RJ-10013 (89104) March 25, 1996
Holographic Storage

Report on Coding Techniques for Holographic Storage

Jonathan J. Ashley, Mario Blaum and Brian H. Marcus *
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA

Abstract

Storing data holographically presents a number of coding challenges. Firstly, identifying the correct constraints creates the need for two-dimensional and even three-dimensional modulation codes. Secondly, the noise produces particular two dimensional error patterns, requiring special configurations of the error-correcting codes.

We present a menu of possible modulation coding and error-correction coding techniques to improve the reliability of data stored holographically.

*We gratefully acknowledge assistance from members of the HOST team, especially John Hoffnagle and Bob Shelby

1 Introduction

We assume that the reader is somewhat familiar with the basic idea of a holographic recording scheme. Here, we give only a brief description. For further information, the reader may consult [29], [15] and the references therein.

A laser illuminates a programmable spatial light modulator (SLM), thereby generating an object beam, which represents a (two-dimensional) page of data. The data page is a pattern of 0's and 1's where a 1 is represented by light and a 0 by dark. A reference beam, which is a simple plane wave generated by the same laser, interferes with the object beam at a spot on the recording medium and writes a grating pattern (the hologram). The original page of data is retrieved at a later time by illuminating the medium with the same reference beam that was used to record it. The data is then collected on a photosensitive array of detectors, known as a charge coupled device (CCD). Several holograms can be recorded in the same physical spot in a scheme known as angular multiplexing by varying the angle of propagation of the reference beam. Such a collection of holograms is called a stack. Since an entire page of data can be retrieved all at once, holographic data storage holds the promise of very high data rate. Since many holograms can be recorded in one stack, it also holds the promise of very high data density.

The purpose of this report is to provide a menu of error-control coding options for improving the performance of a holographic data storage system. We will describe some contributions from the literature as well as new ideas of our own. At this point, most of these ideas have not been adequately tested. We have made some use of experimental data provided by the HOST team's optical tester (with data masks for input rather than an SLM) as well as simulations using a very simple mathematical model of the storage system. In the coming months, we expect an SLM to be integrated to the tester, and this should provide more realistic experimental data.

2 Noise Sources

We view a holographic data storage system as a noisy channel (which we call the holographic data channel). We consider noise to be anything that distorts the data signal, whether in a systematic or statistical way. There are many noise sources that affect the performance of this channel. The most dominant source is the loss in diffraction efficiency due to multiplexing of several holograms in a single stack. It is fairly well accepted that as the number M of holograms written in a stack increases, the diffraction efficiency for retrieving a single hologram decreases as $1/M^2$ [18], [6], [22]. The noise due to scatter of the reference beam in the retrieval process is independent of the number of holograms, and so the signal-to-noise ratio (SNR) is degraded by the same factor, $1/M^2$. Another noise source is crosstalk from adjacent holograms, but the SNR degradation due to crosstalk is believed to be on the order of $1/M$, and so appears to be a less serious problem [13], [7]. However, improvements in optics as well as materials may change this situation.

There are many other noise sources. For instance there may be nonuniformities in the object beam or reference beam. Defects in the storage medium itself can introduce distortions. Since the object beam is really a spatial Fourier transform of the data pattern, the finite extent of the storage medium causes a low-pass filtering effect, so that data content at high spatial frequencies may be lost. Finally, at the detector itself, there are several sources of noise: quantum noise (causing photon counting errors), noise in the camera electronics, and rotations and distortions can cause pixel registration errors, i.e., difficulty in matching the CCD pixels to the SLM pixels.

3 Overview of error-control coding

How can we fight these various noise sources? For some problems, such as rotations causing pixel registration errors, fiduciary marks can be imposed to help match CCD pixels to SLM pixels. For other sources of noise, the data can be coded so as to reduce the impact of the noise. Figure 1 shows the traditional scheme for coding data

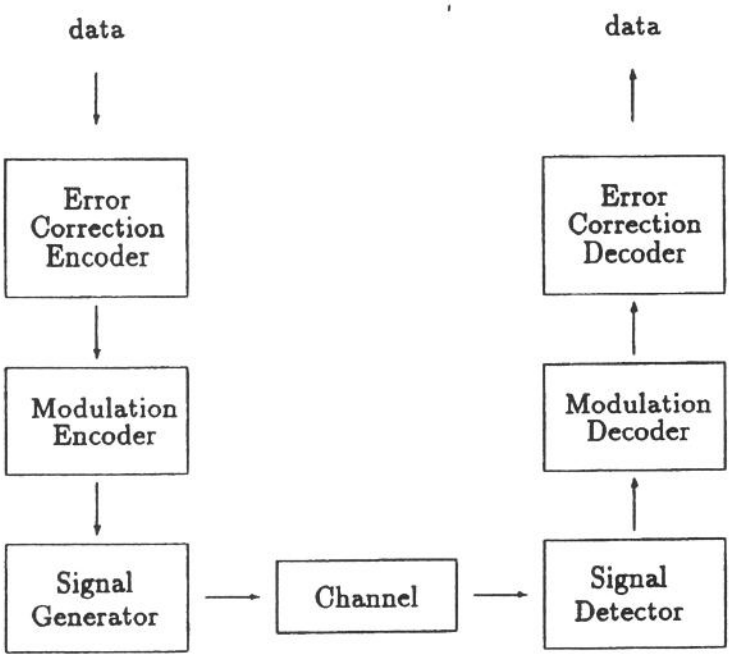

Figure 1: Coding for a Noisy Channel

through a noisy channel.

User data is encoded via an error-correction code (ECC) and then a modulation code. The purpose of the modulation code is to adapt the recorded data to the characteristics of the channel – in other words, to "shape" the data in a way that is less likely to be corrupted by noise. For instance, we will see how a modulation code can be used to compensate for loss of high frequency data content. The purpose of the ECC is to correct errors: by adding redundancy, data patterns can be retrieved correctly. Neifeld and McDonald [24] show how ECC can be used to fight the $1/M^2$ degradation in SNR and therefore improve the overall capacity.

Both kinds of coding are performed digitally, while the outputs from the CCD are essentially analog (photon counts). So, before the modulation decoder can process the data, a detection scheme (also called a detector, not to be confused with photodetector) must make a digital decision on the outputs from the CCD. Ultimately,

the detection scheme, modulation code and ECC should be designed in concert so that they work well together. For instance, the detection scheme can use information based on the constraints of the modulation code, and the ECC may need to correct bursts of errors caused by the modulation decoder. In the next three sections, we will discuss modulation codes, detection schemes and ECC, giving some general ideas and some simple examples.

A conventional magnetic or optical storage system is usually regarded as a one-dimensional channel since each track carries the information contained in a one-dimensional stream of data (if one takes into account the effects of neighboring tracks, then such a system is regarded as a two-dimensional channel). Since holograms represent two-dimensional pages of data, the holographic data channel can be viewed as a two-dimensional channel. With angular multiplexing taken into account, it becomes a three-dimensional channel. The theory and practice of coding and detection is much better developed for one-dimensional channels than for higher-dimensional channels. Nevertheless, we believe that many one-dimensional techniques can be adapted to the holographic data channel.

We will want to use a combination of schemes that yields the highest data rate, highest capacity and lowest bit-error rate (BER). But undoubtedly there will not be a single scheme that performs best with respect to all figures of merit, and so we expect that tradeoffs will need to be made.

In order to achieve a high data rate, the page-oriented structure of the holographic data channel can be exploited as follows. The CCD array can be divided into a fixed number of regions such that within each region data can be fed into a single channel (for this reason we will refer to the regions themselves as channels). In this way, data can be retrieved from the CCD in several parallel channels (say 64 channels). Of course in order to match the output, data will have to be programmed into channels on the SLM which match those of the CCD. A natural question is how to configure these channels in an optimal way. At one extreme, we could divide the CCD into squares of the same size, and at the other extreme into horizontal strips of the same size. We will refer to these two approaches as the square and the strip approach respectively.

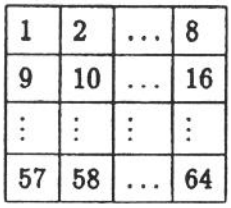

Figure 2: The square approach for the 64 channels

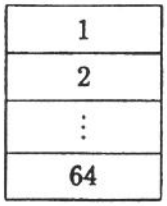

Figure 3: The strip approach for the 64 channels

Figures 2 and 3 illustrate the two approaches. From a hardware implementation point of view, the strip approach appears to be best. In Section 4.2 we will see that the strip approach also appears to work best from the modulation coding point of view. However, for error-correction coding, there appear to be tradeoffs between these two approaches; see Section 6.

4 Modulation coding

A modulation code consists of an encoder, decoder and a modulation constraint, the latter being a constrained set of channel input sequences. The encoder maps arbitrary data to sequences that satisfy the modulation constraint. For instance, in conventional magnetic recording, a modulation encoder maps arbitrary data sequences to run length limited sequences, i.e., sequences where runs of zeros cannot be too long or too short. This constraint helps to overcome problems of intersymbol interference (interference between successive symbols) and also to provide timing control.

In the holographic setting, we assume that data arrives at the SLM in the form of an ordinary 1-dimensional binary sequence. A modulation encoder will map each user

data block of a fixed length m into a pattern of some fixed area A that satisfies the constraint. The code rate of such an encoder is m/A, indicating that A SLM pixels will represent m information bits. Of course, the higher the code rate, the higher the data density.

The capacity of a modulation constraint is defined to be the asymptotic growth rate of the number of patterns that satisfy the constraint. Capacity is an important parameter because it upper bounds the rate of any encoder.

For one-dimensional channels, the general theory shows how to compute capacities of modulation constraints. The theory also shows how to construct encoders and decoders of highest possible rate [20]. In marked contrast, for two-dimensional channels, there is no general formula for capacity, and there is no general method for constructing encoders [28](Chapter 5),[21]. But in many cases one can get good estimates for capacity and construct encoders whose rates are close to capacity.

In Section 4.1 we describe some modulation constraints that seem natural for the holographic data channel. Along the way, we will also indicate some simple modulation encoding schemes. In Section 4.2 we will explore encoding schemes in a more general perspective.

4.1 Examples of Modulation Constraints

4.1.1 Balanced pages

A sequence or array of binary numbers is balanced if it contains the same number of 0's as 1's. If each data page input to the SLM is balanced (or at least roughly balanced), then the overall light intensity will not vary much from page to page, ensuring that the intensity ratio between the the object beam and reference beam will be relatively constant. In this way, an exposure schedule for writing multiple holograms within a stack can be set accurately. Moreover, the balanced condition would be a necessary condition for a detection scheme which uses a global binary threshold.

One way to achieve the balanced page condition is to tile the entire page with balanced sub-arrays of a given relatively small size. This will limit variations in reconstructed pixel intensities within small regions of the CCD array. For instance, there are two possible balanced sub-arrays of size 1×2: 01 and 10. At the detection stage, within each such array we could declare the high pixel value to be 1 and the low pixel value to be 0. Equivalently, we could subtract the received value of the left bit from the received value of the right bit; if the result is positive, we declare the sub-array to be 01, and if the result is negative, we declare the sub-array to be 10. Note that there is a very simple encoder for this modulation constraint: the sub-array pattern 01 represents 0 and the sub-array pattern 10 represents 1 (this is known as differential encoding). The rate of this code is 1/2, and so there is a 2-fold penalty in data density.

Of course, one can use bigger sub-arrays. Since the number of balanced sequences of length n grows exponentially at roughly the same rate as the number of all sequences of length n, the use of bigger sub-arrays will yield a higher code rate and therefore impose a smaller penalty on data density. At the detection stage, within a balanced sub-array we can again use a local threshold – if the number of pixels in the sub-array is $2b$, declare the b pixels of lowest intensity to be 0 and the b pixels of highest intensity to be 1 (as described in [14]). Some very attractive balanced sub-array codes have been designed by Vardy, et al. [30]. These codes are efficient, relatively easy to implement and can be used for error-detection as well.

Large patterns of all dark or all light can create a "ghosting" effect; for instance the presence of a large region of light may cause light to be dispersed beyond the region. The maximum size of a pattern of all dark or all light can be limited by requiring a minimum number of transitions from dark to light and transitions from light to dark within each row and column of the data array. Efficient modulation codes for this kind of constraint, combined with the balanced page constraint, have been constructed in [30].

4.1.2 Avoiding spectral spikes

In holographic recording, the object beam really describes the spatial Fourier transform of the data pattern, input to the SLM. If the transform had very high intensity within a narrow band of spatial frequencies, the dynamic range of the medium may be overwhelmed. Thus, within any narrow band of frequencies, it is desirable to limit the total energy of the data patterns. At DC, this can be achieved by placing a random phase-shifting mask in front or behind the SLM. This also has the effect of smoothing the Fourier transform at moderate frequencies. Modulation constraints which forbid the appearance of certain periodic patterns would also help to smooth the Fourier transform.

4.1.3 High-frequency cut-off

For simplicity consider the case where both the SLM and the CCD are arrays of the same size. In order to make the most efficient use of the channel, we assign each information bit to a single SLM pixel and image it onto a single CCD pixel. (the "one-to-one" method). At this extremely high resolution, the data may be badly distorted. This is a result of the imperfection in the optics, the finite extent of the crystal (which causes high spatial frequencies to be truncated) and the shape of the reference beam. In particular, the truncation of high spatial frequencies causes a low-pass filtering or "smearing" effect. Perhaps the simplest approach to this problem would be to oversample the data, say by representing each information bit as a 2-by-2 square of pixels, i.e., represent a 0 by a 2-by-2 square of 0's and a 1 by a 2-by-2 square of 1's (the "oversampling" method). The corresponding detection scheme would be to sum the photon counts within each square. Preliminary experiments indicate that the oversampling method provides much better resolution than the one-to-one method. Of course, there is a penalty to be paid: the data density is reduced by a factor of 4.

Both the "one-to-one" method and the "oversampling" method can be viewed as (rather trivial) modulation constraints (together with very simple encoders). Between these two extremes there are several intermediate possibilities. Below we have listed

seven constraints in order of increasing severity.

Constraint #1: The One-to-One Method

Constraint #2: Requires that no 0 can be completely surrounded by 1's, and similarly that no 1 can be completely surrounded by 0's, i.e., forbid

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{matrix}$$ and

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}.$$

Constraint #3: Requires that the four nearest neighbors to a 0 not all be 1's and similarly that the four nearest neighbors to a 1 not all be 0's, i.e., forbid

$$\begin{matrix} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{matrix}$$

and

$$\begin{matrix} & 0 & \\ 0 & 1 & 0 \\ & 0 & \end{matrix}$$

Constraint #4: Requires that no three of the four nearest neighbors to a 0 be all 1's and similarly that no three of the four nearest neighbors to a 1 be all 0's, i.e., forbid

$$\begin{matrix} & 0 & \\ 1 & 0 & 1 \\ & 1 & \end{matrix}$$

, etc.

Constraint #5: Requires that no 0 or 1 be isolated either horizontally or vertically, i.e., forbid 1 0 1, 0 1 0,

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$ and $$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}$$.

Constraint #6: Requires that every 0 belong to a 2-by-2 square of 0's and similarly that every 1 belong to a 2-by-2 square of 1's.

Constraint #7: The Oversampling Method

Constraint	#1	#2	#3	#4	#5	#6	#7
Capacity	1.00	0.90	0.86	0.67	0.47	0.46	0.25

Table 1: Capacity estimates for constraints #2 through #6, intermediate to the one-to-one method (#1) and the oversampling method (#7).

It can indeed be verified that as n increases from 1 to 7, Constraint #n becomes increasingly more severe, and so it would be expected that the channel performance would improve while the data density would degrade. Simulations based on a very simple channel model seem to bear this out; see Table 1, which lists estimates of the capacities of the constraints, and Figure 4, which graphs $\log_{10}$(bit-error-rate) as a function of a high-frequency cutoff frequency (measured in multiples of the bit frequency) for each constraint. For cut-off frequencies above 0.35, the more severe the constraint is, the lower the bit-error-rate is.

The simulations above were carried out using test patterns generated by a program which designs "pseudo-random" patterns subject to a given constraint. These test patterns were also used for the capacity estimates listed in Table 1. Encoders for some of these constraints (in particular, Constraint #6) were designed using the method described below in Section 4.2.2.

While Figure 4 shows that there are tradeoffs to be made among these modulation constraints, there is a way to normalize so that the constraints can be compared at any given BER. Namely, if we assume that the cause of the high-frequency cutoff is due solely to the finite extent of the crystal then for any BER, we can compute the capacities of the constraints per unit area. For instance from Figure 4 we see that we can achieve BER $\approx 10^{-4}$ for Constraint #5 with frequency cutoff $\approx .41$, for Constraint #6 with frequency cutoff $\approx .40$ and for Constraint #7 (oversampling) with frequency cutoff $\approx .36$. For this BER, we then have estimated areal capacity for Constraint #5 of $(.47/(.41^2)) \approx 2.796$ bits per unit area, for Constraint #6 of $(.46/(.40^2)) \approx 2.875$ bits per unit area, and for Constraint #7 of $(.25/(.36^2)) \approx 1.93$ bits per unit area. Of course, these comparisons ignore important contributions to high-frequency cutoff such as loss of resolution due to imperfections in the optics.

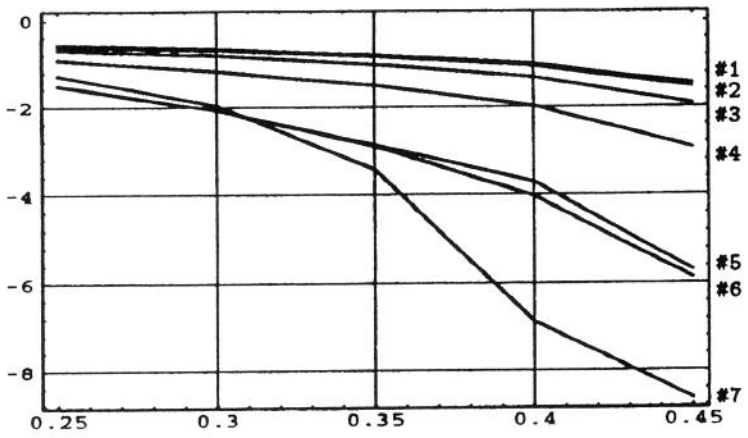

Figure 4: Bit error rate for seven constraints as a function of the high-frequency cutoff.

Another way to protect against the "smearing" effect caused by high frequency truncation is to provide "guarding" around the data pixels. The light from a pixel corresponding to a 1 may leak into an adjacent pixel corresponding to a 0 – possibly causing the 0 to be mis-read as a 1. One approach to this problem would be to represent data by a 2×2 square, where the data bit, 0 or 1, is written in the upper-left-hand corner and the remainder of the square is left dark. Using input masks on the optical tester, the HOST team has shown that this works well – especially if the CCD array is shifted so that each SLM pixel image is centered on the common corner of 4 CCD pixels as in Figure 4.1.3 and the 4 pixel counts are summed; in this way a data '1' is effectively written as a 2×2 square which is most intense at the center of the square and dies off toward the boundary of the square. Of course, as in oversampling there is a 4-fold penalty in data density.

A second approach to "guarding" would be to write data bits only in alternating squares in a checkerboard fashion, always leaving the remaining squares dark. If the CCD array is shifted horizontally so that each SLM pixel image is centered on an edge separating two CCD pixels then each data '1' is effectively written as a 2×1

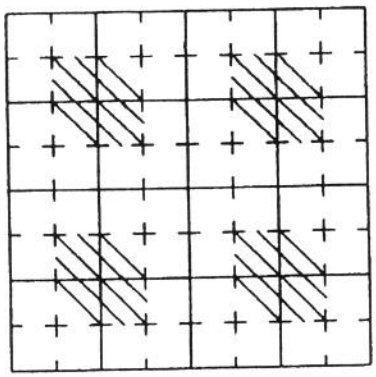

Figure 5: Shifting the CCD array; the dashed lines indicate the SLM grid, the solid lines indicate the CCD grid, and the hashed areas indicate the pixels that carry the data

rectangle. See Figure 6. This entails only a 2-fold penalty in data density, but may yield higher bit error rates.

A third possibility to "guarding" would be to use grey-level encoding; that is, use three intensity levels on the SLM: 0, .5, 1, and then impose the modulation constraint that 0's and 1's cannot be adjacent; in this scheme there would be less variation in the light content between adjacent pixels. This would require that the SLM be programmed to deliver three distinct intensity levels and that the three levels be distinguished accurately. But the modulation constraint (that 0's and 1's cannot be adjacent) would help in this task, and the data density would be higher than in the approaches described above.

4.1.4 Inter-page constraints

It may be that the extent of interference between adjacently multiplexed holograms is data dependent. For instance, a 1 at a particular pixel location in one hologram may cause a 0 at the same pixel location in an adjacent hologram to be mis-read as a 1. There may be other patterns that are more prone to be mis-read and these may suggest natural modulation constraints on the data content of successive pages or of small regions within successive pages.

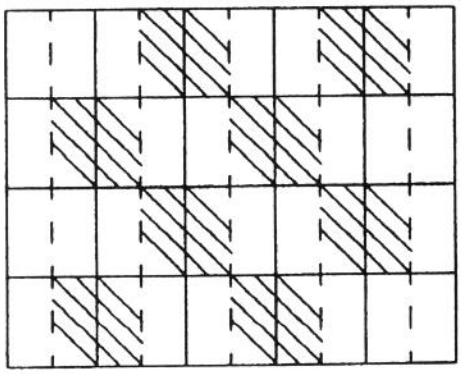

Figure 6: Shifting the CCD array horizontally

4.2 Modulation Encoders

Once we have imposed a modulation constraint, we must design a modulation encoder which maps arbitrary data patterns to patterns that satisfy the constraint. In the preceding section we listed several constraints, and for some of these we indicated some simple encoding methods. In this section, we discuss some general approaches to designing encoders and decoders for two-dimensional modulation constraints. At this point, it appears that the strip encoder method (Section 4.2.2 below) will work best primarily because in that approach we can make use of well-established one-dimensional techniques. This would work well with the strip approach for the division of the CCD array into parallel channels mentioned at the end of Section 3.

4.2.1 Block Encoders

Perhaps the simplest encoding method (block encoding) would be to divide the input sequence into non-overlapping binary blocks of some fixed length m, then divide the SLM array into squares of a fixed size $n \times n$, and encode (in a one-to-one manner) the binary blocks of length m to a set $\mathcal{C}$ of $n \times n$ patterns that satisfy the given constraint. Of course the constraint must be satisfied across the square boundaries,

and so the patterns in $\mathcal{C}$ must be freely concatenatable, i.e., when freely pasted together the constraint must still satisfied. This can often be accomplished by adding an additional artificial constraint near the boundary of the square; sometimes this amounts to buffering each square from each adjacent square.

Decoding is performed simply by inverting the encoding rule. The rate of such an encoder is $m/(n^2)$, since m information bits are represented by n^2 SLM pixels. The encoders constructed in [30], designed to achieve balanced pages, are block encoders.

For some modulation constraints, it is impossible to find any such m, n and $\mathcal{C}$ – even modulation constraints of arbitrarily large capacity [26]. But in many specific cases of interest, this can done – though often by ad hoc methods.

4.2.2 Strip Encoders

There are many variations on the idea of a block encoder. For instance, we might allow the basic shape to be a rectangle rather than a square. Carrying this idea to an extreme, we arrive at the method of strip encoders.

In this method, we make essential use of well-developed one-dimensional techniques. The idea is to divide the SLM array into long horizontal strips of dimension N-by-H, where N is the side dimension of the SLM and H is some fixed height (we assume that H is much smaller than N). Now, we view the 0–1 patterns within the strips as one-dimensional sequences of symbols, where the symbols are binary columns of height H, rather than simply binary numbers. The given modulation constraint on two-dimensional patterns imposes a constraint on the one-dimensional patterns within a strip. Often we can impose an additional artificial constraint on the patterns within a strip (in particular near the top and bottom of the strip) to ensure that whenever they are freely pasted together (in the vertical direction), the constraint is still satisfied across the strip boundaries. Modulation encoders which satisfy constraints only within strips (as opposed to across the boundaries) have been considered by Marcellin and Weber [19].

In this way, we have defined an essentially one-dimensional constraint, and we can use

any of the well-developed one-dimensional coding techniques. For instance, we can use the state splitting algorithm [20], which constructs finite-state machines that can be used as encoders. In this algorithm, the decoders have the desirable property that the extent of error propagation is automatically limited. In 1985, this algorithm was implemented in a software tool, written in APL, at IBM Research. Recently, we wrote a C program which will take as input any encoder generated by the APL program and any user data pattern, and then generate the encoded pattern in a format that can be passed on to the SLM.

We illustrate this with the following simple example. There is an ordinary (one-dimensional) encoder at rate 2:3 from arbitrary user data to the modulation constraint that requires 0's and 1's not be isolated (i.e., the one-dimensional patterns 101 and 010 are forbidden); this code can be obtained by a simple modification of the standard $(0,1)$ run-length-limited encoder contained in [20] (p. 16, Figure 14). Now we can use this one-dimensional encoder to encode arbitrary user data into a one-dimensional sequence $y = y_0 y_1 y_2 \ldots y_N$ and then form a strip of height 2 by simply copying y; so a single strip would look like:

$$y_0 y_1 y_2 \cdots y_N$$
$$y_0 y_1 y_2 \cdots y_N$$

It is a simple matter to verify that this code satisfies Constraint #6 within each strip (of height 2), and in this case the definition of the constraint ensures that there is no problem across the boundaries of the strips. This yields a rate $1:3$ code from arbitrary user data to patterns that satisfy Constraint #6. This gives a substantial improvement (33%) in data density over the oversampling method, although there may be a tradeoff in performance (Figure 4).

One possible disadvantage of the strip encoder approach is that the complexity of the code construction procedure as well as the codes themselves may grow rapidly with the height, H, of the strip.

4.2.3 Two-dimensional encoders

It is natural to ask how the standard one-dimensional finite-state coding model can be generalized from one dimension to two dimensions. For ease of exposition, we imagine that the input data is a two-dimensional page of information, but this could be adapted to the case of one-dimensional input. The idea is to define a local encoding rule of the form

$$\begin{aligned} y_{i,j} &= f(z_{i-1,j}, z_{i,j-1}, x_{i,j}) \\ z_{i,j} &= g(z_{i-1,j}, z_{i,j-1}, x_{i,j}) \end{aligned}$$

where, at position (i,j), $x_{i,j}$ is the data bit to be encoded, $z_{i,j}$ is the state and $y_{i,j}$ is the encoded bit.

In ordinary one-dimensional finite-state coding, one needs only to specify a single initial state in order to start the encoding process; then the local encoding rule generates an arbitrarily long string of encoded data. But here, just to get started, we need to specify a pattern of initial states on big chunks of the positive x-axis, $z_{0,0}, z_{1,0}, \ldots, z_{N,0}$ and the positive y-axis, $z_{0,1}, \ldots, z_{0,N}$. Then the local encoding rule will iteratively generate a constrained pattern on an $N \times N$ square. We remark that the input data symbols $x_{i,j}$ and output symbols $y_{i,j}$ need not be single bits: they may be binary blocks of a certain fixed size (say m, n respectively), and then the code rate will be $\frac{m^2}{n^2}$.

This is a natural approach since it is two-dimensional in nature. However, it suffers from two severe problems. First, at any given stage in the encoding procedure, a large "L-shaped" set of state information must be remembered. This state information must be buffered somewhere. This may be feasible because the amount of information that needs to be buffered is only a small fraction (actually the square root) of the amount of information that will be stored. A more serious problem is that we do not yet have a procedure for actually constructing such encoders for any reasonably large class of practical examples.

5 Detection

The simplest kind of detection scheme for an uncoded system makes a decision on the data value at each pixel based on a global binary threshold. Heanue, et al. [14] compared this with optimal detection schemes appropriate for some simple modulation codes, such as grey-level encoding, differential encoding, and oversampling (mentioned earlier). Under the assumption that the dominant noise source is additive Gaussian noise, that the noise is independent from pixel to pixel and that the SNR degrades as $1/M^2$, they concluded that binary thresholding yields the best performance, i.e., lowest BER for a given capacity.

Under less ideal assumptions, it may be useful to use more sophisticated sequence detection methods. As we said earlier, if each data page is tiled by balanced sub-arrays, we can use a local threshold detector within each sub-array. Perhaps adaptive thresholding or maximum likelihood detection schemes that work well in one dimension could be generalized to the two-dimensional case.

If intersymbol interference is extensive, then partial response decoding could be useful. The idea is that if the point spread function of an isolated pulse can be predicted accurately (perhaps after some equalization), then in spite of the intersymbol interference the data can still be resolved from the received signal. For instance, suppose that the point spread function of an isolated 1 were:

$$\begin{array}{cc} & .5 \\ 1 & .5 \end{array}$$

In other words, if $x_{i,j}$ were the data value at pixel (i,j), then (in the absence of noise) the received value $y_{i,j}$ would be

$$y_{i,j} = x_{i,j} + .5x_{i-1,j} + .5x_{i,j-1}.$$

Using the received value $y_{i,j}$ and estimates of the decoded data $x'_{i-1,j}$ and $x'_{i,j-1}$, we could estimate $x_{i,j}$ by

$$x'_{i,j} = y_{i,j} - .5x'_{i-1,j} - .5x'_{i,j-1}.$$

Just as in the two-dimensional modulation coding scheme (Section 4.2.3), we could then reconstruct the entire transmitted data array $x_{i,j}$, $1 \leq i,j \leq N$ given the received

data array $y_{i,j}$, $1 \leq i,j \leq N$ and an "L-shaped" set of data values $x_{0,0}, x_{1,0}, \ldots, x_{N,0}$, $x_{0,1}, \ldots, x_{0,N}$. But, as in the two-dimensional coding scheme, a memory buffer would be required.

6 Error-Correction Coding

In this section, we study coding techniques to correct possible errors that may occur after the decoding of the modulation code.

The usual convention is to denote an error-correcting code of length n, dimension k and minimum distance d as an $[n,k,d]$ code. The dimension denotes the number of information (user) symbols, while the the length n denotes the number of encoded symbols for each k information symbols. Therefore, $n-k$ redundant symbols are added to the k information symbols in order to achieve error-correcting capability. This error-correcting capability is determined by the minimum (Hamming) distance d between distinct codewords: the code can correct up to $\lfloor \frac{d-1}{2} \rfloor$ symbols. The symbols can have any size, ranging from single bits to bytes of any length. Moreover, in a two-dimensional environment, they can have any shape (rectangular, square, even circular). For details, the reader is referred, for instance, to [16].

A one-dimensional burst error of length t is a set of errors that are confined to t consecutive locations [25]. However, in holographic storage, errors may tend to be either two or three-dimensional in nature, called clusters. This may result from loss of spatial resolution, defects in the material or limited error propagation from the modulation decoder (even if the channel errors were random, the modulation decoder will very likely propagate errors into a relatively small cluster).

Two-dimensional cluster error-correcting codes have been studied for some time in the literature [3][1][9][12]. In [8][11][27], the authors consider codes that can correct many different types of two-dimensional errors, such as "criss-cross" type of errors. In [17], correction of clusters appears in the context of holographic storage.

We propose that ECC be implemented in three levels. The first level (the inner

level) will address error clusters of relatively small size, which we consider to be the dominant error pattern. The second level (the outer level) will correct larger, more catastrophic errors, and the third level (the inter-page level) will correct errors that affect entire pages. Encoding is performed first on the inter-page level, then on the outer level and finally on the inner level. Decoding is performed in the reverse order.

We assume that each page consists of a $2^{10} \times 2^{10}$ array of bits. Recall from Section 3 that each array can be processed by several independent channels. We will assume that there are 64 such channels and therefore each of these channels processes 2^{14} bits. Although in the description to follow we concentrate on these parameters, the techniques can basically be applied to arrays of any size.

Recall that there are two extreme approaches to configuring the division of the CCD array into parallel channels: the square approach (Figure 2) and the strip approach (Figure 3). In the square approach, the data page is divided into 64 squares of $2^7 \times 2^7$ bits each. In the strip approach, it is divided into 64 strips of size $2^4 \times 2^{10}$ bits each.

For purposes of error-correction, which one of the two approaches is better? It depends on the channel and the type of errors that we want to correct. In the course of our discussion, we will make comparisons between the two approaches. Unless the clusters are too large, we favor the strip approach. We assume that the same approach (strip or square) is chosen for both modulation coding and ECC. However, it may be possible to use the strip approach for modulation coding and the square approach for ECC. This would require more communication among the channels, but may provide an opportunity for using parallel ECC decoding methods, advantages of which are argued by Neifeld and Hayes [23].

6.1 The Inner Level

The inner level of ECC will be confined to a single channel (strip or square). We illustrate this with the square approach. Similar designs can be made for the strip approach. We will make use of the well-known class of BCH codes; see [16], page 589.

Interleaving is a standard method for correcting error clusters. Figure 7 represents

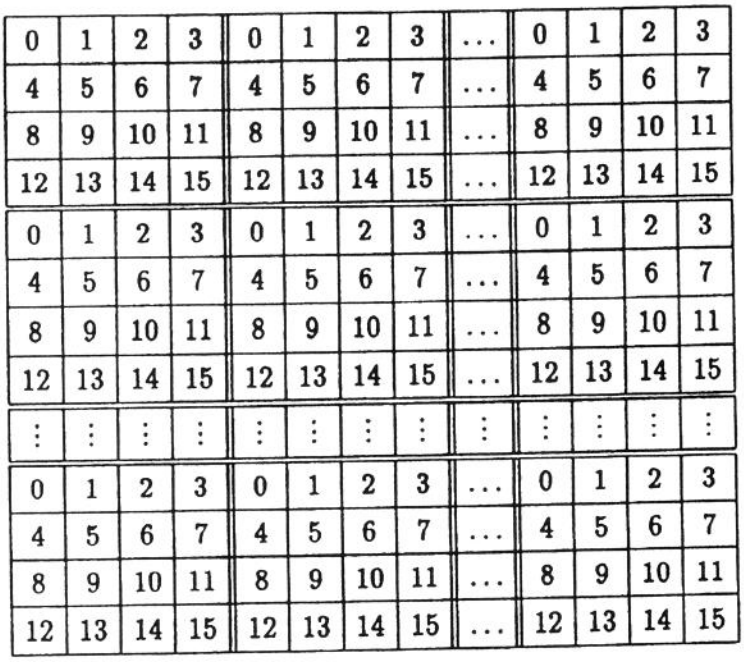

Figure 7: Two-dimensional interleaving of 16 codewords of length 1024 each.

a $2^7 \times 2^7$ square (i.e., one of the 64 channels). Each of the 16 numbers represents a different codeword. We interleave 16 codewords, each of length $2^{10} = 1024$, taken from a BCH code (note that $(16)(1024) = (2^7)(2^7)$). For example, if we use a $[1024, 1013, 4]$ BCH code, then each codeword can correct 1 error. The codewords are decoded independently, and so, within any single channel, any single cluster of size up to 4×4 can be corrected by this approach (since any 4×4 subarray consists of 16 different numbers). The total redundancy will be $(16)(11) = 176$ bits per channel (about 1%). If we want to increase the error-correcting power, for instance, we can use a $[1024, 1003, 6]$ BCH code, which is capable of correcting 2 errors, with a total redundancy of $(16)(21) = 336$ bits per channel (about 2%).

An alternative to BCH codes is to use Reed-Solomon (RS) codes [16]. There are several ways to implement RS codes in a two-dimensional interleaved way. Let us assume that we implement extended RS codes of length 256 over 8-bit bytes, a common standard in the industry. Consider a $2^7 \times 2^7$ square. We can interleave 8 $[256, 254, 4]$ RS codewords as shown in Figure 8.

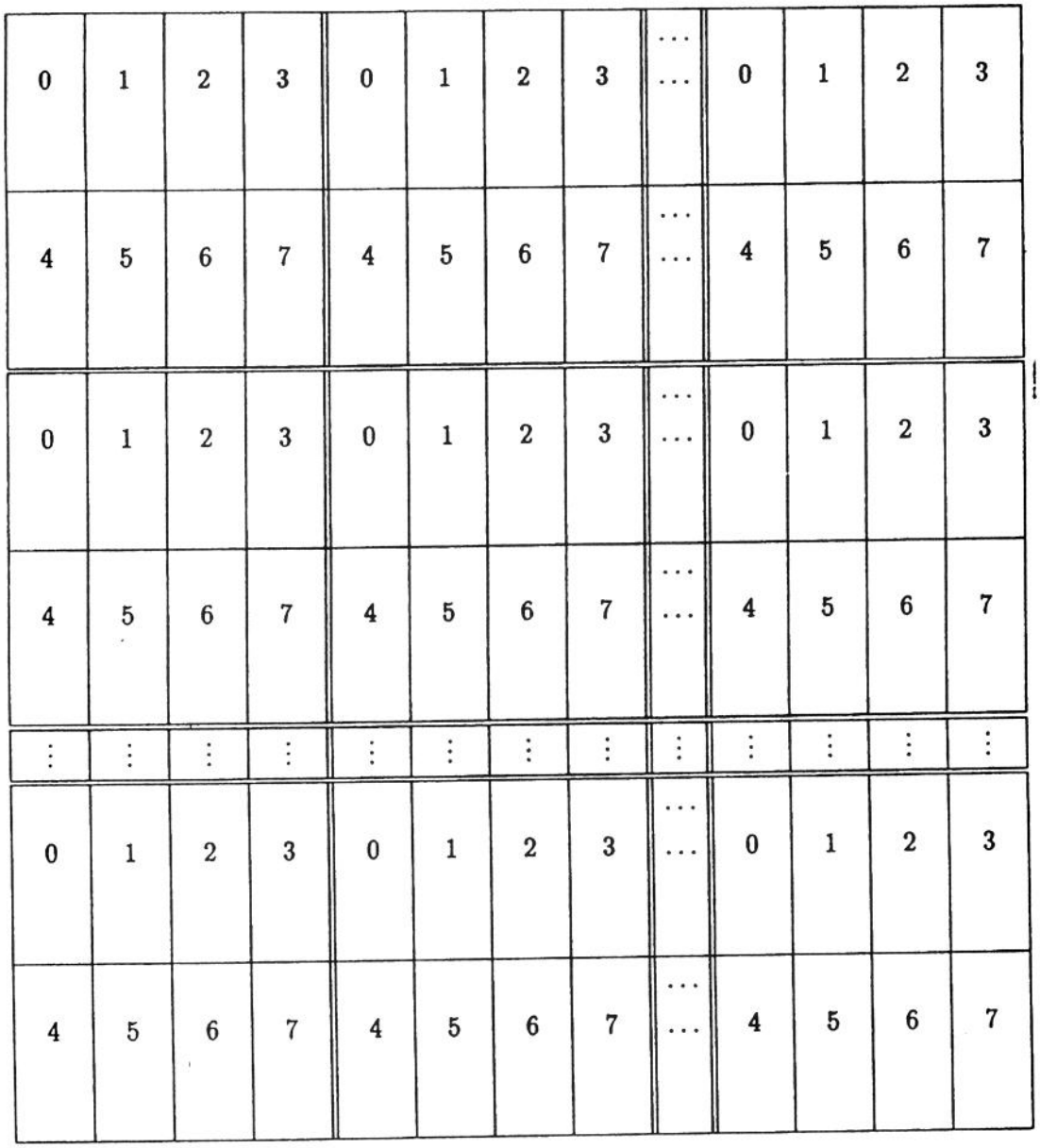

Figure 8: Two-dimensional interleaving of 8 RS codewords of length 256 each over 8-bit bytes.

In Figure 8, each rectangle represents a 4×2-bit byte (i.e., 4 bits by 2 bits). We have 8 different codewords, numbered from 0 to 7. If we implement a $[256, 253, 4]$ RS code, each codeword can correct a 4×2 byte in error. Therefore, we can correct any cluster of size up to 5×7 bits (remember that the scheme using BCH codes of Figure 7 could correct only 4×4 clusters). The redundancy is $(3)(8) = 24$ bytes, or 192 bits. In the BCH scheme the redundancy was 176 bits, so there is a tradeoff here between redundancy against larger cluster-correction.

The codes above are implemented independently for each channel. In this regard, there may be an advantage to the square (as opposed to strip) approach for the (inner level) of ECC. To see this, consider the following two situations: in one situation, there are two error clusters each of which is confined to a single channel – call them channels 1 and 2; in the other situation, one error cluster is confined to channel 1 and the other straddles the boundary between channels 1 and 2. If the inner level ECC is capable of correcting only one error cluster within each channel, then in the first situation we will be able to correct both clusters, while in the second situation we may be able to correct only one cluster. Since the total boundary of the square division is smaller than that of the strip division (roughly because of all rectangles of a given area, the square has the smallest perimeter), we conclude that in the square approach the probability that an error cluster meets more than one channel is lower than in the strip approach. Thus the expected number of clusters that can be corrected will be higher in the square approach.

On the other hand, in the strip approach, even though there may be more error clusters straddling the boundary it may be possible to use a bit more coding and a limited amount of cross-channel communication (communication only between nearest neighbors) to help correct such clusters.

6.2 The Outer Level

While the inner level ECC may be adequate to correct most errors, an outer level ECC may be needed to correct for more severe, but less likely, failures. In this level

of coding, we must assume that the channels can communicate with one another.

An approach with relatively low complexity involves regarding the information within a single channel as a single symbol and the symbols are strung together in a codeword of length 64 (corresponding to the 64 channels). For instance, consider the inner level code depicted in Figure 8 with the 8 RS codewords. In each channel, (8)(253)=2024 of the 8-bit bytes contain information. We consider each set of 2024 information bytes as a single symbol in a code of length 64, using the method in [2][4][5]: the codes given in these references have low complexity since they are based on arrays with simple parity over lines of different slopes (as opposed to the finite field arithmetic required by RS codes). They can also handle very large symbols. One caveat, though, is that the size of the symbols must be a prime number minus one. But here we are lucky: notice that the size of a symbol (in bits) is (2024)(8) = 16192 bits. We can verify that 16193 is a prime number, so symbols of size 16192 will work. We can make two of the 64 symbols redundant, so we have a $[64, 62, 3]$ code over symbols of size 16192 bits. Any symbol in error then will be corrected (say, if one of the 64 channels is defective).

This scheme works fine if the large errors are caused by a channel failure, i.e., if one of the 64 channels fails, then the bits in it are lost. However, the clusters may have a completely different cause, such as a defect in the material. In this case, the cluster will not distinguish between channels and may be spread among them. So, we need to implement an interleaved scheme similar to the one described above within each channel. Here, it is important to distinguish between the square and the strip structure.

If we use the square structure, we interleave as in Figure 9. In this case, we are interleaving 4 $[16, 14, 3]$ codewords as described in, say, [2]. This interleaving scheme can correct any cluster of size up to 128×128. Notice that we are paying a very high price in terms of redundancy: 8 symbols, i.e., $(8)(2^{14}) = 2^{17}$ bits (to this we have to add the redundancy due to the inner code).

On the other hand, if we are using the strip structure, a natural way of interleaving is given in Figure 10. This interleaving scheme allows for correction of a cluster of size

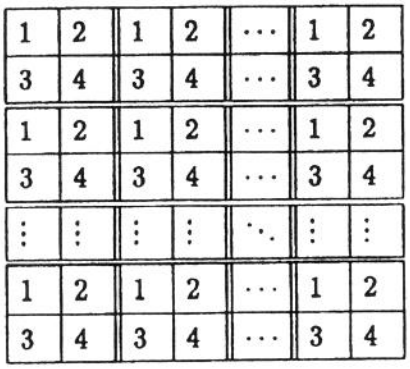

Figure 9: Two-dimensional interleaving of 4 codewords of length 16 each, such that each symbol is a $2^7 \times 2^7$ square.

up to 17×1024. How does this compare with the interleaving scheme of Figure 9? It depends on the size of the cluster. If the size of the cluster does not exceed 17×17, certainly the strip structure is superior, since we need to interleave only two $[32, 30, 3]$ codewords, thus, the redundancy is only 4 symbols, as opposed to 8 symbols with the square approach. Moreover, we could also interleave 3 codewords, say, two $[21, 19, 3]$ codewords and one $[22, 20, 3]$ codeword. In this case, the interleaving scheme can correct a cluster of size up to 33×1024, while the redundancy is 6 symbols. If the clusters do not exceed size 33×33, the strip scheme looks more convenient, since it has less redundancy. However, for clusters larger than this, the square scheme is superior. Again, the codes used for these schemes can be found in [5] or in [4].

We can also use RS codes to correct large clusters. Consider the square structure (a similar treatment can be made for strips). We divide each square into 2048 4×2-bit bytes, similar to the case of the inner level code in Figure 8. If we use the inner code structure of Figure 8, of these 2048 bytes, 2024 of them carry information. Each of the 2024 information bytes represents a different codeword over a $[64, 62, 3]$ RS code (whose symbols are 4×2-bit bytes). Two squares are redundant, as in the first outer level scheme described. Notice that this scheme can correct any cluster of size up to $(2^7 - 3) \times (2^7 - 1)$ (even clusters that straddle the boundaries of the squares). Therefore, in terms of redundancy, it is more efficient than the schemes with large symbols described above. However, a possible disadvantage of this scheme is that it

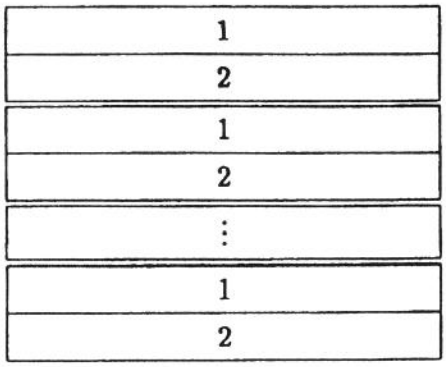

Figure 10: Interleaving of 2 codewords of length 32 each, such that each symbol is a $2^4 \times 2^{10}$ rectangle.

requires us to implement a very large number (2024) of different RS codewords.

6.3 The Inter-Page Level

A third cascaded code can be implemented on whole pages: as the number of pages in a stack grows, the odds that one or more pages are defective increases. Therefore, we may implement a code at the page level, which can correct one or more pages in error. That was the approach taken in [15], where the authors implement a $[12, 8, 4]$ Hamming code bit by bit. This code allows for correcting up to one page in error and detecting two every 12 pages. A more efficient implementation with less redundancy would be to implement $[12, 9, 4]$ RS codes byte by byte (the bytes here can even have size 4 bits): therefore, we are saving one page for data instead of redundancy, and the error-correcting capability is the same. Another possibility is to take the information bytes in each page as one large symbol and use the codes in [5] or in [4], as in the previous subsection. Each page is divided into units of a fixed size, called the granularity. This granularity can be single bits, blocks of various sizes, even whole sectors. By adjusting the granularity, in general we can assure that the size of a page (i.e., the number of units in it) is a prime number minus one, a necessary condition to apply the codes in [5] or in [4]. These codes are also $[12, 9, 4]$ codes, but the symbols are entire pages, and the complexity will be much less than the RS codes above.

There are many more ways of combining codes that can correct errors in holographic storage. The right combination will depend on the error statistics, which are not clear at this stage.

References

[1] C. de Almeida and R. Palazzo, *Two-Dimensional Interleaving Using the Set Partioning Technique,* IEEE Information Theory Symposium, June 1994.

[2] M. Blaum, J. Brady, J. Bruck, and J. Menon, *EVENODD: an optimal scheme for tolerating double disk failures in RAID architectures,* IEEE Trans. on Computers, vol. C-44, pp. 192–202, 1995.

[3] M. Blaum and J. Bruck, *Correcting of Two-Dimensional Clusters by Interleaving of Symbols,* IEEE Information Theory Symposium, June 1994.

[4] M. Blaum, J. Bruck, and A. Vardy, *MDS array codes with independent parity symbols,* IBM Research Report, RJ 9887 (87267), Sept. 1994.

[5] M. Blaum and R.M. Roth, *New array codes for multiple phased burst correction,* IEEE Trans. on Information Theory, vol. IT-39, pp. 66–77, 1993.

[6] D. Brady and D. Psaltis, J. Opt. Soc. Am. A 9, 1167 (1992).

[7] W. J. Burke and P. Sheng, J. Appl. Phys. 48, 681 (1977).

[8] P. Delsarte, "Bilinear forms over a finite field, with applications to coding theory," J. Comb. Th. A, 25 (1978), 226–241.

[9] F. G. Farrell, *Array Codes for Correcting Cluster-Error Patterns,* IEEE Conf. on Elect. Signal Processing, York, July 1982.

[10] E. M. Gabidulin, *Combinatorial Metrics in Coding Theory,* 2nd. Int. Symposium on Inf. Theory, Budapest, pp. 169–76, 1971.

[11] E. M. Gabidulin, *Theory of Codes with Maximum Rank Distance,* Probl. Inform. Transm., vol. 21, No. 1, pp. 3–16, Jan.-Mar. 1985.

[12] E. M. Gabidulin and V. V. Zanin, *Codes Correcting Array Bursts,* The Workshop on Information Protection, Proc., pp. 24-25, Moscow, 1993.

[13] C. Gu, J. Hong, I. McMichael, R. Saxena, and F. Mok, J. Opt. Soc. Am. A 9, 1978 (1992).

[14] J. F. Heanue, M. C. Bashaw, and L. Hesselink, *Channel codes for digital holographic data storage*, preprint, 1995.

[15] J. F. Heanue, M. C. Bashaw, and L. Hesselink, *Volume holographic storage and retrieval of digital data,* Science 265:749-752 (1994).

[16] S. Lin and D. J. Costello, *Error-Control Coding: Fundamentals and Applications,* Prentice-Hall, NJ, 1983.

[17] S. A. Lis and P. D. Henshaw, *Ultra-Dense Optical Mass Storage,* Report prepared for USAF, AFSC, AD-A232 767, 1991.

[18] E.S. Maniloff and K. M. Johnson, J. Appl. Phys., 70, 4702 (1991).

[19] M. Marcellin and H. Weber, *Two-dimensional modulation codes,* IEEE J. Selected Areas on Communications, 10 (1992) 254–266.

[20] B. H. Marcus, P. H. Siegel, and J. K. Wolf, *Finite-state modulation codes for data storage,* IEEE J. Selected Areas on Communications, 10 (1992) 5–37.

[21] N. Markley and M. Paul, *Maximal measures and entropy for $\mathbf{Z}^\nu$ subshifts of finite type,* Classical Mechanics and Dynamical Systems, Dekker Notes 70, ed. by R. Devaney and Z. Nitecki, 135–157.

[22] F. H. Mok, Opt. Lett. 18, 915 (1993).

[23] M. A. Neifeld and J. D. Hayes, *Parallel Error Correction for Optical Memories,* Optical Memory and Neural Networks, Vol. 3, No. 2, pp. 87–98, 1994.

[24] M. A. Neifeld and M. McDonald, *Error correction for increasing the usable capacity of photorefractive memories,* Opt. Lett. 19, 1483 (1994).

[25] T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems,* Prentice-Hall, NJ, 1989.

[26] R. M. Robinson, *Undecidability and nonperiodicity for tilings of the plane,* Inventiones Math., 12 (1971), 177–209.

[27] R. M. Roth, *Maximum-Rank Array Codes and their Application to Crisscross Error Correction,* IEEE Trans. on Information Theory, IT-37, pp. 328–36, March 1991.

[28] K. Schmidt, Algebraic Ideas in Ergodic Theory, AMS–CBMS Reg. Conf. **76**, 1990

[29] G. Sincerbox, *Holographic storage re-visited,* Current Trends in Optics, ed. by C. Dainty, Academic Press, 1994.

[30] A. Vardy, M. Blaum, P. Siegel, and G.Sincerbox, *Conservative Arrays: Multi-Dimensional Modulation Codes for Holographic Recording,* IBM Research Report, RJ 9883 (86130), Sept. 1994.