This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

Coding for Correcting Insertions and Deletions in Bit-Patterned Media Recording

Anantha Raman Krishnan, and Bane Vasic, *Senior Member, IEEE*

***Abstract*—Bit-patterned media is a novel technology for magnetic data storage that is poised to increase recording density beyond 1 Tb/sq. in. However, a significant concern in BPMR is the stringent requirements for synchronization between write clock and island position, errors in which may manifest as insertions and deletions. In this paper, we introduce a method for compensating for synchronization errors by using conventional error-correcting codes. We present a numerical study that provides bounds on achievable coding rates. We also perform a simulation study to demonstrate the applicability of the proposed scheme in practical systems.**

I. Introduction

In the past few years, many novel technologies have been proposed to increase storage densities of magnetic recording systems over 1 Tb/in^2, bit-patterned media recording (BPMR) [1] being the most promising of them. However, BPMR presents challenges that were not a concern in conventional magnetic recording systems, *e.g.*, write synchronization [2]. In BPMR, a high degree of synchronization is required between the write clock and island positions. Errors in synchronization may manifest as insertions or deletions in the bit-stream read out [3]. It is vital that robustness to synchronization errors be maintained. Such robustness can be achieved by using codes capable of correcting insertion and deletion errors, subsequently compensating for written-in errors during the readback. This motivates the investigation of the nature of information transmission in channels with synchronization errors.

For a number of years, characterization of channels with synchronization errors has been an active field of research (e.g. [4] and [5]). In the recent past, this has been further driven by aforementioned systemic requirements in BPMR. However, such channels are are not characterized fully, with only bounds available on capacity of channels (see [6] and references therein for a detailed discussion). Another related area of research has been the design of systems that compensate for insertion and deletion errors. Levenshtein [4] provided a number-theoretic construction of codes capable of correcting a single error (synchronization error or error in bit-value). Although a simple decoding algorithm for this class of codes exists [7], these codes are not systematic, thereby making implementation non-trivial for all but the smallest-length codes. Davey and MacKay [8] proposed watermark codes with a probabilistic decoding algorithm operating on a two-dimensional (2D) trellis for error-correction in insertion/deletion channels. Subsequently, marker codes based on a similar decoder was proposed by Ratzer [9]. These schemes, though observed to be powerful, rely on complex decoding algorithms which are not conducive to system-level implementation. More recently, Ng *et al.* [3] developed a scheme based on 2D error-correcting codes for insertion/deletion errors, in the context of BPMR. Again, implementation in real-life systems necessitates multi-track processing – a considerable challenge with the current state of the art in the technology.

One of the difficulties in addressing the problem of code-construction is the fact that channels with synchronization errors have infinite memory, i.e., a synchronization error affects all subsequent symbols. In this paper, we introduce a coding scheme by which it is possible to transform certain channels with synchronization errors into memoryless channels. This naturally leads to a information encoding scheme by which conventional error-correcting codes may be used to compensate for synchronization errors. In addition, this method is advantageous as it lends itself to single-track processing. We also provide a brief numerical study of the capacity study of this class of channels, and show some results of use of error-correcting codes on these channels.

The rest of the paper is organized as follows: Section II introduces our channel model and data encoding scheme. A numerical analysis of channel capacities obtained is provided in Section III. In Section IV, we show results of our experiments conducted to prove the applicability of our methodology. Finally, Section V concludes the paper.

II. Channel Model

In [3], Ng et al. illustrate how jitters in island positions in BPMR may manifest as synchronization errors, with errors occurring in arbitrary positions. In this section, we give a brief description of a modified synchronization-error channel we consider, and explain how information may be transmitted through this channel.

Consider a channel with binary input and output alphabets. The most general formalism that describes synchronization and bit-flip error is a finite-state machine [8]. At time t_i a bit may be inserted with a probability p_i, deleted with a probability p_d, or successfully transmitted with a probability $1 - p_i - p_d$. The possibility of a change in the value of the transmitted bit can also be encompassed by this model.

In this work, we consider a modification of the channel described above wherein the number of consecutive synchronization errors is restricted, and operates on individual runs of binary sequences. Here, we define the term *run* in a binary sequences as an occurrence of k consecutive, identical symbols.

A. R. Krishnan, and B. Vasic are with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ, 85721 USA e-mail: {ananthak, vasic}@ece.arizona.edu. This work is supported by INSIC-EHDR and IDEMA-ASTC

We can now define a channel with two additional parameters s_i, and s_d, where s_i (s_d resp.) is the maximum number of consecutive insertions (deletions resp.). The probability of j consecutive insertions (deletions resp.) is $(p_i)^j$ ($(p_d)^j$ resp.) for $j = 1, \ldots, s_i$ (s_d resp.). In each run of zeros or ones, only one of the $s_i + s_d + 1$ error events occurs, namely, (i) error-free transmission, (ii) insertion of j bits, $j = 1, \ldots, s_i$, or (iii) deletion of j bits, $j = 1, \ldots, s_d$. We do not consider errors in bit-values introduced by channel. In general, these parameters may be chosen to match the behavior of recording media. For modeling insertions, we use the sticky keyboard channel model [10]. In this model, the bit inserted by $\mathcal{C}$ is identical to the previous bit.

To further illustrate the behavior of the channel, consider as example a channel with parameters $(p_i, p_d, 2, 2)$. Let b_l denote to the l^{th} bit with value b, $b \in \{0, 1\}$. Let b_i represent an bit inserted, and with value b. Similarly, let $\sqcup_l$ represent a deleted bit at position l (note that the receiver cannot identify the position or value of an inserted/deleted bit. These indices are for ease of illustration only).

Suppose that the bitstream $[0_1, 0_2, 0_3]$ is transmitted through this channel. Then, the bit-streams $[0_1, \sqcup_2, 0_3]$ and $[0_1, 0_2, 0_i, 0_i, 0_3]$ is an example of a valid output since it has no more than two consecutive insertions or deletions. However, $[0_1, 0_i, 0_i, 0_i, 0_2, 0_3]$, is not a valid output since there are more than 2 consecutive insertions. Similarly, $[0_1, 1_i, 0_2, 0_3]$ is not a valid output since the inserted bit in this case is not identical to the previous bit.

A. Information Transmission

Before we describe a method for reliable information transmission through the class of channels described above, it is instructive to introduce the simple lemma that follows:

Lemma 1: Consider a channel $\mathcal{C}$ with parameters (p_i, p_d, s, s), having input $\mathbf{x}$. If all the runs in $\mathbf{x}$ have lengths greater than s, then the number of runs in any output $\mathbf{y}$ produced by $\mathcal{C}$ is equal to the number of runs in $\mathbf{x}$.

Proof: First, suppose that $\mathbf{x} = [b_1, \ldots, b_{s_t}], b \in \{0, 1\}$, $s_t > s$. The input $\mathbf{x}$ has exactly one run of s_t bits with value b. Three cases arise: (i) error-free transmission, (ii) one or more insertions, (iii) one of more deletions.

In the case of error-free transmission, output also consists of exactly one run. Next, suppose that there are j deletions during transmission through $\mathcal{C}$, $j = 1, \ldots, s$. The corresponding resultant output of $\mathcal{C}$ is then $\mathbf{y} = [b_1, \ldots, b_j]$, $j = s_t - 1, \ldots, s - s_t$, respectively. Since, $s_t > s$, $\mathbf{y}$ also consists of exactly one run of bit valued b.

Finally, if there are j insertions during transmission, $j = 1, \ldots, s_i$, then the corresponding output $\mathbf{y}$ is $[b_1, \ldots, b_j]$, $j = s_t + 1, \ldots, s_t + s$, respectively (this can be inferred by noting that the inserted bit is identical to the previous bit). In this case as well, the number of runs in $\mathbf{y}$ is one.

This analysis can be extended to inputs with multiple run. To do this, segment the inputs into runs (each, by hypothesis, having length greater than s), and subsequently analyze them individually. The correctness of this analysis arises from the fact that in each run, only one insertion/deletion error-event may occur. ■

Summarizing, when transmitted through $\mathcal{C}$ described in Lemma 1, each run of s_t bits, $s_t > s$, results in a run of j bits, $j = s_t - s, \ldots, s_t + s$. This observation leads naturally to an encoding scheme where symbols are encoded in runs of bits. To explain this, consider a channel with $s_i = s_d = 1$. Algorithm 1 illustrates how information symbols (here binary) may be encoded in runs of channel bits.

Data: Information bits: $b_l,\ l = 1 : n$
Result: Channel bitstream: c
Set current index $i = 1$; channel bitstream $c = [\ ]$;
for $l = 1$ *to* n **do**
  if $b_l = 0$ **then** `/* construct a run consisting of bits` c_i `through` c_{i+1} `*/`
    $c_{i:i+1} = l \mod 2$;
    $i = i + 2$
  else `/* construct a run consisting of bits` c_i `through` c_{i+2} `*/`
    $c_{i:i+2} = l \mod 2$;
    $i = i + 3$
  end
end

Algorithm 1: Encoding binary information in runs of channel bits ($s_i = s_d = 1$)

To illustrate the algorithm, consider the information bits $b = [0, 1, 1, 0]$. The first bit b_1 is 0. To encode b_1, a run consisting of *two* bits, namely c_1 through c_2, is added to c. This can be done by setting $c_1 = c_2 = 1 \mod 2 = 1$ ($c = [11]$). Next, since b_2 is 1, we construct a run consisting of *three* bits, namely, c_3 through c_5, is added to c. This can be done by setting $c_4 = c_5 = c_6 = 2 \mod 2 = 0$ ($c = [11\ 000]$). Proceeding thus, we find that b is encoded as channel bits [11 000 111 00]. In essence, symbols with odd indices are encoded in runs of k 1's, with $k = 2$ for symbol 0 and $k = 3$ for symbol 1. Symbols with even indices are similarly encoded by runs of 0's.

At the receiver, the length of each run is counted in order to determine the output symbol. It must be noted that at the receiver, the lengths of runs may change due to insertions or deletion. Nevertheless, by virtue of Lemma 1, a correct choice of runlengths (more precisely, by choosing runlengths greater than s) assigned to information symbols can lead to the i^{th} information symbol corresponding exactly to the i^{th} run of channel bits.

An important consequence of such an encoding scheme is that transmission through the synchronization-error channel can now be represented as that of transmission through a memoryless channel. That is, any synchronization-error manifests as an error in the corresponding output symbol, and does not affect other symbols. Thus, classical error-correcting codes can be used to compensate for such errors.

The encoding scheme described in Algorithm 1 results in a discrete memoryless channel as illustrated in Fig. 1(a). The input symbols denote the runlengths corresponding to the input bits. The output of the channel correspond to the all the possible values of runlengths at the receiver.

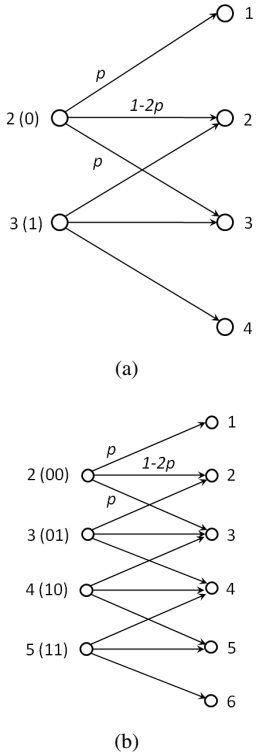


Fig. 1. Transmission of information through a synchronization error channel with parameters $(p, p, 1, 1)$. The inputs are marked with the lengths of the runs, with the encoded symbols marked within parentheses. The outputs correspond to all possible lengths of runs at the receiver: (a) binary information, and (b) quaternary information.

It is easy to see that this methodology is not restricted to encoding of binary-valued sequences, and can be easily extended to larger alphabets. Figure 1(b) shows an example of transmission of information with alphabet sized four.

In order to determine bounds on the achievable rate of error-correcting codes that may be used, we need to calculate the capacity of these channels.

III. Capacity Estimates

In this section, we investigate the capacity limits of channels described above.

Traditionally, the capacity of memoryless channels is obtained by maximizing the mutual information, $I(\mathbf{X};\mathbf{Y})$, between the input alphabet $\mathbf{X}$ and the output alphabet $\mathbf{Y}$, over all input distributions of $\mathbf{X}$. Formally, classical capacity of a channel C is defined as:

$$C = \max_{p(\mathbf{X})} I(\mathbf{X};\mathbf{Y}). \tag{1}$$

This definition assumes that the costs of transmission of different symbols are equal. However, this is not true for the channel formulated in Section II. For instance, in the case of the channel shown in Figure 1(a), the cost of transmitting the information-bit 0 is 2, since 2 channel bits are used for transmission. Similarly, the cost of transmitting the information-bit 1 is 3. In such channels, we use the notion of capacity *per unit cost* [11] (also known as unit-cost capacity).

Formally, for a cost function $c(x)$, defined for each element x in the input alphabet $\mathbf{X}$, the unit-cost capacity is defined as:

$$C_{\text{unit}} = \max_{p(X)} \frac{I(\mathbf{X};\mathbf{Y})}{E[c(\mathbf{X})]}, \tag{2}$$

where $E[c(\mathbf{X})]$ is the expectation of the cost-function, and is defined as

$$E[c(\mathbf{X})] = \sum_{x\in\mathbf{X}} p(x)c(x). \tag{3}$$

It can be seen that in our transmission model, the unit-cost capacity translates to capacity per channel-use for the transmission scheme under consideration. The capacity is a function of the synchronization error probability as well as the size of the input alphabet $\mathbf{X}$.

By changing the alphabet-size at the input, a spectrum of transmission schemes can be obtained. In general, for a channel $\mathcal{C}$, with information alphabet $\mathbf{X}$, we can calculate an associated unit-cost capacity $C(\mathcal{C}, |\mathbf{X}|)$. These capacities constitute lower bounds on the capacity of $\mathcal{C}$.

In this paper, we consider two classes of channels, the first with parameters $(p, p, 1, 1)$ (henceforth referred to as Channel 1), and $(p, p, 2, 2)$ (Channel 2). We choose alphabets of size 2, 4, 8, and 16. For alphabet size $|\mathbf{X}|$, the costs of transmission of were chosen as $2, \ldots, |\mathbf{X}| + 1$.

Remark: Several choices for selection of run-lengths (cost-function) exist. In this study, we chose one logical choice given above. Other choices considered, e.g., costs ranging from 2 through $2|mathbfX|$, yield significant reduction in the capacity. An in-depth study of optimal choices is beyond the scope of this paper.

Figure 2(a) and (b) show the capacity estimates of Channels 1, and 2, respectively, for various values of p and alphabet size $|\mathbf{X}|$.

Firstly, we observe the capacity increases with the size of the alphabet. It is worth noting that since the costs are not equal for all symbols, the maximizing input distribution assigns different probabilities to each symbol. In particular, symbols with low transmission-cost have a higher probability than those with high cost. Another interesting feature is that after a certain threshold, the capacity increases with p. This can be attributed to the fact that beyond the aforementioned threshold, the conditional entropy $H(\mathbf{X}|\mathbf{Y})$ reduces. For instance, for the channel $(0.5, 0.5, 1, 1)$ transmitting binary-valued information, $H(\mathbf{X}|\mathbf{Y})$ is zero. This is because an input with runlength of 2 produces an output with runlength of 1 or 3. Similarly, an input with runlength of 3 produces an output with runlength 2 or 4. Since there is no ambiguity in the input given the output, $H(\mathbf{X}|\mathbf{Y})$ is zero. Finally, we find that for a fixed alphabet size, the capacity of the Channel 1 is lower than that of the Channel 2.

IV. Error Correction Codes

In order to demonstrate the feasibility of implementation of our encoding methodology, we conducted experiments to simulate transmission of coded information through Channel 1 and 2.

Figure 3 illustrates the system model. As shown in the figure, the first step is to transform the distribution of the

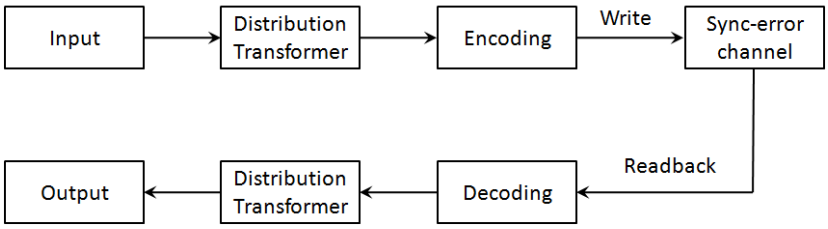
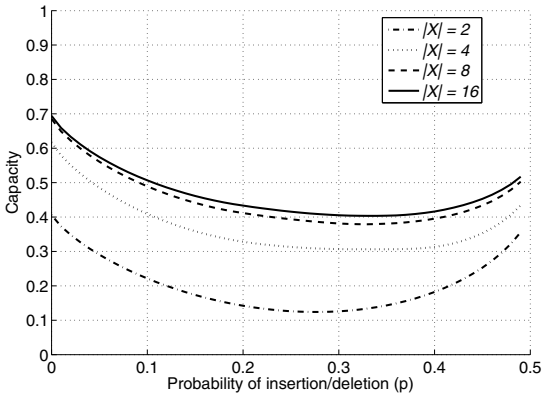
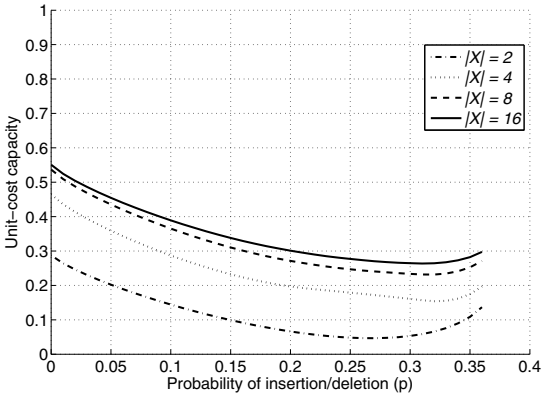


Fig. 3. System model of information transmission



(a)



(b)

Fig. 2. Capacity of various transmission schemes: (a) Channel 1, and (b) Channel 2

information symbols. This step is necessary since the capacity-achieving distribution of the channel-input may be different from that of information symbols (usually uniformly distributed). Such a transformation is readily achieved by employing arithmetic decoder, which is widely used in source coding applications (c.f. [12]). The data is then encoded by using an error-correcting code. Subsequently, the information symbols are encoded in runs of channel bits as described previously. At the output, all the operations are performed in reverse in order to obtain an estimate of the input data.

A. Results

We considered Channel 1 described above, with transmission of binary-valued information (Fig. 1(a)). For our simulations, we chose synchronization-error probability (p) ranging from 0.005 to 0.1.

Three codes were chosen as candidates for simulation. The codes were constructed by methods described in [13], and provide guarantees on error-correction capability under iterative error-correction decoding for BSC. The first code (henceforth referred to as Code 1) was a short-length code with length 190, and rate 0.6. Since the experiments were primarily demonstrative, no attempt at transforming the input distribution were made, i.e., the inputs were uniformly distributed. Thus, the effective code-rate (the number of information bits transmitted per channel use) of the code was calculated as 0.24, which is about 60% of capacity (refer Fig. 2(a)) at the range of p simulated. The second code chosen (Code 2) was of length 530, with an effective code-rate of 0.24. The third code (Code 3) had length 1545, and a rate of 0.8 (effective rate = 0.32; $\sim 80\%$ of capacity). Sum-product algorithm was used for decoding the codes.

Remark: It must be noted that the use of distribution transformers incurs a further loss in the effective rate. However, a detailed discussion of this is beyond the scope of this paper.

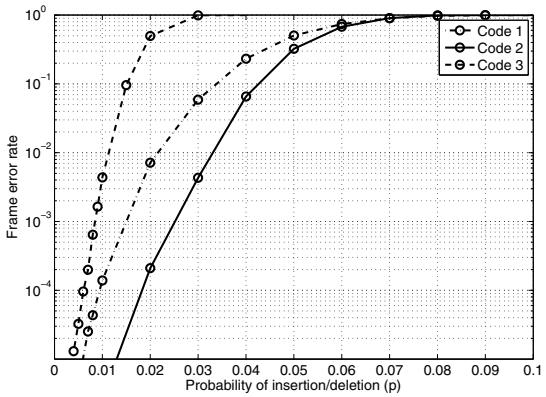


Fig. 4. Performance of LDPC-coded data on Channel 1.

Figuure 4 shows the frame-error rates with respect to the parameter p for Channels 1. As can be seen, even with short- and moderate-length codes, very good frame error-rate performance can be achieved. For a longer-length code (Code 3) working at about 80% of the capacity, good performance was achieved at lower synchronization error probabilities ($\sim$ 0.005). Although effective rates of codes for binary alphabet are low, by using codes over higher alphabets superior guarantees on maximum achievable rates may be achieved.

V. Conclusion

In this paper, we introduced a modified synchronization-error channel for modeling BPMR applications. We proposed an encoding methodology by which conventional error-correcting codes may be used to compensate for synchronization errors. Capacity studies indicate that with large alphabets (8 or 16), our methodology may find application magnetic recording systems.

References

[1] B. Terris *et al.*, "Patterned media for future magnetic data storage," *Microsystem Technologies*, vol. 13, no. 2, pp. 189–196, Nov. 2006.

[2] Y. Tang, K. Moon, and H. J. Lee, "Write synchronization in bit-patterned media," *IEEE Trans. on Magn.*, vol. 45, no. 2, pp. 822–827, Feb. 2009.

[3] Y. Ng, B. V. K. V. Kumar, K. Cai, S. Nabavi, and T. C. Chong, "Picket-shift codes for bit-patterned media recording with insertion/deletion errors," *IEEE Trans. on Magn.*, vol. 46, no. 6, pp. 2268–2271, Jun. 2010.

[4] V. Levenshtein, "Binary codes capable of correcting deletions. insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, 1966.

[5] J. D. Ullman, "On the capabilities of codes to correct synchronization errors," *IEEE Trans. on Inf. Theory*, vol. 13, no. 1, pp. 95–105, 1967.

[6] J. Hu, T. M. Duman, M. F. Erden, and A. Kavcic, "Achievable information rates for channels with insertions, deletions and intersymbol interference with i.i.d inputs," *IEEE Trans. on Comm.*, vol. 58, no. 4, Apr. 2010.

[7] N. J. A. Sloane, "On single-deletion-correcting codes," in *Ohio State University*, 2000, pp. 273–291.

[8] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions and substitutions," *IEEE Transactions on Information Theory*, vol. 47, pp. 687–698, 2001.

[9] E. A. Ratzer, "Marker codes for channels with insertions and deletions," *Annals of Telecommunications*, 2005.

[10] M. Mitzenmacher, "Capacity bounds for sticky channels," *IEEE Transactions on Information Theory*, vol. 54, no. 1, Jan. 2008.

[11] S. Verdú, "On Channel Capacity per Unit Cost," *IEEE. Trans. Inform. Theory*, vol. 36, no. 5, pp. 1019–1030, Sep. 1990.

[12] B. Vasic, O. Milenkovic, and S. McLaughlin, "Scrambling for nonequiprobable signalling," *IEEE Electronics Letters*, vol. 32, no. 17, pp. 1551–1552, Aug. 1996.

[13] D. V. Nguyen, B. Vasic, M. W. Marcellin, and S. K. Chilappagari, "Structured ldpc codes from permutation matrices free of small trapping sets," in *Information Theory Workshop (ITW 2010)*, 2010.