; l++)
     freq[l]=freqComponent[c11][c12][l]+ // frequency of the first difference
             freqComponent[c22][c21][l]+ // frequency of the second difference
             freqComponent[3-c11-c21][3-c11-c21][l]; // frequency carrier
    lenDCM =(UBYTE4) sizeEntropyCode (freq);
    if (lenDCM < minLenDCM)
     {minLenDCM = lenDCM;
      cm1=cm11=c11; // result and reduced of the first differences
      cm12=c12; // subtractor of the first differences
      cm2=cm22=c21; // result and reduced of the second differences
      cm21=c22; }} // subtractor of the second differences
 //further vertical differences are processed similarly
```

Without lines marked with ' •', this program fragment selects alternative difference color models to the RGB model only among models with two differences, that is, it analyzes the effectiveness of using not 49, but 37 color models.

The use of a difference color model as an alternative to RGB, leads to the need to apply its differences to all image pixels that have not been processed by a context-sensitive algorithm, both during encoding and each time during the decoding process. That's why, it slows down these two processes. Although, on the other hand, reducing the size of the compressed image due to the use of difference color models accelerates decoding, as it leads to the processing of compressed data of a smaller volume. Therefore, after determining the differences of the integer color model that provides the minimum predicted entropy code length, we arranged to fall back to the RGB color model if this predicted length decreased by less than 1%. Let's set the indices of the reducible *cm11* and the denominator *cm12* of the first difference and the indices of the reducible *cm21* and the denominator *cm22* of the second difference equal to zero:

```
if (lenRGB-minLenDCM<lenRGB/100)
 {cm11=cm12=cm21=cm22=0;
  requiredDCM=false;
  return; }        .
```

Alternative color model differences must be applied to the input components *R, G,* or *B* in sequence. Therefore, at the end of choosing a difference color model with integer coefficients, we implement a change in the order of subtraction of components, if the result of the first difference is used in the second difference:

```
// if the second difference is defined and uses the result of the first difference
if (cm21!=cm22 && (cm1==cm21 || cm1==cm22))
```

```
{i=cm1; cm1=cm2; cm2=i; // change order differences
 i=cm11; cm11=cm21; cm21=i;
 i=cm12; cm12=cm22; cm22=i; }
```

To speed up the application of the differences of the specified color model, we calculate the coefficients *alfa1* and *alfa2* for the first and second differences:

```
if (cm11!=cm12) alfa1=-medComponent[cm11]+medComponent[cm12]+128;
if (cm21!=cm22) alfa2=-medComponent[cm21]+medComponent[cm22]+128;
```

Then the procedure for applying the differences of the specified color model to any pixel of the image will be written as follows:

```
void codeDCM (int row, int col, UBYTE1 *pixel)
 {if (cm11!=cm12) // the first subtraction is required
   pixel[cm1]=pixel[cm11]-pixel[cm12]+alfa1;
  if (cm21!=cm22) // the second subtraction is required
   pixel[cm2]=pixel[cm21]-pixel[cm22]+alfa2; }
```

We return to the RGB color model in the decoder in the reverse order: first we cancel the application of the second, and then the first difference. Therefore, the procedure for returning from the specified color model to the RGB model for each pixel in the decoding process is implemented as follows:

```
void decodeDCM (UBYTE2 row, UBYTE2 col, UBYTE1 * decodeBytePixel)
 {// if determined, the second subtraction and to DCM components are applied
  if (cm21!=cm22 && appliedDCMBytePixel[cm2])
   if (cm2==cm21) decodeBytePixel[cm2]+=decodeBytePixel[cm22]-alfa2;
   else decodeBytePixel[cm2]=decodeBytePixel[cm21]-decodeBytePixel[cm2]+alfa2;
  // if defined, the first subtraction and to DCM components are applied
  if (cm11!=cm12 && appliedDCMBytePixel [cm1])
   if (cm1==cm11) decodeBytePixel[cm1]+=decodeBytePixel[cm12]-alfa1;
   else decodeBytePixel[cm1]=decodeBytePixel[cm11]-decodeBytePixel[cm1]+alfa1; }     .
```

7. Analysis of the results of the application of difference color models for the compression of whole images

Let us analyze the results of applying selected from alternative difference color models with integer coefficients to test images of the ACT set [15] (Table 1, 3, 4). We can see that on average for this set due to the use of difference color models with integer coefficients the CR is decreased by 0.36 bpb (second row of Table 1). Moreover, about 19% of this decrease occurred due to the shift of the median difference to the middle of the range of possible values (that is, due to the consideration of *alpha,* the third line of Table 1).

For different images, the smallest predicted length of the entropy code is provided by different difference color models (Table 2). But the effectiveness of difference color models differs, first of all, for different types of images. If for photorealistic images the CR is decreased by an average of 0.58 bpb, then for discrete-tone images – only by 0.01 bpb, since they have a low level of correlation between components. For image #1, there was even a fallback to the RGB color model, as alternative difference color models reduced the predicted entropy code length by less than 1%.

The choice of difference color models not from the 49th, but from the first 31st with the same minuends or subtrahends forms models with opposite components in 25% of the images, which insignificantly increases their compression ratio. On the other hand, this does not lead to a drastic acceleration of compression, because most of the time in the process of choosing difference color models is spent on forming a matrix of component frequency arrays and their differences *freqComponent* (10).

If we choose a difference color model from only 16 alternatives (6) according to the symmetric analysis matrix A (5), which contains the entropy lengths of the 3 components and 3 of their

differences [5], taking into account the shifts of the medians after using *ProgresPredict1* and does not take into account the cross-correlation between the components, then we get models with opposite components for 75% of the images (fourth line of Table 2), deterioration of CR for three images, and on average for the set – increase of CR by 0.01 bpb (fourth line of Table 1). Compression compared to the analysis of all alternative color models (7) will speed up by only 2%. If we take into account the asymmetries of the differences of matrix *A* (49 alternative color models without taking into account cross-correlation), then the CR will also deteriorate, but already for 2 pictures (fifth line of Table 1). Therefore, to ensure the smallest CR, a difference color model with integer coefficients should be chosen from all 49 alternatives (7).

Table 1

Image compression ratios of the ACT set after applying various options for formation of difference color models, bpb

Color model	Image Number								Average CR
	1	2	3	4	5	6	7	8	
RGB	1.34	0.58	4.65	3.81	4.14	5.16	0.61	4.32	3.07
Difference CM from the 49th or 31st alternative	1.34	0.57	4.45	3.25	3.69	4.06	0.59	3.71	2.71
Difference CM from 49 alternatives without *alpha*	1.34	0.57	4.62	3.27	3.81	4.21	0.59	3.82	2.78
Difference CM from 16 alternative analysis matrix *A*	1.34	0.57	4.45	3.27	3.69	4.07	0.59	3.76	2.72
Difference CM from 49 alternative analysis matrix *A*	1.34	0.57	4.45	3.27	3.69	4.06	0.59	3.73	2.71

Table 2

Difference color models (without specifying *alpha)* generated for images of the ACT set by different variants of their formation

Color model	Image Number			
	1	2	3	4
Difference CM from 49 alternatives	RGB (R, R-G, B)	B-R, G-B, B	R, R-G, G-B	R, B-G, B-R
Difference CM from 49 alternatives without *alpha*	RGB (R, R-G, B)	B-R, G-B, B	R, R-G, G-B	R, R-G, R-B
Difference CM from the first 31st alternative	RGB (R, R-G, B)	B-R, B-G, B	R, G-R, G-B	R, B-G, B-R
Difference CM from 16 alternatives according to *A*	RGB (R, R-G, B)	R-B, G-B, B	R, G-R, B-G	R, G-B, B-R
Difference CM from 49 alternatives according to *A*	RGB (R, R-G, B)	B-R, G-B, B	R, R-G, G-B	R, G-B, R-B

Continuation of Table 2

Color model	Image Number			
	5	6	7	8
Difference CM from 49 alternatives	G-R, G, G-B	G-R, G, G-B	R-G, B-G, B	R, R-G, B-G
Difference CM from 49 alternatives without *alpha*	G-R, G, G-B	G-R, G, G-B	R-G, B-G, B	R, R-G, B-G
Difference CM from the first 31st alternative	G-R, G, G-B	G-R, G, G-B	R-G, B-G, B	R, R-G, B-G
Difference CM from 16 alternatives according to *A*	R-G, G, B-G	R-G, G, B-G	R-G, G-B, B	R, G-R, B-G
Difference CM from 49 alternatives according to *A*	G-R, G, G-B	G-R, G, G-B	R-G, B-G, B	R, G-R, G-B

Table 3
The time of encoding image files of the ACT set using different variants of the formation of difference color models, s

Color model	Image Number								Average time
	1	2	3	4	5	6	7	8	
RGB	2.19	3.05	1.06	1.93	1.16	2.04	1.25	1.87	1.82
Difference CM from the 49th or 31st alternative	2.62	3.53	1.29	2.17	1.41	2.20	1.50	2.05	2.10
Difference CM from 16 alternatives	2.52	3.52	1.25	2.11	1.36	2.14	1.49	1.99	2.06

Table 4
Decoding time of ACT set image files encoded using different difference color models, s

Color model	Image Number								Average time
	1	2	3	4	5	6	7	8	
RGB	0.58	1.21	0.30	0.54	0.32	0.49	0.49	0.42	0.54
Different from 49th, 31st or 16th alternative	0.58	1.27	0.32	0.60	0.31	0.61	0.50	0.50	0.59

In general, the use of difference color models significantly (on average by more than 13% according to Table 3) slows down encoding not only due to the need to choose such a model from among alternatives, but also due to the orientation to a context-independent algorithm that encodes individual literals. However, the time of image decoding due to the use of these color models is increased by only a tenth of a second (9.3%, second row of Table 4), which, together with a significant reduction in CR, makes it possible to use them effectively in practice.

8. Discussions

In the future, with the aim of further reducing the file sizes of lossless compressed images in the process of progressive hierarchical traversal and speeding up decoding, we plan to increase the efficiency of using symmetric and asymmetric predictors [6] by applying difference color models with integer coefficients to image fragments. We are working on an algorithm for dividing images into large rectangular pieces with different adaptive difference color models by analyzing their median differences.

9. Conclusions

1. It is possible to reduce the CR of images in three-component color models not only due to the data decorrelation of individual components, but also with the help of inter-component decorrelation by switching to difference color models. Inter-component decorrelation should be performed in such a way as to enhance the properties of the image used by the algorithms of prepressing and direct compression of the selected graphic format, for example, to minimize the predicted length of the entropy code (2).
2. The use of difference color models with integer coefficients in the process of progressive hierarchical compression of lossless images makes it possible to reduce the CR of photorealistic images by an average of 0.58 bpb. To ensure the smallest CR, a difference color model with integer coefficients should be chosen from all 49 alternatives (7).
3. In order to increase the efficiency of the application of difference color models with integer difference coefficients, the median differences of the basic components R, G, B should be shifted to the middle of the range of possible values (for example, in color models with a sampling rate of 8 bits – up to 128).
4. Integer difference color models provide significant improvements in the lossless compression efficiency of three components photorealistic images in formats that use predictors, and may therefore be implemented in future standards-level versions of these formats.

10.References

[1] J. Miano, Compressed Image File Format: JPEG, PNG, GIF, XBM, BMP, Addison Wesley, New York, 1999, 264 p.

[2] A. Shportko, Author's certificate № 58216 of Ukraine. HBF-LS Graphics Format Specification. Version 1.0, Kyiv, 2015, 14 p.

[3] D. Selomon, A Guide to Data Compression Methods, Springer, New York, 2002, 295 p.

[4] R. Gonzalez, R. Woods, Digital Image Processing, 4th ed., Pearson, London, 2017, 1192 p.

[5] A. Shportko, The use of differences of colors models for compression of RGB-images without losses, Selection and treatment of information 31 (2009) 90-97.

[6] A. Shportko, V. Postolatii, Development of Predictors to Increase the Efficiency of Progressive Hierarchic Context-Independent Compression of Images Without Losses, Computational Linguistics and Intelligent Systems (COLINS 2021) : Proceedings of the 5th International Conference (Kharkiv, Ukraine, 22-23 April 2021), Kharkiv, Vol. 1. (2021) 1026-1038. URL: http://ceur-ws.org/Vol-2870/paper77.pdf.

[7] A. Shportko, A. Bomba, V. Postolatii, Rejection of the Inefficient Replacements while Forming the Schedule of the Modified Algorithm LZ77 in the Process of Progressive Hierarchical Compression of Images without Losses, Computational Linguistics and Intelligent Systems (COLINS 2022) : Proceedings of the 6th International Conference (Glivice, Poland, 12-13 May 2022), Glivice, Vol. 3171 (2022) 1594-1605. URL: http://ceur-ws.org/Vol-3171/paper113.pdf.

[8] B. Rusyn, O. Lutsyk, Y. Lysak, A. Lukenyuk, L. Pohreliuk, Lossless image compression in the remote sensing applications, 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine (2016) 195-198. doi: 10.1109/DSMP.2016.7583539.

[9] C. Raghavendra, S. Sivasubramanian, A. Kumaravel, Improved image compression using effective lossless compression technique, Cluster Computing 22 (2019) 3911–3916. URL: https://doi.org/10.1007/s10586-018-2508-1.

[10] R. K. Pathria, P. Beale, Statistical Mechanics, Third Edition, Academic Press, 2011, p. 51.

[11] A. Shportko, A. Bomba, L. Shportko, Features of Application of Arithmetic Encoding in the Process of Progressing Hierarchical Compression of Images without Losses, Proceedings of the National University "Lviv Polytechnic". Series: Information Systems and Networks, 783 (2014) 12-22.

[12] G. Wallace, The JPEG still picture compression standard, Communication of ACM, 34 (1991) 30-44.

[13] O. Shehata, Unraveling the JPEG, Parametric Press, 1 (2019). URL: https://parametric.press/issue-01/unraveling-the-jpeg/.

[14] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, Third Edition, Vol. 1, Dialektika, Kiyv, 2020, 648 p.

[15] ACT – Test Files, 2002. URL: http://www.compression.ca/act/act-files.html.