Development of Predictors to Increase the Efficiency of Progressive Hierarchic Context-Independent Compression of Images Without Losses

Alexander Shportko[a] and Veronika Postolatii[b]

[a] *Academician Stepan Demianchuk International University of Economics and Humanities, 4, Acad. S. Demianchuk Str, 33000, Rivne, Ukraine*
[b] *National University "Lviv Polytechnic", 12, St. Bandera Str, 79000, Lviv, Ukraine*

Abstract
The expediency is substantiated, the way of pixel traversal is given and symmetrical and asymmetric predictors for realization of progressive hierarchical context-independent compression of images without losses are offered. The results of the application of the proposed predictors to reduce the entropy of the images of the ACT set in the process of previous transformations are presented. It is shown that the use of combinations of the proposed progressive hierarchical predictors makes it possible, for example, to reduce the compression coefficients of photorealistic images by an average of 0.08 bpp.

Keywords 1
Progressive compression of images, compression without losses, predictors, entropy.

1. Introduction

In today's world, images are an integral part of multimedia information, which is often created, stored and stored on digital media and transmitted through communication channels [1]. Compression of the respective files allows us to increase proportionally the speed of information exchange over the network and reduce the amount of disk space usage. All graphic formats on the principle of image compression are divided into two main classes: with losses and lossless. And if for the vast majority of image compression algorithms with losses it can be provided the desired ratio due to deterioration, the level of lossless image compression depends, in fact, only on the color differences of their pixels and the compression algorithm itself, it is not programmatically regulated and averages only 30 – 70%. Today, designers and developers of the websites often save photorealistic images in JPEG format. Discrete-tone and those where losses are unacceptable are saved in PNG format. Processing of brightness of pixels of images in popular graphic formats which carry out compression without losses (including PNG format [2, p. 249-317]), is mostly carried out consecutively on lines from top to bottom, and in each line it is carried from left to right. As a result, you can output a compressed image in these formats only after decoding all the pixels. Decompressing photos or images with millions of pixels in this way can take several seconds, regardless of the area size and resolution of the output device.

Along with this, to accelerate the output of large images in compression formats with losses, graphic formats that use progressive (translational) hierarchical processing of pixels have already been developed [3, p. 176] (for example, wavelets). In the process of applying this method of image processing, the pixels are bypassed in layers, increasing each time the resolution (progressive component). And in the process of sequential processing of the data of the next layer the data of the previous layers is used (hierarchical component). The image from pixels of the next layer is actually a

COLINS-2021: 5th International Conference on Computational Linguistics and Intelligent Systems, April 22–23, 2021, Kharkiv, Ukraine
EMAIL: ITShportko@ukr.net (A. Shportko); VeronikaShportko@gmail.com (V. Postolatii)
ORCID: 0000-0002-4013-3057 (A. Shportko); 0000-0002-9460-0781 (V. Postolatii)

reduced in several times (usually four) copy of the image from the pixels of the previous layer. So, the last layer coincides with the input image. Therefore, during the progressive hierarchical decoding, the details of the image appear gradually. It is possible to stop such decoding after decompression of the layer with the number of pixels, not less than the output area on each of the axes, without waiting for the reproduction of all pixels of the image. Thus, **the development of methods and graphic format for lossless image compression using the principles of progressive hierarchical processing**, which is the purpose of the study, **is an urgent task today**. This work, in fact, is a logical continuation of the work [4], as it improves predictors described in it and offers new predictors that provide better compression ratios.

2. The analysis of recent research and publications. The principles of compression of images without losses with the use of predictors

As it is known, lossless image compression in graphic formats often occurs in three stages. During the first step brightness of the pixels are converted using predictors. During the second step context-dependent coding reduces redundancies between such fragments. During the last step context-independent coding eliminates redundancies between the predominant values of the brightness of the components. Context-sensitive encoding can reduce the compression ratio (the ratio of the size of the compressed to uncompressed image files) several times due to such fragments. But such fragments are rare in photorealistic images, so the only universal step in lossless image compression is context-independent encoding. The basic principle of such coding: **the code length of an arbitrary element with a higher probability should not exceed the code length of any element with a lower probability**. With regard to images, this principle is based on the fundamental tenet of information theory, according to which to minimize the length of the sequence code each *brightness* (for a separate component of each pixel of the image True Color $brightness = \overline{0, 255}$) with probability of appearance $p_{brightness}$ it is expedient to encode by $-\log_2 p_{brightness}$ bits [5, p. 17]. Therefore, the average code length of the block element after the application of any context-independent algorithm according to the formula of Shannon [5, p. 611; 6], cannot be less than the *entropy of the source*

$$H = -\sum_{brightness} p_{brightness} \times \log_2 p_{brightness} . \tag{1}$$

It is known that the entropy of the source decreases with increasing non-uniformity of the distribution of probabilities (frequencies) between the elements [7]. According to our estimates, the use of a context-independent algorithm (for example, Huffman coding or arithmetic coding [2-5; 7-9]) reduces the compression ratio of images by an average of 33%.

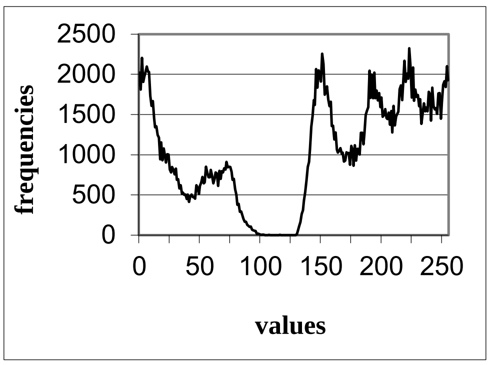
To improve the efficiency of this encoding in the process of lossless image compression it can be used *predictors* that predict the brightness of each component of the next pixel (for example, for the most common 24-bit images – the brightness of red, green and blue components written in integers in individual bytes), using the values of the luminance of the same components of the previously processed adjacent pixels, because these luminance have the highest degree of correlation [7, p. 675]. In the process of using this approach, it is calculated and further encoded the **deviation** Δ_{ij} of the brightness value of the next component of the pixel $brightness_{ij}$ from the predicted value of the selected predictor $predict_{ij}$, namely

$$\Delta_{ij} = brightness_{ij} - predict_{ij} \tag{2}$$

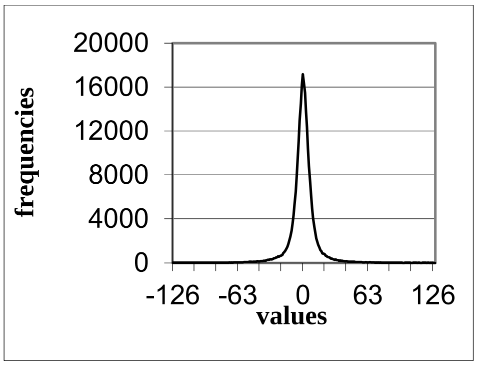
(*i* and *j* run accordingly on all lines and columns of the components of pixels of the image). Adjacent pixels of images often have similar colors. Thus, they have close values of the brightness of the respective components, so the value of the forecast often coincides with the brightness of the next component. It is often close to this value and sometimes is significantly different from it (Figure 1).

That is why most values Δ_{ij} are close to zero. Thus, the use of predictors often increases the uneven distribution of probabilities of brightness values and, as a consequence, reduces entropy (1).

Why do the brightness values of the pixel components deviate from the values predicted by the predictors? The fact is that these deviations are often due to two objectively existing main factors: "strong" changes – the trend and "weak" background fluctuations – noise. Therefore, two opposite types of models are possible: the contribution of noise is insignificant compared to the contribution of evolution; the contribution of the evolution is insignificant compared to the contribution of noise. In the first case ... we will predict the value of ... based on ... the current trend in the second – as equal to the arithmetic mean ... of the previous elements " [8, p. 59].



a)



b)

Figure 1: A division of frequencies of values of green components of image Lena.bmp: a) before the use of predictor (*H*=7,59 of bpb); b) after the use of Left-predictor (*H*=5,34 of bpb)

In addition, during the traditional sequential pixel traversal, predictors can use only the brightness values from the previous lines and on the left in the next line for prediction (Figure 2), which reduces the efficiency of their usage.

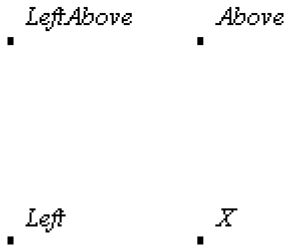


Figure 2: Designation of adjacent elements to the element *X* in a sequential way to bypass the pixels of the image

With this method of traversing pixels, using the notation of adjacent elements for the element *X* in accordance with Figure 2, the most common predictors in C today can be implemented as follows:

```
typedef unsigned char ubyte;
ubyte LeftPredict(ubyte Left, ubyte Above, ubyte LeftAbove)
 {return Left;}
ubyte AbovePredict(ubyte Left, ubyte Above, ubyte LeftAbove)
 {return Above;}
ubyte AveragePredict(ubyte Left, ubyte Above, ubyte LeftAbove)
 {return (Left+Above)/2;}
ubyte PaethPredict(ubyte Left, ubyte Above, ubyte LeftAbove)
 {int pp=Left+Above-LeftAbove;
  int pa, pb, pc;
  pa=abs(pp-Left); pb=abs(pp-Above); pc=abs(pp-LeftAbove);
  if (pa<=pb && pa<=pc) return Left;
  else if (pb<=pc) return Above;
        else return LeftAbove;}
ubyte MedPredict(ubyte Left, ubyte Above, ubyte LeftAbove)
 {if (LeftAbove>=max(Left, Above)) return min(Left, Above);
  else if (LeftAbove<=min(Left, Above)) return max(Left, Above);
        else return Left+Above-LeftAbove; }
```

Predictor *LeftPredict* predicts the value of the next element that is equal to the value on the left. Predictor *AbovePredict* predicts the value that is equal to the value above. And predictor *AveragePredict* predicts the value that is equal to the arithmetic mean of these values. These three predictors belong to the linear static predictors and essentially calculate the arithmetic mean of individual adjacent elements and therefore describe the noise model. The next two predictors belong to nonlinear static predictors and take into account trends relatively to the value predicted in the assumption of equalities of background oscillations of diagonal elements. Therefore, they describe a mixed trend-noise model.

The Paeth predictor *PaethPredict* calculates the value at point X based on the plane passing through the points *Left*, *Above* and *LeftAbove* in three-dimensional space and predicts one of these three values in the direction of the smallest increment relatively to the calculated value.

The *MedPredict* predictor tries to adapt to the local horizontal and vertical edges. The *Left* value is often returned when a horizontal edge is detected. The *Above* value is returned when a vertical edge is detected. If no edge is detected, the value of the plane above the point X passing in the three-dimensional space through the points *Left*, *Above* and *LeftAbove* is returned. The first four of these predictors are used in PNG format, for example. The fourth predictor is used in WinRAR archiver. The last predictor is used in JPEG-LS compression format. A description of other predictors used in the process of sequential pixel traversal can be found in [6].

3. Progressive compression of images without losses. Basic and additional symmetric hierarchical predictors

On the one hand, translational hierarchical compression of images allows to speed up decoding, and on the other – to consider the values of previously processed elements from four, not just from two different sides in the process of using predictors. That is why to achieve the goal of the study we have developed an effective scheme of pixel traversal and appropriate predictors that implement progressive hierarchical compression of images without losses. In particular, for a progressive hierarchical traversal, we propose a scheme in which the pixels of the image are processed sequentially on the first layer, starting from the first, in rows from top to bottom, and in each row - from left to right in steps $h_1 = 2^k$, where k is determined from a condition $k = \left\lfloor \log_2\left(\frac{\max(\min(height; width); 16) - 1}{15}\right)\right\rfloor$, *height* –the number of lines, *width* – the number of columns of pixels of image. This step provides processing on the first layer of at least 16 pixels (if any) on each of the axes (as in the icons), if the image is smaller. On the next layers ($l = \overline{2, k+1}$) intermediate pixels of the image are processed in two passes: on the first pass, those of them which are contained on crossing of diagonals of squares with vertices in adjacent pixels of the previous layer with a step $h_l = 2^{k+2-l}$ both on lines, and on columns are processed. And on the second – unprocessed pixels are processed between adjacent pixels of the previous layer and the first pass with the same step on columns and with twice reduced – on lines (Figure 3).

P	2	P	2	P	...
2	1	2	1	2	...
P	2	P	2	P	...
2	1	2	1	2	...
P	2	P	2	P	...
...	...	...	...	...	...

Figure 3: The scheme of progressive hierarchical processing of pixels of the image on layers, starting with the second layer: P – pixels of the previous layer; 1 – pixels of the first pass of the next layer; 2 – pixels of the second pass of the next layer

The proposed sequence of traversing the pixels of the image allows not only to speed up decoding when the size of the output area is much smaller than the size of the image, but also to use hierarchical predictors to predict the value of each element of the next pixel (in Figure 4 it is marked as *X*) on all the layers, starting with the second. To describe these predictors, we denote the values of the brightness of similar components of the **nearest** (adjacent) previously processed pixels from the previous and next layer or pass using the notation *a, b, c, d, ab, bc* (Figure 4).

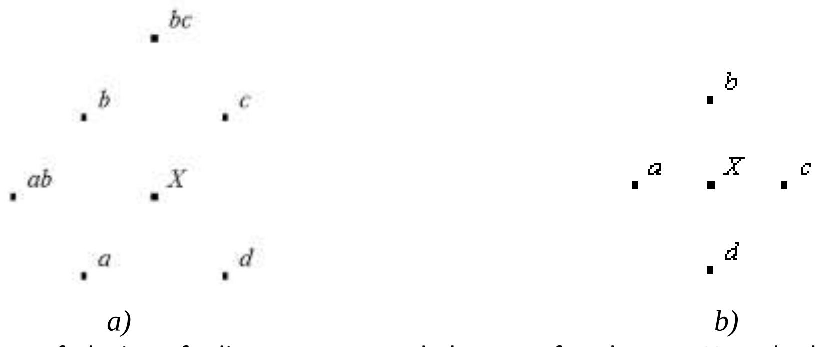


Figure 4: Charts of placing of adjacent processed elements for element *X* on the layers, starting with the second: a) for the first pass; b) for the second passage

Using these notations, we investigate the principles of forecasting the two main symmetric hierarchical predictors, **focused on the basic arithmetic mean** (noise component**) of the two opposite elements of the next four (*a, b, c, d*), which differ least** (trend component), i.e., most likely belong to the same object in the image. In the case when the brightness of the components of close pixels is more affected by weak background oscillations, we predict the value of the next element using the trend-noise predictor *ProgresPredict1*, which returns the basic arithmetic mean. In the case when the increments of the nearest opposite elements are the same, this predictor returns the arithmetic mean among *a, b, c, d*. Predictor *ProgresPredict1* performs an average of 3.5 branching operators and the same number of comparison operations to predict the values of individual brightness of each pixel.

If strong fluctuations in the brightness of the pixels predominate, then let us predict the value of the next element using the noise-trend predictor *ProgresPredict2*. Among four adjacent processed elements *a, b, c, d* this predictor determines and returns the closest value to the basic arithmetic mean, when such a value is unique. If among the adjacent processed elements there are two closest equidistant values to the basic arithmetic mean, then among them the one around which more brightness is concentrated is returned. When the brightness is scattered, it returns less from the nearest equidistant values to the basic arithmetic mean. The general orientation to smaller values makes it possible to shift the non-zero deviations Δ_{ij} calculated according to (2) towards positive values and thus increase the unevenness of their distribution. The *ProgresPredict2* predictor performs an average of 5 branching operators and 6 comparison operations for each prediction.

In C, basic symmetric predictors can be written as follows:

```
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; }
```

```
ubyte ProgresPredict2(ubyte a, ubyte b, ubyte c, ubyte d)
 {ubyte absac, absbd, maxac, minac, maxbd, minbd, prognozn;
  if (a<=c)   {absac=c-a; maxac=c; minac=a;}
  else   {absac=a-c; maxac=a; minac=c; }
  if (b<=d)   {absbd=d-b; maxbd=d; minbd=b; }
  else   {absbd=b-d; maxbd=b; minbd=d; }
  if (absac<=absbd)
   {if (minbd>=minac && minbd<=maxac)   return minbd;
    if (maxbd>=minac && maxbd<=maxac)   return maxbd;
    if (minbd>maxac)   return maxac;
    else   return minac; }
  else
   {if (minac>=minbd && minac<=maxbd)   return minac;
    if (maxac>=minbd && maxac<=maxbd)   return maxac;
    if (minac>maxbd)   return maxbd;
    else   return minbd; }}
```

Let us describe the mechanism for predicting two additional symmetric hierarchical predictors. The noise predictor *ProgresPredict3* returns the arithmetic mean between the average luminance values among *a, b, c, d*. Thus, it does not consider the largest and smallest values among these four brightness values. To do this, the two minimum and maximum values among the opposite brightness values are calculated first, and then the average between the minimum of the two maxima found and the maximum of the two calculated minima is calculated. This predictor performs an average of 4 branching operators and the same number of comparison operations to predict the brightness values of each pixel. The *ProgresPredict3* predictor is effective on the boundaries of the depicted objects.

The trend predictor *ProgrnetworkPredict4* chooses the one around which the most brightness is concentrated among the two opposite elements, which differ the least from the next four. This predictor is effective in areas of images where the influence of the trend significantly prevails. To predict each brightness value, it uses as many branch operators and comparison operations as *PrograsPredict3*.

In C, additional symmetric hierarchical predictors can look like this:

```
ubyte ProgresPredict3(ubyte a, ubyte b, ubyte c, ubyte d)
 {ubyte maxac, minac, maxbd, minbd;
  if (a<=c)   {maxac=c; minac=a; }
  else   {maxac=a; minac=c; }
  if (b<=d)   {maxbd=d; minbd=b; }
  else   {maxbd=b; minbd=d; }
  return (min(maxac,maxbd)+max(minac,minbd))/2; }
```

```
ubyte ProgresPredict4(ubyte a, ubyte b, ubyte c, ubyte d)
 {ubyte absac, absbd, maxac, minac, maxbd, minbd;
  if (a<=c)   {absac=c-a; maxac=c; minac=a; }
  else   {absac=a-c; maxac=a; minac=c; }
  if (b<=d)   {absbd=d-b; maxbd=d; minbd=b; }
  else   {absbd=b-d; maxbd=b; minbd=d; }
  if (absac<=absbd)
 {if (b+d>a+c)   return maxac;
  else   return minac; }
else
 {if (a+c>b+d)   return maxbd;
  else   return minbd; }}
```

Let us analyze the results of the usage of the proposed symmetric predictors of hierarchical traversal to reduce the entropy of the pixel components of the standard test set Archive Comparison Test (ACT), the image characteristics of which are given in table. 1. This set contains both synthesized (№№ 1 (with noise), 2, 7) and photorealistic (all others) images. You can download TIFF versions of these images, for example, from [10].

Table 1
Characteristics of selected images of ACT set

№ file	File name	Size, Kb	Number of unique colors	% of unique colors	Features
1	Clegg.bmp	2101	127696	17.83	Continuous-tone, artificial, noisy, several big objects
2	Frymire.bmp	3622	3622	0.29	Discrete-tone, artificial, one big object
3	Lena.bmp	769	148279	56.56	Continuous-tone, natural, several big objects
4	Monarch.bmp	1153	78617	19.99	Continuous-tone, natural, one big and several small objects
5	Peppers.bmp	769	111344	42.47	Continuous-tone, natural, several big objects
6	Sail.bmp	1153	75748	19.26	Continuous-tone, natural, many medium-sized objects
7	Serrano.bmp	1464	1313	0.26	Discrete-tone, artificial, one big fragmented object
8	Tulips.bmp	1153	118233	30.07	Continuous-tone, natural, several big objects

In the table. 2 for comparison, the line *NonePredict* shows the entropy of the brightness values of the components of the pixels without the usage of predictors, and below – the entropy of the same brightness after using the most common predictors of sequential traversal and symmetric predictors of hierarchical traversal. In the initial passes of the hierarchical pixel traversal, symmetric predictors were not used if they increased entropy.

As it was expected, the usage of noise-trend predictors *ProgresPredict2* and *ProgresPredict4* proved to be more effective for synthesized images, as they are characterized by sharp differences in the brightness of the processed pixels within the depicted objects (trend influence predominates), and trend-noise predictors *ProgresPredict1* and *ProgresPredict3* were more effective for photorealistic images, because their adjacent pixels often have similar but different colors (dominated by noise).

The use of symmetric predictors significantly reduces the entropy on the last layers compared to other currently known sequential traversal predictors, as these predictors consider the effect of four equidistant pixels on different sides, not just left and top pixels, as in sequential traversal. In this case, each processed pixel of the next layer in the middle of the image is used to process eight pixels of the next layer (and not for a maximum of three subsequent pixels, as for predictors of sequential traversal).

The level of correlation of the brightness of the components of adjacent pixels on the last layers is significantly higher than the level of correlation of these brightness on the first layers, especially for synthesized images. That is why the use of sequential bypass predictors (and sometimes the refusal to use them) provide less entropy than the use of progressive predictors on the initial layers, although the efficiency of the latter increases with increasing layer number. Therefore, in order not to increase the entropy in general, for the synthesized image № 2 it is advisable to abandon the use of predictors on both the second and third layers, and for the image № 7 – on the first pass of the second layer.

The noise predictors *ProgresPredict1* and *ProgresPredict3* provide almost the same compression ratio, but *ProgresPredict1* requires fewer operators and comparison operations, so it is chosen as the base. Among the trend predictors *ProgresPredict2* and *ProgresPredict4*, *ProgresPredict2* provides much lower entropy, so it is used as a base. In addition, when performing the basic trend predictor *ProgresPredict2*, on average, 2.5 comparison operations are performed more than the noise predictor

ProgresPredict1, so the predictor *ProgresPredict1* should be preferred to *ProgresPredict2* at the same entropy values after their usage. For comparison: in the process of performing a nonlinear static predictor of sequential traversal of *PaethPredict*, an average of 5.66 comparison operations are performed. *MedPredict* predictor performs 5 comparison operations. Thus, the use of the noise predictor *ProgresPredict1* even instead of nonlinear sequential bypass predictors significantly speeds up encoding/decoding.

Table 2
The entropy of brightness values of the components of pixels of images of the set of ACT after the usage of sequential bypass predictors and symmetric hierarchical bypass predictors, bpb

Predictor	Layer	Passage-way	№ file								Middle entropy
			1	2	3	4	5	6	7	8	
NonePredict	1	1	7.54	4.65	7.75	7.50	7.66	7.32	5.99	7.66	7.01
LeftPredict	1	1	3.57	1.68	5.26	4.57	4.60	5.67	1.74	5.13	4.03
AbovePredict	1	1	3.68	2.23	4.92	4.60	4.72	5.97	1.86	4.94	4.12
AveragePredict	1	1	4.66	2.88	4.86	4.27	4.40	5.59	2.58	4.74	4.25
PaethPredict	1	1	1.90	1.51	4.90	4.26	4.29	5.50	1.35	4.61	3.54
MedPredict	1	1	1.90	1.54	4.84	4.15	4.20	5.43	1.37	4.50	3.49
ProgresPredict1	2	1	7.46	4.61	6.93	6.83	7.18	7.19	5.99	7.33	6.69
	2	2	7.10	4.67	6.30	6.52	6.18	7.04	5.84	7.18	6.35
	3	1	7.27	4.48	6.43	6.44	6.61	7.12	6.00	6.91	6.41
	3	2	6.71	4.59	5.83	6.04	5.71	6.83	5.32	6.54	5.95
	k+1	1	5.65	2.90	4.80	4.10	4.44	5.59	2.85	4.52	4.36
	k+1	2	3.96	1.90	4.44	3.60	3.79	4.92	1.72	3.87	3.53
	Together		4.88	2.57	4.72	4.03	4.22	5.41	2.47	4.41	**4.09**
ProgresPredict2	2	1	7.38	4.61	6.96	6.86	7.19	7.19	5.99	7.31	6.69
	2	2	7.07	4.67	6.29	6.57	6.21	7.04	5.30	7.10	6.28
	3	1	6.98	4.48	6.52	6.57	6.70	7.21	5.58	6.93	6.37
	3	2	6.67	4.59	5.89	6.12	5.84	6.85	4.30	6.52	5.85
	k+1	1	4.72	2.06	5.01	4.49	4.64	5.82	1.90	4.94	4.20
	k+1	2	0.23	1.19	4.65	4.03	4.10	5.24	0.96	4.38	3.10
	Together		2.74	1.80	4.92	4.41	4.48	5.65	1.64	4.82	**3.81**
ProgresPredict3	2	1	7.43	4.61	6.96	6.81	7.15	7.19	5.99	7.34	6.69
	2	2	7.21	4.67	6.37	6.50	6.24	6.97	5.84	7.16	6.37
	3	1	7.25	4.48	6.42	6.44	6.60	7.06	6.00	6.93	6.40
	3	2	6.87	4.59	5.82	6.05	5.72	6.75	5.59	6.55	5.99
	k+1	1	5.67	2.99	4.75	4.07	4.40	5.58	2.85	4.50	4.35
	k+1	2	3.00	2.98	4.38	3.58	3.74	4.93	1.69	3.85	3.52
	Together		4.41	2.66	4.67	4.02	4.18	5.41	2.48	4.40	**4.03**
ProgresPredict4	2	1	7.34	4.61	6.87	6.67	6.95	7.19	5.99	7.27	6.61
	2	2	7.12	4.67	6.45	6.61	6.23	7.05	5.47	7.20	6.35
	3	1	7.28	4.48	6.41	6.37	6.53	7.03	5.70	6.96	6.35
	3	2	6.72	4.52	6.01	6.12	5.94	6.85	4.57	6.68	5.93
	k+1	1	4.94	2.08	4.95	4.44	4.67	5.85	1.99	4.96	4.24
	k+1	2	3.34	1.21	4.64	4.00	4.05	5.34	1.16	4.42	3.52
	Together		4.38	1.83	4.90	4.38	4.46	5.71	1.77	4.88	**4.04**

4. The use of asymmetric hierarchical predictors

It is possible to increase the efficiency of noise-trend predictors by considering in the first pass of each layer the values of adjacent, previously processed elements on the left and top of the same passage, and in the second pass – the two closest processed diagonal values in the previous line (in Figure 4 they are denoted as *ab* and *bc*, respectively). The colors of the pixels of the images are often similar to the colors of close pixels horizontally or vertically, and much less often – to the colors of close pixels diagonally. This (and not only the greater distances to the predicted element) explains much lower efficiency of the use of symmetrical predictors in the first pass relative to the second for each layer of an arbitrary image (see Table 2). It is impossible to consider the values of similar elements on the right and bottom relative to the element *X* in the first passes, because when applying predictors to this element they have not yet been processed (which is why the predictors described below are asymmetric).

Let us consider the principles of forecasting two asymmetric noise-trend predictors developed by us. The first, *ProgresPredict3*, returns the closest value to the basic arithmetic mean not only among the four closest adjacent processed elements *a, b, c, d* as *ProgresPredict2*, but also additionally among *ad* and *ab*, giving them an advantage over other elements at the same distances. To speed up the calculations, the smallest distance to the basic arithmetic mean is determined implicitly, using the fact that the closer to the middle between the points *x* and *y* ($x \le y$) the point *z* is, the smaller the difference is $|(y-z)-(z-x)| = |y-2z+x|$.

The second of the asymmetric predictors, *ProgresPredict4*, determines the smallest absolute diagonal increment relative to the elements *ad* and *ab* and returns the value in the direction of this increment relative to the element *X*, if this increment is less than half of the smaller of the diagonal increments of the nearest elements ($\min(|a-c|,|b-d|)/2$). In this case, among two identical diagonal increments, preference is given to that which is associated with a smaller rectilinear increment around the element *X*. Otherwise, the predictor among the four nearest adjacent processed elements *a, b, c, d* determines the three closest to the basic arithmetic mean (most likely belonging to one image object) and returns the average value among them.

In C, these asymmetric hierarchical predictors are written as follows:

```
ubyte ProgresPredict5(ubyte a, ubyte b, ubyte c, ubyte d, ubyte ab, ubyte bc)
 {ubyte absac, absbd, maxac, minac, maxbd, minbd, prognozn, s, mins;
  if (a<=c)   {absac=c-a; maxac=c; minac=a; }
  else   {absac=a-c; maxac=a; minac=c; }
  if (b<=d)   {absbd=d-b; maxbd=d; minbd=b; }
  else   {absbd=b-d; maxbd=b; minbd=d; }
  if (absac<=absbd)
   {prognozn=ab; mins=abs(maxac+minac-2*ab);
    s=abs(maxac+minac-2*bc);
    if (s<mins)   {prognozn=bc; mins=s; }
    s=abs(maxac+minac-2*minbd);
    if (s<mins)   {prognozn=minbd; mins=s; }
    s=abs(maxac+minac-2*maxbd);
    if (s<mins)   {prognozn=maxbd; mins=s; }
    if (minac>=prognozn || prognozn>maxac)   return minac;
    return prognozn; }
  else
   {prognozn=ab; mins=abs(maxbd+minbd-2*ab);
    s=abs(maxbd+minbd-2*bc);
    if (s<mins)   {prognozn=bc; mins=s; }
    s=abs(maxbd+minbd-2*minac);
    if (s<mins)   {prognozn=minac; mins=s; }
    s=abs(maxbd+minbd-2*maxac);
    if (s<mins)   {prognozn=maxac; mins=s; }
    if (minbd>=prognozn || prognozn>maxbd)   return minbd;
    return prognozn; }}
```

```
ubyte ProgresPredict6a(ubyte a, ubyte b, ubyte c, ubyte d, ubyte ab, ubyte bc)
 {ubyte absac, absbd, maxac, minac, maxbd, minbd, prognozn, minprD, minprGV, prD,
prGV;
  if (a<=c)   {absac=c-a; maxac=c; minac=a; }
  else   {absac=a-c; maxac=a; minac=c; }
  if (b<=d)   {absbd=d-b; maxbd=d; minbd=b; }
  else   {absbd=b-d; maxbd=b; minbd=d; }
  minprD=abs(ab-a); minprGV=abs(a-d); prognozn=d;
  prD=abs(ab-b);
  if (prD<=minprD)
   {prGV=abs(b-c);
    if (prD<minprD || prGV<minprGV)   {minprD=prD; minprGV=prGV; prognozn=c; }}
  prD=abs(bc-b);
  if (prD<=minprD)
   {prGV=abs(a-b);
    if (prD<minprD || prGV<minprGV)   {minprD=prD; minprGV=prGV; prognozn=a; }}
  prD=abs(bc-c);
  if (prD<=minprD)
   {prGV=abs(d-c);
    if (prD<minprD || prGV<minprGV)   {minprD=prD; minprGV=prGV; prognozn=d; }}
  if (2*minprD<min(absac, absbd))   return prognozn;
  return ProgresPredict2(a, b, c, d); }

ubyte ProgresPredict6b(ubyte a, ubyte b, ubyte c, ubyte d, ubyte ab, ubyte bc)
 {UBYTE1 absac, absbd, maxac, minac, maxbd, minbd, prognozn, minpr1, minpr2, pr1, pr2;
  if (a<=c)   {absac=c-a; maxac=c; minac=a; }
  else   {absac=a-c; maxac=a; minac=c; }
  if (b<=d)   {absbd=d-b; maxbd=d; minbd=b; }
  else   {absbd=b-d; maxbd=b; minbd=d; }
  minpr1=abs(ab-b); minpr2=abs(b-c); prognozn=a;
  pr1=abs(ab-a);
  if (pr1<=minpr1)
   {pr2=abs(a-d);
    if (pr1<minpr1 || pr2<minpr2)
     {minpr1=pr1; minpr2=pr2; prognozn=b; }}
  pr1=abs(bc-b);
  if (pr1<=minpr1)
   {pr2=abs(a-b);
    if (pr1<minpr1 || pr2<minpr2)
     {minpr1=pr1; minpr2=pr2; prognozn=c; }}
  pr1=abs(bc-c);
  if (pr1<=minpr1)
   {pr2=abs(c-d);
    if (pr1<minpr1 || pr2<minpr2)
     {minpr1=pr1; minpr2=pr2; prognozn=b; }}
  if (minpr1+minpr2<min(absac,absbd)) return prognozn;
  return ProgresPredict2(a, b, c, d); }
```

The results of the application of the proposed asymmetric hierarchical predictors to reduce the entropy of the pixel components of the ACT image set are given in table 3. From the data in this table, we see that the use of the asymmetric predictor *ProgresPredict5* slightly reduces the entropy of images without noise, and the predictor *ProgresPredict6* significantly reduces the entropy of the image № 1 and compresses the rest of the images worse. It is clear that asymmetric predictors are used to predict the values of six rather than four previously processed elements and are therefore are calculated much longer. Such predictors can be used to provide maximum compression, but they are not suitable for graphic formats where fast decoding is required.

Table 3
Entropy of brightness values of pixel components of images of the ACT set after application of asymmetric hierarchical predictors, bpb

Predictor	Layer	Passageway	№ file								Middle entropy
			1	2	3	4	5	6	7	8	
ProgresPredict5	2	1	7.37	4.61	6.86	6.70	7.10	7.19	5.99	7.27	6.64
	2	2	7.07	4.67	6.32	6.55	6.21	7.05	5.39	7.12	6.30
	3	1	7.01	4.48	6.28	6.43	6.49	7.13	5.29	6.89	6.25
	3	2	6.67	4.59	5.88	6.08	5.81	6.87	4.30	6.51	5.84
	k+1	1	4.90	1.59	4.87	4.30	4.51	5.70	1.67	4.75	4.04
	k+1	2	2.47	1.08	4.57	3.86	3.99	5.13	1.03	4.20	3.29
	Together		3.91	1.62	4.82	4.24	4.37	5.89	1.60	4.66	**3.89**
ProgresPredict6 & ProgresPredict6a	2	1	7.44	4.61	7.03	6.85	7.08	7.19	5.99	7.44	6.70
	2	2	7.17	4.67	6.50	6.69	6.32	7.11	5.36	7.27	6.39
	3	1	7.19	4.48	6.62	6.61	6.65	7.27	5.52	7.12	6.43
	3	2	6.77	4.59	6.08	6.20	6.01	6.96	4.48	6.69	5.97
	k+1	1	2.55	1.80	5.16	4.71	4.78	6.02	1.44	5.21	3.96
	k+1	2	0.04	1.41	4.78	4.14	4.19	5.40	1.13	4.53	3.20
	Together		**2.12**	1.87	5.06	4.56	4.59	5.82	1.68	5.01	**3.84**

5. Complex application of hierarchical predictors

The results of the study indicate that predictors should be used only from the passage and the layer when their application begins to reduce entropy, and among the predictors should be chosen the one that provides the lowest entropy after its usage. Moreover, the level of influence and the nature of noise and trends for different fragments of the image may differ, so they may be effective for different hierarchical predictors. We emphasize that the use of predictors should be abandoned for the whole passage, and not for individual fragments, because the values of the brightness of the image components without and after the use of predictors have different nature of uneven distribution (see Figure 1). Therefore, in the process of progressive hierarchical compression for each pass of the next layer, it is advisable first to determine whether the use of predictors generally reduces the entropy of the brightness of its pixel components. Then, if you decide to use predictors, choose for each homogeneous fragment of pixels minimizes its entropy.

We used basic symmetric hierarchical predictors to decide on the use of predictors **on each pass of the next layer**, as the first of them focuses on photorealistic images and is processed faster than *ProgresPredict3*, and the second predictor provides significantly better compression for synthesized images than *ProgresPredict4*. If they reduce the entropy **for each row of the next pass**, we chose the most efficient among the symmetric predictors and additionally – with each of the asymmetric hierarchical predictors. The results of the usage of these methods of combining hierarchical predictors to reduce the entropy of the pixel components of the ACT image set are given in table 4-6. For comparison, at the beginning of these tables there are the results of a similar application to the same images of combinations of consecutive predictor of PNG format and all consecutive predictors listed above, which provide the lowest entropy for each individual row. The test was performed on a computer with an Intel Pentium 4 processor clocked at 3 GHz and RAM 4 Gb. Let us emphasize that in these experiments, **context-dependent algorithms** that eliminate repetition between individual fragments of images were **not used**.

As evidenced by the data of these tables and tables 2-3, the use of combinations of all symmetric hierarchical predictors reduces the entropy relative to the most effective of the individual studied predictors on average by a set of ACT by more than 0.22 bpb mainly due to the synthesized images. The additional use of asymmetric predictors, on average, slows down encoding by 14-17% and decoding by 5%, but reduces entropy by a maximum of 0.1 bpb. Additional application of *ProgresPredict5* reduced the entropy of only the synthesized image № 2, but this advantage is offset by the context-dependent algorithm. Therefore, for fast hierarchical compression of images, it is advisable to use a combination of symmetrical hierarchical predictors, and to ensure maximum compression it is better to use an additional combination of symmetrical predictors and asymmetric predictor *ProgresPredict6*.

Table 4
The entropy of brightness values of components of pixels of representing the set of ACT after the usage of different combinations of predictors, bpb

Combination	№ file								Middle entropy
	1	2	3	4	5	6	7	8	
Sequential predictors of PNG format	1.89	1.46	4.81	4.23	4.28	5.49	1.33	4.61	3.51
All sequential predictors	1.89	1.46	4.80	4.15	4.20	5.43	1.33	4.50	3.47
Basic symmetric predictors	2.76	1.73	4.75	4.02	4.24	5.44	1.62	4.43	3.63
Additional symmetric predictors	4.22	1.77	4.69	4.00	4.19	5.43	1.75	4.42	3.81
All symmetric predictors	**2.75**	**1.70**	**4.69**	**3.98**	**4.18**	**5.42**	**1.60**	**4.41**	**3.59**
All symmetric predictors and ProgresPredict5	2.73	1.51	4.69	3.98	4.18	5.42	1.46	4.41	3.55
All symmetric predictors and ProgresPredict6	**2.10**	**1.63**	**4.69**	**3.98**	**4.18**	**5.42**	**1.46**	**4.41**	**3.49**

Table 5
Time of encoding of pixels of representing the set of ACT by different combinations of predictors, s

Combination	№ file								Middle time
	1	2	3	4	5	6	7	8	
Sequential predictors of PNG format	1.81	2.37	0.56	0.95	0.56	1.04	0.93	0.98	1.15
All sequential predictors	1.99	2.75	0.66	1.21	0.67	1.15	1.09	1.14	1.33
Basic symmetric predictors	1.52	2.08	0.55	0.70	0.52	0.83	0.80	0.74	0.97
Additional symmetric predictors	1.74	2.57	0.63	0.78	0.66	0.78	1.04	0.78	1.12
All symmetrical predictors	1,85	2,42	0,60	0,97	0,58	0,97	0,96	0,96	1.16
All symmetric predictors and ProgresPredict5	2.00	2.77	0.70	1.11	0.66	1.11	1.08	1.10	1.32
All symmetric predictors and ProgresPredict6	2.06	2.84	0.70	1.15	0.69	1.15	1.13	1.15	1.36

Table 6
Time of decoding of pixels of representing the set of ACT by different combinations of predictors, s

Combination	Number of layers	№ file								Middle time
		1	2	3	4	5	6	7	8	
Sequential predictors (both PNG and all fornats)	4, all	0.59	1.34	0.31	0.56	0.30	0.68	0.39	0.34	0.56
Basic symmetric predictors	4	0.07	0.04	0.02	0.04	0.02	0.04	0.05	0.03	0.04
	all	0.76	1.49	0.30	0.55	0.29	0.50	0.59	0.52	0.63
Additional symmetric predictors	4	0.08	0.04	0.02	0.04	0.03	0.03	0.05	0.04	0.04
	all	0.84	1.51	0.31	0.49	0.43	0.48	0.50	0.60	0.65
All symmetric predictors	4	0.09	0.03	0.02	0.04	0.03	0.03	0.05	0.05	0.04
	all	0.81	1.43	0.32	0.49	0.40	0.48	0.50	0.59	0.63
All symmetric predictors and ProgresPredict5	4	0.15	0.06	0.03	0.03	0.03	0.05	0.04	0.05	0.06
	all	0.89	1.56	0.39	0.46	0.40	0.54	0.53	0.58	0.67
All symmetric predictors and ProgresPredict6	4	0.10	0.04	0.03	0.03	0.03	0.04	0.04	0.04	0.04
	all	0.91	1.57	0.38	0.44	0.35	0.59	0.53	0.51	0.66

The use of hierarchical predictors, although on average provides greater entropy than consecutive ones by 0.13 bpb for synthesized images but achieves the best values for this indicator at 0.08 bpb for photorealistic images. And most importantly: the progressive hierarchical way of traversing pixels allows you to perform decoding much faster than sequential traversal, when the size of the output area is much smaller than the size of images (for example, to fill the output area 128 x 128 pixels (4 layers) such decoding is performed 10 times faster (see Table 6), because the time of hierarchical decoding depends on the smaller among the size of the output area and the size of the image (this was emphasized in [3, p. 175]), and consistent – only on the size of the image. Let us also note that the PNG format should be supplemented with a sequential predictor *MedPredict*, which will significantly reduce the entropy of photorealistic images (for example, the ACT set – by 0.07 bpp).

6. Conclusions and prospects of subsequent researches

1. In the new versions of graphic formats and new lossless image compression formats, it is advisable to implement progressive hierarchical compression, as this allows you to speed up decoding significantly when the size of the output area is smaller than the image size.
2. Reducing the size of images compressed in a progressive hierarchical manner is achieved mainly on the last layers, because the pixels used in the predictors have the highest level of correlation with the predicted pixel on average relative to other layers.
3. In the process of hierarchical compression to increase the efficiency of symmetrical predictors, which consider the brightness of the previously processed four nearest pixels, possibly by taking into account the brightness of the two nearest adjacent pixels processed in the same pass.
4. During context-independent lossless compression for homogeneous image fragments, it is advisable to choose the predictor from several alternatives that allows to minimize the entropy.

In the future, in order to additionally reduce the size of compressed image files without losses and speed up decoding, we plan to adapt context-sensitive compression methods [4, 11, 12] to the hierarchical traversal of image pixels and increase the efficiency of symmetric and asymmetric predictors using different color models [13].

7. References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, Third Edition, Vol. 1, Dialektika, Kiyv, 2020, 648 p. (In Ru).

[2] J. Miano, Compressed image file formats: JPEG, PNG, GIF, XBM, BMP, ACM Press, New York, 1999: Triumph, Moskow, 2003, 336 p., Series: Practice of programming.

[3] D. Selomon, Compression of data, images and sound, Tekhnosfera, Moskow, 2006, 368 p., Series: World of programming: digital signal processing.

[4] A. Shportko, Use of predictors in the process of progressive hierarchical context-independent lossless image compression, Proceedings of the National University "Lviv Polytechnic" 771 (2013) 354-364, Series: Information Systems and Networks.

[5] R. Gonsales, R. Vuds, Digital processing of images, Tekhnosfera, Moskow, 2005, 1072 p.

[6] D. Y. Bredihin, Graphics compression without quality loss, 2004. URL: http://www.compression.ru/download/articles/i_lless/bredikhin_2004_lossless_image_-compression_doc.rar.

[7] Y. Prett, Digital processing of images, the World, Moskow, 1982, Book 2, 480 p.

[8] D. Vatolin, A. Ratushnyak, M. Smirnov, Yu. Yukin, Methods of compression of data. device of archivings, compression of images and video, DIALOG-MIFI, Moskow, 2003, 384 p.

[9] H. D. Kotha1, M. Tummanapally, V. K. Upadhyay, Review on Lossless Compression Techniques, International conference on computer vision and machine learning, J. Phys.: Conf., Ser. vol. 1228,012007 (2019). doi: 10.1088/1742-6596/1228/1/012007.

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

[11] 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.

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

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