EQUALIZATION ALGORITHMS FOR THE TWO-DIMENSIONAL INTERSYMBOL INTERFERENCE CHANNEL

By

YIMING CHEN

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

May 2011

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of YIMING CHEN find it satisfactory and recommend that it be accepted.

Benjamin J. Belzer, Ph.D., Co-Chair

Krishnamoorthy Sivakumar, Ph.D., Co-Chair

Thomas R. Fischer, Ph.D.

ACKNOWLEDGEMENT

I would like to express my gratitude to my advisor, Dr. Benjamin Belzer and co-advisor, Dr. Krishnamoorthy Sivakumar, for their devoted guidance, continuous encouragement and support, and constructive suggestions throughout this research work. The program was one of the most important and formative experiences in my life. Without their help this dissertation would not have been possible.

I would also like to thank my committee member, Dr. Thomas Fischer for his valuable advice and comments, which led to significant improvements of my research. He has generously given his time and expertise to better my work. I thank him for his contribution and good-natured support.

The financial support of this work from the NSF (grant CCR-0098357 and CCF-0635390) and the School of Electrical Engineering and Computer Science are gratefully acknowledged.

I also want to thank all the staff of the School of Electrical Engineering and Computer Science, for their help and support.

Finally, I especially wish to express my deepest gratitude to my beloved wife and my parents for their encouragement, support, love and understanding.

EQUALIZATION ALGORITHMS FOR THE TWO-DIMENSIONAL

INTERSYMBOL INTERFERENCE CHANNEL

Abstract

by Yiming Chen, Ph.D.
Washington State University
May 2011

Co-Chairs: Dr. Benjamin Belzer and Dr. Krishnamoorthy Sivakumar

This dissertation presents several novel iterative soft decision feedback equalization algorithms for detection of binary-valued two-dimensional images corrupted by 2D intersymbol interference (ISI) and additive white Gaussian noise (AWGN). These algorithms exchange weighted soft extrinsic information between maximum-a-posteriori (MAP) detectors employing different row-column or zigzag scan directions. We first use an independent assumption on the a priori probabilities (APP) used in the computation of the BCJR algorithm, where the extrinsic information transmission (EXIT) charts are used to speed up the optimization of the weight and iteration schemes.

Simulation results for the 2×2 averaging mask channel show that, at low signal-to-noise ratios (SNR), the new zigzag algorithm gains about 1 dB over previous iterative row-column soft decision feedback algorithm and over a separable-mask algorithm, two of the best previously published schemes. When the zigzag algorithm is serially concatenated with the row-column algorithm, the concatenated system performs better than four of the

best previously published algorithms.

Based on the definition of the BCJR algorithm, we remove the independence assumption and instead use joint estimation, where we introduce the new two-dimensional extrinsic information flow (TEXIF) chart, which is used to explore the subtraction issue of the extrinsic information for the iterative detection process.

For the redesigned joint iterative row-column soft decision feedback algorithm, the block (BLK) algorithm, has the performance within 0.3 dB from the maximum likelihood bound. To address the increased computational and storage complexity introduced by the joint estimation, we develop a simplified block (SBLK) algorithm with almost the same performance as BLK algorithm.

The redesigned iterative soft decision feedback zigzag algorithm using joint estimation, the block zigzag (BLKZ) algorithm, provides more than 2 dB SNR gain over the ISDFZA on the 2×2 averaging mask channel, and 1 dB gain for the 3×3 averaging mask at high SNRs. When the BLKZ algorithm is concatenated with the joint IRCSDFA, the concatenated system achieves the ML bound at high SNR with the 2×2 averaging mask; with the 3×3 averaging mask it comes within 0.2 dB of the ML bound.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

To continue the historical trend of exponentially increasing storage density, two-dimensional (2D) storage techniques, wherein bits are written in 2D blocks (rather than 1D tracks), are being developed for magnetic and optical disks [12], [36], and also for newer 3D technologies like holographic storage [16]. These new multi-dimensional storage techniques promise density and data read-write rate increases of more than an order of magnitude over current state of the art. However, 2D and 3D storage systems suffer from 2D and 3D intersymbol interference (ISI) due to the low-pass nature of the read-write system. Without effective equalization, 2D and 3D ISI will cause unacceptably high bit error rates (BERs) in next-generation magnetic and optical storage systems.

1.1 2D ISI Channel Model

We considers the detection of an $M \times N$ binary-equiprobable two dimensional (2D) independent and identically distributed (i.i.d.) image $\mathbf{f}$ with elements $f(k,l) \in \{-1,1\}$ from received image $\mathbf{r}$ with elements (pixels)

$$r(m,n) = g(m,n) + w(m,n), \tag{1.1}$$

where $g(m,n)$ is the 2D convolution

$$g(m,n) = \sum_k \sum_l h(m-k, n-l) f(k,l). \tag{1.2}$$

In the above equations $h(k,l)$ are the elements of a finite impulse response 2D blurring mask $\mathbf{h}$, the $w(m,n)$ are zero mean i.i.d. Gaussian random variables (r.v.s), and the double sum is computed over the support of $h(m-k, n-l)$: $\mathcal{S}(m,n)_{h^-} = \{(k,l) : h(m-k, n-l) \neq 0\}$. The model in (1.1) and (1.2) applies, e.g., to 2D data storage systems, which suffer from 2D intersymbol interference (ISI) at high storage densities. Such systems are under active development by industry for next-generation optical disk storage (e.g., [12]), and holographic data storage (e.g., [16]). The binary image is assumed that a boundary of -1 elements surrounds the data set. The model in (1.2) applies, e.g., to 2D data storage systems, which suffer from 2D intersymbol interference (ISI) at high storage densities. Such systems are under active development by industry for next-generation optical disk storage (e.g., [12]), holographic data storage (e.g., [16]), and two-dimensional magnetic recording (e.g., [36]).

Direct maximum likelihood (ML) detection of $\mathbf{f}$ from $\mathbf{r}$ requires comparison of $\mathbf{r}$ with 2^{MN} candidate images, and is therefore impractical for typical image dimensions of $M, N \geq 64$. The standard Wiener filtering solution is significantly inferior to ML detection, especially at high signal-to-noise ratios (SNRs) [25]. Hence, it is desirable to develop a low-complexity 2D detection algorithm that closely approximates the performance of 2D ML

detection. For one dimensional signals, the Viterbi algorithm (VA) provides an efficient method for ML detection of ISI-corrupted data [14]. But the VA does not generalize to two or higher dimensions. In fact, it has been shown in [29] that the 2D ML sequence detection problem is NP complete for certain fixed 2D ISI channels having the form of (1.2). Nonetheless, asymptotically tight union bounds on the performance of 2D ML detection have been developed in [11]; these bounds are useful in assessing the performance of 2D detection algorithms at high SNR.

A number of papers (e.g. [34, 3]) have proposed efficient methods for computing the channel capacity of 2D-ISI channels. The channel capacity is the ultimate performance bound, but practical systems require both channel coding and equalization to approach capacity. The present paper is concerned with equalizer design only. For this reason we use the ML bound as the theoretical performance limit for our equalizer designs; this approach is well established in the literature. While the gains due to equalization and channel coding may not be strictly additive, improved equalizer design usually leads to improved performance of systems that combine equalization and channel coding [10].

A number of 2D decision-feedback VAs (DFVAs) have been constructed, based on row-by-row raster scanning of the image (e.g. [15, 18, 25]). To our knowledge, [25] employed the first iterative algorithm for 2D ISI reduction; the DFVA was run on rows and columns, and bits agreeing in both directions were fixed for subsequent iterations. Subsequent work employed the turbo principle (after turbo coding [2]). In [4], the 2D convolution is decomposed into two 1D computations, and an iterative algorithm exchanges soft information between 1D soft-in soft-out (SISO) detectors. In [37], the binary source image is coded with a low-density parity check (LDPC) error correcting code before transmission over a

separable 2D-ISI channel. Separability is exploited to construct an iterative row-column algorithm in which a non-binary column SISO detector is followed by a binary row SISO detector, followed by an iteration of the LDPC decoder, etc. The LDPC's coding gain enables [37] to approach within less than 1 dB the bit error rate (BER) curve for the non-ISI channel. In [22], soft information is exchanged between maximum *a posteriori* (MAP) row and column detectors; this scheme avoids decision feedback by making decisions on multiple rows/columns, rather then one row/column at a time. An iterative row-column soft decision feedback (SDF) algorithm (IRCSDFA) similar to that of [22], which outperforms the algorithms of [25], [4], [22] and [37] (without LDPC coding) was also recently developed [9].

Approximations of generalized belief propagation (GBP) are developed for the 2D-ISI and related problems in [33]. The GBP-based 2D-ISI equalizer uses exact inference over the sub-region of the image covered by the ISI mask, and passes messages between adjacent sub-regions. The GBP equalizer achieves ML performance for the cases tested in [33], but has been demonstrated only on small (20×20 or smaller) images and with 2×2 ISI masks with relatively low-magnitude coefficients relative to the main tap h_{00}; such cases are easily handled because the nearby boundary conditions greatly aid the estimation, and because the ISI is relatively less compared to masks with a flatter coefficient energy distribution.

In practical applications such as optical or magnetic recording, more complicated 2D ISI channel models than that of (1.1) are sometimes required. Equalization on such channels is considered in, e.g., [26], where a variable-state trellis VA is constructed for multi-level 2D optical storage systems subject to pit-size and pit-position noise.

In [7], we propose a novel iterative soft-decision feedback zigzag algorithm (ISDFZA).

When the ISDFZA is concatenated with the iterative row-column soft-decision feedback algorithm (IRCSDFA) of [9], the system achieves the high-SNR maximum likelihood (ML) bound for the 2×2 averaging mask channel (where $h(k,l) = 1/4$ for $k, l \in \{0, 1\}$), and comes within 0.8 dB of the bound for the 3×3 averaging mask. Comparisons in [7] show that the concatenated ISDFZA-IRCSDFA provides superior or competitive performance to other recently proposed 2D-ISI equalizers. However, the algorithms in [7] assume that pixel estimates passed as extrinsic information are independent, when in fact they are spatially correlated. Also, these algorithms employ log-likelihood ratio (LLR) weights that vary by iteration; optimizing these weight schedules requires EXIT charts to minimize design time.

1.2 Iterative Row-Column Soft Decision Feedback Algorithm (IRCSDFA) Review

The following material is a summary of the papers by Cheng et.al. [9, 8]. The reason for having this part here is because this IRCSDFA will play an important role in the concatenated system which has the best performance in either the independent case or the joint case. And another reason is that, we also redesign this algorithm by using the joint extrinsic information in chapter 3, so the review here could be helpful to have the necessary background knowledge.

Figure 1.1 shows a block diagram of the algorithm scheme, in which the two estimation blocks (row-by-row, column-by-column) exchange weighted soft information. The weights w, which are less than one, prevent the algorithm from converging too quickly. Each detector also processes received image $\mathbf{r}$, which is corrupted by 2D-ISI and by AWGN. The

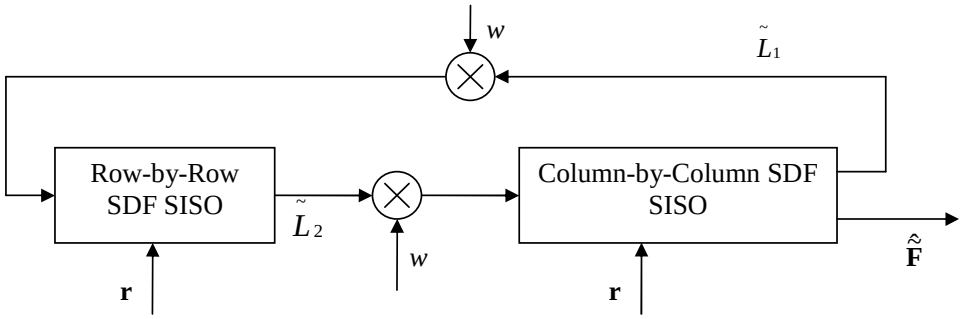


Figure 1.1: IRCSDF algorithm in a 2D ISI channel.

basic element is a *soft decision feedback, soft-input soft-output* detector.

The IRCSDF algorithm is a modified BCJR algorithm, in which soft estimates of branch outputs and state transition probabilities from earlier trellis stages are used as *soft decision feedback* (SDF) to aid the computation of log-likelihood ratios (LLRs) for the current pixel. The SDF branch output computation computes LLRs for inner products between the mask and the image.

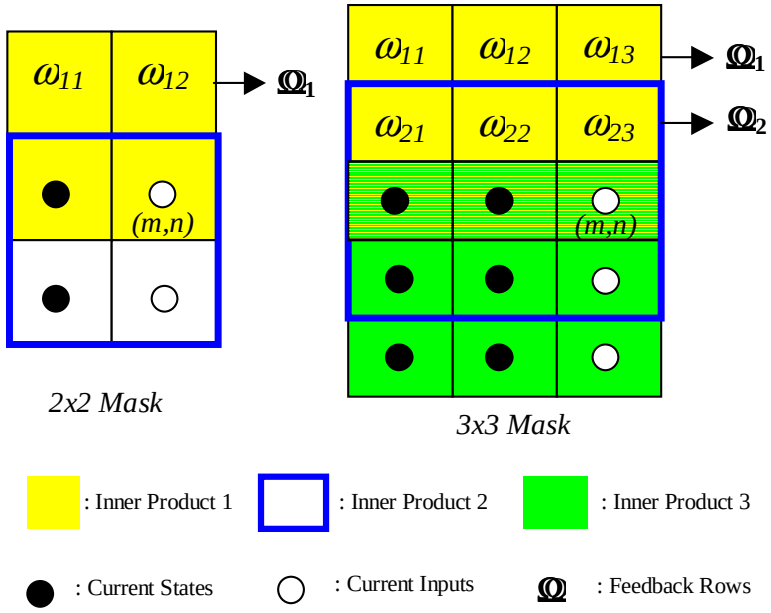


Figure 1.2: State, input, and feedback pixels for the 2×2 mask and 3×3 mask.

For the row-SISO, trellis states and inputs are defined in Figure 1.2; for the 3×3 mask, the trellis has 64-states and 8 branches entering and leaving each state. The IRCSDFA

exchanges weighted LLRs between row and column soft-in/soft-out (SISO) estimators. Each estimator runs a BCJR-like algorithm [1] that uses soft decision feedback in the form of LLRs from previously processed rows or columns. For the 3×3 mask row SISO, trellis states and inputs are defined in Fig. 1.2, where Ω_1 and Ω_2 denote feedback rows. The 2D convolution can be viewed as the 2D inner product between original image $\mathbf{f}$ and the index-reversed and shifted mask $\mathbf{h}^-_{m,n}$ with elements $h(m-k, n-l)$, where mask coefficient $h(0,0)$ is at pixel position (m,n). The inverted mask raster scans through the image row-by-row or column-by-column. It is assumed that a boundary of -1 pixels surrounds the original image $\mathbf{f}$.

Trellis generation for the 3×3 mask on the mth image row is initiated by placing the input marked (m,n) in Fig. 1.2 at the left end of the row, where the initial values of the six state pixels are -1 due to the boundary conditions, and the vector $\mathbf{u}_k = [u_{k0}, u_{k1}, u_{k2}]$ of three input pixels can take $2^3 = 8$ different values. The entire state/input block is then shifted right to pick up the next three input pixels, and the previous three input pixels become the middle three state pixels. The full-state trellis therefore has $2^6 = 64$ states with 8 branches out of each state. At each position (m,n), the trellis branch output vector consists of three 3×3 inner products between the index-reversed mask and the pixel values defined by the trellis. The upper inner product uses feedback from two previously processed rows, the middle uses one feedback row and the lower uses trellis pixels only. The branch metrics are the squared Euclidean distances (SEDs) between the branch output vectors and the received pixel vectors $\mathbf{y}_k = [y_{k0}, y_{k1}, y_{k2}] \triangleq [r(m,n), r(m+1,n), r(m+2,n)]$. At each position (m,n) the trellis branch output is a vector consisting of three 3×3 inner products between the inverted mask and the pixel values defined by the trellis; the upper

inner product, named $x(m,n)$, uses two feedback rows, the middle one,named $x(m+1,n)$, uses one feedback row and the lower one, named $x(m+2,n)$, just uses received pixels. The branch metric is the squared Euclidean distance between the branch output and the real received pixel vector $[r(m,n), r(m+1,n), r(m+2,n)]$. For the 2×2 mask the trellis has 4 states and 4 branches for each state. The column-by-column case is similar to the row-by-row case. In the following description, we continue to assume the 3×3 mask definitions of Fig. 1.2. Given trellis state S_k, input vector $\mathbf{u}_k$, and received vector sequence $y_1^{N_r} \triangleq [y_1, \ldots, y_{N_r}]$, define $\lambda_k^{\mathbf{i}}(s) \triangleq P(\mathbf{u}_k = \mathbf{i}, S_k = s, y_1^{N_r})$, where $\mathbf{i} = [i_0, i_1, i_2]$ and $i_m \in \{-1, 1\}$.

To illustrate the SDF LLR calculation, assume the 3×3 averaging mask $h_{kl} = 1/9$ is used to compute the convolution $c(m,n) = \sum_{k=0}^{2} \sum_{l=0}^{2} f(m-k, n-l)/9$.

For pixel (m,n) at the kth stage, $k \in \{0, 1, \ldots, N\}$, the corresponding received pixel vector is $\mathbf{r} = [r(m,n), r(m+1,n), r(m+2,n)]$, and the actual input vector is $\mathbf{f} = [f(m,n), f(m+1,n), f(m+2,n)]$. To simplify, let $\mathbf{y}_k = [y_{k0}, y_{k1}, y_{k2}] = \mathbf{r}$, and $\mathbf{u} = [u_{k0}, u_{k1}, u_{k2}] = \mathbf{f}$. The log-likelihood ration (LLR) is

$$L_i(k) = \log \left(\frac{P(u_{k0} = +1 | \mathbf{y}_1^{N_r}, \tilde{\mathbf{u}}_i)}{P(u_{k0} = -1 | \mathbf{y}_1^{N_r}, \tilde{\mathbf{u}}_i)} \right), \tag{1.3}$$

where $\tilde{\mathbf{u}}_i$ is the estimation of pixel vector $\mathbf{u}$ from detector i, $i \in \{1, 2\}$. The extrinsic information input to detector i is

$$\tilde{L}_i(k) = \log \left(\frac{P(u_{k0} = +1 | \tilde{\mathbf{u}}_i)}{P(u_{k0} = -1 | \tilde{\mathbf{u}}_i)} \right), \tag{1.4}$$

and the output extrinsic information to the next detector is

$$\tilde{L}_{\mathrm{next}(i)}(k) = [L_i(k) - \tilde{L}_i(k)] \times w_i, \tag{1.5}$$

where $\mathrm{next}(1) = 2, \mathrm{next}(2) = 1$, and w_i is a weighting coefficient with $0 < w_i \leq 1$. By using the input extrinsic information, we can compute the conditional probability of the input pixel:

$$\begin{aligned} P(u_{k0} = +1|\tilde{\mathbf{u}}_i) &= \frac{e^{\tilde{L}_i(k)}}{1 + e^{\tilde{L}_i(k)}} \\ P(u_{k0} = -1|\tilde{\mathbf{u}}_i) &= \frac{1}{1 + e^{\tilde{L}_i(k)}}. \end{aligned} \tag{1.6}$$

Given a particular trellis state S_k, input vector $\mathbf{u}$, and noisy received vectors $\mathbf{y}_k$, define $\lambda_k^{\mathbf{i}}(s) = P(\mathbf{u} = \mathbf{i}, S_k = s, \mathbf{y}_1^{N_r})$, where $\mathbf{i} = [i_0, i_1, i_2], i_m \in \{-1, +1\}$, and $s \in \{0, 1, \ldots, 63\}$. We can then compute the *a posteriori* probability (APP)

$$P(\mathbf{u} = \mathbf{i}|\mathbf{y}_k) = \frac{\sum_s \lambda_k^{\mathbf{i}}(s)}{P(\mathbf{y}_1^{N_r})}. \tag{1.7}$$

As in [1], by setting

$$\begin{aligned}
\alpha_k(s) &= P(S_k = s, \mathbf{y}_1^K) \\
\beta_k(s) &= P(\mathbf{y}_{k+1}^{N_r} | S_k = s) \\
\gamma_{\mathbf{i}}(\mathbf{y}_k, s', s) &= P(\mathbf{u} = \mathbf{i}, S_k = s, \mathbf{y}_k | S_{k-1} = s'),
\end{aligned} \tag{1.8}$$

we have

$$\begin{aligned}
\lambda_k^{\mathbf{i}}(s) &= \sum_{s'} P(\mathbf{u} = \mathbf{i}, S_k = s, S_{k-1} = s', \mathbf{y}_k) \\
&= \sum_{s'} \alpha_{k-1}(s') \gamma_{\mathbf{i}}(\mathbf{y}_k, s', s) \beta_k(s).
\end{aligned} \tag{1.9}$$

The $\alpha_k(s)$ and $\beta_k(s)$ can be computed as

$$\begin{aligned}
\alpha_k(s) &= \sum_{s'} \sum_{\mathbf{i}} \alpha_{k-1}(s') \gamma_{\mathbf{i}}(\mathbf{y}_k, s', s) \\
\beta_k(s) &= \sum_{s'} \sum_{\mathbf{i}} \beta_{k+1}(s') \gamma_{\mathbf{i}}(\mathbf{y}_{k+1}, s, s'),
\end{aligned} \tag{1.10}$$

where $\alpha_0(0) = 1, \alpha_0(s) = 0$ for $s \neq 0$; $\beta_N(0) = 1, \beta_N(s) = 0$ for $s \neq 0$. The SDF output LLRs can be incorporated into the current pixel transition probabilities $\gamma_{\mathbf{i}}(\mathbf{y}_k, s', s)$. The modified γ is the product of a modified conditional channel PDF $p'(\cdot)$, the trellis transition probabilities, and the extrinsic information (from the other detector):

$$
\begin{aligned}
&\gamma_{\mathbf{i}}(\mathbf{y}_k, s', s) \\
&= p'(\mathbf{y}_k|\mathbf{u}=\mathbf{i}, S_k = s, S_{k-1} = s') \times P(\mathbf{u}=\mathbf{i}|s, s') \\
&\times P(S_k = s|S_{k-1} = s') \times P(\tilde{\mathbf{u}}|\mathbf{u}=\mathbf{i}).
\end{aligned} \tag{1.11}
$$

For the given states s', s and input $\mathbf{u}$, $P(\mathbf{u}=\mathbf{i}|s, s')$ is 0 or 1 and $P(S_k = s|S_{k-1} = s')$ is 1/8 based on the trellis. The extrinsic information can be computed as:

$$
P(\tilde{\mathbf{u}}|\mathbf{u}=\mathbf{i}) = \frac{P(\mathbf{u}=\mathbf{i}|\tilde{\mathbf{u}})P(\tilde{\mathbf{u}})}{P(\mathbf{u}=\mathbf{i})}, \tag{1.12}
$$

where $P(\mathbf{u}=\mathbf{i}|\tilde{\mathbf{u}})$ comes from (1.6), and $P(\tilde{\mathbf{u}}) = P(\mathbf{u}=\mathbf{i}) = 1/8$. The modified channel PDF sums over all possible values of the inner products c_{sdf} associated with state transition $s' \rightarrow s$ that are affected by past decisions:

$$
\begin{aligned}
&p'(\mathbf{y}_k|\mathbf{u}=\mathbf{i}, S_k = s, S_{k-1} = s') = P\big(y_{k2}|u_{k0}, u_{k1}, u_{k2}, s, s'\big) \\
&\times \Bigg[\sum_{\Omega_2} P(\Omega_2) P\big(y_{k1}|u_{k0}, u_{k1}, s, s', IP_2(\Omega_2)\big) \\
&\times \Bigg(\sum_{\Omega_1} P(\Omega_1) P\big(y_{k0}|u_{k_0}, s, s', IP_3(\Omega_1, \Omega_2)\big)\Bigg)\Bigg]
\end{aligned} \tag{1.13}
$$

where $\mathbf{\Omega}$ denotes feedback rows, inner product $IP_2(\Omega_2)$ and $IP_3(\Omega_1, \Omega_2)$ are functions of the feedback pixels, and probabilities $P(\Omega_j)$ are obtained from the feedback probabilities $P(\omega_{jl})$:

$$P(\Omega_j = \omega_{j0}, \omega_{j1}, \omega_{j2}) = \prod_{l=0}^{2} P(\omega_{jl}). \tag{1.14}$$

The probabilities $P(\omega_{jl})$ can be computed by (1.6), using feedback LLRs from previously processed rows (or columns) during the current iteration. We note that, since the original image is subject to AWGN, $p'(\mathbf{y}_k|\mathbf{u} = \mathbf{i}, S_k = s, S_{k-1} = s', c_{\mathrm{sdf}j}(\Omega))$ are normal PDFs with mean $c_{\mathrm{sdf}j}(\Omega)$.

Since we have vector inputs and received pixels, to estimate the pixel located on (m, n), we sum the λs over $(m + 1, n)$ and $(m + 2, n)$:

$$\lambda_k^{i0}(s) = \sum_{i1,i2} \lambda_k^{i0,i1,i2}(s). \tag{1.15}$$

The pixel LLR is computed as:

$$L(k) = \log\left(\frac{\sum_s \lambda_k^{i0=+1}(s)}{\sum_s \lambda_k^{i0=-1}(s)}\right). \tag{1.16}$$

If $L(k) > 0$, we decide that pixel (m, n) is $+1$; otherwise, it is detected as -1.

1.3 Iterative Soft Decision Feedback Zigzag Algorithm (ISDFZA) Review

This thesis demonstrates a new iterative soft-decision feedback zig-zig algorithm for reduction or elimination of 2D ISI. At low SNR, the algorithm achieves substantial reductions in required SNR for a given BER, when compared to the best previously published algorithms

for general or separable 2D ISI masks. When the ISDFZA is concatenated with the iterative row-column soft-decision feedback algorithm of [9], the concatenated system achieves superior equalization performance (for a system without error correction coding) on a number of 2D ISI channels, including the 2×2 and 3×3 averaging mask channels. Furthermore, the proposed algorithm is applicable to all classes of 2D-ISI masks, both separable and non-separable.

For the 2D ISI detectors, we also employ a modified BCJR [1] algorithm as in the IRCSDFA, in which SDF from other detectors is used to compute pixel LLRs. In addition to the two causal zig-zag scans shown in Fig. 1.3, 2D convolution can be performed with the two corresponding anti-causal zig-zag scans starting at the upper right corner of the image and having either a horizontal or vertical first move. In these anti-causal scans, image pixel $f(m, n)$ is multiplied by mask coefficient h_{01} rather than h_{00}. For each of the four possible zig-zag scan orders, an ISDFZA MAP detector can be defined as described below. Therefore, the ISDFZA can have one detector, two detectors, or four detectors. (Using three detectors is also possible, but we have not experimented with this due to the asymmetry involved in leaving off one of the complementary scan orders.) For best performance, the detectors should be concatenated in an order that assures the maximum independence (interleaving) of the inputs to each of the detectors [17].

In Figure 1.4, we show the diagram of the ISDFZA, which consists of four inner zigzag detector of different scan directions. The scan direction of each inner zigzag detector is shown in Figure 1.5. The above structures can apply to both 2×2 and 3×3 masks, and with either independence assumption or joint estimation. We will show the ISDFZA trellis definition, the states, the inputs and the feedback pixels definitions in next Chapter 2.

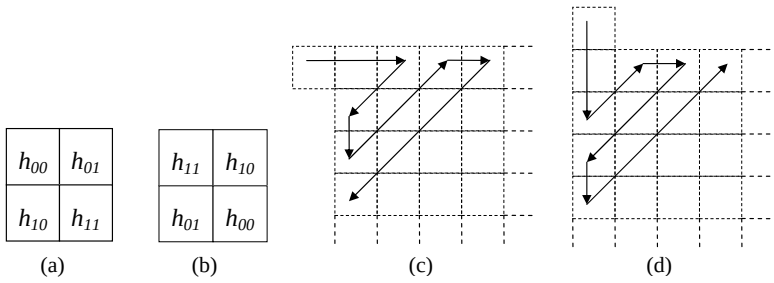


Figure 1.3: (a) 2×2 mask (b) 2×2 mask index-reversed for 2D convolution (c) horizontal and (d) vertical causal zig-zag scans.

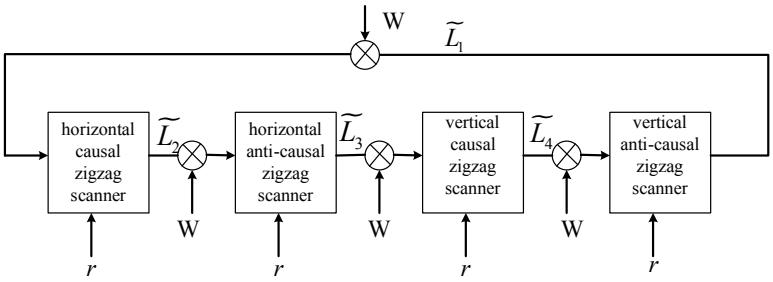


Figure 1.4: Block diagram of the Iterative Soft-Decision Feedback Zigzag Algorithm.

To optimize the weight and iteration schedules of the zig-zag, row-column, and concatenated systems, we applied EXIT charts; this is especially important for systems employing larger (e.g., 3×3 or greater) masks, since optimization via direct simulation of such systems is prohibitively time consuming due to their high trellis complexity.

EXIT charts were introduced by ten Brink in [35]. They allow the performance of a system of concatenated detectors to be predicted based on input/output mutual information curves for individual constituent detectors, which can be computed relatively quickly. First, the mutual information I_A between the input extrinsic information LLR Ξ_A to a given detector and the corresponding source symbol X is computed, where X takes the values -1 and 1 with probability $1/2$. The assumption here, as in [35], is that the conditional PDF $P_A(\xi_A|X)$ of the input extrinsic information is Gaussian with variance σ_A^2; this assumption has been experimentally verified for the ISI detector blocks employed in this thesis. Given

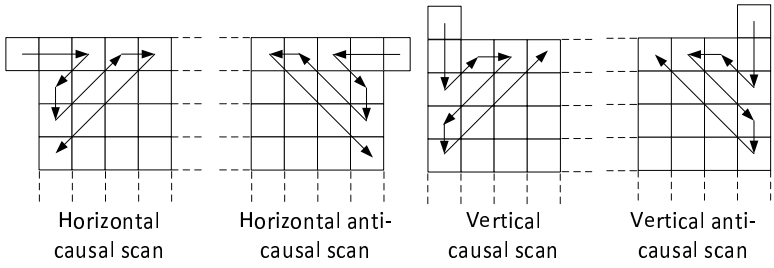


Figure 1.5: Scan directions of four inner zigzag detectors.

this assumption, the relation between the mutual information I_A and the input extrinsic information is

$$I_A(\sigma_A) = 1 - \int_{-\infty}^{\infty} \frac{e^{-((\xi-\sigma_A^2/2)^2/2\sigma_A^2)}}{\sqrt{2\pi}\sigma_A} \times \log_2(1+e^{-\xi})d\xi. \tag{1.17}$$

Experimental histograms indicate that the output extrinsic information LLRs Ξ_E for the detectors we use are not Gaussian. After passing the (conditionally Gaussian) input extrinsic information Ξ_A and the channel LLR (generated by Monte Carlo simulation) to a given detector, experimental conditional PDFs $P_E(\xi_E|X)$ for the output extrinsic LLRs are computed. Then the output mutual information I_E between the output extrinsic LLRs and the corresponding source symbols X is given by the following equation from [35]:

$$I_E = \frac{1}{2} \cdot \sum_{x=-1,+1} \int_{-\infty}^{\infty} P_E(\xi|X=) \times \log_2 \cdot \frac{2P_E(\xi|X=x)}{P_E(\xi|X=-1) + P_E(\xi|X=+1)} d\xi. \tag{1.18}$$

For the weight scheme optimization process for the independence assumption case, the technical details of the EXIT chart optimization are given in Chapter 2.

1.4 Joint Statistics

First, the previously described algorithms with independence assumption have two main limitations: the pixels involved in the extrinsic information exchanged between the constituent detectors are assumed to be independent in [9]. This is mainly a simplifying assumption and is not necessarily true. Secondly, the extrinsic information between detectors is weighted (by a factor less than 1) to account for imperfect information. This leads to a time consuming problem of designing an appropriate weight schedule; as shown in [7], proper design of the weight schedule results in significant performance improvement.

In [5], we redesign the IRCSDF algorithm by dropping the independence assumption on the extrinsic information exchanged between the row and column detectors; we call the resulting algorithm the block (BLK) algorithm. We estimate and exchange joint statistics for the pixels involved in the extrinsic information exchange. To address the increased computational and storage complexity introduced by the joint statistics, we develop a simplified block (SBLK) algorithm. Experimental results demonstrate that the SBLK algorithm performs almost as well as the BLK algorithm.

One additional advantage of exchanging joint statistics is that the new algorithms (both BLK and SBLK) allow a much simpler weight scheme — only a single parameter (constant weight) needed to be chosen. This weight can be designed by simple simulation; in particular, no EXIT chart method is needed here (compared to [7]). The new algorithm offers more than 1 dB gain for both 2×2 and 3×3 masks, and perform within 0.3 dB of the corresponding ML bound. The technical details of this part appear in Chapter 3.

In Chapter 4, we redesign the iterative soft decision feedback zigzag algorithm (ISD-

FZA) of Chen et. al. (IEEE JSAC, Feb. 2010) by using joint estimation, for equalization of two-dimensional intersymbol interference channels with additive white Gaussian noise. In the ISDFZA, source pixel estimates were assumed to be statistically independent; in the new algorithm we jointly estimate adjacent groups of source pixels. The new algorithm, called the block zigzag (BLKZ) algorithm, uses constant LLR weights across iterations, and provides more than 2 dB SNR gain over the ISDFZA on the 2×2 averaging mask channel; at low and medium SNRs it also outperforms the iterative row-column soft decision feedback algorithm (IRCSDFA) using joint extrinsic information (Chen et. al., CISS 2010). For the 3×3 averaging mask, the BLKZ algorithm gives more than 1 dB SNR gain over the independent ISDFZA at high SNRs. When the BLKZ algorithm is concatenated with the joint IRCSDFA, the concatenated system achieves the maximum-likelihood (ML) bound at high SNR with the 2×2 averaging mask; with the 3×3 averaging mask the joint concatenated system gains about 1 dB compared to the concatenated system with independence assumption, and comes within 0.2 dB of the ML bound. This chapter also introduces a new method of LLR subtraction for the BLKZ algorithm based on the algorithm's two-dimensional extrinsic information flow (TEXIF) chart.

1.5 Main Contributions

In this dissertation, the first part of this thesis is a new iterative soft-decision feedback zigzag (ISDFZA) SISO detection algorithm for 2D ISI equalization. The proposed algorithm uses log-likelihood ratio (LLR) weighting schedules for the extrinsic information passed between constituent detector modules to correct for the effects of SDF and to improve

performance. An extrinsic information transfer (EXIT) chart technique is developed to optimize the LLR weight schedules. In contrast to previous algorithms based on row or column scanning, the ISDFZA employs a zig-zag scan to construct a space-varying trellis spanning the entire image; the longer trellis provides improved performance at low SNR. The algorithms and results of [4], [37], [9], and [33] are examples of the previous state-of-the-art for equalization of uncoded images transmitted over 2D ISI channels. For the 2×2 averaging mask, the ISDFZA gains about 1 dB over IRCSDFA of [9] at low SNR, but is somewhat worse at high SNR due error propagation on its longer trellis. But when the ISDFZA is concatenated with the IRCSDFA, the concatenated system gains 1 dB at low SNR, and 0.4 dB at moderate and high SNRs over the IRCSDFA, and is about 0.7 dB better than the separable algorithm of [37] at high SNR. Similar performance gains are demonstrated with various other 2×2 and 3×3 masks. To our knowledge, the concatenated ISDFZA-IRCSDFA system achieves the best performance for uncoded 2D ISI equalization of small (2×2 and 3×3) masks reported previously to this thesis.

The second part of this work redesigns the previous iterative row-column soft decision feedback algorithm (Cheng *et al.*, 2007) for a two-dimensional intersymbol interference channel with additive white Gaussian noise; a 2×2 averaging mask and two 3×3 masks are considered. In particular, we consider the joint estimation of blocks of pixels involved in the exchange of extrinsic information between the detectors; previously, these pixels were considered to be statistically independent. The new algorithm, referred to as the block (BLK) algorithm, provides more than 1 dB SNR gain; the resulting performance is within 0.3 dB from the maximum likelihood bound for the masks considered. To address the increased computational and storage complexity introduced by the joint statistics, we

develope a simplified block (SBLK) algorithm. Experimental results demonstrate that the SBLK algorithm performs almost as well as the BLK algorithm. One additional advantage of exchanging joint statistics is that the new algorithms (both BLK and SBLK) allow a much simpler weight scheme a single parameter (constant weight) to optimize. This weight can be designed by simple simulation; in particular, no EXIT chart method is needed here (compared to [7]). The new algorithm offers more than 1 dB gain for both 2×2 and 3×3 masks, and performs within 0.3 dB of the corresponding ML bound.

The third part of this thesis redisigns the iterative soft decision feedback zigzag algorithm in the first part by using joint extrinsic information. The new joint zigzag algorithm could be applied to both 2×2 and 3×3 mask cases, and significant improvements are observed in several different channel models. The joint ISDFZA provides more than 2 dB SNR gain over the independent ISDFZA on the 2×2 averaging mask channel; at low and medium SNRs it also outperforms the iterative row-column soft decision feedback algorithm (IRCSDFA) using joint extrinsic information (Chapter 3). For the 3×3 averaging mask, the joint zigzag algorithm gives more than 1 dB SNR gain over the independent ISDFZA at high SNRs. When the joint zigzag algorithm is concatenated with the joint IRCSDFA, the concatenated system achieves the maximum-likelihood (ML) bound at high SNR with the 2×2 averaging mask; with the 3×3 averaging mask the joint concatenated system gains about 1 dB compared to the concatenated system with independence assumption, and comes within 0.2 dB of the ML bound, which is the best performance for 2D ISI detection without channel coding, based on our knowledge. In this thesis, we also introduce a new method of LLR subtraction for the joint equalization algorithm based on the algorithm's two-dimensional extrinsic information flow (TEXIF) chart. The TEXIF chart

gives a better view of the 2D extrinsic information flow and it leads to a more intuitive explanation of the subtraction issue for the 2D joint equalization algorithms.

1.6 Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2, the iterative soft-decision feedback zigzag algorithm (ISDFZA) with independence assumption and the serial concatenated system of ISDFZA (with independence assumption) and IRCSDFA (with independence assumption) are presented. The weight and iteration scheme optimization process using EXIT charts is also introduced in this chapter. The material of this chapter was partially presented at the 43th Annual Allerton Conference on Communication, Computing, and Control in September 2006 [28] and in the journal article of "IEEE Journal on Selected Areas in Communications (JSAC)" in Feb. 2010 [7]. In Chapter 3, we consider joint estimation of the extrinsic information on the iterative row column soft decision feedback algorithm (IRCSDFA) for both 2×2 and 3×3 mask channels. The material of this chapter was partially presented at the 44th Conference on Information Sciences and Systems (CISS 2010) in Mar. 2010 [5]. In Chapter 4, we consider the application of joint statistics in the iterative soft decision feedback zigzag algorithm (ISDFZA) for both 2×2 and 3×3 mask channels. In this chapter, we also introduce a new technique called two-dimensional extrinsic information flow (TEXIF) chart. The material of this chapter was partially presented at the 45th Conference on Information Sciences and Systems (CISS 2011) in Mar. 2011 [6]. The final Chapter 5, concludes the thesis and provides some discussion on future research directions.

Chapter 2

Equalization Algorithms for 2D Intersymbol Interference with Independent Assumption

Iterative soft decision feedback zigzag algorithm (ISDFZA) for 2D ISI equalization, is based on a generalized Bahl-Cocke-Jelinek-Raviv (BCJR) algorithms, which is an algorithm for maximum a posteriori decoding of error correction code defined on a trellis. The ISDFZA is described assuming both 2×2 mask and 3×3 mask; generalization to larger mask sizes is straightforward. The ISDFZA uses SISO (Soft Input Soft Output) and SDF (Soft Decision Feedback) information on the input and the feedback pixels, respectively. At each stage k of the trellis, input vector $\mathbf{u}_k$ and received pixel vector $\mathbf{y}_k$ are defined. Since the trellis has four types of moves, the definition of $\mathbf{u}_k$ and $\mathbf{y}_k$ changes with the type of move at stage k as shown in Fig. 2.1. For the horizontal move, we define $\mathbf{u}_k = [u_{k0}, u_{k1}] = [f(m, n+1), f(m-1, n)]$ and $\mathbf{y}_k = [y_{k0}, y_{k1}] = [r(m, n), r(m, n+1)]$. For the vertical move, $\mathbf{u}_k = [u_{k0}, u_{k1}] = [f(m+1, n), f(m, n-1)]$ and $\mathbf{y}_k = [y_{k0}, y_{k1}] = [r(m, n), r(m+1, n)]$. For the up-right move, $\mathbf{u}_k = [u_{k0}, u_{k1}, u_{k2}] = [f(m-1, n+1), f(m-1, n), f(m-2, n)]$, and $\mathbf{y}_k = [y_{k0}, y_{k1}] = [r(m, n), r(m-1, n+1)]$ or $\mathbf{y}_k = [y_{k0}, y_{k1}, y_{k2}] = [r(m, n), r(m-1, n+1), r(m-1, n)]$ for two or three inner prod-

ucts respectively. For the down-left move, $\mathbf{u}_k = [u_{k0}, u_{k1}, u_{k2}] = [f(m+1, n-1), f(m, n-1), f(m, n-2)]$, and $\mathbf{y}_k = [y_{k0}, y_{k1}] = [r(m, n), r(m+1, n-1)]$ or $\mathbf{y}_k = [y_{k0}, y_{k1}, y_{k2}] = [r(m, n), r(m+1, n-1), r(m, n-1)]$ for two or three inner products respectively. We use $\mathbf{y}_1^k$ to denote the received vector sequence $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$, where $k \leq N_r$ and N_r is the number of pixels in received image $\mathbf{r}$.

2.1 Iterative Soft Decision Feedback Zigzag Algorithm

The LLR for detector i is computed as in [2]: $L_i(k) = \log\left(\dfrac{P(u_{k0} = +1|\mathbf{y}_1^{N_r}, \tilde{\mathbf{u}}_{ki})}{P(u_{k0} = -1|\mathbf{y}_1^{N_r}, \tilde{\mathbf{u}}_{ki})}\right)$, where $\tilde{\mathbf{u}}_{ki}$ is the extrinsic estimation of $\mathbf{u}_k$ passed from the previous detector to detector i, $i \in \{1, \ldots, n_d\}$, and $n_d \in \{1, 2, 4\}$ is the number of detectors being used. The extrinsic information input to detector i is $\tilde{L}_i(k) = \log\left(\dfrac{P(u_{k0} = +1|\tilde{\mathbf{u}}_{ki})}{P(u_{k0} = -1|\tilde{\mathbf{u}}_{ki})}\right)$. The output extrinsic information to the next detector is $\tilde{L}_{\mathrm{next}(i)}(k) = \left[L_i(k) - \tilde{L}_i(k)\right] \times w_i$, where $\mathrm{next}(i) = (i \mod n_d) + 1$, $i \in \{1, \ldots, n_d\}$, and w_i is a weighting coefficient with $0 < w_i \leq 1$. The weights w_i reduce the LLR magnitudes to compensate for the unreliability of the soft decision feedback information, which can lead to error propagation as noted in [9]. LLR weighting and/or thresholding has been employed in a number of iterative estimation or decoding algorithms that, due to use of suboptimal constituent detectors or decoders, tend to overestimate the reliability of their output LLRs, e.g. [30, 31, 13].

2.1.1 The Trellis Definition

Trellis Definition for the 2×2 Mask

The ISDFZA uses the well known BCJR algorithm [1] on the trellis defined by the zig-zag scan through the image. In the following, we describe only the modifications to the BCJR algorithm that are unique to the ISDFZA. Let S_k denote the state variable that holds the decimal conversion of the three-bit state vector $\mathbf{s}_k$ at trellis stage k, and let s denote the value of the conversion: $s = \sum_{l=0}^{2} 2^l s_{kl}$. Define $\lambda_k^{\mathbf{i}}(s)$ as the joint probability of a particular trellis state S_k, input vector $\mathbf{u}_k$, and received vectors $\mathbf{y}_1^{N_r}$: $\lambda_k^{\mathbf{i}}(s) = P(\mathbf{u}_k = \mathbf{i}, S_k = s, \mathbf{y}_1^{N_r})$, where $\mathbf{i} = [i_0, i_1]$ for horizontal or vertical moves, $\mathbf{i} = [i_0, i_1, i_2]$ for diagonal moves, $i_m \in \{-1, +1\}$, and $s \in \{0, 1, \ldots, 7\}$. We can then compute the *a posteriori* probability (APP) $P(\mathbf{u}_k = \mathbf{i}|\mathbf{y}_1^{N_r}) = \sum_s \lambda_k^{\mathbf{i}}(s)/p(\mathbf{y}_1^{N_r})$, where $p(\mathbf{y}_1^{N_r})$ denotes the joint probability density function (PDF) of $\mathbf{y}_1^{N_r}$.

The SDF output LLRs can be incorporated into the current pixel transition probabilities $\gamma_{\mathbf{i}}(\mathbf{y}_k, s', s)$, defined as $\gamma_{\mathbf{i}}(\mathbf{y}_k, s', s) = P(\mathbf{u}_k = \mathbf{i}, S_k = s, \mathbf{y}_k|S_{k-1} = s')$. The modified γ is the product of a modified conditional channel PDF $p'(\cdot)$, the trellis transition probabilities, and the extrinsic information from the other detector(s): $\gamma_{\mathbf{i}}(\mathbf{y}_k, s', s) = p'(\mathbf{y}_k|\mathbf{u}_k = \mathbf{i}, S_k = s, S_{k-1} = s')P(\mathbf{u}_k = \mathbf{i}|s, s')P(s|s')P(\tilde{\mathbf{u}}_k|\mathbf{u}_k = \mathbf{i})$. The factor $P(\tilde{\mathbf{u}}_k|\mathbf{u}_k = \mathbf{i})$ is computed from the detector's extrinsic information LLR input as in equation (2) of [9]. For a given trellis stage type, let Ω_j denote the set of feedback pixels (with values ± 1) involved in the computation of the jth inner product IP_j in Fig. 2.1, and let $P(\Omega_j)$ denote its probability. The feedback probabilities $P(\Omega_j)$ are computed using the extrinsic information LLRs from the other detectors, under the assumption that the pixels in each feedback set are statistically

independent. The modified channel PDF sums over all possible values of the feedback pixels associated with state transitions $s' \rightarrow s$ that are affected by past decisions:

$$p'(\mathbf{y}_k|\mathbf{i}, s, s') = \begin{cases} p(y_{k0}|u_{k0}, u_{k1}, s, s') \sum\limits_{\Omega_2} P(\Omega_2) p(y_{k1}|u_{k0}, u_{k1}, s, s', \mathrm{IP}_2(\Omega_2)), \\ p(y_{k0}|u_{k0}, u_{k1}, u_{k2}, s, s') \sum\limits_{\Omega_2} P(\Omega_2) p(y_{k1}|u_{k0}, u_{k1}, u_{k2}, s, s', \mathrm{IP}_2(\Omega_2)), \\ p(y_{k0}|u_{k0}, u_{k1}, u_{k2}, s, s') \sum\limits_{\Omega_2} P(\Omega_2) p(y_{k1}|u_{k0}, u_{k1}, u_{k2}, s, s', \mathrm{IP}_2(\Omega_2)\big) \\ \qquad\qquad \times \sum\limits_{\Omega_3} P(\Omega_3) p(y_{k2}|u_{k0}, u_{k1}, u_{k2}, s, s', \mathrm{IP}_3(\Omega_3)\big), \end{cases} \tag{2.1}$$

for rectilinear, diagonal moves with two inner products and diagonal moves with three inner products respectively. In (2.1), because the noise is AWGN, the $p(y_{kl}|\mathbf{u} = \mathbf{i}, S_k = s, S_{k-1} = s', \mathrm{IP}_j(\Omega_j))$ are normal PDFs with means IP_j; the arguments of the exponential functions in these PDFs are the individual terms of the branch metrics (2.4)-(2.7). Since we have vector inputs and received pixels, to estimate input pixel $u_{k0} = i_0$, we sum the λs over the other inputs $u_{k1} = i_1$ and $u_{k2} = i_2$

$$\lambda_k^{i_0}(s) = \begin{cases} \sum\limits_{i_1} \lambda_k^{i_0, i_1}(s) & \text{(rectilinear moves)} \\ \sum\limits_{i_1} \sum\limits_{i_2} \lambda_k^{i_0, i_1, i_2}(s) & \text{(diagonal moves).} \end{cases} \tag{2.2}$$

The pixel LLR is computed as

$$L(k) = \log\Big(\frac{\sum_s \lambda_k^{i_0=+1}(s)}{\sum_s \lambda_k^{i_0=-1}(s)}\Big). \tag{2.3}$$

If $L(k) > 0$, we detect pixel (m, n) as $+1$; otherwise, it is detected as -1.

Fig. 2.1 shows the state and input block definitions, and the three inner products used in computing the branch metrics, for each of the four trellis stage types. Each trellis stage type has eight states. The state vector for the kth trellis stage is denoted $\mathbf{s}_k = [s_{k0}, s_{k1}, s_{k2}]$, where s_{kl} denotes the lth bit of the state vector at stage number k. When the stage number k is understood or is irrelevant to the discussion, we will drop it and write the state vector as $\mathbf{s} = [s_0, s_1, s_2]$. Similarly the vector of possible input bits at stage k is defined as $\mathbf{i}_k = [i_{k0}, i_{k1}]$ or $\mathbf{i}_k = [i_{k0}, i_{k1}, i_{k2}]$, depending on the trellis stage type. The mapping of state and input bits for each type of 2D state/input block is shown in Fig. 2.1.

The horizontal and vertical stages have four branches per state, while the diagonal stages have eight branches per state and are fully connected. None of the stages have parallel branches. Each trellis stage k corresponds to trellis state s_0 being located at a unique pixel position (m, n). The zig-zag scan thus defines a bijective map $T(\cdot)$ between trellis stages k and pixel locations (m, n): $k = T(m, n)$, and $(m, n) = T^{-1}(k)$.

Trellis generation for a sequence of up-right moves along a diagonal beginning with trellis stage k is initiated by placing state bit s_{k0} of the up-right state/input block at location $(m, n) = T^{-1}(k)$. The vector $\mathbf{i}_k$ of three input pixels can take eight different values. State transitions are determined by shifting the state/input block up and to the right to pick up the next three input pixels. The current inputs are then shifted into the next state: $\mathbf{s}_{k+1} = \mathbf{i}_k$. Trellis generation for the other three stage types is similar.

At each stage, the trellis branch output is a vector consisting of two or three 2×2 inner products between the index-reversed mask and the pixel values defined by the trellis. The horizontal and vertical moves use two inner products, while the up-right and down-left use

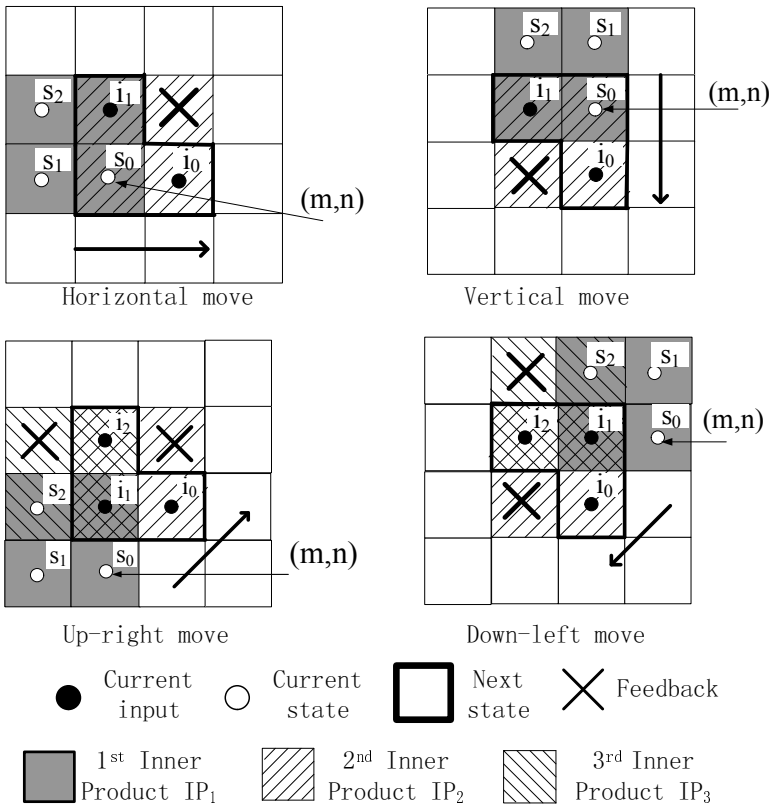


Figure 2.1: State and input definitions for the four trellis stage types, when the mask is 2×2.

two or three inner products. As seen in Fig. 2.1, the first inner product IP_1 uses only trellis pixels, but the second and third inner products IP_2 and IP_3 also use feedback pixels. The feedback pixel LLRs for a given MAP detector are obtained from previous iterations of the same detector (if only one detector is used), or from other MAP detectors employing different zig-zag scan orders (if more than one detector is used.) For the horizontal move, the branch metric δ_h is the squared Euclidean distance (SED) between the branch output and the received pixel vector $[r(m,n), r(m,n+1)]$:

$$\delta_h = [r(m,n) - \mathrm{IP}^h_{m,n}]^2 + [r(m,n+1) - \mathrm{IP}^h_{m,n+1}]^2, \tag{2.4}$$

where $\mathrm{IP}^h_{m,n}$ denotes the 2D inner product between the index-reversed and shifted mask

$\mathbf{h}^{-}_{m,n}$ and the horizontal move state/input block shown in Fig. 2.1; that is, $\mathrm{IP}^{h}_{m,n}$ and $\mathrm{IP}^{h}_{m,n+1}$ are the first and second inner products of the horizontal state/input block. Note that δ_h is a function of the state and input vectors $\mathbf{s}$ and $\mathbf{i}$, and also of the feedback pixel. Similarly, the vertical move branch metric δ_v is

$$\delta_v = [r(m,n) - \mathrm{IP}^{v}_{m,n}]^2 + [r(m+1,n) - \mathrm{IP}^{v}_{m+1,n}]^2. \tag{2.5}$$

For the up-right and down-left moves with three inner products, the branch metrics δ_{ur} and δ_{dl} are

$$\delta_{ur} = [r(m,n) - \mathrm{IP}^{ur}_{m,n}]^2 + [r(m-1,n+1) - \mathrm{IP}^{ur}_{m-1,n+1}]^2 + [r(m-1,n) - \mathrm{IP}^{ur}_{m-1,n}]^2, \tag{2.6}$$

$$\delta_{dl} = [r(m,n) - \mathrm{IP}^{dl}_{m,n}]^2 + [r(m+1,n-1) - \mathrm{IP}^{dl}_{m+1,n-1}]^2 + [r(m,n-1) - \mathrm{IP}^{dl}_{m,n-1}]^2. \tag{2.7}$$

When two IPs are used for the diagonal moves, the last squared terms in δ_{ur} and δ_{dl} are not used. Feedback pixels are specified by LLRs. In [27], we describe a zig-zag DFVA employing an eight state diagonal trellis with four pairs of parallel branches per state, and only one inner product per branch output. The performance of this DFVA is worse than that of [24]; the parallel branch outputs do not have large enough SED between them, leading to increased BERs. The ISDFZA state and input blocks in Fig. 2.1 are therefore

designed to avoid parallel branches; the two or three inner products per branch output in the ISDFZA allows all the input and state bits to affect the branch outputs, leading to larger SEDs between branches and lower BERs. Our simulations show that using two or three inner products on the diagonal moves of the ISDFZA gives the same performance at low SNR, but three inner products give better performance at high SNR. Therefore, we use three inner products in our simulations.

Horizontal and up-right moves have the same state definition, but vertical and down-left moves have another state definition. Thus, in the transition between horizontal/up-right moves and vertical/down-left moves the state definition must change "on the fly." We refer to this change as a *gearshift*. To accommodate gear-shifting, the zigzag scan must include the boundary right column and the boundary bottom row where the pixels are assumed to be -1. Horizontal to down-left gear shifting occurs in the top image row. Down-left to horizontal gear shifting occurs in the bottom boundary row. Vertical to up-right gear shifting occurs in the left image column, and up-right to vertical gear shifting occurs at the right boundary column. Since the gearshifts occur at the edges where at least two of the new state pixels are -1, and since one of the pre-gearshift states becomes a post-gearshift input, each post-shift branch maps to exactly one pre-shift state, so that trellis path continuity is maintained. Fig. 2.2 shows the mapping between pre- and post-gearshift states and branches for the horizontal-to-down-left gearshift. At position (a) the state and input vectors $[s_0, s_1, s_2]$ and $[i_0, i_1]$ are shown. At position (b) the horizontal move has been made and the new horizontal states are shown. At position (c) the state definition has gear-shifted, and the down-left states and inputs are shown. The solid and dotted trellis paths trace the connectivity between the pre- and post-gearshift states. Since the gear shift occurs at the

edge of the image, state bit s_2 is on the boundary and is known to be -1. Hence states $\{-1,-1,1\}$, $\{1,-1,1\}$, $\{-1,1,1\}$, and $\{1,1,1\}$ are eliminated at positions (a), (b) and (c); at position (c), additional states $\{-1,1,1\}$, and $\{1,1,1\}$ are eliminated since state bit s_1 is also on the boundary. The other gearshift directions have similar diagrams.

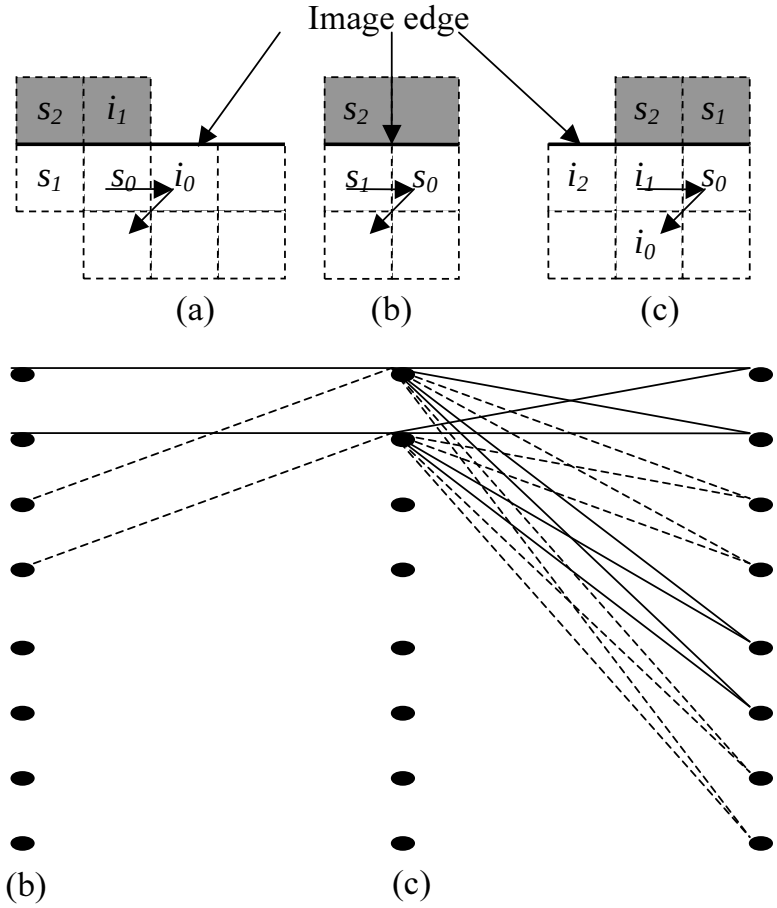


Figure 2.2: Horizontal to down-left gearshift: state definitions and corresponding trellis stages, when the mask is 2×2.

Trellis Definition for the 3×3 Mask

The source image for the 3×3 mask case is assumed to have double boundary columns of -1 pixels immediately to the left and right of the source image, and double boundary rows just above and just below the source image. Fig. 2.3 shows how the trellis states and

inputs for the four basic types of moves are defined for the 3×3 mask. For the horizontal and vertical moves, six pixels constitute the current state, so that there are $2^6 = 64$ states in total; three of these current state pixels become part of the next state. Three pixels constitute the current inputs ($2^3 = 8$ inputs in total), and also become part of the next state. The trellis branch output consists of a vector of two 3×3 inner products (IPs). The first IP has one feedback pixel, and the second IP has three. In total there are three feedback pixels for each of these two moves.

For the up-right and down-left moves, six pixels form the current state (64 states in total), only one of which becomes part of the next state. Five pixels form the current input (32 inputs in total), all of which become part of the next state. The trellis branch output consists of a vector of three IPs: the first IP has no feedback pixel; the second has three feedback pixels, and the third has one. In total there are four feedback pixels for each of these two moves.

As in the 2×2 case, the branch metric is the SED between the branch output and the received pixel vector. For the horizontal move, the metric is $[r(m,n) - \mathrm{IP}^h_{m,n}]^2 + [r(m,n+1) - \mathrm{IP}^h_{m,n+1}]^2$; for the vertical move, it is $[r(m,n) - \mathrm{IP}^v_{m,n}]^2 + [r(m+1,n) - \mathrm{IP}^v_{m+1,n}]^2$; for the down-left move, it is $[r(m,n) - \mathrm{IP}^{dl}_{m,n}]^2 + [r(m+1,n-1) - \mathrm{IP}^{dl}_{m+1,n-1}]^2 + [r(m,n-1) - \mathrm{IP}^{dl}_{m,n-1}]^2$; and for the up-right move, it is $[r(m,n) - \mathrm{IP}^{ur}_{m,n}]^2 + [r(m-1,n+1) - \mathrm{IP}^{ur}_{m-1,n+1}]^2 + [r(m-1,n) - \mathrm{IP}^{ur}_{m-1,n}]^2$. Gear shifting for the 3×3 mask follows a similar scheme to that previously described for the 2×2 mask.

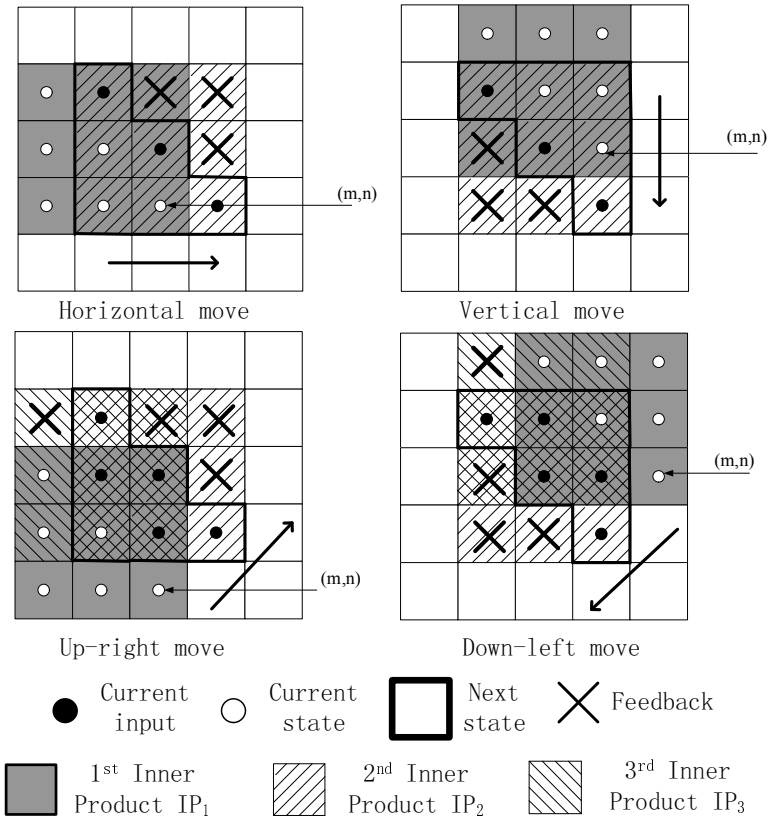


Figure 2.3: State and input definitions for the four trellis stage types, when the mask is 3×3.

Soft Input Consistency

As in 1D ISI, each signal sample occurs multiple times in the ISDFZA trellis. Interior image pixels appear three times; first they are used as input $f(m,n)$ or as state bit s_0; second they are used as state bit s_1; and third they are used as state bit s_2. Similarly, top-row and left-column pixels appear four times in the trellis, and right-column and bottom-row pixels appear twice. Fig. 2.4 provides an example of how the top-row pixel at location $(1,3)$ is used four times in the trellis.

The original image corresponds to a *consistent* trellis path, where pixel $f(m,n)$ assumes the same value wherever it appears (in state or input vectors) on the trellis. But, unlike 1D ISI, there are also a large number of *inconsistent* ISDFZA trellis paths, where

a pixel assumes different values depending on the trellis stage. This fact motivates us to enforce *soft input consistency*, i.e., to favor paths where all image pixels are consistent from start to end.

Fig. 2.5 outlines the proposed soft-input consistency scheme for an interior image pixel that is used three times. The idea is to construct an auxiliary factor graph with one node for each pixel (m, n), and pass messages between each auxiliary node and all trellis stages where its associated pixel (m, n) occurs. Each auxiliary node is connected to three trellis stages, and each trellis stage is connected to three auxiliary nodes. We note that, in his 2004 Ph.D. thesis [23], Marrow used an auxiliary factor graph to enforce bit constraints in different trellis locations of row-column based algorithms.

Message passing on the factor graph shown in Fig. 2.5 proceeds as follows (after [19]). The first step uses the $\lambda_k^{i_0}(s)$ probabilities from (2.2) to compute stage-to-node messages for each bit s_i of state vector $\mathbf{s} = [s_0, s_1, s_2]$. In the following discussion, we drop the superscript i_0 on $\lambda_k^{i_0}(s)$ for convenience. We denote the decimal conversion of the bits in $\mathbf{s}$ as s. The first time pixel (m, n) is used, it is used as state bit s_0; a probability for s_0 is computed by summing the λ probabilities over s_1 and s_2: $\lambda_k(s_0 = j) = \sum_{s_1} \sum_{s_2} \lambda_k(s_0 = j, s_1, s_2)$, $j \in \{-1, 1\}$, where $k = T(m, n)$ is the trellis stage corresponding to pixel (m, n). Then, we convert this probability into the stage-to-node message LLR $\mu_{s\to n}^k(s_0)$ using $\mu_{s\to n}^k(s_0) = \log(\lambda_k(s_0 = 1)/\lambda_k(s_0 = -1))$. In a similar manner, stage-to-node messages $\mu_{s\to n}^k(s_1)$ and $\mu_{s\to n}^k(s_2)$ are computed at the trellis stages where (m, n) is used as states s_1 and s_2. (For the example shown in Fig. 2.4, four stage-to-node messages about pixel $(1, 3)$ are passed to auxiliary node 6 from trellis stages 6, 7, 8, and 14.) In the second step in Fig. 2.5, the three stage-to-node messages are passed down to the kth auxiliary node,

and summed to form the node LLR $L_{\text{node}}(k) = \sum_{i=0}^{2} \mu_{s\to n}^{k}(s_i)$. The summation operation is based on the assumption that the $\lambda_k(s_i)$ probabilities are independent, since they are estimates from widely separated trellis stages. Summing the messages has an averaging effect, thereby encouraging consistency among the state bits connected to auxiliary node k. After all the $L_{\text{node}}(k)$s have been computed, messages are passed back up from the auxiliary nodes to the trellis stages to complete the third step. Consider trellis stage $k \equiv k_0 = T(m,n)$. At this stage, state s_0 is pixel (m,n) with node LLR $L_{\text{node}}(k_0)$, while states s_1 and s_2 are pixels $T^{-1}(k_1)$ and $T^{-1}(k_2)$ (corresponding to two other trellis stages k_1 and k_2) with node LLRs $L_{\text{node}}(k_1)$ and $L_{\text{node}}(k_2)$. (For the example shown in Fig. 2.4, three node-to-stage messages are passed to trellis stage 6 from auxiliary nodes 2, 6, and the boundary auxiliary node corresponding to pixel location $(0,2)$.) The node-to-stage message $\mu_{n\to s}^{k}(s_i)$ passed to state s_i of stage k is $\mu_{n\to s}^{k}(s_i) = L_{\text{node}}(k_i) - \mu_{s\to n}^{k_i}(s_i)$. (The previous stage-to-node message $\mu_{s\to n}^{k_i}(s_i)$ from state bit s_i at stage k to auxiliary node k_i is subtracted in order to ensure that the message passed from node k_i to stage k contains only extrinsic information.) Then, the $\mu_{n\to s}^{k}(s_i)$ are converted to probabilities $\lambda_k(s_i)$ using the standard transformation in [9]. Finally, the fourth step updates the $\lambda_k(s)$ at each stage k by multiplying by the three $\lambda_k(s_i)$ probabilities $\lambda_{k_c}^{i_0}(s) = \lambda_k^{i0}(s)\lambda_k(s_0)\lambda_k(s_1)\lambda_k(s_2)$, where subscript c indicates the soft input consistency update.

2.1.2 The Concatenated System

Fig. 2.6 shows block diagram of the concatenated system. The IRCSDFA uses serially concatenated row and column MAP detectors, as described in [9] and Chapter 1. The ISDFZA employs four zigzag MAP detectors as shown in Fig. 1.4. Within the IRCSDFA, extrinsic

information between rows and columns is weighted according to an optimized version of the weight-schedule reported in [9]. Within the ISDFZA, the four-detector weight schedule described in section 2.1.3 below is employed. Weight schedules for the extrinsic information $\tilde{L}_{ZZ2RC}$ and $\tilde{L}_{RC2ZZ}$ passed between the ISDFZA and the IRCSDFA are also specified in section 2.1.3. The first zigzag detector in the ISDFZA uses $\tilde{L}_{RC2ZZ}$ to compute the modified channel PDFs in equations (2.1). In the system of Fig. 2.6, the numbers m and n of inner iterations of the ISDFZA and IRCSDFA per outer iteration of the concatenated system have a significant effect on system performance. Joint optimization of the iteration and weight schedules is described in section 2.1.3.

2.1.3 Weight Optimization with EXIT Charts

Repeating the procedure we described in Chapter 1 for different inner weights (which in general may vary for different iterations) gives different EXIT curves. By increasing the inner weights, the output mutual information can be increased, though we have not found an explicit expression for this relation.

Unlike turbo codes, there is no convergence threshold SNR at which the EXIT curve is tangent to the line $I_A = I_E$; this lack of threshold is well known from previous 1D turbo equalizer papers [20]. In practice, we adjust the EXIT chart curves so that the trajectory found by taking vertical and horizontal projections between the curves for the two equalizers has uniform vertical and horizontal steps in each iteration. We observed experimentally that equalizing the trajectory step sizes allows the overall concatenated equalizer to achieve a lower BER for a given SNR; this policy tends to maximize the number of iterations between equalizers, thereby slowing the overall convergence rate and avoiding quick

convergence to an erroneous result.

To optimize the inner weights w of the zig-zag equalizer shown in Fig. 1.4, an EXIT chart using the zig-zag equalizer on both axes is drawn. The mutual information I_E at the output of the right-most zig-zag detector is a non-linear function of the inner weights. The outer weight w_4, in Fig. 2.6, which multiplies I_E, cannot be individually optimized this way because I_E is invariant under scaling. We sidestep this problem by setting the zig-zag weights w_1 through w_4 equal at each iteration, thereby reducing the number of degrees of freedom in the optimization. Equalizing the weights makes intuitive sense as it puts each of the constituent detectors on an equal footing within each iteration.

As a starting (sub-optimal) weight schedule for the four-detector 3×3 zig-zag equalizer, we used a schedule for the 2×2 averaging mask ISDFZA, which was optimized by Monte Carlo simulations. Fig. 2.7 shows an EXIT chart for this starting six iteration weight schedule. The I_A versus I_E curves for the first, third, and fifth iterations are plotted with I_A on the horizontal axis, and are labeled "ZZ1", "ZZ3", and "ZZ5"; the curves for even iterations two, four, and six are plotted with I_A on the vertical axis and similarly labeled. The weight schedule varies with iteration k according to $w_i(k) = 0.008 \times (5k^2 + 1)$, for $k = 0, \ldots, 5$, and $i = 1, \ldots, 4$. Overestimation of the LLRs due to soft decision feedback becomes less pronounced with successive iterations, and the LLRs become more reliable, making it advantageous to increase the weights in later iterations. Simple quadratic dependence of the weights on the iteration k works well experimentally, and also facilitates optimization by reducing the dimensionality of the search space. The iteration trajectory is found by starting with $I_A = 0$ on the horizontal axis, projecting a vertical line to the corresponding value of I_E on the ZZ1 curve, then projecting a horizontal line from the ZZ1

curve to the ZZ2 curve, etc.

Optimization of the zig-zag weight schedule using the EXIT chart technique led to the chart shown in Fig. 2.8; this scheme used the weight schedule $w_i(k) = 0.002 \times (k^2 + 1)$, $k = 1, \ldots, 10$, $i = 1, \ldots, 4$. The trajectory steps are clearly more uniform in Fig. 2.8 than in Fig. 2.7. Monte Carlo simulation results in Fig. 2.9 show that the optimized weight schedule gives a performance gain of about 2 dB at BER 10^{-3}.

All the EXIT charts here were drawn with channel SNRs in the middle SNR region, at BERs of about 10^{-3}, to facilitate trajectory optimization. However, it was observed that optimizing the weight schedule at mid-SNR led to improved performance at high SNR as well.

A six iteration weight schedule for the IRCSDFA, found by Monte Carlo simulation in [9], is $w(k) = 0.008 \times (3k^2 + 1)$, $k = 0, \ldots, 5$, where the two weights in Fig. 1.1 are set equal. EXIT chart optimization of this schedule gave only small performance improvements when the number of iterations was limited to six, but larger gains were obtained when eight iterations were allowed [10].

We now consider weight and iteration schedule optimization for the concatenated ISDFZA-IRCSDFA system shown in Fig. 2.6. This is a multi-dimensional optimization problem, involving joint optimization of the inner weights of the ISDFZA and IRCSDFA, the connection weights between them, and also the number of inner iterations m and n of the ISDFZA and IRCSDFA per outer iteration of the concatenated system. The dimensionality can be reduced by using the inner weight schedules found by optimization of the stand-alone zig-zag and row-column equalizers, described above. The connection weights are then fixed by using the 'tail weight' (the last inner weight of each inner system) as the

connection weight. Thus, weight w_4 from the zig-zag system is used as the connection weight from zigzag to row-column. Similarly, weight w_{c2r} from the row-column system is used as the connection weight from row-column to zig-zag. By doing this, we reduce the complexity of the optimization problem; the simulation results in section 2.2 also show that this is a good choice.

The iteration schedule is also optimized using EXIT charts. We draw the EXIT curves for the zig-zag and row-column equalizers under different numbers of inner iterations, combine them in one EXIT chart, and optimize the trajectory for uniform step sizes. With this method, changing the iteration scheme of one of the two systems means changing only one set of curves on the EXIT chart.

After comparing different iteration schedules, we found that two zig-zag and one row-column iterations per outer iteration (i.e., $m = 2$, $n = 1$) in Fig. 2.6 with six outer iterations gave the best simulation performance. Fig. 2.10 shows the EXIT chart and the trajectory of this weight and iteration schedule. For comparison, an EXIT chart for a $m = 1$, $n = 1$ schedule with eight outer iterations is shown in Fig. 2.11. The EXIT charts for both schedules were drawn at channel SNR of 10 dB. The step sizes in Fig. 2.10 are more uniform than those in Fig. 2.11. Simulation results for both schedules can be seen in Fig. 2.14 in section 2.2, and confirm the EXIT chart results.

For the best performing weight and iteration schedule, the six iteration row-column inner weight schedule was used. Even though the six iteration schedule was inferior to the eight iteration schedule for the stand-alone row-column equalizer, the six iteration schedule fit better with the concatenated system's six outer iterations. The $m = 1$, $n = 1$ iteration schedule used the eight iteration row-column inner weight schedule, as that fit best with the

eight outer iterations.

We do not claim that the above described EXIT-chart-based weight and iteration schedule optimization gives the best possible system performance, as we made several simplifying assumptions and did not exhaustively explore the full parameter space. However, the EXIT chart technique provides a feasible, systematic framework for optimization of the proposed equalizers, and allows much faster exploration of the parameter space than optimization by repeated Monte Carlo simulations of the entire equalizer.

2.2 Simulation Results

In this section, we present Monte Carlo simulation results for the ISDFZA and for the concatenated system of subsection 2.1.2. All simulations employ a random 64×64 binary image $f(m, n)$ with pixel values chosen from the alphabet $\{-1, +1\}$, except those in Fig. 2.12, where the image size is 128×128.

The plots presented below show the bit error rate (BER) of the estimated binary input image, versus SNR. The SNR is defined as in [24]:

$$\text{SNR} = 10 \log_{10} \left(\frac{\text{var}\left[\mathbf{f} * \mathbf{h}\right]}{\sigma_w^2} \right), \tag{2.8}$$

where $*$ denotes the 2D convolution in (1.2) and σ_w^2 is the variance of the Gaussian r.v.s $w(m, n)$. To compute the received image $r(m, n)$, we assume a boundary of -1 pixels around $f(m, n)$; the receiver uses this known boundary condition to simplify the trellis near edge pixels. Experimental results show that performance gains due to knowledge of the boundary are negligible at image sizes greater than 64×64, for the 2×2 averaging mask;

Table 2.1: Performance comparison of the concatenated system for the 2×2 mask with $\alpha = 0.5$, image size 20×20.

	SNR (dB)					
Algorithm	8	9	10	11	12	13
GBP	8.9*e*–3	3.0*e*–3	9.9*e*–4	2.5*e*–4	3.9*e*–5	4.1*e*–6
MAP	6.1*e*–3	2.4*e*–3	7.5*e*–4	1.9*e*–4	3.5*e*–5	4.1*e*–6
ZR concatenated	8.333*e*–3	3.5211*e*–3	7.5*e*–4	2.3*e*–4	3.5*e*–5	4.11*e*–6
ML upper bound	2.779*e*–2	6.056*e*–3	1.31*e*–3	2.555*e*–4	3.949*e*–5	4.241*e*–6
Q func lower bound	6.051*e*–3	2.311*e*–3	7.451*e*–4	1.911*e*–4	3.461*e*–5	4.10*e*–6

for larger mask sizes (with deeper boundaries) boundary effects may improve performance somewhat, but only at lower SNRs.

The ISDFZA-IRCSDFA also compares well to the GBP equalizer in [33], for the 6×6 and 20×20 image sizes tested in that paper. As in [33], we consider the mask $[\ 1 \quad \alpha;\ \ \alpha \quad 0\ ]$, where $\alpha = 0.5$ or 0.75, and the semi-colon indicates the start of a new row. Since the SNR definition (2.8) is different than that in [33], the GBP curves were shifted to correspond to (2.8). For the $\alpha = 0.5$ and 20×20 image size case, results are presented in Table 2.1; the MAP algorithm in this table refers to the MAP performance bound reported in [33]. The proposed algorithm performs at least as well as that of [33] at almost all SNRs. The algorithms actually perform slightly better than the ML bound at the highest SNRs shown in the table, as the ML bound has not yet converged. Similar comparison results were observed for the 6×6 image size cases reported in [33].

Fig. 2.12 shows ISDFZA simulation results for one, two, and four detectors, with and without soft input consistency. For comparison, results for the two-row IRCSDFA (with four trellis states) of [9] and for our implementation of the separable algorithm of [37] are also plotted, as is the ML bound of [11]. (Because [37] reports results only for one iteration

of the separable equalization algorithm without LDPC coding, we implemented and tested the separable algorithm for multiple iterations. We found that two iterations of the separable algorithm give virtually all the available performance gain, so two iterations are used for all separable algorithm results reported in this paper.) Enforcing soft input consistency gives an SNR gain of about 3 dB, and is therefore an important part of the ISDFZA. Going from one to four detectors gives an additional 1 dB gain. At SNRs between 2 and 6 dB, the ISDFZA outperforms the IRCSDFA and the separable algorithm by at least 1 dB, when either two or four detectors are used. But at high SNR, the ISDFZA with four detectors is about 0.5 dB worse than the two-row IRCSDFA and about 1 dB worse than the separable algorithm.

Fig. 2.13 shows the performance of the concatenated ISDFZA-IRCSDFA system for the 2×2 averaging mask, after optimizing the weight scheme by simulations. The ISDFZA weight scheme is $w_i(k) = 0.008 \times (5k^2 + 1)$, for $k = 0, \ldots, 5$ and $i = 1, \ldots, 4$, and the IRCSDFA weights are varied with iteration as $w(k) = 0.008 \times (3k^2 + 1)$, as in [9]. One inner iteration of both the ISDFZA and IRCSDFA is done per outer iteration (i.e., $m = n = 1$). For comparison, the four-row IRCSDFA (with 16 trellis states, and the best performance) of [9] and the separable algorithm of [37] are also plotted, as are the ML upper bound and the Q function lower bound from [11]. At low SNR, the concatenated algorithm performs as well or better than ISDFZA of Fig. 2.12. At high SNR (e.g., at BER 10^{-4}) higher reliability extrinsic information from the IRCSDFA reduces the error rate from the ISDFZA MAP detectors, thereby allowing the concatenated algorithm to achieve a gain of about 2 dB over the ISDFZA. The concatenated system outperforms [9] and [37] by at least 0.8 dB at BER 10^{-2} and 0.4 dB at BERs 10^{-3} and 2×10^{-5}, and overlays the

ML bound at BER 2×10^{-5}.

The concatenated system in Fig. 2.13 also outperforms an earlier version reported in the conference publication [28] by about 0.4 dB at BER 2×10^{-5}. Improvements over the earlier version were due to weight schedule optimization of both the ISDFZA and the IRCSDFA, and to changing the order so that the ISDFZA comes first in the outer iteration schedule, as shown in Fig. 2.6.

Fig. 2.14 shows simulation results for the concatenated system with the 3×3 averaging mask. The two iteration schemes mentioned in section 2.1.3 are plotted: the 2 zig-zag 1 row-column 6 outer iterations scheme with $m = 2$ and $n = 1$ (labeled "ZR 2ZZ1RC"), and the 1 zig-zag 1 row-column 8 outer iterations scheme (labeled "ZR 1ZZ1RC"). Also plotted are the ML bound and the optimized IRCSDFA ("single row-column") and the optimized four-detector ISDFZA ("single zig-zag"). At BER 10^{-3}, the $m = 2$, $n = 1$ scheme gains about 0.3 dB over the $m = 1$, $n = 1$ scheme, about 0.7 dB over the IRCSDFA, and about 2.1 dB over the ISDFZA. At BER 10^{-4}, the best concatenated scheme is within about 0.8 dB of the ML bound.

The concatenated ISDFZA-IRCSDFA system was also tested in two "easier" 3×3 channels, i.e., channel A and channel B as defined in [11]. These channels have masks of the form $[\ \beta \quad \alpha \quad \beta;\ \ \alpha \quad 1 \quad \alpha;\ \ \beta \quad \alpha \quad \beta\ ]$, where the semi-colon indicates the start of a new row. In channel A, $\alpha = 0.181$, $\beta = 0.0327$; in channel B, $\alpha = 0.352$, $\beta = 0.0993$. Fig. 2.15 compares performance of the concatenated system (labeled "ZR") and the IRCSDFA (labeled "single RC") on these channels, and also shows the ML upper bounds and Q function lower bound. On channel A, both the concatenated and row-column equalizers achieve the ML bound at high SNR. On channel B, the concatenated system gains about 0.6

dB over the IRCSDFA at BER 10^{-5}, and about 1.5 dB over the algorithm proposed in [4] at BER 10^{-4}. For channel B, the concatenated system outperforms the ML bound at lower SNRs because the ML union bound has not yet converged, and because the concatenated system exploits the known two-pixel-deep boundary, which is not taken into account by the ML bound.

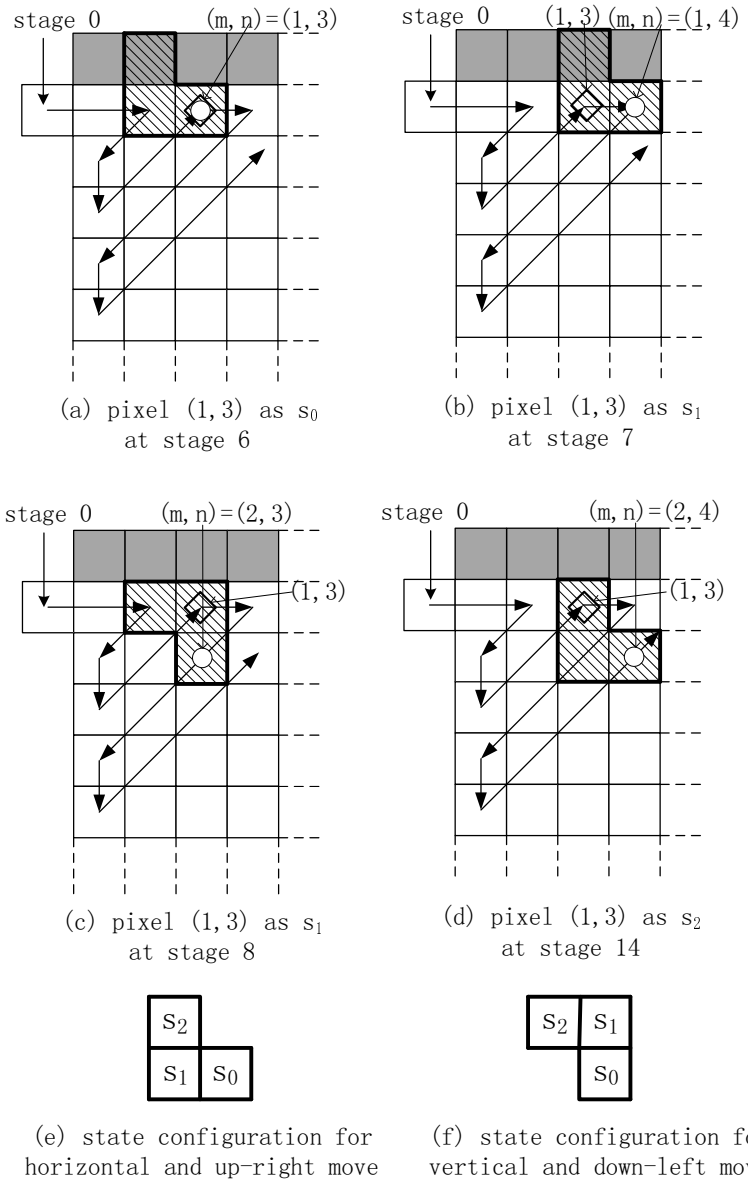
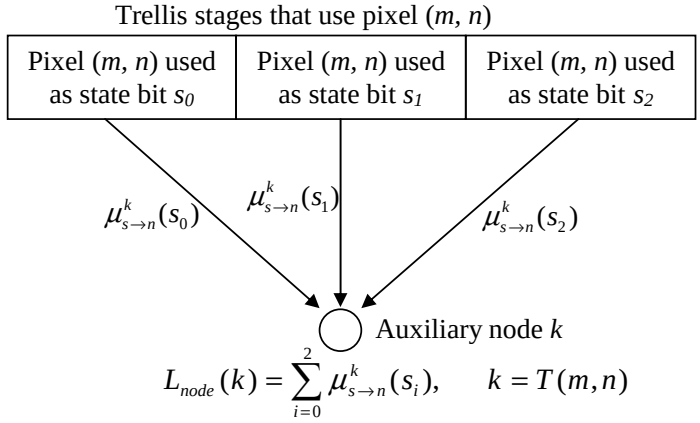


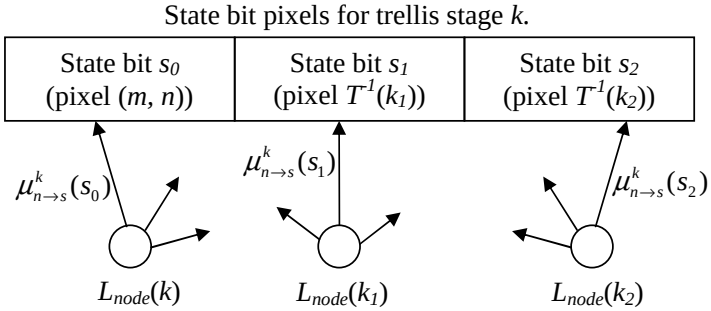
Figure 2.4: Pixel (1,3) is used four times in the zig-zag trellis.

1): Compute stage-to-node LLRs $\mu_{s\to n}^{k}(s_i)$ from the λ_k probabilities.

2): Compute the $L_{node}(k)$ by passing messages to the nodes.



3): Pass node-to-stage LLR messages back up.



4): Update the λ_k probabilities with the node-to-stage messages.

$$\lambda_{k_c}^{i_0}(s) = \lambda_k^{i_0}(s) \cdot \lambda_k(s_0) \cdot \lambda_k(s_1) \cdot \lambda_k(s_2)$$

Figure 2.5: Soft input consistency

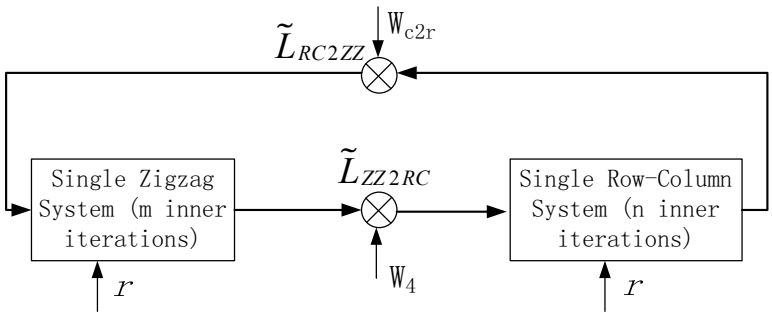


Figure 2.6: Concatenated zig-zag and row-column detectors for 2D-ISI reduction.

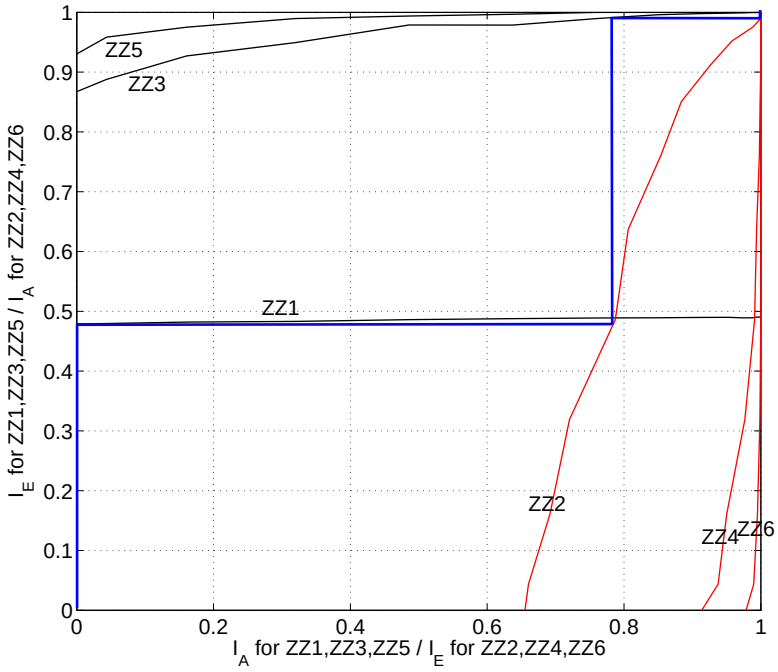


Figure 2.7: The EXIT chart and trajectory of the original 6 iterations weight scheme for the four-detector zig-zag equalizer.

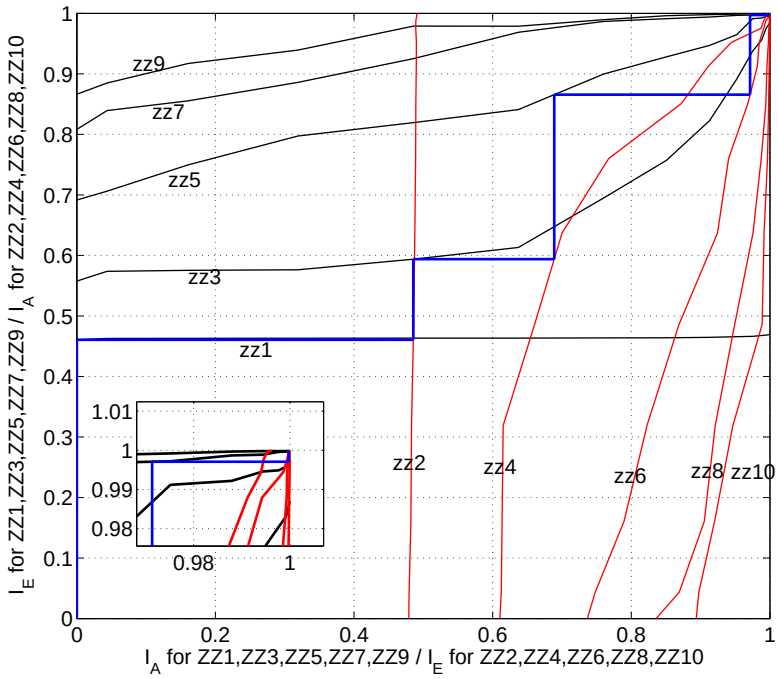


Figure 2.8: The EXIT chart and trajectory of the optimized 10 iterations weight scheme for the four-detector zig-zag equalizer.

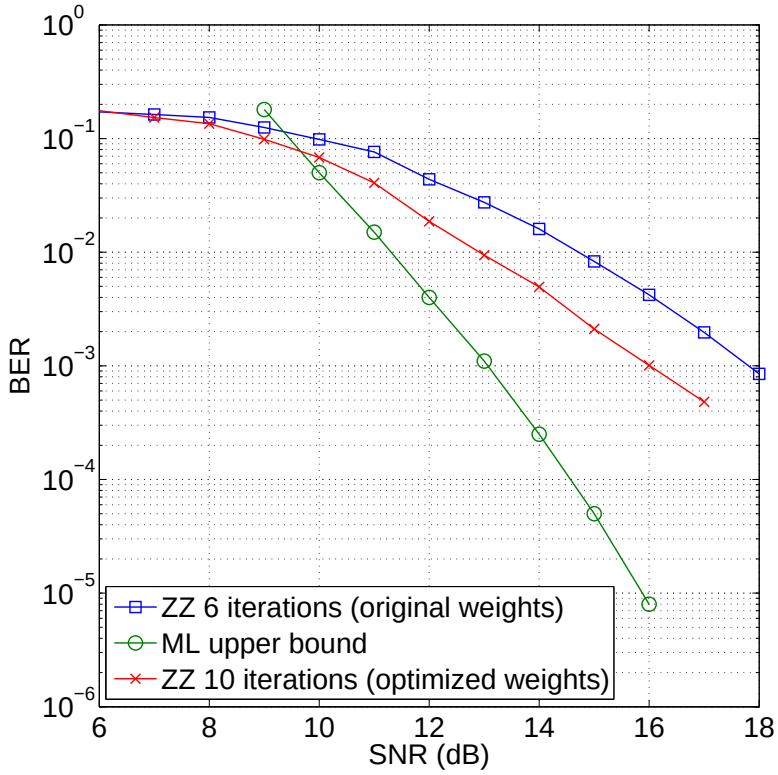


Figure 2.9: The performance of the four-detector zig-zag algorithm with the original weight scheme and the optimized weight scheme found with EXIT charts, on the 3×3 averaging mask channel.

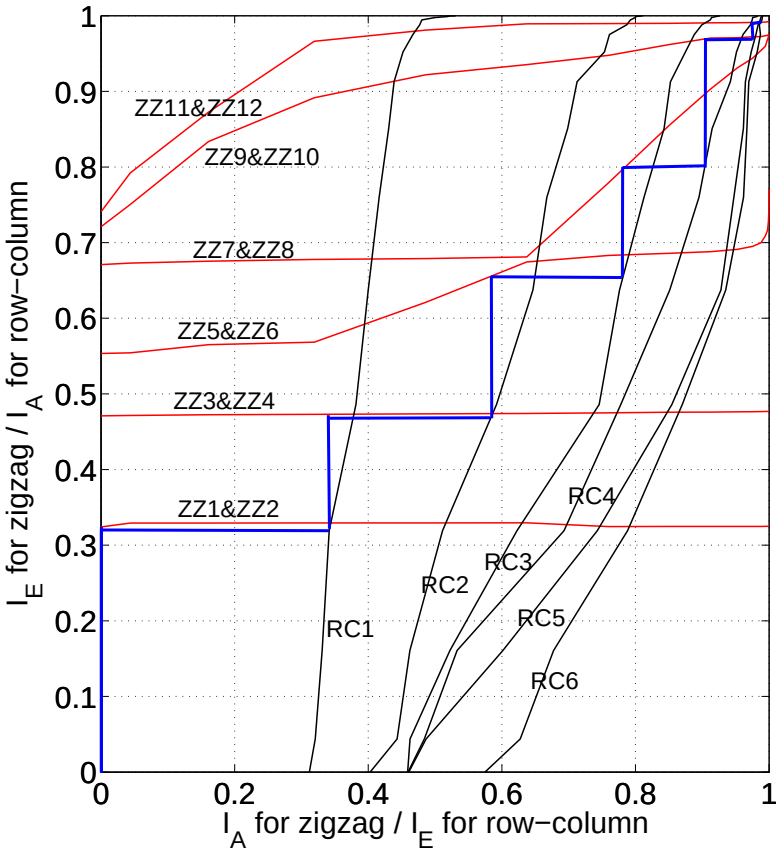


Figure 2.10: The EXIT chart and trajectory of the best six outer iterations weight scheme for the concatenated system at 10 dB. Each outer iteration consists of two zig-zag and one row-column inner iterations. This weight and iteration schedule gave the best performance among all schedules tested for the concatenated system.

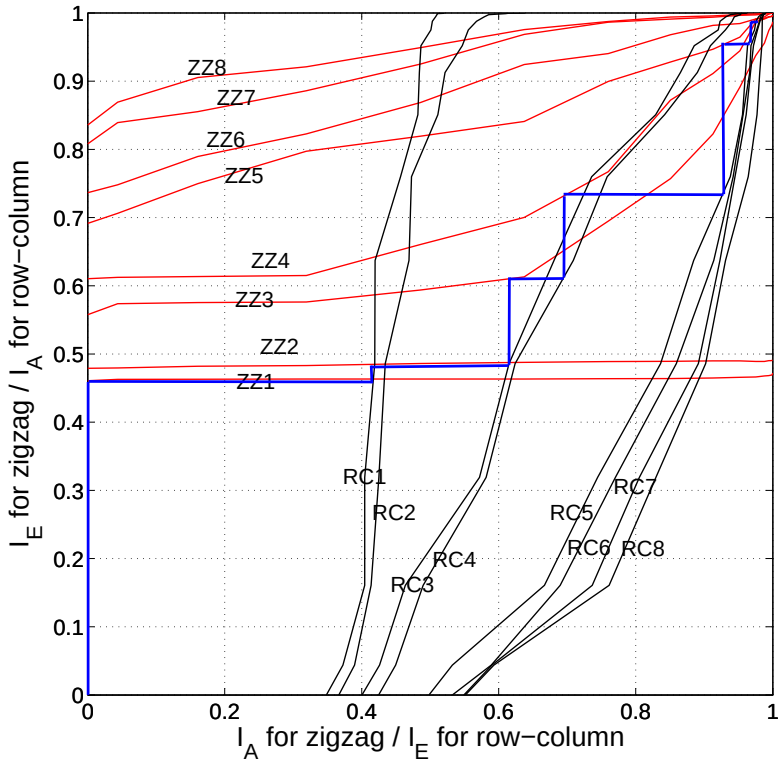


Figure 2.11: The EXIT chart and trajectory of the eight outer iterations weight scheme for concatenated system at 10 dB. Each outer iteration consists of one zig-zag and one row-column iteration. (The trajectory ends at the sixth iteration since the last two iterations are in a very tiny area.)

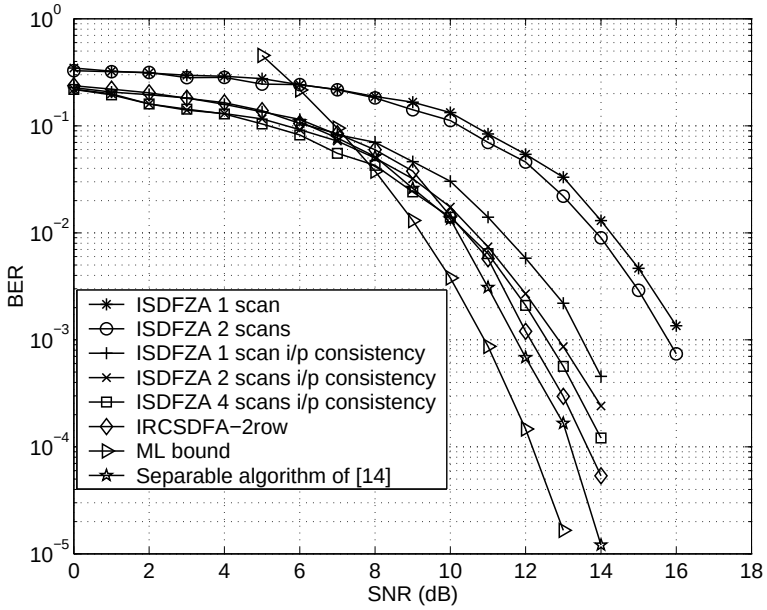


Figure 2.12: Monte Carlo simulation results for the ISDFZA algorithm, when the received image has been blurred by the 2×2 averaging mask and AWGN. Results for the IRCSDFA of [9] and the separable algorithm of [37] are plotted for comparison.

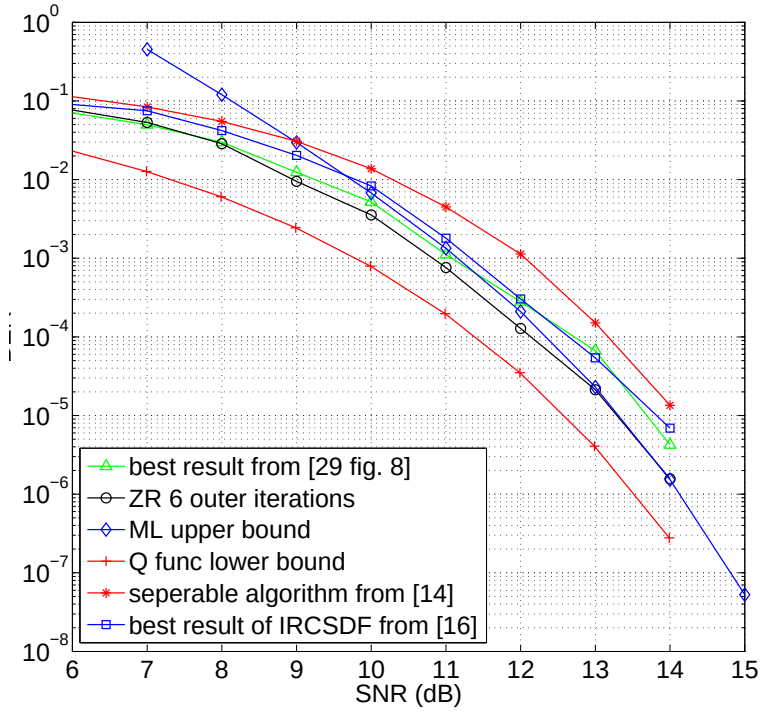


Figure 2.13: Monte Carlo simulations for the ISDFZA-IRCSDFA system with 2×2 averaging mask. Results for the IRCSDFA [9], the separable algorithm [37], and an earlier version [28] of the ISDFZA-IRCSDFA are plotted for comparison.

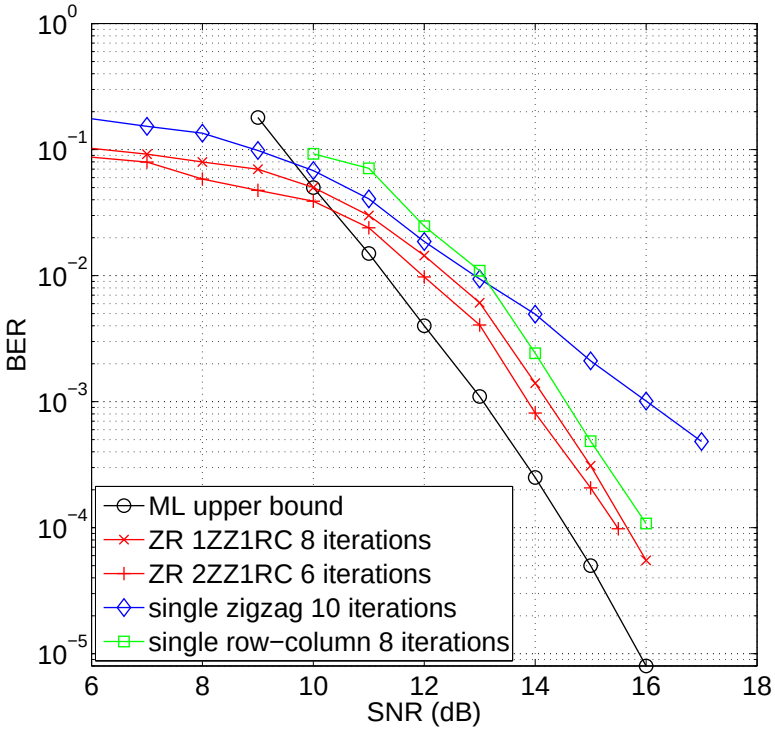


Figure 2.14: Performance comparison of the concatenated system under two different weight and iteration schedules for the 3×3 averaging mask.

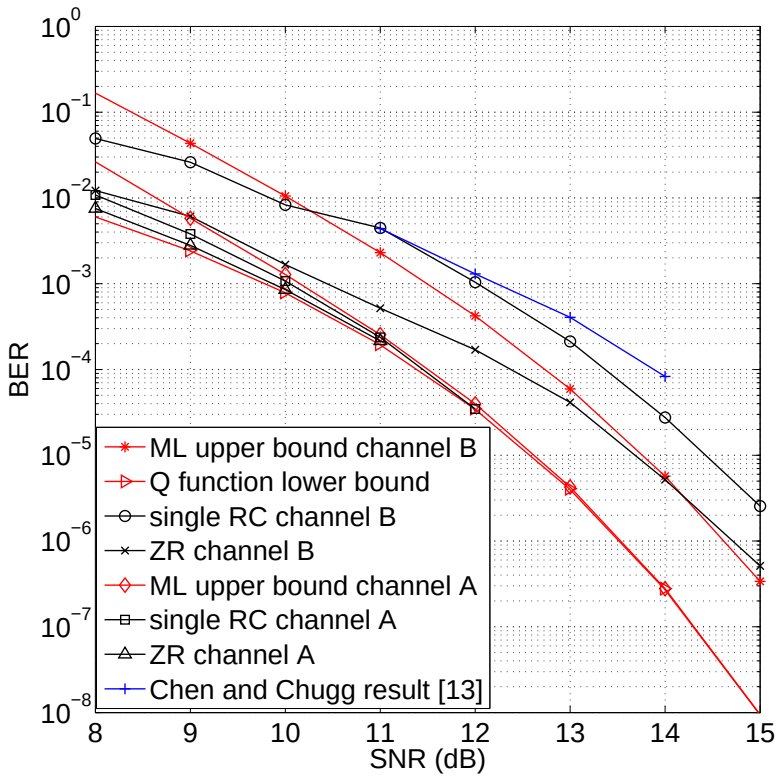


Figure 2.15: The performance of the row-column algorithm and zigzag-row-column concatenated system on channel A and channel B of [4]. Both channels share the same Q function lower bound.

Chapter 3

Iterative Row Column Soft Decision Feedback Algorithm Using Joint Extrinsic Information

In this chapter, we redesign the IRCSDF algorithm by dropping the independence assumption on the extrinsic information exchanged between the row and column detectors; we call the resulting algorithm the block (BLK) algorithm. We estimate and exchange joint statistics for the pixels involved in the extrinsic information exchange. To address the increased computational and storage complexity introduced by the joint statistics, we have developed a simplified version of the block (SBLK) algorithm. Experimental results demonstrate that the SBLK algorithm performs almost as well as the BLK algorithm.

3.1 Block Algorithm

We follow the notations and derivation of the IRCSDF algorithm (assuming independence of pixels in extrinsic information) from [9], [7]. For lack of space, we will present only the most relevant equations from [9], which get modified in the BLK algorithm. Figures 3.1 (a) (2×2 mask) and 3.2 (a) (3×3 mask) show the state (labeled s), input (labeled i), and feedback (labeled ω) pixels for the row detector. Figures 3.1 (d) and 3.2 (d) show similar

information for the column detector. The main modification is in the computation of γ in the traditional BCJR algorithm [1]. Computation of γ in the traditional BCJR algorithm [9] (see equations (4.2), and (3.2) below) requires apriori probability of the input block (two pixels for 2×2 mask and three pixels for 3×3 mask) and apriori probability of the feedback block (two pixels for 2×2 mask and six pixels for 3×3 mask), from the extrinsic information obtained from the previous detector in the overall iterative scheme. In the sequel, we present these equations for the 3×3 mask; obvious changes lead to expressions for the 2×2 case.

$$\gamma_{\mathbf{i}}(\mathbf{y}_k, s', s) = p'(\mathbf{y}_k \mid \mathbf{U} = \mathbf{i}, S_k = s, S_{k-1} = s') \times P(\mathbf{y}_k \mid s, s') \times P(s \mid s'), \tag{3.1}$$

where

$$\begin{aligned} & p'(\mathbf{y}_k \mid \mathbf{U} = \mathbf{i}, S_k = s, S_{k-1} = s') \\ & = P(y_{k2} \mid i_0, i_1, i_2, s, s') \times \Bigg[\sum_{\Omega_2} P(\Omega_2) P(y_{k1} \mid i_0, i_1, s, s', \Omega_2) \\ & \times \sum_{\Omega_1} P(\Omega_1) P(y_{k0} \mid i_0, s, s', \Omega_1, \Omega_2) \Bigg]. \end{aligned} \tag{3.2}$$

Here k represents the trellis stage, $\mathbf{y}_k = (y_{k0}, y_{k1}, y_{k2})$ is the received vector, $\mathbf{i} = (i_0, i_1, i_2)$ is the input vector, s, s' are the current and previous states, and Ω_1, Ω_2 represents the two rows of feedback pixels. In [9], [7], for simplicity, we assumed that the pixel estimation in the input/feedback block were statistically independent. Consequently, marginal probabilities (from the extrinsic information) were multiplied to obtain the required joint

probabilities. In other words, we set (for input pixel block):

$$P(s \mid s') = P(\mathbf{U} = \mathbf{i}) = P(i_0) \times P(i_1) \times P(i_2) \tag{3.3}$$

and for feedback pixel block:

$$\begin{aligned} P(\Omega_1) &= P(\omega_0) \times P(\omega_1) \times P(\omega_2) \\ P(\Omega_2) &= P(\omega_3) \times P(\omega_4) \times P(\omega_5). \end{aligned} \tag{3.4}$$

The independence assumption in (3.3) and (3.4) is strictly speaking not valid in practice; this was verified based on the joint probabilities obtained in our block algorithm. The key idea is to replace equations (3.2)–(3.4) with equations (3.5)–(3.7) shown below, respectively.

$$\begin{aligned} &p'(\mathbf{y_k} \mid \mathbf{U} = \mathbf{i}, S_k = s, S_{k-1} = s') = \\ &P(y_{k2} \mid i_0, i_1, i_2, s, s') \times \Bigg[\sum_{\Omega_1, \Omega_2} P(\Omega_1, \Omega_2) \\ &P(y_{k1} \mid i_0, i_1, s, s', IP_2(\Omega_2)) P(y_{k0} \mid i_0, s, s', IP_1(\Omega_1, \Omega_2)) \Bigg], \end{aligned} \tag{3.5}$$

$$P(s \mid s') = P(\mathbf{U} = \mathbf{i}) = P(i_0, i_1, i_2), \tag{3.6}$$

$$P(\Omega_1, \Omega_2) = P(\omega_0, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5). \tag{3.7}$$

We now address the problem of obtaining the joint probabilities in equations (3.6) and (3.7). In the BCJR algorithm we compute $\alpha_k(s) = P(S_k = s, \mathbf{y}_1^K)$, $\beta_k(s) = P(\mathbf{y}_{k+1}^{N_r} | S_k =$

s) and $\gamma_{\mathbf{i}}(\mathbf{y}_k, s, s') = P(\mathbf{U} = \mathbf{i}, S_k = s, \mathbf{y}_k \mid S_{k-1} = s')$ by a forward-backward procedure. We then compute $\lambda_k^{\mathbf{i}}(s) = \sum_{s'} \alpha_{k-1}(s')\beta_k(s)\gamma_{\mathbf{i}}(\mathbf{y}_k, s, s')$, which gives the unnormalized probability $P(\mathbf{U} = \mathbf{i})$; corresponding log likelihood ratio (LLR) is given by (in the binary case, there is one LLR per pixel, the "other LLR" being 0):

$$L(k) = \log\left(\frac{\sum_s \lambda_k^{i0=+1}(s)}{\sum_s \lambda_k^{i0=-1}(s)}\right). \tag{3.8}$$

In our block (BLK) algorithm, we compute the joint probability of the block of pixels that constitute the state and input (block of four pixels for 2×2 mask and block of nine pixels for 3×3 mask) as shown in Figures 3.1 (a) and 3.2 (a). We first compute the same α, β; γ is computed based on (3.1), (3.5)–(3.7). For computing λ, we do not sum over s'; i.e, $\lambda_k^i(s, s') = P(\mathbf{U} = \mathbf{i}, S_k = s, S_{k-1} = s', \mathbf{y}_1^{N_r})$, which is the joint block probability. Instead of computing two LLRs (one of them being equal to zero) for each pixel, we now have $2^4 = 16$ for 2×2 mask, and $2^9 = 512$ for 3×3 mask LLRs, as shown in equation (3.9). Here, for 2×2 mask $\mathbf{i} = (i_0, i_1)$, $\mathbf{i0} = (-1, -1)$, $S = (s_0, s_1)$, $S_0 = (-1, -1)$, $S' = (s'_0, s'_1)$, $S'_0 = (-1, -1)$. For 3×3 case, $\mathbf{i} = (i_0, i_1, i_2)$, $\mathbf{i0} = (-1, -1, -1)$, $S = (s_0, s_1, s_2, s_3, s_4, s_5)$, $S_0 = (-1, -1, -1, -1, -1, -1)$, $S' = (s'_0, s'_1, s'_2, s'_3, s'_4, s'_5)$, and $S'_0 = (-1, -1, -1, -1, -1, -1)$.

$$L_k(\mathbf{i}, s) = \log\left(\frac{\lambda_k^{\mathbf{i}}(S, S')}{\lambda_k^{\mathbf{i}=\mathbf{i0}}(S = S_0, S' = S'_0)}\right). \tag{3.9}$$

In other words, we compute a 16-valued LLR at each pixel (512-valued LLR for 3×3 mask), instead of a single LLR at each pixel, when we used the independence assumption.

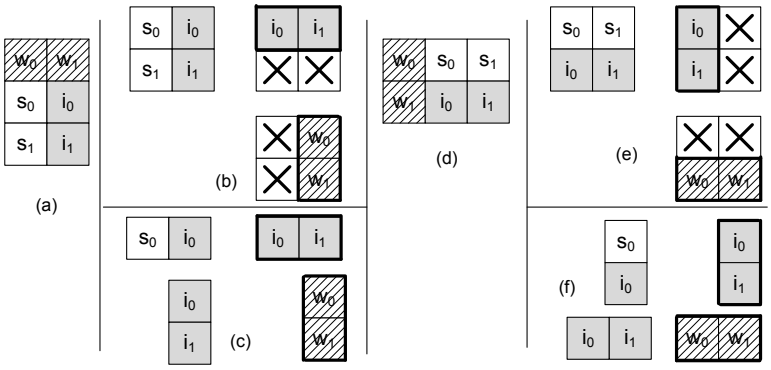


Figure 3.1: Structure definition of the BLK and SBLK algorithm for 2×2 mask, (a) state and input pixels definition of row detector (b) joint block from the row detector and the marginalization applied in the column detector for the BLK algorithm (c) joint pairs from the row detector and their roles in the column detector for the SBLK algorithm (d) state and input pixels definition of column detector (e) joint block from the column detector and the marginalization applied in the row detector for the BLK algorithm (f) joint pairs from the column detector and their roles in the row detector for the SBLK algorithm. (In (b) and (e), 'X' indicates the pixels which are marginalized out)

Before passing this information to the next detector (in an iterative scheme), the input extrinsic LLR is subtracted from the LLR computed in (3.9). It is customary to also multiply the extrinsic LLR by a weight factor less than one, since the extrinsic information is not reliable, at least in the initial iterations. Designing an optimal weight schedule (as a function of iteration number) has been discussed in [7], under the independence assumption. Finally, note that, the joint probability of the four/nine-pixel block can be suitably marginalized to obtain the joint input probability and joint feedback probability required in equations (3.6) and (3.7). This marginalization is shown in Figures 3.1 (b, e) and 3.2 (b, e).

For the current BLK algorithm, we argue next that (a) it is not necessary to subtract the input extrinsic LLR before passing it on to the next detector and (b) a constant weight scheme (single parameter) results in good performance, thereby avoiding the need for a sophisticated EXIT chart based weight scheme design [7].

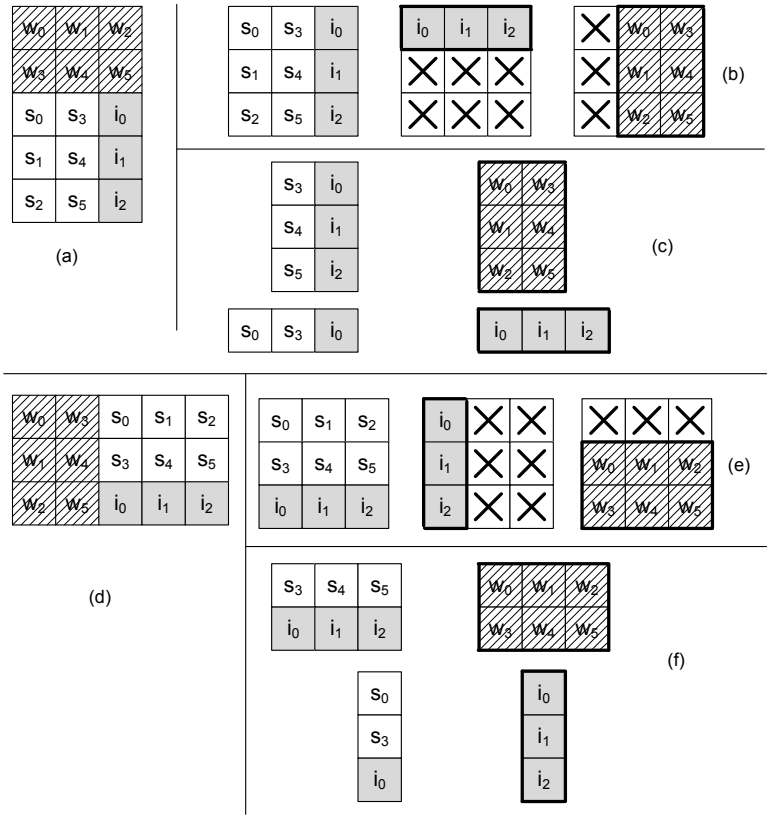


Figure 3.2: Structure definition of the BLK and SBLK algorithm for 3×3 mask, (a) state and input pixels definition of row detector (b) joint block from the row detector and the marginalization applied in the column detector for the BLK algorithm (c) joint pairs from the row detector and their roles in the column detector for the SBLK algorithm (d) state and input pixels definition of column detector (e) joint block from the column detector and the marginalization applied in the row detector for the BLK algorithm (f) joint pairs from the column detector and their roles in the row detector for the SBLK algorithm. (In (b) and (e), 'X' indicates the pixels which are marginalized out)

3.2 Subtraction of Input LLR

As discussed in [32], we subtract the input LLR, which corresponds to an apriori probability (APP), before passing it to the next detector. Based on the modified LLR in (3.9), the λ computation formula and the input APP terms in our BLK algorithm, we derived a similar subtraction term for our BLK algorithm. However, with the subtraction of (joint) input APP, the overall performance of the algorithm, although better than that with independence assumption, was not as good as the one without such subtraction. This observation was

initially puzzling but can be explained based on the two dimensional extrinsic information flow (TEXIF) chart of our BLK algorithm and that of the original algorithm.

TEXIF charts are a form of high-level factor graph [19] for 2D detection algorithms. The difference between a TEXIF chart and a standard factor graph is that the TEXIF chart focuses on the information exchange between different detectors based on the information's 2D structure, while the factor graph only shows the information flow inside one detector, which is of less importance when considering the interface between detectors, and factor graph really does not take into account spatial relationships between the information flows.

TEXIF chart analysis leads to the a better explanation of the subtraction issue for the joint IRCSDFA than our previous argument in [5]. In the joint IRCSDFA, there also exists a 'one step shift' in the horizontal direction for the row detector and the vertical direction for the column detector ((a) in Fig. 3.3) (more details in next chapter Section 4.3). If we consider the TEXIF chart for a 4×4 source image and start from any block in the inner image, we find that there is no loop for either the joint input pixels or the joint feedback pixels as the detection process proceeds through a full iteration. After a sufficient number of iterations, eventually both the joint input trajectory and the joint feedback trajectory end up at the boundary of the image, as shown in Fig. 3.4. The lack of loops in the joint IRCSDFA's TEXIF chart is a better explanation of why we do not need to use any subtraction for that algorithm than the argument presented in [5].

We will find that there exist the 'one step shift' in both the horizontal direction for the row detector and the vertical direction for the column detector ((a) in Fig. 3.3). Based on the similar analysis for ISDFZA, if consider the TEXIF chart of decoding a sample image size 4 x 4, and start form any block in the inner image, we will find that there will be no

loop for either the joint input pixels or the joint feedback pixels as the decoding process going on. And both the joint input trajectory and the joint feedback trajectory will end up at the boundary of the image at some detector (row/column) in some iteration. And that means there is no loop in the TEXIF chart of IRCSDFA, (see Fig. 3.4) which is also a more accurate explanation on why we do not need the subtraction for that algorithm than the argument we presented in our last conference paper [5].

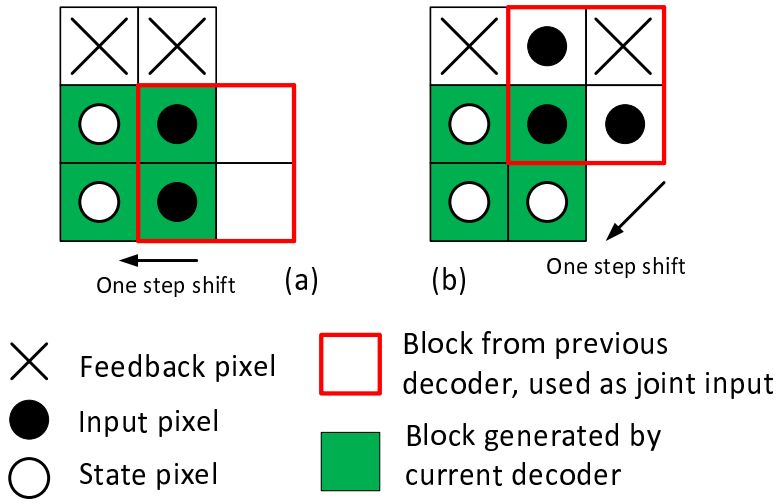


Figure 3.3: (a) 'one step shift' in IRCSDFA (b) 'one step shift' in ISDFZA

This 'one step shift' can only happen when using the joint statistics, since in the independent assumption, pixels in two neighboring detectors has to match each other exactly on the locations. so we could say that the joint statistics and the permutation structure of our RC or ZZ algorithm, enable us to eliminate the loops or enlarge the loop length based on the TEXIF chart of the corresponding algorithm, so that we could ignore the subtraction step or develop a more generalized one.

Another argument supporting the elimination of the subtraction step is that in the BLK (and SBLK to be discussed later) algorithm, the LLR (or APP) after the row detector is independent of the LLR after the column detector in the same iteration. This is unlike the

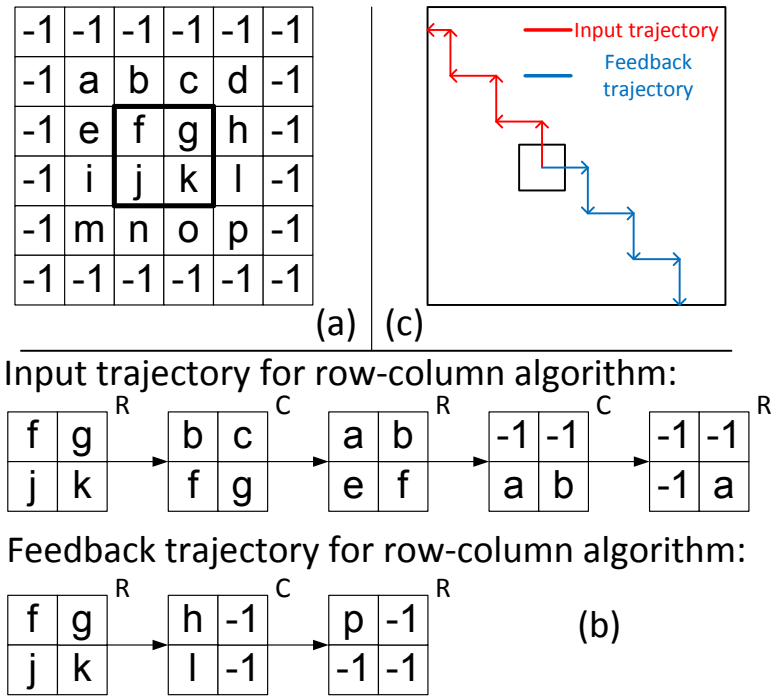


Figure 3.4: (a) Sample image size 4×4 with boundary (b) sample trajectories for the input pixels and feedback pixels for the block [f g; j k] in the row-column algorithm (c) trajectories of both the input pixels and feedback pixels for any inner block, showing absence of loops in the TEXIF chart

case in our previous algorithm [9]. In other words, we can experimentally verify that

$$P(i^R) \times P(i^C) = P(i^R, i^C), \tag{3.10}$$

where i^R is the pixel i after row detector, whereas i^C is the same pixel i after column detector. We do that by verifying

$$
\begin{aligned}
P(i^R \mid R^+)P(i^C \mid C^+) &= P(i^R, i^C \mid R^+, C^+) \\
P(i^R \mid R^+)P(i^C \mid C^-) &= P(i^R, i^C \mid R^+, C^-) \\
P(i^R \mid R^-)P(i^C \mid C^+) &= P(i^R, i^C \mid R^-, C^+) \\
P(i^R \mid R^-)P(i^C \mid C^-) &= P(i^R, i^C \mid R^-, C^-)
\end{aligned}
\tag{3.11}
$$

and

$$
\begin{aligned}
P(R^+)P(C^+) &= P(R^+, C^+) \\
P(R^+)P(C^-) &= P(R^+, C^-) \\
P(R^-)P(C^+) &= P(R^-, C^+) \\
P(R^-)P(C^-) &= P(R^-, C^-).
\end{aligned}
\tag{3.12}
$$

Note that if (3.11) and (3.12) hold, then

$$
\begin{aligned}
P(i^R)P(i^C) &= [P(i^R \mid R^+)P(R^+) + P(i^R \mid R^-)P(R^-)] \\
&\times [P(i^C \mid C^+)P(C^+) + P(i^C \mid C^-)P(C^-)] \\
&= P(i^R \mid R^+)P(i^C \mid C^+)P(R^+)P(C^+) \\
&+ P(i^R \mid R^+)P(i^C \mid C^-)P(R^+)P(C^-) \\
&+ P(i^R \mid R^-)P(i^C \mid C^+)P(R^-)P(C^+) \\
&+ P(i^R \mid R^-)P(i^C \mid C^-)P(R^-)P(C^-) \\
&= P(i^R, i^C \mid R^+, C^+)P(R^+)P(C^+) \\
&+ P(i^R, i^C \mid R^+, C^-)P(R^+)P(C^-) \\
&+ P(i^R, i^C \mid R^-, C^+)P(R^-)P(C^+) \\
&+ P(i^R, i^C \mid R^-, C^-)P(R^-)P(C^-) \\
&= P(i^R, i^C \mid R^+, C^+)P(R^+, C^+) \\
&+ P(i^R, i^C \mid R^+, C^-)P(R^+, C^-) \\
&+ P(i^R, i^C \mid R^-, C^+)P(R^-, C^+) \\
&+ P(i^R, i^C \mid R^-, C^-)P(R^-, C^-) = P(i^R, i^C),
\end{aligned}
\tag{3.13}
$$

which establishes (3.10). In equations (3.11), (3.12), and (3.13), R^+ means "this pixel is decoded as $+1$ after row detector," whereas R^- means "this pixel is decoded as -1 after row detector;" similarly for C^+ and C^-.

We verified (3.11) by plotting the corresponding histograms (for right-hand and left-hand sides) based on Monte Carlo simulation. Equation (3.12) was verified using counts of

the four possible configurations (again, based on Monte Carlo simulation) in both left-hand and right-hand sides of the equation. The norm of the difference vector (between the two sides of the equation) was less than 0.1%.

The above observations and our simulation results show that no subtraction of input LLR (APP) is required for our block algorithm. In the following section, we address the complexity of the block algorithm and present a simplified block (SBLK) algorithm.

3.3 Simplified Block Algorithm

Note that the block algorithm requires the exchange of 16-valued (512-valued) LLR for each pixel for the 2×2 (3×3) mask. This involves significantly more storage and computation compared to our original IRCSDF algorithm, where there was a single LLR for each pixel. We now present a simplified version of the block algorithm, which we call the SBLK algorithm.

The key idea is to store and exchange LLRs only for the joint pairs that we need in the next detector. This is illustrated in Figures 3.1 (c, f) and 3.2 (c, f). For the 2×2 mask, we only need to store two pixel pairs which is $2^2 \times 2 = 8$ LLRs — half the 16 LLRs required in the BLK algorithm. For the 3×3 mask, we only need $2^6 + 2^3 = 72$ LLRs — almost one seventh of the 512 LLRs required in the BLK algorithm. Since we do not need subtraction of the input LLR, the performance of the simplified algorithm is almost as good as that of the block algorithm (see Figures 3.5, 3.6 for details).

3.4 Simulation Results

In this section, we present Monte Carlo simulation results for the BLK and SBLK algorithms, and compare their performance with the previous IRCSDF algorithm (based on independence assumption) and also the ML bound. All simulations employ a random 64×64 binary image $f(m, n)$ with pixel values chosen from the alphabet $\{-1, +1\}$. The plots presented below show the bit error rate (BER) of the estimated binary input image, versus SNR. The SNR is defined as in [25]:

$$\text{SNR} = 10 \log_{10} \left(\frac{\text{Var}[\mathbf{f} * \mathbf{h}]}{\sigma_\omega^2} \right), \tag{3.14}$$

where $*$ denotes the 2D convolution in (1.2) and σ_ω^2 is the variance of the Gaussian rv $w(m, n)$ in (1.2). To compute the received image $r(m, n)$, we assume a boundary of -1 pixels around $f(m, n)$; the receiver uses this known boundary condition to simplify the trellis near edge pixels.

Both the BLK algorithm and the SBLK algorithm gives us significant improvement over our previous algorithm, especially for the 3×3 mask; the results are even better than our concatenated system with zigzag detector [7].

For the 2×2 mask, the BLK and SBLK algorithm (2-rows version) gives us almost 1 dB gain over the corresponding 2-rows version with independence assumption (see Figure 3.5). We applied a constant weight (over all iterations) to the extrinsic LLR (0.5 for BLK and 0.6 for SBLK), with ten iterations in each case. The weights were obtained after a semi-exhaustive search. This is in contrast to the original RC algorithm, which required an iteration-dependent weight schedule optimized based on EXIT charts.

A 3-rows and a 4-rows version of the original RC algorithm provided some improvement (at the cost of added trellis complexity); however, the BLK/SBLK algorithm still outperforms them. Performance of the 3-rows version is shown in Figure 3.5; the 4-rows version has a very similar performance and is not shown. In Figure 3.5, we also compare our results with a modified version of the original algorithm in [9]. The modification is that for the feedback pixels, extrinsic information is used from that provided by the previous detector (Column/Row) instead of from the current detector (Row/Column) as done in [9]. All of the 2-rows, 3-rows and 4-rows version provide some improvement due to this modification; the 2-rows version saw the most gain.

For the 3×3 averaging mask, Figure 3.6 shows that both the BLK and SBLK algorithms provide almost 1.2 dB gain over the original RC algorithm; their performance is only 0.3 dB away from the ML bound. A constant weight was was used (0.3 for both BLK and SBLK) with ten iterations in each case. Results for an easier 3×3 mask (called Channel B [4]) is also shown in Figure 3.6, with some performance improvement. The gains are relatively modest, since we are already quite close to the ML bound.

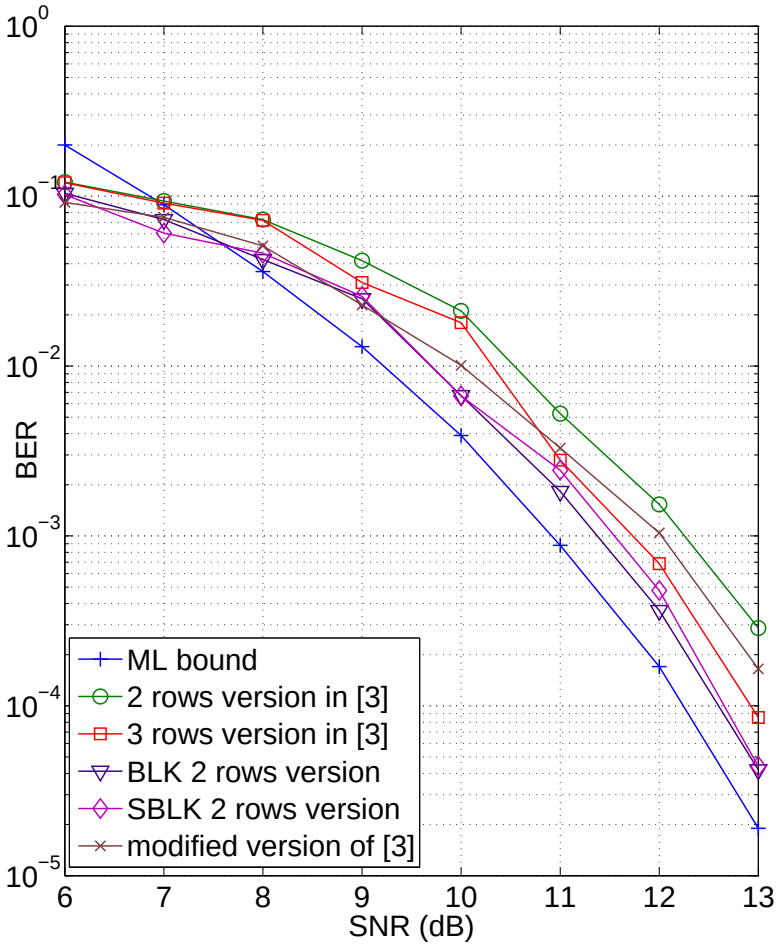


Figure 3.5: Monte Carlo simulations with 2×2 averaging mask for the new BLK and SBLK algorithms, the original RC algorithm with independence assumption in [9] and the ML bound.

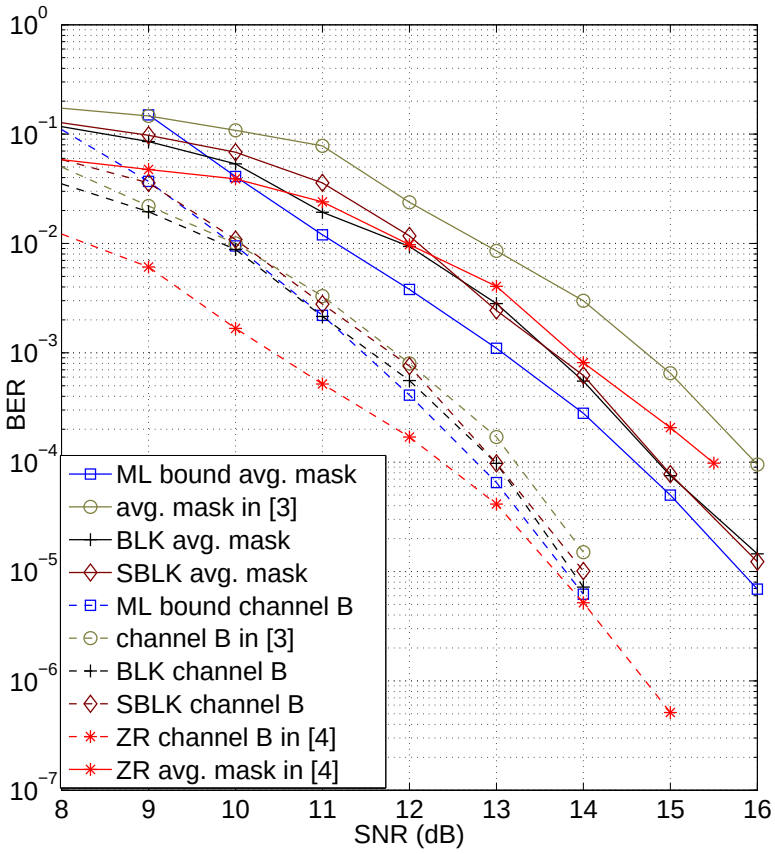


Figure 3.6: Monte Carlo simulations for the new BLK and SBLK algorithm, the original RC algorithm with independence assumption in [9], the concatenated system in [7] and the ML bound. Results for 3×3 averaging mask (solid line) and channel B from [4] (dashed line) are shown.

Chapter 4

Iterative Soft Decision Feedback Zigzag Algorithm Using Joint Extrinsic Information

In this chapter, we redesign the previous iterative soft decision feedback zigzag algorithm (ISDFZA) of Chen et. al. (IEEE JSAC, Feb. 2010), for equalization of two-dimensional intersymbol interference channels with additive white Gaussian noise. In the ISDFZA, source pixel estimates were assumed to be statistically independent; in the new algorithm we jointly estimate adjacent groups of source pixels. The new algorithm, called the block zigzag (BLKZ) algorithm, uses constant LLR weights across iterations, and provides more than 2 dB SNR gain over the ISDFZA on the 2×2 averaging mask channel; at low and medium SNRs it also outperforms the iterative row-column soft decision feedback algorithm (IRCSDFA) using joint extrinsic information (Chen et. al., CISS 2010). For the 3×3 averaging mask, the BLKZ algorithm gives more than 1 dB SNR gain over the independent ISDFZA at high SNRs. When the BLKZ algorithm is concatenated with the joint IRCSDFA, the concatenated system achieves the maximum-likelihood (ML) bound at high SNR with the 2×2 averaging mask; with the 3×3 averaging mask the joint concatenated system gains about 1 dB compared to the concatenated system with independence assumption, and

comes within 0.2 dB of the ML bound. This chapter also introduces a new method of LLR subtraction for the BLKZ algorithm based on the algorithm's two-dimensional extrinsic information flow (TEXIF) chart.

4.1 Introduction

This chapter also considers the detection of an $M \times N$ binary equiprobable two dimensional (2D) independent and identically distributed (i.i.d.) image $\mathbf{f}$ with elements $f(k,l) \in (-1,1)$ from received image $\mathbf{r}$ with elements (pixels)

$$r(m,n) = \sum_k \sum_l h(m-k, n-l) \cdot f(k,l) + w(m,n). \tag{4.1}$$

In (4.1), $h(k,l)$ are elements of a finite impulse response 2D mask $\mathbf{h}$, and the $w(m,n)$ are zero mean i.i.d. Gaussian random variables (r.v.s). The model in (4.1) applies, e.g., to 2D data storage systems, which suffer from 2D intersymbol interference (ISI) at high storage densities. Such systems are under active development for next generation optical disk storage (e.g., [12]), and holographic data storage (e.g., [16]).

In [7], we propose a novel iterative soft-decision feedback zigzag algorithm (ISDFZA). When the ISDFZA is concatenated with the iterative row-column soft-decision feedback algorithm (IRCSDFA) of [9], the resulting system achieves the high-SNR maximum likelihood (ML) bound for the 2×2 averaging mask channel (where $h(k,l) = 1/4$ for $k,l \in \{0,1\}$), and comes within 0.8 dB of the bound for the 3×3 averaging mask. Extensive comparisons in [7] show that the concatenated ISDFZA-IRCSDFA provides superior or competitive performance to other recently proposed 2D-ISI equalization algorithms.

However, the algorithms in [7] assume that pixel estimates passed as extrinsic information between constituent detectors are independent, when in fact they are spatially correlated.

Also, these algorithms employ log-likelihood ratio (LLR) weights that vary by iteration; optimizing these weight schedules requires EXIT charts to minimize design time. In this chapter we redesign the ISDFZA by jointly estimating blocks of adjacent pixels; we call the new algorithm the block zigzag (BLKZ) algorithm. The BLKZ employs a constant weight scheme which is easily optimized via simulation. These changes give almost 2 dB gain for the 2×2 averaging mask assumed in sections 4.2 through 4.4 of this chapter; the technique can also be generalized to larger masks.

When the BLKZ algorithm is serially concatenated with our recently proposed joint-statistics iterative row-column algorithm [5], the new concatenated system gives 0.2 dB gain at low and medium SNR compared to the previous concatenated system with independence assumption [7], and achieves the ML bound at high SNR with a much simpler LLR weight scheme. At high SNR on the 3×3 averaging mask channel, the new joint concatenated system gives 1 dB gain over the previous independent concatenated system, and comes within 0.2 dB of the ML bound.

This chapter is organized as follows. In section 4.2, the required modifications to the ISDFZA of [7] to estimate and exchange joint extrinsic information for the 2×2 averaging mask channel are presented. Section 4.3 discusses subtraction of input extrinsic information from the output LLR before it is passed on to the next detector, based on a two-dimensional extrinsic information flow chart analysis. Section 4.4 presents algorithm structures and a new soft consistency update method. In section 4.5 through section 4.6, we generalize the joint ISDFZA and the corresponding technical details to the 3×3 mask

case, and address the necessary changes. And in section 4.7, we give the details of the joint serial concatenated system containing joint IRCSDFA and joint ISDFZA. Simulation results are presented in section 4.8.

4.2 Joint Iterative Soft Decision Feedback Zigzag Algorithm for 2×2 Mask Case

4.2.1 Joint Extrinsic Information for the ISDFZA

Following the method used in the joint IRCSDFA [5], we develop the joint ISDFZA by defining blocks of adjacent, spatially correlated state and input pixels for which joint statistics will be used; for diagonal moves, these blocks are shown in Fig. 4.1. Similar joint blocks are defined for horizontal and vertical moves. After one BCJR step, we compute the joint probabilities of six pixels (3 for inputs, 3 for states) shown as red outlined blocks in Fig. 4.1. From the state-input block definitions in Fig. 2.3, we see that the two feedback pixels are not adjacent; hence, they are still assumed to be independent. Thus for the extrinsic information it is sufficient to provide just the joint three-pixel state and input blocks shown in Fig. 4.1 (b). In Fig.4.1 (b), the joint information blocks from detectors 1 and 3 in Fig.1.4 rotate 90 degrees counter-clockwise and then pass to the next detector; the joint blocks from detectors 2 and 4 rotate 90 degrees clockwise and pass to the next detector.

The modifications to the diagonal BCJR ([1]) computations to go from the independent to the joint ISDFZA are similar to those done in the joint IRCSDFA [5]. However, we change only the input probabilities to joint and keep the feedback probabilities independent for the reasons mentioned above. The independent ISDFZA has gamma probability:

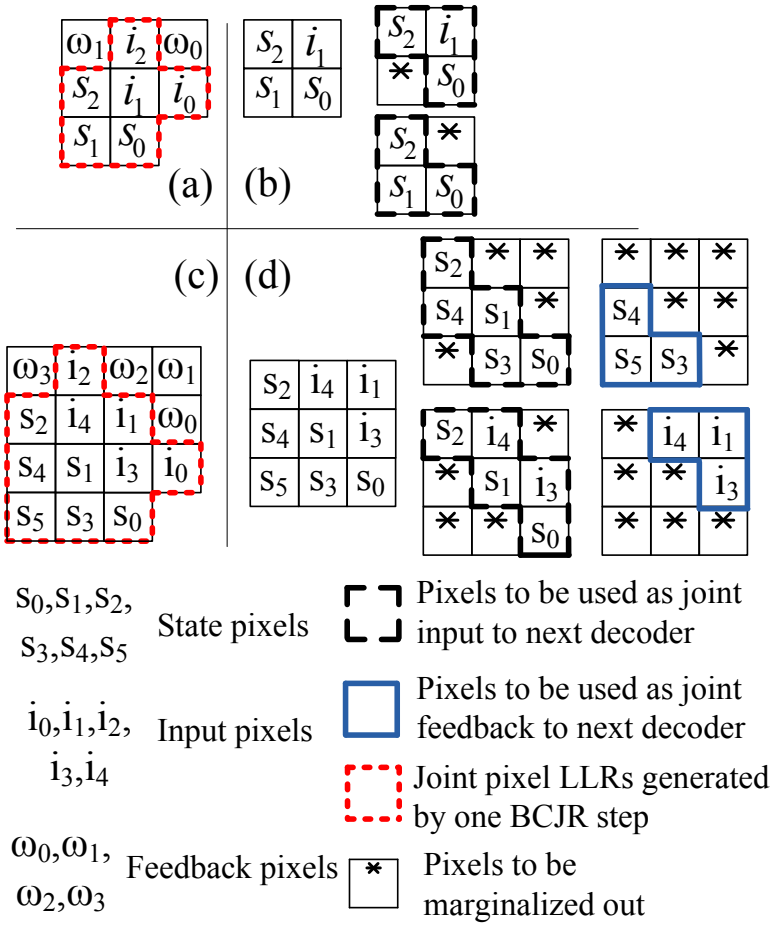


Figure 4.1: Block structure of the BLKZ algorithm for 2×2 mask ((a) and (b)) and 3×3 mask ((b) and (d)): (a), (c) state and input pixels definition for up-right diagonal move; (b), (d) joint block for the diagonal direction showing marginalization needed for the next detector. The other three movement types (i.e. horizontal, vertical, down-left) are similar.

$$\gamma_{\mathbf{i}}(\mathbf{y}_k, s', s) = p'(\mathbf{y}_k \mid \mathbf{u}_k = \mathbf{i}, S_k = s, S_{k-1} = s') \cdot P(\mathbf{u}_k \mid s, s') \cdot P(s \mid s') \cdot P(\tilde{\mathbf{u}}_k | \mathbf{u}_k = \mathbf{i}). \tag{4.2}$$

where

$$P(\tilde{\mathbf{u}}_k | \mathbf{u}_k = \mathbf{i}) \propto P(\mathbf{u}_k = \mathbf{i} | \tilde{\mathbf{u}}_k) = \prod_{l=1}^{2} P(u_{kl} = i_l | \tilde{\mathbf{u}}_k), \tag{4.3}$$

and

$$
\begin{aligned}
&p'(\mathbf{y}_k|\mathbf{u}_k = \mathbf{i}, S_{k-1} = s', S_k = s) \\
&= p(y_{k2}|i_0, i_1, i_2, s', s) \\
&\times \left[\sum_{\Omega_2} P(\Omega_2) p(y_{k1}|i_0, i_1, s', s, \mathrm{IP}_2(\Omega_2))\right. \\
&\left.\times \sum_{\Omega_1} P(\Omega_1) p(y_{k0}|i_0, s', s, \mathrm{IP}_1(\Omega_1, \Omega_2))\right].
\end{aligned}
\tag{4.4}
$$

In (4.4), soft decision feedback is incorporated by averaging the conditional (Gaussian) PDFs of received pixels y_{k1} and y_{k2} over the values of feedback pixels ω_0 and ω_1. The means of the conditional PDFs of y_{k0}, y_{k1}, and y_{k2} in (4.4) are inner products IP_1, IP_2, and IP_3 between the index-reversed mask and the relevant state and input bits for a given trellis branch; IP_2 and IP_3 also depend on feedback pixels ω_0 and ω_1. In the joint ISDFZA, we change (4.3) to

$$
P(\mathbf{u}_k = \mathbf{i}|\tilde{\mathbf{u}}_k) = P(i_0, i_1, i_2, i_3, i_4|\tilde{\mathbf{u}}_k), \tag{4.5}
$$

and leave feedback pixel probabilities $P(\omega_j)$ in (4.4) unchanged.

In the BCJR algorithm we compute $\alpha_k(s) = P(S_k = s, \mathbf{y}_1^{N_r})$, $\beta_k(s) = P(\mathbf{y}_{k+1}^{N_r} \mid S_k = s)$ and $\gamma_{\mathbf{i}}(\mathbf{y_k}, s, s') = P(\mathbf{U} = \mathbf{i}, S_k = s, \mathbf{y_k} \mid S_{k-1} = s')$ by a forward-backward procedure. We then compute $\lambda_k^{\mathbf{i}}(s) = \sum_{s'} \alpha_{k-1}(s')\beta_k(s)\gamma_{\mathbf{i}}(\mathbf{y_k}, s, s')$, which gives the unnormalized probability $P(\mathbf{U} = \mathbf{i})$. In the independent version of the ISDFZA, we compute the binary valued LLR as

$$L(k) = \log\left(\frac{\sum_{s,i_1,i_2} \lambda_k^{i_0=+1}(s)}{\sum_{s,i_1,i_2} \lambda_k^{i_0=-1}(s)}\right). \tag{4.6}$$

The BLKZ algorithm computes a 64-valued LLR at each pixel for the 2×2 mask, instead of the single LLR at each pixel computed by the ISDFZA with independence assumption. In the BLKZ algorithm, we compute the joint probability of the block of six pixels that constitute the state and input as shown in Fig. 4.1 (a). For computing the 64-valued LLRs $L_k(\mathbf{i}, s)$, we do not sum the $\lambda_k^i(s) = \sum_{s'} P(\mathbf{U} = \mathbf{i}, S_k = s, S_{k-1} = s', \mathbf{y_k})$ probabilities over the state s and the inputs i_1 and i_2 as in (4.6). Thus, the 64-valued LLRs are

$$L_k(\mathbf{i}, s) = \log\left(\frac{\lambda_k^{\mathbf{i}}(S)}{\lambda_k^{\mathbf{i}=\mathbf{i_0}}(S = S_0)}\right), \tag{4.7}$$

where $\mathbf{i} = (i_0, i_1, i_2)$, $\mathbf{i_0} = (-1, -1, -1)$, $S = (s_0, s_1, s_2)$, and $S_0 = (-1, -1, -1)$.

4.3 Subtraction Analysis Based on Two-Dimensional Extrinsic Information Flow

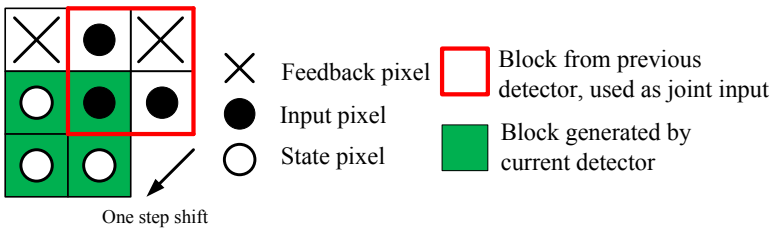


Figure 4.2: 'one step shift' in joint ISDFZA BLKZ

In turbo detection or decoding, the current detector's previously input extrinsic information is subtracted from the current LLR estimate, and the result is passed as extrinsic

information to the next detector; subtraction ensures that only new information developed by the current detector is passed to the next detector. In our initial approach to subtraction in the BLKZ algorithm, we passed 16 LLRs from the four-pixel block shown in Fig. 4.1 (b) as extrinsic information to the next detector, and subtracted them from the 16 LLRs for the same four pixels developed at the end of the next detector's current iteration. This approach is equivalent to factoring out the extrinsic input joint probabilities from the final likelihood ratios, and is consistent with the theory developed for LLR subtraction in standard turbo coding [21]. However, the resulting algorithm had poor performance in bit error rate (BER) vs. SNR simulations. So we investigated new techniques for doing the subtraction. A similar issue arose when developing the joint IRCSDFA in [5].

Fig. 4.2 shows that the joint information block generated by the current detector has a one step shift in the diagonal direction compared to the joint extrinsic information block coming into this detector. This shift does not occur in the independent version of the algorithm, wherein the pixel being computed uses independent extrinsic information from exactly the same pixel location in the previous detector.

In the BLKZ algorithm, the one step shift ensures that the input and output extrinsic blocks share only one common pixel, so that, as shown in Fig. 4.3, in one complete detection iteration (from detector 1 to detector 4) the four output (input) blocks form a loop. Thus for every detector, the output extrinsic information is much less correlated with the input information compared to the independent case, since their locations are no longer the same.

However, the extrinsic information coming into detector 1 from detector 4 in the previous iteration contains some of the information from the previous iteration's detector 1,

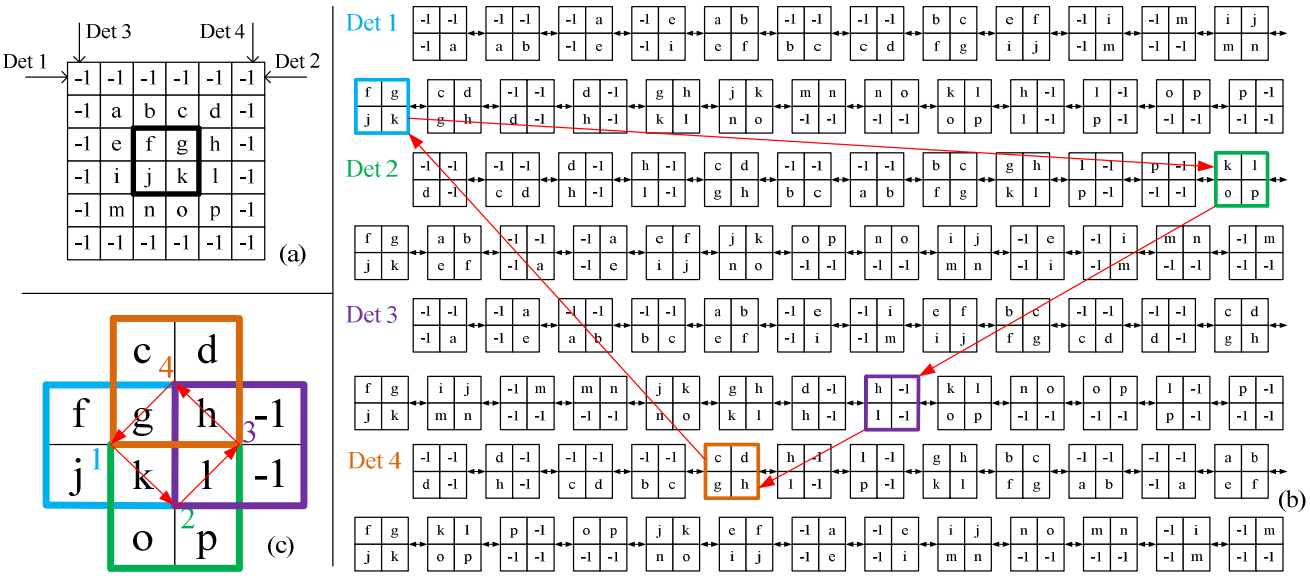


Figure 4.3: Two-dimensional extrinsic information flow (TEXIF) chart and loops in the BLKZ algorithm with 2×2 averaging mask. (a) 4×4 source image with boundary; (b) TEXIF chart of four inner zigzag detectors, showing an example loop; (c) example loop for the BLKZ algorithm (on block [f g; j k]) in (a), showing overlap between the blocks. (The index '1,2,3,4' and the four different colors represent the four inner zigzag detectors.)

since they form a circle. To avoid that effect, we propose the following subtraction procedure. In the log domain, detector 1 does not subtract the incoming extrinsic information from detector 4, as is done in the independent version of the algorithm; instead, it subtracts the extrinsic information from detector 1 in the previous iteration. This procedure is also done in detectors 2, 3, and 4. Since this subtraction procedure is based on the loop structure in the BLKZ algorithm's TEXIF chart, we call it "circular subtraction." Since the feedback pixels in the BLKZ algorithm still use the independence assumption, we do not consider circular subtraction of their extrinsic information. Simulation results in section 4.8 show the effectiveness of circular subtraction.

Next, the details of the LLR subtraction in the BLKZ algorithm are described. For the joint statistics case, we employ generalized LLRs as in [5]. Referring to the BLKZ joint block structure in Fig. 4.1 (a), in the BLKZ algorithm, after one BCJR step, we compute

the $2^6 = 64$ joint probabilities of six pixels. This gives 64 LLRs $L_k(\mathbf{i}, s)$ after converting the probabilities to the log domain, as in (4.7). Among these six pixels, three of them are state pixels, while the other three are input pixels. Thus the 64 LLRs can be divided into eight groups of eight. The eight LLRs in each group share the same input pixels, and differ only in the state pixels. The eight *a priori* input probabilities are converted into eight LLRs $L_k^a(\mathbf{i})$ called 'aLLRs' to discriminate them from the output LLRs. The circular subtraction for the BLKZ algorithm produces 64 extrinsic information LLRs $\tilde{L}_k(\mathbf{i}, s)$ to be passed to the next detector as follows:

$$\tilde{L}_k(\mathbf{i}, s) = L_k(\mathbf{i}, s) - L_k^a(\mathbf{i}), \qquad s \in \{0, 1, \ldots, 8\}. \tag{4.8}$$

Thus, the eight LLRs in one group with fixed input $\mathbf{i}$ (out of the eight groups) subtract exactly one 'aLLR' for input pixels having the same value of $\mathbf{i}$.

The reason the subtraction is done by starting with the 64 LLRs $L_k(\mathbf{i}, s)$ is that, to subtract out the input part in the log domain, the starting group of pixels must contain all three input pixels. Referring again to the BLKZ structure in Fig. 4.1 (a), the six-pixel group outlined in red contains all three input pixels. After the subtraction is done from the 64 LLRs corresponding to these groups, the resulting extrinsic LLRs $\tilde{L}_k(\mathbf{i}, s)$ are converted to joint probabilities and marginalized to the smaller joint blocks shown in Fig. 4.1 (b). The marginalized blocks are then converted back to LLRs and passed as extrinsic information to the next detector.

4.4 Algorithm Structures

In this section we develop the structure of the BLKZ algorithm. This development is presented as an evolutionary process in order to illustrate how different parts of the algorithm affect the performance.

The initial version of the algorithm has two loops. The loop supplying joint input information to the next detector is the "input path", and the other one is the "feedback path" which offers independent feedback information to the next detector. All the BLKZ algorithms in this paper contain these two paths. In this first algorithm (shown in Fig. 4.5), there is no LLR subtraction and no soft consistency update. This algorithm's simulation performance (shown as the curve marked "BLKZ type 1" in Fig. 4.13) is about the same as the ISDFZA with the independence assumption but without the soft consistency update. Since no subtraction is performed here, the computational complexity is reduced.

In the second algorithm we add a feedback soft consistency update to the feedback path, and we call this algorithm "joint ISDFZA with partial soft consistency." The feedback soft consistency update is performed in the same manner as that in [7] and assumes independence of all pixels when they are used as feedback pixels; the independence assumption for the feedback pixels is based on their spatial isolation from one another in Fig. 4.1. The simulation performance of this algorithm is seen in the curve marked "BLKZ type 2" in Fig. 4.13. As in the independent ISDFZA [7], the soft consistency update brings almost 3 dB SNR gain. This algorithm still lacks any LLR subtraction and has almost the same performance as the independent ISDFZA with both soft consistency and subtraction. The greatest advantage of this and the following algorithms is that a constant weight scheme is

applied, giving a great time savings in optimizing the weight scheme.

The third BLKZ algorithm has circular subtraction (described in section 4.3) in the input path and a feedback soft consistency update in the feedback path, and also uses a constant weight scheme. This algorithm's simulation performance algorithm is seen in the curve marked "BLKZ type 3" in Fig. 4.13. Its performance is about 0.2 dB better than the "BLKZ type 2", which shows that the circular subtraction is effective.

4.4.1 Interface Soft Consistency Update

In previous work [7], we showed that the soft consistency update in the independent zigzag algorithm gives about 3 dB SNR gain. In the second algorithm above, the "partial soft consistency" is only applied to the feedback path since we still use the independence assumption there. For the final version of BLKZ, we wish to make the information after the soft consistency update available to the input path, which uses joint statistics. However we cannot simply copy the soft consistency update of [7] directly, as that update is based on the independence assumption.

As shown in Fig. 4.4, we multiply the joint probability of the three input pixels from the input path with their three independent probabilities obtained from the feedback path, and after this update, the new joint probability is used as the joint input for the next detector. This procedure is indicated with the box marked "interface soft consistency" in the final BLKZ algorithm diagram shown in Fig. 4.6. The simulation performance of this algorithm is seen in the curve marked "BLKZ type 4" in Fig. 4.13. Despite this method's obvious sub-optimality in updating a joint probability with a product of marginals, it is evident from Fig. 4.12 that the proposed interface soft consistency update gives more than 1 dB

SNR gain.

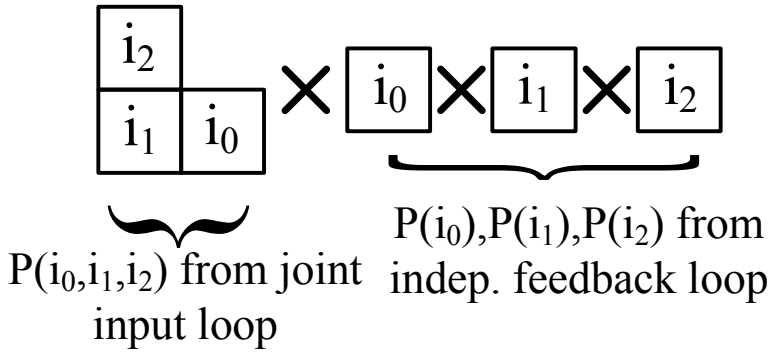


Figure 4.4: Interface soft consistency update.

4.5 Joint Iterative Soft Decision Feedback Zigzag Algorithm for 3×3 Mask Case

Following the method used in the joint ISDFZA [6] 2×2 mask case, we develop the joint ISDFZA for the 3×3 mask; for diagonal moves, these blocks are also shown in (c) and (d) of Fig. 4.1. Similar joint blocks are defined for horizontal and vertical moves. After one BCJR step, we compute the joint probabilities of eleven pixels (5 for inputs, 6 for states) shown as red outlined blocks in Fig. 4.1(c).

For the 3×3 mask, feedback pixels ω_0, ω_1 and ω_2 in Fig. 4.1(c) are adjacent, and hence are estimated jointly and passed to the next detector as joint (8-ary) LLRs. Fig. 4.1(d) shows the marginalization for the input pixels and feedback pixels for the 3×3 mask.

As in the 2×2 case, in (d) of Fig. 4.1, the joint information blocks from detectors 1 and 3 in Fig. 1.4 rotate 90 degrees counter-clockwise and then pass to the next detector; the joint blocks from detectors 2 and 4 rotate 90 degrees clockwise and pass to the next detector.

For the 2×2 case, the three input pixels for diagonal or anti-diagonal moves form a '$\mathbf{L}$'. As shown in Fig. 4.7, for the 2×2 mask there are four '$\mathbf{L}$' orientations in the four zigzag detectors; the 3×3 case is similar. Since every two neighboring detectors have different '$\mathbf{L}$' orientations, the four-pixel joint blocks shown in Fig. 4.1 (b) and (d) are marginalized into two '$\mathbf{L}$'s before passing them as extrinsic information to the next detector.

For the the joint ISDFZA 3×3 case, we now summarize the changes made to the modified BCJR algorithm. For the 3×3 mask, the *a priori* probability (4.3) changes to a joint version.

$$P(\mathbf{u}_k = \mathbf{i}|\tilde{\mathbf{u}}_k) = P(i_0, i_1, i_2, i_3, i_4|\tilde{\mathbf{u}}_k), \tag{4.9}$$

and the conditional channel probability changes to

$$\begin{aligned}
&p'(\mathbf{y_k} \mid \mathbf{u}_k = \mathbf{i}, S_k = s, S_{k-1} = s') = \\
&\begin{cases}
\displaystyle\sum_{\omega_0,\omega_1,\omega_2} P(\omega_0,\omega_1,\omega_2) P(y_{k0} \mid i_1, i_2, s, s', \mathrm{IP}_1(\omega_2)) \\
\times P(y_{k1} \mid i_0, i_1, i_2, s, s', \mathrm{IP}_2(\omega_0,\omega_1,\omega_2)), \\
\text{or} \\
P(y_{k0} \mid i_1, i_3, i_4, s, s') \times \Big[\displaystyle\sum_{\omega_0,\omega_1,\omega_2} P(\omega_0,\omega_1,\omega_2) \\
P(y_{k1} \mid i_0, i_1, i_2, i_3, i_4, s, s', \mathrm{IP}_2(\omega_0,\omega_1,\omega_2)) \Big] \times \\
\Big[\displaystyle\sum_{\omega_2,\omega_3} P(\omega_2,\omega_3) P(y_{k2} \mid i_1, i_2, i_3, i_4, s, s', \mathrm{IP}_3(\omega_2,\omega_3)) \Big].
\end{cases}
\end{aligned} \tag{4.10}$$

In (4.10), the first option shows the computation for horizontal and vertical moves (with input/state block defined in [7]), and the second for diagonal moves. In the second option the joint expectation $E_{\omega_0,\omega_1,\omega_2,\omega_3}[P(y_{k1} \mid i_0,\ldots,i_4,s,s',\mathrm{IP}_2)P(y_{k2} \mid i_1,\ldots,i_4,s,s',\mathrm{IP}_3)]$ is approximated as the product of joint expectations $E_{\omega_0,\omega_1,\omega_2}[P(y_{k1} \mid i_0,\ldots,i_4,s,s',\mathrm{IP}_2)P(y_{k2} \mid i_1,\ldots,i_4,s,s',\mathrm{IP}_3)]$; experiments show that this approximation gives better results than assuming that $P(\omega_0,\omega_1,\omega_2,\omega_3) \approx P(\omega_0,\omega_1,\omega_2)P(\omega_3)$.

The general definition of the joint LLR for joint ISDFZA 3×3 case, is similar to equation (4.7) shown in section 4.2. For the 3×3 joint ISDFZA a 2048-valued LLR is computed at each pixel, with $\mathbf{i} = (i_0, i_1, i_2, i_3, i_4)$, $\mathbf{i_0} = (-1,-1,-1,-1,-1)$, $s = (s_0, s_1, s_2, s_3, s_4, s_5)$, and $s_0 = (-1,-1,-1,-1,-1,-1)$.

4.6 Subtraction Analysis Based on Two-Dimensional Extrinsic Information Flow For 3×3 Mask Case

For the 3×3 case, we still have four similar algorithms structures as the 2×2 case. All the algorithms also contain two paths. One difference with the 2×2 case is that the "input path" not only supplies joint input information to the next detector, but also supplies joint feedback information in the 3×3 case. The "feedback path" still offers independent feedback information to the next detector for the implementation of the 'interface soft consistency update', which also has some differences with the 2×2 case, as described in subsection 4.6.1.

In first and second version of the algorithm, there are not many differences with the 2×2 case, except the "input path" also supplies joint feedback information.

In the third version with subtraction on the "input path", besides the 'circular subtraction' introduced in section 4.3 based on the TEXIF chart analysis, we now also introduce another method to do the subtraction, called the 'grouping method'.

In the joint ISDFZA, incoming extrinsic information is subtracted from the joint LLRs *before* their marginalization into extrinsic information to be passed to the next detector shown in Fig. 4.1(b) and (d). Before marginalization, the joint λ probability can be easily factored as

$$\begin{aligned}\lambda_k^{\mathbf{i}}(s) &= \sum_{s'} \alpha_{k-1}(s')\beta_k(s)\gamma_{\mathbf{i}}(\mathbf{y}_k, s, s') \\ &= \sum_{s'} \alpha_{k-1}(s')\beta_k(s)p(\mathbf{y}_k|\mathbf{u}_k, s, s')P(\mathbf{u}_k|s, s')P(s|s') \\ &\times P(\tilde{\mathbf{u}}_k|\mathbf{u}_k) \\ &= P(\mathbf{u}_k, s, y_1^{N_r})P(\tilde{\mathbf{u}}_k|\mathbf{u}_k).\end{aligned} \tag{4.11}$$

The corresponding joint LLR $L_k(\mathbf{i}, s)$ splits into an extrinsic LLR $L_k^e(\cdot)$ that can be passed to the next detector, and a term $L_k^{e_{in}}(\cdot)$ that depends on the input extrinsic information $\tilde{\mathbf{u}}_k$ from the previous detector:

$$\begin{aligned}L_k(\mathbf{i}, s) &= \log\left[\frac{\lambda_k^{\mathbf{i}}(s)}{\lambda_k^{\mathbf{0}}(\mathbf{0})}\right] = \log\left[\frac{P(\mathbf{u}_k, s, y_1^{N_r})}{P(\mathbf{0}, \mathbf{0}, y_1^{N_r})} \cdot \frac{P(\tilde{\mathbf{u}}_k|\mathbf{u}_k)}{P(\tilde{\mathbf{u}}_k|\mathbf{0})}\right] \\ &= L_k^e(\mathbf{i}, s) + L_k^{e_{in}}(\mathbf{i}, \tilde{\mathbf{u}}_k).\end{aligned} \tag{4.12}$$

The subtraction of the LLR $L_k^{e_{in}}(\mathbf{i}, \tilde{\mathbf{u}}_k)$ from the joint LLR $L_k(\mathbf{i}, s)$ is done by a group-

ing procedure described as follows for the 3×3 mask case; the 2×2 case is similar.

At the beginning of BCJR process, the incoming 32 joint extrinsic LLRs $L_k^{e_{in}}(\mathbf{i}, \tilde{\mathbf{u}}_k)$ for the five pixels as shown in Fig. 4.1(d) are saved in memory; these LLRs include the interface soft consistency information from the previous detector. Referring to the BLKZ joint block structure in Fig. 4.1 (c), after one BCJR step, we compute the $2^{11} = 2048$ joint probabilities of eleven pixels, and convert them into 2048 LLRs $L_k(\mathbf{i}, s)$. Six of these eleven pixels are state pixels, while the other five are input pixels. Thus the 2048 LLRs can be divided into 32 groups of 64. The 64 LLRs in each group share the same input pixels, but have different state pixels. The 64 LLRs in each group with fixed input $\mathbf{i}$ subtract exactly one $L_k^{e_{in}}(\mathbf{i}, \tilde{\mathbf{u}}_k)$ for input pixels having the same value of $\mathbf{i}$:

$$L_k^e(\mathbf{i}, s) = L_k(\mathbf{i}, s) - L_k^{e_{in}}(\mathbf{i}, \tilde{\mathbf{u}}_k), \quad s \in \{0, 1, \ldots, 64\}. \tag{4.13}$$

After the subtraction is done, the resulting extrinsic LLRs $L_k^e(\mathbf{i}, s)$ are converted to joint probabilities and marginalized to the smaller joint blocks shown in Fig. 4.1 (d). The marginalized blocks are converted to LLRs and passed as extrinsic information to the next detector.

Experiments show that the sequential LLR subtraction described previously in equations (4.11)-(4.13) and circular subtraction have almost identical BER vs. SNR performance. One advantage of circular subtraction is that the first iteration through all four detectors does not require a subtraction, whereas sequential subtraction requires three subtractions on the first iteration. All simulations presented in section 4.8 use sequential subtraction, with the exception of Fig. 4.13, where circular subtraction is used in the 'BLKZ

type 3' and 'type 4' algorithms.

Finally we also note that for the joint ISDFZA it does not make sense to create a simplified algorithm similar to the SBLK described in Chapter 3. As the joint IRCSDFA does not need extrinsic information LLR subtraction, as its TEXIF chart is loop free. However, the joint ISDFZA needs subtraction on all 2048 LLRs (for the 3×3 case) after they are converted from probabilities. After the subtraction, all the LLRs are converted back to probabilities and marginalized so that they can be passed as extrinsic information. This probability to LLR conversion is just for the purpose of subtraction. As the result of this conversion, a simplified BLKZ algorithm would bring no savings in either computational complexity or storage; instead, it may increase the complexity, due to the additional marginalization required to produce the simplified blocks.

The fourth and final BLKZ algorithm for 3×3 mask case is shown in Fig. 4.9; one more difference here with the 2×2 case is that the interface soft consistency update is more complicated; it is introduced next.

4.6.1 Interface Soft Consistency Update for 3×3 Mask Case

Generalized from the 2×2 case, the 'interface soft consistency update' is shown in Fig. 4.8 for the 3×3 case. The joint probability of the five input pixels from the input path in Fig. 4.9 is multiplied with their five independent probabilities obtained from the feedback path in Fig. 4.9, and after this update, the new joint probability is used as the joint input for the next detector. Similar processing is done for the joint feedback pixels in the 3×3 case, but not in the 2×2 case where the feedback pixels are considered to be independent due to their spatial separation.

The simulation performance of the BLKZ with ISC is seen in the curve marked "BLKZ type 4" in Fig. 4.13. We show the evolution process in Fig. 4.13 only for the 2×2 case; all simulation results presented for the 3×3 case are from the BLKZ type 4 algorithm, with sequent subtraction.

4.7 Concatenated System with Joint Statistics

In this section we consider a serial concatenation of the joint ISDFZA and the joint IRCSDFA. Each of these two detection algorithms performs several inner iterations before passing joint extrinsic information to the other; after several outer iterations, a hard decision is made on the final output. A similar structure is employed for the concatenated system with independence assumption in [7]; however, the joint system has a more complicated interface between algorithms.

Fig. 4.10 shows a block diagram of the joint concatenated system; in this figure, m and n are the number of inner iterations for the IRCSDFA and ISDFZA respectively. The joint concatenated system needs more inner iterations than the independent system. The 2×2 case needs at least four inner iterations for both the joint ISDFZA and IRCSDFA, and the 3×3 case needs at least three. The dashed lines in the figure indicate the extrinsic information passed on the two interfaces. The constant LLR weights used by these equalization algorithms are as follows: joint IRCSDFA 0.4 (BLK), 0.3 (SBLK); joint ISDFZA 0.01; interface from IRCSDFA to ISDFZA 0.1; interface from ISDFZA to IRCSDFA 0.01.

4.7.1 Subtraction on the Interface of the Concatenated System

For the IRCSDFA to ISDFZA joint interface, if use circular subtraction, for the very first inner ISDFZA iteration, all the four scanners do not need subtraction. (Result not shown here). And if using the second subtraction method, after the IRCSDFA to ISDFZA interface, the first zigzag detector in the very first inner Zigzag iteration need the subtraction, which will be done in a similar manner as we described in the 'new zigzag subtraction grouping method'. The only difference is that, here we do not need the independent pixels' probabilities containing 'soft consistency information', which only exist in the joint zigzag part.

Only the ISDFZA to IRCSDFA interface has subtraction. The IRCSDFA to ISDFZA interface does not have subtraction. It is still an experimental result, we may need to explore the reason for this phenomenon Fig. 4.10. A tentative explanation may be in the diagram for the whole concatenated system (Fig. 4.10), the zigzag system has subtraction after every detector in the inner iterations, however the joint IRCSDFA system has no subtraction in its inner iteration, which means the zigzag could somehow subtract the redundant part in the extrinsic information from the row-column in its inner iteration, even without the subtraction on the IRCSDFA to ISDFZA interface, but since the IRCSDFA has no inner subtraction, it needs the subtraction on the interface.

4.7.2 Interface Soft Consistency Update on the IRCSDFA to ISDFZA Interface for the Concatenated System

Fig. 4.11 parts (a) and (b) show the extrinsic information blocks passed at the ISDFZA-IRCSDFA and IRCSDFA-ISDFZA interfaces for the 3×3 mask case. As shown in Fig. 4.10, at the ISDFZA-IRCSDFA interface incoming ISDFZA joint extrinsic information is subtracted from the row detector output in the first IRCSDFA inner iteration. Similarly, at the IRCSDFA-ISDFZA interface incoming ISDFZA information is subtracted from the second zig-zag detector output in the first ISDFZA inner iteration. (Subtracting from the first zig-zag detector is also effective, but experiments showed that subtracting from the second detector gives slightly better performance.) The interface subtractions use grouping as described in section 4.6.

Fig. 4.11 parts (c) and (d) show the interface soft consistency update applied at the ISDFZA-IRCSDFA interface. After obtaining the the joint block from the input loop of the last ISDFZA zigzag detector, the independent probabilities of the three input pixels and six feedback pixels for the first IRCSDFA row detector are obtained from the feedback loop of the last zigzag detector. The joint block is then marginalized as either a joint input (part (c)) or joint feedback (part (d)) block and multiplied by the input pixel probabilities (part (c)) or feedback probabilities (part (d)) to obtain the joint input and feedback blocks passed to the IRCSDFA.

Table 4.1: Computational complexity comparison of different equalization algorithms for the 3×3 averaging mask.

Algorithm	add / subtract	multiply / divide	exp/log
Joint IRCSDFA BLK (per pixel per iteration)	1.151*e*+6	1.416*e*+6	1.987*e*+5
Joint IRCSDFA SBLK (per pixel per iteration)	1.619*e*+5	2.039*e*+5	3.179*e*+4
Joint ISDFZA (per pixel per iteration)	1.695*e*+7	4.439*e*+6	6.358*e*+5
Joint Concatenated System (per pixel per outer iteration)	5.131*e*+7	1.757*e*+7	2.505*e*+6
Independent IRCSDFA from [9] (3 rows version) (per pixel per iteration)	1.935*e*+3	3.398*e*+3	3.91*e*+2
Independent Concatenated System from [7] (per pixel per outer iteration)	6.604*e*+5	1.052*e*+6	8.821*e*+4

4.8 Simulation Results

In this section, we present Monte Carlo simulation results for the joint equalization algorithms, and compare their performance with previous algorithms based on the independence assumption and with the ML bound. All simulations employ a random 64×64 binary image $f(m, n)$ with pixel values chosen from the alphabet $\{-1, +1\}$. The plots presented below show the bit error rate (BER) of the estimated binary input image, versus SNR. The SNR is defined as in [25]:

$$\text{SNR} = 10 \log_{10} \left(\frac{\text{Var}[\mathbf{f} * \mathbf{h}]}{\sigma_\omega^2} \right), \tag{4.14}$$

where $*$ denotes the 2D convolution in (1.2) and σ_ω^2 is the variance of the Gaussian r.v. $w(m, n)$ in (1.2). To compute the received image $r(m, n)$, we assume a boundary of -1 pixels around $f(m, n)$; the receiver uses this known boundary condition to simplify the trellis near edge pixels.

4.8.1 2×2 masks

Simulation results for the 2×2 averaging mask are shown in Fig. 4.14. At high SNR, the BLK and SBLK joint IRCSDFAs with two-row state-input block (and four-state trellis)

shown in Fig. 3.2 (a) have almost identical performance; they provide about 0.3 dB gain over the performance curve for the three-row independence-assumption IRCSDFA (with eight-state trellis) shown in Fig. 4.14. (The four-row independent IRCSDFA has the same high-SNR performance as the three-row.) Although we implemented three- and four-row versions of the BLK/SBLK algorithm, they did not provide the performance improvements over the two-row joint version that we observed with the independence assumption versions in [7]. This is because the joint block becomes rectangular, and it is therefore not easy to obtain the required joint input and feedback probabilities in such a way that they can be easily exchanged between row and column detectors. The versions implemented therefore used some partial independence assumptions (e.g, the final joint probability is the product of a joint probability and an independent probability) that apparently prevent further performance improvements by adding more rows to the state-input block.

At BER 10^{-4}, the joint ISDFZA "type 4" algorithm performs about 1.6 dB better than the independent ISDFZA, and even outperforms the joint IRCSDFA by about 0.2 dB. However, at BER 10^{-5} the joint ISDFZA loses about 0.4 dB to the joint IRCSDFA.

At 14 dB SNR, the 2×2 joint concatenated system performs as well as the ML bound. The previous independence-assumption concatenated system also hits the ML bound at high SNR. However, the new algorithm uses a much simpler constant-weight LLR weighting scheme, which makes it possible to quickly optimize the weights by repeated simulations, rather than using the more complicated EXIT-chart optimization methods described in [7].

The joint concatenated system actually performs about 0.3 dB below the ML bound and independent concatenated system at BER 2×10^{-5}. We observed a similar (but smaller)

effect in the 3×3 simulation results shown in Fig. 4.15. We believe this occurs because the ML bound is not tight at low and medium SNRs, and because the joint algorithms may be better at exploiting known boundary conditions at the image edges. To verify that 64×64 is a large enough image size, we tested 128×128 input images at SNR=14dB with the 3×3 averaging mask and obtained a BER of 2.91×10^{-4} for the joint concatenated system, which is only slightly higher than the result shown in Fig.4.15; thus, we argue that 64×64 images are large enough to avoid undue influence of boundary effects.

4.8.2 3×3 masks

Simulation results for the 3×3 averaging mask are shown in Fig. 4.15. At high SNR, the BLK and SBLK joint IRCSDFAs with three-row state-input block shown in Fig. 3.2 (g) have almost identical performance; they provide about 1.1 dB gain over the independence-assumption IRCSDFA at BER 10^{-4}, and about 0.6 dB gain over the independent concatenated algorithm. At BER 10^{-3}, the joint ISDFZA performs about 1.1 dB better than the independent ISDFZA, but loses about 1.2 dB to the BLK and SBLK algorithms.

The 3×3 joint concatenated system outperforms its independent counterpart by 1 dB at BER 10^{-4}, outperforms the joint IRCSDFA by about 0.3 dB at BER 2×10^{-5}, and performs at slightly less than 0.2 dB above the ML bound at BER 10^{-5}. To our knowledge, this is the best performance for the 3×3 averaging mask published to date.

Table 4.1 compares computational complexity between the joint equalization algorithms and the previous independent ones. All comparisons are done on 64×64 source images. The simplified joint IRCSDFA (SBLK) has only about one-third the computational complexity of the independent concatenated system, yet has about a 0.6 dB performance

gain in Fig. 4.15; this shows the significant advantages of using joint estimation. When comparing the complexity of the joint concatenated system to that of the simplified joint IRCSDFA or the independent concatenated system, it is clear that achieving the last few tenths of a dB improvement requires a large increment of computational complexity.

Fig. 4.16 shows simulation results for the 3×3 "channel B" mask from [4]; this sampled Gaussian mask is easier to equalize than the averaging mask. All LLR weights in the channel B test are set equal to those for the averaging mask, in order to verify that one set of weights can be used for different masks. At SNR 14 dB, the independent and joint concatenated systems achieve ML performance, and the joint IRCSDFA comes within 0.1 dB of ML. However, at BER 10^{-4} the joint concatenated system outperforms its independent counterpart by 0.9 dB and outperforms the joint IRCSDFA by 1.5 dB.

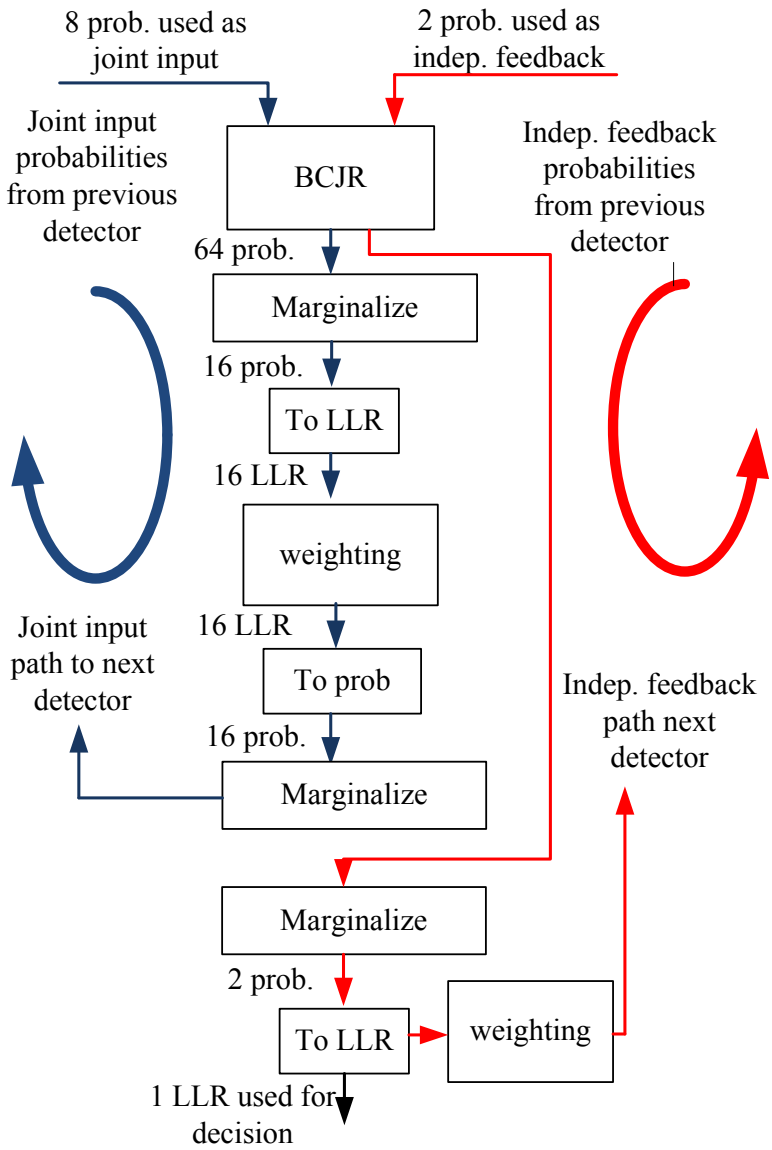


Figure 4.5: The first joint ISDFZA algorithm structure without ‘LLR subtraction’ or ‘partial soft consistency update’

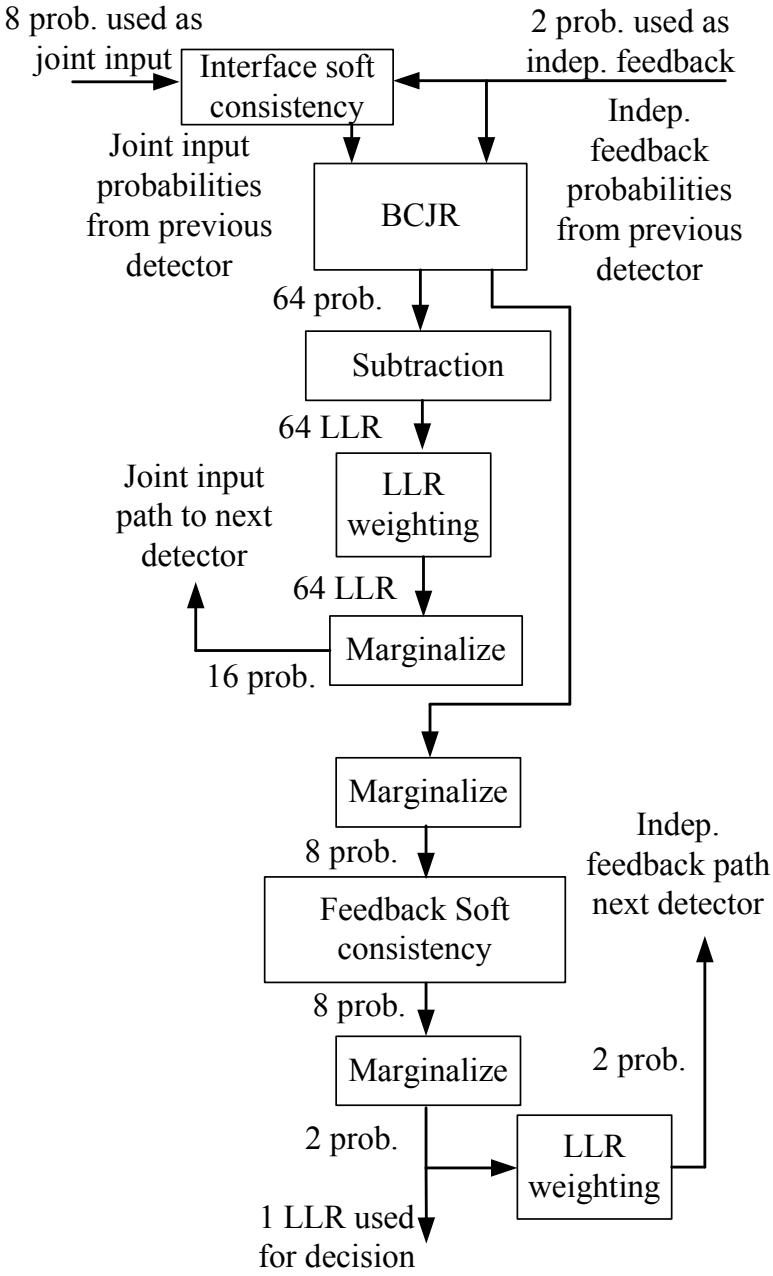


Figure 4.6: The final version of the BLKZ algorithm (2×2 case) with LLR subtraction and interface soft consistency updates for both input and feedback.

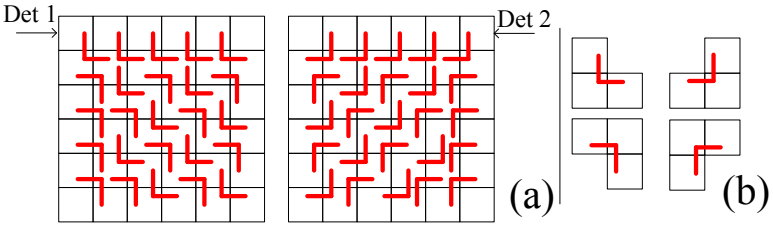


Figure 4.7: (a) The joint input *a priori* probabilities needed by the first and second zigzag detector (b) The four directions of '**L**' as joint input groups

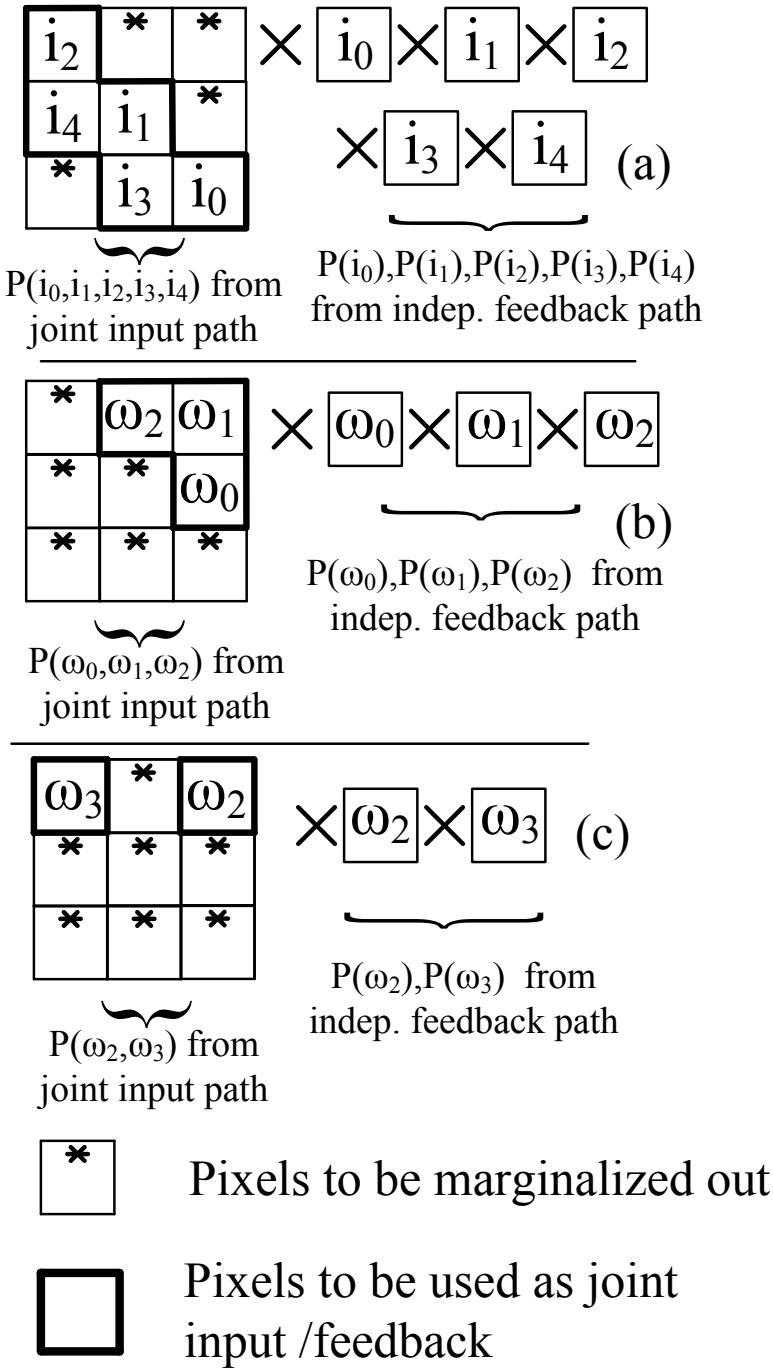


Figure 4.8: Interface soft consistency update for 3×3 case: (a) 'interface soft consistency' update for the input pixels; (b) 'interface soft consistency' update for the feedback pixels for the second inner product; (c)'interface soft consistency' update for the feedback pixels for the third inner product.

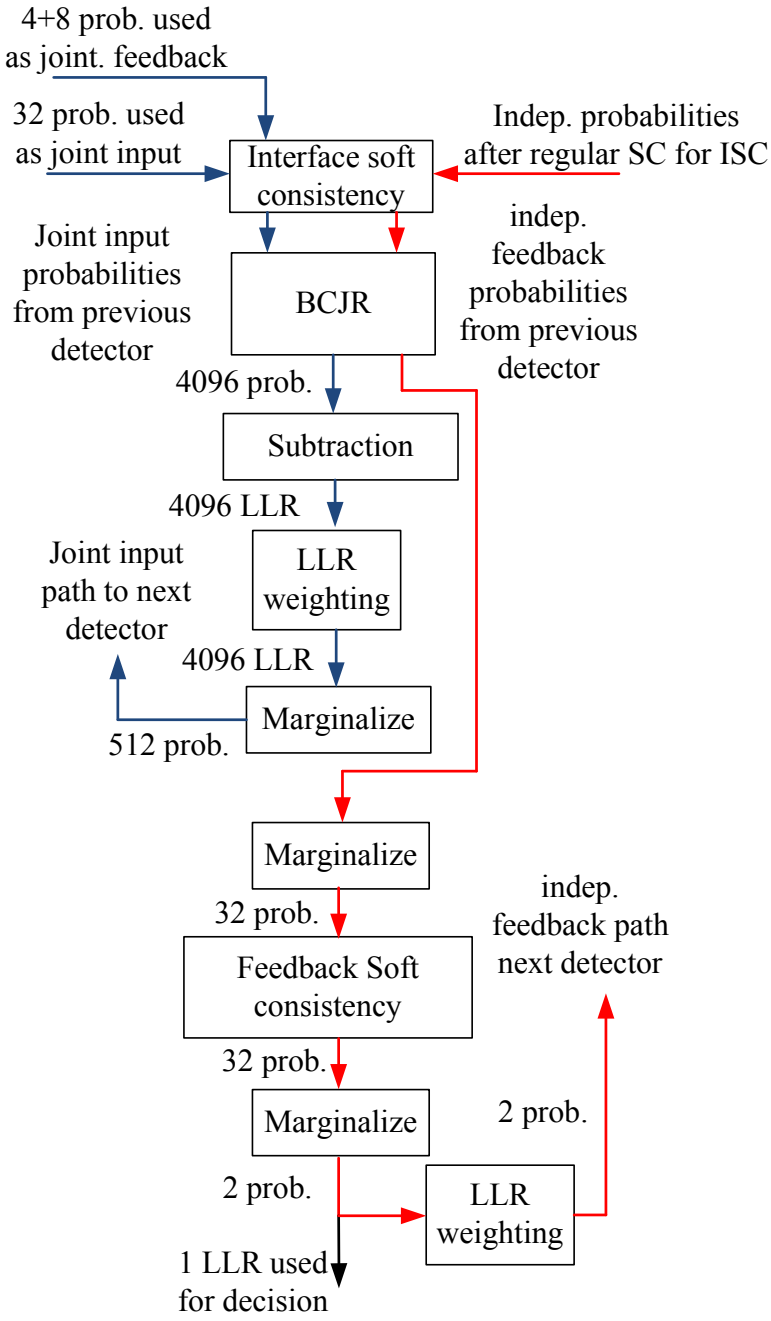


Figure 4.9: The final version of the BLKZ algorithm (3×3 case) with LLR subtraction and interface soft consistency updates for both input and feedback.

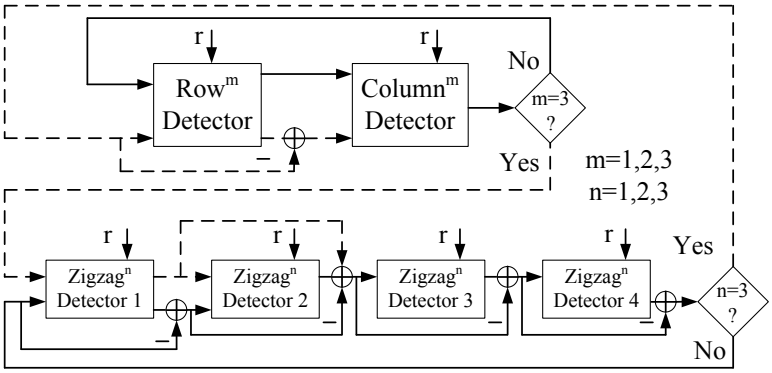


Figure 4.10: Diagram of the concatenated system of IRCSDFA and ISDFZA.

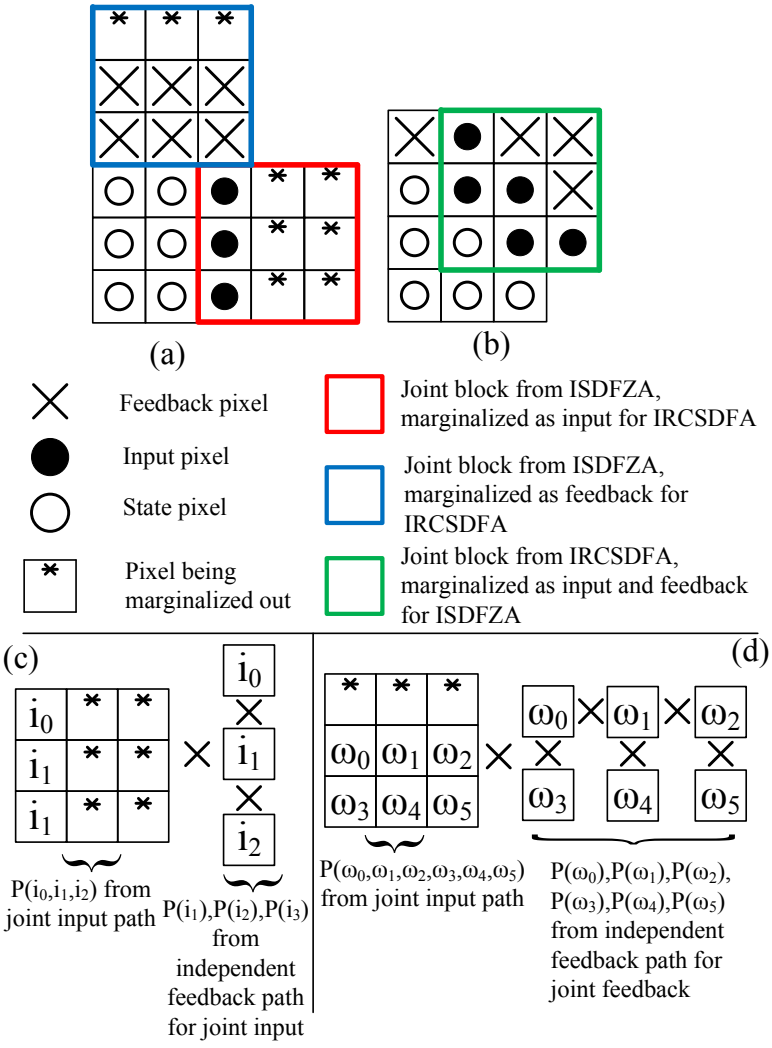


Figure 4.11: TEXIF chart for information exchange in the joint concatenated system: (a) the ISDFZA to IRCSDFA interface; (b) the IRCSDFA to ISDFZA interface; (c) the interface soft consistency update for input pixels on the ISDFZA to IRCSDFA interface; (d) the interface soft consistency update for feedback pixels on the ISDFZA to IRCSDFA interface.

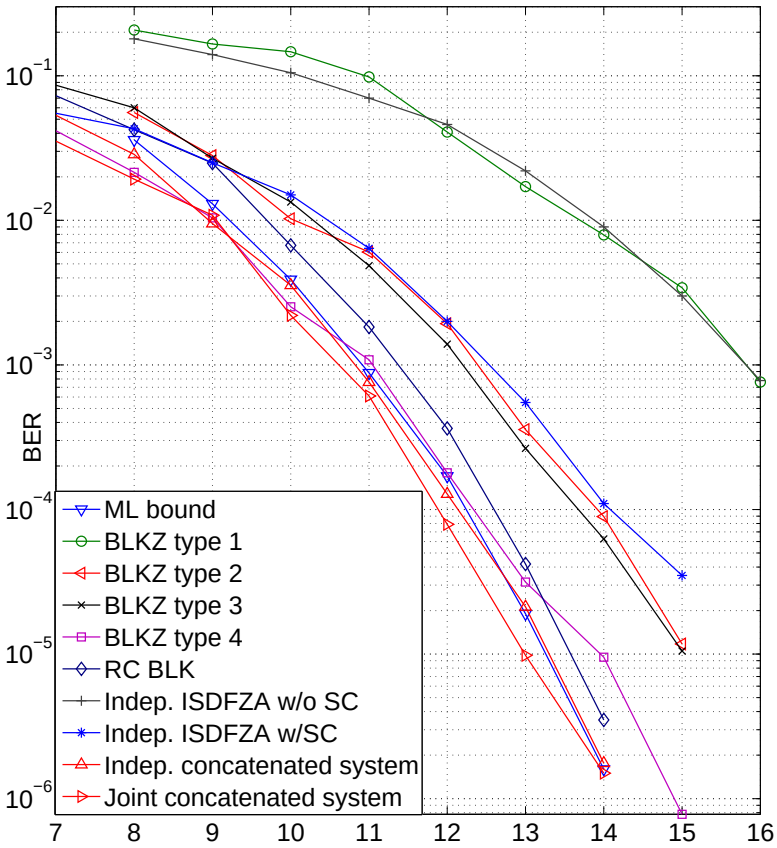


Figure 4.12: Monte Carlo simulations with 2×2 averaging mask for the four types of BLKZ algorithms, the joint IRCSDFA (RC BLK), the independent ISDFZA of [7] with and without soft consistency update, and the ML bound computed according to [11]. All curves are based on a 64×64 image.

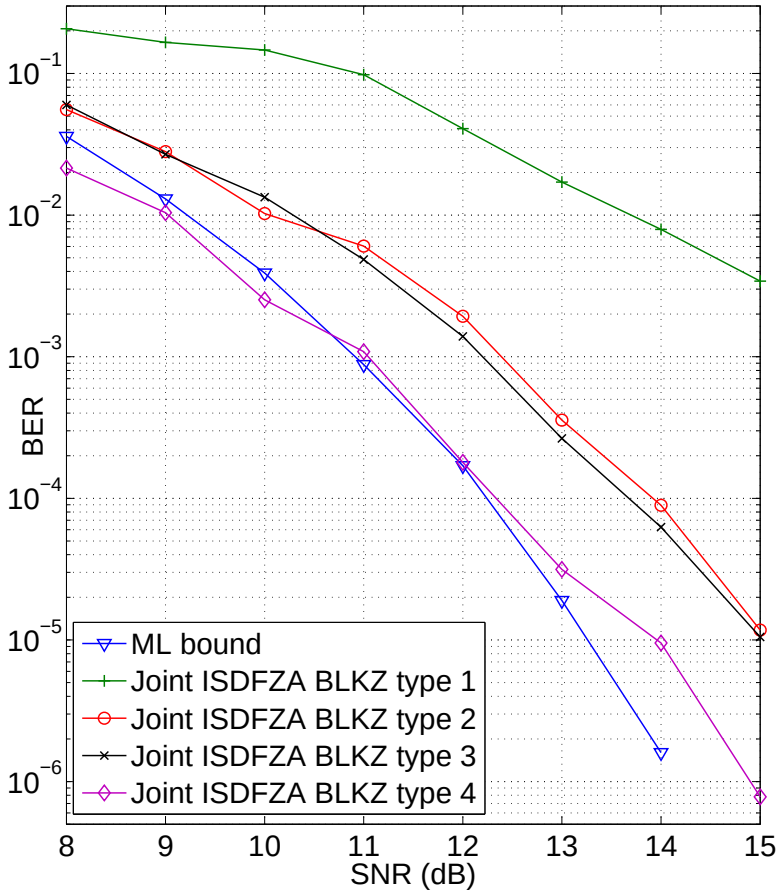


Figure 4.13: Monte Carlo simulations results show the evolution process of four different joint ISDFZA algorithm structrues, for 2×2 averaging mask.

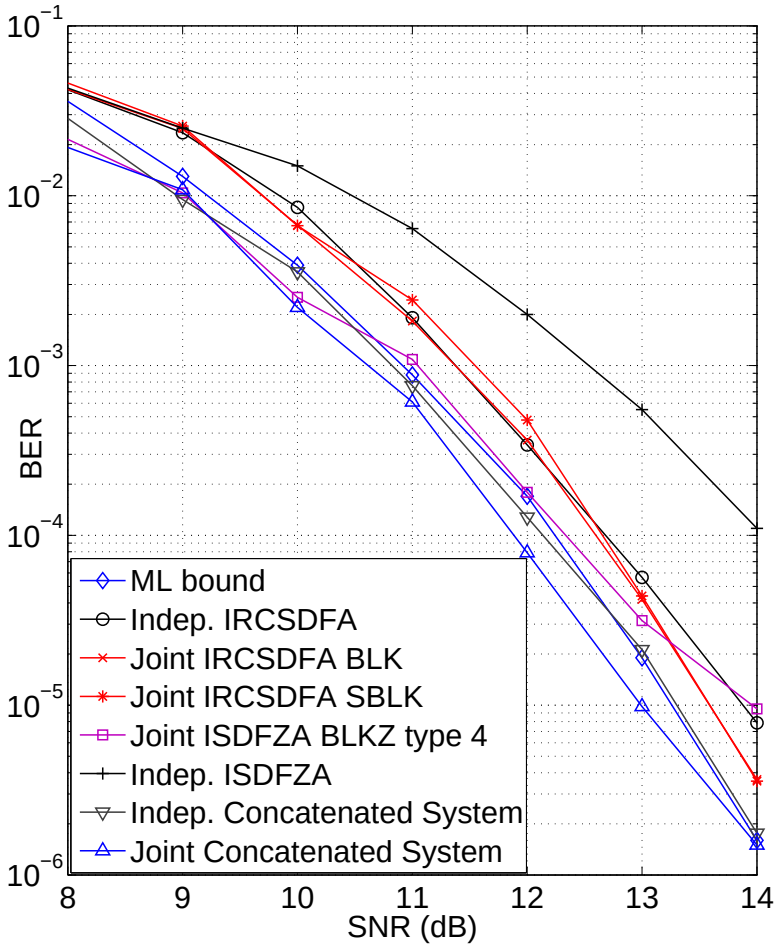
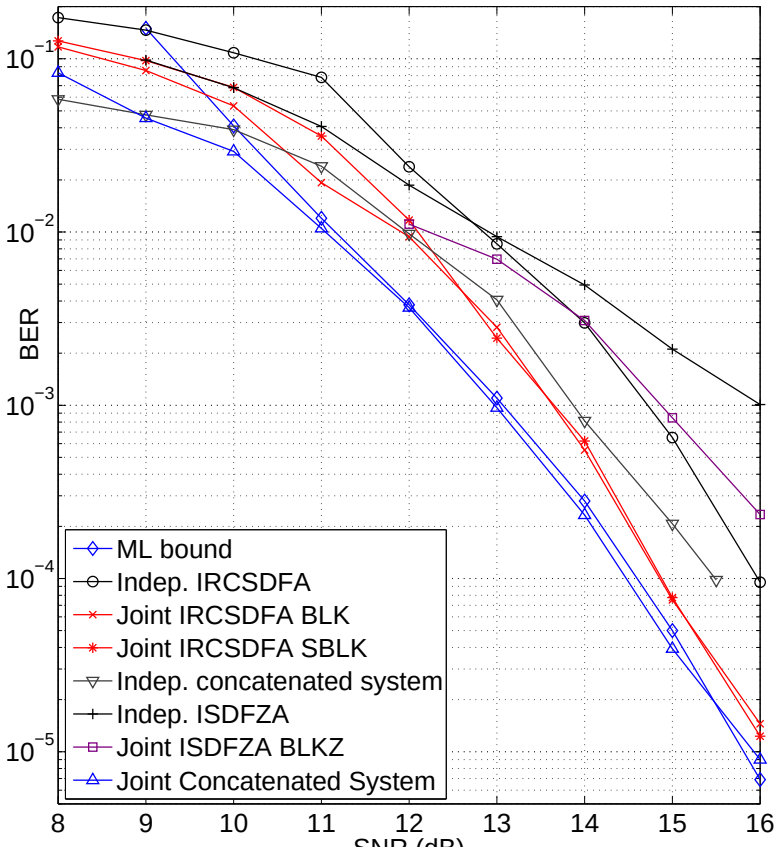


Figure 4.14: Monte Carlo simulations results for 2×2 averaging mask.



Figu

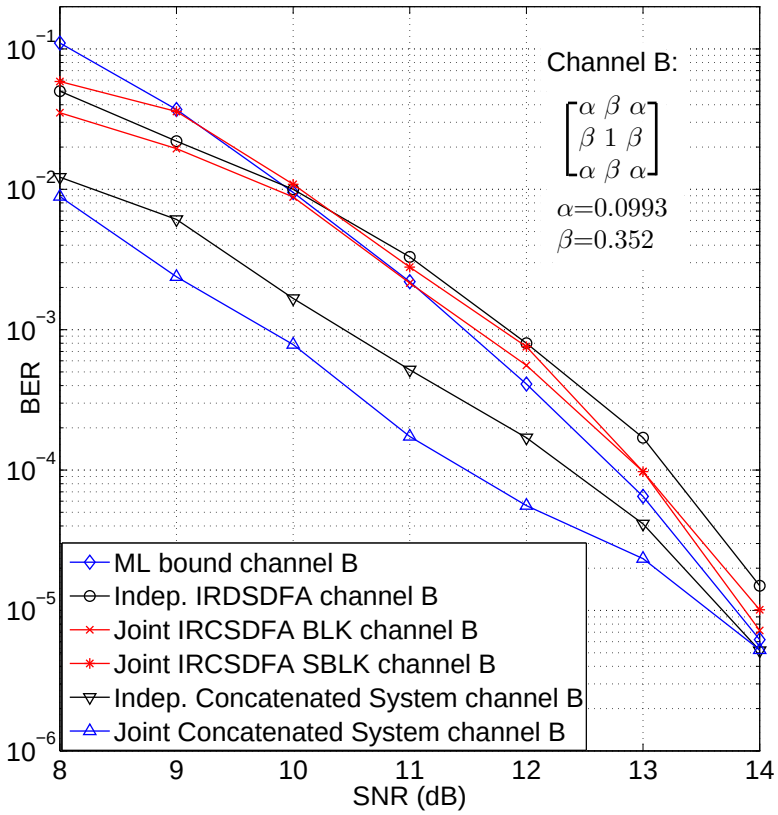


Figure 4.16: Monte Carlo simulations results for 3×3 channel B from [4].

Chapter 5

Conclusion

5.1 Summary of Thesis Contribution

In this thesis, we first introduced a novel equalization algorithm for two dimensional intersymbol interference channels, i.e. the iterative soft decision feedback zigzag algorithm (ISDFZA) for both 2×2 mask case and 3×3 mask case. We used the independent assumption for the modified BCJR algorithm as the core of this equalization algorithm. This ISDFZA applies quadratic weight scheme to the extrinsic information (log-likelihood ratio). We used the extrinsic information transimission (EXIT) chart to optimize the weight scheme, which is much more efficient than the optimization just by simulation. We then concatenated the zigzag algorithm with the previous iterative row column soft decision feedback algorithm (IRCSDFA), which has better performance than either one of the single algorithms. The zigzag algorithm gains about 1 dB over the row-column algorithm and over a separable-mask algorithm, two of the best previously published schemes. When the zig-zag algorithm is concatenated with the row-column algorithm, the concatenated system performs as well as or better than four of the best previously published algorithms, at both low and high signal-to-noise ratios (SNR), for a variety of 2×2 and 3×3 convolution masks;

in several cases, the system performs within less than 0.1 dB of the maximum-likelihood performance bound.

Subsequently, we modified the algorithms by using the joint extrinsic information, which we believe are more accurate based on the algorithm definition. First we implemented the change to the iterative row column soft decision feedback algorithm. The new algorithm, referred to as the block (BLK) algorithm, provides more than 1 dB SNR gain for the 2×2 averaging mask; the resulting performance is within 0.3 dB from the maximum likelihood bound for the 3×3 averaging masks. To address the increased computational and storage complexity introduced by the joint statistics, we have developed a simplified version of the block (SBLK) algorithm. Experimental results demonstrate that the SBLK algorithm performs almost as well as the BLK algorithm. One important advantage of the joint statistics is that these algorithms have better performance with a constant weight scheme than the previous linear or quadratic ones, which simplify the optimization process.

Later, we implemented the joint statistics to the iteratve soft decision feedback zigzag algorithm, by jointly estimating adjacent groups of source pixels. The new algorithm, called the block zigzag (BLKZ) algorithm, uses constant LLR weights across iterations, and provides more than 2 dB SNR gain over the ISDFZA on the 2×2 averaging mask channel; at low and medium SNRs it also outperforms the iterative row-column soft decision feedback algorithm (IRCSDFA) using joint extrinsic information. For the 3×3 averaging mask, the BLKZ algorithm gives more than 1 dB SNR gain over the independent ISDFZA at high SNRs. When the joint ISDFZA is concatenated with the joint IRCSDFA, the concatenated system achieves the maximum-likelihood (ML) bound at high SNR with the 2×2 averaging mask, with about 0.3 dB SNR gain in medium SNR region than the

previous independent concatenated system; with the 3×3 averaging mask the joint concatenated system gains about 1 dB compared to the concatenated system with independence assumption, and comes within 0.2 dB of the ML bound, which is the best performance in this two dimensional intersymbol interference channel detection area without channel coding.

In this thesis, we also introduced a new method of LLR subtraction for the BLKZ algorithm based on the algorithm's two-dimensional extrinsic information flow (TEXIF) chart. This TEXIF chart could enable us have a better view on the extrinsic information flow and give a better explanation on the subtraction issue for the joint algorithms.

5.2 Future Work

The 'uncorrelated assumption' for the computing the conditional probability in gamma probability of the modified BCJR algorithms in this joint ISDFZA for 3×3 mask channel may need be further explored, since a better approximation may exist.

The 'interface soft consistency update' (ISC) part may need to be studied more carefully, so that we could provide a more theoretical justification for this update.

We also plan to add channel coding, like LDPC code, serial concatenated convolutional code (SCCC), which will provide coding gains. The performance of the equalizer with channel coding will be compared to channel capacity. This is another future direction.

Since the joint statistics will bring higher dimensional LLRs, this will increase the complexity in both computation and storage exponentially. So, to explore the trade off between the performance and the time/space cost will be another interesting topic.

We are currently explore the application of the IRCSDFA and ISDFZA to the problem of two-dimensional meganetic recording (TDMR).

Bibliography

[1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.

[2] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[3] J. Chen and P. H. Siegel, "On the symmetric information rate of two-dimensional finite-state ISI channels," *IEEE Trans. Inform. Theory*, vol. 52, no. 1, pp. 227–236, Jan. 2006.

[4] X. Chen and K. M. Chugg, "Near-optimal data detection for two-dimensional ISI/AWGN channels using concatenated modeling and iterative algorithms," in *Proc. IEEE International Conference on Communications, ICC'98*, 1998, pp. 952–956.

[5] Y. Chen, B. Belzer, and K. Sivakumar, "Iterative row-column soft-decision feedback algorithm using joint extrinsic information for two-dimensional intersymbol interference," in *Proceedings of the 44th Conference on Information Sciences and Systems (CISS 2010)*, Princeton, NJ, March 2010.

[6] ——, "Iterative soft-decision feedback zigzag algorithm using joint extrinsic information for two-dimensional intersymbol interference," in *Proceedings of the 45th Conference on Information Sciences and Systems (CISS 2011)*, Baltimore, MD, March 2011.

[7] Y. Chen, T. Cheng, P. Njeim, B. Belzer, and K. Sivakumar, "Iterative soft decision feedback zig-zag equalizer for 2D intersymbol interference channels," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, Feb. 2010.

[8] T. Cheng, B. J. Belzer, and K. Sivakumar, "Image deblurring with iterative row-column soft-decision feedback algorithm," in *Proc. 39th Conf. on Info. Sci. and Systems (CISS2005)*, Johns-Hopkins U., Baltimore, MD, Mar. 2005, CD-ROM.

[9] ——, "Row-column soft-decision feedback algorithm for two-dimensional intersymbol interference," *IEEE Signal Processing Letters*, vol. 14, pp. 433–436, July 2007.

[10] T. Cheng, "Equalization and coding for the two-dimensional intersymbol interference channel," Ph.D. dissertation, Washington State University, 2007.

[11] K. M. Chugg, "Performance of optimal digital page detection in a two-dimensional ISI/AWGN channel," in *Proc. Asilomar Conf. on Signals, Systems and Comp.*, Nov. 1996, pp. 958–962.

[12] W. Coene, "Coding and signal processing for two-dimensional optical storage (TwoDos)," Mar. 2004, powerpoint presentation available at http://cm.bell-labs.com/cm/ms/events/WGIR04/pres/coene.ppt.

[13] D. Fertonani, A. Barbieri, and G. Colavolpe, "Reduced-complexity BCJR algorithm for turbo equalization," *IEEE Trans. Commun.*, vol. 12, pp. 2279–2287, Dec. 2007.

[14] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[15] J. F. Heanue, K. Gürkan, and L. Hesselink, "Signal detection for page-access optical memories with intersymbol interference," *Applied Optics*, vol. 35, no. 14, pp. 2431–2438, May 1996.

[16] G. T. Huang, "Holographic memory," *MIT Technology Review*, vol. 9, Sept. 2005.

[17] R. Koetter, A. C. Singer, and M. Tuechler, "Turbo equalization," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 67–80, Jan. 2004.

[18] R. Krishnamoorthi, "Two-dimensional Viterbi like algorithms," Master's thesis, Univ. Illinois at Urbana Champaign, 1998.

[19] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[20] S.-J. Lee, A. C. Singer, and N. R. Shanbhag, "Linear turbo equalization analysis via BER transfer and EXIT charts," *IEEE Trans. Sig. Proc.*, vol. 53, pp. 2883–2897, Aug. 2005.

[21] S. Lin and J. D. J. Costello, *Error Control Coding*, 2nd ed. Pearson Prentice Hall, 2004.

[22] M. Marrow and J. K. Wolf, "Iterative detection of 2-dimensional ISI channels," in *Proc. Info. Theory Workshop*, Paris, France, Mar./Apr. 2003, pp. 131–134.

[23] M. Marrow, "Detection and modeling of 2-dimensional signals," Ph.D. dissertation, University of California at San Diego, 2004.

[24] C. Miller, B. R. Hunt, M. A. Neifeld, and M. W. Marcellin, "Binary image reconstruction via 2-D Viterbi search," in *Proc. IEEE International Conference on Image Processing, (ICIP97)*, vol. 1, 1997, pp. 181–184.

[25] C. L. Miller, B. R. Hunt, M. W. Marcellin, and M. A. Neifeld, "Image restoration using the Viterbi algorithm," *Journal of the Optical Society of America A*, vol. 16, pp. 265–274, February 2000.

[26] A. Moinian, L. Fagoonee, W. M. J. Coene, and B. Honary, "Sequence detection based on a variable state trellis for multidimensional ISI channels," *IEEE Trans. Magnetics*, vol. 43, pp. 580–587, Feb. 2007.

[27] P. Njeim, T. Cheng, B. J. Belzer, K. Sivakumar, and Y. Zhu, "Image detection in 2D intersymbol interference with zig-zag decision-feedback Viterbi algorithm," in *Proc. 38th Conference on Information Sciences and Systems (CISS'04)*, Princeton, NJ, Mar. 2004, pp. 195–200.

[28] P. M. Njeim, T. Cheng, B. J. Belzer, and K. Sivakumar, "Image detection in 2D intersymbol interference with iterative soft-decision feedback zig-zag algorithm," in *Proc. 43rd annual Allerton Conf. on Comm., Computing, and Control*, Univ. of Illinois, Urbana-Champaign, IL, Sept. 2005, CD-ROM, paper 43.

[29] E. Ordentlich and R. M. Roth, "On the computational complexity of 2D maximum-likelihood sequence detection," *Hewlett-Packard Technical Report HPL-2006-69*, 2006, online at http://www.hpl.hp.com/techreports/2006/HPL-2006-69.html.

[30] L. Papke, P. Robertson, and E. Villebrun, "Improved decoding with the SOVA in a parallel concatenated (turbo-code) scheme," in *Proc. IEEE Int. Conf. Commun. (ICC96)*, Dallas, TX, June 1996, pp. 102–106.

[31] R. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 1003–1010, Aug. 1998.

[32] W. Ryan, "A turbo code tutorial."

Available online at http://www2.engr.arizona.edu/~ryan/publications/turbo2c.pdf

[33] O. Shental, N. Shental, S. Shamai, I. Kanter, A. J. Weiss, and Y. Weiss, "Discrete-input two-dimensional Gaussian channels with memory: estimation and information rates via graphical models and statistical mechanics," *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1500–1513, April 2008.

[34] J. B. Soriaga, M. Marrow, P. H. Siegel, and J. K. Wolf, "On achievable rates of multi-stage decoding on two-dimensional ISI channels," in *Proc. 2005 IEEE Int. Symp. on Info. Theory*, Sept. 2005, pp. 1348–1352.

[35] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[36] R. Wood, M. Williams, A. Kavcic, and J. Miles, "The feasibility of magnetic recording at 10 terabits per square inch on conventional media," *IEEE Trans. Magnetics*, vol. 45, no. 2, pp. 917–923, Feb. 2009.

[37] Y. Wu, J. A. O'Sullivan, N. Singla, and R. S. Indeck, "Iterative detection and decoding for separable two-dimensional intersymbol interference," *IEEE Trans. Magnetics*, vol. 39, no. 4, pp. 2115–2120, July 2003.