USC-SIPI REPORT #367

Iterative Detection for Page-Oriented Optical Data Storage Systems

by

Nopparit Intharasomabat

August 2005

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Viterbi School of Engineering
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Suite 400
Los Angeles, CA 90089-2564 U.S.A.

Acknowledgements

My doctoral study has been long and interesting journey, in which I have explored many facets of life during this period. There are many enjoyable experiences mixed with a few hardships. Thanks to a group of special people in my life that hardships are more bearable and good experiences become great experiences. I would like to express my most sincere thanks to each and every single one of them. In addition, I would also like to make a special acknowledgement to the following people.

First and foremost, I would like to express my deep appreciation towards my thesis advisor, Dr. Alexander A. Sawchuk who provides me with invaluable discussions, guidance and encouragements. His vast knowledge and incredible patience have made my academic career an enjoyable one.

I am also truly grateful for my dissertation committee members: Dr. B. Keith Jenkins, Dr. Vijay P. Kumar, Dr. C.C. Kuo and Dr. Wlodek Proskurowski for their constructive comments, and their invaluable time, consideration and kindness for serving on my committee. In addition, they are also my teachers whose knowledge I have learned benefits me greatly in this thesis.

Also, I thank many EE staffs including Gloria Halfacre, Dr. Alan Weber, Seth Scafani, Tim Boston and Diane Demetras who provide me with great assistances.

Thanks are extended to many of my friends, office mates, and research group members: Dr. Hyukjune Chung, Changki Min, Tawei Ho, Yunsong Huang, Zahir Alpaslan, Chiao Wang, Dr. Dhawat (Eddie) Pansantiankul, Ian Lee, Charchai Meesookho, Withaya Sankara, etc. I would especially like to thank my roommates: Sunan Tugsinavisut, Dr. Phoom Sagetong, etc. who share many of my experiences during these past few years.

Most importantly, I would like to express my deepest gratitude to my parents, Admiral Chamnan Intharasombat (RTN.) and Mrs. Boonsong Intharasombat whose unwavering love, trust and support have been immeasurable. My brothers: Ekkarat (Boat), Chainarong (Brig), and Napat (Bay) have undoubtedly played important roles in my study, and their supports are greatly appreciated. A special thank goes to my grandmother whose kind and thoughtful words have helped me through many difficult times during my study.

Last but definitely not least, I would like to deeply thank my lovely girlfriend, Donnapa Artpraserdkul who stands by with incredible patience, understanding and encouragement. Her love and encouragements had helped me through several turbulences during my study. She had shown me the true meaning of love, care and support.

TABLE OF CONTENTS

List of Variables (Alphabetical Order)

A	Pixel pitch
b	Number of bits in quantized system
b_S	Blur factor coefficient
C_1	Number of loops performed on the classification stage
C_2	Number of loops performed on the verification stage
d	Rayleigh limit distance
D	One unit delay
$d(x,y)$	Digital output
E_1	Energy level of an isolated pixel *one*
$f(x,y)$	Recording channel output
$h(x,y)$	Point-spread function
$h(x_0,y_0)$	Point-spread function elements at the closest neighbor locations
H	Transfer function
L	Constraint length
m	Number of total bits on a code rate
M	Number of levels
n	Number of user information bits on a code rate
N	Noise random variable

P_N	Probability of noise
$P(e)$	Probability of error
$r(x,y)$	Initial received intensity
$\hat{r}(x,y)$	Estimated pixel intensity
$\hat{r}_a(x,y)$	Added intensity
$\hat{r}_r(x,y)$	Residual intensity
$\widetilde{r}_v(x,y)$	Difference between the received intensity and the reconstructed value
R	Code rate
R_1	Class *zero*
R_2	Class *one*
R_3	Class *unclassified*
$s(x,y)$	Detector output
SNR	Signal to noise ratio level
T_1	Lower threshold value
T_2	Upper threshold value
T_{3A}	Update pixel *zero* sensitivity value
T_{3B}	Update pixel *one* sensitivity value
T_4	Fixed threshold value

T_5	Allowable deviation value
T_{lower}	Lower symbol update sensitivity value
T_{upper}	Upper symbol update sensitivity value
$u(x, y)$	User data
$u_M(x, y)$	Modulated user data
$v(x, y)$	Current digital pixel value
$v_a(x, y)$	Mapped current digital pixel value
$valid\,(x, y)$	Pixel validity value
x	Horizontal axis coordinate
$x(n)$	Sample input
y	Vertical axis coordinate
$y(n)$	Sample output
$Z_{\max}$	Maximum noise tail (multiple of standard deviation)
Δd_c	Difference between received intensity and corrected intensity
γ	Interpage interference coefficient
σ	Additive white Gaussian noise standard deviation

List of Tables

List of Figures

Abstract

Data storage requirements have rapidly increased in the past few years due to the development of several storage-intensive applications. Beside high storage capacity, other desirable characteristics of data storage systems are high data transfer rate and fast access time. Page-oriented optical data storage systems (PODS) with their parallel readout channels and volumetric storage area have good potential for becoming the next generation data storage system. In particular, we develop signal processing techniques for two-photon PODS systems.

We concentrate on modulation and detection aspects of the system including protecting data against sources of interference that cause errors. The three main interference sources are intersymbol interference (ISI), interpage interference (IPI) and additive noise. Our main effort is to mitigate ISI and IPI in the presence of noise. The level of ISI and IPI becomes more pronounced as the data bit packing density increases. Typical methods of combating an ISI problem (a channel with memory) are equalization-based methods. However, the computational complexity of equalization methods depends on the channel memory length, which is directly proportional to the level of ISI and IPI.

We describe an iterative algorithm and extensions that mitigate ISI and IPI in the presence of additive noise, while maintaining acceptable bit-error-rate. The algorithm uses a decision-feedback equalization-like approach. The algorithm effectively removes the ISI effects contributed by neighboring pixels from the received intensity and either makes a decision based on this result or defers making a decision if the result cannot be classified with a strong certainty. The decision is later verified by checking the consistency between the estimated value and the received intensity. The pixels that fail the consistency test are assigned a new value by following a set of correction rules.

Our algorithm is further extended to mitigate a combination of ISI and IPI. The IPI problem is essentially a three-dimensional ISI problem and the relevant data pages extend from several preceding pages to several proceeding pages. To solve this causality problem, a pipelined processing is implemented. We also extend our algorithm to implement multi-level encoding. A joint error correction/detection method is used to combat ISI effect with non-binary encoding.

Chapter 1

Introduction

1.1 Motivation and Objective

Information has become an integral part of our everyday life, and data storage devices have certainly become crucial elements in various functions and applications such as movie archival, database systems, data acquisition systems, etc. The demands for larger and faster storage systems are apparent and continue to grow from many emerging storage intensive applications. A good data storage system is qualitatively associated with high storage capacity. However, with the invention of many real-time systems and algorithms, the access time and the data transfer rate have become critical parameters for data storage systems. The data accuracy, in terms of the bit-error-rate, is also an important property of data storage systems. Therefore, the overall performance of a data storage system can be measured by storage capacity, access time, data transfer rate, and data accuracy. It is possible to obtain trade-offs between these four factors and improvements in all of them by adopting various signal processing techniques.

Currently, nearly all data storage systems are planar type, which store information on a planar storage media surface. They typically have a single readout channel, although in some systems, a limited number of these channels can read and write data in parallel. These planar data storage systems include magnetic type (hard disk) and optical type (compact disc or CD, and digital versatile disc or DVD). Both systems store data on their media surfaces. In magnetic storage systems, the media surface is coated with magnetic grains. A physical data bit is a contiguous region on the surface that contains many magnetic grains. The data is recorded by arranging the magnetic polarity of these bits. In optical storage systems, the media surfaces are made of reflective material, and the data is recorded by etching two different depths representing two different binary values. Figure 1.1 illustrates basic magnetic and optical disc systems and shows the limitation of the storage areas in a planar type data storage system. However, the major disadvantage of planar type data storage is the lack of parallelism in its readout subsystem, leading to low overall data transfer rates.

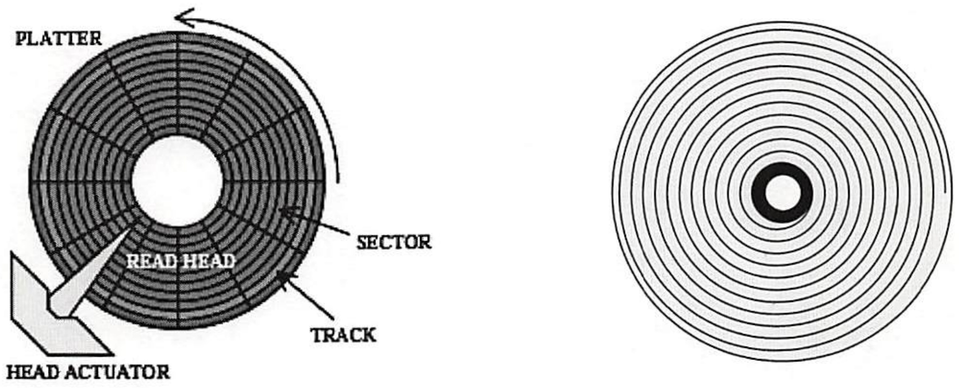


Figure 1.1: Data storage systems (Top view): Left: Magnetic hard disk. The disk contains multiple platters and each platter contains multiple tracks and each track contains multiple sectors. Right: Optical disk system. Optical disk has a single track. A spiral read pattern is shown.

There are various algorithms that deal with signal detection from one-dimensional (serial) data streams. Well-known algorithms are the Viterbi algorithm (VDA) [9], partial response maximum likelihood (PRML) [23], etc. In this thesis, the data streams are two-dimensional (2D) streams, which are the output of a page-oriented memory. The two-dimensional data streams must be treated differently from one-dimensional streams because of additional interference not present in a typical one-dimensional stream.

Page-oriented optical data storage systems are essentially three-dimensional (3D) systems, and are affected by three major types of signal interference: inter-symbol interference (ISI); inter-page interference (IPI); and measurement noise. Thus, a three-dimensional signal processing approach is required. Some efforts have been made to extend existing one-dimensional algorithms to higher dimension (2D and 3D) format. These one-dimensional algorithms perform well against one-dimensional inter-symbol interference. However, due to differences in the characteristics of the two-dimensional data, they have been shown to be inefficient with the two-dimensional data stream format. Another aspect of the algorithm that must be considered is the computational complexity. Some algorithms have relatively good performance, but have extremely high computational complexity. Page-oriented optical data storage contains multiple parallel readout channels and thus produces a very high data throughput rate. To avoid data processing bottlenecks, a new approach that has a good balance between good performance and low complexity must be explored.

1.2 Organization

This thesis is divided into seven chapters, and they are organized as follows. In chapter 2, we provide a background on conventional data storage systems, next generation data storage systems and their detection techniques. It includes the system level description, the channel model description, noise model description, and detector configuration description. Earlier works done by various researchers are also reviewed. In chapter 3, the proposed iterative decoding algorithm is described in detail, and an example is provided. In chapter 4, bit-error-rate analysis and comparison of the two dimensional interference are presented. In addition, a complexity analysis of the algorithm is provided. In chapter 5, the adaptation of the algorithm to inter-page interference problem is addressed. This adaptation aims at two and three-dimensional interference. The reorganization of the algorithm structure is described, and its performance is presented. Chapter 6 describes grayscale-encoding applications in page-oriented optical data storage. The problem statement and solution are shown. In this chapter, a simple joint modulation/error correction scheme that takes advantage of the available unused pixels is described. Lastly, some future work and possible extensions are discussed.

1.3 Contributions

In this thesis, we describe algorithms that mitigate the effect of two-dimensional (2D) inter-symbol interference and 3D inter-page interference to improve the overall bit-error-rate performance. We also describe data pattern configurations that allow higher data packing density and lower calculation load. The goals of this thesis are to explore new approaches to increase data capacity and data transfer rate while maintaining acceptable data accuracy (bit-error-rate). We later describe an error correction scheme that works in conjunction with the detection algorithms presented in this thesis to allow grayscale encoding.

Chapter 2

Background

This chapter provides preliminary background information on conventional data storage systems including magnetic and optical, some next-generation data storage systems, and data storage system signal processing techniques.

2.1 System Background

2.1.1 Mainstream data storage systems

First, we examine some common data storage systems and their capacity barriers. There exist two major types of data storage systems: magnetic recording systems and optical recording systems. These data storage systems are classified by their storage medium. Their succinct descriptions are described in the next few sections.

2.1.1.1 Magnetic recording systems

Magnetic recording systems are the most common media in data storage systems, and the most common format is magnetic hard disks. The media are available in various form factors, capacities and transfer rates. The physical characteristics and brief operation process of the system are described in this section. The media are flat disks called platters, which are coated with ferromagnetic films that contain small magnetic grains. Each of these small grains has north and south magnetic poles.

Data is stored on the disk in the form of magnetized areas representing bits. Each bit occupies a contiguous area of the disk containing many grains. During the recording process, the hard disk write head induces all of the grains in the bit area to align in the same direction. As the head proceeds along the disk, it changes the polarity of grains in a local area to record the bits.

Magnetic recording systems represent bit information using the flux reversal signal rather than individual bit polarity. A flux reversal signal is generated when there is a change in the polarity between two adjacent bits. The flux reversal signal is encoded using modulation-encoding schemes. Modulation encoding is the mapping between digital data and the actual analog physical signals encoded on the disk. There are several types of modulation encoding for magnetic recording systems such as: frequency modulation (FM), modified frequency modulation (MFM), and run length limited (RLL). The most efficient and commonly used encoding is the run length limited encoding.

In magnetic recording systems, the storage capacity is constantly being increased in newer generation hard disks using various techniques. The most common approach is by reducing the magnetic grain size, which allows tighter grain packing density leading to higher capacity. However, as the grain size decreases, the separation between two adjacent bits becomes smaller. As a result, the adjacent bits start to interact with each other and become magnetically unstable. This effect is called the *superparamagnetic* effect (SPE) [35]. This particular parameter indicates the highest data packing density in which the data can be reliably stored on the media. The location and severity of the SPE are unpredictable as shown in Fig. 2.1. The SPE can be avoided by reducing the magnetic field strength and thus lowering the bit interactions. However, the reduced magnetic field strength leads to lower signal level and more sophisticated detection procedures are required. The superparamagnetic packing density is generally estimated to be between 20 and 40 gigabits per square inch (GB/in.^{2}).

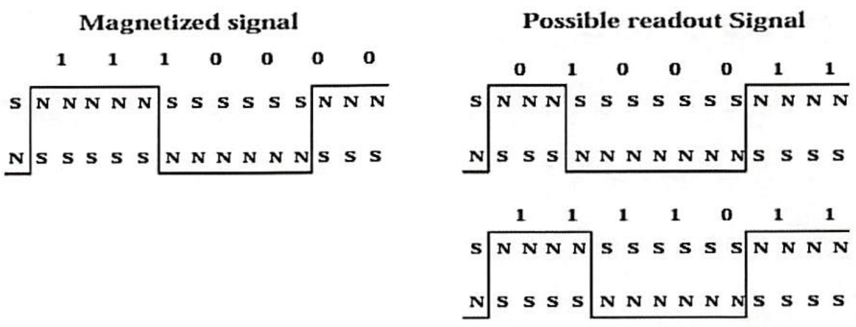

Magnetized signal
1 1 1 0 0 0 0
S N N N N N S S S S S S N N N
N S S S S S N N N N N N S S S
Possible readout Signal
0 1 0 0 0 1 1
S N N N S S S S S S S N N N N
N S S S N N N N N N N S S S S
1 1 1 1 0 1 1
S N N N N S S S S S S N N N N
N S S S S N N N N N N S S S S

Figure 2.1: Superparamagnetic effect (SPE): Using RLL(2,7) modulation encoding Left: original magnetized signal. Right: when recording density exceeds the SPE, unpredictable polarity interaction errors occur, leading to multiple possible readout signals.

Another SPE remedy is an anti-ferromagnetic-coupled technique created by IBM [10]. This technique inserts a layer of ruthenium (*Ru*, three atoms thick) to separate two layers of magnetic bits, allowing the recorded bits to be smaller. The anti-ferromagnetic-coupled technique helps to increase the superparamagnetic constraint in media development. This technology extends the superparamagnetic effect limitation, allowing the packing density to approach 100 GB/in.^{2}. However, the superparamagnetic effect problem continues to be an imminent problem and is likely to reach its limit in the near future.

2.1.1.2 Optical recording systems

The compact disc (CD) and digital versatile disc (DVD) are currently the mainstream of optical data storage systems. The surface of optical disc is made of reflective material. During the recording process, two different depths (flat surface and pit) are etched onto the reflective surface to represent the two binary levels. Figure 2.2 is a simple schematic of material structure and readout mechanism of the optical disc. The readout mechanism consists of the readout laser and detector placed at pre-calibrated angles. When the light falls on the detector, the system detects a bit *one* signal, and a bit *zero* is detected if the light falls outside of the detector.

The packing density limitation of these systems is apparent due to their physical configurations. The media are primarily designed for removable data storage purposes, and their physical and software specifications must conform to their respective standards to ensure readout compatibility. The common standard capacities are 640 and 700 MB (Megabyte) for CD and 4.7 GB (Gigabyte) for single-sided single-layered DVD disc. These specifications allow little room for modifications, and capacity diversities in these systems are uncommon.

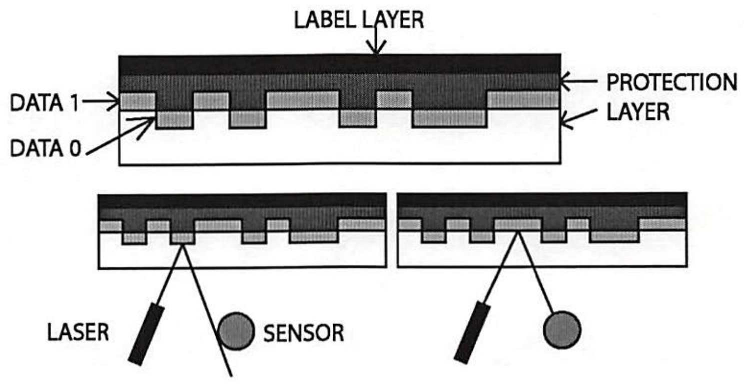


Figure 2.2: Optical disk system: cross section of media surface and readout mechanism.

2.1.2 Next generation data storage systems

In recent years, several next generation data storage systems have been proposed and are under investigation. In this section, we describe three next generation data storage systems, holographic optical data storage, persistent spectral hole burning, and two-photon absorption optical data storage.

A class of systems called page oriented optical data storage systems (PODS) is particularly interesting. It provides many attractive advantages that include higher capacity, faster throughput transfer rate, and potentially lower cost. There are two basic types of page oriented optical data storage systems: holographic and imaging. The former is described here in this section and the latter is the basic system for this thesis that is described in detail in the next section.

2.1.2.1 Holographic Optical Data Storage Systems

Holographic systems are one of the most actively developed data storage systems. In holographic optical data storage, the system uses a storage medium capable of storing a high spatial frequency interference pattern. The recording and readout processes use a laser to make a spatially coherent recording and readout system. Figure 2.3 (top) shows the recording process. For recording, the light beam is split into a data beam and a reference beam. The pixels on a two-dimensional spatial light modulator (SLM) page represent the binary data to be encoded. Each pixel on the SLM can block or transmit light depending on the binary data encoded. The SLM produces the encoded beam. This encoded beam made to interfere with the reference beam through the recording medium volume. The data is stored in storage medium in the form of interference patterns. Multiple pages of data can be superimposed onto the same medium volume by varying the characteristics of the reference beam such as beam angle, or wavelength.

During the readout process, a known reference beam is used to illuminate the medium. The resulting output is the reconstruction of the stored data page that is projected onto two-dimensional detector array. Other superimposed pages can be reconstructed by illuminating the medium with appropriate reference beams. Figure 2.3 (bottom) illustrates the readout process of holographic data storage. In the past, it has been difficult to construct holographic data storage systems due to lack of suitable components and more importantly the lack of suitable storage media. Advances in optics, electronics, and chemistry have allowed the realization of such systems. In recent years, many researchers have demonstrated many implementations of the systems [1], [5], [6], [7].

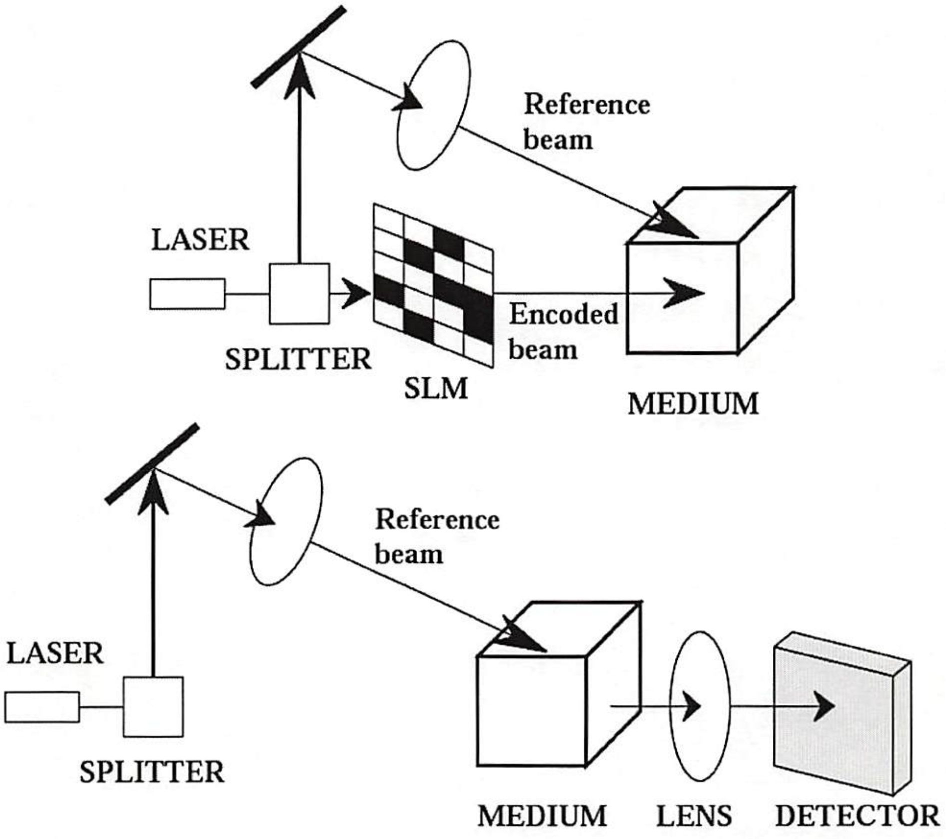


Figure 2.3: Holographic page-oriented optical data storage: Top: recording process, Bottom: readout process.

2.1.2.2 Persistent Spectral Hole Burning Optical Data Storage Systems

Persistent spectral hole burning (PHB) is a technique that employs a phenomenon occurring in solid-state materials. These materials have a temporal frequency response with broad inhomogeneous absorption lines. The absorption line is the composition of multiple zero-phonon lines (ZPL). These lines describe the absorption characteristic of the molecules. A process called "bleaching" changes part of the absorbing molecules to selectively burn a narrow spectral null (spectral hole) in the inhomogeneous absorption line. To create these holes at a specific spatial location on the medium, that spatial location is illuminated with a narrow band laser tuned to specific frequencies within the inhomogeneous absorption line. Multiple holes can be burned onto the medium spectrum at specific spatial locations. The presence and absence of holes at designated frequencies represent the binary value of *zero* and *one* respectively. The data is read out from each location on the media by inspecting the absorption line. Figure 2.4 shows the process of absorption line modification.

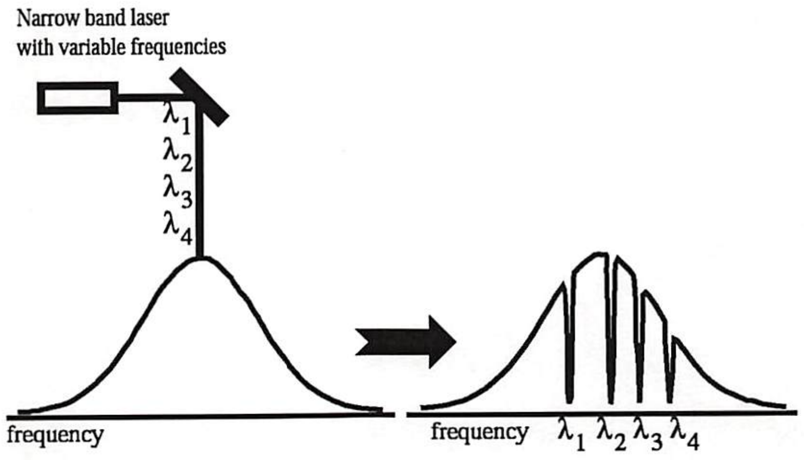


Figure 2.4: Persistent spectral hole burning: inhomogeneous absorption line hole burning using narrow band laser with tunable variable frequencies.

The maximum number of spectral holes that can be burned onto the material is the ratio of the bandwidth of the inhomogeneous absorption line to the bandwidth of the

ZPL. With the same laser spot size (determined by the diffraction limit) as conventional optical data storage, the optimistic estimate of the areal density is on the order of 10 Gbits/cm^2. Clearly, this technology holds a great potential for massively large data storage systems. However, one system requirement is very difficult to achieve: the system must be maintained at extremely low temperatures to sustain sharp resolution of the ZPL lines. The resolution of the ZPL lines determines the sensitivity of the detector. The ideal operating temperature is a few degrees above absolute zero temperature (0°K, - 273°C, or – 459.4°F). A few material requirements must also be addressed to construct a realizable system. These materials must be able to produce high inhomogeneous to ZPL bandwidth ratio at room temperature and to allow multiple read processes without gradual erasure of the data (material degradation). Another disadvantage of these systems is the lack of readout parallelism, although a transfer rate of 20-30 Mbps can be achieved due to its multi-level encoding. This technology is under development and has great potential in storage capacity improvement.

2.1.2.3 Two-Photon Absorption Optical Data Storage Systems

Two-photon absorption systems benefit from the physical characteristic of specially synthesized polymers. The molecules of these polymers efficiently absorb light (photons) of a specified wavelength. The molecule absorbs photon (energy) to move through the energy band gap to reach the excited state. The chemical characteristics of the molecules change at the excited state and yield useful properties (such as changes in temporal frequency response, etc.). In some cases, the energy band gap is too wide to be reached by absorbing one photon, and two photons are needed to reach the excited state. The molecule absorbs the first photon to reach a virtual state and simultaneously absorbs the second photon to reach the excited state. Thus, lower energy photons can be used in a two-photon system allowing lower power consumption. Figure 2.5 shows the simple absorption diagram of the two-photon system. The advantages of two-photon technology are apparent in the application of volumetric data storage. The first advantage is in low power consumption. The second advantage derives directly from the unique absorption process. The system can selectively excite molecules in a specific area or volume without perturbing the molecules in the penetration path. This is possible because the coincidence of two photons is needed to excite a molecule at a targeted area. The third advantage is the application to multi-level signaling. The excitation probability depends on the intensity of the recording beam, and the number of molecules being excited can be adjusted to multiple levels by simply changing the intensity level of the write beam.

The efficiency of the system depends on the media's ability to absorb two photons. Currently, there is a physical system under development that shows promising results [34]. In this thesis, we adopt a page-oriented optical data storage model that uses two-photon absorption technology and describe it in detail in the next section. The terms imaging page-oriented optical data storage and two-photon absorption optical data storage will be used interchangeably.

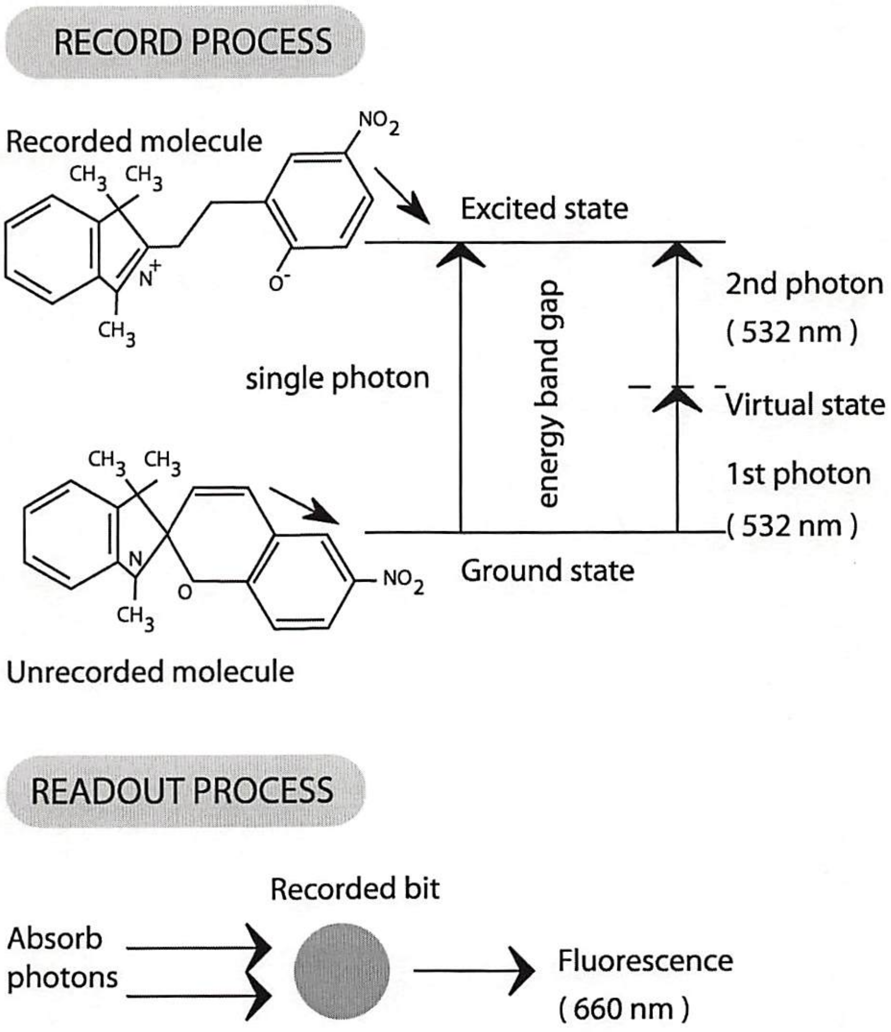


Figure 2.5: Two photon absorption diagram.

2.2 Imaging Page-Oriented Optical Data Storage Systems

In this section, imaging page-oriented optical data storage (PODS) systems used in this thesis are described in detail. The imaging PODS system is modeled using two-photon absorption technology. It utilizes three-dimensional (volumetric) storage as compared to two-dimensional planar-based systems. The additional dimension greatly increases the storage capacity, and also relaxes the packing density constraint leading to a lower bit-error-rate of the playback process. In addition, PODS systems allow for a faster data transfer rate due to its parallel output channels. Conventional data storage has a serial readout channel, and uses a single data processing unit, which limits the output data transfer rate. Imaging PODS systems provide parallel readout channels. Multiple processing units can be used in parallel to reduce the load on a single processor or additional processing can be added to lower the bit error probability of the modulation encoding and decoding process.

Figure 2.6 shows the typical arrangement for an imaging PODS disk. The shape of the media disk is similar to a regular compact disk with thicker depth. The data pages are arranged along the cross section of the disk (gray wedges in Fig. 2.6 top view). A media disc contains multiple tracks arranged in concentric rings, and each track contains multiple two-dimensional (2D) data pages. This configuration is similar to that in a compact disc or DVD disc, whereas the PODS disk has much greater potential capacity. While CD or DVD disc record bits serially on a track, PODS records 2D parallel pages of data on a track. The potential of capacity improvement is limited by only the thickness of the media in recording and playback process.

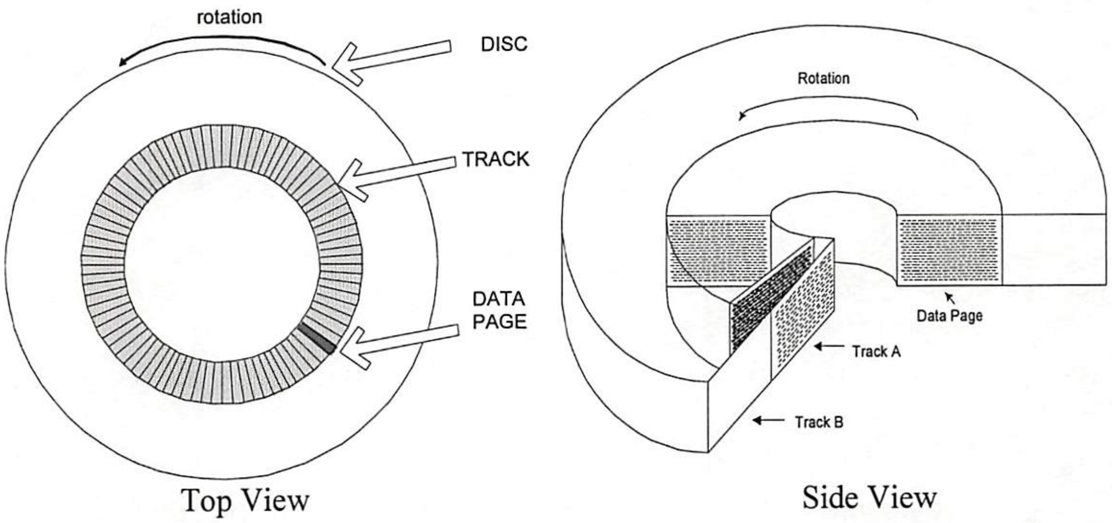


Figure 2.6: Cross section of media disk.

The illustration of the typical physical arrangement of a two-photon page oriented data storage system is shown in Fig. 2.7. The system consists of three main components: storage media disk, an imaging lens that optically transfers the data, and a detector array. The media disk is made of a light sensitive polymer that locally changes its fluorescence characteristics when it absorbs light at certain wavelengths. Data are recorded by the use of two intersecting beams that imprint the data on the data pixels. The data are read out by illuminating the rotating disk with a readout beam sheet. The readout beam covers an individual data page at a single readout instant. The page illumination readout allows for parallel readout process. Each data pixel on a data page independently reacts to the readout beam and fluoresces, and the resulting fluorescence is proportional to its recorded value. The image of emitting data page is projected onto the detector array through the imaging lens. Similar to conventional optical disc, a constant data throughput rate is desired, the media disc rotates at constant angular velocity around its axis giving a constant page throughput rate, and the overall data throughput rate depends on the number of parallel output data channels (number of data pixels on a single page).

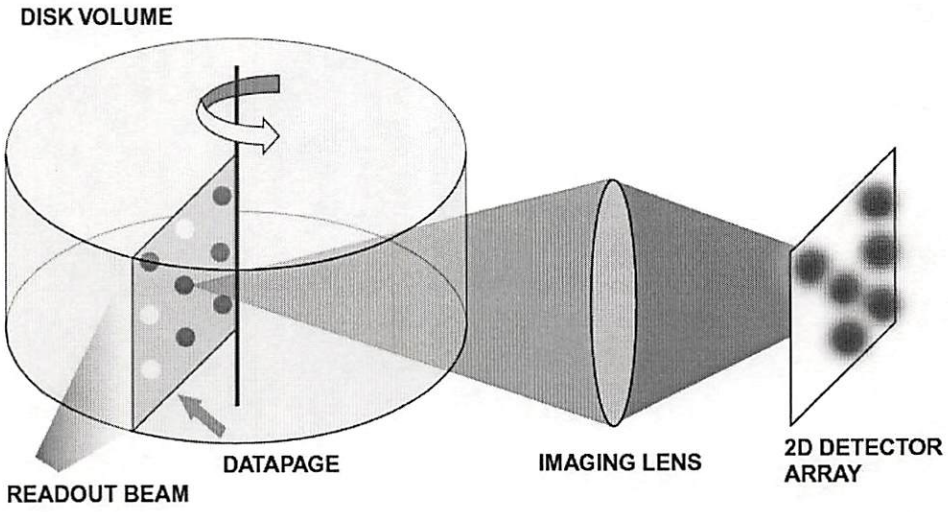


Figure 2.7: Readout process of two-photon imaging optical data storage.

2.2.1 Pixel Coordinate System

An important contribution of this thesis is the use of an unconventional coordinate system in the PODS system. Two possible detector layout configurations are shown in Fig. 2.8. In a conventional sensor (detector) array, a rectangular grid with equal spacing of A units in both the vertical and horizontal direction (rectangular grid, shown on the left). Another alternate configuration is shown on the right, where a pixel is equidistant to all its neighbors. The configuration has the shape of hexagon, and we will term it hexagonal grid. By adopting the hexagonal grid, the areal density of the media is improved by approximately 15% for a detector array with moderate size. The areal density improvement may offer a larger potential capacity depending on other system parameters. A portion of the increased potential capacity can also be allocated to offer data redundancy coding leading to higher overall data integrity.

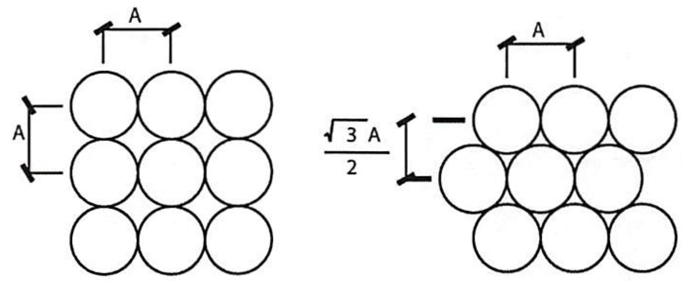


Figure 2.8: Detector grid coordinate system Left: Rectangular coordinate system, Right: Hexagonal coordinate system.

There are trade-offs between the two comparing coordinate systems. The rectangular grid is widely adopted by many system integrators and spatial light modulators. Also, the detector arrays with such geometry are much more common, and sometimes commercially available. In addition, most signal processing algorithms assume a rectangular two-dimensional data format, which allows much easier indexing both in hardware and software implementations. However, the disadvantage of the rectangular grid compared to the hexagonal grid is its lower packing density and overall storage capacity potential. On the other hand, hexagonal grid formation provides a higher packing density and it may be beneficial to algorithms that require processing of neighbor data pixels because the six neighbors surrounding a given data pixel are equidistant. This effective circular symmetry may reduce the detection signal processing involving weights of neighbors. This advantage becomes more apparent in hardware implementation. The hexagonal grid configuration is adopted in this thesis. Although, the hexagonal grid may offer higher packing density, it may also increase interactions between the neighbor data pixels leading to higher inter-symbol interference and lower overall signal-to-noise ratio. These system issues are discussed and explored in this thesis.

2.2.2 System Model

From Fig. 2.9, the PODS system consists of three main components: recording subsystem (transmission), storage medium (channel), and playback subsystem (reception). The recording subsystem conditions and transforms the data to make it more resistant to the external noise sources. It is divided into two encoding stages: error correction encoding (ECC) and modulation encoding. The error correction encoding accepts binary data and adds redundancies to ensure the data integrity and enable the correction of errors that may occur in the data stream. Figure 2.9 shows that a modulation encoder follows the ECC encoder. Modulation encoding transforms a binary data stream into analog signals that are physically placed on the recording medium. The readout subsystem contains complementary components that reverse the process and recovers the binary data.

Both modulation encoding and ECC are necessary for data storage systems. It is an interesting research question whether signal processing should be used for more complex modulation or for ECC. Though ECC units can successfully recover data from highly corrupted streams, there is a tradeoff between higher correctibility (ability to detect and correct errors) in ECC and the additional overhead due to redundant pixels needed for such feature. As the ability to correct errors grows, the overhead due to redundant pixels generally grows. As a result, the total usable data becomes low. Thus, there must be a balance between modulation encoding and the overall detection algorithm. The key emphasis in this research is on the use of new detection schemes.

A system description of the detection process is shown in Fig. 2.10.

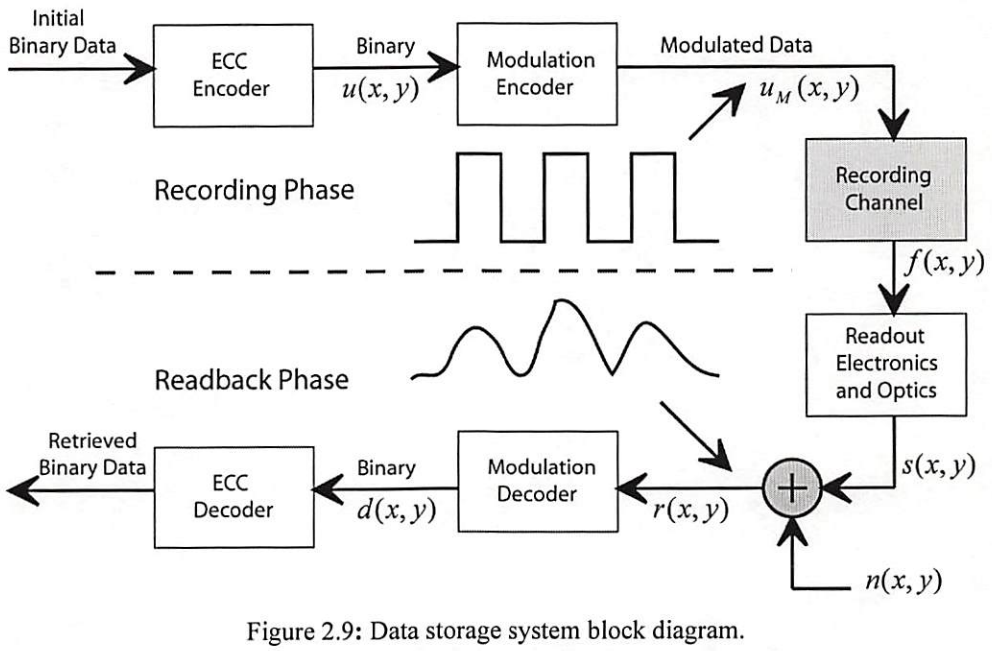


Figure 2.9: Data storage system block diagram.

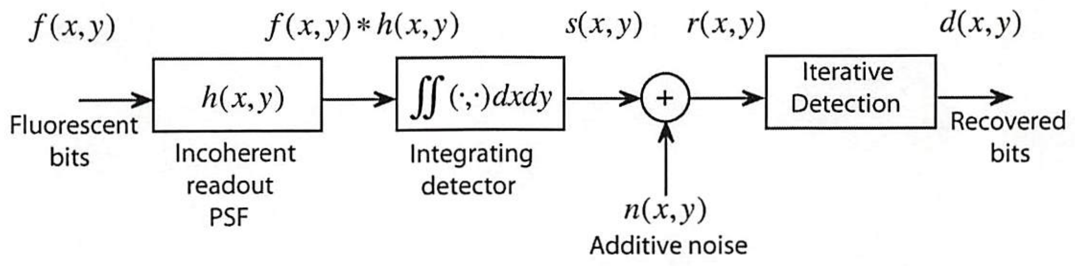


Figure 2.10: Model for the digital optical storage channel detection system.

The optical imaging in the system is described by the incoherent point-spread function denoted by $h(x,y)$. For two-photon storage, the recording and playback processes are spatially incoherent, so that the received signal is the magnitude of the light intensity. Therefore, the optical detector used in the system is modeled as a spatial integration of the received signal. It collects photons over its effective active area and translates the total photon count into a proportional electrical signal level. The commonly-used detectors are charged-coupled device (CCD) or complementary metal oxide silicon (CMOS). The data storage system generally suffers from noise interference (measurement error) from both internal and external sources. In this thesis, we assume additive noise for simplicity, although other more complicated noise models could be adopted. Finally, the information is processed by the detection algorithm, which is presented in chapter 3.

The complexity of the overall system depends mainly on the combined complexity of the modulation decoding and the error correction decoding systems. Our goal is to find a balanced configuration that lowers the overall system complexity. A very important parameter that is considered in the encoding/decoding design is the *code rate* which is defined by

$$R = \frac{n}{m}, \tag{2.1}$$

where R is the code rate, n is number of user information bits in the codeword, and m is the total number of bits in the codeword. Essentially, the code rate is defined by the ratio of the number of information pixels to the total number of pixels, and is always less than or equal to one. Each stage of operation adds its own redundancy or recovery information, and thus reduces the overall code rate. The overall code rate is the product of the ECC code rate and the modulation code rate. The key objective is to find an acceptable overall code rate that satisfies both capacity and data integrity requirements.

2.2.3 Modulation Encoding

The main concentration in this thesis is on the modulation detection scheme. While we describe relatively simple modulation-encoding schemes, they can improve the performance of the output data significantly. In this thesis, the encoding is done as follows. As shown in Fig. 2.11, a cluster of pixels on the data page contains nine pixels, and the number of information pixels contained in the cluster depends on the modulation code rate. The encoder allocates enough data pixels from the data stream to fill the information pixels in the cluster and places the data pixels into a two-dimensional configuration. The configuration varies depending on the target code rate and the modulation technique used. Figure 2.11 illustrates some simple data modulation schemes for various code rates.

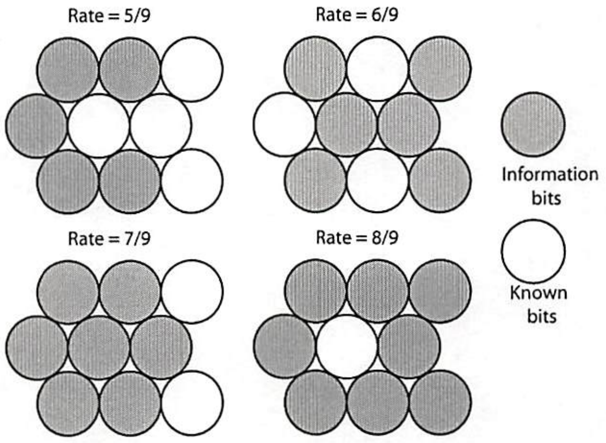
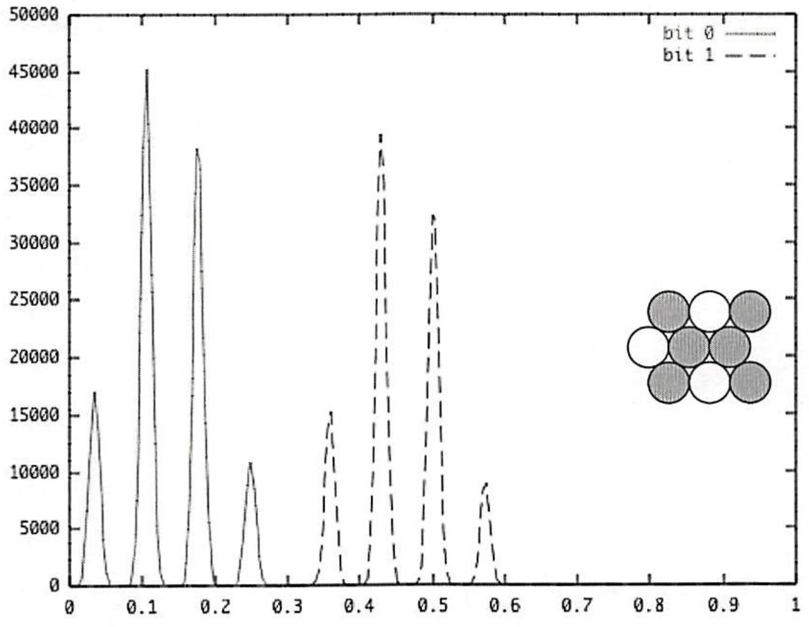


Figure 2.11: Data pixels formation examples.

Figure 2.11 also shows that there are two types of data pixels recorded on the data page: information pixels and known pixels. Information pixels are the user data from the data stream and are shown in dark circles. The known pixels are the additional information used for decoding purposes. These pixels generally contain an analog zero value for use in combating the intersymbol interference, and are shown in white circles.

More efficient modulation encoding can be incorporated into the encoding process to achieve a higher performance. The difficulty in decoding the data arises mainly from the ISI effect. The strength of the interfering ISI is directly proportional to the number of interfering neighbors. Intuitively, a better modulation configuration has a lower average number of interfering neighbors. In Figure 2.12, histograms of noiseless data streams are shown. The two histograms have the same code rate of 6/9, but are encoded with different modulation codes (encoding format). Figure 2.12 shows the two different histograms of pixels encoded with two configurations at the same code rate. The top histogram shows a significant gain in performance over the bottom histogram because *zero* pixels are more uniformly interspersed with information pixels.



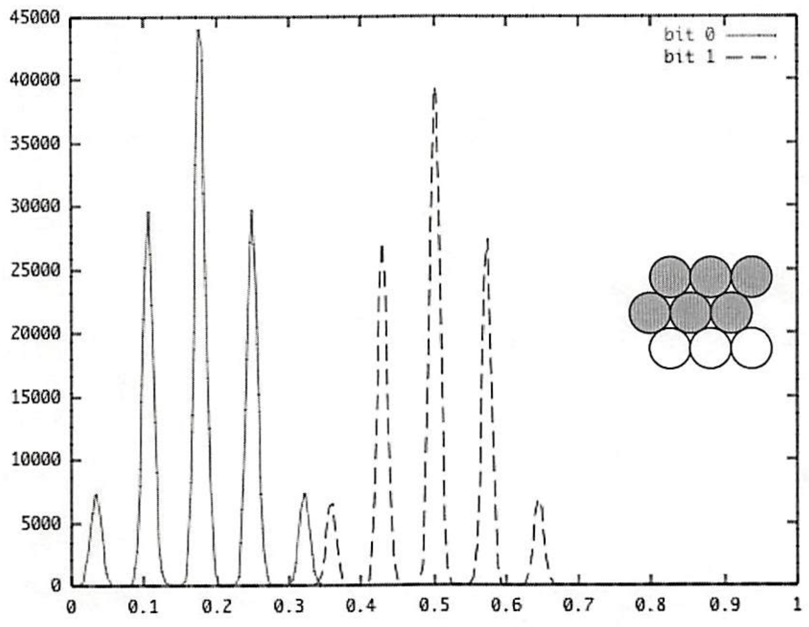


Figure 2.12: Histograms of data streams, Noiseless System, the encoding patterns are shown next to its respective histogram. Top: configuration 1, Bottom: configuration 2.

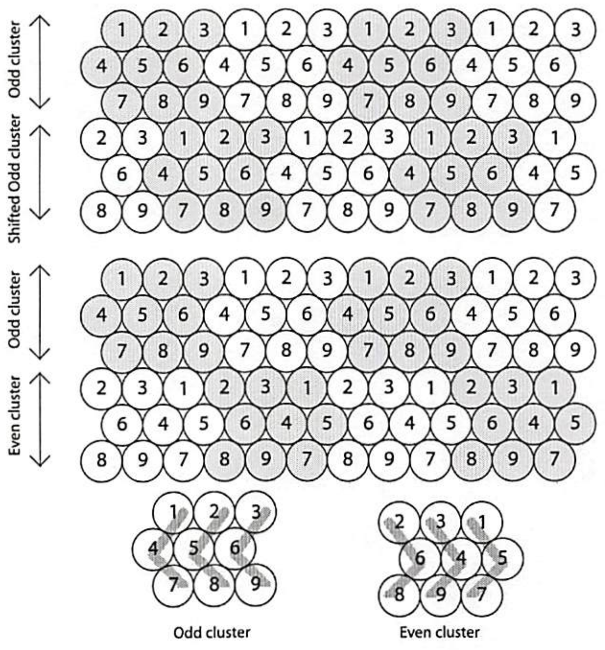


Figure 2.13: Data pixel arrangement: Two ways of placing data pixels on the data page are shown.

Figure 2.13 shows the data pixel arrangement on the data page. Since a cluster is not symmetric, there are two ways of placing the data pixels on the data page as illustrated in Fig. 2.13. Both formations are arranged such that the data pixels with the same numerical label (co-channel) are evenly spaced two-dimensionally which allows future interleaving scheme implementation.

2.2.4 Channel Model

The optical data storage channel can be viewed as a communications channel. The only difference is that in data storage channel, the transmitting and receiving processes do not occur at the same time. The model of the data storage channel can be characterized by the type of channel used and noise type involved. Typically, optical channels possess a low-pass characteristic due to the diffraction limitations of the finite aperture. The finite aperture property causes the light passing through the optical lens to spread out to the surrounding area by convolution with the optical point-spread function instead of concentrating the light on the intended confined area. Thus, the appropriate channel model is an intersymbol interference (ISI) channel. Figure 2.14 shows the simulated illumination pattern of a single isolated pixel *one*.

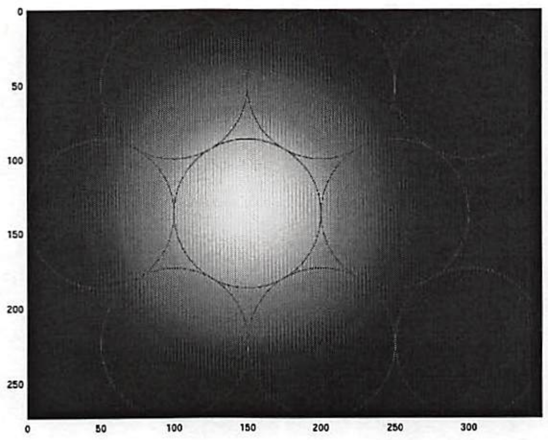


Figure 2.14: Spreading caused by the optical point-spread function, circles define the pixel region.

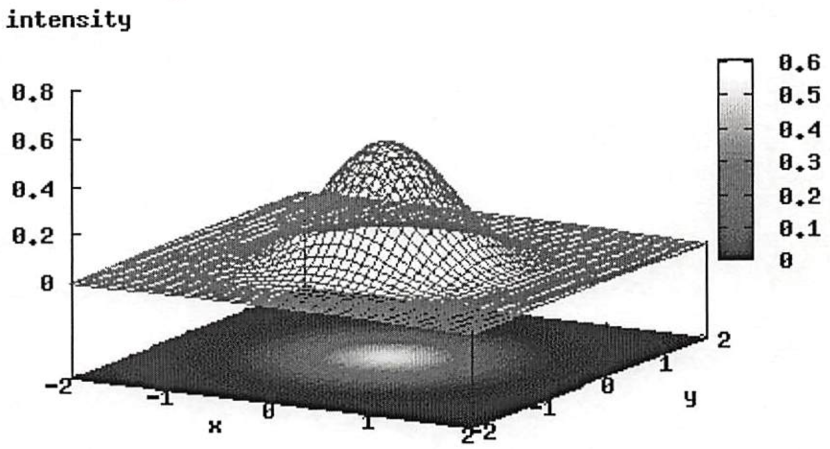


Figure 2.15: 2D profile of point-spread function.

The circles shown in Fig. 2.14 outline the effective borders of each detector (sensor). As shown in Fig. 2.14, significant energy lands on adjacent detectors outside the intended area. This results in the decrease of the signal level in the intended pixel and also in the increase of the interference noise level in the adjacent pixels.

The diffraction effects are modeled as a low pass filter and are described mathematically by

$$h(x,y) = \frac{1}{\pi b_S^2}\left[\frac{J_1(2\pi\sqrt{x^2+y^2}/b_S}{\sqrt{x^2+y^2}/b_S}\right]^2, \tag{2.2}$$

where b_S is a parameter called the blur factor. The point-spread function is characterized by the property of imaging optics used in the system. The blur factor b_S indicates the degree of severity of the inter-symbol interference effect. As the blur factor increases, the extent of the blur becomes higher and less energy is concentrated on the center pixel. The point-spread function described in Eq. (2.2) is adopted because it corresponds to the diffracted-limited impulse response of a circular aperture, and is itself circularly symmetric. The circularly symmetric property gives all signals equidistant relative to the center the same value, and corresponds to the symmetry of the hexagonal coordinate system described, thus simplifying the calculation. Expressing Eq. (2.2) in polar coordinate using the $r^2 = x^2 + y^2$ transformation confirms its circular symmetry. Additionally, this point-spread function is also isotropic, which further simplifies the signal processing.

Other point-spread functions can also be appropriately assigned for different system environments. Possible alternative point-spread functions are *sinc*-based (non-isotropic) and Gaussian-based (isotropic).

2.2.5 Noise Model

In optical systems, the signal may suffer from several unwanted signal contributions that can be stochastic or systematic and lead to erroneous received signals. It is desirable to suppress these unwanted signals as much as possible. In a data storage system, these errors can be classified into two major categories: electronic noise and media noise. Electronic noise includes all interference that arises from system electronics such as quantization noise, ambient noise, etc. Media noise includes environmental variations of the system, and media deformations. In this section, a few of these noise sources are briefly described.

2.2.5.1 Photon Noise

Photon noise is also called shot noise. It is caused by the quantum nature of light signals. The arrival time of the photons at the detector is random with a Poisson distribution. Photon noise is generally insuppressible which also implies that the received intensity does not necessarily represent the true value of the pixel. Photon noise is generally more problematic in systems with low light levels in which the photon count is small, and the small difference in photon count is significant in relative to the received photon count. Conversely, the effect of photon noise is negligible in systems with high light levels.

2.2.5.2 Dark Current Noise

Dark current noise is also known as thermal noise, and is caused by the thermal agitation of electrons. Electrons from the detector can be freed by thermal vibration. They are indistinguishable from the detected photoelectrons and are added to the total photon count. The dark current noise can be thought of as the detector's signal output when no light is present. This type of noise depends on the operating temperature of the detector. There are two effective ways of suppressing dark current noise. The first is by reducing the operating temperature of the detector, and the second is to calculate the dark current average and subtract the average from the detected value.

2.2.5.3 Readout Noise

The readout noise is associated with the detector's amplifier, which translates the photo current generated by the detected photon to a voltage. This type of noise depends on the readout rate (frequency) of the detector and cannot be eliminated but is manageable by adjusting the readout rate to an appropriate rate in which the noise is constant. Its value increases exponentially when the readout rate exceeds the maximum allowable level. This noise is also known as on-chip electronics noise.

2.2.5.4 Quantization Noise

The signal is generally processed in the digital domain rather than in the analog domain. Therefore, analog signals must be converted to their digital representations. The analog value can be thought of as a digital value with infinite number of quantization levels. The round-off error that occurs from the finite number of levels is called quantization noise. Generally, the output of typical analog to digital converter produces an 8, 10, 12 or 16 bit data stream depending on the system configurations. The general signal-to-noise ratio of quantization noise is defined as in [4] by

$$SNR = 6b + 11(dB), \tag{2.3}$$

where b is the number of bits used in the system.

2.2.5.5 Blooming

This effect is the result of the over-saturation of potential well of a detector element. When the potential well is saturated, some of the electrons spill over to the adjacent pixels. Adjusting the exposure time or the addition of anti-blooming gate in the detector can eliminate blooming. An anti-blooming gate detects the saturation and diverts the excess potential without overflowing to the adjacent pixels. Optical data storage systems generally operate in low power mode, and only adequate light for its operation is generated. A common cause of blooming is stray ambient light.

2.2.5.6 Dead Pixels

Dead pixels are the pixel elements on the detector that do not respond to light signals. Dead pixels produce constant values (black or white). This kind of noise varies with the particular sample of detector used in a system. The problem can be easily solved by replacing the defective sensor. In applications such as data storage systems, dead pixels are quite problematic because the data cannot be reconstructed and reconditioned at the receiving end. A more sophisticated solution is to add error correction coding to allow reconstruction of pixel at the receiving end.

2.2.5.7 Media Deformity

The media noise comes from the media degradation such as shrinkage, fabrication imperfections (bubbles), and wear from usage. This type of distortion is unpredictable and the most difficult to suppress. The general solutions for this type of distortion are sturdy construction, design safety factor, and data redundancy.

For simplicity, we model the overall noise as additive white Gaussian noise (AWGN). The noise effects are overcome using a combination of the modulation encoding/decoding, detection algorithm and error correcting code (ECC) encoding/decoding techniques. The probability density function of the noise signal is described as

$$P_N(N) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(N)^2}{2\sigma^2}}, \qquad (2.4)$$

where σ is the standard deviation of the noise sample, and N is the noise signal. A random number generator is used to generate a large number of independent noise samples. The statistical properties of the samples in the limit approach the density of Eq. (2.4).

2.3 Previous Work

There are several previous efforts in signal processing which extend and enhance the performance of data storage systems. They can be divided into three categories: equalization/detection algorithms, modulation encoding/decoding procedures, and error correction coding/decoding procedures.

2.3.1 Equalization/Detection Algorithms

2.3.1.1 Viterbi Decoding Algorithm (VDA)

The Viterbi decoding algorithm is a well known sequence estimation technique described by Andrew Viterbi [9], [32], [33] in 1967. The algorithm is based on a maximum likelihood estimator, and is suited for data sequences with heavy inter-symbol interference (ISI) or signal smearing in several applications of digital communications.

The VDA uses the principle that for a given received sequence of symbols, the particular reconstructed sequence that generates the minimum number of symbol differences is the sequence with the lowest probability of error. There are several possible reconstructed sequences and each sequence can be described as a graph with connected nodes. A sequence tracer or trellis is constructed to simplify the task of identifying the correct sequence.

Figure 2.16 illustrates a trellis diagram containing multiple nodes which are connected by branches. Each branch has branch word (weight). Each branch word is constructed from the encoder state machine and the next branch word depends on two parameters: the encoder input value and the current encoder state. The VDA uses measurement quantities called branch metric and path metric, where branch metric is the difference between the received symbol and branch word. There are many path metrics in the trellis and each path metric is the sum of all branch metrics along a unique path originating from the start of the sequence and ending at the end of the sequence. Another important parameter in VDA system is the number of states in the state machine, which is determined by the constraint length (*K*) or the number of relevant bits used for encoding a symbol. An example of trellis tracing is shown in Fig. 2.16. This VDA algorithm uses 4-state state machine which is shown by the vertical dimension of the graph. The branch metrics are shown on the line connecting between two nodes. The number written on top of each node represents the smallest path metric originating from the start of the sequence and ending at that node. The sequence always starts at state 0, denoted by S0 having the path metric value of zero. The two variables $s(t)$ and $r(t)$ shown in the figure represent transmitted symbols and received symbols respectively. The received symbols are subtracted from the branch weights and the result is added to the path metric of the previous node. At each time instance, the path metrics are compared and the node with the smallest path metric is selected as the next node. The calculation continues until the sequence ends. The thick line shown on the figure represents the path with the smallest probability of error. A longer sequence provides better performance at the expense of higher memory storage requirements.

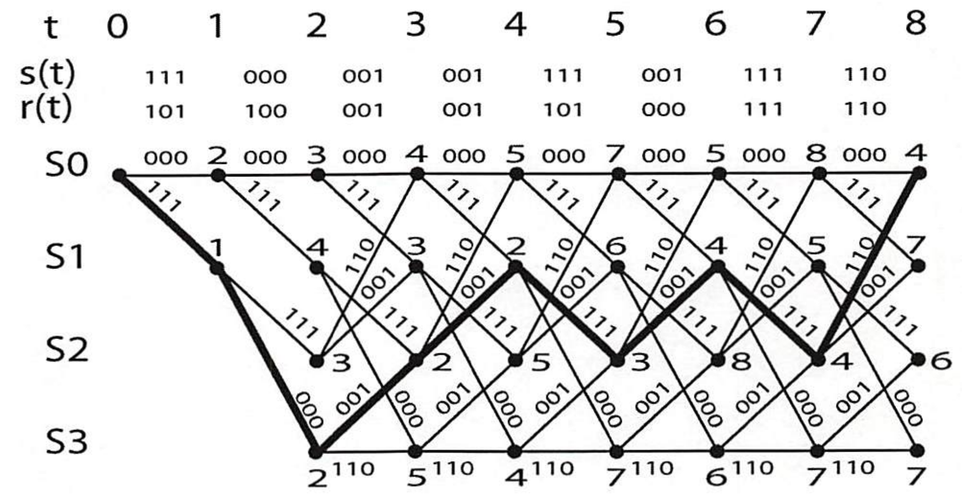


Figure 2.16: Trellis Diagram and example decoding.

2.3.1.2 Partial response maximum likelihood (PRML)

Partial response maximum likelihood [23], [35] is a sequence detection method used mainly in magnetic recording systems and can be viewed as a two step process. The two steps are partial response signaling and maximum likelihood detection. PRML method uses a modified signaling which allows the received signal pulses to be closer and thus the data packing density to be larger. At the receiving end, the receiver receives the partial response pulse sequence which is the input of ML decoder. While there are several kinds of partial response channels, a particular channel called the PR class IV is a commonly used and powerful channel. Figure 2.17 shows the block diagram of the PR4 channel, which produces an output described by

$$y(n) = x(n) - x(n-2) \tag{2.5}$$

for input $x(n)$ and where $x(n-2)$ is an input from delayed by two time units.

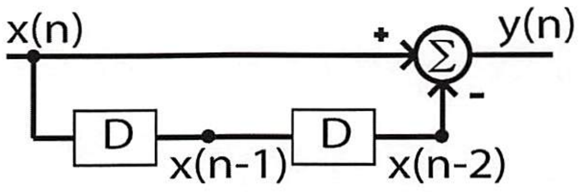


Figure 2.17: Block diagram of PR4 channel.

Figure 2.18 shows the isolated response pulse for the PR4 channel and its sampled data. The purpose of PRML is to allow the adjacent pulses to be closer than the Nyquist spacing, which allows higher packing density. Figure 2.19 shows the superposition of two pulses separated by time delay of T. The superposition is applied to all partial response channels.

The important principle of PRML is that every sampled output at time interval T consists of only three distinct levels -1, 0, and +1, and that inputs can be recovered from current measurements and the previous samples. Equation (2.6) describes the data sample recovery procedure.

$$x(n) = y(n) + x(n-2) \tag{2.6}$$

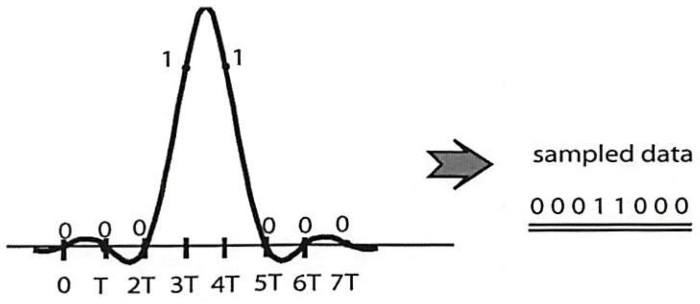


Figure 2.18: Isolated response pulse and its sampled data.

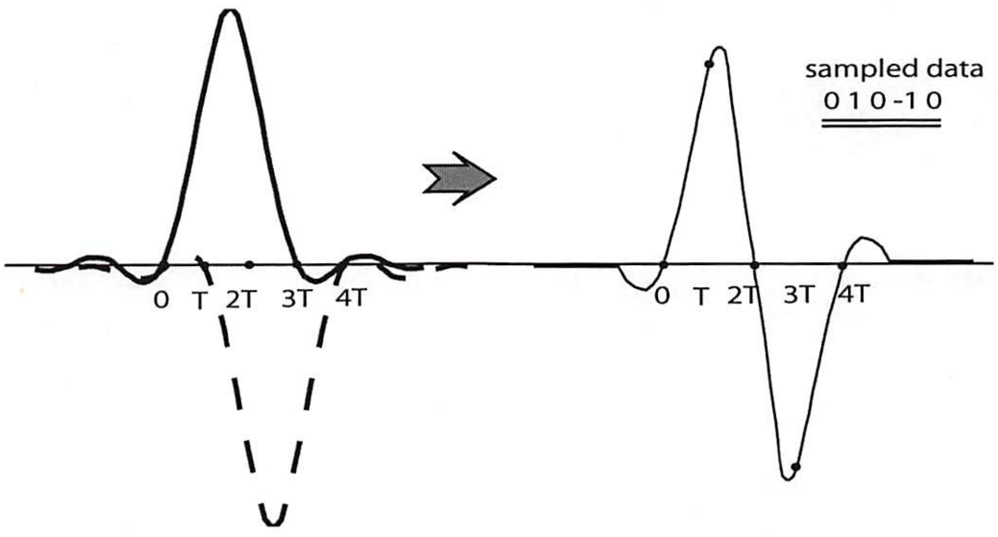


Figure 2.19: Two pulses separated by T and their superimposed pulse shape.

Partial response signaling performs well when the noise level is low. However, in the presence of significant additive noise, it can perform poorly. Maximum likelihood is coupled with partial response signaling to enhance the performance. Several ML algorithms can be used in PRML including the Viterbi decoding algorithm described earlier.

The transfer function of the PR4 signal is a member of the PR polynomial family given by

$$H = (1 - D) \bullet (1 + D)^n \tag{2.7}$$

where D is one unit time delay, and n is the order. PR4 has n value of 1. Higher-order members of PR family have wider response signals that occupy several samples. Higher order polynomials allow higher packing density at the expense of higher complexity in ML decoding.

2.3.2 Modulation Encoding

The two major kinds of interference in data storage are inter-symbol interference (ISI) and system noise. Modulation coding/decoding is specifically used to combat the ISI effect. The ISI effect occurs when the energy from adjacent symbols spread out into the detected signal of interest. The key idea of the modulation coding is to place the symbols in a configuration such that the influences from adjacent symbols become minimal. Modulation becomes particularly more difficult when dealing with higher dimension such as two-dimensional and three-dimensional data.

Modulation codes with code rate of n/m reduce the basic capacity by factor of n/m. However, the modulation coding allows easier decoding and higher bit-error-rate performance. There are several modulation encoding/decoding techniques developed for optical data storage systems. Some classical modulation coding/decoding techniques are described in this section.

2.3.2.1 Frequency Modulation and Modified Frequency Modulation

Frequency modulation (FM) is a digital encoding technique used extensively in magnetic recording. Magnetic recording represents data using magnetic flux reversals. FM changes the frequency of the flux reversal depending on the data bits. The FM encoding procedure uses an encoding table shown in Table 2.1 to convert user data to channel bits. The problem with this encoding is that it uses two channel bits to represent one user bit, which is not efficient.

User data	**Channel bits**	**Percentage of occurrences**
0	RN	50
1	RR	50

Table 2.1: Frequency modulation encoding table, R in channel bit represents a flux reversal in magnetic recording, and N for non-reversal

Modified frequency modulation (MFM) is an improvement on FM. It does not insert a flux reversal at the beginning of a bit sequence. It only inserts flux reversals for symbol *one* and between consecutive symbol *zero*s. MFM significantly increases the information in channel bits without losing the clock signal. Table 2.2 shows the coding of MFM.

User data	Channel bits	Percentage of occurrences
0 (before 0)	RN	25
0 (before 1)	NN	25
1	NR	50

Table 2.2: Modified frequency modulation encoding table

2.3.2.2 Run Length Limited (RLL) Code

Run length limited (RLL) codes are a family of codes that improves results by converting user data patterns into patterns of channel bits with more desirable characteristics. This conversion uses a conversion table known as a codebook. There are 2^m possible code words in the codebook, where m is the number of bits in user data.

The RLL code combines clock information and data and encodes them into a data stream. There are two parameters that define RLL code: the run length, and run limit. The run length and run limit represent the minimum and maximum number of zeros between two closest ones of the adjacent symbols respectively. The number of code words in a RLL code may be significantly lower than 2^m possible code words to preserve the run-limited relationship of the code. There are many RLL codes in the family such as 2/3, 4/5, 2/7, etc, where the first and second numbers represent the run-length and run-limit value respectively. Table 2.3 shows an example of RLL with run-length of 2 and run-limit of 7. The data and channel bits have variable lengths to satisfy the run length limited conditions.

DATA	Channel bits	Percentage of occurrences
10	0100	25
11	1000	25
000	000100	12.5
010	100100	12.5
011	001000	12.5
0010	00100100	6.25
0011	00001000	6.25

Table 2.3: 2/7 RLL code example

2.3.2.3 Modulation Encoding for Page-Oriented Optical Data Storage

Some modulation techniques have been developed specifically for page-oriented optical data storage [3-6], where two-dimensional nature of the data streams is considered. There are two approaches: static encoding and dynamic encoding. Static encoding [3-4] converts data into two-dimensional patterns with pre-determined code words. There are two variations of static encoding: fixed length type, which is a two-dimensional extension of group code; and variable length, which is a two-dimensional extension of RLL code. Static encoding has low encoding/decoding complexity and good performance. The disadvantage of static encoding is its lower code rate because the code is generally designed for worst-case scenario. Dynamic encoding [5-6] encodes data depending on the statistical properties of the data page. Dynamic encoding generally yields good code rates, however at the expense of higher encoding/decoding complexity.

2.3.3 Error Correction Coding/Decoding

The key objective of error correction (ECC) coding/decoding as shown in Fig. 2.9 is to deliver the data message across the corrupted data channel with minimal errors. The ECC is the protection layer on top of the modulation coding which has a higher bit-error-rate requirement. The key method for combating errors is adding data redundancy that simplifies data detection and recovery. Multiple layers of error correcting codes can be applied to a data stream. The data integrity increases with each increasing layer at the expense of overall storage capacity. The block diagram as shown in Fig. 2.20 describes a more detailed description of the ECC encoder shown in Fig. 2.9.

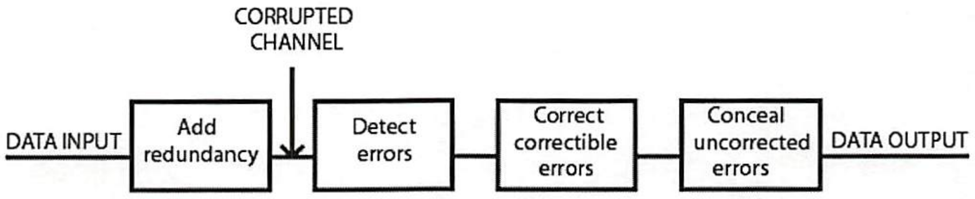


Figure 2.20: Error handling sequence in error correction decoding process.

There are several well-known error-correction procedures used in data storage such as Reed-Solomon (RS) code [30], Turbo code [2], LDPC [18], etc [16], [33], [35], [37]. However, RS is among the most commonly used error correcting codes for data storage applications and specifically for optical data storage due to its flexibility and efficiency to correct burst errors, which is a major distortion in data storage system.

2.3.3.1 Reed Solomon Code

The Reed-Solomon code was invented in 1960 by Irving S. Reed and Gustave Solomon [16], [30], [35], [36]. It is a type of error-correcting code (ECC) used in systems vulnerable to channel noise. Specifically, it belongs to a class of linear block code and is used commonly in optical data storage systems (CD and DVD disc). The Reed-Solomon code has burst-error correction capability, which is suitable for storage media with surface defects. The RS code became popular when Berlekamp [36] invented an efficient decoding algorithm. In this section, we describe a common implementation of the Reed Solomon code in optical data storage.

A specific implementation of RS code in optical data storage such as CD utilizes a combination of RS codes and an interleaver. In a CD system, two layers of Reed Solomon coding are inserted and they are called the inner RS and outer RS. There is an interleaver separating the two layers. This scheme is called cross-interleaved Reed Solomon (CIRC). The key idea for two layers of RS is such that each layer can relax the correction ability constraint, and the interleaver is inserted to alleviate and decimate the continuity of the burst errors. A diagram of typical CD ECC implementation is illustrated in Fig 2.21. The inner RS (also called C2 level) has the code rate of 24/28 and is capable of correcting 2 errors, while the outer RS (C2) has the code rate of 28/32 and is also capable of correcting 2 errors. Both the inner and outer RS have limited error correction abilities, however the interleaver, which redistributes the symbols around the data blocks (specifically 109 blocks), helps minimize the length of the burst errors. Without the interleaver, a single RS unit must have large error correction abilities, which requires high redundancy.

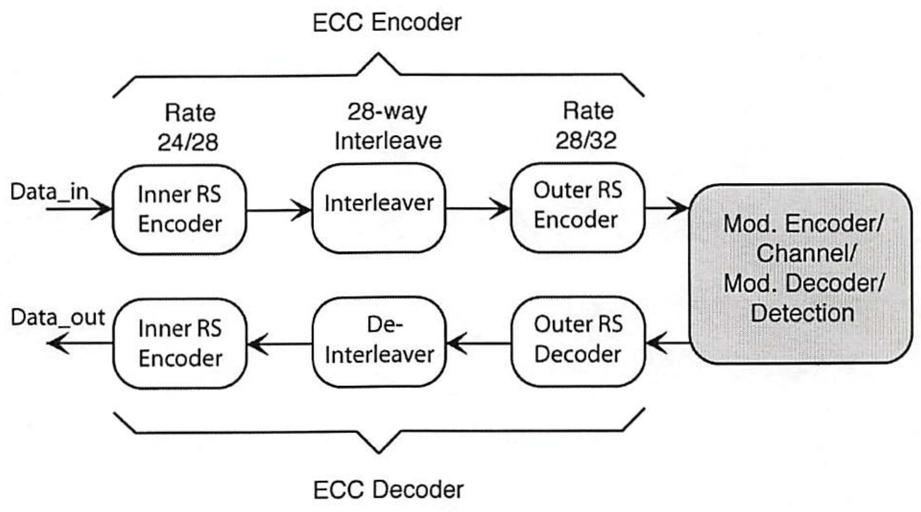


Figure 2.21: Typical ECC in compact disc (CD).

2.3.3.2 Reed Solomon Product Code (RSPC)

The Reed Solomon Product Code (RSPC) is used primarily in DVD recording. It considers the data to be encoded as a two-dimensional block. Two RS codes are applied to the data, one which encodes row data and another which encodes column data, hence the term product code. A specific ECC format of DVD media is illustrated in Fig. 2.22. The RSPC performs encoding on a block of data, where one block of data contains sixteen data sectors and one data sector contains twelve rows of 172 data bytes. The final dimension of a data block is 192 bytes by 172 bytes. After the ECC encoding, calculation on PO bytes (row parity) is performed, and followed by a calculation on PI bytes (column parity). The encoded data has dimension of 208 bytes by 182 bytes. In addition, the PO row is interleaved evenly in the vertical direction. This format combines 12 rows of data with 1 row of PO as illustrated in Fig. 2.22. The ECC code used in RSPC has vertical and horizontal code descriptions as follows: RS(208,192,17) and RS(182,172,11) respectively. This product code allows the correction of 8 error bytes and 5 error bytes in column and row respectively.

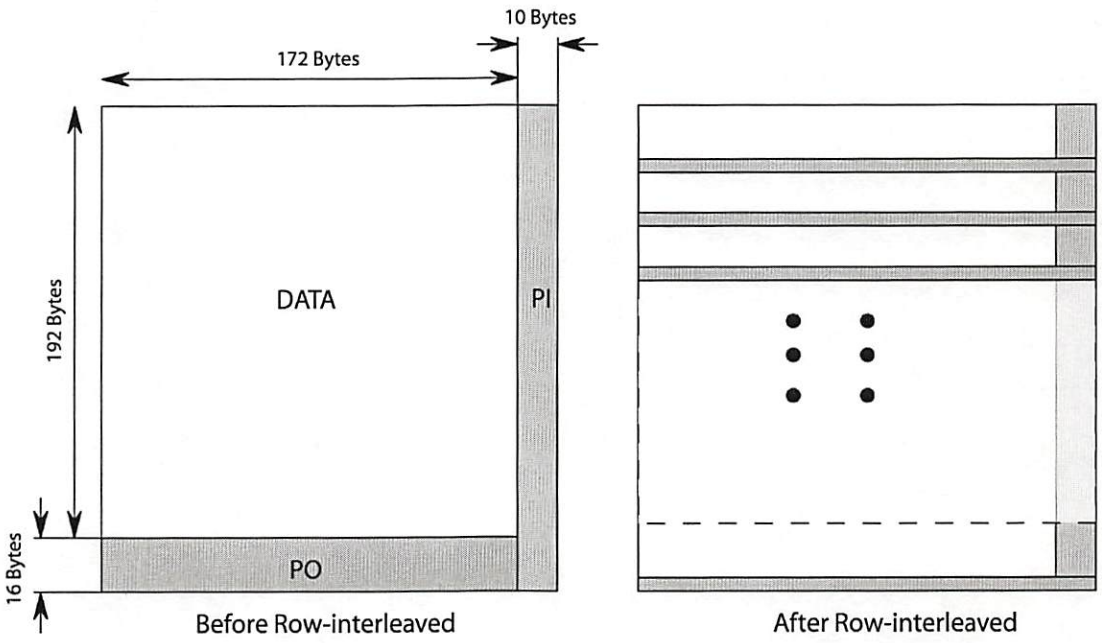


Figure 2.22: Typical ECC in digital versatile disc (DVD).

Chapter 3

Iterative Detection Techniques for PODS

In this chapter, we describe an iterative technique for detection of binary pixel values in a two-dimensional PODS system. The technique incorporates decision feedback procedure to iteratively refine the estimated pixel values based on the interactions with its respective estimated neighbor pixels.

3.1 Algorithm Assumptions

Our proposed technique aims at accurately detecting the pixels that are difficult to recover because of intersymbol interference (ISI), interpage interference (IPI) and noise using local information gathered from neighboring pixels. These data interference significantly corrupt and degrade the signals read from the storage medium. The technique produces acceptable bit-error-rate (BER) performance results with lower computational complexity compared to other techniques. During the algorithm design process, we make the following experimental assumptions, which provide reasonable operating guidelines for the algorithm:

1) The two-dimensional point-spread function (PSF) or transfer function of the read-back signal from an isolated pixel is known and well-defined. In other words, the shape and intensity of the read-back signal due to known stored pixels can be predicted exactly.

2) The optical system is assumed to be spatially incoherent and linear, thus superposition of the intensity signals from center and neighbor pixels at the detector is linear.

3) The locations of the information pixels are known in the multidimensional media at both the transmitting (recording) end and receiving (detector) end.

4) The system assumes perfect registration of the sensor array with the corresponding data pixels. The center of the individual sensor is perfectly aligned to the center of the intended data pixel on the data page.

5) The binary valued data pattern is recorded using a modulation code whose analog representation is known.

In subsequent chapters, we relax some of these assumptions to adapt to more complex scenarios, but we begin here with a description of the basic algorithm.

3.2 Algorithm Description

The detection algorithm described in this thesis employs a maximum likelihood (ML-like) approach, and *a priori* knowledge of known pixels at predefined locations to derive the most probable data pixel values. The algorithm consists of a sequence of computation processes that identify the most probable value of the data pixels.

The complete block diagram of our algorithm is shown in Fig 3.1. The algorithm consists of two sequential operating stages: classification and verification. Data pixels are processed iteratively through the algorithm. Due to its iterative nature, the outputs from the first and subsequent iterations are expected to have some decoding errors. The algorithm retains the results and confidence information obtained during the previous iterations and feeds them back as the additional input for the next iteration. As the number of iterations grows, more confidence information is collected and better estimation is carried out leading to gradual improvement in performance.

The analog detector output is quantized to an 8-bit binary number to generate digital input to the algorithm denoted by $r(x,y)$ in Fig. 3.1, where x and y represent horizontal and vertical indices of the data pixel on the page respectively. All subsequent calculations are performed using 8-bit finite precision arithmetic. The 8-bit finite precision is used in consideration of reduced hardware requirement and it conforms to common general-purpose DSP processor input. The number of quantization levels could be increased at the expense of increased hardware processor complexity and power consumption. However, we have found experimentally that an increase in the number of quantization levels has little or no effect on the overall BER. The output of the algorithm $d(x,y)$ is a digital representation of the detected binary value at each pixel. In the next sections, we describe each stage of the algorithm in detail.

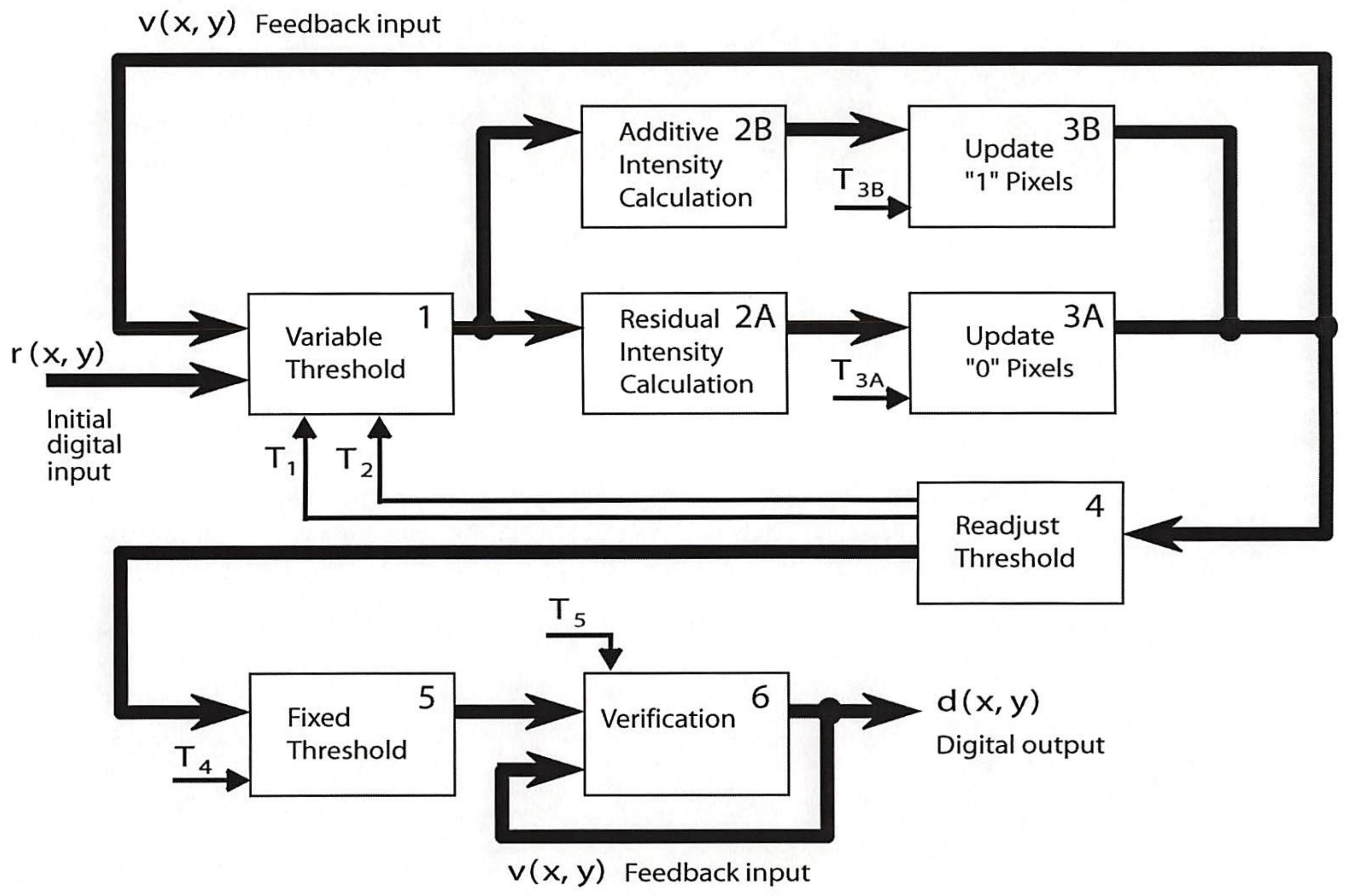


Figure 3.1: Block diagram of iterative detection algorithm: Two stages: Classification (blocks 1 through 5); Verification (block 6).

3.2.1 Algorithm Parameters

3.2.1.1 Data Pixel Classes for Binary Data

We begin by describing the algorithm for binary (two-level) data storage. The algorithm is also extendable to handle non-binary signals. The goal of the algorithm is to differentiate and classify measured data pixels as either binary *zero* or *one*. The algorithm operates over several iterations in which some data pixels remain unclassified until they can be confidently classified. During the operations, the data pixels are internally classified into three distinct classes: $R1$, $R2$ and $R3$, representing *zero* pixels, pixels *one* pixels, and unclassified pixels respectively.

3.2.1.2 Operation Parameters

Two additional internal parameters are also defined for calculations and book keeping purposes. The first parameter is the data pixel value denoted by $v(x, y)$. It records and updates the classified value of the data pixels at various steps of the algorithm. The data pixels belonging to classes $R1$, $R2$ and $R3$ have the $v(x, y)$ value of *zero*, *one* and $r(x, y)$ respectively, where $r(x, y)$ denotes the value of the initial quantized input value.

The second parameter is the pixel validity denoted by $valid(x, y)$, defined mainly for book keeping purposes. The validity is set to one if the corresponding data pixel has been classified. The data pixels belonging to class $R1$ or $R2$ are classified and considered valid and their $valid(x, y)$ values are assigned to one. The remaining unclassified data pixels (class $R3$) have $valid(x, y)$ value of *zero*. Initially, all data pixels are unclassified and belong to class $R3$, where $v(x, y)$ are set to $r(x, y)$, and $valid(x, y)$ to *zero*. Figure 3.2 shows the class and parameters assignment described in this section.

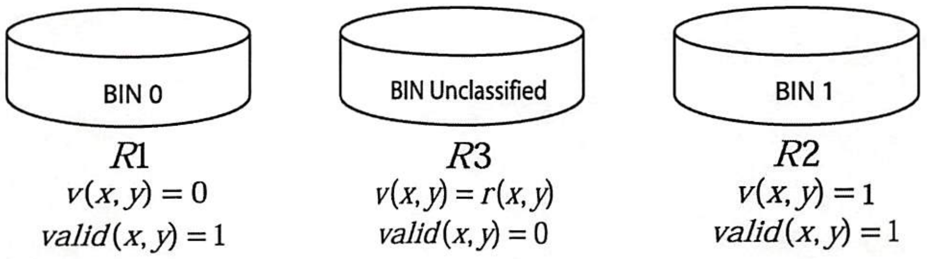


Figure 3.2: Class and parameters assignment.

3.2.2 Algorithm Setup

3.2.2.1 Point-Spread Function Lookup Table Setup

The readout system point-spread function (PSF) is a crucial part of the calculation which depends on the physical characteristics of the readout optical system, and is assumed fixed and known. For computational efficiency, we calculate the PSF coefficients offline during the calibration process and store them in a lookup table. In this thesis, we concentrate on optical systems having a circularly symmetric Jinc PSF described in Eq. (2.2). However, alternative choices of detector arrays (PSF) can be used in the algorithm.

Figure 3.3 shows a plot of Jinc point-spread functions for different b_S values, where b_S is a blur factor parameter. Figure 3.3 also shows the influence of parameter b_S on the distribution of point-spread function energy: as parameter b_S increases, the energy pattern becomes more disperse. In Fig. 3.3, the curves show the cross-section of the PSF at the axis plane. However, the specific PSF is circularly symmetric and these curves also represent any cross-section views of the planes passing through the origin.

Ideally, the data pixel energy is bounded within its corresponding detector cell. In reality, the smearing effect causes the energy to fall outside of the intended region. Table 3.1 indicates the percentage of energy contained within the intended center pixel, its first level neighbors, and its second level neighbors, where one hundred means that all the energy is detected by the center pixel. The neighbor pixels have comparatively lower energy levels relative to their center pixel but the sum of energy from all neighbors is significant and drastically affects the detection process. The PSF at various blur parameters have different zero crossing locations and the measured energy values may also vary depending on these locations. The PSF table setup procedure is performed during the system calibration process.

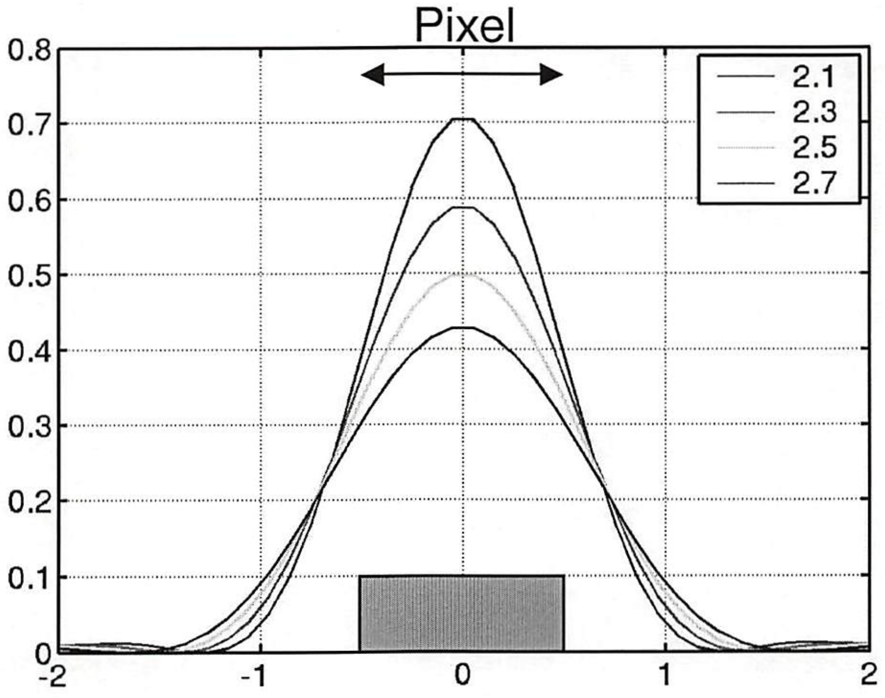


Figure 3.3: Jinc point-spread function profile with various values of blur factor parameter.

Blur factor	Center pixel (%)	1st neighbor[1] (%) per neighbor	2nd neighbor[2] (%) per neighbor
1.64[3]	58.84	3.51	0.25
2.1	42.50	5.59	0.46
2.3	37.06	6.46	0.51
2.5	32.48	7.18	0.47
2.7	28.62	7.71	0.37

Table 3.1: Concentration of signal strength in center pixel and its neighbors due to the Jinc point-spread function.

[1] 1st neighbor: the pixel adjacent to the center pixel (based on a hexagonal grid with 6 neighbors)
[2] 2nd neighbor: the pixel adjacent to the 1st neighbor
[3] $b_S \approx 1.64$ is the Rayleigh limit equivalent, which represents the closest distance two points can be distinctly resolved. This Rayleigh limit is specific to normalized Jinc point spread function.

3.2.2.2 Known Data Pixel Decoding

Certain data pixels with known *zero* or *one* values are inserted as part of a data page, and their locations are known in advance for a particular system and page format. The *a priori* knowledge contained in these pixels helps in decoding data corrupted by ISI effect. These pixels are inserted during the encoding process to ensure ISI protection. Additionally, they can be used to derive noise characteristics. Using assumption (3) stated in section 3.1, we assume that the decoder knows the locations of both information data and *a priori* data. The *a priori* knowledge arrives at the receiver in the form of known values (generally *zero*) at specific known locations on the page. Thus, the known data pixels can be efficiently separated from the data page and readily decoded with no errors.

For most communication channels, the operating environment is often unpredictable and the task of an equalizer/detector is to counteract the corruption caused by the channel. However, the distortion may be time-varying and is often difficult to recover. By attaching the known data pixels to the data stream, the need to recover these pixels that could otherwise be corrupted is eliminated. These pixels also reveal the noisy channel characteristic. The benefit of the known data pixels is two-fold in this algorithm. They not only reveal the channel characteristics but are also used in the decoding algorithm as additional information. The number of known data pixels contained in the data stream determines the data stream code rate, R as defined in

Eq. (2.1). Lower code rates contain a higher percentage of known pixels. Higher code rates provide higher effective data storage capacity. Our proposed algorithm takes advantage of local information and thus benefits greatly from lower code rates at the expense of lower effective storage capacity.

In this thesis, we use a hexagonal grid arrangement as described in section 2.2.1. The fundamental coding cluster consists of nine data pixels arranged as shown in Fig. 2.11. In a known data pattern, a portion of the coding cluster (N symbols) are immediately classified as a pre-assigned value (*zero* is preferably used as the pre-assigned value to reduce the unnecessary inter-symbol interference effect), where N is the number of known data pixels in the cluster. Figure 3.4 shows the percentage of pixels whose value is known *a priori* for different code rates. In this thesis, we adopt a simple coding cluster assumed in Fig 2.11 containing nine pixels. The code rate can be further improved by a more efficient modulating coding cluster (using more pixels covering larger neighborhood).

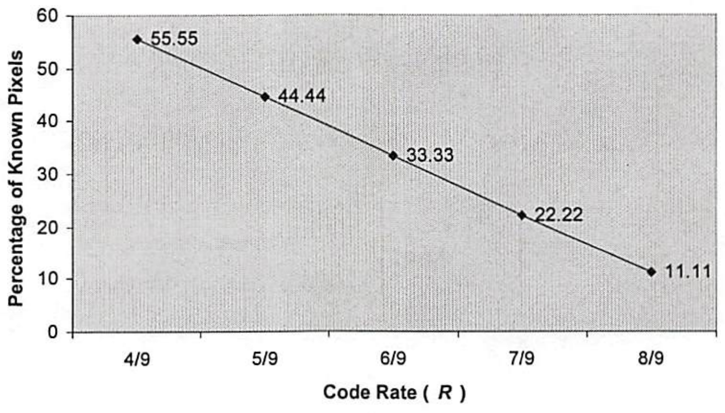


Figure 3.4: Percentage of known pixels for various values of code rate R.

In this section (section 3.2.2), we have discussed some setup procedures involving our proposed algorithm. These setup procedures are performed on every data page, and require a small amount of setup time. They can be performed before or inside the classification stage described in the next section.

3.2.3 Algorithm Stages

Referring to Fig. 3.1, the algorithm begins with the noisy readout intensity signal from the physical storage medium, denoted by $r(x, y)$. The algorithm is essentially composed of two sequential stages as follows:

1) Classification stage (blocks 1 through 5 in Fig. 3.1): performing iterative classification of the data pixels using successive calculation steps to determine the most likely candidate.
2) Verification stage (block 6 in Fig. 3.1):: performing iterative verification to check for consistency of the classified data pixels

3.2.3.1 Classification Stage

The classification stage consists of blocks 1 through 5 illustrated in Fig. 3.1. The objective of this stage is to attempt to classify as many data pixels as possible within a given number of iterations (time limited) using *a priori* knowledge (known data pixels) and additional confidence information collected during the calculation iterations. The resulting output is passed to the verification stage at the end of the classification stage. The classification stage consists of successive low complexity calculations and testing rules to determine the most likely symbol candidate for each data pixel and is divided into the following five calculation steps: variable threshold step, residual intensity calculation step, update step, threshold readjustment step and fixed threshold step. The next several sections will describe each procedure in detail.

3.2.3.1.1 Variable Threshold Step

The first step of the classification stage is the variable threshold step. It performs threshold detection using two threshold values: upper and lower threshold. The data pixels are classified into three separate bins (classes) described in the previous section. All measured data pixels $r(x,y)$ are classified into one of the following bins: bin 0, bin 1, and an unclassified bin using the decision rules described by

$$r(x,y) \in \begin{cases} bin\,0 & if \quad r(x,y) < T_1^i \\ unclassified\ bin & if \quad T_1^i \le r(x,y) \le T_2^i \\ bin\,1 & if \quad r(x,y) > T_2^i \end{cases}, \qquad (3.1)$$

where, T_1^i is the lower threshold, T_2^i is the lower threshold, and superscript *i* denotes the iteration number.

The objective of this calculation is to spend the least amount of computing power to classify some portions of data pixels. Unlike normal threshold detection, the procedure does not classify every data pixel; rather it only attempts to classify data pixels that have a high probability of being correct. Referring to the typical histogram of measured *zero* and *one* pixels with ISI as shown in Fig 2.12, the data pixels are considered to have a high probability of being correct if their value falls in a region of the histograms that is a considerable distance from the overlap region of the *zero* and *one* histograms. In our algorithm, this classification step is repeated every iteration with updated thresholds.

Figure 3.5 shows a representative sample histogram of the received intensity. The distribution of levels corresponding to binary *zero* and binary *one* are the curves on the left and right respectively. Figure 3.5 also shows two vertical lines representing two decision threshold levels. The lower and upper decision threshold levels are shown on the left and right respectively, and are denoted by T_1^i and T_2^i respectively where the superscript indicates the iteration index. The initializations of these two parameters are described in the following subsection.

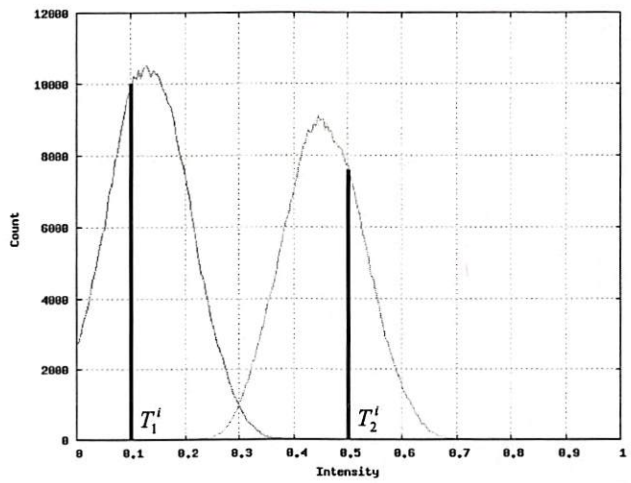


Figure 3.5: Histogram of received intensity and the decision threshold levels.

3.2.3.1.1A Threshold Parameter Initialization

On the first iteration, the threshold values are initialized to set the starting points for the latter iterations. From Fig. 3.5, the threshold values can be arbitrarily initialized as long as they reside in a region having a high probability of being correct. The goal is to assign the values of T_1^0 and T_2^0 such that when the threshold step is performed, it is certain that any data pixels that is classified to R_1 or R_2 are correctly classified. The initialization of these two parameters can be done in two different approaches: by calculation or by system calibration, which are described in detail in the next sections.

3.2.3.1.1B Calculation Approach

The first approach considers two scenarios, one for each symbol. We first describe parameter T_1^0 setup. From Fig. 3.5, we want the T_1^0 parameter to be sufficiently less than the lowest value of pixel *one*. We assume that the noisy channel is modeled by AWGN described by Eq. (2.4), and the noise tail is infinitely long. We must choose an acceptable error probability denoted by $P(e)$. Optimistically, we want to find an error probability that allows less than one error pixel out of one data page, although it can be chosen arbitrarily. Thus, the lowest value of pixel *one* is approximately defined by the isolated intensity of pixel *one* subtracted by the maximum destructive noise. The setup derivation is described by

$$|Z_{\max}| > \mathrm{erf}^{-1}(1 - P(e)) \text{ and} \tag{3.2}$$

$$T_1^0 \le E_1 - |Z_{\max}|\sigma_n, \tag{3.3}$$

where $Z_{\max}$ denotes the maximum multiple of noise standard deviation, and E_1 denotes the isolated pixel *one* intensity.

In Fig. 3.6, an example is shown, where an acceptable error probability is selected and the corresponding parameter $Z_{\max}$ is calculated.

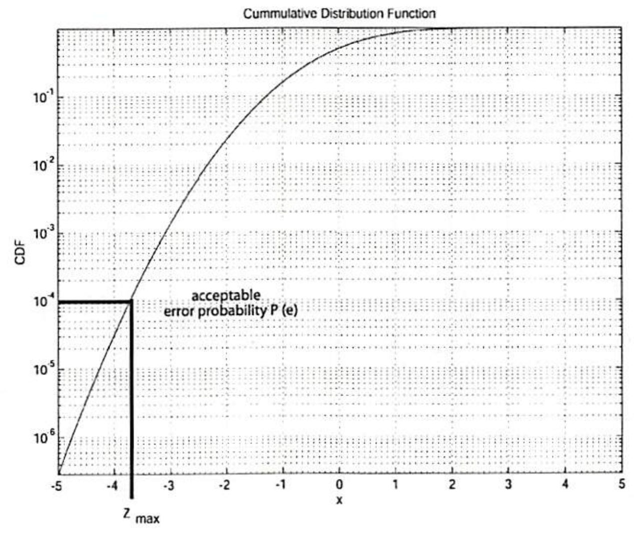


Figure 3.6: Parameter $Z_{\max}$ selection example: the parameter is selected from the cumulative distribution function of zero-mean, $\sigma_n = 1$ noise variable using a corresponding error probability.

Similarly, the parameter T_2^0 needs to be sufficiently greater than the highest value of pixel *zero* and specified by

$$T_2^0 \ge E_{\text{max neighbor ISI}} + |Z_{\max}|\sigma_n \text{ and} \tag{3.4}$$

$$E_{\text{max neighbor ISI}} = \begin{pmatrix}\text{number of neighbor pixels} \\ \text{who are information pixels}\end{pmatrix} * h(x_0, y_0), \qquad (3.5)$$

where $E_{\text{max neighbor ISI}}$ is the maximum ISI generated from the neighbor pixels containing information pixels and $h(x_0, y_0)$ is the readout PSF coefficient of pixels immediately adjacent to the center pixel. The number of information pixels surrounding the center pixel depends on the code rate and the modulation encoding configuration, and the $h(x_0, y_0)$ coefficient is consistent for all neighbor pixels due to circular symmetric PSF used. A higher code rates contain more information neighboring pixels leading to higher value of $E_{\text{max neighbor ISI}}$.

3.2.3.1.1C Calibration Approach

The second approach uses a simpler initialization method. During the calibration process, we randomly generate data to fill multiple data pages. The generated data pages convolve with known PSF and the output signal is collected to construct two histograms corresponding to original pixel values of *zero* and *one*. The two histograms are superimposed and the location of the overlap region R_3 can be identified. The intensity range of R_3 extends from an intensity spectrum that has very low pixel *one* probability of occurrence to an intensity spectrum that has very low pixel *zero* probability of occurrence. The value of probability of occurrence can be chosen arbitrarily. Using the knowledge of R_3, we initialize the threshold values

T_1^0 and T_2^0 such that the threshold levels are not within the overlap region boundaries to ensure the accuracy of the classification.

3.2.3.1.2 Residual Intensity Calculation and Pixel zero Update

The output from the last step provides inputs to two parallel branches labeled blocks 2A and 2B, and blocks 3A and 3B in Fig. 3.1. Here, the algorithm intends to update two symbols, pixel *zero* and pixel *one*. While the update procedures can be carried out either in cascade or in parallel, it is more efficient to do them in parallel. In this section, we describe the procedure necessary for updating pixel *zero*, and the following section describes the complement procedure necessary for updating pixel *one*. There are two major types of pixels occupying the unclassified region R_3: pixels of value *zero* who have a majority of neighbor pixels of value *one* and pixels of value *one* who have a majority of neighbor pixels of value *zero*. In this section, we emphasize processing of the first type, and in the next section, we emphasize the latter.

3.2.3.1.2A Residual Intensity Calculation

This step is a key part of the algorithm, in which the algorithm tries to estimate the uncorrupted intensity value of the unclassified data pixels by subtracting out the ISI effects of the previously classified neighboring data pixels. The calculation is described by

$$\tilde{r}_r(xy) = r(x,y) - \sum_{\substack{(x_k,y_k)\in \\ \text{neighbor pixels}}} h(x_k - x, y_k - y)\cdot v(x_k,y_k)\cdot valid(x_k,y_k), \qquad (3.6)$$

where $\tilde{r}_r(x,y)$ is a new variable called the residual intensity, $r(x,y)$ is the received intensity, $h(x,y)$ is the system point-spread function, $v(x_k,y_k)$ and $valid(x_k,y_k)$ are the classified value and pixel validity of the neighbor pixel respectively.

The convolution in Eq. (3.6) computes the ISI contribution to the pixel at coordinate (x,y) by convolving the system PSF with each of the nearest neighbor pixels that are classified with high accuracy as a binary *one*. This contribution is summed over the neighbors and subtracted from the received intensity to produce the residual intensity.

Equation (3.6) indicates that only the data pixels belonging to R_1 and R_2 (those previously classified with high confidence in earlier iteration) are used in the calculation. To simplify and reduce the redundancy in the calculation, the point-spread function is pre-computed and stored in the lookup table as described in the algorithm setup section. Figure 3.7 describes graphically the effect of the residual intensity calculation step where it shows an unclassified center pixel surrounded by a number of previously classified neighboring pixels and a few unclassified neighboring pixels. The ISI effect originating from previously classified neighboring pixels can be effectively removed from the received intensity as illustrated by the outward arrows. The ideal scenario occurs when all neighboring pixels are correctly classified and their ISI contributions removed, in which case the residual intensity represents the sum of the intensity from center pixels and noise interference. However, some neighboring pixels usually remain unclassified, and the residual intensity represents the sum of the intensity from center pixels, noise interference and ISI contributions from unclassified neighboring pixels. Due to the decision feedback procedure, the number of unclassified pixels on subsequent iterations is less than or equal to the number of unclassified pixels from its previous iteration, which allows the residual intensity to approach its true value as algorithm progresses. The residual intensity from each iteration provides the input value necessary for update procedure on the next section.

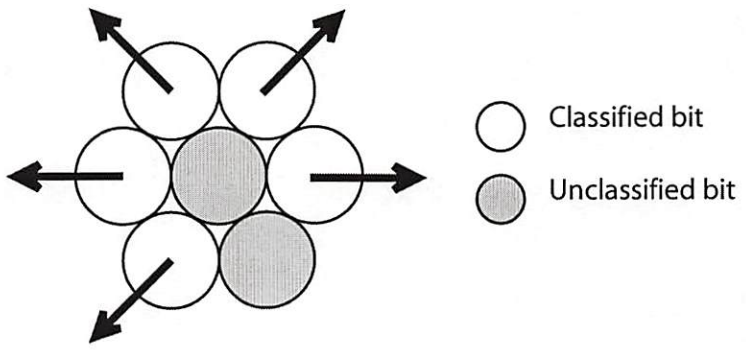


Figure 3.7: Residual intensity calculation procedure.

3.2.3.1.2B Pixel zero Update

The output from the previous step (residual intensity) denoted by $\tilde{r}(x,y)$ is processed through an updating procedure. This procedure updates the unclassified data pixels using the update rule described by

$$\begin{array}{ll} \text{if } \tilde{r}_r(x,y) < T_{3A} & \text{, pixel is assigned to } R_1 \\ \text{if } \tilde{r}_r(x,y) \geq T_{3A} & \text{, pixel is assigned to } R_3 \end{array}, \tag{3.7}$$

where T_{3A} is the pixel *zero* update sensitivity. The pixel *zero* update sensitivity value is set to be less than intensity level of an isolated pixel *one* with an additional margin for the effects of additive noise.

The purpose of the pixel update step is to verify whether the output from the residual intensity can be unambiguously identified as a number of a class (in this particular case class *zero*). The procedure follows a simple logic that if the true value of center data pixel is *zero* and the received intensity falls into the overlap region R_3, then the received intensity of the data pixels results mainly from the ISI contribution of the neighbor data pixels. The data pixel with true value of *zero* is mistakenly detected as pixel *one* if it has significant number of neighbor data pixels with value of *one*. Therefore when the residual intensity falls significantly below the threshold T_{3A}, it is certain that the data pixel is a pixel *zero*. The data pixel is then placed in bin 0, $v(x,y)$ is set to 0, and $valid(x,y)$ to 1. When the update is successful, it means we have classified a sufficient number of neighbors around the center data pixel to be certain that it has a true value of *zero*.

From the description of the procedure, it can be seen that *one* pixels are not updated in this step. A parallel update procedure which updates pixel *one* is described in the following section. Any data pixels that fail the update test fall into one of two categories: 1) a pixel *zero* with many unclassified neighbors, 2) a pixel *one* with a small number of unclassified neighbors. The remaining pixels that fail to pass the update test remain in the unclassified bin. Figure 3.8 summarizes the operation of pixel update step.

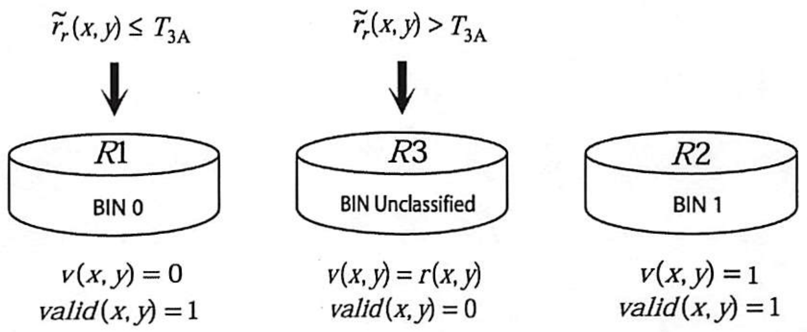


Figure 3.8: Pixel *zero* update process.

An example of two-step update procedure involving residual intensity calculation and pixel *zero* update procedure is illustrated in Fig. 3.9. Figure 3.9 shows the overlap region and two types of clusters occupying this region. The solid arrows leading from each clusters point to their respective received intensity value. The residual intensity calculation effectively removes the ISI contributions from neighbor pixels, and the amount being removed depends on the number of previously classified neighbor pixels surrounding each center pixels with value of *one*. From Fig. 3.9, the top cluster has one classified neighbor pixel with value of *one*, while the bottom cluster has three. The dashed line arrows indicate the residual intensities of both clusters after the ISI contributions have been removed. The bottom cluster crosses the threshold value and the center pixel is classified as pixel *zero*, while the top cluster fails the update test and remains unclassified.

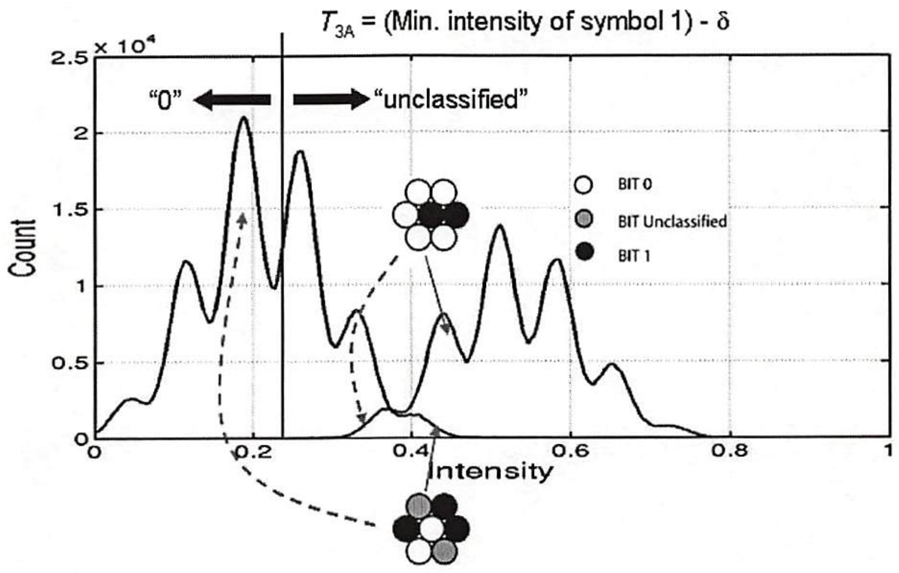


Figure 3.9: Residual intensity calculation and Pixel zero update process example diagram.

3.2.3.1.3 Additive Intensity Calculation and Pixel one Update

In this section, we emphasize on the processing of data pixels in the unclassified region R_3 that are *one* pixels with majority of neighbor pixels being *zero* pixels. The procedure is complementary to the residual intensity calculation procedure described in the previous section. It updates the unclassified *one* pixels using information from the variable threshold step.

3.2.3.1.3A Added Intensity Calculation

Complementary to the residual intensity calculation, the added intensity calculation effectively emphasizes and adds the effects of surrounding neighbor *zero* pixels which are previously ignored in the residual intensity calculation. This step requires an additional data pixel mapping as described by

$$v_a(x,y) = \begin{cases} 0 & \text{if the bit is classified } \textit{information} \text{ bit with } v(x,y)=0 \text{ or } \textit{unclassified} \\ 1 & \text{if the bit is classified } \textit{information} \text{ bit with } v(x,y)=1 \end{cases}, \quad (3.8)$$

where $v_a(x,y)$ denotes the mapped pixel value. In this mapping, the unclassified pixels are mapped to pixel *zero*, and the classified information pixels are mapped to their binary complement i.e. a pixel *zero* is mapped to *one* and vice-versa. Figure 3.10 shows simple mapping examples.

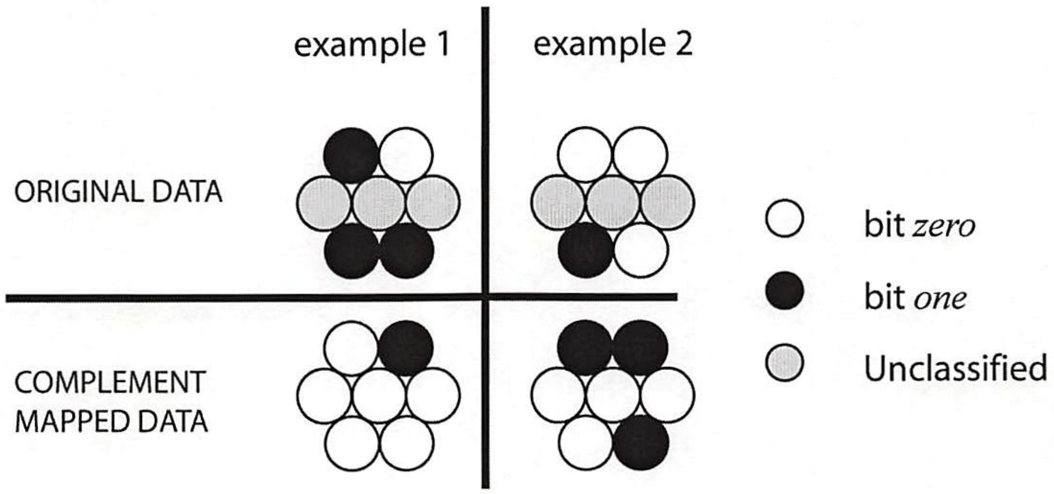


Figure 3.10: Added intensity calculation procedure example.

Next, a new variable called added intensity denoted by $\widetilde{r}_a(x,y)$ is calculated by convolving the newly mapped data with the point-spread function and adding the original received intensity as described by

$$\widetilde{r}_a(x,y) = r(x,y) + \sum_{\substack{(x_k,y_k)\in \\ \text{neighbor pixels}}} h(x_k - x, y_k - y)\cdot v_a(x_k,y_k)\cdot valid(x_k,y_k). \quad (3.9)$$

The convolution in Eq. (3.9) computes the complement ISI contribution to the pixel at coordinate (x,y) by convolving the system PSF with the mapped values of each of the classified nearest neighbor pixels. This contribution is summed over the neighbors and added to the received intensity to produce the added intensity. It turns out that the added intensity is bounded by the maximum received intensity of each respective symbol.

3.2.3.1.3B Pixel one Update

The updating of pixel *one* adopts a similar logic to the pixel *zero* update. The added intensity of pixel *zero* is known to be bounded by the maximum received intensity of pixel *zero*, therefore any pixels whose added intensities exceed this value can be correctly classified as pixel *one*. This procedure updates the unclassified data pixels using the update rule described by

$$\begin{array}{ll} \text{if } \widetilde{r}_a(x,y) > T_{3B} & \text{, pixel is assigned to } R_2 \\ \text{if } \widetilde{r}_a(x,y) \le T_{3B} & \text{, pixel is assigned to } R_3 \end{array}, \quad (3.10)$$

where T_{3B} is the pixel *one* update sensitivity. The pixel *one* update sensitivity value is set to be more than intensity level of the maximum ISI effect with an additional margin for the effects of additive noise.

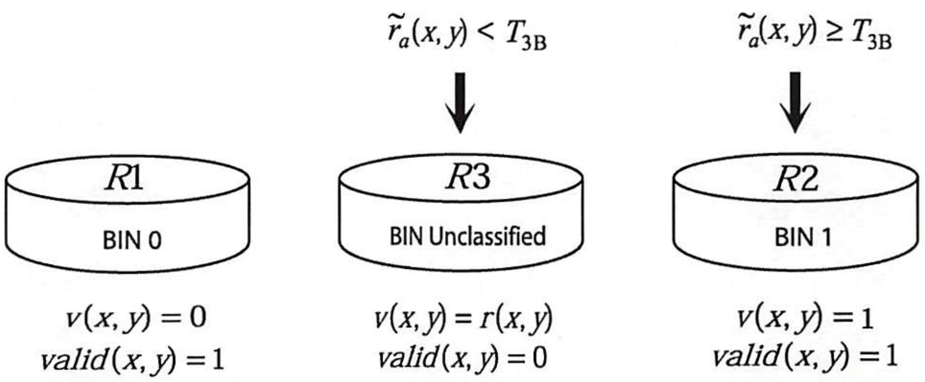


Figure 3.11: Pixel *one* update process.

An example of the two-step update procedure involving added intensity calculation and pixel *one* update procedure which are complementary to the procedures described in the previous section is illustrated in Fig. 3.12. Figure 3.12 shows the overlap region, two types of clusters occupying this region, and their corresponding mapped counterparts. The clusters and their corresponding mapped counterparts are shown adjacent to each other where they are shown on the left and right respectively. The solid arrows leading from each clusters point to their respective received intensity value. The clusters are mapped to their corresponding counterpart as described earlier and the intensities generated by mapped counterparts are added to the received intensity to produce the added intensity. The dashed line arrows indicate the added intensities of both clusters. The bottom cluster which has higher number of neighbor pixels with value of *zero* advances to the right along the intensity axis. Figure 3.12 shows that the added intensity from the bottom cluster crosses the threshold value and its center pixel is classified as pixel *one*, while the top cluster fails the update test and remains unclassified.

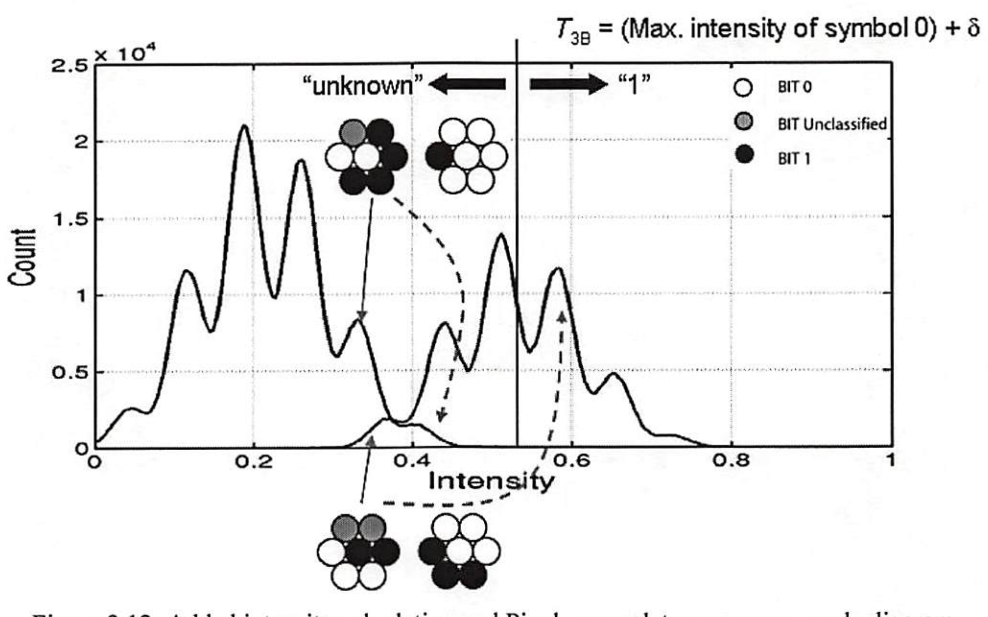


Figure 3.12: Added intensity calculation and Pixel *one* update process example diagram.

3.2.3.1.4 Parameter Readjustment and Stopping Criteria

The procedure is iterative and some parameters must be adjusted for the next iteration. The update of parameters T_1^i and T_2^i is needed. Currently, the threshold is adjusted by a fixed interval value according to

$$T_1^{i+1} = \begin{cases} T_1^i + \delta & \text{if } T_1^i + \delta < T_{1,\min} \\ T_1^i & \text{if } T_1^i + \delta \geq T_{1,\min} \end{cases}, \qquad (3.11)$$
$$T_2^{i+1} = \begin{cases} T_2^i - \delta & \text{if } T_2^i - \delta < T_{2,\max} \\ T_2^i & \text{if } T_2^i - \delta \geq T_{2,\max} \end{cases}$$

where δ is the update interval. The parameter δ depends on the rate of convergence needed and at the same time, the update must avoid entering the overlapping region.

Because of its iterative nature, there is possibility that the algorithm will not satisfy its conditions, or that it is not convergent. Therefore, stopping criteria are set so that the algorithm does not consume computation time when additional significant performance gain cannot be achieved. In this algorithm, two indices are used as the stopping criteria: the number of iterations and the number of new classified data pixels. Limiting the number of iterations is crucial when dealing with a system having a throughput constraint requiring that the data stream must be available at a specific time. This index is dependent on the individual system constraints. Another index is the number of new classified data pixels. The algorithm sometimes continues to operate but yield low number of new classified data pixels. A constraint must be set so that the computation power is conserved.

3.2.3.1.5 Fixed Threshold Step

The last step of the classification stage is called the fixed threshold step. This step is outside the iterative loop of the classification stage. When the algorithm has exhausted all its attempts, most likely there are some remaining data pixels left unclassified. In previous steps, we have exploited all reliable information, and we must use an alternative method to classify the remaining pixels. To prepare the data for the next stage, another threshold decision is performed and described by

$$\begin{array}{ll} \text{if } r(x,y) \geq T_4 & \text{, pixels assigned as } \textit{one} \\ \text{if } r(x,y) < T_4 & \text{, pixels assigned as } \textit{zero} \end{array}, \tag{3.12}$$

where T_4 is a decision threshold, that can generally be any value within the range of intensity values. For better results, we set it somewhere inside the histogram overlap region, and for the best prediction, set the threshold level at the minimum error probability level determined by the intersection of the histograms representing two symbols as shown in Fig. 3.13. The parameter $v(x,y)$ is assigned as 0 or 1 depending on the threshold result, and parameter $valid(x,y)$ becomes 1. While the fixed threshold classifies the data pixels into classes, it also introduces errors by incorrectly classifying some data pixels. Therefore, the data pixels classified in the fixed threshold step are marked for further processing in the following steps. Given that we set the threshold level at the minimum error probability, this step alone can guarantee us the result of equal or greater performance than the simple threshold algorithm.

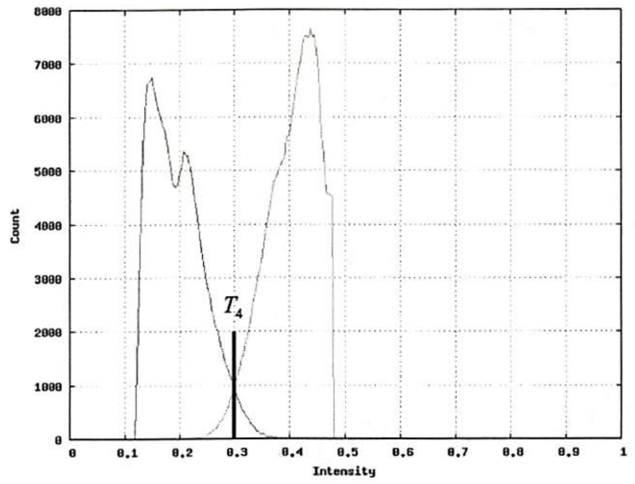


Figure 3.13: Fixed threshold step.

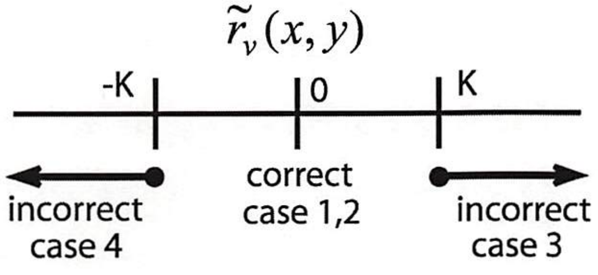
3.2.3.2 Verification Stage

The verification step attempts to detect and correct errors that have been introduced in the last step of the classification stage. Following the previous stage, all data pixels have been classified with a value of either 0 or 1. However, some of the data pixels that have been classified by fixed threshold step are erroneous, so the verification step is performed to ensure the correctness of data and further reduce the probability of error. At this stage of the algorithm, the received intensity and an estimated pixel value defined by $\hat{r}(x,y)$ are known. We then perform a calculation to verify that the estimated data are consistent with the original data. In this step, only the data pixels marked for processing in the fixed threshold step are examined. This restriction is made to reduce the propagation of error to the correctly classified data pixels. In a practical implementation, it also helps reduce the power consumption. The verification equation is described as

$$\tilde{r}_v(x,y) = \left[\hat{r}(x,y) + \sum_{\substack{(x_k, y_k) \in \\ \text{neighbor pixels}}} h(x_k - x, y_k - y) * v(x_k, y_k) \right] - r(x,y), \qquad (3.13)$$

where $r(x,y)$ is the original received intensity, $v(x,y)$ is the classified value of the data pixel and $\tilde{r}_v(x,y)$ is the difference between the calculated intensity and the original intensity.

The quantity in the brackets is the estimated intensity that the center pixel should produce at the detector including all the ISI contribution from its nearest neighbors. The original measurement $r(x,y)$ is subtracted from this bracketed quantity to produce $\hat{r}_v(x,y)$, which is essentially the difference between the estimated intensity and the actual received intensity. There are four possible outcomes of this test as shown in Fig. 3.14.



Case 1: true value = 0; estimate = 0

Case 2: true value = 1; estimate = 1

Case 3: true value = 0; estimate = 1

Case 4: true value = 1; estimate = 0

Figure 3.14: Outcome of the verification check.

In case 1 and 2, the prediction is correct, and the magnitude of $\hat{r}_v(x,y)$ is small and bounded within a margin of *K*, a predefined threshold value. In case 3, and 4, the prediction is incorrect and the magnitude of $\hat{r}_v(x,y)$ is expected to be significant.

When the outcome is identified, we proceed to the correction phase. If the outcome falls into case 1 or 2, the result is assumed to be correct. However, if the outcome falls into case 3, or 4, the result is assumed to be incorrect and update must be performed. The correction is carried out according to

$$v(x,y) = \begin{cases} v(x,y) & ,\text{if } |\hat{r}_v(x,y)| \le K \\ 1 - v(x,y) & ,\text{otherwise} \end{cases} . \quad (3.14)$$

Equation (3.14) limits the correction procedure to two cases: correct estimate and incorrect estimate.

For a two-level (binary) system, the update rule is relatively simple and it follows the second row of Eq. (3.14) in which the correction value is the binary complement of the old value; i.e. if the old value is a pixel *zero*, the corrected value is a pixel *one* and vice-versa. The correction procedure is simple and efficient, but relies on Eq. (3.13) to produce parameter $\hat{r}_v(x,y)$, and if a pixel contains many incorrect neighbor pixels, the procedure cannot readily verify and correct that pixel. The solution is to allow the verification procedure to run iteratively and make the correction on a future iteration. Similar to the iterative portion of the classification stage, the verification stage has an iteration limit and the iteration is terminated when the number of iterations exceeds a predefined value or there is insignificant update progress in a particular iteration.

3.2.4 Operation Progress Example

Figure 3.15 show an example of the normal processing steps in the algorithm. The example shown has the following experimental parameters: rate $= 6/9$, $\sigma = 0.04$, $b_S = 2.5$, and page size 1200 x 1200 pixels. Only a 24 x 24 pixel portion of the 1200 x 1200 pixel array is shown for simplicity. The processing steps are shown in chronological order and some steps are omitted to make the example more concise. At any particular step, there are three types of pixels present: unclassified pixel, pixel *zero*, and pixel *one*. These three types of pixels are represented graphically by light gray circle, white circle, and black circle respectively. During the fixed threshold step and the verification step, all pixels are classified and the representation of unclassified pixels is no longer necessary. However, there may exist error pixels after the fixed threshold step (step 11) and we represent these error pixels graphically by black squares marked with cross inside.

Step 1 shows the start of the algorithm in which all pixels are unclassified. Step 2 shows the data page after the pre-process step which includes known pixel decoding. The next two steps are the iterative classification of the algorithm: the variable threshold step and the update step. Step 3 classifies some pixels using threshold values, which are then used in step 4 to update more pixels.

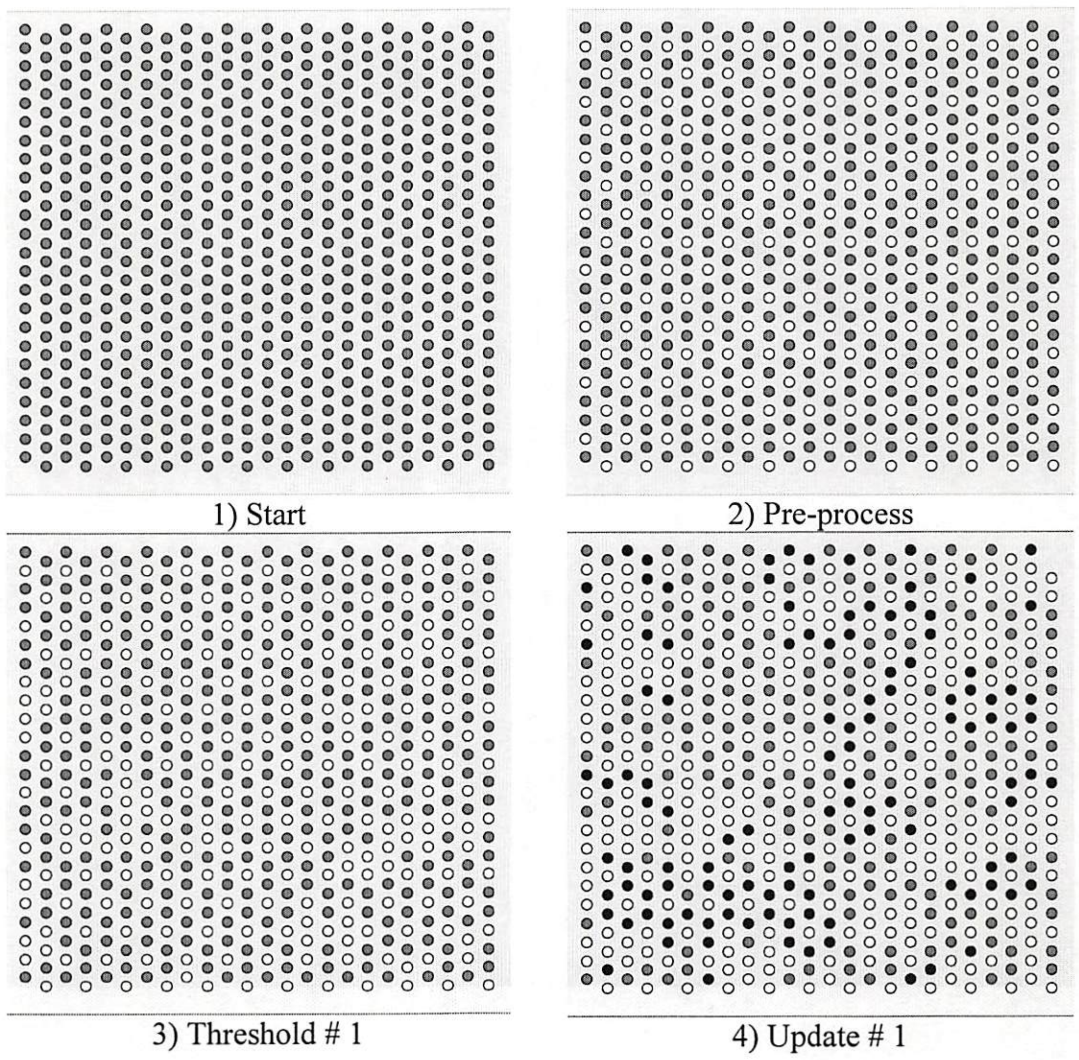


Figure 3.15: Example.

Steps 5 and 6, and 7 and 8 show the processing of 2nd and 3rd iterations respectively. The views shown are a small part of the complete 1200x1200 pixel page. While some iterations do not show any significant change or update, updates may occur in some other part of the same page and are not visible in this display.

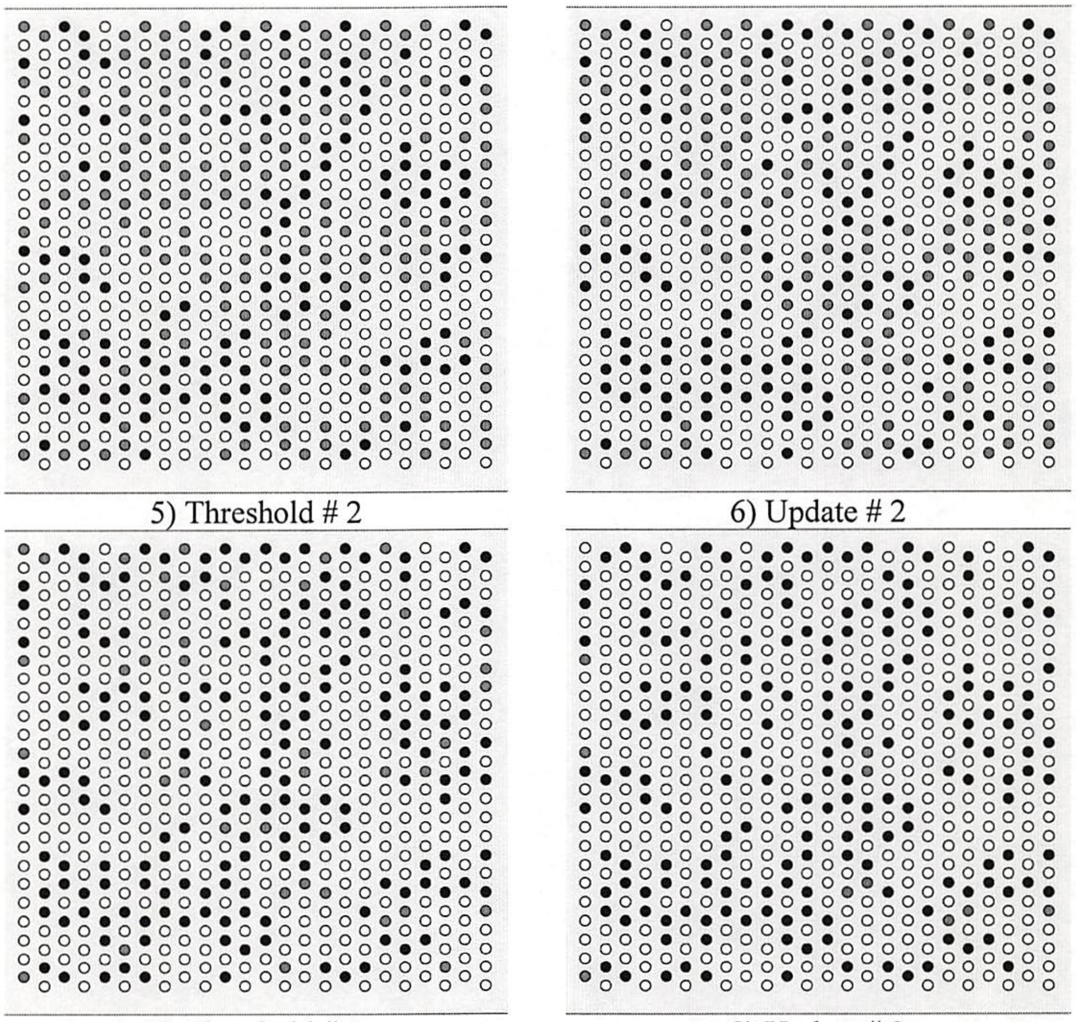
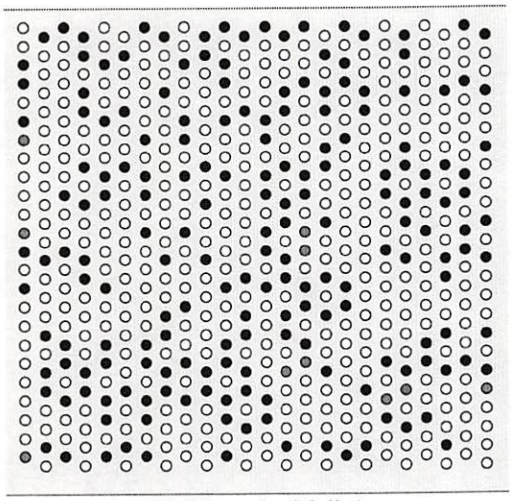
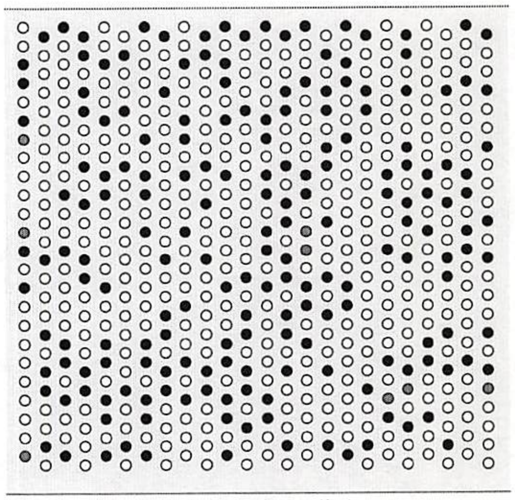
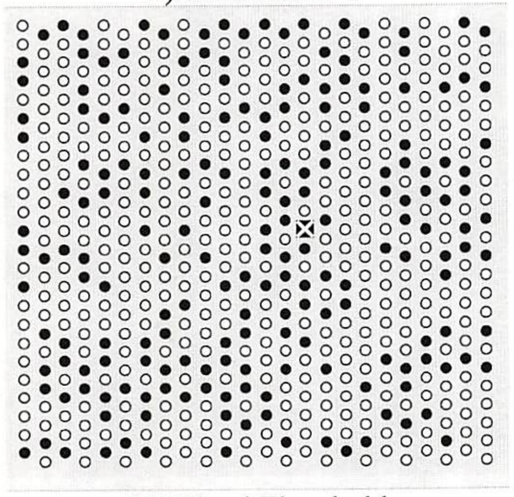


Figure 3.15: Example (Continued).

Step 9 and 10 show the iterative steps for 4th iteration. However, the number of updates in this iteration is significantly smaller compared to the previous iterations, and the iterative loop is terminated. Step 11 performs fixed threshold using Eq. (3.12), which introduces errors represented by black squares marked with cross inside. Step 12 verifies the consistency of the classification output with the received intensity and corrects the errors if inconsistency occurs.


9) Threshold # 4


10) Update # 4


11) Fixed Threshold

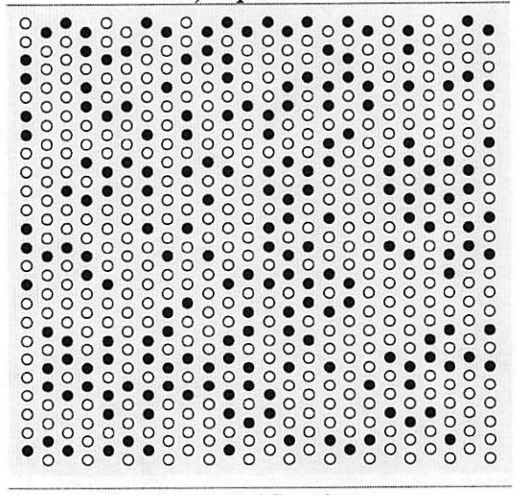

12) Verification

Figure 3.15: Example (Continued).

For this particular example, the algorithm continues through four iterations in the classification stage and two iterations in verification stage. The resulting BER yielded is 5.9 x 10^{-5}, and the signal-to-noise level is 11.6784 dB. The number of iterations required depends on various parameters such as code rate, noise level, etc. The performance of the algorithm described in this chapter is discussed in the following chapter.

Chapter 4

Bit-Error-Rate Performance Analysis and Complexity Analysis of the Iterative Procedure in PODS Systems

In this chapter, we present the simulation results of bit-error-rate (BER) performance of the iterative algorithm described in Chapter 3. Bit-error-rate is defined by the total number of errors divided by total number of transmitted bits and is an appropriate performance index for the inner modulation coding part of a data storage system.

4.1 Simulation Environment Parameters

In this section, we describe the experimental parameters used in the test simulation. Multiple test data pages are generated to test the algorithm performance. Each test data page is 1200 pixels wide and 1200 pixels high. The pixels are arranged in hexagonal grid format described in Chapter 2. The data on the page is filled with binary user data, which is randomly generated with equal probability for each symbol. The starting random-generating seed value is also randomly generated to ensure statistical randomness of data. The inter-symbol interference (ISI) effect is modeled by the point-spread function (PSF) of Eq. (2.2), specifically the Jinc function with various blur parameters described in chapter 2. This PSF characterizes the two-dimensional footprint of the data point as it appears at the detector array.

Lastly, the data page is corrupted by additive white Gaussian noise (AWGN), which is randomly generated in a manner similar to the signal but independently from the signal data set. The severity of the noise environment is measured by a parameter called signal-to-noise ratio (SNR) defined by

$$SNR = \frac{VAR\left(r(x,y) * h(x,y)\right)}{\sigma_n^2}, \tag{4.1}$$

where σ_n is the AWGN standard deviation. A higher SNR indicates better signal reception and is desired. The parameter SNR is generally measured in logarithmic decibels or dB scale. This definition of signal-to-noise ratio simply describes the ratio between the variance of a noiseless data page to the variance of AWGN noise.

Input data to the algorithm is received as a raw analog value ranging between zero and one. At the quantizer, the input signal is transformed and is represented digitally by an 8-bit value as described in chapter 3.

4.2 BER Performance with Different Code Rates

Generally, higher data accuracy is achieved by creating additional redundancy to assure that there are sufficient correlated data to decode the original data. In capacity constrained systems such as data storage systems of interest here, redundancy is not an attractive solution. In this thesis, the algorithm attempts to decode the data without data redundancy. When dealing with ISI effects, modulation-coding techniques help reduce the effect of ISI by appropriately limiting ISI-contributing data pixels while achieving maximum capacity simultaneously. Using lower code rate allows the insertion of known non-information data pixels whose ISI effects are known making the decoding simpler. In general, as the code rate grows, the total effects of ISI become worse resulting in lower BER performance.

Figure 4.1 shows the output stream bit-error-rate performance as a function of SNR for various code rates and a fixed moderate-to-high level of blur (ISI) using the algorithm described in Chapter 3. As described in Eq. (4.1), a higher value of SNR reflects a lower noise level, thus yielding a better BER performance. The top curve shows poor performance using a simple threshold procedure. The second, third, fourth and fifth curves from the top show the BER performance of the iterative detection procedures with descending code rate values. As expected, they also show the increasing performance as the code rate value decreases. The graphs show that the result from the proposed algorithm outperforms the simple threshold by several orders of magnitude depending on the SNR level.

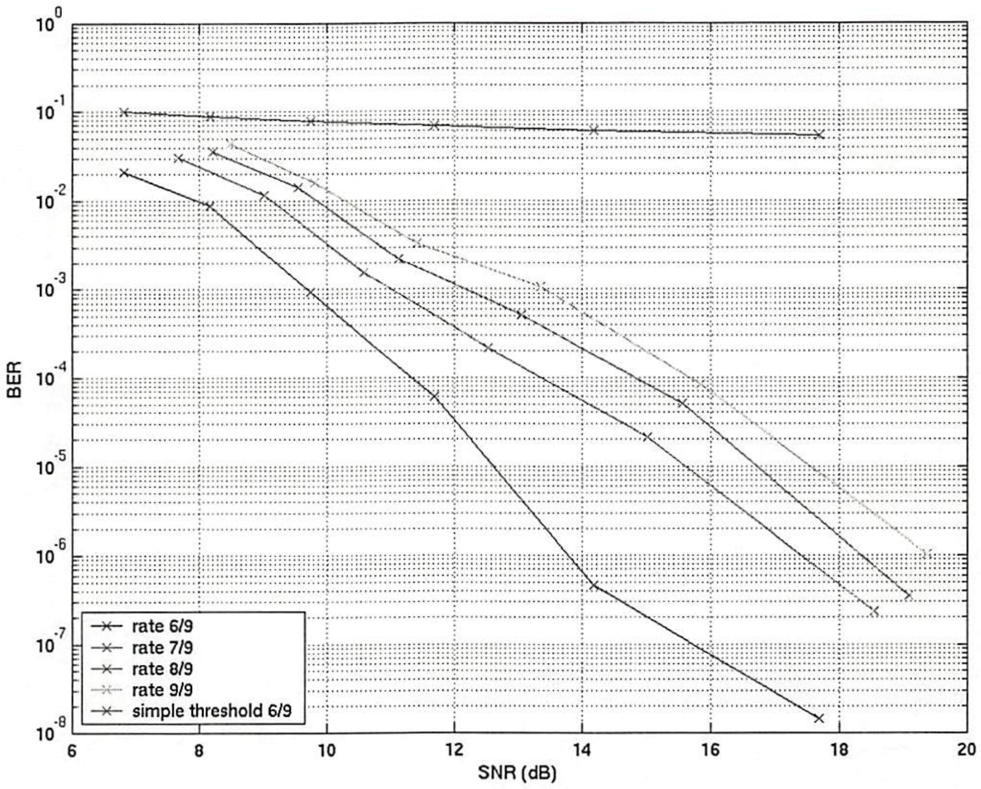


Figure 4.1: BER Performance of the two-dimensional modulation encoding and decoding procedure for different code rate, (blur factor $b_S = 2.5$).

4.3 BER Performance with Different Blur Factors

The blur factor is the index that indicates the severity of the inter-symbol interference effect. A higher value of blur factor represents a higher level of ISI effect. The relationship between the blur factor value and the level of ISI effect depends on the system point-spread function and not necessarily linear. Figure 4.2 illustrates the bit-error-rate performance when different degrees of ISI effect are applied to the data page. To emphasize the severity of the inter-symbol interference imposed on the test system, the blur factor corresponding to the Rayleigh limit is approximately 1.64, which is significantly lower than our test parameters.

The Rayleigh resolution criterion states that to resolve two distinct objects, the two objects must be separated by at least the Rayleigh minimum distance. In our particular system, the Rayleigh minimum distance is given by

$$d = \begin{cases} \dfrac{1.22 * b_S}{2} & \text{, jinc based PSF} \\ b_S & \text{, sinc based PSF} \end{cases}, \tag{4.2}$$

where d represents the Rayleigh minimum distance and the distance between the center of PSF to its first zero. Figure 4.2 shows the intensity footprint and the combine intensity cross section of two bright spots spatially separated by the Rayleigh limit distance (based on jinc function).

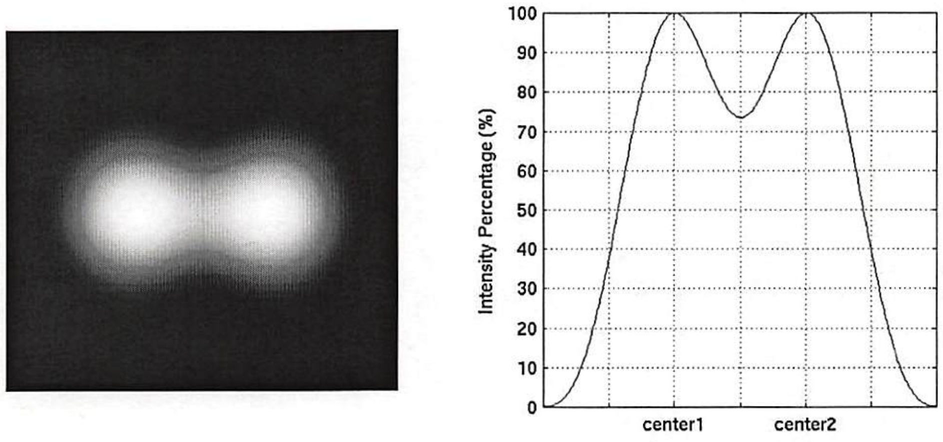


Figure 4.2: Rayleigh limit resolution (Jinc function): Left: Footprint of two bright spots separated by Rayleigh limit distance, Right: The cross-section of the combined intensity

The result shown in Fig 4.3 is obtained using the algorithm described in Chapter 3 in the presence of a moderate amount of additive white Gaussian noise (the SNR ranges from 10 to 15 dB). Figure 4.3 shows that the performance improves as the blur factor is reduced, and it also shows that after a specific blur value threshold, the BER becomes very small for all code rates less than one. The blur parameters used are much higher than the allowable Rayleigh limit and does not represent a practical system implementation.

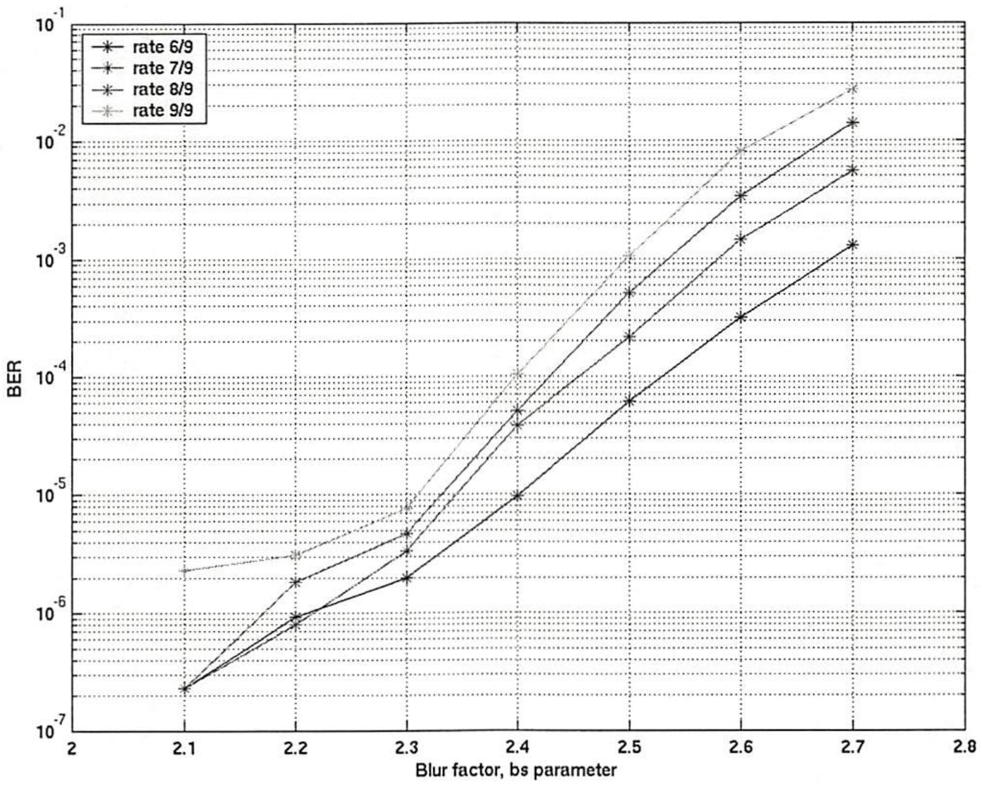


Figure 4.3: BER Performance of two-dimensional data at different blur factor (AWGN noise standard deviation $\sigma = 0.04$).

We recall from chapter 2 that a typical data storage system consists of three major signal processing steps: modulation, detection/equalization and ECC. To demonstrate the acceptable bit-error-rate performance in detection unit, the output BER of detection must satisfy the input BER of the ECC unit in order to achieve a desired low BER for the whole system.

As an illustrative example, we choose the Reed-Solomon (RS) code is to implement ECC unit because of its flexibility, efficiency, and availability. Figure 4.4 shows the relationship between input BER and output BER for various common types of RS codes. Typical data storage systems expect a BER of better than 10^{-15} after ECC decoding (1 bit out of approximately 100 TB). From Fig. 4.4, Reed-Solomon can achieve this overall BER level with an input BER in the range of 10^{-3} to 10^{-4} using ECC codes with reasonable code rates.

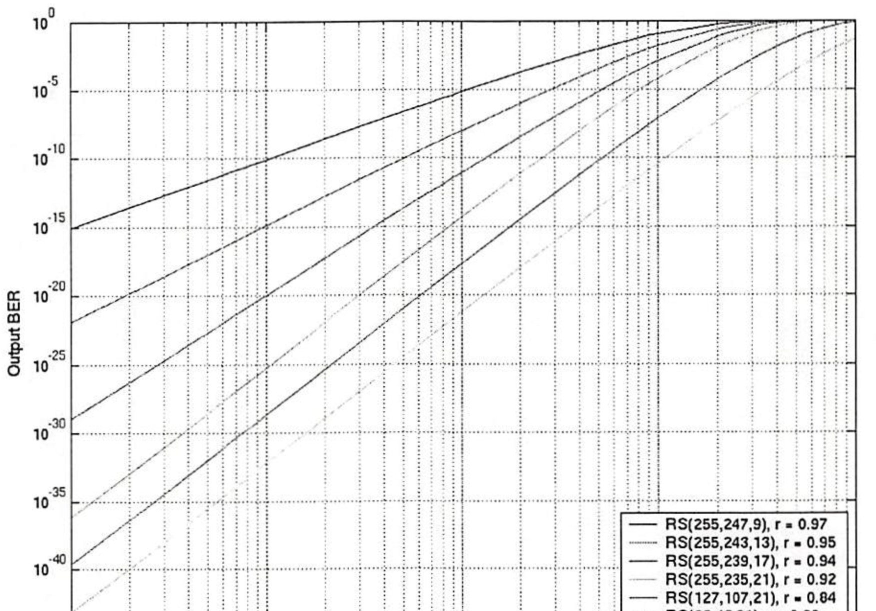


Figure 4.4: Reed-Solomon code input/output bit-error-rate requirement.

4.4 Complexity Analysis

One of the main design goals for this algorithm is to decode data with low calculation cost while maintaining acceptable data accuracy. The complexity of the algorithm also depends on how the algorithm is implemented. There are several factors such as the structure of the implementation (cascaded or hierarchical style), amount of resources available (single or multiple processors), etc. For analysis purposes, in this thesis we assume that a single processor is used and that the algorithm is implemented in serial fashion as described in Fig. 3.1.

4.4.1 Classification Stage

The classification stage consists of an iterative part and a non-iterative part. The iterative part includes the variable threshold, residual intensity calculation, pixel *zero* update, added intensity calculation, and pixel *one* update. The non-iterative part includes the preprocessing step and the fixed threshold step.

4.4.1.1 Non-Iterative Part

The preprocessing step consumes a fixed amount of computation time, and involves classifying known data pixels. We can bypass this step by marking the sensor pixels that correspond to these known data pixels. However for a worst-case analysis, we assume that this step is included, and that a constant amount of time per pixel (depending on the transition speed of the registers) is spent in this step. Another non-iterative procedure is the fixed threshold step, and it consumes a variable amount of computation time depending on the number of remaining unclassified data pixels, which can range from zero to the whole page. The procedure performs one comparison (addition) per pixel.

4.4.1.2 Iterative Part

The iterative part consists of several steps. However, the steps that carried the highest computational load are residual intensity and added intensity calculation. We examine these two steps first and later analyze the remaining iterative steps. Residual and added intensity calculations involve convolving classified data pixels' values to the available known point-spread function. As described in chapter 3, the system point-spread function is stored in a look-up table. Table look-up is computationally more efficient than costly multiplication operations. Only valid pixel values are retrieved, requiring a condition logic check, and the validation can be done through simple AND operations. In added intensity calculation, there is an additional overhead of bit-complementing. Bit-complementing takes one fast logic operation (~ operation) in binary system, however for *M-ary* system, it requires one addition operation. To satisfy worst-case analysis, this overhead is concluded as addition operation. The remaining iterative procedures are the pixel *zero* and *one* update procedures, which require simple comparison implemented as one addition operation. It can be seen that the residual (added) intensity calculation and update pixel *zero* (*one*) as shown in Fig. 3.1 can be combined as one step rather than two sequential steps. However, the two steps are separated to avoid one long step requiring longer clock cycle to process. Thus, they are separately analyzed.

The iterative part of the classification stage takes C1 iterations to complete depending on the environment parameters. Parameter C1 is constrained by two factors: time and update efficiency yield. In our particular implementation, the update efficiency yield is the number of pixels being updated on a particular iteration. The iteration is terminated when the update efficiency yield is low or the allotted time for the classification stage runs out. Therefore, the iteration parameter C1 is bounded and independent of the page size. The constraints controlling the parameter C1 are arbitrarily set depending on the system requirement. For a time constraint system, the time allotted time is short, while a power constraint system, the minimum number of updated pixels is set higher. However, these adjustments must be done with the overall performance in mind.

4.4.2 Verification Stage

The verification stage consists of one processing step. The verification includes summation of all local contributions including the center pixel, and compares the summation to the actual received intensity. The step is carried out totally by addition operations and consumes C2 iterations to complete. Again, parameter C2 is constrained by two factors: time and update efficiency. In most cases, this stage takes a very short processing time and update efficiency becomes more dominant. We can conclude that C2 is bounded and independent of the page size.

4.4.3 Complexity Summary

We summarize the complexity requirements for all processing steps here.

Classification Add ops. $= n^2 + C1 \cdot \lfloor 2n^2 + 6n^2 + n^2 + 12n^2 + n^2 \rfloor + n^2$

$= 2n^2 + 22 * C1 * n^2$

AND ops. $= C1 \cdot \lfloor 6n^2 + 6n^2 \rfloor$

$= 12 * \mathrm{C1} * n^2$

Verification Add ops. $= 9 * C2 * n^2$

Classification complexity $= O(C1 \cdot n^2)$

Verification complexity $= O(C2 \cdot n^2)$

Total complexity $= O([C1 + C2] \cdot n^2)$

CONDITIONS: $C1$ and $C2$ bounded

$C1$ and $C2 << n$

Asymptotic complexity $= O(n^2)$

Table 4.1 shows cost details for each step of the algorithm. In this analysis, the addition operation is used as the basic operation. We also assume that the AND operation has a much shorter execution time than the addition operation. Single processor and cascaded operation are also assumed.

STAGE	**STEP**	**Cost**	**Iteratio**
CLASSIFICATION	Preprocess	n^2 additions	1
	Variable threshold	$2n^2$ additions	C_1
	Residual intensity	$6n^2$ additions + $6n^2$ AND	C_1
	Update pixel *zero*	n^2 additions	C_1
	Added intensity	$12n^2$ additions + $6n^2$ AND	C_1
	Update pixel *one*	n^2 additions	C_1
	Fixed Threshold	n^2 additions	1
VERIFICATION	Verification	$9n^2$ additions	C_2

Table 4.1: Cost analysis (single processor unit)

For pages with $n x n$ pixels, the algorithm complexity becomes linear if parallel operation is utilized. Parallel processing is possible through the use of smart pixels, which are detectors having attached processing capability at each pixel. If we consider page-oriented optical memory, which has a parallel readout channel at fast data transfer rates, it seems more logical to perform the algorithm in parallel manner to reduce the bottleneck. In Fig. 4.5, simple hardware implementations are shown for some steps of the algorithm. Thus it may be feasible to embed this simple hardware into each sensor.

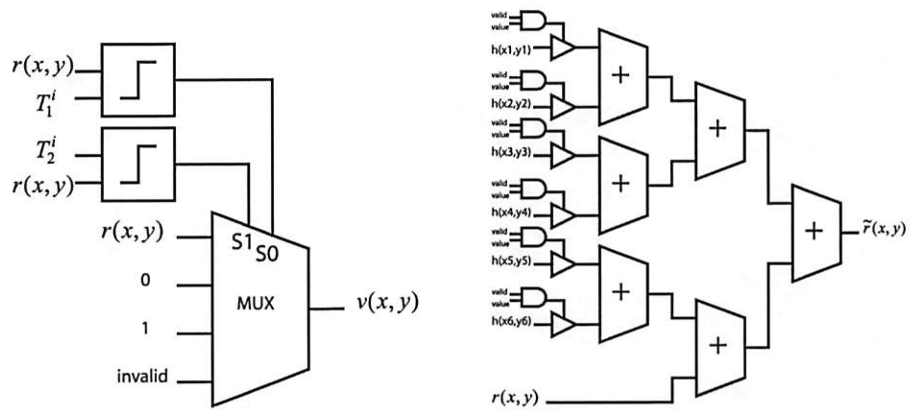


Figure 4.5: Simple hardware implementations of some steps in the algorithm; Left: Variable threshold step, Right: residual (added) intensity, and verification steps.

4.4.4 Complexity Comparison with Viterbi Detection Algorithm

Our research goal is to design an algorithm that has low computational complexity while maintaining acceptable bit-error-rate performance. The Viterbi detection algorithm (VDA) is a detection algorithm that is used extensively in various communications applications and it is an appropriate benchmark to compare with our algorithm. The VDA performs remarkably well in systems corrupted by one-dimensional ISI. The review of VDA algorithm is described in Chapter 2. In some recent efforts [17], some researchers have implemented the VDA in a page-oriented optical data storage system.

The VDA is generally used for sequence detection of data streams generated by a convolutional encoder. First, we describe the one-dimensional implementation of the VDA and convolutional encoder. A one-dimensional data stream enters the convolutional encoder and the output is called convolutional code, which is usually described using two parameters: code rate (*K*) and constraint length (*L*). The design parameter that is relevant to our analysis is the constraint length which is defined as the number of symbols relevant to the encoding of a transmitted symbol. The constraint length is typically between 5 and 9 symbols and a higher constraint length generally provides a higher BER performance. The VDA uses a trellis to perform decoding and based its operations on a unit called ACS (Add Compare Select) unit. The algorithm can be thought of as an organized way of performing exhaustive search on all possible paths on the trellis. Nevertheless, the complexity of the algorithm is high and the asymptotic computational complexity of one-dimensional VDA is $O(2^L)$ per symbol.

The two-dimensional data stream poses a different scenario and requires a different approach to the problem. There are many possible implementations and we describe one here. The approach treats two-dimensional data stream as two separable one-dimensional data stream and two separate VDA operations perform on vertical and horizontal data streams independently. Assuming the ISI effects along vertical and horizontal axes are the same and their channel memory lengths (*L*) are the same, the asymptotic computational complexity is $O(2^{2L})$ per symbol and for a data page of size $n \times n$, the complexity is $O(2^{2L}n^2)$. The overall channel memory lengths are *2L* because the ISI interference arrives from two directions along the axis and each has channel memory length of L. For short memory length ($L = 2\text{-}3$), the complexity of the algorithm is still manageable. However, the real disadvantage is the BER performance which becomes lower due to the incomplete processing of data corrupted by two-dimensional ISI.

Table 4.2 shows the computational complexity and constraint comparison between several algorithms including 2D Viterbi, conventional equalization, and our iterative detection algorithm. The Viterbi algorithm has computational complexity that grows exponentially proportional to the system constraint defined by the channel memory. A conventional equalization method has a fixed computational cost, however most operations are floating point multiplications which are computationally expensive. Our proposed iterative detection algorithm has the same order of asymptotic computational complexity as conventional equalization, however all operations are addition or faster operations.

	Complexity	**Constraint**
2D Viterbi	$O(2^{2L} n^2)$	L = Constraint length (channel memory length)
Equalization (Wiener, etc.)	$O(w^2 n^2)$	w = Equalization window size
Iterative Detection	$O(C_a n^2)$	C_a = Cost per pixel (number of loops)

Table 4.2: Comparison of computational complexities and constraints

Chapter 5

Extension to Three-Dimensional Intersymbol Interference (Interpage Interference) for Higher Density PODS Systems

5.1 Interpage Interference (IPI), or Three-Dimensional Intersymbol Interference

In previous chapters, our algorithm is designed to mitigate two-dimensional intersymbol interference (ISI). Two-dimensional ISI arises when the packing density of the media is increased due to pixel pitch reduction, such that the pixels on a given page become too close to each other. This technique of increasing the packing density takes advantage only of the planar storage area and the storage capacity limitation is determined by how many pixels can be packed on a page. However, page-oriented optical storage has a structural benefit over conventional planar based memory, because the information is recorded on volumetric media rather than on planar media as in magnetic storage and we can take advantage of the extra dimension.

In previous discussions of two-dimensional ISI, the page spacing (page separation) is assumed to be sufficiently large so that the ISI introduced from page-to-page is negligible. By reducing the interpage spacing, further capacity improvement can be achieved. However, the effect of ISI may propagate to adjacent pages. The effective amount of ISI resulting from page-to-page crosstalk is determined by the page spacing and the readout beam profile. as shown in Fig. 2.7

Ideally, the readout beam profile has a small beam width and illuminates only the intended target page. However, in practical implementations, the beam may be wider and may illuminate neighboring pages. The portion of the beam that illuminates the adjacent pages causes them to fluoresce. Thus, the detector array collects photons originating from not only within the intended page but also the neighboring pages. This type of ISI effect is called inter-page interference (IPI), or three-dimensional ISI. Figure 5.1 shows the rough structure of the readout sheet beam where L, W and D are the length, width and depth of the sheet beam respectively. The data pixels are recorded on the plane L-D. Parameters L and D determine the size of data page, and W is the space between adjacent pages. W may be smaller than the allowable page separation leading to inter-page interference crosstalk (IPI). To simplify the IPI model, we assume that the readout beam profile of the illumination beam sheet along the width axis is modeled by a Gaussian beam profile whose center is equidistant from the sides of the beam and whose irradiance is constant along the D axis. The readout beam profile is shown in Fig. 5.2a, and the cross section of the readout beam is shown in Fig. 5.2b. Figure 5.2b also shows white rectangular borders representing the cross-sectional boundaries of the data pages

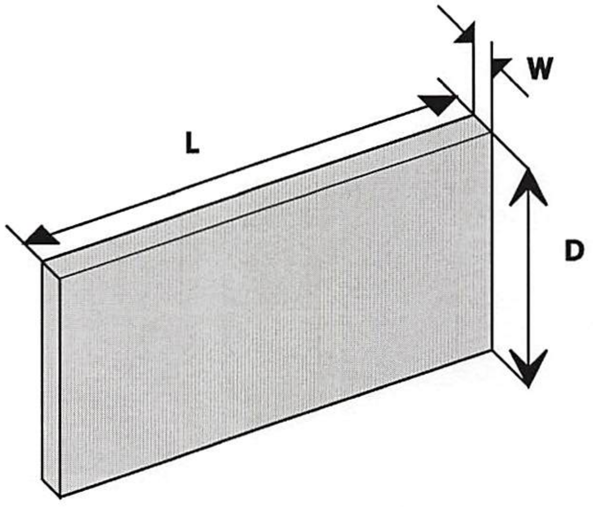
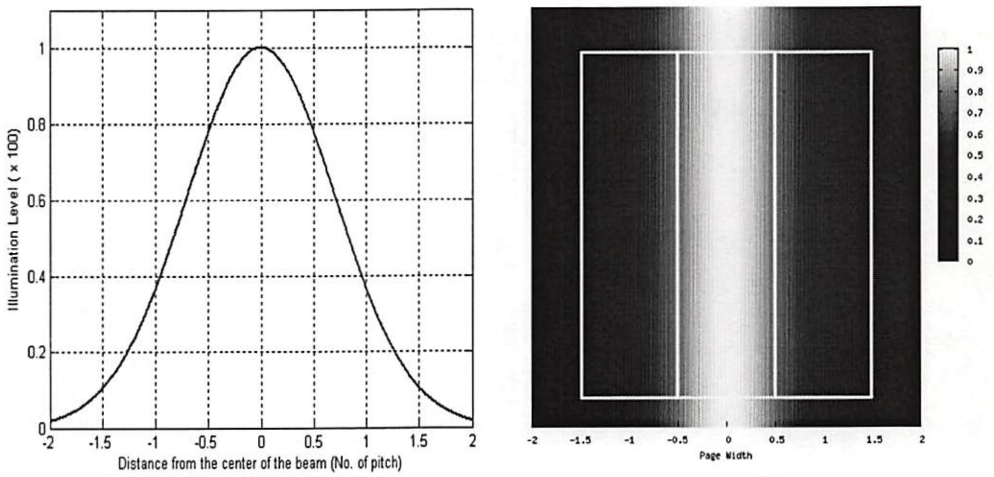


Figure 5.1: Illumination beam sheet structure, when illumination is from the top along D axis.



a) b)

Figure 5.2: Readout irradiance and cross section of beam profile.

With IPI, there are multiple light sources originating from multiple pages. We model the intended page (at the middle of the beam profile) as the only perfectly aligned and in-focus page. The pages located before and after the intended page are generally out-of-focus and also may be misaligned with respect to the detectors.

Because the spacing between pages is very small compared to the distance from pages to the detector, we initially assume perfect registration to simplify our model.

At the detector, the adjacent pages are modeled as a weighted image of the adjacent pages, because the adjacent pages are only partially illuminated by the readout beam. The model of IPI channel becomes the weighted sum of all contributing pages including the intended page, which has the weight of one. The accumulated intensity is described by

$$\hat{r}_I(x,y) = \sum_{\substack{J \in \\ \mathit{CONTRIBUTING} \\ \mathit{PAGES}}} \gamma_J \cdot r_J(x,y), \tag{5.1}$$

where γ_J represents the weight coefficient for page J, $r_J(x,y)$ represents the received intensity for page J and $\hat{r}_I(x,y)$ is the combined received intensity of page I at the detector. As mentioned earlier, the readout beam is generally very narrow and illuminates only the intended page, however in the case of IPI, only the immediate adjacent pages are considered significant contributing pages. The intensity originating from pages further away are considered negligible. Thus the expression described in Eq. (5.1) can be further simplified to

$$\hat{r}_I(x,y) = \gamma_{I-1} r_{I-1}(x,y) + \gamma_I r_I(x,y) + \gamma_{I+1} r_{I+1}(x,y), \tag{5.2}$$

where

$$\gamma_{I-1} + \gamma_I + \gamma_{I+1} = 1. \tag{5.3}$$

While Eq. (5.2) shows three contributing pages, the number of included pages is not restricted to three. However, we simplify our model to include only three center pages that provide the most significant contributions, and we assume that other pages are negligible. Because of the nature of the readout system, only a readout beam with an appropriate wavelength can illuminate the pages. Thus the fluorescing pages (emitting light at a different wavelength) cannot become secondary light sources, and thus cannot create new fluorescent signal contributions. An additional source of noise is light scattered from within the storage media, which may create additional additive signals at the detector. Figures 5.3a and 5.3b show the simulated pages (*I*-*1*), and (*I*+*1*) respectively. Figure 5.3c shows an isolated pixel *one* on page (*I*), and Fig. 5.3d shows the combined detected intensity. As shown in Fig. 5.3, the IPI produces additional interference at the detector. As the page spacing becomes smaller, the weight coefficients on the neighbor pages become larger and the IPI effect consequently grows.

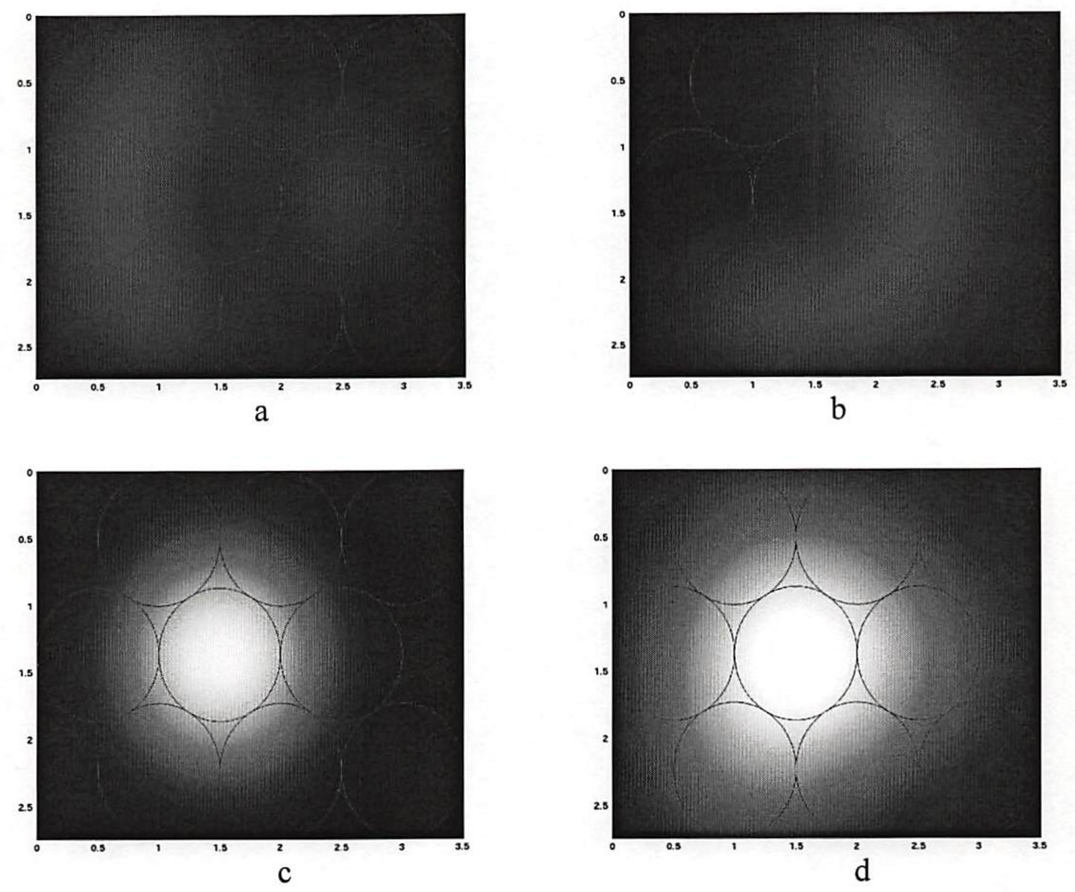


Figure 5.3: Simulated inter-page inter-symbol interference; a) Page *I-1* contribution, b) Page *I+1* contribution, c) Page *I* with ISI, without IPI effect, d) Page *I* with ISI and IPI effect ($\gamma = 0.15$).

5.2 Three Stage Pipeline Detection Procedure

A pipeline procedure is a type of parallel processing suitable for any computationally large algorithm that can be expressed as a sequence of many smaller independent tasks that together require long processing time. These independent tasks can be carried out in different stages using different computing units simultaneously. Smaller tasks require shorter computation times leading to faster overall processing. The pipeline architecture is particularly suitable for data-dependent detection algorithms such as the IPI detection algorithm described in Chapter 3 and section 5.1. From Eq. (5.2), there are interdependencies between the current sample and the preceding sample. Extending the data buffers to include all relevant samples solves the dependency of the preceding sample on the current sample, however, this dependency requires the current sample to wait for the completion of the preceding sample processing. Because the typical serial implementation of IPI has long processing time, and data dependency, it is computationally more efficient to implement it in pipeline fashion.

Figure 5.4 shows an example of a long processing time problem and implementation comparison. Our example has two sequential tasks: task A and task B. Task A consists of one task: task1, and task B consists of two smaller tasks: task2 and task3. Assuming that task1, task2, and task3 take equal time, task B takes twice as much time as task A. In serial processing, there are two computing units performing task

A and task B as shown in Fig. 5.4a. At time instance 2, it is shown that DATA2 cannot immediately proceed through task A until DATA1 finishes task B and remains idle for the amount of time it takes for DATA1 to finish task B. This slows down the processing time and reduces the overall data throughput. The pipeline implementation as shown in Fig. 5.4b divides task B into smaller tasks which eliminate or minimize the delay allowing an overall higher data throughput.

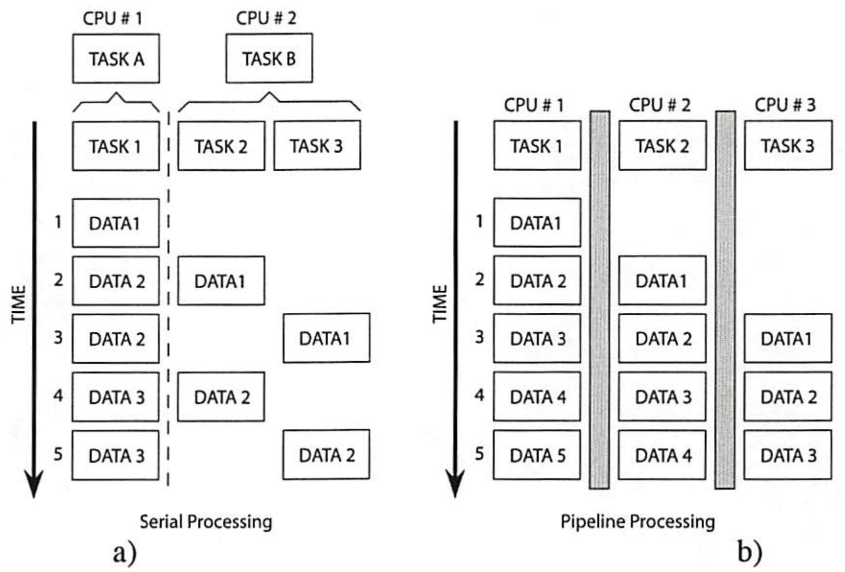


Figure 5.4: Long clock cycle problem and pipeline solution. a) Serial processing, b) Pipeline solution

Figure 5.5a shows an example of the data dependency problem. The algorithm has three sequential tasks with three computing units. Task1 and task3 require the current sample as its only input, and task2 requires two inputs: the current sample and the result of task2 processing from the previous sample. To achieve accurate results, the inputs to all tasks must have the most up-to-date result. At time instance 3, task 2 needs data2 and updated data1, however, data1 does not have time to update its value in the memory at the end of time instance 2, leading to obsolete data and calculation errors on the next time instance. Pipeline processing as shown in Fig. 5.5b eliminates data dependency by a procedure called data forwarding, which sends the updated data1 directly to task2. Thus, we have shown two advantages of pipeline implementation over serial implementation for the IPI algorithm. However, we need to modify the data buffer to implement effective the pipeline algorithm. We discuss the modifications in the next section.

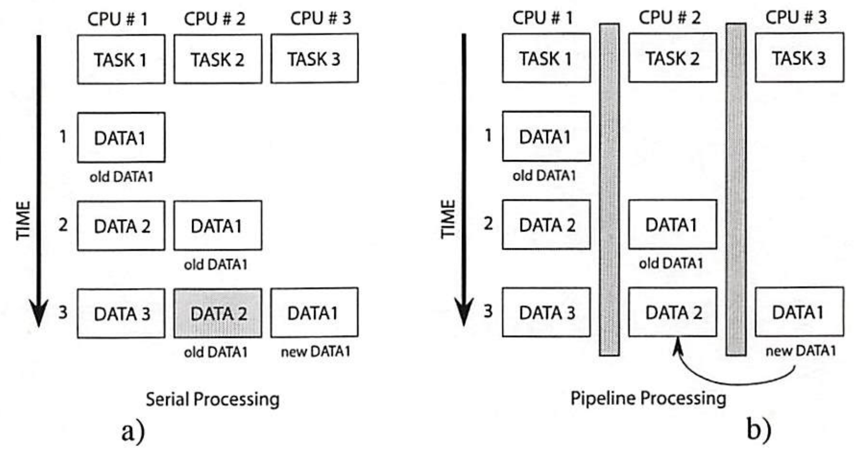


Figure 5.5: Data dependency problem and pipeline solution. a) Serial processing, b) Pipeline solution

5.2.1 Modified Buffer Slots

As described in Chapter 3, the detection procedure for the two-dimensional detection algorithm consists of a sequence of calculations performed in cascade as shown in Fig. 3.1. We have extended the procedure described in chapter 3 to the detection of data corrupted by IPI. As indicated in Eq. (5.2), the intensity calculations rely on the information originating not only from the current page (*I*), but also from the previous page (*I-1*), and the page behind it (*I+1*). In the original two-dimensional algorithm, only the current page (*I*) is available at any given time in the processing buffer, while the data from pages (*I-1*) and (*I+1*) are not available. Figure 5.6 shows the data buffering method in the original two-dimensional algorithm. It can be clearly shown that page (*I*) is the only page available to the processing unit, while page (*I-1*) has already become obsolete, and page (*I+1*) has not been retrieved yet.

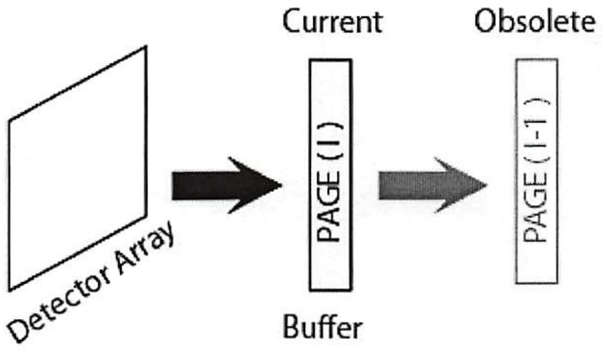


Figure 5.6: Data buffering in the two-dimensional algorithm.

To overcome this limitation, we expand the buffer size to include all necessary pages. The additional buffer pages require extra time overhead to fill them, and this effectively delays the start of the processing. Although, the buffer length of the new extended buffer is longer than the buffer for two-dimensional algorithm shown in Fig. 5.6, the processing throughput for each page is not affected due to the pipeline processing architecture. Figure 5.7 shows the new extended buffer for the pipeline algorithm, and the new intermediate buffer slots shown as buffers 2, 4, and 6.

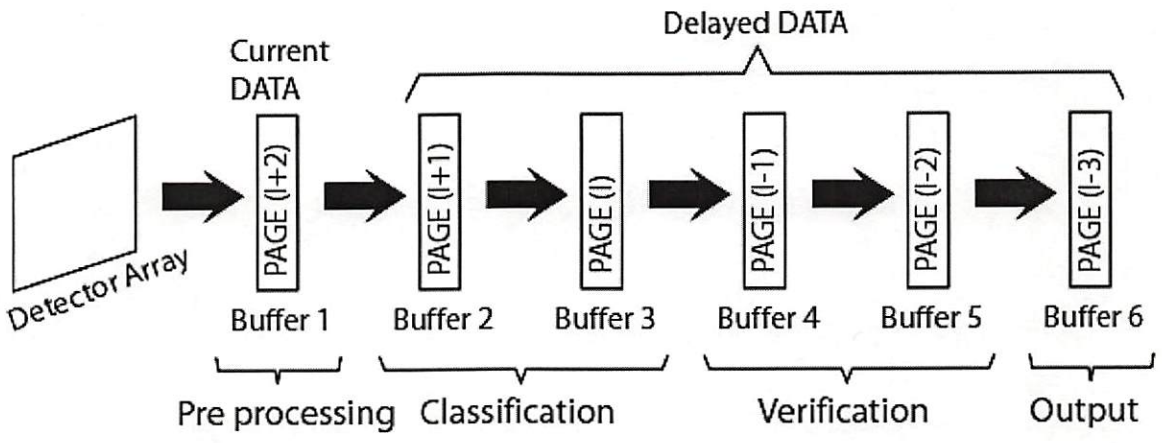


Figure 5.7: Modified data buffering in the pipeline algorithm.

One restriction of the pipeline process is a condition called the *race condition* which must be avoided. The race condition is an undesirable situation that occurs when the system attempts to perform read and write operations from the memory (buffers) simultaneously. One way to avoid the race condition is to serialize the data access (read or write). The serialization is carried out through the scheduling of the read and write access. All processes requesting data access are put in a queue, and at any given time, only one data access can be performed to avoid simultaneous accesses causing the race condition. However, this approach creates an apparent latency in the processing time. When two processes are requesting simultaneous accesses, one process must halt its process to wait for its turn to access the data, creating a bottleneck. The bottleneck not only slows down the algorithm but also, for a time sensitive system such as a data storage system, there may be insufficient time to complete the processing. To simplify the scheduling process and creating a faster pipeline processing in this system, a longer buffer similar to the diagram shown in

Fig. 5.7 is created to accommodate the intermediate pages. The intermediate pages are the pages between stages, and are available for read access and remain unchanged throughout the calculations in one cycle. The scheduling process is no longer necessary because at any given time, both read and write access can be performed on a buffer page without violating the race condition.

5.2.2 Pipeline Partitioning

Using the extended buffer described in the last section, pages (*I-1*), (*I*) and (*I+1*) which are necessary for processing the interpage interference, are included. Both two and three-dimensional algorithm use the partially classified binary data, $v(x,y)$ as the reliable information in the calculation rather than the received intensity $r(x,y)$. Although page (*I+1*) is available in the buffer but it is chronologically behind page (*I*), and has yet to be processed and therefore not useful for processing. The intermediate page (*I+1*) must first be pre-processed such that it arrives at the calculating stage with adequately reliable information to be useful. However, the intermediate page also relies on other pages to correctly classify its content. To overcome such data dependency and make the algorithm more efficient, we utilize a pipeline strategy.

By partitioning the two-dimensional algorithm into multiple independent tasks, these tasks can be carried out simultaneously by a pipeline process. Figure 5.8 illustrates the partitioning of the serialized two-dimensional algorithm into pipelined three-dimensional algorithm. As with the original algorithm, the pipeline contains three separable stages: pre-process, classification and verification, which are described in later sections.

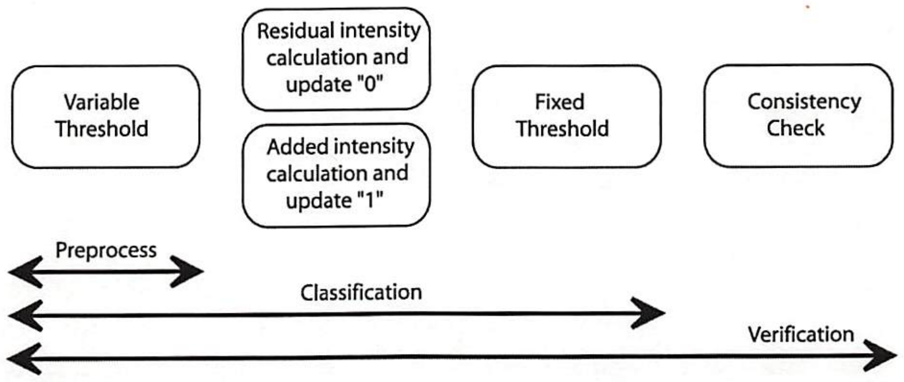

Variable Threshold
Residual intensity calculation and update "0"
Added intensity calculation and update "1"
Fixed Threshold
Consistency Check
Preprocess
Classification
Verification

Figure 5.8: Partition of original 2D algorithm.

To partition the stages, the necessary operations necessary for each stage are identified. Similar to the original ISI algorithm, the pipeline algorithm consists of a succession of staged calculations that implement the decoding procedure. To maximize the efficiency of the pipeline system, two factors must be considered: load balancing and task redundancy. The design objective is to distribute the computational load to all stages, and minimize the task redundancy among the stages.

5.2.3 Algorithm Stages

5.2.3.1 Preprocess Stage

The preprocessing stage performs preliminary operations to classify pixels whose measured intensities place them reliably within the *zero* or *one* classification ranges. The input of this stage is the received intensity from the detection, $r(x, y)$ and the output of this stage provides the intermediate data for the next stage (classification stage). The calculations carried out in this stage are not iterative and consume the shortest amount of time among the three stages. However, we separate its operation from the other stages to eliminate the data dependency problem. The calculations involved in this stage are classification of the non-information (known) pixels, and variable threshold step to separate pixels with a high probability of being correct as described in Chapter 2. Efficient pipeline implementation requires that the tasks are divided equally and distributed to each stage. This procedure is called load balancing and is an important design issue in the pipeline algorithm. The pre-processing stage has a relatively shorter processing time as compared to other stages and it remains idle until the end of the other stages. This problem needs to be addressed in further development.

5.2.3.2 Classification Stage

Similar to ISI problem, the detection stage performs iterative procedures including the variable threshold, residual (added) intensity calculations, and pixel *zero* (and *one*) updating steps using the data from pages *I-1*, *I+1*, and *I*. The most computationally expensive operations in this stage are the residual and added intensity calculations and the modified version of the residual (added) intensity test of Eq. (3.6) and (3.9) and is described by

$$\tilde{r}_{r,I}(x,y) = r_I(x,y) \mp h(x,y) * \left[\sum_{k=I-1}^{I+1} \gamma_k \cdot v_k(x,y) \cdot valid_k(x,y) \right] \qquad (5.4)$$

where k is the page index, and $\tilde{r}_{r,I}(x,y)$ is the residual (added) intensity signal of page *I*. Equation (5.2) shows that data from multiple pages are accessed simultaneously. This stage has the highest calculation load as shown in Table 4.1. The length of this stage puts a constraint on the length of the stage cycle for all stages. The solution is to allocate additional sub-stages and divide the separable calculation tasks inside the classification stage. The divided calculation tasks are then placed inside the newly allocated sub-stages. This solution allows a shorter stage cycle at the expense of extra component resources. Thus, there is a trade-off between stage cycle and component resources that must be considered.

5.2.3.3 Verification Stage

The last stage of the algorithm is to perform consistency verification. Similar to the two-dimensional algorithm, there are some classified pixels whose intensity is close to a decision boundary, and thus have low confidence. The verification stage is included to verify these pixels. The modified verification test of Eq. (3.13) is described by

$$\tilde{r}_{v,I}(x,y) = \left[\hat{r}_I(x,y) + h(x,y) * \left[\sum_{k=I-1}^{I+1} \gamma_k \cdot v_k(x,y)\right]\right] - r_I(x,y), \tag{5.5}$$

where $\tilde{r}_{v,I}(x,y)$ is the residual (added) intensity signal of page *I*. In addition, the data dependency causes incomplete classification in classification stage as shown in Eq. (5.2). It indicates that the calculation depends on three pages, pages *I*, *I-1*, and *I+1*. Pages *I-1* and *I+1* are referred to as the donor page, and page *I* is referred to as the recipient page. Page *I-1* is chronologically in front of the page being processed (page *I*), which does not cause any data dependency or incompleteness. However, page *I+1* is chronologically behind page *I*, and is only partially decoded during classification stage leading to low performance decoding. The solution is to delay the final decoding and verification until all donor pages are complete. In the verification stage, both donor pages are decoded, and we can verify the result with much better accuracy.

5.2.3.4 Algorithm Example

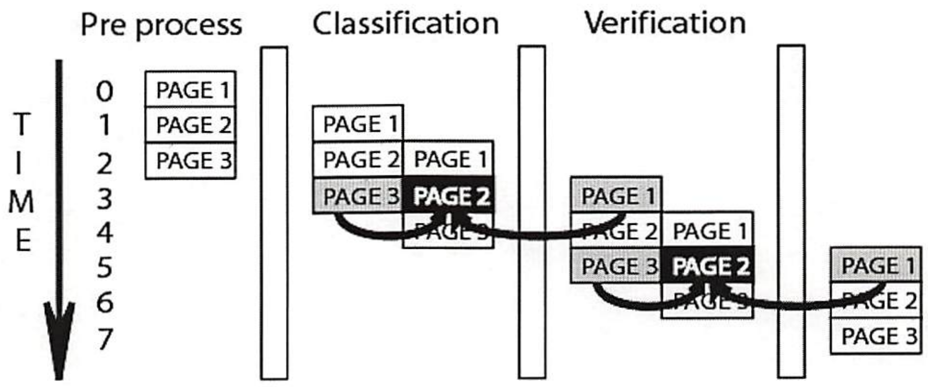


Figure 5.9: Three stages pipeline diagram of iterative detection.

An example of iterative detection using the pipeline process is shown in Fig. 5.9. In Fig. 5.9, the time progression advances vertically downward as shown on the left of the diagram. In this example, the target page being processed is page 2. At time instance 1, page 2's known pixels are extracted and some simple classification is done. The page information is passed to the classification stage. The classification stage performs the variable threshold, residual (added) calculations, and update *zero* (*one*) using the information from pages 1, 2, and 3. The donor pages are represented in the figure by gray rectangles, and the recipient page is represented by black rectangles. In pipeline processing, the data forwarding restriction rule indicates that the data in proceeding stages cannot be used or consumed in the preceding stages, this restriction prevents the *race condition* problem. Therefore, the classification stage occurs at time instance 3 rather than 2, because of the data forwarding restriction. Finally, the page information is passed to the verification stage. The procedure is similar to the procedure in classification stage, but the donor pages (pages 1 and 3) are decoded and provide more accurate information than in the detection stage. This stage is delayed to time instance 5 to avoid the data forwarding restriction.

5.2.4 Bit-Error-Rate Performance Results

We performed a simulation of the combined ISI and IPI pipeline detection to determine the BER as a function of other system variables. Multiple data pages are randomly filled with binary data values similar to the simulations carried out for ISI problem as described in Chapter 4. Each data page is 1200 pixels wide and 1200 pixels high. To create data randomness, the number of bits generated needs to larger than the reciprocal of the smallest BER. We generally need to generate at least one hundred pages to ensure accurate BER testing. The pipeline structure allocates memory for six data pages for processing as described in Fig. 5.7.

Figure 5.10 shows the experimental simulation results of 2D and 3D versions of the iterative page-oriented detection technique for various cases of severe ISI (pitch much smaller than Rayleigh limit) and IPI. The overall procedure is similar to a maximum likelihood detection procedure with much lower computational load (linear with page size compared to exponential for other methods). The vertical axis is the bit-error rate for the detected data as a function of the data signal-to-noise ratio (SNR) assuming additive white Gaussian noise.

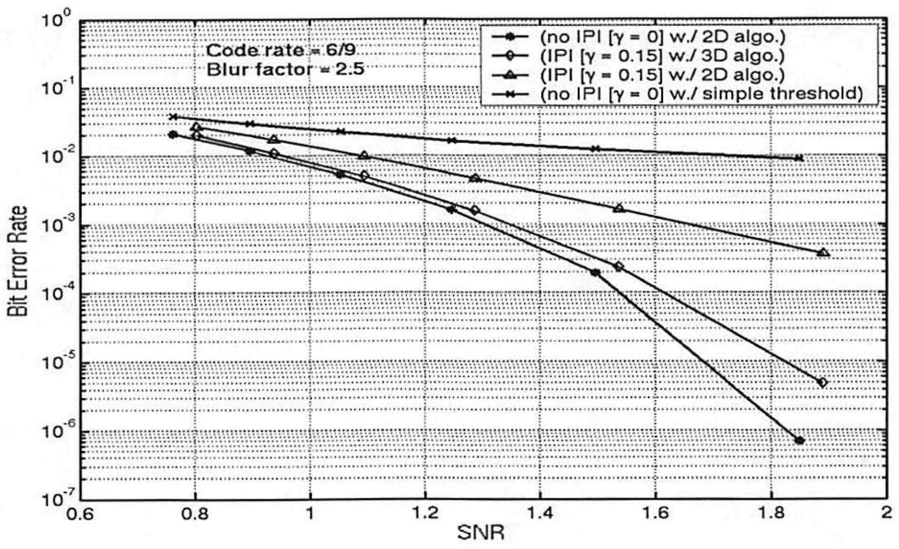


Figure 5.10: Bit-error-rate performance of modified iterative procedure as a function of SNR for different levels of inter-page interference.

The top curve in Fig. 5.10 shows the poor results for 2D detection of data that has ISI and *zero* IPI ($\gamma = 0$) using simple threshold decision only. The second and the third curves from the top show processing on data having ISI and IPI ($\gamma = 0.15$). There is an improvement using only a 2D version of the iterative algorithm operating on ISI within a single page. The third curve from the top shows how results with the full 3D iterative algorithm are even better than the two preceding cases. Its result is comparable to the performance of the 2D iterative algorithm (at the bottom) assuming data with zero IPI ($\gamma = 0$).

Chapter 6

Multi-Level (Grayscale) Encoding/Decoding in Imaging PODS

6.1 Multi-Level Encoding System

In most data storage systems, the recorded data is represented in a two-level binary format. The binary format has the advantage of simplicity and signal robustness. An alternate method to improve the capacity uses a multi-level representation in which the data is represented by non-binary symbols. It satisfies the capacity-improving requirement with lower spatial bit packing density requirements. Similar to many data storage systems, one implementation constraint is the lack of compatible storage media. For media with grayscale capability, multi-level recording is possible including the two-photon technology described. The capacity of a multi-level recording system is $\lceil \log_2 M \rceil$ times the binary recording system, where M is the number of levels in the recording system. Finally, grayscale recording allows us to record more data onto the same spatial location which allows the use of a PSF with lower blur factor while maintaining acceptable results and operations.

Grayscale representation uses M symbols (elements) in the alphabet set, where M is the number of discrete levels in the intensity range. Binary storage is a special case of the grayscale system with M equal to two. Normalizing the intensity signal over a range from zero to one, the symbols in the alphabet set are uniformly distributed along this range. Figure 6.1 shows the symbol separations and noise margins of binary (top) and grayscale systems for M=4 (bottom). The symbol separation is given by $1/(M-1)$. In general, a large symbol separation is preferred to allow proportionally large noise margin. Low noise margin indicates that the system can tolerate lower magnitude of noise leading to higher bit-error-rate. Therefore, the noise margin must be sufficiently large to ensure robustness of the system. Figure 6.1 shows that the grayscale system has significantly lower symbol separation and thus lower noise margin.

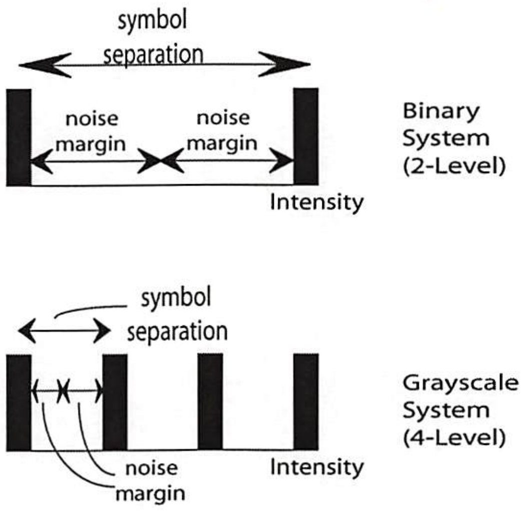


Figure 6.1: Noise margin and symbol separation in binary (Top) and grayscale (Bottom) recording.

Data symbols recorded on the media are either information symbols or known symbols. In the binary system, the information symbols are simply the user information, and the known symbols are defined by the system to help in the decoding process. Due to the effects of ISI and the difficulty of decoding known symbols, known symbols are intentionally set to *zero* to help counter ISI effects and avoid decoding errors as described in Chapter 2. However, in the grayscale system, the information symbol structure is more complex and it is difficult to decode based solely on the information symbols. Thus, additional decoding information is embedded in the known symbols to help improve the decoding accuracy. These known symbols carry parity information, thus we replace the term “known symbol” with “parity symbol”. The parity symbols themselves suffer the same interference effects as the information symbols and it is necessary to explore a reliable retrieval method to prevent a parity symbol detection error leading to error propagation problems.

6.2 Parity Symbol Encoding

In this section, we describe a low-complexity joint error-correcting/detection scheme using embedded decoding side information. To reliably decode the side information, we encode these side information symbols such that the probability of error is lowest. We define the parity symbols on the data page to have two states: odd and even, thus two distinct symbols are needed. For any grayscale system, the two symbols with the highest noise margins are symbol 0 and symbol $M-1$. We define symbol 0 to represent the even state and symbol $M-1$ to represent the odd state. The parity pixel is described by

$$d(x, y) = \left[\sum_{(x_i, y_i) \in \text{neighbor pixels}} \left(LSB(d(x_i, y_i))\right) \bmod 2 \right] * (M - 1), \qquad (6.1)$$

where $d(x_i, y_i)$ is binary representation of the user symbol value, represented with $\lceil \log_2 M \rceil$ bits. Equation (6.1) essentially sums up all the least significant bits (*LSB*) of all information symbols around parity pixels and assigns parity symbol values as either odd or even. The bracketed term in Eq. (6.1) can also be replaced with a parity check gate with an appropriate number of inputs.

6.3 Multi-Level Detection

Multi-level decoding follows the same general procedure as for the binary system using the algorithms described in chapter 3. The decoder adds an error detection procedure to the algorithm. Figure 6.2 shows the detection procedure for grayscale recording, which consists of two operations: decoding and error detection. In Fig. 6.2, the box labeled 1 represents the decoder described in chapter 3. The decoder outputs two types of intermediate outputs: decoded information symbols and decoded parity symbols as shown in box 2 of Fig. 6.2. The decoded information symbols are the estimated grayscale symbols and the parity symbols are encoded with parity values as described in the previous section. The decoder treats the parity symbols as ordinary grayscale symbols. We use the two intermediate outputs to independently calculate two parity symbol values in box 3 and 4 of Fig. 6.2. The results of the two calculations are then compared in box 5. A parity mismatch indicates information symbol errors which must be further corrected in box 6. The processed symbols are fed back to the decoder for future calculation. Detailed descriptions of each component in the algorithm are described in the following sections.

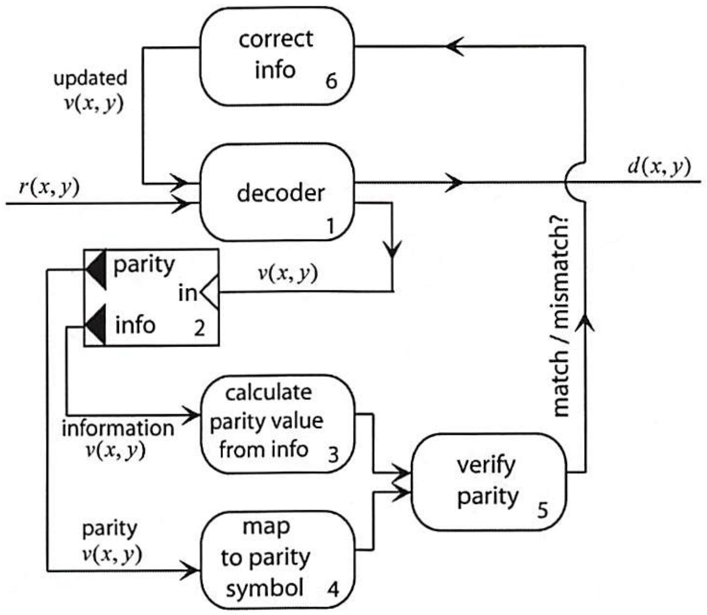


Figure 6.2: Detection procedure for grayscale recording.

6.3.1 Multi-Level Decoder

This section describes a multi-level decoder modified from the binary decoder described in chapter 3. The difference between the multi-level decoder and binary decoder is the number of adjacent alphabet symbols. The binary decoder has only one, while the multi-level decoder has two. Therefore, the decoder is modified to handle this scenario. The decoding procedure divides the decision region into multiple regions to detect each alphabet symbol. To detect a specific symbol *I*, we define a region which measured symbols have a high probability of being symbol *I*. We then process only the pixels whose received intensity value lies in that region. From this region, we detect the two adjacent symbols *I-1* and *I+1* using a modified update 0 and update 1 respectively. These modified update procedures change the update sensitivity values according to the symbol being detected. The lower symbol (Modified update 0) procedure is described by

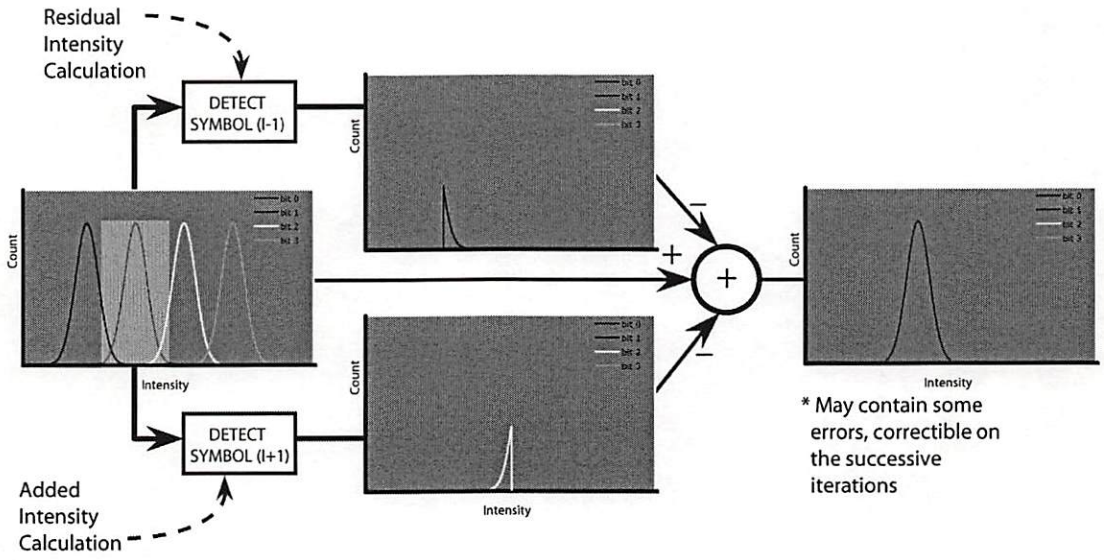
$$\begin{array}{ll} \text{if } \tilde{r}_r(x,y) < T_{lower} & , pixel \text{ is assigned to symbol } I-1 \\ \text{if } \tilde{r}_r(x,y) \geq T_{lower} & , pixel \text{ is assigned to symbol } I \end{array} , \qquad (6.2)$$

where T_{lower} is the lower symbol update sensitivity. The lower symbol update sensitivity value is set to be less than intensity level of an isolated symbol *I* with an additional margin for the effects of additive noise. In the same way, the upper symbol (Modified update 1) procedure is described by

$$\begin{array}{ll} \text{if } \tilde{r}_a(x,y) > T_{upper} & , pixel \text{ is assigned to symbol } I+1 \\ \text{if } \tilde{r}_a(x,y) \leq T_{upper} & , pixel \text{ is assigned to symbol } I \end{array} , \qquad (6.3)$$

where T_{upper} is the upper symbol update sensitivity. The upper symbol update sensitivity value is set to be greater than the intensity level of the sum of isolated symbol *I*, the maximum ISI effect and the additional margin for the effects of additive noise.

From these two procedures, some pixels are classified either as symbol *I-1* or *I+1*. The remaining pixels are most likely pixels having the symbol value *I*. The decoder then processes each of the remaining alphabet symbols with their corresponding lower and upper symbol update sensitivity values as described in (6.2) and (6.3). Figure 6.3 illustrates a specific example of 4-level system and the figure shows the decoding process for symbol 01.



Residual Intensity Calculation
DETECT SYMBOL (I-1)
Count
Intensity
DETECT SYMBOL (I+1)
Added Intensity Calculation
+
−
−
* May contain some errors, correctible on the successive iterations

Figure 6.3: Multi-level decoder example (M=4), detect symbol 01.

6.3.2 Parity Symbol Decoding from Information Symbols

From Fig. 6.2, two parity calculations are performed to verify the data correctness. Each calculation uses separate inputs to ensure independent results which can be impartially compared. In this section, we describe the first parity calculation as shown in box 3 of Fig. 6.2. In Fig. 6.2, box 2 separates the decoder output into two portions: information symbol and parity symbol. The two portions are extracted from the predefined locations of the information symbols and parity symbols used in modulation encoding. The parity symbol calculation in this section receives uses the information symbol portion and the procedure is similar to the parity symbol encoding procedure described in section 6.2. Equation (6.1) takes all relevant information symbols and calculates the corresponding parity symbols. These parity symbols are later compared with the result of the second parity calculation described on the next section.

6.3.3 Parity Symbol Decoding from Parity Symbols

The second parity calculation described in this section is shown in box 4 of Fig. 6.2. It uses the parity symbol portion of the decoder output. The procedure simply maps the symbols to their corresponding symbols using Table 6.1. Table 6.1 shows a few alphabet sets of *M*-level encoding, the symbols used for parity, and mapping of code symbols to even and odd parity. In this table, the pixels whose intensity are in the lower half of the intensity range are mapped to even parity and the pixels whose intensity are in the upper half of the intensity range are mapped to odd parity. These parity symbols are designed to have much higher noise margin than the information symbols, thus we assume that the mapped decoded parity symbols are correct with very high probability. For example, in 4-level encoding, the parity symbols are represented by symbol *zero* (even parity) and symbol 3 (odd parity). However, due to interference and noise, the decoded values of even parity can either be symbol 0 or symbol 1 and the decoded values of odd parity can either be symbol 2 or symbol 3.

Table 6.1 shows that pixels with decoded symbol value of 0 or 1 are mapped to symbol 0 and the pixels with decoded symbol value of 2 or 3 are mapped to symbol 3. The parity mapping step is an important step because the parity symbols are a crucial part in decoding the information symbols.

	Alphabet	**Parity values**	**Symbols mapping to even parity**	**Symbols mapping to odd parity**
4-level	{0,1,2,3}	{0,3}	{0,1}	{2,3}
8-level	{0,1,2,3,4,5,6,7}	{0,7}	{0,1,2,3}	{4,5,6,7}
M-level	$\{0,\ldots,M\text{-}1\}$	$\{0,M\text{-}1\}$	$\{0,\ldots,(M/2)-1\}$	$\{(M/2),\ldots,M\text{-}1\}$

Table 6.1: Decoded parity table for some grayscale encoding methods.

6.3.4 Information Symbol Correction

The results from sections 6.3.2 and 6.3.3 are the two independently calculated parity symbols of corresponding parity symbol locations. These two calculated parity values are compared to verify the information symbol correctness. A mismatch indicates that one or more symbols surrounding a given parity symbol are incorrect. From this indication, we locate and correct some incorrect information symbols. The symbols with mismatch parity are flagged and marked for correction. A candidate selection example for 4-level system is shown in Fig. 6.4. Table 6.2 shows possible error scenarios for the 4-level system example.

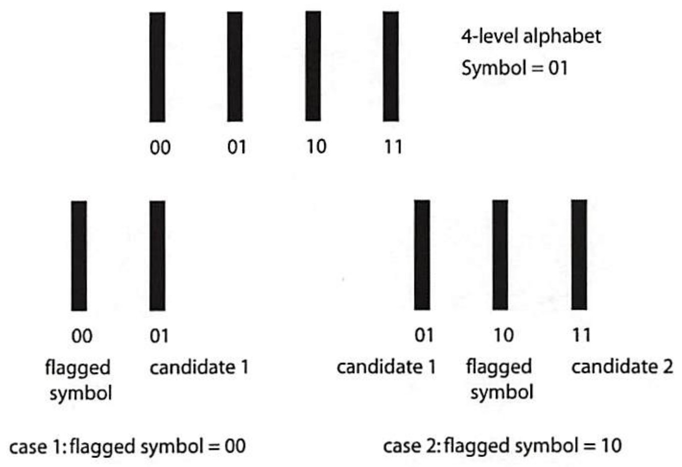


Figure 6.4: An example of candidate selection procedure.

Symbol	Error Symbol	Candidates
00	01	00,10
01	00	01
01	10	01,11
10	01	00,10
10	11	10
11	10	01,11

Table 6.2: Candidate selection example for grayscale encoding with M=4.

We consider a 4-level system example which has four alphabet symbols: 00, 01, 10, and 11 as shown in Fig. 6.4. Assume that the stored symbol is symbol 01, and that the symbol can be incorrectly classified either as symbols 00 or 10. In both cases, the classified symbol has a different LSB from the stored symbol. This property allows us to easily identify the correction candidates by choosing the candidates with a different LSB from the classified symbol. Depending on the classified symbol, the candidates are chosen from table 6.2. Next, we choose the correction symbol from the correction candidates by the calculation described by

$$\Delta d_c(x,y) = \left| r(x,y) - \left[c_c(x,y) - v(x,y) + \sum_{(x_i,y_i)\in \text{neighbors}} v(x_i,y_i) \right] * h(x,y) \right| \qquad (6.4)$$

and

$$v(x,y) = \begin{cases} c_1(x,y) & ;\text{if } \Delta d_1 \le \Delta d_2 \\ c_2(x,y) & ;\text{otherwise} \end{cases}, \qquad (6.5)$$

where $\Delta d_c(x,y)$ is difference between the received intensity and estimated intensity that replaces the flagged symbol, $v(x,y)$ with a new center symbol, $c_c(x,y)$, $c_1(x,y)$ is the candidate 1 value, and $c_2(x,y)$ is the candidate 2 value. The difference values resulting from the two candidates are compared and the candidate with the lower value is selected as more likely to be correct. The flagged symbol is then replaced with this candidate. The candidate symbols have twice the noise margin as the original symbol as shown Fig. 6.5. Therefore, the noise tolerance is much higher leading to lower error probability. A 4-level encoding example is shown Fig. 6.6.

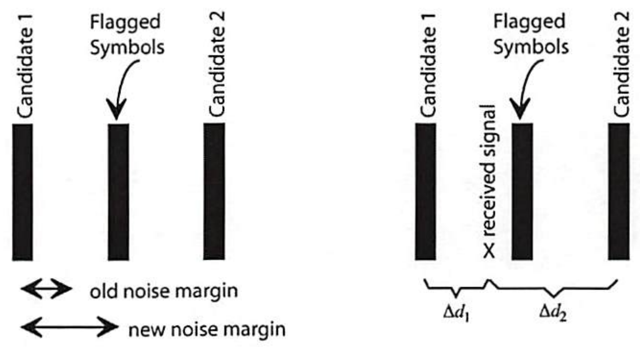


Figure 6.5: Candidate selection and symbol correction.

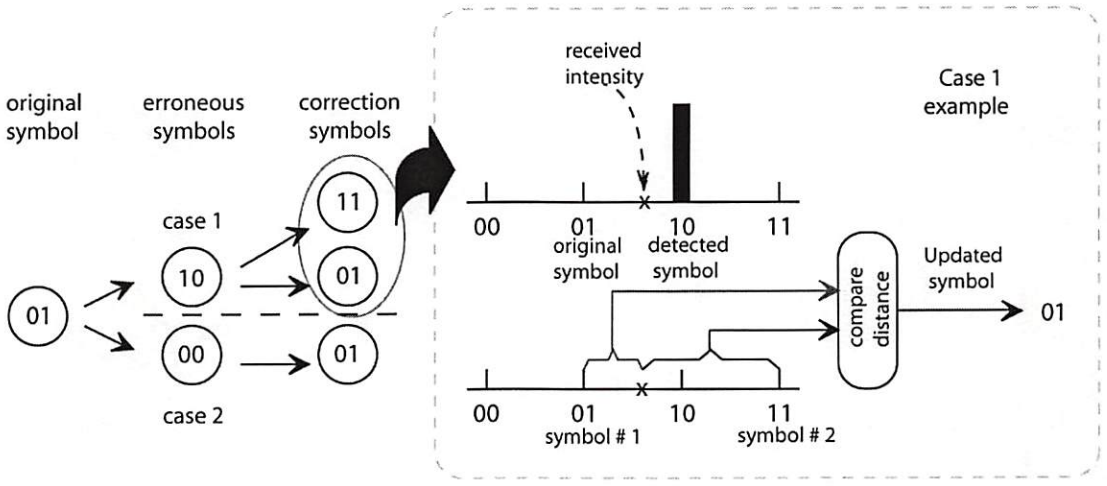


Figure 6.6: Information symbol correction example (M=4).

6.4 Rate 6/9 Implementation

We describe here a specific implementation using a modulation code with code rate 6/9 and pixels arranged in a hexagonal grid. This configuration allows every information symbol to be surrounded by exactly three parity symbols and the parity symbols to be surrounded by exactly six information symbols. Figure 6.7 shows a section of a data page, and illustrates the information/parity arrangement. This unique configuration allows a parity symbol to verify all six symbols surrounding it, thus an error in an information symbol will flag parity mismatch on three parity symbols. These three mismatch parity symbols form a triangle (triplet) surrounding the error symbol, which can be readily corrected as discussed in previous chapter. For simplicity, only isolated error symbols are considered. The simplified constraint relaxes the detection/correction rules greatly. There are more complex scenarios regarding even-numbered errors and more complex rules can be added to cover those situations.

Figure 6.7 shows an original data page and decoded data page, where the gray circles represent parity symbols and the white circles represent information symbols. In the decoded data page, the parity symbols are recomputed based on the decoded information symbols surrounding them and compared with the decoded parity symbols. Figure 6.7 also shows how the isolated error symbols (shown in light grey) generate three parity mismatches around them. Two adjacent symbols in error as shown in the lower right corner create a detection exception and are not detected by this simple algorithm. This algorithm uses a rule-based detection and the exceptions, can be easily detected by including additional detection rules. This rule-based algorithm can also be used to flag symbols as erasures, where erasure is a known error location. This allows some error correcting schemes that are capable of correcting erasures to easily correct them without wasting detection effort.

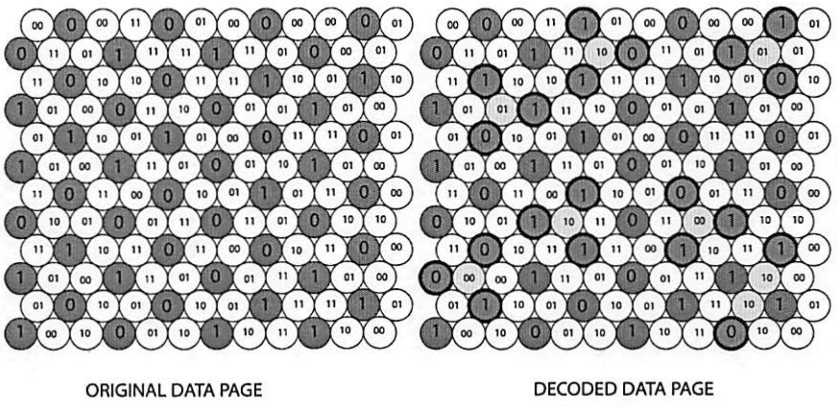


Figure 6.7: Example of parity detection (M=4).

6.5 Bit-Error-Rate Performance

Figure 6.8 shows the bit-error-rate performance of the parity check scheme for multi-level recording. The plot illustrates the result for a grayscale system with $M = 4$ with the rate 6/9 implementation. The system is corrupted by moderately severe inter-symbol interference and various levels of additive noise. The top curve shows the poor performance from simple threshold detection algorithm. The second and third curves from the top show the result from the same ISI level, and the bottom curve shows the result from a lower ISI level. The second curve implements the regular iterative algorithm, while the third curve includes the parity check feature in the detection. The third curve shows improvement of as much as one to two orders of magnitude over the second curve, and in some intervals, it equals the performance of the bottom curve with smaller ISI effect.

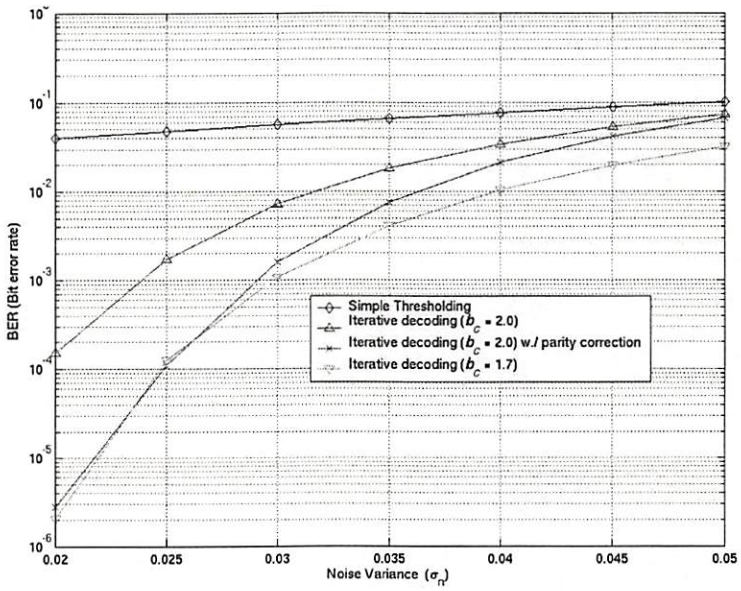


Figure 6.8: Bit-error-rate performance for multi-level (M=4) data storage using the parity check scheme.

Chapter 7

Conclusions

7.1 Summary

Page oriented optical data storage (PODS) has demonstrated the potential to store large amount of data and to retrieve the data with very high data transfer rate. These properties provide an attractive combination of capacity, transfer rate and access time. As in any data storage system, there are many types of interference associated with imaging PODS. The main data corruption sources are intra-page intersymbol interference (ISI), inter-page inter-symbol interference (IPI) and additive measurement noise.

The most common detection scheme used in data storage is threshold decision, which is based on a maximum *a priori* criterion that tries to minimize error probability. However, threshold decision schemes are not very effective against ISI, and yield relatively low bit-error-rate (BER) performance. There are various alternative powerful detection schemes. Some algorithms such as Viterbi algorithm provide higher BER performance and are optimal in the limit at the expense of higher computation complexity. PODS systems have massively high output data stream rates due to its parallelism, thus data bottlenecks can occur with high complexity algorithm. Thus the preferred approach is to incorporate a low complexity algorithm that provides an acceptable BER performance with a low complexity ECC scheme. This dissertation describes a low complexity detection algorithm for imaging PODS that mitigate the interference while maintaining acceptable data integrity.

Chapter 2 discusses background material on current data storage systems and signal processing techniques associated with them. The chapter also looks at some next generation data storage systems. Among next generation data storage systems, two-photon absorption technology is discussed, which is the model for our page-oriented optical data storage (PODS). Lastly, it discusses the imaging PODS including its systems arrangement, general detector configuration, system setup, noise model and channel model.

In chapter 3, the detailed description of the proposed iterative procedure is given. The procedure is a low complexity alternative to existing detection schemes that produces acceptable BER with data store in a multi-dimensional format. The algorithm essentially consists of two separable stages: classification and verification, and they operate in an iterative manner with preset stopping criteria. The low complexity factor makes it possible for the computing unit to be embedded into the sensor. The implementation can be carried out by the use of smart pixels.

Algorithm performance is discussed in chapter 4. Comparisons are provided in the basis of bit-error-rate. The bit-error-rate results are produced with different parameters such as code rate, blur factor, and noise level. Chapter 4 also discusses the computational complexity of the algorithm.

In chapter 5, inter-page inter-symbol interference (IPI) and modified algorithms to alleviate IPI are briefly discussed. The modified algorithm is adapted from the two-dimensional version described in chapter 3. Adaptation of the algorithm requires the data from multiple pages of data, causing a data dependency problem. To overcome this problem, a pipeline scheme is adopted. Similar to the original two-dimensional algorithm, the modified algorithm allows the capacity to be enhanced while maintaining an acceptable bit-error-rate (BER) performance, and when used in conjunction with low complexity error correcting codes (ECC), produces acceptable BER for the output bit stream. The overhead cost of the pipeline system is the additional buffer memory to process pages that are still in the pipeline.

7.2 Challenges and Future Work

Imaging page oriented optical data storage (PODS) is a promising technology that has become a topic of recent research interest. The system provides an attractive combination of capacity and speed. Yet, there are definite needs for improvements in various areas of the system. In the field of signal processing, modulation coding, detection/equalization, encryption, and error correction coding remain active as areas of interest for PODS system development. The following topics show promising development benefits.

7.2.1 Non Linear Quantization Methods for Reduced Fixed-Point Precision

In implementation of general DSP algorithms, an increase in the number of quantization levels leads to lower quantization error and more accurate processing results. However, in many power-constrained and time-sensitive systems, this generally means longer latency and more power consumption. Many soft-decision based algorithms usually require the data representation to be in a floating point or representation or fixed point binary representation with many levels. In our present implementation, the data is represented in 8-bit binary values. However, as many as three gate delays are wasted for each calculation because of the number of gate levels required to operate on 8-bit binary values. Different quantization methods may be devised to reduce the fixed-point precision while retaining minimum quantization errors and accuracy of the processing. It may be possible that bit budgets can be assigned to regions where the quantization errors are more pronounced.

7.2.2 Interleaving Scheme to Improve the Performance of the Algorithm

A problem of page-oriented data storage systems is burst error, in which large spatially connected groups of pixels are in error due to a local material defect. Most data storage systems suffer from inevitable media degradations such as: wear, scratches, and manufacturing defects (bubbles). This problem can be overcome by combination of multiple layers of error correcting coding and interleaving. Interleaving rearranges the spatial location of the data pixels such that the data pixels with continuous content are distributed over a much larger area. Statistically, this reduces the overall error probability and makes it easier to correct the data. This extension must be considered together with binary error-correction coding (ECC) that is employed at the outer layers of the storage process as shown in Fig. 2.9.

7.2.3 Detection Algorithm for Overcoming Mis-registration of Data Pixels and Detectors

Physical degradation of the recording media such as shrinkage and deformation can cause data pixels to be misaligned with the detector array. In addition, mechanical and optical system misalignment may occur. If these degradations are within an acceptable range, improved detection algorithms may be able to tolerate such problems and yields acceptable results.

7.2.4 Optimizing the Joint Modulation/Detection Scheme to Increase the Systems Performance

The overall system performance requires all components to work together efficiently. In most previous work, the development of modulation-coding and detection have always been separated. It may be possible to further improve the performance of the systems by jointly optimizing them.

7.2.5 Extension to the Application of Image Restoration

Application of the algorithm to image restoration may be possible since the format of storage is in a two-dimensional page, resembling an image. The binary image application is obvious. However, with development of multi-level extension, grayscale image restoration can also be performed.

References

[1] J. Ashley et al., “Holographic data storage,” IBM Journal of Research & Development 44(3), 341-368 (2000).

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "*Near Shannon limit error-correcting coding and decoding: Turbo codes*," in Proc. IEEE Int. Conf. Commun. 1993.

[3] M. Blaum, J. Bruck and A. Vardy, "*Interleaving schemes for multidimensional cluster errors*," IEEE Trans. Inform. Theory, vol.44, pp. 730-743, 1998. 12

[4] F.R. Boddeke, *Quantitative Fluorescence Microscopy*, pp. 56 (Delft University Press, 1999)

[5] Geoffrey W. Burr, Jonathan Ashley, Hans Coufal, et.al. , "Modulation coding for pixel-matched holographic data storage," Optics Letters 22, 639-641 (1997).

[6] G.W. Burr, G. Barking, H. Coufal, J.A. Hoffnagle, C.M. Jefferson, and M.A. Neifeld. “Gray-scale data pages for digital holographic data storage,” Optics letters, 23, 1218-1220 (1998)

[7] T. Etzion and A. Vardy, "*Two-dimensional interleaving schemes with repetitions: constructions and bounds*," IEEE Trans. Inform. Theory, vol.48, pp. 428-458, 2002.

[8] Lina Fagoonee, Abdolhosein Moinian, Bahram Honary, and Wim M. J. Coene, "Nonlinear signal-processing model for signal generation in multilevel two-dimensional optical storage," Optics letters 29, 385-387 (2004).

[9] G.D. Forney, “The Viterbi Algorithm,” Proceedings of the IEEE (1973).

[10] R. Ham, “Chemistry Online” November 2001, pp.4.

[11] Simon Haykins, *Adaptive filter theory* (Prentice-Hall, Inc. 1996)

[12] J.F. Heanue, M.C. Bashaw, and L. Hesselink, "*Volume Holographic Storage and Retrieval of Digital Data*," Science, Aug. 1994, pp. 749-752.

[13] N. Intharasombat and A.A. Sawchuk, "Capacity Improvement in Imaging Page-Oriented Optical Data Storage Using 2-D Modulation and Iterative Detection," *2003 Topical Meeting on Optical Data Storage Technical Digest,* Vancouver, BC, in OSA *Trends in Optics and Photonics (TOPS)* vol. 88, pp. 184-186, OSA, Washington, DC, 2003.

[14] N. Intharasombat and A.A. Sawchuk, "Iterative Detection for Imaging Page-Oriented Optical Data Storage," *2003 Topical Meeting on Optics in Computing,* OSA Technical Digest, (Optical Society of America, Washington, DC, 2003), pp. 173-175.

[15] N. Intharasombat and A.A. Sawchuk, "Modulation Encoding and Iterative Detection for Imaging Page-Oriented Optical Data Storage," Frontiers in Optics, Optical Society of America Annual Meeting, Tucson, AZ, October 2003; *OSA Annual Meeting Program 2003 Technical Digest*, paper WS1, (Optical Society of America, Washington, DC, 2003).

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

[17] C. Miller, B.R. Hunt, M.W. Marcellin, M.A. Neifeld, "Image restoration with the Viterbi algorithm", JOSA A, **17**(2) pp. 265 (2000)

[18] D.J.C. MacKay, "Good Error-Correcting Codes based on Very Sparse Matrices," IEEE Transactions on Information Theory, **45**(2) 399-431 (1999)

[19] M.A.Neifeld and M.McDonald, "*Error correction for increasing the usable capacity of photorefractive memories*," Opt. Lett., vol. 19, pp.1483, 1994. 13

[20] D.E. Pansatiankul and A.A. Sawchuk, "Fixed-Length Two-Dimensional Modulation Coding for Imaging Page-Oriented Optical Data Storage Systems," *Applied Optics*, vol. 43, pp. 275-290, (2003).

[21] D.E. Pansatiankul and A.A. Sawchuk, "Variable-Length Two-Dimensional Modulation Coding for Imaging Page-Oriented Optical Data Storage Systems," Appl. Opt. 43, 5319-5333 (2003).

[22] D.E. Pansatiankul and A.A. Sawchuk, "Bit-Error-Rate Performance Improvements of Three-Dimensional Modulation Codes for Imaging Page-

Oriented Optical Data Storage Systems," *2003 Topical Meeting on Optical Data Storage Technical Digest,* Vancouver, BC, in OSA *Trends in Optics and Photonics (TOPS)* vol. 88, pp. 102-104, OSA, Washington, DC, 2003.

[23] J.G. Proakis, *Digital Communication 4th edition* (McGraw-Hill, 2000)

[24] I.S. Reed, G. Solomon, "Polynomial codes over certain finite fields," J. SIAM, 8(2), 300-304, (1960).

[25] M. Rornagnoli, W.E. Moerner, F.M. Schellenberg, M.D. Levenson, and G.C. Bjorklund, "Beyong the bottleneck: submicrosecond hole burning in phthalocyanine, " Journal of Optical Social of America B, 1(3), 341-348 (1984)

[26] Leo Selavo, "Dynamic Data Encoding for Page Oriented Memories," University of Pittsburgh, Thesis (2004).

[27] Leo Selavo, Donald M. Chiarulli and Steven P. Levitan, "Real-time adaptive encoding for 3D optical memories," Three- and four-dimensional optical data storage conference, Proceedings of SPIE, 4459, (2001).

[28] C.E. Shannon, `*A mathematical theory of communication*', Bell System Technical Journal, 27, 379-423 and 623-656, (July and October 1948).

[29] Shelby, R. M.; Hoffnagle, J. A.; Burr, G. W.; Jefferson, C. M.; Bernal, M.-P.; Coufal, H.; Grygier, R. K.; Gunther, H.; Macfarlane, R. M.; Sincerbox, G. T. *Pixel--matched holographic data storage with megabit pages*. Opt. Lett. 1997, 22 (19), 1509--1511.

[30] M. Sudan, "Deocoding of Reed-Solomon Codes beyond the Error-Correction Bound," Journal of Complexity, **13**(1), 180-193 (1997)

[31] S.A. Vanstone, P.C. van Oorschot, *An introduction to error correcting codes with applications* (Kluwer academic publishers, 1989)

[32] A.J. Viterbi, "Error bounds in convolutional codes and an asymptotically optimal decoding algorithm", *IEEE Transactions on Information Theory*, 1967.

[33] A.J. Viterbi, J.K. Omura, *Principles of Digital Communication and Coding* (McGrawHill, New York, 1979)

[34] M.M. Wang, S.C. Esner, F.B. McCormick, I. Cokgor, A.S. Dvornikov, and P.M. Rentzepis, "Experimental characterization of a two-photon memory," Opt. Lett. 22, 558-560 (1997)

[35] John Watkinson, *The art of data recording* (Butterworth-Heinemann, 1994).

[36] S.B. Wicker, V.K. Bhargava, *Reed-Solomon Codes and Their Applications,* (Wiley-IEEE Press, 1999)

[37] Zining Wu, *Coding and Iterative Detection for Magnetic Recording Channel* (Kluwer academic publishers, 1999).