entropy

MDPI

Article

A New Transformation Technique for Reducing Information Entropy: A Case Study on Greyscale Raster Images

Borut Žalik [1,*], Damjan Strnad [1], David Podgorelec [1], Ivana Kolingerová [2], Luka Lukač [1], Niko Lukač [1], Simon Kolmanič [1], Krista Rizman Žalik [1] and Štefan Kohek [1]

[1] Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška Cesta 46, SI-2000 Maribor, Slovenia; damjan.strnad@um.si (D.S.); david.podgorelec@um.si (D.P.); luka.lukac@um.si (L.L.); niko.lukac@um.si (N.L.); simon.kolmanic@um.si (S.K.); krista.zalik@um.si (K.R.Ž.); stefan.kohek@um.si (Š.K.)

[2] Department of Computer Science and Engineering, University of West Bohemia, Technická 8, 306 14 Plzeň, Czech Republic; kolinger@kiv.zcu.cz

* Correspondence: borut.zalik@um.si

Abstract: This paper proposes a new string transformation technique called *Move with Interleaving* (MwI). Four possible ways of rearranging 2D raster images into 1D sequences of values are applied, including scan-line, left-right, strip-based, and Hilbert arrangements. Experiments on 32 benchmark greyscale raster images of various resolutions demonstrated that the proposed transformation reduces information entropy to a similar extent as the combination of the Burrows–Wheeler transform followed by the Move-To-Front or the Inversion Frequencies. The proposed transformation MwI yields the best result among all the considered transformations when the Hilbert arrangement is applied.

Keywords: computer science; algorithm; string transformation; information entropy; Hilbert space filling curve

check for updates

Citation: Žalik, B.; Strnad, D.; Podgorelec, D.; Kolingerová, I.; Lukač, L.; Lukač, N.; Kolmanič, S.; Žalik, K.R.; Kohek, Š. A New Transformation Technique for Reducing Information Entropy: A Case Study on Greyscale Raster Images. *Entropy* **2023**, *25*, 1591. https://doi.org/10.3390/e25121591

Academic Editor: Mateu Sbert

Received: 7 November 2023
Revised: 20 November 2023
Accepted: 22 November 2023
Published: 27 November 2023

1. Introduction

Information entropy is a measure for an uncertainty in data [1]. Data with lower entropy have reduced diversity and, consequently, are more predictable. The concept was introduced by Shannon [2]. It finds applications in various disciplines including computer science [3], mathematics [4], chemistry [5], mechanics [6], and statistics [7]. In computer science, we are often interested in determining the minimum number of bits required to encode a message X, where $X = \langle x_i \rangle$ is a sequence of symbols from the alphabet $\Sigma_X = \{x_i\}$. Each symbol $x_i \in \Sigma_X$ is assigned a probability p_i, which is calculated as the ration of the number of occurrences of x_i in X to the number of all symbols in X. Shannon's information entropy is calculated with Equation (1), and provides a lower bound on the average number of bits required to represent symbols $x_i \Sigma_X$.

$$H(X) = -\sum_{i=0}^{|\Sigma_X|-1} p_i \log_2(p_i). \tag{1}$$

The entropy is strongly related to the efficiency of various compression algorithms; lower entropy leads to better compression [8,9]. However, there are known techniques that can influence the information entropy of X [10], including predictions and transformations. This paper initially considers three such transformation techniques: Move-To-Front, Inversion Frequencies, and the Burrows–Wheeler Transform. A new transformation technique is proposed later.

This paper is divided into five Sections. Section 2 provides a brief explanation of Move-To-Front, Inversion Frequencies, and the Burrows–Wheeler Transform. Section 3 introduces the new transformation method named Move-With-Interleaving (MwI), and discusses various possibilities for arranging the data from the raster images into sequences X, which are then transformed. The results of applying the considered transformations on 32 benchmark greyscale raster images are presented in Section 4. Section 5 concludes the paper.

2. Background

String transformation techniques, including Move-To-Front [11], Inversion Frequencies [12], Sorted Inversion Frequencies [13], Frequency Count [14], Weighted Frequency Count [15], Distance Coding [16], Time Stamp [17], Burrows–Wheeler Transform [18], have attracted a lot of research attention in the past. In this paper, however, we will limit our focus to the most known techniques, namely Move-To-Front, Inversion Frequencies, Burrows–Wheeler Transform, and their combinations [19].

2.1. Move-To-Front Transform

Move-To-Front (MTF) transformation was introduced independently by Ryabko [11], and, shortly thereafter, by Bentley et al. [14]. It is one of the self-organising data structures [20]. MTF transforms $X = \langle x_i \rangle$, $x_i \in \Sigma_X$, into $Y = \langle y_i \rangle$, $y_i \in \Sigma_Y = \{0, 1, 2, \cdots, |\Sigma_X| - 1\}$. Not all elements from Σ_Y need to exist in Y. MTF changes the domain from Σ_X to the set of natural numbers including 0. The lengths of the sequences X and Y remain the same, i.e., $|X| = |Y|$. MTF utilises a list L with random access, and operates through the following steps:

- Initialisation: fill L with all $x_i \in \Sigma_X$,
- For each $x_i \in X$:
  - Find the index l in L where x_i is located;
  - Send l to Y;
  - Increment the positions of all x_k, where $0 \leq k < l$;
  - Move x_i to the front of L.

Let us consider an example in Table 1, where $X = \langle \text{barbara}\,|\,\text{barbara} \rangle$, $\Sigma_X = \{\text{a, b, r, } |\}$ and $H(X) = 1.806$. MTF transforms X into $Y = \langle 1, 1, 2, 2, 2, 2, 1, 3, 3, 2, 3, 2, 2, 2, 1 \rangle$, with $H(Y) = 1.457$. In this example, Y contains fewer symbols than $|\Sigma_X|$, although this is not always the case.

Table 1. MTF Transform: an example.

X		/ [1]	b	a	r	b	a	r	a	|	b	a	r	b	a	r	a
L	0	a	b	a	r	b	a	r	a	|	b	a	r	b	a	r	a
	1	b	a	b	a	r	b	a	r	a	|	b	a	r	b	a	r
	2	r	r	r	b	a	r	b	b	r	a	|	b	a	r	b	b
	3	|	|	|	|	|	|	|	|	b	r	r	|	|	|	|	|
Y			1	1	2	2	2	2	1	3	3	2	3	2	2	2	1

[1] Initialisation.

MTF reduces the information entropy in data by revealing local correlations. In fact, the sequences of the same symbols are transformed into 0, pairs of symbols are transformed into 1, triplets are transformed into 2, and so on. In some cases, repeated MTF transformations further reduce H [21].

The Inverse Move-To-Front (IMTF) Transform is straightforward. The input consists of the sequence of indices $Y = \langle y_i \rangle$ and the alphabet Σ_X. List L should be initialised in the same manner as in the MTF case (see Table 1). After that, indices $l = y_i$ are taken one by one from Y. The symbol x_i at index l in L is read and sent to X. L is then rearranged in the same way as during the transformation.

2.2. Inversion Frequencies

Transformation Inversion Frequencies (IF) was proposed by Arnavut and Magliveras [12,22]. IF accepts $X = \langle x_i \rangle$ as an input, where x_i is from the alphabet Σ_X, and transforms it into $Y = \langle y_i \rangle$, $y_i \in \Sigma_Y$, where $\Sigma_Y = \{0, 1, 2, \cdots, |X| - 1\}$. Similarly to MTF, IF transforms the input symbols into the domain of natural numbers, but this time, the limit is $|X|$ instead of $|\Sigma_X|$ as in the case of MTF. Of course, not all elements from Σ_Y need to be present in Y.

For each $x_i \in \Sigma_X$, IF stores the position (i.e., an index) of its first appearance in X, and calculates an offset for all subsequent occurrences of x_i. However, all symbols $x_j \in \Sigma_X$, $0 \leq j < i$, that have been used up to this point, are skipped over. The partial results for each x_i are stored in auxiliary sequences A_{x_i}, which are merged in Y at the end.

Let us transform $X = \langle \text{barbara}\,|\,\text{barbara} \rangle$ with IF, where $\Sigma_X = \{\text{a, b, r, }|\}$. The partial transformations are:

- $A_{\text{a}} = \langle 1, 2, 1, 2, 2, 1 \rangle$;
- $A_{\text{b}} = \langle 0, 1, 2, 1 \rangle$;
- $A_{\text{r}} = \langle 0, 0, 1, 0 \rangle$;
- $A_{|} = \langle 0 \rangle$.

The first 'a' is located at position 1 in X and, therefore, the first entry into A_{a} is 1. To reach the next 'a', two symbols ('r' and 'b') have to be skipped, and therefore, the next entry into A_{a} is 2. The remaining entries in A_{a} are obtained using the same principle. First, 'b' is located at index 0. Two symbols ('a' and 'r') exist before the next 'b'. However, 'a' was already used, giving the offset 1. The first appearance of 'r' in X is at the position 2. However, as 'b' and 'a' were already used they should be skipped, and therefore, the first entry in A_{r} is 0. All the auxiliary arrays are then merged into $Y = \langle 1, 2, 1, 2, 2, 1, 0, 1, 2, 1, 0, 0, 1, 0, 0 \rangle$ with $H(Y) = 1.566$. Expectantly, the values in the auxiliary sequences become smaller gradually, with all entries being zero for the last symbol.

Inverse Inversion Frequency (IIF) transformation requires information about the lengths of auxiliary sequences, i.e., the frequencies of the symbols in X, in addition to Y and Σ_X. In our example, $F = \langle 6, 4, 4, 1 \rangle$. However, F could be avoided by the introduction of a guard, which should not be an element in Σ_X. The guard then separates the elements from auxiliary sequences. As we know that the last auxiliary array only contains zeros, it can be avoided. If the guard is -1, then $Y = \langle 1, 2, 1, 2, 2, 1, -1, 0, 1, 2, 1, -1, 0, 0, 1, 0, -1 \rangle$. When the occurrence of the last symbol exceeds $|\Sigma_X|$, $|Y| < |X|$ could be advantageous, for example, for compression.

2.3. Burrows–Wheeler Transform

One of the ideas on how to transform X could be the generation of all possible permutations, and then selecting the one with the highest local correlations. The consecutive number of this permutation should be stored to reproduce the X. Unfortunately, the number of permutations grows exponentially by $|X|$, and this approach is, therefore, not applicable in practice. However, one of the permutations is obtained by sorting. The sorted sequence offers many good properties; among others, the local correlations are also emphasised. Unfortunately, an inverse transformation, which would convert the sorted sequence into its unsorted source, is not known. Burrows–Wheeler Transform (BWT), one of the most surprising algorithms in Computer Science [23], constructs the permutation of X, where the same symbols tend to be close together. In addition, only $O(1)$ additional information is needed to restore X. Transformation, as suggested by Burrows and Wheeler [18], consists of four steps:

1. Generating $|X|$ permutations of X through rotational shift-right operations;
2. Sorting the obtained permutations lexicographically;
3. Reading the BWT(X) from the last column of the sorted permutations;
4. Determining the position of X in the sorted array of permutations. This position is essential for reconstruction, and is considered a BWT index, i_{BWT};

The construction of BWT for $X = \langle \text{barbara} \,|\, \text{barbara} \rangle$ is shown in Table 2. The majority of the same symbols are placed together in the obtained result $Y = \langle \text{rbbbbrrr} \,|\, \text{aaaaaa} \rangle$. The position of X, $i_{BWT} = 9$, should be stored for reconstruction.

Table 2. BWT: an example.

i	Step 1	Step 2	Step 3	Step 4
0	barbara | barbara	abarbara | barbar	r	
1	arbara | barbarab	arabarbara | barb	b	
2	rbara | barbaraba	ara | barbarabarb	b	
3	bara | barbarabar	arbarabarbara | b	b	
4	ara | barbarabarb	arbara | barbarab	b	
5	ra | barbarabarba	a | barbarabarbar	r	
6	a | barbarabarbar	barabarbara | bar	r	
7	| barbarabarbara	bara | barbarabar	r	
8	barbarabarbara |	barbarabarbara |	|	
9	arbarabarbara | b	barbara | barbara	a	←
10	rbarabarbara | ba	rabarbara | barba	a	
11	barabarbara | bar	ra | barbarabarba	a	
12	arabarbara | barb	rbarabarbara | ba	a	
13	rabarbara | barba	rbara | barbaraba	a	
14	abarbara | barbar	| barbarabarbara	a	

$i_{BWT} = 9$ is needed to reconstruct X from Y. The first column C of the sorted array of permutations is obtained from Y straightforwardly by sorting (see Table 3). The first symbol is easily obtained from C, as it is pointed by $i_{BWT} = 9$, $C_9 = \text{'b'}$ and $X = \langle \text{b} \rangle$. The symbol 'b' is the fourth 'b' in C, and, therefore, it can be found in Y at the position 4. $C_4 = \text{'a'}$ is inserted into $X = \langle \text{ba} \rangle$. The found symbol was the fifth 'a' in C, so the fifth 'a' is searched for in Y. It is found at position 13, where the corresponding C_{13}='r' is added into $X = \langle \text{bar} \rangle$. The process continues until index i_{BWT} is reached again.

Table 3. Reconstructing BWT.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Y	r	b	b	b	b	r	r	r	|	a	a	a	a	a	a
C	a	a	a	a	a	a	b	b	b	b	r	r	r	r	|

3. Materials and Methods

Continuous-tone greyscale raster images (i.e., photographs) are used in our study, and therefore, the new transformation technique, introduced in Section 3.1, is designed accordingly. There are various transformations commonly applied to images among with Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) among the most widely used. These transformations are most frequently used for the spectral analysis of the image for the quantification of higher frequencies for lossy compression. The rare exception is JPEG2000 with LGT 5/3 wavelet which enables lossless compression. However, previous studies demonstrate higher compression ratios for more advanced prediction-based encoders for lossless compression, such as JPEG-LS [24,25], FLIF, and JPEG XL lossless [26]. Prediction methods are more commonly used for reducing information entropy for lossless image compression [27]. However, these methods are domain-dependent, whereas the proposed transformation methods are general.

Images are 2D structures that should be rearranged into 1D sequences to apply the aforementioned transformation techniques. However, this rearrangement can be performed in different ways, as discussed in Section 3.2.

3.1. Move with Interleaving

Let I be a greyscale raster image consisting of pixels $p_{x,y}$, $0 \leq x < N, 0 \leq y < M$, where $N \times M$ is the image resolution, and $p_{x,y} \in [0, 255]$. The pixels are arranged in the sequence $X = \langle x_i \rangle$, $x_i \in [0, 255]$, $|X| = N \times M$, on which transformation is applied in order to reduce the information entropy.

Neighbouring pixels $p_{x,y} \in I$, and therefore also consecutive symbols $x_i \in X$, often reveal local correlations. However, in the majority of cases in photographs, these correlations do not manifest as the sequences of the same values or repeated patterns; instead, these values tend to be just *similar enough* within a certain tolerance δ. It can be assumed that the suitable value for δ depends on the specific image, and therefore, it is experimentally determined in Section 4. The values of x_i change importantly only when the scene in I changes drastically (for example, such as during the transition from a branch of a tree the image background [28]). MTF would, in such a case (see Section 2.1), issue a long sequence of large indices to bring enough similar values near to the beginning of the list. Unfortunately, this means that the values in Y would be considerably dispersed, which is undesirable from the information entropy perspective.

The proposed transformation is based on the idea of MTF and, therefore, utilises list L with random access. Let $x_i \in X$ represent the value to be transformed. The position l of x_i in L should be found and l is sent to Y. The updating process of the L operates in two modes:

Mode 1: MTF is applied when $l \leq \delta$.

Mode 2: The temporal array T is filled with $2 \cdot \delta$ interleaved values, starting with x_i when $l > \delta$. The values from T are then inserted at the front of L, shifting all the remaining values in L accordingly. This is why we named this transformation *Move with Interleaving* (MwI).

Algorithm 1 presents a pseudocode for the MwI transform, which is demonstrated by an example, where the alphabet $\Sigma_X \in [0, 15]$. Given $X = \langle 7, 9, 11, 10, 2, \cdots \rangle$ as the first five elements of X, and let $\delta = 3$, the following steps are performed:

1. Function InitialiseL in Line 4 of Algorithm 1 obtains the first element from X ($x_0 = 7$) and $\delta = 3$ as input, and populates the list L. The first element in L becomes x_0, while $2 \cdot \delta$ elements from interval $[7 - 3, 7 + 3]$ are interleaved around this value. The remaining elements of L are then filled from the smallest up to the largest possible value from the interval $[0, 15]$, according to the alphabetical order in Σ_X. x_0 also becomes the first element in Y (see Line 5) to enable the same initialisation of L during the decoding phase. The situation after the initialisation is therefore:

$$\begin{aligned} &i = 0 \\ &X = \langle 7, 9, 11, 10, 2, \cdots \rangle \\ &L = \langle 7, 8, 6, 9, 5, 10, 4, 0, 1, 2, 3, 11, 12, 13, 14, 15 \rangle. \\ &Y = \langle 7 \rangle \end{aligned}$$

2. The remaining elements from X are then transformed within the For-loop starting at Line 6 and ending in Line 15. Let us follow the algorithm for some elements from X.

$i = 1$: Function in Line 7 finds the position $l = 3$ for $x_1 = 9$. l is inserted into Y in Line 8. The MTF transform in Line 10 is applied as $l \leq \delta$ to obtain the following situation:

$$\begin{aligned} &i = 1 \\ &X = \langle 7, 9, 11, 10, 2, \cdots \rangle \\ &L = \langle 9, 7, 8, 6, 5, 10, 4, 0, 1, 2, 3, 11, 12, 13, 14, 15 \rangle. \\ &Y = \langle 7, 3 \rangle \end{aligned}$$

$i = 2$: For $x_2 = 11$, function GetPosition returns $l = 11$, which is inserted into Y. As $l > \delta$, function FillT in Line 12 generates the interleaved values around $x_2 = 11$. It returns $T = \langle 11, 12, 10, 13, 9, 14, 8 \rangle$. Function ModifyL in Line 13 moves T in front of L, while other values are placed after it. We obtained:

$$
\begin{aligned}
&i = 2\\
&X = \langle 7, 9, 11, 10, 2, \cdots \rangle\\
&T = \langle 11, 12, 10, 13, 9, 14, 8 \rangle\\
&L = \langle 11, 12, 10, 13, 9, 14, 8, 7, 6, 5, 4, 0, 1, 2, 3, 15 \rangle.\\
&Y = \langle 7, 3, 11 \rangle
\end{aligned}
$$

$i = 3$: GetPosition returns $l = 2$ for $x_3 = 10$ and inserts l into Y. Function MTF is applied because $l < \delta$. The obtained situation is:

$$
\begin{aligned}
&i = 3\\
&X = \langle 7, 9, 11, 10, 2, \cdots \rangle\\
&L = \langle 10, 11, 12, 13, 9, 14, 8, 7, 6, 5, 4, 0, 1, 2, 3, 15 \rangle.\\
&Y = \langle 7, 3, 11, 2 \rangle
\end{aligned}
$$

$i = 4$: $x_4 = 2, l = 13, l > \delta$. Sequence $T = \langle 2, 3, 1, 4, 0, 5 \rangle$ contains only 6 elements this time, as the values being outside $[0, 15]$ are not inserted. The obtained situation is therefore:

$$
\begin{aligned}
&i = 4\\
&X = \langle 7, 9, 11, 10, 2, \cdots \rangle\\
&T = \langle 2, 3, 1, 4, 0, 5 \rangle\\
&L = \langle 2, 3, 1, 4, 0, 5, 10, 11, 12, 13, 9, 14, 8, 7, 6, 15 \rangle.\\
&Y = \langle 7, 3, 11, 2, 13 \rangle
\end{aligned}
$$

Algorithm 1 Transformation MwI

```
1:  function MwI($X$, $\delta$)                       ▷ Returns transformed sequence $Y$
2:                                                    ▷ $X$: input sequence; $\delta$: tolerance
3:      $i = 0$
4:      $L$ = InitialiseL($x_i$, $\delta$)            ▷ Initialisation of $L$ is done according to $x_{i=0}$
5:      $Y_0 = x_i$                                   ▷ The first entry in $Y$ is $x_{i=0}$
6:      for $i \leftarrow 1$ to $|X|$ do              ▷ For all other $x_i$
7:          $l$ =GetPosition($L$, $x_i$)              ▷ Find the position of $x_i$ in $L$
8:          $Y$ = AddToY($Y$, $l$)                    ▷ Store the position in $Y$
9:          if $l \leq \delta$ then                   ▷ If position is smaller that $\delta$
10:             $L$ = MTF($l$, $L$)                   ▷ Then rearrange $L$ according to MTF
11:         else                                      ▷ otherwise
12:             $T$ = FillT($x_i$, $\delta$)          ▷ Fill temporal sequence $T$
13:             $L$ = ModifyL($L$, $T$)               ▷ Place symbols in $T$ in front of $L$
14:         end if
15:     end for
16:     return $Y$                                    ▷ Returns transformed sequence
17: end function
```

The inverse MwI transformation (IMwI) is shown in Algorithm 2. As can be seen, it completely mimics the transformation procedure. The first element in Y represent the absolute value of x_0, and it is obtained in Line 3. x_0 is utilised to populate the list L in Line 4, and depended on the output sequence X in Line 5. All other elements in Y are processed with the *for-loop* starting in Line 6. The specific position l is obtained from Y (Line 7), the value v is retrieved from L at the position l (Line 8), and stored in X in Line 9. After that, the algorithm evaluates l with regard to δ and applies either MTF (Line 11) or resets the content of L in Lines 13 and 14. When all indices from X have been processed, the reconstructed values are returned in Line 17.

Algorithm 2 Inverse MwI transformation

```
function IMwI($Y$, $\delta$)                       ▷ Returns restored sequence $X$
                                                   ▷ $Y$: input sequence of indices; $\delta$: tolerance
    $x_0 = Y_0$                                    ▷ The first entry in $X$ is $Y_0$
    $L$ = InitialiseL($x_0$, $\delta$)             ▷ Initialisation of $L$ is done according to first element
    AddToX($X$, $x_0$)                             ▷ $x_0$ is sent to the reconstructed sequence $X$
    for $i \leftarrow 1$ to $|Y| - 1$ do           ▷ For all other $y_i$
        $l = Y_i$                                  ▷ Get the position from $Y$
        $v = L_l$                                  ▷ Get the value from $L$
        $X$ = AddToX($X$, $v$)                     ▷ Store the value in $X$
        if $l \leq \delta$ then                    ▷ If position is smaller that $\delta$
            $L$ = MTF($l$, $L$)                    ▷ Then rearrange $L$ according to MTF
        else                                       ▷ Otherwise
            $T$ = FillT($v$, $\delta$)             ▷ Fill temporal sequence $T$
            $L$ = ModifyL($L$, $T$)                ▷ Place symbols in $T$ in front of $L$
        end if
    end for
    return $X$                                     ▷ Returns restored sequence
end function
```

3.1.1. Time Complexity Estimation

The worst-case time complexity analysis for the considered transformation techniques is performed in this subsection.

MTF: In the worst-case scenario, the last element of L should always be moved to the front. There are $|\Sigma_X|$ elements in L and consequently, $T_{MTF}(X) = |\Sigma_X| \cdot |X|$. Since $|\Sigma_X| << |X|$, $T_{MTF}(X) = O(|X|)$.

IF: For each $x_i \in \Sigma_X$, all elements in X are always visited, resulting in $T_{IF}(X) = |\Sigma_X| \cdot |X|$. Again, since $|\Sigma_X| << |X|$, $T_{IF}(X) = O(|X|)$.

BWT: The algorithm presented in Section 2.3 has, unfortunately, A time complexity of $O(|X|^2 \log |X|)$, which limits its practical use for longer sequences. Later, it was shown that BWT can be constructed from the suffix array in linear time [23], and since there are known algorithms for constructing the suffix array in linear time [29,30], BWT itself can be obtained in $T_{BWT}(X) = O(|X|)$ time.

BWT+MTF: Based on the above analysis, the combination BWT, followed by MTF, works in $T_{BWT+MTF}(X) = T_{BWT}(X) + T_{MTF}(X) = O(|X|)$.

BWT+IF: Similarly, as above, the combination BWT, followed by IF, operates in time complexity $T_{BWT+IF}(X) = T_{BWT}(X) + T_{IF}(X) = O(|X|)$.

MwI: MwI operates in two modes. In mode 1, then $T_{MwI}(X) = T_{MTF}(X) = O(|X|)$. In mode 2, the algorithm performs two tasks. Firstly, it fills the auxiliary sequence T with $\Delta = 1 + 2 \cdot \delta$ elements. After that, it applies MTF Δ times resulting in a total of $T_{MwI}(X) = \Delta \cdot |\Sigma_X| \cdot |X|$ operations. Since $\Delta \leq |\Sigma_X| << |X|$, $T_{MwI}(X) = O(|X|)$.

3.2. Rearranging Raster Data in the Sequence

Images are typically rearranged into sequences using a Scan-line order, which is a heritage of television (see Figure 1a). Three other possibilities shall be used for our experiments:

- Left-right scan (Figure 1b);
- Strip scan (Figure 1c);
- Hilbert scan (Figure 1d).

The Strip arrangement requires a user-defined parameter h for the width of the strip. Its value is evaluated in Section 4. A well-established approach for transforming multidimensional data into a one-dimensional form is through the use of space-filling curves.

The Hilbert curve [31] has been frequently applied to images [10,32,33]. An implementation based on the state diagram [34] has been used for mapping between 2D images and 1D sequences, and vice versa. The complete Hilbert curve can only be constructed on images with resolutions equal to powers of 2 in both directions. However, images of different resolutions are quite common. Therefore, the Hilbert curve is cut off accordingly, as shown in Figure 1d.

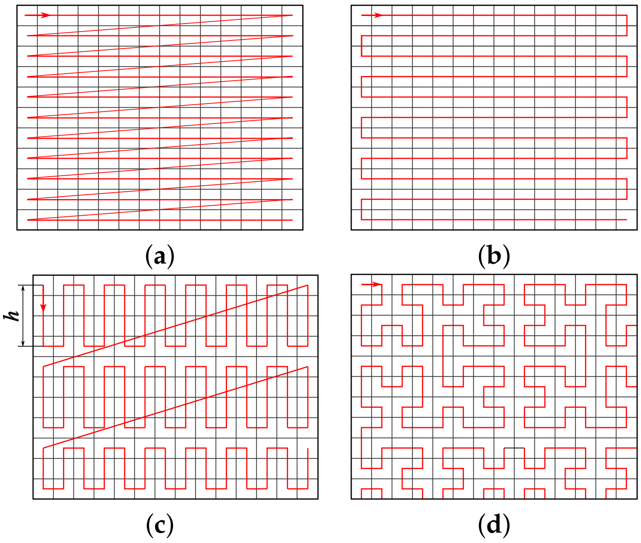

Figure 1. Different arrangements of pixels into a sequence: (**a**) Scan-line; (**b**) Left-right; (**c**) Strip; and (**d**) Hilbert.

4. Experiments

Figure 2 shows 32 benchmark 8-bit greyscale images used in the performed experiments. Table 4 gives the resolutions of these images in the second column, and their information entropies in the third one.

The information about the proposed transformation MwI is given in the fourth and fifth columns: firstly, the best values of δ, and secondly, the achieved information entropies. On average, the best value of δ is 12. However, $\delta = 11$ was used for further experiments, since 17 out of the 32 images achieved the best reduction in entropy with $\delta < 12$. The decrease in information entropy is shown in columns 6, 7, and 8 of Table 4 for MTF, IF, and MwI, respectively. MwI considerably outperformed MTF and IF.

BWT was used before MTF, IF, and MwI in the last three columns of Table 4. BWT had a considerably positive effect only on MTF and IF, but not on MwI. MwI, with its transformation mechanism, is capable of entirely replacing BWT. The last row of Table 4 shows the rank achieved by all the considered transformations. The ranking was as follows: BWT in front of IF was in the first place, MwI was in the second, BWT followed by MTF was in the third, while MwI after BWT, IF, and MWI were in the fourth, fifth, and the sixth places, respectively.

(1) Baboon (2) Ballons (3) Barb (4) Barbara

(5) Bark (6) Board (7) Boats (8) Cameraman

(9) Earth (10) Flower (11) Flowers (12) Fruits

(13) Girl (14) Gold (15) Lena (16) Malamute

(17) Maltese (18) Man (19) Mango (20) Mercury

(21) Monarch (22) Mosaic (23) Mushrooms (24) Parrots

(25) Pens (26) Peppers (27) Rainier (28) Sailing

(29) Sun (30) Sunrise (31) Yachts (32) Zelda

Figure 2. Testing raster images.

Table 4. Information about the images' resolutions and their entropies H, and the entropies obtained by different transformations—all for the Scan-line order.

I	$N \times M$	H	δ_{opt}	MwI(δ_{opt})	MTF	IF	MwI [1]	BWT MTF	BWT IF	BWT MwI [1]
(1)	512 × 512	7.357	25	6.471	7.308	7.415	6.567	6.656	6.680	6.661
(2)	720 × 576	7.346	5	4.074	5.649	5.497	4.172	3.989	3.920	4.008
(3)	720 × 576	7.484	13	6.033	6.936	6.868	6.037	6.285	6.112	6.248
(4)	512 × 512	7.343	16	5.351	6.769	6.888	6.370	6.156	6.020	6.518
(5)	512 × 512	7.325	40	6.757	7.517	7.849	7.016	6.792	6.873	6.786
(6)	720 × 576	6.828	9	4.526	5.454	5.364	4.535	4.617	4.459	4.625
(7)	720 × 576	7.088	11	5.043	6.008	5.909	5.043	5.266	5.095	5.271
(8)	256 × 256	6.904	11	5.551	6.349	6.323	5.551	5.834	5.667	5.799
(9)	512 × 512	7.155	17	5.544	6.740	6.702	5.598	5.569	5.511	5.555
(10)	512 × 480	7.410	8	4.764	6.375	6.288	4.803	4.553	4.463	4.515
(11)	500 × 362	7.305	17	5.608	6.832	6.668	5.641	5.823	5.629	5.808
(12)	512 × 480	7.366	10	4.840	6.294	6.218	4.844	4.805	4.657	4.740
(13)	720 × 576	7.288	12	5.111	6.489	6.455	5.114	5.166	5.067	5.178
(14)	720 × 576	7.530	14	5.262	6.485	6.308	5.268	5.516	5.382	5.547
(15)	512 × 512	7.348	10	5.184	6.629	6.720	5.189	5.240	5.118	5.233
(16)	1616 × 1080	7.792	14	5.427	6.994	6.865	5.439	5.326	5.153	5.321
(17)	2238 × 2446	6.964	6	3.912	5.056	4.932	4.036	3.927	3.961	4.051
(18)	1024 × 1024	7.524	15	5.480	6.900	6.834	5.511	5.662	5.523	5.682
(19)	1360 × 732	7.729	6	4.138	5.739	5.673	4.204	4.060	3.930	4.083
(20)	732 × 529	4.711	16	3.707	4.352	4.313	3.728	3.850	3.687	3.854
(21)	768 × 512	7.18	8	4.770	6.082	5.935	4.787	4.770	4.632	4.768
(22)	512 × 512	2.983	0	0.125	0.125	0.173	0.133	0.133	0.126	0.143
(23)	481 × 321	7.585	12	6.042	7.230	7.095	6.042	5.961	5.788	5.875
(24)	768 × 512	7.256	7	4.460	5.842	5.652	4.482	4.590	4.455	4.638
(25)	512 × 480	7.482	13	5.203	6.977	6.845	5.216	4.908	4.819	4.846
(26)	512 × 512	7.594	14	5.258	6.840	6.967	5.288	5.459	5.373	5.451
(27)	1920 × 1080	7.088	21	4.632	5.127	5.022	4.658	4.732	4.510	4.711
(28)	512 × 768	7.131	9	4.774	5.865	5.677	4.775	4.966	4.845	5.039
(29)	2100 × 2034	6.950	5	3.599	4.533	4.261	3.641	3.551	3.360	3.713
(30)	6000 × 2908	7.328	8	4.290	4.741	4.523	4.310	4.411	4.269	4.493
(31)	512 × 480	7.560	8	5.093	6.421	6.473	5.105	5.098	4.976	5.025
(32)	720 × 576	7.334	11	4.830	6.536	6.436	4.830	4.941	4.889	4.975
Average		7.102	12.2	4.870	6.039	5.973	4.935	4.949	4.842	4.974
Rank					6	5	2	3	1	4

[1] $\delta = 11$ was used.

Table 5 presents the average entropy of all 32 benchmark images when different pixel arrangements were used to obtain sequence X. The results are quite intriguing, and deserve further analysis. For example, MTF significantly benefited from the Strip order, but performed poorly on the Scan-line and Left-right orders. The same pattern also applies to IF. On the other hand, the effect of the arrangement type was reduced when BWT was used in front of MTF or IF. Even more, BWT followed by MTF or IF was the best when the Scan-line order was applied. The pipeline BWT followed by MwI yielded worse results compared to using MwI alone. Therefore, it can be concluded that MwI efficiently replaced BWT. It can be observed that MwI was also not very sensitive to the data arrangements. However, the Hilbert arrangement was the most suitable, as, in this case, MwI achieved the best result between all the tested transformation and data arrangements.

Table 5. Average entropies achieved for different arrangements of the pixels in sequences.

Order	MTF	IF	MwI	BWT MTF	BWT IF	BWT MwI
Scan-line	6.039	5.973	4.935	4.949	4.842	4.974
Left-right	6.004	5.905	4.925	4.959	4.843	4.977
Strips	5.111 [1]	5.036 [2]	4.962 [3]	5.118 [4]	5.000 [4]	5.149 [1]
Hilbert	5.349	5.905	4.754	5.012	4.889	5.050

[1] width of the strip $h = 4$. [2] width of the strip $h = 12$. [3] width of the strip $h = 8$. [4] width of the strip $h = 16$.

Besides the formal analysis provided in Section 3.1.1, it is even more important to consider how efficient the algorithm is in practice. Table 6 shows the CPU time spent on three techniques, all achieving a similar reduction in information entropy for seven images ranging from the smallest to the largest. The Scan-line order was used, and MwI was consistently the fastest in all cases.

Table 6. The CPU time spent in seconds for three transformation techniques, all achieving similar reductions in information entropy.

Image	No. of Pixels	BWT and MTF	BWT and IF	MwI
(8)	65,536	0.042	0.068	0.037
(1)	262,144	0.260	0.308	0.162
(6)	414,720	0.432	0.465	0.217
(18)	1,048,576	1.365	1.589	0.616
(16)	1,745,280	2.554	2.781	0.992
(27)	2,073,600	8.910	9.451	2.901
(30)	17,448,000	31.966	32.669	9.331

Personal computer with AMD Ryzen 5 5500 processor clocked at 3.60 GHz and equipped with 32 GB of RAM, running the Windows 11 operating system, was used in the experiments. The algorithms were programmed in C++ using MS Visual Studio, version 17.4.2.

5. Discussion

This paper introduces a transformation technique named *Move with Interleaving* (MwI). It operates in two modes. The first mode is the classical Move-To-Front, where the considered symbol x_i from the alphabet Σ_X is moved to the front of the list L. In the second mode, MwI moves $2 \cdot \delta$ symbols interleaved around x_i in front of L. As a result of MwI, less oscillating transformed values are obtained, which exhibit lower information entropy. The approach proves to be especially beneficial in the sequences of symbols where local correlations are manifested as similar symbols within a certain tolerance, rather than as completely identical symbols, or symbols that reveal repeating patterns. Continuous-tone raster images are typical examples of such data, and were used in this paper to illustrate the concept.

Pixels, which define a raster image, can be arranged into a sequence in various ways, with Scan-order being used the most frequently. Three other possibilities have been tried in this study: Left–right, Strip, and the Hilbert arrangement. The proposed MwI was compared against Move-To-Front (MTF) and Inversion Frequencies (IF) transformations, both individually, and after applying the Burrows–Wheeler transform (BWT).

A total of 32 benchmark 8-bit greyscale raster images with different resolutions and contexts were used in the experiments. The effect of the aforementioned transformations on the information entropy can be summarised as follows:

- When BWT is not used before MwI, it is considerably more efficient than MTF and IF.
- MwI is as efficient as BWT, followed by MTF or IF.
- BWT followed by MwI yields worse results in comparison to the results obtained by MwI alone.

- MwI is less sensitive to the arrangements on the input data compared to MTF and IF.
- MwI is the most efficient transformation technique when the Hilbert data arrangement is used.
- BWT, for its operations, requires the knowledge of the whole sequence in advance, while MwI operates incrementally and can, therefore, also be used in streaming applications.
- Implementing MwI is easier compared to BWT, as it does not require the implementation of a prefix array for computational efficiency.

At this point, it is worth mentioning that string transformation techniques are less efficient than the modern prediction methods in the domain of 2D continuous-tone raster images [27,35]. In future work, it would be interesting to investigate the combination of prediction-based methods and the proposed MwI transformation. A comprehensive comparison with other string transformation techniques and data domains, such as audio, should be conducted. And finally, open challenges remain: how to set δ for each individual data sequence, or even better, how to dynamically modify it during the processing of the considered data sequences.

Author Contributions: Conceptualisation, B.Ž.; methodology, D.S., Š.K. and N.L.; software, B.Ž. and K.R.Ž.; validation I.K. and D.P.; formal analysis, D.P. and Š.K.; investigation, B.Ž., D.P. and I.K.; resources, I.K.; data curation, Š.K. and L.L.; writing—original draft preparation, B.Ž.; writing—review and editing, D.S., I.K., Š.K., N.L., S.K. and K.R.Ž.; visualisation, L.L. and S.K.; supervision, I.K. and B.Ž.; project administration, I.K. and D.P.; funding acquisition, I.K. and B.Ž. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Slovenian Research and Innovation Agency under Research Project J2-4458, Research Programme P2-0041, and the Czech Science Foundation under Research Project 23-04622L.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2006.
2. Shannon, C.E. A Mathematical Theory of Communication. *AT&T Tech. J.* **1948**, *27*, 379–423.
3. Liu, S.; Xu, M.; Qin, Y.; Lukać, N. Knowledge Graph Alignment Network with Node-Level Strong Fusion. *Appl. Sci.* **2022**, *12*, 9434. [CrossRef]
4. Gray, R.M. *Entropy and Information Theory*, 2nd ed.; Springer: New York, NY, USA, 2011.
5. Sabirov, D.S.; Shepelevich, I.S. Information Entropy in Chemistry: An Overview. *Entropy* **2021**, *23*, 1240. [CrossRef] [PubMed]
6. Vasco-Olmo, J.M.; Díaz, F.A.; García-Collado, A.; Dorado-Vicente, R. Experimental evaluation of crack shielding during fatigue crack growth using digital image correlation. *Fatigue Fract. Eng. Mater. Struct.* **2013**, *38*, 223–237. [CrossRef]
7. Ben-Naim, A. Entropy, Shannon's Measure of Information and Boltzmann's H-Theorem. *Entropy* **2017**, *19*, 48. [CrossRef]
8. Sayood, K. *Introduction to Data Compression*, 4th ed.; Morgan Kaufman: Waltham, MA, USA, 2012.
9. Rahman, M.A.; Hamada, M. Lossless Image Compression Techniques: A State-of-the-Art Survey. *Symmetry* **2019**, *11*, 1274. [CrossRef]
10. Salomon, D.; Motta, G. *Handbook of Data Compression*, 5th ed.; Springer: London, UK, 2010.
11. Ryabko, B.Y. Data compression by means of a 'book stack'. *Probl. Pereda. Inform.* **1980**, *16*, 265–269.
12. Arnavut, Z.; Magliveras, S.S. Block sorting and compression. In Proceedings of the IEEE Data Compression Conference, DCC'97, Snowbird, UT, USA, 25–27 March 1997; Storer, J.A., Cohn, M., Eds.; IEEE Computer Society Press: Los Alamitos, CA, USA, 1997; pp. 181–190.
13. Abel, J. Improvements to the Burrows-Wheeler Compression Algorithm: After BWT Stages. 2003. Available online: https://api.semanticscholar.org/CorpusID:16110299 (accessed on 1 November 2023).
14. Bentley, J.L.; Sleator, D.D.; Tarjan, R.E.; Wei, V.K. A Locally Adaptive Data Compression Scheme. *Commun. ACM* **1986**, *29*, 320–330. [CrossRef]
15. Deorowicz, S. Improvements to Burrows-Wheeler Compression Algorithm. *Softw. Pract. Exper.* **2000**, *30*, 1465–1483. [CrossRef]

16. Binder, E. Distance Coding. 2000. Available online: https://groups.google.com/g/comp.compression/c/96DHNJgf0NM/m/Ep15oLxq1CcJ (accessed on 14 November 2023).
17. Albers, S. Improved randomized on-line algorithms for the list update problem. *SIAM J. Comput.* **1998**, *27*, 682–693. [CrossRef]
18. Burrows, M.; Wheeler, D.J. *A Block-Sorting Lossless Data Compression Algorithm*; Technical Report No. 124; Digital Systems Research Center: Palo Alto, CA, USA, 1994.
19. Abel. J. Post BWT stages of the Burrows-Wheeler compression Algorithm. *Softw. Pract. Exper.* **2010**, *40*, 751–777. [CrossRef]
20. Dorrigiv, R.; López-Ortiz, A.; Munro, J.I. An Application of Self-organizing Data Structures to Compression. In *Experimental Algorithms, Proceedings of the 8th International Symposium on Experimental Algorithms, SEA 2009, Dortmund, Germany, 3–6 June 2009*; Vahrenhold, J., Ed.; Lecture Notes in Computer Science 5526; Springer: Berlin, Germany, 2009; pp. 137–148.
21. Žalik, B.; Lukač, N. Chain code lossless compression using Move-To-Front transform and adaptive Run-Length Encoding. *Signal Process. Image Commun.* **2014**, *29*, 96–106. [CrossRef]
22. Arnavut, Z. Move-To-Front and Inversion Coding. In *Proceedings of the IEEE Data Compression Conference, DCC'2000, Snowbird, UT, USA, 28–30 March 2000*; Cohn, M., Storer, J.A., Eds.; IEEE Computer Society Press: Los Alamitos, CA, USA, 2000; pp. 193–202.
23. Adjeroh, D.; Bell, T.; Mukherjee, A. *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*, 2nd ed.; Springer Science + Business Media: New York, NY, USA, 2008.
24. Lee, Y.L.; Han, K.H.; Sullivan, G.J. Improved lossless intra coding for H. 264/MPEG-4 AVC. *IEEE Trans. Image Process.* **2006**, *15*, 2610–2615.
[PubMed]
25. Khademi, A.; Krishnan, S. Comparison of JPEG 2000 and other lossless compression schemes for digital mammograms. *IEEE Trans. Image Process.* **2005**, *25*, 693–695.
26. Barina, D. Comparison of Lossless Image Formats. *arXiv* **2021**, arXiv:2108.02557
27. Ulacha, G.; Łazoryszczak, M. Lossless Image Coding Using Non-MMSE Algorithms to Calculate Linear Prediction Coefficients. *Entropy* **2023**, *25*, 156. [CrossRef] [PubMed]
28. Kohek, Š.; Strnad, D.; Žalik, B.; Kolmanič, S. Interactive synthesis and visualization of self-organizing trees for large-scale forest succession simulation. *Multimed. Syst.* **2019** *25*, 213–227. [CrossRef]
29. Nong, G.; Zhang, S.; Chan, W.H. Two efficient algorithms for linear time suffix array construction. *IEEE Trans. Comput.* **2011**, *60*, 1471–1484. [CrossRef]
30. Kärkkäinen, J.; Sanders, P.; Burkhardt, S. Linear work suffix array construction. *J. ACM* **2017** *53*, 918–936. [CrossRef]
31. Bader, M. *Space-Filling Curves—An Introduction with Applications in Scientific Computing*; Springer: Berlin, Germany, 2013.
32. Chung, K.-L.; Huang, Y.-L.; Liu, Y.-W. Efficient algorithms for coding Hilbert curve of arbitrary-sized image and application to window query. *Inf. Sci.* **2007** *17*, 2130–2151. [CrossRef]
33. Žalik, B.; Mongus, D.; Rizman Žalik, K.; Lukač, N. Boolean Operations on Rasterized Shapes Represented by Chain Codes Using Space Filling Curves. *J. Vis. Commun. Image Represent.* **2017** *49*, 420–430. [CrossRef]
34. Lawder, J.K.; King, P.J.H. Using state diagrams for Hilbert curve mappings. *Int. J. Comput. Math.* **2001** *78* 327–342. [CrossRef]
35. Žalik, B.; Strnad, D.; Kohek, Š.; Kolingerová, I.; Nerat, A.; Lukač, N.; Lipuš, B.; Žalik, M.; Podgorelec, D. FLoCIC: A Few Lines of Code for Raster Image Compression. *Entropy* **2023**, *25*, 533. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.