

766659

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

**TRANSMISSION OF 2-D IMAGES FOR TURBO
CODES**

**Ph.D. Thesis by
Kenan BÜYÜKATAK, M.Sc.
(504002209)**

Date of submission : 18 April 2005

Date of defence examination: 4 August 2005

Supervisor (Chairman): Assoc. Prof.Dr. Sedef KENT

Members of the Examining Committee Prof.Dr. Osman Nuri UÇAN (İÜ.)

Prof.Dr. Mahmut ÜN (İÜ.)

Prof.Dr.Tamer ÖLMEZ (İTÜ.)

Assoc. Prof.Dr. Işıl CELASUN (İTÜ.)

AUGUST 2005

ACKNOWLEDGEMENTS

I would like to thank Assoc.Prof.Dr. Sedef KENT, my advisor and “guide” for her continuous and constructive guidance during this thesis. She has been very patient in working with me without complaining even on the weekends. I am also very grateful to Prof.Dr. Osman Nuri UÇAN for providing me most valuable support and directing me to real-to-life problems, some of which are investigated in this thesis. I also would like to thank my friend and colleague, Captain Ersin GÖSE, for many stimulating discussions and also for his help in some of calculations.

Last but not the least, I would like to express my deep gratitude to my wife, Gülseri and my son, Alperen and daughter, Nazlı Sıla for providing me a working milieu with love and understanding.

August, 2005

Kenan BÜYÜKATAK

TABLE OF CONTENTS

Page No

LIST OF ABBREVIATIONS	v
LIST OF FIGURES	vi
LIST OF SYMBOLS	viii
SUMMARY	ix
ÖZET	xii
1. INTRODUCTION.....	1
1.1. Literature Review.....	2
2. TURBO CODES.....	10
2.1. Introduction.....	10
2.2. Encoders for TurboCodes.....	11
2.2.1. RSC Component Codes.....	12
2.2.2. Trellis Termination.....	15
2.2.3. Interleaving.....	15
2.2.4. Convolutional Code Representation.....	17
2.2.4.1. State Diagram Representation.....	17
2.2.4.2. Trellis Diagram Representation.....	18
2.2.5. Puncturing.....	18
2.3. Turbo Decoding.....	19
2.3.1. MAP Algorithm.....	21
2.3.1.1. Terminology.....	21
2.3.1.2. Formulation.....	22
2.3.1.3. Max-Log-MAP and Log-MAP Algorithms.....	27
3. TRANSMISSION AND EQUALIZATION OF THE IMAGES WITH TURBO CODES.....	33
3.1. Equalization with 1-D Adaptive Filters.....	33
3.1.1. Introduction.....	33
3.1.2. Implementation.....	37
3.1.2.1. System Model.....	38
3.1.2.2. Simulation Results.....	38
3.1.3. Evaluations.....	39
3.2. Adaptive Wiener-Turbo System.....	42
3.2.1. Introduction.....	42
3.2.2. System Model.....	43
3.2.2.1. Bit Plane Decomposition.....	44
3.2.2.2. Encoding the Bit Plane Slices.....	51
3.2.2.3. Wiener based 2D denoising Filtering.....	53
3.2.2.4. Decoding the Bit Plane Slices.....	60
3.2.3. Simulation Results.....	63
3.2.4. Evaluations.....	63
3.3. Iterative Cellular Image Processing -Turbo System.....	65
3.3.1. Introduction.....	65

3.3.2. System Model.....	66
3.3.3. Iterative Cellular Image Processing Algorithm.....	67
3.3.3.1. Introduction.....	67
3.3.3.2. Implementation.....	67
3.3.4. Simulation Results.....	70
3.3.5. Evaluations.....	70
4. COMPRESSION AND TRANSMISSION OF 2-D IMAGES WITH TURBO	
CODES.....	72
4.1. Bit Sliced Compression.....	73
4.1.1. Introduction.....	73
4.1.2. Method.....	74
4.1.3. Experimental Results.....	75
4.1.4. Evaluations.....	77
4.2. JPEG Compression.....	78
4.2.1. Introduction.....	78
4.2.2. Method.....	79
4.2.3. Experimental Results.....	84
4.2.4. Evaluations.....	88
4.3. Progressive Transmission and Compression of Turbo Coded	
Images.....	89
4.3.1. Introduction.....	89
4.3.2. Method.....	89
4.3.3. Advantages.....	90
4.3.4. System Model.....	91
4.3.4.1. Image Partitioner.....	91
4.3.4.2. Bit Plane Slicer.....	92
4.3.4.3. Turbo Encoder.....	92
4.3.4.4. Turbo Decoder.....	94
4.3.4.5. Sub-image Combiner.....	94
4.3.5. Simulation Results.....	96
4.3.6. Evaluations.....	97
4.4. Image-oriented Compression and Progressive Transmission.....	100
4.4.1. Introduction.....	100
4.4.2. Method.....	100
4.4.3. System Model.....	101
4.4.4. Mathematical evaluations.....	105
4.4.4.1. Example of a special case.....	108
4.4.5. Simulation Results.....	112
4.4.6. Evaluations.....	113
5. CONCLUSIONS AND DISCUSSIONS.....	114
REFERENCES.....	117
APPENDICES.....	124
BIOGRAPHY.....	150

LIST OF ABBREVIATIONS

1D, 2B	: One-Dimensional, İki Boyutlu
2D or 2-D	: Two-Dimensional
APP	: A posteriori Probability
AWGN	: Additive White Gaussian Noise
AW-TS	: Adaptive Wiener-Turbo System
BER	: Bit Error Rate
BPS	: Bit Plane Slicing
BPSK	: Binary Phase Shift Keying
BSCT	: Bit Sliced Compression Turbo
dB	: Decibell
CC	: Constituent Code, Convolutional Code
CSI	: Channel State Information
DCT	: Discrete Cosine Transform
DEC	: Decoder
DFT	: Discrete Fourier Transform
ENC	: Encoder
EXT	: Extrinsic
FEC	: Forward Error Correcting
FIR	: Finite Impulse Response
ICIPA	: Iterative Cellular Image Processing Algorithm
IC-TS	: Iterative Cellular Image Processing Turbo System
IDFT	: Inverse Discrete Fourier Transform
JPEG	: Joint Photographic Experts Group
LLR	: Logarithmic Likelihood Ratio
LMMSE	: Linear Minimum Mean Square Error
LMS	: Least Mean Square
LOG	: Logarithmic
LSB	: Least Significant Bit
LTI	: Linear Time Invariant
MAP	: Maximum a Posteriori Probability
MIMO	: Multi Input-Multi Output
MMSE	: Minimum Mean Square Error
MSB	: Most Significant Bit
NSC	: Non-recursive Systematic Convolutional
PTCTC	: Progressive Transmission Compression of Turbo Coded Image
RLS	: Recursive Least Square
RSC	: Recursive Systematic Convolutional
SISO	: Soft Input Soft Output
SNR	: Signal to Noise Ratio
SOVA	: Soft Output Viterbi Algorithm
TC	: Turbo Code
QAM	: Quadrature Amplitude Modulation
QPSK	: Quadrature Phase Shift Keying
WT	: Wiener-Turbo

LIST OF FIGURES

	<u>Page No</u>
Figure 1.1 : Block Diagram of Digital Communications System.....	2
Figure 2.1 : General Structure of Turbo Encode.....	11
Figure 2.2 : Example of Non-Systematic Convolutional Code.....	14
Figure 2.3 : Example of Recursive Systematic Convolutional Code.....	14
Figure 2.4 : Trellis termination strategy for RSC encoder.....	15
Figure 2.5 : Random Interleaving in Turbo Encoder.....	16
Figure 2.6 : The State Transitions (path) for Input Information Sequence.....	17
Figure 2.7 : The Trellis Diagram for Convolutional Code.....	18
Figure 2.8 : Puncturing Process.....	19
Figure 2.9 : General Turbo Decoder Structure.....	21
Figure 2.10 : Diagram of forward recursion for MAP algorithm.....	25
Figure 2.11 : Diagram of backward recursion for MAP algorithm.....	26
Figure 2.12 : Diagram of forward recursion for max-log-MAP algorithm.....	30
Figure 2.13 : Diagram of backward recursion for max-log-MAP algorithm.....	31
Figure 3.1 : General Adaptive Filter.....	33
Figure 3.2 : Simplified Error Function with Respect to the Filter Coefficients..	34
Figure 3.3 : Turbo Decoder.....	38
Figure 3.4 : Transmission of the Image.....	40
Figure 3.5 : Bit Error Rates for 0-7 dB SNR (1. iteration).....	41
Figure 3.6 : Bit Error Rates for 0-7 dB SNR (2. iteration).....	41
Figure 3.7 : Bit Error Rates for 0-7dB SNR (3. iteration).....	41
Figure 3.8 : AW-TS System Model.....	44
Figure 3.9 : The idea of N Plane Slicing.....	46
Figure 3.10a : Separating an Image into its Constituent Bit Planes for which N=8	47
Figure 3.10b : Separating an Image into its Constituent Bit Planes for which N=8	48
Figure 3.10c : Separating an Image into its Constituent Bit Planes for which N=8	49
Figure 3.11 : The idea of plane combining.....	50
Figure 3.12 : Reconstructed Image from the Various Combinations of Bit Planes.....	51
Figure 3.13 : Encoder of AW-TS System.....	52
Figure 3.14 : Decoder of AW-TS System.....	62
Figure 3.15 : IC-TS System Model.....	66
Figure 3.16 : Iterative-Cellular Image Processing Algorithm Block Diagram.....	68
Figure 3.17 : Neighborhood Relations (template) for m=3.....	68
Figure 4.1 : The effects of the various slice combinations for “cameraman.tiff” image.....	76
Figure 4.2 : BSCT System Model.....	77
Figure 4.3 :Example of JPEG compression/Decompression.....	83
Figure 4.4 :System model of JPEG-Turbo system.....	84
Figure 4.5 : Compressed, Turbo Processed and Restore images at SNR=2dB...	85
Figure 4.6 : Compressed, Turbo Processed and Restore images at SNR=4dB....	86
Figure 4.7 : Compressed, Turbo Processed and Restore images at SNR=6dB....	87

Figure 4.8	: PTCTC System Model.....	91
Figure 4.9	: Division the image into partitions.....	91
Figure 4.10	: Encoding Strategy of PTCTC System.....	93
Figure 4.11	: Encoder of PTCTC System.....	94
Figure 4.12	: The flow chart of PTCTC system.....	95
Figure 4.13	: Progressively Receiving Image.....	98
Figure 4.14	: Bit Error Rates of the Planes.....	99
Figure 4.15	: System Model.....	102
Figure 4.16	: Encoding Strategy of the Proposed system.....	103
Figure 4.17	: Flow Chart of the Whole System.....	105
Figure 4.18	: Masks position after 1st iteration.....	110
Figure 4.19	: Masks position after 2nd iteration.....	111
Figure 4.20	: Masks position after 3rd iteration.....	111
Figure 4.21	: Masks position after 4th iteration.....	112
Figure A.1	: Noisy 2D image and various simulation results for SNR=2dB.....	125
Figure A.2	: Noisy 2D image and various simulation results for SNR=1dB.....	126
Figure A.3	: Noisy 2D image and various simulation results for SNR=0dB.....	127
Figure A.4	: Noisy 2D image and various simulation results for SNR=-1dB.....	128
Figure A.5	: Noisy 2D image and various simulation results for SNR=-2dB.....	129
Figure A.6	: Noisy 2D image and various simulation results for SNR=-3dB.....	130
Figure A.7	: Bit Error Performance of conventional Turbo and AW-TS with various bit planes.....	131
Figure B.1	: Noisy 2D image and various simulation results for SNR=1dB.....	132
Figure B.2	: Noisy 2D image and various simulation results for SNR=0dB.....	133
Figure B.3	: Noisy 2D image and various simulation results for SNR= -1dB.....	134
Figure B.4	: Noisy 2D image and various simulation results for SNR= -2dB.....	135
Figure B.5	: Noisy 2D image and various simulation results for SNR= -3dB.....	136
Figure B.6	: Bit Error Performance of conventional Turbo and IC-TS with various bit planes.....	137
Figure C.1	: Reconstructed Image with various Bit Planes at SNR= 2dB.....	138
Figure C.2	: Reconstructed Image with various Bit Planes at SNR= 1dB.....	139
Figure C.3	: Reconstructed Image with various Bit Planes at SNR= 0dB.....	140
Figure C.4	: Reconstructed Image with various Bit Planes at SNR= -1dB.....	141
Figure C.5	: Reconstructed Image with various Bit Planes at SNR= -2dB.....	142
Figure C.6	: Reconstructed Image with various Bit Planes at SNR= -3dB.....	143
Figure D.1	: Reconstructed images for mask size M=25 and their (plane allocation) histograms.....	144
Figure D.2	: Reconstructed images for mask size M=10 and their (plane allocation) histograms.....	145
Figure D.3	: Reconstructed images for mask size M=10 and their (plane allocation) histograms.....	146
Figure D.4	: Reconstructed images for mask size M=25 and their (plane allocation) histograms.....	147
Figure D.5	: Reconstructed images for mask size M=25 and their (plane allocation) histograms.....	148
Figure D.6	: Reconstructed images for mask size M=10 and their (plane allocation) histograms.....	149

LIST OF SYMBOLS

C	: Channel Capacity
R	: Information Rate
$\mathbf{d}_k$	: Binary Input Data Sequence
$\mathbf{x}_k^p$	: Parity Data
$\mathbf{x}_k^s$	: Systematic Data
r	: Convolutional Encoder Rate
M	: Convolutional Encoder Memory
$S_{i,j}$	: The nodes, where j is the state and i is the time index.
$\mathbf{y}_k^s$	: Received Sequence of Parity Values
$\mathbf{y}_k^p$	: Received Sequence of Systematic Values
$\Lambda(d_k)$	: A Log-likelihood Representation of the Estimate of d_k
s_i	: State of the Trellis at the time i
$\mathbf{y}$	: Received Observation Vector
$\mathbf{z}_i$	: A-priori Information to the Decoder
m_i	: Message Bit associated with the state transistion s_i to s_{i+1}
x_i	: Output Bit associated with the state transition s_i to s_{i+1} .
$P[x]$	: Probability of x
$P[y/x]$	: Probability of y conditioned on x .
$\mathbf{x}(n)$	: The Input Signal
$\mathbf{d}(n)$	: The Reference Input
$\hat{\mathbf{d}}(n)$	: The Adaptive FIR Filter Output
$\mathbf{e}(n)$	: The Error Signal
J_n	: The Square Error
$H(w_1, w_2)$	: Spectrum of $h(n_1, n_2)$
$P(w_1, w_2)$	: Power Spectra of $H(w_1, w_2)$
m_g	: Noisy Image
$P_x^{\varepsilon}(w_1, w_2)$	: Spectrum Estimate for a Signal $x(n_1, n_2)$
$n_{i,j}$	: Gaussian Noise

TRANSMISSION OF 2-D IMAGES FOR TURBO CODES

SUMMARY

Error corrective coding has become a vital part of modern digital wireless systems, enabling reliable and efficient transmission to be achieved over noisy and fading channels. Over the past decade, turbo codes have been widely considered to be the most powerful error control code due to its practical importance. It has low decoding complexity and enhances the transmitted information's efficiency and accuracy with the rate close to the channel capacity.

This thesis investigates an application of binary turbo codes over image transmission problem in the noisy channel and aims to achieve the three objectives: "error correction", "progressive transmission" and "image oriented compression" (Binary turbo coding is preferable in low SNR case, due to symbol distances are far from each other).

First, we begin with the investigation of the transmission of the 2-D images via turbo codes. The transmission of the pictorial information over wireless channels are extremely sensitive to the bit errors and they severely degrades the quality of the image at the receiver. This necessitates the provision of protection methods against these possible channel effects.

Second, we continue to develop some methods that reduces the bandwidth requirements. Generally, high speed hardware enhancements are not likely to solve this problem and the need for rapid data transfer grows continuously.

Third, a search is also made for image oriented compression and progressive transmission. This proposed method is very usefull when frequent incremental visual inspections of images are needed. Especially in quick search, this method is attractive.

The organization of this work is as follows:

Chapter 1 is the introduction section, giving some basic concepts related to general communication system, Shannons theorem etc. We investigate the literature review of Turbo Codes in this section.

Chapter 2 gives the general information of Turbo coding/decoding concept. It can be seen that they have a very powerful error correcting capability, reasonable complexity, and flexibility in terms of accommodating different block lengths and code rates.

In Chapter 3, the problem of image transmission via Turbo codes is investigated.

First, A satellite test image with 256-gray level is transmitted frame by frame (i.e. 1-D transmission case). Every 8-bit pixel of the image is converted to BPSK signal, coded and transmitted over the communication channel. The channel is considered as FIR (and Linear-Time invariant, but the channel parameters are unknown) plus AWGN (A common model in satellite communication). That means, the magnitude and phase of data are distorted and corrupted additionally (at the same time) by the AWGN. To compensate the channel effect and predict the inverse channel coefficients are more difficult (due to AWGN) in this situation. LMS, RLS and Kalman based adaptive algorithms are implemented for equalization and the results showed that above 4dB SNR, the proposed system equalized the channel effects successively.

Second, the image is transmitted with its neighborhood properties (i.e. 2-D transmission case). For this purpose, the 2D image is sliced into its Bit Planes. This approach reveals using the binary turbo codes enabling to carry the neighborhood relations to the receiver. Two different systems AW-TS (Adaptive Wiener-Turbo System) and ICIPA-TS (Iterative Cellular Image Processing Algorithm-Turbo System) are proposed here and the results are discussed. By the help of these systems, we obtained 2dB additional SNR improvements compared to that of conventional (in the literature) turbo coding systems.

In Chapter 4, Some new efficient techniques are proposed which successfully achieves the objectives of “compression” and “progressive image transmission”. They provided high compression ratios, with high image quality as desired and enabled an efficient transmission with high Bit Error Rates.

- First, we achieved “Compression”.

The compressions are based “bit plane slicing” and “JPEG “approach. They are compared in terms of their compression ratios and image qualities at the receiver. It can be seen in this chapter that the fundamental advantage of bit plane based algorithm is that it can recover the image at very low SNR's, but JPEG based algorithm needs SNR to be high. On the other hand, it could achieve much greater compressions. So we concluded that, JPEG based compression and transmission would need considerably less transmission time. But, in the cases of BER is important, bit plane based algorithm should be preferred.

- Second, we achieved “progressive transmission”.

In progressive Transmission technique, the quality of the image could be controlled at each stage. This enables the user to receive a more refined image. According to these techniques;

- i) Users could determine the compression ratio before transmission depending on the fast inspection.
- ii) The Compression Ratio was determined image orientedly. The basic approach is to segment the image into regions and allocate the number of Bit Planes (resulting the compression) according to their DCT coefficients. If the

DCT magnitudes of subimages are high, more planes are allocated, meaning this region has a greater detail in. An image with lower details have less number of planes and may be more efficiently compressed. This method allowed a lossy image transfer with fast extraction of an image, while the bandwidth of communication channel is preserved.

- Third, we achieved the image oriented compression and progressive transmission. The procedure is based on the idea that only partial information from the original image is provided to the receiver at each step. The receiver will use the limited information and perform the enhancement. In the next step, additional information about the image will be sent to the receiver (like WWW browsers). The new data, together with data already on the receiver side, is used to build a new and more complete image at the receiver, achieving a better image representation. As the procedure is iterated, for more enhancement, the receiver gathers more image data so that eventually it is able to recreate the image in its original form.

Finally, in Chapter 5 we present some concluding remarks together with suggestions for future work.



TURBO KODLU 2-B GÖRÜNTÜLERİN İLETİLMESİ

ÖZET

Hata Düzeltme Kodlamaları, Gürültülü ve sönümlü ortamlarda güvenilir ve etkin haberleşmeyi mümkün kılan günümüz modern kablosuz iletişimin vazgeçilmez bir unsuru haline gelmiştir. Son yüzyıldan beri, Turbo Kodlama bu alanda en güçlü bir Hata Düzeltme aracı olarak görülmektedir. Düşük Kodlama karmaşıklığına sahip olacak şekilde bilginin, maksimum kanal kapasitesi hızına yakın, verimli ve etkin bir şekilde iletilebilmesi bu sayede mümkün olabilmektedir.

Bu tez, İkili Turbo Kodlamanın, gürültülü kanallarda, görüntü iletimine ilişkin bir uygulaması niteliğinde olup, üç amacın gerçekleştirilmesini hedeflemektedir. Bunlar, “Hata Düzeltme”, “İlerlemeli İletim” ve “Görüntü bağımlı sıkıştırma” dır. (İkili turbo kodlama, semboller arasındaki mesafelerinin uzak olması nedeniyle, işaret/gürültü oranı düşük durumlarda tercih edilir).

İlk olarak, 2 boyutlu görüntülerin Turbo Kodlama yoluyla iletilmesi araştırılmıştır. Kablosuz ortamda resim bilgisinin iletilmesinde meydana gelebilecek bit hataları, alıcıda elde edilecek resmin kalitesinde son derece önemli bir role sahiptir. Bu yüzden resim iletiminde kanal etkilerine karşı bazı koruma önlemlerinin kullanılması bir zorunluluktur.

İkinci olarak, iletim band genişliği ihtiyacını azaltmaya yönelik bazı metodlar geliştirilmiştir. Genellikle yüksek hızlı donanım kullanımı bu problemi çözmekte yeterli olmayıp yüksek veri hızı gereksinimi gün geçtikçe de artmaktadır.

Üçüncü olarak resim-ilişkili sıkıştırma ve ilerlemeli iletim olanakları araştırılmıştır. Önerilen bu metot da görüntünün her aşamada artırımı olarak iletimi söz konusudur. Özellikle hızlı taramaların önemli olduğu durumlarda bu metod ilgi çekici neticeler verir.

Tezin yapısı aşağıdaki gibidir:

Bölüm 1 de, haberleşme sistemleriyle ilgili temel bilgiler, Shannon Teoremi v.b. gibi genel bilgileri içeren giriş bölümü yer alır. Literatürde mevcut olan geleneksel Turbo Kodlamanın gelişim süreci bu bölümde verilmiştir.

Bölüm 2 de Turbo Kodlama/Kod çözme sistemine ilişkin temel ilkeler yer alır. Buradan, güçlü bir hata düzeltme aracı olan bu kodlamanın, farklı blok uzunluğu ve farklı kodlama hızları için makul bir yapı karmaşasına sahip olduğu anlaşılabılır.

Bölüm 3 de, Turbo Kodlama yoluyla görüntü iletimi problemi ele alınmıştır.

İlk olarak, 256 seviyede bir uydu görüntüsünün dizi dizi (1 Boyutlu olarak) iletimi düşünülmüştür. Resme ait 8 bitlik her piksel önce BPSK işaretlerine dönüştürülmüş, kodlanmış ve iletim kanalına verilmiştir. Kanal, sonlu darbe cevaplı (ve doğrusal-zamanla değişmeyen, fakat kanal parametrelerinin bilinmediğinin varsayıldığı) ve Toplamsal Beyaz Gürültünün de ilave olarak mevcut olduğu bir kanaldır. Yani veriler genlik ve faz bozulmasının yanında Beyaz gürültüye de maruz bırakılmaktadır. Kanal etkilerinin azaltmak ve ters kanal filtre katsayılarını bulmak, bu durumda (Beyaz gürültü etkisinden dolayı) daha da zorlaşmaktadır. Kanal dengelemesi LMS, RLS ve Kalman tabanlı algoritmalar ile gerçekleştirilmiş olup, sonuçlar 4 dB ve yukarısı için kanalın başarılı bir biçimde dengelendiğini göstermiştir.

İkinci olarak, resim komşuluk ilişkileri de dikkate alınarak (2 Boyutlu olarak) iletilmiştir. Bu amaçla, 2 boyutlu resim Bit Düzlemlerine ayrıştırılmıştır. Bu yaklaşım, 4 bitlik durum için 16 seviyede olan bir resmin hem İkili Turbo Kodlarıyla iletimini mümkün kılmış, hem de komşuluk ilişkilerini alıcıya taşımış, böylece kod çözme işleminden önce resim işleme algoritmalarını kullanabilmemize olanak sağlamıştır. Literatürde mevcut olmayan bu iki yaklaşımın beraber düşünülmesi ile AW-TS ve ICIPA-TS sistemleri oluşturulmuş ve mevcut geleneksel Turbo Kodlama yöntemlerine göre 2dB lik bir kazanç sağlanmıştır.

Bölüm 4 de, “sıkıştırma” ve “aşamalı görüntü iletimi”ni gerçekleştiren bir yöntem geliştirilmiştir. Bu yöntem, istenilen oranda yüksek kalitede bir sıkıştırma ve yüksek hata başarımlı iletim olanağı sağlar.

- İlk olarak “Sıkıştırma” işlemi gerçekleştirilmiştir..

Sıkıştırma, “Bit Düzlemleri” ve “JPEG” yaklaşımlarıyla ayrı ayrı gerçekleştirilmiştir. Bu iki yaklaşım, sıkıştırma oranları ve alıcıda elde edilen resim kalitesi anlamında kıyaslanmıştır. Bu bölümde görülür ki, Bit Düzlemi yaklaşımı iletim ile elde edilen en büyük avantaj, resmin alıcıda çok düşük SNR’lı durumlarda bile elde edilebilmesidir. Fakat JPEG tabanlı sıkıştırmalı iletimin tatminkar olabilmesi için SNR’ın daha yüksek oranda olması gerekir. Diğer yandan JPEG’li iletimde daha yüksek oranda bir sıkıştırma gerçekleştirilebildi. Bu da bize “iletim zamanı” açısından bu yöntemin diğerine göre daha avantajlı olduğu anlamına gelir. Ancak Bit Hata Oranının önemli olduğu durumlarda, “Bit Düzlemi sıkıştırması”na dayalı iletim tercih edilmelidir.

- İkinci olarak, “Aşamalı iletim” gerçekleştirilmiştir.

İlerlemeli Görüntü iletiminde, görüntü kalitesi ilerlemeli iletim tekniği ile her adımda kontrol edilebilir ve daha detaylı görüntü bilgilerini alabilir. Bu yöntem sayesinde;

- i) Hızlı tarama ihtiyacına bağlı olarak, iletimden önce sıkıştırma oranı önceden kullanıcı tarafından ayarlanabilir.

- ii) Sıkıştırma oranı resim ilintili olarak otomatik ayarlanabilir. Yöntem, resmi küçük alt resimlere bölmek ve her bir alt resme detay derecesine göre düzlem sayısı tahsis etmekten ibarettir. Düzlem sayısı (detay bilgisi) alt resimlerin DCT katsayıları ile ilişkilidir. Eğer alt resimlerin DCT katsayılarının genlik değerleri büyükse, o bölge fazla detaya sahiptir anlamına gelir ve o alt resme daha fazla düzlem tahsis edilir. Daha az detaya sahip alt resimler böylece daha az sayıda düzlemle gönderilerek sıkıştırma sağlanmış olur. Bu şekilde kanalın band genişliği değiştirilmeksizin kayıplı ama hızlı resim iletimi sağlanmış olur.
- Üçüncü olarak, Resim ilintili sıkıştırma ve aşamalı iletim gerçekleştirilmiştir. Yöntem, orijinal görüntünün kısmi bilgilerinin sırasıyla her bir adımda alıcıya aşamalı olarak gönderilmesi şeklindedir. Her bir aşamada gelen bu kısmi bilgiler görüntünün iyileştirilmesinde kullanılır. Böylece Her bir adımda verici tarafından görüntü hakkında ilave bilgiler gönderildikçe (WWW. Tarayıcılarında olduğu gibi) alıcıda, bir önceki adımda elde edilen bilgilerin de kullanılmasıyla, daha detaylı görüntü elde edilmiş olur. Bu işlem yinelendikçe alıcıda daha fazla bilgi toplanmış olur ve orijinal resme her bir adımda daha fazla yaklaşmış olur.

Son olarak Bölüm 5’de tezde elde edilen sonuçlar değerlendirilmiş ve bu doğrultuda gelecekte yapılabilecek çalışmalar için öneriler getirilmiştir.

1. INTRODUCTION

Error correcting codes, sometimes referred to as forward error correcting (FEC) codes or channel codes, have become an invaluable tool in ‘closing’ the link budget of wireless-based digital communications systems. That is, by allowing a system to operate at a lower signal to noise ratio than would otherwise be the case, a desired quality of service over a link can be achieved within the transmit power or antenna gain constraints of the system. This property of error correcting codes is often referred to as ‘power efficiency’. Figure 1.1 shows an error correcting code within a digital communications system. The channel encoder adds code bits to the transmission bit stream, based on the data bits at its input. These extra bits are used by the channel decoder at the receiver to correct errors introduced into the transmission stream by a noisy or fading channel. The disadvantages of error correcting codes are two-fold. Firstly, the injection of extra bits into the transmission stream has the effect that, if the original rate of transmission of ‘useful’ data bits is to remain the same, the symbol rate over the channel must be increased, thus increasing the bandwidth needed to transmit the signal. Frequently, bandwidth expansion is not an option in modern wireless systems, where frequency spectrum is often highly regulated and bandwidth is costly. If the bandwidth is strictly limited then the effective data rate is decreased by the inclusion of an error correcting code. This problem is particularly troublesome at a time when new technologies are evolving, such as, Digital Audio and Video Broadcasting, Wireless Local Loop and Bluetooth that propose high-speed data transmission over wireless channels. A second disadvantage of error correcting codes is that they add complexity to the design of a communications system. The decoder complexity can affect the development time, physical size, power consumption, memory requirements and total delay of a receiver.

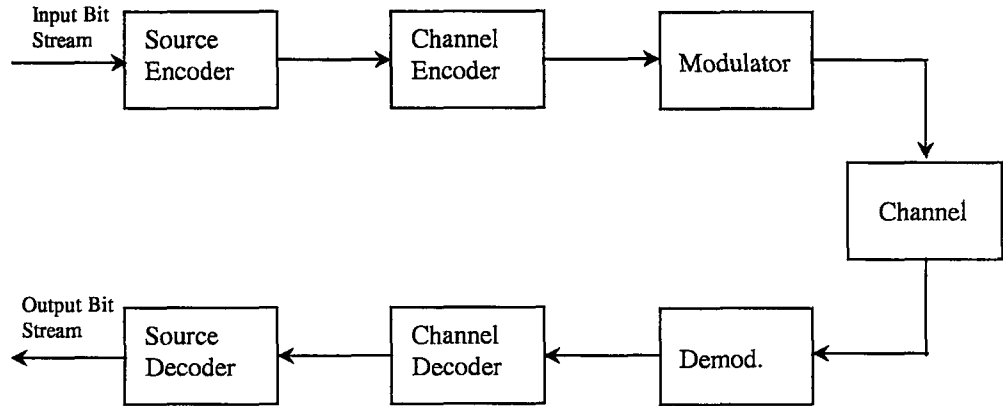


Figure 1.1: Block Diagram of Digital Communications System.

The fundamental approach to the problems of efficiency and reliability in communication systems is contained in the Noisy Channel Coding Theorem developed by Claude Shannon in 1948. This theorem states that it is possible to transmit data with arbitrarily low probability of error if the communication system operates below channel capacity and if the very large block length of data is transmitted. After Shannon determined the capacity of channels, we finally know of practical encoding and decoding algorithms that closely approach this limit. This remarkable feat was first achieved by the invention of turbo codes in Berrou *et al.*, (1993). Since then, turbo codes have essentially revolutionized the coding field and became one of the central research problems in recent years. While there has been a great deal of excellent theoretical work on turbo codes, it is fair to say practice still leads theory by a considerable margin.

1.1. Literature Review

The Shannon theorems for capacity and error probability are based on some information theoretic definitions. Recall from information theory Cover and Thomas (1991) that Entropy is defined as the average measure of randomness of a source of information and is expressed as

$$H(x) = E \left\{ \log \left(\frac{1}{P(x)} \right) \right\} = \sum_{i=0}^{M-1} P_i x_i \quad (1.1)$$

where source X chooses its elements from the set $\{x_0, \dots, x_{M-1}\}$ with probability of occurrence P_i , $i = 0, \dots, M-1$. Considering X as the channel input random variable

and Y as the channel output random variable, the mutual information between X and Y is defined as the measure of uncertainty remaining in X after observation of Y and defined as

$$I(X;Y) = H(X) - H(X|Y) \quad (1.2)$$

Channel capacity is defined as the maximum of $I(X; Y)$ with respect to the source probability distribution P_i , $i = 0, \dots, M-1$. So the channel capacity is a representation of the maximum capability of data transmission of the channel. The Shannon theorem for capacity describes that

- A given communication system has a maximum rate of information C known as the **channel capacity**.
- If the information rate R is less than C , then one can approach arbitrarily small error probabilities by using intelligent coding techniques.

To get lower error probabilities, the encoder has to work on longer blocks of signal data. This entails longer delays and higher computational requirements.

Thus, if the source symbols encoded into N -bit words, and $P_w(e)$ denotes the probability of decoding a binary N -bit word in error, then, if transmission is done with rate $R \leq C$, there exists a channel code such that $P_w(e) \rightarrow 0$ as $N \rightarrow \infty$ and the transmission may be accomplished without error in the presence of noise.

Unfortunately, Shannon's theorem is not a constructive proof -it merely states that such a coding method exists. The proof can therefore not be used to develop a coding method that reaches the channel capacity. The negation of this theorem is also true: if $R > C$, then errors cannot be avoided regardless of the coding technique used.

Although the theorem says that with large enough block length of data arbitrarily low error probability is achievable, the decoding complexity of the system increases exponentially with the block length. Turbo coding is an efficient method of increasing block length, while keeping the overall complexity linear. Turbo codes were originally introduced by Berrou *et al.*, (1993) as binary error correcting codes with a near Shannon performance (only 0.7 dB from Shannon limit) which are built from the parallel concatenation of two recursive systematic convolutional (RSC) codes employing powerful iterative decoding algorithm. In fact turbo codes are based on two concepts: Concatenated coding and iterative decoding. Iterative decoding is

actually the "turbo-principle", which is the real reason behind remarkable performance of turbo code.

The concatenation principle is based on the fact that the correction power of forward error correcting (FEC) codes increase with the length of the block of a block code, or the constraint length of a convolutional code, and approaches Shannon bound at very large block or constraint length. The fact that the decoding complexity increases exponentially with increase of the block length, leads to design of a long complex code from much shorter easily decodable component codes. Concatenation is to feed the output of one encoder to the input of another encoder. This concatenation can be done for more than two encoders. The first encoder is called the outer encoder and the last one is called the inner encoder. The resulting code is much more complex than each of the component codes. However, its decoding is easy since the decoders will be applied for each of the component codes in turn from the inner decoder to the outer decoder. The concatenation scheme will be completed with the application of interleaver between the outer and the inner codes to prevent error propagation between the component codes. Concatenation approach has been well known in coding community for long time as iterated codes (Elias, 1954), product codes (Hirasawa *et al.*, 1984), concatenated codes (Forney, 1966) and their generalized version (Hirasawa *et al.*, 1980).

In turbo codes, the same sequence of data is applied to two RSC encoders in parallel, with an interleaver (a pseudo random interleaver) between them. If a given data sequence happens to terminate the first encoder fast, it is likely that after interleaving it will not terminate the second encoder and as the result at least one of the encoders has a large Hamming distance, which satisfies the requirements for design of good FEC. More importantly, they use turbo-principle which is in fact the iterative decoding principle. It is based on the re-application of the decoded word information not only to the inner code, but also to the outer code and repeat the process as many times as necessary. The information that passes between decoders is a soft information and the decoder is a soft input-soft output decoder. The soft information is the log likelihood ratio (LLR) of each bit. The LLR is a measure of total information about a specific bit. This information comes from two sources. Some part of this information comes from the received data bits which is called the intrinsic information. The other part comes from another decoder when the code is decoded in one of the decoders. This part of information is called extrinsic information and it is this part which should be passed between decoders. The intrinsic information is already available to the next decoder and should not be passed through iterations. The coding community received turbo codes with dubiety at the beginning until the

results were reproduced by other researchers (Robertson, 1994; Divsalar and Pollara, 1995a). Since then, there have been extensive attempts on the code construction and weight distribution (Divsalar and Ellise, 1998; Dolinar and Divsalar, 1995), performance bounds (Benedetto and Montorsi, 1996; Duman and Salehi, 1999; Sason and Shamai, 2000), and specially on iterative decoding algorithm (Benedetto *et al.*, 1996c). Later on turbo principle turned out to be used in many detection and decoding problems such as serial concatenation (Benedetto *et al.*, 1998a; Benedetto *et al.*, 1998b; Narayanan and Stuber, 1999) and coded modulation. Coded modulation principle was introduced by Ungerboeck (1984). In Ungerboeck (1984), it is integrated a convolutional code with a bandwidth-efficient modulation scheme, and showed that significant coding gain can be achieved compared to the uncoded schemes without sacrificing data rate or bandwidth. The technique, called Trellis-Coded Modulation (TCM), attracted many interests for further research (Gose *et al.*, 2003). After introduction of turbo codes, turbo coded modulation schemes, combination of turbo codes and TCM to improve spectral efficiency of the transmission, have been brought to the picture of communication systems (Le Goff *et al.*, 1994; Robertson and Woertz, 1995; Robertson and Woertz, 1996; Robertson and Woertz, 1998) on the basis of Ungerboeck's TCM. The work of Le Goff *et al.* (1994) presented a rate $R = (m - \tilde{m}) / m$ turbo code in conjunction with M-ary Phase Shift Keying (PSK) or Quadrature Amplitude Modulation (QAM) schemes. The codes process $(m - \tilde{m})$ bits from the three parallel data streams of the turbo encoder, and after proper puncturing, map the codeword to a $2m$ -point modulation scheme. The codes are decoded by soft demodulation and turbo decoding. Turbo Trellis Coded Modulation (TTCM) in Benedetto *et al.* (1996b) consists of a parallel concatenation of two rate $m/(m + 1)$ codes which have m systematic bits and 1 parity bit generated by RSC encoders. The output of each RSC encoders are punctured to obtain the desired code rate. The applied modulation schemes are 8-PSK, 16-QAM, or 64-QAM. The performance of 1 dB of the Shannon limit is reported for both 2 bits/s/Hz and 4 bits/s/Hz bandwidth efficiencies.

The history of employing multiple transmit and multiple receive antennas or Multiple Input-Multiple Output (MIMO) systems goes back to 1995 and 1996 and the independent works of Telatar (1995) and Foschini and Gans (1998). They provided a measure of capacity for MIMO systems and showed that the achievable rates on fading channels will be increased greatly by the use of these systems if the channel parameters can be estimated at the receiver and if the propagation coefficients between pairs of transmitter/receiver antennas can be considered statistically independent. It has been shown that the assumption of independent fade coefficient will be correct if the antennas at each transmitter and receiver side are separated by a

few carrier wavelength. The major result of their work is that the capacity of a multi-antenna system far exceeds that of a single-antenna system in fading environment. In fact fading is not a disturbing phenomenon anymore and MIMO wireless channels are capable of increasing capacity as well as introducing diversity gains. The initial idea of combining coding and antenna diversity, the delay diversity scheme was proposed by Wittenben (1993). The scheme used two transmit antennas and transmitted the same information from both antennas simultaneously but with a delay of one symbol interval. In 1998, the innovation in wireless communication proposed by Tarokh *et al.* (1998a). They invented a new family of codes called space-time codes for transmission using multiple transmit antennas over Rayleigh or Rician wireless channels. This family of code is actually space-time Trellis code that combines signal processing at the receiver with coding technique appropriate to multiple transmit antennas (Naguib *et al.*, 2000). It is focused on merging antenna diversity with proper channel coding in order to employ both coding and antenna diversity. The coding is achievable through expansion of the channel signal in antenna space. Since then, lots of work have been done on how to design the coding and modulation schemes to improve the performance of the system. There are more mathematical analysis on Trellis space-time codes in Tarokh *et al.* (1999a). A new modern technology for high data rate wireless communication based on the use of concatenated space-time coded modulation with multiple transmit and/or multiple receive antennas was proposed in Naguib *et al.* (1998). A general construction of PSK space-time coded modulation is developed in Hammons and Gamal (2000). Some codes for space-time trellis coded modulation are derived in Baro *et al.* (2000). Space-time trellis codes can also be combined with turbo codes. All the above works are dealing with space-time Trellis codes. Space-time Trellis codes perform extremely well in slow fading environment and reach to the outage capacity (Telatar, 1995; Foschini and Gans, 1998). However, for the fixed number of transmit antennas, the decoding complexity of this scheme increases exponentially with transmission rate. Some efforts have been done in reducing the complexity of decoding using sphere decoding algorithm (Damen *et al.*, 2000). The algorithm of Damen *et al.* (2000) is applicable for integer constellations. In October 1998, Alamouti discovered a simple transmit diversity scheme using two transmit antennas (Tarokh *et al.*, 1999a). The scheme is much less complex than space-time Trellis codes but it has a loss in performance compared to space-time trellis codes due to the lack of coding gain of the scheme. The simplicity and still good performance of Alamouti's scheme was the motivation of the work by Tarokh, Jafarkhani, and Calderbank (Tarokh *et al.*, 1998a). The work is the generalization of the idea to introduce space-time block codes (Tarokh *et al.*, 1999b; Tarokh *et al.*, 1999c). They

introduced the theory of generalized orthogonal designs and constructed space-time block codes for any number of transmit antennas. However they showed that generalization of Alamouti's codes to more than two transmit antennas is not always possible. These codes have a very simple maximum-likelihood decoding algorithm which is only based on linear processing. They exploit the full diversity given by transmit and receive antennas. The approach of space-time block code has been the basis of the extensive research on design of high performance codes for wireless channels in the absence of channel state information. When the fading is fast or there is high mobility there is no time for estimation of channel coefficients. Therefore in many situations the receiver does not have the channel state information (CSI). Information-theoretic aspects of a mobile multiple antenna communication link in the absence of channel state information is analyzed in Marzetta and Hochwald (1999). It is shown in Marzetta and Hochwald (1999) that there is no point in making the number of transmitter antennas greater than the length of the coherence interval in systems without CSI. The transmitted signals that achieve capacity are mutually orthogonal with respect to time among the transmitter antennas. When the coherence interval becomes large compared with the number of transmitter antennas, the capacity approaches to the capacity of the case of known propagation coefficients. Using these results, Hochwald and Marzetta (2000) introduced unitary space-time modulation over Rayleigh flat fading channels, a category of space-time block codes which are well tailored for piecewise-constant fading models. Unitary space-time code uses the constellations of $T \times M$ space-time signals $\{\phi_l, l=1, \dots, L\}$, where T represents the coherence interval for which the fading coefficients are approximately constant and $M < T$ is the number of transmitter antennas. The challenge job is how to design unitary space-time constellations. A method of designing unitary space-time constellation is introduced in Hochwald et al. (2000). Unitary space-time modulation is a non differential approach to the problem of transmit diversity without channel information. The differential approach to the problem of transmit diversity with unknown channel information is the focus of Tarokh et al. (1998a), Tarokh and Jafarkhani (2000), Hochwald and Sweldens (2000), Clarkson et al. (2001), Hughes (2000a) and Hughes (2000b). In Tarokh et al. (1998b), two equivalent differential detection scheme for the simple transmit diversity of Tarokh et al. (1999a) are proposed. It can be thought of as a joint channel and data estimation. It suffers from error propagation because it uses the detected sequence at time $t-1$ for channel estimation and these estimates are used to detect the data at time t . A differential encoding and decoding method on the basis of Tarokh et al. (1999a) is proposed in Tarokh and Jafarkhani (2000). It is based on the orthogonal design of block codes. The encoding and decoding algorithm is simple and it has a 3 dB loss compared to

Tarokh et al. (1999a). The extension of the method to more than two transmit antennas is nontrivial and it uses the theory of generalized orthogonal design. Differential unitary space-time modulation is a modification of unitary space-time codes to work in the continuously changing fading environment which is proposed independently in Hochwald and Sweldens (2000) and Hughes (2000a). In Hughes (2000b), it is characterized all optimal differential space-time modulation techniques that fit within the framework of Hochwald and Sweldens (2000) and Hughes (2000a) as unitary group codes. The complexity of the maximum likelihood decoder for differential unitary space-time codes is exponential both in the number of antenna and the rate (Hochwald and Sweldens, 2000). An algorithm for fast decoding of the differential unitary codes is proposed in Clarkson et al. (2001). The complexity of this method is polynomial in the number of antennas and the rate, while its error performance is very close to the maximum likelihood algorithm. The branch of works on space-time codes without CSI is evaluated as a possible future research direction and is not the concern of this thesis. This thesis is focused on the case of perfect channel state information and deals with other aspects of space-time coding. Recently, there has been growing interest in combining all the above mentioned concepts in coding theory, including space-time codes, turbo principle, serial and parallel concatenation, and TCM to get more improvement in the performance of wireless communication systems. The codes in Lin and Blum (2000) employ a serial concatenation coding scheme with a space-time Trellis code as the outer code, a simple rate-1 recursive code (a differential code) as the inner code and an interleaver in between. The system has a rate $1/N$ space-time Trellis encoder (N transmit antennas), N individual block interleavers, and N separate rate-1 recursive code. The output of the inner encoders will be mapped into a set of signal constellation points by a modulator mapping. The decoder is made of N soft input-soft output inner decoders, N de-interleavers at the output of each inner decoder and the output of de-interleavers are fed to the soft input soft output outer decoder and the iterative decoding will be done between the set of inner decoders and outer decoder. It is shown that the system outperforms the normal space-time Trellis code for applications that can tolerate delay. In Lin and Blum (2001) the serial concatenation of the outer code (a block code) and an inner recursive space-time Trellis code is considered and the performance bounds are calculated. Turbo space-time coded modulation scheme is proposed in Cui and Haimovich (2001a). Cui and Haimovich applied the original form of turbo codes to space-time codes. In their work two systematic space-time trellis codes are concatenated in parallel and connected to $N = 2$ transmit antennas. At the receiver there may be M receive antennas. The decoder has an iterative structure and the outperformance of the approach in comparison to

the ordinary space-time Trellis code is due to the application of turbo principle. The performance analysis of the approach is given in Cui and Haimovich (2001b). The performance bounds in Cui and Haimovich (2001b) are based on the union bounds. Two scenarios are considered by Tajkovic. The first one similar to Cui and Haimovich (2001a) is the concatenation of two recursive space-time Trellis codes in parallel and sending the coded data through two transmit antennas. The second one is the serial concatenation of a convolutional code with a recursive space-time Trellis code with an interleaver between them. It is shown by simulation that the first structure outperforms Tarokh's about 1dB at frame error rate of 10^{-2} and the second one outperforms Tarokh's approach only at high SNRs. In Liew et al. (2000) serial concatenation of three different turbo codes with space-time block codes are considered and their performance are compared over Rayleigh fading channels. It is shown that concatenation of space-time block codes with turbo codes results a coding gain and the possible achieved gain depends on the turbo coding scheme, since different turbo codes with different modulations have different level of error protections for the bits. In Bahceci and Tolga (2001), the concatenation of turbo codes and unitary space-time codes is studied and a coding gain of 10 dB at a bit error rate of 10^{-5} is achieved compared to the uncoded unitary space-time codes. Gose *et al.*, (2003) investigated the performance of Turbo Coding for Multi-path channels. In 2004, Turbo code Design is investigated for block fading channels (Boutros *et al.*, 2004). In Rekh *et al.*, (2005) an analysis is made for the optimal choice in the size and structure of interleaver. optimal interleaver design.

2. TURBO CODES

2.1. Introduction

Turbo-codes have received considerable attention since their introduction in 1993. This is due to their powerful error correcting capability, reasonable complexity, and flexibility in terms of accommodating different block lengths and code rates.

In a simplified turbo code, there are two convolutional encoders in parallel. The information bits are scrambled before entering the second encoder. The codeword in a turbo code consists of the input bits - i.e. the code is systematic – followed by the parity check bits from the first encoder and then the parity bits from the second encoder, as depicted in Figure 2.1. The simplified turbo code block diagram in Figure 2.1 shows only two branches. In general, we can have multiple turbo encoders with more than two branches. The convolutional code at every branch is called the constituent code (CC). The CCs can have similar or different generator functions. We will concentrate on the usual configuration with two branches having the same CC.

The Turbo-code (TC) encoder considered here consists of two 8-state, rate 1/2 recursive systematic convolutional (RSC) encoders operating in parallel with the data bits, d_k , interleaved between the two RSC encoders. Without puncturing, the overall code rate is 1/3. Other code rates are obtained by puncturing the coded bits. Standard practice has been to only puncture the parity bits, x_k^p . However, it is known that a significant increase in Hamming distance can be achieved by also puncturing a small number of data bits.

Interleaving is a key component of Turbo-codes. An interleaver is a permutation that maps N input data sequence $d_1, d_2, \dots, d_N$ into the same sequence of data with a new order. If the input data sequence is $d = (d_1, d_2, \dots, d_N)$, then d^P is the permuted data sequence where P is the interleaving matrix with only one nonzero element equal to one in each column and row. Every interleaver has a corresponding de-interleaver that acts on the interleaved data sequence and puts them back into its original order. The de-interleaving matrix is simply the transpose matrix of interleaving matrix, i.e., P^T . Although interleavers can be defined and implemented in a number of different ways. In general, two interleaver types have been commonly investigated. They are

“random” interleaver and the so-called “S-random” or “spread” interleaver (Divsalar and Pollara 1995a; Divsalar and Pollara 1995b; Divsalar and Pollara 1995c). In Divsalar, Pollara (1995a) and Benedetto, Montorsi G. (1996). It was recognized early on that good spreading properties are desirable for both fast convergence and good distance properties. The details of Turbo coding performance, based interleaver design, can be found in Takeshita and Costello (2000).

2.2. Encoders for Turbo Codes

Figure 2.1 shows the fundamental diagram of a turbo encoder of rate 1/3. In the diagram x_k^s is the systematic bit, and x_{k1}^p , and x_{k2}^p are the parity check bits. The overall weight of the code is the sum of the weights of the x_k^s , x_{k1}^p , and x_{k2}^p sequences over a block. The decoder is a feedback decoder consists of two serially concatenated decoders with a de-interleaver in between which decodes the data iteratively.

The binary input data sequence is represented by $d_k = (d_1, \dots, d_n)$. The input sequence is passed into the input of a convolutional Encoder1 (Viterbi, 1971; Sklar, 1988), and a coded bit stream, x_{k1}^p is generated. The data sequence is then interleaved. That is, the bits are loaded into a matrix and read out in a way so as to spread the positions of the input bits. The bits are often read out in a pseudo-random manner. The interleaved data sequence is passed to a second convolutional Encoder2, and a second coded bit stream, x_{k2}^p is generated. The code sequence that is passed to the modulator for transmission is a multiplexed (and possibly punctured) stream consisting of systematic code bits and parity bits from both the first encoder x_{k1}^p and the second encoder x_{k2}^p .

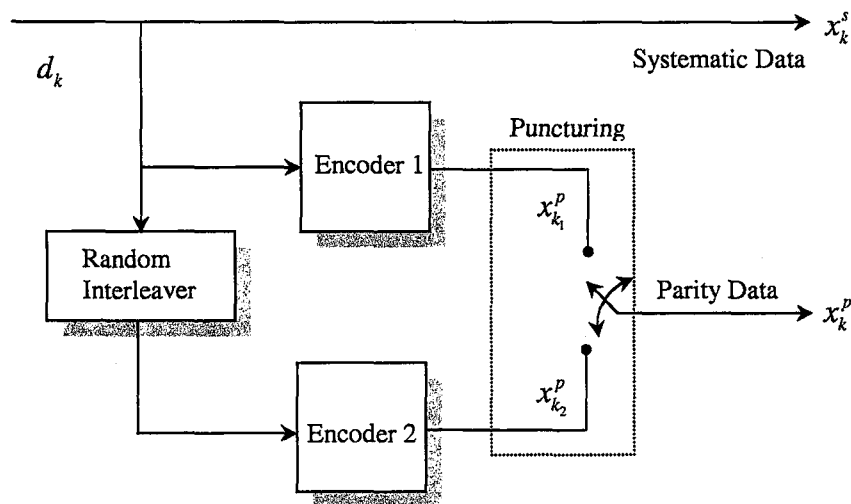


Figure 2.1: General Structure of Turbo Encoder

2.2.1. RSC Component Codes

Encoder1 and Encoder2 are *Recursive Systematic Convolutional* (RSC) codes - that is, convolutional codes which use feedback (they are 'recursive') and in which the uncoded data bits appear in the transmitted code bit sequence (they are 'systematic'). A simple RSC encoder is shown in Figure 2.2 along with a non-systematic (NSC) encoder, for comparison. The RSC encoder is rate $1/2$, with constraint length $K = 3$, and generator polynomial $G = \{g_1, g_2\} = \{7, 5\}$, where g_1 is the feedback connectivity and g_2 is the output connectivity, in octal notation. An RSC component encoder has two output sequences. One is the data sequence, $x_k^s = (x_1^s, \dots, x_N^s)$. The other is the $x_k^p = (x_1^p, \dots, x_N^p)$ parity sequence.

To understand why RSC component codes are used in a turbo code encoder, rather than the conventional NSC codes, it is necessary to first discuss the structure of error control codes. The minimum number of errors an error control code can correct is determined by the minimum Hamming distance of the code - the minimum number of bit positions in which any two codewords differ. The linear nature of turbo codes (at least, those using BPSK/QPSK modulations) means that the minimum Hamming distance of the code can be determined by comparing each possible codeword with the all-zeroes codeword. This process simplifies analysis of the code somewhat, and the minimum Hamming distance is then equal to the minimum *codeweight* (number of '1's) which occurs in any codeword.

The minimum Hamming distance tends to determine the Bit Error Rate (BER) performance of the code at high Signal to Noise Ratio (SNR) - the *asymptotic* performance. A high minimum Hamming distance results in a steep rate of fall of BER as SNR becomes large, whereas a low value results in a slow rate of fall. In the case of a turbo code, this rate of fall at high SNR is so slow, it is termed an *error floor*. At low SNR, however, codewords with codeweights larger than the minimum value must be considered. It is then that the overall *distance spectrum* of the code becomes important. This is the relationship between the codeweight and the number of codewords with that codeweight. Now, RSC codes have an infinite impulse response. That is, if a data sequence consisting of a '1' followed by a series of '0's enters the RSC encoder, a code sequence will be generated containing both ones and zeroes for as long as the subsequent data bits remain zero. This property means that RSC encoders will tend to generate high weight code sequences for groups of data bits spread far apart in the input sequence. An NSC code, however, will return to the

all-zeroes state after $K-1$ input zeros, where K is the constraint length of the encoder. The infinite impulse response property of RSC codes is complemented in turbo codes by the interleaver between component encoders. An interleaver is a device for permuting a sequence of bits (or symbols) at its input into an alternate sequence with a different ordering at the output. Turbo codes tend to make use of pseudo-random interleavers, whose role is to ensure that most groups of data bits which are close together when entering one RSC encoder are spread far apart before entering the other RSC encoder. The result is a composite codeword which will *often* have a high codeweight. Some further details of interleaving will be discussed in the next section.

This does not, however, mean that turbo codes tend to exhibit high minimum Hamming distances, and therefore good asymptotic performance. In fact, the opposite is usually true. The pseudo-random nature of most turbo code interleavers tends to result in a mapping such that a few combinations of input bit positions which cause low codeweight sequences in one RSC component code are permuted into combinations of positions which generate low codeweight sequences in the second RSC code. The result in such a case is a low composite codeweight. Such pseudo-random mappings often lead to turbo codes having a low minimum codeweight compared to, say, NSC-based convolutional codes, resulting in a marked error floor at high SNR. It is clear from this brief discussion that interleaver design is crucial in ensuring that a turbo code/interleaver combination has the lowest possible error floor.

Then, the significance of using RSC component codes, at low SNR, is the distance spectrum of the code as a whole becomes significant in determining BER performance, and that the combination of RSC code and pseudo-random interleaving produces codewords with higher codeweights most of the time. This results in there being *fewer* code-words with relatively low codeweights than a comparable convolutional code. It can be shown in constructing theoretical bounds for turbo codes that it is the *number* of codewords at each weight, as well as the actual codeweight, which determines the error probability of a code. The low multiplicity of low codeweight sequences associated with turbo codes, sometimes referred to as *spectral thinning*, leads to their good BER performance at low SNR. A full analysis of the turbo code characteristics described here can be found in Perez *et al.* (1996).

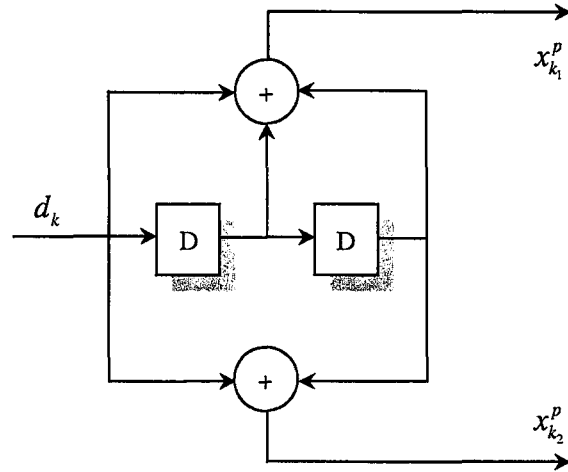


Figure 2.2: Example of Non-Systematic Convolutional Code

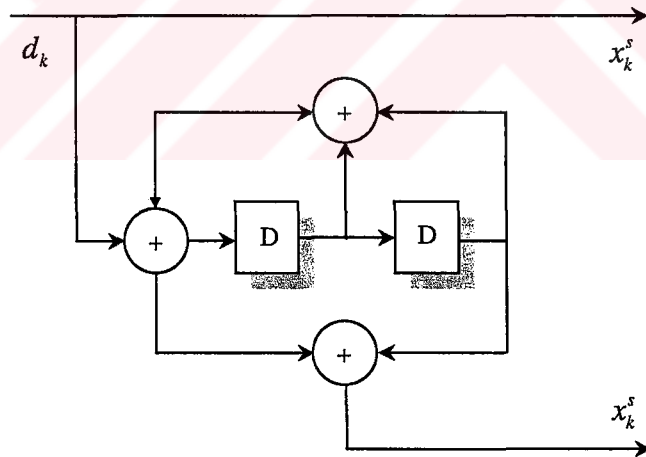


Figure 2.3: Example of Recursive Systematic Convolutional Code

2.2.2. Trellis Termination

For the conventional convolutional encoder, the trellis is terminated by inserting $m=K-1$ additional zero bits after the input sequence. These additional bits drive the conventional convolutional encoder to the all-zero state (trellis termination). However, this strategy is not possible for the RSC encoder due to the feedback. The additional termination bits for the RSC encoder depend on the state of the encoder and are very difficult to predict. Furthermore, even if the termination bits for one of the component encoders are found, the other component encoder may not be driven to the all-zero state with the same m termination bits due to the presence of the interleaver between the component encoders. Figure 2.4 shows a simple strategy which overcomes this problem.

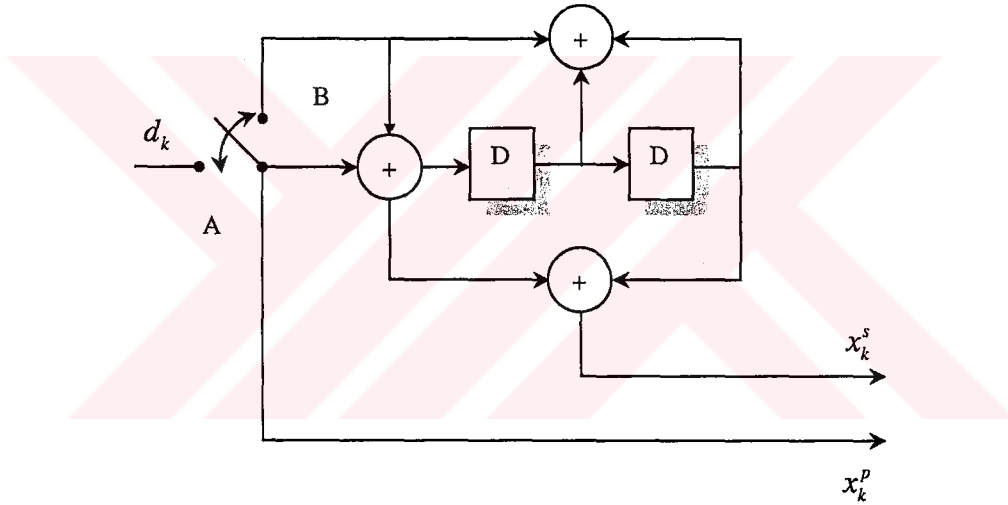


Figure 2.4: Trellis termination strategy for RSC encoder.

For encoding the input sequence, the switch is turned on to position A and for terminating the trellis, the switch is turned on to position B.

2.2.3. Interleaving

As shown in Figure 2.5, an interleaver is used to scramble the order of the input bits before feeding them into the second encoder (a deinterleaver at the receiver reverses the scrambling effect of the interleaver). Most error control codes work much better when errors in the received sequence are spread far apart. A familiar role of interleavers in communications is to break up error patterns in the overall code

structure. The interleaving between the two component codes then ensures that these bursts are adequately spread before entering the outer decoder. The trellis decoding nature of most convolutional codes means that uncorrected errors at the output of the decoder will tend to occur in bursts.

It is important to note that the interleaver within this work has a pseudorandom pattern, known to both the transmitter and receiver. It randomly generates mapping between input and output positions. That is, for an interleaver of length, the input sequence $i = 0, 1, \dots, (N-1)$ is scrambled according to a pseudo-random number set to form output sequence $\pi(i) = 0, 1, \dots, (N-1)$. The advantage of random interleavers is their ease of generation, but the disadvantage is that it is not possible to guarantee the minimum spreading properties and therefore BER performance of the interleaver. The interleaver serves to break up any recurrent error patterns between the two codes. Since both encoders receive the same input bits (but in different orders), the systematic nature (one of the output bits is the same as the input bit) of the encoders makes the systematic output of one encoder redundant to that of the other. Thus, the systematic output of the lower encoder is usually not transmitted, resulting in an overall code rate of $1/3$. This code rate may be increased through a selective removal of bits from the overall code output stream.

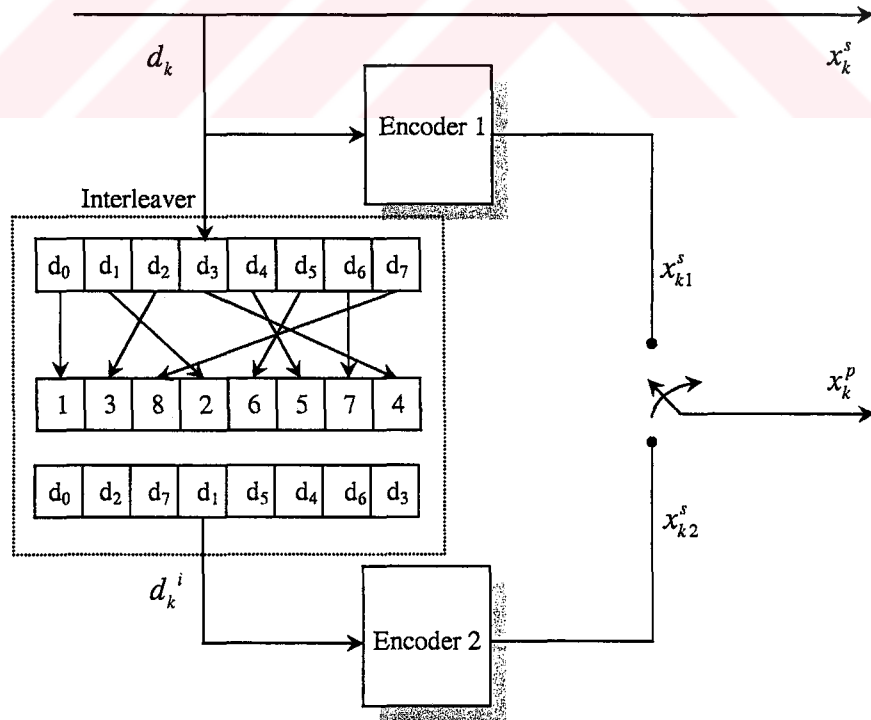


Figure 2.5: Random Interleaving in Turbo Encoder.

2.2.4. Convolutional Code Representation

Convolutional representation can be considered to be k -input, n output Finite Impulse Response filters. They can be thought as finite state machines and can be represented by state diagrams, graphs or trellises. The state of a convolutional encoder is determined by the contents of its shift registers. For a rate $r=1/n$ convolutional encoder with memory M , we have 2^M possible internal states. Once the encoder receives a new input bit, it can only make a transition to one of two other states, depending on whether the input bit is a 0 or 1. When the encoder makes a transition from one another state, a particular sequence of n bits is output.

2.2.4.1. State Diagram Representation

The operation of convolutional encoder can be specified by a state diagram which consists of some nodes corresponding to the possible internal states of encoder. The nodes are connected by branches and the path information between the states is labeled with the notation x/c , which represents input information bit x and output encoded bits c . It is customary to begin convolutional encoding from the all zero state. For example, the input information sequence $x=\{1011\}$ (begin from the all zero state) leads to the state transition sequence $s=\{10, 01, 10, 11\}$ and produces the output encoded sequence $c=\{11, 10, 00, 01\}$. Figure 2.6 shows the path taken through the state diagram for the given example.

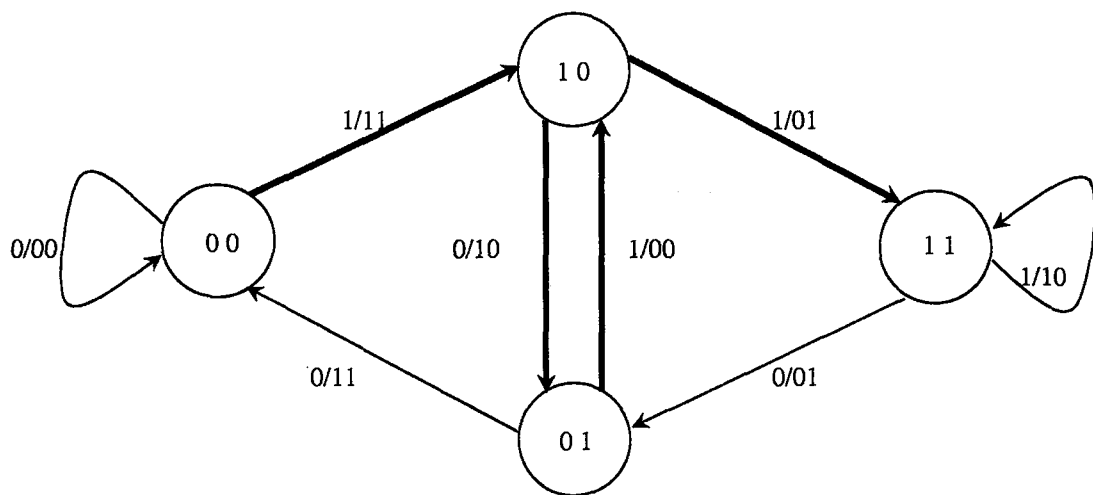


Figure 2.6: The state transitions (path) for input information sequence {1011}.

2.2.4.2. Trellis Diagram Representation

The state diagram can be expanded into a trellis diagram. It shows all possible state transitions at each time step. The nodes in a trellis diagram correspond not only to a particular state, but also a particular discrete time step. Frequently, a legend accompanies the trellis diagram to show the state transitions and the corresponding input and output bit mappings (x/c). This compact representation is very helpful for decoding convolutional codes. The nodes are labeled S_{ij} , where j is the state and i is the time index. Branches connect nodes at time i to nodes at time $i+1$. Figure 2.7 shows the trellis diagram for the encoder in Figure 2.3.

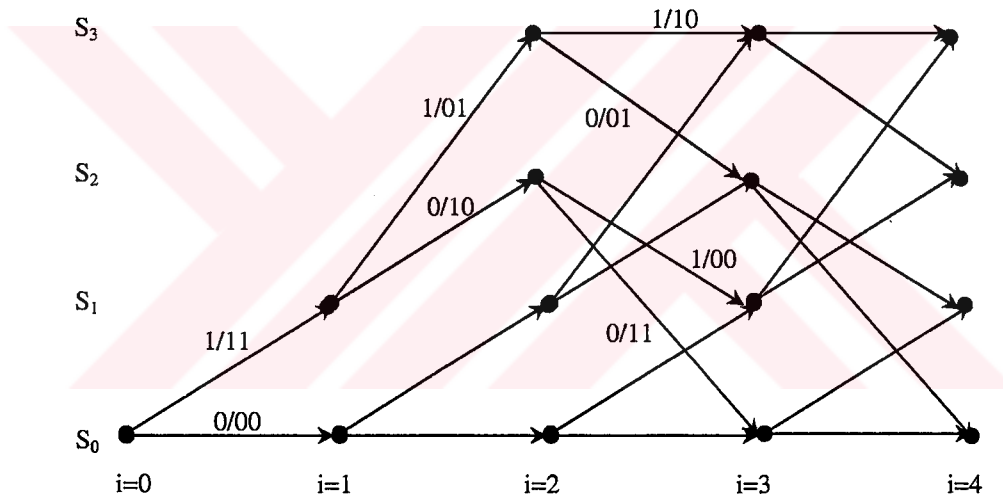


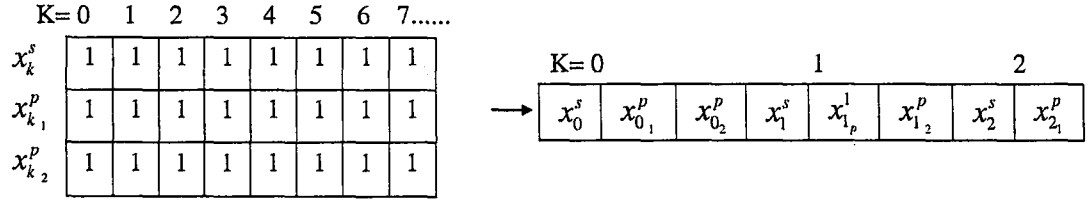
Figure 2.7: The Trellis Diagram For Convolutional Code.

2.2.5. Puncturing

Different code rates are achieved by puncturing the parity bit sequences x_{k1}^p and x_{k2}^p . Puncturing the data bit sequence x_k^s leads to a severe degradation in turbo code performance. Figure 2.8 illustrates the puncturing process. A number '1' in the tables represents a code bit that is included in the transmitted code bit sequence, and a number '0' represents a code bit that is excluded, or *punctured*. On the right of each table is the list of code bits which are included in the transmitted code sequence. In *a*), the code is unpunctured and is of code rate $1/3$ whereas in *b*), alternate parity bits

from each component encoder are punctured at each time interval k . The result is a $1/2$ rate code.

a. Rate 1/3



b. Rate 1/2

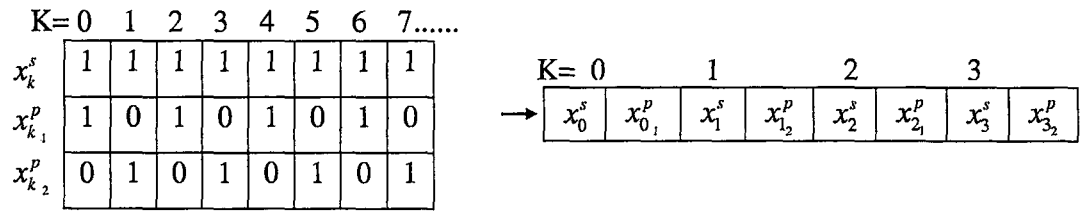


Figure 2.8: Puncturing Process

2.3. Turbo Decoding

The decoder works in an iterative way. Figure 2.9 shows a block diagram of a turbo decoder. Only one loop is performed at a time. In practice the number of iterations does not exceed 18, and in many cases 6 iterations can provide satisfactory performance (Berrou *et al.*, 1993). Actually, the term turbo codes is given for this iterative decoder scheme with reference to the turbo engine principle. The first decoder will decode the sequence and then pass the hard decision together with a reliability estimate of this decision to the next decoder. Now, the second decoder will have extra information for the decoding; a priori value together with the sequence.

The input to the turbo decoder is a sequence of received code values $R_k = \{y_k^s, y_k^p\}$ from the demodulator. The turbo decoder consists of two component decoders DEC_1 to decode sequences from ENC_1 and DEC_2 to decode sequences from ENC_2 . Each of these decoders is a Maximum A Posteriori (MAP) decoder, which will be described in the next section.

DEC_1 , takes as its input the received sequence of systematic values y_k^s and the received sequence of parity values $y_{k_1}^s$ belonging to the first encoder ENC_1 . The

output of DEC₁ is a sequence of soft estimates EXT₁ of the transmitted data bits d_k . EXT₁ is called *extrinsic* data, in that it does not contain any information which was given to DEC₁ by DEC₂. This information is interleaved, then passed to the second decoder DEC₂. The interleaver is identical to that in the encoder (Figure 2.5). DEC₂ takes as its input the (interleaved) systematic received values y_k^s and the sequence of received parity values y_{k2}^p from the second encoder ENC₂, along with the interleaved form of the extrinsic information EXT₁, provided by the first decoder DEC₂ outputs a set of values, which, when de-interleaved using an inverse form of the interleaver, constitute soft estimates EXT₂ of the transmitted data sequence d_k . This extrinsic data, formed without the aid of parity bits from the first code, is feedback to DEC₁.

This procedure is repeated in an iterative manner. The iterative decoding process adds greatly to the BER performance of turbo codes. For example, Berrou and Glavieux achieved BER = 10^{-5} at $\frac{E_b}{N_0}$ within 0.7 dB of the Shannon limit, using a rate 1/2 turbo

code and 18 decoding iterations. However, after several iterations, the two decoders' estimates of d_k will tend to converge. At this point, DEC₂ outputs a value $\Lambda(d_k)$; a log-likelihood representation of the estimate of d_k . This log likelihood value takes into account the probability of a transmitted '0' or '1' based on systematic information and parity information from both component codes. More negative values of $\Lambda(d_k)$ represent a strong likelihood that the transmitted bit was a '0' and more positive values represent a strong likelihood that a '1' was transmitted. $\Lambda(d_k)$ is de-interleaved so that its sequence coincides with that of the systematic and first parity streams. Then a simple threshold operation is performed on the result, to produce hard decision estimates, $\hat{d}_k$, for the transmitted bits. The decoding estimates EXT₁ and EXT₂, do not necessarily converge to a correct bit decision. If a set of corrupted code bits form a pair of error sequences that neither of the decoders is able to correct, then EXT₁ and EXT₂ may either diverge, or converge to an incorrect soft value. In the next three sections, we will look at the algorithms used in the turbo decoding process, within DEC₁ and DEC₂.

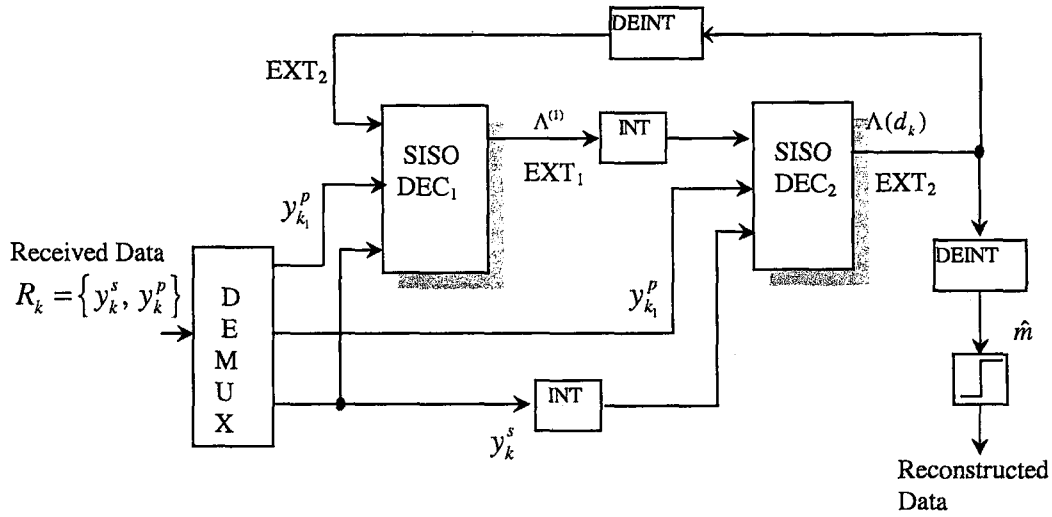


Figure 2.9: General Turbo Decoder Structure

2.3.1 MAP Algorithm

2.3.1.1 Terminology

- s_i represents the state of the trellis at the **time i**. It can be any of the 2^m possible states. ($m = \text{Constraint Length} - 1$).
- **L** represents the state of the decoder
- y and $\hat{y}$ represent the received observation vector. It is the output of the Demodulator. It consists of the sequence $y(0), y(1), y(2) \dots y(L-1)$, where L is the length of the Convolutional Coded block and the values of y are the soft-outputs of the previous block.
- z_i is the a-priori information to the decoder. This is the output of the previous channel decoder block. It is in the form of a Log Likelihood Ratio (LLR). These LLR's must be converted into proper values before processing.
- x_i is the message bit associated with the state transition s_i to s_{i+1} .
- x_i is the output bit associated with the state transition s_i to s_{i+1} .

$P[x]$ represents the probability of x and $P[y/x]$ is the probability of y conditioned on x .

2.3.1.2. Formulation

The MAP Algorithm calculates the a posteriori probability (APP) of each state transition, message bit or symbol produced by a markov process, given the noisy observation y . Once the APPs are calculated for all possible values of the desired quantity, a hard decision is made by taking the quantity with highest probability. When used for Turbo decoding, The MAP Algorithm calculates the APPs of message bits $P[m_i=1|y]$ and $P[m_i=0|y]$, which are then put into LLR form according to $\Lambda_i = \ln \frac{P_i[m_i=1|y]}{P_i[m_i=0|y]}$. The version of the MAP algorithm used for turbo decoding

does not make hard-decisions on the message bits until after the last decoder iteration. Before finding the APPs for the message bits, the MAP algorithm first finds the probability $P[s_i \rightarrow s_{i+1}|y]$ of each valid state transition given the noisy channel observation y . From the definition of conditional probability

$$P[s_i \rightarrow s_{i+1} | y] = \frac{P[s_i \rightarrow s_{i+1}, y]}{P[y]} \quad (2.1)$$

The properties of the Markov process can be used to partition the numarator as

$$P[s_i \rightarrow s_{i+1}, y] = \alpha(s_i) \gamma(s_i \rightarrow s_{i+1}) \beta(s_{i+1}),$$

$$\alpha(s_i) = P[s_i, (y_0, \dots, y_{i-1})]$$

$$\gamma(s_i \rightarrow s_{i+1}) = P[s_{i+1}, y_i | s_i] \quad (2.2)$$

$$\beta(s_{i+1}) = P[(y_{i+1}, \dots, y_{L-1}) | s_{i+1}].$$

The term $\gamma(s_i \rightarrow s_{i+1})$ is the branch metric associated with the transition $s_i \rightarrow s_{i+1}$, which can be expressed as

$$\gamma(s_i \rightarrow s_{i+1}) = P[s_{i+1} | s_i] P[y_i | s_i \rightarrow s_{i+1}] = P[m_i] P[y_i | x_i] \quad (2.3)$$

where m_i and x_i are the message and output associated with the state transition $s_i \rightarrow s_{i+1}$. Note that if the states s_i and s_{i+1} are not connected in the trellis diagram, then the above probability is zero.

The first term on the right hand side of the above equation is obtained using the a priori information z_i and

$$P[m_i] = \begin{cases} \frac{e^{z_i}}{1+e^{z_i}} & \text{for } m_i = 1 \\ \frac{1}{1+e^{z_i}} & \text{for } m_i = 0 \end{cases}, \quad (2.4)$$

$$\ln P[m_i] = z_i m_i - \ln(1 + e^{z_i}).$$

The second term is a function of the modulation and channel model. For flat fading channel,

$$P[y_i | x_i] = \frac{1}{\sqrt{\pi N_0 / E_s}} \exp \left\{ \frac{-E_s}{N_0} \sum_{q=0}^{n-1} [y_i^q - a_i^{(q)} (2x_i^{(q)} - 1)]^2 \right\}. \quad (2.5)$$

The probability $\alpha(s_i)$ can be found according to the forward recursion.

$$\alpha(s_i) = \sum_{s_{i-1}} \alpha(s_{i-1}) \gamma(s_{i-1} \rightarrow s_i) \quad (2.6)$$

Likewise, $\beta(s_i)$ can be found according to the backward recursion.

$$\beta(s_i) = \sum_{s_{i+1}} \beta(s_{i+1}) \gamma(s_i \rightarrow s_{i+1}) \quad (2.7)$$

Once the a posteriori probability of each state transition $P[s_i \rightarrow s_{i+1} | y]$ is found, the message bit probabilities can be found according to

$$P[m_i = 1 | y] = \sum_{s_i} P[s_i \rightarrow s_{i+1} | y] \quad (2.8)$$

and

$$P[m_i = 0 | y] = \sum_{s_0} P[s_i \rightarrow s_{i+1} | y]$$

Where $S_1 = \{s_i \rightarrow s_{i+1} : m_i = 1\}$ is the set of all state transitions associated with a message bit of 1, and $S_0 = \{s_i \rightarrow s_{i+1} : m_i = 0\}$ is the set of all state transitions associated with a message bit of 0. The log-likelihood ratio then becomes

$$\Lambda_i = \ln \frac{\sum_{S_1} \alpha(s_i) \gamma(s_i \rightarrow s_{i+1}) \beta(s_{i+1})}{\sum_{S_0} \alpha(s_i) \gamma(s_i \rightarrow s_{i+1}) \beta(s_{i+1})} \quad (2.9)$$

As a summary, the MAP algorithm can be divided into 3 major steps :

1. Forward Recursion : In this step the α 's are calculated in this step.
2. Backward Recursion : β 's are calculated in this step.
3. Log-Likelihood Ratios: The Log-Likelihood ratios of all the decoded message bits from $i = 0$ to $L-1$ are calculated in this step. These LLR's are then used by the next stage decoder in decoding the other convolutional code.

1. The following diagram depicts the flow for forward recursion.

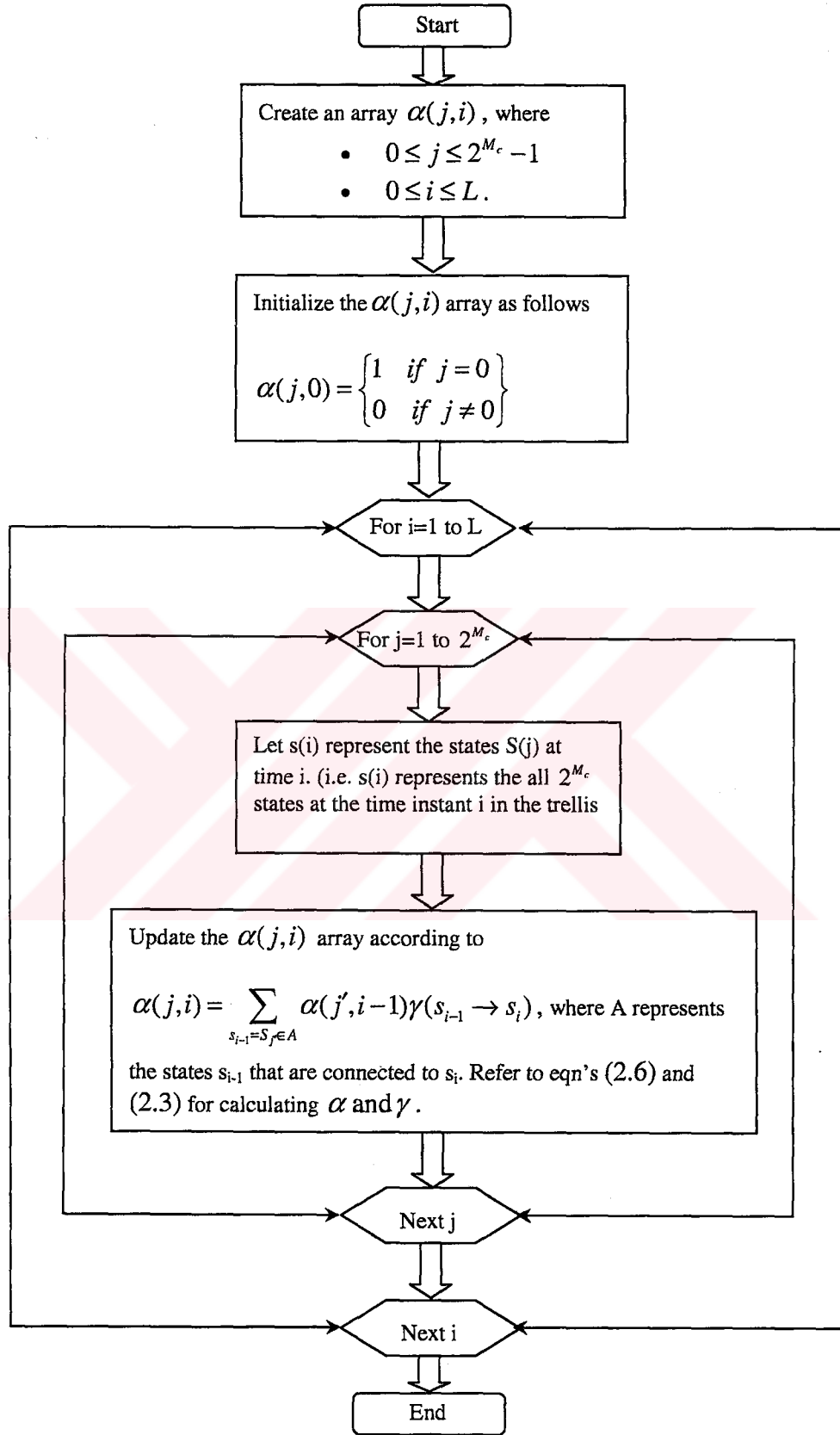


Figure 2.10: Diagram of Forward Recursion for MAP Algorithm

2. The following diagram depicts the flow for Backward recursion.

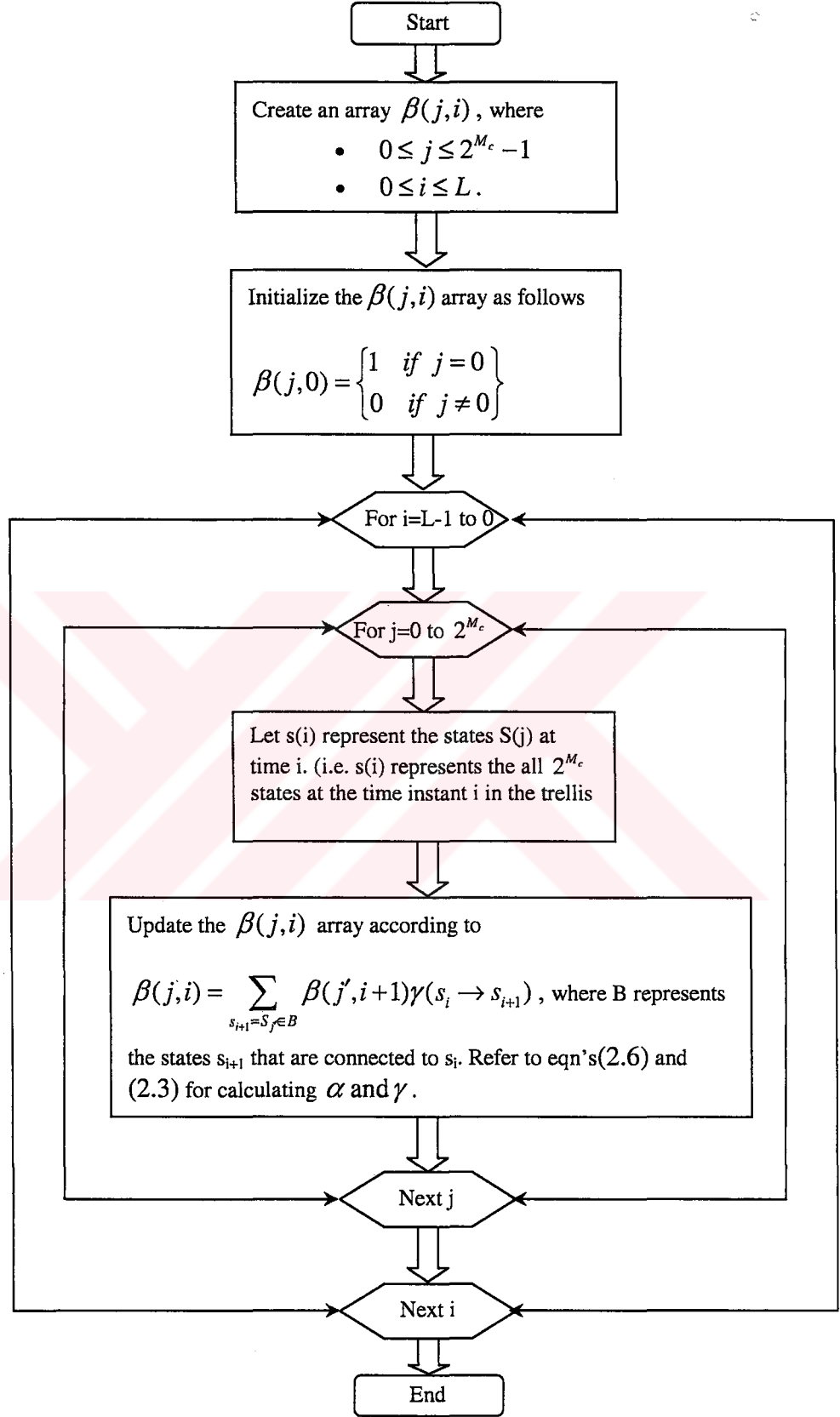


Figure 2.11: Diagram of Backward Recursion for MAP Algorithm

3. Computing the LLR's

This is a fairly straight-forward step and doesn't require any flowcharts.

For all $i = (0, 1, \dots, L-1)$, find out the LLR using the following formula :

$$\Lambda_i = \ln \frac{\left\{ \sum_{S_1} \alpha(j, i) + \gamma(s_i \rightarrow s_{i+1}) + \beta(j', i+1) \right\}}{\left\{ \sum_{S_0} \alpha(j, i) + \gamma(s_i \rightarrow s_{i+1}) + \beta(j', i+1) \right\}} \quad (2.10)$$

Where, S_1 represents the set of all state transitions for which the input message bit is "1" and S_0 represents all the state transitions for which the input message bit is "0". Both S_1 and S_0 are for time "i" only.

There ends the MAP algorithm.

2.3.1.3. Max-Log-MAP and Log-MAP Algorithms

The MAP algorithm is able to calculate precise estimates of the a posteriori probability of each message bit. However, the MAP algorithm suffers from two rather severe practical problems. First, it is computationally intensive, requiring 6×2^M multiplications per estimated bit and an equal number of additions. Second it is sensitive to the round-off errors that occur when representing numerical values with finite precision. These two problems can be alleviated by performing the entire algorithm in the log-domain, rather than waiting until the last step to take the logarithm of the likelihood ratio.

The main benefit of executing the algorithm in the log-domain is that multiplication becomes addition. However addition is not straight-forward when performed in the log domain. To illustrate how addition is performed in the log-domain, consider the Jacobian logarithm

$$\begin{aligned} \ln(e^x + e^y) &= \max(x, y) + \ln(1 + \exp\{-|y - x|\}) \\ &= \max(x, y) + f_c(|y - x|). \end{aligned} \quad (2.11)$$

This suggests that addition, when performed in the log-domain, becomes a maximization operation followed by a correction function $f_c\{\cdot\}$. Furthermore, it is

noted that when y and x are dissimilar, the correction function is close to zero. Thus, a reasonable approximation to the above is found by

$$\ln(e^x + e^y) \approx \max(x, y) \quad (2.12)$$

There are two classes of MAP algorithms that work in the log-domain--the max-log-MAP algorithm and the log-MAP algorithm. Both algorithms perform multiplications in the log-domain as additions. The difference between the two algorithms rests in how they compute addition in the log-domain. The max-log-MAP approximates addition in the log-domain as a maximization operation, per (2.12). The log-MAP algorithm is a refinement over the max-log-MAP algorithm, and computes addition in the log domain as a maximization followed by a correction term, per (2.11). Since the correction depends only on the difference between the two values, it can be stored as a one-dimensional look-up table. In Robertson et al. (1977), it is shown that excellent results can be achieved with as few as eight entries in the look-up table. In Fossorier et al. (1998), it is shown that the max-log-MAP algorithm is a generalization of the SOVA, and a modification to the SOVA is presented that makes it identical to the max-log-MAP algorithm.

Let $\bar{\gamma}(s_i \rightarrow s_{i+1})$ denote the natural logarithm of $\gamma(s_i \rightarrow s_{i+1})$

$$\begin{aligned} \bar{\gamma}(s_i \rightarrow s_{i+1}) &= \ln \gamma(s_i \rightarrow s_{i+1}) \\ &= \ln P[m_i] + \ln P[y_i | x_i]. \end{aligned} \quad (2.13)$$

For the flat-faded channel with BPSK modulation, the above becomes

$$\begin{aligned} \bar{\gamma}(s_i \rightarrow s_{i+1}) &= \ln P[m_s] - \frac{1}{2} \ln(\pi N_0 / E_s) + \left\{ \frac{-E_s}{N_0} \sum_{q=0}^{n-1} [y_i^q - a_i^{(q)}(2x_i^{(q)} - 1)]^2 \right\} \\ &= \lambda(s_i \rightarrow s_{i+1}), \end{aligned} \quad (2.14)$$

Now let

$$\begin{aligned} \bar{\alpha}(s_i) &= \ln \alpha(s_i) = \ln \left\{ \sum_{s_{i-1}} \exp[\bar{\alpha}(s_{i-1}) + \bar{\gamma}(s_{i-1} \rightarrow s_i)] \right\} \\ &= \max_{s_{i-1} \in A} [\bar{\alpha}(s_{i-1}) + \bar{\gamma}(s_{i-1} \rightarrow s_i)], \end{aligned} \quad (2.15)$$

where A is the set of states s_{i-1} that are connected to state s_i . For the max-log-MAP algorithm, $\max^*(x, y) = \max(x, y)$, while for the log-MAP algorithm $\max^*(x, y) = \max(x, y) + f_c(|y - x|)$. Note that for max-log-MAP, the iterative computation

of $\bar{\alpha}(s_i)$ follows exactly the same procedure as the Viterbi algorithm, and in fact the Viterbi algorithm is used to perform the forward recursion. For the log-MAP algorithm, $\bar{\alpha}(s_i)$ is computed using a modification of the Viterbi algorithm, where the correction term is added during each add compare select (ACS) operation. Thus the forward recursion of the log-MAP algorithm is performed using the “generalized” Viterbi algorithm.

Likewise $\bar{\beta}(s_i)$ denote the natural logarithm of $\beta(s_i)$. Then

$$\begin{aligned}\bar{\beta}(s_i) &= \ln \beta(s_i) = \ln \left\{ \sum_{s_{i+1}} \exp \left[\bar{\beta}(s_{i+1}) + \bar{\gamma}(s_i \rightarrow s_{i+1}) \right] \right\} \\ &= \max_{s_{i+1} \in B}^* \left[\bar{\beta}(s_{i+1}) + \bar{\gamma}(s_i \rightarrow s_{i+1}) \right],\end{aligned}\tag{2.16}$$

where B is the set of states s_{i+1} that are connected to state s_i . Again, this recursion can be executed using the Viterbi algorithm (max-log-MAP) or the generalized Viterbi algorithm (log-MAP). Note however, that the backwards recursion requires the Viterbi algorithm to be run in reverse (i.e. starting at the end of the trellis and proceeding to the beginning). Once $\bar{\alpha}(s_i)$ and $\bar{\beta}(s_i)$ have been found for all states in the trellis, the LLR can be found using

$$\begin{aligned}\Lambda_i &= \ln \left\{ \sum_{s_1} \exp \left[\bar{\alpha}(s_i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(s_{i+1}) \right] \right\} \\ &\quad - \ln \left\{ \sum_{s_0} \exp \left[\bar{\alpha}(s_i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(s_{i+1}) \right] \right\} \\ &= \max_{s_1}^* \left[\bar{\alpha}(s_i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(s_{i+1}) \right] \\ &\quad - \max_{s_0}^* \left[\bar{\alpha}(s_i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(s_{i+1}) \right].\end{aligned}\tag{2.17}$$

Thus, the max-log-MAP and log-MAP algorithm proceeds as follows:

1. For forward recursion.

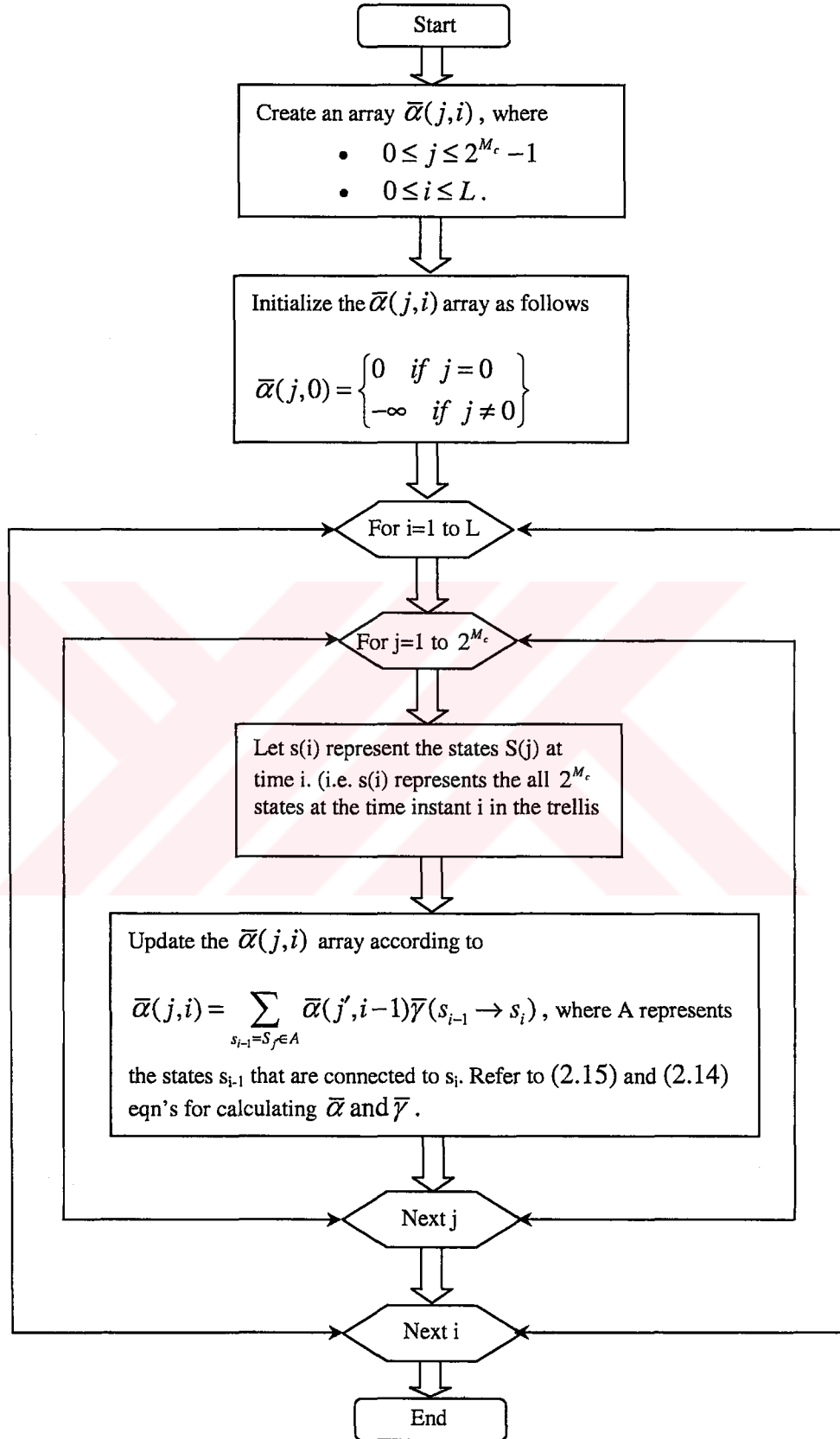


Figure 2.12: Diagram of Forward Recursion for max-log-MAP Algorithm

2. For backward recursion.

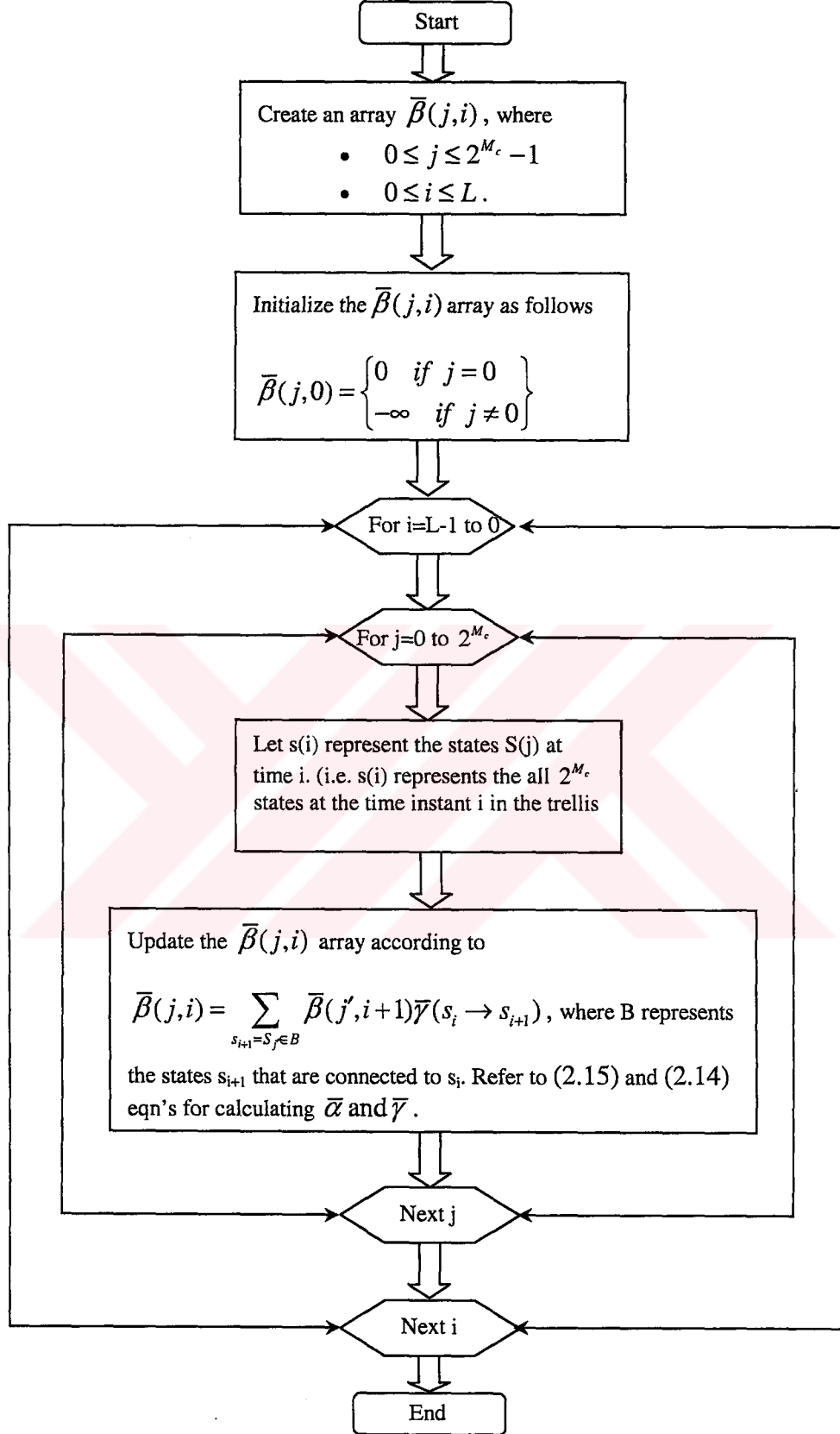


Figure 2.13: Diagram of Backward Recursion for max-log-MAP Algorithm

3. Computing the LLR's

For $i = (0, 1, \dots, L-1)$, find out the LLR using the following formula :

$$\Lambda_i = \max_{s_1}^* \sum_{S_1} \bar{\alpha}(j, i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(j', i+1) - \max_{s_0}^* \sum_{S_0} \bar{\alpha}(j, i) + \bar{\gamma}(s_i \rightarrow s_{i+1}) + \bar{\beta}(j', i+1) \quad (2.18)$$

Where, $S_1 = \{(s_i = S_j) \rightarrow (s_{i+1} = S_j) : m_i = 1\}$ represents the set of all state transitions for which the a message bit of "1" and $S_0 = \{(s_i = S_k) \rightarrow (s_{i+1} = S_k) : m_i = 0\}$ is the set of all the state transistions associated with a message bit is "0".

3. TRANSMISSION AND EQUALIZATION OF THE IMAGES WITH TURBO CODES

3.1 Equalization with 1-D Adaptive Filters

3.1.1 Introduction

Adaptive filters are digital filters capable of self adjustment. These filters can change in accordance to their input signals. An adaptive filter is used in applications that require differing filter characteristics in response to variable signal conditions.

The adaptive filter requires two inputs: The input signal $x(n)$, the reference input $d(n)$. An adaptive filter has the ability to update its coefficients. New coefficients are sent to the filter from a coefficient generator. The coefficient generator is an adaptive algorithm that modifies the coefficients in response to an incoming signal. The digital filter is typically a special type of finite impulse response (FIR) filter, but it can be also an infinite impulse response (IIR) or other type of filter. Adaptive filters have uses in a number of applications, including noise cancellation, linear prediction, adaptive signal enhancement, and adaptive control.

Figure 3.1 shows a block diagram of an adaptive filter model. The equalization system is modeled by an FIR filter with adjustable coefficients. The adaptive FIR filter output $\hat{d}(n)$ is compared with the input $d(n)$ to produce the error signal $e(n)$. The error signal represents the difference between the input and the model output. The error $e(n)$ is then used as the input to an adaptive control algorithm, which corrects the individual tap weights of the filter. This process is repeated through several iterations until the error signal $e(n)$ becomes sufficiently small. The resultant $\hat{d}(n)$ output now is the filtered version of $y(n)$.

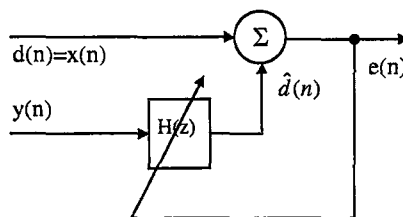


Figure 3.1: General Adaptive Filter

The estimated signal $\hat{d}(n)$ is expressed by the following equation.

$$\hat{d}(n) = \sum_{m=0}^{M-1} h_m(n)x(n-m) = \mathbf{h}(n)^T \mathbf{x}(n) \quad (3.1)$$

where

$$\mathbf{h}(n) = \begin{bmatrix} h_0(n) \\ h_1(n) \\ \vdots \\ h_{M-1}(n) \end{bmatrix} \quad \text{and} \quad \mathbf{x}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-M+1) \end{bmatrix} \quad (3.2)$$

The error which is the difference between the signal and the estimated signal needs to be minimized by choosing the best filter coefficients. The error squared we would like to minimize for each n is given by

$$J_n = e^2(n) = [d(n) - \mathbf{h}(n)^T \mathbf{x}(n)]^2. \quad (3.3)$$

Because we would like to minimize the square error, the term, *least mean square (LMS)* algorithm, is used. Assume that a simplified error function with respect to the filter coefficients is shown in Figure 3.2. The slope or the first derivative of J_n with respect to $\mathbf{h}(n)$ is given by

$$\frac{\partial J_n}{\partial \mathbf{h}(n)} = -2e(n)\mathbf{x}(n) \quad (3.4)$$

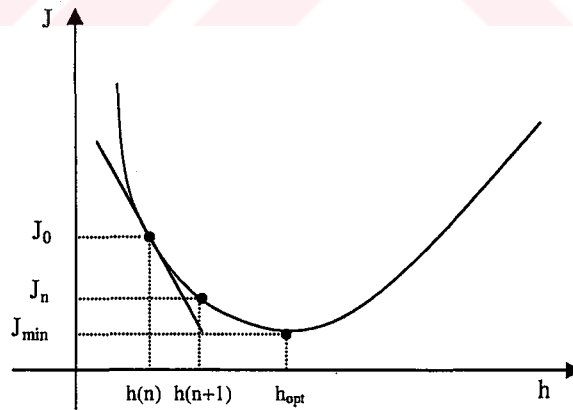


Figure 3.2: Simplified Error Function with Respect to the Filter Coefficients

Let h_{opt} the best or the optimum filter coefficients at which the squared error is minimized. If $h(n)$ is at the left-hand side of h_{opt} , the filter coefficients in the next step, $h(n+1)$, should move to the right (or positive direction). Note that the slope is negative in this case. On the other hand, if $h(n)$ is at the right-hand side of h_{opt} , the

filter coefficients in the next step, $\mathbf{h}(n+1)$, should move to the left (or negative direction). The slope is positive in this case. Thus, the filter coefficients in the next step should be given by

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu \left(-\frac{\partial J_n}{\partial \mathbf{h}(n)} \right) \quad (3.5)$$

When the slope is negative, positive correction will be made. And when the slope is positive, negative correction will be made. The μ is termed the adaptation parameter. If it is too large there will be too much fluctuation in the filter coefficients. If it is too small it will take a long time for the filter coefficients to converge. By plugging equation (3.1) into equation (3.2), one obtains the following adaptive rule:

$$\mathbf{h}(n+1) = \mathbf{h}(n) + 2\mu e(n)\mathbf{x}(n) \quad (3.6)$$

The summary of the adaptive algorithm is as follows:

- i) Initialize $\mathbf{h}(n)$ with any numbers. Random numbers are usually used.
- ii) Compute $e(n) = d(n) - \mathbf{h}(n)^T \mathbf{x}(n)$.
- iii) Update $\mathbf{h}(n+1) = \mathbf{h}(n) + 2\mu e(n)\mathbf{x}(n)$.
- iv) Go to ii) and continue.

Main problem with the LMS algorithm is that it takes a long time for the filter coefficients to converge because they are adjusted at an identical rate. Kalman filter or the RLS (Recursive Least Squares) adaptive algorithm improves this greatly at the expense of larger computational complexity.

A sum of squared error criterion can be alternatively expressed as

$$J_n = \sum_{i=0}^n b^{n-i} e^2(i) = e^2(n) + b e^2(n-1) + \dots + b^n e^2(0) = \sum_{i=0}^n b^{n-i} [d(i) - \mathbf{x}(i)^T \mathbf{h}(n)]^2 \quad (3.7)$$

where b is termed the forgetting factor ($0 < b < 1$). To minimize the error criterion one needs to set to zero the derivative of it with respect to the filter coefficients:

$$\frac{\partial J_n}{\partial \mathbf{h}(n)} = 2 \sum_{i=0}^n b^{n-i} [-\mathbf{x}(i)] [d(i) - \mathbf{x}(i)^T \mathbf{h}(n)] = 0 \quad (3.8)$$

$$\mathbf{R}(n)\mathbf{h}(n) = \mathbf{p}(n) \quad (3.9)$$

The solution is:

$$\mathbf{h}(n) = \mathbf{R}(n)^{-1} \mathbf{p}(n) \quad (3.10)$$

We would like to have a recursive solution. One can show that recursive expressions of the $M \times M$ matrix $\mathbf{R}(n)$ and the $M \times 1$ vector $\mathbf{p}(n)$ are given, respectively, by

$$\begin{aligned} \mathbf{R}(n) &= b\mathbf{R}(n-1) + \mathbf{x}(n)\mathbf{x}(n)^T \\ \mathbf{p}(n) &= b\mathbf{p}(n-1) + d(n)\mathbf{x}(n). \end{aligned} \quad (3.11)$$

A recursive expression of the inverse matrix of $\mathbf{R}(n)$ is given by

$$\mathbf{R}(n)^{-1} = \frac{1}{b} \left[\mathbf{R}(n-1)^{-1} - \frac{\mathbf{R}(n-1)\mathbf{x}(n)\mathbf{x}(n)^T \mathbf{R}(n-1)^{-1}}{b + \mathbf{x}(n)^T \mathbf{R}(n-1)^{-1} \mathbf{x}(n)} \right] \quad (3.12)$$

Now the solution becomes

$$\begin{aligned} \mathbf{h}(n) &= \mathbf{R}(n)^{-1} \mathbf{p}(n) \\ &= \mathbf{h}(n-1) + \left[\frac{\mathbf{R}(n-1)^{-1} \mathbf{x}(n)}{b + \mathbf{x}(n)^T \mathbf{R}(n-1)^{-1} \mathbf{x}(n)} \right] [d(n) - \mathbf{h}(n-1)^T \mathbf{x}(n)] \\ &= \mathbf{h}(n-1) + \mathbf{k}(n)e(n|n-1) \end{aligned} \quad (3.13)$$

where

$$\mathbf{k}(n) = \left[\frac{\mathbf{R}(n-1)^{-1} \mathbf{x}(n)}{b + \mathbf{x}(n)^T \mathbf{R}(n-1)^{-1} \mathbf{x}(n)} \right] : \text{Kalman gain} \quad (3.14)$$

$\mathbf{h}(n-1)$: The best filter coefficients based on all the data upon time $n-1$.

$e(n|n-1) = d(n) - \mathbf{h}(n-1)^T \mathbf{x}(n)$: Tentative estimation error made on the basis of

coefficients $\mathbf{h}(n-1)$. The RLS adaptive algorithm is summarized below.

- i) Initialize $\mathbf{h}(n-1)$ with random numbers.
- ii) Compute $e(n|n-1) = d(n) - \mathbf{h}(n-1)^T \mathbf{x}(n)$
- iii) Update $\mathbf{h}(n-1) + \mu \mathbf{k}(n)e(n|n-1)$ where μ is the adaptation parameter.
- iv) $\mathbf{R}(n)^{-1} = \mathbf{R}(n-1)^{-1} - \mathbf{k}(n)\mathbf{x}(n)^T \mathbf{R}(n-1)^{-1}$
- v) Increase n , go to ii) and continue.

It can be shown that

$$\mathbf{k}(n) = \mathbf{R}(n)^{-1} \mathbf{x}(n) \quad (3.15)$$

Now the two methods can be compared to each other.

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mu \mathbf{R}(n)^{-1} e(n) \mathbf{x}(n) \quad (\text{RLS or Kalman method})$$

$$\mathbf{h}(n) = \mathbf{h}(n-1) + 2\mu e(n) \mathbf{x}(n-1) \quad (\text{LMS or steepest descent method})$$

The RLS method converges much faster than the LMS method at the cost of larger computational complexity. There are $O(M^3)$ multiplications for the RLS method compared to $O(M^2)$ multiplications for the LMS method in each iteration.

3.1.2. Implementation

The performance of Turbo codes, with adaptive equalizer, (LMS and RLS) over transmitting image will be presented in this chapter. The system is made up of turbo codes and the 1-D adaptive noise removal filtering. For experimental results, a satellite image with 100x100 pixel resolution with 256-gray level is taken into consideration. While transmitting, the image is corrupted due to the transmission conditions. So it is needed to minimize the effects and transmit the image with optimum errors (Buyukatak *et al.*, 2003). For simulation, the transmission channel was chosen with the transfer function $G(z) = 1 - 0.7z^{-1} - 0.28z^{-2} + 0.16z^{-3}$ and the noise is the additive white Gaussian noise (AWGN). Every 8-bit pixel of the image is converted to Binary Phase Shift Keying (BPSK) signal. After coding, it is transmitted frame by frame over the communication channel and corrupted by AWGN. The channel, whose the transfer function is as above, is equalized before decoding by LMS, RLS and Kalman algorithms.

After equalization process, the remarkable coding gains are achieved by MAP algorithm. The basic idea is that two or more a posteriori probability decoders exchange soft information. One of the decoders calculates the a posteriori probability distribution of the information sequence and passes that information to the next decoder. The new decoder uses this information and computes its own version of the probability distribution. This exchange of information is called an iteration. After a certain number of iterations, a decision is made at the second decoder. For each iteration, the probability that we decode in favour of the correct decision will improve. At the last iteration, the hard decision is made using the soft decision of the last decoder.

The training data is used to estimate the inverse channel coefficients. In order to equalize the Turbo coded data, these algorithms become satisfactory since only a few hundred data samples (400, in the simulation) are enough to estimate the inverse channel coefficients.

3.1.2.1. System Model

The system consists of turbo encoder (transmitter) and turbo decoder (receiver) sections as shown in Figure 3.3. The decoder employs two identical systematic recursive convolutional encoders (RSC) connected in parallel with an interleaver preceding the second recursive convolutional encoder. The information bits are encoded by both RSC encoders. The first encoder operates on the input bits in their original order, while the second one operates on the input bits as permuted by the interleaver. The decoding algorithm involves the joint Estimation of two Markov processes one for each constituent code. Because the two Markov processes are defined by the same set of data, the estimates can be refined by sharing information between the two decoders in an iterative fashion. The output of one decoder can be used as a priori information by the other decoder. The iteration process is done until the outputs of the individual decoders are in the form of hard bit decisions. In this case, there is not any advantage to share information anymore.

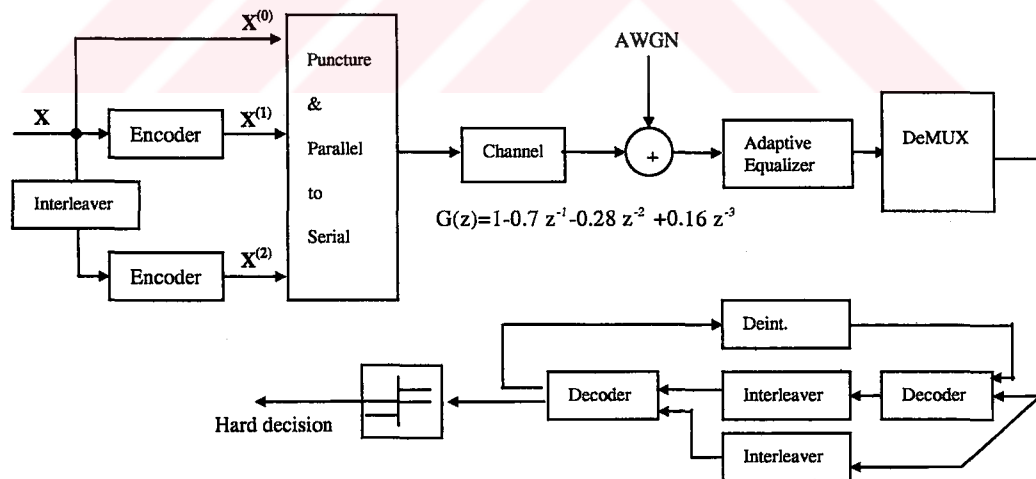


Figure 3.3: Turbo Decoder

3.1.2.2. Simulation Results

In this section, we present simulation results that illustrate the performance of the proposed algorithm over transmitting the satellite (a part of Izmit-Marmara/TURKEY) image. In our example, $\frac{1}{2}$ rate RSC encoder with known

channel model is investigated. Here the generator matrix is $g=[111:101]$, a random interleaver is used and the frame size is chosen as $N=800$. First, the image, pixel by pixel, is converted (256 gray level) binary data. The data are then coded via RSC encoder and a random interleaver. The bit error performance are compared for various iteration numbers for various adaptive inverse filtering processes (i.e. RLS, LMS, Kalman). It is clear that as the iteration number increases, the Bit Error Rates (BER) decrease for any signal to noise ratios (SNR). But for the same SNR values, we obtain different performances depending on which adapting process is used. The best performance is obtained for RLS algorithm. After 4 dB (SNR), it is seen that the BER goes to zero (i.e. all decisions are correct), although the other algorithms gives some Bit Error Rates.

Figure 3.4 (a) shows the original image, (b) and (d) shows the transmitted image that is effected by the channel and AWGN noise with 6 dB and 4 dB respectively. Figure 3.4 (c) and (e) shows the turbo processed images for which 6 dB and 4 dB respectively. It can be seen that the original image can not be identified after transmitting process due to the channel and noise effects. If the image have been coded before transmitting, the great recovery would be achieved after turbo decoding. Figure 3.5, 3.6 and 3.7 shows the Bit Error Rates for 1st, 2nd and 3rd iterations for various signal to noise ratios.

3.1.3. Evaluations

Simulation results are based on steady-state mean-square estimation error (3.3) and in Linear Time Invariant (LTI) channel with $H(z)=1-0.7z^{-1}-0.28z^{-2}+0.16z^{-2}$. Three of the most prominent families of adaptive filtering algorithms are LMS, RLS and Kalman family used for compensating the channel effect. We can see that in terms of Bit Error Rate, the RLS algorithm outperforms the LMS and Kalman algorithm.

Though we have presented results for only one set of channel taps, we found similar results when using different amplitudes, phases, channel lengths, and filter lengths provided that the channel is Minimum Phase (and hence, has an inverse system). In the simulation, we have both AWGN noise and $H(z)$ channel effect, so the adaptive algorithms try to equalize $H(z)$ under the AWGN. For his reason, equalizing the LTI channel fails at lower SNR's.

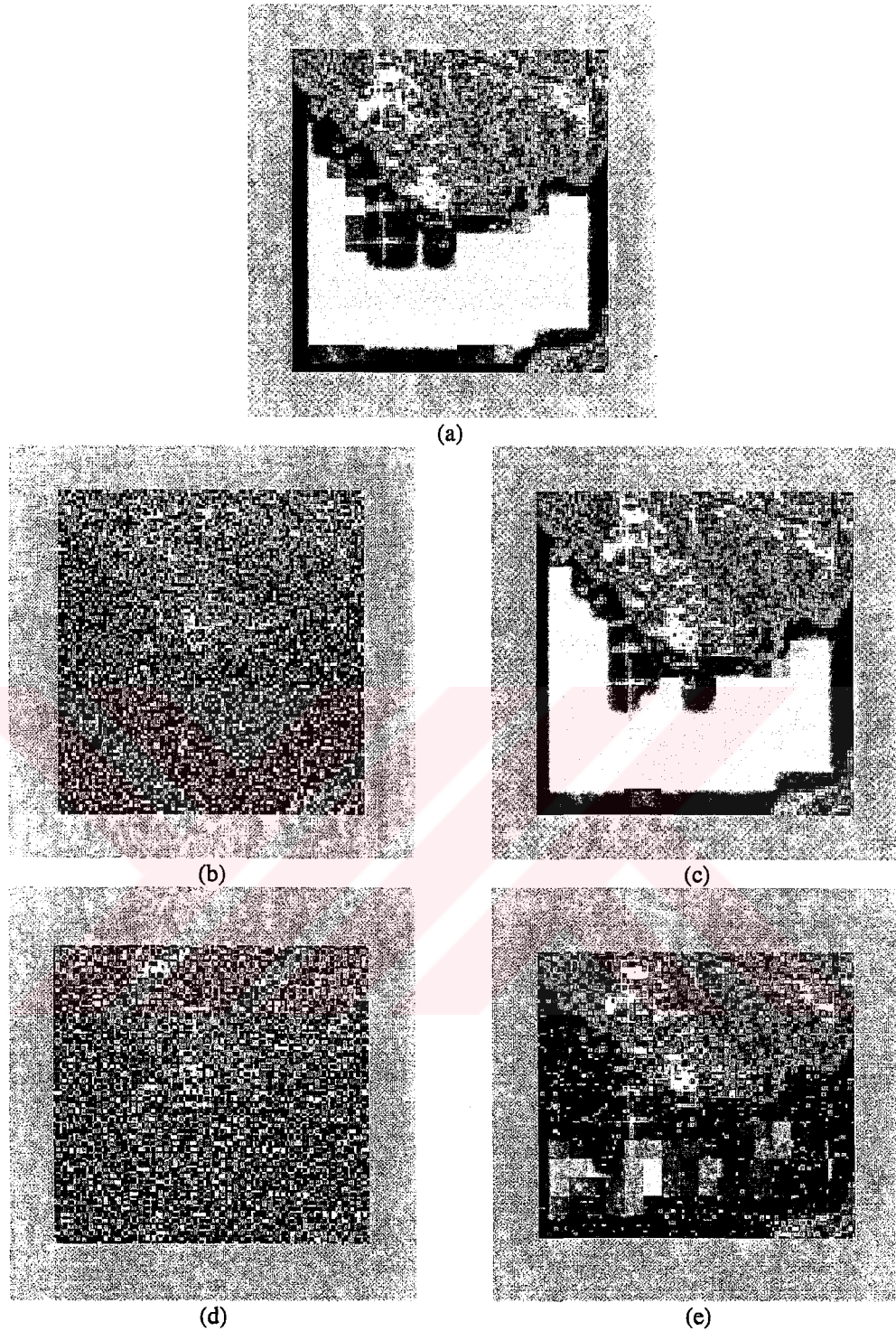


Figure 3.4: Transmission of the Image

(a) Original image

(b) Corrupted image with $H(z)=1-0.7z^{-1}-0.28z^{-2}+0.16z^{-2}$ at 6 dB SNR

(c) Turbo processed image with $H(z)=1-0.7z^{-1}-0.28z^{-2}+0.16z^{-2}$ at 6 dB SNR

(d) Corrupted image with $H(z)=1-0.7z^{-1}-0.28z^{-2}+0.16z^{-2}$ at 4 dB SNR

(e) Turbo processed image with $H(z)=1-0.7z^{-1}-0.28z^{-2}+0.16z^{-2}$ at 4 dB SNR

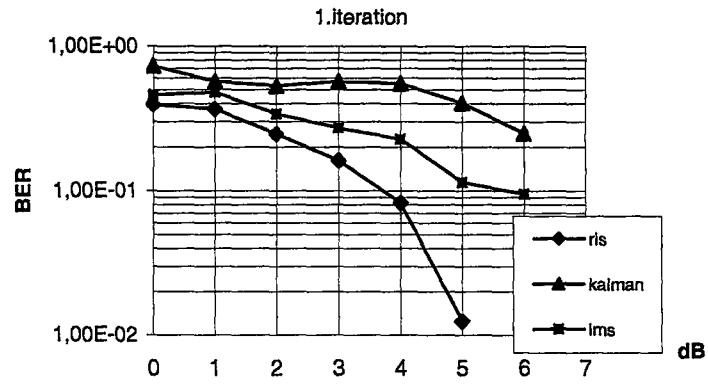


Figure 3.5: Bit Error Rates for 0-7 dB SNR (1. iteration)

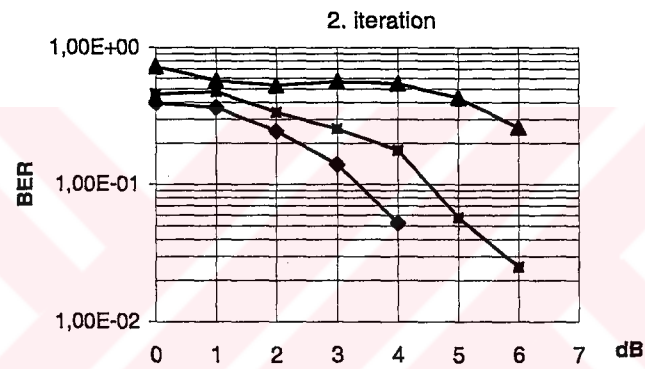


Figure 3.6: Bit Error Rates for 0-7 dB SNR (2. iteration)

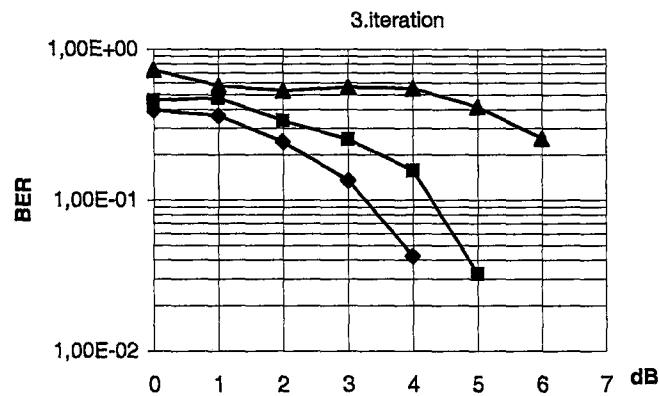


Figure 3.7: Bit Error Rates for 0-7dB SNR (3. iteration)

3.2. ADAPTIVE WIENER-TURBO SYSTEM

3.2.1. Introduction

The performance of Adaptive Wiener-Turbo System (AW-TS) is investigated for AWGN channel model. It is a combination of turbo codes and the 2D adaptive noise removal filtering (Buyukatak *et al.*, 2004). In AW-TS, each binary correspondence of the amplitude values of each pixel is grouped in bit-planes. Then each bit of bit planes are transmitted and at the receiver side, a combined structure, denoted as Wiener-Turbo decoder is employed. Wiener-Turbo decoder is an iterative structure with a feedback link from the second Turbo decoder and Wiener filtering. The decoding process continues iteratively until the desired output is obtained. The advantage of neighborhood relation of pixels is taken into account in AW-TS scheme, resulting improvement of image enhancement. Although each bit slice is transmitted in serial way, the bit string is reassembled at the receiver in such a way that the original neighborhood matrix properties before decoding process. Thus instead of classical binary serial communication and decoding, in our scheme, the coordinates of the pixels are kept as in their original input data matrix. Instead of transmission the binary equivalence with length of N of amplitude value of the pixels, each pixel value is mapped to corresponding binary N -level their binary correspondences are mapped regarding the quantization transmitted in bit slices.

At the receiver, after each bit slice is decoded and output bits are mapped to the previous places by reassembling the first bit from the first bit slice, second bit from the second bit slice, and lastly the last bit from the last bit slice for each pixel. Hence, by collecting all the pixels, the recovery of the image is achieved. In the case of compression, due to the resolution, not all the bits but some of them can be taken into account and corresponding quantized amplitude values of the pixels are found. Thus we can conclude that, in AW-TS both BER and image enhancement can be achieved.

A sample image with 150x150-pixel resolution and 16-gray level is considered as a test image. The problem is to receive the image that is extracted from any image resources, to the processing stations (ground stations, airplanes, ship etc.). While transmitting, it is generally corrupted due to transmission conditions. So it is needed

to minimize the noise effects. Due to the fact that Binary Turbo codes accept only binary inputs, the image must be converted to binary fashion first and then coded before transmission. The solution is found as using “bit planes” (Gonzales and Woods, 1992). While the plane symbols are being transmitted, they are corrupted with distorted channel, which is generally to be AWGN. At the receiver side, before MAP based decoding, the noise reduction is done in order to obtain the highest probability of the symbols. The basic idea in Turbo scheme is that two or more a posteriori probability decoders exchange soft information. One of the decoders calculates the a posteriori probability distribution of the bit planes and passes this information to the next decoder. The new decoder uses this information and computes its own version of the probability distribution. This exchange of information is called iteration. After a certain number of iterations, a decision is made at the second decoder. In each iteration, the probability that is decoded in favor of the correct decision of each bits of the planes will improve. At the last iteration, the hard decision is made using the soft decision of the last decoder.

3.2.2. System Model

The system consists of image slicer, turbo encoder (transmitter), iterative 2D adaptive filter, turbo decoder (receiver), and image combiner sections as shown in Figure 3.8.

The decoder employs two identical systematic recursive convolutional encoders (RSC) connected in parallel with an interleaver preceding the second recursive convolutional encoder. Both RSC encoders encode the information bits of the bit slices. The first encoder operates on the input bits in their original order, while the second one operates on the input bits as permuted by the interleaver. The decoding algorithm involves the joint estimation of two Markov processes one for each constituent code. Because the two Markov processes are defined by the same set of data, the estimated data can be refined by sharing information between the two decoders in an iterative fashion. The output of one decoder can be used as a priori information by the other decoder. The iteration process is done until the outputs of the individual decoders are in the form of hard bit decisions. In this case, there is not any advantage to share information anymore.

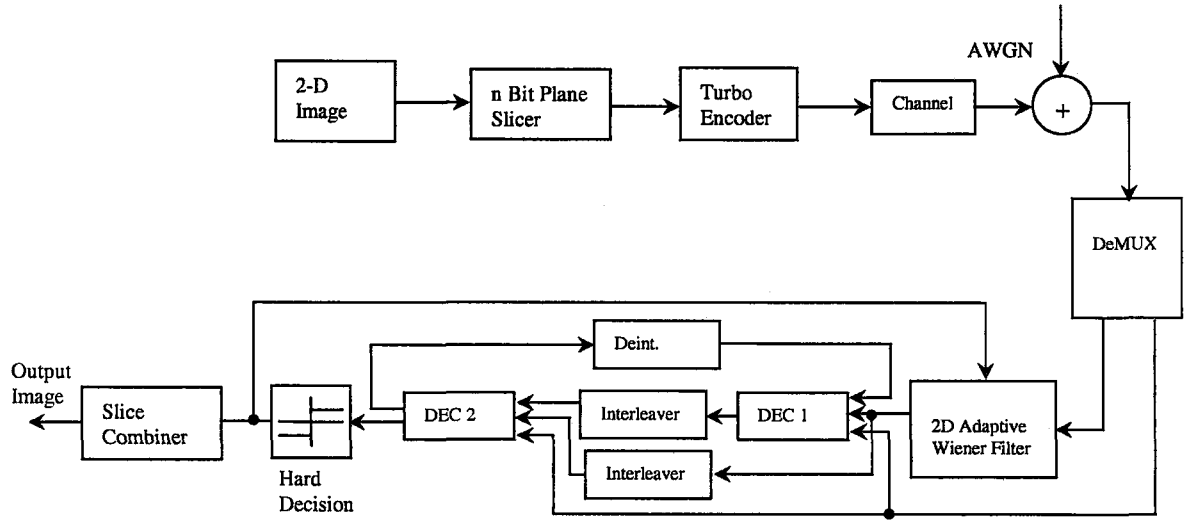


Figure 3.8: AW-TS System Model

3.2.2.1. Bit Plane Decomposition

The gray levels of an n -bit gray-scale image can be represented in their binary representation $a_{n-1} a_{n-2} \dots a_1 a_0$, where each a_j $0 \leq j \leq n-1$ is 0 or 1. Based on this property, a simple method of decomposing the image into a collection of binary images is to separate the a_j 's into N 1-bit bit planes (Gonzales and Woods, 1992). The j -order bit plane is generated by collecting the a_j bits of each pixel. The main drawback of this approach is that small changes in gray level can have significant impact on the complexity of bit planes. If a pixel of intensity 127 (i.e. $(0111111)_2$) is adjacent to a pixel of intensity 128 (i.e. $(1000000)_2$), for instance, every bit plane will contain a corresponding 0 to 1 (or 1 to 0) transition.

An alternative decomposition approach (which reduces the effect of small gray-level variations) is to first represent the image by an n -bit Gray code. The n -bit Gray code

$g_{n-1} g_{n-2} \dots g_1 g_0$ that corresponds to $a_{n-1} a_{n-2} \dots a_1 a_0$ can be computed by the following manner:

$$g_i = a_i \oplus a_{i+1} \quad 0 \leq i \leq n-2$$

$$g_{n-1} = a_{n-1}$$

Here, $\oplus$ denotes the exclusive OR operation. This code has the unique property that successive code words differ in only one bit position. Thus small changes in gray

level are less likely to affect all n bit planes. For example, when gray levels 127 and 128 are adjacent, only the 7th bit plane will contain a 0 to 1 transition, because the Gray codes that correspond to 127 and 128 are 11000000 and 01000000, respectively.

For example, taking only the first digit:

- All values above 128 in an 8 bit image will have the first digit equal to 1 – all values below will contain a zero –therefore the image is being thresholded around 127, a mid-grey. The next digit will then represent a thresholding halfway through the interval that the first one has defined, i.e. if the first two binary digits are 1 then the number is above $128 + (1/2 \times 128) = 192$. If the first two digits are 0, then the number will be below $127 - (1/2 \times 128) = 63$.

What this means is that by separating the image into bit planes, we immediately have a method of identifying more important and less important information. These planes may be easier to encode than the image in its entirety gray coding. To take this one step further we can alter the way in which consecutive pixels are represented by altering the binary code. Gray coding simply changes the order of the code so that a small change in pixel value, as you might see in consecutive pixels in an image, for example, 127 and 128, changes only 1 of the bit planes. This means that more of the bit planed images will contain areas of continuous tone and this in turn means that the image will more easily compress.

Bit Plane Slicing

Instead of highlighting intensity ranges, highlighting the contribution made to the total image appearance by specific bit plays an important role in our study. An application of this technique is also for data compression in the image processing area. Imagine that the image is composed of $N-1$ bit planes, ranging from plane 0 for least significant bit to plane $N-1$ for the most significant bit. We can slice an image into the following bit-planes. Zero is the least significant bit (LSB) and $N-1$ is the most significant bit (MSB). Figure 3.9 illustrates these ideas.

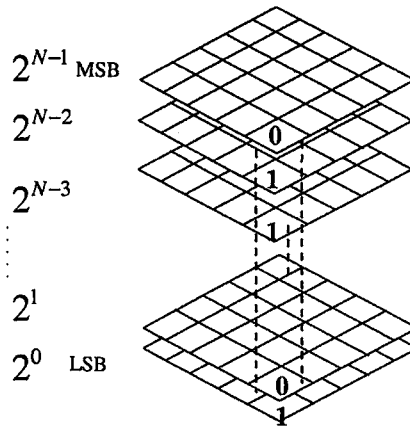
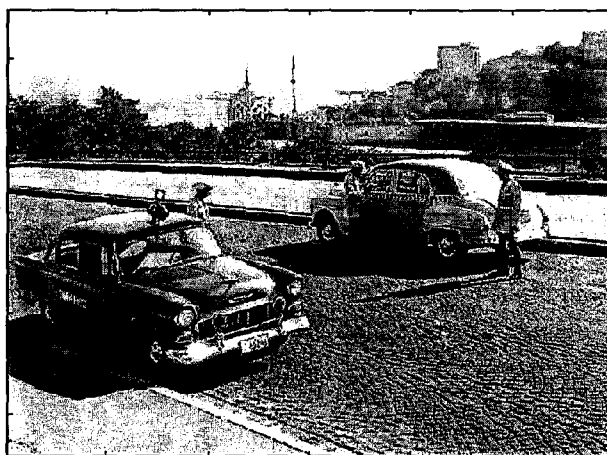


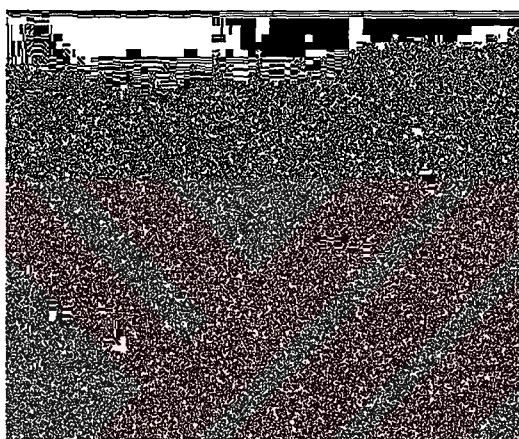
Figure 3.9: The idea of N Plane Slicing

Each bit plane contains a representation of each pixel in the image as a single binary digit either 0 or 1. Because of the way binary code is constructed, it is normally found that the most significant information within the image is contained in the top few bit planes, i.e. those represented by the first few digits of the code. This scheme reveals that only some highest order bits contain visually significant data. Plane $N-1$ corresponds exactly with an image threshold at gray-level 2^{N-1} .

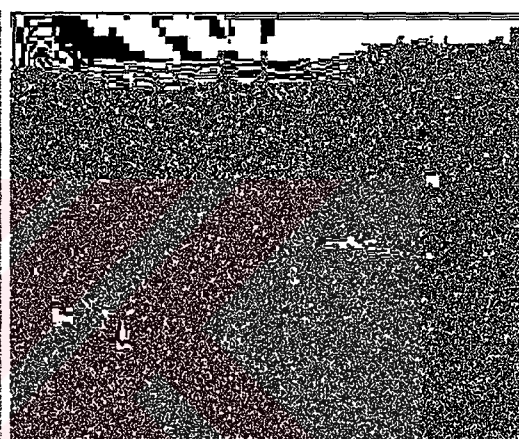
Figure 3.10a, 3.10b and 3.10c illustrates the results from separating an image into its constituent bit planes for which $N=8$ in the binary and gray coding case. Note that, each bit plane carries some part of the information of the original image. The planes are all bi-level, as the pixels are only represented as 1's or 0's. Because of the nature of binary code each individual bit plane represents the image being thresholded at a different 'resolution'. Note that only the three highest order bits contain visually significant data. The other bit planes contribute to more subtle details in the image. Also plane 7 corresponds exactly with an image thresholded at gray level 128.



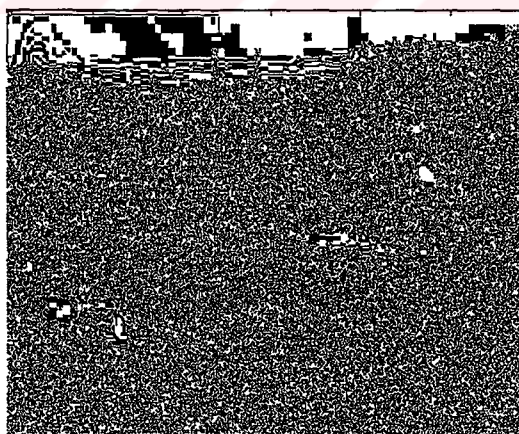
(a)



(b)



(d)



(c)



(e)

Figure 3.10a: Separating an Image into its Constituent Bit Planes for which $N=8$.

(a) Original Image (1950's Istanbul, Dolmabahce)

From (b) to (c): The 0th and 1st planes of eight Binary coded bit planes (left column)

From (d) to (e): The 0th and 1st planes of eight Gray coded bit planes (right column)

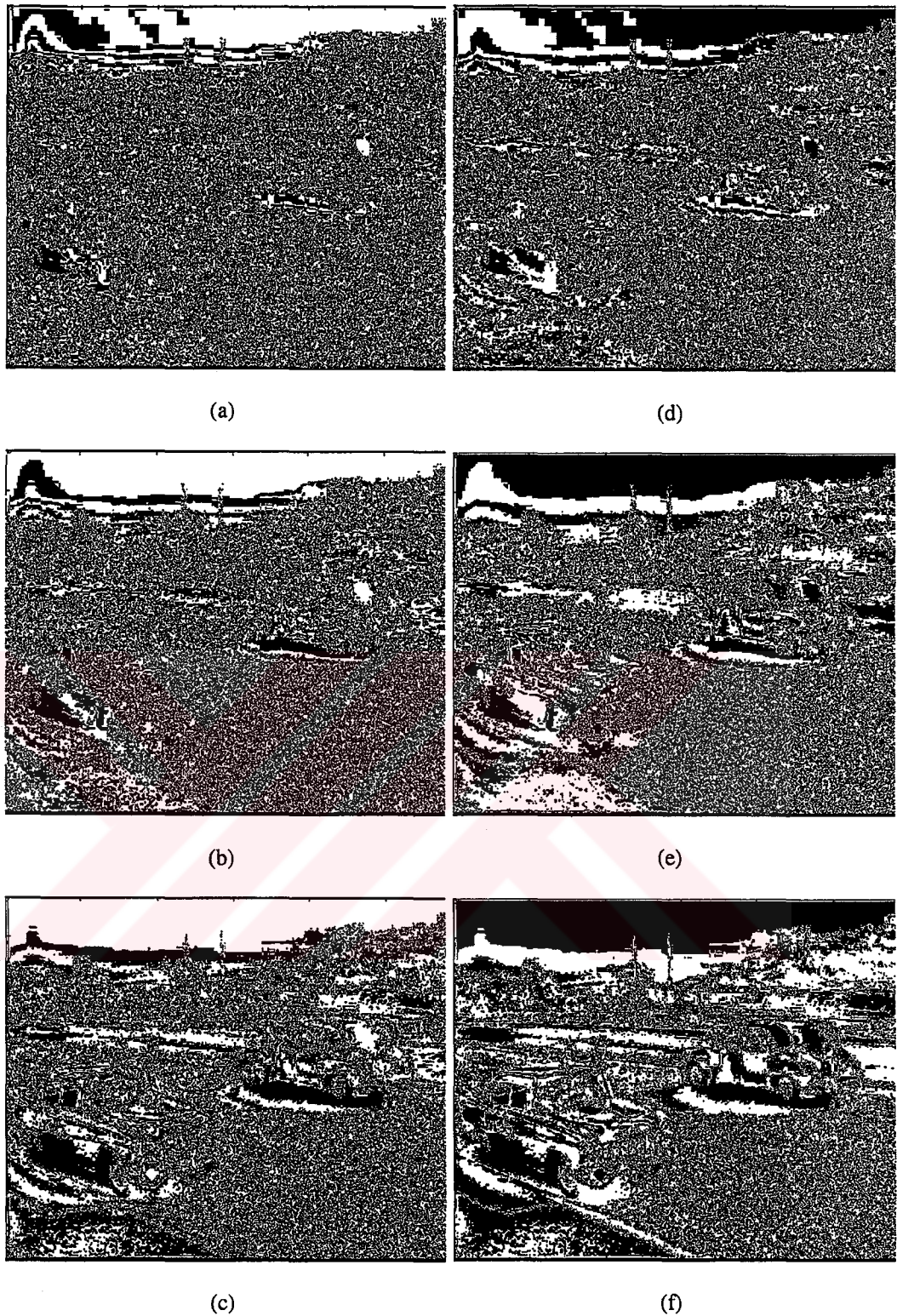
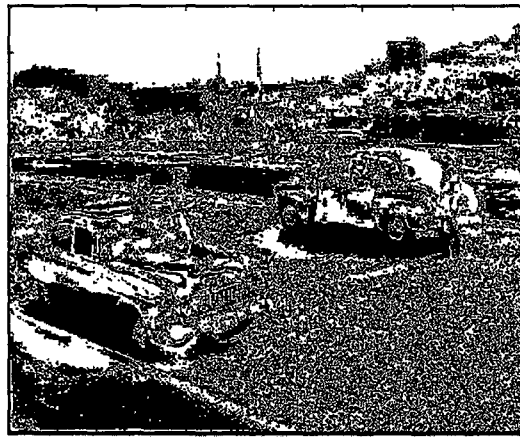


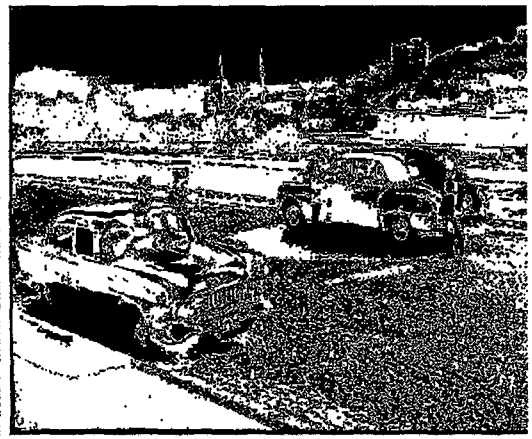
Figure 3.10b: Separating an Image into its Constituent Bit Planes for which $N=8$.

From (a) to (c): The 2nd, 3rd and 4th planes of eight Binary coded bit planes (left column)

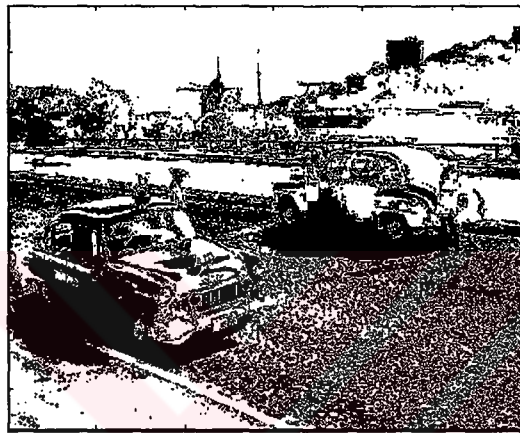
From (d) to (f): The 2nd, 3rd and 4th planes of eight Gray coded bit planes (right column)



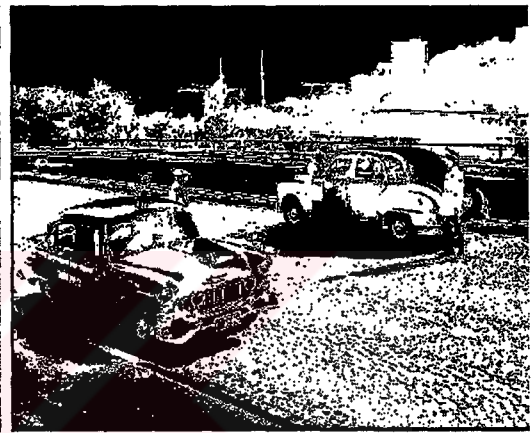
(a)



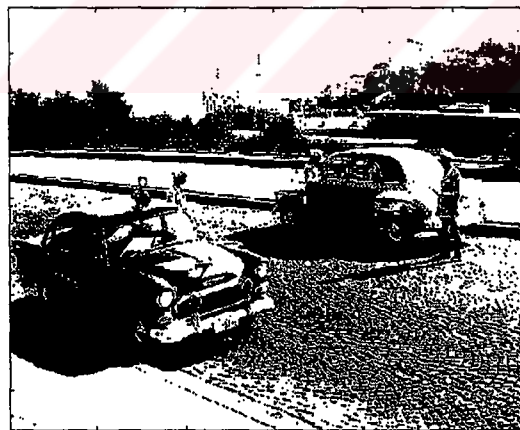
(d)



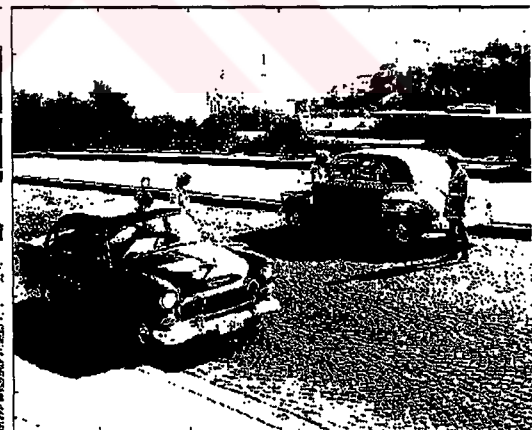
(b)



(e)



(c)



(f)

Figure 3.10c: Separating an Image into its Constituent Bit Planes for which $N=8$.

From (a) to (c): The 5th, 6th and 7th planes of eight Binary coded bit planes (left column)

From (d) to (f): The 5th, 6th and 7th planes of eight Gray coded bit planes (right column)

Bit Plane Combining

Bit plane combining is exactly the inverse process of bit plane slicing. In other words, the planes are recombined at the order from most significant to least significant. Image reconstruction due to bit-plane combining may be lossy or lossless: lossy, by discarding less-significant bit planes (the result is an approximation of the original image); lossless, by taken into consideration all planes. If all planes are taken into account, the image is reconstructed exactly. Figure 3.11 shows the idea of combining the planes for the pixel of $X(m,n)=113 = (01110001)_2$.

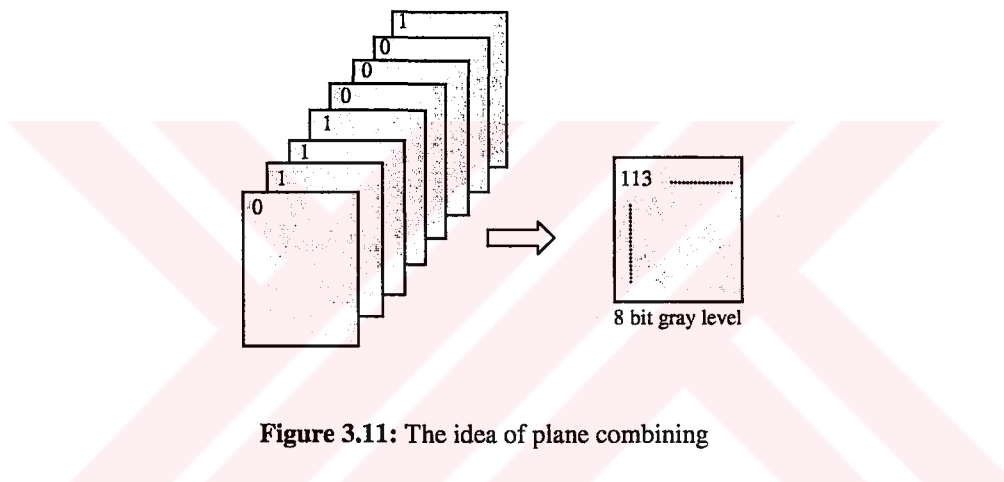


Figure 3.11: The idea of plane combining

Figure 3.12 shows the reconstructed image from the various combinations of bit planes. When the bit planes are all reassembled, the exact (lossless) image (Fig.3.12 (f)) is obtained. Especially, in the case of data rate is important, some planes can be ignored until the changes in gray level have an acceptable impact on the image. This approach will increase the data rate.

Transforming the image in a binary fashion is also very suitable before turbo coding. If the image have been coded and transmitted without being sliced, we would have lost the neighborhood relationship and it would be useless to have 2D denoising filters before decoding. When the image is first sliced and then coded, the neighborhood properties would be saved and evaluated at the receiver. As a result, the noise effect would be reduced before MAP based decoding algorithm, so the performance would highly improve.

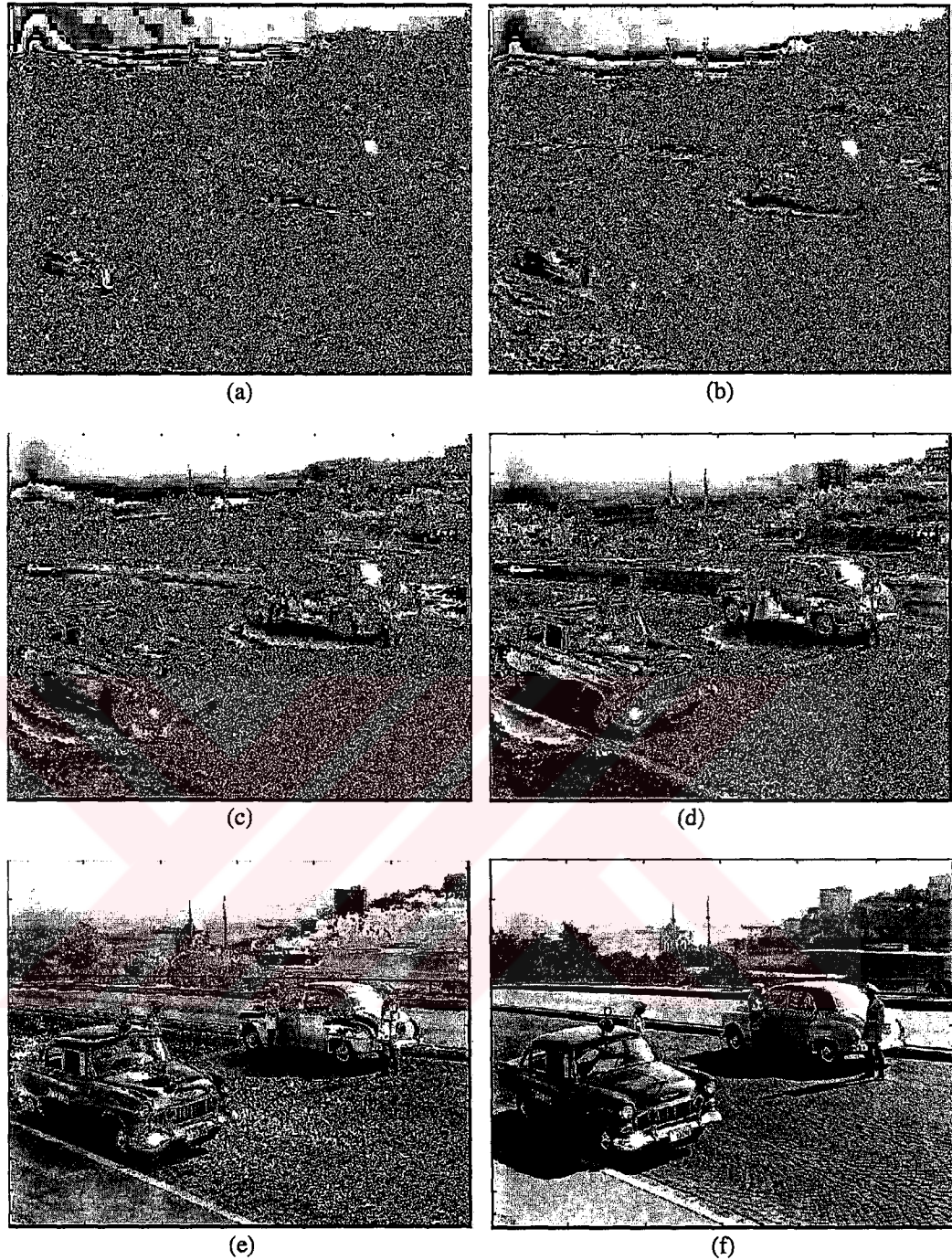


Figure 3.12: Reconstructed Image from the Various Combinations of Bit Planes.
 (a) Bit Plane 0+1+2 (b) Bit Plane 0+1+2+3 (c) Bit Plane 0+1+2+3+4
 (d) Bit Plane 0+1+2+3+4+5 (e) Bit Plane 0+1+2+3+4+5+6 (f) All Bit Planes

3.2.2.2 Encoding the Bit Plane Slices

Consider a half-rate RSC encoder with M memory size. If the d_k is an input at time k the output X_k is equal to

$$X_k = d_k \quad (3.16)$$

Remainder $r(D)$ can be found using feedback polynomial $g^{(0)}(D)$ and feed forward polynomial is $g^{(1)}(D)$. The feedback variable is

$$r_i = d_k + \sum_{j=1}^K r_{k-j} g_j^{(0)} \quad (3.17)$$

and RSC encoder output Y_k which called parity data (Valenti, 1998) is

$$Y_k = \sum_{j=0}^K r_{k-j} g_j^{(1)} \quad (3.18)$$

RSC encoder with memory $M=2$ and rate $R=1/2$ which feedback polynomial $g^{(0)} = 7$ and feed forward polynomial $g^{(1)} = 5$ is illustrated in Figure 3.13 and it has a generator matrix such as;

$$G(D) = \begin{bmatrix} 1 & \frac{1+D+D^2}{1+D^2} \end{bmatrix} \quad (3.19)$$

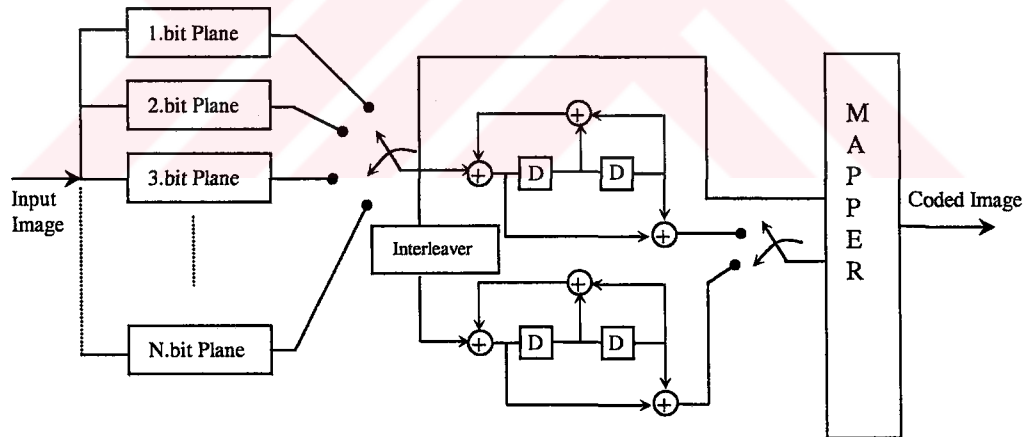


Figure 3.13: Encoder of AW-TS System

3.2.2.3. Wiener Based 2D Denoising Filtering

Introduction

2D Wiener filter is commonly used to restore linearly degraded images. The Wiener filter is the Mean-Square-Error-optimal stationary linear filter and is associated with Norbert Wiener and Andrei N. Kolmogorov back to 1940's. The Wiener filter is based on the assumption that the power spectra of the ideal image and the noise are known. Although the Wiener filter is optimally derived, its true success in restoring real-world images depends on accurate estimation of the image and noise power spectra. In typical problems, estimating the noise power spectrum is relatively easy. In this thesis, we assume the noise power spectrum is known. Mentioned in many textbooks, a set of prototype images of the ideal signal is often suggested to be used for the estimation of power spectrum. However, in practical situations, it is unlikely that such a set of prototypes will be available, and consequently, the single copy of the degraded image is often used for the power spectrum estimation. This will lead to undesirable restored results because of the lack of proper and sufficient images for the spectrum estimation. In this study, we assume we only have access to the degraded image. One intuitively reasonable solution is to use Wiener filter iteratively. In the absence of the ideal image, the filter uses the closest available image to estimate power spectrum, when a better image (the result of the last Wiener filtering) is available; it is used to improve the estimate of power spectrum. This method is the so-called interactive Wiener filter introduced in Hillery Chin, (1991). Both the conventional Wiener filter and the iterative Wiener filter require the assumption that the signal and noise fields are homogeneous. But in a typical image, image characteristics differ considerably from one region to another, so the adaptive Wiener filter is proposed. In this section the theory of Wiener filter, the implementation of Wiener filter based on homogenous image model, the iterative Wiener filter and adaptive Wiener filter will be discussed and achieved.

Brief Discussion of Wiener Filtering Theory

The goal of Wiener filtering is to obtain an estimate of the original signal from a degraded version of the signal. The degraded signal $g(n_1, n_2)$ can be represented by

$$g(n_1, n_2) = f(n_1, n_2) + v(n_1, n_2) \quad (3.20)$$

where $f(n_1, n_2)$ is the uncorrupted, undegraded signal and $v(n_1, n_2)$ is the noise. Given the degraded signal $g(n_1, n_2)$ and some knowledge of the nature of $f(n_1, n_2)$ and $v(n_1, n_2)$ we want to come up with a function $h(n_1, n_2)$ that will output a good estimate of $f(n_1, n_2)$. This estimate is $p(n_1, n_2)$, and is defined by the following:

$$\begin{aligned} p(n_1, n_2) &= g(n_1, n_2) * h(n_1, n_2) \\ P(w_1, w_2) &= G(w_1, w_2) H(w_1, w_2) \end{aligned} \quad (3.21)$$

The Wiener filter generates $h(n_1, n_2)$ that minimizes the mean square error, which is defined by:

$$E\{e^2(n_1, n_2)\} = E\{(g(n_1, n_2) - f(n_1, n_2))^2\} \quad (3.22)$$

This is a linear minimum mean square error (LMMSE) estimation problem, since it is a linear estimator, and the goal is to minimize the mean squared error between $g(n_1, n_2)$ and $f(n_1, n_2)$. Using the orthogonality principle, we can solve this signal estimation problem. The principle states that the error $e(n_1, n_2) = g(n_1, n_2) - f(n_1, n_2)$ is minimized by requiring that $e(n_1, n_2)$ be uncorrelated with any random variable. From the orthogonality principle,

$$E\{e(n_1, n_2)g^*(m_1, m_2)\} = 0 \quad \text{for all } (n_1, n_2) \text{ and } (m_1, m_2)$$

We have:

$$\begin{aligned} E\{f(n_1, n_2)g^*(m_1, m_2)\} &= E\{(e(n_1, n_2) + p(n_1, n_2))g^*(m_1, m_2)\} \\ &= E\{p(n_1, n_2)g^*(m_1, m_2)\} \\ &= E\{(g(n_1, n_2) * h(n_1, n_2))g^*(m_1, m_2)\} \\ &= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h(k_1, k_2) E\{g(n_1 - k_1, n_2 - k_2)g^*(m_1, m_2)\} \end{aligned} \quad (3.23)$$

Rewrite (3.23), we have:

$$\begin{aligned} R_{fg}(n_1 - m_1, n_2 - m_2) \\ = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h(k_1, k_2) R_g(n_1 - k_1 - m_1, n_2 - k_2 - m_2) \end{aligned} \quad (3.24)$$

So:

$$R_{fg}(n_1, n_2) = h(n_1, n_2) * R_g(n_1, n_2) \quad (3.25)$$

$$H(w_1, w_2) = \frac{P_{fg}(w_1, w_2)}{P_g(w_1, w_2)} \quad (3.26)$$

The filter in (3.26) is called the noncausal Wiener filter. Suppose $f(n_1, n_2)$ is uncorrelated with $v(n_1, n_2)$

$$E\{f(n_1, n_2)v^*(m_1, m_2)\} = E\{f(n_1, n_2)\}E\{v^*(m_1, m_2)\} \quad (3.27)$$

Noting that $f(n_1, n_2)$ and $v(n_1, n_2)$ are zero-mean process, we obtain:

$$\begin{aligned} R_{fg}(n_1, n_2) &= R_f(n_1, n_2) \\ R_g(n_1, n_2) &= R_f(n_1, n_2) + R_v(n_1, n_2) \end{aligned} \quad (3.28)$$

and,

$$P_{fg}(w_1, w_2) = P_f(w_1, w_2) \quad (3.29)$$

$$P_g(w_1, w_2) = P_f(w_1, w_2) + P_v(w_1, w_2) \quad (3.30)$$

so,

$$H(w_1, w_2) = \frac{P_f(w_1, w_2)}{P_f(w_1, w_2) + P_v(w_1, w_2)} \quad (3.31)$$

If we further impose the additional constraint that $f(n_1, n_2)$ and $v(n_1, n_2)$ are samples of Gaussian random field, equation (3.22) becomes minimum mean square error (MMSE) estimation problem, then the Wiener filter in equation (3.31) becomes the optimal minimum mean square error estimator. The Wiener filter in (3.31) is a zero-phase filter. Since the power spectra $P(w_1, w_2)$ and $P_v(w_1, w_2)$ are real and nonnegative, $H(w_1, w_2)$ is also real and nonnegative. Therefore, the Wiener filter affects the spectral magnitude but not the phase. By observing (3.31), if we let $P_v(w_1, w_2)$ approach 0, $H(w_1, w_2)$ will approach 1, indicating that the filter tends to preserve the high SNR frequency components. If we let $P_v(w_1, w_2)$ approach infinity, $H(w_1, w_2)$ will approach 0, indicating that the filter tends to attenuate the low SNR frequency components.

Practical Issues of Wiener Filter

Although the Wiener filter is optimally derived, the success of Wiener filter in restoring real-world images depends on accurate estimation of the image power

spectrum. To obtain an accurate estimate of the power spectrum, an ensemble of many samples of the ideal image (as random variables) is required. However, in practical applications it is unlikely that there will be an ensemble of ideal image samples (that is, a set of prototype images) available for the estimation. Actually, in most applications, only the image to be restored is available and all prior knowledge about the ideal image signal has to be estimated from it. Hence, the power spectrum estimated from this single copy of the degraded image is far from the true power spectrum of the ideal image. For these reasons, it is expected that the restoration filter is no longer optimal because of the lack of accurate prior information. In general, an image is an inhomogeneous random field, and ideally, a different model may be required at each pixel. This will make the complexity of the whole image model unmanageable, so simpler image model, which makes the analysis feasible, needs to be assumed. The simplest model is to assume the image is homogenous. We denote the Wiener filters with inhomogeneous image models as adaptive Wiener filters and discuss them in section 4. Here, we point out that, in this project, we assume:

- * Only the image to be restored is available
- * The power spectrum of noise is known, which is a zero-mean white Gaussian.

Implementation

The following are the steps used to implement Wiener filter with homogenous image model assumption. We have to use Discrete Fourier Transform (DFT) to implement Wiener filter in digital computer.

1) Generate the noisy image

SNR and NMSE are defined as:

$$SNR = 10 \log \frac{Var\{f(n_1, n_2)\}}{Var\{v(n_1, n_2)\}} \quad (3.32)$$

$$NMSE = \{f(n_1, n_2), g(n_1, n_2)\} = 100 \frac{Var\{f(n_1, n_2) - g(n_1, n_2)\}}{Var\{f(n_1, n_2)\}} \% \quad (3.33)$$

2) Subtract the mean of noisy image m_g from the degraded image $g(n_1, n_2)$:

$g_0(n_1, n_2) = g(n_1, n_2) - m_g$ Compute $G_0 = (k_1, k_2)$, which is the DFT of $g_0(n_1, n_2)$, with the size $N_1 \times N_2$. N_1, N_2 are the image size.

3) Estimate the power spectrum of the original image and noise

Since we assume we know the noise is zero-mean white Gaussian, so the power spectrum of noise $P_v(k_1, k_2)$ is a matrix with every element is σ_v^2 . One conventional method for power spectrum estimation is to divide the image into smaller regions and use periodogram-averaging method. For a signal $x(n_1, n_2)$ the spectrum estimate $P_x^\xi(w_1, w_2)$ is obtained by

$$P_x^\xi(w_1, w_2) = \frac{1}{N} |X(w_1, w_2)|^2 \quad (3.34)$$

where N is the number of points in a region D . The periodogram $P_x^\xi(w_1, w_2)$ in

(3.34) is the Fourier transform of $P_x^\xi(n_1, n_2)$ given by:

$$P_x^\xi(n_1, n_2) = \frac{1}{N} \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x(k_1, k_2) x^*(k_1 - n_1, k_2 - n_2) \quad (3.35)$$

There is a tradeoff between resolution and variance of $P_x(w_1, w_2)$ estimation. If the image is divided into $K_1 \times K_2$ sections, the periodogram is obtained for each of the smaller segments, and the spectrum estimate $P_x^\xi(w_1, w_2)$ is taken as the average of the periodograms, so the variance is reduced by a factor of $K_1 K_2$ and the resolution is reduced by a factor of K_1 in w_1 and K_2 in w_2 . Since we have the degraded image available only, we have to estimate the power spectrum of ideal image from the degraded image. The power spectrum of degraded image is:

$$P_g(k_1, k_2) = \frac{1}{N_1 \times N_2} |G_0(k_1, k_2)|^2 \quad (3.36)$$

where, $G_0(k_1, k_2)$ is the DFT of $g_0(n_1, n_2) = g(n_1, n_2) - m_g$

From equation (3.30), $P_f(w_1, w_2) = P_g(w_1, w_2) + P_v(w_1, w_2)$

4) Find the Wiener filter

From equation (3.31)

$$H(k_1, k_2) = \frac{P_f(k_1, k_2)}{P_f(k_1, k_2) + P_v(k_1, k_2)}, \quad \text{and} \quad (3.37)$$

$$h(n_1, n_2) = \text{IDFT} \{ H(k_1, k_2) \}$$

5) Apply Wiener filter

In theory, if we apply $h(n_1, n_2)$ or $H(k_1, k_2)$ directly, there is circular convolution effect. We should use FIR filter design method (like window method) to find a $h'(n_1, n_2)$, which has smaller size, in order to alleviate the effect. By observing $h(n_1, n_2)$, we find that most part of $h(n_1, n_2)$, except for the four corners (remember $h(n_1, n_2)$ is the IDFT of $H(k_1, k_2)$), are nearly 0, the results of applying $h(n_1, n_2)$ and $h'(n_1, n_2)$ should be almost the same, and we show the result with applying $h(n_1, n_2)$ directly.

$$P_0(k_1, k_2) = G_0(k_1, k_2)H(k_1, k_2) \quad (3.38)$$

$$p_0(n_1, n_2) = IDFT(P_0(k_1, k_2)) \quad (3.39)$$

6) Add back mean to get the restored image. $p(n_1, n_2) = p_0(n_1, n_2) + m_f$. Since we assume the zero-mean Gaussian noise, $m_f = m_g$ and $p(n_1, n_2) = p_0(n_1, n_2) + m_g$

Iterative Wiener Filter

By using the degraded image as a prototype, an (initial) estimate of power spectrum is determined, and subsequently, a restored image can be obtained with Wiener filtering. It is quite plausible that the restored image represents a better prototype than the given degraded image, and can be used to obtain a better estimate of the power spectrum. This suggests an iterative procedure whereby successively restored images are used to update the power spectrum estimates and subsequently to improve the restoration. This is called iterative Wiener filtering.

Adaptive Wiener Filter

The Wiener filters discussed in the previous two sections are developed under the assumption that the image is homogeneous. This results a space-invariant filter. In general, an image is an inhomogeneous random field, so a homogeneous model cannot fit it very well. From the discussion in the previous sections, we know the Wiener filter requires estimating the signal mean m_f , noise mean m_v , signal power spectrum $P_f(w_1, w_2)$, and noise power spectrum $P_v(w_1, w_2)$. Instead of assuming a fixed $m_f, m_v, P_f(w_1, w_2)$ and $P_v(w_1, w_2)$ for the entire image, they can be estimated locally. This type of Wiener filters is called adaptive Wiener filter. A prior probability model of the image is of central importance for adaptive Wiener filter design. Modeling the statistics of natural images is a challenging task. Ideally, to represent an inhomogeneous image, a different model may be required at each pixel.

This will cause computational overload in processing images. In order to avoid such unmanageable complexities, simpler image models that make the analysis feasible but still take into account the inhomogeneity of the image are needed. In this section, we will discuss Wiener filtering with inhomogeneous image model developed by (A. D. Hillery). The following are the rules used to implement Wiener filter with bit-sliced plane image model. In our problem, Wiener filter is applied to all planes of the image and hence, we still assume that the additive noise on the bit-plane $v(n_1, n_2)$ is zero mean and white with variance of σ_v^2 . Let $f^j(n_1, n_2)$ shows each planes from 1st to 4th (i.e. $j=1, 2, 3, 4$). Their power spectrums $P_v^j(w_1, w_2)$ are then given by $P_v^j(w_1, w_2) = (\sigma_v^j)^2$. Consider a small local region in which the plane $f^j(n_1, n_2)$ is assumed homogeneous. Within the local region, the plane $f^j(n_1, n_2)$ is modeled by,

$$f^j(n_1, n_2) = m_f^j + \sigma_f^j w^j(n_1, n_2) \quad (3.40)$$

where m_f^j and σ_f^j are the local mean and standard deviation of $f^j(n_1, n_2)$ and $w^j(n_1, n_2)$ is zero-mean white noise with unit variance effecting each plane. Within the local region, then, the Wiener filter $H^j(w_1, w_2)$, $h^j(n_1, n_2)$ is given by:

$$H^j(w_1, w_2) = \frac{P_f^j(w_1, w_2)}{P_f^j(w_1, w_2) + P_v^j(w_1, w_2)} = \frac{(\sigma_f^j)^2}{(\sigma_f^j)^2 + (\sigma_v^j)^2}; \quad j=1, 2, 3, 4 \quad (3.41)$$

$$h^j(n_1, n_2) = \frac{(\sigma_f^j)^2}{(\sigma_f^j)^2 + (\sigma_v^j)^2} \delta(n_1, n_2); \quad j=1, 2, 3, 4 \quad (3.42)$$

Then, the restored planes $p^j(n_1, n_2)$ within the local region can be expressed as

$$\begin{aligned} p^j(n_1, n_2) &= m_f^j + (g^j(n_1, n_2) - m_f^j) * \frac{(\sigma_f^j)^2}{(\sigma_f^j)^2 + (\sigma_v^j)^2} \delta(n_1, n_2) \\ &= m_f^j + \frac{(\sigma_f^j)^2}{(\sigma_f^j)^2 + (\sigma_v^j)^2} (g^j(n_1, n_2) - m_f^j); \quad j=1, 2, 3, 4 \end{aligned} \quad (3.43)$$

If we assume that m_f^j and σ_f^j are updated at each symbol,

$$p^j(n_1, n_2) = m_f^j(n_1, n_2) + \frac{(\sigma_f^j)^2(n_1, n_2)}{(\sigma_f^j)^2(n_1, n_2) + (\sigma_v^j)^2(n_1, n_2)} (g^j(n_1, n_2) - m_f^j(n_1, n_2)) \quad (3.44)$$

From degraded planes, we can estimate $(\sigma_g^j)^2(n_1, n_2)$ only, and since

$$(\sigma_g^j)^2(n_1, n_2) = (\sigma_f^j)^2(n_1, n_2) + (\sigma_v^j)^2(n_1, n_2), \quad \text{Equation changes to (3.44)}$$

$$p^j(n_1, n_2) = m_g^j(n_1, n_2) + \frac{(\sigma_g^j)^2(n_1, n_2) - (\sigma_v^j)^2(n_1, n_2)}{(\sigma_g^j)^2(n_1, n_2)} (g^j(n_1, n_2) - m_g^j(n_1, n_2)). \quad (3.45)$$

$$j = 1, 2, 3, 4$$

It can be observed that the performance of adaptive Wiener filter applied to the planes (without using Turbo scheme) is satisfying only above 8 dB or higher SNR's. The objective in our study is to obtain better results at much lower SNR's. So after the bit planes are (Wiener based) filtered, the MAP based turbo-decoding algorithm is considered. So, in equation (3.44), $p^j(n_1, n_2)$ is taken the degraded planes as the new inputs for the turbo decoders. The resulting output planes can be expressed as:

$$\begin{aligned} p_{i+1}^{(1)}(n_1, n_2) &= m_{g_i}^{(1)}(n_1, n_2) + \frac{(\sigma_{g_i}^{(1)})^2(n_1, n_2) - (\sigma_v^{(1)})^2(n_1, n_2)}{(\sigma_{g_i}^{(1)})^2(n_1, n_2)} (p_i^{(1)}(n_1, n_2) - m_{g_i}^{(1)}(n_1, n_2)) \\ p_{i+1}^{(2)}(n_1, n_2) &= m_{g_i}^{(2)}(n_1, n_2) + \frac{(\sigma_{g_i}^{(2)})^2(n_1, n_2) - (\sigma_v^{(2)})^2(n_1, n_2)}{(\sigma_{g_i}^{(2)})^2(n_1, n_2)} (p_i^{(2)}(n_1, n_2) - m_{g_i}^{(2)}(n_1, n_2)) \\ p_{i+1}^{(3)}(n_1, n_2) &= m_{g_i}^{(3)}(n_1, n_2) + \frac{(\sigma_{g_i}^{(3)})^2(n_1, n_2) - (\sigma_v^{(3)})^2(n_1, n_2)}{(\sigma_{g_i}^{(3)})^2(n_1, n_2)} (p_i^{(3)}(n_1, n_2) - m_{g_i}^{(3)}(n_1, n_2)) \\ p_{i+1}^{(4)}(n_1, n_2) &= m_{g_i}^{(4)}(n_1, n_2) + \frac{(\sigma_{g_i}^{(4)})^2(n_1, n_2) - (\sigma_v^{(4)})^2(n_1, n_2)}{(\sigma_{g_i}^{(4)})^2(n_1, n_2)} (p_i^{(4)}(n_1, n_2) - m_{g_i}^{(4)}(n_1, n_2)) \end{aligned} \quad (3.46)$$

where i is the iteration index for each plane, between the Wiener process and turbo decoder (called WT iteration index). i can be taken from one to the desired number. If it is taken one, the MAP processed plane (at the output of the decoder) enters to the Wiener filter only one times. In this case, we have $P_1^j(n_1, n_2)$ and $P_2^j(n_1, n_2)$ for each plane and for reconstructing the image, only $P_2^j(n_1, n_2)$ is taken into consideration. If WT index is taken zero, the output of the decoders never refiltered by Wiener. WT index is taken as 1 in our study. The simulation results have shown that it is not necessary to take WT index too high.

3.2.2.4. Decoding the Bit Plane Slices

We consider the MAP approach for decoding. The goal of the MAP algorithm is to find the a posteriori probability of each bit of The BPSs produced by the underlying Markov process, given the noisy observation $\mathbf{y}$. Once the a posteriori probabilities are

calculated for all possible values of the desired quantity, a hard decision is made by taking the quantity with the highest probability. When used for turbo decoding, The MAP algorithm calculates the a posteriori probabilities of the bits for each planes, $P[m_i = 1 | y]$ and $P[m_i = 0 | y]$, which are then put into LLR form according to

$$\Lambda_k = \ln \frac{P[m_k = 1 | y]}{P[m_k = 0 | y]} \quad (3.47)$$

Before finding the a posteriori probabilities for the message bits, the MAP algorithm first finds the probability $P[s_i \rightarrow s_{i+1} | y]$ of each valid state transition given the noisy channel observation y . From the definition of conditional probability

$$P[s_i \rightarrow s_{i+1} | y] = \frac{P[s_i \rightarrow s_{i+1}, y]}{P[y]} \quad (3.48)$$

The properties of the Markov process can be used to partition the numerator as

$$P[s_i \rightarrow s_{i+1}, y] = \alpha(s_i) \gamma(s_i \rightarrow s_{i+1}) \beta(s_{i+1}),$$

Where

$$\begin{aligned} \alpha(s_i) &= P[s_i, (y_0, \dots, y_{i-1})] \\ \gamma(s_i \rightarrow s_{i+1}) &= P[s_{i+1}, y_i | s_i] \\ \beta(s_{i+1}) &= P[(y_{i+1}, \dots, y_{L-1}) | s_{i+1}]. \end{aligned} \quad (3.49)$$

The probability $\alpha(s_i)$ can be found according to the forward recursion.

$$\alpha(s_i) = \sum_{s_{i-1}} \alpha(s_{i-1}) \gamma(s_{i-1} \rightarrow s_i) \quad (3.50)$$

Likewise, $\beta(s_i)$ can be found according to the backward recursion.

$$\beta(s_i) = \sum_{s_{i+1}} \beta(s_{i+1}) \gamma(s_i \rightarrow s_{i+1}) \quad (3.51)$$

Once the a posteriori probability of each state transition $P[s_i \rightarrow s_{i+1} | y]$ is found, the message bit probabilities can be found according to

$$P[m_i = 1 | y] = \sum_{s_i} P[s_i \rightarrow s_{i+1} | y] \quad (3.52)$$

and
$$P[m_i = 0 | y] = \sum_{S_0} P[s_i \rightarrow s_{i+1} | y]$$

Where $S_1 = \{s_i \rightarrow s_{i+1} : m_i = 1\}$ is the set of all state transitions associated with a message bit of 1, and $S_0 = \{s_i \rightarrow s_{i+1} : m_i = 0\}$ is the set of all state transitions associated with a message bit of 0. The log-likelihood ratio then becomes

$$\Lambda_i = \ln \frac{\sum_{S_0} \alpha(s_i) \gamma(s_i \rightarrow s_{i+1}) \beta(s_{i+1})}{\sum_{S_1} \alpha(s_i) \gamma(s_i \rightarrow s_{i+1}) \beta(s_{i+1})} \quad (3.53)$$

Now let

$$\hat{\beta}(s_i) = \ln \beta(s_i) = \ln \left\{ \sum_{s_{i+1}} \exp \left[\hat{\beta}(s_{i+1}) + \hat{\gamma}(s_i \rightarrow s_{i+1}) \right] \right\} \quad (3.54)$$

Likewise, let

$$\hat{\alpha}(s_i) = \ln \alpha(s_i) = \ln \left\{ \sum_{s_{i-1}} \exp \left[\hat{\alpha}(s_{i-1}) + \hat{\gamma}(s_{i-1} \rightarrow s_i) \right] \right\} \quad (3.55)$$

So, LLR can be found as (Valenti, 1998; Gross and Gulak, 1998a; Hageneauer, 1996)

$$\Lambda_i = \ln \left\{ \sum_{s_i} \exp \left[\hat{\alpha}(s_i) + \hat{\gamma}(s_i \rightarrow s_{i+1}) + \hat{\beta}(s_{i+1}) \right] \right\} - \ln \left\{ \sum_{s_0} \exp \left[\hat{\alpha}(s_i) + \hat{\gamma}(s_i \rightarrow s_{i+1}) + \hat{\beta}(s_{i+1}) \right] \right\} \quad (3.56)$$

This soft bit information is de-interleaved and evaluated at the hard decision section. Figure 3.14 achieves equation (3.56) and combines the BPSs.

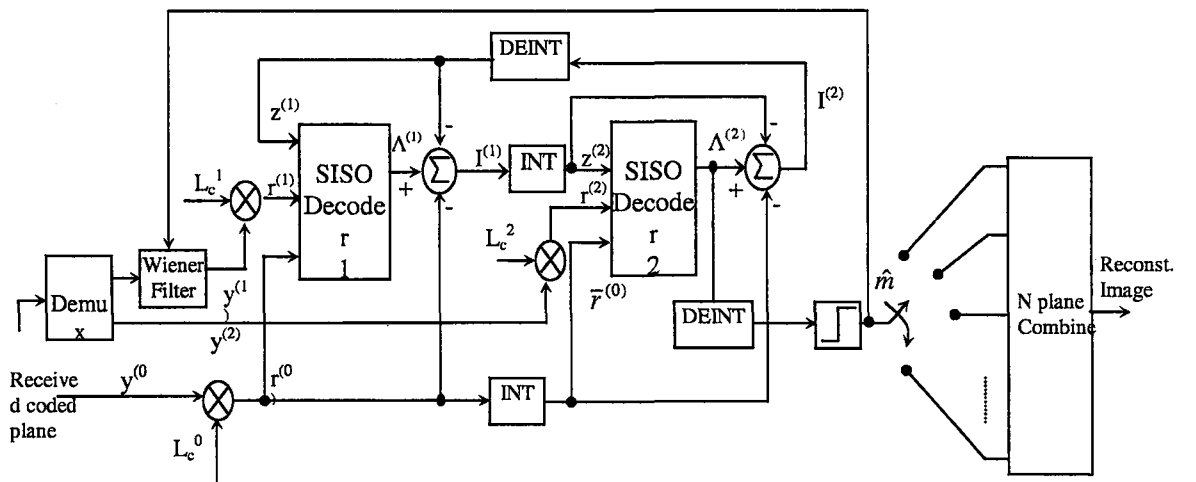


Figure 3.14: Decoder of AW-TS

3.2.3. Simulation Results

In this section, we present simulation results that illustrate the performance of the proposed (AW-TS) algorithm over transmitting the image. In our example, $\frac{1}{2}$ rate RSC encoder with AWGN channel model is investigated. Here the generator matrix is $g = [111:101]$, a random interleaver is used and the frame size is chosen as $N=150$, iteration (between wiener filter and decoder) is taken 1. At first, the pixels of the image is converted to 16 gray levels and then sliced to four bit-planes. All planes are then coded via RSC encoder and a random interleaver. The coded planes are corrupted with $SNR = 2\text{dB}, 1\text{dB}, 0\text{dB}, -1\text{dB}, -2\text{dB}$ and -3dB and transmitted. Figure A.1, A.2, A.3, A.4, A.5 and A.6 show the corrupted, traditional turbo processed, the well known Wiener image processed and the proposed AW-TS processed images.

The results have shown that, to recover the image corrupted at 2 dB SNR or below, the well known image-processing and conventional Turbo algorithms are not satisfying as seen from figures A.1 to A.6. The proposed AW-TS algorithm gives good results from 0 dB SNR to -3 dB. This is a challenging result and we have at least 4dB and 2.5dB additional SNR improvement compared to that of classical image processing and conventional turbo coding systems, respectively. Also from Figure A.7, we see that the BER of 3rd bit plane is better than the others. The BER of 2nd bit is better than 0th and 1st bit planes. The BER of 1st plane is better than that of the 0th bit plane. This is an expected result, since the neighborhood relations of 3rd bit plane is more than the others. The neighborhood relationships decrease as 2nd bit, 1st bit, 0th bit planes, respectively. At 2dB SNR, the BER of 0th bit plane of the proposed AW-TS system is same as that of conventional turbo system and above 2 dB SNR level, it becomes worse. But due to the fact that the 0th bit plane is the least significant bit plane and contains visually less significant data in general, the resulting recovered image is still satisfying. If the BER of 3rd bit plane would be high instead of the 0th bit plane, the reconstructed image would be degraded much more compared to the previous case. So, obtaining good results depend on the local neighborhood relations of the planes. Increasing the number of the bit planes enables us to study with more detailed images. But if the image is not very detailed, it is no need to increase the number of the bit planes in order to avoid complexity.

3.2.4. Evaluations

The images being transmitted over noisy channels are extremely sensitive to the bit errors, which can severely degrade the quality of the image at the receiver. This

necessitates the application of error control codes in the image transmission. This study presents an efficient image transmission by means of a new proposed AW-TS (Adaptive Wiener-Turbo System) method, which takes the advantage of the superior performance of error control codes, Turbo codes. In AW-TS, the original image is partitioned into 2^N quantization levels, where N is denoted as bit planes. Then each of the N -bit-plane is coded by Turbo encoder and transmitted over Additive White Gaussian Noise (AWGN) channel. At the receiver side, bit-planes are re-assembled taking into consideration of neighborhood relationship of pixels in 2-D images. Each of the noisy bit-plane values of the image is evaluated iteratively by a combined equalization block, which is composed of 2-D adaptive noise-removal filtering and Turbo decoder. In AW-TS, there is a feedback link between Wiener filtering and Turbo decoder and process iteratively. Our scheme employs a pixel-wise adaptive Wiener-based Turbo decoder and uses statistics (mean and standard deviation of local image) of estimated values of local neighborhood of each pixel. It has extraordinary satisfactory results of both bit error rate (BER) and image enhancement performance for less than 2 dB Signal-to-Noise Ratio (SNR) values, compared to separately application of traditional turbo coding scheme and 2-D filtering. We can also achieve compression by using AW-TS systems. In compression we use less memory storage and increase data rate up to $N-1$ times by simply choosing any number of bit slices, sacrificing resolution.

For comparison, the classical image processing (based Wiener filtering) algorithm, conventional Turbo coding, Adaptive Wiener-Turbo algorithm (AW-TS) methods are evaluated and the results are compared. We have seen that more than 4 dB additional SNR improvement is achieved in the proposed algorithm, AW-TS compared to that of Turbo and 2.5 dB additional gain compared to Wiener methods applied separately. Hence, we conclude that AW-TS system will be a compromising approach in 2-D image transmission, recovery of noisy signals.

3.3. Iterative Cellular Image Processing -Turbo System

3.3.1. Introduction

In this chapter, to compress, to improve bit error performance and also to enhance 2D images, a new scheme, called Iterative Cellular-Turbo System (IC-TS) is introduced, (Osman *et al.*, 2003). In IC-TS, the original image is partitioned into 2^N quantization levels, where N is denoted as bit planes. Then each of the N -bit-plane is coded by Turbo encoder and transmitted over Additive White Gaussian Noise (AWGN) channel. At the receiver side, bit-planes are re-assembled taking into consideration of neighborhood relationship of pixels in 2-D images. Each of the noisy bit-plane values of the image is evaluated iteratively using IC-TS structure, which is composed of equalization block; Iterative Cellular Image Processing Algorithm (ICIPA) and Turbo decoder. In IC-TS, there is an iterative feedback link between ICIPA and Turbo decoder. ICIPA uses mean and standard deviation of estimated values of each pixel neighborhood (Osman *et al.*, 2003). It has extra-ordinary satisfactory results of both Bit Error Rate (BER) and image enhancement performance for less than -1 dB Signal-to-Noise Ratio (SNR) values, compared to traditional turbo coding scheme and 2-D filtering, applied separately. We can also achieve compression by using IC-TS systems. In compression we use less memory storage and increase data rate up to $N-1$ times by simply choosing any number of bit slices, sacrificing resolution. Hence, we conclude that IC-TS system will be a compromising approach in 2-D image transmission, recovery of noisy signals and image compression.

The new unsupervised iterative algorithm uses the advantage of stochastic properties and neighborhood relations between the cells of the input image. In ICIPA scheme; first regarding to the stochastic properties of the data, all possible quantization levels are determined and then 2D input image is processed using a function, based on averaging and neighborhood relationship, and after that a parameter is assigned to each cell. Then Gaussian probability values are mapped to each cell regarding to all possible quantization levels and the attended value of the parameter. A maximum selector defines the highest probability value for each cell. In the case of complex data, first iteration output is fed into input till a sufficient output is found.

3.3.2. System Model

The system consists of image slicer, turbo encoder (transmitter), ICIPA, turbo decoder (receiver), and image combiner sections as shown in Figure 3.15. In IC-TS,

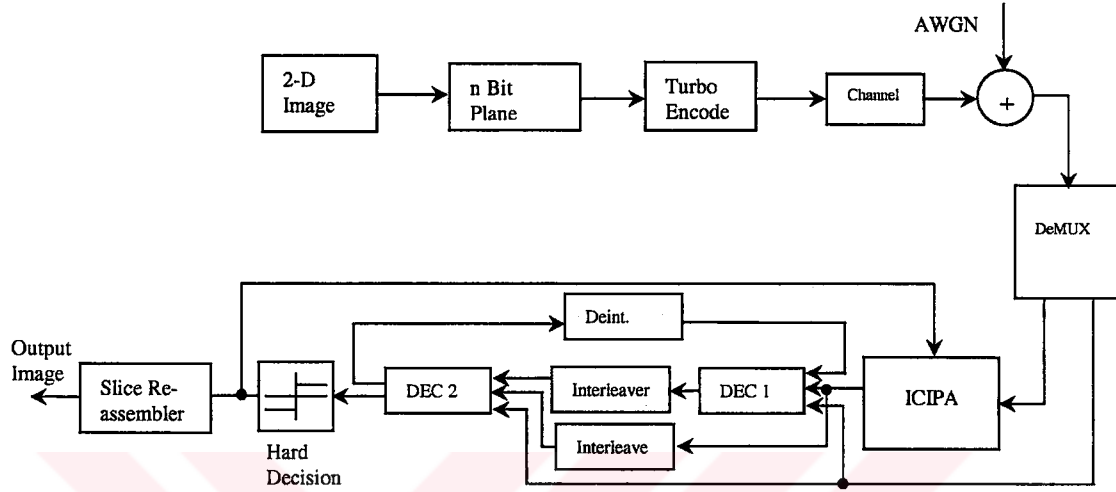


Figure 3.15: IC-TS System Model.

binary correspondence of each pixel amplitude value is grouped in bit-planes. Then these bit planes are transmitted and at the receiver side, a combined structure, denoted as ICIPA-Turbo decoder is employed. ICIPA-Turbo decoder is an iterative structure with a feedback link from the second Turbo decoder and ICIPA filtering. The decoding process continues iteratively till the desired output is obtained. The advantage of neighborhood relation of pixels is taken into account in IC-TS scheme, resulting improvement of image enhancement. Although each bit slice is transmitted in serial way, the bit string is reassembled at the receiver to preserve the original neighborhood matrix properties before decoding process. Thus instead of classical serial bit stream communication and decoding, in our scheme, the coordinates of the pixels are kept as in their original input data matrix. Each pixel value is mapped to corresponding binary N-level and their binary correspondences are mapped regarding to the quantization of the transmitted bit slices.

In IC-TS, data rate can be increased up to $(N-1)$ times, by only transmitting most important significant bit. Thus our bit slicing can be also an efficient way of compression technique. To obtain more accurate 2D images, other bits can also be transmitted.

3.3.3. Iterative Cellular Image Processing Algorithm

3.3.3.1. Introduction

In this section, a new iterative image processing algorithm is introduced and will be denote as “iterative cellular image processing algorithm” (ICIPA). The new unsupervised iterative algorithm uses the advantage of stochastic properties and neighborhood relations between the cells of the input image. In ICIPA scheme; first regarding to the stochastic properties of the data, all possible quantization levels are determined and then 2D input image is processed using a function, based on averaging and neighborhood relationship, and after that a parameter C is assigned to each cell. Then Gaussian probability values are mapped to each cell regarding to all possible quantization levels and the attended value C. A maximum selector defines the highest probability value for each cell. In the case of complex data, first iteration output is fed into input till a sufficient output is found. We have applied ICIPA algorithm to various synthetic examples and then a real data, the ruins of Hittite Empire. Satisfactory results are obtained. We have observed that de-noising property of our scheme is the best in the literature. It is interesting that the corrupted data with Additive White Gaussian Noise (AWGN) up to 97% ratio, can be de-noised by using our proposed ICIPA algorithm.

3.3.3.2. Implementation

This algorithm has iterative and cellular characteristics. Figure 3.16 shows the block diagram of proposed restoration algorithm. Here there are four blocks and one switch. At the very first iteration, switch is in up position, but for all other iterations the switch is in down position. In the first block, neighborhood parameter C matrix is computed. For a cell the $C_{i,j}$ the output values of the neighbor cells are multiplied by some weights, summed then averaged according to the weights whereas the output of the cell $C_{i,j}$ is multiplied by 1. This can be formulized as,

$$C'_{i,j} = \frac{1}{\left(1 + \sum_{z=1}^m \frac{8}{z}\right)} \sum_{k=i-m}^{i+m} \sum_{l=j-m}^{j+m} \left(\frac{1}{\text{Max}(|k-i|, |l-j|)} \right)^2 C_{k,l}^{(t-1)} \quad (3.57)$$

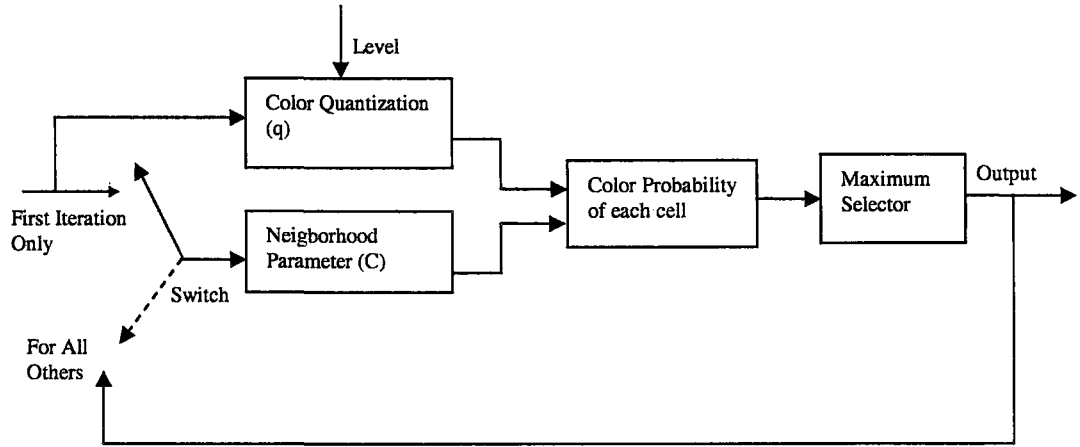


Figure 3.16: Iterative-Cellular Image Processing Algorithm Block Diagram.

here i and j are row and column indices, t is iteration index, m is the level of neighborhood which effects the cell and maximum is an operator which finds out the maximum of the values in the parentheses in the case of the conditions below.

$$Max(|k-i|, |l-j|) = |l-j| \begin{cases} |k-i| & \text{if } |k-i| \geq |l-j| \text{ and } |k-i| \neq 0, \\ |l-j| & \text{if } |l-j| \geq |k-j| \text{ and } |l-j| \neq 0, \\ 1 & \text{if } |k-i| = |l-j| = 0 \end{cases} \quad (3.58)$$

m can take any integer value and this designates the template dimensions. For $m=3$, the template is shown in Figure 3.17. at the first iteration $t=1$. $C(t-1)$ indicates the cells of the input image. Effect of the neighborhoods w_z decreases when the level of neighborhood increases. This effect can be shown as,

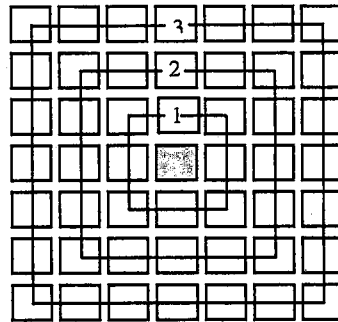


Figure 3.17: Neighborhood Relations (template) for $m=3$.

$$w_z = \left(\frac{1}{z}\right)^2, \quad z = 1 \dots m \quad (3.59)$$

and w_0 shows the loop of itself and it is 1. We cannot use (3.57) completely for every cell, since if the $C_{i,j}$ is close enough to the border of the image, the neighborhood level overflows and frames of the neighborhoods is quite different than template. In this type of border conditions, common parts of the template and image are considered and weighted average is computed. Before calculating the color probability of the cells, quantization levels are obtained from the image according to the color level. These quantization levels are chosen from the colors which are encountered in the image. To calculate the color probability of the cells, we use logarithmic form of Gaussian probability different than template. In this type of border conditions, common parts of the template and image are considered and weighted average is computed. Before calculating the color probability of the cells, quantization levels are obtained from the image according to the color level. These quantization levels are chosen from the colors which are encountered in the image. To calculate the color probability of the cells, we use logarithmic form of Gaussian probability density function.

$$\Pr(C_{i,j}^{(t)} = q_l) = -\frac{(C_{i,j}^{(t)})^2}{2\sigma^2} - \frac{(C_{i,j}^{(t)} - q_l)^2}{2\sigma^2} \quad (3.60)$$

Here l is quantization index, q_l shows the quantization or color level and σ^2 is the variance of the noise. Calculation of the variance of the noise is very easy for synthetic data but not so easy for real data. After computing the color probabilities of the cells, a maximum selector is applied. Most probable color is chosen from the quantization levels according to the color probabilities which are given in Equation (3.60). In this new method, the object size and template size are very correlated with each other. Thus, choosing the template level and the object size that we are trying to find out is very important and can be given as,

$$\frac{\text{number of pixel in the template}}{\text{number of pixel in the object}} \quad (3.61)$$

In simulations, we add Gaussian noise to synthetic data. We indicate noiseless real data as $x_{i,j}$ and noisy data as $C_{i,j}$. Noisy data can be shown as,

$$C_{i,j} = x_{i,j} + n_{i,j} \quad (3.62)$$

here $n_{i,j}$ is Gaussian noise. Additional noise in percentage can be calculated as,

$$n_p = \sqrt{\frac{\sum_i \sum_j (x_{i,j} - C_{i,j})^2}{\sum_i \sum_j x_{i,j}^2}} \quad (3.63)$$

3.3.4. Simulation Results

This study presents an efficient image transmission by means of a new proposed IC-TS (Iterative Cellular-Turbo System) method.

For experimental results, a sample image with 150x150 pixel resolution with 16-gray level is taken and 1/2 rate RSC encoder with AWGN channel model is investigated. Here the generator matrix is $g = [111:101]$, a random interleaver is used and the frame size is chosen as $N=150$, iteration (between ICIPA and decoder) is taken 1. At first, the pixels of the image is converted to 16 gray levels and then sliced to four bit-planes. All planes are then coded via RSC encoder and a random interleaver. The coded planes are corrupted with $SNR = 1\text{dB}, 0\text{dB}, -1\text{dB}, -2\text{dB}$ and -3dB and transmitted. Figure B.1, B.2, B.3, B.4 and B.5 show the corrupted, traditional turbo processed, ICIPA processed and reconstructed images via IC-TS system. Figure B.6 shows Bit Error Performance of conventional Turbo and IC-TS with various bit planes.

For comparison, ICIPA algorithm, conventional Turbo coding and IC-TS methods are evaluated and the results are compared. The results have shown that, to recover the image corrupted at 1 dB SNR or below, the well known image-processing and conventional Turbo algorithms are not satisfying as seen from the Figures B.1 to B.5. The proposed IC-TS algorithm gives good results from 0 dB SNR to -3 dB. So, more than 4 dB additional SNR improvement is achieved in the proposed algorithm (compared to that of conventional Turbo method). Hence, we conclude that IC-TS system will be a compromising approach in 2-D image transmission, in recovery of noisy signals.

3.3.5. Evaluations

In IC-TS, the original image is partitioned into 2^4 quantization levels. Then each of the 4-bit-plane is coded by Turbo encoder and transmitted over Additive White Gaussian Noise (AWGN) channel. Each of the noisy bit-plane values of the image is evaluated iteratively by a combined equalization block, which is composed Iterative

Cellular Image Processing Algorithm and Turbo decoder. In IC-TS, there is a feedback link between ICIPA and Turbo decoder. The result has an extra-ordinary satisfactory results of both bit error rate (BER) and image enhancement performance for less than 2 dB Signal-to-Noise Ratio (SNR) values. We can also achieve compression by using IC-TS systems. So, we use less memory storage and increase the data rate.

For comparison, the classical image processing (based Wiener filtering) algorithm, conventional Turbo coding, Adaptive Wiener-Turbo algorithm (AW-TS) methods are evaluated and the results are compared. We have seen that important additional SNR improvement is achieved in the proposed algorithm (as seen in figure B.6). Below 2dB SNR, the improvement is more than that of high SNR's. Hence, we conclude that ICIPA-TS system will be a compromising approach in 2-D image transmission, recovery of noisy signals.



4. COMPRESSION AND TRANSMISSION OF 2-D IMAGES WITH TURBO CODES

The compression is highly important for images obtained at a great cost, such as space, radar, optical or medical images, etc. In this case, even marginal loss of data may destroy some details required during further processing, or add artifacts that lead to erroneous interpretation.

Many applications require an image compression method that would perform:

- Lossless image compression with good compression ratios in order to use data storage more efficiently;
- Reconstruction of an image approximation if only the data carrying the details of the image is available. Obviously, it is preferable to obtain good-looking sketches by extraction of just small parts of the compressed data.

Typically, users had to choose different methods depending on whether the compression or fast inspection is desired.

This chapter is organized as follows. Chapter 4.1 and 4.2 achieves image transmission based bit plane slicing and JPEG compression approach. Especially in quick search of the image, chapter 4.1, for more compressions of the image, chapter 4.2 is attractive. When all the data is received, the entire original image will have been already constructed (lossless).

In Chapter 4.3, Progressive Transmission and Compression is handled. This scheme refers to image compression techniques that allow both, original image reconstruction without loss of any details and construction of picture approximations with the accuracy level depending on the amount of data available. This approach is very useful for frequent incremental visual inspections of images are needed.

In Chapter 4.4, image-oriented compression and progressive transmission is accomplished. The idea is to segment the image into regions and allocate the number of BPS according to their DCT coefficients. If the DCT magnitudes of subimages are high, more planes are allocated, meaning this region has a greater detail in. An image with lower details have less number of planes and may be more efficiently

compressed. This method allows a lossy image transfer with fast extraction of an image, while the bandwidth of communication channel is preserved.

4.1. Bit-Sliced Compression

4.1.1. Introduction

Low bandwidth requirements for image transfer over the communication channel are all too familiar. Image data is inherently suitable for processing that results in reduction of its size at the expense of image quality, which can still remain satisfactory for many applications.

High speed hardware enhancements are not likely to solve the problem. Our appetite for rapid data transfer will grow as more and more speed is available from processing hardware, while today's communication bottleneck will remain.

We present an approach that addresses this issue, proposing a new approach of compression and transmission for gray scale images using a powerful error correction code –Turbo code-. The algorithm can be easily extended to coloured images, either by processing each of the colour planes separately, or by transforming the image from RGB representation to other convenient representations.

The compression, where possible self similarity within the image, is identified by the concept of Bit Plane Slicing (BPS). This scenario reduces the significant amount of data required to reproduce the image at the receiver. Traditionally the other methods, transmitting the whole information of the images, have been time consuming, but the proposed approach promise to speed up the process.

The first step in compression is to partition the image into 2^N quantization levels, where N is denoted as the number of bit planes. Depending on the importance of the data it contains, each plane is allocated to get the portion of the total image's detail, such that the compressed image has the minimum data before transmission. We named this procedure “Bit Sliced Compression Turbo” (BSCT) system.

The second step, by the help of the approach explained above, we keep on reducing one bit slice at a time till we achieve satisfied recovery. Imagine that the image is composed of $N-1$ bit planes, ranging from plane 0 for least significant bit to plane $N-1$ for the most significant bit. In terms of N bit planes, plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image and plane $N-1$ contains all

the high-order bits. In other words, Plane zero is the Least Significant Bit (LSB) and Plane N-1 is the Most Significant Bit (MSB). Data rate can be increased up to (N-1) times, by only transmitting most important significant bit. Especially in quick search, such interesting results get importance. Then the other bits can be transmitted to obtain more accurate 2D images. Maximum resolution is obtained if all the bit slices from most significant to least significant are decoded at the receiver side without sacrificing any bandwidth efficiency. Thus, our bit slicing approach can be also an efficient way of compression technique.

4.1.2. Method

The compression and transmission performance of BSCT over Additive White Gaussian Noise (AWGN) channel is investigated. Proposed scheme is a combination of TC for transmission and BPS for compression. In AWGN environment, 2D adaptive noise removal filtering is used at the receiver side to achieve a better error performance. In BSCT, each binary correspondence of the amplitude values of each pixel is grouped in bit-planes. Then each bit symbol of bit planes are transmitted and at the receiver side, a combined structure, denoted as Wiener-Turbo decoder is employed. Wiener-Turbo decoder is an iterative structure with a feedback link between the second Turbo decoder and Wiener filter. The decoding process works iteratively till the desired output is obtained. The advantage of neighborhood relation of pixels is taken into account in BSCT scheme to get improvement on image enhancement. Although each bit slice is transmitted in serial way, the bit string is reassembled at the receiver in such a way that the original neighborhood matrix relations are taken into consideration before decoding process. Thus, instead of classical binary serial communication and decoding, in our scheme, the coordinates of the pixels are kept as in their original input data matrix. Instead of transmitting the N binary representation of pixels, each bit of the pixel is mapped to corresponding slice matrix.

In BSCT, data rate can be increased up to (N-1) times, by only transmitting most important significant BPS. Then the other BPSs can be transmitted to obtain more accurate 2D images. Maximum resolution is obtained if all BPSs from most significant to least significant planes are decoded at the receiver side without sacrificing bandwidth efficiency. Thus, our bit slicing approach can also be an efficient way of compression technique.

At the receiver, after each BPS is decoded, output bits are mapped to the previous places by reassembling the first bit from the first bit slice, second bit from

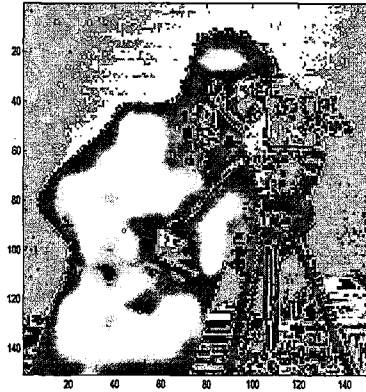
the second bit slice, and lastly the last bit from the last bit slice for each pixel. Hence, by collecting all the pixels, the recovery of the image is achieved. In the case of compression due to the resolution, not all the BPSs but some of them can be taken into account. Thus, corresponding quantized amplitude values of the pixels are found. Resolution loss is a general case in compression but by sacrificing resolution, we use less memory storage and increase data rate up to $(N-1)$ times by simply choosing any number of bit slices.

4.1.3. Experimental Results

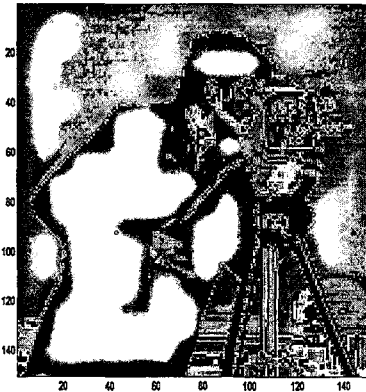
A sample image with 150x150-pixel resolution and 16-gray level is considered as a test image. The first step of the transmission is based on partitioning the image into four BPSs, which can be coded independently. So, some important and highly protected neighborhood relations are transmitted to the decoder. Each bit plane carries some portions of the details for the image as in Figure 4.1.

Figure 4.2 shows the proposed BSCT system. Here, the encoder is a $\frac{1}{2}$ rate Recursively Systematic Convolutional encoder. The generator matrix is $g = [111:101]$. Interleaver is in a random fashion. The frame size is chosen as $N=150$ and the iteration (between wiener filter and decoder) number is taken 1. The channel model is considered to be AWGN. While transmission, BPSs are corrupted with several Signal to Noise Ratios (i.e. SNR = 2dB, 1dB, 0dB, -1dB, -2dB and -3dB). Figure C.1, C.2, C.3, C.4, C.5 and C.6 show the reconstructed images from various combinations of BPSs via BSCT system.

At the receiver side, the bit-planes are reassembled by considering the neighborhood relationship of pixels in 2-D images. Each of the noisy bit-plane is evaluated iteratively by a combined block, which is composed of 2-D adaptive noise-removal filter and Turbo decoder.



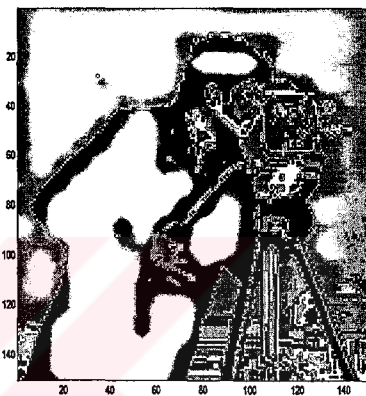
(a)



(b)



(c)



(d)



(e)



(f)

Figure 4.1: The effects of the various slice combinations for “cameraman.tif” image.

- (a) 0.Bit Plane+1.Bit plane (b) 0.Bit Plane+1.Bit plane+2.Bit plane
(c) 0.Bit Plane+1.Bit plane+2.Bit plane+3.Bit plane (d) 1.Bit plane+2.Bit plane
(e) 1.Bit Plane+2.Bit plane+3.Bit plane (f) 2.Bit plane+3.Bit plane

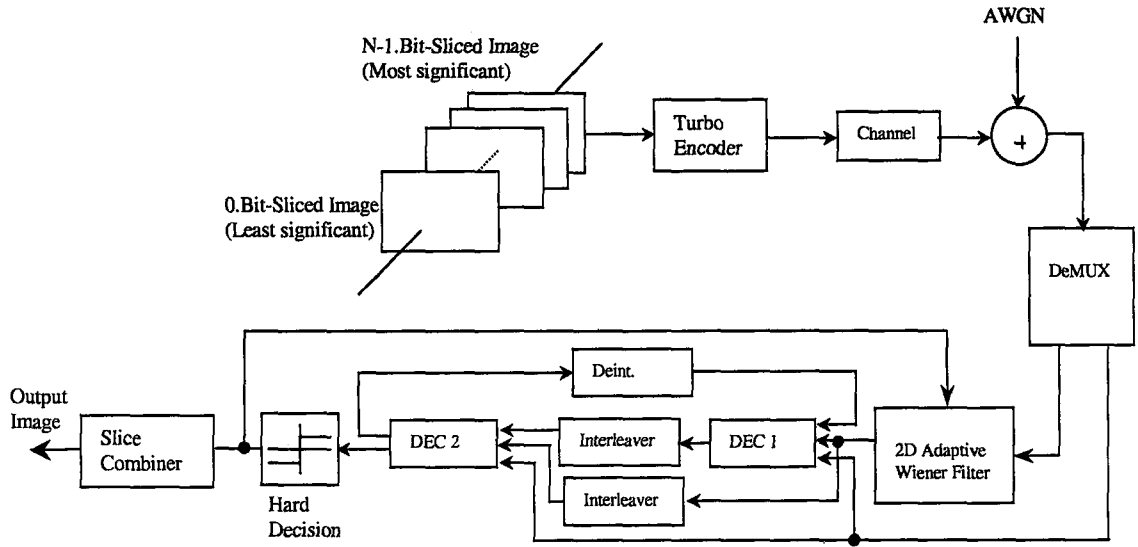


Figure 4.2: BSCT System Model

In BSCT, there is an iterative feedback link between Wiener filter and Turbo decoder. Our scheme employs a pixel-wise adaptive Wiener-based Turbo decoder and uses statistics (mean and standard deviation of local image) of local neighborhood of each pixel. Before MAP based decoding, the noise reduction is achieved with Wiener filter in order to obtain the highest probability of the symbols.

4.1.4. Evaluations

In this work, we propose a reliable and efficient compression-transmission system for 2D images using BPS (for compression) and TC (for error correction). In some cases, images need not be reproduced 'exactly'. An approximation of the original image is enough for most purposes, as long as the error between the original and the compressed image is tolerable. Also, during transmission, noisy channels severely degrade the quality of the image at the receiver. This forced using turbo code in the image transmission.

The traditional methods have been time consuming, but the proposed method promise to speed up the process enabling to get a better Bit Error Rate. The idea in compressing the image is to segregate it into different layers according to the certain neighborhood properties, which can be exploited by BPS. Also, some of the finer details in the image can be extracted with transmitting more slices. Initially, when 3rd Bit Slice (the most significant) is transmitted, %75 compression ratio (maximum

rate) is achieved. We keep on increasing one bit slice more in the transmission till we achieve satisfactory recovery as desired.

In compression, we used less memory storage and increased data rate up to $N-1$ ($N=4$, and $4-1=3$, in this study) times by simply choosing any number of bit slices, sacrificing resolution. Hence, we conclude that BSCT system will be a compromising approach in 2-D image transmission, recovery of noisy signals and image compression.

4.2. JPEG Compression

4.2.1. Introduction

JPEG is a standardized image compression mechanism that stands for Joint Photographic Experts Group. It is designed for compressing either full-color or gray-scale images of natural, real-world scenes. It works well on photographs, naturalistic artwork, and similar material; not so well on lettering, simple cartoons, or line drawings. JPEG handles only still images, but there is a related standard called MPEG for motion pictures.

JPEG is "lossy," meaning that the decompressed image isn't quite the same as the one it is started with. (There are lossless image compression algorithms, but JPEG achieves much greater compression than is possible with lossless methods.) JPEG is designed to exploit known limitations of the human eye, notably the fact that small color changes are perceived less accurately than small changes in brightness. Thus, JPEG is intended for compressing images that will be looked at by humans.

A useful property of JPEG is that the degree of lossiness can be varied by adjusting compression parameters. This means that the image maker can trade off file size against output image quality. We can make so small files if we don't mind poor quality; this is useful for applications such as indexing image archives. Computer-drawn images, such as ray-traced scenes, usually fall between photographs and cartoons in terms of complexity. The more complex and subtly rendered the image, the more likely that JPEG will do well on it. The same goes for semi-realistic artwork (fantasy drawings and such). But icons that use only a few colors are handled better by GIF. JPEG has a hard time with very sharp edges: a row of pure-black pixels adjacent to a row of pure-white pixels, for example. Sharp edges tend to come out blurred unless you use a very high quality setting. Edges this sharp are rare in

scanned photographs, but are fairly common in GIF files: consider borders, overlaid text, etc. The blurriness is particularly objectionable with text that's only a few pixels high. When we have a GIF with a lot of small-size overlaid text, it is not recommended to JPEG it.

Plain black-and-white (two level) images is not suitable to be converted to JPEG; they violate all of the conditions given above. It is needed at least about 16 gray levels before JPEG is useful for gray-scale images. It is noted that GIF is lossless for gray-scale images of up to 256 levels, while JPEG is not.

Gray-scale images do not compress by such large factors. Because the human eye is much more sensitive to brightness variations than to hue variations, JPEG can compress hue data more heavily than brightness (gray-scale) data. A gray-scale JPEG file is generally only about 10%-25% smaller than a full-color JPEG file of similar visual quality. But the uncompressed gray-scale data is only 8 bits/pixel, or one-third the size of the color data, so the calculated compression ratio is much lower. The threshold of visible loss is often around 5:1 compression for gray-scale images.

4.2.2. Method

The JPEG standard proposes compression / decompression algorithms aimed primarily at both gray scale and colour images. The standard proposes a *lossy* symmetrical encoding technique based on DCT and a uniform quantiser. DCT compression involves first dividing the image into 8x8 pixel blocks. A block may include *low-frequency* elements, a uniform background for example, or many *high-frequency* elements if it contains a lot of detail. Each block's information is transformed to the frequency domain. The transformed data is then filtered and quantised, a fancy way of saying that the high-frequency elements are thrown away as they are of less significance than low-frequency elements. The removal of these high frequency components forms the compression action. The more these high-frequency elements are discarded, the smaller the resulting file and the lower the resolution of the reconstituted image. Although JPEG represents a great step forward, it is not perfect. It suffers from two shortcomings: blocking, the undesired visibility of colour block boundaries, and undesired colour shift.

If one such 8x8 8-bit subimage is:

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} \quad (4.1)$$

which is then shifted by 128 results in

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix} \quad (4.2)$$

and then taking the DCT and rounding to the nearest integer results in

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \quad (4.3)$$

Quantization:

The human eye is fairly good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. This fact allows one to get away with greatly reducing the amount of information in the high frequency components. This is done by simply dividing each component in the frequency domain by a constant for that component,

and then rounding to the nearest integer. This is the main lossy operation in the whole process. As a result of this, it is typically the case that many of the higher frequency components are rounded to zero, and many of the rest become small positive or negative numbers.

A common quantization matrix is:

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (4.4)$$

Using this quantization matrix with the DCT coefficient matrix from above results in:

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.5)$$

For example, using -415 (the DC coefficient) and rounding to the nearest integer
 $\text{Round}(-415/16) = \text{Round}(-25.9375) = -26$

Decoding:

Decoding to display the image consists of doing all the above in reverse. Taking the DCT coefficient matrix (after adding the difference of the DC coefficient back in)

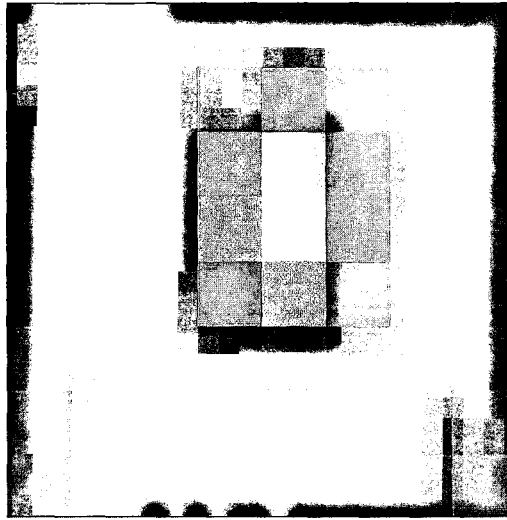
$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.6)$$

and multiplying it by the quantization matrix from above results in

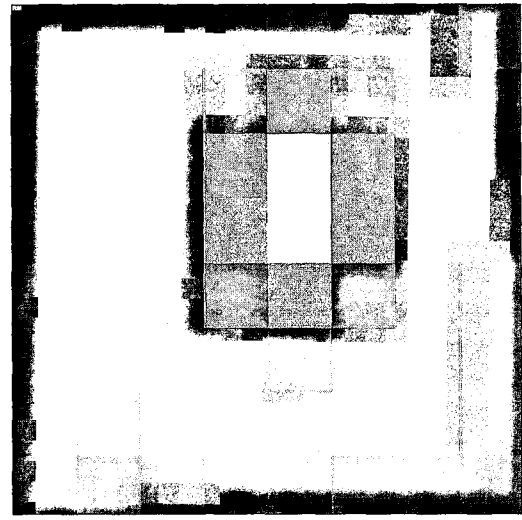
$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.7)$$

which closely resembles the original DCT coefficient matrix for the top-left portion. Taking the inverse DCT results in an image with values (still shifted down by 128)

$$\begin{bmatrix} -68 & -65 & -73 & -70 & -58 & -67 & -70 & -48 \\ -70 & -72 & -72 & -45 & -20 & -40 & -65 & -57 \\ -68 & -76 & -66 & -15 & 22 & -12 & -58 & -61 \\ -62 & -72 & -60 & -6 & 28 & -12 & -59 & -56 \\ -59 & -66 & -63 & -28 & -8 & -42 & -69 & -52 \\ -60 & -60 & -67 & -60 & -50 & -68 & -75 & -50 \\ -54 & -46 & -61 & -74 & -65 & -64 & -63 & -45 \\ -45 & -32 & -51 & -72 & -58 & -45 & -45 & -39 \end{bmatrix} \quad (4.8)$$



(a)



(b)

Figure 4.3: Example of JPEG compression/Decompression
(a) Original Image (b) Restored Image

and adding 128 to each entry

$$\begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix} \quad (4.9)$$

This is the uncompressed subimage and can be compared to the original subimage (also see images to the right) by taking the difference (original - uncompressed) results in error values

$$\begin{bmatrix} -8 & -8 & 6 & 8 & 0 & 0 & 6 & -7 \\ 5 & 3 & -1 & 7 & 1 & -3 & 6 & 1 \\ 2 & 7 & 6 & 0 & -6 & -12 & -4 & 6 \\ -3 & 2 & 3 & 0 & -2 & -10 & 1 & -3 \\ -2 & -1 & 3 & 4 & 6 & 2 & 9 & -6 \\ 11 & -3 & -1 & 2 & -1 & 8 & 5 & -3 \\ 11 & -11 & -3 & 5 & -8 & -3 & 0 & 0 \\ 4 & -17 & -8 & 12 & -5 & -7 & -5 & 5 \end{bmatrix} \quad (4.10)$$

with an average absolute error of about 5 values per pixels:

$$\frac{1}{64} \sum_{x=1}^8 \sum_{y=1}^8 |e(x, y)| = 4.8125 \quad (4.11)$$

i.e. The error is most noticeable in the bottom-left corner where the bottom-left pixel becomes darker than the pixel to its immediate right.

4.2.3. Experimental Results

Figure 4.4 shows the proposed system. A sample image with 150x150-pixel resolution and 256-gray level is considered as a test image. The first step of the transmission is to compress the image with JPEG. Here, the encoder is a $\frac{1}{2}$ rate Recursively Systematic Convolutional encoder. The generator matrix is $g = [111:101]$. Interleaver is in a random fashion. The frame size is chosen as $N=150$. The channel model is considered to be AWGN. While transmission, the DCT coefficients (obtained as a result of JPEG compression) are corrupted with several Signal to Noise Ratios (i.e. SNR = 2dB, 4dB, 6dB). Figure 4.5, 4.6 and 4.7 show the reconstructed images obtained JPEG-Turbo system.

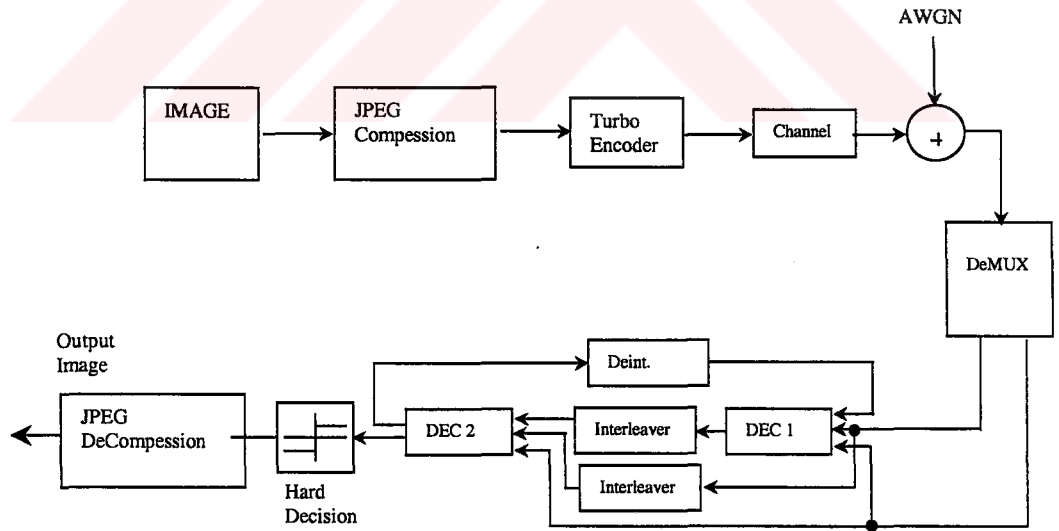
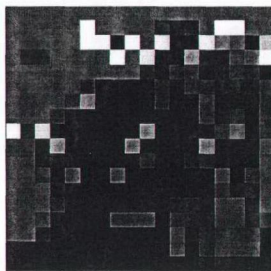


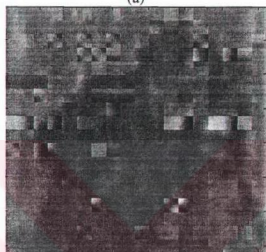
Figure 4.4: System model of JPEG-Turbo system



(a)



(b)



(c)



(d)



(e)



(f)

Figure 4.5: Compressed, Turbo Processed and Restore images at SNR=2dB

(a) Original Image

(b) Restored image with 1 coefficient (compression ratio = % 98)

(c) Restored image with 2 coefficients (compression ratio = %94)

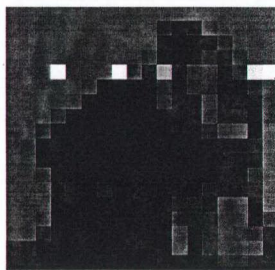
(d) Restored image with 4 coefficients (compression ratio = %75)

(e) Restored image with 6 coefficients (compression ratio = %44)

(f) Restored image with 8 coefficients (compression ratio = %0)



(a)



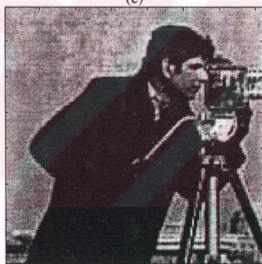
(b)



(c)



(d)



(e)



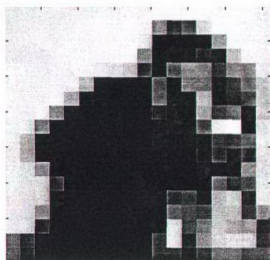
(f)

Figure 4.6: Compressed, Turbo Processed and Restore images at SNR=4dB

- (a) Original Image
- (b) Restored image with 1 coefficient (compression ratio = % 98)
- (c) Restored image with 2 coefficients (compression ratio = %94)
- (d) Restored image with 4 coefficients (compression ratio = %75)
- (e) Restored image with 6 coefficients (compression ratio = %44)
- (f) Restored image with 8 coefficients (compression ratio = %0)



(a)



(b)



(c)



(d)



(e)



(f)

Figure 4.7: Compressed, Turbo Processed and Restore images at SNR=6dB

(a) Original Image

(b) Restored image with 1 coefficient (compression ratio = % 98)

(c) Restored image with 2 coefficients (compression ratio = %94)

(d) Restored image with 4 coefficients (compression ratio = %75)

(e) Restored image with 6 coefficients (compression ratio = %44)

(f) Restored image with 8 coefficients (compression ratio = %0)

4.2.4. Evaluations

Making image files smaller before transmission is a win for speeding up, bandwidth requirements and storing needs. JPEG can easily provide 20:1 compression making a big difference for transmission time! In this work, we proposed JPEG compression and Turbo transmission for 2D images (named as JPEG-Turbo system).

In Figure 4.5. SNR is taken 2dB, it can be seen that the results are not satisfactory at that level of noise. In BSTC system (at the 2dB and above), the output image was high quality for which 0% compression ratio and we had still successive approximations for different compression ratios. it can be concluded that, the reason of this fact is due to usage of image processing algorithms before decoding. Satisfactory recoveries are reached at above 4dB SNR in JPEG-Turbo system. In this system, there is no any neighborhood relations between the (compressed) coded data, so it is not meaningful to use 2D filters before MAP algorithm. Also in BSCT system, we know that the data is transmitted in the time domain, but in JPEG-Turbo system, we transmit the DCT coefficients, so in the noisy environments where bit-error is high, the corrupted DCT coefficients played a dominant role for recovering the image.

In the proposed system, successive approximation is achieved in the environment for which SNR is high. The performance has a compromising approach while maintaining a very good degree of compression. Figure 4.7 (c), (d) and (e) shows this fact, in this case we have 94%, 75% and 44% attractive compressions.

Evaluations of BSTC and JPEG-Turbo systems can be summarized as; The fundamental advantage of BSCT system is it can recover the image at very low SNR's, but JPEG-Turbo system needs SNR to be high. On the other hand, JPEG achieved much greater compressions, so JPEG based transmission will need considerably less transmission time.

For the future work, the compression algorithm can be significantly improved if the coefficients selection would be adaptive, i.e. in each DCT block we would select a different number of coefficients with the largest amplitude. Thus smooth regions of an image can be represented by a small number of coefficients, whereas edges and high frequency textures would be represented by large number of coefficients. This will solve the problem of edges, while leaving the algorithm efficient.

4.3. Progressive Transmission and Compression of Turbo Coded Images

4.3.1. Introduction

This chapter describes a method, that allows progressively image transfer over communication channel, in which the quality of the image passed to the receiver can be adjusted, Sherwood and Zeger (1998). Images are converted into the appropriate format – bit plane sliced and turbo coded- on the transmitter side. With the desired image quality specified, the transmitter sends coded bit planes of image through the channel. The receiver cumulatively builds the coded data for the image while applying additional error correction –MAP (Maximum A posteriori Probability) algorithm- operations on the data obtained so far. The ability to specify desired image quality allows the receiver to satisfy constraints imposed by line bandwidth and response time. This method is illustrated with 4 bit gray scale images for simulation and can be extended for color images.

The Progressive Transmission and compression of Turbo Coded Image (PTCTC) procedure is based on the idea that only partial information from the original image is provided to the receiver at each iteration –i.e. bit plane iteration-. The receiver will use the limited information it has obtained, perform enhancement if desired or necessary, and displays the image. In the next iteration, additional information about the image will be sent to the receiver (like WWW browsers). The new data, together with data already on the receiver side, is used to build a new and more complete image at the receiver, achieving a better image representation. As the procedure is iterated, for more enhancement, the receiver gathers more image data so that eventually it is able to recreate the image in its original form.

4.3.2. Method

The PTCTC procedure will slice the original image by partitioning it along its bit planes and transmitting them with Turbo Coding scheme. For an 8 bit gray scale image this will amount to 8 planes, for an 4 bit gray scale image it will amount to 4 planes. In the case of true color (24 bit) image the same procedure can be applied for each of the three bands (RGB) producing 24 such slices (for 8 planes case). For simplicity of implementation, we consider only 4 bits/pixel for grayscale images; however, all the following assertions can be easily adapted to 8 or more bits/pixel color images considering each color component separately.

Proposed scheme uses the combination of Turbo Coding for transmission (error correction) and BPS for compression. During transmission, noisy channels severely degrade the quality of the image at the receiver. This necessitated using the error control code in the progressive transmission.

When the bit plane of image is transmitted from the communication channel, it is added to the existing image data (if available) at the receiver side. Since bit planes arrive as coded from RSC encoders, they are decoded via MAP algorithm and pasted on "top" of the planes currently defining an image. The additional bit plane provides a better color or gray scale resolution as resolution gets increased by a factor of two. Maintenance of the order in which bit planes are composed to represent an image is controlled by the transmitter.

This scheme allows both: (1) original image reconstruction without loss of any details; and (2) construction of picture approximations (sketches) with the accuracy level depending on the amount of data available. This approach is very suitable where, frequent incremental visual inspections of images is necessary. By the term "image inspection" we mean here fast extraction of image sketches with desired levels of accuracy. The approximations of the image can be progressively updated in parallel to data extraction. The speed of this operation depends on the amount of data that should be retrieved from the storage for sketch reconstruction.

4.3.3. Advantages

The power of this method comes from the ability to control the amount of data to be transferred. In some cases, the first iteration can prove sufficient for a given application, considering constraints of bandwidth and response time.

Notably, we can compare this to sending an image by fax. In many cases, the low quality of the received image can be satisfactory. The advantage here is that if it is not, the receiver can ask for more image data and combine it with what it has received previously. The crucial point is that the receiver assembles data received in a previous iterations with a data from the most recent one to produce a higher quality image.

Also worth mention is that by transferring images in this way we are not introducing any overhead in data transmission and the stateless nature of the transmitter and receiver is preserved. No image data is sent twice through the channel. Repeated

receiver requests for more data, if needed, are the only overhead which is, for all practical purposes, negligible.

This technique achieves high compression ratios by taking only consideration the most significant bit planes. Furthermore, the quality of the image is controlled using progressive transmission – in each stage the user receives a more refined image.

4.3.4. System Model

The system consists of image partitioner, bit plane slicer, plane selector, turbo encoder (transmitter), turbo decoder (receiver), and subimage combiner as shown in Figure 4.8.

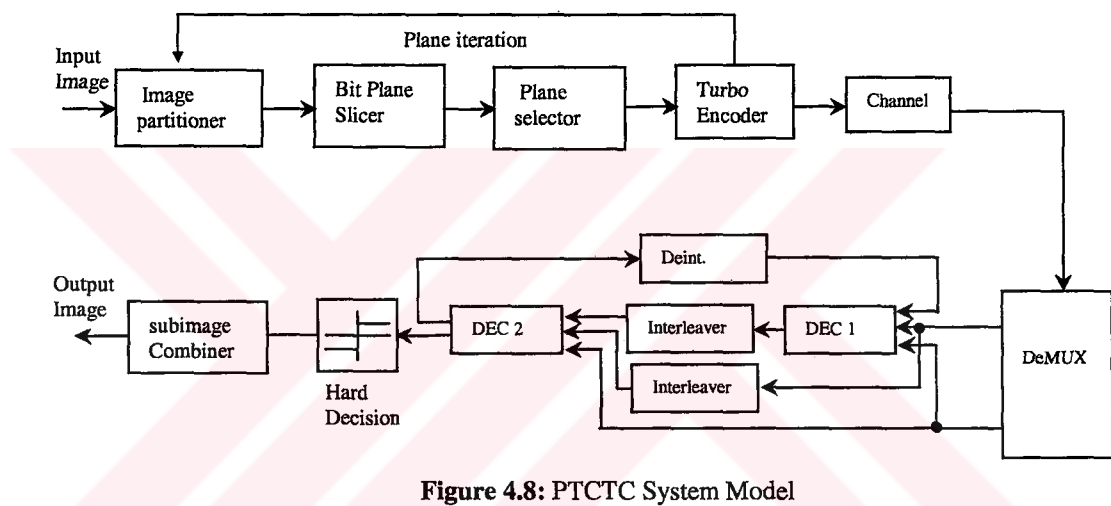


Figure 4.8: PTCTC System Model

4.3.4.1. Image Partitioner

The image is first sub-divided into regions in image partitioner section. The subimage size of the image can be adjusted. But if it is taken considerably too low, the bit error rate may increase due to MAP algorithm. Figure 4.9 shows how the image is being partitioned.

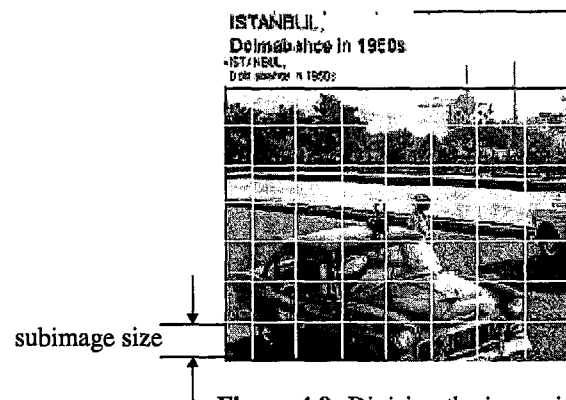


Figure 4.9: Division the image into partitions

4.3.4.2. Bit Plane Slicer

This decomposition reveals that only some highest order bits contain visually significant data. Data rate, in this case, can be increased up to $(N-1)$ times, by only transmitting most important significant bit planes. Especially in quick search, such interesting results get importance. Then the other bit planes are transmitted to obtain more accurate images. Maximum resolution is obtained if all the planes from most significant to least significant are decoded at the receiver side without sacrificing any bandwidth efficiency. What this means is that by separating the image into bit planes, we immediately have a method of identifying more important and less important information. This means that more of the bit planed images will contain areas of continuous tone and this in turn means that the image will more easily compress. Thus, our bit slicing approach can be also an efficient way of compression technique.

In any plane, each pixel representation is a single binary digit either 0 or 1. Because of the way binary code is constructed, it is normally found that the most significant information within the image is contained in the top few bit planes, i.e. those represented by the first few digits of the code. The images are all bilevel, as the pixels are only represented as 1's or 0's. Because of the nature of binary code each individual bit plane represents the image being thresholded at a different 'resolution'.

4.3.4.3. Turbo Encoder

In our study, the subimages are sliced into 4 planes i.e. each pixel in the image is represented by 4 bits (or 16 gray levels), which is ranging from plane 0 for the least significant bit to plane 3 for the most significant bit. In terms of 4-bit bytes, plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image and plane 3 contains all the high-order bits. Note that the most significant bit plane (3.Bit Plane) contains visually significant data. The other bit planes contribute to more subtle details in the image. Also plane 3 corresponds exactly with an image thresholded at gray level 8.

We begin from the MSB bit planes of each subimages. Figure 4.10 shows the encoding strategy. At the first plane iteration, all the 3rd planes are coded. At the receiver side, these are decoded and so, the binary image is obtained. At the next iteration, the 2nd planes are processed. In this case we have 4 gray level accuracy. At third iteration, the 1st planes and at fourth iteration, 0th planes are sent enabling 8 and 16 level accuracy is achieved, respectively.

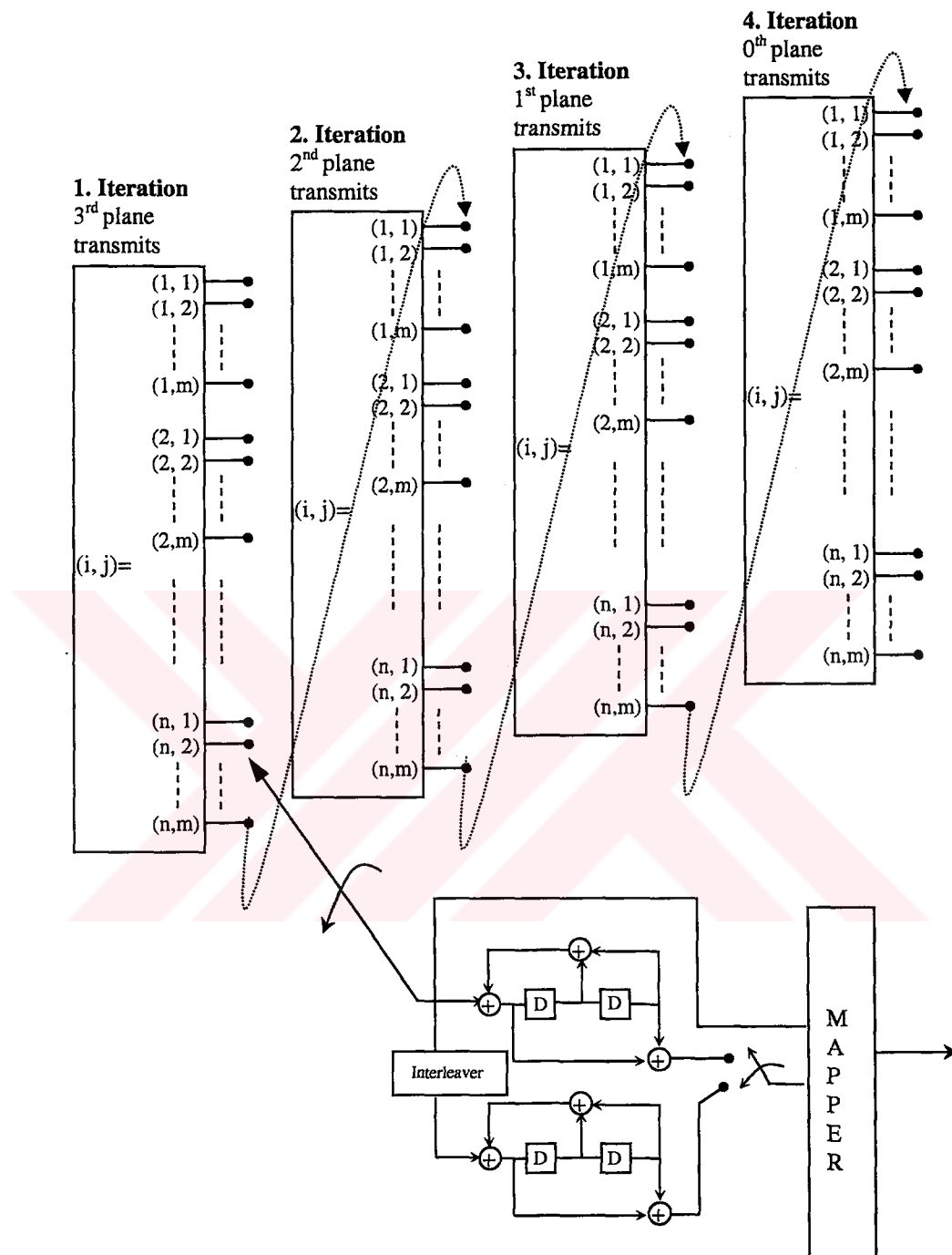


Figure 4.10: Encoding Strategy of PTCTC System

4.3.4.4. Turbo Decoder

The incoming coded data (from the most significant to least significant) are decoded in this section. At the output of the hard-decision, the planes are progressively collected at the top of the image (see figure 4.11). The result is progressively enhancing image.

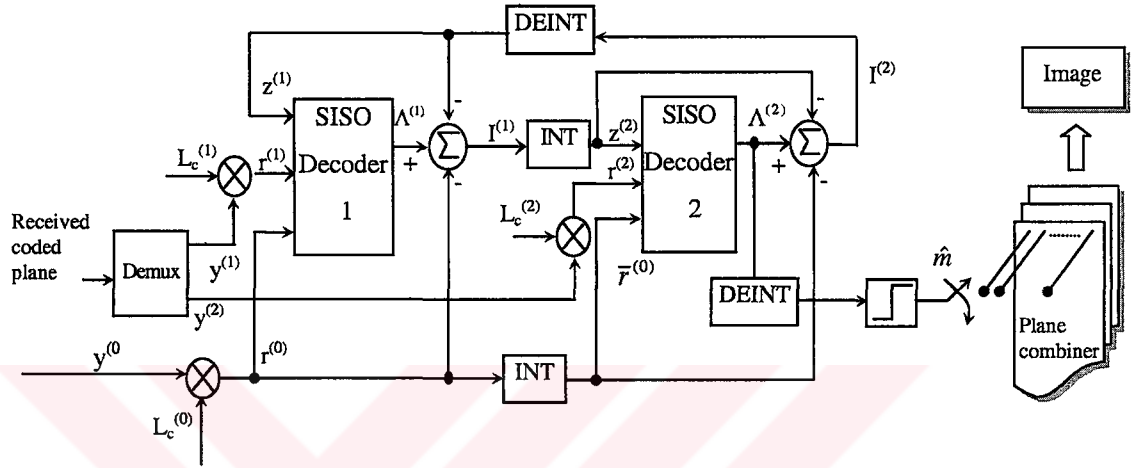


Figure 4.11: Encoder of PTCTC

4.3.4.5. Sub-image combiner

The proposed scheme ends with this section. When the soft bit information of the planes is de-interleaved and hard decisioned, they are evaluated to recover the subimages. The subimages are then put in their original places, resulting to reconstruct the whole image. This process is exactly the inverse process of bit plane slicing. In other words, the planes are recombined at the order from most significant to least significant. It may be lossy or lossless: lossy, by discarding less significant bit planes (the result is an approximation of the original image); lossless, by taken into consideration all planes. If all planes are taken into account, the image is reconstructed exactly.

The flow chart in Figure 4.12 illustrates the idea of the proposed system, which can be examined at transmitter and receiver sides.

Transmitter side

1. Partition the image and obtain subimages,

Receiver side

1. Take the coded data.
2. Encode it and take $k=3$ (i.e. the incoming data is the most significant),
3. Put it into (1,1) pixel (i.e. $i=1, j=1$) of the 3rd bit plane,
4. Increase i and j respectively, put the encoded data at these locations.
5. When the whole surface of 3rd plane is spanned, decrease k . This means the lower bit plane will be taken into account.
6. Repeat from 1 to 6 until $k=0$. When the last plane is reconstructed, we have 4 bit planes (whose their pixels are made up of binary values).
7. Multiply the 3rd plane by 2^3 , 2nd plane by 2^2 , 1st plane by 2^1 and last 0th plane 2^0 and add all them resulting to get recovered image.
8. If the resolution is satisfying at the end of step 5, step 6 can be skipped.

4.3.5. Simulation Results

In this section, we present simulation results that illustrate the performance of the proposed (PTCTC) algorithm. In our example, $\frac{1}{2}$ rate RSC encoder with AWGN channel model is investigated. Here the generator matrix is $g = [111:101]$, a random interleaver is used and the subimage size is chosen as 100. At first, the pixels of the image is converted to 16 gray levels and then sliced to four bit-planes. All planes are then coded via RSC encoder and a random interleaver. The coded planes are corrupted with $\text{SNR} = 4\text{dB}$ for simulation and they are transmitted progressively. At the receiver side Figure 4.13 (a), (b), (c) and (d) is obtained at 1st, 2nd, 3rd and 4th iterations, respectively. At the first iteration, we have %75 compression ratio. We see that the text (the region is circled as **I**) in the image can be read easily at 1st iteration (i.e. by taking only the most significant plane). This level of compression is very suitable for fax communication but the details are lost. Two ships which are sailing in the region **II** are only selectable in the second iteration. But we have loss of some other details. The sea (region **III**) is not recognized until iteration 3. After the last iteration, the other part of the city (and also the whole details which exists in the original image) begins to appear (region **IV**).

The results have shown that increasing the number of the bit planes enables us to study with more details. But if the image is not very detailed, it is no need to increase the number of the bit planes in order to avoid complexity.

4.3.6. Evaluations

This algorithm losslessly compresses and transmits an image into several segments corresponding each to information about the image at various levels of amplitude accuracy. With the transmission of the first MSB bit planes (As a matter of fact, the correct order of operations is important for less time consuming and best image quality at first sight. i.e. 3rd bit plane must always be transmitted first and then 2nd, 1st and last 0th. The image is reconstructed at its lowest accuracy level. Transmission the further planes reconstructs the image at progressively better levels of accuracy. The transmission whole bit planes reconstructs the original image exactly.

This algorithm is particularly useful in bandwidth-constrained storage/retrieval applications when the user requires the closest approximation to the original image, for a given bandwidth constraint. Less accurate approximations take less bandwidth to transmit while allowing the user to make a decision as to whether the full resolution is desired or not, minimizing total transmission time per decision. This algorithm is a progressively lossless compression/decompression algorithm and requires no additional storage since all successive approximations of the image can be reconstructed directly from the upcoming image bit planes. Furthermore, the lossless compression ratio obtained with this algorithm is comparable to or better than the best standard lossless compression algorithms, which do not allow for progressively accurate approximations.

Figure 4.14 shows the BER values of the planes during the transmission. As decoder iteration increases, BERs get lower due to MAP algorithm.

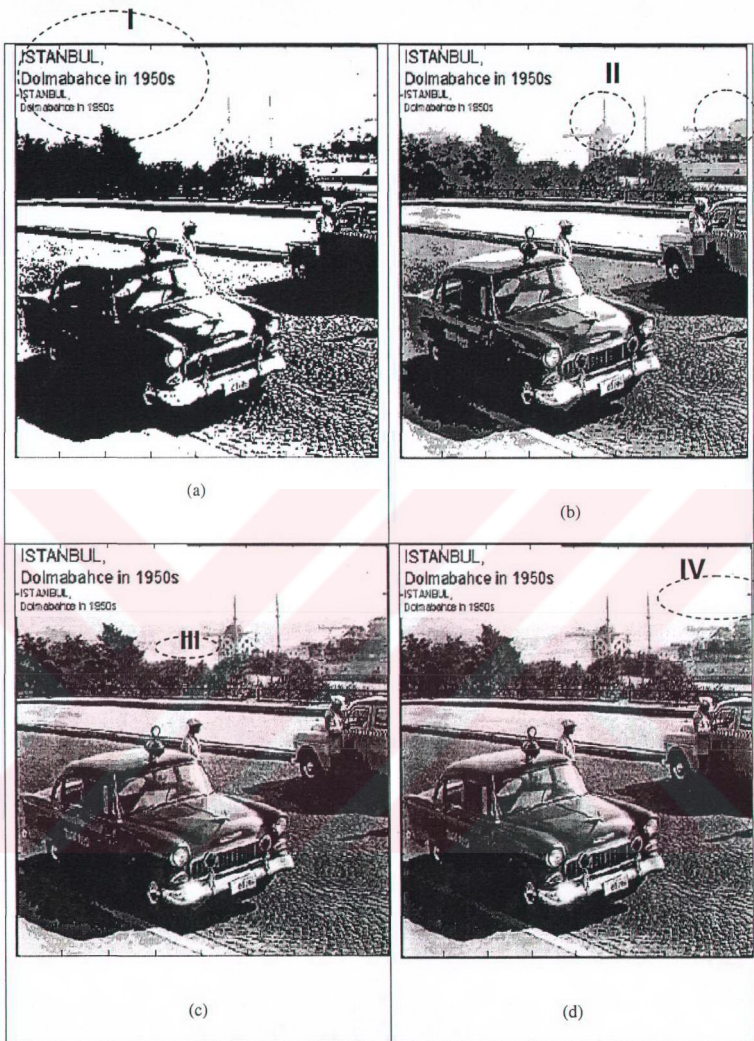
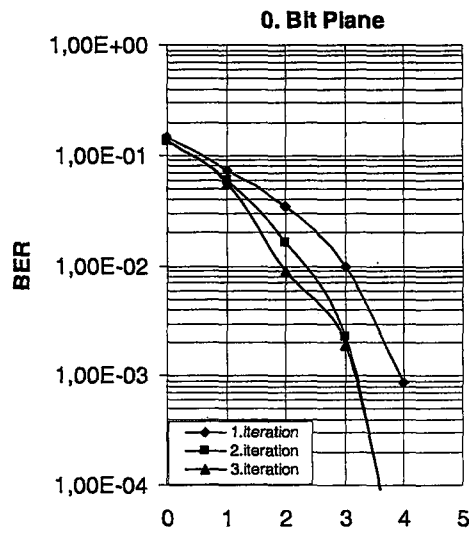
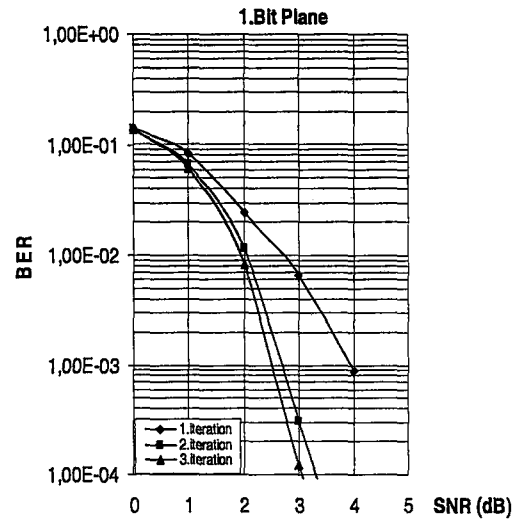


Figure 4.13: Progressively Receiving Image.

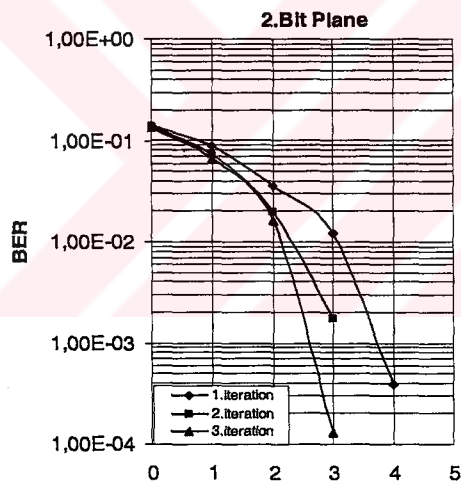
- (a) Only 3rd bit plane is received (Plane Iteration = 1, compression ratio = 75%)
- (b) 3rd and 2nd bit planes are received (Plane Iteration = 2, compression ratio = 50%)
- (c) 3rd, 2nd and 1st bit planes are received (Plane Iteration = 3, compression ratio = 25%)
- (d) All bit planes are received (Plane Iteration = 4, compression ratio = 0%)



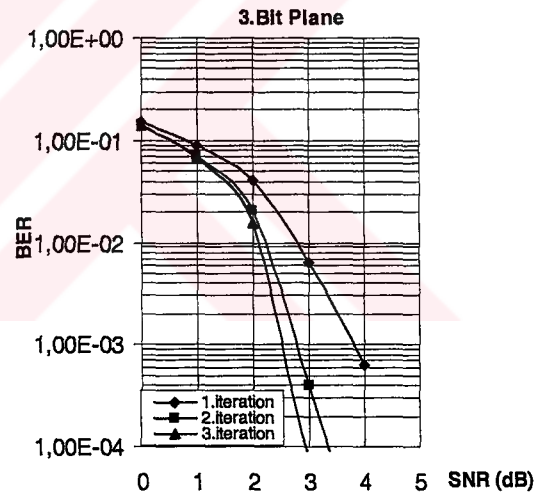
(a)



(b)



(c)



(d)

Figure 4.14: Bit Error Rates of the Planes.

(a) BER of 0.bit plane (b) BER of 1.bit plane (c) BER of 2.bit plane (d) BER of 3.bit plane

4.4. Image Oriented Compression and Progressive Transmission

4.4.1. Introduction

Often a rough approximation (preview) of an image is sufficient for the user to decide whether or not it should be transmitted in greater detail. The refinement methods can be classified as “*detail on demand*”, where the receiver asks greater detail, “*progressive refinement*”, where the system transmits and displays more detail automatically and “*error correction*”, where the effect of the noise has a tolerable impact on the resulted image.

We present an approach that addresses this issue. A progressive image transmission with image oriented compression capability has been developed to provide image transfer, in which the quality of the image passed to the receiver can be controlled. It is important to note that the method discussed, allows for a lossy and non lossy image transfer with a negligible overhead, while the bandwidth of communication channel is preserved.

4.4.2. Method

In this chapter, we propose a method, that allows Turbo coded images to transfer efficiently, in which the data amount and the quality passed to the receiver can be adjusted. This efficient method successfully achieves three objectives - progressive transmission, image oriented lossless/lossy compression and error correction.

- The objective of Progressive Transmission is based on the idea that only partial information from the original image is provided to the receiver at each iteration
- The idea of image oriented compression is based on partitioning the image into small regions (subimages) and convert them to the appropriate format – bit plane sliced (BPS)- and then allocate the number of BPS according to their DCT coefficients. If the DCT magnitudes of subimages are high, this region has a greater detail (VISL, 2005) and hence more planes are allocated. An image with lower details have less number of planes and may be more efficiently compressed. This method allows a lossy image transfer with fast extraction of an image, while the bandwidth of communication channel is preserved.

- For the objective of error correction, Turbo codes have been considered. They are the most powerful error control code due to their practical importance which have low decoding complexity and enhances the transmitted information's efficiency and accuracy with the rate close to the channel capacity.

We propose here a combined method, which allows to transmit the reduced data of the desired image quality specified, progressively transmission in order to avoid time consuming and enhancement, to get a better quality of the image while saving bandwidth limitations.

For compression, the similarity within the image, is identified by the concept of Bit Plane Slicing (BPS). This scenario reduces the significant amount of data required to reproduce the image at the receiver. Traditionally the other methods, transmitting the whole information of the images, have been time consuming, but the proposed approach promise to speed up the process.

The ability to specify desired image quality allows the receiver to satisfy constraints imposed by line bandwidth and response time. This method is illustrated with 4 bit gray scale images for simulation and can be extended for color images.

4.4.3. System Model

The system consists of image partitioner, DCT, Bit Plane Allocation, plane selector, turbo encoder (transmitter), turbo decoder (receiver), and subimage combiner sections as shown in Figure 4.15.

Image partitioner, turbo encoder, turbo decoder, subimage combiner was mentioned in the previous sections and 2-D DCT concept is assumed to be well known. But, the priority of plane coding in this system is so different. Figure 4.15 shows the general system model and Figure 4.16 expresses the coding strategy at the transmitter side.

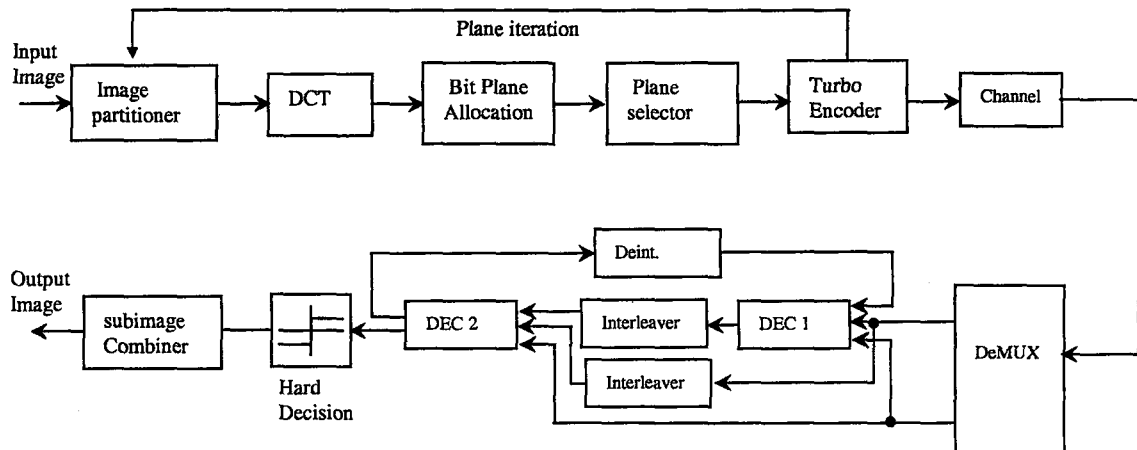


Figure 4.15: System Model.

When the soft bit information of the planes is de-interleaved and hard decisioned, they are evaluated in the image combiner section. The subimages are then reconstructed progressively as the plane iteration increases.

The flow chart in Figure 4.17 illustrates the general idea of the whole system. This chart can be examined at transmitter and receiver sides.

Transmitter side

1. Partition the image and obtain subimages,
2. Calculate DCT coefficients
4. Allocate BPSs to each subimage according to 2
5. Take MSB bit plane (of the first subimage)
6. Code this plane and send it over.
7. Take the next subimage and proceed to 5. Repeat until all subimages are processed.
6. Take lower significant bit as MSB and repeat 5

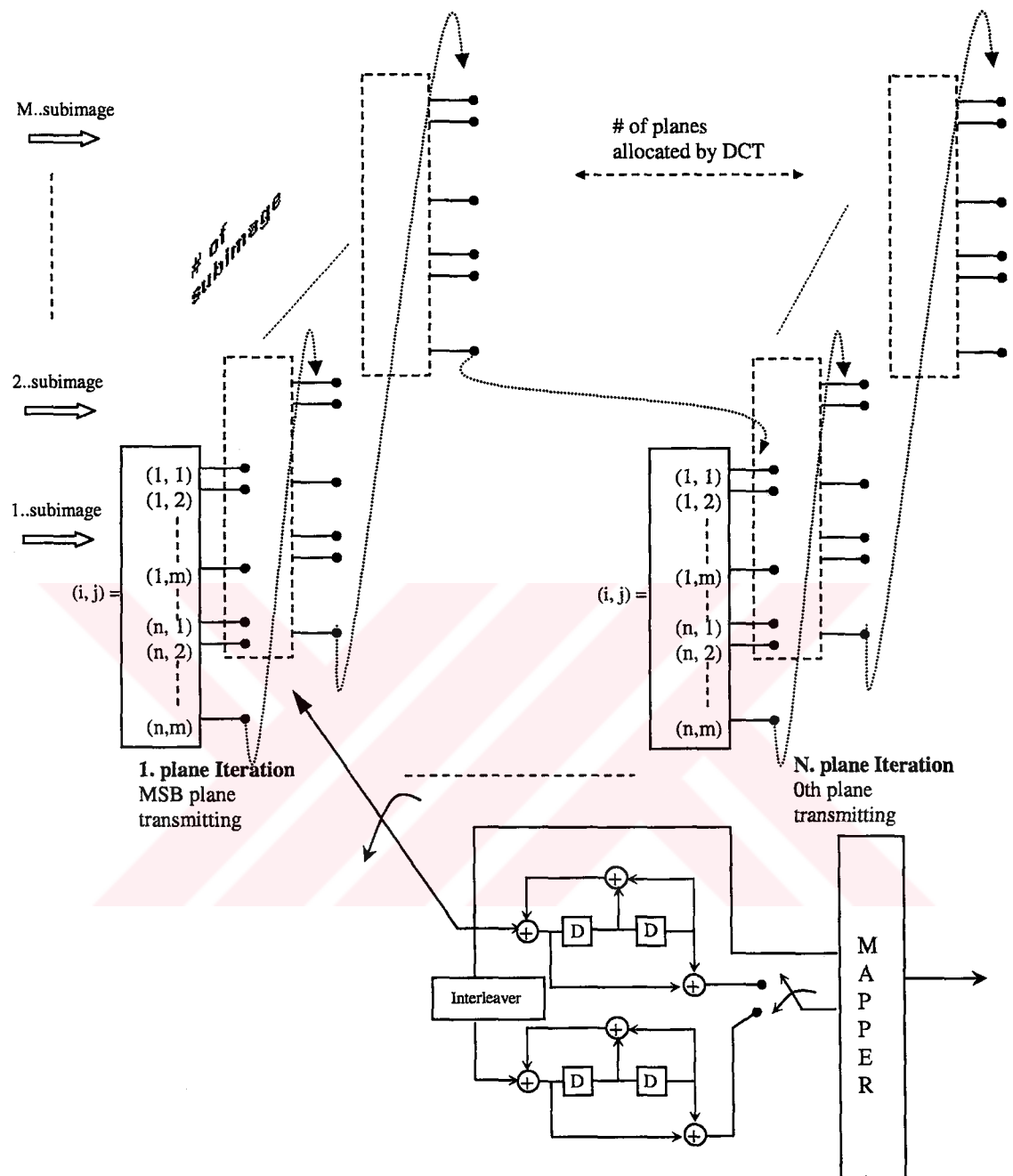


Figure 4.16: Encoding Strategy of the Proposed System.

Receiver side

1. Take the coded data.
2. Encode it and take $k=3$ (i.e. the incoming data is the most significant),
3. Put it into (1,1) pixel (i.e. $i=1, j=1$) of the 3rd bit plane,
4. Increase i and j respectively, put the encoded data at these locations.
5. When the whole surface of 3rd plane is spanned, decrease k . This means the lower bit plane will be taken into account.
6. Repeat from 1 to 6 until $k=0$. When the last plane is reconstructed, we have 4 bit planes (whose their pixels are made up of binary values).
7. Multiply the 3rd plane by 2^3 , 2nd plane by 2^2 , 1st plane by 2^1 and last 0th plane 2^0 and add all them resulting to get recovered image.
8. If the resolution is satisfying at the end of step 5, step 6 can be skipped.

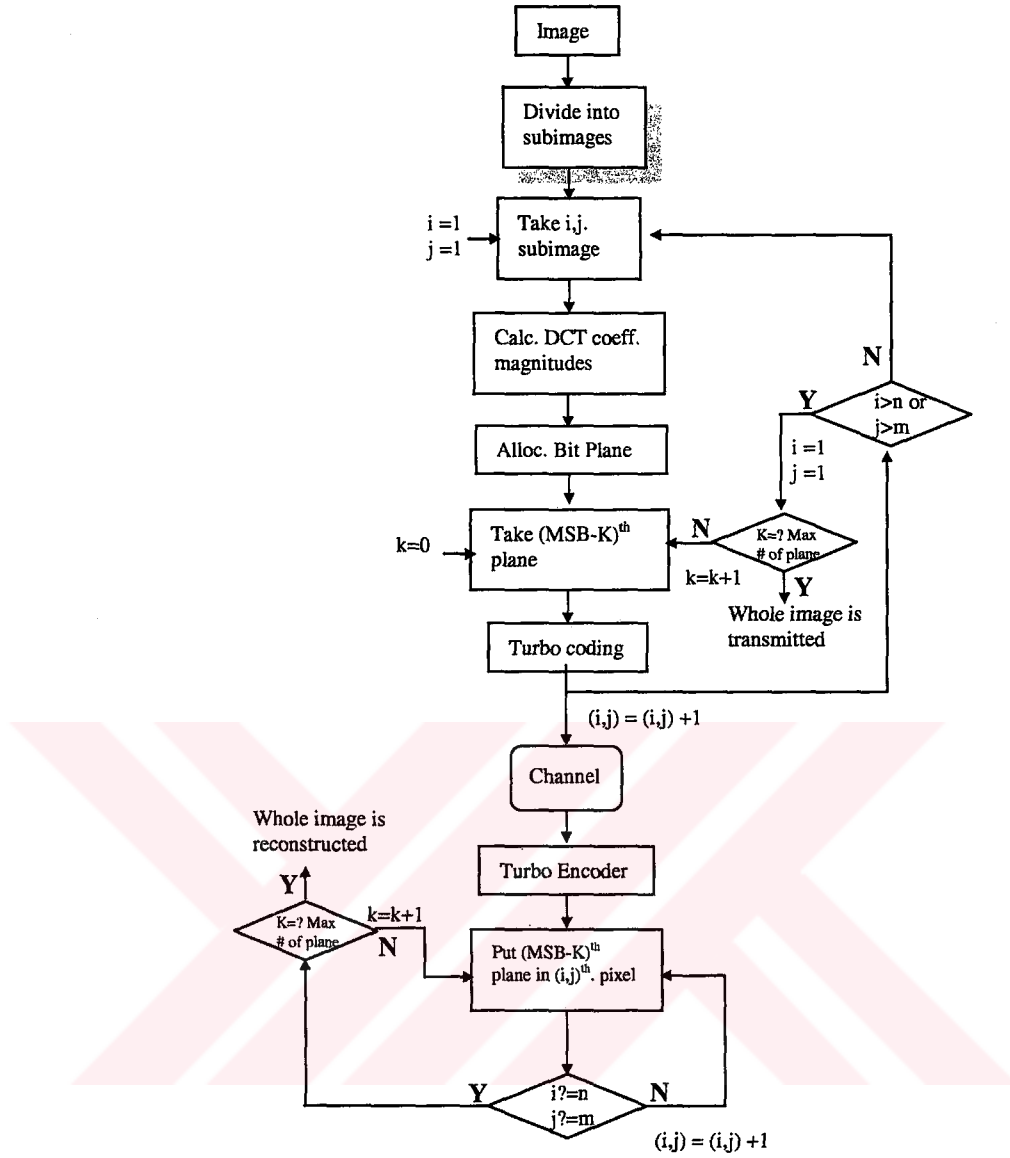


Figure 4.17: Flow chart of the whole system

4.4.4. Mathematical Evaluations

Let I be the image whose the size is $U \times V$.

$$[I(i, j)]_{U \times V} = \begin{bmatrix} I_{1,1}(i, j) & I_{1,2}(i, j) & \cdots & I_{1,r}(i, j) \\ I_{2,1}(i, j) & \cdots & \cdots & \cdots \\ \vdots & & \ddots & \\ I_{p,1}(i, j) & \cdots & \cdots & I_{p,r}(i, j) \end{bmatrix}_{U \times V} \quad (4.12)$$

where, $\mathbf{I}_{x,y}(i, j)$, ($x = 1 \dots p, y = 1 \dots q$) is the subimage,
 $p = \frac{U}{\text{mask size}}, r = \frac{V}{\text{mask size}}$ and mask size is the subimage size. According to the

bit decomposition concept, each subimages can be expressed in terms of their bit planes as:

$$\mathbf{I}_{x,y}(i, j) = 2^{N-1} \mathbf{I}_{x,y}^{(N-1)}(i, j) + 2^{N-2} \mathbf{I}_{x,y}^{(N-2)}(i, j) + \dots + 2^0 \mathbf{I}_{x,y}^{(0)}(i, j) \quad (4.13)$$

where, $\mathbf{I}_{x,y}^{(N-m)}(i, j)$, ($m = 1 \dots N$) is the (N-m)th bit plane of (x, y)th subimage.

$$\mathbf{I}_{x,y}(i, j) = \sum_{k=1}^N 2^{N-k} \mathbf{I}_{x,y}^{(N-k)}(i, j) \quad (4.14)$$

Let's define

$$\bar{\mathbf{a}}^{(x,y)} \triangleq \begin{bmatrix} a_{N-1}^{(x,y)} & a_{N-2}^{(x,y)} & \dots & a_0^{(x,y)} \end{bmatrix} \in [0,1] \quad (4.15)$$

The entries of $\bar{\mathbf{a}}^{(p,r)}$ is determined by the DCT magnitudes of the (p,r)th subimage. This vector tells us which planes are taken into account for the reconstruction. Then $\mathbf{I}(i, j)$ (taking $M=N$) can be expressed as:

$$\mathbf{I}(i, j) = \begin{bmatrix} \sum_{k=1}^M \overbrace{a_{N-k}^{(1,1)} 2^{N-k} \mathbf{I}_{1,1}^{N-k}(i, j)}^{(1,1)\text{th subimage}} & \sum_{k=1}^M \overbrace{a_{N-k}^{(1,2)} 2^{N-k} \mathbf{I}_{1,2}^{N-k}(i, j)}^{(1,2)\text{th subimage}} & \dots & \sum_{k=1}^M \overbrace{a_{N-k}^{(1,r)} 2^{N-k} \mathbf{I}_{1,r}^{N-k}(i, j)}^{(1,r)\text{th subimage}} \\ \vdots & \ddots & & \vdots \\ \sum_{k=1}^M \overbrace{a_{N-k}^{(p,1)} 2^{N-k} \mathbf{I}_{p,1}^{N-k}(i, j)} & \dots & & \sum_{k=1}^M \overbrace{a_{N-k}^{(p,r)} 2^{N-k} \mathbf{I}_{p,r}^{N-k}(i, j)}^{(p,r)\text{th subimage}} \end{bmatrix} \quad (4.16)$$

$$= \sum_{k=1}^M 2^{N-k} \begin{bmatrix} a_{N-k}^{(1,1)} \mathbf{I}_{1,1}^{N-k}(i, j) & a_{N-k}^{(1,2)} \mathbf{I}_{1,2}^{N-k}(i, j) & \dots & a_{N-k}^{(1,r)} \mathbf{I}_{1,r}^{N-k}(i, j) \\ \vdots & \ddots & & \vdots \\ a_{N-k}^{(p,1)} \mathbf{I}_{p,1}^{N-k}(i, j) & \dots & & a_{N-k}^{(p,r)} \mathbf{I}_{p,r}^{N-k}(i, j) \end{bmatrix} \quad (4.17)$$

Define

$$\bar{\mathbf{I}}_{x,y} \triangleq \begin{bmatrix} 2^{N-1} \mathbf{I}_{x,y}^{(N-1)} & 2^{N-2} \mathbf{I}_{x,y}^{(N-2)} & \dots & 2^0 \mathbf{I}_{x,y}^{(0)} \end{bmatrix}^T \quad (4.18)$$

$\Rightarrow$ (p,r) th subimage, $\mathbf{I}_{p,r}(i, j)$, can be written as: $\bar{\mathbf{I}}_{p,r} * \bar{\mathbf{a}}^{(p,r)}$, where (*) is the inner product operation. So, the image can be formed as:

$$\mathbf{I}(i, j) = \begin{bmatrix} \bar{\mathbf{I}}_{1,1} * \bar{\mathbf{a}}^{(1,1)} & \bar{\mathbf{I}}_{1,2} * \bar{\mathbf{a}}^{(1,2)} & \dots & \bar{\mathbf{I}}_{1,r} * \bar{\mathbf{a}}^{(1,r)} \\ \vdots & \ddots & & \vdots \\ \bar{\mathbf{I}}_{p,1} * \bar{\mathbf{a}}^{(p,1)} & \dots & & \bar{\mathbf{I}}_{p,r} * \bar{\mathbf{a}}^{(p,r)} \end{bmatrix} \quad (4.19)$$

Hence, compression and progressive transmission approaches can be expressed as the following.

Compression:

Compression ratio is determined by $\bar{\mathbf{a}}^{(x,y)} \in [0,1]$ vector. This ratio is $\% \frac{\varphi}{N}$ where φ is the number of zero entries of $\bar{\mathbf{a}}^{(x,y)}$. For example let $\bar{\mathbf{a}}^{(1,1)} = [1 \ 1 \ 0 \ 0]^T$, and $N=4$, (i.e. we have 4 planes, representing the image into 16 gray levels) the compression ratio is $\% \frac{2}{4} \cdot 100 = \%50$. In this case 3th and 2nd planes are taken into account for the decomposition of (1,1)th subimage, 1st and 0th planes are ignored, resulting some contrast leakage.

Progressive Transmission

The reconstruction of the image at the receiver is in the progressive fashion. The contrast becomes better as the iteration increases.

- **First iteration: (M=1)**

M is taken 1 in equation (4.17). So;

$$\Rightarrow \mathbf{I}(i, j) = 2^{N-1} \begin{bmatrix} a_{N-1}^{(1,1)} \mathbf{I}_{1,1}^{N-1}(i, j) & a_{N-1}^{(1,2)} \mathbf{I}_{1,2}^{N-1}(i, j) & \dots & a_{N-1}^{(1,r)} \mathbf{I}_{1,r}^{N-1}(i, j) \\ \vdots & \ddots & & \vdots \\ a_{N-1}^{(p,1)} \mathbf{I}_{p,1}^{N-1}(i, j) & \dots & & a_{N-1}^{(p,r)} \mathbf{I}_{p,r}^{N-1}(i, j) \end{bmatrix} \quad (4.20)$$

$$\triangleq \mathbf{I}(i, j) \Big|_{M=1}$$

This relates to binary image (2 gray level) in the progressively transmission system. i.e. at the first iteration we have 2 level image.

- **Second iteration: (M=2)**

$$\Rightarrow \mathbf{I}(i, j) = \mathbf{I}(i, j)|_{M=1} + 2^{N-2} \begin{bmatrix} a_{N-2}^{(1,1)} \mathbf{I}_{1,1}^{N-2}(i, j) & a_{N-2}^{(1,2)} \mathbf{I}_{1,2}^{N-2}(i, j) & \dots & a_{N-2}^{(1,r)} \mathbf{I}_{1,r}^{N-2}(i, j) \\ \vdots & \ddots & & \vdots \\ a_{N-1}^{(p,1)} \mathbf{I}_{p,1}^{N-2}(i, j) & \dots & \dots & a_{N-1}^{(p,r)} \mathbf{I}_{p,r}^{N-2}(i, j) \end{bmatrix} \quad (4.21)$$

$$\triangleq \mathbf{I}(i, j)|_{M=2}$$

At the second iteration we have 4 gray level image and hence more contrast, compared to that of first iteration.

- **Nth iteration: (M=N)**

$$\Rightarrow \mathbf{I}(i, j) = \mathbf{I}(i, j)|_{M=N-1} + 2^0 \begin{bmatrix} a_0^{(1,1)} \mathbf{I}_{1,1}^0(i, j) & a_0^{(1,2)} \mathbf{I}_{1,2}^0(i, j) & \dots & a_0^{(1,r)} \mathbf{I}_{1,r}^0(i, j) \\ \vdots & \ddots & & \vdots \\ a_0^{(p,1)} \mathbf{I}_{p,1}^0(i, j) & \dots & \dots & a_0^{(p,r)} \mathbf{I}_{p,r}^0(i, j) \end{bmatrix} \quad (4.22)$$

This relates to 2^N gray level image (Maximum contrast).

4.4.4.1. Example of a Special Case

Let $N=4$ and the image partitioned into four subimages. In this case $\mathbf{I}(i, j)$ image can be expressed as

$$\mathbf{I}(i, j) = \begin{bmatrix} \mathbf{I}_{11}(i, j) & \mathbf{I}_{12}(i, j) \\ \mathbf{I}_{21}(i, j) & \mathbf{I}_{22}(i, j) \end{bmatrix} \quad (4.23)$$

and every subimage can be written as:

$$\mathbf{I}_{x,y}(i, j) = 2^3 \mathbf{I}_{x,y}^{(3)}(i, j) + 2^2 \mathbf{I}_{x,y}^{(2)}(i, j) + 2^1 \mathbf{I}_{x,y}^{(1)}(i, j) + 2^0 \mathbf{I}_{x,y}^{(0)}(i, j) \quad (4.24)$$

According to the DCT coefficients of each subimage, let (1,1) subimage, $\mathbf{I}_{11}(i, j)$, is represented by 2 planes. Let subimage (1,2), $\mathbf{I}_{12}(i, j)$, is by 1 plane. Subimage (2,1), $\mathbf{I}_{21}(i, j)$, is by 3 planes and last subimage (2,2), $\mathbf{I}_{22}(i, j)$, is represented by 4 planes.

Compression

The entries of $\bar{\mathbf{a}}^{(p,r)}$ vector becomes;

$$\begin{aligned} \bar{\mathbf{a}}^{(1,1)} &\triangleq [1 \ 1 \ 0 \ 0], & \bar{\mathbf{a}}^{(1,2)} &\triangleq [1 \ 0 \ 0 \ 0], & \bar{\mathbf{a}}^{(2,1)} &\triangleq [1 \ 1 \ 1 \ 0], \\ \bar{\mathbf{a}}^{(2,2)} &\triangleq [1 \ 1 \ 1 \ 1] \end{aligned} \quad (4.25)$$

For this case, using $\% \frac{\Phi}{N}$ compression formula, we have %50 compression for (1,1) subimage, %75 compression for (1,2) subimage, %25 compression for (2,1) subimage and no compression for (1,1) subimage. As a result, %62 compression ratio is achieved. If we express the subimages with respect to the bit planes,

$$\mathbf{I}_{x,y}(i,j) = \bar{\mathbf{I}}_{x,y} * \bar{\mathbf{a}}^{(x,y)} \quad (4.26)$$

$$\text{where, } \bar{\mathbf{a}}^{(x,y)} = \begin{bmatrix} a_3^{(x,y)} & a_2^{(x,y)} & a_1^{(x,y)} & a_0^{(x,y)} \end{bmatrix}^T$$

and

$$\bar{\mathbf{I}}_{x,y} = \begin{bmatrix} 2^{4-1} \mathbf{I}_{x,y}^{(4-1)} & 2^{4-2} \mathbf{I}_{x,y}^{(4-2)} & 2^{4-3} \mathbf{I}_{x,y}^{(4-3)} & 2^{4-4} \mathbf{I}_{x,y}^{(4-4)} \end{bmatrix} = \begin{bmatrix} 8\mathbf{I}_{x,y}^{(3)} & 4\mathbf{I}_{x,y}^{(2)} & 2\mathbf{I}_{x,y}^{(1)} & \mathbf{I}_{x,y}^{(0)} \end{bmatrix} \quad (4.27)$$

for the (1,1) subimage

$$\begin{aligned} \mathbf{I}_{1,1}(i,j) &= \bar{\mathbf{I}}_{1,1} * \bar{\mathbf{a}}^{(1,1)} = \begin{bmatrix} 8\mathbf{I}_{1,1}^{(3)} & 4\mathbf{I}_{1,1}^{(2)} & 2\mathbf{I}_{1,1}^{(1)} & \mathbf{I}_{1,1}^{(0)} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ &= 8\mathbf{I}_{1,1}^{(3)} + 4\mathbf{I}_{1,1}^{(2)} \end{aligned} \quad (4.28)$$

meaning only 3rd and 2nd planes are contributed for the recovery to this subimage.

for the (1,2) subimage

$$\begin{aligned} \mathbf{I}_{1,2}(i,j) &= \bar{\mathbf{I}}_{1,2} * \bar{\mathbf{a}}^{(1,2)} = \begin{bmatrix} 8\mathbf{I}_{1,2}^{(3)} & 4\mathbf{I}_{1,2}^{(2)} & 2\mathbf{I}_{1,2}^{(1)} & \mathbf{I}_{1,2}^{(0)} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ &= 8\mathbf{I}_{1,2}^{(3)}. \end{aligned} \quad (4.29)$$

meaning only 3rd is contributed for the recovery to this subimage.

for the (2,1) subimage

$$\begin{aligned} \mathbf{I}_{2,1}(i, j) &= \bar{\mathbf{I}}_{2,1} * \bar{\mathbf{a}}^{(2,1)} = \begin{bmatrix} 8\mathbf{I}_{2,1}^{(3)} & 4\mathbf{I}_{2,1}^{(2)} & 2\mathbf{I}_{2,1}^{(1)} & \mathbf{I}_{2,1}^{(0)} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ &= 8\mathbf{I}_{2,1}^{(3)} + 4\mathbf{I}_{2,1}^{(2)} + 2\mathbf{I}_{2,1}^{(1)} \end{aligned} \quad (4.30)$$

meaning only 3rd, 2nd and 1st planes are contributed for the recovery to this subimage.

for the (2,2) subimage

$$\begin{aligned} \mathbf{I}_{2,2}(i, j) &= \bar{\mathbf{I}}_{2,2} * \bar{\mathbf{a}}^{(2,2)} = \begin{bmatrix} 8\mathbf{I}_{2,2}^{(3)} & 4\mathbf{I}_{2,2}^{(2)} & 2\mathbf{I}_{2,2}^{(1)} & \mathbf{I}_{2,2}^{(0)} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\ &= 8\mathbf{I}_{2,2}^{(3)} + 4\mathbf{I}_{2,2}^{(2)} + 2\mathbf{I}_{2,2}^{(1)} + \mathbf{I}_{2,2}^{(0)} \end{aligned} \quad (4.31)$$

meaning all planes are contributed for the recovery to this subimage.

Progressive Transmission

- **First iteration: (M=1)**

$$\mathbf{I}(i, j) = 2^3 \begin{bmatrix} \mathbf{a}_3^{(1,1)} \mathbf{I}_{11}^{(3)}(i, j) & \mathbf{a}_3^{(1,2)} \mathbf{I}_{12}^{(3)}(i, j) \\ \mathbf{a}_3^{(2,1)} \mathbf{I}_{21}^{(3)}(i, j) & \mathbf{a}_3^{(2,2)} \mathbf{I}_{22}^{(3)}(i, j) \end{bmatrix} = 8 \begin{bmatrix} 1 * \mathbf{I}_{11}^{(3)}(i, j) & 1 * \mathbf{I}_{12}^{(3)}(i, j) \\ 1 * \mathbf{I}_{21}^{(3)}(i, j) & 1 * \mathbf{I}_{22}^{(3)}(i, j) \end{bmatrix} \triangleq \mathbf{I}(i, j)|_{M=1} \quad (4.32)$$

Each masks position over the image, after the first iteration, is shown the in figure 4.18.

$\mathbf{I}_{1,1}^3(i, j)$	$\mathbf{I}_{1,2}^3(i, j)$
$\mathbf{I}_{2,1}^3(i, j)$	$\mathbf{I}_{2,2}^3(i, j)$

Figure 4.18: Masks position after 1st iteration

- **Second iteration: (M=2)**

$$\begin{aligned} \mathbf{I}(i, j) &= \mathbf{I}(i, j)|_{M=1} + 2^2 \begin{bmatrix} a_2^{(1,1)} \mathbf{I}_{11}^{(2)}(i, j) & a_2^{(1,2)} \mathbf{I}_{12}^{(2)}(i, j) \\ a_2^{(2,1)} \mathbf{I}_{21}^{(2)}(i, j) & a_2^{(2,2)} \mathbf{I}_{22}^{(2)}(i, j) \end{bmatrix} = 4 \begin{bmatrix} 1 * \mathbf{I}_{11}^{(2)}(i, j) & 0 * \mathbf{I}_{12}^{(2)}(i, j) \\ 1 * \mathbf{I}_{21}^{(2)}(i, j) & 1 * \mathbf{I}_{22}^{(2)}(i, j) \end{bmatrix} \\ &= 4 \begin{bmatrix} \mathbf{I}_{11}^{(2)}(i, j) & 0 \\ \mathbf{I}_{21}^{(2)}(i, j) & \mathbf{I}_{22}^{(2)}(i, j) \end{bmatrix} \triangleq \mathbf{I}(i, j)|_{M=2} \end{aligned} \quad (4.33)$$

Each masks position over the image, after the second iteration, is shown the in figure 4.19.

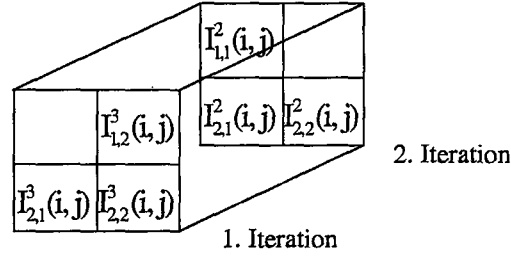


Figure 4.19: Masks position after 2nd iteration

- **Third iteration: (M=3)**

$$\mathbf{I}(i, j) = \mathbf{I}(i, j)|_{M=2} + 2^1 \begin{bmatrix} a_1^{(1,1)} \mathbf{I}_{11}^{(1)}(i, j) & a_1^{(1,2)} \mathbf{I}_{12}^{(1)}(i, j) \\ a_1^{(2,1)} \mathbf{I}_{21}^{(1)}(i, j) & a_1^{(2,2)} \mathbf{I}_{22}^{(1)}(i, j) \end{bmatrix} = \mathbf{I}(i, j)|_{M=2} + 2 \begin{bmatrix} 0 & 0 \\ \mathbf{I}_{21}^{(1)}(i, j) & \mathbf{I}_{22}^{(1)}(i, j) \end{bmatrix} \triangleq \mathbf{I}(i, j)|_{M=3}$$

Each masks position over the image, after the third iteration, is shown the in figure 4.20.

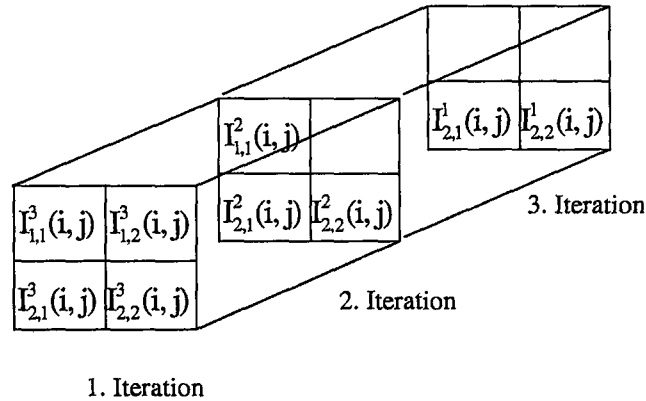


Figure 4.20: Masks position after 3rd iteration

(4.34)

- **Fourth iteration: (M=4)**

$$\mathbf{I}(i, j) = \mathbf{I}(i, j)|_{M=3} + 2^0 \begin{bmatrix} a_0^{(1,1)} \mathbf{I}_{11}^{(0)}(i, j) & a_0^{(1,2)} \mathbf{I}_{12}^{(0)}(i, j) \\ a_0^{(2,1)} \mathbf{I}_{21}^{(0)}(i, j) & a_0^{(2,2)} \mathbf{I}_{22}^{(0)}(i, j) \end{bmatrix} = \mathbf{I}(i, j)|_{M=3} + 2 \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_{22}^{(0)}(i, j) \end{bmatrix} \quad (4.35)$$

Each masks position over the image, after the fourth iteration, is shown the in figure 4.21.

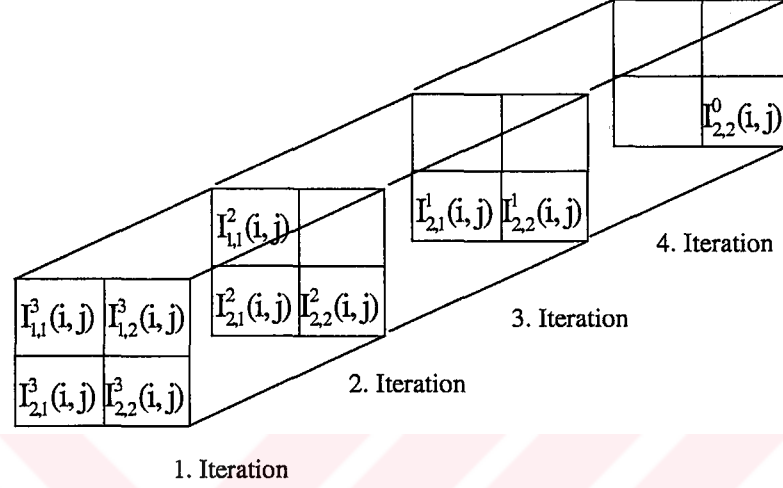


Figure 4.21: Masks position after 4th iteration

is obtained. As the iteration increases, from 1st to 4th, image enhances progressively achieving %62 compression ratio.

4.4.5. Simulation Results

The mask size is taken as 25x25 and 10x10 for progressive transmission for 3 different images. The results are shown from figures D.1 to D.5. The right columns are the histograms of the bit plane allocation scheme which is reserved for each mask. In Figure D.1 (a), the numbers of allocated planes are written on each mask for good understanding. For example, the (1,1) mask is represented with 1 plane, whereas (6,7) mask is represented with 4 plane (they are circled in the figure). These are determined according to their detail levels. The plane allocation is achieved automatically. From Figure D.1 (b), we can see that, totally 50x1 planes, 14x2 planes, 9x3 planes and 27x4 planes are transmitted for the reconstruction. So the compression ratio is $(10 \times 10 \times 4 - 50 \times 1 - 14 \times 2 - 9 \times 3 - 27 \times 4) / (10 \times 10 \times 4) = \%46$. This is better (even for the same compression ratio) than that of the non adaptive allocation case. Because, most of the planes are used for more detailed areas, whereas the planes are shared by each masks equally for non-adaptive case. This causes to lost details and does not highlight the fine changes. If we want to get some greater details (accepting

to use more planes and hence less compression ratio), we can chose the allocation procedure which has the histograms expressed in (d) and (f). The compression ratio is $(10 \times 10 \times 4 - 21 \times 1 - 17 \times 2 - 16 \times 3 - 46 \times 4) / 10 \times 10 \times 4 = \%28$ for this case. Also for (f), we have $(10 \times 10 \times 4 - 21 \times 1 - 17 \times 2 - 16 \times 3 - x \times 4) / 10 \times 10 \times 4 = \%7.5$ compression ratio.

Figure D.2 shows the image which is progressively transmitted and reconstructed for which mask size is 10×10 . When the changes occur in small areas, if the mask size is chosen great, it is not possible to allocate the planes efficiently (i.e. especially when the area of the changes are comparably small to that of mask size). The smaller mask size is taken, the evaluations of details in the small areas can be taken into account. For these cases, we have $(25 \times 25 \times 4 - 374 \times 1 - 64 \times 2 - 61 \times 3 - 126 \times 4) / 25 \times 25 \times 4 = \%52$ (for b), $(25 \times 25 \times 4 - 110 \times 1 - 126 \times 2 - 138 \times 3 - 251 \times 4) / 25 \times 25 \times 4 = \%29$ (for (d)), $(25 \times 25 \times 4 - 70 \times 1 - 7 \times 2 - 9 \times 3 - 539 \times 4) / 25 \times 25 \times 4 = \%9$ (for (f)).

4.4.6. Evaluations

In this section, the image is segmented into regions and allocate BPSs according to their DCT coefficients and they are analyzed and exploited separately. The planes of the segmented regions is then coded using turbo codes. The size of the encoded regions can be gradually reduced to allow progressively a more refined description of the transmitted image. In simulation, the Subimage size selection is taken as $M=25$ and 10, the results of these cases are shown in Figures D.1 and D.6. This progressive approach could be useful for slower communication channels and allows a user to reconstruct, view, and optionally reject an image *as it arrives*, rather than waiting for the whole image, compressed or not, to arrive. Nevertheless, in the case of non-progressive compression, neither an approximation nor the original picture is available until the whole image data is retrieved.

5. CONCLUSIONS AND DISCUSSIONS

Recent advances in image transmission and compression techniques result in very efficient and resourceful representations of images in noiseless environment. When transmitted in noisy channels unprotected, those bit streams are easily corrupted and both the objective and subjective reconstructed image qualities are unacceptable in most cases. In fact, detailed analysis of the simulations shows that a single bit of error may destroy the whole bit stream due to error propagation. Thus, some form of channel protections are necessary for robust image communications.

This thesis investigates the efficient 2D image transmission over the communication channel achieving the objectives of “error correction”, “progressive transmission” and “image oriented lossless/lossy compression” under the Turbo Coding error protection scheme. We begun the study with the investigation of the transmission of the 2-D images via turbo codes. Then we continued to develop some methods that reduces the bandwidth requirements (i.e. progressive transmission). And last, we searched the image oriented compression.

At first, two new methods AW-TS (Adaptive Wiener-Turbo System) and ICIPA-TS (Iterative Cellular Image Processing Algorithm-Turbo System) are proposed. In the literature the transmission is usually done without taking into account any characteristics of the source data. This fact motivated the research. Before transmission, the 2-D image is partitioned into N Bit Planes and then Each of them is coded by Turbo encoder and transmitted. At the receiver side, before decoding, bit-planes are re-assembled taking into consideration of neighborhood relationship. The noisy bit-planes of the image are evaluated iteratively by a combined equalization block, which is composed of 2-D adaptive noise-removal filtering and Turbo decoder. Hence the advantages of the superior performance of Turbo codes with image processing algorithms are combined resulting to get a better bit error performance. So the resulting BER is at least 2dB better than the conventional error correction scheme. By simply choosing any number of bit slices, data rates can also be increased up to $(N-1)$ times resulting less memory requirement. This coding strategy became very suitable for the compression in our work. The results show that below 2 dB SNR's, the conventional error correction (Turbo coding) algorithm is not satisfying. The proposed algorithm also gives good results at 0 dB or at lower SNR's.

This is a challenging result and we have 2dB additional SNR improvement. Hence we obtained satisfactory results of both bit error rate (BER) and image enhancement performance below 2 dB SNR, compared to separately application of traditional turbo coding scheme and 2-D filtering. For simplicity of simulation, we consider only 4 bits/pixel images; So N is taken as 4. however, all the following assertions can be easily adapted to 8 or more bits/pixel images.

Second, we handled Compression (Bit-sliced and JPEG) and Progressive transmission. The advantages of Bit-sliced and DCT based compressions are discussed. The progressive transmission technique sends the important data up front, so we can see the important properties of the whole image even when only a small part of the data is transmitted. This approach is very suitable where, frequent incremental visual inspections of images are necessary. The approximations of the image are updated progressively. The speed of this operation depends on the amount of data that should be retrieved from the storage for sketch reconstruction. The power of this method comes from the ability to control the amount of data to be transferred. In some cases, the first iteration can prove sufficient for a given application, considering constraints of bandwidth and response time. Notably, we can compare this to sending an image by fax. In many cases, the low quality of the received image can be satisfactory. The advantage here is that if it is not, the receiver can ask for more image data and combine it with what it has received previously. The crucial point is that the receiver assembles data received in a previous iterations with a data from the most recent one to produce a higher quality image.

Third, Image-oriented compression is accomplished. we segmented the image into regions and allocate the number of BPS according to their DCT coefficients. When DCT magnitudes of subimages are high, more planes are allocated, meaning this region has a greater detail in. An image with lower details have less number of planes and could be more efficiently compressed. This method allows a lossy image transfer with fast extraction of an image, while the bandwidth of communication channel is preserved. Hence, we can conclude that the new approaches will be very compromising in 2-D image transmission, recovery of noisy signals and image compression. The results of all implementations are given at the ends of the relating chapters.

The progressive transmission of color images with 256 colors is currently unresolved by this method. The issue of color map adjustment as transmission progresses is still an open problem. Due to the fact that color map entrees can contain random values from the color three-space, the best way to perform color quantization as image bit

planes are added to the final image is unclear. We will try to address this problem in our future work. A possible solution would be application of wavelet compression instead of image bit plane partitioning.



REFERENCES

- Bahceci I. and Tolga M. Duman**, (2001). Combined Turbo Coding and unitary space-time modulation, *Proc. of IEEE ISIT'2001*, pp. 106-120, Washington DC.
- Baro S., Bauch G., and Hansmann A.**, (2000). Improved codes for space-time trellis coded modulation, *IEEE Commun. Lett.*, vol. 46, no. 2, pp. 20-22.
- Benedetto S. and Montorsi G.**, (1996). Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes, *IEEE Trans. on Inform. Theory*, vol. 42, no. 2, pp.409-428.
- Benedetto S., Divsalar D., Montorsi G., and Pollara F.**, (1996a). A soft-input soft-output maximum a posteriori (map) module to decode parallel and serial concatenated codes. *TDA Progress Report 42-127*, pp. 1-20.
- Benedetto S., Divsalar D., Montorsi G., and Pollara F.**, (1996b). Parallel concatenated trellis coded modulation". *Proc. IEEE International Conference on Communications, ICC'96*, pp. 974-978, Dallas.
- Benedetto S., Montorsi G., Divsalar D., and Pollara F.**, (1996c). Soft-output decoding algorithms in iterative decoding of turbo codes, *TDA Progress Report 42-124*, pp. 63-83.
- Benedetto S., Divsalar D., Montorsi G., and Pollara F.**, (1998a). Analysis, design, and iterative decoding of double serially concatenated codes with interleavers, *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 231-244.
- Benedetto S., Divsalar D., Montorsi G., and Pollara F.**, (1998b). Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding, *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 909-926.
- Berrou C., Glavieux A. and Thitimajshima P.**, (1993). Near Shannon limit error correcting coding and decoding, *Proc. Intl. Conf. Comm.*, pp. 1064-1070.
- Boutros J.J., Calvanese E. and Guillen A., Fabregas I.**, (2004). Turbo code Design for block fading channels, *Allerton's Conference on Communication and Control, Illinois, October 2004*.

- Buyukatak K., Gose E., Ucan O.N., Kent S., Osman O., (2003).** Channel Equalization and Noise Reduction Based Turbo Codes“, *Recent Advances on Space Technology*.
- Buyukatak K, Ucan O. N, Kargin S, Kent S, Gose E., (2004).** Uyarlamalı wiener-turbo (UW-TS) sistemi ve bu sistemin iki boyutlu işaretlerin iletiminde kullanılması, *Havacılıkta İleri Teknolojiler ve Uygulamaları Sempozyumu (HITEK)*.
- Clarkson K. L., Sweldens W., and Zheng A., (2001).** Fast multiple antenna differential decoding, *IEEE Trans. Commun.*, vol. 46, no. 2, pp. 253-261.
- Cover T. M. and Thomas J. A., (1991).** Elements of information theory, *Wiley, NewYork*.
- Cui D. and Haimovich A. M., (2001a).** Performance of parallel concatenated space-time codes, *IEEE Commun. Lett.*, vol. 5, no. 6, pp. 236-238.
- Cui D. and Haimovich A. M., (2001b).** Error performance analysis of turbo space-time coded modulation over fading channels, *Proc. IEEE ICC'01*, Helsinki, Finland.
- Damen O., Chkeif A., and Belfiore J. C., (2000).** Lattice code decoder for space-time codes, *IEEE Commun. Letter*, vol. 4, no. 5, pp. 161-163,.
- Divsalar D. and Pollara F., (1995a).** Turbo Codes for deep space communications, *TDA Progress Report 42-120*, pp. 29-39.
- Divsalar D. and Pollara F., (1995b).** Multiple Turbo Codes for deep-space communications, *JPL, TDA Progress Report 42-121*.
- Divsalar D. and Polara F., (1995c).** Multiple Turbo Codes, *MILCOM'95*, pp. 279-285.
- Divsalar D. and Ellise Mc. R. J., (1998).** On the design of concatenated coding systems with interleavers, *TMO Progress Report 42-134*, pp. 1-22.
- Dolinar S. and Divsalar D., (1995).** Weight distributions for Turbo Codes using random and nonrandom permutations, *TDA Progress Report 42-122*, pp. 56-65.
- Duman T. M. and Salehi M., (1999).** Performance bounds for turbo coded modulation systems, *IEEE Trans. Commun.* vol. 47, no. 4, pp. 511-521.
- Forney Jr G. D., (1966).** Concatenated Codes, *M.I.T Press*, Cambridge, MA.

- Foschini G. J. and Gans M. J.,** (1998). On limits of wireless communications in a fading environment when using multiple antennas, *Wireless Personal Commun.*, pp. 311-335.
- Fossorier M.P.C., Burkert F., Lin S. and Hagenauer J.,** (1998). On the equivalence between SOVA and max-log-MAP decodings, *IEEE Commun. Letters*, vol. 2, pp. 137-139.
- Gonzales R.C., Woods R.E.,** (1992). Digital Image Processing, *ISBN 0-201-50803-6*.
- Gose E., Buyukatak K., Osman O., Ucan O.N., Pastaci H.,** (2003). Performance Of Turbo Decoding For Time-Varying Multipath Channels, *Recent Advances on Space Technology*.
- Gose E., Pastaci H., Buyukatak K., Ucan O.N. and Osman O.,** (2004). Performance of Transmit Diversity-Turbo Trellis Coded Modulation (TD-TTCM) over Genetically estimated WSSUS Mimo Channels, *Frequenz* 58 (2004)11-12, pp. 249-255.
- Gross W.J. and Gulak P.G.,** (1998a). Simplified MAP Algorithm suitable for implementation of Turbo Decoders, *Electronic Letters* online no:19981126.
- Hageneauer J.,** (1996). Iterative decoding of binary block and convolutional codes, *IEEE Trans. Inform. Theory*, vol. 42, pp. 429-445.
- Hammons A. R. and Gamal H. E.,** (2000). On the theory of space-time codes for PSK modulation, *IEEE Trans. Inform. Theory*, vol. 46, no. 2, pp.524-542.
- Hillery A. D. and Chin R. T.,** (1991). Iterative Wiener filters for image restoration, *IEEE Trans. Signal Processing*, vol. 39, no. 8, 1892-1899,
- Hirasawa S., Kasahara M., Sugiyama Y., and Namekawa T.,** (1980). Certain generalizations of concatenated codes-exponential error bound and decoding complexity, *IEEE Trans. Inform. Theory*, pp. 527-534.
- Hirasawa S., Kasahara M., Sugiyama Y., and Namekawa T.,** (1984). Modified product codes, *IEEE Trans. Inform. Theory*, pp. 299-306.
- Hochwald B. M. and Marzetta T. L.,** (2000). Unitary space-time modulation for multiple-antenna communications in Rayleigh flat fading, *IEEE Trans. Inform. Theory*, vol. 46, no. 2, pp. 543-564.

- Hochwald B. M. and Sweldens W.**, (2000). Differential unitary space-time modulation, *IEEE Trans. Commun.*, vol. 48, no. 12, pp. 2041-2052.
- Hochwald B. M., Marzetta T. L., Richardson T. J., W. Sweldens, and R. Urbanke**, (2000). Systematic design of unitary space-time constellations, *IEEE Trans. Inform. Theory*, vol. 46, no. 6, pp. 1962-1979.
- Hughes B. L.**, (2000a). Differential space-time modulation, *IEEE Trans. Inform. Theory*, vol. 46, no. 7, pp. 2567-2578.
- Hughes B. L.**, (2000b). Further results on differential space-time modulation. In Sensor Array and Multichannel Signal Processing Workshop, *Proceedings of the 2000 IEEE*, pp. 163-167.
- Le Goff S., Glavieux A., and Berrou C.**, (1994). Turbo codes and high spectral efficiency modulation". Proc. IEEE International Conference on Communications Supercomm, *ICC'94*, pp. 645-649, New Orleans.
- Liew T.H., Pliquet J., Yeap B. L., Yang L. L., and Hanzo L.**, (2000). Comparative study of space-time block codes and various concatenated turbo coding schemes, *Proc. of IEEE PIMRC'00*, pp. 741-745, London, UK.
- Lin X. and Blum R. S.**, (2000). Improved space-time codes using serial concatenation, *IEEE Commun. Lett.*, vol. 4, no. 7, pp. 221-223.
- Lin X. and Blum R. S.**, (2001). Guidelines for serially concatenated space-time code design in flat rayleigh fading channels, *Third IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications*, pp. 27-250.
- Marzetta T. L. and Hochwald B. M.**, (1999). Capacity of a mobile multiple antenna communication link in rayleigh flat fading. *IEEE Trans. Inform. Theory*, vol. 45, no. 1, pp. 139-157.
- Sherwood P. G. and Zeger K.**, (1998). Error protection for progressive image transmission over memoryless and fading channels. *IEEE Trans. Communications*, Vol. 46, No. 12, pp. 1555-1559. December.
- Naguib A. F., Tarokh V., Seshadri N., and Calderbank A.R.**, (1998). A space-time coding modem for high data rate wireless communication, *IEEE Trans. Commun.*, vol. 16, no. 8, pp. 1459-1475.
- Naguib A. F., Seshadri N., and Calderbank A. R.**, (2000). Increasing data rate over wireless channels, *IEEE Signal Processing Magazine*, pp. 76-92.

- Narayanan K. and Stuber G. L.,** (1999). A serial concatenation approach to iterative demodulation and decoding, *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 956-961.
- Osman O., Uçan O.N., Albora M.,** (2003). Iterative Cellular Image Processing Algorithm (ICIPA), *Istanbul University –Journal of Electrical and Electronics Engineering (IU-JEE)*, Vol.3, No.1, pp.775-782.
- Perez L., Seghers J., and Costello D.,** (1996). A distance spectrum interpretation of Turbo Codes, *IEEE Trans. on Information Theory*, vol. 42, no. 6, pp. 1698–1709.
- Robertson P.,** (1994). Illuminating the structure of code and decoder of parallel concatenated recursive systematic Turbo Codes, *Proc. IEEE Global Communications Conference*, pp. 1298-1303.
- Robertson P. and Woerz T.,** (1995). Coded modulation scheme employing turbo codes, *Electronic Letters*, vol. 31, no. 18, pp. 1546-1547.
- Robertson P. and Woerz T.,** (1996). A novel bandwidth efficient coding scheme employing Turbo Codes, *Proc. ICC'96*, pp. 962-967.
- Robertson P. and Woerz T.,** (1998). Bandwidth-efficient Turbo Trellis-coded modulation using punctured component codes, *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 206-218.
- Robertson P., Hoener P. and Villebrun E.,** (1977). Optimal and sub-optimal maximum a posteriori algorithms suitable for Turbo Decodings, *European Trans. on Telecommun.*, vol. 8, pp. 119-125.
- Rauschenbach U.,** Progressive Image Transmission Using Levels of Detail and Regions of Interest, *University of Rostock*, Computer Science Department.
- Rekh S., Rani S., Shanmugam A.,** (2005). Optimal Choice of interleaver for Turbo Codes, *Academic Open Internet Journal*, Volume 15 (<http://www.acadjournal.com/2005/v15/part6/p7/>).
- Sason I. and Shamai S. (Shits),** (2000). Improved upper bounds on the ml decoding error probability of parallel and serial concatenated turbo codes via their ensemble distance spectrum, *IEEE Trans. Inform. Theory*, vol. 46, no. 1, pp. 24-47, Jan.
- Sklar B.,** (1988). Digital Communications: Fundamentals and applications, Chapter 6, *Prentice-Hall International, Inc.*
- Slim J.,** Two-dimensional Signal and Image Processing, ISBN 0-13-934563-9, page: 525-540

- Takeshita Y. and Costello J.,** (2000). New Deterministic interleaver designs for Turbo Codes, *IEEE Trans. On Information Theory*, vol. 46, no.6, pp.1988-2006.
- Tarokh V., Seshadri N., and Calderbank A.r.,** (1998a). Space-time codes for high data rate wireless communications: Performance analysis and code construction, *IEEE Tran. Inform. Theory*, vol. 44, no. 2, pp. 744-765.
- Tarokh V., Alamouti S. M., and Poon P.,** (1998b). New detection scheme for transmit diversity with no channel estimation, *ICUPC '98*, volume 2, pp. 917-920.
- Tarokh V., Jafarkhani H., and Calderbank A. R.,** (1999a). Space-time block codes from orthogonal designs, *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1456-1467.
- Tarokh V., Naguib A. F., Seshadri N., and Calderbank A.R.,** (1999b). Space-time codes for high data rate wireless communication: Performance criteria in the presence of channel estimation errors, mobility, and multiple paths, *IEEE J. Select Areas Commun.*, vol. 47, no. 2, pp. 199-207.
- Tarokh V., Jafarkhani H., and Calderbank A. R.,** (1999c). Space-time block coding for wireless communications: Performance results, *IEEE J. Select Areas Commun.*, vol. 17, no. 3, pp. 451-460.
- Tarokh V., Jafarkhani H., and Calderbank A. R.,** (1999d). Space-time block codes from orthogonal designs, *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1456-1467.
- Tarokh V. and Jafarkhani H.,** (2000). A differential detection scheme for transmit diversity, *IEEE J. Select Areas Commun.*, vol. 18, no. 7, pp. 1169-1174.
- Telatar I.E.,** (1995). Capacity of multi-antenna Gaussian channels, *AT&T-Bell Labs Internal Tech. Memo*.
- Tujkovic D.,** Space-time Turbo Coded modulation. submitted to *EURASIP Journal on Applied Signal Processing*, special issue on space-time coding and its application.
- Ungerboeck G.,** (1984). Channel coding with multilevel phase signals. *IEEE Tran. Inform. Theory*, pp. 784-791.
- Valenti Matthew C.,** An Introduction to Turbo Codes, *Virginia Polytechnic Inst. & S.U., Blacksburg, Virginia*.

- Valenti Matthew C.**, (1998). Iterative Detection and Decoding for Wireless Communications, *A Proposal for Current and Future Work toward Doctor of Philosophy degree*.
- VISL.**, (2005). Vision Research and Image Sciences Lab., *DCT-Based Image Compression*, Israel.
- Viterbi A. J.**, (1971). Convolutional Codes and their performance in communication Systems, *IEEE Trans. Communication Technology*, vol. COM-19, no. 15, pp. 751–772.
- Wittenben A.**, (1993). A new bandwidth efficient transmit antenna modulation diversity scheme for linear digital modulation, *Proc. IEEE International Conference on Communications, ICC'93*, pp. 1630-1634.



APPENDICES

Appendix A. Noisy 2D image and various simulation results from 2dB to -3dB SNR and their Bit Error Performances (for AW-TS).

Appendix B. Noisy 2D image and various simulation results from 1dB to -3dB SNR and their Bit Error Performances (for IC-TS).

Appendix C. Reconstructed Image with various Bit Planes from 2dB to -3dB SNR (for BSCT system)

Appendix D. Reconstructed images for mask size $M=10$ and 25 and their plane allocation histograms (for progressive transmission).



Appendix A. Noisy 2D image and various simulation results from 2dB to -3dB SNR and their Bit Error Performances (for AW-TS).

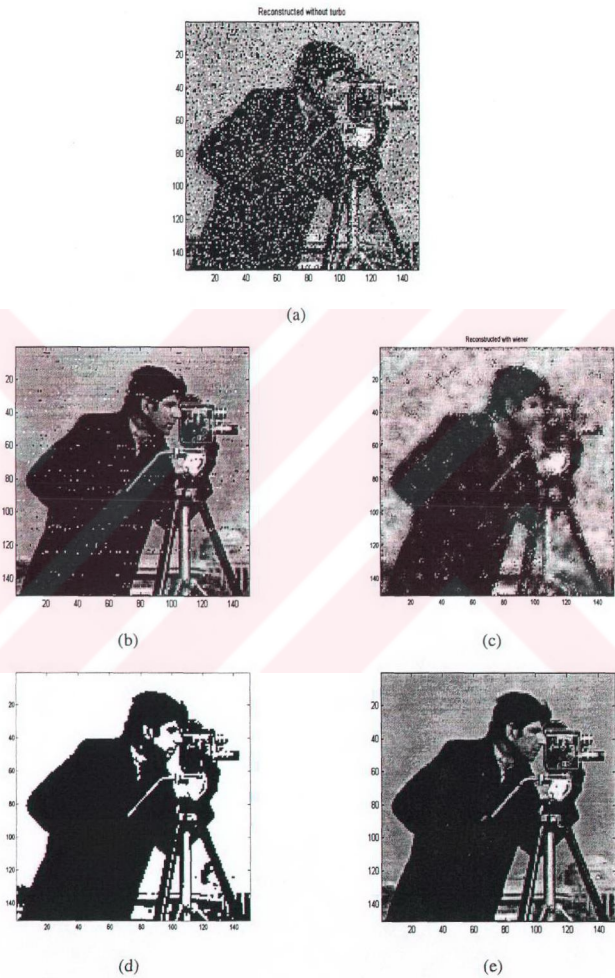
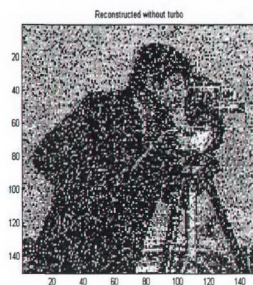


Figure A.1: Noisy 2D image and various simulation results for SNR=2dB.
(a) Corrupted image (b) Turbo processed image (c) Wiener processed image
(d) Most significant part of AW-TS output (e) AW-TS processed image



(a)



(b)



(c)



(d)



(e)

Figure A.2: Noisy 2D image and various simulation results for SNR=1dB.

(a) Corrupted image with SNR= 1dB (b) Turbo processed image (c) Wiener processed image
(d) Most significant part of AW-TS output (e) AW-TS processed image

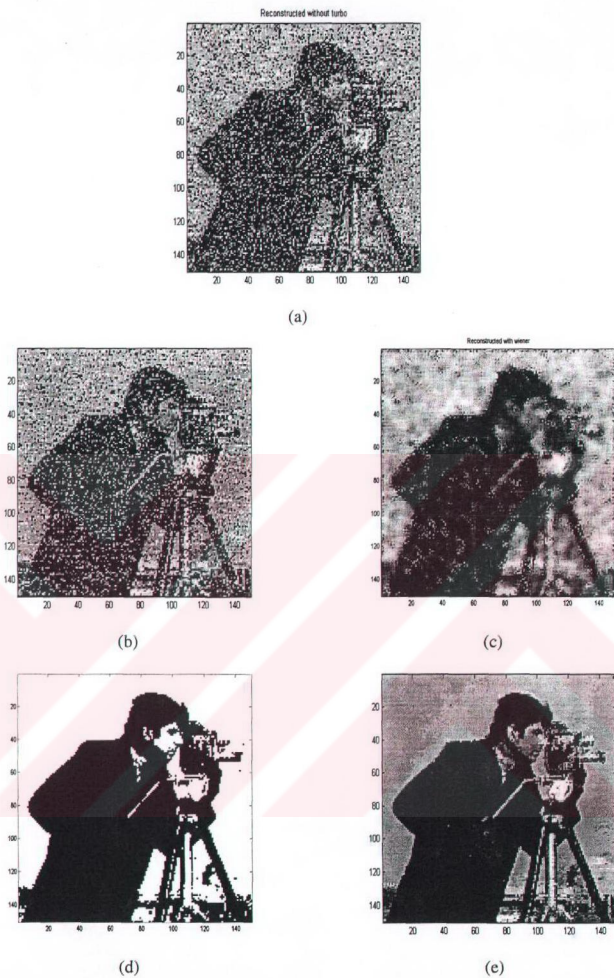
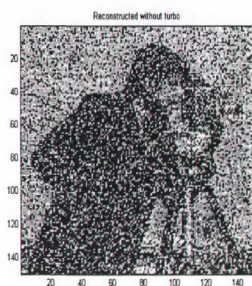
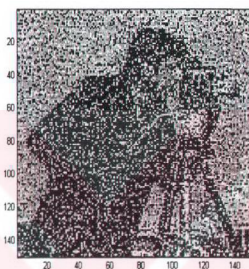


Figure A.3: Noisy 2D image and various simulation results for $\text{SNR}=0\text{dB}$.

(a) Corrupted image with $\text{SNR}=0\text{dB}$ (b) Turbo processed image (c) Wiener processed image
(d) Most significant part of AW-TS output (e) AW-TS processed image



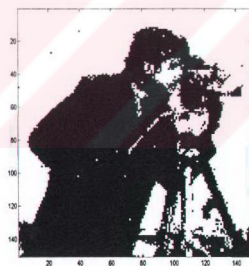
(a)



(b)



(c)



(d)



(e)

Figure A.4: Noisy 2D image and various simulation results for SNR= -1dB.

(a) Corrupted image with SNR= -1dB (b) Turbo processed image (c) Wiener processed image
(d) Most significant part of AW-TS output (e) AW-TS processed image

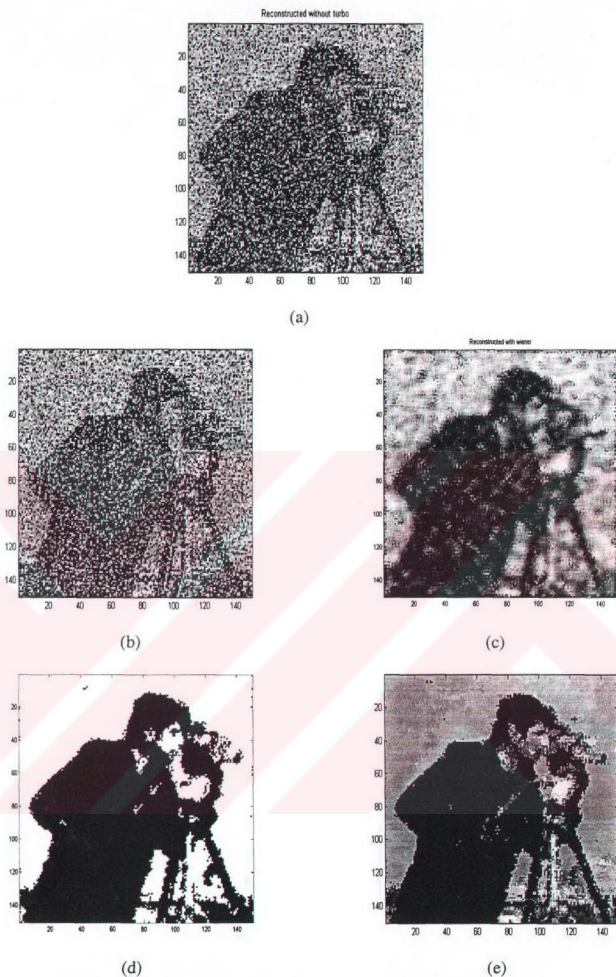


Figure A.5: Noisy 2D image and various simulation results for SNR=-2dB.

(a) Corrupted image with SNR=-2dB (b) Turbo processed image (c) Wiener processed image
(d) Most significant part of AW-TS output (e) AW-TS processed image

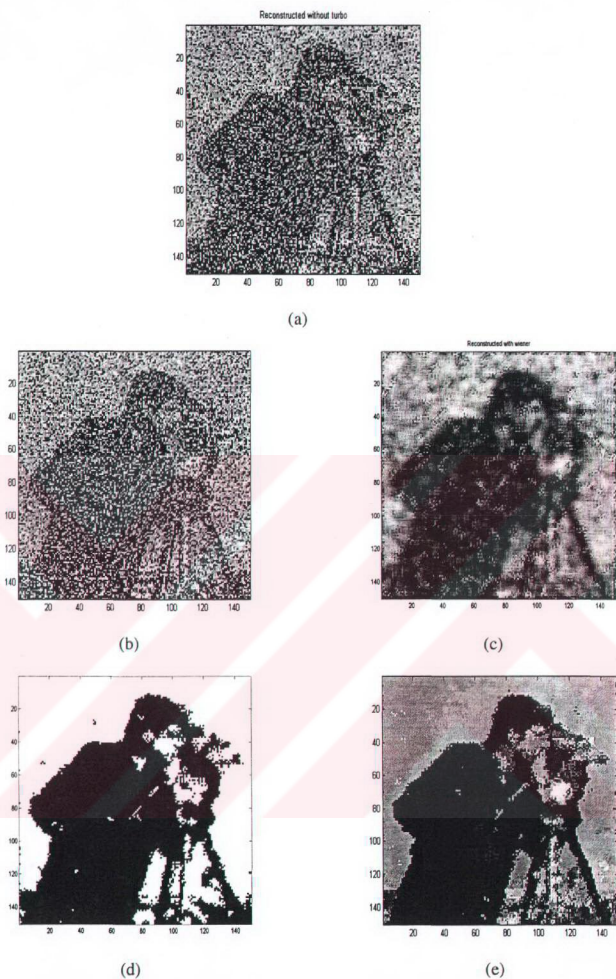


Figure A.6: Noisy 2D image and various simulation results for $\text{SNR}=-3\text{dB}$.
 (a) Corrupted image with $\text{SNR}=-3\text{dB}$ (b) Turbo processed image (c) Wiener processed image
 (d) Most significant part of AW-TS output (e) AW-TS processed image

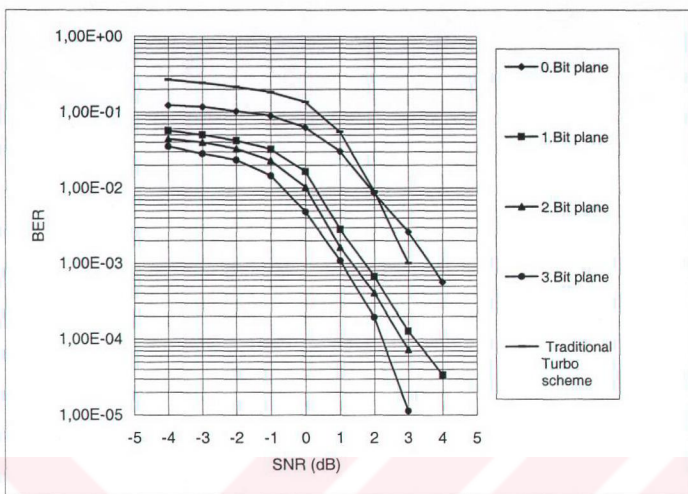


Figure A.7: Bit Error Performance of conventional Turbo and AW-TS with various bit planes.

Appendix B. Noisy 2D image and various simulation results from 1dB to -3dB SNR and their Bit Error Performances (for IC-TS).

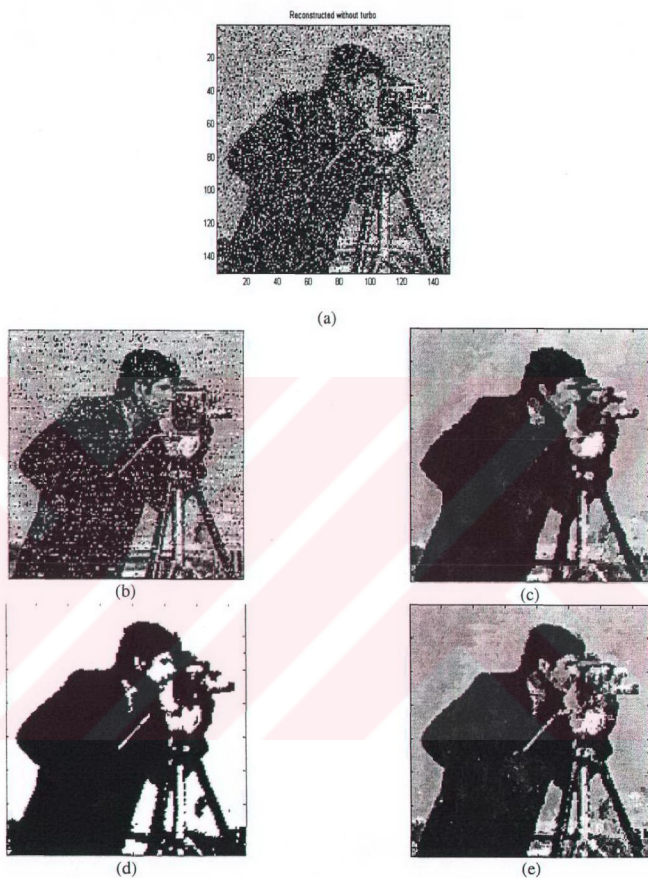


Figure B.1: Noisy 2D image and various simulation results for SNR=1dB.
 (a) Corrupted image with SNR= 1dB (b) Turbo processed image (c) ICIPA processed image
 (d) Most significant part of IC-TS output (e) IC-TS processed image

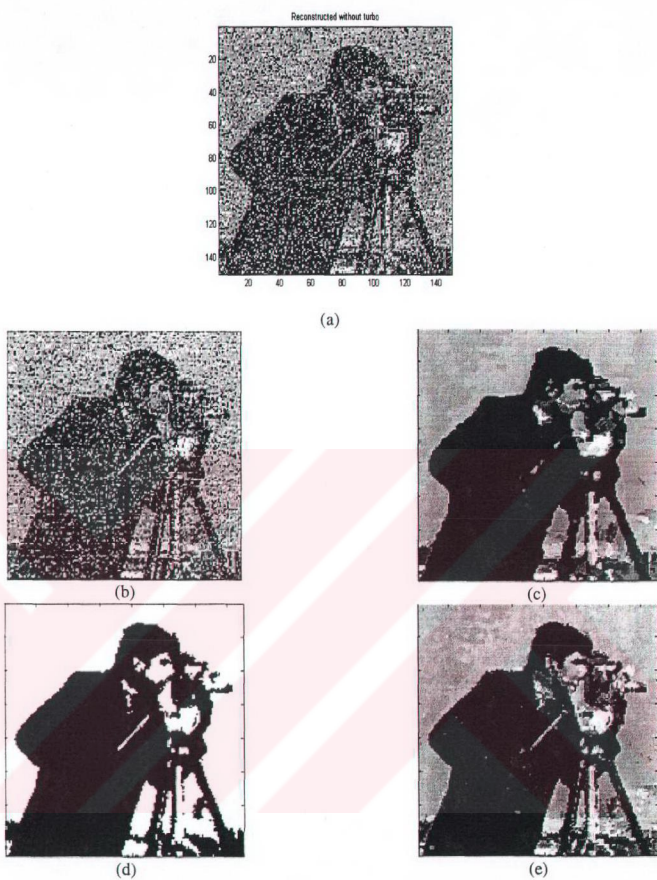


Figure B.2: Noisy 2D image and various simulation results for SNR=0dB.
 (a) Corrupted image with SNR= 0dB (b) Turbo processed image (c) ICIPA processed image
 (d) Most significant part of IC-TS output (e) IC-TS processed image

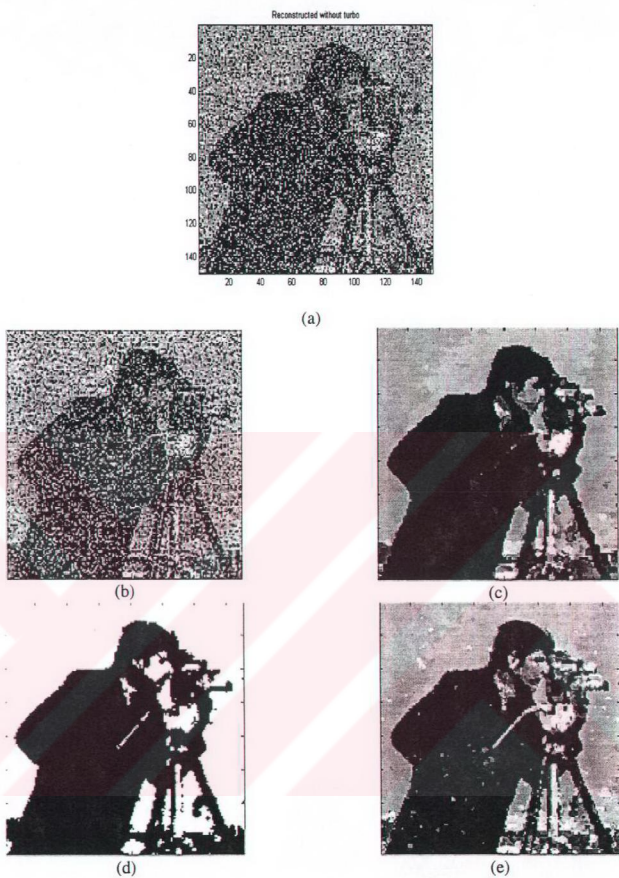
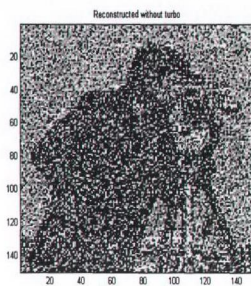


Figure B.3: Noisy 2D image and various simulation results for SNR= -1dB.

(a) Corrupted image with SNR= -1dB (b) Turbo processed image (c) ICIPA processed image
 (d) Most significant part of IC-TS output (e) IC-TS processed image



(a)



(b)



(c)



(d)



(e)

Figure B.4: Noisy 2D image and various simulation results for $\text{SNR} = -2\text{dB}$.

(a) Corrupted image with $\text{SNR} = -2\text{dB}$ (b) Turbo processed image (c) ICIPA processed image
(d) Most significant part of IC-TS output (e) IC-TS processed image

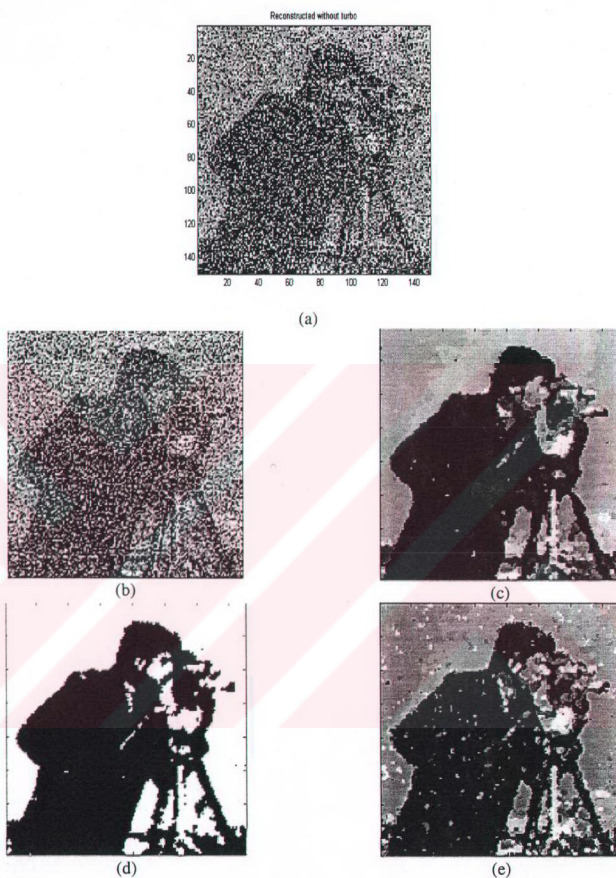


Figure B.5: Noisy 2D image and various simulation results for SNR=-3dB.

(a) Corrupted image with SNR=-3dB (b) Turbo processed image (c) ICIPA processed image
(d) Most significant part of IC-TS output (e) IC-TS processed image

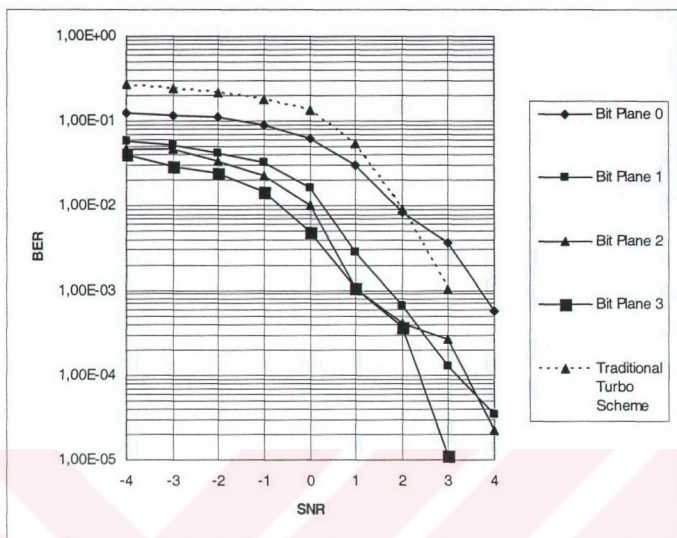


Figure B.6: Bit Error Performance of conventional Turbo and IC-TS with various bit planes.

**Appendix C. Reconstructed Image with various Bit Planes from 2dB to -3dB SNR
(for BSCT system)**



Figure C.1: Reconstructed Image with various Bit Planes at SNR= 2dB.

- (a) Original Image
- (b) Reconstructed with Traditional Turbo scheme without compression
- (c) Reconstructed with BSCT system, for which all Bit planes are transmitted (Compression Ratio is %0)
- (d) Reconstructed with BSCT system, for which 4., 3. and 2.Bit planes are transmitted (Compression Ratio is %25)
- (e) Reconstructed with BSCT system, for which 4. and 3.Bit planes are transmitted (Compression Ratio is %50)
- (f) Reconstructed with BSCT system, for which 4. Bit plane is transmitted (Compression Ratio is %75)



(a)



(b)



(c)



(d)



(e)



(f)

Figure C.2: Reconstructed Image with various Bit Planes at SNR= 1dB.

(a) Original Image

(b) Reconstructed with Traditional Turbo scheme without compression

(c) Reconstructed with BSCT system, for which all Bit planes are transmitted (Compression Ratio is %0)

(d) Reconstructed with BSCT system, for which 4., 3. and 2.Bit planes are transmitted (Compression Ratio is %25)

(e) Reconstructed with BSCT system, for which 4. and 3.Bit planes are transmitted (Compression Ratio is %50)

(f) Reconstructed with BSCT system, for which 4. Bit plane is transmitted (Compression Ratio is %75)



Figure C.3: Reconstructed Image with various Bit Planes at SNR= 0dB

- (a) Original Image
- (b) Reconstructed with Traditional Turbo scheme without compression
- (c) Reconstructed with BSCT system, for which all Bit planes are transmitted (Compression Ratio is %0)
- (d) Reconstructed with BSCT system, for which 4., 3. and 2.Bit planes are transmitted (Compression Ratio is %25)
- (e) Reconstructed with BSCT system, for which 4. and 3.Bit planes are transmitted (Compression Ratio is %50)
- (f) Reconstructed with BSCT system, for which 4. Bit plane is transmitted (Compression Ratio is %75)



Figure C.4: Reconstructed Image with various Bit Planes at SNR= -1dB

- (a) Original Image
- (b) Reconstructed with Traditional Turbo scheme without compression
- (c) Reconstructed with BSCT system, for which all Bit planes are transmitted (Compression Ratio is %0)
- (d) Reconstructed with BSCT system, for which 4., 3. and 2.Bit planes are transmitted (Compression Ratio is %25)
- (e) Reconstructed with BSCT system, for which 4. and 3.Bit planes are transmitted (Compression Ratio is %50)
- (f) Reconstructed with BSCT system, for which 4. Bit plane is transmitted (Compression Ratio is %75)



Figure C.5: Reconstructed Image with various Bit Planes at SNR= -2dB

- (a) Original Image
- (b) Reconstructed with Traditional Turbo scheme without compression
- (c) Reconstructed with BSCT system, for which all Bit planes are transmitted (Compression Ratio is %0)
- (d) Reconstructed with BSCT system, for which 4., 3. and 2. Bit planes are transmitted (Compression Ratio is %25)
- (e) Reconstructed with BSCT system, for which 4. and 3.Bit planes are transmitted (Compression Ratio is %50)
- (f) Reconstructed with BSCT system, for which 4. Bit plane is transmitted (Compression Ratio is %75)



Figure C.6: Reconstructed Image with various Bit Planes at SNR= -3dB

(a) Original Image

(b) Reconstructed with Traditional Turbo scheme without compression

(c) Reconstructed with BSCT system, for which all Bit planes are transmitted (Compression Ratio is %0)

(d) Reconstructed with BSCT system, for which 4., 3. and 2. Bit planes are transmitted (Compression Ratio is %25)

(e) Reconstructed with BSCT system, for which 4. and 3. Bit planes are transmitted (Compression Ratio is %50)

(f) Reconstructed with BSCT system, for which 4. Bit plane is transmitted (Compression Ratio is %75)

Appendix D. Reconstructed images for mask size $M=10$ and 25 and their plane allocation histograms (for progressive transmission).

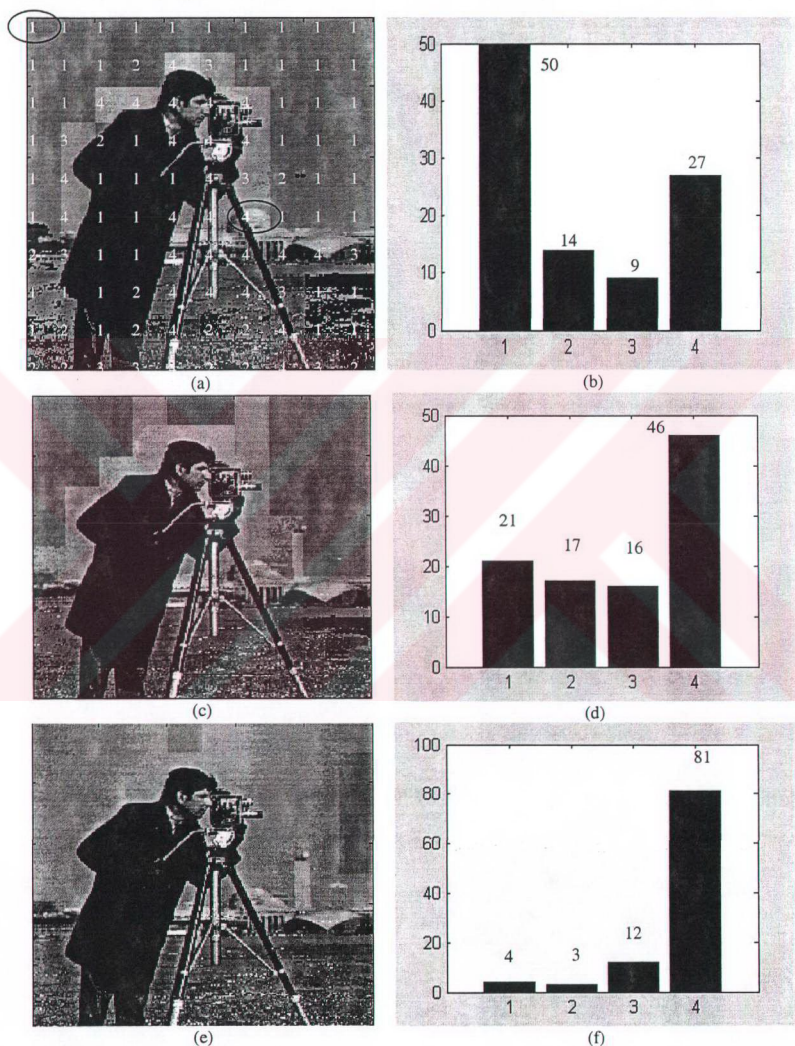
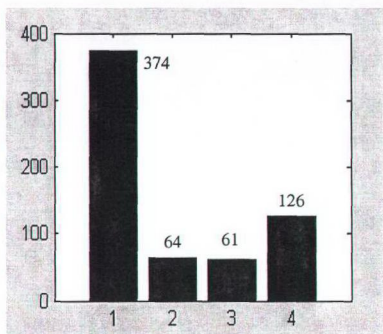


Figure D.1: Reconstructed images for mask size $M=25$ and their (plane allocation) histograms.

- (a) Compression Ratio is % 46 (c) Compression Ratio is % 28 (e) Compression Ratio is % 7.5
 (b) Plane allocation histogram of (a) (d) Plane allocation histogram of (c) (f) Plane allocation histogram of (e)



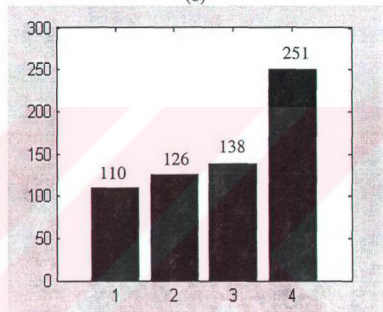
(a)



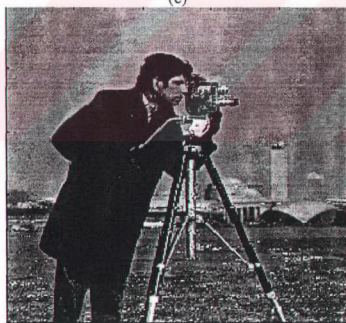
(b)



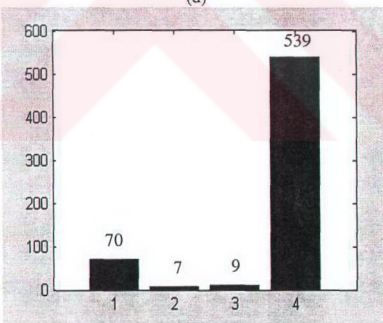
(c)



(d)



(e)



(f)

Figure D.2: Reconstructed images for mask size $M=10$ and their (plane allocation) histograms.

(a) Compression Ratio is % 52

(c) Compression Ratio is % 29

(e) Compression Ratio is % 9

(b) Plane allocation histogram of (a)

(d) Plane allocation histogram of (c)

(f) Plane allocation histogram of (e)

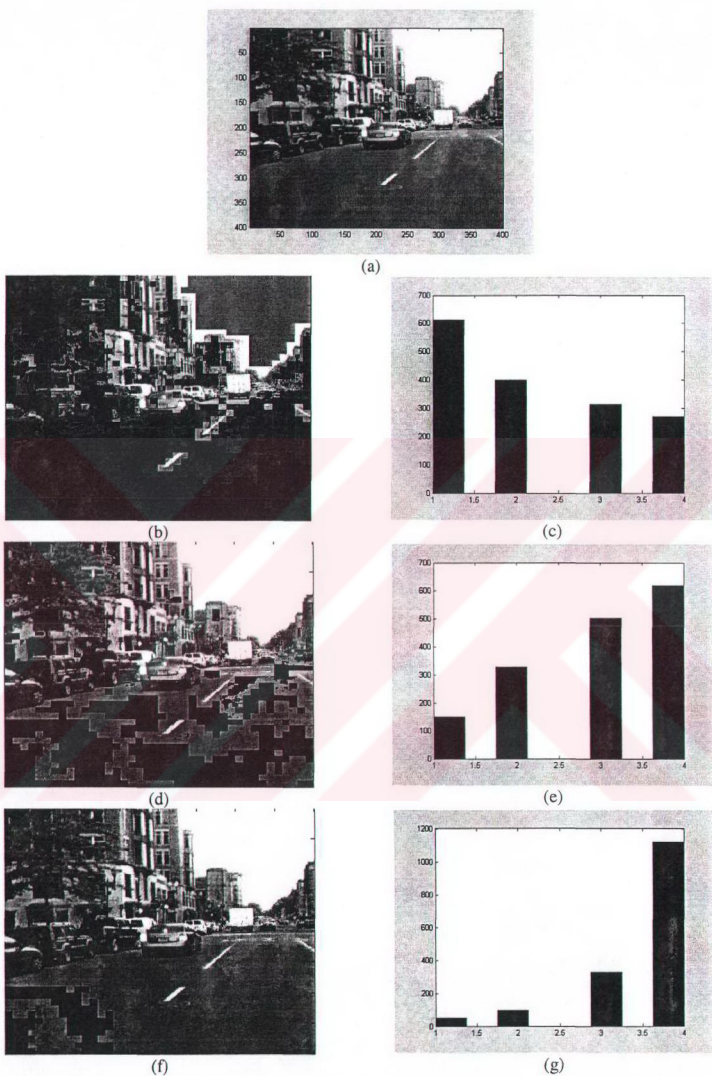


Figure D.3: Reconstructed images for mask size $M=10$ and their (plane allocation) histograms.

(a) Original Image

(b) Compression Ratio is % 86

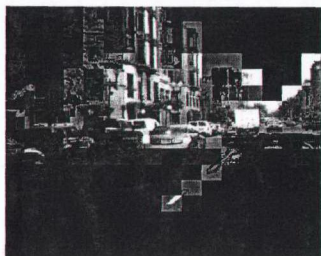
(d) Compression Ratio is % 33

(f) Compression Ratio is % 12

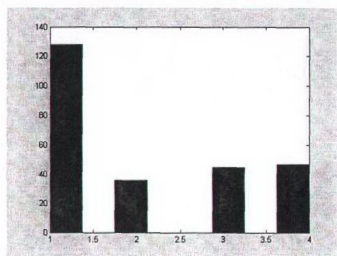
(c) Plane allocation histogram of (a)

(e) Plane allocation histogram of (c)

(g) Plane allocation histogram of (e)



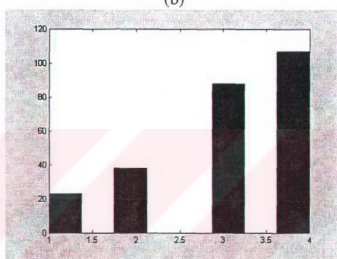
(a)



(b)



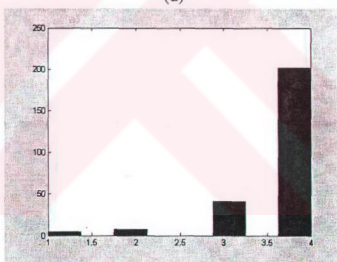
(c)



(d)



(e)



(f)

Figure D.4: Reconstructed images for mask size $M=25$ and their (plane allocation) histograms.

(a) Compression Ratio is % 88

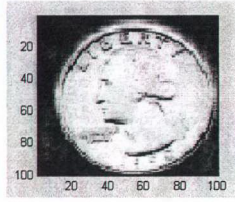
(c) Compression Ratio is % 29

(e) Compression Ratio is % 8

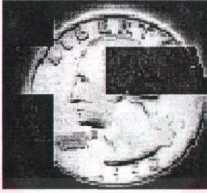
(b) Plane allocation histogram of (a)

(d) Plane allocation histogram of (c)

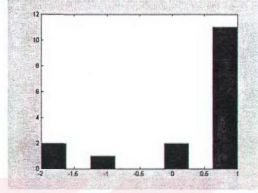
(f) Plane allocation histogram of (e)



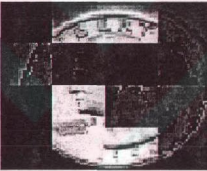
(a) Original Image



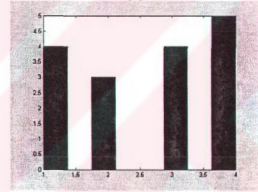
(b)



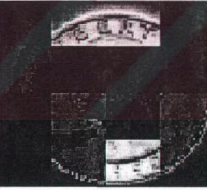
(c)



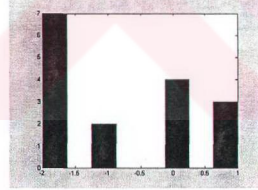
(d)



(e)



(f)



(g)

Figure D.5: Reconstructed images for mask size $M=25$ and their (plane allocation) histograms.

(a) Original Image

(b) Compression Ratio is % 18

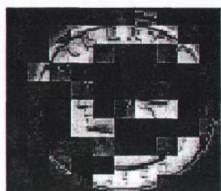
(d) Compression Ratio is %52

(f) Compression Ratio is % 83

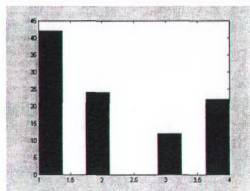
(c) Plane allocation histogram of (b)

(e) Plane allocation histogram of (d)

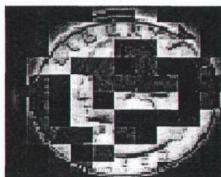
(g) Plane allocation histogram of (f)



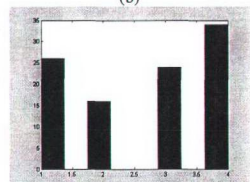
(a)



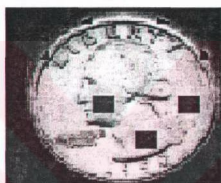
(b)



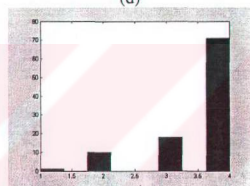
(c)



(d)



(e)



(f)

Figure D.6: Reconstructed images for mask size $M=10$ and their (plane allocation) histograms.

(a) Compression Ratio is % 87 (c) Compression Ratio is % 50 (e) Compression Ratio is % 11
 (b) Plane allocation histogram of (a) (d) Plane allocation histogram of (c) (f) Plane allocation histogram of (e)

BIOGRAPHY

Kenan Büyükkatak was born in Ankara, Turkey on 11 June 1966. He graduated from Electrical and Electronics Department of Istanbul Technical University in 1988. He received M.Sc. honours degree in Electronics Engineering from Middle East Technical University in 1997. He is pursuing Ph.D. at Electrical & Electronics Engineering Department of Istanbul Technical University on Turbo-Coding and Image Transmission Systems. He has currently been working in Turkish Air Force Academy as a Major. His research area is Turbo coding, satellite communications, Image Transmission and Processing, Adaptive and stochastic Signal Processing, neural networks and microcontrollers. Büyükkatak is married and has a daughter and a son.