Enhanced decoding by error detection on a channel with correlated 2-dimensional errors.

Pål Ellingsen[*]
The Selmer Center
University of Bergen
5020 Bergen, Norway
e-mail: paale@ii.uib.no

Øyvind Ytrehus
The Selmer Center
University of Bergen
5020 Bergen Norway
e-mail: oyvind@ii.uib.no

Paul Siegel
Center for Magnetic Recording Research
University of California
5020 Gilman Drive
92037 La Jolla, USA

Abstract — **We apply principles from digital image correction to enhance the correction of two-dimensionally correlated unidirectional errors on a two-dimensional grid system.**
A restoration technique presented in [1], based on Markov random fields, is used to find an estimate of the error pattern. This estimate then in turn provides *a priori* information for use in a soft decoder for the actual code (e.g. LDPC decoder).

I. Channel model

We will study encoded information on a two dimensional lattice with 2D correlated errors. Further, we will assume that errors are asymmetric so that only the transition $0 \longrightarrow 1$ occurs in a received codeword.

Definition 1 (Matrix OR) *Assume A and B are matrices with dimensions $d_1 \times d_2$ where $d_1 \cdot d_2 = n$, and with coordinates a_i and b_i respectively. Then the OR of these matrices is defined as*

$$A \vee B \triangleq a_i \vee b_i, \qquad 1 \le i \le n$$

The received word Y can then be defined as the combination of

$$Y = C \vee X$$

Y - received word
C - original codeword
X - error pattern
$\vee$ - OR operator on matrices as defined above

The channel model can also be illustrated as a two dimensional grid with black and white squares denoting 1 and 0 bits respectively, as seen in Fig. 1.

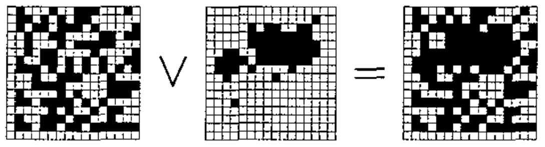


Fig. 1: Graphic description of channel model

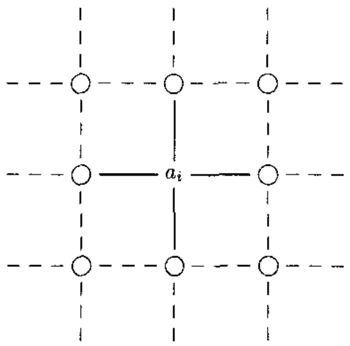
[*]This research was supported by the RCN (Research Council of Norway) under grant 157668/432

II. Error model

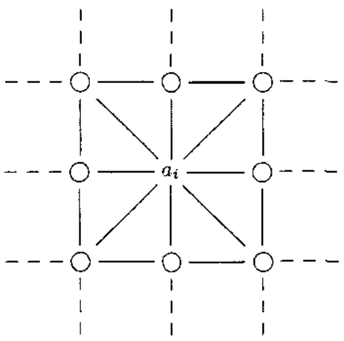
We will use an error model with correlated errors. The errors are assumed to be correlated in the sense that the value of a bit in the error pattern depends on the values of its neighboring bits. To model the correlation between the bits, we will use a Markov Random Field (MRF), and this implies that the Gibbs distribution gives the statistical properties of the errors. We are studying a system where local dependencies in a 2-dimensional space are very important. To describe such dependencies, we will introduce the concept of neighborhoods and the related concept of cliques.

A. Neighborhoods and cliques

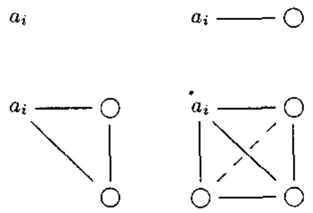
Consider a set of random variables $A = \{A_i | i \in I\}$ for some index set I, where the variables are organized in a two dimensional grid. Let the variables correspond to the vertices and the statistical dependencies between the variables correspond to edges in an undirected graph G. We shall use this setup to model both codewords and errors in our system. Two connected vertices in G are said to be *neighbors*, and a *neighborhood* $\mathcal{N}_i$ of a vertex a_i can be defined as the set of vertices that are connected to a_i in G. Different sizes of neighborhoods can be defined for an MRF. By convention, a node is not a neighbor of itself. On a regular lattice we define the first order neighborhood to be the four closest neighbors of a node as seen below, the second order neighborhood as the eight closest neighbors and so on. The collection of all neighborhoods $\mathcal{N} = \{\mathcal{N}_i \mid \forall i \in I\}$ in a graph, is called a *neighborhood system.*



Within the neighborhood of a vertex a_i, we define a *clique* to be any collection of vertices that contains a_i and forms a fully connected subgraph of G, i. e. that the vertices are mutual neighbors relative to the neighborhood system $\mathcal{N}$. In the case of a first order neighborhood, all nodes within distance 1 of the center are said to be neighbors, and the cliques become the center node a_i and all pairs of (a_i, a_j) where a_j is a neighbor of a_i. In a second order neighborhood, all nodes within distance $\sqrt{2}$ are defined as neighbors:



and in this case the cliques becomes any configuration of



The collection of all cliques of size i in a neighborhood system $\mathcal{N}$ is called $\mathcal{C}_i$. The set $\mathcal{C}$ of all cliques in a graph can then be partitioned into the subsets $\mathcal{C}_i$ for $1 \leq i \leq n$

B. Markov random fields (MRF)

An MRF can be seen as a generalization of Markov chains, but while a Markov chain is often defined over a domain of time as a sequence of random variables, an MRF can be defined in space to describe dependencies between variables on a grid of dimension 2 or higher.

Just as a Markov chain $\{\dots, a_k, a_{k-1}, a_{k-2}, \dots\}$ satisfies

$$P(a_i|a_{i-1}, a_{i-2}, \dots) = P(a_i|a_{i-1}, a_{i-2}, \dots, a_{i-n})$$

for some n, an MRF should satisfy

$$P(a_i|a_{I-\{i\}}) = P(a_i \mid \mathcal{N}_i)$$

where I is the set of indices of a and $\mathcal{N}_i$ is the neighborhood of a_i as defined above.

In the following we shall use an MRF defined over a second order neighborhood system to model the 2-D correlated errors.

C. Probability distribution

The fact that the errors of our channel can be represented by an MRF does not immediately enable us to analyze the error patterns statistically. By assuming that the dependencies in a collection of random variables can be represented by an MRF, the joint probability of the variables is given by the so called *Gibbs distribution.*

Definition 2 (Gibbs distribution) *A set of random variables is said to be a* Gibbs random field *(GRF) if the joint distribution of the variables takes the following form:*

$$P(X = x) = \frac{1}{Z} \exp\left[-\frac{1}{T} U(X)\right] \tag{1}$$

This distribution is called a Gibbs distribution.

- Z is a constant called *the partition function* and can be expressed as $Z = \sum_{x \in \mathbb{X}} e^{-\frac{1}{T}U(x)}$, so that Z^{-1} becomes a normalizing constant in the expression.
- $U(x)$ is called *the energy function* and is a function of the values of the variables forming cliques in the field. It can be written as

  $$U(x) = \sum_{c \in C} V_c(x)$$

  We can expand this expression further by summing over the cliques of the same degree separately

  $$\sum_{c \in C} V_c(x) = \sum_{a \in C_1} V_1(a) + \sum_{a,b \in C_2} V_2(a,b) + \sum_{a,b,c \in C_3} V_3(a,b,c) + \dots$$

  where C_i is the collection of all cliques of degree i, so that V_i is a function of i variables forming a clique, and $\sum_{C_i} V_i$ mean that we sum over all possible cliques in the field of degree i.
- T is called the *temperature* (this is a legacy from the distribution's origin in statistical physics). The parameter T influences the degree of cohesion between the variables on a grid, so that a higher temperature corresponds to a lower degree of cohesion in the sense that the values of the variables becomes more and more independent, while a lower temperature gives a higher probability of the formation of large clusters of variables with the same value. We shall assume that the temperature is 1 in our simulations, even if the parameter will be used in the theoretical treatment of the decoding algorithm.

The Clifford-Hammersley theorem states that for a set of variables $\mathcal{F}$ with a neighborhood system $\mathcal{N}$, $\mathcal{F}$ is an MRF with respect to $\mathcal{N}$ if and only if $\mathcal{F}$ is a GRF with respect to $\mathcal{N}$. See [5].

Unfortunately, Z is very hard to compute. Since we have to consider all possible of values of x in order to find Z, the computational complexity of the task is a formidable $O(2^n)$, effectively preventing us from computing the absolute probabilities for the configurations of X. It is nevertheless possible to use the Gibbs distribution to find an estimate of the error patterns generated by the channel.

III. Error estimation

We want to find an estimate of the error pattern that was added to the codeword, based on the assumptions about the dependence between errors given in the previous sections. In order to avoid computing the constant Z in the Gibbs distribution, we will do a MAP estimation of the errors. That is, given a received word Y, we want to find an estimate of the most likely error pattern X that was added to C. Some terminology is needed in order to develop this. Let A be a set of random variables defined on the set $\mathcal{L}$, and let the elements of A be indexed by $1 \leq i \leq n$. If $A_i = a_i$ for each variable A_i, where $a_i \in \mathcal{L}$, we call $\{a_1, \dots, a_n\} = a$ a *configuration* of A

A. MAP estimation

MAP estimation of the error pattern X based on the received word Y can be formulated as the maximization of the *a posteriori* probability $P(X = x|Y = y)$ with respect to x. That is, we want to find a configuration x that makes the probability $P(X = x|Y = y)$ as high as possible.

Bayes rule gives us

$$P(X = x|Y = y) = \frac{P(X = x)P(Y = y|X = x)}{P(Y = y)}$$

Since $P(Y = y)$ does not depend on $P(X = x)$, we can maximize over

$$P(X = x)P(Y = y|X = x) \qquad (2)$$

To find the probabilities $P(Y = y|X = x)$, we must take care to remember that the error pattern X is now considered as the original information that we want to estimate, and the codeword C is to be considered as errors obscuring the information. In the following, we shall make some assumptions about X and C.

- The variables are bipolar, with 1 corresponding to 0 and -1 corresponding to 1 in the channel model.
- The codeword C, when treated as errors, can be seen as random bipolar variables so that

$$P(C = c) = \prod_i P(c_i) = (\frac{1}{2})^n$$

Under these assumptions only the transition $1 \rightarrow -1$ takes place, and the possibility of doing so is $\frac{1}{2}$ since c_i is supposed to be a random binary variable. The value of $P(Y|X)$ is given by the channel characteristics and the assumption that there is an equal probability that a 1 bit and a -1 bit in the codeword will coincide with a -1 bit in the error pattern. The resulting probabilities are seen in Table 1.

$x \backslash y$	1	-1
1	$\frac{1}{2}$	$\frac{1}{2}$
-1	0	1

Tab. 1: Transition probabilities

The conditional probabilities in the table can be expressed as an exponential function by

$$P(Y_i = y_i|X_i = x_i) = \lim_{\epsilon \to 0} \frac{1}{2} \exp\left[\frac{-y_i(1 - x_i)\ln 2}{1 - y_i + \epsilon}\right]$$

We can then express the probability of a given y conditioned on a configuration x by

$$P(Y = y|X = x) = \prod_i \lim_{\epsilon \to 0} \frac{1}{2} \exp\left[\frac{-y_i(1 - x_i)\ln 2}{1 - y_i + \epsilon}\right] \qquad (3)$$

Substituting (1) and (3) into (2), we can find the joint probability by

$$P(X = x, Y = y) = \left[Z^{-1} e^{-\frac{1}{T}U(x)}\right] \prod_i \lim_{\epsilon \to 0} \frac{1}{2} \exp\left[\frac{-y_i(1 - x_i)\ln 2}{1 - y_i + \epsilon}\right]$$

Since the natural logarithm is strictly increasing, the following equality holds:

$$\arg\max_X(P(X,Y)) = \arg\max_X(\ln(P(X,Y)))$$

In order to avoid computing Z in the above expression, we take the logarithm of both sides and eliminate constants to get

$$V(x) = U(x) + \sum_i \lim_{\epsilon \to 0} \left[\frac{-y_i(1 - x_i)\ln 2}{1 - y_i + \epsilon}\right]$$

where $V(x) = \ln[P(X = x, Y = y)]$.

We define the partial functions V_i of $U(x)$ according to [3, 4]

$$\begin{aligned} V_1(x_i) &= \alpha x_i \\ V_2(x_i, x_{i'}) &= \beta_{i,i'} x_i x_{i'} \\ V_3(x_i, x_{i'}, x_{i''}) &= \cdots = 0 \end{aligned}$$

Note that the expression for V_2 implies that $V_2(x_i, x_{i'}) = \beta_{i,i'}$ for $x_i = x_{i'}$and $V_2(x_i, x_{i'}) = -\beta_{i,i'}$for $x_i \neq x_{i'}$.

From this we get a new expression for $V(x)$:

$$V(x) = \sum_i \left[\alpha x_i + \beta_{i,i'} \sum_{i' \in \mathcal{N}_i} x_i x_{i'} + \lim_{\epsilon \to 0} \left[\frac{-y_i(1 - x_i)\ln 2}{1 - y_i + \epsilon}\right]\right],$$

and splitting the last term into a constant and a non-constant term yields

$$V(x) = \sum_i \left[\alpha x_i + \beta_{i,i'} \sum_{i' \in \mathcal{N}_i} x_i x_{i'} + \lim_{\epsilon \to 0} \left[\frac{x_i y_i \ln 2}{1 - y_i + \epsilon}\right] + \lim_{\epsilon \to 0} \left[\frac{-y_i \ln 2}{1 - y_i + \epsilon}\right]\right].$$

Since the last term only depends on y, we can find the MAP configuration by simplifying the expression to:

$$\begin{aligned} \mathcal{V}(x) &= \sum_i \left[\alpha x_i + \beta \sum_{i' \in \mathcal{N}_i} x_i x_{i'} + \lim_{\epsilon \to 0} \left[\frac{x_i y_i \ln 2}{1 - y_i + \epsilon}\right]\right] \\ &= \sum_i \left[\alpha + \beta \sum_{i' \in \mathcal{N}_i} x_{i'} + \lim_{\epsilon \to 0} \left[\frac{y_i \ln 2}{1 - y_i + \epsilon}\right]\right] x_i \qquad (4) \end{aligned}$$

B. Optimization of $\mathcal{V}(x)$

To do a global optimization of the expression above with respect to x would become computationally infeasible as the size of x increases. Instead we can use the local dependencies between bits to do a local optimization along the lines of the PDFE in [1, 2] or the partial binary segmentation algorithm of [4]. It is apparent that when $\mathcal{V}(x)$ is factored as in (4), we can always choose the value of x_i so that each term in the sum becomes positive, and thus the sum is non-decreasing. For each pixel we compute the value of $\left[\alpha + \beta \sum_{i' \in \mathcal{N}_i} x_{i'} - \lim_{\epsilon \to 0} \left[\frac{-y_i \ln 2}{1 - y_i + \epsilon}\right]\right]$ and set the value of x_i so that this expression is positive. This procedure is iterated until we converge on a solution or a maximum number of iterations is reached.

IV. Error generation

Generation of two dimensional burst errors is done by the use of a Monte Carlo Markov chain technique called the Metropolis algorithm.We do not have very strict requirements for the generated sample configurations, other than that they should be "somewhat likely" to occur given the condition that the variables' distribution is given by the Gibbs distribution.

The Metropolis algorithm is a general method for generating samples from a joint distribution of two or more variables, and can be applied to distributions that are either continuous or discrete as long as it is possible to compute the difference of the likelihoods for two configurations of the variables.

We would like to sample the joint distribution $A = \{A_1, \ldots, A_n\}$. This is achieved by generating random changes to the components A_i of A, and accepting or rejecting these changes based on how they affect the likelihood of the configuration. In our case, the natural change to a component of a configuration would be to flip the bit value.

Given an initial configuration A, a new configuration A^* is obtained as explained above by flipping a bit. Then, the difference of the likelihood of the new configuration and the old configuration is calculated by

$$\Delta U = U(A^*) - U(A) =$$
$$\sum_{a^* \in C_1} V_1(a^*) + \sum_{a^*, b^* \in C_2} V_2(a^*, b^*) + \sum_{a^*, b^*, c^* \in C_3} V_3(a^*, b^*, c^*) + \ldots$$
$$- \sum_{a \in C_1} V_1(a) + \sum_{a, b \in C_2} V_2(a, b) + \sum_{a, b, c \in C_3} V_3(a, b, c) + \ldots$$

and the new configuration is accepted with probability 1 if the new likelihood is higher than the old one. Otherwise, the new configuration is accepted with probability $e^{-\Delta U/T}$, so the probability of accepting the new configuration becomes:

$$P(A \to A^*) = \begin{cases} 1 & \Delta U \geq 0 \\ e^{-\Delta U/T} & \Delta U < 0 \end{cases}$$

A pass through all the components in A in this way is called a *sweep* over the variables in A. In our case, we generate a sample from the distribution by doing 4 sweeps over A, resulting in the evaluation of a total of $4n$ new configurations. This should result in a sample that has high enough probability to be detected by the estimation algorithm described above.

V. Performance of estimation algorithm

The performance of the estimation algorithm depends heavily on the value of β, which determines the degree of clustering in the error pattern. A critical performance parameter is the probability ε that not all bits in the error pattern are detected by the estimation algorithm. A bit that belongs to the error pattern, but is not detected as such, is given a high probability of being correct, and can hence be the source of errors that are hard to correct. Therefore, ε is an important measure of the reliability of the algorithm. As can be seen in Fig. 2, $P(\varepsilon)$ is high for $\beta < 0.5$, reflecting the fact that small and very irregular error clusters appear in this range. $P(\varepsilon)$ drops sharply initially, but levels out when $\beta > 1$ as a result of the clusters becoming bigger and more coherent. As we shall see later, this is also reflected in the performance of the algorithm.

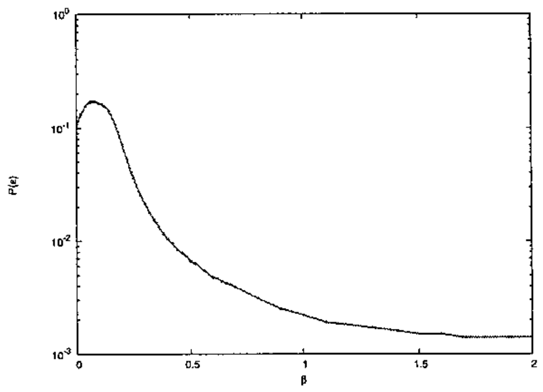


Fig. 2: $P(\varepsilon)$ for fixed $\beta = 0.2$ in the estimation algorithm.

VI. Decoding

Having obtained an estimate

$$\hat{X} = \{\hat{X}_1, \ldots, \hat{X}_i, \ldots, \hat{X}_n\}$$

of the error pattern, it can be used to find likelihood ratios for input to the decoder. For each bit, we set the likelihood ratio to

$$L_i = \frac{P(C_i = -1 | Y_i, \hat{X}_i)}{P(C_i = 1 | Y_i, \hat{X}_i)}$$

The resulting probabilities can be seen in Table 2. In the

$\hat{x}$ \ y	-1	1
-1	1	0
1	$\frac{1-\rho}{0+\rho}$	0

Tab. 2: Input probabilities to the decoder

table, ρ is the probability that a bit belonging to the error pattern is incorrectly estimated as a 1-bit. The parameter ρ must be estimated by simulation, but should in general be small, indicating a relatively certain -1-bit.

VII. Simulations

A regular LDPC code is used as the error correcting code component We shall use different values of β in the simulations, and assume $\alpha = \gamma = \cdots = 0$ in the estimation algorithm. We shall also assume that the receiver does not know the value of β used by the noise generating process. The components of the simulator is then connected as shown in Fig. 3 The simulations show that there is a large performance gain

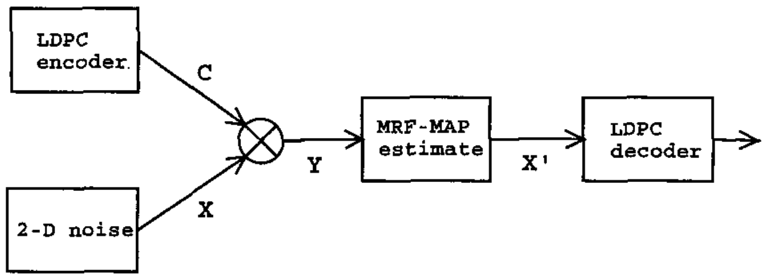


Fig. 3: System model

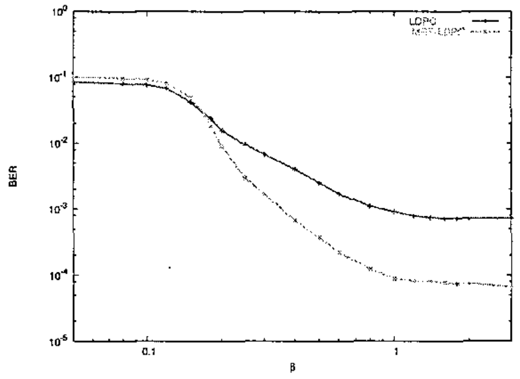


Fig. 4: Performance under varying β with rate $\frac{3}{8}$

for some choices of parameter using the LDPC-MRF combination described above. The value of β has great influence over the relative performance of the two decoding methods. Looking in Fig. 4 at the performance of a code in combination with the MAP error estimate and alone, under varying β, we can observe that the performance difference between the two decoders increases as β increases. This is due to the effect described in Section V: as β increases, the reliability of the error estimate also increases. We also notice that the drop in BER levels off at about $\beta = 1$ corresponding to the reliability of the estimate leveling off from the same point. The performance of the decoder could also be measured under varying bit error probabilities, but because the bit error probability depends on the parameter β in the Gibbs distribution in a way that makes it hard to predict the average error probability over codewords, we fix the value of β to $\beta = 0.2$ and $\beta = 1.0$ which gives an average error rate of about 0.12 and 0.02 respectively, and study the performance of joint LDPC - MRF decoding for different code rates using these parameters. We see in Fig. 5 that the effect of the MRF estimator gives very good results in combination with the LDPC code when the code rate is sufficiently low, while the performance gap between the two decoders gets smaller as the code rate grows. This occurs because the MRF-LDPC decoder needs a certain amount of information from the code itself to determine the value of the bits in the error pattern, even if the MRF estimator provides a perfect estimate of the errors.

VIII. Conclusions

We have observed significant performance gains using the combined error detection and correction methods described above. We have also seen that the performance of the algorithm depends on the reliability of the error estimate. We have however not done any optimization of the error correcting codes used in the simulations. There should be a potential for further performance improvements by either constructing the codes to assure a maximum spatial spread of the bits in each parity check, or use interleaving to achieve the same result. It is also possible to extend the use of cluster error detection to other channels like the binary symmetric channel.

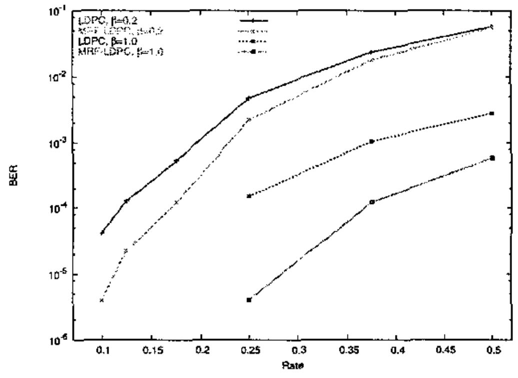


Fig. 5: Performance under varying rate with $\beta = 0.2$ and $\beta = 1.0$

References

[1] M. A. Neifield, K. M. Chugg and B. M. King, "Parallel data detection in page-oriented optical memory", Optics Letters, vol. 21, no. 18, pp. 1481-1483, Sept. 1996.

[2] M . Neifeld and B. M. King, "Parallel detection algorithms for page oriented optical memories", Applied Optics, vol. 37, no. 26, pp. 6275-6297, Sept. 1998

[3] S. Z. Li, "Modeling image analysis problems using Markov random fields", Handbook of Statistics, vol. 20, pp. 1-43, 2000.

[4] Shridhar, M., Ahmadi, M., El-Gabali, M., "Restoration of noisy images modeled by Markov random fields with Gibbs distribution", Circuits and Systems, IEEE Transactions on, vol. 36 , no. 6 , pp. 884 - 890, June 1989.

[5] R. Kindermann and J. L. Snell, "Markov random fields and their applications", AMS, Providence, R. I., 1980.

[6] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods", Technical Report CRG-TR-93-1, Dep. of Computer Science, University of Toronto, 1993.

[7] R. Chellappa, A. Jain (ed.), "Markov Random Fields", Academic Press, San Diego, 1993.