entropy

MDPI

Article

Extended Multi WLS Method for Lossless Image Coding

Grzegorz Ulacha [1,*], Ryszard Stasiński [2] and Cezary Wernik [1]

[1] Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, ul. Żołnierska 49, 71-210 Szczecin, Poland; cwernik@wi.zut.edu.pl
[2] Department of Informatics and Telecommunications, Poznań University of Technology, ul. Piotrowo 3, 60-965 Poznań, Poland; ryszard.stasinski@put.poznan.pl
* Correspondence: gulacha@wi.zut.edu.pl

Received: 8 July 2020; Accepted: 19 August 2020; Published: 22 August 2020

Abstract: In this paper, the most efficient (from data compaction point of view) and current image lossless coding method is presented. Being computationally complex, the algorithm is still more time efficient than its main competitors. The presented cascaded method is based on the Weighted Least Square (WLS) technique, with many improvements introduced, e.g., its main stage is followed by a two-step NLMS predictor ended with Context-Dependent Constant Component Removing. The prediction error is coded by a highly efficient binary context arithmetic coder. The performance of the new algorithm is compared to that of other coders for a set of widely used benchmark images.

Keywords: image coding; lossless coding; arithmetical coder

1. Introduction

Image compression methods can be divided into lossy and lossless. Important applications of lossless image and video compression techniques include archiving of 2D, 3D, and 4D (3D video sequences) medical images [1–5], as well as astronomical and satellite ones [6,7]. Lossless mode is often required at some stage of photographs, advertising materials, TV productions, and films graphic processing (post-production [8]), etc. In such cases, lossy versions of compression methods, such as JPEG, JPEG2000 (for static images) [9], MPEG2, and MPEG4 [10] (for video sequences), cannot be used. Although these standards have appropriate lossless modes, they are not particularly efficient.

Samples of multimedia signals are usually strongly correlated, which means that simple entropy coders do not compress them effectively. Correlation can be significantly reduced by signal prediction, but then another problem emerges: changes in signal statistical properties. In the 1990s, several solutions were proposed that used linear and non-linear prediction for lossless image compression. The first image coder that solved the above problems relatively well was the context coder CALIC [11]. At that time, it was considered too complex for practical purposes; nevertheless, the standardized JPEG-LS was a context coder too [12]. Then followed more efficient but also more complex techniques: TMW^{Lego} [13], Multi-WLS [14], MRP 0.5 [15]. The more recent best algorithms are Blend-20 [16], and then the improved versions of MRP 0.5: GPR-BP [17], and MRP-SSP [18]. Further analysis of currently existing solutions is presented in Section 1.2.

In this paper, the currently best image lossless coding algorithm is presented, extended multi weighted least squares (EM-WLS; Section 3.2), and its simplified version, locally adaptive ordinary least squares (LA-OLS; Section 3.1). The algorithms have in common the cascade structure proposed in Section 2; their first stages are either EM-WLS, or LA-OLS predictors. Section 6.4 shows that it is much better than any older method, including the newest GPR-BP, and MRP-SSP. EM-WLS is an extended version of the WLS technique, which is also a basis for the very good multi-WLS algorithm [14].

The first version of AVE-WLS was introduced [19]; the current version using the cascade prediction system has been significantly improved (Section 2). Several new formulas are added, i.e., additional NLMS stages (Section 3.3), enhanced cumulated prediction error cancellation stage (Section 4), and a new arithmetic coder (Section 5).

1.1. Basics of Prediction Coding

Optimization of data encoding consists of minimizing the average number of bits per single symbol generated by a source S (in the case of lossless, image compression symbols are pixel color values; in this paper, we work with 8-bit luminance).

Depending on the sources, we can divide them into those without memory (discrete memoryless source (DMS)) and sources with memory (conditional source model (CSM)) [20]. Considering this classification from the Markov model point of view, in the first case, the lower limit on the bit average is the unconditional entropy ($H(S_{\mathrm{DMS}})$, also named zero-order entropy). In the second case, we deal with the conditional entropy of the kth order, defined as $H(S|C^{(k)})$, where in the case of images context, $C^{(k)}$ may be defined, e.g., by k neighbor pixels (Figure 1).

In general, for the source entropy $H(S)$, the following relation holds: $H(S) \leq H(S|C^{(k)}) \leq H(S_{\mathrm{DMS}})$. Since image samples take on many values (at least 256), it is impractical to determine and apply a Markov model for them. It is usually assumed that removal of inter-dependencies between pixels is possible by using predictive techniques; prediction errors are coded. A linear predictor of order r is used to estimate the value of a sample:

$$\hat{x}(0) = \sum_{i=1}^{r} w(i) \cdot P(i), \tag{1}$$

where $P(i)$ is pixel from the currently coded pixel $x(0) = P(0)$ neighborhood (Figure 1), and $w(i)$ is a predictor coefficient from vector $\mathbf{w} = [w(1), w(2), ..., w(r)]$ [21]. To simplify notation in this work, indices of the currently coded pixel are usually omitted, while indices i of pixels $P(i)$ and prediction errors $e(i)$ show their distances from the processed element $P(0)$ or $e(0)$ (Figure 1):

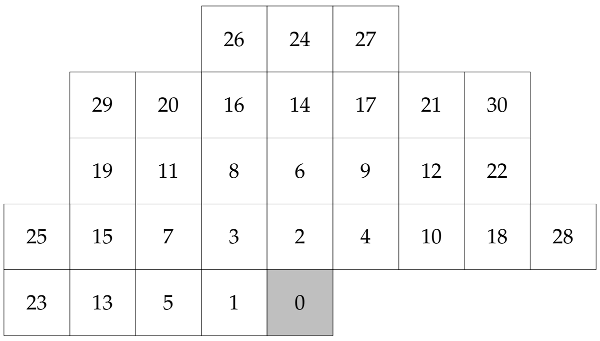

Figure 1. Numbering of neighborhood pixels or errors of $P(0)$ or $e(0)$.

The closest pixels provide the most information about the coded one, so the neighboring pixels are ordered in accordance with their Euclidean distances from $P(0)$ (Figure 1). The numbering of pixels with the same distance from $P(0)$ is usually determined clockwise (this is discussed in more detail in Section 3.2.3). The prediction model can be a linear model of the kth order or a more complex nonlinear solution, but we reduce it to a predictor of one value $\hat{x}(0)$, which is then subtracted from the current $P(0)$, see Formula (2). In this way, we create a not explicitly defined first-order Markov model. As a result, a sequence of prediction errors can be expected to form a source for which the first-order entropy value is noticeably smaller than the zero-order value. The estimated pixel value (expected value rounded to the closest integer) is subtracted from the real pixel value:

$$e(0) = x(0) - \lfloor \hat{x}(0) - 0.5 \rfloor, \tag{2}$$

and the difference (prediction error $e(0)$) coded. In this way, we obtain a differential image, in which probability distribution of its samples is close to the Laplace distribution [21,22]. This allows efficient coding of prediction errors using one of the static or adaptive entropy methods, among which arithmetic coding is most effective (Section 5).

Notably, it is extremely difficult to determine image entropy $H(S)$ because it is hard to define a Markov model that considers all interdependencies between pixels. Namely, the dependencies may extend quite far. An example is the research use of linear prediction of order $r_2 = 106$ in the second stage of the proposed-here cascade structure (Section 3.3). The number of Markov model states grows exponentially with its order. Here, if we assume that symbols are 8-bit pixel values, the number is at least 256^{106}. Therefore, only the bit average, being the average number of bits necessary for coding a pixel, is used as the basic benchmark for testing lossless codecs (Section 6).

1.2. Adaptive Predictive Techniques

The effectiveness of image compression depends on the correct choice of one or more prediction models. Linear prediction is most often used (Formula (1)), where a vector $\mathbf{w}$ containing r predictor coefficients $w(i)$ is usually determined by minimizing the mean square error (MMSE). This vector can be determined once for the whole image by a forward adaptation method or individually for each coded pixel using a backward adaptation technique. The advantage of backward adaptation is the possibility of using relatively high prediction orders (there is no need to provide prediction coefficients to the decoder), which allows for high compression efficiency. However, the disadvantage of this solution is the necessity of updating prediction coefficients in both the encoder and decoder for each pixel, which is a time-symmetric approach.

In methods with forward adaptation, it is reasonable to divide images into blocks (e.g., 8×8 or 16×16 pixels) and define an individual predictive model for each. One of the first solutions of this type was the method presented by Memon [23], where one of eight fixed models was assigned to each block of 8×8 pixels, producing the smallest absolute error. As such, the header information associated with a block required only three bits, forming the model number.

Subsequent methods used the mean square error for determining the best prediction coefficients. Unfortunately, this was associated with very large header information, as predictor coefficients required large numbers of bits. To reduce the size of the header, the blocks with similar characteristics were grouped into clusters, and all were associated with a common prediction model [24]. Through using vector quantization techniques (as well as fuzzy clustering [25]), optimized sets of, e.g., 16 prediction models were created, so that even for a large prediction order, the header size did not significantly increase the bit average. Matsuda et al. [15] used a technique of combining adjacent blocks belonging to the same category into groups (associated with the same predictor) to create larger blocks. Next, a map of the blocks with variable sizes was saved using an effective technique of encoding quadtrees.

In the above-mentioned above block techniques, it is easy to prove that it is possible to reach lower values of first-order entropy than for the MMSE method [15]. In the paper [26], it was proposed to replace MMSE criterion with minimum mean absolute error (MMAE), which improved the results for a block method. The poblem of discrepancy between MMSE and first-order entropy criterions was discussed in paper [27].

In the case of backward adaptation in local training windows, the MMSE criterion is closer to optimal one than for forward adaptation approaches. Analysis of this issue is presented in Section 6.1. This type of solution was used in the lossless image codec proposed in this work.

2. Cascade Prediction Model

Considering the features of adaptive methods described in Section 1.2, to achieve the highest possible compression efficiency in our work, we decided to use a backward adaptation approach, and developed a highly effective cascade prediction model whose final high-efficiency form is discussed in this section. A similar approach was used earlier in lossless audio compression solutions [28]. The cascade is calculated as follows: In the main stage image, pixels are processed:

$$y_1(0) = \sum_{i=1}^{r_1} w_{MP}(i) \cdot P(i), \tag{3}$$

where coefficients of the predictor form vector $\mathbf{w}_{MP}$ of order r_1. They are computed using the WLS or OLS methods described in Sections 3.1 and 3.2, respectively. In the following stages, prediction errors from previous stages are transformed:

$$y_j(0) = \sum_{i=1}^{r_j} w_j(i) \cdot e_{j-1}(i), \text{ for } j > 1, \tag{4}$$

$$e_1(0) = P(0) - y_1(0), \tag{5}$$

$$e_j(0) = e_{j-1}(0) - y_j(0), \text{ for } j > 1. \tag{6}$$

The final prediction error is obtained at the output of a cascade structure shown in Figure 2. It is given by:

$$e(0) = P(0) - \lfloor y_1(0) + y_2(0) + y_3(0) + C_{\text{mix}} + 0.5 \rfloor, \tag{7}$$

where $y_1(0)$ and $y_{2/3}(0)$ are signal estimates provided by the Main Predictor, the locally adaptive Ordinary Least-Squares (OLS) method (LA-OLS, Section 3.1) or the extended multi WLS method (EM-WLS, Section 3.2), and then by Normalized Least Mean Square predictors (NLMS+, Section 3.3) respectively. Finally, C_{mix} is the cumulated prediction error correction defined in Section 4 (Context-Dependent Constant Component Removing (CDCCR)). The last two blocks (Golomb code and Context Adaptive Binary Arithmetic Coder (CABAC)) are used for effective compression of predictive errors. Details are provided in Section 5.

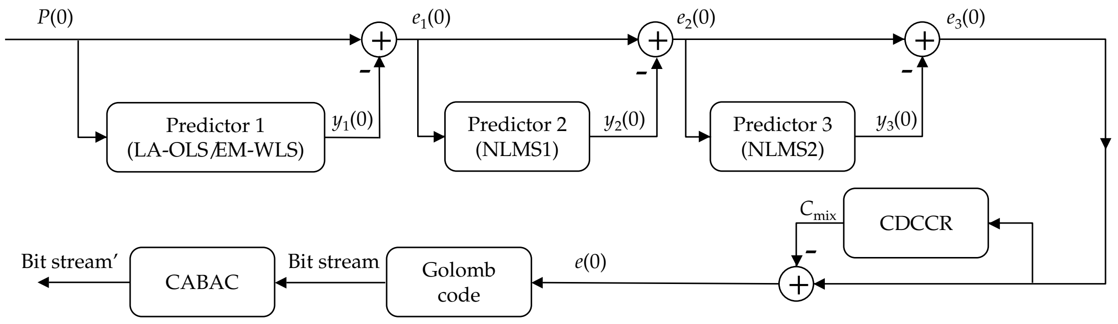

Figure 2. Cascade of predictors forming the data modelling part of the coder.

The proposed solution can be considered to be a model offering the highest efficiency. To reduce implementation complexity (Section 6.4), individual blocks can be easily modified (e.g., by reducing prediction orders in the first 3 blocks) or even deleted. For example, block Predictor 1 may contain one of two versions of the Main Predictor: EM-WLS or the faster LA-OLS. The blocks are described in the following sections, and the effects of blocks removal are discussed in more detail in Section 6.

3. Stages of Adaptive Predictive Cascade

In this section, we present the blocks in the first three stages of the cascade structure from Figure 2. In particular, two versions of the Main Predictor are defined: complex but more efficient EM-WLS and simplified LA-OLS.

3.1. Locally Adaptive OLS Method

In Section 1.2, we list examples of codecs using predictive methods with forward adaptation. Among them, the most effective are those that adapt the prediction model to local features of an image, e.g., for 8×8 or 16×16 pixel blocks. In the case of backward adaptation, similarly sized Q windows are used. The main difference is that the currently encoded pixel $P(0)$ is not in that Q windows.

In the OLS method, prediction coefficients are calculated for each coded pixel, minimizing the prediction mean square error in a certain limited area Q. The vector of OLS predictor coefficients $\mathbf{w}_{MP}$ (Main Predictor) is computed from the following formula [29]:

$$\mathbf{w}_{MP} = \mathbf{R}^{-1} \cdot \mathbf{P}, \tag{8}$$

where $\mathbf{R}$ is the experimental signal autocovariance matrix:

$$\mathbf{R}(j,i) = \sum_{y \in Q} \sum_{x \in Q} \Psi_{(y,x)} \cdot P_{(y,x)}(i) \cdot P_{(y,x)}(j), \tag{9}$$

and vector $\mathbf{P}$ is:

$$\mathbf{P}(j) = \sum_{y \in Q} \sum_{x \in Q} \Psi_{(y,x)} \cdot P_{(y,x)}(0) \cdot P_{(y,x)}(j), \tag{10}$$

where Ψ is a weighting function, which takes a constant value of 1 for classic OLS; pixels P are taken from a training window Q around the coded pixel located at position (y, x) (Figure 3). It consists of W image sub-rows of size $2W + 1$ preceding the estimated pixel and W pixels preceding it in the current row (Q is an area extending W pixels to the left and up from the coded pixel $P(0)$ and to the right in rows preceding it). The best results were obtained for $W = 10, r = 18$ (Section 6.2).

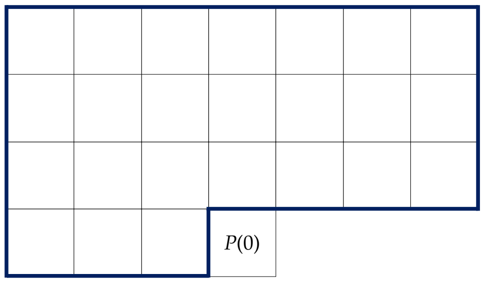

Figure 3. Training window Q for window parameter $W = 3$.

The default vector $\mathbf{w}_{MP} = [0.620, 0.625, -0.125, 0.125, -0.125, -0.125]$ is used in border areas when $\mathbf{R}$ is ill-conditioned. With the Q training window defined in this way and coding of successive pixels row by row using a fast sliding window procedure, the adaptation of the $\mathbf{R}$ matrix and vector $\mathbf{P}$ is performed relatively fast in comparison to the WLS method discussed in Section 3.2. It involves deleting data obtained from the extreme left column of the Q area and including the new extreme right column of the training window [30].

In the paper, the improved version of this scheme is analyzed. Firstly, a more robust version of Formula (8) is used [31,32]:

$$\mathbf{w}_{MP} = (\mathbf{R} + u_{\text{bias}} \cdot \mathbf{I})^{-1} \cdot \mathbf{P}, \tag{11}$$

where $u_{\text{bias}} = 800$ (term $u_{\text{bias}}\mathbf{I}$ guarantees non-singularity of (9)).

Secondly, predictor coefficients obtained from (11) are weighted by coefficients:

$$\widetilde{w}(j) = w(j) \cdot \frac{\sqrt[4]{\overline{d}_j}}{\sum_{i=1}^{r} \sqrt[4]{\overline{d}_i}}, \tag{12}$$

where $\overline{d}_j = 1/\sqrt{(\Delta x_j)^2 + (\Delta y_j)^2}$ is the inverse Euclidean distance from the currently coded pixel. The new coefficients substitute $w(j)$ in (3), which is a novel idea introduced in this algorithm.

Another improvement proposed in this work is the introduction of a weighting function promoting pixels, for which prediction errors at their positions in window Q are small:

$$\Psi_{(y,x)} = \frac{1}{4 + |e_{(y,x)}(0)|}. \tag{13}$$

In this case, $u_{\text{bias}} = 100$. As such, the weighted sum $\sum\limits_{y \in Q} \sum\limits_{x \in Q} \Psi_{(y,x)} \cdot e^2_{(y,x)}(0)$ of squared errors is minimized, similar to the WLS method described in Section 3.2.

3.2. Multi WLS Method

The use of a more complex weighting function compared to Formula (13)) requires a significant increase in implementation complexity at the stage of determining the **R** matrix and **P** vector. EM-WLS is based on the WLS concept, so this section begins with its description.

3.2.1. Weighted Least-Squares (WLS)

In general, the vector of WLS predictor coefficients $\mathbf{w}_{MP}$ is computed from Formulas (9)–(11) [21]. The weighting function Ψ reflects similarity between neighborhoods of size m around the coded $P(0)$ and some other $P_{\text{off}}(0)$ pixels [14,19] (Figure 4). The initial form of the function was relatively simple [14]:

$$\Psi = \frac{1}{1 + \sum_{k=1}^{m} (P(k) - P_{\text{off}}(k))^2}. \tag{14}$$

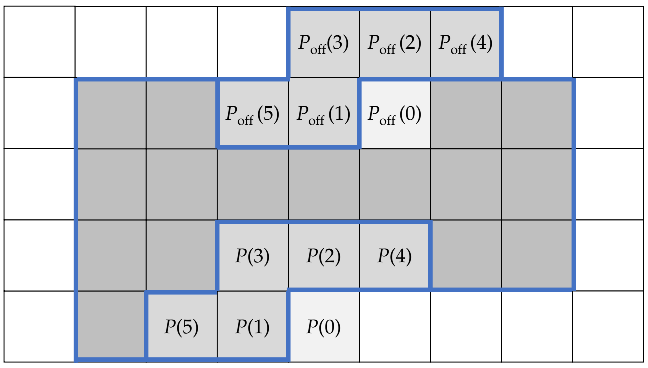

Figure 4. Neighborhoods of pixels $P(0)$, and $P_{\text{off}}(0)$ for $m = 5$ and $W = 3$.

3.2.2. AVE-WLS Method

A previous paper on AVE-WLS [19] followed suggestions from [33,34], as did we, concerning the optimal form of LS predictors. The first one was a statement that for each coded pixel, an optimal predictor order exists [33]. It was not known a priori, so in [19], instead of searching for it, we propose computing the averaged value of the WLS predictor vectors $\mathbf{w}_{MP(j)}$ for orders from $j = r_{\min}$ to $j = r_{\max}$:

$$\mathbf{w}_{\text{AVE-MP}} = \frac{1}{r_{\max} - r_{\min} + 1} \cdot \sum_{j=r_{\min}}^{r_{\max}} \mathbf{w}_{MP(j)}. \tag{15}$$

Implemented here, the predictor orders range from $r_{\min} = 4$ to $r_{\max} = 24$. Additionally, $W = 14$ (Figure 3). If vectors $\mathbf{w}_{MP(j)}$ should be extended to $r_{\max}$, it is achieved by zero-padding.

In this paper, a complex weighting function is proposed. Equation (16) is an enhanced version of the formula from [19]:

$$\Psi_{(y,x)} = \alpha \cdot \frac{\lambda_2 + 0.8^{\sqrt{(\Delta y_0)^2 + (\Delta x_0)^2}}}{\left(\lambda_1 + \sum_{k=1}^{m} \left(\overline{d}_k \cdot (P(k) - P_{\text{off}}(k))\right)^2\right)^{\gamma}}, \tag{16}$$

where $\lambda_1 = 64$, $\lambda_2 = 0.25$, $\overline{d}_k = 1/\sqrt{(\Delta y_k)^2 + (\Delta x_k)^2}$ is the inverse of Euclidean distance between pixels $P(k)$ and $P(0)$, neighborhood size $m = r_{\max}$ (Formula (15)), and $\gamma = 1.18$. The expression appearing as a power of number 0.8 determines the Euclidean distance between pixels $P(0)$ and $P_{\text{off}}(0)$. The scaling factor α depends on two threshold values calculated by Algorithm 1:

Algorithm 1: Algorithm for calculating α weight (Formula (16))

```
1 $t_1 = \max\left(P_{\text{off}}(0) - P_{\text{off}}(i)\right)$, for $i = \{1, 2, -1\}$
  // $P_{\text{off}}(-1)$ indicates the right neighbor of $P_{\text{off}}(0)$)
2 $t_2 = \max\left(P_{\text{off}}(0) - P_{\text{off}}(i)\right)$, for $i = \{3, 4\}$
3 $\alpha = 1$
4 if $t_1 \geq 12$ then
5     $\alpha = t_1^{-0.25}$
6 else if $t_2 \geq 25$ then
7     $\alpha = t_2^{-0.25}$
8 end
```

The second suggestion concerning LS predictors optimization [34] was a proposition to use not all, but only $r_{\max}$ most similar (correlated) pixels taken from a range of r_{ext} pixels, $r_{\text{ext}} > r_{\max}$, in the basic prediction Formula (1). In this paper, we propose to set $r_{\max} = 24$ and $r_{\text{ext}} = 48$, but similarity is tested using only 10 pixels with indices from 15 to 48, and located in Q (Figure 4). The similarity measure here is the smallest cumulative absolute distance of pixels from the Q region:

$$\rho_{\text{dist}}(k) = \sum_{y \in Q} \sum_{x \in Q} \left| P_{(y,x)}(0) - P_{(y,x)}(k) \right|. \tag{17}$$

Apart from the chosen 10 pixels, the remaining ones are the closest to the coded one. The minimization rule (17) does not apply to the 14 nearest pixels; they are considered by default. In this way, vector $\mathbf{Z} = \{z(1), z(2), ..., z(r_{\max})\}$ is calculated from $r_{\max} = 24$ pixels $P(i)$ from the neighborhood of size $r_{\text{ext}} = 48$. Formula (3) takes the form:

$$y_1(0) = \sum_{i=1}^{r_{\max}} w_{\text{AVE-MP}}(i) \cdot z(i). \tag{18}$$

Finally, u_{bias} is also optimized in this paper using ridge regression [35]. Firstly, the initial vector $\mathbf{w}_{\text{AVE-MP}}$ is calculated for $u_{\text{bias}} = 0$, then the following term is computed:

$$u_{\text{bias}} = c \cdot \frac{\sum_{y \in Q} \sum_{x \in Q} \Psi_{(y,x)} \cdot e^2_{(y,x)}(0)}{\sum_{i=1}^{r_{\max}} \left(w_{\text{AVE-MP}}(i) \right)^2}, \tag{19}$$

where $e(0)$ is defined in (2), respectively; $w_{\text{AVE-MP}}(i)$ is the coefficients of the mentioned above initial vector $\mathbf{w}_{\text{AVE-MP}}$. Constant c can be evaluated as $4 \cdot r_{\max}/W^2 \approx 0.5$.

3.2.3. Extended Multi WLS Method

Another idea introduced in this work, apart from the proper selection of the neighborhood (Section 6.3), is replacement of the arithmetic mean of 22 prediction models for successive orders from $r_{\min} = 3$ to $r_{\max} = 24$ (Formula (15)) by a weighted mean whose weights are determined adaptively after coding of each pixel using an improved ALCM+ method. The approach of weighted combination of predictive models (for calculation of the Main Predictor) is called the extended multi WLS method (EM-WLS).

The original version of the Activity Level Classification Model (ALCM) technique [36] was developed in the 1990s for image coding purposes. In comparison to the classic LMS solution, it is characterized by a lower, though similar, computational complexity. Originally, the method operated on linear predictors of the fifth or sixth order. In each iteration, only two predictor coefficients were modified, and only by adding/subtracting a constant ($\mu = 1/256$ for 8-bit samples). Here, the number of coefficients is 22, and up to six properly selected coefficients are modified, hence the name ALCM+.

In the proposed solution, Formula (15) is extended to the following:

$$\mathbf{w}_{\text{ALCM-MP}} = \sum_{j=r_{\min}}^{r_{\max}} \beta_j \cdot \mathbf{w}_{\text{MP}(j)}, \tag{20}$$

where weights β_i are initialized with $1/(r_{\min} - r_{\max} + 1)$, so the sum of weights is 1. Formula (18) takes the form:

$$y_1(0) = \sum_{i=1}^{r_{\max}} w_{\text{ALCM-MP}}(i) \cdot z(i). \tag{21}$$

To determine the updated β_j, it is necessary to calculate predictions $\hat{x}_{MP(j)}(0)$ given by WLS models for orders of predictors from $r_{\min} = 3$ to $r_{\max} = 24$. A 22-element vector $\mathbf{g} = \{\hat{x}_{MP(j)}(0)\}$ is created:

$$\hat{x}_{MP(j)}(0) = \sum_{i=1}^{r_j} w_{MP(j)}(i) \cdot z(i), \text{for } j = \{1, 2, ..., 22\}, r_j = \{3, 4, ..., 24\}. \tag{22}$$

After coding the current pixel $P(0)$, six weights β_j are adapted. For this purpose, the highest three and the lowest values three are taken from the vector $\mathbf{g}$. Let us denote these elements as the smallest:

$$g(q_{(1)}) \le g(q_{(2)}) \le g(q_{(3)}), \tag{23}$$

and the largest:

$$g(p_{(3)}) \le g(p_{(2)}) \le g(p_{(1)}). \tag{24}$$

If condition $g(q_{(1)}) < g(p_{(1)})$ is true, then the following adaptation coefficients are used (Algorithm 2):

Algorithm 2: Adaptation of weights β_j.

```
1 if $y_1(0) < P(0)$ then
2     $\beta^{(n+1)}_{g(p_{(k)})} = \beta^{(n)}_{g(p_{(k)})} + \mu_{(k)}$ , for $k = \{1, 2, 3\}$
3     $\beta^{(n+1)}_{g(q_{(k)})} = \beta^{(n)}_{g(q_{(k)})} - \mu_{(k)}$ , for $k = \{1, 2, 3\}$
4 else if $y_1(0) > P(0)$ then
5     $\beta^{(n+1)}_{g(p_{(k)})} = \beta^{(n)}_{g(p_{(k)})} - \mu_{(k)}$ , for $k = \{1, 2, 3\}$
6     $\beta^{(n+1)}_{g(q_{(k)})} = \beta^{(n)}_{g(q_{(k)})} + \mu_{(k)}$ , for $k = \{1, 2, 3\}$
7 end
```

The modifying value $\mu_{(k)}$ is determined as:

$$\mu_{(k)} = \min\{2^{-12}; \overline{\mu}_{(k)}\}, \tag{25}$$

where:

$$\overline{\mu}_{(k)} = \frac{|P(0) - y_1(0)|}{2^6 \cdot \sum_{j=1}^{3} \alpha_j \cdot \left(g(p_{(j)}) - g(q_{(j)})\right)}, \tag{26}$$

and $\alpha_j = \{1; 0.75; 0.5\}$.

3.3. Normalized Least Mean Square (NLMS) Method

Similar to [28,37], in this paper, we use the cascaded NLMS method for tuning the results obtained from the Main Predictor. This approach allows introduction of high prediction orders while maintaining relatively low implementation complexity. Here we have two cascaded NMLS filters with orders $r_3 < r_2$. The values of prediction coefficients are initially set to 0, i.e., $\mathbf{w}_j^{(n=0)} = [0, ..., 0]$, where j is the prediction stage (for NLMS $j = \{2, 3\}$) of the cascade prediction model shown in Figure 2. The general coefficient update formula for the NLMS method is [21]:

$$w_j^{(n+1)}(i) = w_j^{(n)}(i) + \mu_j(i) \cdot \overline{e}_j^{(n)}(0) \cdot e_{j-1}^{(n)}(i), \text{ for } i = \{1, 2, ..., r_j\}, \tag{27}$$

where

$$\overline{e}_j^{(n)}(0) = \operatorname{sgn}(e_j^{(n)}(0)) \cdot \min\{|e_j^{(n)}(0)|; \varphi\}, \tag{28}$$

is the bounded prediction error. The experimentally found bound $\varphi = 14$.

In the NLMS approach, the learning coefficient $\mu_j(i)$ adapts to the signal. Here, we propose an enhanced formula for it [31]:

$$\mu_j(i) = \frac{\overline{d}_i}{2^3 \cdot \sqrt{\overline{\sigma}^2} \cdot \left(10 + \sum_{k=1}^{r_j} \sqrt{\overline{d}_k} \cdot (e_j^{(n)}(k))^2\right)}, \tag{29}$$

where $e_j^{(n)}(k)$ is the kth signal sample (Figure 1). Here, the prediction error from the preceding stage of the algorithm, $\overline{\sigma}^2$, is the weighted average value of all variances $\widetilde{\sigma}^2$:

$$\widetilde{\sigma}^2 = \frac{1}{\delta} \sum_{k=1}^{m} \overline{d}_k \cdot (P(k) - \widetilde{p})^2, \tag{30}$$

where $m = 10$, $\widetilde{p} = \frac{1}{\delta} \sum_{k=1}^{m} \overline{d}_k \cdot P(k)$ and $\delta = \sum_{k=1}^{m} \overline{d}_k$.

The orders of NLMS predictors used in this paper are $r_2 = 96$, and $r_3 = 30$ for LA-OLS, and $r_2 = 106$ and $r_3 = 42$ for EM-WLS.

4. Cancelling Cumulative Prediction Error

4.1. Context-Dependent Constant Component Removal

It was observed for the first time in [11] that predictors tend to produce a DC error component, which diminished their efficiency. Then, the algorithm for computing the correction value for cancelling this component was constructed. According to Formula (7), in our algorithm, the final pixel estimate and its rounded version are:

$$\dot{x}(0) = y_1(0) + y_2(0) + y_3(0) + C_{\text{mix}}, \tag{31}$$

$$\hat{x}(0) = \lfloor \dot{x}(0) + 0.5 \rfloor, \tag{32}$$

where the components come from the Main Predictor, NLMS1, NLMS2, and the cumulative error correction stages. The latter is computed by using an extended formula from our previous works [38]:

$$C_{\text{mix}} = \sum_{j=1}^{12} \alpha_j \cdot C_j, j = 4(f-1) + k, \tag{33}$$

where C_j components are computed using context-based error bias correction methods mentioned in [38]: there are 4 context definitions for each technique, $k = \{1,2,3,4\}$; hence, $j = \{1,2,3,4\}$ for JPEG-LS, $f = 1$; $j = \{5,6,7,8\}$ for CALIC, $f = 2$; $j = \{9,10,11,12\}$ for the slightly modified median method, and $f = 3$. For $j \leq 8$, C_j is obtained from $C_j = S_{\text{map}}(j)/N_{\text{map}}(j)$, where $S_{\text{map}}(j)$ are elements of $\mathbf{S}_{\text{map}} = [S_{\text{JPEG-LS}_1}(i_1), S_{\text{JPEG-LS}_2}(i_2), S_{\text{JPEG-LS}_3}(i_3), S_{\text{JPEG-LS}_4}(i_4), S_{\text{CALIC}_1}(i_1), S_{\text{CALIC}_2}(i_2), S_{\text{CALIC}_3}(i_3), S_{\text{CALIC}_4}(i_4)]$ being current sums of prediction errors $e_3(0)$ (outputs of NLMS2 stage), where i_k is fixed and indicates the number of contexts chosen for coding of current pixel for the kth technique. $\mathbf{N}_{\text{map}} = [N_{\text{JPEG-LS}_1}(1,i_1), N_{\text{JPEG-LS}_2}(2,i_2), N_{\text{JPEG-LS}_3}(3,i_3), N_{\text{JPEG-LS}_4}(4,i_4), N_{\text{CALIC}_1}(1,i_1), N_{\text{CALIC}_2}(2,i_2), N_{\text{CALIC}_3}(3,i_3), N_{\text{CALIC}_4}(4,i_4)]$ consist of current numbers of appearances of context number i_k. For $j = \{9,10,11,12\}$, C_j is the median $C_{\text{Median}}(i_k)$ obtained as the middle value of a vector $\mathbf{V}_{\text{MED}}(i_k)$ containing up to 128 values of sorted prediction errors $e_3(0)$ for every context number i_k.

From a statistical point of view, careful averaging of uncertain measurements leads to improved measurement results. This is why the constant component is removed as in Formula (33) in this paper. At the same time, the approach reduces variations in histogram shapes, which may manifest in their strong asymmetry: for a given context i, there may be several maximum values $\mathbf{S}_{\text{map}}(j)$ not positioned at its center (Laplace distribution is symmetric and has one maximum) [39].

As in [38], four indices pointing at four context systems are applied to each method, which are described in the next subsection. A novelty is Formula (33) being adaptive:

$$\alpha_j = \frac{\beta_j}{\sum\limits_{i=1}^{12} \beta_i}, \tag{34}$$

where:

$$\beta_j = \overline{\omega}_j \cdot \sqrt[3]{\frac{N_{\text{map}}(j)}{\theta_{\text{map}}(j)}}, \tag{35}$$

where $\theta_{\text{map}}(j)$ is the actual cumulated final square prediction error for index j ($\boldsymbol{\theta}_{\text{map}} = [\theta_{\text{JPEG-LS}}(i_1), \theta_{\text{JPEG-LS}}(i_2), \theta_{\text{JPEG-LS}}(i_3), \theta_{\text{JPEG-LS}}(i_4), \theta_{\text{CALIC}}(i_1), \theta_{\text{CALIC}}(i_2), \theta_{\text{CALIC}}(i_3), \theta_{\text{CALIC}}(i_4), \theta_{\text{Median}}(i_1), \theta_{\text{Median}}(i_2), \theta_{\text{Median}}(i_3), \theta_{\text{Median}}(i_4)]$), which is updated as follows:

$$\theta_{\text{map}}^{(n+1)}(j) = \theta_{\text{map}}^{(n)}(j) + (P(0) - \dot{x}(0))^2. \tag{36}$$

The experimentally found weights were: $\overline{\omega}_j = \{0.275; 0; 0.4; 0.15; 0.2; 0.3; 0.1; 0.35; 0.2; 0.2; 0.325; 0.2\}$; optimization of weights for a particular image is also possible. When a count $N_{\text{map}}(j)$ reaches 128, then it is halved, similar to $S_{\text{map}}(j)$. In this case, the denominator of (35) is scaled: $\theta_{\text{map}}^{(n+1)}(j) = 0.5 \cdot (\theta_{\text{map}}^{(n)}(j) + 1000)$. In the case of median updating when halving the 128-element vector $V_{\text{MED}}(i_k)$ (contains sorted prediction errors $e_3(0)$ for context number i_k), the greatest 32 and the smallest 32 are rejected.

4.2. Contexts for Correcting Prediction Error Bias

Three of four context definitions for the error bias correction methods are similar to those described in [38], but there are some improvements (Sections 4.2.1–4.2.3). The fourth approach is completely new (Section 4.2.4).

4.2.1. Approach for $k = 1$

The context definitions in [11] use values of t samples surrounding the coded pixel $P(0)$, and their mean. For $t = 8$, the samples are $P(1)$, $P(2)$, $P(3)$, $P(4)$, $P(5)$, $P(6)$, GradNorth $= 2P(2) - P(6)$, and GradWest $= 2P(1) - P(5)$. The numbering is the same as in Figure 1. We take an 8-bit word and set each bit to 1 if the corresponding sample value is greater than the mean of the t samples, and 0 in the opposite case. As such, 1 of the 256 context numbers is defined. Additionally, we can quantize the pixel standard deviation to q values; the number of contexts grows to $q \cdot 2^t$. In this paper, $t = 8$ and $q = 4$ are used, which produces 1024 contexts i_1. The arithmetic mean is replaced by the expectation value $\overline{y}_3(0) = y_1(0) + y_2(0) + y_3(0)$. The variance (multiplied by 8) is:

$$\hat{\sigma}^2(0) = (\overline{y}_3(0) - \text{GradNorth})^2 + (\overline{y}_3(0) - \text{GradWest})^2 + \sum_{i=1}^{6} (\overline{y}_3(0) - P(i))^2. \tag{37}$$

Quantization levels were 3 experimentally found thresholds for $\hat{\sigma}^2(0)$: 300, 2000, 8000. A similar idea was presented in [40].

4.2.2. Approach for $k = 2$

The context-defining approach proposed here is similar to that used in JPEG-LS [12]. There are 6 contexts obtained from value ranges of neighbor pixel differences d_1, d_2, and d_3. Ranges are defined by difference signs and 2 thresholds, q_1 and q_2, which give $6^3 = 216$ contexts (a six-state quantizer with 5 thresholds $\{-q_2, -q_1, 0, q_1, q_2\}$). Differences d_j are defined as follows: $d_1 = \overline{y}_3(0) - P(4)$, $d_2 = \overline{y}_3(0) - P(1)$, and $d_3 = \overline{y}_3(0) - P(2)$. In addition, a three-bit number $b_0 b_1 b_2$ is introduced; the first bit b_0 is set if condition $|P(1) - P(5)| > q_3$ is fulfilled as the b_1 sign of $e(1)$ is used. Finally, $b_2 = 1$ if $\overline{y}_3(0)$ is greater than the average of pixels coded up to that moment. The experimentally found thresholds q_j were {5, 18, 20}. This produces a total of $6^3 \cdot 2^3 = 1728$ contexts i_2.

4.2.3. Approach for $k = 3$

The third approach is based on a simplified vector quantization method [41]. Initial data analysis resulted in defining 16 vectors of centroids (each containing 3 pixels from the coded pixel neighborhood: $\mathbf{V} = \{P(1), P(2), P(4)\}$). Centroids are initialized as shown in Algorithm 3.

Algorithm 3: Centroids initialization.

```
1 $\underset{j\in 0;15}{\text{count}}(j) = 1$
2 $\underset{j\in 0;15, i\in 0;2}{\text{centroid}}(j,i) = j \cdot 2^4.$
```

The simplified adaptive method for updating centroids is based on Euclidean distances of the current vector $\mathbf{V} = \{P(1), P(2), P(4)\}$ from all 16 centroids. The 4-bit $(b_0 b_1 b_2 b_3)$ label l of the closest centroid is concatenated with the other 6 bits (defined below) into a 10-bit context number i_2 (i.e., they have 1024 contexts i_3). The lth centroid is adapted as follows:

$$\underset{i\in 0;2}{\text{centroid}}(l,i) := \frac{\text{count}(l) \cdot \text{centroid}(l,i) + \mathbf{V}(i)}{\text{count}(l) + 1}. \tag{38}$$

Then, the counter count(l) is increased by 1.

Next, two bits $b_4 b_5$ of context number i_3 are decisions if the neighboring pixel $P(1)$ and $P(2)$ values are close enough to te expected coded pixel value $\overline{y}_3(0)$: $|\overline{y}_3(0) - P(j)| \geq 7$ and $j = \{1,2\}$. Bits $b_6 b_7$ are obtained from the comparison of $\overline{y}_3(0)$ with values of $P(1)$ and $P(2)$. If $P(j) < \overline{y}_3(0)$, then the bit is zero, $j = \{1,2\}$.

Bit b_8 carries information if the current $\overline{y}_3(0)$ is greater than average of pixels coded up to that moment. Finally, bit b_9 is defined on the basis of the following majority formula: if at least 5 pixels from the set $\{P(3), P(4), P(5), P(6), P(7), P(8), P(9)\}$ are greater than $\overline{y}_3(0)$, then the bit is zero.

4.2.4. Approach for k = 4

This approach is completely new. We start with reorganizing values $\{P(1), P(2), \overline{y}_3(0)\}$ in ascending order; a new set $\{p_{r_1} \leq p_{r_2} \leq p_{r_3}\}$ is obtained. Then, (non-negative) differences are defined: $d_1 = p_{r_2} - p_{r_1}$ and $d_2 = p_{r_3} - p_{r_2}$. The differences are quantized using two thresholds: 5 and 18. There are 6 possible orderings of the set, and 3^2 quantization levels for differences, which produeces 54 combinations. After adding a five-bit number $54 \cdot 2^5 = 1728$, contexts i_4 are obtained. Bit zero of the number (b_0) is set if p_{r_2} (median value) is greater than average of pixels coded up to that moment. The next bit (b_1) is the error $e(1)$ sign. Two following bits $(b_2 b_3)$ respond to two questions linked with pixel $P(4)$: is it lower than $\overline{y}_3(0)$ and is substantially different than $\overline{y}_3(0)$, i.e., $|\overline{y}_3(0) - P(4)| \geq q_3$. The experimentally found q_3 value was 20. Finally, the fifth bit (b_4) is one if $|P(1) - P(5)| \geq q_3$.

5. Context Adaptive Binary Arithmetic Coder (CABAC)

Among the practical applications of prediction errors entropy coders, the most effective are the adaptive arithmetic ones, although various variations of Huffman code are also used, including Rice and Golomb [21]. When measuring the characteristics of prediction errors, it is possible to determine the approximate type of distribution of the currently encoded value $e(0)$ quite well. Based on this assumption, a contextual multi-value arithmetic encoder is usually constructed using not one but t probability distributions associated with context numbers from 0 to $t - 1$. Theoretically, as the number of contexts increases, an improvement in compression efficiency is expected. Initially, we do not know the probability distributions; their shapes emerge when collecting incoming data. Hence, for large t, the problem of context dilution arises, i.e., for too long time shapes of some, probability distributions are not formed. We need a quick determination of their approximate target form. Therefore, a compromise should be found between the number of contexts and the speed of probability distribution adaptation. Most often, 8 [11], 16, or even 20 contexts are used [42]. An analysis of the influence of the number of contexts on bit average was presented [43].

Another approach consists of assuming an initial knowledge of the approximate probability distributions for each context. In this case, an arithmetic encode should be built that can determine which histogram is updated by the coded pixel.

This is a complex problem as it requires finding the best possible mathematical description of distributions matching a given image or class of images. This approach was analyzed based on Laplace, Gaussian, Student's t, and Generalized Gaussian Distribution (GGD) [22]. The conclusions were followed [44], where a GGD was used. A similar solution using Student's t distribution was also used in the TMW method [45], where a principle of blending probability distributions was introduced. However, further research showed that an important issue is the influence of the implemented prediction models on the type of distributions, where parameters describe the actual distributions of prediction errors sufficiently well [46].

A less-often-used approach consists of omitting the prediction stage and coding adaptively pixels instead of prediction errors [47].

Among the most effective codecs, the most often used is an adaptive version of a multivalue arithmetic coder with a reduced number of coding symbols [48] obtained using a non-linear quantization stage. A reduced number of symbols results in an increase in adaptation speed (there are fewer contexts). Even faster probability distribution adaptations can be achieved when using the binary version of the arithmetic encoder. Therefore, our proposed solution to the entropy coder is a completely new context adaptive binary arithmetic coder (CABAC). The absolute error value $|e(0)|$ is coded by an adaptive Golomb code, then compressed by two arithmetic coders. Additionally, if the error is non-zero, then the third coder for sign compression is activated. This two-stage approach significantly improves coder adaptability to the quickly changing properties of image prediction error.

5.1. Short-Term Estimation of Probability Distribution

The presented rules for determining the context number are an extension of ideas from previous works [14,15,42,43]. Firstly, values ω_1 and ω_2 are computed:

$$\begin{aligned}\omega_1 = \max\Big\{&2.3\cdot\big|e(1)\big|, 2\cdot\big|e(2)\big|, 1.6\cdot\big|e(4)\big|, 0.95\cdot\Big(\big|e(3)\big|+\big|e(4)\big|\Big),\\ &1.25\cdot\Big(\big|e(5)\big|+\big|e(10)\big|\Big), 1.3\cdot\big|e(3)\big|, 1.375\cdot\Big(\big|e(1)\big|+\big|e(2)\big|\Big),\\ &0.4\cdot\Big(\big|e(6)\big|+\big|e(7)\big|\Big), 0.4\cdot\Big(\big|e(8)\big|+\big|e(9)\big|\Big)\Big\},\end{aligned} \tag{39}$$

$$\omega_2 = \frac{1}{\delta}\sum_{j=1}^{m}\overline{d}_j\cdot|e(j)|, \tag{40}$$

where $\delta = \sum_{j=1}^{m}\overline{d}_j$ ad $m = 28$. For $\overline{d}_j$, see the description of Formula (16). Next, the computed parameters are:

$$\omega_3 = \max\Big\{2.1\cdot\omega_1, 10.2\cdot\omega_2\Big\}, \tag{41}$$

$$\begin{aligned}\omega_4 = \max\Big\{&\big|P(1)-P(3)\big|, \big|P(2)-P(4)\big|, 1.1\cdot\big|P(1)-P(2)\big|,\\ &0.7\cdot\big|P(2)-P(3)\big|, 0.9\cdot\big|P(1)-P(4)\big|, 0.9\cdot\big|P(3)-P(4)\big|\Big\},\end{aligned} \tag{42}$$

which are used for final calculation of ω:

$$\omega = \omega_3 + 0.48\cdot\omega_4. \tag{43}$$

ω is quantized using $t-1$ thresholds $\mathbf{T}_h(j)$, which for $t=16$, gives a 4-bit number b_{medium} of short-term probability distribution $\mathbf{T}_h = \{3, 7, 12, 18, 24, 31, 39, 49, 59, 72, 90, 115, 140, 170, 210\}$.

5.2. Medium-Term Estimation of Probability Distribution

Golomb code is particularly well suited for coding data with a geometric distribution [49]. Parameter m_G is chosen so that for p, the parameter of geometric probability distribution $p^{m_G}\approx 1/2$. The value of m_G is searched for each coded $|e(0)|$ among the 6 probability distributions of the form

$G(i) = (1-p)p^i$. They are defined by m_G values $\mathbf{m} = \{1,1,2,3,4,12\}$. The current p parameter is calculated as $p = (\mathcal{K}-1)/\mathcal{K}$, where $\mathcal{K} = \omega_2$ for $m = 48$ (40). Then, m_G is evaluated [49,50]:

$$m_G = \left\lceil -\frac{\log_{10}(1+p)}{\log_{10} p} \right\rceil. \tag{44}$$

According to a previous observation [51], $m_G \approx ln(2)\mathcal{K}$. Then, the value of $ln(2)\mathcal{K}$ is quantized using thresholds $\{0.01, 1.5, 3.6, 11.0, 16.0\}$; the obtained index b_{Golomb} with values of $\{0,1,...,5\}$ is used to select element of set $\mathbf{m}$. The value b_{Golomb} is a part of a context number, and is constant when coding bits of Golomb word representing current $|e(0)|$. Golomb word consists of unary coded group number $u_G = \lfloor |e(0)|/m_G \rfloor$, and for $m_G > 1$, the group element number $v_G = |e(0)| - u_G \cdot m_G$ (remainder of division by m_G) is coded using phased-in binary code, which is the variant of the Huffman code for sources with m_G equally probable symbols [49]. Specifying the $k = \lceil \log_2 m_G \rceil$ parameter means that in each group, the first $l = 2^k - m_G$ elements v_G are coded using $k-1$ bits, and the remaining $m - l$ elements are coded as number $v_G + l$ using k bits [21]. The value $|e(0)|$ is transformed into two bitstreams representing u_G and v_G in the Golomb code block (Figure 2).

5.3. Context Number Calculation

Binary sequences u_G and v_G are coded by separate binary arithmetic coders. The context number for u_G is computed as follows:

$$ctx_u = 6 \cdot (2^4 \cdot b_{\text{Golomb}} + b_{\text{medium}}) + b_{\text{unary}}, \tag{45}$$

where b_{unary} is in the range $\{0,1,...,5\}$ and denotes the number of currently coded u_G bits (starting with the most significant one). If there are more than six bits, then $b_{\text{unary}} = 5$. Hence, there are 576 ctx_u contexts. The number of contexts for v_G is 192:

$$ctx_v = 2^4 \cdot (2 \cdot b_{\text{Golomb}} + b_{\text{phased-in}}) + 2^3 \cdot b_\omega + 2^2 \cdot b_{\text{binary}} + b_{\text{unary2}}, \tag{46}$$

where $b_{\text{unary2}} = \min\{b_{\text{unary}}, 3\}$ and b_{binary} is the most significant bit of v_G, b_ω is a one-bit number obtained by quantizing ω using threshold 49, and $b_{\text{phased-in}}$ is 0 for the first coded bit of v_G and 1 otherwise. If $b_{\text{phased-in}} = 0$, then $b_{\text{binary}} = 0$.

5.4. Long-Term Adaptation of Probability Distribution

Each context number is associated with counters of its zeros and ones: $n(0)$ and $n(1)$. The counter values cannot grow infinitely; when the sum $n(0) + n(1)$ reaches a value $N_{\max}$, both counts are halved. For u_G, use $N_{\max} = 2^{10}$, and the counters' initial values are $n(0) = n(1) = 1$. For v_G, the values are $N_{\max} = 2^{11}$, and initially, $n(0) = n(1) = 16$.

5.5. Sign Coding

Separate error $e(0)$ sign coding is rather uncommon in binary arithmetic coders; hence, it is a particular feature of the coder presented in this paper. There are 32 contexts for sign $e(0)$ coding (5-bit context number). Bits of the context number are: signs of neighbor errors $\text{sgn}(e(1))$ and $\text{sgn}(e(2))$ (Figure 1). Then, bit b_ω; see the comment to (46). Finally, the last two bits are obtained from the four-state quantization of $|e(0)|$ using thresholds $\{1,3,16\}$. Initial counts of zeros and ones are set to $n(0) = n(1) = 2$. For the sign coder, $N_{\max} = 2^{10}$.

6. Performance and Complexity Analysis of New Algorithms

6.1. Generalized Criterion for Minimizing the Bit Average of a Coder

In [26], they observed that minimization of the mean square error is not equivalent to minimization of the first-order entropy and the average bitrate of a coded image. This observation prompted us to search for the more accurate global static predictors from this point of view for each of 45 test images using the Minkowsky vector distance criterion [52]:

$$L_M = \Big(\sum_{n \in Q} \Big| x^{(n)}(0) - \hat{x}^{(n)}(0) \Big|^M \Big)^{\frac{1}{M}}, \tag{47}$$

where $x^{(n)}(0)$ and $\hat{x}^{(n)}(0)$ are reference and actual vector elements, e.g., actual samples and their estimates (1), respectively; Q is the data training area for predictive model learning. It appeared that best predictors were obtained for M values between 0.6 and 0.9 (compromise $M = 0.75$). For comparison, in [26], the MMAE criterion meant that $M = 1$; for $M = 2$, the criterion was equivalent to MMSE. In [26], some improved results were obtained in comparison to MMSE; however, the improvements were not as evident for the final bitrate as for prediction error entropy. This was due to the extremely difficult task of joint optimization of modelling and entropy coding stages, at least for online methods. In the case of advanced offline techniques such as MRP [15], the optimization simply resulted in the iterative nature of such algorithms and in their high computational complexity (forward adaptation). Replacing MMSE by some version of the Minkovsky criterion led to an important increase in potentially enhanced method computational complexity, making it impractical.

The results of [52] were so striking that we decided to continue experiments with the Minkovsky criterion. It appeared that the optimum value M can be strongly variable. We started with OLS prediction, followed by an adaptive arithmetic coder, and the value jumped from approximately 0.75 [52] to 1.3, partly due to the locality of the OLS predictors. After adding some kind of cumulated predictor error removal (compare with Section 4), the value increased to 1.5. The 3ST-OLS technique [31] has an additional NLMS+ stage (compare with Section 3.3), and for it, the optimal M is 1.7. Finally, for the proposed EM-WLS version, the best M value is 1.9 (Figure 5), and the gain in output data entropy with respect to MMSE criterion is very small and unlikely to be exploited in practice (of course, in the paper, we present the algorithm based on MMSE).

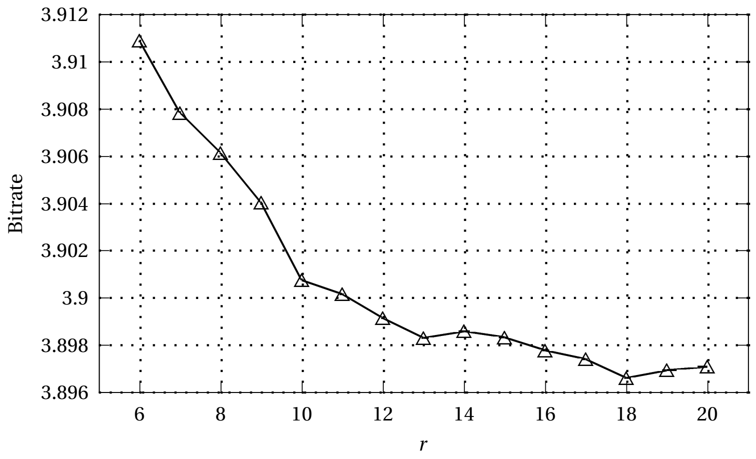

Figure 5. Dependence of the bit average on the LA-OLS prediction order (average for a set of 45 test images).

In summary, structures of the most effective data compacting algorithms seem to be collections of ad-hoc concepts that work, testing for how close to 2 the optimum M value is to the Minkovsky criterion for predictor optimization seems to validate these concepts, or not. For example, our findings highlight a non-obvious fact: multi-stage multimedia lossless coders are often more accurate than one-stage coders, in accordance with what is known about linear predictors. The added stages could even deteriorate the result; when data are scarce, combined total linear predictor length may be prolonged beyond limits given by length criteria, such as Akaike, MDL, etc. However, as in the examples above, the addition of a stage to a coder may push the optimum M value towards 2, and this seems to be the effect that matters.

In line with the findings presented here, the cascaded EM-WLS algorithm is close to optimal for the Minkovsky criterion (Figure 6), providing the advantages of an online approach (backward adaptation) and excellent performance (see Section 3.2).

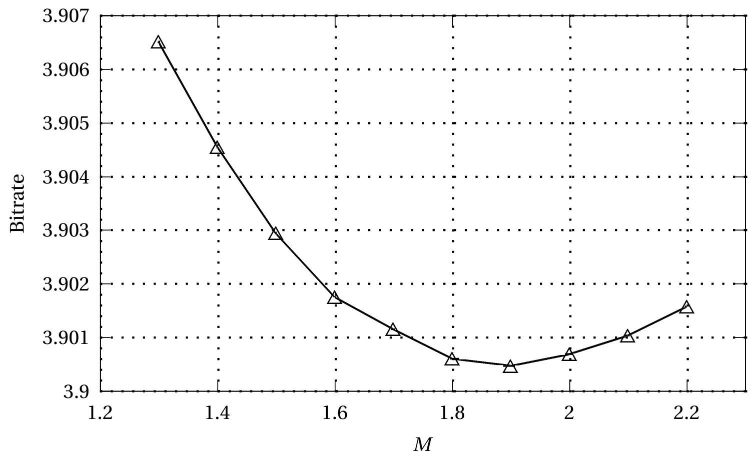

Figure 6. Relationship between the bit average and M, the parameter of the Minkovsky minimization criterion applied to the Main Predictor (20) in the EM-WLS method for a database of 45 test images.

6.2. Performance Analysis of Algorithm with LA-OLS in Main Predictor Stage

Testing of this method started with checking if each of the novel aspects of the method improved overall technique performance. Table 1 compares the average bitrates per pixel for the LA-OLS method (last column) with algorithm versions with an omitted feature. In Test 1, Formula (8) was implemented instead of (11); in Test 2, predictor coefficients were not weighted (12); in Test 3, there were no NLMS stages; and in Test 4, there was no error bias cancelling stage. As can be seen, the results for the full method were superior. The situation was similar when using EM-WLS as the Main Predictor.

Table 1. Bitrate comparison of simplified and full LA-OLS versions.

Test	1	2	3	4	LA-OLS
Bitrate for 45 images	3.91414	3.90987	3.96234	3.97458	3.90510

Figure 5 shows the bit average for a set of 45 test images as a function of the LA-OLS predictor order. Table 2 presents results for LA-OLS predictor orders from 6 to 20, size of optimal training window size W, and execution times of the versions, times are for coding of *Lennagray* image on i5 3.4 GHz PC. As can be seen, the best parameters are obtained for $r = 18$, and $W = 10$. For this case total execution time is 5.86 s, which is very close to that for GLICBAWLS method [32]. Additional time needed for the realisation of NLMS and bias correction stages is quite big when these stages are appended to described in [53] CoBALP$_{\text{ultra}}$ algorithm, its execution time extends from 2.07 s to 2.62 s.

Table 2. Performance of LA-OLS for a set of OLS orders (average for a set of 45 test images).

r	W	Bitrate for 45 Images	Execution Time (s)
6	6	3.91090	2.80
7	6	3.90785	2.94
8	6	3.90614	3.02
9	8	3.90404	3.30
10	8	3.90076	3.48
11	8	3.90017	3.65
12	8	3.89914	3.87
13	8	3.89832	4.08
14	10	3.89860	4.63
15	10	3.89834	4.90
16	10	3.89778	5.20
17	10	3.89741	5.52
18	10	3.89661	5.86
19	10	3.89695	6.21
20	10	3.89710	6.58

6.3. Effects of Neighborhood Selection in Linear Prediction

The most accurate result is obtained when the closest possible neighborhood of the coded pixel (in terms of Euclidean distance) is selected, consisting of r pixels $P(i)$ in (1), where r is the predictor order. This was confirmed by experiments. Notably, there are groups of pixels equidistant to the coded pixel $P(0)$. For $r \leq 30$, these are pixels with numbers as shown in Figure 7a, where equidistant pixels are numbered clockwise: {1, 2}, {3, 4}, {5, 6}, {7, 8, 9, 10}, {11, 12}, {13, 14}, {15, 16, 17, 18}, {19, 20, 21, 22}, {23, 24}, {25, 26, 27, 28}, and {29, 30}. In the case of orders r belonging to the set {2, 4, 6, 10, 12, 14, 18, 22, 24, 28, 30}, pixel numbering is irrelevant, as long as we maintain the rule of the Euclidean distance minimization. Examples can be found in previous works for $r = 6$ [54], for $r = 10$ [30], for $r = 12$ [33,55–57], and for $r = 18$ [29].

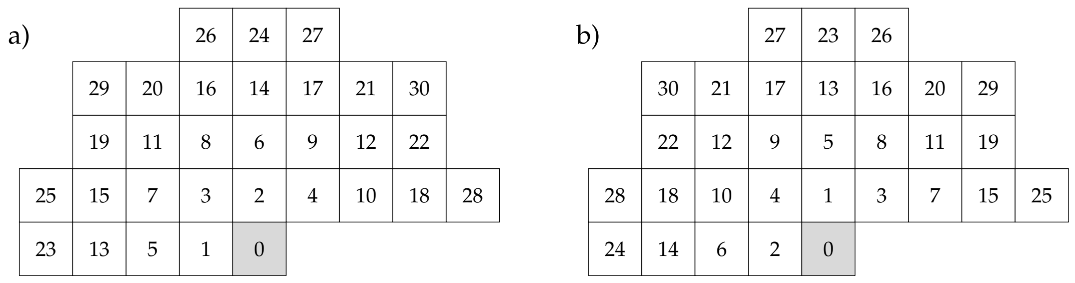

Figure 7. Two methods of numbering the pixels in the neighborhood of the currently coded pixel $P(0)$.

A problem arises for other prediction orders, e.g., when $r = 5$, we have to decide which pixel, $P(5)$ or $P(6)$, completes the set of four nearest pixels: $P(1)$, $P(2)$, $P(3)$, and $P(4)$. For example, in [58], the fifth neighbor was pixel $P(5)$ (Figure 7a). The finding seems to be justified by the fact that we minimized the number of image rows, to which the procedure of calculating the predicted value must have access. This is important for optimizing execution time or resource usage for hardware solutions.

In many cases, the speed of calculations has high priority. In such cases, low orders of prediction are implemented as a compromise, and even some simplifications are introduced to reduce the complexity of Equation (8) calculation to close to that for the model of order $r = 5$ while maintaining compression efficiency close to that offered by order $r = 6$ [54].

Some solutions in the literature prefer the Manhattan (l_1 norm) over the Euclidean distance. For example, in [32], the neighborhood for distance $l_1 \leq 3$ ($r = 12$) was used, and in [44,46] distances were $l_1 \leq 4$ ($r = 20$) and $l_1 \leq 5$ ($r = 30$), respectively.

In our opinion, the best approach to the numbering of neighborhood pixels is to minimize the Euclidean distance. In the case of equidistant pixels, counterclockwise numbering should be used (Figure 7b). Our experiments confirmed the advantage of this approach over clockwise numbering. This advantage is most noticeable when using linear predictors of orders 3, 5, and 7. This is particularly important when using multiple prediction models of different orders together, as in Section 3.2.3.

An intuitive explanation is that, assuming the same level of correlation of equidistant neighbors from pixel $P(0)$, the pixels on the right side of $P(0)$ should be selected first. This is because the number of neighbors to the left of $P(0)$ is predominant, for example, for $r = 14$, there are seven of them; another three are located directly above the coded pixel, and on its right side, there are only four of them. The dominance of left-handed neighbors introduces some imbalance of information. Therefore, counterclockwise numbering, at least for some prediction orders, reduces this disproportion.

The authors of [26] used one step further in this direction by relaxing the Euclidean distance criterion for a model of order $r = 14$. They decided to use only three rows, keeping the six neighbors to the left (two columns in each of three rows) equal to that of those to the right of the coded pixel $P(0)$ (together with two pixels directly above it: four columns in each of two rows).

6.4. Performance Analysis of New Algorithms

Section 6.2 confirms the usefulness of all blocks in the proposed cascade model (see Figure 2) for total coder compression efficiency maximization. Additionally, the dependence is shown of coding time on parameters r and W when using LA-OLS as the Main Predictor. The highest efficiency was obtained for $r = 18$ and $W = 10$. For these settings, the encoding time of the Lennagrey image (512×512 pixels) is 5.86 s.

When EM-WLS was used in the first predictive block, the coding time increased significantly up to 107.4 s (decoding time was similar due to the same process for calculating predictions of coded pixels in the decoder). The time is not dependent on image content; it is only linearly proportional to the number of pixels in the encoded image.

Table 3 presents the time contributions of the EM-WLS cascade model coding stages to the whole coding time. It shows that the total time contribution of the last three stages is less than 1% (two last entries of Table 3). The same holds for total processing time of both NLMS blocks. Among the most complex operations is the determination of predictive model, requiring the solution of 22 matrix equations, which is performed twice due to the use of ridge-regression (Formulas (11) and (19)). This is performed using Cholesky decomposition, which requires 13.21% of the coding time. However, the most complex calculation is that of filling the **R** matrix and **P** vector (see Formulas (9) and (10)). The high computational complexity of this step is due to each coded pixel, for as much as $W \cdot (W + 1) \cdot r \cdot (r + 5)$, multiplications and additions should be performed despite symmetry of the **R** matrix. For $r = 24$ and $W = 14$, this necessitates 146,160 multiplication and addition operations.

Table 3. Time contributions of algorithm steps to the total coding time.

Filling the R Matrix and P Vector	**Solving Matrix Equations**	**Two NLMS Blocks**	**CDCCR**	**CABAC and Golomb Code**
85.20%	13.21%	0.62%	0.85%	0.12%

Tables 4–7 compare LA-OLS and EM-WLS performance to those of provided in the literature for other efficient lossless image coding techniques. Two sets of test images were used for this purpose [59,60]. The execution times of CALIC and JPEG-LS, JPEG2000, and BMF) are below one

second. The methods in Table 4 are faster than EM-WLS, but perform worse: Blend-20 is three times faster, LA-OLS codes the Lennagray image in 5.86 s (Pentium i5 3.4 GHz), in contrast to 52.5 s for Vanilic WLS-D. The coding time using CoBALP$_{\text{ultra2}}$, GLICBAWLS, 3ST-OLS, SWAP, and RALP is a few seconds.

Among methods with the most complex implementations, EM-WLS stands out with an execution time 107.4 s. This is still 3.9 times less than for MRP 0.5. GPR-BP, MRP-SSP, and TMW^{Lego} are even more computationally complex. As can be seen, being less complex, EM-WLS is, on average, closer to optimum than these algorithms. However, decoders for MRP 0.5 and, hence, GPR-BP and MRP-SSP, are relatively time efficient due to the use of prediction with forward adaptation, whereas the EM-WLS decoder complexity is similar to that of a coder. EM-WLS files are on average 11.98% shorter than those of JPEG-LS.

Table 4. Bitrate comparison of some state-of-the-art algorithms for the first image database [59].

Images	**JPEG-LS** [12]	**CALIC** [11]	**OLS** [14]	**GLICBAWLS** [32]	**CoBALP$_{\text{ultra2}}$** [31,53]	**Vanilc WLS-D** [61]	**3ST-OLS** [31]
Balloon	2.889	2.78	2.690	2.640	2.673	2.626	2.580
Barb	4.690	4.31	3.939	3.916	3.881	3.815	3.832
Barb2	4.684	4.46	4.310	4.318	4.247	4.231	4.219
Board	3.674	3.51	3.388	3.392	3.339	3.332	3.296
Boats	3.930	3.78	3.638	3.628	3.591	3.589	3.544
Girl	3.922	3.72	3.576	3.565	3.523	3.523	3.471
Gold	4.475	4.35	4.273	4.276	4.232	4.229	4.208
Hotel	4.378	4.18	4.162	4.177	4.067	4.074	4.047
Zelda	3.884	3.69	3.549	3.537	3.568	3.501	3.504
Bit average	**4.058**	**3.864**	**3.725**	**3.717**	**3.665**	**3.658**	**3.633**

Table 5. Bitrate comparison of some state-of-the-art and proposed algorithms for the first image database [59].

Images	**TMW^{Lego}** [13]	**LA-OLS**	**MRP 0.5** [15]	**Multi-WLS** [14]	**Blend-20** [16]	**AVE-WLS** [62]	**Extended Multi-WLS**
Balloon	2.60	2.576	2.579	2.60	2.566	2.549	<u>2.546</u>
Barb	3.84	3.832	3.815	3.75	3.768	3.712	<u>3.705</u>
Barb2	4.24	4.214	4.216	4.18	4.175	4.134	<u>4.126</u>
Board	3.27	3.288	3.268	3.27	3.272	3.242	<u>3.240</u>
Boats	3.53	3.537	3.536	3.53	3.520	3.495	<u>3.494</u>
Girl	3.47	3.467	3.465	3.45	3.449	3.411	<u>3.409</u>
Gold	4.22	4.198	4.207	4.20	4.185	4.170	<u>4.169</u>
Hotel	4.01	4.040	4.026	4.01	4.007	3.979	<u>3.977</u>
Zelda	3.50	3.499	3.495	3.51	3.498	3.485	<u>3.483</u>
Bit average	**3.631**	**3.628**	**3.623**	**3.611**	**3.605**	**3.575**	**<u>3.572</u>**

Table 6. Bitrate comparison of some state-of-the-art algorithms for the second image database [60].

Images	**JPEG2000** [47]	**FLIF 0.3** [47]	**WebP Lossless 0.6** [47]	**SWAP** [63]	**RALP** [57]	**TMW** [45]	**GLICBAWLS** [32]	**PMO** [47]
Airplane	4.013	3.794	3.894	3.58	3.71	3.601	3.668	3.632
Baboon	6.107	6.078	5.891	5.86	5.81	5.738	5.666	5.727
Balloon	3.031	2.856	2.925	2.49	2.55	2.649	2.640	2.673
Barb	4.600	4.500	4.547	4.12	4.12	4.084	3.916	3.997
Barb2	4.789	4.656	4.668	4.55	4.51	4.378	4.318	4.287
Camera	4.535	4.285	4.274	4.39	4.24	4.098	4.208	3.960
Couple256	3.915	3.677	3.703	3.75	3.63	3.446	3.543	3.415
Gold	4.603	4.518	4.464	4.30	4.32	4.266	4.276	4.476
Lennagrey	4.303	4.252	4.145	3.95	3.95	3.908	3.901	3.944
Peppers	4.629	4.595	4.495	4.25	4.27	4.251	4.246	4.267
Bit average	**4.453**	**4.321**	**4.301**	**4.124**	**4.111**	**4.042**	**4.038**	**4.038**

Table 7. Bitrate comparison of some state-of-the-art and new algorithms for the second image database [60].

Images	**BMF** [15]	**Vanilc WLS-D** [61]	**xMRP** [64]	**MRP 0.5** [15]	**LA-OLS**	**GPR-BP** [17]	**MRP-SSP** [18]	**Extended Multi-WLS**
Airplane	3.602	3.575	3.590	3.591	3.568	3.451	3.536	3.547
Baboon	5.714	5.678	5.662	5.663	5.643	5.641	5.635	5.622
Balloon	2.649	2.626	2.613	2.579	2.576	2.544	2.548	2.546
Barb	3.959	3.815	3.817	3.815	3.832	3.821	3.764	3.705
Barb2	4.276	4.231	4.226	4.216	4.214	4.184	4.175	4.126
Camera	4.060	3.995	3.971	3.949	4.001	3.964	3.901	3.920
Couple256	3.448	3.459	3.389	3.388	3.414	3.339	3.323	3.345
Gold	4.238	4.229	4.216	4.207	4.198	4.178	4.173	4.169
Lennagrey	3.929	3.856	3.885	3.889	3.881	3.880	3.877	3.847
Peppers	4.241	4.187	4.208	4.199	4.153	4.170	4.163	4.101
Bit average	**4.012**	**3.965**	**3.958**	**3.950**	**3.948**	**3.917**	**3.910**	**3.893**

7. Conclusions

In this work, different approaches to lossless compression of images, such as forward and backward adaptation, were analyzed. The paper contains some remarks on neighborhood selection in pixel prediction (Section 6.3) and notes on relationships between Minkovsky distance, final prediction error first-order entropy, and eventual coder average data rate (Section 6.4). The proposed EM-WLS algorithm construction was influenced by these observations. When compared to the best algorithms, on average, the proposed EM-WLS lossless image coding technique is currently the most efficient in terms of data compaction. The algorithm is less computationally complex than its main competitors. It is based on the AVE-WLS approach, being an expanded version of WLS, and has a cascade form, where the EM-WLS predictor is followed by a two-stage NLMS section, and by a final Context-Dependent Constant Component Removing stage. The new sophisticated binary context arithmetic coder is much less computationally complex than the preceding data modelling stage; hence, it can be used in other image compression methods. In the proposed universal cascade architecture (Figure 2), the Main Predictor module can be converted to an LA-OLS coder with lower implementation complexity (while maintaining high compression efficiency) due to the simpler prediction value calculation technique, as shown in Section 3.1.

Author Contributions: Conceptualization, G.U. and R.S.; methodology, G.U.; software, G.U.; validation, G.U., R.S., and C.W.; formal analysis, G.U.; investigation, G.U. and R.S.; resources, G.U.; data curation, G.U.; writing—original draft preparation, G.U., R.S., and C.W.; writing—review and editing, G.U., R.S., and C.W.; visualization, G.U. and C.W.; supervision, G.U.; project administration, G.U.; funding acquisition, G.U. and R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kassim, A.A.; Yan, P.; Lee, W.S.; Sengupta, K. Motion compensated lossy-to-lossless compression of 4-D medical images using integer wavelet transforms. *IEEE Trans. Inf. Technol. Biomed.* **2005**, *9*, 132–138. [CrossRef] [PubMed]
2. Sanchez, V.; Nasiopoulos, P.; Abugharbieh, R. Efficient 4D motion compensated lossless compression of dynamic volumetric medical image data. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, USA, 31 March–4 April 2008; pp. 549–552. [CrossRef]
3. Scharcanski, J. Lossless and Near-Lossless Compression for Mammographic Digital Images. In Proceedings of the International Conference on Image Processing, Atlanta, GA, USA, 8–11 October 2006; pp. 2253–2256. [CrossRef]
4. Strom, J.; Cosman, P. Medical Image Compression with Lossless Regions of Interest. *Signal Process.* **1997**, *59*, 155–171. [CrossRef]
5. Wang, Z.; Li, G.; Xie, X. A Near-Lossless Image Compression Algorithm Suitable for Hardware Design in Wireless Endoscopy System. *EURASIP J. Adv. Signal Process.* **2007**, 48–61. [CrossRef]
6. Chen, X.; Canagarajah, C.; Vitulli, R.; Nunez-Yanez, J. Lossless Compression for Space Imagery in a Dynamically Reconfigurable Architecture. In Proceedings of the International Workshop on Applied Reconfigurable Com-puting (ARC2008), London, UK, 26–28 March 2008; Volume 4943, pp. 332–337. [CrossRef]
7. CCSDS. (Ed.) *Lossless Data Compression. Recommendation for Space Data System Standards*; CCSDS: Washington, DC, USA, 2007.
8. Andriani, S.; Calvagno, G.; Erseghe, T.; Mian, G.A.; Durigon, M.; Rinaldo, R.; Knee, M.; Walland, P.; Koppetz, M. Comparison of lossy to lossless compression techniques for digital cinema. In Proceedings of the International Conference on Image Processing (ICIP '04), Singapore, 24–27 October 2004; Volume 1, pp. 513–516. [CrossRef]
9. Marcellin, M.; Gormish, M.; Bilgin, A.; Boliek, M. An Overview of JPEG2000. In Proceedings of the Data Compression Conference, Snowbird, UT, USA, 28–30 March 2000; pp. 523–541.
10. Richardson, I. *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia/Iain E. G. Richardson*; John Wiley & Sons Ltd.: Hoboken, NJ, USA, 2004.
11. Wu, X.; Memon, N. Context-based, adaptive, lossless image coding. *IEEE Trans. Commun.* **1997**, *45*, 437–444. [CrossRef]
12. Weinberger, M.J.; Seroussi, G.; Sapiro, G. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Trans. Image Process.* **2000**, *9*, 1309–1324. [CrossRef] [PubMed]
13. Meyer, B.; Tischer, P. TMW^{Lego}—An Object Oriented Image Modelling Framework. In Proceedings of the Data Compression Conference, Snowbird, UT, USA, 27–29 March 2001; IEEE Computer Society: Washington, DC, USA, 2001; p. 504.
14. Ye, H.; Deng, G.; Devlin, J.C. A Weighted Least Squares Method For Adaptive Prediction in Lossless Image Compression. In Proceedings of the Picture Coding Symposium, Saint-Malo, France, 23–25 April 2003; pp. 489–493.
15. Matsuda, I.; Ozaki, N.; Umezu, Y.; Itoh, S. Lossless coding using variable block-size adaptive prediction optimized for each image. In Proceedings of the 13th European Signal Processing Conference, Antalya, Turkey, 4–8 September 2005; pp. 1–4.
16. Ulacha, G.; Stasinski, R. Performance optimized predictor blending technique for lossless image coding. In Proceedings of the 36th International Conference on Acoustics, Speech and Signal Processing (ICASSP'11), Prague, Czech Republic, 22–27 May 2011; pp. 1541–1544. [CrossRef]

17. Dai, W.; Xiong, H. Gaussian Process Regression Based Prediction for Lossless Image Coding. In Proceedings of the Data Compression Conference, Snowbird, UT, USA, 26–28 March 2014; pp. 93–102. [CrossRef]
18. Dai, W.; Xiong, H.; Wang, J.; Zheng, Y.F. Large Discriminative Structured Set Prediction Modeling with Max-Margin Markov Network for Lossless Image Coding. *IEEE Trans. Image Process.* **2014**, *23*, 541–554. [CrossRef] [PubMed]
19. Ulacha, G.; Stasinski, R. Enhanced Lossless Image Coding Methods Based on Adaptive Predictors. In Proceedings of the International Conference on Systems, Signals and Image Processing IWSSIP, Rio de Janeiro, Brazil, 17–19 June 2010; pp. 312–315.
20. Przelaskowski, A. Hybrid Lossless Coder of Medical Images with Statistical Data Modelling. In Proceedings of the International Conference on Computer Analysis of Images and Patterns CAIP: Computer Analysis of Images and Patterns, Warsaw, Poland, 5–7 September 2001; Volume 2124, pp. 92–101. [CrossRef]
21. Sayood, K. *Introduction to Data Compression*, 5th ed.; Morgan Kaufmann Publ./Elsevier Inc.: Cambridge, MA, USA, 2018. [CrossRef]
22. Ye, H.; Deng, G.; Devlin, J.C. Parametric probability models for lossless coding of natural images. In Proceedings of the 11th European Signal Processing Conference EUSIPCO-02, Toulouse, France, 3–6 September 2002; pp. 514–517.
23. Memon, N.; Sayood, K. An asymmetric lossless image compression technique. In Proceedings of the International Conference on Image Processing, Washington, DC, USA, 23–26 October 1995; Volume 3, pp. 97–100. [CrossRef]
24. Golchin, F.; Paliwal, K.K. Classified adaptive prediction and entropy coding for lossless coding of images. In Proceedings of the International Conference on Image Processing, Santa Barbara, CA, USA, 26–29 October 1997; Volume 3, pp. 110–113. [CrossRef]
25. Aiazzi, B.; Alparone, L.; Baronti, S. Near-lossless image compression by relaxation-labelled prediction. *Signal Process.* **2002**, *82*, 1619–1631. [CrossRef]
26. Hashidume, Y.; Morikawa, Y. Lossless image coding based on minimum mean absolute error predictors. In Proceedings of the SICE Annual Conference, Takamatsu, Japan, 17–20 September 2007; pp. 2832–2836. [CrossRef]
27. Wang, X.; Wu, X. On Design of Linear Minimum-Entropy Predictor. In Proceedings of the IEEE 9th Workshop on Multimedia Signal Processing, MMSP'07, Crete, Greece, 1–3 October 2007; pp. 199–202. [CrossRef]
28. Huang, H.; Franti, P.; Huang, D.; Rahardja, S. Cascaded RLS–LMS Prediction in MPEG-4 Lossless Audio Coding. *IEEE Trans. Audio Speech Lang. Process.* **2008**, *16*, 554–562. [CrossRef]
29. Ye, H.; Deng, G.; Devlin, J.C. Adaptive linear prediction for lossless coding of greyscale images. In Proceedings of the IEEE International Conference on Image Processing, Vancouver, BC, Canada, 10–13 September 2000; Volume 1, pp. 128–131. [CrossRef]
30. Wu, X.; Barthel, E.U.; Zhang, W. Piecewise 2D autoregression for predictive image coding. In Proceedings of the International Conference on Image Processing. ICIP98 (Cat. No.98CB36269), Chicago, IL, USA, 7 October 1998; Volume 3, pp. 901–904. [CrossRef]
31. Ulacha, G.; Stasinski, R. Three-Stage OLS Method for Improved Lossless Image Coding. In Proceedings of the International Conference on Systems, Signals and Image Processing (IWSSIP), Maribor, Slovenia, 20–22 June 2018; pp. 1–4. [CrossRef]
32. Meyer, B.; Tischer, P. Glicbawls-Grey Level Image Compression by Adaptive Weighted Least Squares. In Proceedings of the Data Compression Conference, Snowbird, UT, USA, 27–29 March 2001; p. 503.
33. Deng, G.; Ye, H.; Marusic, S.; Tay, D. A method for predictive order adaptation based on model averaging. In Proceedings of the International Conference on Image Processing, Barcelona, Spain, 14–17 September 2003; Volume 2, pp. 189–192. [CrossRef]
34. Wu, X.; Zhai, G.; Yang, X.; Zhang, W. Adaptive Sequential Prediction of Multidimensional Signals with Applications to Lossless Image Coding. *IEEE Trans. Image Process.* **2011**, *20*, 36–42. [CrossRef]
35. Hoerl, A.E.; Kennard, R.W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **1970**, *12*, 55–67. [CrossRef]
36. Sayood, K. *Lossless Compression Handbook*; Academic Press: San Diego, CA, USA, 2003.
37. Schuller, G.; Yu, B.; Huang, D. Lossless coding of audio signals using cascaded prediction. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Proceedings (Cat. No.01CH37221), Salt Lake City, UT, USA, 7–11 May 2001; Volume 5, pp. 3273–3276. [CrossRef]

38. Ulacha, G.; Stasinski, R. Effective context lossless image coding approach based on adaptive prediction. *World Acad. Sci.* **2009**, *57*, 63–68.
39. Topal, C.; Gerek, O. Pdf sharpening for multichannel predictive coders. In Proceedings of the 14th European Signal Processing Conference (EUSIPCO-06), Florence, Italy, 4–8 September 2006; pp. 1–4.
40. Golchin, F.; Paliwal, K.K. A lossless image coder with context classification, adaptive prediction and adaptive entropy coding. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, WA, USA, 15 May 1998; pp. 2545–2548.
41. Kau, L.-J.; Lin, Y.-P. Lossless image coding using a switching predictor with run-length encodings. In Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763), Taipei, Taiwan, 27–30 June 2004; pp. 1155–1158. [CrossRef]
42. Deng, G.; Ye, H. Lossless image compression using adaptive predictor symbol mapping and context filtering. In Proceedings of the International Conference on Image Processing (Cat. 99CH36348), Kobe, Japan, 24–28 October 1999; Volume 4, pp. 63–67. [CrossRef]
43. Aiazzi, B.; Alparone, L.; Baronti, S. Context modeling for near-lossless image coding. *IEEE Signal Process. Lett.* **2002**, *9*, 77–80. [CrossRef]
44. Matsuda, I.; Shirai, N.; Itoh, S. Lossless Coding Using Predictors and Arithmetic Code Optimized for Each Image. In Proceedings of the International Workshop on Visual Content Processing and Representation, Madrid, Spain, 18–19 September 2003; Volume 2849, pp. 199–207. [CrossRef]
45. Meyer, B.; Tischer, P. TMW—A new method for lossless image compression. In Proceedings of the International Picture Coding Symposium (PCS'97), Berlin, Germany, 10–12 September 1997; pp. 533–538.
46. Ueno, H.; Morikawa, Y. A new distribution modeling for lossless image coding using MMAE predictors. In Proceedings of the 6th International Conference on Information Technology and Applications, Hanoi, Vietnam, 9–12 November 2009; pp. 249–254.
47. Matsuda, I.; Ishikawa, T.; Kameda, Y.; Itoh, S. A Lossless Image Coding Method Based on Probability Model Optimization. In Proceedings of the 26th European Signal Processing Conference (EUSIPCO), Rome, Italy, 3–7 September 2018; pp. 156–160. [CrossRef]
48. Deng, G. Transform domain LMS-based adaptive prediction for lossless image coding. *Signal Process. Image Commun.* **2002**, *17*, 219–229. [CrossRef]
49. Salomon, D. *Data Compression-The Complete Reference*, 4th ed.; Springer: Berlin/Heidelberg, Germany, 2007.
50. Bhaskaran, V.; Konstantinides, K. *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed.; Kluwer Academic Publishers: Palo Alto, CA, USA, 1997. [CrossRef]
51. Sugiura, R.; Kamamoto, Y.; Harada, N.; Moriya, T. Optimal Golomb-Rice Code Extension for Lossless Coding of Low-Entropy Exponentially-Distributed Sources. *IEEE Trans. Inf. Theory* **2018**, *64*, 3153–3161. [CrossRef]
52. Ulacha, G.; Stasinski, R. Paths to future image lossless coding. In Proceedings of the ELMAR-2012, Zadar, Croatia, 12–14 September 2012; pp. 63–66.
53. Ulacha, G.; Stasinski, R. New context-based adaptive linear prediction algorithm for lossless image coding. In Proceedings of the 2014 International Conference on Signals and Electronic Systems (ICSES), Poznan, Poland, 11–13 September 2014; pp. 1–4. [CrossRef]
54. Jakhetiya, V.; Jaiswal, S.; Tiwari, A. A novel predictor coefficient interpolation approach for lossless compression of images. In Proceedings of the 2011 IEEE International Instrumentation and Measurement Technology Conference, Binjiang, China, 10–12 May 2011; pp. 1–4. [CrossRef]
55. Li, X.; Orchard, M.T. Edge-directed prediction for lossless compression of natural images. *IEEE Trans. Image Process.* **2001**, *10*, 813–817.
56. Takamura, S.; Matsumura, M.; Yashima, Y. A study on an evolutionary pixel predictor and its properties. In Proceedings of the 16th IEEE International Conference on Image Processing (ICIP'09), Cairo, Egypt, 7–10 November 2009; pp. 1921–1924.
57. Kau, L.-J.; Lin, Y.-P. Least squares-adapted edge-look-ahead prediction with run-length encodings for lossless compression of images. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, USA, 31 March–4 April 2008; pp. 1185–1188. [CrossRef]
58. Ye, H.; Deng, G.; Devlin, J.C. Least squares approach for lossless image coding. In Proceedings of the Fifth International Symposium on Signal Processing and Its Applications (IEEE Cat. No.99EX359) (ISSPA '99), Brisbane, QLD, Australia, 22–25 August 1999; Volume 1, pp. 63–66.

59. Test Image Database 1. 2020. Available online: http://wernik.zut.edu.pl/shared/Image1.zip (accessed on 20 August 2020).
60. Test Image Database 2. 2020. Available online: http://wernik.zut.edu.pl/shared/Image2.zip (accessed on 20 August 2020).
61. Weinlich, A.; Amon, P.; Hutter, A.; Kaup, A. Probability Distribution Estimation for Autoregressive Pixel-Predictive Image Coding. *IEEE Trans. Image Process.* **2016**, *25*, 1382–1395. [CrossRef] [PubMed]
62. Ulacha, G.; Stasinski, R. AVE-WLS Method for Lossless Image Coding. In Proceedings of the 10th International Conference on Image and Graphics (ICIG 2019), Beijing, China, 23–25 August 2019; LNCS 11903; Springer International Publishing: Beijing, China, 2019; pp. 23–34. [CrossRef]
63. Kau, L.; Lin, Y.; Lin, C. Lossless image coding using adaptive, switching algorithm with automatic fuzzy context modelling. *IEE Proc. Vis. Image Signal Process.* **2006**, *153*, 684–694. [CrossRef]
64. Hsieh, F.Y.; Wang, C.M.; Lee, C.C.; Fan, K.C. A Lossless Image Coder Integrating Predictors and Block-Adaptive Prediction. *J. Inf. Sci. Eng.* **2008**, *24*, 1579–1591.

Sample Availability: Database of the images are available on the webpage http://wernik.zut.edu.pl/shared/, the files Image1.zip and Image2.zip.