applied sciences

MDPI

*Article*

# Novel Block Sorting and Symbol Prediction Algorithm for PDE-Based Lossless Image Compression: A Comparative Study with JPEG and JPEG 2000

**Časlav Livada [1,*], Tomislav Horvat [2] and Alfonzo Baumgartner [1]**

[1] Chair of Visual Computing, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, Josip Juraj Strossmayer University of Osijek, Kneza Trpimira 2B, 31000 Osijek, Croatia
[2] Department of Electrical Engineering, University North, 104. Brigade 3, 42000 Varaždin, Croatia
* Correspondence: caslav.livada@ferit.hr

**Abstract:** In this paper, we present a novel compression method based on partial differential equations complemented by block sorting and symbol prediction. Block sorting is performed using the Burrows–Wheeler transform, while symbol prediction is performed using the context mixing method. With these transformations, the range coder is used as a lossless compression method. The objective and subjective quality evaluation of the reconstructed image illustrates the efficiency of this new compression method and is compared with the current standards, JPEG and JPEG 2000.

**Keywords:** block sorting; compression; objective and subjective quality assessment; partial differential equations; symbol prediction



**Citation:** Livada, Č.; Horvat, T.; Baumgartner, A. Novel Block Sorting and Symbol Prediction Algorithm for PDE-Based Lossless Image Compression: A Comparative Study with JPEG and JPEG 2000. *Appl. Sci.* **2023**, *13*, 3152. https://doi.org/10.3390/app13053152

Academic Editor: Zhengjun Liu

Received: 16 January 2023
Revised: 22 February 2023
Accepted: 24 February 2023
Published: 28 February 2023



## 1. Introduction

Partial differential Equation (PDE)-based image compression techniques use PDEs to model the image data and convert it into a set of parameters that can be efficiently encoded. There are several PDE-based image compression techniques that differ in the way they model and transform the image data. PDE-based image enhancement methods were invented to restore missing image regions by uniformly transferring information from the known surrounding regions. The filling effect has also become the main feature of PDE-based inpainting methods such as [1–4]. The main idea is to consider the image data as Dirichlet boundary conditions and interpolate the data in the inpainting regions by solving the appropriate boundary value problems.

**Why use PDE-based image compression?** PDE-based image compression methods offer several advantages over traditional compression techniques such as JPEG and MPEG. Firstly, they offer higher compression ratios while maintaining better image quality. This is achieved by modeling the image as a set of partial differential equations and transforming it into a set of parameters that can be efficiently encoded. The resulting compressed image contains much less data, causing it to be easier to store and transmit [5]. Secondly, PDE-based compression methods are more robust to noise and distortion. Unlike traditional compression techniques, PDE-based methods rely on modeling the image using differential equations, which can adapt to changes in the image data, causing them to be more tolerant to noise and distortion. This causes them to be particularly useful in scenarios where the image quality needs to be maintained despite noise and other distortions [6,7]. Thirdly, PDE-based compression methods are versatile and can be adapted to various applications. For instance, they can be used for compressing medical images, satellite images, and video streams, among others. This causes them to be a valuable tool in various domains, including medical imaging, remote sensing, and multimedia [8,9]. Overall, PDE-based image compression methods offer a compelling alternative to traditional compression techniques.

They offer higher compression ratios, better image quality, and are more robust to noise and distortion. Moreover, they are versatile and can be adapted to different applications.

**Existing PDE-based compression algorithms.** The Embedded Zero-Tree Wavelet (EZW) algorithm is a compression technique that uses the wavelet transform to decompose an image into sub-bands, and then applies a PDE-based thresholding to produce a sparse representation of the image. The resulting sparse representation is then encoded using entropy coding. EZW is widely used in image and video compression [10]. The SPIHT (Set Partitioning In Hierarchical Trees) algorithm is a wavelet-based compression algorithm that uses a PDE-based algorithm to sort and partition the wavelet coefficients in a hierarchical tree structure. This tree structure is then used to encode the wavelet coefficients using binary arithmetic coding. SPIHT has been used in various applications, such as satellite imagery and medical imaging [11].

The Geometric Image Compression (GIC) algorithm is a PDE-based compression algorithm that uses a geometric representation of images based on a set of partial differential equations. GIC models the image data as a series of curves, areas, and volumes, and then applies PDE-based compression techniques to the curves, areas, and volumes to create a compact representation of the image. GIC has proven to be very powerful in compressing geometric data such as 3D meshes and point clouds [12]. The Variational Image Compression (VIC) algorithm is a PDE-based compression algorithm that uses a variational approach to model the image data. VIC solves an optimization problem that minimizes an energy function containing both fidelity to the original image and a regularization term that promotes the smoothness of the compressed image. The resulting compressed image is then encoded using entropy coding. VIC has shown good performance in compressing natural and medical images [13].

Variational and PDE-based interpolation and inpainting techniques have been used to interpolate the scattered data. Lower order PDEs often result in singularities at isolated interpolation points on images, higher-order PDEs provide smoother solutions, but the violation of an extremum principle can result in undesirable overshoots and undershoots [14]. Recently, PDE-based image compression techniques have been developed using PDE-based interpolation of scattering data [15–20]. The idea is to keep only a small number of pixels and reconstruct the remaining data using PDE-based interpolation. There are many subdivision strategies, in particular methods based on quadtree decompositions [21,22]) and adaptive triangulation ideas can be found in [23–25]. In R-EED [18], the adaptive triangulation from [17] is replaced by a subdivision into a rectangular structure with various modifications.

**Reliability and efficiency.** PDE-based image compression techniques have been proposed as a means of reducing the size of image data while preserving image quality. However, these methods have both strengths and limitations that affect their reliability and efficiency. One of the limitations of PDE-based compression techniques is their high computational complexity, which can cause them to be computationally intensive and time-consuming, especially for large images or real-time applications [26]. Another limitation of PDE-based compression techniques is their sensitivity to image content. PDE-based methods rely on modeling the image data using PDEs, which may not always accurately capture the structure and content of the image. In some cases, PDE-based methods may not perform well for certain types of images, such as those with sharp edges or high-frequency content [27].

Most PDE-based compression techniques are lossy, which means that they discard some information from the original image in order to achieve compression; while lossy compression can be effective in reducing the size of the image data, it can also result in perceptual distortions or artifacts in the compressed image, which can affect its quality and reliability [28]. In addition, PDE-based compression techniques may not always achieve the same level of compression performance as other state-of-the-art image compression methods, such as those based on deep learning or neural networks. In some cases, PDE-

based methods may not be able to achieve the same level of compression while preserving the same level of image quality [7,29].

In conclusion, while PDE-based image compression techniques have shown promise in reducing the size of image data, their reliability and efficiency depend on various factors, such as the specific method used, the nature of the image data being compressed, and the compression requirements of the application. Further research is needed to improve the reliability and efficiency of PDE-based compression methods, and to address their limitations and challenges.

**Our contribution.** The aim of this paper is to investigate the compression capabilities of unaided PDE-based compression methods and PDE-based compression methods using data transformations and symbol prediction. The compression methods and ratios for grayscale compression and binary tree structure compression are evaluated in terms of the extent to which the quality of a reconstructed image depends on the choice of compression methods. The improvement of the compression algorithms is performed by applying data transforms to the input data stream. The data transformations used are delta coding, context tree weighting, Burrows–Wheeler transform, prediction by partial fitting, and dynamic Markov coding.

**Related work.** PDE-based compression was evaluated with range coding and Burrows–Wheeler transform for grey values only [30] and concluded that it performed better than EED with this combination. In [31], further analysis of the range coder was performed in combination with context tree weighting, prediction by partial adaptation, and dynamic Markov coding.

**Organization of the paper.** We begin with a brief introduction to PDE-based inpainting in Section 2. Section 3 discusses possible encoders for grey-level compression. In Section 4, we describe data transformations applied to grey values to increase their effectiveness. In Section 5, we introduce the context-blending method as a means to improve the compression. In Section 6, we perform a detailed objective and subjective quality analysis of four images. Section 7 concludes our work with a summary and an outlook on future work.

## 2. Partial Differential Equations in Image Compression

Partial differential equations are mainly used for image preprocessing [32–34] or as a tool for the postprocessing of image errors arising during coding. Partial differential equations are particularly useful for data compression and interpolation in scattered data, which was enabled by the improvement of binary tree coding by Distasi et al. [23].

### *2.1. Pde-Based Interpolation*

The main goal of PDE-based interpolation is to reconstruct the original image from the known pixels without losing their primary information. With PDE-based interpolation, the image can be reconstructed from sparse data with relative accuracy.

The concept of diffusion is known mainly from the physical context. It is a process that compensates for concentration differences without creating or destroying mass. This idea can be applied to image processing tasks, and we will formulate it in a continuous framework.

Let $\Omega \subset \mathbb{R}^n$ be an $n$-dimensional image domain. An unknown scalar-valued function $v : \Omega \to \mathbb{R}$ must be recovered, of which only its values on a subset $\Omega_1 \subset \Omega$ are known. The goal is to find an interpolating function $u : \Omega \to \mathbb{R}$ that is smooth and close to $v$ in $\Omega \setminus \Omega_1$ and identical to $v$ in $\Omega_1$.

This problem can be embedded in an evolution environment with an evolution parameter $t \geq 0$. Its solution $u(x, t)$ yields the desired interpolating function as its steady state ($t \to \infty$). The evolution is initialized with a function $f : \Omega \to \mathbb{R}$ initialized to $v$ identical to $\Omega_1$ and set to $\Omega \setminus \Omega_1$ to any value:

$$f(x) := \begin{cases} v(x) & \text{if } x \in \Omega_1 \\ 0 & \text{else.} \end{cases} \tag{1}$$

The evolution is considered

$$\partial_t u \;=\; (1-c(x))\, Lu \;-\; c(x)\,(u-f) \tag{2}$$

with $f$ as initial value,

$$u(x,0) \;=\; f(x), \tag{3}$$

and reflecting (homogeneous Neumann) boundary conditions on the image boundary $\partial\Omega$. The function $c : \Omega \to \mathbb{R}$ is the characteristic function on $\Omega_1$, i.e.,

$$c(x) := \begin{cases} 1 & \text{if } x \in \Omega_1 \\ 0 & \text{else,} \end{cases} \tag{4}$$

and $L$ is some elliptic differential operator. The idea is to solve the steady state equation

$$(1-c(x))\, Lu \;-\; c(x)\,(u-f) = 0 \tag{5}$$

with reflecting boundary conditions. In $\Omega_1$, $c(x) = 1$ such that the interpolation condition $u(x) = f(x) = v(x)$ is fulfilled. In $\Omega \setminus \Omega_1$, it follows from $c(x) = 0$ that the solution has to satisfy $Lu = 0$. This elliptic PDE can be regarded as the steady state of the evolution equation

$$\partial_t u = Lu \tag{6}$$

with the Dirichlet boundary conditions provided by the interpolation data on $\Omega_1$.

**Specific Smoothing Operators.** There are many possibilities for the elliptic differential operator $L$. The simplest and best studied uses the Laplace operator $Lu := \Delta u$, which leads to *linear isotropic diffusion* [35]:

$$\partial_t u = \Delta u, \tag{7}$$

also called *homogeneous diffusion (HD)*.

A prototype for a higher-order differential operator is the biharmonic operator $Lu := -\Delta^2 u$ providing the *biharmonic smoothing (BS)* evolution

$$\partial_t u = -\Delta^2 u. \tag{8}$$

Using it for interpolation comes down to thin plate spline interpolation [36], rotationally invariant multidimensional generalizations of the cubic spline interpolation.

The higher-order nonlinear diffusion considered in this paper is related to the differential operator based on Laplacian $Lu := -\Delta(g(|\Delta u|^2)\,\Delta u)$ proposed by [37]. *The fourth-order nonlinear diffusion* equation is

$$\partial_t u = -\Delta(g(|\Delta u|^2)\,\Delta u) \tag{9}$$

where the diffusivity function $g$ is

$$g(|\Delta u|^2) = \frac{1}{\sqrt{1+|\Delta u|^2/\lambda^2}}. \tag{10}$$

with some contrast parameter $\lambda > 0$. Since the highest derivative operator is 4, it will be denoted as *fourth-order Charbonnier diffusion (4ChD)*.

Finally, the nonlinear anisotropic diffusion considers $Lu := \operatorname{div}(g(\nabla u_\sigma \nabla u_\sigma^\top)\,\nabla u)$, namely *edge-enhancing diffusion (EED)* [38] with

$$\partial_t u = \operatorname{div}\left(g(\nabla u_\sigma \nabla u_\sigma^\top)\,\nabla u\right). \tag{11}$$

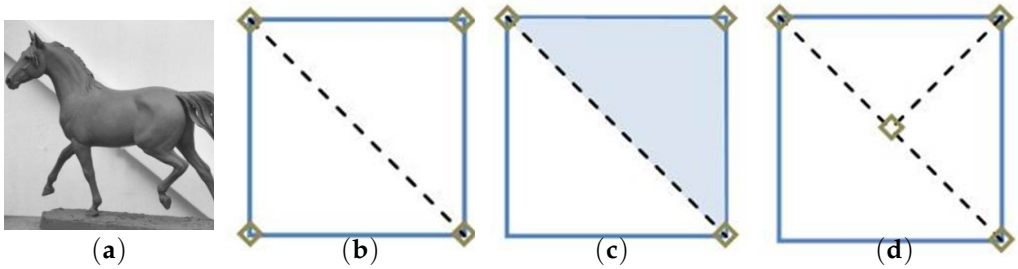
where the diffusivity function $g$ is the *Charbonnier diffusivity* [39]

$$g(|\nabla u|^2) = \frac{1}{\sqrt{1+|\nabla u|^2/\lambda^2}}. \tag{12}$$

### 2.2. Binary Tree Triangular Coding

In the B-Tree Triangular Coding Scheme (BTTC), the image is decomposed into a large number of triangular areas so that they can be reconstructed to a satisfactory quality using vertex interpolation. The data obtained during triangulation is stored in a binary tree structure.

This process is shown in Figure 1, where an initial image, Figure 1a, is approximated by its four vertices. The interpolation is performed using the information about the location of the vertices and their grey values. If the original image is presented by $(f_{i,j})$ and its approximation is $(u_{i,j})$ then an error can be defined $(e_{i,j})$ as $e_{i,j} := |u_{i,j} - f_{i,j}|$. If $(e_{i,j})$ satisfies $e_{i,j} \leq \varepsilon$ for every image element $(i,j)$ with given error threshold parameter $\varepsilon > 0$, the representation with triangles is considered to be sufficiently good.



(**a**) (**b**) (**c**) (**d**)

**Figure 1.** BTTC process. (**a**) Original image. (**b**) Triangle division process. (**c**) Reconstruction of inner values by interpolation. (**d**) Repeated triangle division process.

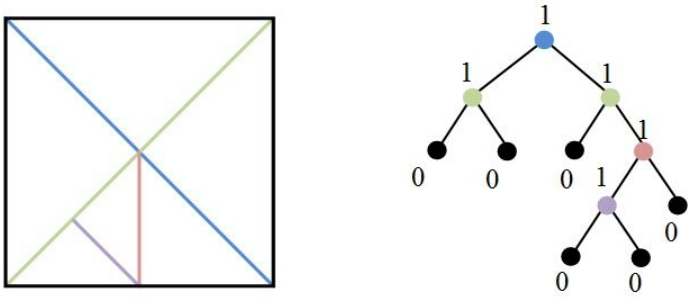
If the above equality is not satisfied, the triangle containing $(i,j)$ is divided into two similar triangles by reducing its height on the hypotenuse, Figure 1d. This is repeated until the given threshold for triangle formation is reached.

During this process, two images are generated. One is the mask image containing the coordinates of the vertices, and the other is a sparse image containing the information of the grey values in the same vertices.

### 2.3. Binary Tree Structure

During the triangulation process, a binary tree structure is formed in which each triangle is represented by a node, while the triangles that are no longer divided are represented by a leaf. A triangle is divided into two subordinate triangles. To save this tree structure, a preorder traversal is performed and a 1 is stored for each node and a 0 for each leaf. Additional space saving is achieved by storing two numbers *minimum tree depth* and *maximum tree depth*. The minimum tree depth is the tree depth up to which all nodes are dividing (all values are 1 s), and the maximum tree depth is the depth above which nodes are not divided any more (all values are 0 s).

The process of creating a binary tree structure is shown in Figure 2. In Figure 2a, the process is shown for a square image and, in Figure 2b, the resulting binary tree is generated. First, the image is divided by its main diagonal, then the process is repeated for the newly created triangles. To compress the grey values in all vertices, a zigzag traversal of the sparse image is performed and stored in the grey value stream. The pseudocode can be seen in the following text below:




(**a**) (**b**)

**Figure 2.** Process of creating the B-tree structure. (**a**) Process of dividing the triangles. (**b**) Corresponding binary tree structure.

Main algorithm: `Image` is grayscale image:

```
Step 1.  roottriangle = new Triangle(0,0,Image.width,Image.height)
Step 2.  Btree root = new Node(roottriangle,'')
Step 3.  CreateChildNodes(root)
Step 4.  EncodeImage(root)
```

Recursive procedure CreateChildNodes(node):

```
Step 1.  Let t = node.triangle
Step 2.  midx = t.x + t.wdith/2, midy = t.y + t.width/2
Step 3.  topleft = new Triangle(t.x,t.y,midx-t.x,midy-y)
Step 4.  botright = new Triangle(midx,midy,t.x+t.width-midx,y+t.height+midy)
Step 5.  tlaverage = ComputeAverageColorValue(topleft)
Step 6.  braverage = ComputeAverageColorValue(botright)
Step 7.  If tlaverage≠braverage then
             node.left = new Node(tltriangle,node.code+'0')
             node.right = new Node(brtriangle,node.code+'1')
             CreateChildNodes(node.left)
             CreateChildNodes(node.right)
```

Procedure EncodeImage(node):

```
Step 1.  If node=NULL return
Step 2.  Print node.code
Step 3.  EncodeImage(node.left)
Step 4.  EncodeImage(node.right)
```

Finally, due to the high similarity of the tree structure, the data stream is compressed using Huffman coding [40].

The final compressed file format consists of the following:

- Image height and width (4 byte);
- Minimum and maximum binary tree depth (2 byte);
- Binary tree structure in binary form (1 bit for every node);
- Huffman coded gray values.
  - First value of gray value in stream (1 byte);
  - Minimum and maximum Huffman binary tree depth (2 byte);
  - Huffman binary tree binary characters (1 bit for every node);
  - Gray values coded with Huffman coding.

We have further improved this encoding by a lossy requantization step that reduces the number of grey values in the original image from 256 to 64.

Binary Tree Decoding and Interpolation

Decompression of the encoded image is performed in two steps. In the first step, a reconstruction of the mask image is performed, and the stored grey values are placed in appropriate locations so that the sparse image can be created. For the reconstruction of the mask vertices, a tree is created in the same order as it was saved. The second step is the image interpolation, where the vertex mask is used as the interpolation mask. In their BTTC scheme, Distasi et al. [23] used linear interpolation. In this study, homogeneous diffusion, biharmonic and triharmonic smoothing, absolute monotone Lipschitz extension (AMLE), Charbonnier diffusion, and edge-enhancing diffusion are tested.

To check the quality of the interpolation methods, two methods of quantitative error analysis were used. These methods are: mean absolute error and mean square error of the decoded image $u_{i,j}$ and the original image $v_{i,j}$.

Average absolute error (AAE):

$$\mathrm{AAE}\,(u,v) := \frac{1}{nm}\sum_{i,j}|u_{i,j}-v_{i,j}|, \tag{13}$$

where $m$ depicts the image height and $n$ depicts the image width. The smaller value of $AAE$ represents a smaller deformation of the decoded image compared to the original one.

Mean square error (MSE):

$$\mathrm{MSE}\,(u,v) := \frac{1}{nm}\sum_{i,j}|u_{i,j}-f_{i,j}|^2. \tag{14}$$

The square error in the $MSE$ dampens the small differences between two image elements but emphasizes the big differences. The smaller values of $MSE$ represent a smaller error.

Implementing different types of interpolation on a test image *horse* is shown on Figure 1a; the results are shown in Table 1.

**Table 1.** Average absolute error (AAE) and mean square error (MSE) for the sparse data interpolation of the image *horse*.

| PDE Method | AAE | MSE |
|---|---|---|
| Homogeneous diffusion | 14.98 | 366.94 |
| Biharmonic smoothing | 13.16 | 373.52 |
| Triharmonic smoothing | 16.25 | 491.15 |
| AMLE | 14.91 | 376.99 |
| Charbonnier diffusion | 18.66 | 572.03 |
| Edge-enhancing diffusion | 12.52 | 353.12 |

According to the results shown in Table 1, the best results were achieved by using the edge-enhancing diffusion; therefore, this compression algorithm will be referred to as *EEDC—Edge-Enhancing Diffusion Compression*.

## 3. Gray Value Compression

To increase the compression ratio of the EEDC, a first possible solution is the gray value compression. A randomly selected portion of the grey values is selected as an example of the data to be compressed and is shown in Figure 3. This data stream is generated by the binary triangular encoding method described above.

[... 164 108 204 220 204 156 140 140 92 76 140 180 100 148 132 76 60
84 180 172 180 156 140 188 172 156 60 140 84 92 124 116 116 116
124 164 132 140 156 172 124 116 140 196 156 60 60 116 140 124 116 ...]

**Figure 3.** Random part of the gray value stream acquired using the EEDC.

When looking at the data stream in Figure 3, the regularity, repetition, and redundancy stand out, i.e., all the phenomena that compression methods use to increase the compression ratio. The compression methods described and used in this article are: Huffman coding, arithmetic coding, range coding, and LZ coding family.

### *3.1. Huffman Coding*

Huffman encoding is a lossless variable-length prefix encoding. This greedy algorithm considers the occurrence of each symbol in an optimal way, i.e., the characters that occur more frequently receive shorter codewords, while the characters that occur less frequently receive longer codewords [41]. In this way, Huffman coding reduces the number of bits needed to represent a set of symbols.

In this subsection, EEDC compression without modification of the grey values is compared with JPEG and JPEG 2000 compression methods at five compression ratios (0.8, 0.4, 0.2, 0.1, and 0.05 bpp). The compression analysis is performed for the previously mentioned Figure 1a).

The smaller the value of the AAE parameter, the better the quality of the reconstructed image. At a compression ratio of 0.8 bpp, as we can see from Table 2, the JPEG and JPEG 2000 compression methods are better than EEDC. At 0.4 bpp, JPEG 2000 achieves the best result, followed by EEDC, while JPEG has the worst result. At 0.2 bpp, JPEG 2000 is still better than EEDC, but, at 0.1 and 0.05 bpp, EEDC achieves the best quality of the reconstructed image. Figure 4 shows a graphical relationship of AAE between the above compression methods at different compression ratios.

**Table 2.** Comparison of Average Absolute Error for image *horse* on an unmodified compression algorithm.

| bpp | JPEG | JPEG 2000 | EEDC |
|---|---|---|---|
| 0.8 | 1.65 | 1.39 | 2.08 |
| 0.4 | 2.91 | 2.29 | 2.67 |
| 0.2 | 6.20 | 3.78 | 3.80 |
| 0.1 | 10.96 | 6.75 | 5.94 |
| 0.05 | / | 12.52 | 9.31 |

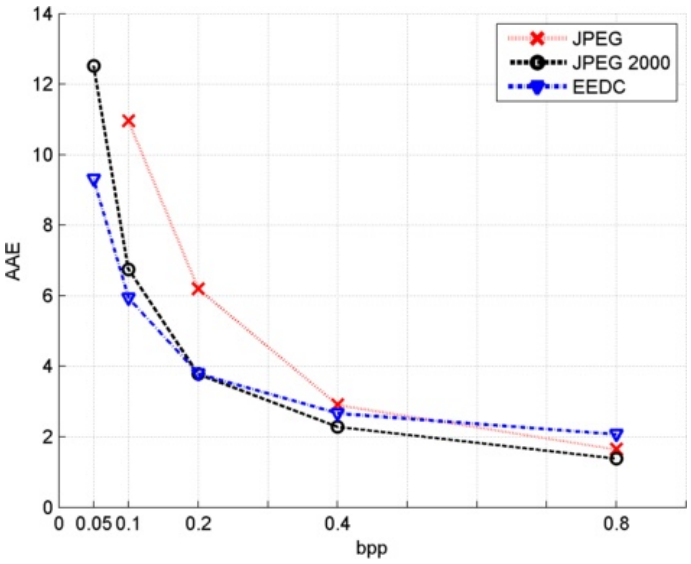



**Figure 4.** AAE for image *horse* and unmodified gray value compression algorithm.

### *3.2. Arithmetic Coding*

Arithmetic coding, similar to Huffman coding, belongs to a group of entropy coders; it assigns codes to symbols so that the code length corresponds to the probability of the occurrence of symbols, while other entropy coders decompose the input data stream into constituents, i.e., symbols, and replace each symbol with a codeword. Arithmetic coding encodes the entire message into a single number $n$ for which $n \in \langle 0, 1 \rangle$ [42]. Instead of the Huffman coding used in the original EEDC compression, an arithmetic coding is used, and the encoder is called *EEDC-Arith*. An equivalent analysis is performed as in the previous chapter. The encoders shown in Figure 5 were obtained by applying the new compression algorithm on the gray values series.

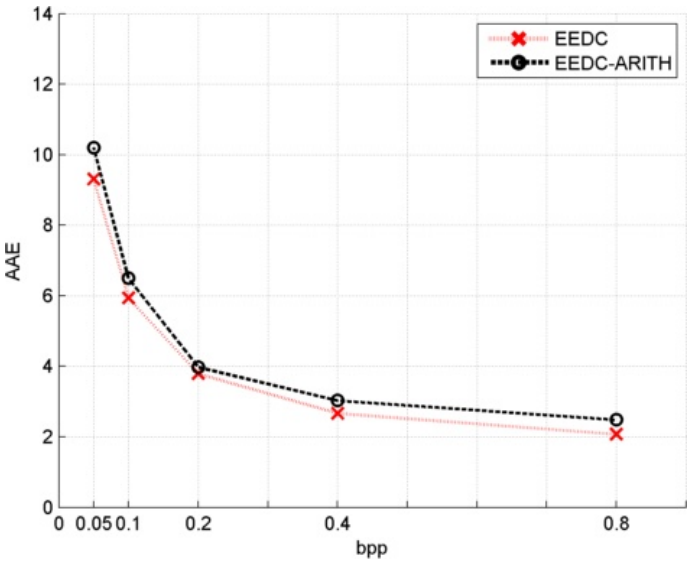



**Figure 5.** AAE for image *horse* and modified gray values compression using arithmetic coding.

From the graphical representation of the results in Figure 5, it can be concluded that the introduction of arithmetic coding did not yield better results, so this compression method is discarded.

### *3.3. Range Coding*

Range coding also belongs to the family of entropy coders and is very similar to arithmetic coding, with the main difference being the choice of the interval within which to search for a number that can represent a range of data as in [43]. In arithmetic coding, the interval within which a codeword is searched is $\langle 0, 1 \rangle$, while in range coding an interval is defined at $\langle 0, N \rangle$, where $N$ is the arbitrarily chosen upper bound of the interval.

Looking at the results presented in Figure 6, we can see that the quality of the reconstructed image has not improved, i.e., the direct indicator—AAE—is not lower at any compression ratio. The EEDC with Huffman coding for gray value compression is better than the EEDC with range coding.

### *3.4. Lz-Family Coding*

In this chapter, a family of compression algorithms is analyzed, based on the work of Lempel and Ziv [44,45]. These algorithms have a different approach to symbol compression. Instead of assigning code words to known symbols a priori, the code words are assigned to recurring symbols in the input data stream. The length of the recurring symbols can adopt the values from 1 to a certain constant value. There are three basic Lempel–Ziv algorithms, LZ77, which was described by Lempel and Ziv in 1977 [44], LZ78, which was described in 1978 [45], and LZW, which was described by Welsh in 1984 [46]. In addition to the three main algorithms, there are numerous other versions, but we will not discuss them in this article.

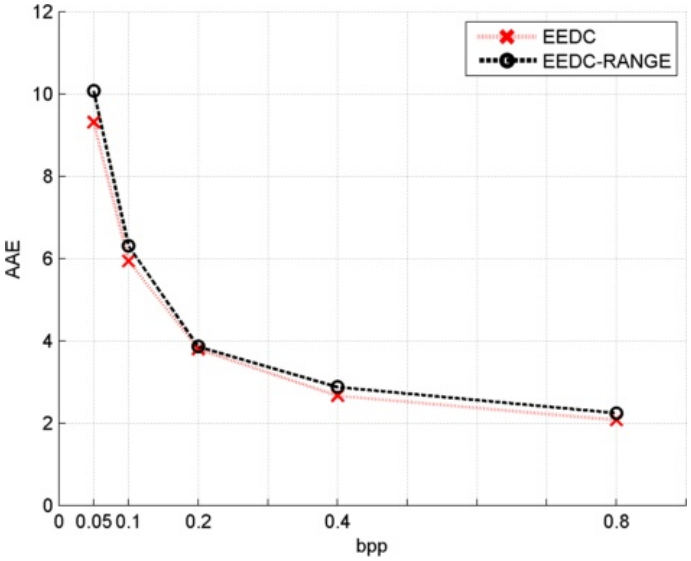



**Figure 6.** AAE for image *horse* and modified gray values compression with range coding.

#### 3.4.1. LZ77

The LZ77 algorithm is also known as the *sliding window compression algorithm*. This encoding algorithm holds $n - L_S$ symbols ($L_s$ is the maximum string length to be compressed, and $n$ is the size of the input buffer) to find a substring within $n - L_S$ that corresponds to the prefix of the input stream to be encoded. The found prefix is encoded with the position of the substring, its length, and the first symbol of the prefix. The length of the prefix can vary from 0 to $n - L_S$, and the greater it is, the greater the coding efficiency.

#### 3.4.2. LZ78

The LZ78 coding algorithm is also a *dictionary-based coding algorithm*. Instead of searching for the largest substring within a constant number of symbols as the LZ77 algorithm does, the LZ78 algorithm searches for the largest substring within all symbols that have appeared and represent the prefix to be compressed. In the case of an infinitely large input string, the location of the largest substring is unbound and complicates the encoding. To avoid this situation, the input string is divided into large blocks of length $n$ so that the largest strings can only begin at certain locations and that the location determines the length of the substring.

#### 3.4.3. Lzw Coding Algorithm

LZW uses the same coding principles as LZ78, but features technical improvements. In the LZW algorithm, when incrementally parsing the input stream, each word begins with the extension character of the previously parsed word. LZW uses fixed-length codewords, while LZ78 uses variable-length codewords. Welsh suggests 12-bit codewords in their article [46]. Due to the change in incremental parsing, the codeword contains only the index of the corresponding words in the dictionary, so each stream contains only one character.

#### 3.4.4. Mutual Comparison of the Lz-Family Coders

In this subsection, LZ77, LZ78, and LZW are compared with each other, and the procedure that achieves the best results is then compared with the original EEDC. The results of the mutual comparison in terms of coding efficiency are shown in Table 3.

From the results in Table 3, we can conclude that the best algorithm is LZ77. Although algorithms LZ78 and LZW are improvements of algorithm LZ77, they showed worse results in this example because their disadvantage is that they do not work well on small files. They only show their full potential with files that contain large amounts of data. At the beginning of the encoding process, the initialization of the dictionary occupies a lot of free memory due to the large amount of different data. As new entries are added to the

dictionary, the free space becomes smaller and smaller. For comparison with the original EEDC algorithm, only LZ77 is used.

**Table 3.** Average absolute error for image *horse* and LZ-family coders.

| bpp | LZ77 | LZ78 | LZW |
|---|---|---|---|
| 0.8 | 3.01 | 3.77 | 3.62 |
| 0.4 | 3.51 | 3.92 | 3.84 |
| 0.2 | 4.33 | 4.71 | 4.58 |
| 0.1 | 7.12 | 7.90 | 7.33 |
| 0.05 | 14.22 | 15.20 | 14.78 |

Although the LZ77 algorithm is the best in its family, the LZ77, according to Figure 7, performs worse than the unmodified EEDC algorithm with Huffman coding. Therefore, the LZ77 algorithm is also rejected because it did not improve.

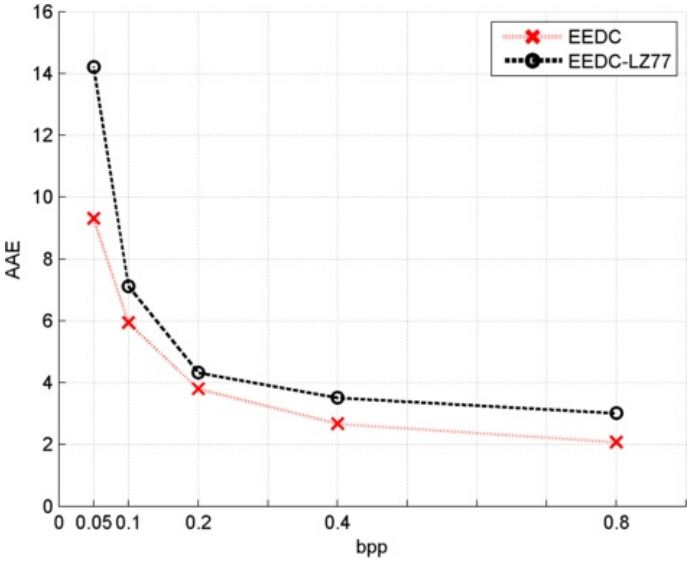



**Figure 7.** AAE for image *horse* and modified gray values compression with LZ77 coding.

### *3.5. Comparative Analysis of the Described Coders*

In this subsection, all the described coders are compared at five compression rates (0.8, 0.4, 0.2, 0.1, and 0.05 bpp) on the test image *horse*. The cumulative results are shown in Table 4.

**Table 4.** Comparison of average absolute error for image *horse* and all modified compression algorithms.

| bpp | JPEG | JPEG 2000 | EEDC | EEDC-Arith | EEDC-Range | EEDC-LZ77 | EEDC-LZ78 | EEDC-LZW |
|---|---|---|---|---|---|---|---|---|
| 0.8 | 1.65 | 1.39 | 2.08 | 2.49 | 2.24 | 3.01 | 3.77 | 3.62 |
| 0.4 | 2.91 | 2.29 | 2.67 | 3.03 | 2.88 | 3.51 | 3.92 | 3.84 |
| 0.2 | 6.20 | 3.78 | 3.80 | 3.98 | 3.86 | 4.33 | 4.71 | 4.58 |
| 0.1 | 10.96 | 6.75 | 5.94 | 6.50 | 6.31 | 7.12 | 7.90 | 7.33 |
| 0.05 | / | 12.52 | 9.31 | 10.20 | 10.08 | 14.22 | 15.20 | 14.78 |

From careful examination of Table 4, it can be concluded that all of the described changes in grey level compression did not improve the quality of the reconstructed image. Changing the compression algorithm alone does not produce better results, so a modification of the input data stream is required.

## 4. Data Transformation Impact on Gray Values Compression

To increase the efficiency of compression, it is necessary to transform an input data stream, as shown in Figure 3. The methods for transforming input data streams described in this article are:

- Delta coding.
- CTW—Context Tree Weighting.
- BWT—Burrows Wheeler Transformation.
- PPM—Prediction by Partial Matching.
- DMC—Dynamic Markov Compression.

### *4.1. Delta Coding*

Delta coding is a data stream transformation in which symbols are stored as the difference between the current character and the previous character [47]. Fewer bits are needed to represent the differences between successive characters than the characters themselves. The histogram of the image *horse* can be seen in Figure 8.

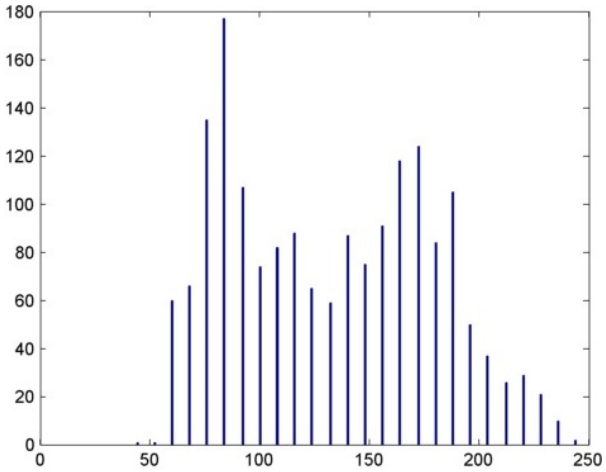



**Figure 8.** Histogram of the image *horse*.

After applying delta coding on the histogram data, the results can be seen of Figure 9.

When analyzing Figure 9, the frequency distribution is better for entropy coders because the differences in the frequencies of occurrence of each element are greater. For example, in Figure 9, there are about 250 elements with value 0 and about 10 elements with value 100. This difference should be exploited by Huffman coding and increase the effectiveness of the compression.

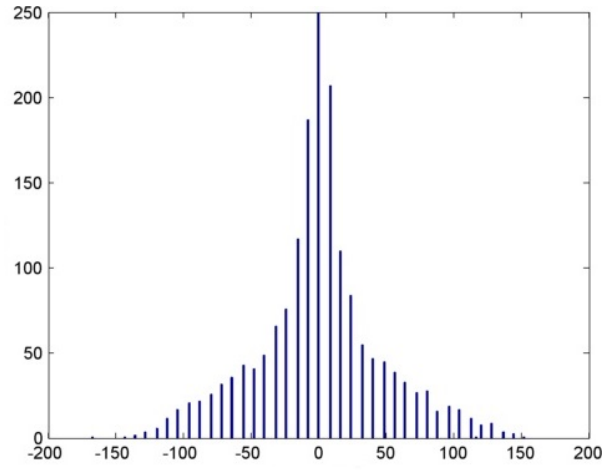



**Figure 9.** Histogram of the image *horse* after applying delta coding.

### *4.2. Context Tree Weighting*

The context tree weighting method was first described by Willems, Shtarkov, and Tjalkens in their 1995 article [48]. Context Tree Weighting (CTW for short) is an effective implementation of weighting the distribution of codes. The main components of the CTW codec are the CTW model estimator and the entropy encoder. The CTW model estimator is responsible for modeling the input stream, i.e., evaluating the probability of the next symbol occurring, while the entropy encoder uses the predicted probabilities to compress the input stream.

A very important part of the CTW algorithm is the context tree, which is dynamically created during encoding and decoding. The context is defined as all previous symbols of an input stream that is being encoded. Each context that occurs is stored as a path in a contextual tree. The encoding process consists of four stages:

- Searching for a path in the contextual tree that matches the current context.
- In every node in the contextual tree, the probability of the next symbol $P_e$ is predicted using data that are stored in the node itself (estimated probability is calculated using *Krichevski–Trofimov* estimation method or with *Zero-Redundancy* estimation method);
- Calculating the weighted probability $P_w$ on all $P_e$ values;
- Sending the weighted probability to entropy coder.

### *4.3. Burrows–Wheeler Transformation*

The Burrows–Wheeler transformation, or BWT for short, was described in a 1994 article by Burrows and Wheeler [49]. In BWT, the input data are not treated as a string, but the input stream is divided into blocks and each block is encoded independently. This transformation is most efficient when the input data are processed as one block. The basic idea is to reorder a current stream $S$ with $N$ symbols into another stream $L$, and the mathematical term for the reordering is *permutation*.

The encoding algorithm adopts a stream $S$ consisting of $N$ symbols $S[0], \ldots, S[N-1]$ selected from the sorted alphabet $X$. The encoder creates an $N \times N$ matrix $M$ and stores $S$ in its first row followed by $N-1$ copies of the stream $S$ cyclically shifted one symbol to the left. The matrix is sorted lexicographically by row, and the output of the BWT algorithm is the last symbol in each permuted stream and the row number where the original stream is found.

### *4.4. Prediction by Partial Matching*

Prediction by Partial Matching was invented in 1984 by Cleary and Witten [50]. It is a statistical method for modeling input data streams with a limited context, which can be viewed as merging multiple contextual models to predict the next symbol. The main idea of prediction by partial matching is to exploit the knowledge about the previous $K$ symbols to create a conditional probability for the current symbol.

### *4.5. Dynamic Markov Coding*

Dynamic Markov coding is an adaptive statistical compression procedure in two phases described by Cormack and Horspool in 1987 [51]. The first phase of this coding process consists of a finite state machine for predicting the probability of the next symbol and the second phase consists of an entropy encoder, usually arithmetic coding that performs the compression. This algorithm was originally developed for binary data, i.e., machine code, images, and sound. This algorithm reads a bit from the input stream, assigns it a certain probability value based on its previous occurrence, and switches it to the next state depending on its value. The algorithm starts with a small finite state machine that is assigned to new states during the encoding process, causing it to be an adaptive process. Thus, a finite state automaton can grow very quickly and fill the entire free memory. This algorithm is divided into two parts:

- Probability calculation;
- New states addition.

The probability calculation is performed by counting 0 and 1 in the input stream. Let it be assumed that the finite machine has been in the state $S$ several times in the past and has received $s_0$ zeros and $s_1$ ones. The easiest way to assign a probability is this simple expression:

$$\frac{s_0}{s_0 + s_1} \quad probability\ that\ the\ input\ data\ is\ 0.$$

$$\frac{s_1}{s_0 + s_1} \quad probability\ that\ the\ input\ data\ is\ 1.$$

### *4.6. Possible Combinations of Transformation and Compression*

Using the information in the last two sections, which describe all compression methods used and all data transformations, a table of all possible combinations can be created, Table 5.

**Table 5.** Possible combinations of coders and transformations.

| Compression Method | Input Stream Transformation | | | | |
|---|---|---|---|---|---|
| | Delta Coding | CTW | BWT | PPM | DMC |
| Huffman coding | ✓ | ✓ | ✓ | ✓ | |
| Arithmetic coding | ✓ | ✓ | ✓ | ✓ | ✓ |
| Range coding | ✓ | ✓ | ✓ | ✓ | ✓ |
| LZ77 | ✓ | | ✓ | ✓ | |
| LZ78 | ✓ | | ✓ | ✓ | |
| LZW | ✓ | | ✓ | ✓ | |

From Table 5, it can be seen that, with delta coding, Burrows–Wheeler transform, and Prediction by Partial Matching, all compression methods can be used. With the context tree weighting, all entropy encoders can be used, while with Dynamic Markov coding, only certain entropy encoders can be used, i.e., arithmetic coding and range coding. For all possible combinations, an analysis of the AAE parameter is performed for the image *horse* with a compression ratio of 0.2 bpp, Table 6.

**Table 6.** *AAE* for all possible combinations of the compression and transformation.

| Compression Method | Input Stream Transformation | | | | |
|---|---|---|---|---|---|
| | Delta Coding | CTW | BWT | PPM | DMC |
| Huffman coding | 3.89 | 3.91 | 3.84 | 4.12 | / |
| Arithmetic coding | 4.02 | 3.91 | 3.88 | 4.09 | 4.53 |
| Range coding | 3.91 | 3.84 | **3.61** | 3.93 | 4.11 |
| LZ77 | 4.21 | / | 3.91 | 4.51 | / |
| LZ78 | 4.55 | / | 4.17 | 5.01 | / |
| LZW | 4.34 | / | 3.99 | 4.89 | / |

For the compression to be more efficient than the original *EEDC*, the value of the AAE parameter must be lower than 3.80. From the results in Table 6, it can be seen that the entropy coders achieve better results compared to dictionary coders, which do not use their full potential due to the small size of the input data. The best result was obtained by a combination of range coding with Burrows–Wheeler transform; an AAE value of 3.61 was achieved. In further analysis, this combination will be *EEDC-RANGE (BWT)*.

### *4.7. Analysis of the EEDC-RANGE(BWT) Algorithm*

In this subsection, the new algorithm is tested at five compression ratios on the standard test image *horse*. Table 7 shows the values of the average absolute error for five encoders: JPEG, JPEG 2000, EEDC, and EEDC-RANGE (BWT).

**Table 7.** AAE values comparison.

| bpp | JPEG | JPEG 2000 | EEDC | EEDC-RANGE (BWT) |
|---|---|---|---|---|
| 0.8 | 1.65 | 1.39 | 2.08 | 2.02 |
| 0.4 | 2.91 | 2.29 | 2.67 | 2.52 |
| 0.2 | 6.20 | 3.78 | 3.80 | 3.61 |
| 0.1 | 10.96 | 6.75 | 5.94 | 5.65 |
| 0.05 | / | 12.52 | 9.31 | 8.68 |

By carefully observing Table 7, it can be seen that the novel algorithm beats its predecessor in all compression ratios. *EEDC-RANGE(BWT)* is better than the JPEG at 0.1, 0.2, and 0.4 bpp while it is only better than JPEG 2000 at 0.05, 0.1, and 0.2 bpp. The graphical representation is shown on Figure 10.

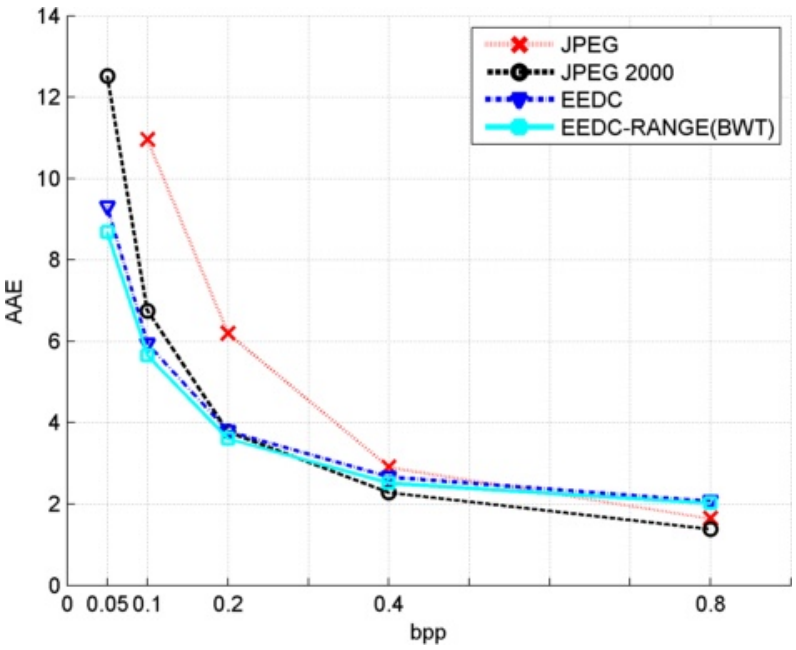



**Figure 10.** AAE comparison for image *horse*.

Figure 11 shows a series of reconstructed images at all compression ratios for all compression methods. Looking at the images shown, it is clear that the differences are more pronounced at higher compression ratios. The JPEG method was never able to produce a reconstructed image at 0.05 bpp. At 0.1 and 0.2 bpp, this new method is obviously better than other methods. At 0.4 and 0.8 bpp, it is difficult to see the differences between the tested methods. The analysis and the results obtained in this section have shown that the compression of grey values directly affects the quality of the reconstructed image. The improvement is due to the introduction of the Burrows–Wheeler transform and range coding. Still, this method is no better than JPEG 2000 at 0.4 and 0.8 bpp. The question arises, "Is it possible to achieve better results than JPEG 2000 at all compression ratios?", and the answer to this question follows in the next section.

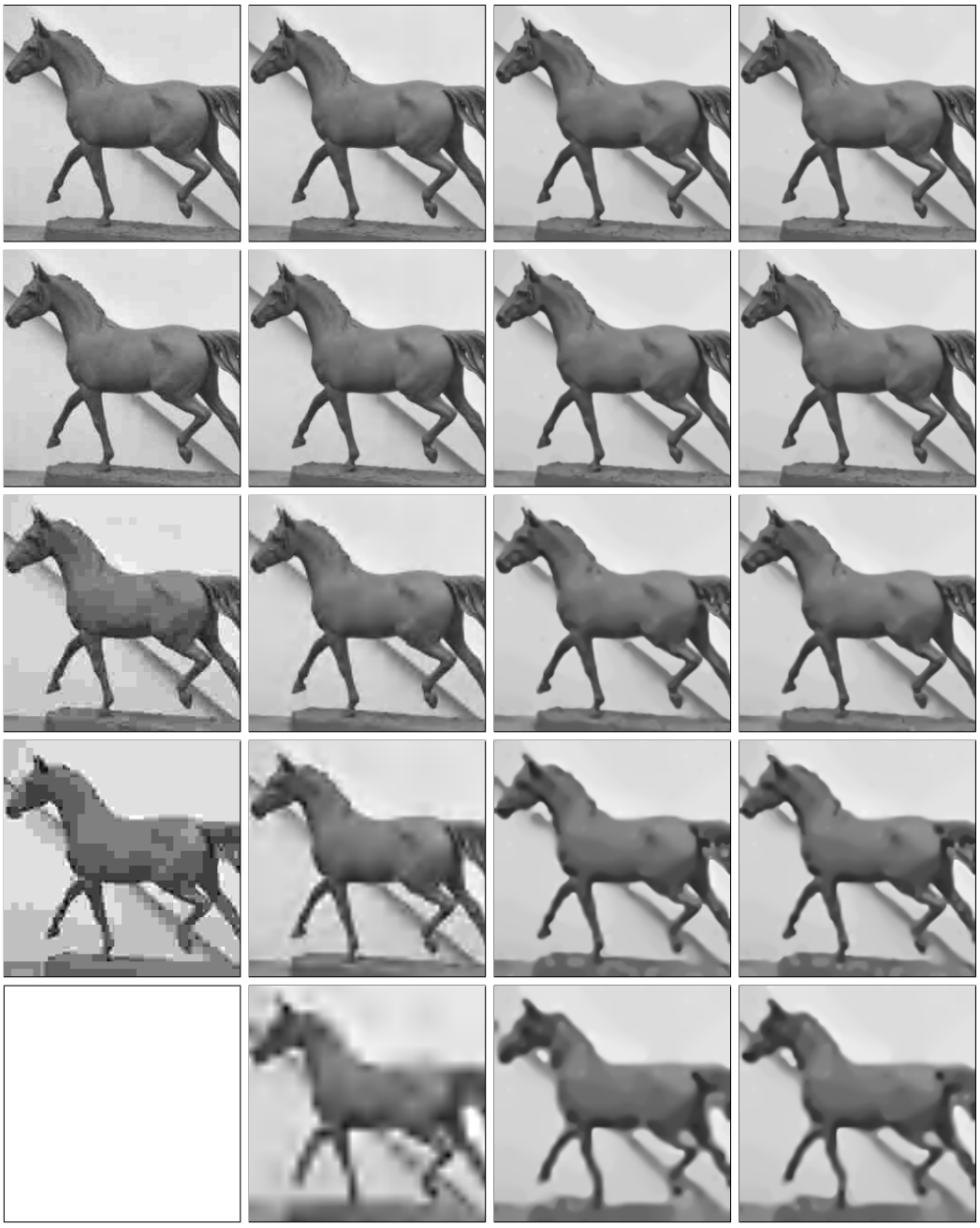



**Figure 11.** Comparative overview of reconstructed images ranging from 0.8 bpp (**top row**) to 0.05 bpp (**bottom row**). Compression methods: (**a**) JPEG, (**b**) JPEG 2000, (**c**) EEDC i, and (**d**) EEDC-RANGE (BWT).

## 5. Binary Tree Structure Compression

In the original compression method (EEDC), the binary tree structure is not compressed in any way. This structure, a stream of 1s and 0s, is stored in groups of 8 bits—*bytes*. In this section, the compression of the binary tree structure is analyzed using the compression methods and data transformation algorithms described earlier. An example of a binary tree structure can be seen in Figure 12.

[... 1 1 0 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1 1 ...]

**Figure 12.** A segment of the binary tree structure obtained using triangular coding.

When looking at Figure 12, certain blocks/sequences can be seen, but they have a size of 2 or 3 bits, which cannot be effectively used for increasing the compression ratio. The compression method *EEDC-RANGE (BWT)* is the basis of this research and, in this binary tree structure, compression is implemented in it. After applying data transformation and compression methods, the results can be seen in Table 8.

**Table 8.** *AAE* for possible combinations of data transformation and compression at binary tree structure coding.

| Compression Method | Data Stream Transformation | | | | | |
|---|---|---|---|---|---|---|
| | None | Delta Coding | CTW | BWT | PPM | DMC |
| Huffman coding | 4.75 | 6.12 | 4.22 | 4.68 | 4.66 | / |
| Arithmetic coding | 5.02 | 7.10 | 4.08 | 5.11 | 4.21 | 3.80 |
| Range coding | 5.33 | 7.24 | 3.99 | 5.39 | 4.13 | 3.84 |
| LZ77 | 4.68 | 5.88 | / | 4.65 | 4.34 | / |
| LZ78 | 4.89 | 6.01 | / | 4.93 | 4.48 | / |
| LZW | 4.76 | 5.94 | / | 4.87 | 4.42 | / |

The data reported in Table 8 proves that not a single value was below the reference value of 3.61 obtained by *EEDC-RANGE (BWT)* at 0.02 bpp. The best compression was achieved with dictionary coders, especially LZ77, due to the repetition of subsets in the data stream. By using delta coding, a new character, $-1$, appeared, which caused compression to be more difficult, as shown by the results in Table 6. Rearranging the bits, as BWT does, does not provide better results. The best result is obtained by combining DMC with arithmetic coding, but it is still above the value of 3.61.

*Context Mixing Method*

The shortcoming of methods that use contexts to predict the next symbol (DMC, PPM, and CTW) is that the context must be coherent. The best symbol prediction for images is adjacent horizontal and vertical pixels, but they do not form a coherent context. The previously mentioned methods do not provide a mechanism for combining statistics from contexts. *Context mixing method* combines predictions from a large number of independent models using weighted averaging [52].

The input data are represented as a stream of 1 s and 0 s. For each bit, each model independently provides two characters $n_0, n_1 \geq 0$, which can be used as the probability that the next symbol will be 0 or 1. Considering these two numbers together, they are the model's assertion that the next bit will be 0 with probability $n_0/n$ or 1 with probability $n_1/n$, where $n = n_0 + n_1$ is the model's relative confidence in this prediction. Since the models are independent of each other, the confidence is only meaningful when comparing two predictions of the same model. The models are combined by a weighted summation of $n_0$ and $n_0$ over all models as follows:

$$\begin{aligned} S_0 &= \epsilon + \sum w_i n_{0i} \quad \textit{evidence for } 0 \\ S_1 &= \epsilon + \sum w_i n_{1i} \quad \textit{evidence for } 1 \\ S &= S_0 + S_1 \quad \textit{total evidence} \\ p_0 &= \frac{S_0}{S} \quad \textit{probability that the next bit is } 0 \\ p_1 &= \frac{S_1}{S} \quad \textit{probability that the next bit is } 1, \end{aligned} \tag{15}$$

where $w_i \geq 0$ is the weight of the $i$'th model, $n_{0i}$ and $n_{1i}$ are outputs $n_0$ i $n_1$ of the $i$-th model, and $\epsilon > 0$ is a constant that guarantees $S_0, S_1 > 0$ i $p_0, p_1 \in (0, 1)$.

After coding each bit, the weights are adjusted in favor of the models that correctly predict that bit. Let $x$ be the first bit to be encoded in the sequence. The cost of the optimal

encoding of $x$ is $\log_2 1/p_x$. From the partial derivative of the encoding cost with respect to each $w_i$ in (15) with the constraint that the weights must be non-negative, we obtain the weight adjustment term:

$$w_i \leftarrow max\left[0, w_i + \frac{(x-p_1)(Sn_{1i} - S_1 n_i)}{S_0 S_1}\right], \tag{16}$$

where $n_i = n_{0i} + n_{1i}$. The term $(x - p_1)$ is the prediction error. The weights tend to grow logarithmically because the term $S_0 S_1$ grows along with the weights.

The predictions from Equation (15) are sent to arithmetic coder. If the input in the coder is $x$ then the output from the coder is the length of $x$ together with a number from a half-open interval $[p_{<x}, p_{<x} + p(x))$ where $p_{<x}$ is a probability that a random picked string is lexicographically smaller than $x$. Within the interval, there is a number with a base $B$ encoding that has maximum $1 + \log_b 1/p(x)$ digits [42]. When $p(x)$ is expressed as a product of conditional probabilities

$$p(x_1, x_2, \ldots, x_n) = \prod_i p(x_i | x_1, x_2, \ldots, x_{i-1})$$

and only 1 s and 0 s are used. Then, the arithmetic code can be computed efficiently so that it starts with the interval $[0, 1)$, and, for each bit $x_i$, the interval is divided into two parts proportional to $p_0$ and $p_1$ from Equation (15) and replaced by a sub-interval corresponding to $p_{xi}$. If the range is $[low, high)$ and the probability that $x_i$ is 0; then, the interval updates as follows:

$$\begin{aligned} mid &= low + p_0(high - low) \\ [low, high) &\leftarrow [low, mid) \ \ if\ x_i = 0 \\ &\quad\ [mid, high) \ \ if\ x_i = 1 \end{aligned} \tag{17}$$

As the interval shrinks, the leading digits of the *low* and *high* will match; then, these digits are outputted immediately.

When the interval shrinks, the leading digits of *low* and *high* coincide and these digits are immediately output.

The context shuffling method applied to binary tree compression consists of 18 context models (10 models for general use, 4 models for long contexts, and 4 models for short contexts) and 8 sets of weights that choose a 4-bit context consisting of the 4 most significant bits of the previous data stream. This method was implemented in an *EEDC-RANGE (BWT)* encoder and was tested for the *horse* image with a compression ratio of 0.2 bpp. The best result for the *AAE* value is 3.61 using *EEDC-RANGE (BWT)*. When context mixing is used, the value of the *AAE* parameter is 3.47, which is significantly better than the previous algorithm. The new algorithm is named *EEDC-BSSP* (Block-Sorting Symbol Prediction) because it uses the block-sorting algorithm (Burrows–Wheeler Transform) for compressing the grey values and the symbol prediction method (Context Mixing Method) for compressing the binary tree structure.

## 6. Quality Analysis of the Reconstructed Image

The measures used to determine the quality of the reconstructed image are divided into objective and subjective. The objective measurements are determined using mathematical operations or by the use of measuring devices WB06. The subjective measurements are determined by human judgement.

For quality analysis of the *EEDC-BSSP* compression method, four images are used—*horse, beauty, mask, pills*—as can be seen on Figure 13.

The images have a size of 257 × 257 pixels. The images are compressed using JPEG, JPEG 2000, and EEDC-BSSP compression algorithms with five compression ratios (0.8, 0.4, 0.2, 0.1, and 0.05 bpp). The images are then decompressed and the reconstructed images are subjected to further analysis.

(**a**) (**b**) (**c**) (**d**)

**Figure 13.** Test images: (**a**) horse, (**b**) beauty, (**c**) mask, and (**d**) pills.

*6.1. Objective Quality Analysis*

Objective quality analysis is based on the difference of the original $f_{i,j}$ and reconstructed image $u_{i,j}$. Two metrics are already mentioned and they are *AAE*, (13), and *MSE* (14).

The signal-to-noise ratio (*SNR*) is a ratio of the power of the signal and power of the background noise:

$$\mathrm{SNR}(f,g) = 10 \cdot log_{10}\left(\frac{\sigma^2(g)}{\mathrm{MSE}}\right). \tag{18}$$

The amplitude of image elements has a range:

$$\mathrm{MAX} := |0.2^q - 1| \tag{19}$$

where $q$ is the number of bits needed to display the amplitudes of the original image. The MSE does not take MAX into consideration so the Peak Signal-to-Noise Ratio (PSNR) is introduced:

$$\mathrm{PSNR}\,(u,v) := 10\log_{10}\frac{\mathrm{MAX}^2}{\mathrm{MSE}} = 20\log_{10}\frac{\mathrm{MAX}}{\sqrt{\mathrm{MSE}}}. \tag{20}$$

*SSIM* (Structure Similarity) is a novel method of calculating similarity of two images [27]. The main idea is that the human visual system is suited for structural information processing and this measurement strives to measure differences in this information between original and reconstructed images. Let $x_i$ and $y_i$ be two discrete signals where $i = 1, 2, \ldots, N$. The average value of discrete signal $x_i$ is

$$\mu_x := \frac{1}{N}\sum_i^N x_i. \tag{21}$$

The average value of discrete signal $y_i$ is

$$\mu_y := \frac{1}{N}\sum_i^N y_i. \tag{22}$$

The variance of $x_i$ is

$$\sigma_x^2 := \frac{1}{N-1}\sum_i^N (x_i - \mu_x)^2. \tag{23}$$

The variance of $y_i$ is

$$\sigma_y^2 := \frac{1}{N-1}\sum_i^N (y_i - \mu_y)^2. \tag{24}$$

The covariance of $x_i$ and $y_i$ is

$$\sigma_{xy} := \frac{1}{N-1}\sum_i^N (x_i - \mu_x)(y_i - \mu_y). \tag{25}$$

The *SSIM* is locally calculated over a window of determined size (usually $8 \times 8$). For every local window, three parameters are calculated: brightness, contrast, and structure. The value that depicts the change of brightness intensity is calculated by:

$$l(x,y) := \frac{2\mu_x\mu_y}{\mu_x^2 + \mu_y^2}. \tag{26}$$

The contrast is calculated using the variance between the original and reconstructed image as

$$c(x,y) := \frac{2\sigma_x\sigma\mu_y}{\sigma_x^2 + \sigma_y^2}. \tag{27}$$

The structural similarity uses the covariance of two images and is calculated by

$$s(x,y) := \frac{\sigma_{xy}}{\sigma_x\sigma_y}. \tag{28}$$

These three components are combined

$$\mathrm{SSIM}(x,y) := [l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{\gamma} \tag{29}$$

where $\alpha, \beta$, and $\gamma$ are parameters that define the relative importance of each component. The special case is when $\alpha = \beta = \gamma = 1$; the resulting SSIM index is provided with the next expression.

$$\mathrm{SSIM}(x,y) = \frac{4\mu_x\mu_y\sigma_{xy}}{\left(\mu_x^2 + \mu_y^2\right)\left(\sigma_x^2 + \sigma_y^2\right)}. \tag{30}$$

*MS-SSIM* (Multi-Scale Structure Similarity) is based on *SSIM* [53], but the contrast and structure are calculated at different frequency scales, so it tests the image quality at different observation distances. The brightness is calculated only at the last scale, and the MS-SSIM is calculated as a combination of all the previously calculated coefficients at all other scales. The low-pass filter is iteratively applied to the original and reconstructed images, and the filtered image is requantized by a factor of 2. The original image is referred to as scale 1 and the largest scale as scale $M$. The change in brightness intensity (26) is calculated only at scale $M$ and denoted as $l_M(x,y)$. On a random scale $j$, the contrast $c_j(x,y)$ is computed with (27) and the structural similarity $s_j(x,y)$ is computed with (28).

$$\mathrm{MS-SSIM}(x,y) := [l(x,y)]^{\alpha_M}\prod_{j=1}^{M}[c(x,y)]^{\beta_j}[s(x,y)]^{\gamma_j}. \tag{31}$$

Similar to (29), the exponents $\alpha_M, \beta_j$, and $\gamma_j$ are used to assign the relative importance to the different components. Just as with the SSIM, the result of this measurement can range from 0 (no similarity) to 1 (identical images). The disadvantage of this measurement is that the calculation requires longer than with SSIM.

*VIF* (Visual Information Fidelity) [54] quantifies the information divided between the reference image and the distorted image relative to the information contained in the reference image. It uses *NSS* (Natural Scene Statistic) together with the distortion model of the image and the model of the human visual system, *HVS*. The result can range from 0 (no similarity) to 1 (identical images).

### *6.2. Review and Analysis of Objective Measurement Results*

In this subsection, a comparative analysis of three encoders—JPEG, JPEG 2000, and EEDC-BSSP—is performed for five compression rates and four images in Figure 13. These images were selected based on their characteristics to comprehensively test the new compression method. The image *horse* contains a large number of color gradients from one intensity to another, and the details are concentrated in a few parts of the image (head and tail). The image *beauty* was chosen because it contains parts with more details (face and

hair) and also features parts with less details (wall in the background). The image *mask* contains a small amount of intensity gradients, but the main reason this image was chosen is the faster transition between gradients and scattered details throughout the image. The image *pills* is selected because it contains a large amount of details and repetitive shapes.

#### 6.2.1. Objective Quality of the Test Image Horse

The reconstructed images for each compression method and ratio are shown in Figure 14. At the lowest compression ratio (top row), the images look almost the same. At 0.4 bpp, the differences between JPEG 2000 and EEDC-BSSP are imperceptible, while the degradation begins for the image compressed with JPEG. At 0.2 bpp, artifacts begin to appear on the JPEG image, while degradation is observed on the JPEG 2000 image. On the EEDC-BSSP image, loss of detail is evident but structure is preserved. At 0.1 bpp, the JPEG image is severely degraded, the structure and details of the JPEG-2000 image are distorted, and the EEDC-BSSP image still has the original structure but lacks details. At 0.05 bpp, the JPEG cannot reproduce the image, JPEG 2000 is very blurred, while the EEDC-BSSP image is degraded in its structure and details, but still a picture of a horse is recognizable. The data shown in Tables 9–11 are the numerical evidence of the image degradation shown in Figure 14.

From Table 9, it can be seen that, the lower the parameter *AAE*, the smaller the difference between the original and the reconstructed image, i.e., the smaller the distortion. Furthermore, the parameter *MSE* should be lower, while the parameter *SNR* should be larger, which means that the reconstructed image has better visual quality.

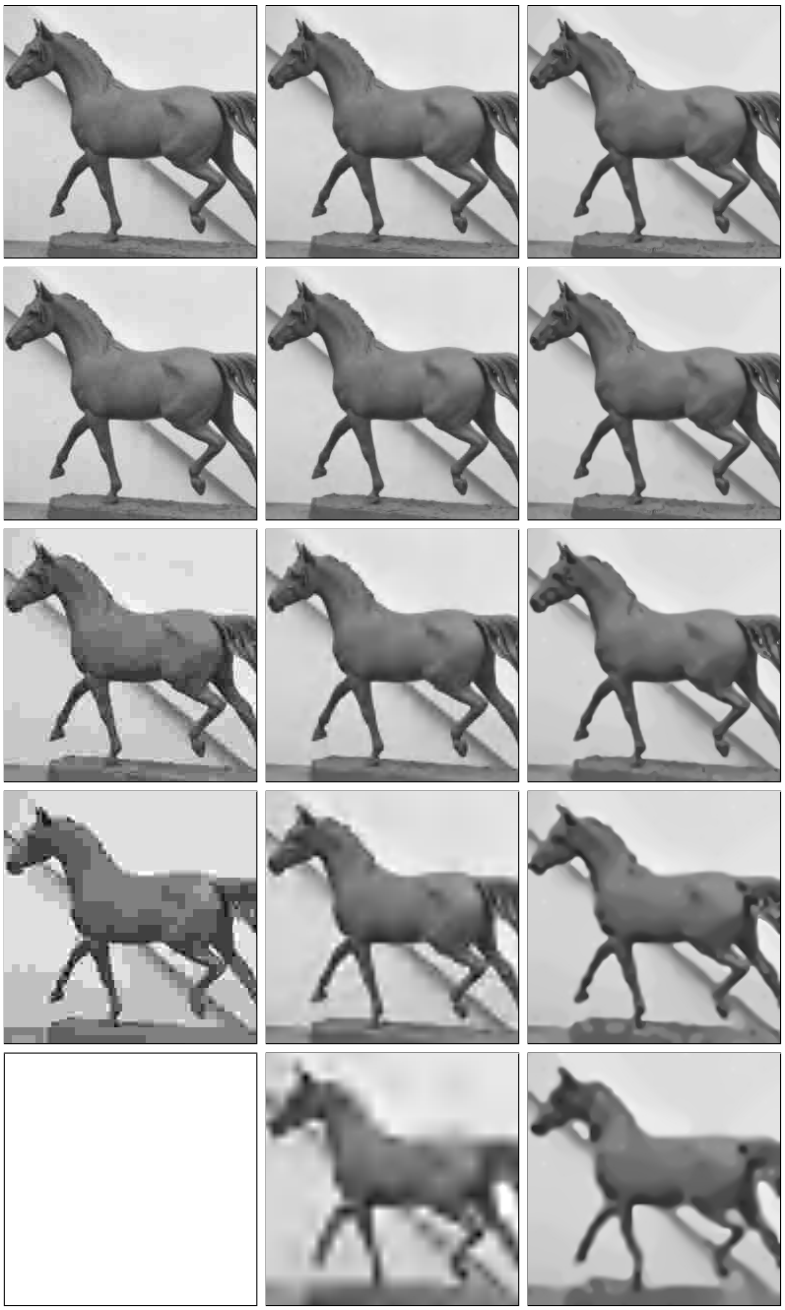
**Table 9.** Results of the AAE, MSE, and SNR metrics for image *horse*.

| bpp | AAE | | | MSE | | | SNR | | |
|---|---|---|---|---|---|---|---|---|---|
| | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP |
| 0.8 | 1.65 | 1.39 | 1.97 | 6.51 | 3.50 | 6.22 | 27.09 | 29.88 | 27.44 |
| 0.4 | 2.91 | 2.29 | 2.45 | 19.74 | 11.15 | 11.04 | 22.28 | 24.89 | 24.91 |
| 0.2 | 6.20 | 3.78 | 3.47 | 75.08 | 36.45 | 29.00 | 16.61 | 19.66 | 20.64 |
| 0.1 | 10.96 | 6.75 | 5.46 | 231.92 | 115.60 | 88.25 | 11.57 | 14.60 | 15.82 |
| 0.05 | / | 12.52 | 8.84 | / | 353.12 | 257.27 | / | 9.47 | 11.05 |

**Table 10.** Results of the PSNR, SSIM, and MS-SSIM metrics for image *horse*.

| bpp | PSNR | | | SSIM | | | MS-SSIM | | |
|---|---|---|---|---|---|---|---|---|---|
| | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP |
| 0.8 | 39.99 | 42.70 | 40.19 | 0.97 | 0.98 | 0.97 | 1.00 | 1.00 | 0.99 |
| 0.4 | 35.18 | 37.66 | 37.70 | 0.93 | 0.96 | 0.96 | 0.99 | 0.99 | 0.99 |
| 0.2 | 29.38 | 32.51 | 33.51 | 0.84 | 0.92 | 0.93 | 0.94 | 0.98 | 0.98 |
| 0.1 | 24.48 | 27.50 | 28.67 | 0.74 | 0.84 | 0.88 | 0.87 | 0.95 | 0.95 |
| 0.05 | / | 22.65 | 24.03 | / | 0.72 | 0.87 | / | 0.82 | 0.89 |

In Table 10, the parameter PSNR is similar to SNR, i.e., its value increases with the quality of the reconstructed image. The values of SSIM, MS-SSIM in Table 10, and VIF in Table 11 range from 0 (no similarity) to 1 (identical images). According to the SSIM values, EEDC-BSSP is better than the other two compression methods at compression ratios of 0.05, 0.1, and 0.2 bpp, while it is the same as JPEG 2000 at 0.4 bpp and the same as JPEG at 0.8 bpp, but JPEG 2000 has the best result at low compression. MS-SSIM has similar results. The VIF results tend to be better for EEDC-BSSP at all compression ratios except 0.8 bpp.



(**a**) (**b**) (**c**)

**Figure 14.** Comparative overview of reconstructed images for image *horse* ranging from 0.8 bpp (**top row**) to 0.05 bpp (**bottom row**). Compression methods: (**a**) JPEG, (**b**) JPEG 2000, and (**c**) EEDC-BSSP.

**Table 11.** Results of the VIF metric for image *horse*.

| bpp | JPEG | JPEG 2000 | EEDC BSSP |
|---|---|---|---|
| 0.8 | 0.75 | 0.80 | 0.75 |
| 0.4 | 0.59 | 0.66 | 0.67 |
| 0.2 | 0.38 | 0.50 | 0.54 |
| 0.1 | 0.23 | 0.33 | 0.38 |
| 0.05 | / | 0.17 | 0.23 |

6.2.2. Objective Quality of the Test Image Beauty

The reconstructed images for each compression method and ratio are shown in Figure 15. At the lowest compression ratio (top row), the quality of the JPEG image appears to immediately drop; JPEG 2000 has some edges, while EEDC-BSSP is smooth. JPEG has the worst image quality for all compression ratios, while JPEG 2000 and EEDC-BSSP are on par with only a slight difference. EEDC-BSSP is smoother in detail. At 0.05 bpp, the face is not recognizable on the JPEG 2000 image, while recognizable facial features are preserved on EEDC-BSSP. The results of the objective quality assessment are shown in Tables 12–14.

(**a**) (**b**) (**c**)

**Figure 15.** Comparative overview of reconstructed images for image *beauty* ranging from 0.8 bpp (**top row**) to 0.05 bpp (**bottom row**). Compression methods: (**a**) JPEG, (**b**) JPEG 2000, and (**c**) EEDC-BSSP.

**Table 12.** Results of the AAE, MSE, and SNR metrics for image *beauty*.

| bpp | AAE | | | MSE | | | SNR | | |
|---|---|---|---|---|---|---|---|---|---|
| | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP |
| 0.8 | 2.53 | 2.08 | 3.01 | 15.53 | 8.23 | 17.16 | 25.05 | 27.84 | 24.70 |
| 0.4 | 4.34 | 3.27 | 4.10 | 41.19 | 22.82 | 33.87 | 20.82 | 23.41 | 21.68 |
| 0.2 | 8.34 | 4.91 | 5.12 | 131.02 | 48.28 | 51.51 | 15.80 | 19.83 | 19.31 |
| 0.1 | 12.40 | 8.08 | 7.82 | 270.45 | 140.47 | 128.15 | 12.50 | 15.46 | 15.81 |
| 0.05 | / | 13.46 | 12.20 | / | 384.84 | 300.65 | / | 10.86 | 11.89 |

From Table 12, it can conclude that EEDC-BSSP performs better than JPEG 2000 only at 0.1 and 0.05 bpp, while it is better than JPEG at all compression ratios except 0.8 bpp.

**Table 13.** Results of the PSNR, SSIM, and MS-SSIM metrics for image *beauty*.

| bpp | PSNR | | | SSIM | | | MS-SSIM | | |
|---|---|---|---|---|---|---|---|---|---|
| | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP |
| 0.8 | 36.22 | 38.98 | 35.79 | 0.95 | 0.97 | 0.94 | 0.99 | 0.99 | 0.99 |
| 0.4 | 31.98 | 34.55 | 32.83 | 0.89 | 0.93 | 0.91 | 0.98 | 0.99 | 0.98 |
| 0.2 | 26.96 | 31.01 | 30.48 | 0.76 | 0.88 | 0.87 | 0.91 | 0.97 | 0.97 |
| 0.1 | 23.81 | 26.66 | 27.05 | 0.68 | 0.79 | 0.80 | 0.85 | 0.92 | 0.93 |
| 0.05 | / | 22.28 | 23.35 | / | 0.66 | 0.73 | / | 0.78 | 0.85 |

The SSIM, MS-SSIM parameters in Table 13, and VIF in Table 14 show similar results to the AAE, MSE, and SNR values. EEDC-BSSP is very similar to JPEG, but JPEG 2000 outperforms this new compression method at 0.2, 0.4, and 0.8 bpp, respectively.

**Table 14.** Results of the VIF metric for image *beauty*.

| bpp | JPEG | JPEG 2000 | EEDC BSSP |
|---|---|---|---|
| 0.8 | 0.69 | 0.74 | 0.65 |
| 0.4 | 0.53 | 0.60 | 0.55 |
| 0.2 | 0.32 | 0.49 | 0.46 |
| 0.1 | 0.21 | 0.32 | 0.34 |
| 0.05 | / | 0.19 | 0.23 |

#### 6.2.3. Objective Quality of the Test Image Mask

The reconstructed images for each compression method and ratio are shown in Figure 16. At the lowest compression ratio (top row) because of the low detail of this image, it is quite difficult to distinguish between images at 0.8 and 0.4 bpp. At 0.2 bpp, JPEG starts to degrade and artifacts appear, especially at 0.1 bpp. JPEG 2000 and EEDC-BSSP are quite similar, except that JPEG 2000 emphasizes the artifacts that degrade the image more.

Tables 15–17 show the results that follow the quality of the reconstructed images on Figure 16.

**Table 15.** Results of the AAE, MSE, and SNR metrics for image *mask*.

| bpp | PSNR | | | SSIM | | | MS-SSIM | | |
|---|---|---|---|---|---|---|---|---|---|
| | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP |
| 0.8 | 2.53 | 2.08 | 3.01 | 15.53 | 8.23 | 17.16 | 25.05 | 27.84 | 24.70 |
| 0.4 | 4.34 | 3.27 | 4.10 | 41.19 | 22.82 | 33.87 | 20.82 | 23.41 | 21.68 |
| 0.2 | 8.34 | 4.91 | 5.12 | 131.02 | 48.28 | 51.51 | 15.80 | 19.83 | 19.31 |
| 0.1 | 12.40 | 8.08 | 7.82 | 270.45 | 140.47 | 128.15 | 12.50 | 15.46 | 15.81 |
| 0.05 | / | 13.46 | 12.20 | / | 384.84 | 300.65 | / | 10.86 | 11.89 |

6.2.4. Objective Quality of the Test Image Pills

Figure 17 represents all the reconstructed images of image *pills*. Due to the high level of detail, i.e., the high frequencies in the image, it is very difficult to detect errors in the images at low compression ratios (0.8 and 0.4 bpp). The quality of JPEG images deteriorates rapidly as the compression rate increases. JPEG 2000 and EEDC-BSSP go head-to-head at high compression rates, and, at the highest compression rate (0.05 bpp), JPEG 2000 loses its integrity, while EEDC-BSSP managed to preserve the shapes. The results are consistent with the quality of the reconstructed images quality Tables 18–20.

(**a**) (**b**) (**c**)

**Figure 16.** Comparative overview of reconstructed images for image *mask* ranging from 0.8 bpp (**top row**) to 0.05 bpp (**bottom row**). Compression methods: (**a**) JPEG, (**b**) JPEG 2000, and (**c**) EEDC-BSSP.

(**a**) (**b**) (**c**)

**Figure 17.** Comparative overview of reconstructed images for image *pills* ranging from 0.8 bpp (**top row**) to 0.05 bpp (**bottom row**). Compression methods: (**a**) JPEG, (**b**) JPEG 2000, and (**c**) EEDC-BSSP.

**Table 16.** Results of the PSNR, SSIM, and MS-SSIM metrics for image *mask*.

| bpp | PSNR | | | SSIM | | | MS-SSIM | | |
|---|---|---|---|---|---|---|---|---|---|
| | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP |
| 0.8 | 36.22 | 38.98 | 35.79 | 0.95 | 0.97 | 0.94 | 0.99 | 0.99 | 0.99 |
| 0.4 | 31.98 | 34.55 | 32.83 | 0.89 | 0.93 | 0.91 | 0.98 | 0.99 | 0.98 |
| 0.2 | 26.96 | 31.01 | 30.48 | 0.76 | 0.88 | 0.87 | 0.91 | 0.97 | 0.97 |
| 0.1 | 23.81 | 26.66 | 27.05 | 0.68 | 0.79 | 0.80 | 0.85 | 0.92 | 0.93 |
| 0.05 | / | 22.28 | 23.35 | / | 0.66 | 0.73 | / | 0.78 | 0.85 |

**Table 17.** Results of the VIF metric for image *mask*.

| bpp | JPEG | JPEG 2000 | EEDC BSSP |
|---|---|---|---|
| 0.8 | 0.69 | 0.74 | 0.65 |
| 0.4 | 0.53 | 0.60 | 0.55 |
| 0.2 | 0.32 | 0.49 | 0.46 |
| 0.1 | 0.21 | 0.32 | 0.34 |
| 0.05 | / | 0.19 | 0.23 |

Tables 18–20 show the results that follow the quality of the reconstructed images on Figure 17.

**Table 18.** Results of the AAE, MSE, and SNR metrics for image *pills*.

| bpp | PSNR | | | SSIM | | | MS-SSIM | | |
|---|---|---|---|---|---|---|---|---|---|
| | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP |
| 0.8 | 1.75 | 1.48 | 1.92 | 6.55 | 3.95 | 5.98 | 26.15 | 28.44 | 26.67 |
| 0.4 | 3.29 | 2.47 | 2.59 | 22.05 | 12.19 | 12.40 | 20.88 | 23.56 | 23.41 |
| 0.2 | 6.17 | 4.25 | 4.19 | 72.37 | 39.12 | 38.71 | 15.74 | 18.41 | 18.46 |
| 0.1 | 11.94 | 7.23 | 6.51 | 243.65 | 114.56 | 102.60 | 10.55 | 13.68 | 14.13 |
| 0.05 | / | 13.34 | 10.29 | / | 362.35 | 257.54 | / | 8.20 | 9.92 |

**Table 19.** Results of the PSNR, SSIM, and MS-SSIM metrics for image *pills*.

| bpp | PSNR | | | SSIM | | | MS-SSIM | | |
|---|---|---|---|---|---|---|---|---|---|
| | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP | JPEG | JPEG 2000 | EEDC BSSP |
| 0.8 | 39.97 | 42.17 | 40.36 | 0.97 | 0.98 | 0.97 | 1.00 | 1.00 | 0.99 |
| 0.4 | 34.70 | 37.27 | 37.20 | 0.92 | 0.96 | 0.96 | 0.98 | 0.99 | 0.99 |
| 0.2 | 29.54 | 32.21 | 32.25 | 0.82 | 0.90 | 0.92 | 0.94 | 0.97 | 0.97 |
| 0.1 | 24.26 | 27.54 | 28.02 | 0.68 | 0.82 | 0.85 | 0.84 | 0.93 | 0.94 |
| 0.05 | / | 22.54 | 24.02 | / | 0.69 | 0.76 | / | 0.77 | 0.86 |

**Table 20.** Results of the VIF metric for image *pills*.

| bpp | JPEG | JPEG 2000 | EEDC BSSP |
|---|---|---|---|
| 0.8 | 0.76 | 0.81 | 0.77 |
| 0.4 | 0.59 | 0.68 | 0.68 |
| 0.2 | 0.39 | 0.50 | 0.52 |
| 0.1 | 0.21 | 0.34 | 0.37 |
| 0.05 | / | 0.18 | 0.24 |

### *6.3. Subjective Quality Analysis*

A competent measure of image quality is the subjective image quality, i.e., although two images may have identical values in the objective quality parameters, the human visual system can distinguish these two images. The standard procedures for subjective assessment are specified in Recommendation ITU-R BT.500-11 [55].

The subjective measurements can be divided into several groups in the ITU-R 500 recommendation, with the basic classification being general and alternative methods. Two general methods are *Double stimulus impairment scale—DSIS* and *Double stimulus continuous quality scale—DSCQS*. The alternative methods are *Single stimulus methods—single stimulus categorical rating and single stimulus continuous quality scale*.

The double stimulus impairment scale method is a method in which a pair of images is observed, one of which is the original image and the other the reconstructed image. For

each image, the observers provide a rating on the image distortion compared to the original image. The ratings can vary from 1 to 5, with 1 representing complete distortion of the image and 5 representing imperceptible distortion. The ratings are provided in Table 21.

**Table 21.** Image distortion ratings.

| Rating | Distortion |
|---|---|
| 1 | Particularly irritating |
| 2 | Irritating |
| 3 | Slightly irritating |
| 4 | Noticeably |
| 5 | Not noticeably |

After image grading, the average grade for all images is calculated and that grade is the *Mean Opinion Score—MOS*:

$$MOS = \sum_{i=1}^{5} i \cdot p(i) \tag{32}$$

where $p(i)$ is a share of grade $i$ in the total number of ratings. The double stimulus impairment scale provides stable results for the small amount of deterioration caused by changes in the level of distortion.

In double stimulus, the continuous quality scale method original image is first shown in the original image, then in the same image or a deteriorated image. The observer rates the quality of the second image. The evaluation of the reference image and the test image are performed using descriptive scores via Table 22.

**Table 22.** Image quality ratings.

| Rating | Quality |
|---|---|
| 1 | Unwatchable |
| 2 | Barely watchable |
| 3 | Watchable |
| 4 | Good |
| 5 | Excellent |

For optimal conditions in the determining subjective quality, it is necessary to provide the same screen, distance, and viewing angle for all observers.

### *6.4. Review and Analysis of Subjective Measurement Results*

In this article, the subjective assessment of image quality is performed using the double stimulus continuous quality scale. The image evaluation was performed by 40 observers, 8 of whom were female and 32 of whom were male. The observers first saw the original image, then the same image or the decoded image was shown. The subjective evaluation was performed using Table 22. After grading, the cumulative MOS was calculated for each tested image. The analysis for the image *horse* is shown in Table 23.

**Table 23.** MOS values for image *horse* at various compression ratios.

| bpp | JPEG | JPEG 2000 | EEDC BSSP |
|---|---|---|---|
| 0.8 | 4.53 | 4.52 | 4.13 |
| 0.4 | 3.41 | 3.80 | 4.24 |
| 0.2 | 2.06 | 2.81 | 2.65 |
| 0.1 | 1.23 | 1.61 | 1.96 |
| 0.05 | / | 1.13 | 1.57 |

As can be seen from Table 23, the competition was again between JPEG 2000 and EEDC-BSSP. EEDC-BSSP provided better results at high compression ratios and even received the best score at 0.4 bpp. At 0.8 bpp, JPEG was surprisingly rated the best. A graphical representation can be found in Figure 18.

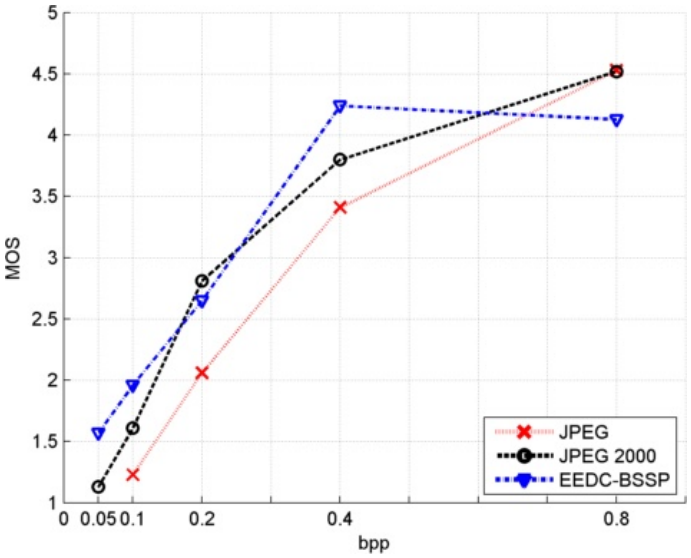



**Figure 18.** MOS values comparison for EEDC-BSSP with JPEG and JPEG 2000 for image *horse*.

From Figure 18, it is easy to see which compression method was found to be the best for a given compression ratio.

The quantitative analysis of the MOS result for the test image *beauty* is provided in Table 24 with its graphical representation on Figure 19.

**Table 24.** MOS values for image *beauty* at various compression ratios.

| bpp | JPEG | JPEG 2000 | EEDC BSSP |
|---|---|---|---|
| 0.8 | 4.51 | 4.56 | 4.52 |
| 0.4 | 3.25 | 4.03 | 3.71 |
| 0.2 | 2.07 | 3.05 | 2.84 |
| 0.1 | 1.31 | 1.67 | 1.53 |
| 0.05 | / | 1.11 | 1.42 |

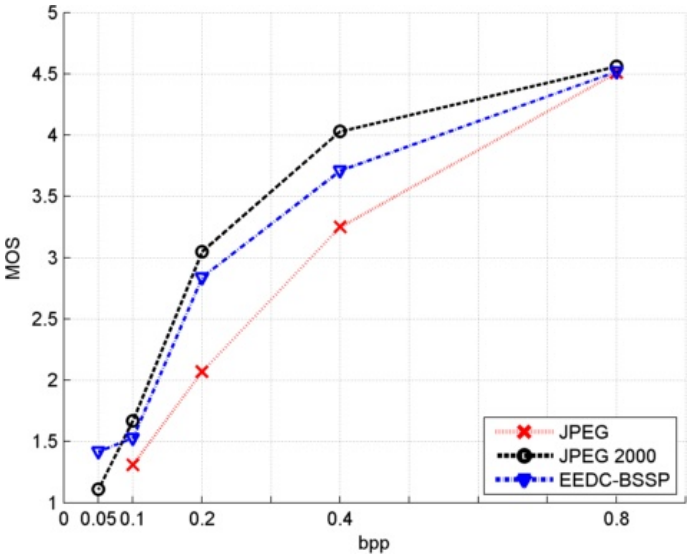



**Figure 19.** MOS values comparison for EEDC-BSSP with JPEG and JPEG 2000 for image *beauty*.

The values provided and the corresponding graph show that, for image *beauty*, JPEG 2000 performs best at all compression ratios except 0.05 bpp, and JPEG performs by far the worst at all compression ratios.

The calculated values of MOS for the test image *mask* are listed in Table 25. The graph of agreement is shown in Figure 20.

**Table 25.** MOS values for image *mask* at various compression ratios.

| bpp | JPEG | JPEG 2000 | EEDC BSSP |
|---|---|---|---|
| 0.8 | 4.91 | 4.95 | 4.92 |
| 0.4 | 3.50 | 4.04 | 4.46 |
| 0.2 | 2.91 | 3.36 | 3.16 |
| 0.1 | 2.05 | 2.16 | 2.17 |
| 0.05 | / | 1.16 | 1.11 |

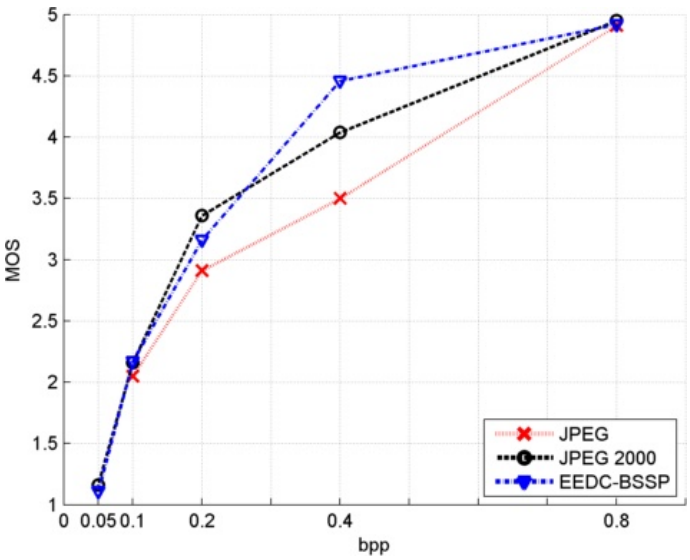



**Figure 20.** MOS values comparison for EEDC-BSSP with JPEG and JPEG 2000 for image *mask*.

For the image *mask*, JPEG was rated the worst for all compression ratios, but still JPEG 2000 and EEDC-BSSP are very close according to the results. It is really hard to distinguish these two algorithms for this image. The results for these two algorithms are similar except that, at 0.2 bpp, JPEG 2000 was rated as better than EEDC-BSSP, but, at 0.4 bpp, the roles reversed, i.e., EEDC-BSSP was better than JPEG 2000.

The numerical values of the MOS rating for the *pills* image are provided in Table 26 and the graph for this image is shown in Figure 21.

**Table 26.** MOS values for image *pills* at various compression ratios.

| bpp | JPEG | JPEG 2000 | EEDC BSSP |
|---|---|---|---|
| 0.8 | 4.73 | 4.84 | 4.00 |
| 0.4 | 3.16 | 3.92 | 3.62 |
| 0.2 | 2.40 | 2.34 | 2.73 |
| 0.1 | 1.22 | 1.91 | 1.76 |
| 0.05 | / | 1.00 | 1.14 |

For the image *pills*, JPEG 2000 was generally rated as the best compression method, but had a lower value than EEDC-BSSP at 0.2 bpp (seen as a "drop" at 0.2 bpp in Figure 21). Interestingly, JPEG was ranked better than EEDC-BSSP at 0.8 bpp, but all other values followed the same regularity as the previous images.

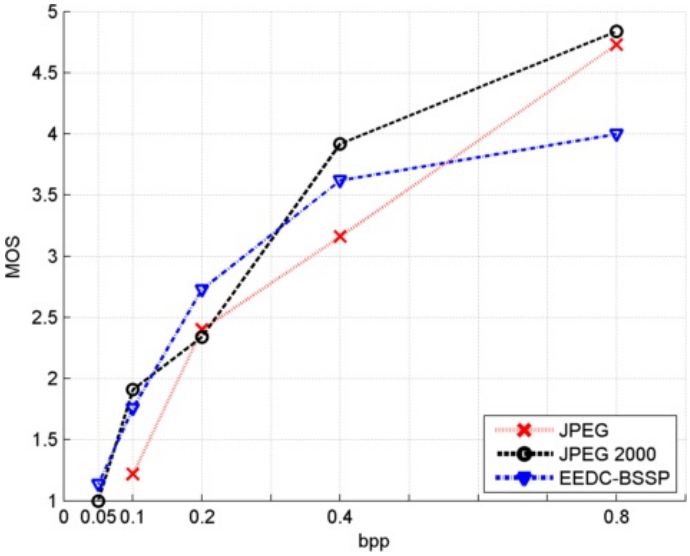



**Figure 21.** MOS values comparison for EEDC-BSSP with JPEG and JPEG 2000 for image *pills*.

## 7. Discussion

The objective and subjective image quality analyses performed in this study provide insight into the effectiveness of the presented algorithm *EEDC-BSSP* compared to the two standardized image compression methods, *JPEG* and *JPEG2000*. The four test images (*horse*, *beauty*, *mask*, and *pills*) were compressed at five different compression ratios ranging from 0.8 (lowest) to 0.05 (highest). The results for each image are presented in an image box allowing for a clear visualization of the quality loss of each image.

The objective image analysis was performed using several metrics, including AAE, MSE, SNR, PSNR, SSIM, MS-SSIM, and VIM. These metrics were presented in tables and used to evaluate the performance of each compression method for the four test images. The objective image analysis showed that EEDC-BSSP outperformed JPEG and JPEG2000 in terms of image quality at high compression rates.

The subjective image analysis was performed by calculating the MOS for each image using the double stimulus impairment method. The values of MOS were presented in tables and a graph, which showed a clear comparison of the performance of the three compression methods. The subjective analysis confirmed the superiority of EEDC-BSSP over JPEG and JPEG2000 at high compression ratios, as evidenced by the higher values of MOS.

Overall, the objective and subjective analyzes show that the presented algorithm EEDC-BSSP is more effective in preserving the image quality than the two standardized compression methods evaluated in this study. The results of this study could have important implications for applications that require high compression ratios without a significant loss of image quality, such as medical imaging and satellite imagery. Future work could include further optimization of the EEDC-BSSP algorithm to improve its performance for specific applications.

## 8. Conclusions

In this work, we have presented two extensions of the PDE-based image compression algorithm EEDC and demonstrated its superior performance compared to JPEG and JPEG 2000 at high compression rates. In particular, we analyzed the impact of grey-level compression and binary tree compression on the quality of the reconstructed image. Although PDE-based methods are more computationally intensive than transform-based compression methods (DCT in JPEG and DWT in JPEG 2000), the clever use of the available data and its statistics enables remarkable performance.

In our future work, we intend to optimize the EEDC-BSSP method for specific applications. Possible improvements include automatic selection of the diffusion and convergence rate parameters, i.e., whether decomposition and inpainting can be achieved in fewer iterations.

**Author Contributions:** Conceptualization, Č.L. and A.B.; methodology, Č.L.; software, A.B.; validation, T.H.; formal analysis, A.B.; investigation, Č.L.; resources, Č.L.; data curation, T.H.; writing—original draft preparation, Č.L.; writing—review and editing, Č.L., T.H., and A.B.; visualization, Č.L.; supervision, Č.L.; project administration, Č.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

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

## References

1. Bertalmío, M.; Sapiro, G.; Caselles, V.; Ballester, C. Image inpainting. In Proceedings of the ACM SIGGRAPH, New Orleans, LA, USA, 23–28 July 2000; pp. 417–424.
2. Chan, T.F.; Shen, J. Non-texture inpainting by curvature-driven diffusions (CDD). *J. Vis. Commun. Image Represent.* **2001**, *12*, 436–449. [CrossRef]
3. Grossauer, H.; Scherzer, O. Using the complex Ginzburg–Landau equation for digital impainting in 2D and 3D. In *Scale-Space Methods in Computer Vision, Volume 2695 of Lecture Notes in Computer Science*; Springer: Berlin, Germany, 2003; pp. 225–236.
4. Tschumperlé, D.; Deriche, R. Vector-valued image regularization with PDEs: A common framework for different applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 506–516. [CrossRef]
5. Kansal, K.; Singh, P. A review on image compression using partial differential equation. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2013**, *3*, 1192–1202.
6. Li, H.; Li, C.; Li, X.; Xu, X. An overview of partial differential equation-based image processing techniques. *J. Electron. Imaging* **2015**, *24*, 013011.
7. Saha, S.; Pramanik, S. A review on image compression using partial differential equations. *J. Comput. Theor. Nanosci.* **2018**, *15*, 1574–1584.
8. Li, S.; Zhang, Y.; Xie, X.; Li, X. Recent progress on partial differential equation-based image processing: Denoising, segmentation, inpainting, and beyond. *J. Electron. Imaging* **2017**, *26*, 011007.
9. Tai, X.C.; Chan, T.F. A survey on image denoising using PDEs. *J. Vis. Commun. Image Represent.* **2018**, *55*, 247–267.
10. Shapiro, J.M. Embedded image coding using zero trees of wavelet coefficients. *IEEE Trans. Signal Process.* **1993**, *41*, 3445–3462. [CrossRef]
11. Said, A.; Pearlman, W.A. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. Circuits Syst. Video Technol.* **1996**, *6*, 243–250. [CrossRef]
12. Taubin, G. A signal processing approach to fair surface design. In Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 15 August 1995; ACM : New York, NY, USA, 1995; pp. 351–358.
13. Kolouri, S.; Rohde, G.K. Variational image compression with a scale hyperprior. *IEEE Trans. Image Process.* **2018**, *27*, 4937–4950.
14. Caselles, V.; Morel, J.M.; Sbert, C. An axiomatic approach to image interpolation. *IEEE Trans. Image Process.* **1998**, *7*, 376–386. [CrossRef] [PubMed]
15. Galić, I.; Weickert, J.; Welk, M.; Bruhn, A.; Belyaev, A.; Seidel, H.P. Towards PDE-based image compression. In *Variational, Geometric and Level-Set Methods in Computer Vision, Volume 3752 of Lecture Notes in Computer Science*; Springer: Berlin, Germany, 2005; pp. 37–48.
16. Kostler, H.; Sturmer, M.; Freundl, C.; Rude, U. PDE based video compression in real time. In *Technical Report 07–11. Lehrstuhl fur Informatik 10*; University of Erlangen-Nurnberg: Erlangen, Germany, 2007.
17. Galić, I.; Weickert, J.; Welk, M.; Bruhn, A.; Belyaev, A.; Seidel, H.P. Image compression with anisotropic diffusion. *J. Math. Imaging Vis.* **2008**, *31*, 255–269. [CrossRef]
18. Schmaltz, C.; Weickert, J.; Bruhn, A. Beating the quality of JPEG 2000 with anisotropic diffusion. In *Pattern Recognition, Volume 5748 of Lecture Notes in Computer Science*; Springer: Berlin, Germany, 2009; pp. 452–461.
19. Mainberger, M.; Hoffmann, S.; Weickert, J.; Tang, C.; Johannsen, D.; Neumann, F.; Doerr, B. Optimising spatial and tonal data for homogeneous diffusion inpainting. In *Scale Space and Variational Methods in Computer Vision, Lecture Notes in Computer Science*; Springer: Berlin, Germany, 2012 ; Volume 66, pp. 26–37.

20. Schmaltz, C.; Peter, P.; Mainberger, M.; Ebel, F.; Weickert, J.; Bruhn, A. Understanding, optimising, and extending data compression with anisotropic diffusion. *Int. J. Comput. Vis.* **2014**, *108*, 222–240. [CrossRef]
21. Strobach, P. Quadtree-structured recursive plane decomposition coding of images. *IEEE Trans. Signal Process.* **1991**, *39*, 1380–1397. [CrossRef]
22. Sullivan, G.J.; Baker, R.J. Efficient quadtree coding of images and video. *IEEE Trans. Image Process.* **1994**, *3*, 327–331. [CrossRef]
23. Distasi, R.; Nappi, M.; Vitulano, S. Image compression by B-tree triangular coding. *IEEE Trans. Commun.* **1997**, *45*, 1095–1100. [CrossRef]
24. Demaret, L.; Dyn, N.; Iske, A. Image compression by linear splines over adaptive triangulations. *Signal Process.* **2006**, *86*, 1604–1616. [CrossRef]
25. Bougleux, S.; Peyré, G.; Cohen, L. Image compression with anisotropic triangulations. In Proceedings of the Tenth International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009.
26. Liu, X.; Sun, J.; Tang, X. An iterative total variation algorithm for compressive image sensing. *IEEE Trans. Image Process.* **2017**, *26*, 4787–4799.
27. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef]
28. Goh, P.; Ramesh, V. Multiscale edge-based PDEs for image compression. *IEEE Trans. Image Process.* **2010**, *19*, 645–656.
29. Chen, D.; Zhang, X.; Jiang, J. Learning-based image compression: Past, present and future. *Signal Process. Image Commun.* **2020**, *84*, 115797.
30. Livada, C.; Galic, I.; Zovko-Cihlar, B. EEDC image compression using Burrows-Wheeler data modeling. In Proceedings of the ELMAR 2012, Zadar, Croatia, 12–14 September 2012; pp. 1–5.
31. Livada, C.; Galic, I.; Zovko-Cihlar, B. EEDC image compression enhancement by symbol prediction. In Proceedings of the ELMAR 2013, Zadar, Croatia, 25–27 September 2013; pp. 59–63.
32. Chan, T.; Zhou, H. Feature preserving lossy image compression using nonlinear PDE's. *Adv. Signal Process. Algorithms Arch. Implementations VIII* **1998**, *3461*, 316–327.
33. Ford, G. Application of inhomogeneous diffusion to image and video coding. In Proceedings of the 13th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 3–6 November 1996; Volume 2, pp. 926–930.
34. Kopilovic, I.; Szirányi, T. Artifact reduction with diffusion preprocessing for image compression. *Opt. Eng.* **2005**, *44*, 1–14.
35. Iijima, T. Basic theory on normalization of pattern (in case of typical one-dimensional pattern. *Bull. Electrotech. Lab.* **1962**, *26*, 368–388.
36. Duchon, J. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *Rairo Math. Model. Methods Appl. Sci.* **1976**, *10*, 5–12. [CrossRef]
37. Didas, J.S.; Weickert, B.B. Stability and local feature enhancement of higher order nonlinear diffusion filtering. In *Pattern Recognition*; Lecture Notes in Computer Science; Kropatsch,W., Sablatnig, R., Hanbury, A., Eds.; Springer: Berlin, Germany, 2005; Volume 3663.
38. Weickert, J. Theoretical foundations of anisotropic diffusion in image processing. *Comput. Suppl.* **1996**, *11*, 221–236.
39. Charbonnier, P.; Blanc-Féraud, L.; Aubert, G.; Barlaud, M. Deterministic edge-preserving regularization in computed imaging. *Trans. Image Process.* **1997**, *6*, 298–311. [CrossRef]
40. Huffman, D. A method for the construction of minimum redundancy codes. *Proc. Ire* **1952**, *40*, 1098–1101. [CrossRef]
41. Mathur, I.K.; Loonker, S.; Saxena, D. Lossless Huffman Coding Technique For Image Compression and Reconstruction using Binary Trees. *Int. J. Comput. Technol. Appl.* **2012**, *3*, 76–79.
42. Howard, P.G.; Vitter, J.S. Analysis of arithmetic coding for data compression. *Inf. Process. Manag.* **1992**, *28*, 749–763. [CrossRef]
43. Martin, G.N.N. Range encoding: An algorithm for removing redundancy from a digitized message. *Video Data Rec. Conf. Southampt.* **1979** .
44. Lempel, A.; Ziv, J. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **1977**, *23*, 337–343.
45. Lempel, A.; Ziv, J. Compression of Individual Sequences via Variable-Rate Coding. *IEEE Trans. Inf. Theory* **1978**, *24*, 530–536.
46. Welsh, T.A. A technique for high-performance data compression. *Computer* **1984**, *17*, 8–19.
47. Salomon, D. *Data Compression: The Complete Reference*, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 2002.
48. Willems, F.; Shtarkov, Y.; Tjalkens, T. The Context Tree Weighting Method: Basic Properties. *IEEE Trans. Inf. Theory* **1995**, *41*, 653–664. [CrossRef]
49. Burrows, M.; Wheeler, D. *A Block-Sorting Lossless Data Compression Algorithm*; Digital Systems Research Center, SRC Research Report 124; Digital Systems Research Center: Palo Alto, CA, USA, 1994.
50. Cleary, J.; Witten, I. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Trans. Commun.* **1984**, *32*, 196–402. [CrossRef]
51. Cormack, G.V.; Horspool, R.N.S. Data compression using dynamic Markov modelling. *Comput. J.* **1987**, *30*, 541–550. [CrossRef]
52. Mahoney, M.V. *Adaptive Weighing of Context Models for Lossless Data Compression*; Florida Institute of Technology CS Dept Technical Report CS-2005-16; Florida Institute of Technology: Melbourne, FL, USA, 2005.
53. Wang, Z.; Simoncelli, E.P.; Bovik, A.C. Multiscale structural similarity for image quality assessment. In Proceedings of the 37th Asilomar Conference on Signals, Systems & Computers, Pacific Grove, CA, USA, 9–12 November 2003; Volume 2, pp. 1398–1402.

54. Sheikh, H.R.; Bovik, A.C. Image information and visual quality. *IEEE Trans. Image Process.* **2003**, *15*, 430–444. [CrossRef] [PubMed]
55. BT.500-11. *Methodology for the Subjective Assessment of the Quality of Television Pictures*; International Telecommunication Union/ITU Radiocommunication Sector: Geneva, Switzerland, 2012.