Feature

Interleaving for Combating Bursts of Errors

Yun Q. Shi, Xi Min Zhang, Zhi-Cheng Ni, and Nirwan Ansari

© EYEWIRE

Abstract

To ensure data fidelity, a number of random error correction codes (ECCs) have been developed. ECC is, however, not efficient in combating bursts of errors, i.e., a group of consecutive (in one-dimensional (1-D) case) or connected (in two- and three- dimensional (2-D and 3-D) case) erroneous code symbols owing to the bursty nature of errors. Interleaving is a process to rearrange code symbols so as to *spread* bursts of errors over multiple code-words that can be corrected by ECCs. By converting bursts of errors into random-like errors, interleaving thus becomes an effective means to combat error bursts. In this article, we first illustrate the philosophy of interleaving by introducing a 1-D block interleaving technique. Then multi-dimensional (M-D) bursts of errors and optimality of interleaving are defined. The fundamentals and algorithms of the state of the art of M-D interleaving—the *t*-interleaved array approach by Blaum, Bruck and Vardy and the successive packing approach by Shi and Zhang—are presented and analyzed. In essence, a *t*-interleaved array is constructed by closely tiling a building block, which is solely determined by the burst size *t*. Therefore, the algorithm needs to be implemented each time for a different burst size in order to maintain either the error burst correction capability or optimality. Since the size of error bursts is usually not known in advance, the application of the technique is somewhat limited. The successive packing algorithm, based on the concept of 2×2 basis array, only needs to be implemented once for a given square 2-D array, and yet it remains optimal for a set of bursts of errors having different sizes. The performance comparison between different approaches is made. Future research on the successive packing approach is discussed. Finally, applications of 2-D/3-D successive packing interleaving in enhancing the robustness of image/video data hiding are presented as examples of practical utilization of interleaving.

Introduction

In data manipulation and transmission, errors may be caused by a variety of factors including noise corruption, limited channel bandwidth, and interference between channels and sources. It is well known that many error correction codes (ECCs) have been developed to correct errors in order to ensure data fidelity. In doing so, redundancy has been added to ECC, resulting in what is known as random error correction codes. There are basically two different types of ECCs. One is known as block codes, the other as convolutional codes. The frequently used block codes are often denoted by a pair of two integers, i.e., (n, k), and one block code is completely defined by 2^k binary sequences, each is an n-tuple of bits, known as code-worde. Note that for simplicity only binary code symbols are considered in this article. Specifically, consider the commonly used BCH codes, which are one kind of block codes, named after the three inventors Bose, Chadhuri, and Hocquenghem [1, 2]. The notation BCH (31, 6) indicates that there are at most 2^6 distinct messages, each represented by six bits and encoded by a code-worde consisting of 31 bits in this BCH code. Instead of six bits, 31 bits are used to represent an input symbol, implying an added redundancy. According to channel coding theory, the minimum Hamming distance between any two different code-words in the BCH (31, 6) code is 15, and the error correction capability of the code is seven. In other words, a 31-bit code-worde in the BCH (31, 6) code can be correctly decoded as long as there are no more than seven error bits regardless of the bit error positions within the code-worde. That is why random ECC refers to its ability to correct *random* bit errors within a code-worde. While error positions can be random the number of error bits within one code-worde that can be corrected, referred to as the random error correction capability of the code, is critical.

Bursts (or clusters) of errors are defined as a group of *consecutive* error bits in the one-dimensional (1-D) case or *connected* error bits in multi-dimensional (M-D) cases. In this sense, we can see that the channel has memory. One example can be several consecutive transmitted error bits in a mobile communication system caused by a multipath fading channel. Another example can be an area formed by many connected error bits in a two-dimensional (2-D) barcode. In [3], a bursty channel is defined as a channel over which errors tend to occur in bunches, or "bursts," as opposed to random patterns associated with a Bernoulli-distributed process. Therefore, a random error correction code, when applied, may not be powerful enough to correct the bursts of errors and at the same time it may be a waste for other occasions (1-D case) when there are no bursts of errors, or regions (M-D case) where there are no bursts of errors. For instance, consider a case in which there is one burst of errors consisting of 60 consecutive bits. Obviously, the BCH (31, 6) code is not be able to correct this burst of errors. One may think of using a more powerful BCH code to combat this burst of errors. For example, BCH (255, 9) seems to be a suitable candidate since it can correct 63 errors in a 255-bit code-worde. Indeed, this error burst consisting of 60 consecutive error bits can be corrected by this powerful BCH code. However, for a vast majority of time, the error correction capability at the expense of high redundancy (each code-worde now consists of 255 bits) has been wasted. This example demonstrates that using random ECC to combat bursts of errors is not efficient.

Although some codes, including Fire codes [4, 3], suitable for correcting bursts of errors have been developed, they are not efficient for random error correction. In most practical systems, unfortunately, both types of errors may exist. By far, interleaving before applying random ECCs is a most frequently used and efficient way to combat bursts of errors and random errors. In this article, the philosophy of interleaving is first illustrated through a 1-D block interleaver. Then M-D bursts of errors and optimality of interleaving are defined. Afterwards, the state of the art of M-D interleaving techniques is presented. In particular, the first comprehensive M-D interleaving technique, i.e., the t-interleaved array approach developed by Blaum et al. [5] and the successive packing (SP) approach developed by Shi and Zhang [6] are introduced and analyzed. In essence, a t-interleaved array is constructed by packing a building block, which is solely determined by the burst size t. In the 2-D case, for a given burst size, t_0, a specific algorithm, which can correct arbitrarily-shaped error burst of size t_0 and is optimal in the sense of the interleaving degree, is implemented. It is observed that when the burst size, t, increases, i.e., $t > t_0$, the algorithm with a set of new parameters has to be implemented once again in order to correct larger arbitrarily-shaped error bursts. When the burst size decreases, i.e., $t < t_0$, the interleaved array obtained with respect to t_0 is not optimal any more. Since the size of error bursts is not known in advance (this is usually the case in reality), the application of the technique is somewhat limited. In practice, the size of a 2-D array, say, a digital image is normally given and the burst size is often unknown *a priori*, the successive packing algorithm, based on the concept of the 2×2 basis array, only needs to be implemented once for a given square 2-D array and yet it remains optimal for

Yun Q. Shi (shi@njit.edu), Xi Min Zhang, Zhi-Cheng Ni, and Nirwan Ansari are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA.

a set of bursts of errors having different sizes. The performance of these two interleaving techniques is compared and comments are made. Future research on the successive packing approach is discussed. Finally, applying 2-D/3-D SP interleaving techniques to enhance the robustness of image/video data hiding is presented as practical examples of interleaving techniques.

Philosophy of Interleaving

The 1-D interleaving techniques have been well documented in error correction coding texts, e.g., in [3]. The main idea is to mix up the code symbols from different code-words so that when the code-words are reconstructed at the receiving end error bursts encountered in the transmission are spread across multiple code-words. Consequently, the errors occurred within one code-worde may be small enough to be corrected by using a simple random error correction code. This can be seen clearly from a simple example. Consider a code in which each code-worde contains four code symbols. Furthermore, the code possesses the random error correction capability. Without loss of generality, we assume the one-random-error-correction capability, i.e., any single code symbol error occurred in one code-worde can be corrected. Suppose there are 16 symbols, which correspond to four code-words. That is, code symbols from 1 to 4 form a code-worde, from 5 to 8 another code-worde, and so on. One of the 1-D interleaving procedures, known as block interleaving, first creates a 4×4 2-D array, called block interleaver as shown in Figure 1. The 16 code symbols are read into the 2-D array in a column-by-column (or row-by-row) manner. The interleaved code symbols are obtained by writing the code symbols out of the 2-D array in a row-by-row (or column-by-column) fashion. This process has been depicted in Figure 1 (a), (b), and (c). Now take a look at how this interleaving technique can correct error bursts. Assume a burst of errors involving four consecutive symbols as shown in Figure 1 (c) with shades. After de-interleaving as shown in Figure 1 (d), the error burst is effectively spread among four code-words, resulting in only one code symbol in error for each of the four code-words. With the one-random-error-correction capability, it is obvious that no decoding error will result from the presence of such an error burst. This simple example demonstrates the effectiveness of 1-D interleaving technique in combating 1-D bursts of errors, i.e., how the interleaving spreads code symbols over multiple code-words so as to convert a burst of errors occurred with the interleaved array into random-like errors in the de-interleaved array. In other words, the pair of interleaving and de-interleaving can equivalently convert a bursty channel into a random-like channel. Consequently, random error correction codes can be used efficiently to correct bursts of errors.

2-D and M-D Bursts of Errors

Scenarios, where M-D error bursts may occur, include magnetic and optical (say, holographic) datastorage, charge-coupled devices (CCDs), 2-D barcodes, and information hiding in digital images and video sequences. In particular, it is worth mentioning that in the holographic recording a laser beam illuminates a programmable spatial light modulator thereby generating an object beam, which represents a 2-D page of data. An entire page of data can be retrieved all at once, thus achieving a very high data rate. Therefore, the reliability issue of M-D information has arisen as an important task, having both theoretical and practical significance.

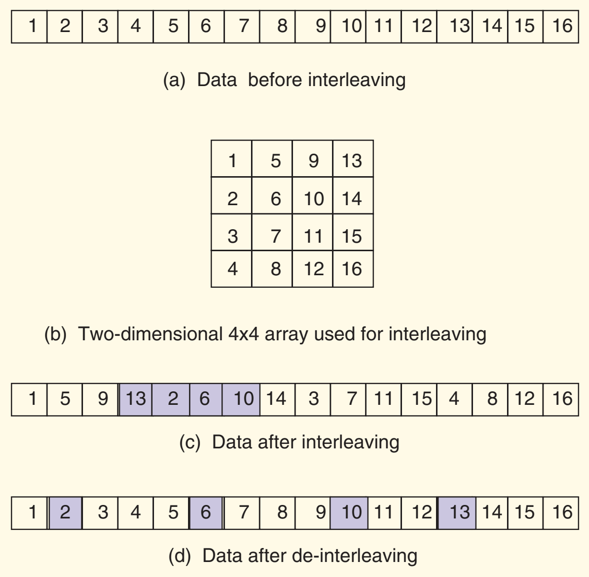


Figure 1. 1-D block interleaving and its performance.

Instead of defining a burst of errors as a rectangular area or a circular area, Blaum et al. defined a 2-D burst of errors as an arbitrarily-shaped, connected area [5]. Consider Figure 2, where all the code symbols (assigned to the elements of the 2-D array) marked with triangles form a 2-D error burst. Note that all of these symbols are connected to each other and the connectivity here is constrained to the horizontal and vertical directions, referred to as 4-connection. This definition can be generalized to the M-D case. The size of a burst is defined as the total number of code symbols contained in the burst. Hence, the size of the error burst in Figure 2 is 10.

1-D Interleaving: Not Effective for Combating 2-D Error Bursts

To enhance the reliability of M-D digital data, which is of crucial importance in the information age, codes that can correct M-D bursts of errors are desired. Here, we show that it is necessary to develop M-D interleaving techniques by demonstrating via an example that 1-D interleaving-based techniques are not effective in correcting 2-D error bursts.

It is known that 2-D barcodes are information storage media in which the source information is stored as a bit stream on a printed label [7]. Examples of 2-D barcode applications include bracelets that are used in hospitals to carry patients' entire medical histories; labels put on parts that are used in automotive assembling processes to carry a unique identification number and other pertinent information applicable to production, tracking, and statistical process control [8]; and the labels that are used as portable data files that accompany packages in shipping industry [9]. When the United Parcel Service (UPS) developed its own 2-D bar codes, 1-D interleaving technique was used to combat 2-D bursts. That is a sequence of code symbols is first 1-D interleaved. The 1-D interleaved symbols are then written into a 2-D array (printed on a 2-D label) according to some writing pattern. Specifically, there are two different patterns used by the UPS: the Boustrophedonic Pattern and the Spiral Data Pattern, [9], shown in Figure 3 (b) and (e), respectively.

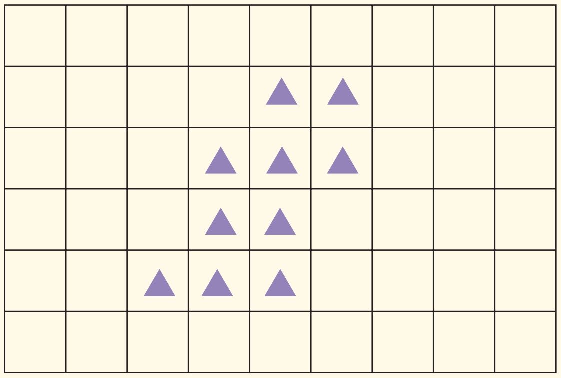


Figure 2. A 2-D burst of errors of size 10.

Now, consider again the scenario of the 16 code symbols discussed earlier. After the 16 symbols have been 1-D interleaved (refer to Figure 1 (c)), they are written into a 4×4 2-D array according to either the Boustrophedonic Pattern or the Spiral Data Pattern. The 4×4 2-D array obtained by applying these two writing patterns are shown in Figure 3 (a) and (d), respectively. Let us examine the performance of these two procedures by checking if they can combat a 2×2 2-D error burst, shown in Figure 3 (a) with shades. Figure 3 (c) indicates that this 2×2 error burst has not been spread effectively so that there are two code symbols in error in the second code-worde, indicating that the error burst cannot be corrected by using the code of one-random-error-correction capability. That is, the 1-D interleaving technique plus the Boustrophedonic Pattern writing procedure cannot combat the 2×2 burst of errors. The same observation may be obtained for the Spiral Data Pattern writing procedure, as shown in Figure 3 (d) and (f).

Summing up, combining the 1-D interleaving technique and some writing procedure to constitute a 2-D interleaved array may not be able to combat 2-D burst errors effectively. This does not come as a surprise because the 2-D nature has not been taken into account with the 1-D procedure. Therefore, it is necessary to develop efficient M-D interleaving techniques to secure the reliability of M-D digital data.

M-D Interleaving I: *t*-Interleaved Array

As mentioned above, while some 2-D error burst correction codes have been developed [4, 10], an M-D interleaving technique followed by a simple random error correction code has become the most common approach to correcting M-D error bursts. Some M-D interleaving techniques for combating M-D bursts of errors have been proposed [12, 11, 13, 98]. Some theoretical aspects of the task, in terms of the definitions of 2-D and M-D bursts, the optimality of interleaving, the existence of the optimal interleaving and

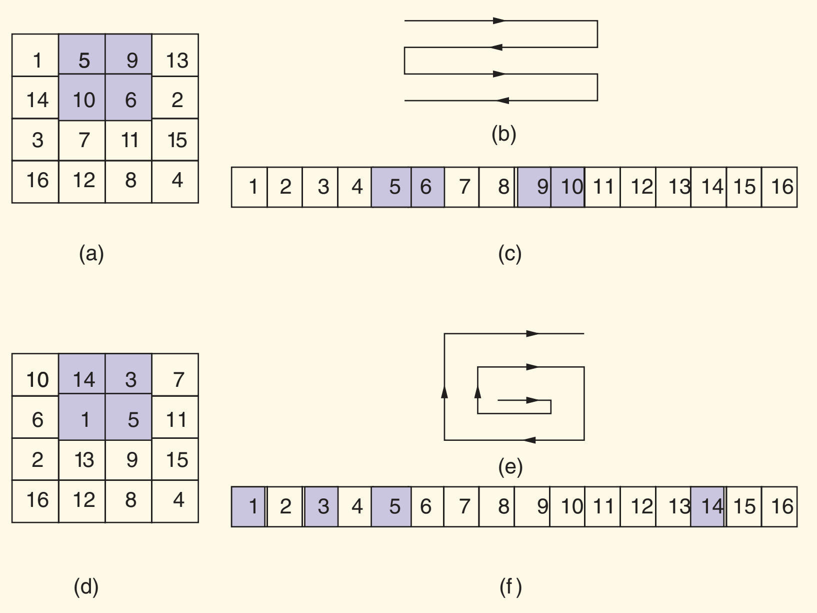


Figure 3. Boustrophedonic pattern and spiral data pattern.

so on have been studied in [adbelghaffar 88, 5]. It has been noticed that the organization of raster-graphics memory encounters the same problems as that faced by the interleaving technique [14]. Taking into account [12] (dealing with an error burst of circular-shape) is rather brief, and that [5] is the first comprehensive presentation of the M-D interleaving scheme, we will elaborate on the t-interleaved array technique [5] in this section.

8	1	2	3	4	5	6	7
3	4	5	6	7	8	1	2
6	7	8	1	2	3	4	5
1	2	3	4	5	6	7	8
4	5	6	7	8	1	2	3
7	8	1	2	3	4	5	6
2	3	4	5	6	7	8	1
5	6	7	8	1	2	3	4

(a)

1	2	3	4	5	6	7	8
5	6	7	8	9	10	1	2
9	10	1	2	3	4	5	6
2	3	4	5	6	7	8	9
5	6	7	10	1	2	3	4
8	1	2	9	4	5	10	7
3	10	5	6	7	8	1	2
6	7	8	1	2	3	4	9

(b)

Figure 4. Two 4-interleaved arrays with different interleaving degrees. (a) Interleaving degree is 8. (b) Interleaving degree is 10.

t-Interleaved Array, Interleaving Degree and Optimality

Blaum et al. [5] introduced the concept of the *t*-interleaved array. Consider two 4-interleaved arrays shown in Figure 4. By a 4-interleaved array, it is meant that no matter how we choose four 4-connected elements in the array we always have these four elements marked with distinct numbers. Assuming that in Figure 4 all elements in the 2-D array denoted by the same number form one code-word, we can then conclude that whenever a burst of errors of size four takes place within the 4-interleaved 2-D array each code-worde will encounter at most one error. If further assuming that the code has the one-random-error-correction capability, we can see that the error burst can be corrected. Through this discussion, it is observed that a *t*-interleaved array together with a random error correction code having the one-random-error-correction capability can combat an error burst of size four. Without loss of generality, only one error burst is discussed in this article.

A close look at Figure 4 (a) and (b) reveals that there are a total of eight and 10 distinct numbers, i.e., eight and 10 distinct code-words in (a) and (b), respectively. The total number of distinct code-words is referred to as the *interleaving degree*. The optimality of interleaving is achieved if the interleaving degree reaches its lower bound. The lower bound for correcting arbitrarily-shaped error bursts has been proved to be: $t^2/2$ if t is even and $(t^2+1)/2$ if t is odd [5]. Thus, for a 4-interleaved array, the lower bound of the interleaving degree is equal to eight. Therefore the code depicted in Figure 4 (a) is optimal, while that in Figure 4 (b) is not. Blaum et al. have shown that optimality can be guaranteed for the 1-D and 2-D case; however, this is not always true for the 3-D case [5, 15].

Basic Ideas and Algorithms

Based on the concept of the *t*-interleaved array, Blaum et al. proposed their algorithms for 2-D and 3-D interleaving. Here, we present only one of their two optimal 2-D interleaving algorithms. Assume that A is a 2-D array and m is a positive integer. One labels the coordinates of the array

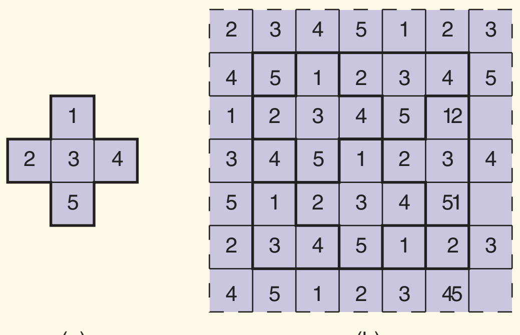

(a) (b)

Figure 5. 3-interleaved array and its fundamental building block, Lee sphere of radius 1. (a) Lee sphere of radius 1. (b) 3-interleaved array.

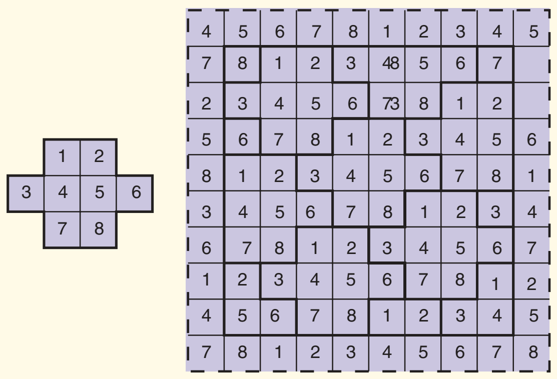

(a) (b)

Figure 6. 4-interleaved array and its fundamental building block (4-sphere). (a) 4-sphere. (b) 4-interleaved array.

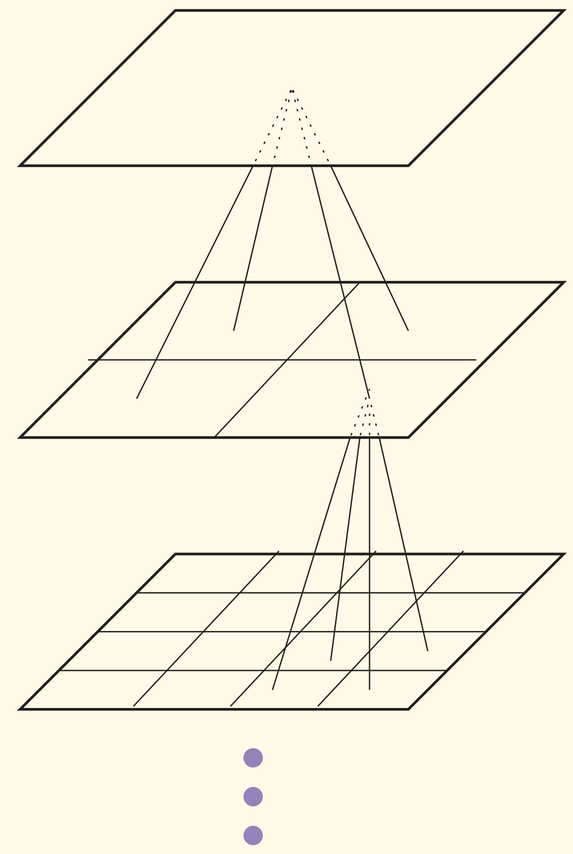


Figure 7. 2-D successive doubling.

toroidally on m, i.e., assign the same number to elements (x, y) and (x', y') of A if $x' \equiv x$ and $y' \equiv y$ modulo m. Let b be a fixed positive integer. Then for each $a = 0, 1, \cdots, m-1$, the elements $(x, a+xb)$, with both coordinates taken modulo m, are labeled by the integer a. For odd t, $b = t$ and $m = (t^2+1)/2$; for even t, $b = t+1$ and $m = t^2/2$. It can be verified that the 4-interleaved array shown in Figure 4(a) can be constructed using the above algorithm with $b = 5$, and $m = 8$. It is observed that the algorithm is implemented differently according to the burst size t as well as whether it is odd or even.

The key idea of the Blaum et al. approach is based on Lee-spheres and close tiling. Linking Lee spheres with odd burst sizes and creating some spheres for even burst sizes, Blaum et al. used these spheres as fundamental building blocks to construct interleaved arrays. A Lee sphere of radius 1 is shown in Figure 5 (a), where one can see that the maximum *4-distance* (only considered in either horizontal or vertical directions) from the central element to any other elements in the sphere is equal to one. This Lee sphere can be used as a fundamental building block to construct (closely tile) a 3-interleaved array as shown in Figure 5 (b). Note that, in Figure 5 (b), there is a square array of 5×5 enclosed by solid lines and formed by several whole and partial Lee spheres of radius 1 (also bounded by solid lines). The fundamental building block used to closely tile a 4-interleaved array is shown in Figure 6 (a), while the constructed 4-interleaved array is shown in Figure 6 (b). Similarly, there is an 8×8 square array bounded by solid lines and formed by several whole and partial fundamental building blocks (enclosed by soled lines). It is further noted that the 3- and 4-interleaved arrays shown in Figures 5 (b) and 6 (b), respectively, can in turn be used as building blocks to closely tile 3- and 4-interleaved array of a larger size. This can be verified by the observing that the five central elements in the first row in Figure 5 (b): 3, 4, 5, 1, 2 with dashed lines repeat themselves in the last row within the solid line square, and so do the eight central elements in the first row of Figure 6 (b): 5, 6, 7, 8, 1, 2, 3, 4. The same is true for the five and eight elements, respectively, in the bottom row, the left-most column, and the right-most column of Figure 5 (b) and Figure 6 (b). By close tiling, it is meant that translated building blocks are used to construct a larger block, which is the union of translated building blocks and there is no overlapping among the building blocks translated in the process. (For more information on these two concepts, i.e., the Lee sphere and close tiling, interested readers may refer to [15].) Blaum et al. have shown that if one labels each element in the fundamental building block with a distinct number and uses the building block to closely (meaning no uncovered elements) tile (meaning no overlapping between blocks) a large enough 2-D area, then one can produce an interleaved array. In this interleaved array, each element in any arbitrarily-shaped, connected subset consisting of t elements is labeled with a distinct number. All numbers of the same kind form a code-worde. Consequently, the error burst of size t can be corrected by one-random-error-correction codes.

Comments and Discussions

Though it can effectively spread arbitrarily-shaped 2-D burst errors of size t, the above characterization of the technique [5] does reveal some of its limitations. Firstly, the technique is based on the size of a burst of errors, t. For combating bursts of errors of size t equal to a specific t_0, one needs to implement an algorithm with a set of parameters to construct an interleaving code. When the size t increases, i.e., $t > t_0$, one needs to implement an algorithm with a new set of parameters to construct another interleaving code. That is, the interleaved array constructed for a specific t_0 may not be able to correct a burst of errors of size t as $t > t_0$. Since in reality, e.g., in an application of 2-D bar-codes, *the size of error bursts may not be known exactly a priori*, the implementation of the technique may become cumbersome and ineffective.

Secondly, when the actual size of a burst, t, is less than t_0, with which the interleaving algorithm is applied, the technique is no longer optimal. This can be justified as follows. In [5], the optimality means that the interleaving degree reaches its lower bound. As mentioned before, in the 2-D case the interleaving degree, associated with an interleaving scheme designed for some burst size t in [5], is guaranteed to reach its lower bound. Furthermore it is known that the lower bound of the interleaving degree is a monotonically increasing function of the burst size t. Specifically, the lower bound is $t^2/2$ for even t and $(t^2+1)/2$ for odd t. Therefore, with respect to the implementation of the interleaving scheme designed for a burst size t_0, when the actual size of an error burst, t, is smaller than t_0, the achieved interleaving degree with t_0 is larger than the lower bound that corresponds to t. That is, the interleaving scheme designed for a burst size t_0 is not optimal for a smaller bust size, t.

In many applications, *the size of a given 2-D or M-D array is known*. For instance, a digital (watermarked) image may be known to have a size of 512 by 512 pixels. Under the circumstances, one may wonder if it is possible to develop a 2-D interleaving technique, which is optimal for all (if possible) or (at least) for many of the possible error burst sizes. Therefore, it can be implemented only once for a given 2-D array. Motivated by these observations, a novel 2-D interleaving technique, called successive packing approach, has been proposed [6].

M-D Interleaving II: Successive Packing

Given that digital images, video frames, charge-coupled devices (CCDs), and 2-D bar-codes are all in the form of 2-D arrays, without loss of generality, square arrays of $2^n \times 2^n$ are considered here. The utilization of $2^n \times 2^n$ arrays will be further justified later.

2-D Code-words and 1-D Sequence of Code Symbols

In general, the code-words in the 2-D case are of 2-D in nature. 1-D code-words, either row-type, or column-type, or other-type, can be considered as special cases of 2-D code-words. The successive packing technique is able to handle 2-D code-words because all the code symbols in the 2-D code-words are first linked into a 1-D sequence of code symbols. Without loss of generality, the quartering indexing scheme is described below for illustrative purposes. That is, a square array of $2^n \times 2^n$ is viewed as consisting of four quadrants, each quadrant itself consisting of its own four quadrants; the process repeats itself until it reaches a level where all four quadrants are of 2×2. This is referred to as 2-D successive doubling, as shown in Figure 7. These 2×2 arrays are the fundamental structure. When the quartering indexing scheme is applied, each code symbol, assigned to an element of the array has a pair of subscripts. The first subscript represents the index of the 2×2 array in which the code symbol is located, while the second subscript indicates the index of the code symbol within the 2×2 array. To convert the quartering index, $s_{i,j}$ into the 1-D index, s_k, we apply the following operation: $k = 4i + j$.

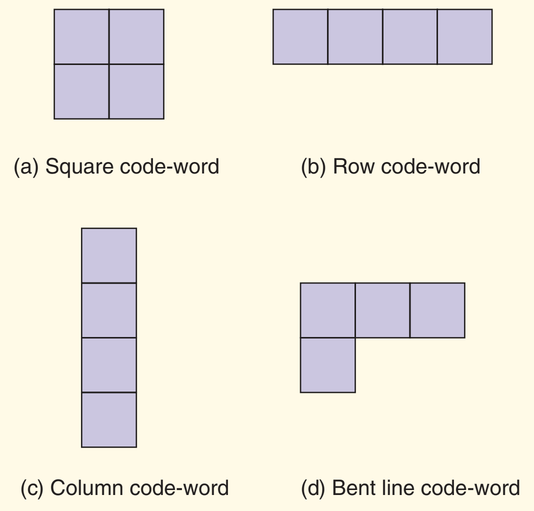


Figure 8. Four different types of 2-D code-words having four code symbols.

Quartering indexing is not the only choice for the proposed interleaving technique. Actually, code-words can be of any shape. Several shapes of a code-worde consisting of four code symbols are shown in Figure 8. Obviously, for any given shape of 2-D code-words, it is always possible to label the code symbols into a 1-D sequence with a possibly more complicated bookkeeping scheme.

The Successive Packing Algorithm

Now we present the successive packing interleaving technique in the 2-D case in a general and compact way, that allows straightforward generalization to the M-D case.

The 2-D interleaving using the successive packing proceeds as follows. Consider a 2-D array of $2^n \times 2^n$ for 2-D interleaving.

When $n = 0$, i.e., an array of 1×1 is considered for interleaving, the interleaved array is the original array itself. That is,

$$S_1 = [s_0] \tag{1}$$

where s_0 represents the element in the array, and S_1 the array. Note that the subscript in the notation S_1 represents the total number of elements in the interleaved array. Hence, when $n = 1$, i.e., for a 2×2 array, the interleaved array is denoted by S_4; when $n = 2$, the interleaved array is S_{16}. In general, for a given n, the interleaved array is denoted by $S_{2^{2n}}$.

The procedure is carried out successively. Given an interleaved array S_i, the interleaved array of S_{4i} can be generated according to

$$S_{4i} = \begin{bmatrix} 4 \times S_i + 0 & 4 \times S_i + 2 \\ 4 \times S_i + 3 & 4 \times S_i + 1 \end{bmatrix} \qquad (2)$$

where the notation of $4 \times S_i + k$ with k = 0, 1, 2, 3 represents a 2-D array that is generated from S_i. This indicates that $4\times S_i + k$ has the same dimensionality as S_i. Furthermore, each element in $4\times S_i + k$ is indexed in such a way that its subscript equals to four times of that of the corresponding element in S_i plus *k*. By the corresponding element, we mean the element occupying the same position in the 2-D array. It appears that S_{4i} is derived from S_i by *packing* S_i four times. This explains why the term successive packing is used.

According to the above rule, we have

$$S_4 = \begin{bmatrix} 4 \times S_1 + 0 & 4 \times S_1 + 2 \\ 4 \times S_1 + 3 & 4 \times S_1 + 1 \end{bmatrix} = \begin{bmatrix} s_0 & s_2 \\ s_3 & s_1 \end{bmatrix}. \qquad (3)$$

Similarly, we have S_{16} as follows.

$$\begin{aligned} S_{16} &= \begin{bmatrix} 4 \times S_4 + 0 & 4 \times S_4 + 2 \\ 4 \times S_4 + 3 & 4 \times S_4 + 1 \end{bmatrix} \\ &= \begin{bmatrix} s_0 & s_8 & s_2 & s_{10} \\ s_{12} & s_4 & s_{14} & s_6 \\ s_3 & s_{11} & s_1 & s_9 \\ s_{15} & s_7 & s_{13} & s_5 \end{bmatrix}. \end{aligned} \qquad (4)$$

The resemblance between the successive packing interleaving and the fast Fourier transform (FFT) is observed. Firstly, the successive doubling mentioned before is also used in FFT. Secondly, after the successive doubling, what is left here is a 2×2 basis array, which is expressed in (3) and depicted in Figure 9. This 2×2 basis array is the counterpart of the basic butterfly computation structure used in FFT. Thirdly, both techniques work on a group of data whose dimensionality is an integer power of two to facilitate utilization of digital computers.

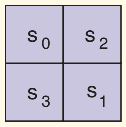


Figure 9. 2 × 2 basic array.

Main Results

It has been proved in [6] that in a 2-D interleaved array of $2^n\times 2^n$, A, generated with the successive packing technique, any square error burst of $2^k\times 2^k$ with $1 \le k \le n-1$ and any rectangular error burst of $2^k\times 2^{k+1}$ or $2^{k+1}\times 2^k$ with $0 \le k \le n-1$ can be spread so that each element in the burst falls into a distinct block in the de-interleaved array, where the block size, *K*, is 2^{2n-2k} for the burst of $2^k\times 2^k$, and $2^{2n-2k-1}$ for the burst of $2^k\times 2^{k+1}$ or $2^{k+1}\times 2^k$. This indicates that, if a distinct code symbol is assigned to each element in a block and all the code symbols associated with the block form a distinct code-worde, then this technique guarantees that the error burst can be corrected with a one-random-error-correction code, provided the code is available. (Note that a code capable of correcting one code symbol error within a code-worde of two code symbols does not exist in reality. Therefore, though the error burst of $2^{n-1}\times 2^n$ or $2^n\times 2^{n-1}$ can be effectively spread in the de-interleaved arrays as described above, they in fact cannot be corrected with a one-random-error-correction code.) Furthermore, the interleaving degree equals to the size of the burst error, hence minimizing the number of code-words required for an interleaving scheme. In other words, the interleaving degree obtained by the successive packing interleaving is indeed the lower bound [6]. In this sense, the successive packing interleaving technique is optimal. If a coding technique has a strong random-error-correction capability, say, it can correct one error in every code-worde of size eight, then any error burst of $2^{n-1}\times 2^{n-2}$ or $2^{n-2}\times 2^{n-1}$ can be corrected. If a code, on the other hand, has a weaker random-error-correction capability, say, it can only correct one random error within a code-worde of size 64, then only smaller error bursts, i.e., any burst of $2^{n-3}\times 2^{n-3}$ in the interleaved array can be corrected by the successive interleaving.

Performance Comparison

In this subsection, the performance of the three techniques—the UPS' technique, the *t*-interleaved array interleaving technique [5], and the SP interleaving technique [6]—is compared by means of an example, followed by some general comments.

An Example

Figure 10 shows the interleaved 2-D arrays of 8×8 obtained by applying the three techniques and the de-interleaved array, from which the following observations can be made.

Firstly, look at an error burst of 2×2 located in the center of the interleaved arrays. It involves the following four elements: s_{20}, s_{62}, s_{43}, s_1 in the interleaved array using SP; s_{35}, s_{27}, s_{28}, s_{36}, using UPS'; and s_{35}, s_{43}, s_{60}, s_4 using the *t*-interleaved array method. After de-interleaving, SP spreads the four elements in the error burst into four different quadrants, while either the UPS' method or the t-interleaved array method has two error elements located within one quadrant. This implies that the SP interleaving can spread error burst farther away in 2-D sense, and if a code can only correct one random error in each code-worde consisting of 16 code symbols, then only SP can correct the $2{\times}2$ error burst. In other words, only SP can correct the $2{\times}2$ error burst with four code-words, while the other two methods require more than four code-words.

Secondly, it is noticed that, with respect to the interleaved 2-D array of $8{\times}8$, SP can optimally spread and correct with an one-random-error-correction code the following types of bursts: any burst of $2{\times}4$, $4{\times}2$, or $2{\times}2$, according to the Main Results presented above; while the t-interleaved array method can spread and correct with a one-random-error-correction code arbitrarily-shaped error bursts of size four with an interleaving degree of eight (refer to [5]). Since any error burst of size four in an array of $8{\times}8$ can be entirely included in either a burst of $2{\times}2$, or a burst of $2{\times}4$, or a burst of $4{\times}2$, it is clear that SP can also spread and correct with a one-random-error-correction code any arbitrarily-shaped error burst of size four in the interleaved 2-D array of $8{\times}8$. As far as the efficiency is concerned, if a burst of size four is of $2{\times}2$, then as shown above, SP reaches an interleaving degree of four, and hence is more efficient. If, more general, a burst of size four is included in a burst of $2{\times}4$ or $4{\times}2$, then the interleaving degree is 8, indicating that in this case SP achieves the same efficiency as the technique [5] does.

Thirdly, in addition to the three types of bursts described above, SP can also optimally spread and correct with a one-random-error-correction code any bursts of $1{\times}2$, $2{\times}1$, or $4{\times}4$ within the interleaved array of $8{\times}8$, according to the Main Results presented earlier. Thus SP is versatile and effective.

Comment 1

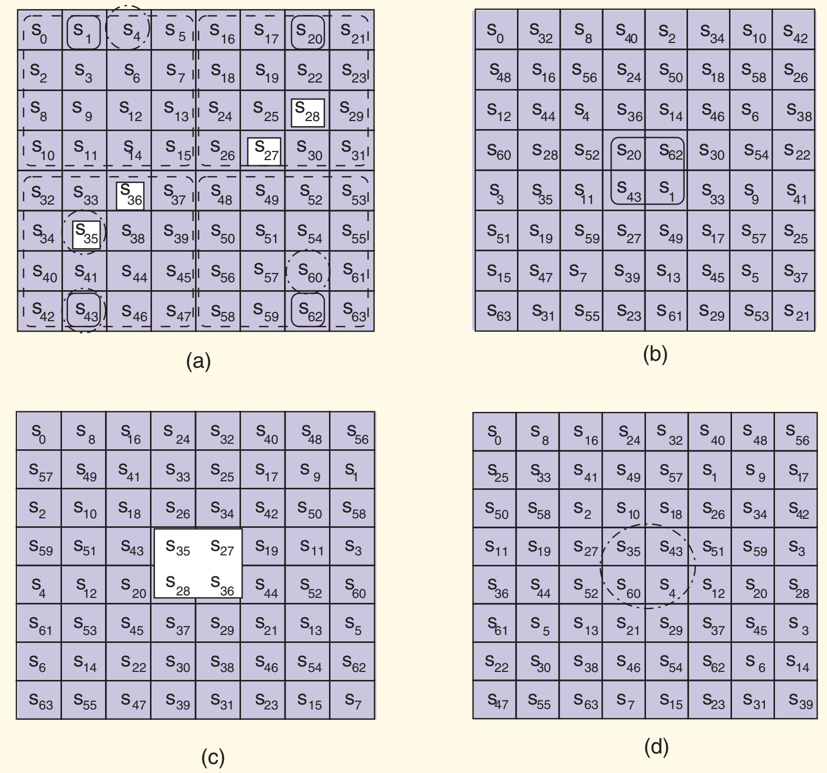
Both examples presented in Introduction (Figure 3) and above demonstrate that the UPS' approach (using 1-D interleaving plus a special writing procedure to form a 2-D interleaved array) does not work well in combating 2-D error bursts.

Comment 2

While the SP technique works for 2-D arrays of $2^n \wedge 2^n$, the technique of [5] works for 2-D arrays of $m{\wedge}m$, where $m = t^2/2$ for even t and $m = (t^2+1)/2$ for odd t with t being the size of error size $t = 4$ can be generalized to the case of any error burst of size $t = 2^p$, where p is a positive integer [6]. That is, both the t-interleaved array technique and the successive packing technique can be applied to a square array of $2^{2p-1}{\times}2^{2p-1}$, and both can correct optimally an arbitrarily-shaped error bursts of size $t = 2^p$ (with the same interleaving degree $m = t^2/2 = 2^2p - 1$). It has been noticed that, however, SP can optimally correct other error bursts of different sizes according to the Main Results presented earlier in this article. Therefore, SP is versatile.

Comment 3

Needless to say that there is a certain constraint, yet practical, on using the SP technique. That is, it is assumed that given 2-D arrays are of $2^n \times 2^n$ shape, and the error bursts considered are of $2^k \times 2^k$ shape with $1 \le k \le n-1$, or $2^k \times 2^{k+1}$ or $2^{k+1} \times 2^k$ with $0 \le k \le n-1$. To our best knowledge, 2-D burst error-correction codes (not involving interleaving) [4, 10] address error bursts of rectangular shapes only. Furthermore, as discussed previously, the assumption made with SP is similar to that made in



s_0	s_1	s_4	s_5	s_{16}	s_{17}	s_{20}	s_{21}
s_2	s_3	s_6	s_7	s_{18}	s_{19}	s_{22}	s_{23}
s_8	s_9	s_{12}	s_{13}	s_{24}	s_{25}	s_{28}	s_{29}
s_{10}	s_{11}	s_{14}	s_{15}	s_{26}	s_{27}	s_{30}	s_{31}
s_{32}	s_{33}	s_{36}	s_{37}	s_{48}	s_{49}	s_{52}	s_{53}
s_{34}	s_{35}	s_{38}	s_{39}	s_{50}	s_{51}	s_{54}	s_{55}
s_{40}	s_{41}	s_{44}	s_{45}	s_{56}	s_{57}	s_{60}	s_{61}
s_{42}	s_{43}	s_{46}	s_{47}	s_{58}	s_{59}	s_{62}	s_{63}

(a)

s_0	s_{32}	s_8	s_{40}	s_2	s_{34}	s_{10}	s_{42}
s_{48}	s_{16}	s_{56}	s_{24}	s_{50}	s_{18}	s_{58}	s_{26}
s_{12}	s_{44}	s_4	s_{36}	s_{14}	s_{46}	s_6	s_{38}
s_{60}	s_{28}	s_{52}	s_{20}	s_{62}	s_{30}	s_{54}	s_{22}
s_3	s_{35}	s_{11}	s_{43}	s_1	s_{33}	s_9	s_{41}
s_{51}	s_{19}	s_{59}	s_{27}	s_{49}	s_{17}	s_{57}	s_{25}
s_{15}	s_{47}	s_7	s_{39}	s_{13}	s_{45}	s_5	s_{37}
s_{63}	s_{31}	s_{55}	s_{23}	s_{61}	s_{29}	s_{53}	s_{21}

(b)

s_0	s_8	s_{16}	s_{24}	s_{32}	s_{40}	s_{48}	s_{56}
s_{57}	s_{49}	s_{41}	s_{33}	s_{25}	s_{17}	s_9	s_1
s_2	s_{10}	s_{18}	s_{26}	s_{34}	s_{42}	s_{50}	s_{58}
s_{59}	s_{51}	s_{43}	s_{35}	s_{27}	s_{19}	s_{11}	s_3
s_4	s_{12}	s_{20}	s_{28}	s_{36}	s_{44}	s_{52}	s_{60}
s_{61}	s_{53}	s_{45}	s_{37}	s_{29}	s_{21}	s_{13}	s_5
s_6	s_{14}	s_{22}	s_{30}	s_{38}	s_{46}	s_{54}	s_{62}
s_{63}	s_{55}	s_{47}	s_{39}	s_{31}	s_{23}	s_{15}	s_7

(c)

s_0	s_8	s_{16}	s_{24}	s_{32}	s_{40}	s_{48}	s_{56}
s_{25}	s_{33}	s_{41}	s_{49}	s_{57}	s_1	s_9	s_{17}
s_{50}	s_{58}	s_2	s_{10}	s_{18}	s_{26}	s_{34}	s_{42}
s_{11}	s_{19}	s_{27}	s_{35}	s_{43}	s_{51}	s_{59}	s_3
s_{36}	s_{44}	s_{52}	s_{60}	s_4	s_{12}	s_{20}	s_{28}
s_{61}	s_5	s_{13}	s_{21}	s_{29}	s_{37}	s_{45}	s_3
s_{22}	s_{30}	s_{38}	s_{46}	s_{54}	s_{62}	s_6	s_{14}
s_{47}	s_{55}	s_{63}	s_7	s_{15}	s_{23}	s_{31}	s_{39}

(d)

Figure 10. Performance comparison of three 2-D different interleaving techniques. (a) De-interleaved array. (b) Interleaved array with the SP technique. (c) Interleaved array with the UPS' technique (Boustrophedonic Pattern). (d) Interleaved array with technique of [5, Example 2.3].

Figure 11. The original and marked Lena image. (a) Original Lena image. (b) Marked Lena image.

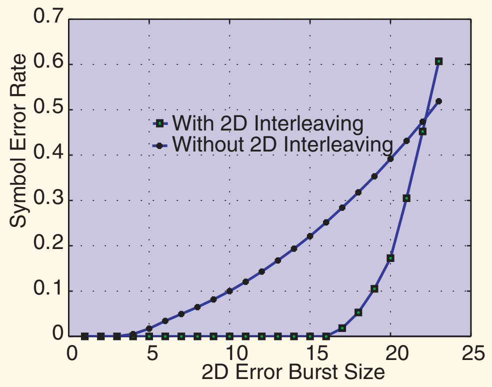


Figure 12. Test results of square error bursts.

the development of the fast Fourier transform technique, i.e., both are based on integer powers of 2 (base-2). Therefore, the assumption is practical and reasonable.

Future Research

How to make SP workable for a 2-D array having dimensionality different from $2^n \times 2^n$, say, $3^n \times 3^n$ (base-3) is a future research subject. It is expected that other basis arrays different from the 2×2 basis array depicted in Figure 9 may be necessary, and combinations of different basis arrays may be able to solve this problem. In both interleaving techniques [5, 6], the ECC with one-random-error-correction capability is assumed. In [16, 17], a more general situation has been examined for the *t*-interleaved array method. Therefore, the investigation in this regard for the successive packing approach may be another subject for future research.

Applications of 2-D/3-D SP Interleaving Techniques to Enhance Robustness of Image/Video Data Hiding

Recently, watermarking and data hiding have received extensive attention. Data hiding is a process to hide some data (representing some useful information) into a cover media imperceptibly. The hidden data may be used for authentication, copyright protection, and data system security. Robustness is one of the basic requirements for imperceptible data hiding in some applications. Error correction codes have been adopted to improve the robustness of watermark signal in [e.g., 18, 19]. As analyzed, however, ECC is only suitable to correct random errors, and will not be efficient for correction of bursts of errors. When cropping or random rows/columns removal takes place in a marked image, often called stego-image, bursts of errors do occur in watermarked still images. Frame loss and 3-D error clusters are typical bursts of errors that can occur to watermarked video sequences. Transmission error may be another source of bursts of errors. When bursts of errors occur, how to extract and detect the hidden data correctly becomes a challenge. While using ECC alone to correct these bursts of errors are not efficient, surprisingly, combating bursts of errors using an interleaving technique, a common tool used in communication systems, has been neither recognized nor addressed so far in the data hiding community.

In this section, we report on recent investigations on applying 2-D/3-D successive packing interleaving techniques to combat bursts of errors occurred in watermarked still images/video sequences, [20]. Experimental results demonstrate that the robustness of hidden data inside still images/video sequences against bursts of errors is significantly improved by using 2-D/3-D SP interleaving followed by ECC.

2-D SP Interleaving for Still Image Data Hiding

Consider the "Lena" image ($256\times256\times8$) shown in Figure 12 (a), a widely used image for image processing experiments. The data hiding is carried out in the block discrete cosine transform (DCT) domain. First, the image is split into non-overlapped blocks of 8×8 pixels each. Then, DCT transform is applied to each block. The largest three AC DCT coefficients are used to embed bits. If the bit to be embedded is "1", to the coefficient is added a quantity Δ (e.g., $\Delta = 6$ is empirically used in our simulations). If the bit is "0", from the coefficient Δ is subtracted. We use six bits to represent one symbol. Two types of ECC are used in our simulations. In simulations involving 2-D bursts of errors and cropping, we used the BCH (31, 6) code. For random rows/columns removal, also known as jitter attack, we used the BCH (63, 7) code. Hence, 99 symbols are embedded in the former case and 48 symbols in the latter case. The image is *scanned* three times. Each scan embeds 1/3 of the total bits (about 1024 bits) in an AC coefficient having the same position within each block. These 1024 bits are first interleaved using the 2-D SP interleaving technique, and are then embedded into the AC coefficients. For the data extraction, parts damaged by the error burst are replaced with "0"s. Without 2-D interleaving, we simply embed bits block by block, say, from left to right, from top to bottom through out the whole image. In each block, we embed three bits. The experimental results are shown in Figures 12 to 14, respectively.

In Figure 12, the horizontal axis represents the size of the 2-D error burst. For example, size 2 means that the error burst is a square area of 2×2 blocks (each block is of 8×8 pixels). That is, the square error burst has the size of 16×16 pixels. The vertical axis stands for the symbol error rate (SER). In the simulation, we consider all possible positions of the error burst. Then the arithmetic average of SERs corresponding to the bursts occupying all possible positions is reported in the figure. As shown in the figure, without interleaving, the SER emerges when the error burst size is larger than four. With 2-D SP interleaving, the SER is still zero even when the error burst size is 16. This indicates that even when a quadrant of the marked Lena image has been in error, the SER is still zero, implying significant improvement of robustness. Note that when the error burst size is larger than 22, the SER with interleaving will be larger than that without interleaving. In this case, almost half of the image has been damaged. The SER for both algorithms with and without interleaving has been almost 50%, practically rendering both algorithms useless in this case.

Cropping is a test function of the well-known benchmark software StirMark [21]. In Figure 13, the horizontal axis represents the size of cropping. For example, a cropping of 10 means that five percent of the stego-image is cropped from the top, from the bottom, from the left, and from the right while keeping the central part intact. That is, with respect to a value, say, x, along the horizontal axis, cropping will be conducted from four boundaries of an image inwards by $x/2\%$, while leaving the central portion intact. From this figure, it is observed that with interleaving the SER remains to be zero even when the cropping size is 10, while without interleaving, the SER emerges when the cropping size is only two. Obviously, the robustness against cropping attack has been improved with interleaving. Note that when the cropping size is close to 50, in which case almost all the image has been cropped, the SER with interleaving is higher than that without interleaving.

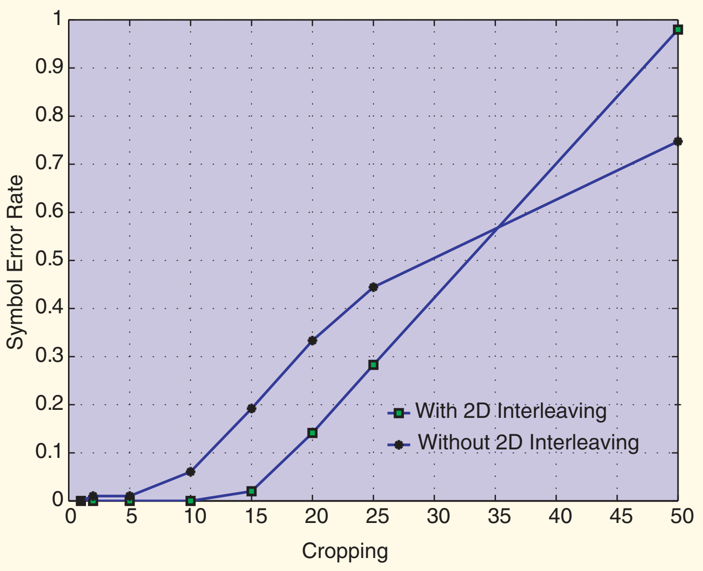


Figure 13. Test results of cropping.

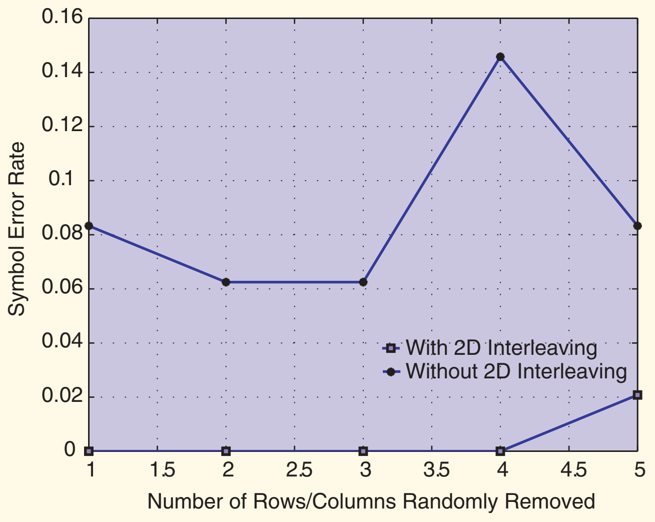


Figure 14. Test results of random rows/columns removal.

Jitter attack is another testing function of the SirMark. It randomly removes a few rows and/or a few columns. The experimental results for jitter attack are depicted in Figure 14. The horizontal axis represents the number of columns and/or rows randomly removed, while SER is represented by the vertical axis. It is observed that the SER with interleaving is constantly well below that without interleaving.

3-D SP Interleaving for Video Sequence Data Hiding

A video sequence is a set of successive frames. Each frame is a 2-D image. Here we consider two types of bursts of errors that may occur to the data embedded in a video sequence. The first type of error bursts is frame loss. Frame loss may occur in video transmission, especially when the video is transmitted through a bursty and noisy channel. The second type of error bursts is what is known as 3-D error bursts. Since there is a high correlation among the successive frames, a 2-D error burst sometimes leads to errors approximately in the same location of the succeeding frames, thus causing a 3-D error burst.

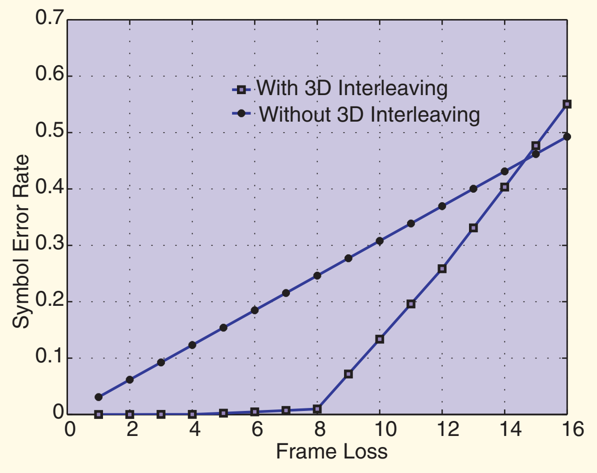


Figure 15. Test results of video frame loss.

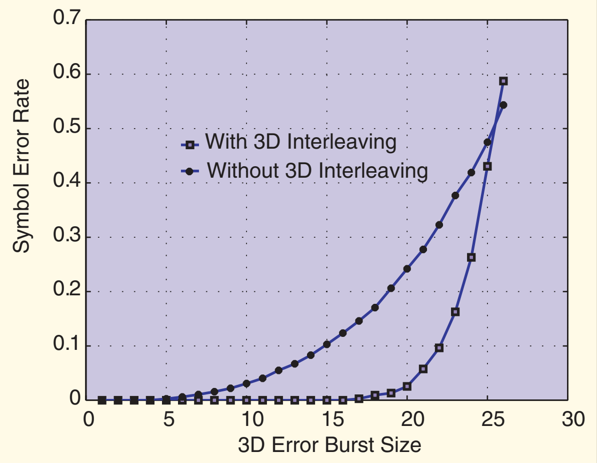


Figure 16. Test results of 3-D error burst.

For these two types of error, again, we conducted our simulation in a way similar to that used for image data hiding. The testing video sequence has 32 frames. Each frame is a gray image of 256×256. We split each frame into non-overlapping blocks of 8×8 pixels each. As a result, we have 32×32×32 blocks within the entire sequence. DCT transform is applied to each block and the data bits are embedded into the three largest AC DCT coefficients in each block. Again, six bits are used to represent one symbol and the BCH (31, 6) code is used. Data embedding is carried out in three scans. In each scan, we embed $32 \times 32 \times 32 = 32,768$ bits (equivalent to 1,057 symbols) in an AC coefficient having the same position in each block. These 32,768 bits are first interleaved using the 3-D SP interleaving technique, which is a straightforward extension from 2-D SP to the 3-D case [22, 23], before being embedded into the AC coefficients. In the data extraction, error parts are filled with "0"s. Without 3-D interleaving, we simply embed bits block by block, say, from left to right, from top to bottom, and from front to rear. In each block, we embed three bits. Figures 15–16 show the simulation results, demonstrating that 3-D interleaving can greatly improve the robustness of data hiding.

In Figure 15, the horizontal axis denotes the number of lost frames (that are consecutive). From this figure, it is seen that when eight frames are lost, the SER with interleaving is almost zero while that without interleaving is about 25 percent. Note that when 15 frames are lost, the error rate with interleaving will be higher than that without interleaving, in which case almost half of the 32 frames are lost. The SER for both algorithms with and without interleaving is almost 50%, implying that the hidden data have been severely damaged.

In Figure 16, the horizontal axis is the size of the 3-D error burst. For example, size 2 means that the error burst is a cubic volume of 2×2×2 blocks (each block is of 8×8 pixels). That is, the cubic error burst has the size of 16×16 pixels in two consecutive frames. With interleaving, the SER is still zero even when the error burst size is 16 (one eighth of the blocks is lost), while without interleaving the SER is more than 12%. Clearly, the significant improvement on the robustness of hidden data against 3-D bursts of errors has been achieved.

Summing up, our initial investigation of applying 2-D/3-D SP interleaving techniques to enhance the robustness of hidden data in still images/video sequences has been reported. It is shown that either in still image or in video sequence data hiding, the SP interleaving can greatly enhance the robustness of hidden data against bursts of errors. Therefore, 2-D/3-D successive packing interleaving can play a promising role in enhancement of robustness in image/video data hiding. It is expected that the 2-D interleaving that can make 2-D data more robust and reliable can also find important applications in the areas such as 2-D bar-codes and holographic storage.

Conclusions

In this article, we first distinguish bursts of errors from random errors. Then we illustrate the philosophy of interleaving by means of an example, i.e., interleaving can convert a bursty channel into a random-like one. Consequently, interleaving, together with a wide spectrum of readily available random error correction codes, can combat effectively bursts of errors.

This article focuses on the state of the art of 2-D/3-D interleaving techniques, i.e., the fundamental ideas, algorithms, and performances of the *t*-interleaved array and the successive packing techniques. In a former technique, Blaum, Bruck and Vardy have defined an M-D burst of errors as an arbitrarily-shaped connected volume, and the optimality of M-D interleaving. They have formulated fundamental building blocks and proposed to tile closely building blocks to form *t*-interleaved arrays to combat bursts of errors. They have proved that the optimality can be guaranteed in the 2-D case, but not in the 3-D case. In the two-dimensional case, for each given burst size, t_0, a specific algorithm is implemented, which can correct arbitrarily-shaped error burst of size t_0 and is optimal. When t increases ($t > t_0$), the algorithm with a set of new parameters needs to be implemented once again in order to correct larger error bursts. When t decreases ($t < t_0$), the interleaved array obtained with t_0 is not optimal any more. However, in reality the size of error bursts is normally not known in advance, while the size of 2-D/3-D arrays is often given. This somewhat restricts the practical utilization of the technique. Based on these observations, the successive packing approach has been developed by Shi and Zhang. It can be implemented for a given 2-D/3-D array once, and has been proved to remain optimal for a set of bursts of errors of different sizes. Therefore, it is practical. The performance comparison between the UPS' approach (using 1-D block interleaving plus a 2-D writing procedure), the *t*-interleaved array approach, and the successive packing approach is illustrated via an example along with observations and comments. It has been observed that both the *t*-interleaved array technique and the successive packing technique can be applied to a square array of $2^{2p-1} \times 2^{2p-1}$ where p is a positive integer, and both can correct an arbitrarily-shaped error burst of size 2^p with the same interleaving degree. The SP technique, however, can correct optimally some other error bursts of different sizes. Therefore, SP is versatile. Successive doubling and 2×2 basic array are the key elements of the successive packing technique, bearing a resemblance to the mechanism of the fast Fourier transform. Some future research on the successive packing approach has been discussed.

As a practical application, we have applied successive packing 2-D/3-D interleaving to enhance the robustness of still image/video sequence data hiding. The initial investigation in this regard is promising and has demonstrated significant improvements of the robustness of image/video data hiding.

Acknowledgement

This work was supported in part by New Jersey Commission of Science and Technology via NJCWT and NJWINS, New Jersey Commission of Higher Education via NJ-I-TOWER, and NSF via IUCRC.

References

[1] R. C. Bose and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes,” *Inform. Control*, vol. 3, pp. 68-79, March 1960

[2] A. Hocquenghem, *Codes Correcteurs d'Erreurs*, Chiffres, vol. 2, pp. 147–156.

[3] S.B. Wicker, *Error Control System for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, Inc, 1995.

[4] H. Imai, “Two-dimensional Fire codes,” *IEEE Transactions on Information Theory*, vol. 19, no. 6, pp. 796–806, 1973.

[5] M. Blaum, J. Bruck, and A. Vardy, “Interleaving schemes for multidimensional cluster errors,” *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 730–743, 1998.

[6] Y.Q. Shi and X.M. Zhang, “A new two-dimensional interleaving technique using successive packing,” *IEEE Transactions on Circuits and Systems, Part I: Fundamental Theory and Application*, Special Issue on Multidimensional Signals and Systems, vol. 49, no. 6, pp. 779–789, June 2002.

[7] T. Pavlidis, J. Swartz, and Y.P. Wang, “Information encoding with two-dimensional bar-codes,” *IEEE Computer Magazine*, pp. 18–28, June 1992.

[8] J. Pjatek, “Automotive industry recommends 2-D standards,” *Automatic I. D. News*, pp. 54–56, Aug. 1994.

[9] MaxiCode Briefing Document, UPS Inc., 1996, http://www.maxicide.com.

[10] K.A.S. Abdel-Ghaffar, R.J. Mceliece, and H.C.A. Van Tilborg, “Two-dimensional burst identification codes and their use in burst correction,” *IEEE Transactions on Information Theory*, vol. 34, no. 3, pp. 494–504, 1988.

[11] K.A.S. Abdel-Ghaffar, “Achieving the Reiger bound for burst errors using two-dimensional interleaving schemes,” in *Proc. IEEE Int. Symp. Information Theory*, Ulm Germany, 1997, p. 425.

[12] C. de Almeida and R. Palazzo, Jr., “Two-dimensional interleaving using the set partitioning technique,” in *Proc. IEEE Int. Symp. Information Theory*, Trondheim, Norway, 1994, p. 505.

[13] M. Blaum, J. Bruck, and P.G. Farrell, “Two-dimensional interleaving schemes with repetitions,” in *Proc. IEEE Int. Symp. Information Theory*, Ulm Germany, 1997, p. 342.

[14] B. Chor, C.E. Leiserson, and R.L. Rivest, “An application of number theory to the organization of raster-graphics memory,” *Journal of the Association for Computing Machinery*, vol. 33, no. 1, pp. 86–104, 1986.

[15] S.W. Golomb and L.R. Welch, “Perfect codes in the Lee metric and the packing of polyominnoes,” *SIAM J. Appl. Math.*, vol. 18, no. 2, pp. 302–317, Jan. 1970.

[16] T. Etzion and A. Vardy, “Two-dimensional interleaving schemes with epetitions: constructions and bounds,” *IEEE Transactions on Information Theory*, vol. 48, no. 2, pp. 428–457, 2002.

[17] V.C. da Rocha, Jr., W.P.S. Guimarães, and P. Farrell, “Two-dimensional interleaving schemes with burst error-correcting codes,” *IEE Electronics Letters*, vol. 38, no. 18, pp. 1042–1043, 2002.

[18] J. Huang, G. Elmasry, and Y.Q. Shi, “Power constrained multiple signaling in digital image watermarking,” in *Proc. of 1998 IEEE Workshop on Multimedia Signal Processing*, Los Angeles, CA, USA, Dec. 1998, pp. 388–393.

[19] J. Huang and Y.Q. Shi, “Reliable information bit hiding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 916–920, Oct. 2002.

[20] Y.Q. Shi, Z.C. Ni, J. Huang, N. Ansari, and W. Su, “A successive 2-D/3-D interleaving to enhance robustness of image/video data hiding,” *IEEE International Symp. on Circuits and Systems,* Bangkok, Thailand, May 2003. (Submitted)

[21] F. Petitcolas et al. (1997, Nov.). *Stirmark2.* Available www.cl.cam.ac.uk/~fapp2/watermarking

[22] X.M. Zhang, Y.Q. Shi, and S. Basu, “On successive packing approach to multidimensional (M–D) interleaving,” Presented at *Fifteenth International Symp. on Mathematical Theory of Networks and Systems (MTNS02)*, University of Notre Dame. Available: http://www.nd. edu/~mtns

[23] G.F. Elmasry, “Detection and robustness of digital image watermarking signals: A communication theory approach,” Ph.D. dissertation, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Aug. 1999.

Dr. Yun Q. Shi has joined the Department of Electrical and Computer Engineering at the New Jersey Institute of Technology, Newark, NJ since 87, and is now a professor there. He obtained his B.S. and M.S. degrees from the Shanghai Jiao Tong University, Shanghai, China; his M.S. and Ph.D. degrees from the University of Pittsburgh, PA. His research interests include theory of multidimensional systems and signal processing (robust stability of linear systems, 2-D spectral factorization, 2-D/3-D interleaving), visual signal processing and communications, digital multimedia data hiding, computer vision, applications of digital image processing and pattern recognition to industrial automation and biomedical engineering. Prior to entering graduate school, he had industrial experience in a radio factory as a principal design and test engineer in numerical control manufacturing and electronic broadcasting devices. Some of his research projects are currently supported by several federal and New Jersey State funding agencies.

He is an author/coauthor of more than 120 journal and conference proceedings papers in his research areas and a book on *Image and Video Compression for Multimedia Engineering* (99). He has been an IEEE senior member (93-), the chairman of Signal Processing Chapter of IEEE North Jersey Section (96-), an editorial board member of *International Journal of Image and Graphics* (99-), a member of IEEE CASS Technical Committee of Visual Signal Processing and Communications as well as Technical Committee of Multimedia Systems and Applications (01-), a member of IEEE SPS Technical Committee of Multimedia Signal Processing (02-). He is currently an IEEE CASS Distinguished Lecturer (02–03). He was a co-general chair of IEEE 2002 International Workshop on Multimedia Signal Processing, a formal reviewer of the *Mathematical Reviews* (87–99), an Associate Editor for *IEEE Transactions on Signal Processing* in the area of Multidimensional Signal Processing (94–96), the guest editor of the special issue on Image Sequence Processing for the *International Journal of Imaging Systems and Technology* (98), one of the contributing authors in the area of Signal and Image Processing to the *Comprehensive Dictionary of Electrical Engineering* (98). His biography has been selected by Marquis Who's Who for inclusion in *Who's Who in Science and Engineering*.

Xi Min Zhang received his B.E. degree from Xi'an Jiaotong University, Xi'an, China, and his M.E. degree from Nanyang Technological University, Singapore in 1991 and 1999, respectively. Since 1999, he has been a research assistant in New Jersey Institute of Technology, Newark, NJ, where he is working towards his Ph.D. degree. All of his degrees are in electrical engineering.

From 1991 to 1997, he was with CAAC, China as a systems engineer. In the summer of 2002, he worked as an intern in Mitsubishi Electric Research Labs, Murray Hill, NJ. He received the best student paper award at SPIE's VCIP02. His research interests include M-D interleaving for various information security issues, MPEG video codec, streaming video through network, error resilient coding, artificial neural network and digital watermarking.

Zhicheng Ni received his B.E. degree from Southeast University, Nanjing, China, his M.E. degree from Nanyang Technological University, Singapore in 1994 and 2000, respectively. Since 2000, he has been a research assistant in New Jersey Institute of Technology, Newark, NJ, where he is working towards the Ph.D. degree, all in electrical engineering. From 1994 to 1998, he was an electronic engineer in a nuclear power research institute, China. His research interests include fractal and interleaving coding, digital watermarking and authentication.

Nirwan Ansari received the B.S.E.E. (summa cum laude), M.S.E.E., and Ph.D. from NJIT, University of Michigan, and Purdue University in 1982, 1983, and 1988, respectively. He joined the Department of Electrical and Computer Engineering, NJIT, in 1988, and has been Professor since 1997. He is a technical editor of the IEEE Communications Magazine as well as the Journal of Computing and Information Technology, is the General Chair of ITRE2003, was instrumental, while serving as its Chapter Chair, in rejuvenating the North Jersey Chapter of the IEEE Communications Society which received the 1996 Chapter of the Year Award, served as the Chair of the IEEE North Jersey Section and served in the IEEE Region 1 Board of Governors during 2001–2002, and currently serves in various IEEE committees. He was the 1998 recipient of the NJIT Excellence Teaching Award in Graduate Instruction, and a 1999 IEEE Region 1 Award. His current research focuses on various aspects of high speed networks and multimedia communications. He authored with E.S.H. Hou Computational Intelligence for Optimization (1997, and translated into Chinese in 2000), and edited with B. Yuhas Neural Networks in Telecommunications (1994), both published by Kluwer Academic Publishers. He has also contributed over 150 technical papers in his areas of research.