

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI[®]

**Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

**ADAPTIVE DIGITAL IMAGE COMPRESSION
BASED ON
SEGMENTATION AND BLOCK CLASSIFICATION**

by

Mahmoud R. El-Sakka

**A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering**

Waterloo, Ontario, Canada, 1997

©Mahmoud R. El-Sakka 1997



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

395 Wellington Street
Ottawa ON K1A 0N4
Canada

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-44784-7

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Over the last few decades, many good image compression schemes have been developed. The performance of these schemes varies from low to high compression ratios with low to high levels of degradation of the decompressed images. Since the end users of decompressed images are usually human beings, consequently, it is natural that attempts should be made to incorporate some of the human visual system properties into the encoding schemes to achieve even further compression with less noticeable degradations.

This thesis presents a new digital image compression scheme which exploits one of the human visual system properties—namely that of, recognizing images by their regions—to achieve high compression ratios. It also assigns a variable bit count to each image region that is proportional to the amount of information it conveys to the viewer. The new scheme copes with image non-stationarity by adaptively segmenting the image—using quad-trees segmentation approach—into variable-block sized regions, and classifying them into statistically and perceptually different classes. These classes include, a smooth class, a textural class, and an edge class. Blocks in each class are separately encoded. For smooth blocks, a new adaptive prediction technique is used to encode block averages. Meanwhile, an optimized DCT-based technique is used to encode both edge and textural blocks.

Based on extensive testing and comparisons with other existing compression techniques, the performance of the new scheme surpasses the performance of the JPEG standard and goes beyond its compression limits. In most test cases, the new compression scheme results in a maximum compression ratio that is at least twice of JPEG, while exhibiting lower objective and subjective image degradations. Moreover, the performance of the new block-based compression is comparable to the performance of the state-of-the-art wavelet-based compression technique and provides a good alternative when adaptability to image content is of interest.

Acknowledgements

First and foremost, I would like to thank and praise *Allah* almighty, the only One *God*, for enlightening my way and directing me through each and every success I have reached or may reach.

I would like to express my heartfelt gratitude to my supervisor Professor *Mohamed Kamel* for his constant encouragement, support, and invaluable guidance. He has shaped my view of what constitutes valuable research. He was always there when I needed him, not only at the research level, but also at the personal level. Professor *Kamel* always treated me as a son, rather than just a student. May *Allah* reward him in this world and in the hereafter for all his good deeds.

To Professor *Mohamed Nazih El-Derini* (of the Alexandria University, Egypt), I would like to extend my sincere thanks for his wisdom and vision for encouraging and pushing me to travel and seek knowledge.

I would also like to gratefully recognize the contribution of Professor *John Robinson* for his valuable comments on my work.

I am thankful to my defense examination committee members: Professor *Rabab Ward* (of the University of British Columbia), Professor *George Freeman* (of the Electrical & Computer Engineering Department), Professor *Mohamed Kamel*, Professor *Andrew Wong*, Professor *John Robinson*, and Professor *Otman Basir* for their meticulous reading of this thesis.

My thanks also go to members of the PAMI group, especially Professor *Andrew Wong* and Professor *Mohamed Kamel* (the PAMI directors), as well as Dr. *Puiwing Wong* (the former PAMI assistant director) for their trust, respect, support, and co-operation during my tenure as a computer systems administrator for the PAMI group.

This work has been supported in part by *NSERC, OGS, UW, FOE, and ICR*. This support is greatly appreciated.

To my friends I would like to express my sincere appreciation, especially Dr. *Khaled Hassanein* and Dr. *Gasser Auda* for reviewing several drafts of this dissertation and providing suggestions which helped in enhancing its presentation. I would also like to thank my friend Dr. *Hamada Ghenniwa* for attending many dry runs in preparation for the defense presentation of this thesis.

To my family back home in Egypt, I would like to extend my profound thanks and appreciation for their deep understanding and moral support. My wholehearted thanks go to my beloved wife *Hiba El-Ghazali* for her love, patience, encouragement, and understanding. Finally, I would like to thank my beloved daughter *Yomna* for filling my life with fun and pleasantness ever since she was born a year ago.

Dedication

To

**my mother and the memory of my father,
my wife and my daughter,
my brothers and my sister,
my parents in law,
my brothers in law and my sisters in law,
my nephews and my niece,
the rest of my family and the rest of my wife's family,
and finally
to whom it may concern.**

Contents

Abstract	iv
Acknowledgements	v
Dedication	vi
Abbreviations	xvi
1 Introduction	1
1.1 Introduction to Digital Image Compression	1
1.2 Motivations	3
1.3 Objectives	5
1.4 Outline of Thesis	6
2 Image Compression Techniques: A Review	7
2.1 Types of Image Information	7
2.2 Classifications of Image Compression Techniques	8
2.3 Waveform Compression Techniques	9
2.3.1 Non-predictive Waveform Techniques	11
2.3.2 Predictive Waveform Techniques	11

2.4	Transform Compression Techniques	12
2.4.1	Block-transform Techniques	13
2.4.2	Filter-transform Techniques	14
2.4.3	DCT-based Versus Wavelet-based Compressions	16
2.5	Model-based Compression Techniques	17
2.6	Hybrid Compression Techniques	19
2.7	Quantization	19
2.7.1	Scalar Quantization	20
2.7.2	Vector Quantization	20
2.8	Previous Work Most Relevant to This Thesis	21
3	ABC-SC: A New Adaptive Compression Technique	26
3.1	An Overview of the Proposed Compression Technique	26
3.1.1	Quad-tree Representation	28
3.1.2	Homogeneity Criteria	32
3.1.3	Adaptive Differential Pulse Code Modulator	35
3.1.4	Discrete Cosine Transform	37
3.1.5	Quantizers	41
3.1.6	Zigzag-ordering	44
3.1.7	Run-length Encoder	44
3.1.8	Arithmetic Encoder	46
3.1.9	Post-processing	47
3.2	Parameters Adjustment	50
3.2.1	Thresholding and Quantization Functions	50
3.2.2	Neighboring Block Averages Parameter	66
3.2.3	Run-length Parameter	66

4	Results and Discussion	75
4.1	Performance Metrics	75
4.2	Results	78
4.2.1	Two-class Case	80
4.2.2	Three-class Case	89
4.2.3	ADPCM Effect	92
4.2.4	Lossless Encoder Effect	92
4.2.5	Post-processing Effect	96
4.2.6	Quad-tree Overhead	96
4.2.7	Execution Time	98
4.3	Discussion	99
5	Concluding Remarks and Future Work	103
5.1	Conclusions	103
5.2	Future Work	105
A	Results of the Comparison with JPEG/IJPEG	106
B	Results of the Comparison with SPIHT-A/SPIHT-B	121
	Bibliography	135

List of Tables

2.1	Performance results of various encoding techniques for the 512×512 Lena image.	25
3.1	All possible relative pixel values of A, B, and C.	38
3.2	The ADPCM prediction rules.	39
3.2	The ADPCM prediction rules, (continued).	40
3.3	The outputs of all non-texture-related functions.	62
3.3	The outputs of all non-texture-related functions, (continued).	63
3.3	The outputs of all non-texture-related functions, (continued).	64
4.1	A summary of the rate-distortion performance for decompressed images shown in Figures 4.2 and 4.3.	82
4.2	A summary of the rate-distortion performance for decompressed images shown in Figures 4.5–4.7.	85
4.3	The average compression/decompression time (in seconds) for the Lena image.	99
A.1	A summary of rate-distortion performance for the decompressed images shown in Figures A.2–A.13.	120

B.1	A summary of rate-distortion performance for the decompressed images shown in Figures B.1–B.12.	134
------------	--	------------

List of Figures

1.1	An example showing that HVS recognizes images by their regions, not by the intensity value of their pixels.	4
1.2	An example showing that HVS focuses on edge information when trying to recognize a scene.	5
2.1	A classification of image compression techniques.	10
2.2	General block-transform encoding/decoding block diagrams.	13
2.3	General filter-transform encoding/decoding block diagrams.	15
2.4	General model-based encoding/decoding block diagrams.	18
3.1	Block diagram of ABC-SC.	29
3.2	Detailed encoder block diagram.	30
3.3	Detailed decoder block diagram.	31
3.4	A Quad-tree segmentation example.	33
3.5	A textural candidate block and its augmented block.	35
3.6	The ADPCM block diagram.	36
3.7	The zigzag-ordering sequence.	44
3.8	A post-processing example.	51
3.9	The naming conventions of the locations within the corner.	52

3.10	Another post-processing example.	53
3.11	The 8 original images which are used during the functions' identification process.	56
3.11	The 8 original images which are used during the functions' identification process, (continued).	57
3.12	The non-texture-related functions' identification algorithm.	59
3.13	An example of an envelope curve for rate-distortion curves which are generated for K_{ADPCM}^{smooth} during the functions' second identification iteration.	60
3.14	Segmentation results for the Lena image at $QF equals 200$	67
3.15	Pixels distribution for the Lena image.	68
3.16	The effect of both v and QF on the utilization of the prediction rules.	69
3.17	The values of v as a function of QF	70
3.18	The relation between QF and compression ratio for different r values, normalized to the $r = 6$ case.	72
3.19	The values of r as a function of QF	73
3.20	The relation between QF and compression ratio for the adaptive r value case, normalized to the case of not using run-length encoding (i.e., the $r = 0$ case).	74
4.1	An example showing how PSNR and $RMSE$ are sometimes misleading when judging the actual loss of fidelity.	77
4.2	ABC-SC, IJPEGE, and JPEG compression results for the Lena image.	81
4.3	ABC-SC and IJPEGE compression results for the Lena image.	83
4.4	Comparison of rate distortion results between ABC-SC, IJPEGE, JPEG, and some other recent segmentation based encoding techniques for the Lena image.	84

4.5	ABC-SC, SPIHT-A, and SPIHT-B compression results I for the Lena image.	86
4.6	ABC-SC, SPIHT-A, and SPIHT-B compression results II for the Lena image.	87
4.7	ABC-SC, SPIHT-A, and SPIHT-B compression results III for the Lena image.	88
4.8	The effect of texture-quality-factor on the performance of ABC-SC for the Lena image.	90
4.9	The compressed file-sizes for the Lena image.	91
4.10	The effect of texture-quality-factor on the performance of ABC-SC for the Barbara image.	93
4.11	The ADPCM effect for the Lena image.	94
4.12	The effect of the lossless encoder on the performance of ABC-SC for the Lena image.	95
4.13	Enhancement results for the Lena image due to the post-processing. . . .	97
4.14	The quad-tree overhead in ABC-SC for the Lena image.	98
A.1	Original test images.	107
A.2	ABC-SC, IJPEGE, and JPEG compression results for the Monarch image.	108
A.3	ABC-SC, IJPEGE, and JPEG compression results for the Boats image. . .	109
A.4	ABC-SC, IJPEGE, and JPEG compression results for the Rocks image. . .	110
A.5	ABC-SC, IJPEGE, and JPEG compression results for the Barbara image. .	111
A.6	ABC-SC, IJPEGE, and JPEG compression results for the Zelda image. . .	112
A.7	ABC-SC, IJPEGE, and JPEG compression results for the Peppers image. .	113
A.8	ABC-SC, IJPEGE, and JPEG compression results for the Goldhill image. .	114
A.9	ABC-SC, IJPEGE, and JPEG compression results for the F16 image. . . .	115
A.10	ABC-SC, IJPEGE, and JPEG compression results for the Woman image. .	116
A.11	ABC-SC, IJPEGE, and JPEG compression results for the Tulips image. . .	117

A.12	ABC-SC, IJPEG, and JPEG compression results for the Bridge image. . .	118
A.13	ABC-SC, IJPEG, and JPEG compression results for the Man image. . . .	119
B.1	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Monarch image.	122
B.2	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Boats image.	123
B.3	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Rocks image.	124
B.4	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Barbara image.	125
B.5	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Zelda image.	126
B.6	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Peppers image.	127
B.7	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Goldhill image.	128
B.8	ABC-SC, SPIHT-A, and SPIHT-B compression results for the F16 image.	129
B.9	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Woman image.	130
B.10	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Tulips image.	131
B.11	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Bridge image.	132
B.12	ABC-SC, SPIHT-A, and SPIHT-B compression results for the Man image.	133

Abbreviations

ABC-SC	Adaptive Block Compression method based on Segmentation and Classification
ADPCM	Adaptive Differential Pulse Code Modulation
AM-BTC	Absolute Moment Block Truncation Coding
bpp	bits per pixel
BTC	Block Truncation Coding
CR	Compression Ratio
DCT	Discrete Cosine Transform
DM	Delta Modulation
DPCM	Differential Pulse Code Modulation
EZW	Embedded Zero-tree Wavelet approach
HVS	Human Visual System
I-ADPCM	Inverse ADPCM
I-DCT	Inverse DCT
IJPEG	Improved JPEG
JPEG	Joint Photographic Experts Group
KLT	Karhunen-Loève Transform
PCM	Pulse Code Modulation
PSNR	Peak Signal-to-Noise-Ratio
QMF	Quadrature Mirror Filter
QF	Quality-Factor
RL	Run Length
RMSE	Root Mean Squared Error
SPIHT	Set Partitioning In Hierarchical Trees
SPIHT-A	SPIHT with Arithmetic encoder
SPIHT-B	SPIHT with Binary encoder
TQR	Texture-Quality-Ratio
VQ	Vector Quantization

Chapter 1

Introduction

It is widely believed that a picture is worth more than a thousand words. However, dealing with digital pictures (images) requires far more computer memory and transmission time than that needed for plain text. To be able to handle, efficiently, the huge amount of data associated with images, compression schemes are needed. Image compression is a process intended to yield a compact representation of an image, hence, reducing the image storage/transmission requirements.

This chapter provides a general introduction to the digital image compression field and its applications. It also presents the motivations and objectives of this thesis. Finally, it gives an outline of later chapters in the thesis.

1.1 Introduction to Digital Image Compression

Since the very beginning of digital image processing in the 1950's, image compression has been recognized as an important field. This is due to the large amount of data which need to be handled for transmission or storage of digital images. The objective of image compression is to achieve a low bit rate in the digital representation of an input image (i.e.,

compact digital image representation) with a minimal perceived loss of picture quality.

To appreciate the need for image compression, consider the storage and transmission requirements of a typical gray scale digital image of size 512×512 with 256 gray levels. Such an image requires at least 262144 bytes of storage space. To transmit this uncompressed image over a 64 Kbits/second channel, it would take more than half minute. With a good compression scheme that offers an excellent quality of the reproduced image with, for example, a bit rate of 0.25 bit/pixel, or less, these requirements can be dramatically reduced. With such a compression scheme, the storage required to save the above image may be reduced to 8192 bytes, or less. At the same time, its transmission time over the above mentioned communication channel may be reduced to less than a second.

Applications of image compression are numerous. These applications range from:

1. image communication applications, including facsimile machines, television sets, picture phones, and video conferencing; and
2. image retrieval applications, including desktop publishing, real estate, education, security, internet shopping, and printing industry;

up to:

1. the use of satellite imagery for weather and other earth-resource applications (i.e., remote sensing applications);
2. the control of remotely piloted vehicles in military applications;
3. space control applications; and
4. hazardous waste control applications.

All these applications require transmission and storage of a huge amount of image data. Hence, without a doubt, one can say that image compression is still an important present day research issue, and it is likely to stay as such for years to come.

1.2 Motivations

Decompressed images are usually observed by human beings. Therefore, their fidelities are subject to the capabilities and limitations of the *Human Visual System* (HVS). A significant property of the HVS is the fact that it *recognizes images by their regions* and not by the intensity value of their pixels [1]. Figure 1.1 illustrates this property. Although the two rectangular stripes in the figure have an identical intensity grey-level, 64, they appear to have different brightnesses. This is because they are placed in different surroundings.

In addition to the above property, when an observer looks to an image trying to understand it, he/she searches for distinguishing features such as edges (not pixel values) and, mentally, combines them together into recognizable groupings. In support of this, Figure 1.2(a) shows the Goldhill image, whereas Figure 1.2(b) shows the image after applying a Sobel operator [2] and a threshold. Even though Figure 1.2(b) consists of just a small number of lines, we can easily understand the content of the image and associate it with the original Goldhill image. It can also be concluded from this figure that when the HVS tries to recognize a scene, it focuses on edge information more than textural information.

Studies on some psycho-visual properties of the HVS [3] indicate that:

1. the collection of the individual pixel intensity values without any interaction between them is not what produces the visual perception,
2. the edge regions in an image is responsible for our perception, and
3. edge regions are of higher importance to our perception than textural regions.

This suggests that this HVS property (recognizing images by their regions) can be exploited by segmenting images into regions based on the amount of information each region

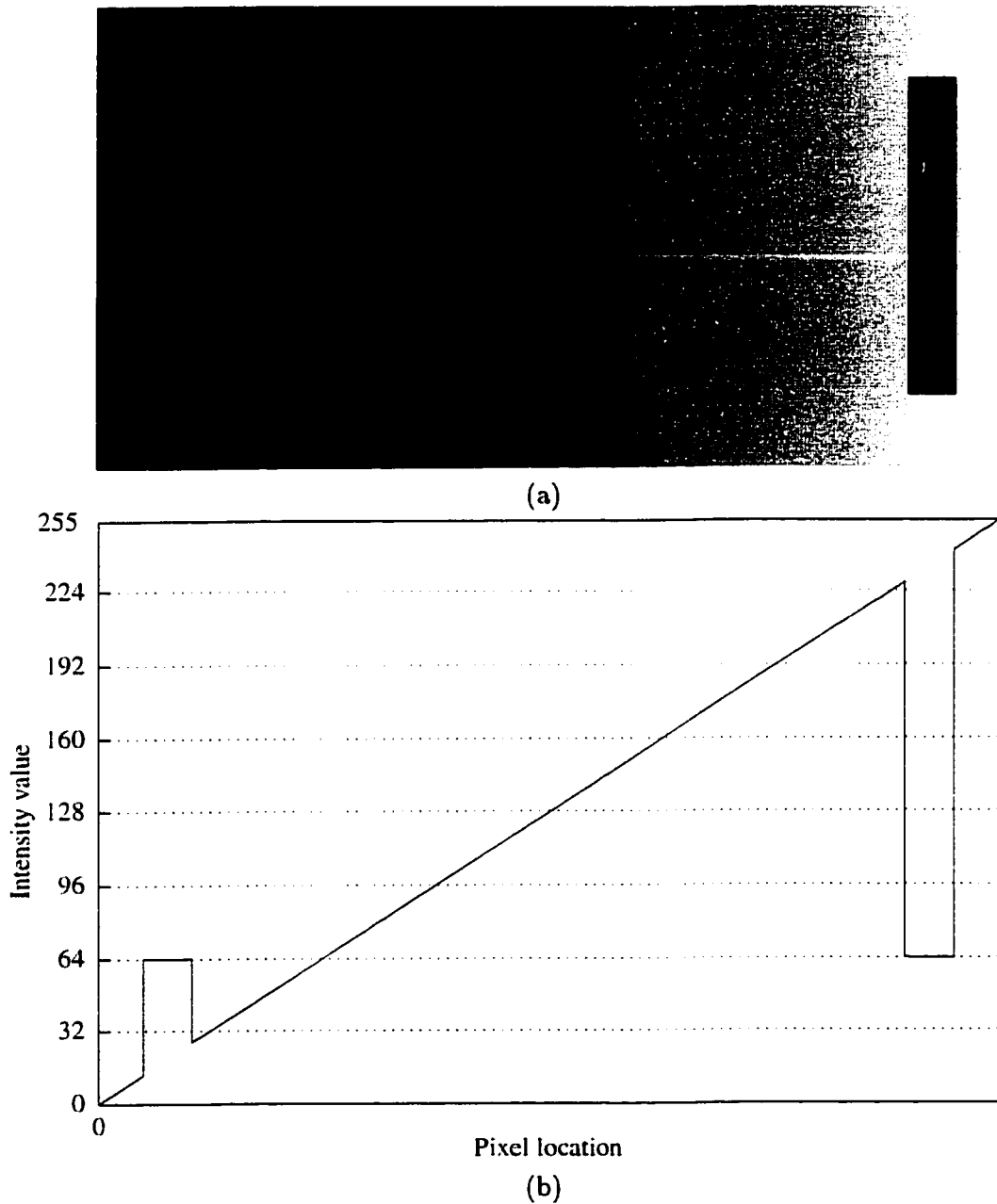


Figure 1.1: An example showing that HVS recognizes images by their regions, not by the intensity value of their pixels. (a) is an image having two rectangular strips which are intensity identical but as they lie on backgrounds with different intensity, they appear to have different brightness; and (b) shows the pixel intensity values along the horizontal center line of the image in (a).



Figure 1.2: An example showing that HVS focuses on edge information when trying to recognize a scene. (a) The Goldhill image and (b) the image after applying a Sobel operator and a threshold.

conveys to the viewer. Then, regions of each category could be encoded using a distinct encoding procedure. This encoding procedure should preserve the main visual characteristics of this particular category (focusing on useful information) while reducing the existing correlation (reducing redundant information), and neglecting some of the irrelevant details (omitting irrelevant information).

1.3 Objectives

The objectives of this thesis is to develop a general technique for digital image compression that meets a number of diverse requirements, such as:

1. achieving high compression ratios by exploiting the HVS property of recognizing images by their regions, and assigning bits in each region proportional to the amount of information conveyed;

2. exceeding the compression performance of the current image compression standards, and providing a state-of-the-art block-based compression technique;
3. giving the users the ability to trade off between desired compression and image quality;
4. being dynamically adaptive without requiring training or a priori knowledge about the image being compressed;
5. having modest computational complexity; and
6. being amenable to hardware implementation.

1.4 Outline of Thesis

This thesis is divided into five chapters and two appendices. Chapter 1 provides a general introduction to the image compression field, as well as the motivations and the objectives of this thesis. Chapter 2 classifies and reviews image compression techniques. In Chapter 3, the proposed compression technique is described. In Chapter 4, the choice of performance metrics for evaluating the new technique and the results of its performance in comparison with existing techniques are presented and discussed. Chapter 5 gives some concluding comments as well as some possible improvements and potential extensions of this work for future research. Finally, Appendix A and Appendix B present more results of the comparison with other compression techniques.

Chapter 2

Image Compression Techniques: A Review

Research in visual communications has advanced very rapidly in the last few decades. Over the years, an enormous number of different image compression techniques have been developed for still-images. All these techniques try, by one way or another, to represent image data by a minimal (in some sense) amount of information sufficient to maintain a certain quality level.

In this chapter, types of image information are identified. Then, image compression techniques are classified. Further, the basic idea, as well as some examples, of each class of compression techniques are briefly stated. Finally, previous work most relevant to this thesis is reviewed.

2.1 Types of Image Information

Generally, images carry three main types of information: *redundant*, *irrelevant*, and *useful*. Redundant information is the deterministic part of the information which can be repro-

duced, without any loss, from other information contained in the image (i.e., *inter-pixel redundancy*), for example, low varying background information. Irrelevant information is the part of information that has enormous details which is beyond the limit of perceptual significance (i.e., *psycho-visual redundancy*). Useful information is the part of information which is neither redundant nor irrelevant.

The ultimate goal of any lossy compression system is to encode an image in its minimal representation while ensuring that it can be clearly reconstructed with the minimal possible distortion. This compression task can be accomplished by focusing on the useful information while reducing the redundant information and omitting the irrelevant information. When more useful information is considered, less distortion and lower compression ratios are achieved. On the other hand, when more redundant and irrelevant information are reduced, higher compression ratios are reached without introducing much distortion.

2.2 Classifications of Image Compression Techniques

Image compression techniques can be classified in several ways. One such classification is based on the reversibility property of the compression technique, i.e., whether the technique is reversible or irreversible. Reversible techniques (also called information-lossless, or lossless, techniques) are able to reconstruct the same original image exactly. Conversely, irreversible techniques (also called information-lossy, or lossy, techniques) introduce some distortions which should be kept as un-noticeable as possible.

Another way of classification is often made based on the adaptivity property of the compression technique, i.e., whether the technique is fixed or adaptive, in a sense that the parameters used are fixed or adjusted as a function of the local image data.

Perhaps the most comprehensive way to classify image compression techniques can be

based on the compression space-domain, i.e., the domain where the compression technique is applied. These domains include:

1. *spatial-space-domain*, where pixel intensity values are combined in an appropriate way and encoded;
2. *transform-space-domain*, where an image is transformed to another domain—which is significantly different from the image pixel intensity domain—and the transformed coefficients are combined in an appropriate way and encoded; and
3. *feature-space-domain*, where higher-level image features—which is above the level of individual pixel intensity values—are extracted and their parameters are encoded.

Spatial-, transform-, and feature-space-domain compression techniques are called waveform, transform, and model-based compression techniques, respectively. Figure 2.1 shows this classification of image compression techniques with examples of each class. The following sections (Section 2.3–Section 2.6) give brief description of the basic idea, with some examples, of each class.

2.3 Waveform Compression Techniques

In waveform compression techniques, image pixel intensity values (or some simple variations of them) are quantized and, then, encoded. These techniques attempt to exploit the existing correlation among pixel intensity values to reduce the inter-pixel redundancy. Simplicity is the main advantage of waveform compression techniques. On the other hand, they do not achieve high compression ratios. Typically, they achieve compression in the order of a single digit to 1 ratio.

Waveform compression techniques can be further classified into two main categories: predictive and non-predictive compression techniques. The main difference between these

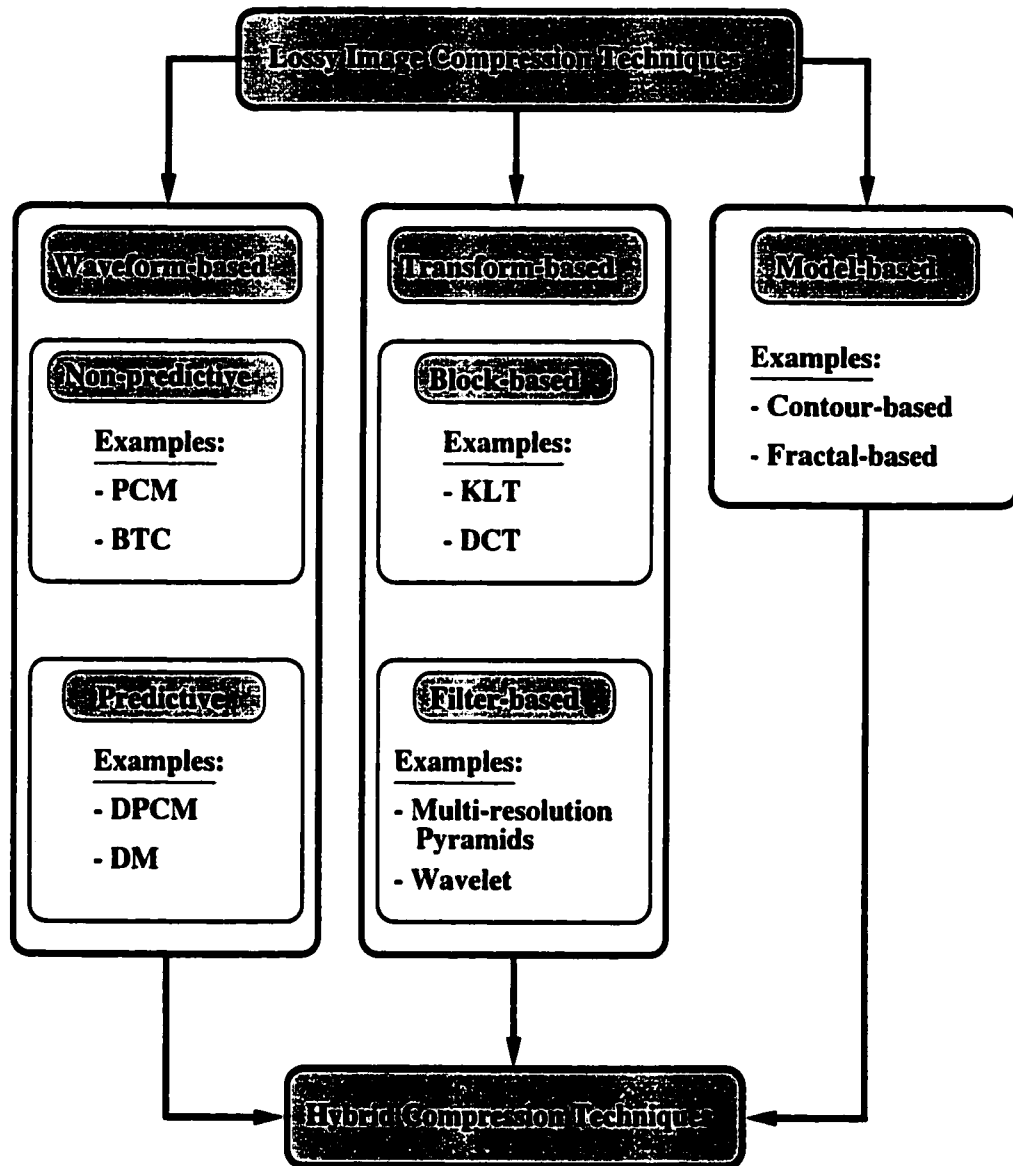


Figure 2.1: A classification of image compression techniques.

two categories is whether pixel intensity values are predicted before encoding or not.

2.3.1 Non-predictive Waveform Techniques

Non-predictive waveform compression techniques include:

1. *Pulse Code Modulation (PCM)*—the simplest waveform compression technique—, which is nothing but quantizing and encoding the image pixel intensity values; and
2. *Block Truncation Coding (BTC)*, where an image is divided into small blocks and pixels are quantized into two levels such that some of the original block statistics are preserved [4, 5]. In the basic form, the first and the second moments (i.e., the block mean and the variance, respectively) are preserved.

2.3.2 Predictive Waveform Techniques

In a general predictive waveform encoding scheme, the correlation between the neighboring pixel intensity values is used to form a prediction for each pixel. Then, the predicted value is subtracted from the actual pixel intensity value to form an error image, which is much less correlated than the original image data. Finally, the error image is quantized and encoded. Predictive waveform compression techniques include:

1. *Differential Pulse Code Modulation (DPCM)*, where the current encoded pixel intensity value is predicted—by using the most recently encoded pixel intensity values—and the prediction error is quantized and encoded [6]; and
2. *Delta Modulation (DM)*, which is a special case of DPCM where the prediction errors are quantized into only two levels [6].

2.4 Transform Compression Techniques

In transform compression techniques [7, 8], the transformed coefficients are quantized and, then, encoded. The main differences among these techniques come from the properties and the characteristics of the transformation that each technique uses. These properties and characteristics include:

1. the capability to reduce the correlations among transformed coefficients, hence, achieving compression (*correlation reduction property*);
2. the capability to concentrate a large amount of energy in a small set of transformed coefficients, hence, many transformed coefficients can be discarded without seriously affecting the fidelity of the reconstructed image (*energy compaction property*);
3. the existence of efficient way to compute the transformed coefficients and their inverse (*fast implementation characteristic*); and
4. the ability to decompose the two-dimensional transformation, and its inverse, into two one-dimensional transformations (*separability characteristic*).

In general, transform compression techniques perform significantly better and achieve higher compression ratios than those achieved by waveform compression techniques. However, they are computationally more expensive than waveform compression techniques.

Transform compression techniques can be further classified into two main categories: block- and filter-transform compression techniques. The main difference between these two categories exist in the way of transforming images. In the former category, the image is first sub-divided into blocks, then, these blocks are transformed individually. Meanwhile, in the latter category, the image is transformed as a whole using a filtering scheme.

2.4.1 Block-transform Techniques

A general block-transform encoding scheme involves:

1. sub-dividing images into smaller $m \times m$ blocks,
2. applying a transformation on each block, and finally
3. quantizing and encoding the transformed coefficients.

The decoding scheme reverses the encoding scheme. Figure 2.2 shows a block diagram of the basic block-transform encoding/decoding schemes.

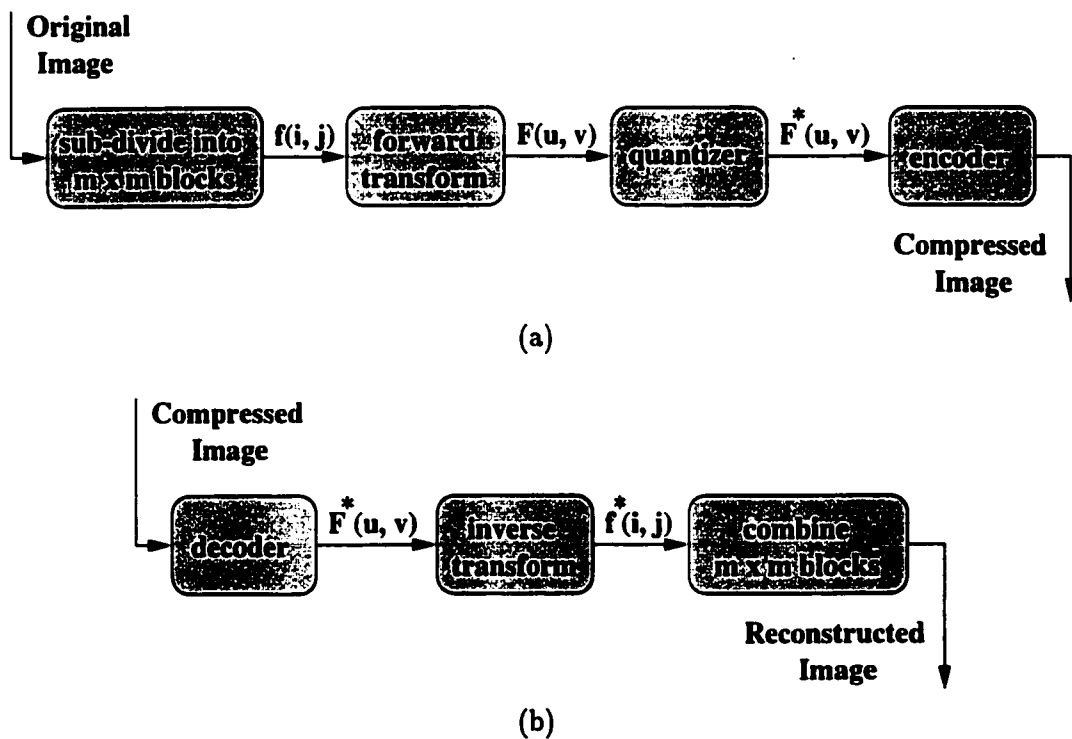


Figure 2.2: General block-transform encoding/decoding block diagrams. (a) encoder and (b) decoder.

A block transformation can be interpreted in several ways. One way is to consider it as a rotation of the block coordinate axes, where the various transformations merely differ in the nature of the rotation. Another interpretation is to view the transformation as a decomposition of the original block data into a set of transformed coefficients for a given set of basis functions, where the various transformations merely differ in their basis functions. Examples of block-transform compression techniques include the *Karhunen-Loève Transform (KLT)* [9] and the *Discrete Cosine Transform (DCT)* [10, 11].

The KLT is the optimal transformation, in an energy-packing sense [12]. Unfortunately, the KLT basis functions—which are the eigen vectors of the blocks covariance matrix—are image dependent. This means that they must be estimated, individually, for each image to be compressed. Furthermore, there is no fast algorithm to implement the KLT. Hence, the KLT utilization for image compression is limited.

The DCT, on the other hand, uses image independent cosine basis functions. In fact, the DCT is the closest image independent transformation to the KLT. Furthermore, there are so many fast algorithms to implement it [13, 14, 15, 16, 17, 18]. Because of these advantages, the DCT has become, by far, the most widely used block-transform for image compression.

2.4.2 Filter-transform Techniques

A general filter-transform encoding scheme involves:

1. performing a set of filtering operations on an image to divide it into n spectral sub-image components, and
2. quantizing and encoding each of these components to form the compressed image.

Figure 2.3 shows block diagrams of the basic filter-transform encoding/decoding schemes. The basis for these techniques is that, each sub-image component does not need to be

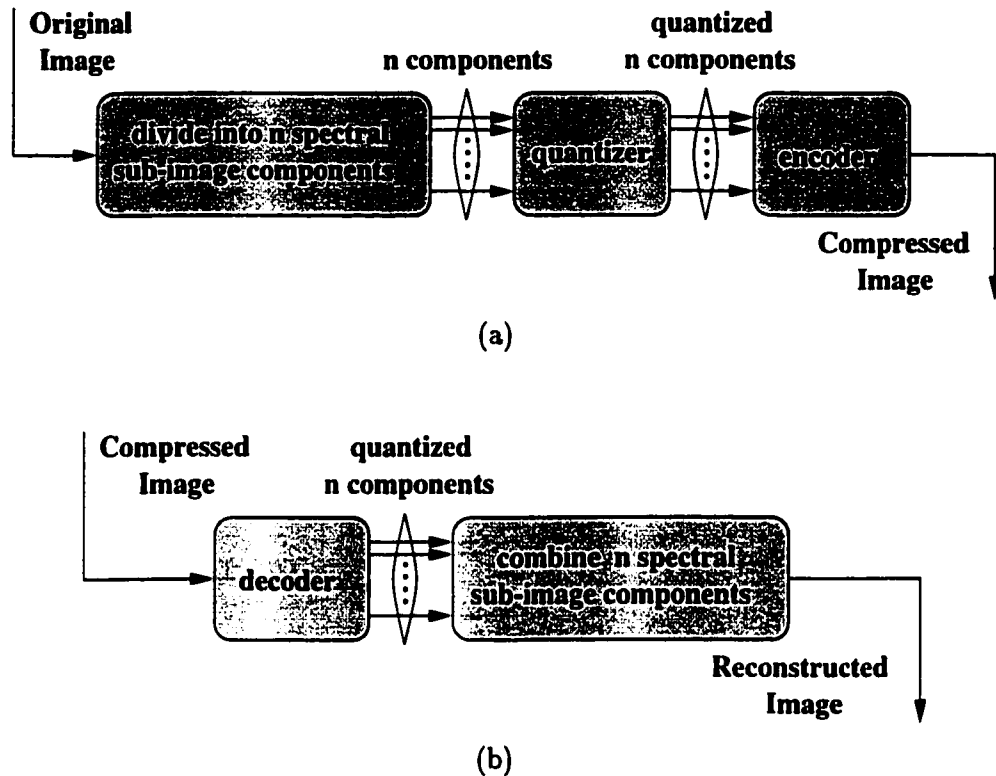


Figure 2.3: General filter-transform encoding/decoding block diagrams. (a) encoder and (b) decoder.

quantized with the same quantizer. For example, in some encoders, some sub-image components are often encoded with minimal, or no, quantization—in general, the higher the frequency component, the more coarsely the quantizer used is. Examples of filter-transform compression techniques include *multi-resolution pyramids* and *wavelet* encoding techniques.

In multi-resolution pyramid encoding schemes, the original image is successively low-pass filtered and sub-sampled to produce a pyramid which consists of reduced resolution versions of the original image. Each low-pass filtered image is, then, expanded (by up-sampling and filtering) to the dimension of the next level to provide a predicted image

for that level. By subtracting these predicted images from their corresponding images in the pyramid, a residual pyramid of images is created. Finally, all the residual images, as well as the lowest resolution image in the pyramid, are quantized and encoded. Since the residual images have smaller variances and are less correlated than the original image, they can be encoded efficiently. Hence, compression can be achieved. Depending on which low-pass filter is used, and how the low-pass filtered image is sub-sampled, many variations are possible, see for example [19, 20, 21, 22].

The wavelet encoding scheme can be interpreted as a multi-resolution image decomposition in a set of *independent* and *spatially oriented*—horizontal, vertical, and diagonals—frequency representations, where the band-widths of the different multi-resolution levels are on a logarithmic scale. The collection of the impulse responses of the wavelet band-pass filters is called a wavelet family. A wavelet family is a set of functions that are generated from a single function—which must satisfy certain conditions such as decaying to zero rapidly (i.e., small wave), and having a zero DC value—using dilation and translation operations [23]. In 1989, Mallat showed that the wavelet representation of a signal can be computed using a pyramid filter structure of a *Quadrature Mirror Filter* (QMF) filter pair [24, 25]. Note that, compression efficiency and overall performance depend not only on the choice of the wavelet filter bank, but also on the compression method that will be used on the sub-band filtered data—which is the case with other compression schemes. Examples of excellent wavelet compression implementations include Shapiro's [26] and Said-Pearlman's [27].

2.4.3 DCT-based Versus Wavelet-based Compressions

Without a doubt, both the DCT-based and the wavelet-based compression approaches provide a localized frequency representation of the original image. However, there are some differences between them. These differences lie, primarily, in:

1. the way they transform images, and
2. the manner in which the transformed data are organized and encoded.

At low bit-rate compression, while the DCT-based encoding schemes may suffer from the blocking artifact, the wavelet-based encoding schemes suffer from the ringing effect. Besides, from a computational cost point of view, the wavelet-based encoding schemes are more expensive than the DCT-based encoding schemes.

2.5 Model-based Compression Techniques

In model-based compression techniques, also called knowledge-based compression techniques, an image, or some portion of an image, is modeled and the model parameters are used for image synthesis. At the encoding side, the model parameters are estimated by analyzing the image. Meanwhile, at the decoder side, the image is synthesized from the estimated and quantized model parameters. In other words, a model-based compression technique can be viewed as an analysis/synthesis system. Figure 2.4 shows block diagrams of the basic model-based encoding/decoding schemes.

Model-based compression techniques have the prospect of further reducing the bit rate, compared to waveform and transform compression techniques. However, estimating model parameters and synthesizing an image from them is likely to be, computationally, very expensive. Examples of these techniques include *contour-* and *fractal-based* compression techniques.

Contour-based compression schemes are based on the idea that a small number of well-placed contours can retain a great deal of information pertaining to the objects they represent [28]. A contour-based compression technique encodes an image by a set of locations and intensities of the extracted image contours. Using this set of locations and intensities, an approximate version of the original image is generated [29].

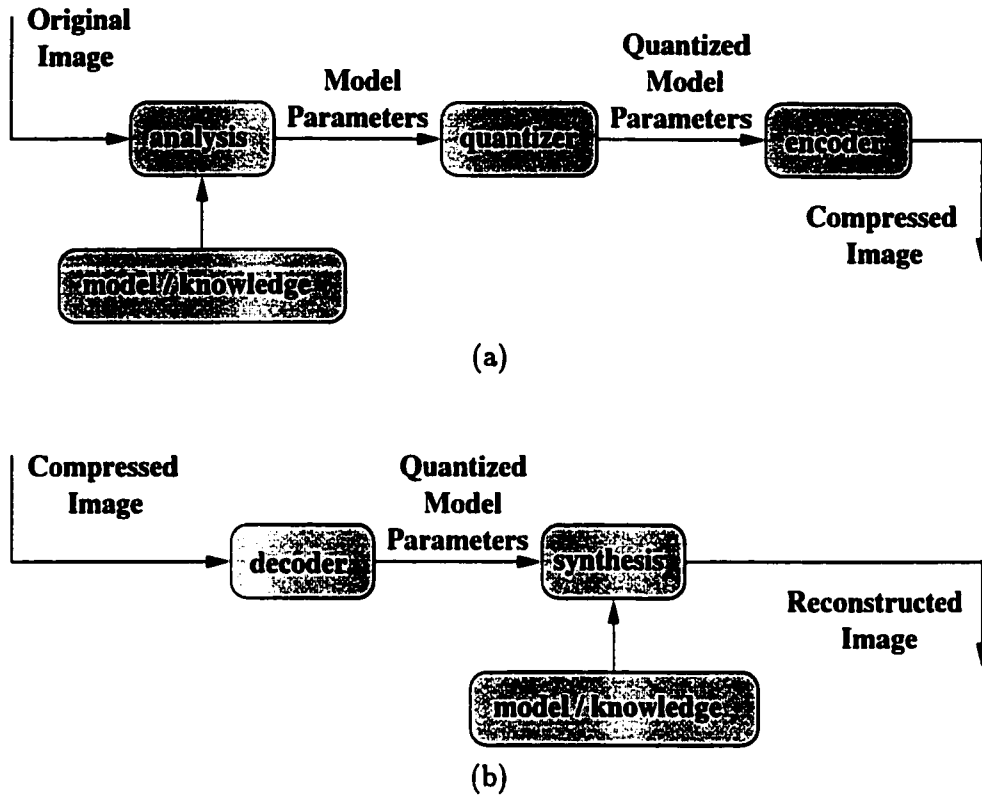


Figure 2.4: General model-based encoding/decoding block diagrams. (a) encoder and (b) decoder.

Fractal compression schemes achieve compression by exploiting the self-similarities between different image regions. Basically, a fractal-based compression technique encodes a given image as a set of mathematical functions. In fact, it consists of four basic steps:

1. partitioning a given image into blocks, called range-blocks;
2. defining an equal number of other larger image blocks which may overlap, called domain-blocks;
3. identifying a corresponding domain-block for each range-block; and
4. for each domain-range block pair, identifying an intensity transform that maps the

domain-block intensity values onto the intensity values of its corresponding range-block.

A decoding process may start from any arbitrary image. Then, the above mentioned intensity transforms are iteratively applied. This yields a sequence of images that converges to an approximation of the original image [30, 31].

2.6 Hybrid Compression Techniques

Hybrid compression techniques refer to compression schemes that combine together more than one compression approach. Examples of hybrid techniques include block-transforms/DPCM compression [32], DCT/fractal compression [33], DCT/contour-based compression [34], and wavelet/contour-based compression [34].

Generally, hybrid techniques may achieve better performance than any of the individual compression schemes being combined. At the same time, their complexity is not much higher than the complexity of any of the individual compression schemes being combined.

2.7 Quantization

Generally, output data values from a quantization process have less resolution than its input data values. This, in turn, is translated into a reduction in the psycho-visual redundancy. Quantization of data can be performed in either one of two ways:

1. individually (scalar quantization), or
2. jointly (vector quantization).

2.7.1 Scalar Quantization

In scalar quantization [35], a data value is quantized to a particular quantization level out of L possible levels. There are two main issues to be addressed with scalar quantization:

1. what is a best number of quantization levels to use, and
2. where to locate these levels.

The number of the quantization levels represents a tradeoff. When more quantization levels are used, more bits are assigned to each quantized value, hence, less compression ratios are achieved. At the same time, when more quantization levels are used, less distortion is achieved, and vice versa.

The location of the quantization levels may be selected to divide the space uniformly (uniform quantization) or non-uniformly (non-uniform quantization). Uniform quantization is a straightforward process. However, it may not be optimal. In non-uniform quantization, the probability density function of the data values to be quantized is exploited to minimize the error between the quantized and un-quantized data values [36, 37].

2.7.2 Vector Quantization

Prior to 1980, almost all image compression techniques employed scalar quantization techniques to quantize image data. In 1980, and based on Shannon's theory [38], the idea of the scalar quantization was generalized to the notion of *Vector Quantization* (VQ) [39]. Since then, much attention has been directed to VQ [40, 41].

The basic idea of VQ is to divide the image into small blocks (vectors), then, match each block to one of a set of predetermined prototype blocks so that the distortion between them is minimized. The set of all predetermined prototype blocks is usually called a *codebook*, where each prototype block is called a *codeword*. The codebook is generated a

priori from a collection of representative images. Compression is achieved by transmitting the index of the block's best matched codeword. Decoding in the VQ approach is much simpler than encoding. The decoder, which has an identical codebook as that of the encoder, uses the codeword index and a simple table look-up operation to reconstruct the image.

As with scalar quantization, there are two main issues to be addressed with VQ:

1. how to design a good codebook that is representative of all the possible occurrences of pixel value combinations in a block, and
2. how to find a best match, efficiently, in the codebook during the encoding process.

Linde *et al* [39] and Equitz [42] have suggested various clustering techniques to design good codebooks. Partitioning the search space and tree search algorithms have been proposed to speed up the code vector search process [43, 44].

The VQ can be considered as a waveform compression technique. However, it may be used in a hybrid scheme in conjunction with any other compression technique.

2.8 Previous Work Most Relevant to This Thesis

The compression technique proposed in this thesis relies on classifying image-blocks into different perceptual classes. This process requires segmenting an image into non-overlapping homogeneous regions. Many segmentation techniques have been proposed in the literature [45, 46, 47, 48]. One of the most simple, yet efficient, segmentation techniques is *quad-tree* segmentation [49].

In [50, 51], quad-trees are used to segment a given image into smooth-blocks of variable sizes and non-smooth blocks of a fixed smaller size. In [50], three-level 4×4 to 16×16 bottom-up quad-trees were used. The DCT followed by a fixed step-size quantizer was

used to encode each of these blocks but with coarser quantization applied to smooth-blocks (large-blocks). In [51], two-level 4×4 to 2×2 top-down quad-trees were used. The block average was exploited to encode smooth-blocks while *Absolute Moment Block Truncation Coding* (AM-BTC) [4] and AM-BTC with lookup tables (based on block activities) were used to encode non-smooth-blocks. The results reported in both of these papers showed good-quality reconstructed images with compression ratios of less than 12:1, i.e., at a bit rate of more than 0.66 *bits per pixel* (bpp). The reason for these high bit rates might be either:

1. using the same fixed step-size quantizer for all DCT coefficients regardless of their locations within the transformed-block, or
2. using the BTC encoding algorithm which has a limited compression capability.

In [52], three-level 16×16 to 4×4 top-down quad-trees were used. The non-smooth-blocks were further classified into 8 classes according to the orientation of their edges. Then, VQ in the DCT domain was used to encode the AC-coefficients of each class, separately, while the DC coefficients were DPCM encoded. Although both textural- and edge- (non-smooth-) blocks represent high frequency data, textural-blocks usually convey a low amount of information to the viewer and should, therefore, be assigned less bits. In [53], this idea was exploited. In the first stage, an initial four-level 32×32 to 4×4 top-down quad-tree segmentation of the input image was performed to locate regions which have homogeneous mean values. Then, the mean of each block was encoded to form the mean image which was, then, smoothed to generate an interpolated image. The interpolated image was subtracted from the input image to obtain a residual image. The residual image was segmented by another four-level 32×32 to 4×4 top-down quad-trees and their leafs were classified into three classes (smooth, textural, and edge classes). Finally, the classified residual-blocks were vector-quantized at different bit rates according to their

levels of information (i.e., according to the class to which they belong). The reported results show that these two techniques achieved good-quality reconstructed images with compression ratios between 10:1 and 32:1, i.e., at bit rates between 0.8 bpp and 0.25 bpp, depending on the nature of the original image. The main disadvantage of these two techniques is the long time needed to execute them. For example, the latter technique typically takes about 10 minutes of CPU time on a SUN 3/260 computer with a floating point accelerator to encode a 512×512 image. Most of this time is devoted to searching the codebooks.

Another simple, yet efficient, segmentation technique, which segments a given image into variable size smooth regions, is binary-tree segmentation. In binary-trees, a straight line is used to binary-divide a given image domain into two sub-image domains. The orientation and the location of this line are chosen so as to minimize the reconstruction error. This procedure is repeated until the reconstruction error goes under a certain threshold.

In [54, 55], instead of using quad-trees, the binary-trees have been used, and each leaf region was represented by a first order polynomial function. The polynomial function coefficients were determined using the least-square approximation method. The main difference between these two techniques is that the former restricts the dividing line orientations to one of four possible orientations, namely, 0° , 45° , 90° , and 135° . Hence, the latter technique provides a more general framework, yet requires higher computational complexity, than the former technique. The reported results show that these two techniques obtained a fair-quality reconstructed images with compression ratios between 80:1 and 160:1, i.e., at bit rates between 0.1 bpp and 0.05 bpp.

Other segmentation techniques, such as region growing [56, 57], perform a more precise isolation on statistically-homogeneous regions. However, both the number of regions and their shapes are determined solely by the contents of the examined image. This implies

that a very large number of bits may be needed to represent the shape and the location of each region.

Another approach for compressing images [34], is to decompose images, instead of segmenting them, into three components (namely, strong edge, smooth, and textural components). The model used is the curvature energy minimization model (developed in [3]). The intensity and geometric information of strong edge contours in the strong edge component were encoded separately using the chain encoding technique [58, 59]. Two alternatives for encoding the smooth and textural components were suggested, namely, entropy-encoded fixed block-size adaptive DCT encoding and entropy-encoded subband encoding. It has been reported that this technique has better performance over the JPEG continuous-tone image compression standard [60]. However, its main disadvantage is that it is extremely time consuming. The main bulk of complexity resides in the three-component decomposition. For example, it typically takes about 23 minutes of CPU time on a SUN SPARC-station 1 just to decompose a 256×256 image. Table 2.1 presents performance results of this technique as well as several other encoding techniques mentioned in this section.

Table 2.1: Performance results of various encoding techniques for the 512×512 Lena image.

Method	Ref.	CR	RMSE
<i>Segmentation</i> : tri-level quad-trees {4, 8, 16}, <i>Encoding</i> : DCT.	[50]	6.62 : 1	5.50
<i>Segmentation</i> : di-level quad-trees {4, 2}, <i>Encoding</i> : average/AM-BTC.	[51]	5.88 : 1 6.40 : 1 10.53 : 1	5.47 5.70 9.10
<i>Segmentation</i> : tri-level quad-trees {16, 8, 4}, block classification, <i>Encoding</i> : DPCM/VQ in DCT domain.	[52]	22.28 : 1 23.95 : 1 25.89 : 1 28.17 : 1	5.87 6.11 6.37 6.68
<i>Segmentation</i> : quadruple-level quad-trees {32, 16, 8, 4}, block classification, <i>Encoding</i> : VQ.	[53]	22.04 : 1 28.88 : 1	6.90 7.88
<i>Segmentation</i> : binary-trees, <i>Encoding</i> : 1 st order polynomial fitting.	[55]	80.00 : 1 114.00 : 1	13.23 14.84
image decomposition, <i>Encoding</i> : chain encoding/DCT/subband.	[34]	10.94 : 1 16.43 : 1 32.39 : 1 64.00 : 1	3.20 3.94 5.62 8.52

Chapter 3

ABC-SC: A New Adaptive Compression Technique

One approach to develop efficient compression technique is to:

1. know the different types of information that are carried within images,
2. understand the main characteristics of each type, and
3. handle each of them with an appropriate resolution according to its importance and impact on the HVS.

In this chapter, the new compression technique, which was developed by following the above three steps, is described and a way to identify its parameters is suggested.

3.1 An Overview of the Proposed Compression Technique

The proposed compression technique is called *Adaptive Block Compression method based on Segmentation and Classification* [61, 62], hence, the naming abbreviation *ABC-SC*. A

general block diagram of ABC-SC is shown in Figure 3.1, and detailed encoding/decoding block diagrams are shown in Figures 3.2 and 3.3, respectively.

In the ABC-SC technique, a given input image is first subdivided into super-blocks of 32×32 pixels. Then, for each of these super-blocks, a quad-tree with minimum sub-blocks of 8×8 pixels is built. A block is declared to be a leaf block if its homogeneity measure satisfies a certain threshold or if the bottom of the tree is reached, otherwise the tree is traversed down further. The leaf-blocks are then classified into three perceptual classes according to the amount of information each block conveys to the viewer. These perceptual classes are: a smooth class, a textural class, and an edge class. As a result of this classification, three segments of the image are produced (one per each perceptual class). Each of these segments is then encoded separately. For the smooth image segment, only the quantized average of each block is encoded using a new *Adaptive Differential Pulse Code Modulation* (ADPCM) technique. In ADPCM, different linear prediction rules, including 2^{nd} and 3^{rd} order two-dimensional prediction rules, are utilized to predict the current block average—only one rule per prediction is applied. The choice of a prediction rule is based on the differences between the neighboring encoded block averages. For the textural and the edge image segments, blocks are DCT-transformed. Then, the transformed coefficients are quantized, where the quantization matrix for each of the two classes is optimized based on the amount of information conveyed in each region—coarser quantization is applied to the textural-blocks than to the edge-blocks. This is due to the fact that the textural-blocks convey less information to the viewer than edge-blocks. Since the averages of the adjacent blocks are strongly correlated, the DC-coefficients are ADPCM encoded. Meanwhile, the AC-coefficients of each block are *zigzag*-ordered and then *Run-Length* (RL) encoded [63]. Finally, each group of encoded data, the quad-tree structures (also called the *side information*); the encoded average of each smooth block with the encoded DC-coefficients; and the encoded AC-coefficients, are arithmetically and

separately encoded to produce the compressed image.

At the receiver's side, each group of data is arithmetically decoded. Then, using the decoded side information, the quad-trees are reconstructed. Next, the quantized coefficients are multiplied by the quantization step-size and decoded, to form a reconstructed image segment. For the smooth segment, the *Inverse ADPCM* (I-ADPCM) is applied. Meanwhile, the run-length decoding, the inverse zigzag-ordering, the *Inverse DCT* (I-DCT), as well as the inverse ADPCM are applied to the edge and textural image segments. Finally, all of these reconstructed image segments are gathered to form the reconstructed image.

At high compression levels, smooth-block boundaries may become visible and viewers might suffer from the annoying intra-blocking effect. The reason for having this blocky appearance is that smooth-blocks were represented only by their quantized averages, although their pixel intensity values might not be close enough to this quantized average. Since it is always desired to reach high compression ratios and still have a good reconstruction quality, a post-processing filtering scheme should be applied to reduce this annoying intra-blocking-effect and return the lost grey-levels continuity back to the reconstructed image.

3.1.1 Quad-tree Representation

Quad-tree is an efficient data structure that provides an effective compromise between the accuracy, with which the region boundaries are determined, and the number of bits required to specify the segmentation information. Quad-tree starts by dividing a given image into blocks of equal size. Then, the homogeneity of each of these blocks is measured (Section 3.1.2). If the homogeneity of a given block does not exceed a certain threshold (i.e., *non-leaf-block*), it is further subdivided into four smaller quadrant sub-blocks. This dividing process goes on until the homogeneity criterion is satisfied or a certain minimum

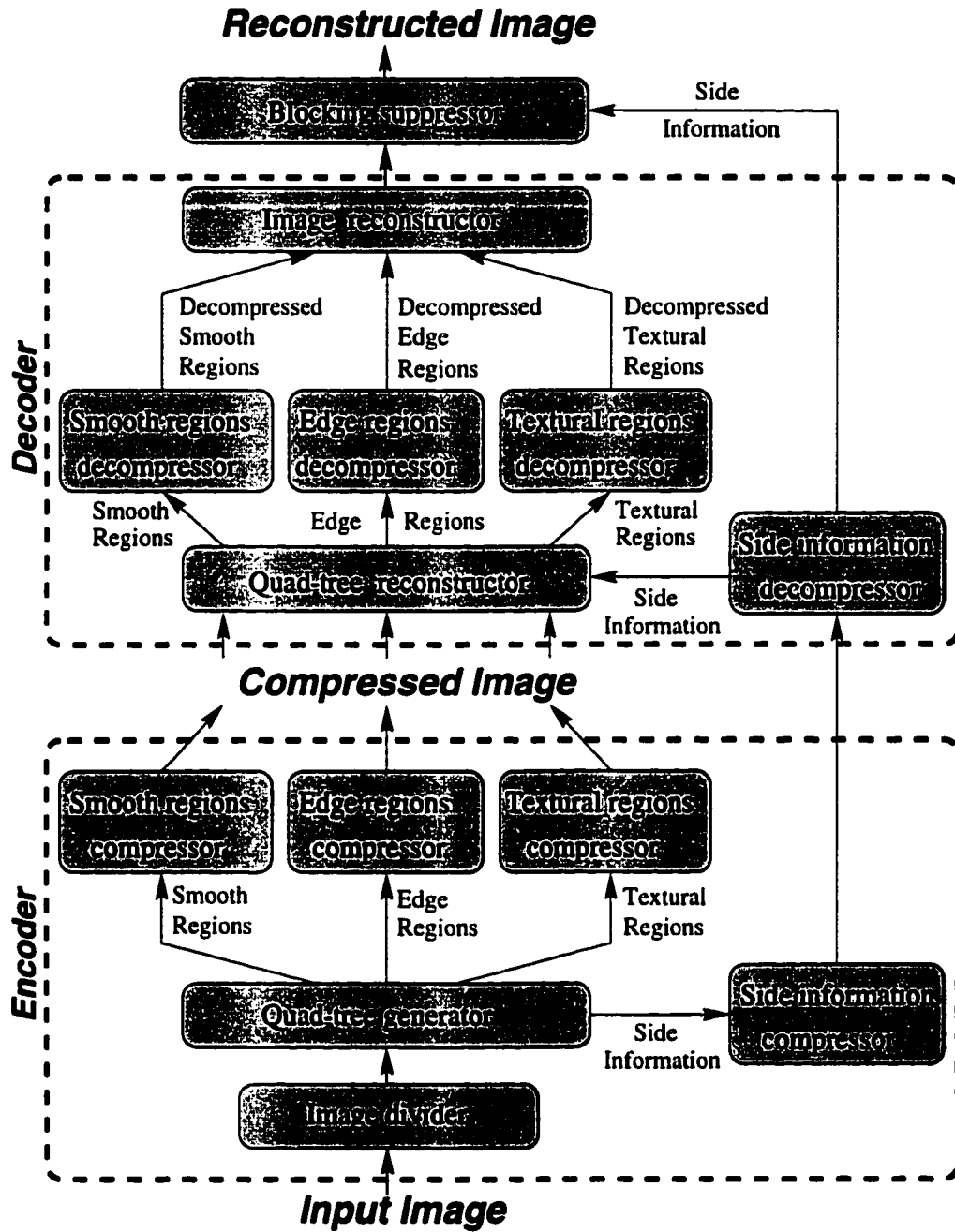


Figure 3.1: Block diagram of ABC-SC.

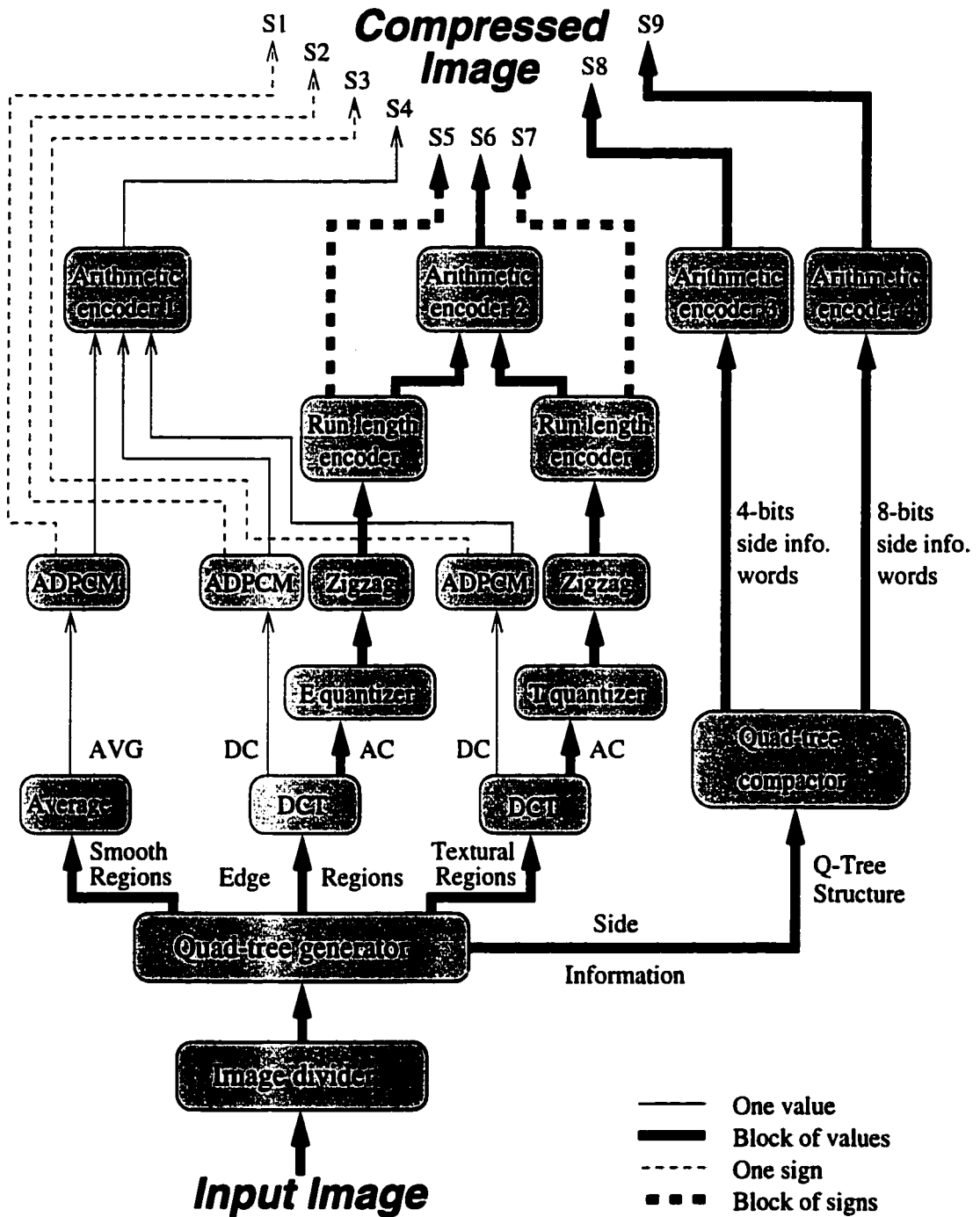


Figure 3.2: Detailed encoder block diagram.

block-size is reached (i.e., *leaf-blocks*). Figure 3.4 shows an example of segmenting an image-block and its corresponding quad-tree. Quad-trees require only a small overhead bit rate (side information). This is due to the restriction put on the shapes and the sizes of their leafs (to be selected from a predetermined set of options). In fact, the quad-tree structure is represented by one bit of side information per node to indicate whether that node is a non-leaf or a smooth leaf node. However, if the node is corresponding to an 8×8 block, two bits are needed to indicate whether the node is a smooth, an edge, or a textural leaf node. The side information corresponding to each four neighboring nodes are then combined together to form a 4-bit or an 8-bit side information word. This word is arithmetically and separately encoded.

3.1.2 Homogeneity Criteria

Smoothness Criterion

Many smoothness criteria have been proposed in previous studies [64, 65, 66, 67]. As a simple and adequate measure, the block variance is selected to be the smoothness criterion in this work. To identify an $m \times m$ block to be a smooth-block, its variance σ_m^2 must be less than or equal to a certain threshold T_m , i.e.,

$$\sigma_m^2 \leq T_m. \quad (3.1)$$

Since ABC-SC is not sensitive to segmentation errors, it is believed that the variance criterion is sufficient for the purpose of this research. This is the case since even if a segmentation error is made, it would only result in a small change in the bit rate.

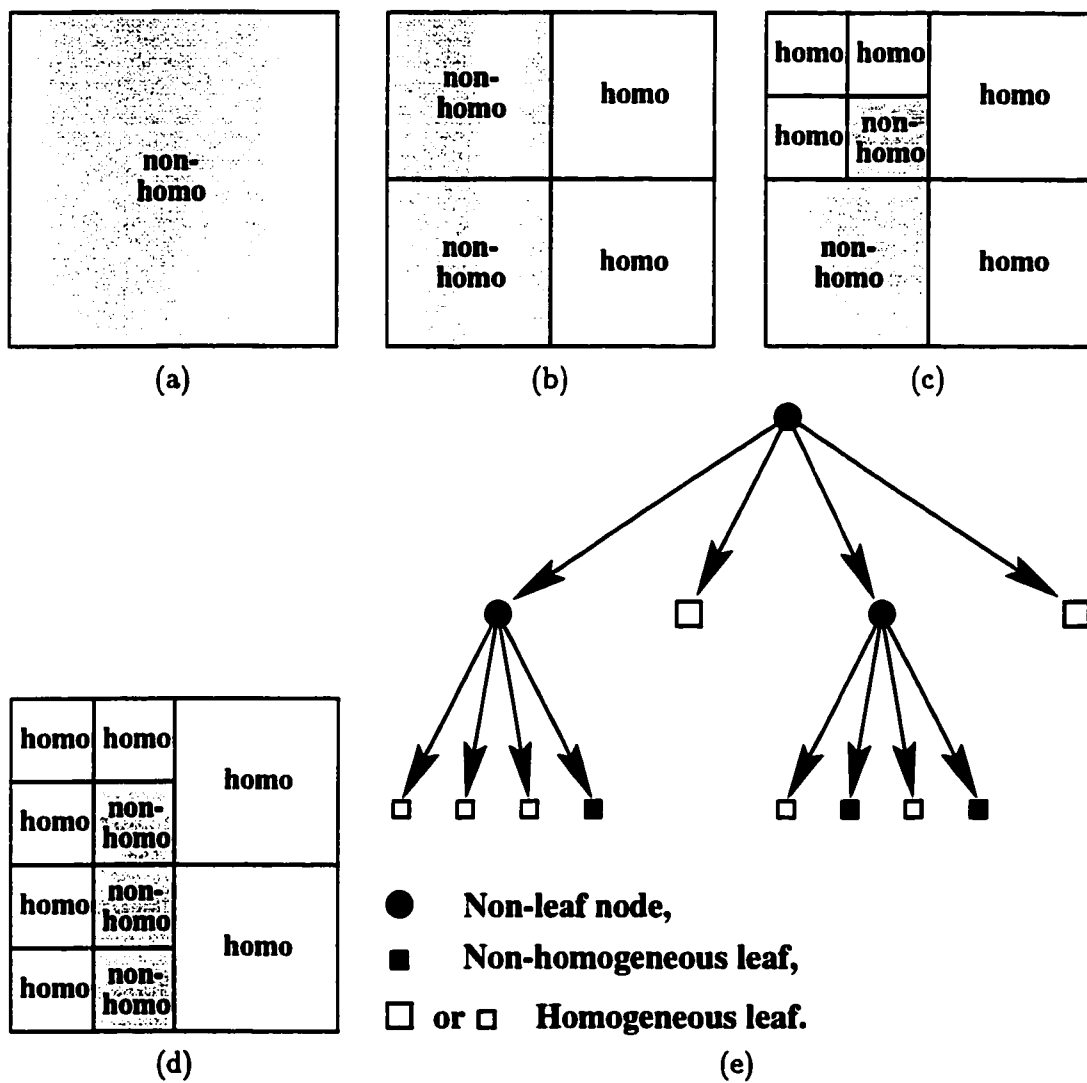


Figure 3.4: A Quad-tree segmentation example. (a)–(d) the process of segmenting a block and (e) the corresponding quad-tree structure.

Textural Criterion

It was found [53] that a block belonging to the textural class must have high level of details and exceed a minimum size. Also, the block's contents together with the contents of its surrounding blocks must be homogeneous. It was shown, experimentally, that 8×8 may be an adequate size for textural-blocks. Therefore, the candidates for the textural class are those 8×8 blocks which do not satisfy the smoothness criterion. To test the textural surrounding of a given candidate block, a 16×16 augmented block which has the candidate block at its center, is considered, as shown in Figure 3.5. This augmented block is then broken up into four non-overlapped sub-blocks of size 8×8 . Next, the variances of these four blocks (σ_1^2 , σ_2^2 , σ_3^2 , and σ_4^2), as well as the variance of the candidate block (σ_5^2), are calculated. The candidate block is declared as a textural-block if the average of these 5 variances is greater than or equal to a threshold $T_{average}$, and the relative absolute deviation between each of them and their average is less than or equal to a threshold $T_{deviation}$, i.e.,

$$\sigma_{avg}^2 \geq T_{average}, \quad (3.2)$$

and

$$\frac{|\sigma_i^2 - \sigma_{avg}^2|}{\sigma_{avg}^2} \leq T_{deviation}, \quad i = 1, 2, \dots, 5, \quad (3.3)$$

where

$$\sigma_{avg}^2 = \frac{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_4^2 + \sigma_5^2}{5}. \quad (3.4)$$

In other words, these conditions ensure that blocks classified into the textural class are located in a region exhibiting a homogeneous high value variance.

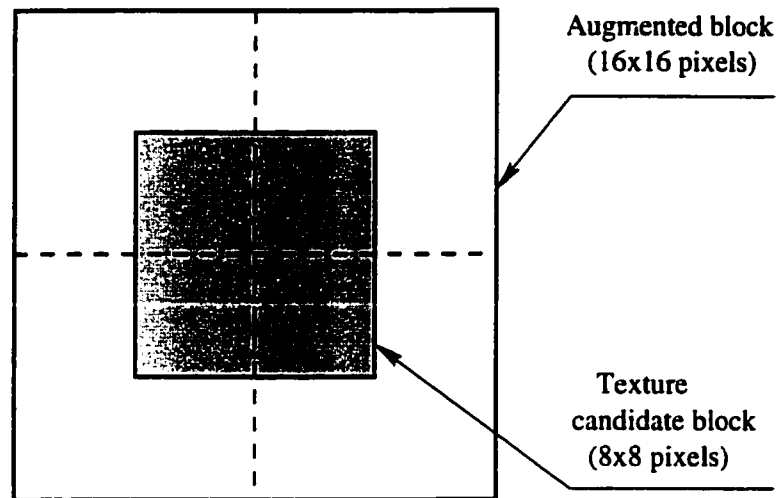


Figure 3.5: A textural candidate block and its augmented block.

Edge Criterion

If a block is neither classified as a smooth-block nor a textural-block and the bottom of the quad-tree is reached, then the block is simply declared an edge-block.

3.1.3 Adaptive Differential Pulse Code Modulator

In ADPCM, as shown in Figure 3.6, an attempt is made to predict the block average to be encoded. Note that the DC-coefficient is exactly m times the block average. The prediction is made using the already encoded block averages. Only the prediction error, i.e., the difference between the predicted and the current block average values, is quantized. The absolute quantized prediction error is, then, arithmetically encoded while the sign of each non-zero quantized prediction error is encoded separately in a single bit.

During the adaptive prediction process, different linear prediction rules are utilized to predict the current block average (only one rule per prediction is applied). These rules include 2^{nd} order two-dimensional prediction rules, (e.g., $\frac{1}{4}A + \frac{3}{4}C$, $\frac{3}{4}A + \frac{1}{4}C$, $1\frac{1}{4}A - \frac{1}{4}C$, and $-\frac{1}{4}A + 1\frac{1}{4}C$), and 3^{rd} order two-dimensional prediction rules (e.g., $A - \frac{1}{4}B + \frac{1}{4}C$,

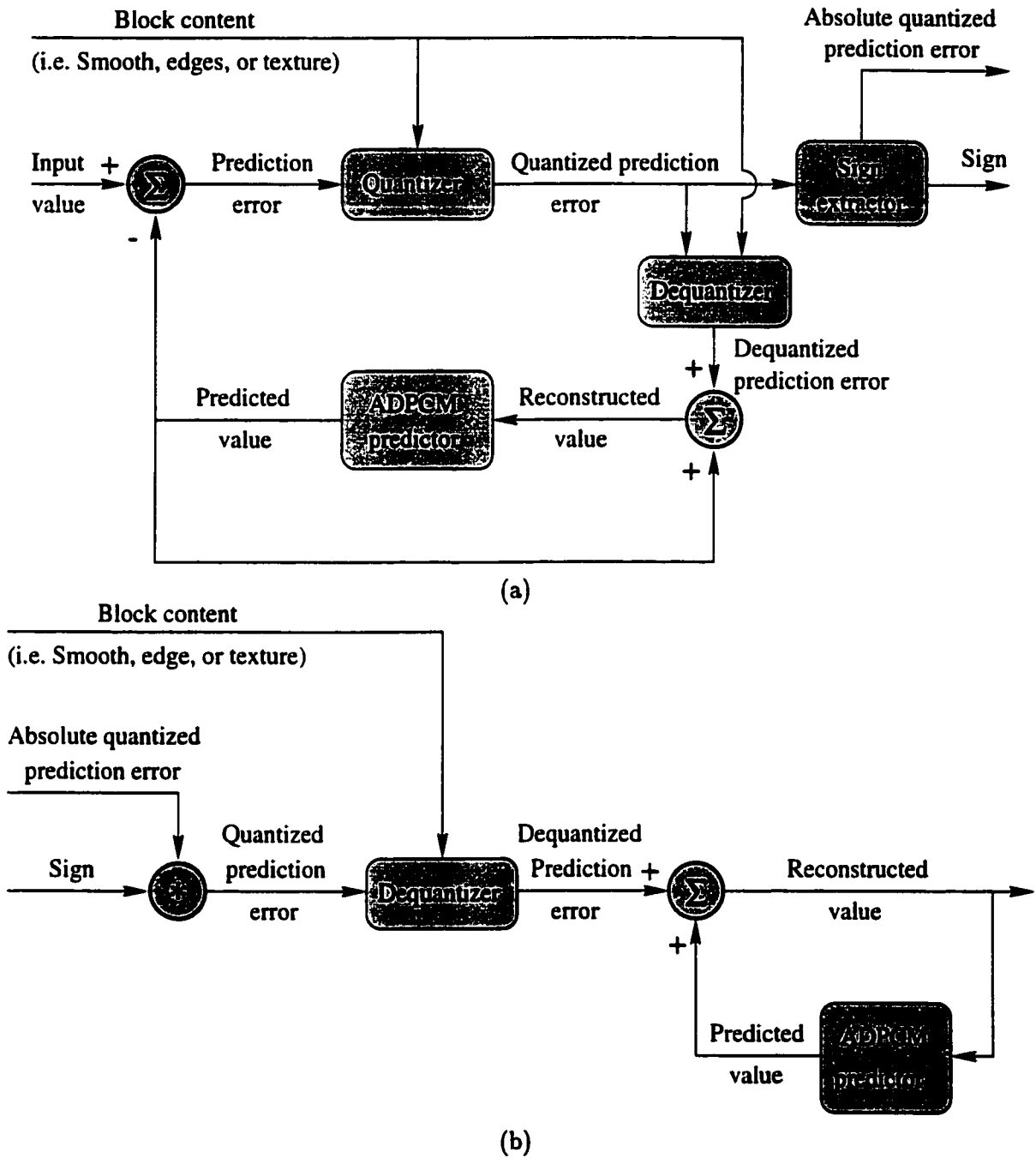


Figure 3.6: The ADPCM block diagram. (a) encoder and (b) decoder.

$\frac{1}{4}A - \frac{1}{4}B + C$, $A - \frac{1}{2}B + \frac{1}{2}C$, $\frac{1}{2}A - \frac{1}{2}B + C$, and $\frac{3}{4}A - \frac{1}{2}B + \frac{3}{4}C$), where A , B , and C are the neighboring block averages.

The choice of a prediction rule is based on encoded block averages local statistics, namely, the differences between the neighboring encoded block averages. Table 3.1 shows all the different cases of A , B , and C with all the possible relative distances between them, whereas Table 3.2 shows the corresponding prediction rule, the prediction range, and the expected shape of this area for each case. Note that the term *small* or *large* in Table 3.1 means that the difference between the neighboring encoded block averages is less or greater, respectively, than a certain value v . In this work, the value of v is empirically determined so that the prediction rule usage is partitioned as evenly as possible (Section 3.2.2).

3.1.4 Discrete Cosine Transform

The DCT [10, 11] is a block-transform encoding technique which transforms a given image to the frequency domain. During this transformation, the DCT attempts to reduce the correlation that exists among image pixel intensity values. Moreover, the DCT has an excellent *energy compaction* property, which means that a large amount of the energy is concentrated in a small set of transformed coefficients. Hence, by concentrating on this set of coefficients, high compression can be achieved, without seriously affecting the reconstructed quality.

For an $m \times m$ image-block, the DCT and the I-DCT coefficients can be calculated according to (3.5) and (3.6), respectively.

$$F(u, v) = \frac{C(u)C(v)}{m} \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} f(i, j) \cos\left(\frac{(2i+1)\pi u}{2m}\right) \cos\left(\frac{(2j+1)\pi v}{2m}\right) \quad (3.5)$$

Table 3.1: All possible relative pixel values of A, B, and C.

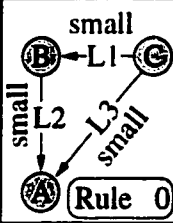
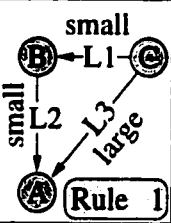
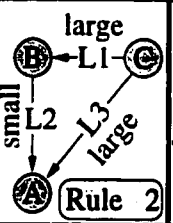
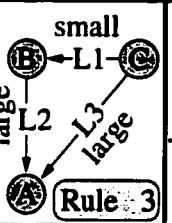
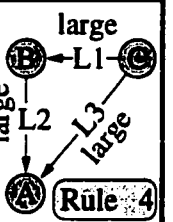
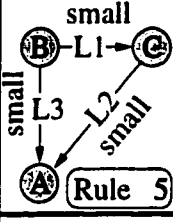
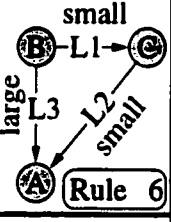
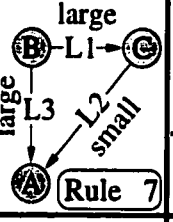
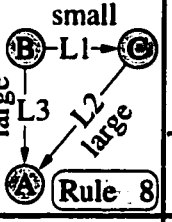
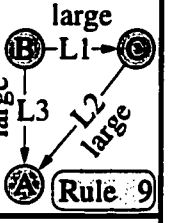
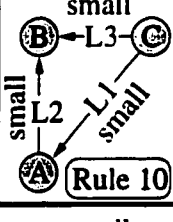
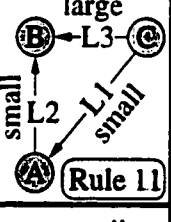
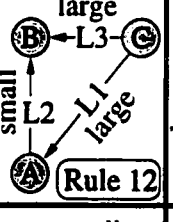
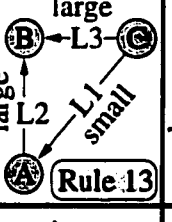
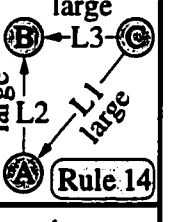
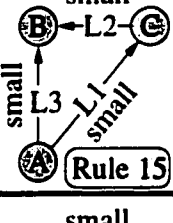
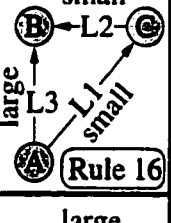
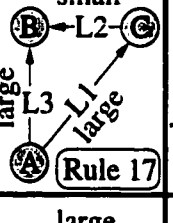
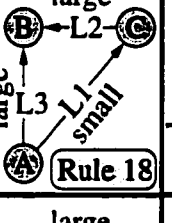
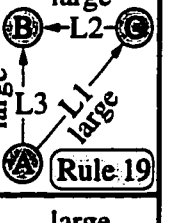
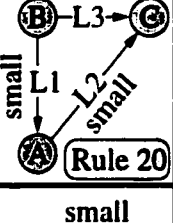
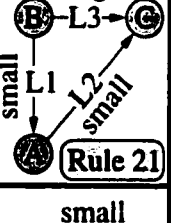
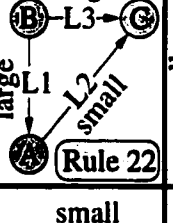
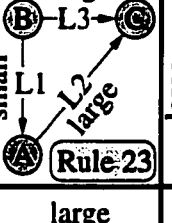
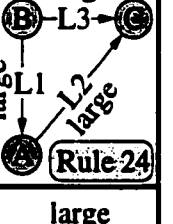
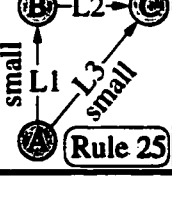
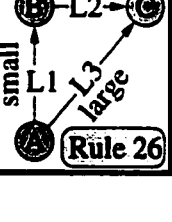
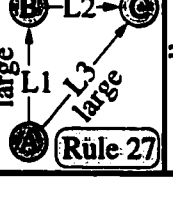
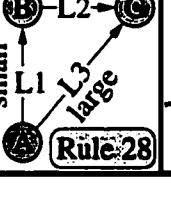
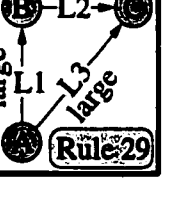
	L1 is small & L2 is small		L1 is large & L2 is small	L1 is small & L2 is large	L1 is large & L2 is large
	L3 is small	L3 is large			
$A \geq B \geq C$	 Rule 0	 Rule 1	 Rule 2	 Rule 3	 Rule 4
$A \geq C > B$	 Rule 5	 Rule 6	 Rule 7	 Rule 8	 Rule 9
$B > A \geq C$	 Rule 10	 Rule 11	 Rule 12	 Rule 13	 Rule 14
$B \geq C > A$	 Rule 15	 Rule 16	 Rule 17	 Rule 18	 Rule 19
$C > A \geq B$	 Rule 20	 Rule 21	 Rule 22	 Rule 23	 Rule 24
$C > B > A$	 Rule 25	 Rule 26	 Rule 27	 Rule 28	 Rule 29

Table 3.2: The ADPCM prediction rules.

	Rule no.	Prediction rule	Prediction range	Expected shape
$A \geq B \geq C$	Rule 0	$\frac{3}{4}A - \frac{1}{2}B + \frac{3}{4}C$	$\in [C, A]$	flat
	Rule 1	$\frac{3}{4}A - \frac{1}{2}B + \frac{3}{4}C$	$\in (C, A)$	flat
	Rule 2	$\frac{1}{2}A - \frac{1}{2}B + C$	$\in [C, B)$	edge
	Rule 3	$A - \frac{1}{2}B + \frac{1}{2}C$	$\in (B, A]$	edge
	Rule 4	$\frac{3}{4}A - \frac{1}{2}B + \frac{3}{4}C$	$\in (C, A)$	strong edge/texture
$A \geq C > B$	Rule 5	$\frac{3}{4}A + \frac{1}{4}C$	$\in [C, A]$	flat
	Rule 6	$A - \frac{1}{2}B + \frac{1}{2}C$	$> A$	edge
	Rule 7	$1\frac{1}{4}A - \frac{1}{4}C$	$\geq A$	edge/texture
	Rule 8	$A - \frac{1}{2}B + \frac{1}{2}C$	$> A$	edge
	Rule 9	$A - \frac{1}{4}B + \frac{1}{4}C$	$> A$	strong edge/texture
$B > A \geq C$	Rule 10	$\frac{1}{4}A + \frac{3}{4}C$	$\in [C, A]$	flat
	Rule 11	$\frac{1}{2}A - \frac{1}{2}B + C$	$< C$	edge
	Rule 12	$\frac{1}{2}A - \frac{1}{2}B + C$	$< C$	edge
	Rule 13	$-\frac{1}{4}A + 1\frac{1}{4}C$	$\leq C$	edge/texture
	Rule 14	$\frac{1}{4}A - \frac{1}{4}B + C$	$< C$	strong edge/texture

Table 3.2: The ADPCM prediction rules, (continued).

	Rule no.	Prediction rule	Prediction range	Expected shape
$B \geq C > A$	Rule 15	$\frac{3}{4}A + \frac{1}{4}C$	$\in (A, C)$	flat
	Rule 16	$A - \frac{1}{2}B + \frac{1}{2}C$	$< A$	edge
	Rule 17	$A - \frac{1}{2}B + \frac{1}{2}C$	$\leq A$	edge
	Rule 18	$1\frac{1}{4}A - \frac{1}{4}C$	$< A$	edge/texture
	Rule 19	$A - \frac{1}{4}B + \frac{1}{4}C$	$< A$	strong edge/texture
$C > A \geq B$	Rule 20	$\frac{1}{4}A + \frac{3}{4}C$	$\in (A, C)$	flat
	Rule 21	$\frac{1}{2}A - \frac{1}{2}B + C$	$> C$	edge
	Rule 22	$-\frac{1}{4}A + 1\frac{1}{4}C$	$> C$	edge/texture
	Rule 23	$\frac{1}{2}A - \frac{1}{2}B + C$	$\geq C$	edge
	Rule 24	$\frac{1}{4}A - \frac{1}{4}B + C$	$> C$	strong edge/texture
$C > B > A$	Rule 25	$\frac{3}{4}A - \frac{1}{2}B + \frac{3}{4}C$	$\in (A, C)$	flat
	Rule 26	$\frac{3}{4}A - \frac{1}{2}B + \frac{3}{4}C$	$\in (A, C)$	flat
	Rule 27	$A - \frac{1}{2}B + \frac{1}{2}C$	$\in (A, B)$	edge
	Rule 28	$\frac{1}{2}A - \frac{1}{2}B + C$	$\in (B, C)$	edge
	Rule 29	$\frac{3}{4}A - \frac{1}{2}B + \frac{3}{4}C$	$\in (A, C)$	strong edge/texture

$$f(i, j) = \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} \frac{C(u)C(v)}{m} F(u, v) \cos\left(\frac{(2i+1)\pi u}{2m}\right) \cos\left(\frac{(2j+1)\pi v}{2m}\right) \quad (3.6)$$

where

$$C(x) = \begin{cases} 1 & \text{for } x = 0, \\ \sqrt{2} & \text{for } x = 1, 2, \dots, m-1, \end{cases}$$

$f(i, j)$: An input image pixel intensity value,

$F(u, v)$: A transformed coefficient value, and

m : The block width.

Conventionally, the DCT coefficient with a zero frequency in both dimensions—i.e., $F(0, 0)$ —, is called the DC-coefficient, since it is related to the block average (in fact, it is exactly m times the block average). The remaining DCT coefficients are called AC-coefficients. The AC-coefficients generally diminish as the frequency variables—i.e., the transformed-block coordinates u and v —increase. This property is exploited while encoding the AC-coefficients.

3.1.5 Quantizers

ADPCM Quantizer

Typically, ADPCM prediction errors have a greatly reduced variance, compared to the variance of the original average values. However, more bit rate reduction can be achieved by quantizing these prediction errors prior to encoding them. For this purpose, a simple scalar quantizer is designed. In this quantizer, each prediction error value is quantized using a simple scalar value defined by

$$\text{ADPCM-quantization-step} = \left\lfloor \frac{256}{K_{ADPCM}^{class}(QF)} \right\rfloor, \quad (3.7)$$

where $K_{ADPCM}^{class}(QF)$ is a class dependent positive non-zero integer function of the *Quality-Factor* QF , and *class* can be any of the three perceptual classes. The K_{ADPCM}^{class} is used as a scaling factor to compromise between the compression ratio and the quality of the reconstructed images. The range of K_{ADPCM}^{class} is

$$2 \leq K_{ADPCM}^{class} \leq 256. \quad (3.8)$$

On the other hand, the K_{ADPCM}^{class} domain is arbitrarily set between 1 and 256, where a maximum reconstruction quality, with a minimum compression ratio, is achieved when $QF = 256$ while a maximum compression ratio, with a minimum reconstruction quality, is achieved when $QF = 1$.

AC Quantizer

For most natural scene images, the majority of the AC-coefficients have small magnitudes. Hence, these coefficients have the smallest impact on the reconstructed image quality. Consequently, they can be coarsely quantized, or even discarded entirely, with the trade-off of having a little image distortion.

The role of the AC quantizer is to eliminate selectively or quantize coarsely the AC-coefficients that carry the least information. In this quantizer, the AC-coefficient in row i and column j is quantized using a scalar value defined by

$$\text{AC-quantization-step} = \begin{cases} \left\lfloor \frac{q_{ij}}{K_{AC}^{class}(QF)/256} \right\rfloor & \text{if } q_{ij} * 256 > K_{AC}^{class}(QF), \\ 1 & \text{otherwise.} \end{cases} \quad (3.9)$$

where $q_{ij} \in Q$, Q is an 8×8 normalization matrix and $K_{AC}^{class}(QF)$ is a class dependent positive non-zero integer function of the quality-factor QF while *class* is restricted

only to either textural or edge class. The Q normalization matrix, shown in (3.10), is adopted after the JPEG standard [60]. The role of this Q matrix, which was heuristically determined, is to weight the AC-coefficients according to their perceptual importance.

$$Q = \begin{pmatrix} * & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \quad (3.10)$$

Similar to the K_{ADPCM}^{class} , the K_{AC}^{class} is used as a scaling factor to compromise between the compression ratio and the quality of the reconstructed images. As the smallest and the largest element in Q are equal to 10 and 121, respectively, and the range of the AC-coefficients is between $\pm(2^8 - 1) \times block_size/2$, the range of K_{AC}^{class} will be

$$2 \leq K_{AC}^{class} \leq 30976. \quad (3.11)$$

At $K_{AC}^{class} = 2$, all of the AC-coefficients will be quantized to zero, i.e., a given image-block belongs to this class will be represented by its DC-coefficient only. On the other hand, at $K_{AC}^{class} = 30976$ all of the AC-coefficients will be un-quantized at all. Meanwhile, the K_{AC}^{class} domain is arbitrarily set between 1 and 256, where a maximum reconstruction quality, with a minimum compression ratio, is achieved when $QF = 256$ while a maximum compression ratio, with a minimum reconstruction quality, is achieved when $QF = 1$.

3.1.6 Zigzag-ordering

After quantization, many of the AC-coefficients are set to zero, especially at higher frequencies. To take advantage of these zeros, the 2-D block of the AC-coefficients is reformatted into 1-D vector using a zigzag-order, as shown in Figure 3.7. Zigzag-ordering places low-frequency coefficients, which are more likely to have non-zero values, before high-frequency coefficients. As a result of this ordering process, the AC-coefficients are re-arranged in order of both increasing spatial frequency and decreasing absolute value.

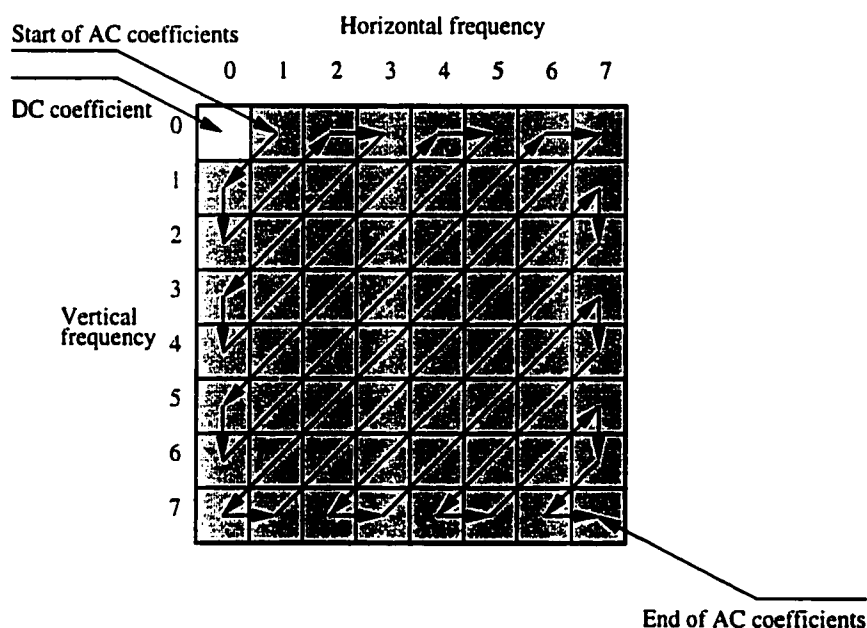


Figure 3.7: The zigzag-ordering sequence.

3.1.7 Run-length Encoder

After performing the zigzag-ordering process, several long runs of zero values are created. To capitalize on this situation, these runs of zero values are run-length encoded, as in [63] (i.e., these runs of zero values are replaced with a count of their number). In run-length encoding, each non-zero AC-coefficient is represented by a composite word of $s + r$ bits.

The composite word consists of two parts: the s most significant bits, and the remaining r least significant bits. The first part is devoted to represent the magnitude of the coefficient, whereas the second part is devoted to represent the current coefficient position relative to the previous non-zero coefficient, i.e., the run-length of the zero coefficients between the non-zero coefficients. This means that the second part's range is between 0 and $2^r - 1$. A distinct composite word, which is $2^r - 1$, is defined to represent a run-length of 2^r zero coefficients. If the run-length exceeds 2^r zero coefficients, it is encoded by using multiple composite words. In addition, another distinct composite word, which is 0, is used to encode the *end of block* (EOB). The EOB word specifies that all the remaining coefficients in the block are quantized to zero. Finally, the sign of each non-zero coefficient is determined and encoded, separately, by using a single bit per sign.

It might seem that run-length encoding expands, instead of compressing, the number of bits required to encode the AC-coefficients by r bits. However, this is not true, since not all the AC-coefficients are represented by run-length words. In fact, run-length words are reserved only for non-zero coefficients or runs of more than $2^r - 1$ zero coefficients. Also, EOB word saves so many words that may be needed to represent the trail run of zero coefficients at the end of the block. Moreover, the bulk of the most significant s bits in any run-length word are zeros which are dealt with, efficiently, through the arithmetic encoding.

The value of s depends on the values of both the AC-coefficients and their quantization. It can be determined according to the expression given by

$$s = \left\lceil \log_2 \left(\frac{\text{maximum } |\text{AC coefficient value}|}{\text{minimum AC quantization value}} \right) \right\rceil. \quad (3.12)$$

As the range of the AC-coefficients is between $\pm(2^8 - 1) \times \text{block_size}/2$, s can be re-written

as

$$s \approx 10 - \lceil \log_2 (\text{minimum AC quantization value}) \rceil. \quad (3.13)$$

Note that the value of the AC-quantization is a function of QF . Hence, the s will be a function of the QF too.

As both textural and edge-blocks are considered to be among the 8×8 blocks (i.e., there is a maximum of 64 coefficients per block), the range of r is between 0 and 6, where $r = 0$ means that run-length encoder is not presented at all. Note that the increasing/decreasing of the r value will decrease/increase the overall number of run-length words required to encode the AC-coefficients. However, the length of each of these words will be longer/shorter. At high compression ratios, most of the high frequency AC-coefficients are vanished to zero. Hence, after encoding a very few AC-coefficients, the EOB word will be used to declare that all the remaining coefficients in the block are quantized to zero. This means that the possibility of having long runs of zero values becomes lower. Therefore, a smaller value of r might be a good choice. On the other hand, at low compression ratios, the possibility of having an un-vanished high frequency AC-coefficient becomes higher. This means that the possibility of having long runs of zero values becomes higher too. Therefore, a larger value of r might be a good choice. In this work, the value of r is empirically determined to improve the compression performance of ABC-SC (Section 3.2.3).

3.1.8 Arithmetic Encoder

Although the ADPCM, the DCT, and the run-length encoding techniques reduce the inter-pixel correlations significantly, the resulting codewords are still correlated slightly. To further reduce the remaining correlations, the resulting codewords are arithmetically encoded [68]. The arithmetic encoding technique works by representing a given sequence

of codewords by an interval of real numbers between 0 and 1. As the number of codewords increases, the interval needed to represent them becomes smaller. Consequently, the number of bits needed to specify this interval increases. The reduction of the interval corresponding to a given codeword is based on the probability of this codeword. Instead of using a fixed probability model, an adaptive binary-tree state model is used [69], where each codeword to be compressed is decomposed into bits in order to fit into the model. In this model, the most significant bit of each codeword starts the tree at its root node. As more bits are considered, the tree is traversed downwards. The adaptability of this model comes from the way of updating the frequencies based on the seen codewords. At the beginning, all counts are equally initialized (reflecting no initial information). Then, as each codeword is seen, the counts are updated to approximate the observed frequencies. As both of the encoder and decoder use the same initial values (e.g., equal counts) and the same updating algorithm, their models will remain in step concurrently.

The arithmetic encoding technique compresses a sequence of codewords at least as compact as the Huffman encoding technique [70]. Moreover, it accommodates adaptive models easier, and it is computationally less expensive than the adaptive Huffman encoding technique.

3.1.9 Post-processing

As most of the compression techniques start to achieve very high compression ratios, artifacts which severely degrade the perceived quality of the reconstructed images start to appear. As ABC-SC is a block-based encoding technique, the most noticeable artifact is generally the discontinuities present at block boundaries, which is called blocking-artifacts. The blocking-artifacts distinctively occur in smooth regions, where blocks are represented by their quantized averages. As pixels intensity values might not be close enough, the difference between two adjacent block averages might be significantly large,

especially at high compression ratios, due to the coarse quantization. Hence, the boundaries of these blocks might become visible. To cope with this annoying intra-blocking degradation artifact without increasing the bit rate, a post-processing filtering scheme is needed to restore the nice continuity of the grey-scale intensities over the entire reconstructed image regions.

In [71, 72], a low-pass filtering scheme is carried out only on the block boundary pixels to remove the high frequency part of the visible block boundaries. However, this approach might introduce unnecessary blurring to the edge regions and hence, degrades edges. Several techniques attempt to overcome this smoothing edges problem by estimating the edge locations in the reconstructed image first to avoid blurring [73, 74]. This, however, is a very difficult task for very high compression ratios, where the actual edge information is somewhat scrambled. Note that ABC-SC decoder receives the blocks classification information within the compressed image data and hence, the edge location estimation process is no longer needed.

The proposed post-processing filtering scheme consists of two phases. In the first phase, the smooth regions are adaptively blurred, whereas in the second phase, the connection between the smooth and the non-smooth regions is performed.

In the adaptive blurring phase, each pixel in the smooth region is blurred twice: one in a row-wise order and the other in a column-wise order (or vice versa, i.e., in a column-wise order then in a row-wise order), where the blurring starts at the boundary rows/columns and ends at the center of the block. The following adaptive formulas are used to blur upper rows, lower rows, left columns, and right columns, respectively.

$$\hat{f}(x, y) \Leftarrow \frac{1}{6} \sum_{\Delta x = -1}^0 \sum_{\Delta y = -1}^1 \hat{f}(x + \Delta x, y + \Delta y) \quad (3.14)$$

$$\hat{f}(x, y) \leftarrow \frac{1}{6} \sum_{\Delta x = 0}^1 \sum_{\Delta y = -1}^1 \hat{f}(x + \Delta x, y + \Delta y) \quad (3.15)$$

$$\hat{f}(x, y) \leftarrow \frac{1}{6} \sum_{\Delta x = -1}^1 \sum_{\Delta y = -1}^0 \hat{f}(x + \Delta x, y + \Delta y) \quad (3.16)$$

$$\hat{f}(x, y) \leftarrow \frac{1}{6} \sum_{\Delta x = -1}^1 \sum_{\Delta y = 0}^1 \hat{f}(x + \Delta x, y + \Delta y) \quad (3.17)$$

where $\hat{f}(x, y)$ is the reconstructed pixel intensity value at row x and column y . The order of performing the row- or the column-wise blurring process is adaptively determined based on the values of the vertical and the horizontal boundaries variations (B_v and B_h , respectively). These are defined by

$$B_v = \sum_{\forall \text{ upper pixels}} |\hat{f}(x, y) - \hat{f}(x - 1, y)| + \sum_{\forall \text{ lower pixels}} |\hat{f}(x, y) - \hat{f}(x + 1, y)| \quad (3.18)$$

and

$$B_h = \sum_{\forall \text{ left pixels}} |\hat{f}(x, y) - \hat{f}(x, y - 1)| + \sum_{\forall \text{ right pixels}} |\hat{f}(x, y) - \hat{f}(x, y + 1)|, \quad (3.19)$$

where x and y represent the block boundary row and column locations, respectively. Note that the upper, lower, left and right pixels mean the block boundary pixels in the uppermost row, lowermost row, leftmost column and rightmost column, respectively. For blocks with B_v greater than B_h , the row-wise blurring process is carried out before the column-wise. Otherwise, the column-wise blurring process is carried out first. This could also be viewed as a two-step process. The first step serves to reduce major variations at boundary pixels of smooth-blocks in one direction. The second step implements a fine tuning operation in a perpendicular direction to that of the first step. After performing both of the row- and the column-wise blurring processes, a constant value is added to

each pixel in the block to maintain the block average as it was before blurring. The entire blurring process is performed on all smooth-blocks starting with the 32×32 blocks down to the 8×8 blocks. Figure 3.8 demonstrates the steps of the blurring process.

In the second post-processing phase (i.e., the connection between the smooth and non-smooth regions), the pixels in the block corners at the boundary between the smooth and non-smooth regions are adjusted in order to reduce the boundary stair-like shape. This task is accomplished in two steps. The first step interpolates any smooth corner which is horizontally and vertically neighbored by non-smooth-blocks. This is carried out by diagonally linear interpolating corners using the two neighboring non-smooth pixels in the direction of the diagonal extension. The second step is to blur any non-smooth corner which is surrounded by smooth blocks. This is carried out by averaging each pixel in the corner with its three neighbor (towards the corner) pixels. The interpolation and the averaging processes start from the corner pixel and continue diagonally towards the block center. Figure 3.9 demonstrates the definition of locations within the corner and Figure 3.10 demonstrates the steps of the connection process.

3.2 Parameters Adjustment

3.2.1 Thresholding and Quantization Functions

There is no doubt that the impact of the quantization functions on the ABC-SC performance is affected by the values of the thresholding parameters. The T thresholding parameters control the routing of the blocks to the different compression modules. In the same time, the impact of the T thresholding parameters on the ABC-SC performance is also affected by the definition of the K quantization functions, which control the allowed amount of distortions that might occur to the blocks of each class. Therefore, care should be taken when identifying these parameters and functions.

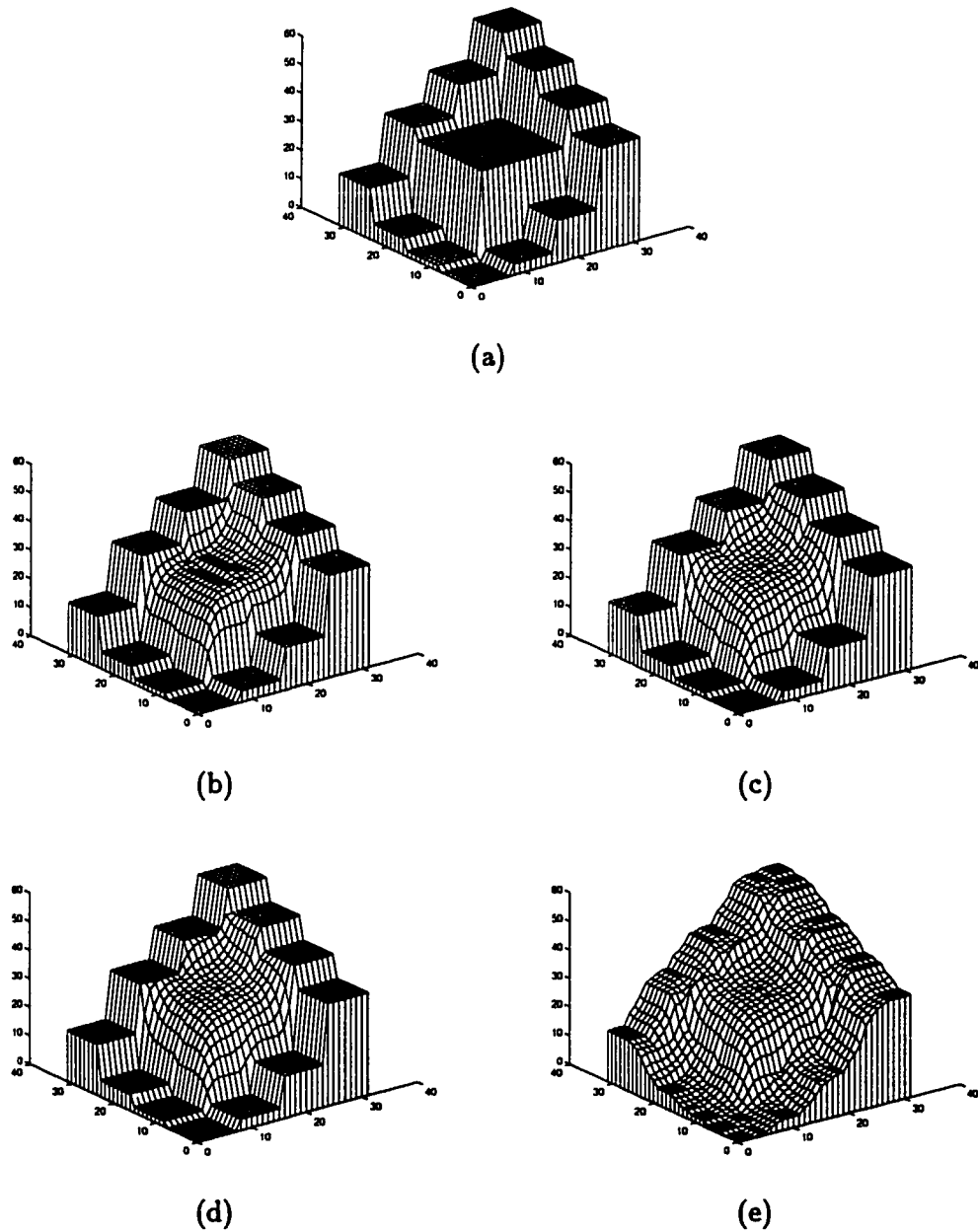


Figure 3.8: A post-processing example. (a) before post-processing, where $B_h = 432$ and $B_v = 344$; (b) after horizontal blurring; (c) after vertical blurring; (d) after average adjustment; and (f) after full blurring.

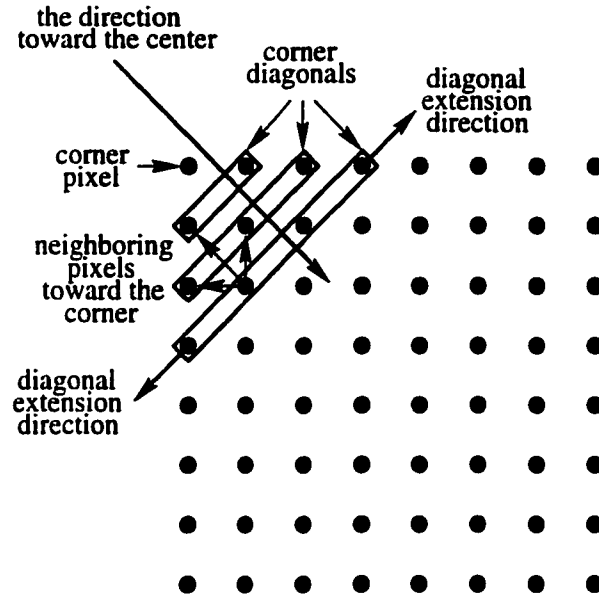


Figure 3.9: The naming conventions of the locations within the corner.

In this work, the parameters and functions' identification process is divided into two phases. In the first phase, all the non-smooth-blocks are dealt with as if they are edge-blocks, i.e., only two classes of blocks are considered (smooth and edge classes). Consequently, only the non-texture-related parameters and functions (i.e., T_m , K_{ADPCM}^{smooth} , K_{ADPCM}^{edge} , and K_{AC}^{edge}) are identified in this phase. Then, in the second phase, the rest of the parameters and functions (i.e., the texture-related parameters functions, namely, $T_{average}$, $T_{deviation}$, $K_{ADPCM}^{texture}$, and $K_{AC}^{texture}$) are identified, based on the value of the already-identified non-texture-related parameters and functions.

Non-texture-related Functions

To add more adaptability to ABC-SC, each of the T_m thresholding parameters is considered a function of the QF , rather than just a scalar value. The role of these T_m thresholding functions depends on the value of the QF . For high QF values, the role of these functions is to facilitate the conditions on blocks to be classified as non-smooth-

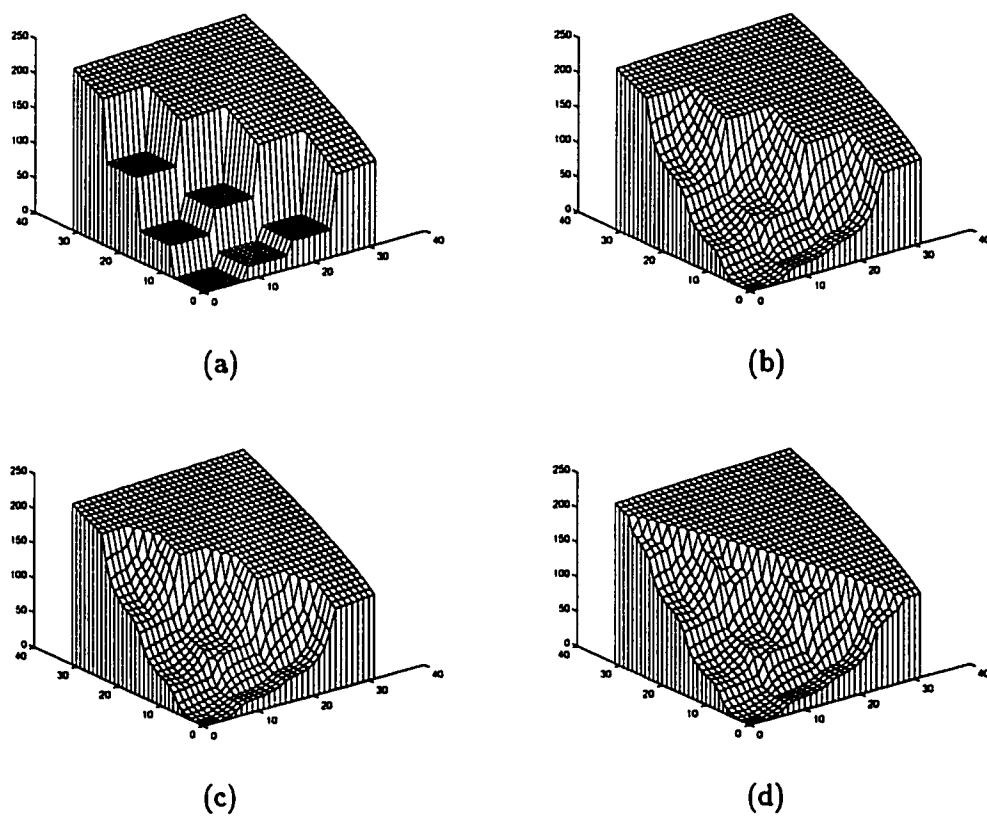


Figure 3.10: Another post-processing example. (a) before post-processing; (b) after smooth region blurring; (c) after smooth corner interpolation; and (d) after non-smooth corners blurring.

blocks. Hence, the blocks gain less distortions. On the other hand, for low QF values, their role is to ease the conditions on blocks to be classified as smooth-blocks. Hence, they can be compressed further.

Theoretically speaking, any decreasing function can be considered as a valid definition for any of the T_m thresholding functions. Also, any increasing function can be considered as a valid definition for any of the non-texture-related K quantization functions, namely, K_{ADPCM}^{smooth} , K_{ADPCM}^{edge} , and K_{AC}^{edge} . However, as the performance of ABC-SC strongly depends on the definition of these functions, they should be chosen carefully.

Since optimizing both of the T and the K functions simultaneously is not an easy optimization problem, we propose an iterative solution for identifying them sequentially. The basic idea is to adjust only one function at a time, while fixing the rest of the functions, and repeat doing this adjustment in sequence until no change occurs during one full iteration.

During this identification process, each of these functions is assumed to be a continuous piece-wise linear function between some pre-determined QF values, namely, 1, 8, 16, 32, 64, 96, 128, 160, 192, 224, 240, 248, and 255. At $QF = 256$, all the T_m thresholding functions are set to zero so that all blocks are classified as edge-blocks. In the same time, the K_{AC}^{edge} and K_{ADPCM}^{edge} quantization functions are set to a large number. This allows all the AC- and the DC-coefficients to be preserved without any quantization at all. Hence, the lowest reconstruction degradation is achieved.

Note that the piece-wise linear assumption is a reasonable assumption, as each of these functions is either an increasing or a decreasing function. Also note that the domains of the piece-wise segments are selected to be equal, except at low and high QF values where the changing rate of the T_m thresholding functions and the K quantization functions are expected to be higher than the changing rate for the rest of the QF values. With this assumption, our goal now is to determine the output values of each function at these

pre-determined QF values.

During this phase, determining the functions output values is based on 8 different 512×512 images. These 8 images are selected to represent different image classes including, edge images (e.g., the Monarch and the Boats images); textural images (e.g., the Rocks and the Barbara images); smooth images (e.g., the Zelda and the Peppers images); and natural scene images (e.g., the Goldhill and the F16 images). Figure 3.11 shows these 8 images.

In this phase, the determination of the output values of a given function is achieved by considering all the valid output values of this function. Then, for each QF value, a rate-distortion curve, based on compressing the 8 images, is generated. Note that each point on this curve corresponds to one of the valid output values. Then, an envelope curve for all these rate-distortion curves, which maximizes the compression ratio and, at the same time, minimizes the root mean squared error, is generated. The envelope curve may be approximated by a continuous piece-wise linear curve. The first segment of this curve is the line tangent to the rate-distortion curve corresponding to $QF = 255$ starting from the rate-distortion point corresponding to $QF = 256$. The second segment of this curve is the line tangent to the next rate-distortion curve (i.e., the curve corresponding to $QF = 248$) starting from the current tangent point (which is located on the curve corresponding to $QF = 255$), and so on. Finally, the output values corresponding to the tangent points between the envelope and each rate-distortion curve are determined and assigned to the value of the function at the corresponding QF values. This procedure is repeated for all the non-texture-related functions until no change occurs to any of them over a one full iteration. Note that after the adjustment of each function, the performance of ABC-SC is either improved, from a rate-distortion point of view, or at worst it is remained the same, if the unchanged function case is encountered. Hence, the procedure convergence is guaranteed. Figures 3.12 and 3.13 show the algorithmic steps of

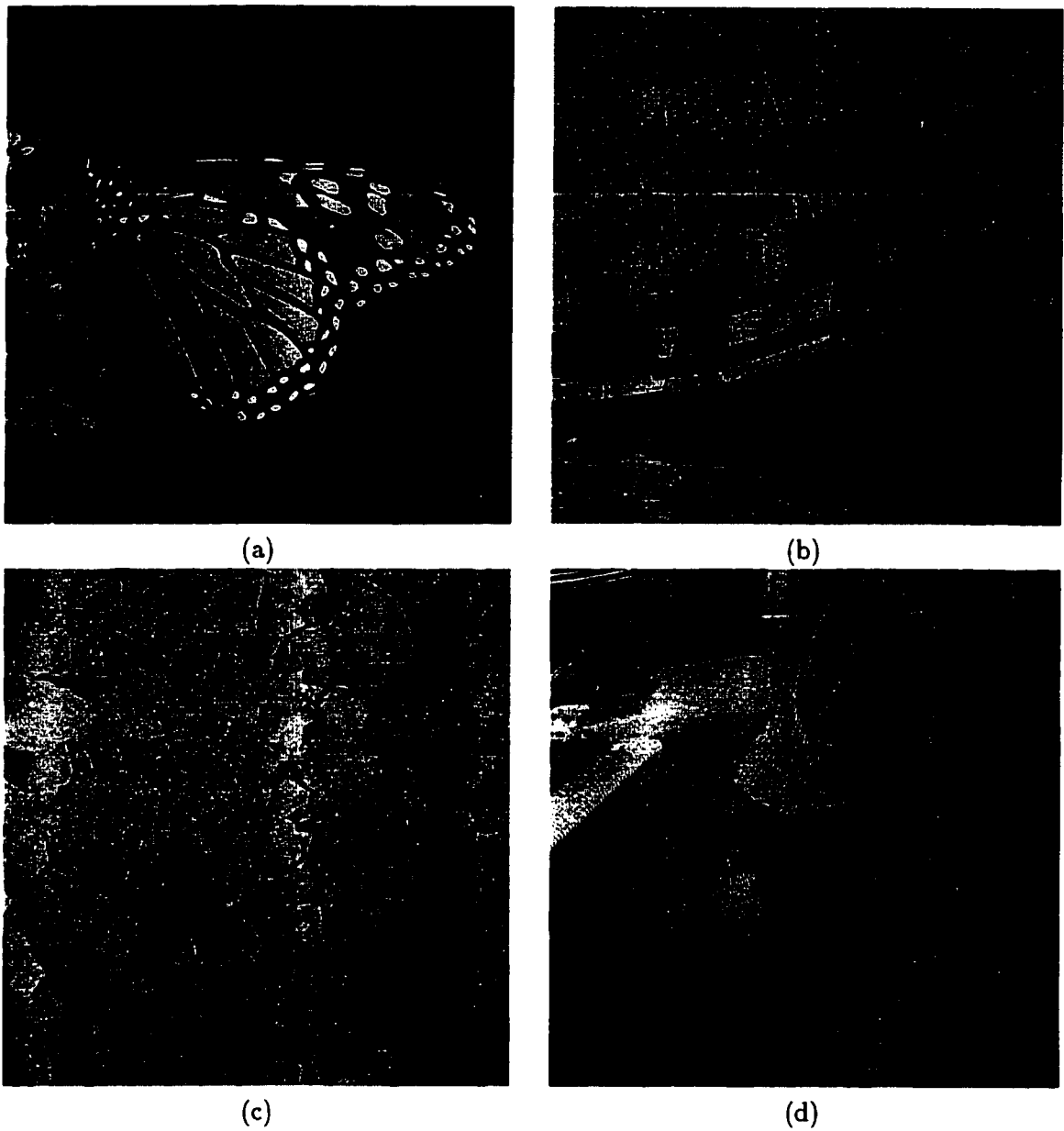


Figure 3.11: The 8 original images which are used during the functions' identification process. (a) Monarch; (b) Boats; (c) Rocks; (d) Barbara; (e) Zelda; (f) Peppers; (g) Goldhill; and (h) F16.

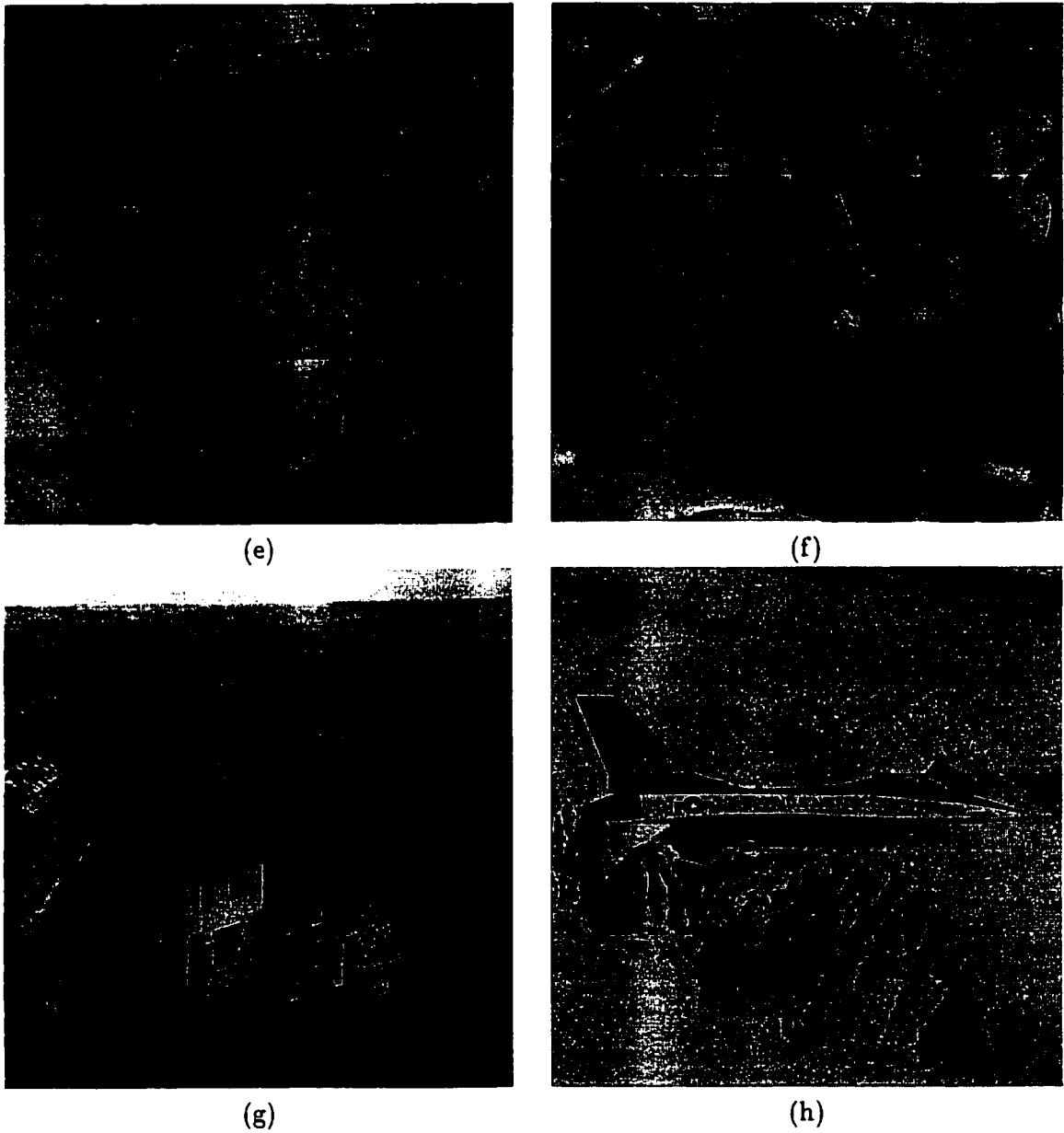


Figure 3.11: The 8 original images which are used during the functions' identification process, (continued).

the non-texture-related functions identification procedure, and an example of an envelope curve for rate-distortion curves which are generated for K_{ADPCM}^{smooth} during the functions' second identification iteration, respectively.

Note that this identification procedure is insensitive to initial conditions, i.e., any increasing function can be considered as a valid initial definition for any of the T_m thresholding functions, and any decreasing function can be considered as a valid initial definition for any of the K quantization functions. To show this insensitivity, the initial definitions of K_{ADPCM}^{smooth} and K_{ADPCM}^{edge} quantization functions are selected to be constant functions, as given by (3.20) and (3.21), respectively.

$$K_{ADPCM}^{smooth} = 256 \quad (3.20)$$

$$K_{ADPCM}^{edge} = 256 \quad (3.21)$$

Meanwhile, the initial definitions of T_{32} , T_{16} , and T_8 functions are selected to be continuous piece-wise linear decreasing functions, as given by (3.22)–(3.24), respectively.

$$T_{32} = \begin{cases} \lfloor -3/8 \times QF + 60 \rfloor & \text{if } 1 \leq QF < 64 \\ \lfloor -2/8 \times QF + 52 \rfloor & \text{if } 64 \leq QF < 160 \\ \lfloor -1/8 \times QF + 32 \rfloor & \text{if } 160 \leq QF < 255 \end{cases} \quad (3.22)$$

$$T_{16} = \begin{cases} \lfloor -9/2 \times QF + 528 \rfloor & \text{if } 1 \leq QF < 64 \\ \lfloor -4/2 \times QF + 368 \rfloor & \text{if } 64 \leq QF < 160 \\ \lfloor -1/2 \times QF + 128 \rfloor & \text{if } 160 \leq QF < 255 \end{cases} \quad (3.23)$$

Algorithm: *Non-texture-related-functions-identification***Step 0:** Begin algorithm:**Step 1:** Let functions be $\{ K_{AC}^{edge}, K_{ADPCM}^{edge}, K_{ADPCM}^{smooth}, T_8, T_{16}, \text{ and } T_{32} \}$.**Step 2:** Let QF-values be $\{ 1, 8, 16, 32, 64, 96, 128, 160, 192, 224, 240, 248, \text{ and } 255 \}$.**Step 3:** Repeat:**Step 3.1:** For each function in functions do:**Step 3.1.1:** For each QF-value in QF-values do:**Step 3.1.1.1:** Let valid-outputs be samples from all possible valid outputs for function (QF-value).**Step 3.1.1.2:** For each valid-output in valid-outputs do:**Step 3.1.1.2.1:** Compress the 8 images at quality-factor = QF-value, while assuming that function (QF-value) = valid-output.**Step 3.1.1.2.2:** Calculate the average compression ratio over the 8 images.**Step 3.1.1.2.3:** Decompress the 8 images.**Step 3.1.1.2.4:** Calculate the average Root Mean Squared Error (RMSE) over the 8 images.**Step 3.1.1.3:** End for.**Step 3.1.1.4:** Generate rate-distortion (QF-value) curve over the 8 images.**Step 3.1.2:** End for.**Step 3.1.3:** Generate envelope (function) for all the rate-distortion (QF-value) curves generated in this iteration.**Step 3.1.4:** For each QF-value in QF-values do:**Step 3.1.4.1:** Let the current value of function (QF-value) be the value of valid-output corresponding to the tangent point between envelope (function) and rate-distortion (QF-value).**Step 3.1.5:** End for.**Step 3.2:** End for.**Step 4:** Until no change is happened to any function during a one full iteration.**Step 5:** End algorithm.

Figure 3.12: The non-texture-related functions' identification algorithm.

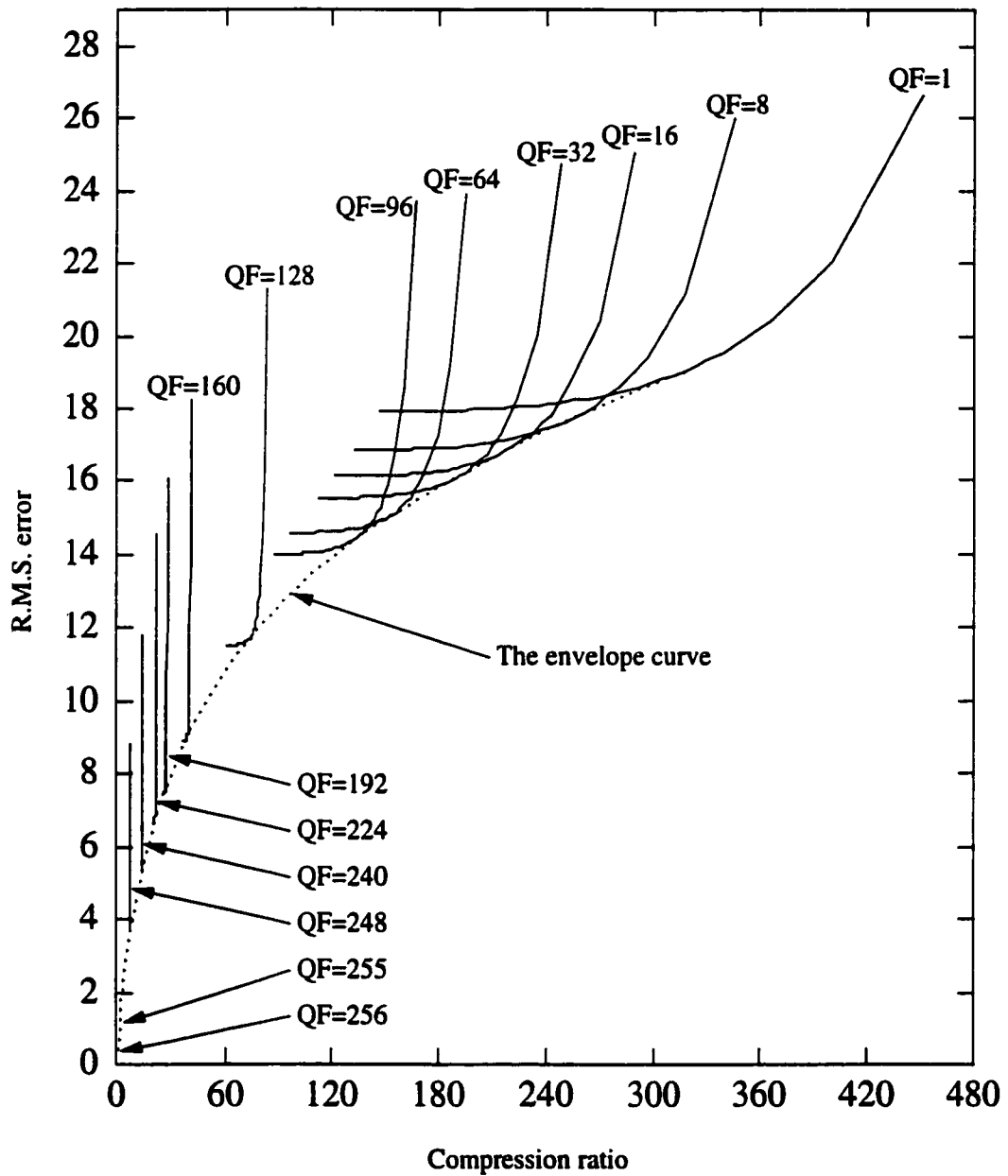


Figure 3.13: An example of an envelope curve for rate-distortion curves which are generated for K_{ADPCM}^{smooth} during the functions' second identification iteration.

$$T_8 = \begin{cases} \lfloor -54 \times QF + 5184 \rfloor & \text{if } 1 \leq QF < 64 \\ \lfloor -16 \times QF + 2752 \rfloor & \text{if } 64 \leq QF < 160 \\ \lfloor -2 \times QF + 512 \rfloor & \text{if } 160 \leq QF < 255 \end{cases} \quad (3.24)$$

After the second full iteration, all functions are converged to their final definitions. Table 3.3 shows the definition of each function (the initial and after each iteration definitions), where the changed values are highlighted. Note that the K_{ADPCM}^{smooth} and K_{ADPCM}^{edge} quantization functions took almost an extra iteration, than T_m thresholding functions, to converge, i.e., the initial conditions might change the number of the iterations needed to let the identification procedure converge.

Texture-related Functions

In general, the objective of the T and K parameter and function identification process is to maximize the compression ratios and, at the same time, minimize the reconstruction errors. However, the texture-related parameters and functions (i.e., $T_{average}$, $T_{deviation}$, $K_{ADPCM}^{texture}$, and $K_{AC}^{texture}$) might have different objectives according to the application. For example, in some image compression applications, more reconstruction errors in the textural regions might be allowed in order to achieve higher compression ratios. The reason is that textural regions usually convey a low amount of information to the viewer. However, there are image processing applications where the textural regions might be of interest. Hence, reducing their reconstruction errors may be more significant, to the viewer, than achieving higher compression ratios.

ABC-SC deals with these different situations by allowing users to define the quality of the reconstructed textural regions relative to the quality of the reconstructed edge regions (Texture-Quality-Ratio or, simply, TQR). If this ratio is less than one, this means that

Table 3.3: The outputs of all non-texture-related functions. (a) initial conditions; (b) after the first iteration; and (c) after the second iteration.

QF	K_{AC}^{edge}	K_{ADPCM}^{edge}	K_{ADPCM}^{smooth}	T_8	T_{16}	T_{32}
1	—	256	256	5130	523	59
8	—	256	256	4752	492	57
16	—	256	256	4320	456	54
32	—	256	256	3456	384	48
64	—	256	256	1728	240	36
96	—	256	256	1216	176	28
128	—	256	256	704	112	20
160	—	256	256	192	48	12
192	—	256	256	128	32	8
224	—	256	256	64	16	4
240	—	256	256	32	8	2
248	—	256	256	16	4	1
255	—	256	256	2	0	0

(a)

Table 3.3: The outputs of all non-texture-related functions, (continued).

QF	K_{AC}^{edge}	K_{ADPCM}^{edge}	K_{ADPCM}^{smooth}	T_8	T_{16}	T_{32}
1	<u>10</u>	<u>14</u>	<u>12</u>	<u>4300</u>	<u>700</u>	<u>58</u>
8	<u>12</u>	<u>15</u>	<u>13</u>	<u>2600</u>	<u>600</u>	57
16	<u>13</u>	<u>16</u>	<u>14</u>	<u>2000</u>	<u>500</u>	<u>56</u>
32	<u>14</u>	<u>16</u>	<u>14</u>	<u>1700</u>	<u>400</u>	<u>50</u>
64	<u>16</u>	<u>17</u>	<u>19</u>	<u>1400</u>	240	36
96	<u>18</u>	<u>18</u>	<u>20</u>	1216	176	<u>26</u>
128	<u>32</u>	<u>32</u>	<u>32</u>	<u>550</u>	112	<u>16</u>
160	<u>64</u>	<u>44</u>	<u>52</u>	<u>224</u>	48	12
192	<u>96</u>	<u>56</u>	<u>72</u>	128	<u>36</u>	8
224	<u>128</u>	<u>72</u>	<u>96</u>	<u>80</u>	<u>20</u>	4
240	<u>208</u>	<u>96</u>	<u>144</u>	<u>40</u>	<u>12</u>	2
248	<u>512</u>	<u>160</u>	<u>160</u>	<u>20</u>	<u>6</u>	1
255	<u>4096</u>	256	256	<u>0</u>	0	0

(b)

Table 3.3: The outputs of all non-texture-related functions, (continued).

QF	K_{AC}^{edge}	K_{ADPCM}^{edge}	K_{ADPCM}^{smooth}	T_8	T_{16}	T_{32}
1	10	<u>12</u>	12	<u>4000</u>	700	58
8	12	<u>12</u>	13	2600	600	57
16	13	<u>12</u>	<u>16</u>	2000	500	56
32	14	<u>13</u>	<u>16</u>	1700	400	50
64	16	<u>15</u>	19	1400	240	36
96	18	<u>16</u>	20	1216	176	26
128	32	<u>26</u>	32	550	112	16
160	64	<u>36</u>	<u>48</u>	224	48	12
192	96	<u>52</u>	<u>64</u>	128	36	8
224	128	<u>68</u>	<u>92</u>	80	20	4
240	208	<u>88</u>	<u>128</u>	40	12	2
248	512	160	<u>208</u>	20	6	1
255	4096	256	256	0	0	0

(c).

the user is willing to sacrifice some of the reconstruction quality in the textural regions in order to achieve higher compression ratios. On the other hand, if this ratio is greater than one, this means that the user is interested in the textural regions. Hence, less degradation is allowed in these regions even if the compression ratio is decreased. Finally, if this ratio is equal to one, this means that both of the edge and the textural regions have the same interest to the user and there is no need to differentiate between them (two-class case), i.e., as done in the first phase.

The main difference between edge- and textural-blocks is the level of activities within the block and not in its average (i.e., the AC-coefficients and not the DC-coefficient). This suggests that the texture-related K quantization functions, $K_{ADPCM}^{texture}$ and $K_{AC}^{texture}$, may be defined as follows.

$$K_{ADPCM}^{texture} = K_{ADPCM}^{edge} \quad (3.25)$$

$$K_{AC}^{texture} = TQR \times K_{AC}^{edge} \quad (3.26)$$

The values of $T_{average}$ and $T_{deviation}$ have been determined experimentally by visually examining many textural and non-textural-blocks in the 8 images shown in Figure 3.11. Based on these examinations, the following values are suggested.

$$T_{average} = 400 \quad (3.27)$$

$$T_{deviation} = 0.96 \quad (3.28)$$

According to experimentations, we found that these thresholding values are quite successful in separating textural-blocks from edge-blocks. Figure 3.14 shows a sample

of segmentation results for the Lena image at QF equals 200, and Figure 3.15 shows the pixel distribution for the different image classes for the Lena image at different QF values.

3.2.2 Neighboring Block Averages Parameter

To achieve a maximum benefit from all prediction rules, it is required to choose the value of v so that all these prediction rules are utilized evenly. To do so empirically, the 8 images, shown in Figure 3.11, are re-considered. This time, these images are used for studying the effect of v on partitioning the usage of the prediction rules. This can be done by compressing these images and counting the number of times each prediction rule is used. Since the variance of these counts gives an indication about how even the utilization of the prediction rules is (the higher/lower the variance value, the less/more even utilization of the prediction rules is), the required v value is chosen to be the value corresponding to the minimum variance for each QF value. Figure 3.16 shows the variance values for different values of v and QF , and Figure 3.17 shows the chosen v values as a function of QF .

3.2.3 Run-length Parameter

The last parameter that need to be adjusted is r , the width of the run-length field. Similar to v , the 8 images are re-considered. This time, for each r value in $\{1, 2, 3, 4, 5, 6\}$, all these images are compressed then decompressed. Then, the average overall QF -rate relation is generated. The required r value is chosen to be the value corresponding to the maximum compression ratio for each QF value. Figure 3.18 shows the relation between the QF and the compression ratio for different r values, after subtracting the compression ratio for the $r = 6$ case from each of them (i.e., normalized to the $r = 6$ case) Figure 3.19 shows the chosen r values as a function of QF .

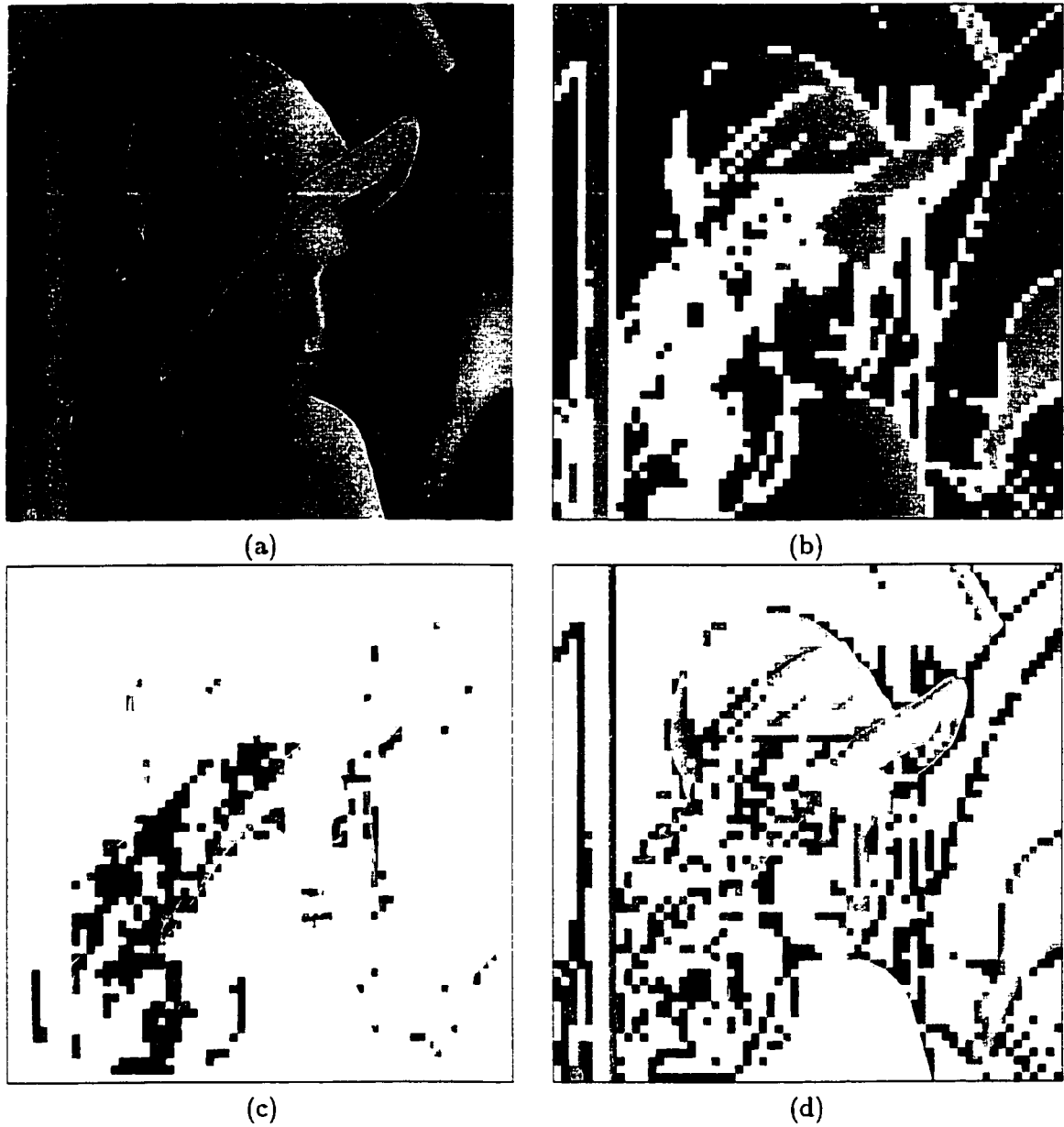


Figure 3.14: Segmentation results for the Lena image at QF equals 200. (a) the original Lena image; (b) smooth segment image; (c) textural segment image; and (d) edge segment image.

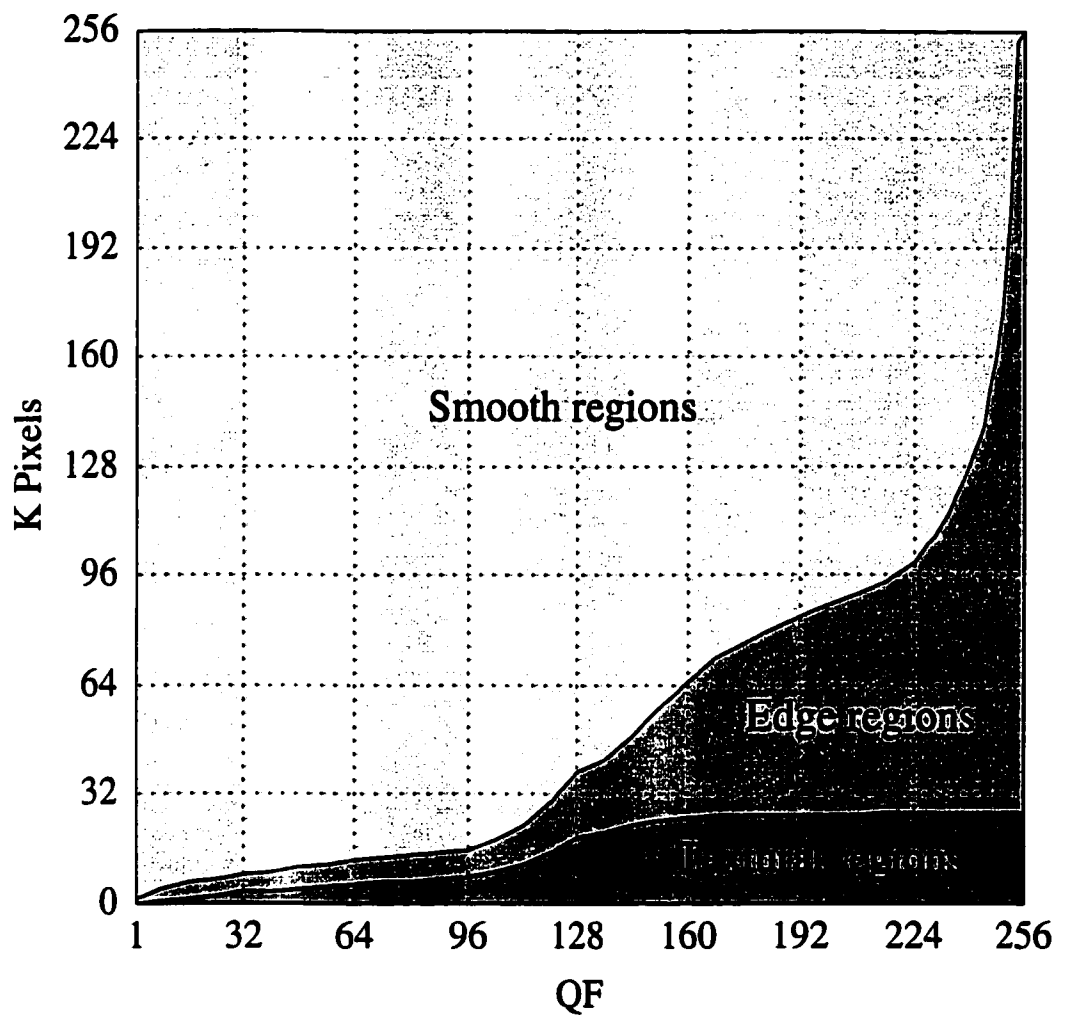


Figure 3.15: Pixels distribution for the Lena image.

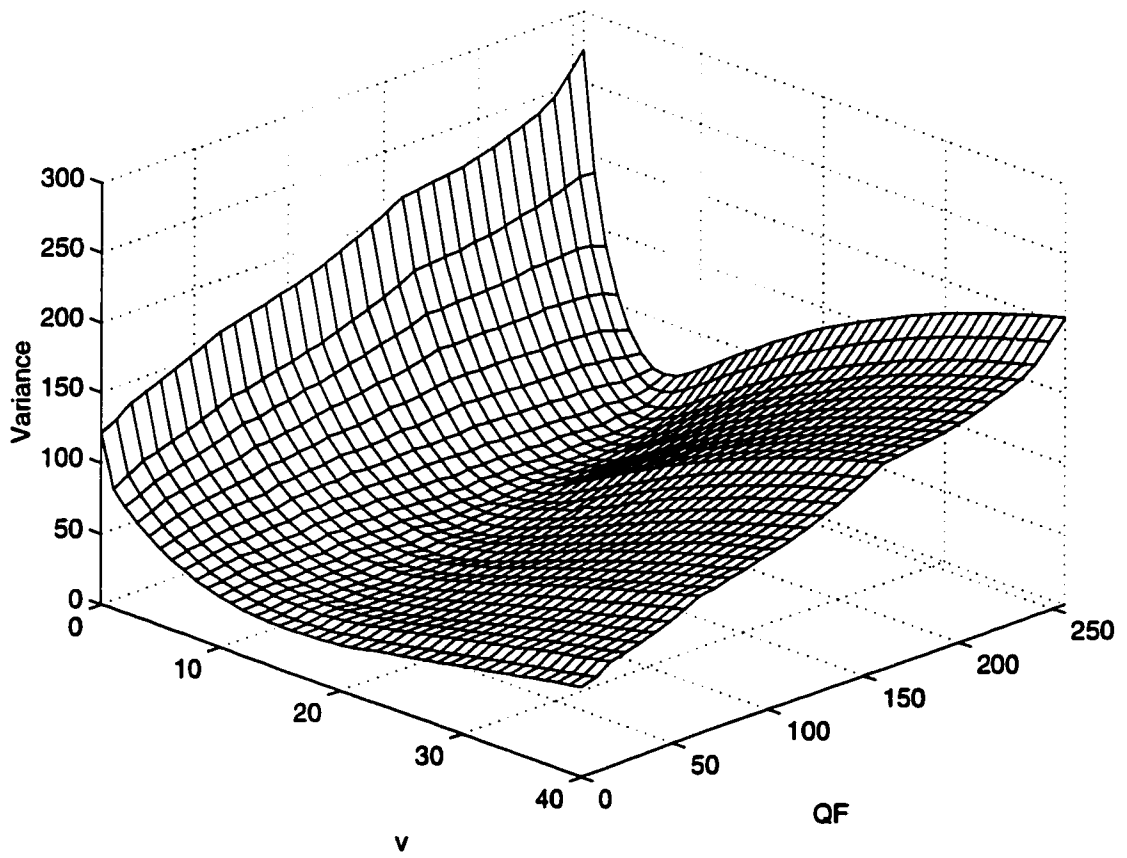


Figure 3.16: The effect of both v and QF on the utilization of the prediction rules.

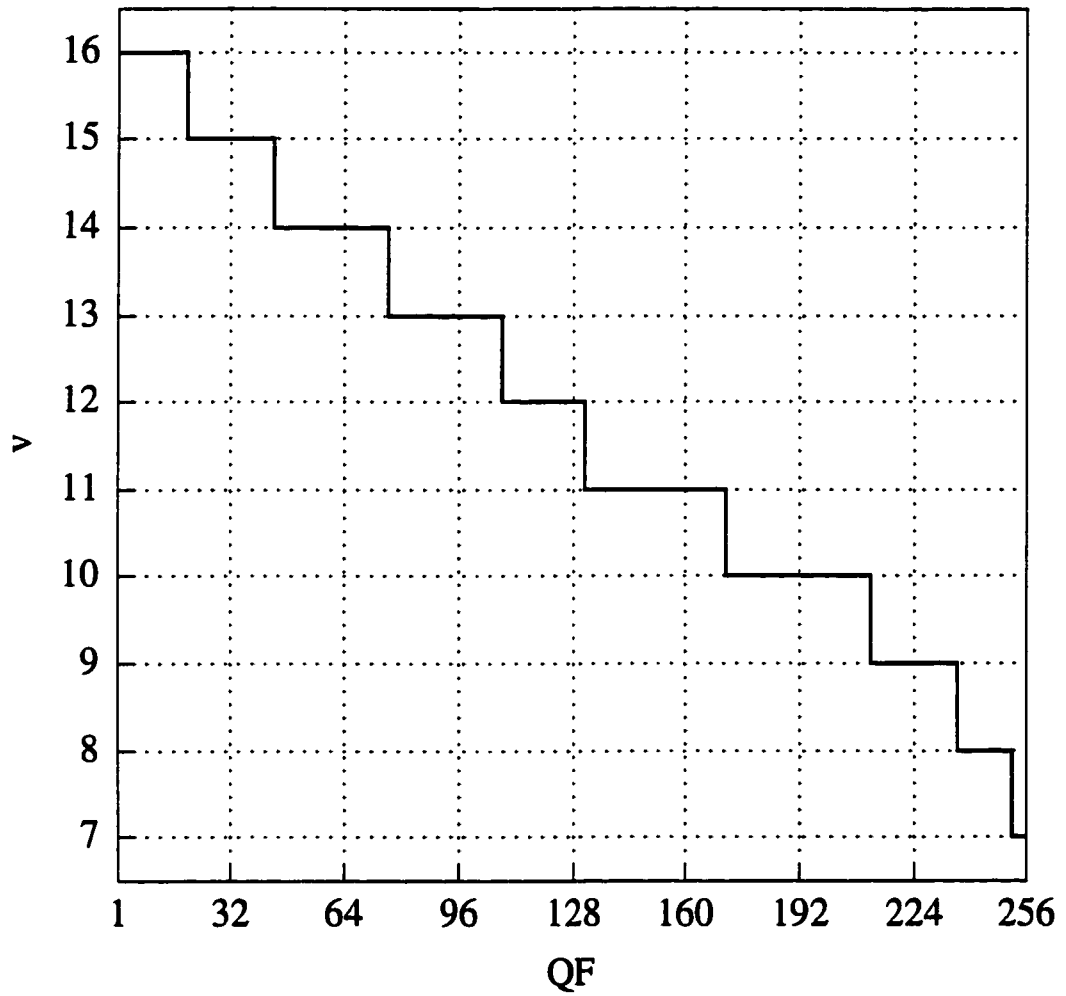


Figure 3.17: The values of v as a function of QF .

Note that ABC-SC is insensitive to the value of r . This appears in the maximum peak to peak difference in compression ratio between any two cases of r (which is always less than 2) for the same value of QF . Hence, just a minor improvement in the compression ratio is gained when using the adaptive r value. However, the improvement due to the presence of the run-length encoding is not minor. Figure 3.20 shows the effect of using run-length encoding with the adaptive r value, after subtracting the compression ratio for the $r = 0$ case from it (i.e., normalized to the the case of not using run-length encoding).

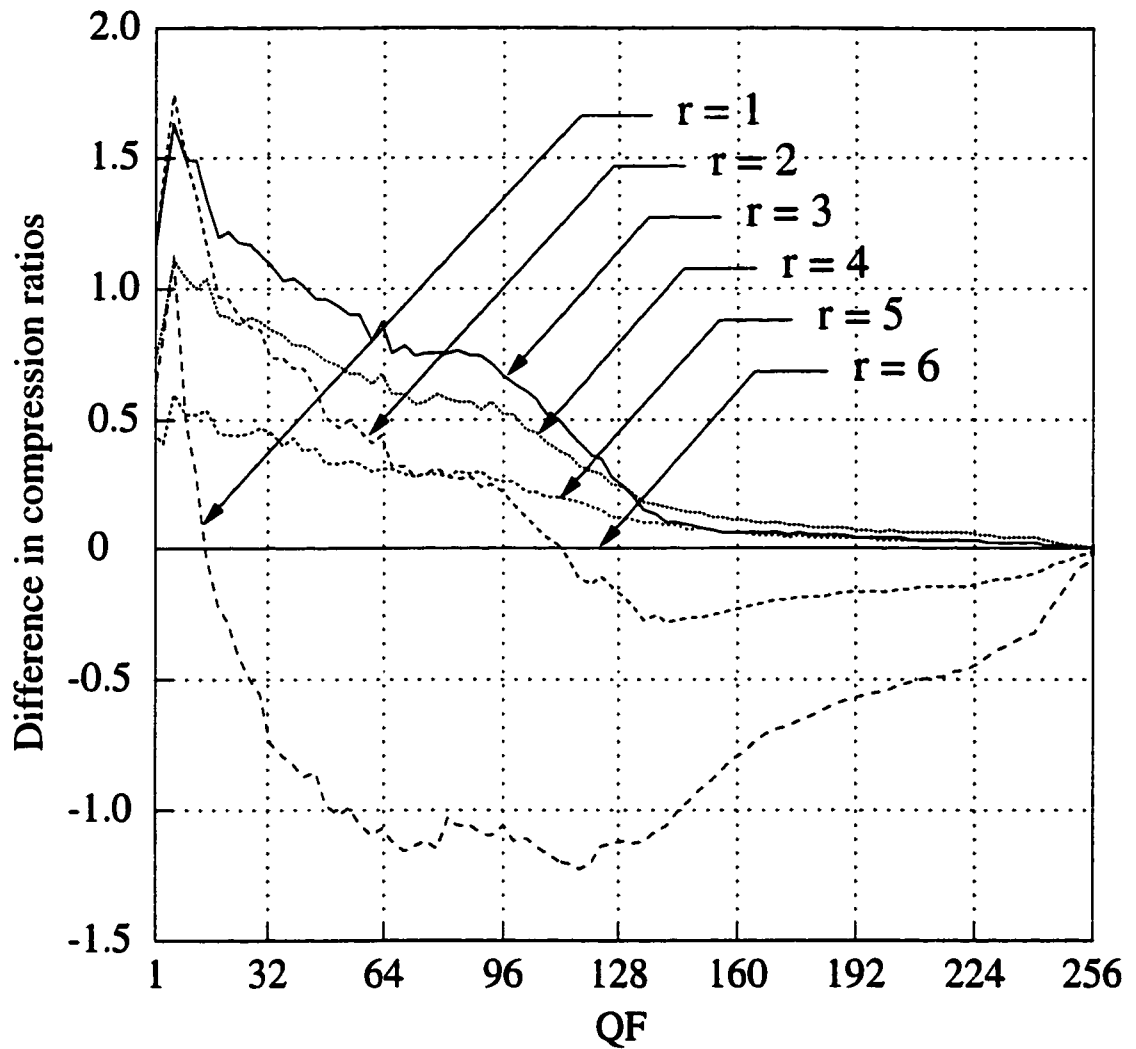


Figure 3.18: The relation between QF and compression ratio for different r values, normalized to the $r = 6$ case.

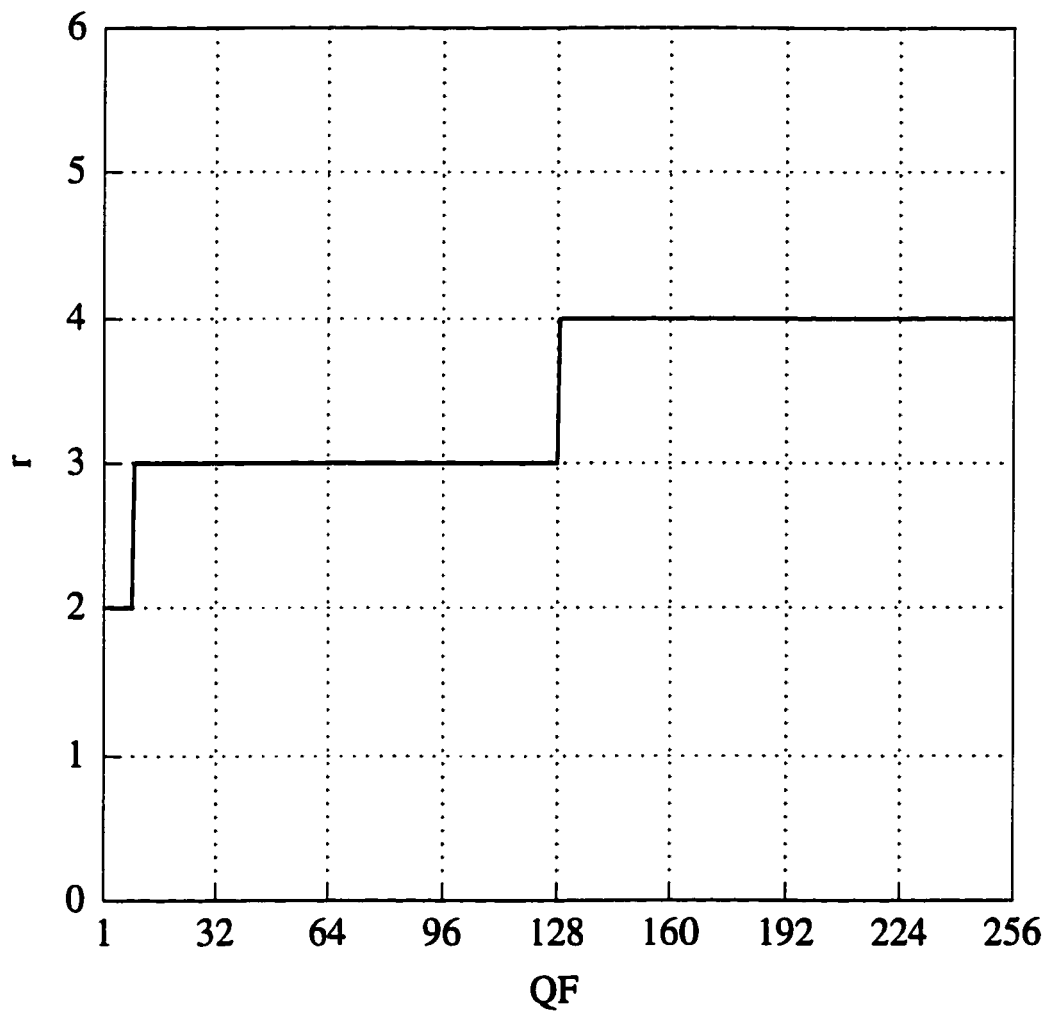


Figure 3.19: The values of r as a function of QF .

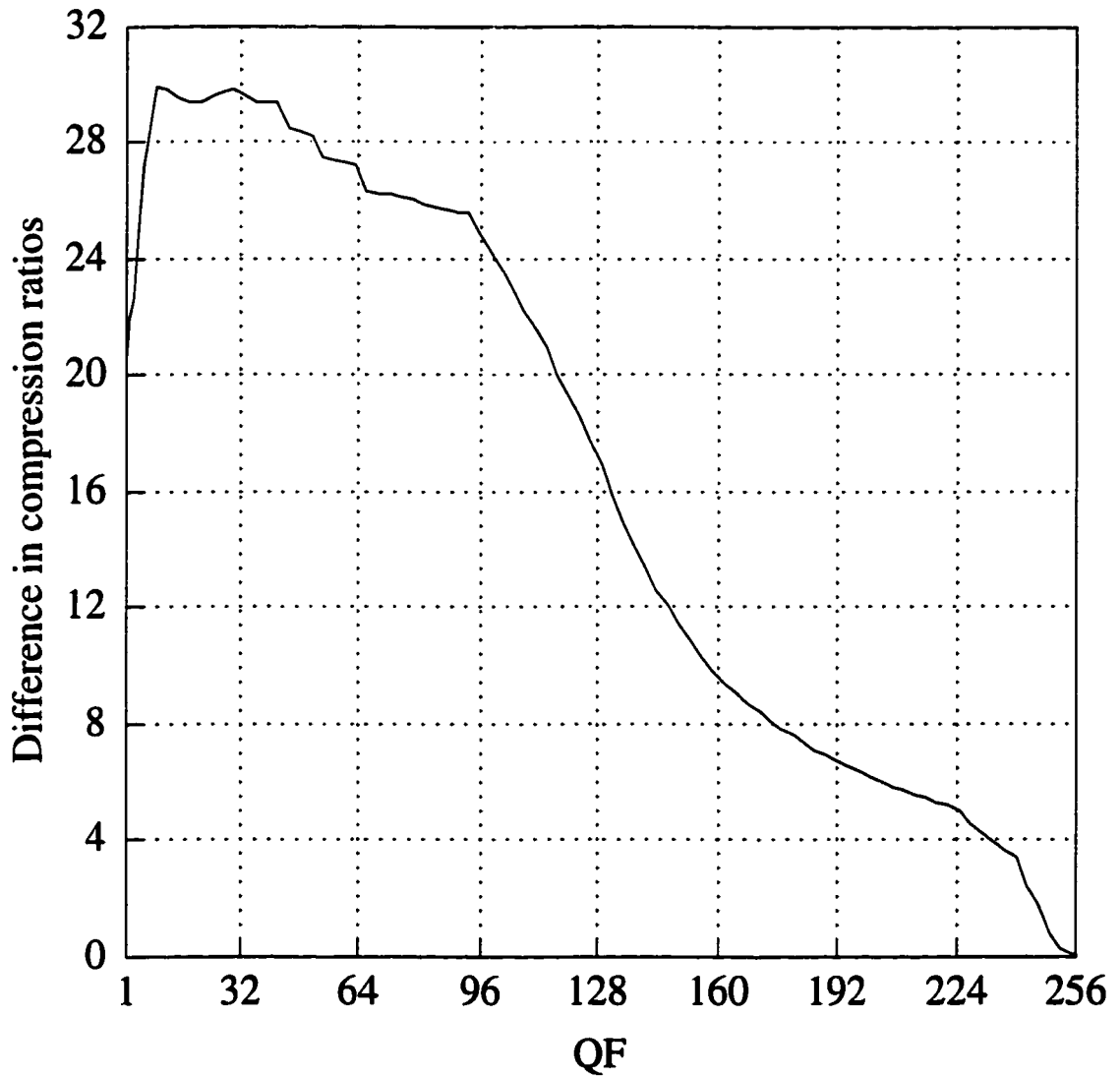


Figure 3.20: The relation between QF and compression ratio for the adaptive r value case, normalized to the case of not using run-length encoding (i.e., the $r = 0$ case).

Chapter 4

Results and Discussion

This chapter shows results and evaluation of the performance of the ABC-SC technique on different images. It also provides representative performance comparisons with other well known image compression techniques. Additional comparative results are included in Appendix A and Appendix B.

4.1 Performance Metrics

Since ABC-SC is a lossy technique, its reconstructed image may have some loss of information which may be visually useful. To assess the loss of fidelity in reconstructed images, a measurement of distortion should be used. As most decompressed images ultimately are viewed by human beings, it is appropriate to measure their qualities by subjective evaluations of human observers. Unfortunately, a simple convenient subjective evaluating mechanism does not exist.

As alternative measures, *Peak Signal-to-Noise-Ratio (PSNR)* and *Root Mean Squared*

Error (*RMSE*) are usually used to accomplish this task.

$$PSNR = 10 \log_{10} \left(\frac{255}{RMSE} \right)^2 dB \quad (4.1)$$

$$RMSE = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (\hat{f}(x, y) - f(x, y))^2} \quad (4.2)$$

where,

- $f(x, y)$: The original image,
- $\hat{f}(x, y)$: The reconstructed image,
- M : The height of the image, and
- N : The width of the image.

Although PSNR (and RMSE) are widely used in the lossy-compression literature, they are sometimes misleading and not indicative of the actual loss of fidelity. This is especially critical for low values of PSNR (or the high values of *RMSE*). For example, consider the images in Figure 4.1(b)–(d). They are a histogram flattening, a Gaussian blurred, and a zero mean Gaussian noisy versions, respectively, of the 512×512 Lena image shown in Figure 4.1(a). The histogram flattening is an image enhancement technique which considerably improves the appearance of an image by increasing its dynamic range. On the other hand, the Gaussian blurring is a low pass filtering technique which removes most of the image details and reduces its fidelity. Even though, the *PSNR* and the *RMSE* between any of these three images and the original Lena image are identical. Their values are 26.29 dB and 12.36, respectively. Hence, according to any of the two measurements, the quality of any of these images is considered the same, which is not true.

Note that since *PSNR* is a log function of *RMSE*, it exaggerates differences in the near-lossless range (perceptually lossless range) and suppresses differences in the highly-

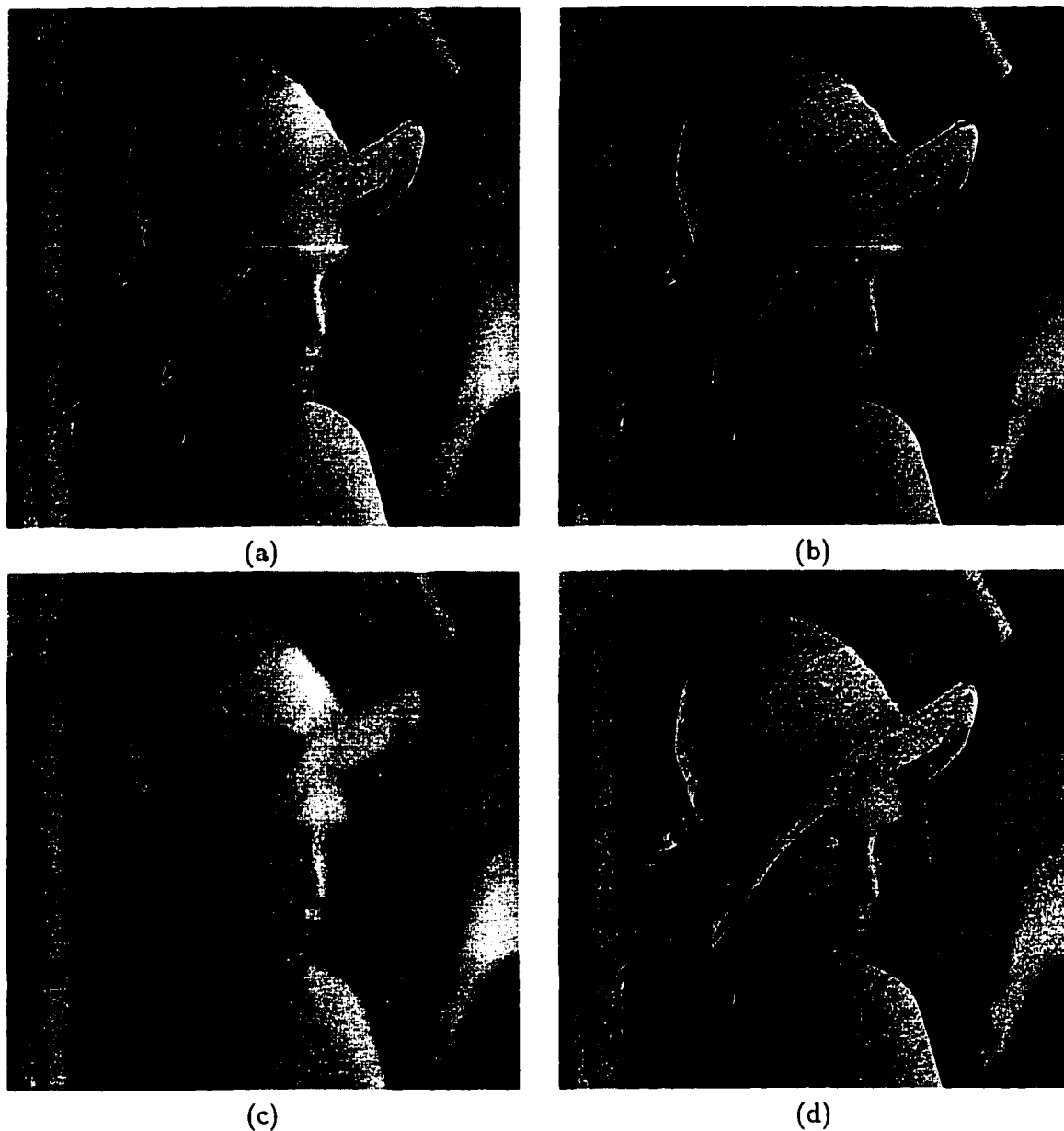


Figure 4.1: An example showing how PSNR and $RMSE$ are sometimes misleading when judging the actual loss of fidelity. (a) the original Lena image; (b) histogram flattening version of the Lena image ($PSNR = 26.29$ dB and $RMSE = 12.36$); (c) Gaussian blurred version of the Lena image using a mask of size 15×15 with a standard deviation = 2.992 ($PSNR = 26.29$ dB and $RMSE = 12.36$); and (d) noisy version of the Lena image using random Gaussian noise ($PSNR = 26.29$ dB and $RMSE = 12.36$).

lossy range. This makes the *PSNR*, as a measure of improvement, non-quantitative. For example, improving *RMSE* from 0.8 to 0.2 leads, virtually, to an invisible improvement and from 20 to 5 leads, visually, to a big significant improvement. However, the improvement of *PSNR* are the same in both cases and equals to 12.04 *dB* (i.e., from 50.07 *dB* to 62.11 *dB* or from 22.11 *dB* to 34.15 *dB*, respectively).

Since there is a lack of other alternative measures and based on the above arguments, the *RMSE* measure is used during this work to assess the reconstructed images degradation. However, when *RMSE* gets large, a subjective judgement should be considered, for example by visually examining the loss of fidelity in some samples of the reconstructed images¹.

Finally, the *Compression Ratios* (CRs) are calculated from the actual size of the compressed files, not entropy estimates, as follows:

$$CR = \frac{\text{image width} \times \text{image height}}{\text{actual compressed file size}} \quad (4.3)$$

4.2 Results

The validity of ABC-SC is demonstrated by compressing then decompressing several images, and comparing the input with the reconstructed images. Also other existing compression techniques are used in the comparison, including the *Joint Photographic Experts Group* (JPEG) [60] and the *Set Partitioning in Hierarchical Trees* (SPIHT) [27]. JPEG is a widely accepted industrial standard for continuous-tone natural scene image compression, whereas SPIHT is believed to be the state-of-the-art still-image lossy-compression method. While JPEG is a fixed block-size DCT-based compression technique, SPIHT

¹Since output devices' resolution is one of the major factors that influences image visual-quality, it is worth mentioning that all images in this work are printed on a 600 dot per inch HP laser jet printer.

is an embedded wavelet-based compression technique which is based on the *Embedded Zero-tree Wavelet approach* (EZW) [26].

The JPEG version that is used in this thesis is obtained via the internet from the *Independent JPEG Group*, the source code is available by anonymous FTP from

ftp.uu.net:/graphics/jpeg/jpegsrc.v6.tar.gz.

This JPEG version has two modes of operation: the baseline JPEG mode and the *Improved JPEG* (IJPEG) mode. The only difference between these two modes of operations is that the IJPEG employs a multiple-pass Huffman encoder to determine an optimal symbol code.

Similarly, the SPIHT version that is used in this thesis is obtained via the internet, where the source code is available by anonymous FTP from

ipl.rpi.edu:/pub/EW_Code/codetree.tar.gz.

SPIHT exists in slow and fast versions. The slow version (SPIHT-A) uses an *Arithmetic* encoder to improve compression and the fast version (SPIHT-B) produces a *Binary-uncoded* bit stream. Hence, the former version has the ability to compress images slightly more than that in the latter version.

During this work, ABC-SC is extensively tested and compared with JPEG, IJPEG², SPIHT-A and SPIHT-B. However, bear in mind that while testing against JPEG is fair enough (since both of ABC-SC and JPEG are DCT-based techniques), testing against SPIHT (the state-of-the-art) is a kind of an ultimate comparison.

From now on, the *TQR* parameter has been set to 1.00, unless otherwise specified. This means that each block is classified into one of two perceptual classes, i.e., either a

²In most image compression literature, researchers compare their compression results with JPEG results, but not with IJPEG results, even though the latter performance outperforms the former.

smooth or a non-smooth-block. The 512×512 Lena image³, shown in Figure 3.14(a), is chosen to demonstrate the results.

4.2.1 Two-class Case

Figure 4.2(a)–(c) shows the decompressed Lena images using ABC-SC at QF equals 147, IJPEG at QF equals 6, and JPEG at QF equals 2, respectively. While the compression ratios of these three reconstructed images are almost the same (62.47:1, 59.89:1, and 62.22:1, respectively), ABC-SC has the smallest *RMSE* reconstruction error (8.43, 9.86, and 20.42, respectively). From the subjective-quality point of view, the ABC-SC reconstructed image is much better than the IJPEG reconstructed image. This appears especially at smooth areas, where the blocking-effect is almost removed completely with ABC-SC. The ABC-SC reconstructed image is also far much better than the JPEG reconstructed image.

Figure 4.2(d) shows the rate-distortion curves for the Lena image using ABC-SC, IJPEG, and JPEG. As the curves indicate, ABC-SC outperforms both JPEG and IJPEG for all compression ratios under consideration. Moreover, ABC-SC can achieve high compression ratios—with a reasonable *RMSE* reconstruction error—which can not be achieved at all by either JPEG or IJPEG. In fact, for most of the testing images, ABC-SC resulted in at least twice the maximum compression ratio achieved with IJPEG and more than five times the maximum compression ratio achieved with JPEG.

Figure 4.3(a)–(b) shows the decompressed Lena images using ABC-SC at QF equals

³This 8 bpp grey-levels version of the Lena image is the *Y* luminance component of the original 24 bpp *RGB* (Red, Green, and Blue) Lena image, where the *Y* pixel values are obtained from the *RGB* pixel values using the following linear transform.

$$Y(i, j) = 0.299 \times R(i, j) + 0.587 \times G(i, j) + 0.114 \times B(i, j).$$

The other way of getting an 8 bpp grey-levels version from a 24 bpp *RGB* image is by using only the *G* component. However, the *G* component is darker and has slightly less fidelity than the *Y* component.

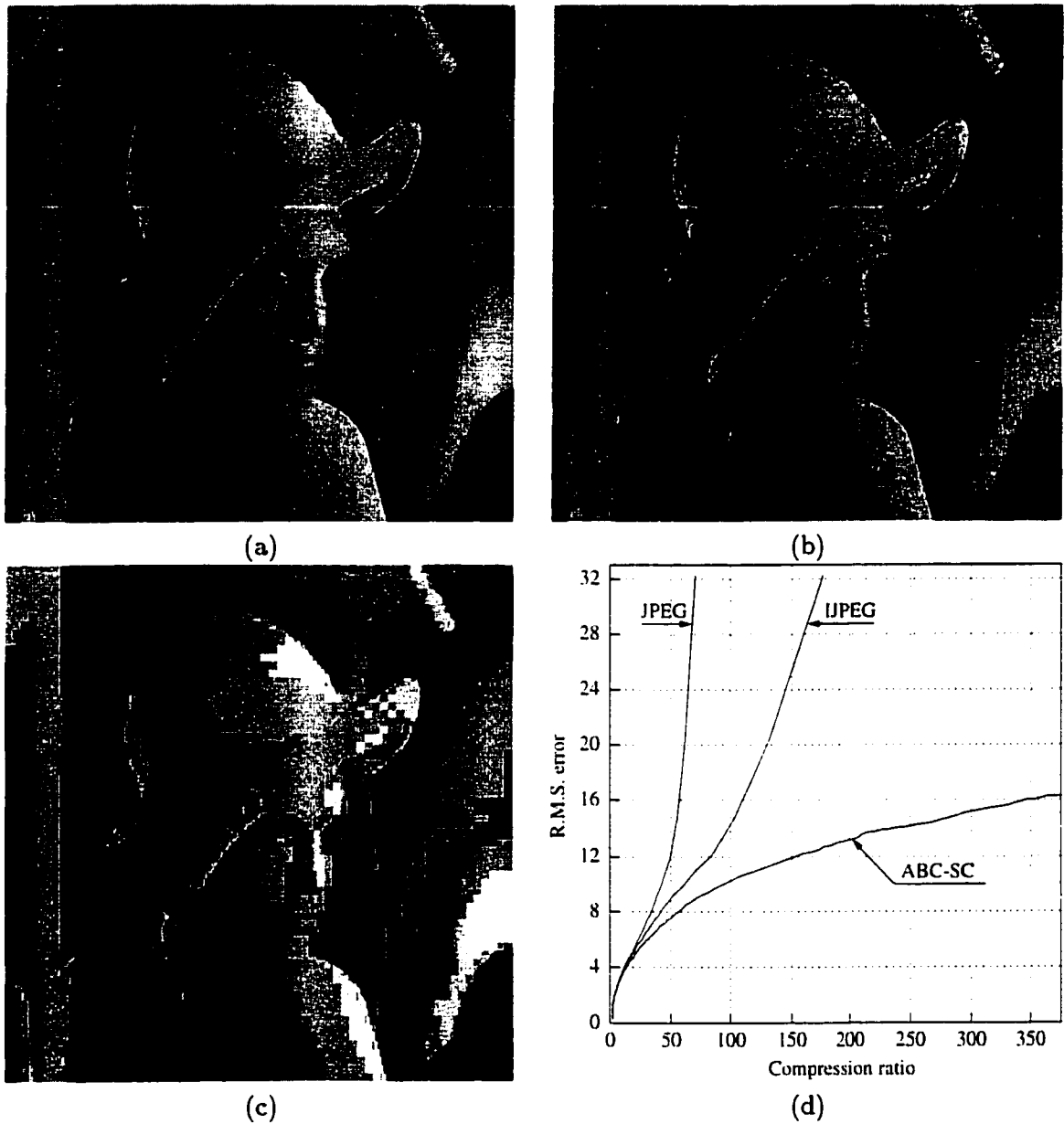


Figure 4.2: ABC-SC, IJPEG, and JPEG compression results for the Lena image. (a) ABC-SC ($QF = 147$, $TQR = 1.00$, $CR = 62.47$, and $RMSE = 8.43$); (b) IJPEG ($QF = 6$, $CR = 59.89$, and $RMSE = 9.86$); (c) JPEG ($QF = 2$, $CR = 62.22$, and $RMSE = 20.42$); and (d) rate-distortion curves.

184 and IJPEg at QF equals 11. The compression ratios are 36.81:1 and 36.69:1 and the $RMSE$ reconstruction errors are 6.50 and 7.39. From the subjective-quality point of view, both images are almost the same, except for some minor blocking-artifact in the flat areas in Figure 4.3(b), e.g., the Lena's cheek and shoulder areas.

Figure 4.3(c)–(d) shows the decompressed Lena images using ABC-SC at QF equals 89 and IJPEg at QF equals 1. The compression ratios are 177.01:1 and 176.05:1 and the $RMSE$ reconstruction errors are 12.66 and 32.23. From the subjective-quality point of view, the difference is clear. Note that this is the maximum compression ratio that IJPEg can achieve for the Lena image. Table 4.1 provides a summary of rate-distortion performance for the decompressed images shown in Figures 4.2 and 4.3.

Table 4.1: A summary of the rate-distortion performance for decompressed images shown in Figures 4.2 and 4.3.

Compression technique	Figure 4.2		Figure 4.3(a)–(b)		Figure 4.3(c)–(d)	
	CR	$RMSE$	CR	$RMSE$	CR	$RMSE$
ABC-SC	62.47	8.43	36.81	6.50	177.01	12.66
IJPEg	59.89	9.86	36.69	7.39	176.05	32.27
JPEg	62.22	20.42				

Figure 4.4 shows the rate-distortion performance of ABC-SC, IJPEg, JPEg, and a number of other segmentation-based compression techniques—which have been reported in recent compression literature (Section 2.8). As seen from the figure, ABC-SC outperforms all these techniques, except for the technique which was reported in [34]. Nevertheless, the improvement in the $RMSE$ reconstruction error is minor—less than 0.5—and takes place in the perceptually lossless range. This means that it is really hard to see any difference between the two reconstructed images, for example, compare Figure 4.3(a)

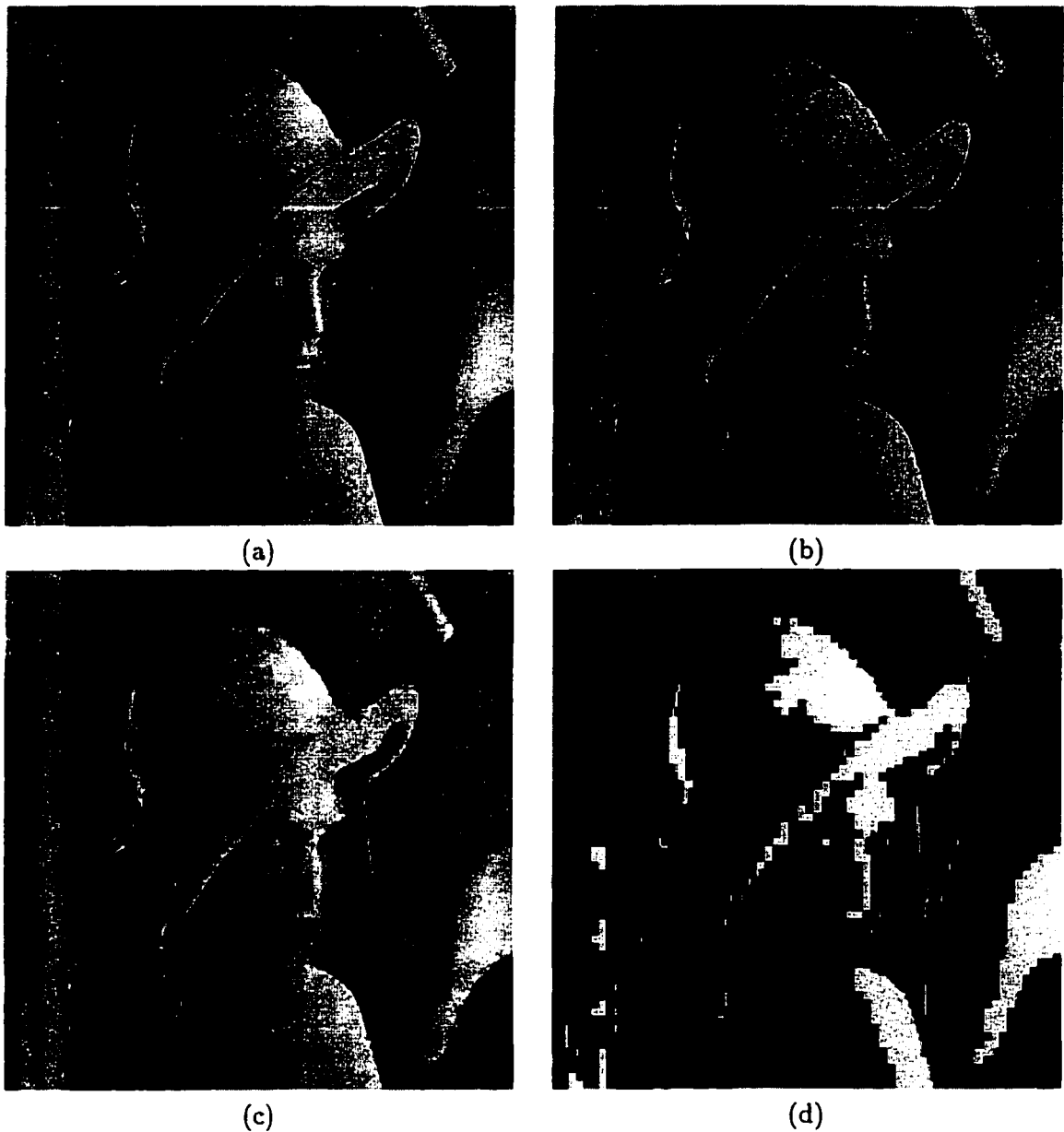


Figure 4.3: ABC-SC and IJPEg compression results for the Lena image. (a) ABC-SC ($QF = 184$, $TQR = 1.00$, $CR = 36.81$, and $RMSE = 6.50$); (b) IJPEg ($QF = 11$, $CR = 36.69$, and $RMSE = 7.39$); (c) ABC-SC ($QF = 89$, $TQR = 1.00$, $CR = 177.01$, and $RMSE = 12.66$); and (d) IJPEg ($QF = 1$, $CR = 176.05$, and $RMSE = 32.27$).

with Figure 9(d) in [34]. On the other hand, from the practical point of view, the technique in [34] is practically unacceptable, since it takes about 23 minutes of CPU time to decompose just a 256×256 image. The ABC-SC average execution time on a comparable task is less than 1 second (see Section 4.2.7).

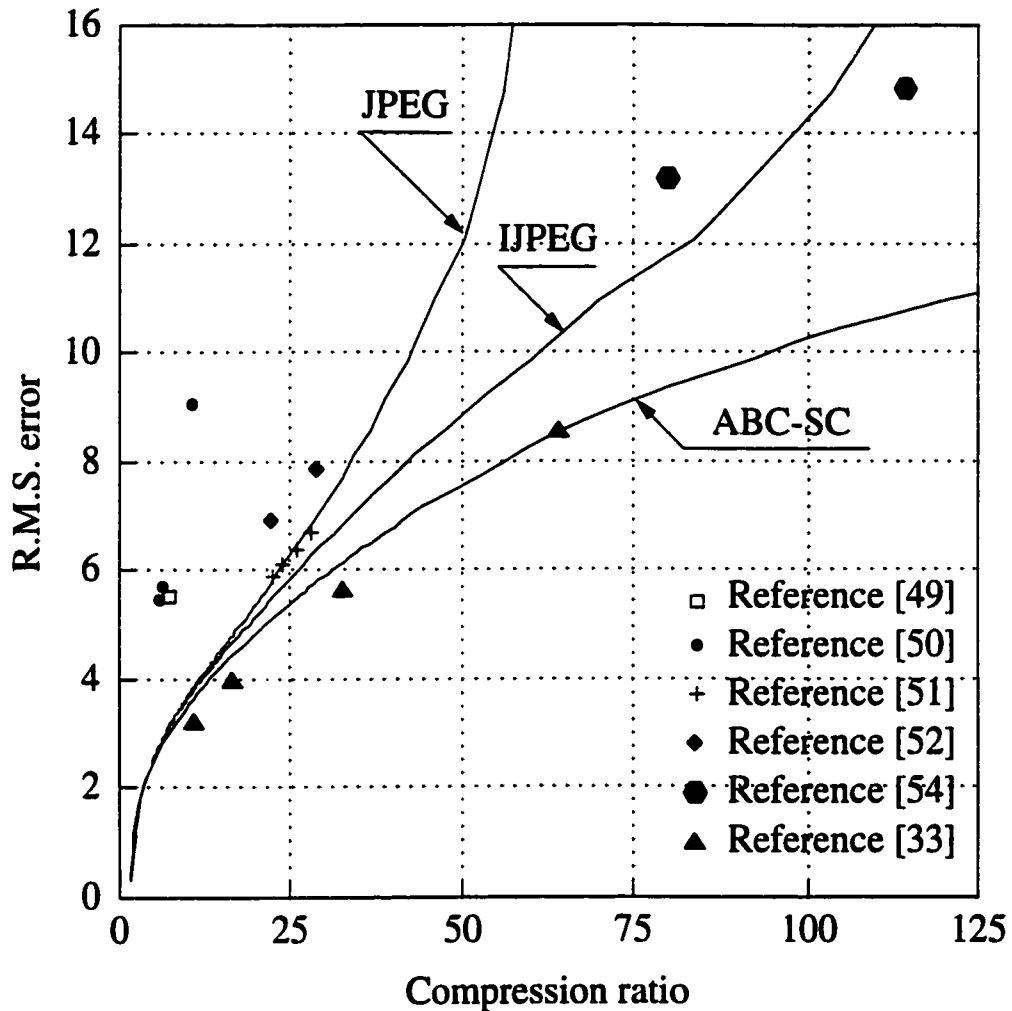


Figure 4.4: Comparison of rate distortion results between ABC-SC, IJPEG, JPEG, and some other recent segmentation based encoding techniques for the Lena image.

Figure 4.5(a)–(c) shows the decompressed Lena images using ABC-SC at QF equals 32, SPIHT-A, and SPIHT-B, respectively. While the compression ratio of any of these

three reconstructed images is 235.11, the *RMSE* reconstruction errors are 13.97, 12.29, and 12.95, respectively. This does not represent a major differences. From the subjective-quality point of view, while ABC-SC start suffering from the blocking-artifact, both SPIHT-A and SPIHT-B are suffering from the ringing-effect—which is a well-known artifact in sub-band encoders in general, especially near strong edges. However, in ABC-SC, there is always a chance for restoring the blocking artifact—since the exact deformity locations are known. At the same time, in SPIHT-A and SPIHT-B, there is no chance at all to restore ringing-effect.

Figure 4.5(d) shows the rate-distortion curves for the Lena image using ABC-SC, SPIHT-A, and SPIHT-B. The figure shows that the ABC-SC rate-distortion curve is following both SPIHT-A and SPIHT-B rate-distortion curves, while maintaining a minor increase in the *RMSE* reconstruction error.

Figures 4.6 and 4.7 show another two sets of reconstructed Lena images using ABC-SC, SPIHT-A, and SPIHT-B, where the compression ratios are equal to 91.95 and 374.49, respectively. From the subjective-quality point of view, there is no major difference between images in the first set. However, the blocking and the ringing-artifacts are obvious in the second set of reconstructed images. Table 4.2 provides a summary of rate-distortion performance for the decompressed images shown in Figures 4.5–4.7.

Table 4.2: A summary of the rate-distortion performance for decompressed images shown in Figures 4.5–4.7.

Compression technique	Figure 4.5		Figure 4.6		Figure 4.7	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	235.11	13.97	91.95	9.90	374.49	16.34
SPIHT-A	235.11	12.29	91.95	8.37	374.49	14.70
SPIHT-B	235.11	12.95	91.95	8.83	374.49	15.12

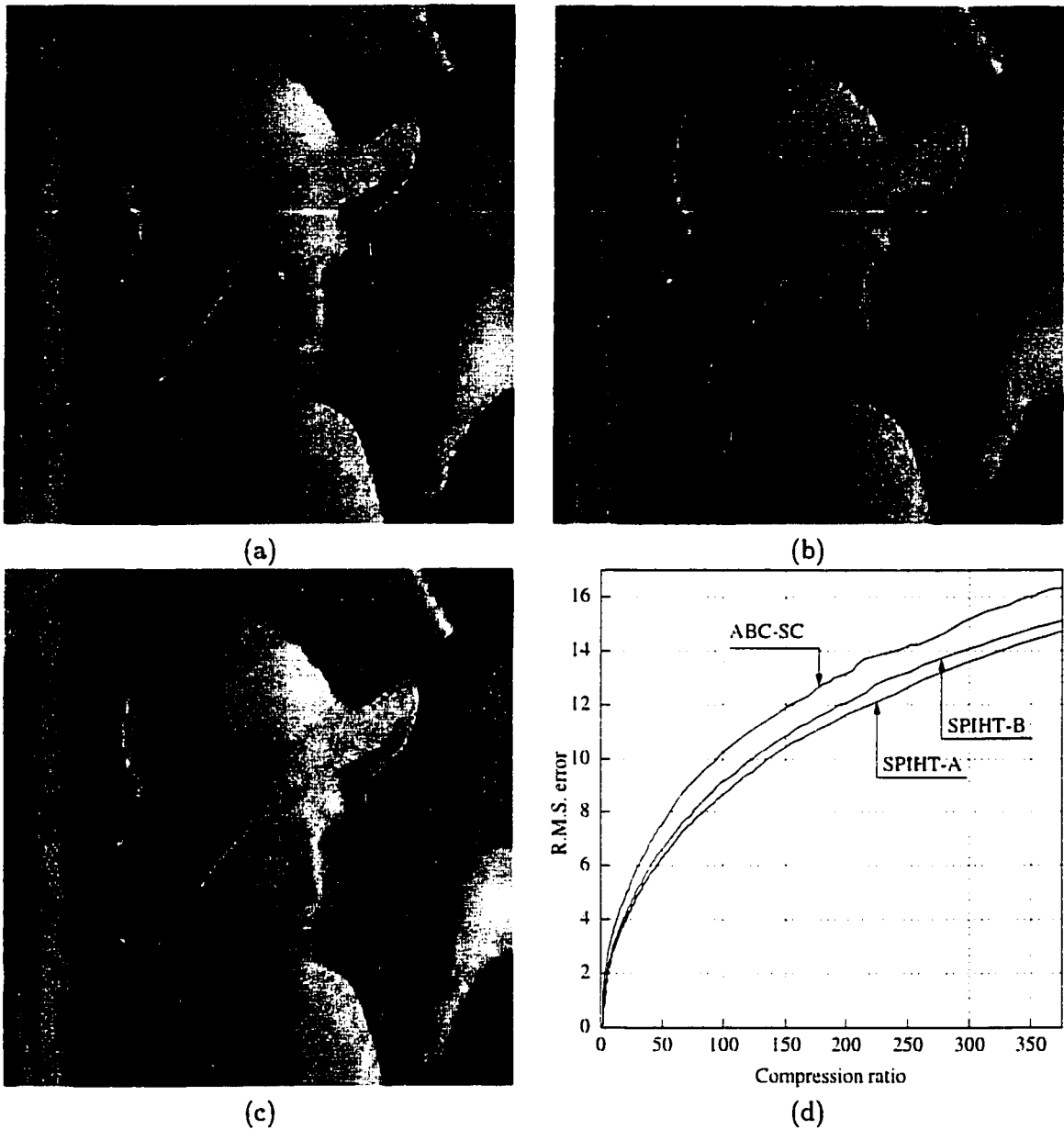


Figure 4.5: ABC-SC, SPIHT-A, and SPIHT-B compression results I for the Lena image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 235.11$, and $RMSE = 13.97$); (b) SPIHT-A ($CR = 235.11$, and $RMSE = 12.29$); (c) SPIHT-B ($CR = 235.11$, and $RMSE = 12.95$); and (d) rate-distortion curves.

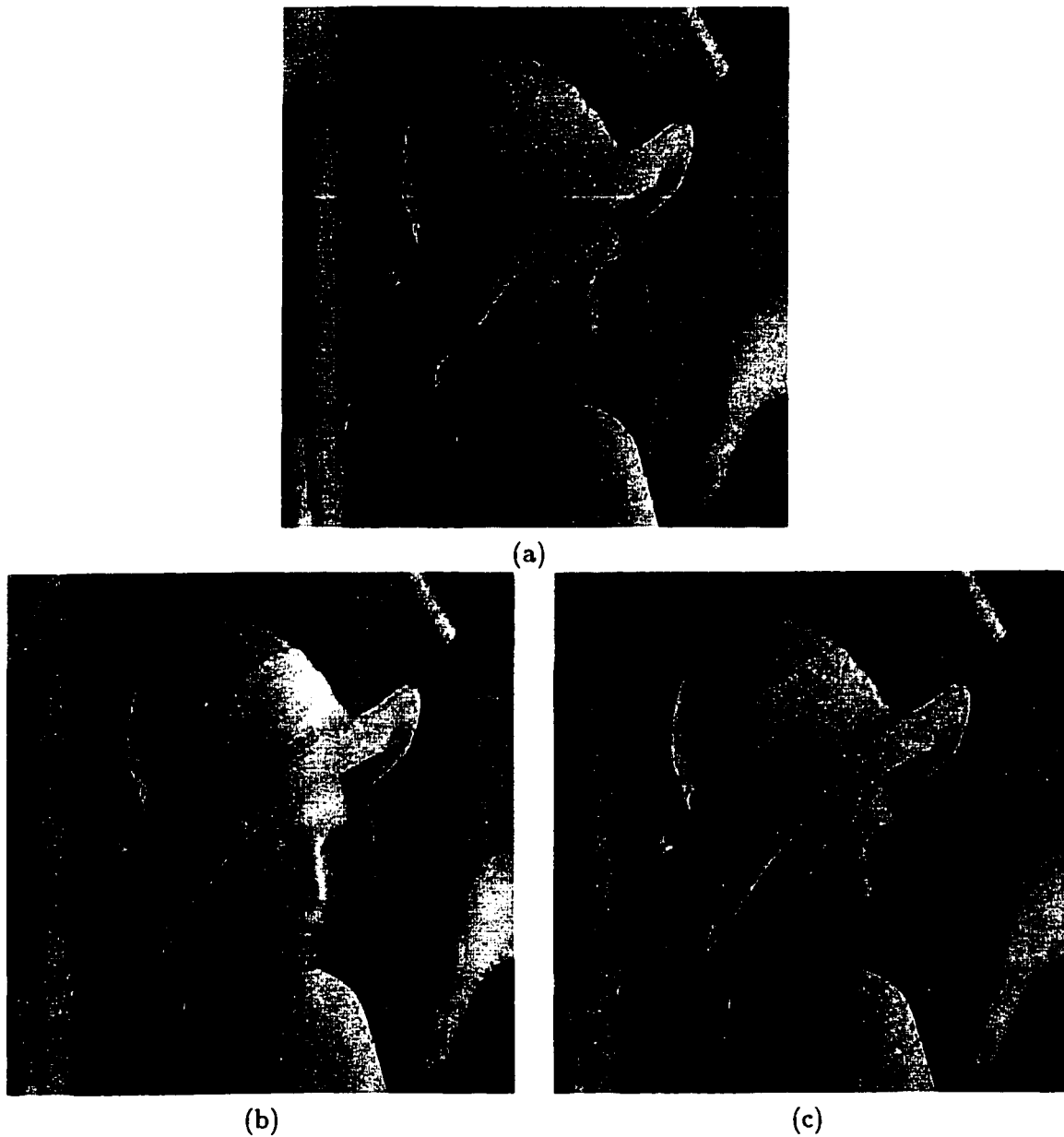


Figure 4.6: ABC-SC, SPIHT-A, and SPIHT-B compression results II for the Lena image. (a) ABC-SC ($QF = 128$, $TQR = 1.00$, $CR = 91.95$, and $RMSE = 9.90$); (b) SPIHT-A ($CR = 91.95$, and $RMSE = 8.37$); and (c) SPIHT-B ($CR = 91.95$, and $RMSE = 8.83$).

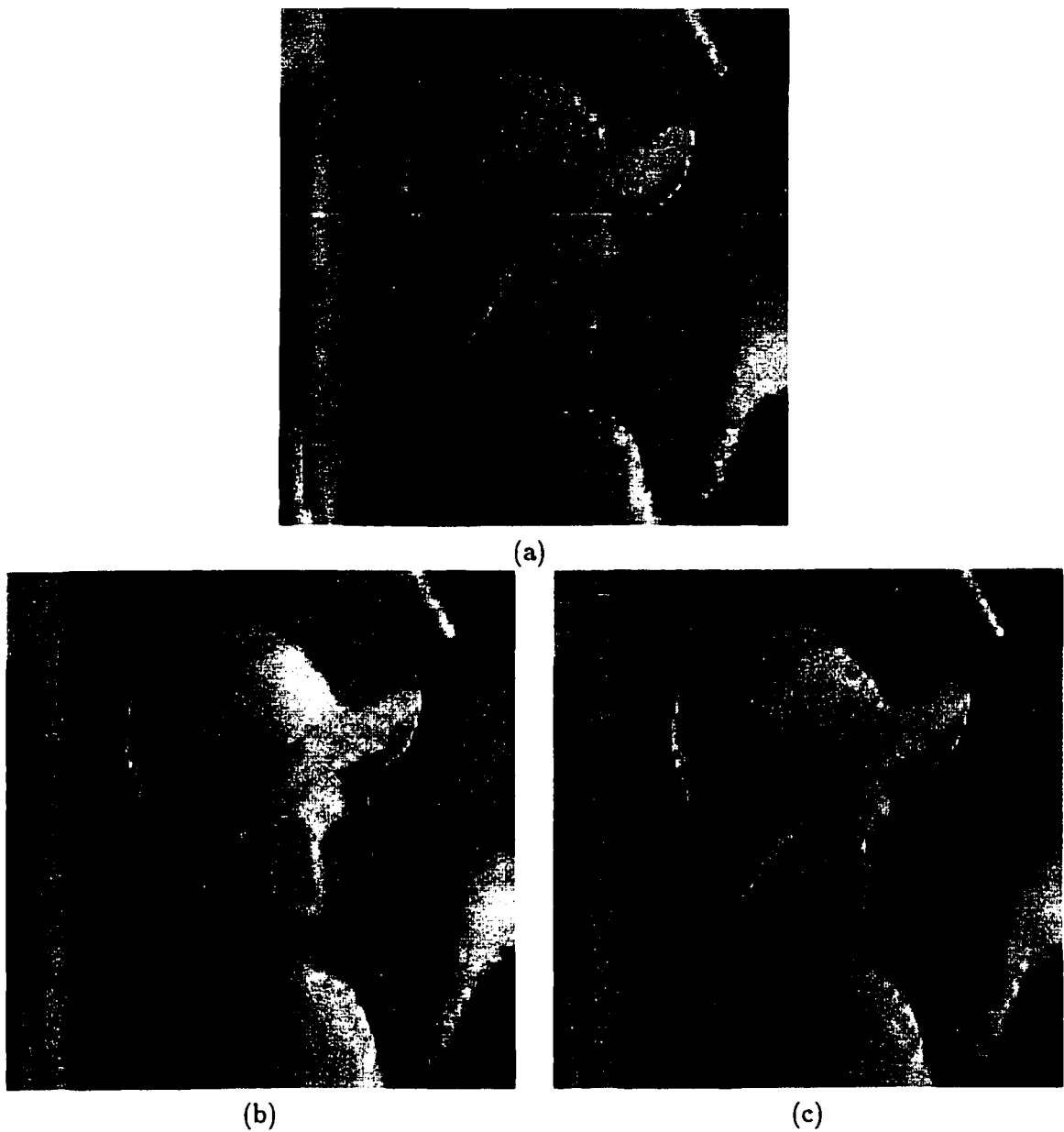


Figure 4.7: ABC-SC, SPIHT-A, and SPIHT-B compression results III for the Lena image. (a) ABC-SC ($QF = 1$, $TQR = 1.00$, $CR = 374.49$, and $RMSE = 16.34$), (b) SPIHT-A ($CR = 374.49$, and $RMSE = 14.70$), and (c) SPIHT-B ($CR = 374.49$, and $RMSE = 15.12$).

According to Figures 4.5–4.7, we believe that although ABC-SC is maintaining a minor increase in the *RMSE* reconstruction error, it does not really make a major difference in the subjective-quality of the reconstructed images.

For more comparative results with JPEG/IJPEG and SPIHT-A/SPIHT-B, see Appendix A and Appendix B.

4.2.2 Three-class Case

Figure 4.8(a)–(b) shows two reconstructed Lena images using ABC-SC at QF equals 200, where the TQR parameter is set to 1.0 and 0.3. Although the difference between the *RMSE* reconstruction errors in these two images is 1.28—between 6.11 and 7.38—, it is hard to see any major difference between them. This is because all the differences occur in textural areas which convey low amount of information to the viewer. Hence, degradation in these areas is less noticeable. To clarify the locations of these differences, Figure 4.8(c)–(d)⁴ shows the absolute differences between images in Figure 4.8(a)–(b), respectively, and the original Lena image, shown in Figure 3.14(a). Note that most of the differences between these two error images are located in the feather area, which coincides with the location of the Lena image textural segment shown in Figure 3.14(c).

Interestingly, this increase in the *RMSE* reconstruction error is associated with a decrease in the compressed image file-size—the compressed file-size of the image shown in Figure 4.8(b) is reduced by 16% compared to the compressed file-size of the image shown in Figure 4.8(a), i.e., the compression ratio increased from 32.05:1 to 38.14:1.

Figure 4.9 demonstrates the amount of reduction in the compressed Lena image file-size—due to setting the TQR parameter to 0.3—for different QF values. Figure 4.9(a) shows the compressed file sizes when TQR is set to 1.00 (i.e., two-class case). Figure 4.9(b)

⁴For the sake of improving the fidelity, each pixel in these error images is multiplied by 8 before printing.

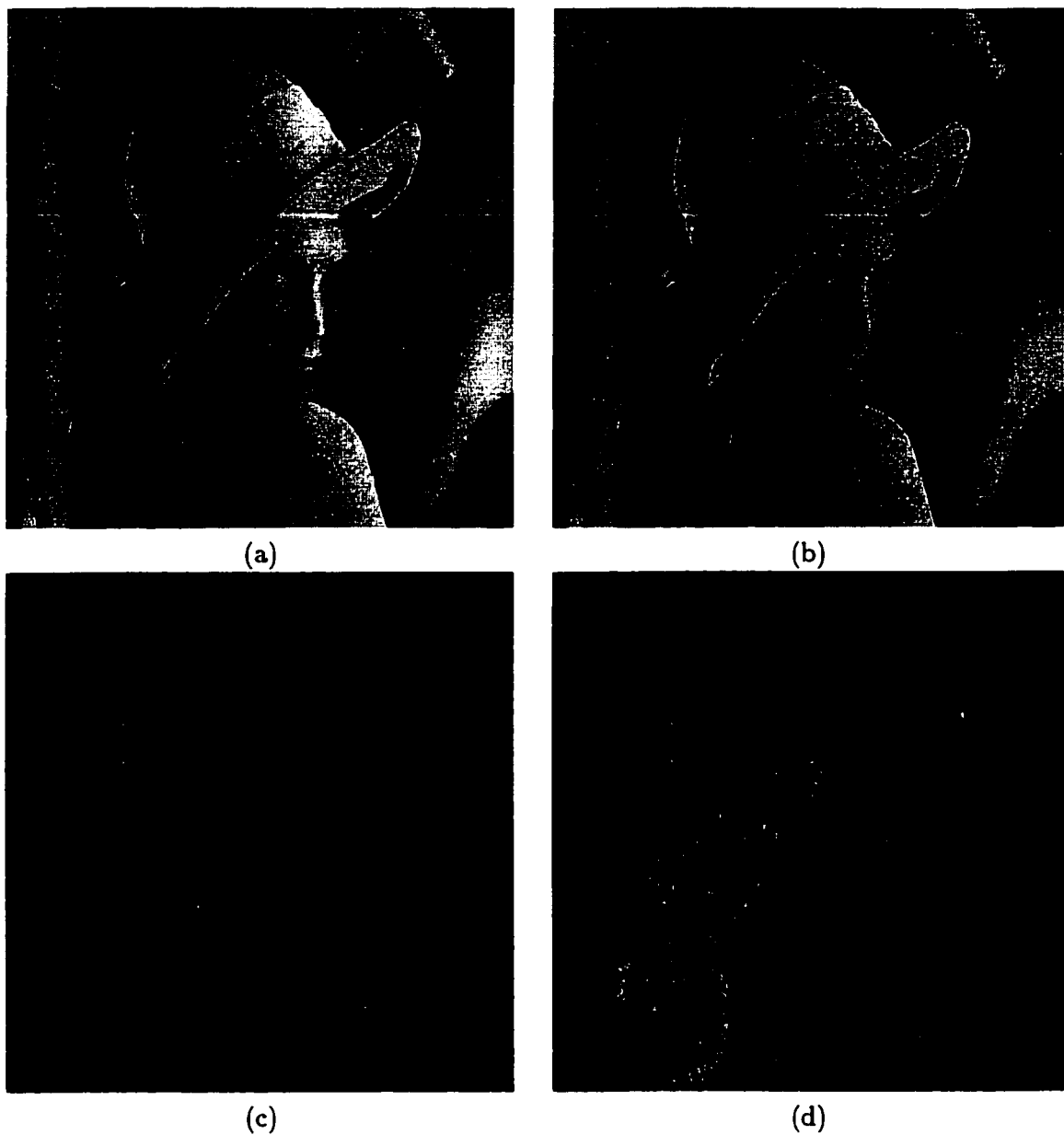


Figure 4.8: The effect of texture-quality-factor on the performance of ABC-SC for the Lena image. (a) $QF = 200$, $TQR = 1.00$, $CR = 32.05$, and $RMSE = 6.11$; (b) $QF = 200$, $TQR = 0.30$, $CR = 38.14$, and $RMSE = 7.38$; (c) and (d) the absolute differences between the original Lena image and the images in (a) and (b), respectively.

shows the compressed file sizes—normalized to the file-sizes in Figure 4.9(a)—when TQR is set to 0.3 (i.e., three-class case). This amount of reduction in the compressed Lena image file-size can be as high as 18.64% compared to the two-class case (i.e., increasing in the compression ratio from 63.47:1 to 78.02:1), when the QF is set to 146.

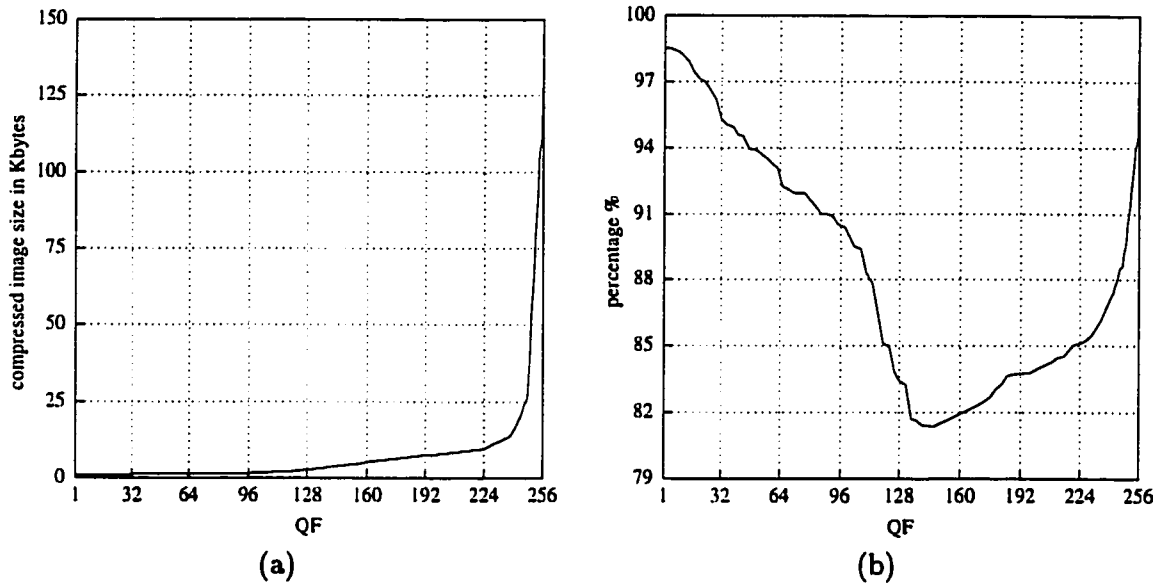


Figure 4.9: The compressed file-sizes for the Lena image. (a) two-class case and (b) three-class case at $TQR = 0.30$, normalized to the two-class case.

Figure 4.10(a)–(c) shows three reconstructed Barbara images using ABC-SC at QF equals 137, where the TQR parameter is set to 1.0, 0.5, and 2.0, respectively. The compression ratios are 54.53:1, 73.45:1, and 34.54:1, respectively, and the $RMSE$ reconstruction errors are 14.90, 16.43, and 12.63, respectively. Figure 4.10(d) shows the textural segment for the Barbara image, shown in Figure 3.11(d). The figure shows that when the TQR parameter is set to a value less than 1.0, more compression is achieved with a higher $RMSE$ reconstruction error. However, most of these errors occur in a non-noticeable areas (e.g., in the stripes of Barbara clothes). On the other hand, when it is set to a value greater than 1.0, less compression is achieved with a lower $RMSE$

reconstruction error, where most of the improvement occurs in the textural areas. Note that the appropriate TQR value is left to the user to set it according to the application of interest.

4.2.3 ADPCM Effect

Figure 4.11 shows the rate-distortion curves for the Lena image using three different versions of ABC-SC. The only difference between these three versions is the DC-coefficient predictor. In the first version, the ADPCM predictor—introduced in Section 3.1.3—is used. In the second version, the JPEG/IJPEG DC-coefficient predictor is used—where the predicted value is always the previous DC-coefficient value of the block to the left of the current block. Finally, in the third version, there is no prediction at all. The figure shows the superiority of the ADPCM predictor version over the two other versions. We believe that this superiority justifies the additional complexity introduced by the ADPCM predictor, which is an insignificant extra complexity after all.

4.2.4 Lossless Encoder Effect

As an alternative to the arithmetic encoder, we have also considered the Huffman encoder. Figure 4.12 shows the rate-distortion curves for the Lena image using two different versions of ABC-SC. The only difference between these two versions is the lossless encoder used. While the first version uses the arithmetic encoder mentioned in Section 3.1.8, a two-pass Huffman encoder is used in the second version. The figure shows a minor rate-distortion improvement when the arithmetic encoder is used. Note that the Huffman encoder is considered to be an optimal lossless encoder only when all the symbols' probabilities are governed by the 2^{-n} formula, where n is any positive integer number.

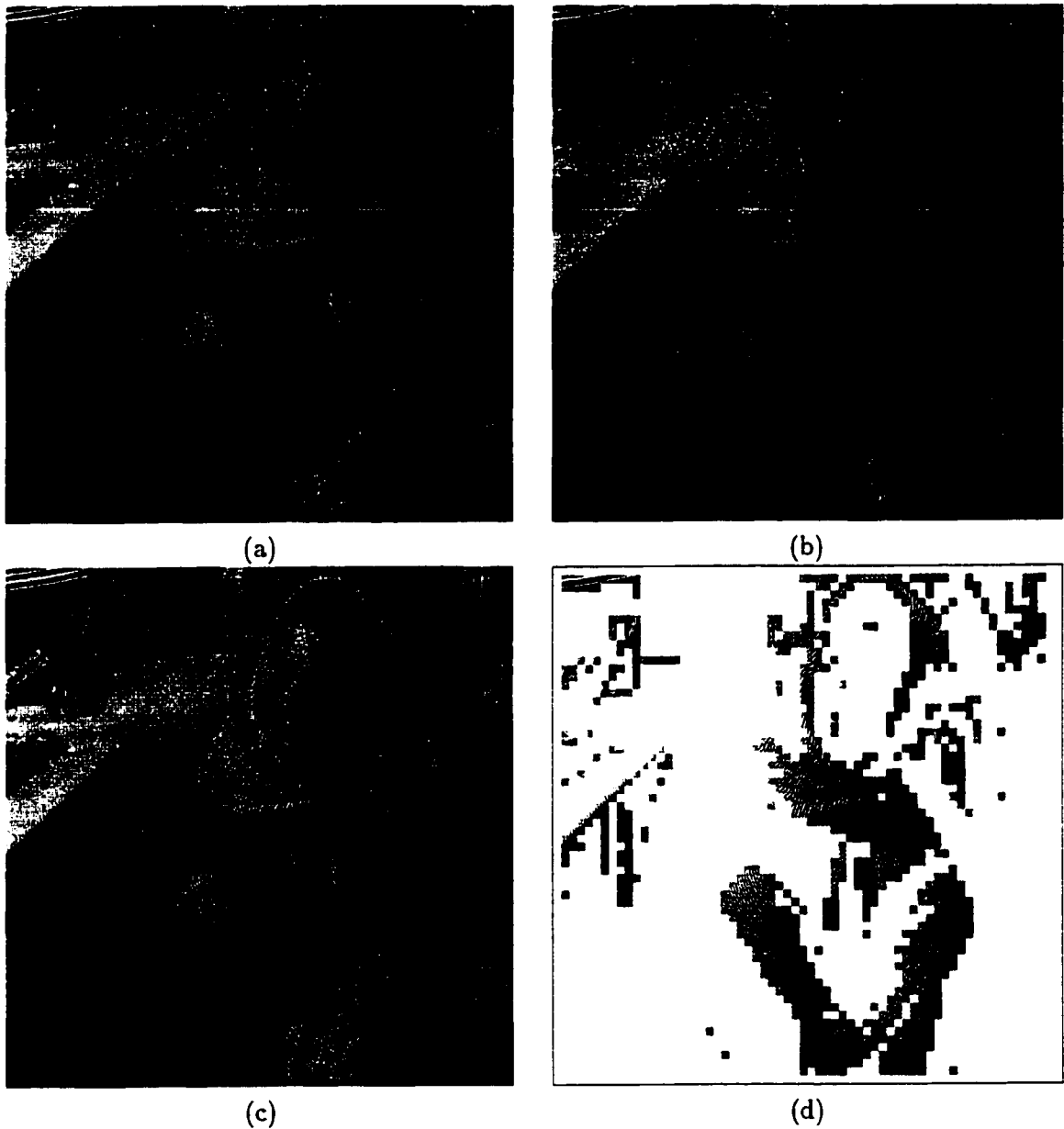


Figure 4.10: The effect of texture-quality-factor on the performance of ABC-SC for the Barbara image. (a) $QF = 137$, $TQR = 1.00$, $CR = 54.53$, and $RMSE = 14.90$; (b) $QF = 137$, $TQR = 0.50$, $CR = 73.45$, and $RMSE = 16.43$; (c) $QF = 137$, $TQR = 2.00$, $CR = 34.54$, and $RMSE = 12.63$; and (d) textural segment image.

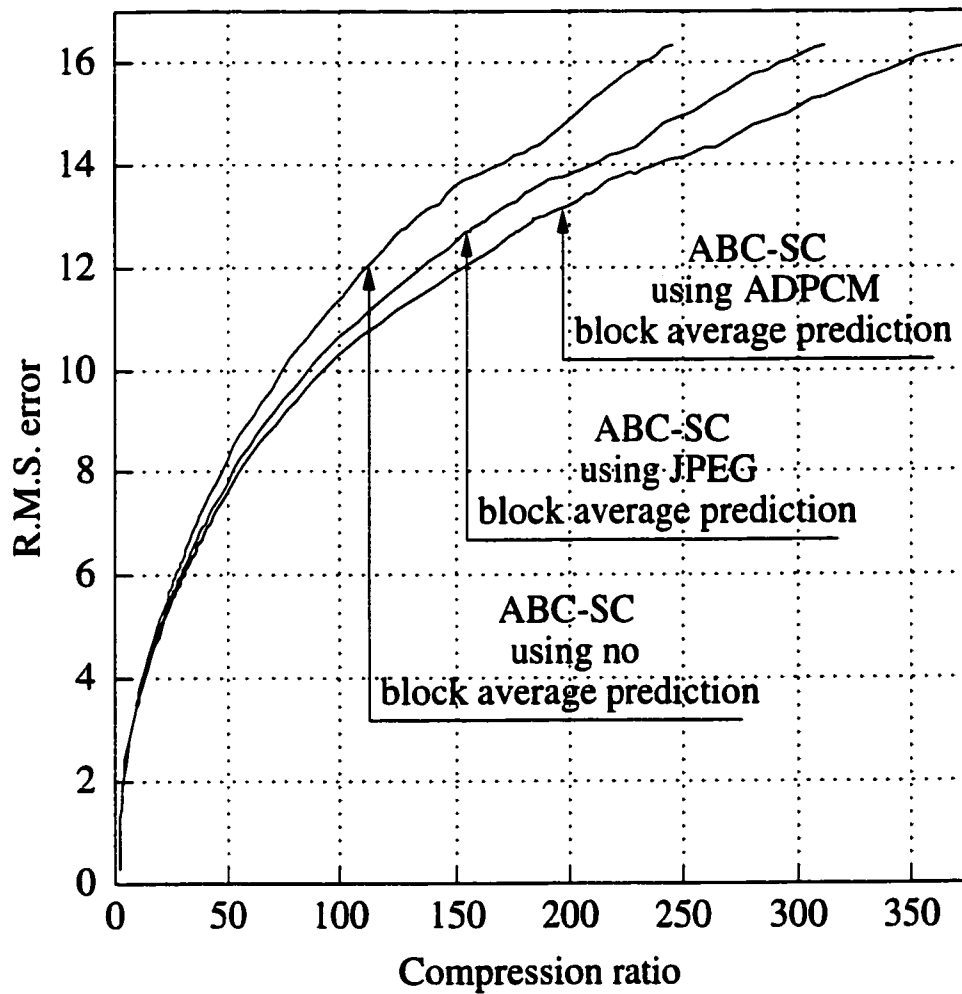


Figure 4.11: The ADPCM effect for the Lena image.

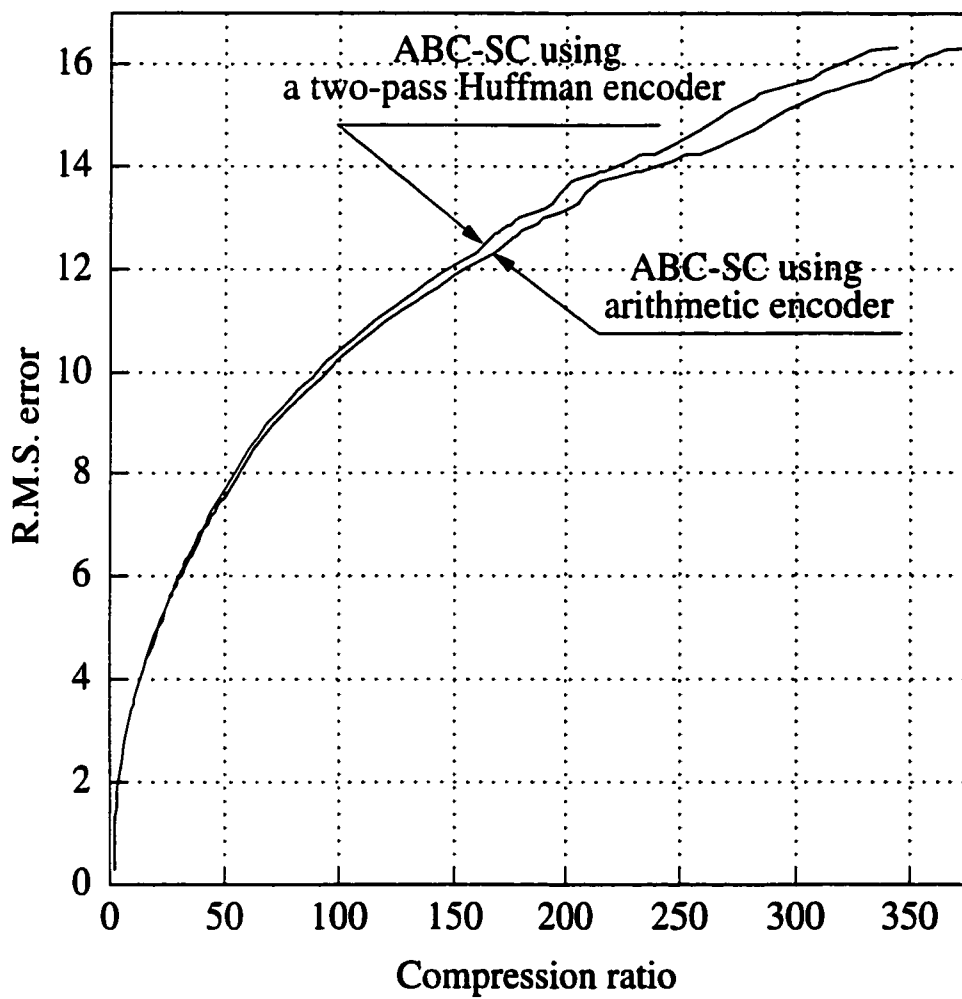


Figure 4.12: The effect of the lossless encoder on the performance of ABC-SC for the Lena image.

4.2.5 Post-processing Effect

Figure 4.13(a)–(b) shows two decompressed Lena images using ABC-SC before and after post processing. While the compression ratios of this two images are the same and equal to 105.07:1, the *RMSE* reconstruction error is improved by 1.15—from 11.61 to 10.46. Figure 4.13(c)⁵ shows the absolute difference between images in Figures 4.13(a) and 4.13(b), which is basically the blocking-artifact locations. This means that most of the improvement in the *RMSE* reconstruction error is due to the block-artifact removal.

4.2.6 Quad-tree Overhead

Figure 4.14 shows the quad-tree overhead in ABC-SC for the Lena image at *TQR* equals 1.00 (i.e., two-class case) and 0.3 (i.e., three-class case). Figure 4.14(a) shows the quad-tree overhead in bytes and Figure 4.14(b) shows the quad-tree overhead normalized to the compressed image file-size. The figure shows that the maximum quad-tree overhead is as small as 444 and 587 bytes for the two- and three-class cases at *QF* equals 233, respectively. When the *QF* increases, the quad-tree overhead increases until it reaches its maximum value. However, its percentage share in the compressed image file is decreasing from about quarter of the whole compressed image file to almost zero. This means that, at high compression ratios, the cost of determining the positions of the few significant coefficients represents a significant portion of the compressed-image file-size. As the compression ratio decreases, further coefficients are considered. Consequently, the cost of determining their locations is increased. However, this cost is negligible relative to the coefficient encoding cost.

⁵For the sake of improving the fidelity, each pixel in this error image is multiplied by 8 before printing.

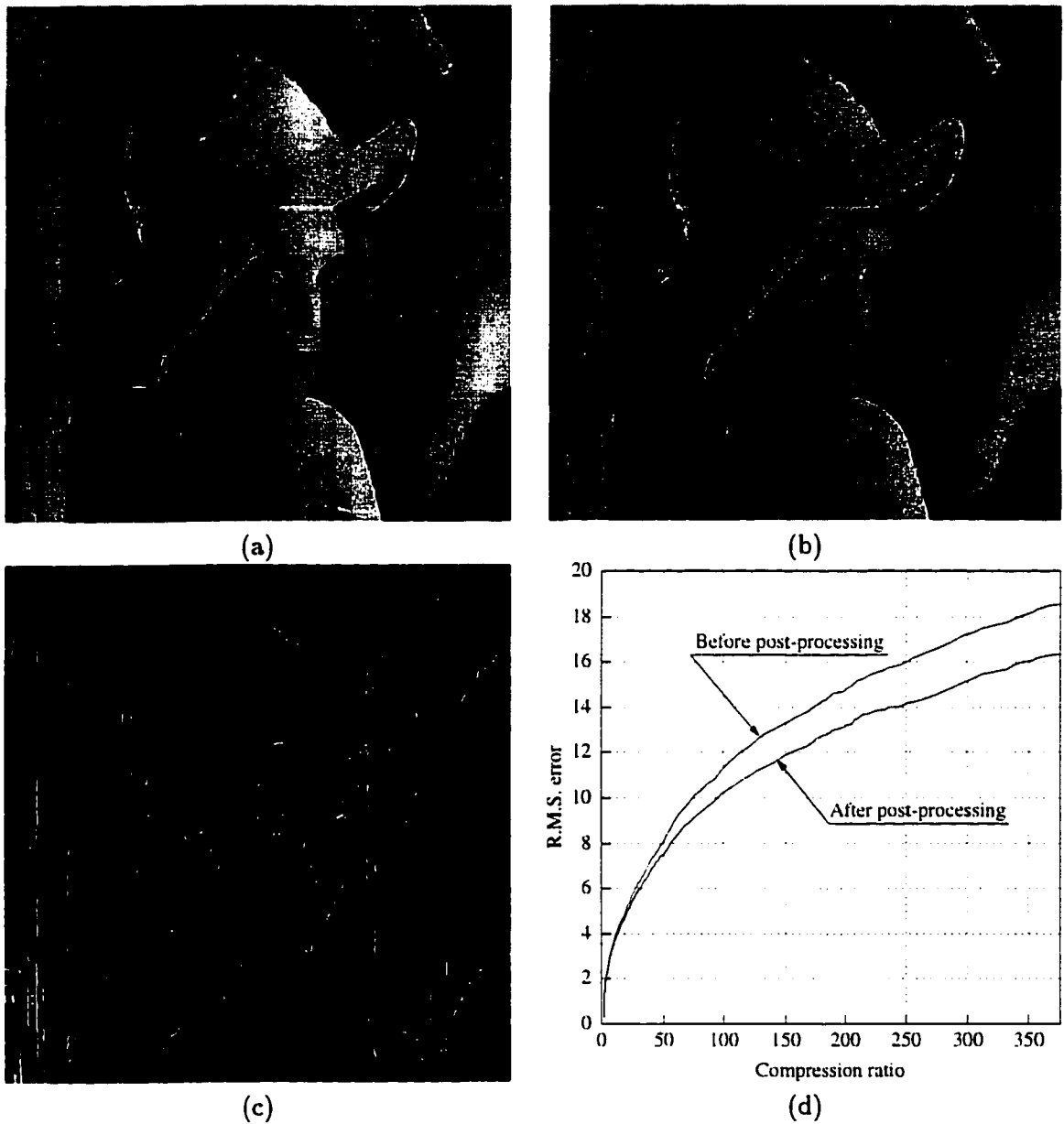


Figure 4.13: Enhancement results for the Lena image due to the post-processing ($QF = 122$, $TQR = 1.00$, and $CR = 105.07$). (a) before post-processing ($RMSE = 11.61$); (b) after post-processing ($RMSE = 10.46$); (c) absolute difference between images (a) and (b); and (d) rate-distortion curves before and after post-processing.

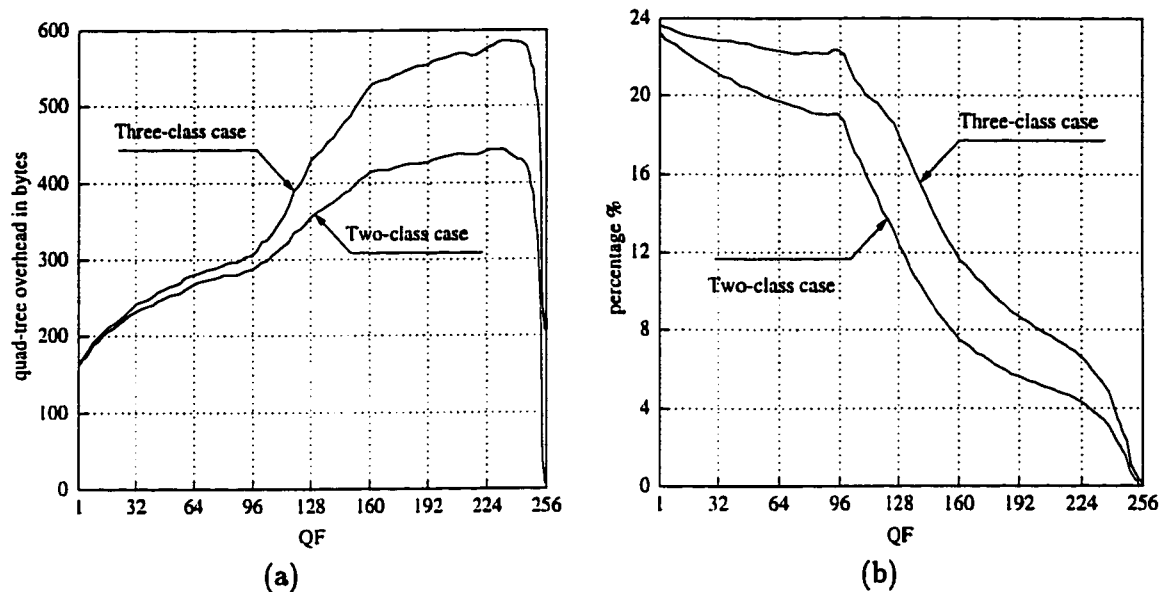


Figure 4.14: The quad-tree overhead in ABC-SC for the Lena image at $TQR = 1.00$ (i.e., two-class case) and at $TQR = 0.30$ (i.e., three-class case). (a) overhead in bytes and (b) overhead normalized to the compressed file-size.

4.2.7 Execution Time

Table 4.3 shows the average amount of time in seconds required to execute ABC-SC at TQR equals 1.00 and 0.3, SPIHT-A, SPIHT-B, JPEG, and IJPEG on a SUN Ultra 1 computer for the Lena image. These averages are calculated based on actual runs over all QF values, in case of ABC-SC, JPEG, and IJPEG; and equivalent compression ratios, in case of SPIHT-A and SPIHT-B.

From the table, we can conclude the following:

1. The ABC-SC compression/decompression time falls between the SPIHT-A and SPIHT-B compression/decompression times.
2. The increase in the average compression time due to dealing with three classes, instead of two classes, is minor (i.e., 0.151 seconds). Meanwhile, the average decompression time decreased by 0.031 seconds.

Table 4.3: The average compression/decompression time (in seconds) for the Lena image.

Average execution time	ABC-SC		SPIHT-A	SPIHT-B	JPEG	IJPEG
	$TQR = 1.0$	$TQR = 0.3$				
Compression	0.919	1.070	1.253	0.763	0.109	0.122
Decompression	0.846	0.815	1.051	0.532	0.084	0.081
Enhancement	0.414	0.414				

3. The enhancement time is almost half the decompression time. However, this time always can be waived, especially at low compression ratios, where the blocking-artifact is non-noticeable.
4. The execution time of JPEG/IJPEG is, dramatically, less by an order of magnitude, than the execution time of the rest of the techniques.

Finally, it is worth mentioning that we have not made a major effort to reduce the complexity or optimize the implementation of ABC-SC. In fact, we believe that ABC-SC could be further optimized and we anticipate that its execution time could be reduced to a value in the vicinity of JPEG/IJPEG execution time, since both ABC-SC and JPEG/IJPEG are DCT-based techniques. However, this needs further investigation to support this believe.

4.3 Discussion

Based on the results reported in Section 4.2, it is clear that, in spite of the fact that ABC-SC and the JPEG/IJPEG are both DCT-based techniques, the former consistently outperforms the latter not only from the rate-distortion point of view, but also from the subjective-quality point of view (i.e., having sharper edges with smoother outlook

reconstructed images). This is true especially at high compression ratios. This is because ABC-SC attempts to distribute the degradation over the entire image taking into account the relative importance of the different image regions. This is accomplished by compressing each region according to the amount of information it conveys to the viewer. On the other hand, JPEG/IJPEG treats different image regions in the same manner, and hence, distributes the degradation uniformly over the entire image.

ABC-SC moves block-based compression beyond the limits of JPEG/IJPEG. In fact, for most testing images, the maximum compression ratio that can be achieved with ABC-SC is, at least, twice that of IJPEG and five times that of JPEG, with even less objective and subjective degradations.

For all the testing images, ABC-SC produced a performance which is comparable to the state-of-the-art wavelet compression technique from both the rate-distortion and the subjective-quality points of view.

The execution time of ABC-SC is somewhere between the execution time of SPIHT-A and SPIHT-B. However, we have not made a major effort yet to reduce the complexity or optimize the implementation of ABC-SC. In fact, ABC-SC is a similar technique to the JPEG/IJPEG technique and the latter execution time is less, by an order of magnitude, than the execution time of ABC-SC, SPIHT-A, and SPIHT-B. Therefore, we believe that ABC-SC has the potential to be further optimized and its execution time can be dropped by an order of magnitude, or so, too.

ABC-SC can be considered as a multi-adaptive compression technique, where the adaptability is achieved through the following features:

1. adaptive block-size determination, where a block-size is determined according to the information and the activities within blocks;
2. adaptive multi-block-encoding techniques, where the encoding technique is deter-

- mined according to the information and the contents of blocks;
3. adaptive block-average prediction, where a prediction rule is chosen based on some encoded block average local statistics;
 4. adaptive arithmetic encoding, where the adaptive probability model is updated based only on the already encoded codewords; and
 5. adaptive post-processing, where the post-processing scheme used for a given block is chosen based on its content and the contents of its surrounding blocks.

The adaptability of ABC-SC gives at least two advantages:

1. better consistent image quality, and
2. low average bit-rate for a wide variety of images.

Although the ABC-SC uses a simple segmentation criterion, which is sufficient to identify the different block classes, it is not sensitive to errors that might occur in the segmentation process.

While the current mechanism for scaling the quantization matrix in JPEG/IJPEG is known not to be very good at low QF values, ABC-SC introduces another scaling mechanism which exhibits an excellent behavior at all QF values. This excellent behavior of the proposed quantization mechanism—which is a group of non-linear scaling functions of the QF —may be attributed to:

1. the distinction between the quantization of each perceptual class of blocks, and
2. the distinction between the quantization of the DC- and the AC-coefficients.

Hence, ABC-SC can achieve very high compression ratios while maintaining a reasonable reconstructed image fidelity. In fact, it can be stated that ABC-SC shows the capabilities of the DCT transform more than JPEG/IJPEG.

ABC-SC is one of very few of compression techniques which give users the ability to adjust and control the quality of the reconstructed textural areas—independent from the rest of the reconstructed image areas. This feature gives ABC-SC further flexibility and controllability more than other compression techniques which deal with images, globally, as a whole, e.g. JPEG/IJPEG and SPIHT-A/SPIHT-B.

It should be noted that:

1. there is no training of any kind and no ensemble statistics of images are required (except the parameters adjustment which might be considered as a sort of ensemble statistic);
2. in ABC-SC, as the QF decreases, the degradation gradually increases, rather than making a sudden drop in quality;
3. the significance of the variable block-size is more noticeable at high compression ratios, whereas it is almost negligible at low compression ratios;
4. while the ABC-SC parameters have been adjusted based on minimizing the *RMSE* reconstruction errors, however, it had an insight about the locations of the useful, the irrelevant, and the redundant information;
5. although ABC-SC parameters have been adjusted based on some general images, these parameters are still valid over wider classes of images; and
6. since ABC-SC is a memoryless block technique, it is amenable to parallel processing. Moreover, the DCT transform can be computed using fast algorithms and efficiently implemented using Very Large Scale Integration (VLSI) techniques [75, 76, 77, 78].

Chapter 5

Concluding Remarks and Future Work

This thesis began by showing the motivation for this work. After providing a review of image compression techniques, the proposed compression technique was introduced and described. The previous chapter showed the results achieved with the proposed compression technique, as well as comparisons with other compression techniques. This final chapter has two sections. In the first section, conclusions based on observations made throughout this thesis, and on the previously reported results, are presented. In the second and final section of this chapter, recommendations to further improve the proposed compression technique are listed.

5.1 Conclusions

In this thesis, a new adaptive compression technique is proposed. Adaptability has been incorporated into many aspects of this technique, including adaptive block-size determination, adaptive multi-block-encoding techniques, adaptive block-average prediction,

adaptive arithmetic encoding, and adaptive post-processing. The proposed compression technique also exploits one of the HVS properties, which is, recognizing images by their regions, to get high compression ratios. Based on extensive test results and comparisons with other existing compression techniques, the following conclusions can be made:

1. ABC-SC produces good-quality reconstructed images at both high and low bit-rates;
2. it also produces a good rate-distortion performance as well as good subjective-quality reconstructed images;
3. in spite of the fact that ABC-SC and the JPEG/IJPEg techniques are both DCT-based techniques, the former technique consistently exhibits a significantly better performance than the latter, especially at high compression ratios;
4. ABC-SC moves block-based compression beyond the limits of JPEG/IJPEg—usually, the maximum compression ratio that can be achieved with ABC-SC is, at least, twice that of IJPEg and five times that of JPEG, with even less objective and subjective degradations;
5. in all of the testing cases, the ABC-SC performance is comparable to that of the wavelet compression technique from both the rate-distortion and the subjective-quality points of view;
6. the execution time of ABC-SC is somewhere between the execution time of SPIHT-A and SPIHT-B, the slow and the fast versions of SPIHT. However, ABC-SC has a potential to be further optimized so that its execution time can be reduced by an order of magnitude or so;
7. ABC-SC provides users with the ability to give certain classes of image segments more importance over others, so that they are less affected by the compression process; and

8. ABC-SC provides a good alternative to wavelet-based compression techniques, especially when adaptability to image content is of interest.

5.2 Future Work

The post-processing filtering technique introduced in this thesis is simple, efficient, and its blocking artifact reduction is quite remarkable. However, we still believe that there is more work to be done in this direction to further improve the overall performance.

ABC-SC classifies the segmented blocks into several perceptually distinct classes and encodes each class separately. It uses the same DCT-based compression scheme, but with a different bit assignment, to encode each of them. We believe that a moderate improvement to ABC-SC could be achieved by using a specialized encoding scheme for each class. However, a considerable effort should be made to search for such appropriate compression schemes for each class. With this improvement, ABC-SC can become a fully heterogeneous compression technique.

Although we have put some effort into optimizing the current implementation of ABC-SC from an execution time point of view, some further improvements in this regard could be realized by optimizing various modules of this technique, especially the segmentation procedure.

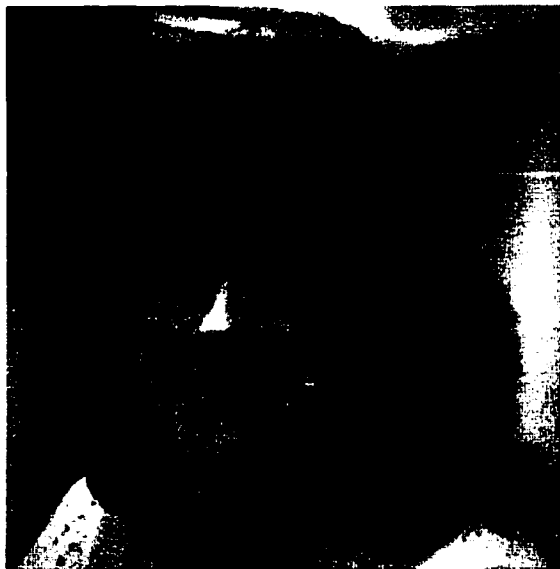
The current ABC-SC implementation version does not have a progressive mode of operation. However, it can be easily modified to handle this mode by coming up with an efficient scheme to reorder the compressed image data to facilitate an effective progressive mode operation.

Finally, the current ABC-SC implementation version deals only with 8-bit images. It would be interesting to find suitable thresholding and quantization functions for color images, hence, extending the scope of ABC-SC to deal with color images too.

Appendix A

Results of the Comparison with JPEG/IJPEG

This appendix provides additional comparison results with the JPEG and IJPEG compression techniques. Figure A.1 shows four additional testing images, namely, the Woman, the Tulips, the Bridge, and the Man images. Figures A.2–A.13 show some supplementary examples of decompressed images and rate-distortion curves for the Monarch, the Boats, the Rocks, the Barbara, the Zelda, the Peppers, the Goldhill, the F16, the Woman, the Tulips, the Bridge, and the Man images—shown in Figures 3.11 and A.1—using ABC-SC, IJPEG, and JPEG. In addition, Table A.1 provides a summary of rate-distortion performance for the decompressed images shown in Figures A.2–A.13.



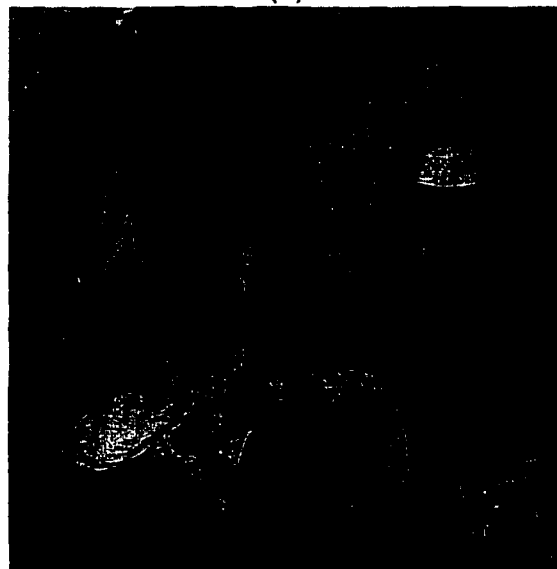
(a)



(b)



(c)



(d)

Figure A.1: Original test images. (a) Woman; (b) Tulips (c) Bridge; and (d) Man.

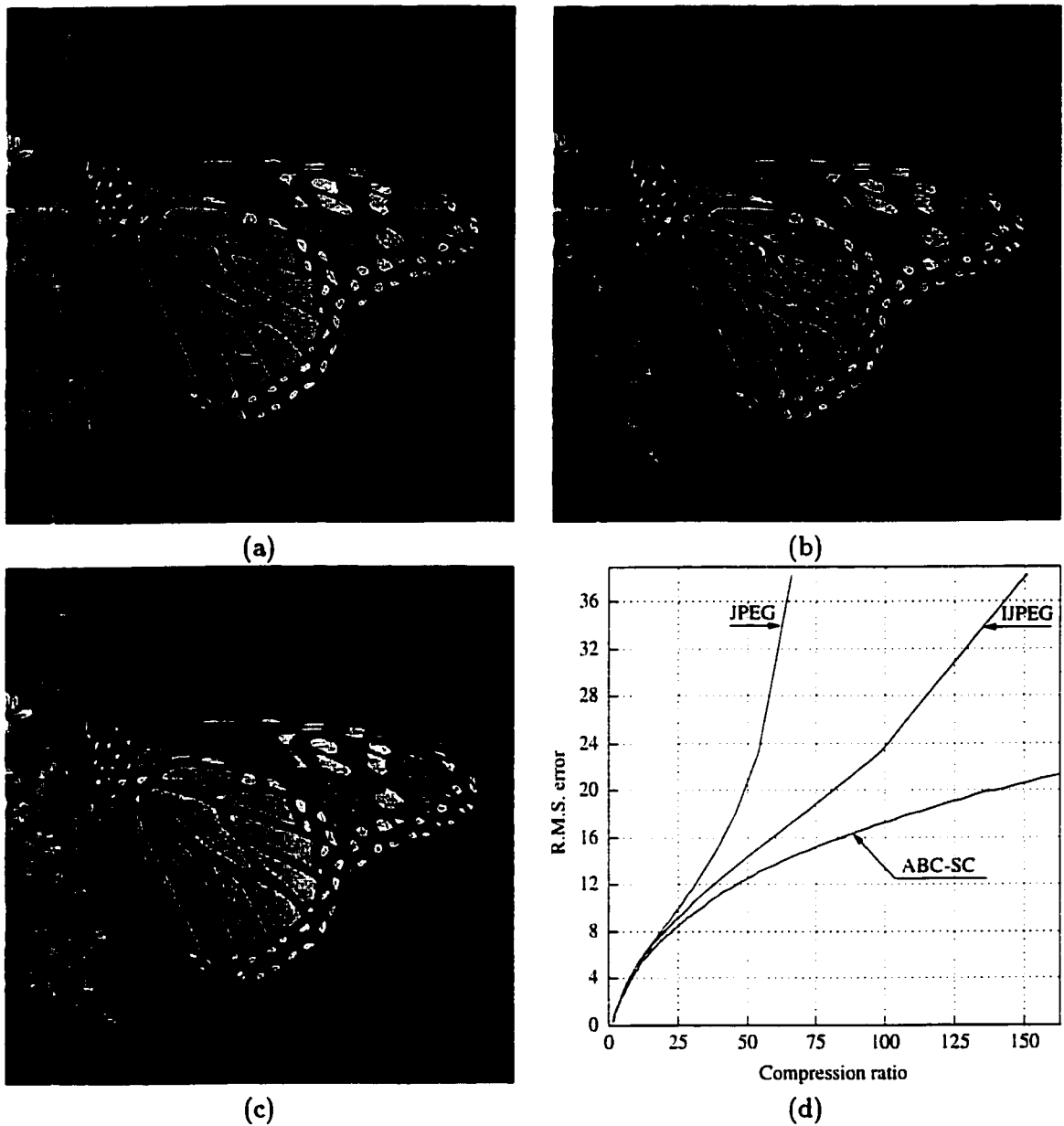


Figure A.2: ABC-SC, IJpeg, and Jpeg compression results for the Monarch image. (a) ABC-SC ($QF = 138$, $TQR = 1.00$, $CR = 41.04$, and $RMSE = 11.25$); (b) IJpeg ($QF = 6$, $CR = 40.90$, and $RMSE = 12.59$); (c) Jpeg ($QF = 4$, $CR = 40.49$, and $RMSE = 15.58$); and (d) rate-distortion curves.

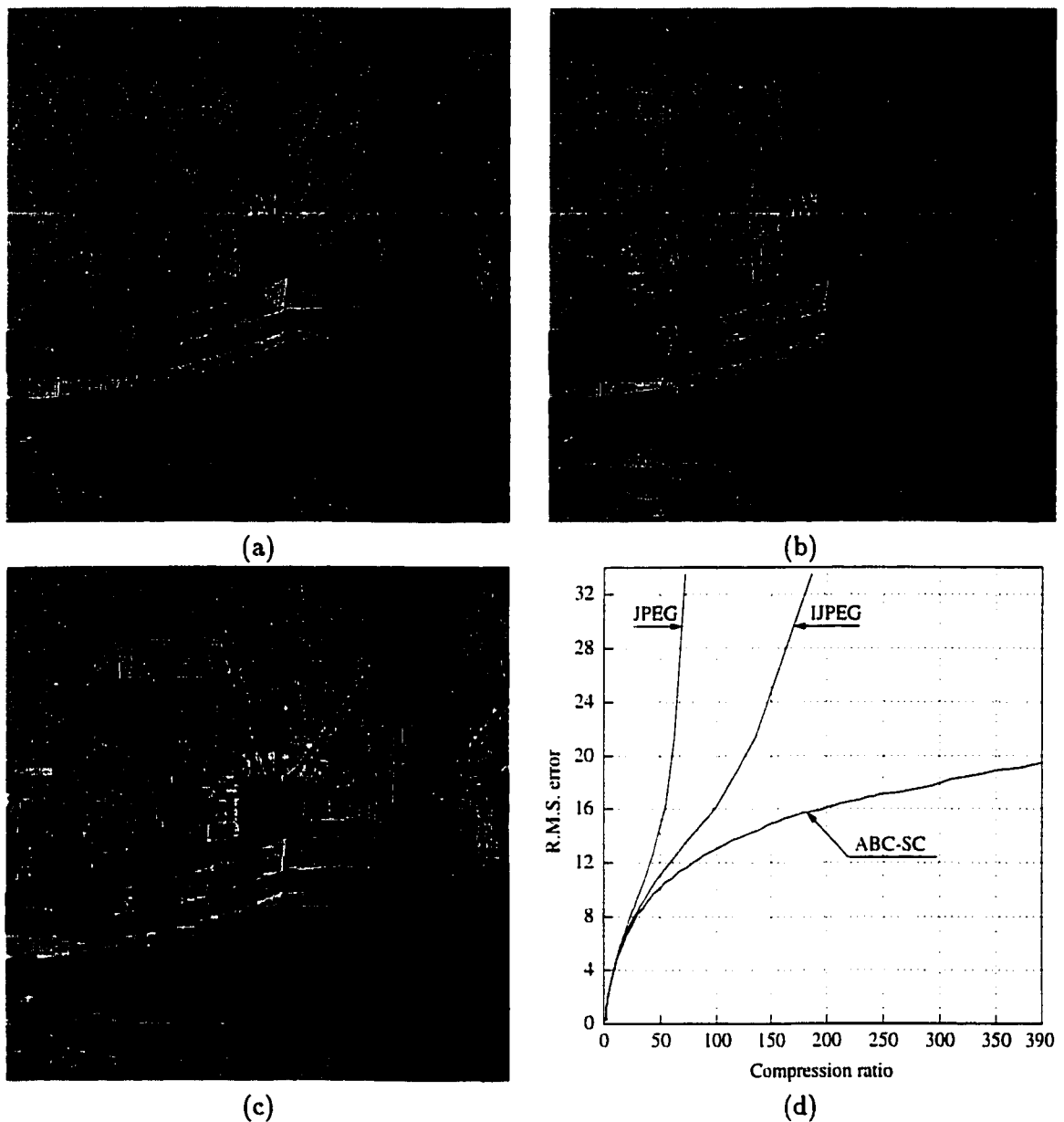


Figure A.3: ABC-SC, IJPEG, and JPEG compression results for the Boats image. (a) ABC-SC ($QF = 141$, $TQR = 1.00$, $CR = 54.72$, and $RMSE = 10.54$); (b) IJPEG ($QF = 6$, $CR = 54.36$, and $RMSE = 11.62$); (c) JPEG ($QF = 3$, $CR = 54.58$, and $RMSE = 16.19$); and (d) rate-distortion curves.

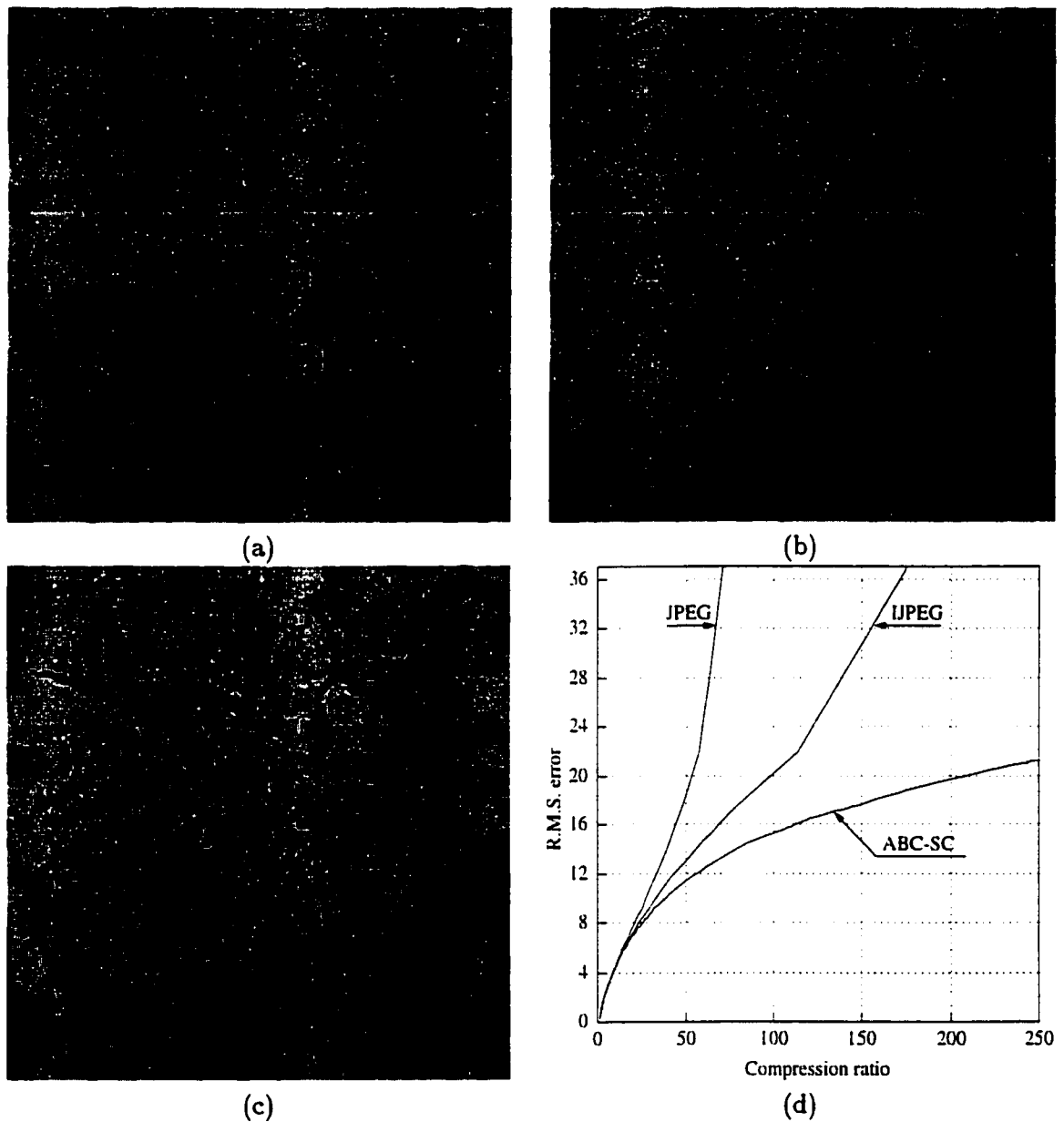


Figure A.4: ABC-SC, IJPEG, and JPEG compression results for the Rocks image. (a) ABC-SC ($QF = 140$, $TQR = 1.00$, $CR = 40.91$, and $RMSE = 10.37$); (b) IJPEG ($QF = 6$, $CR = 40.52$, and $RMSE = 11.59$); (c) JPEG ($QF = 4$, $CR = 40.83$, and $RMSE = 14.61$); and (d) rate-distortion curves.

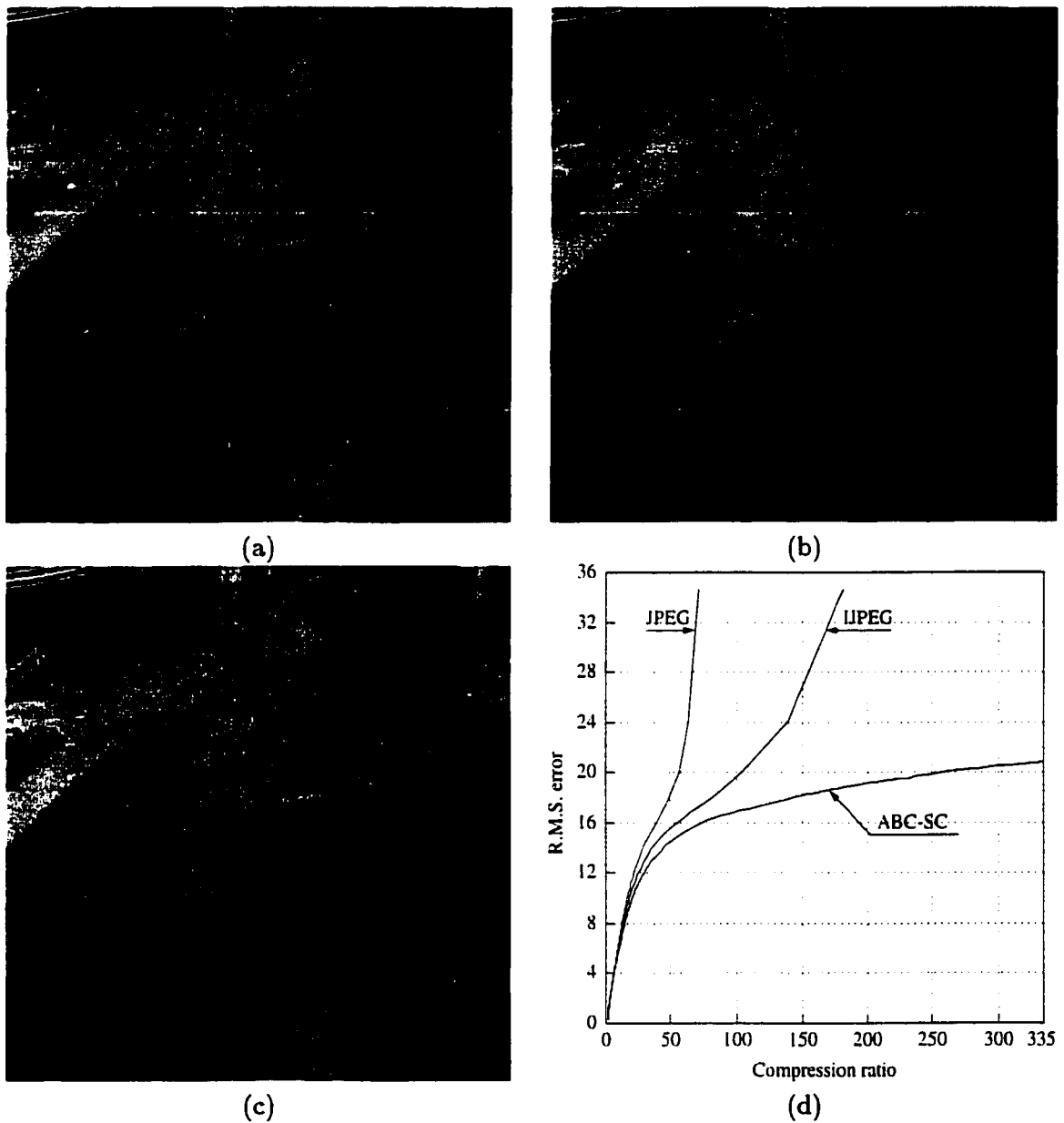


Figure A.5: ABC-SC, IJPEG, and JPEG compression results for the Barbara image. (a) ABC-SC ($QF = 137$, $TQR = 1.00$, $CR = 54.53$, and $RMSE = 14.90$); (b) IJPEG ($QF = 6$, $CR = 54.26$, and $RMSE = 15.97$); (c) JPEG ($QF = 4$, $CR = 49.05$, and $RMSE = 18.03$); and (d) rate-distortion curves.

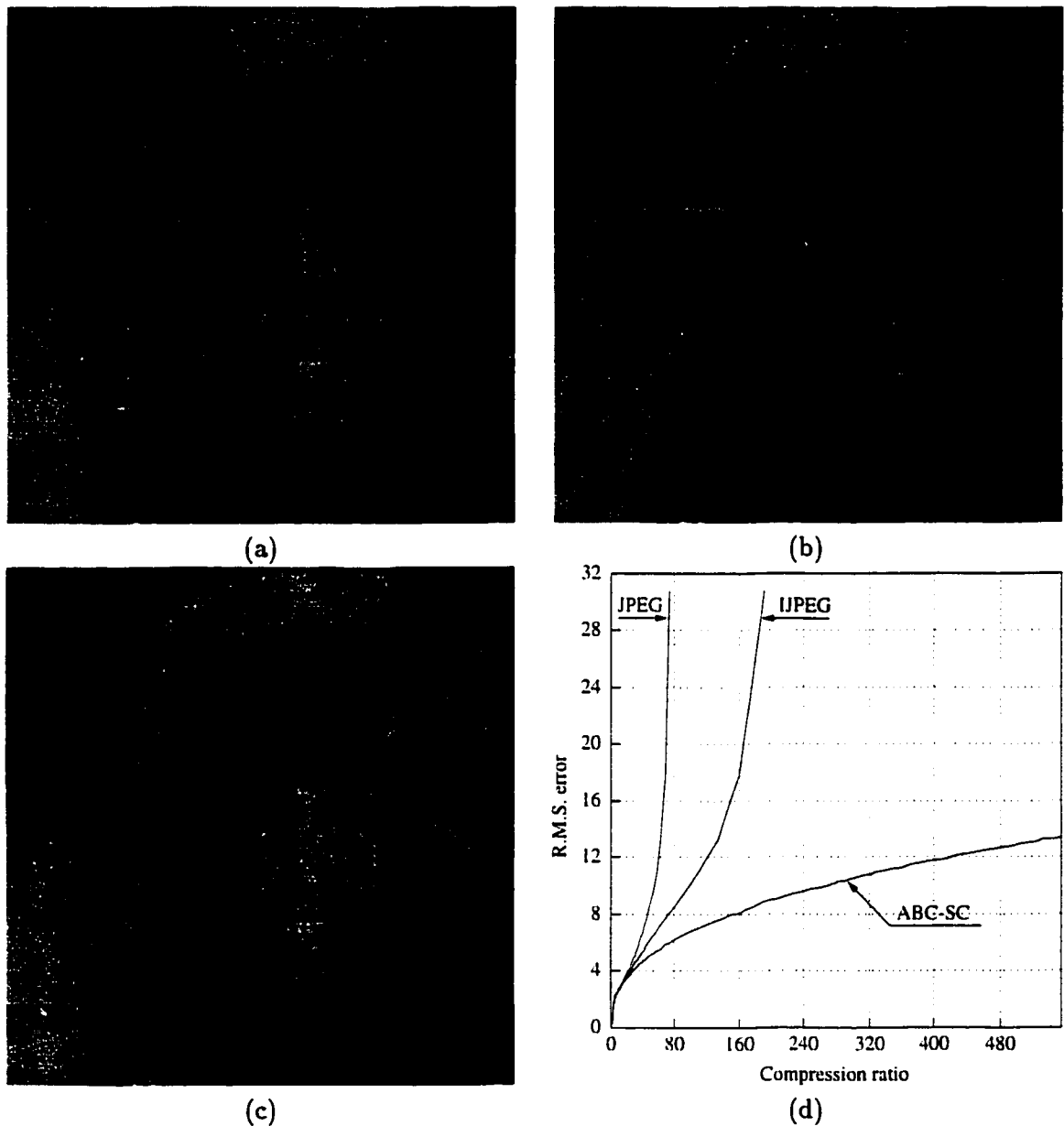


Figure A.6: ABC-SC, IJPEG, and JPEG compression results for the Zelda image. (a) ABC-SC ($QF = 184$, $TQR = 1.00$, $CR = 56.91$, and $RMSE = 5.42$); (b) IJPEG ($QF = 9$, $CR = 55.81$, and $RMSE = 6.71$); (c) JPEG ($QF = 4$, $CR = 56.93$, and $RMSE = 10.99$); and (d) rate-distortion curves.

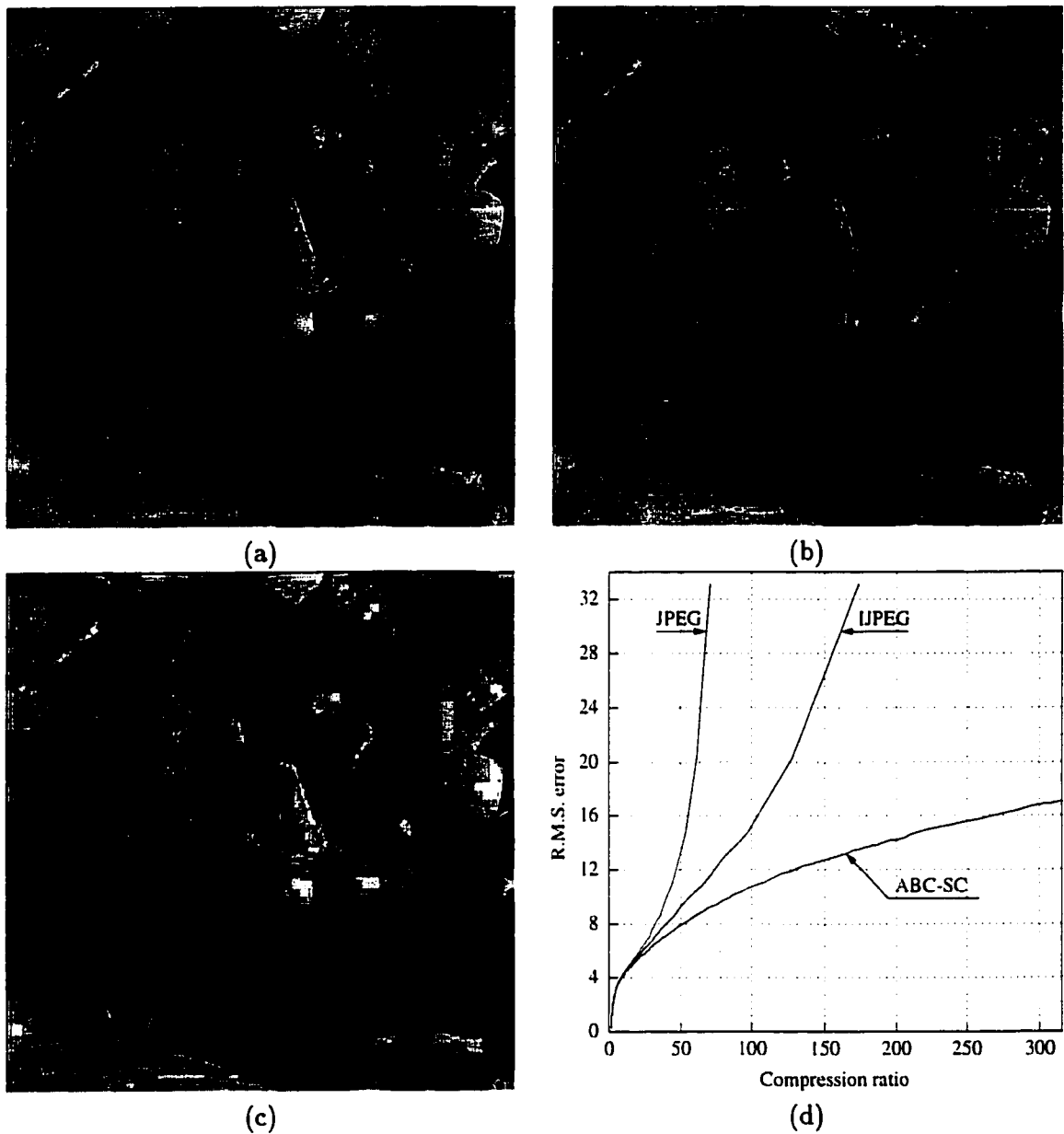


Figure A.7: ABC-SC, IJPEG, and JPEG compression results for the Peppers image. (a) ABC-SC ($QF = 146$, $TQR = 1.00$, $CR = 57.31$, and $RMSE = 8.39$); (b) IJPEG ($QF = 6$, $CR = 57.27$, and $RMSE = 10.10$); (c) JPEG ($QF = 3$, $CR = 54.20$, and $RMSE = 14.80$); and (d) rate-distortion curves.

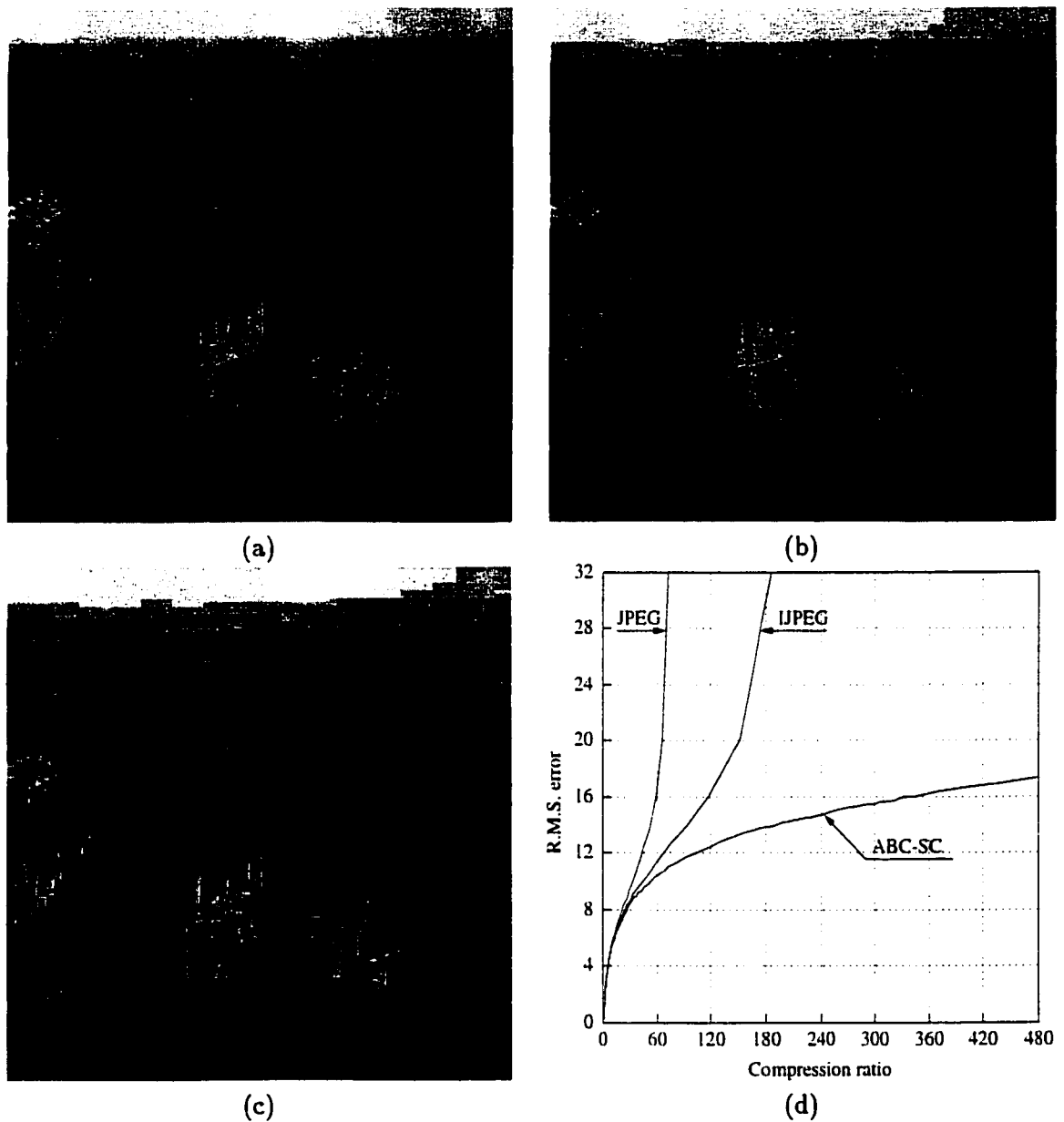


Figure A.8: ABC-SC, IJPEG, and JPEG compression results for the Goldhill image. (a) ABC-SC ($QF = 146$, $TQR = 1.00$, $CR = 62.67$, and $RMSE = 10.56$); (b) IJPEG ($QF = 6$, $CR = 61.87$, and $RMSE = 11.56$); (c) JPEG ($QF = 3$, $CR = 58.55$, and $RMSE = 15.96$); and (d) rate-distortion curves.

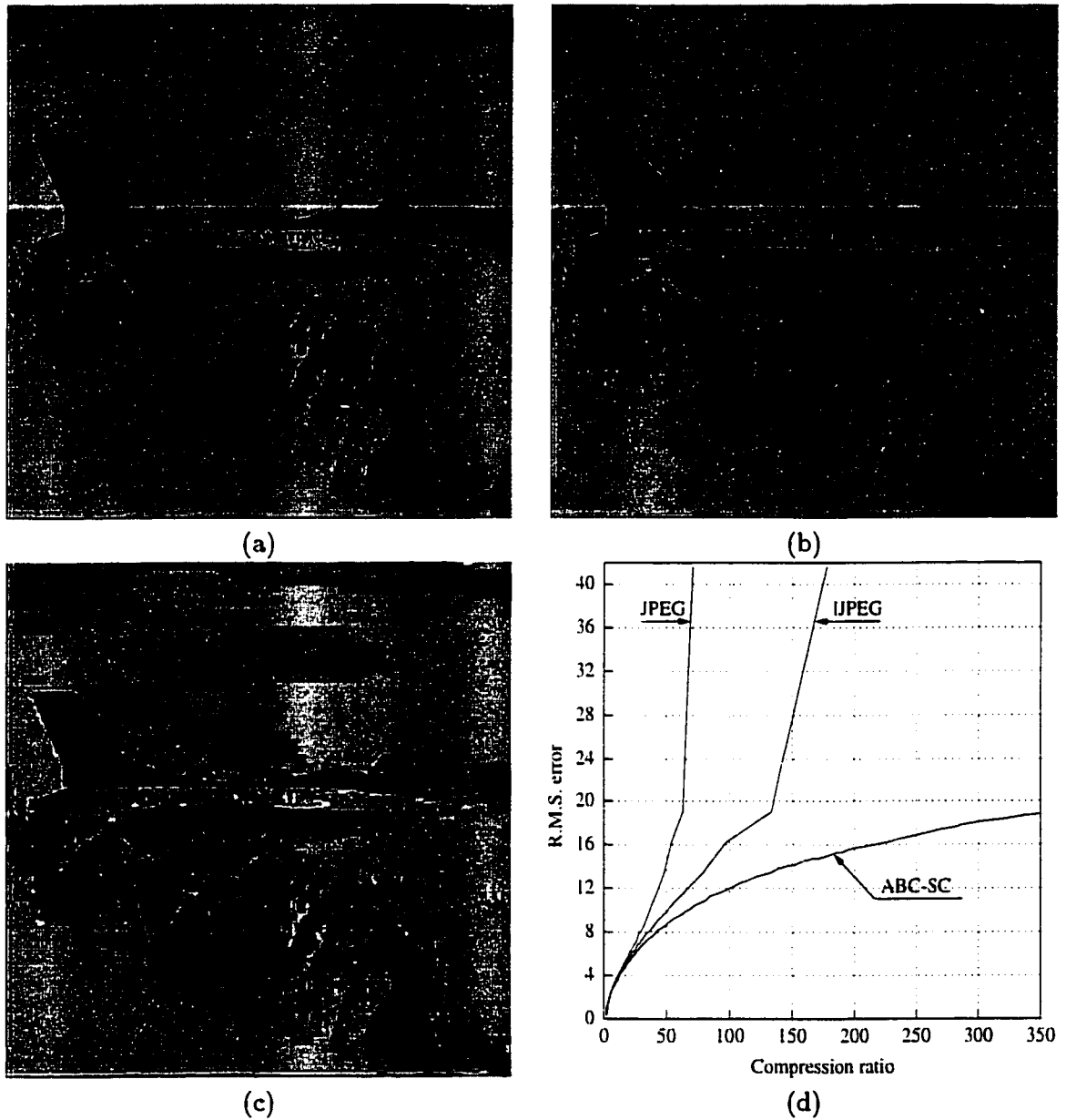


Figure A.9: ABC-SC, IJPEG, and JPEG compression results for the F16 image. (a) ABC-SC ($QF = 141$, $TQR = 1.00$, $CR = 56.09$, and $RMSE = 9.24$); (b) IJPEG ($QF = 6$, $CR = 55.40$, and $RMSE = 10.73$); (c) JPEG ($QF = 3$, $CR = 53.55$, and $RMSE = 16.27$); and (d) rate-distortion curves.

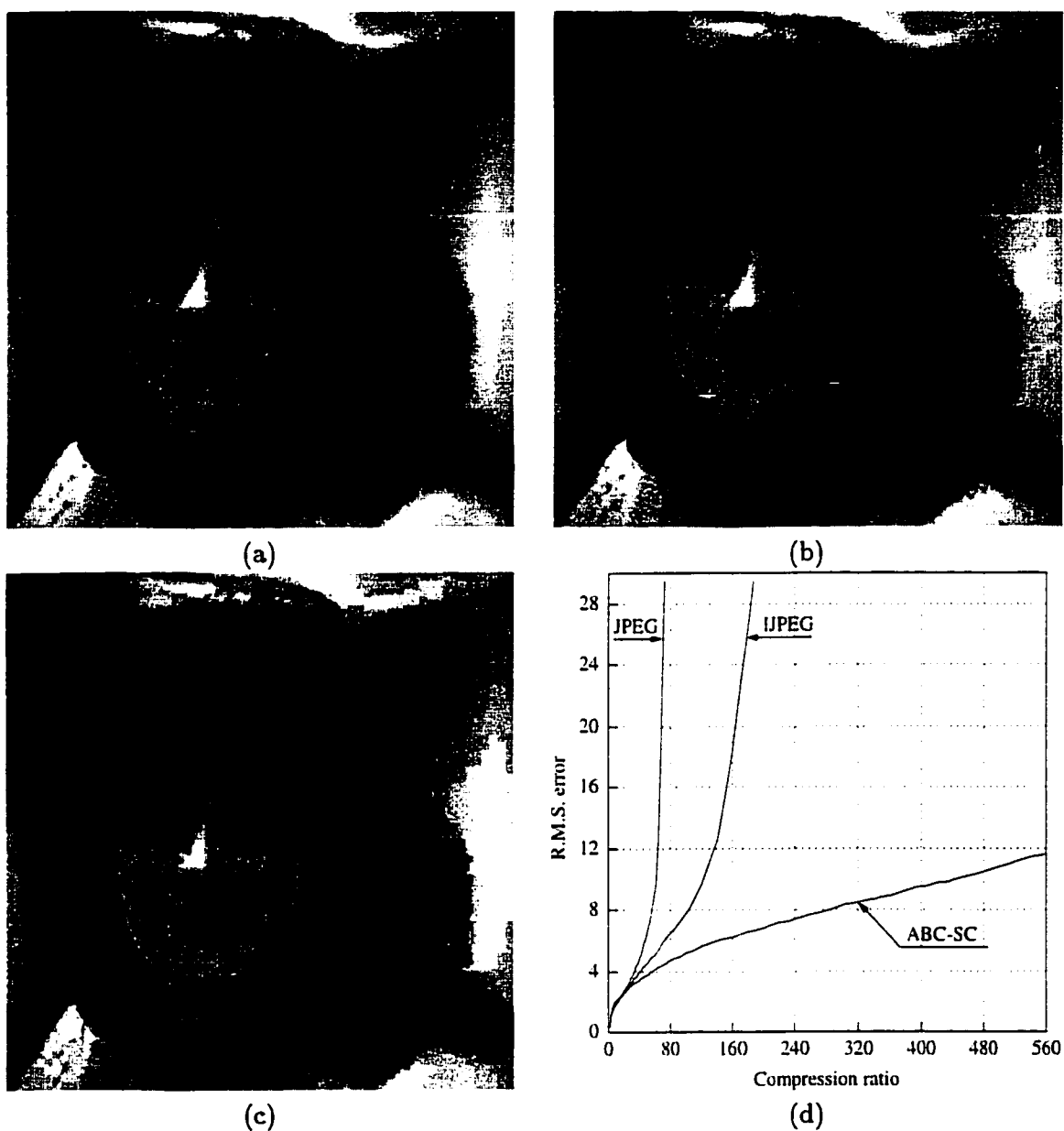


Figure A.10: ABC-SC, IJPEG, and JPEG compression results for the Woman image. (a) ABC-SC ($QF = 197$, $TQR = 1.00$, $CR = 64.63$, and $RMSE = 4.28$); (b) IJPEG ($QF = 10$, $CR = 64.11$, and $RMSE = 5.40$); (c) JPEG ($QF = 4$, $CR = 60.49$, and $RMSE = 9.66$); and (d) rate-distortion curves.

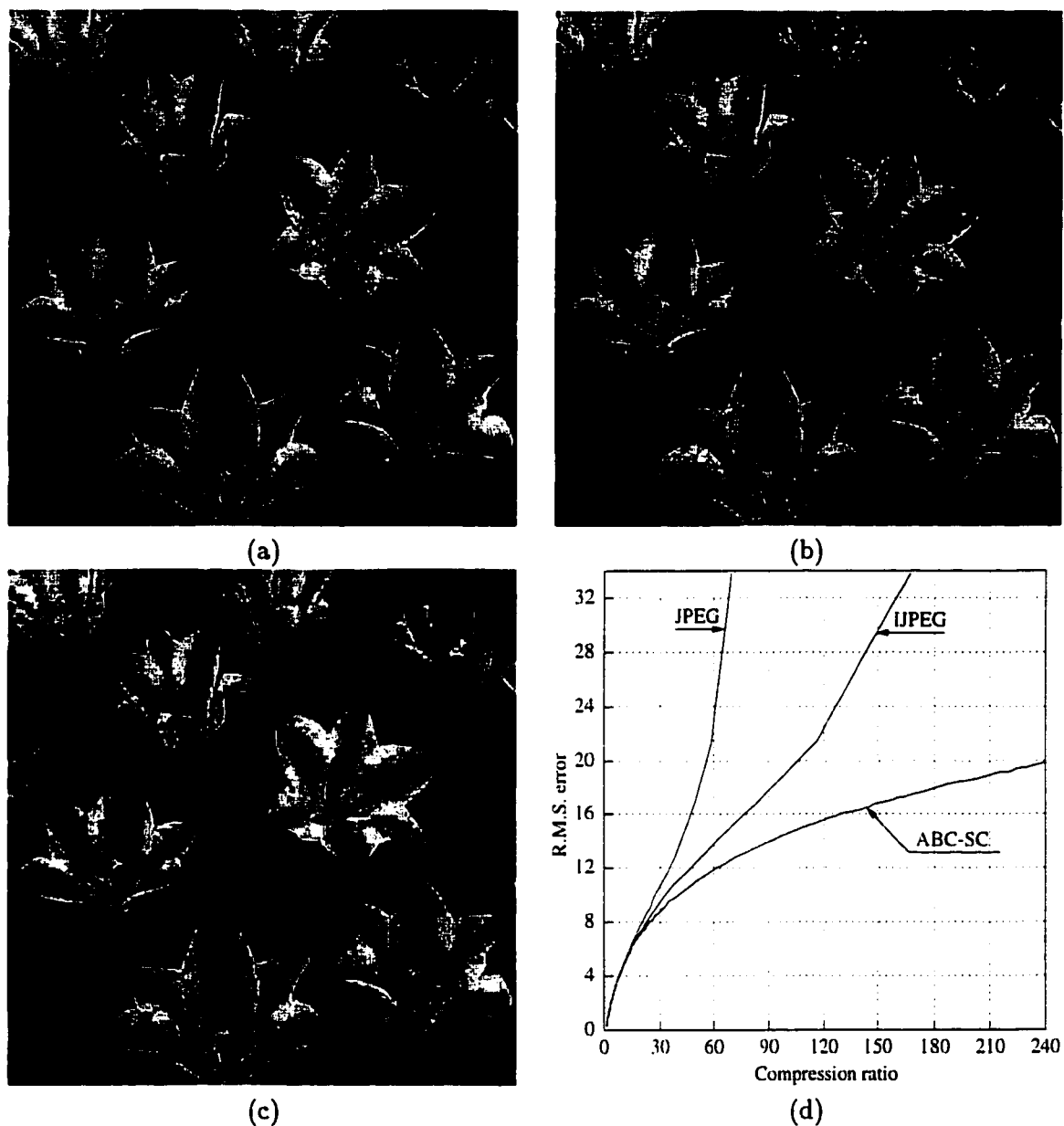


Figure A.11: ABC-SC, IJPEG, and JPEG compression results for the Tulips image. (a) ABC-SC ($QF = 140$, $TQR = 1.00$, $CR = 44.89$, and $RMSE = 10.53$); (b) IJPEG ($QF = 6$, $CR = 44.04$, and $RMSE = 11.70$); (c) JPEG ($QF = 4$, $CR = 43.03$, and $RMSE = 14.49$); and (d) rate-distortion curves.

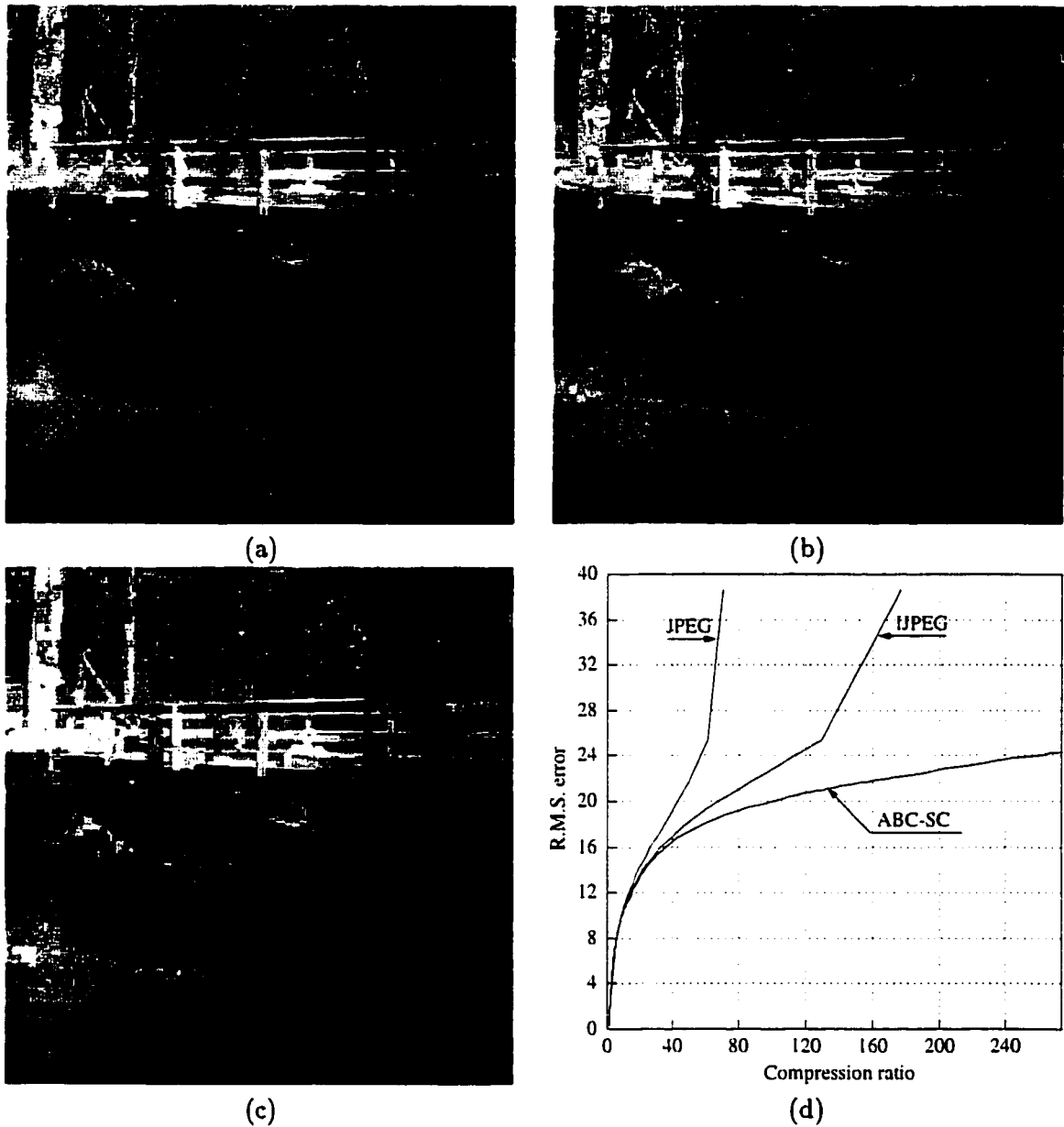


Figure A.12: ABC-SC, IJPEG, and JPEG compression results for the Bridge image. (a) ABC-SC ($QF = 126$, $TQR = 1.00$, $CR = 49.98$, and $RMSE = 17.42$); (b) IJPEG ($QF = 5$, $CR = 47.03$, and $RMSE = 17.92$); (c) JPEG ($QF = 3$, $CR = 49.87$, and $RMSE = 21.72$); and (d) rate-distortion curves.

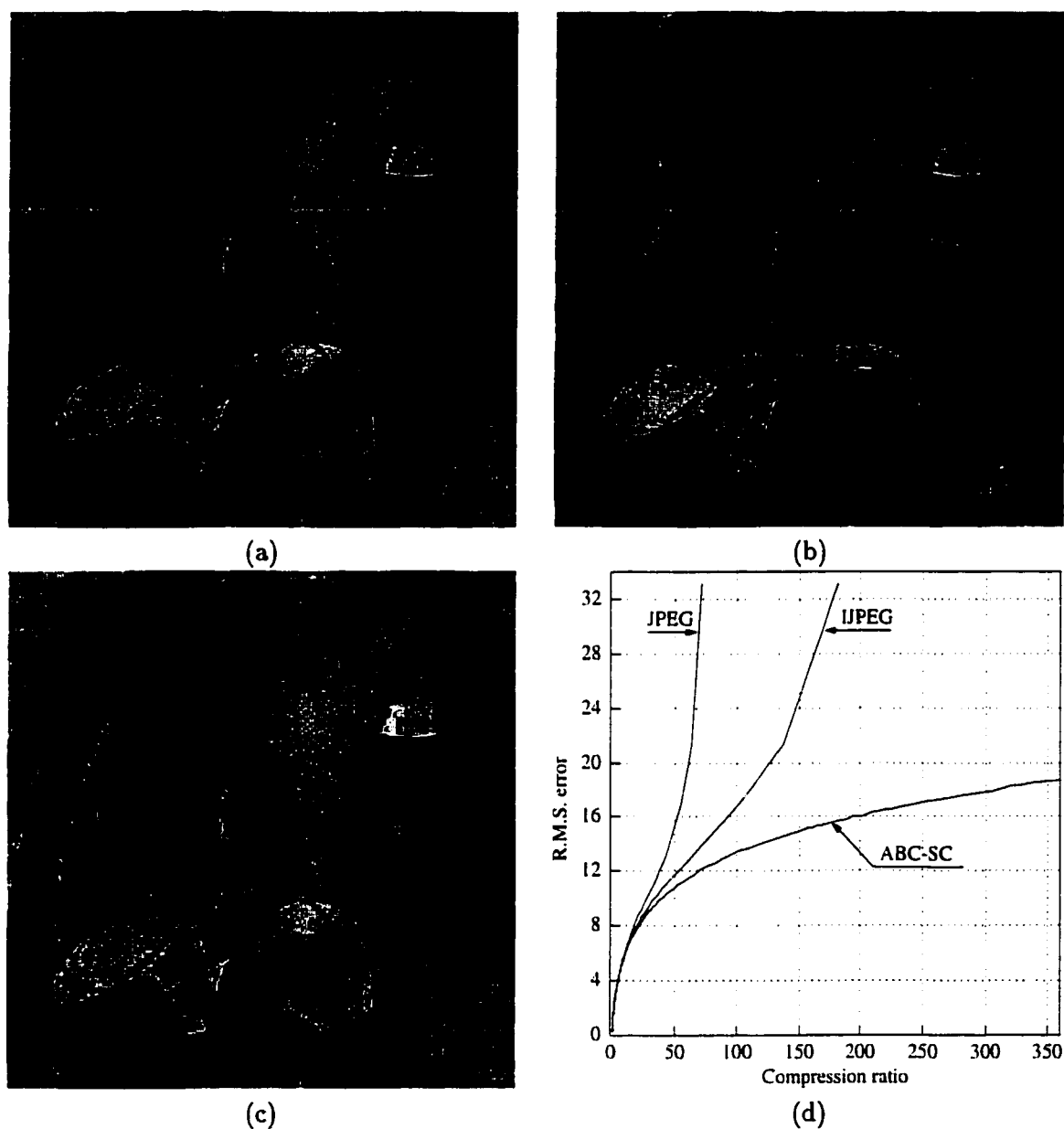


Figure A.13: ABC-SC, IJPEG, and JPEG compression results for the Man image. (a) ABC-SC ($QF = 142$, $TQR = 1.00$, $CR = 54.64$, and $RMSE = 11.12$); (b) IJPEG ($QF = 6$, $CR = 54.08$, and $RMSE = 12.10$); (c) JPEG ($QF = 4$, $CR = 48.64$, and $RMSE = 14.71$); and (d) rate-distortion curves.

Table A.1: A summary of rate-distortion performance for the decompressed images shown in Figures A.2–A.13.

Compression technique	Monarch		Boats		Rocks	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	41.04:1	11.25	54.72:1	10.54	40.91:1	10.37
IJPEG	40.90:1	12.59	54.36:1	11.62	40.52:1	11.59
JPEG	40.49:1	15.58	54.58:1	16.19	40.83:1	14.61

Compression technique	Barbara		Zelda		Peppers	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	54.53:1	14.90	56.91:1	5.42	57.31:1	8.39
IJPEG	54.26:1	15.97	55.81:1	6.71	57.27:1	10.10
JPEG	49.05:1	18.03	56.93:1	10.99	54.20:1	14.80

Compression technique	Goldhill		F16		Woman	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	62.67:1	10.56	56.09:1	9.24	64.63:1	4.28
IJPEG	61.87:1	11.56	55.40:1	10.73	64.11:1	5.40
JPEG	58.55:1	15.96	53.55:1	16.27	60.49:1	9.66

Compression technique	Tulips		Bridge		Man	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	44.89:1	10.53	49.98:1	17.42	54.64:1	11.12
IJPEG	44.04:1	11.70	47.03:1	17.92	54.08:1	12.10
JPEG	43.03:1	14.49	49.87:1	21.72	48.64:1	14.71

Appendix B

Results of the Comparison with SPIHT-A/SPIHT-B

This appendix provides additional comparison results with the SPIHT-A and SPIHT-A compression techniques. Figures B.1–B.12 show some supplementary examples of decompressed images and rate-distortion curves for the Monarch, the Boats, the Rocks, the Barbara, the Zelda, the Peppers, the Goldhill, the F16, the Woman, the Tulips, the Bridge, and the Man images—shown in Figures 3.11 and A.1—using ABC-SC, SPIHT-A, and SPIHT-B. In addition, Table B.1 provides a summary of rate-distortion performance for the decompressed images shown in Figures B.1–B.12.

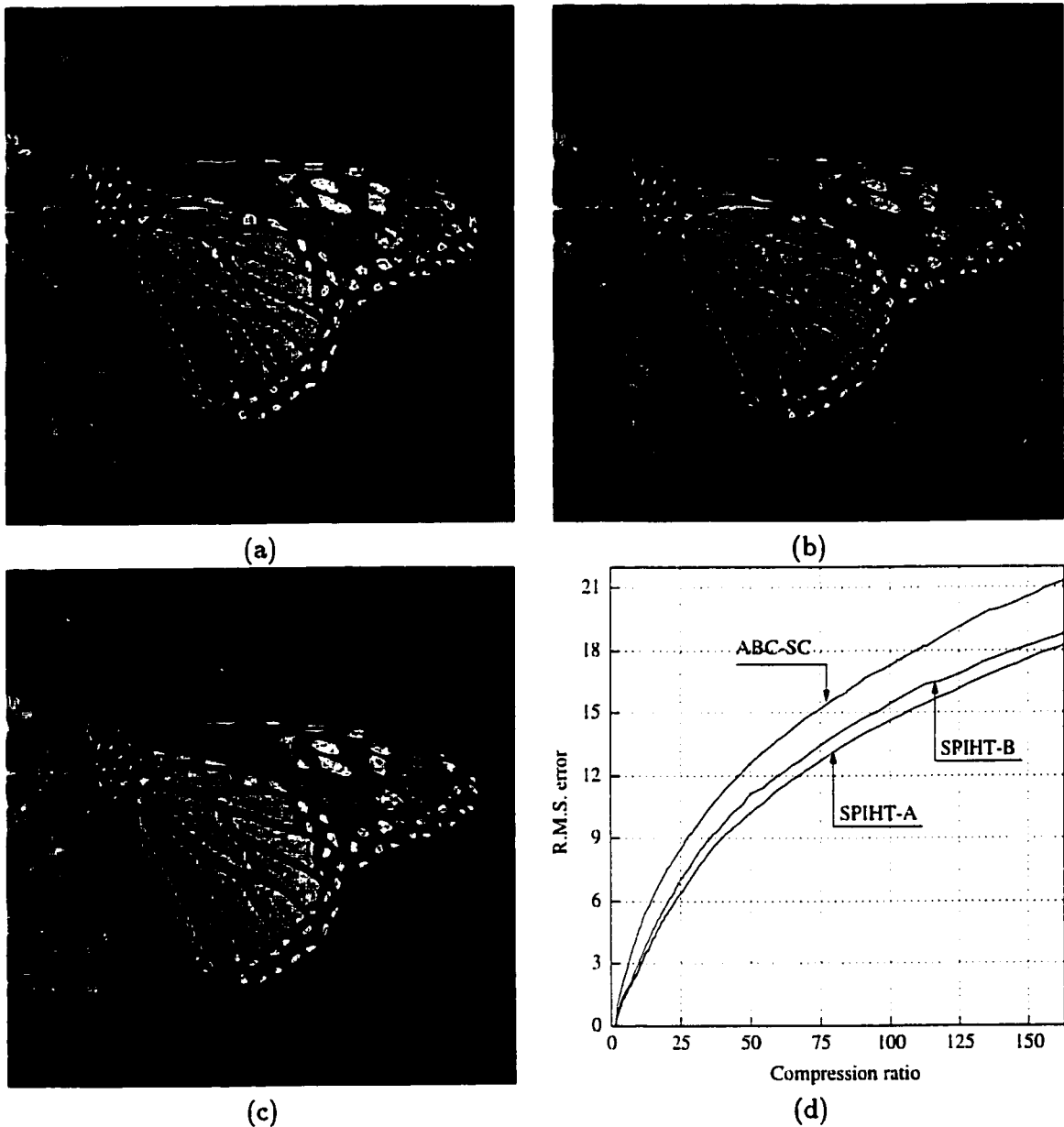


Figure B.1: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Monarch image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 104.03$, and $RMSE = 17.61$); (b) SPIHT-A ($CR = 104.03$, and $RMSE = 14.89$); (c) SPIHT-B ($CR = 104.03$, and $RMSE = 15.77$); and (d) rate-distortion curves.

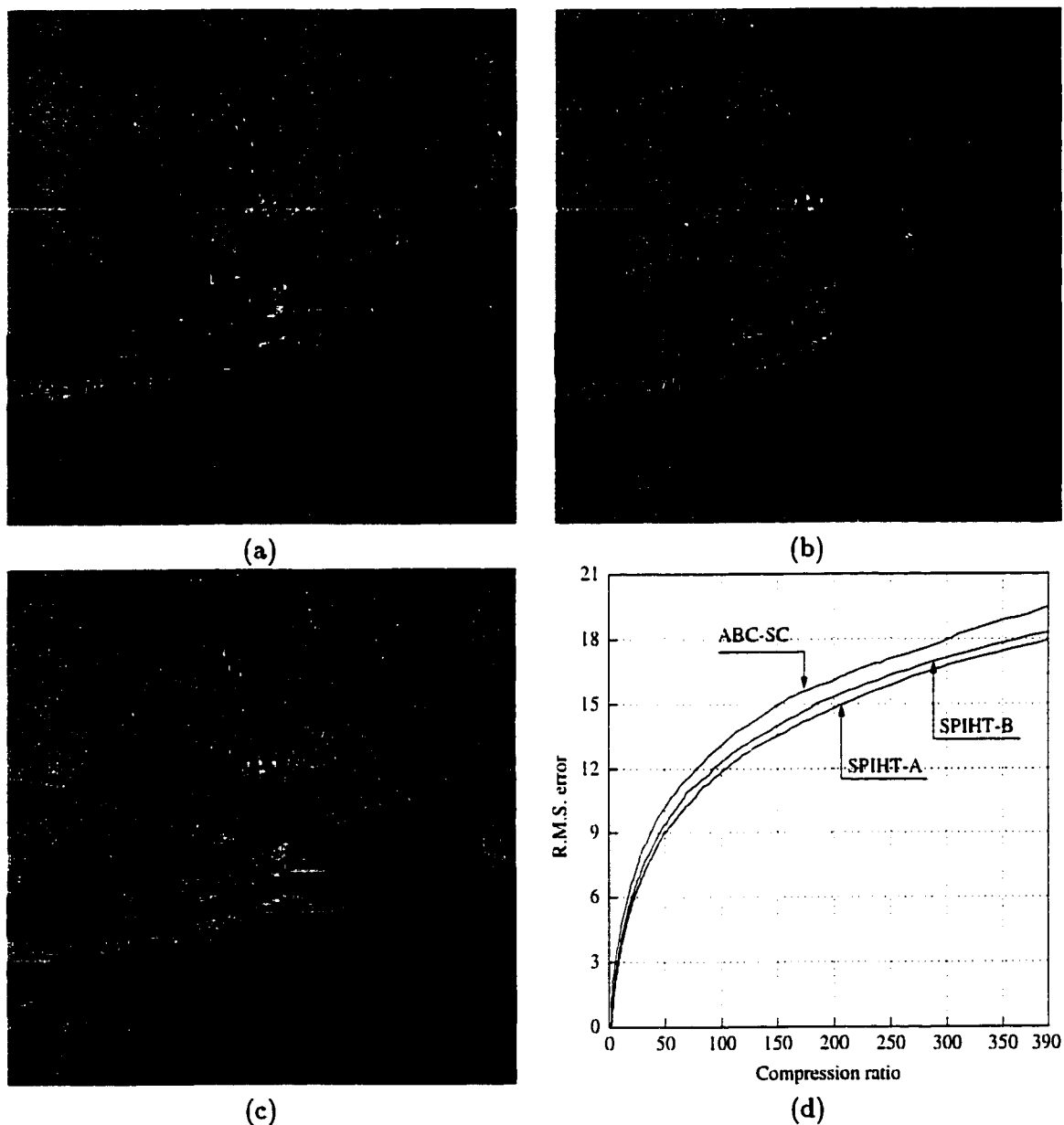


Figure B.2: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Boats image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 212.09$, and $RMSE = 16.38$); (b) SPIHT-A ($CR = 211.92$, and $RMSE = 15.11$); (c) SPIHT-B ($CR = 212.09$, and $RMSE = 15.60$); and (d) rate-distortion curves.

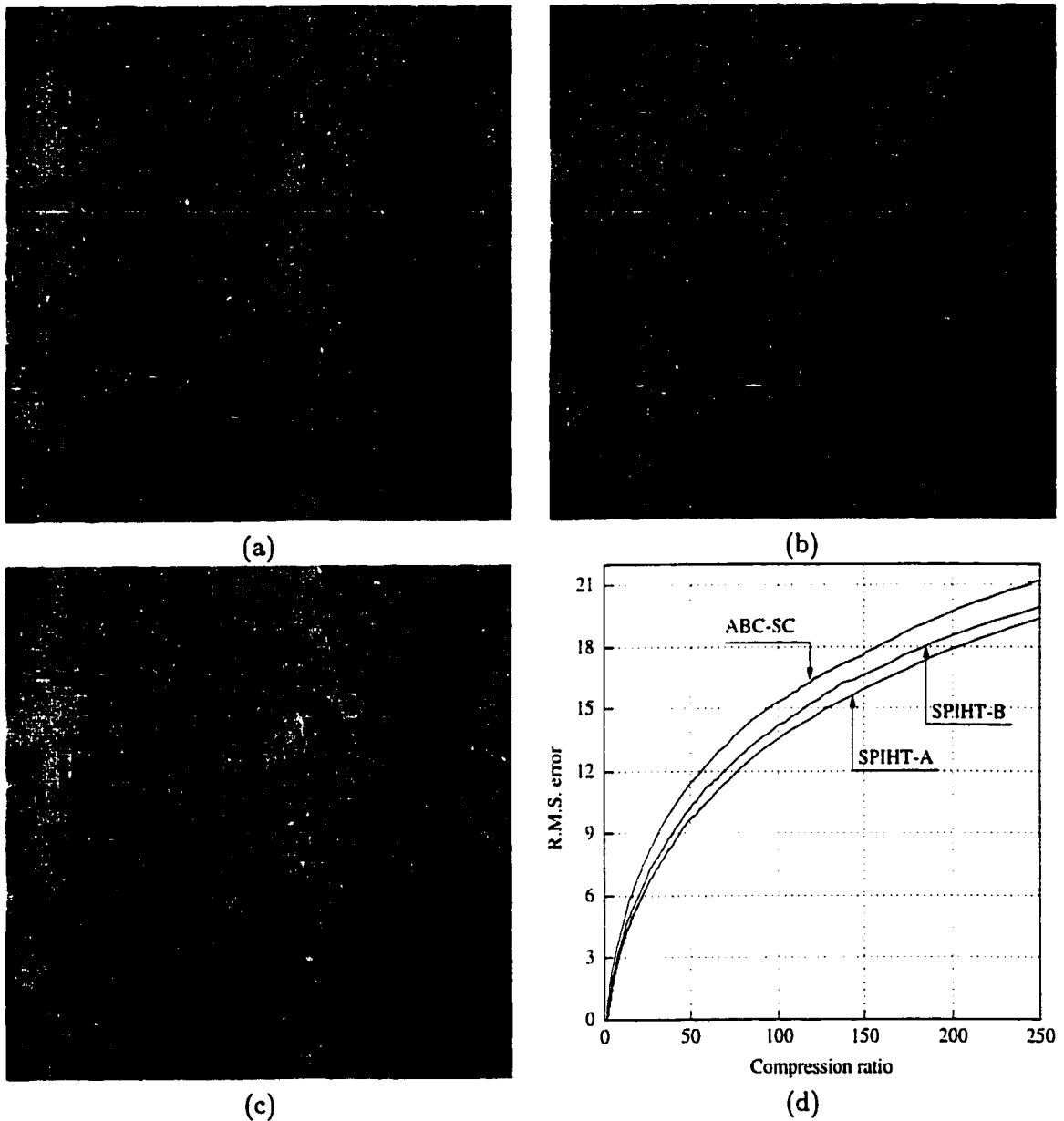


Figure B.3: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Rocks image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 156.22$, and $RMSE = 17.93$); (b) SPIHT-A ($CR = 156.22$, and $RMSE = 16.21$); (c) SPIHT-B ($CR = 156.22$, and $RMSE = 16.87$); and (d) rate-distortion curves.

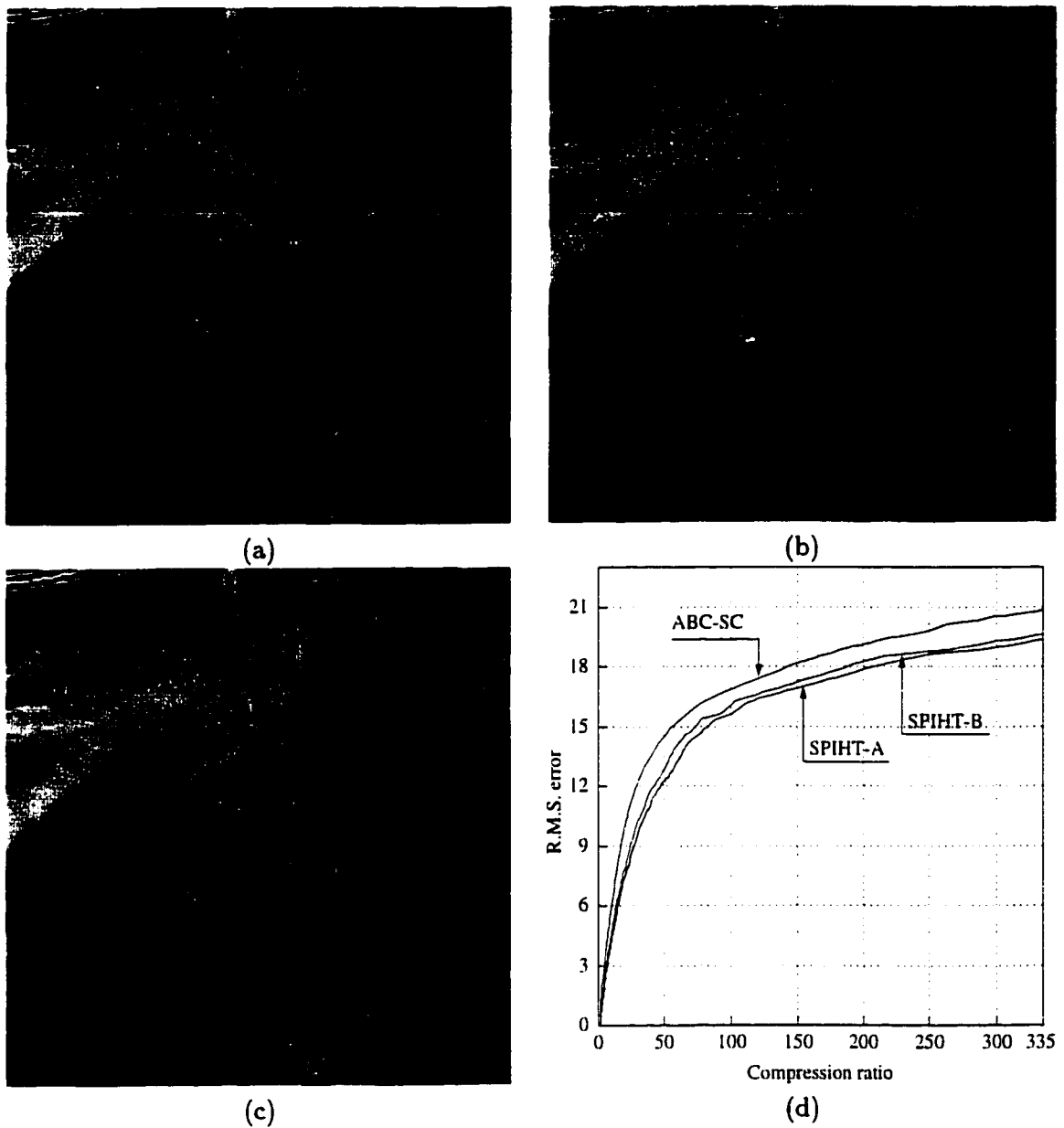


Figure B.4: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Barbara image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 209.38$, and $RMSE = 19.28$); (b) SPIHT-A ($CR = 209.38$, and $RMSE = 18.04$); (c) SPIHT-B ($CR = 209.38$, and $RMSE = 18.44$); and (d) rate-distortion curves.

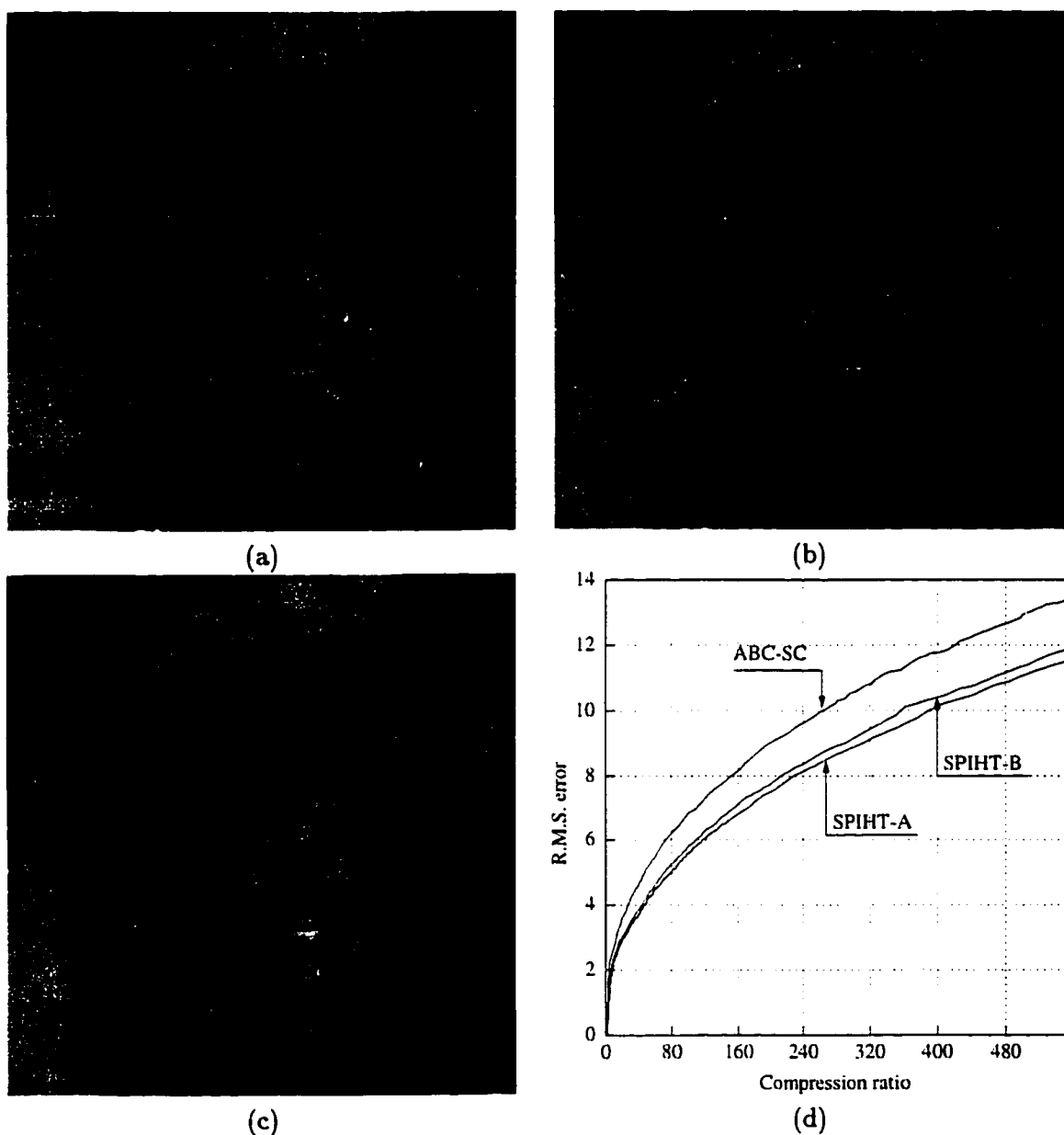


Figure B.5: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Zelda image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 391.85$, and $RMSE = 11.75$); (b) SPIHT-A ($CR = 391.85$, and $RMSE = 10.00$); (c) SPIHT-B ($CR = 391.85$, and $RMSE = 10.34$); and (d) rate-distortion curves.

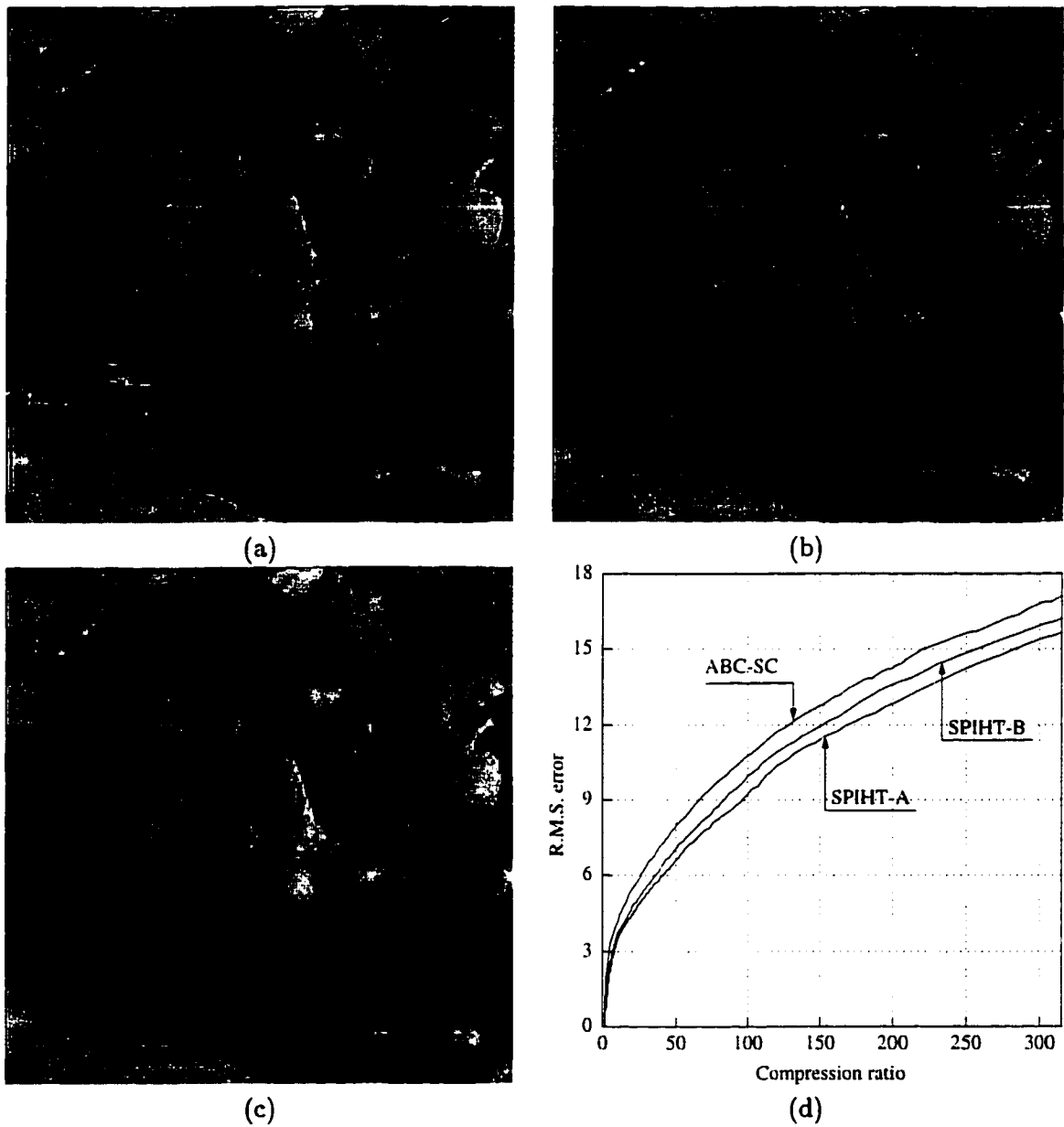


Figure B.6: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Peppers image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 188.32$, and $RMSE = 13.94$); (b) SPIHT-A ($CR = 188.32$, and $RMSE = 12.50$); (c) SPIHT-B ($CR = 188.32$, and $RMSE = 13.20$); and (d) rate-distortion curves.

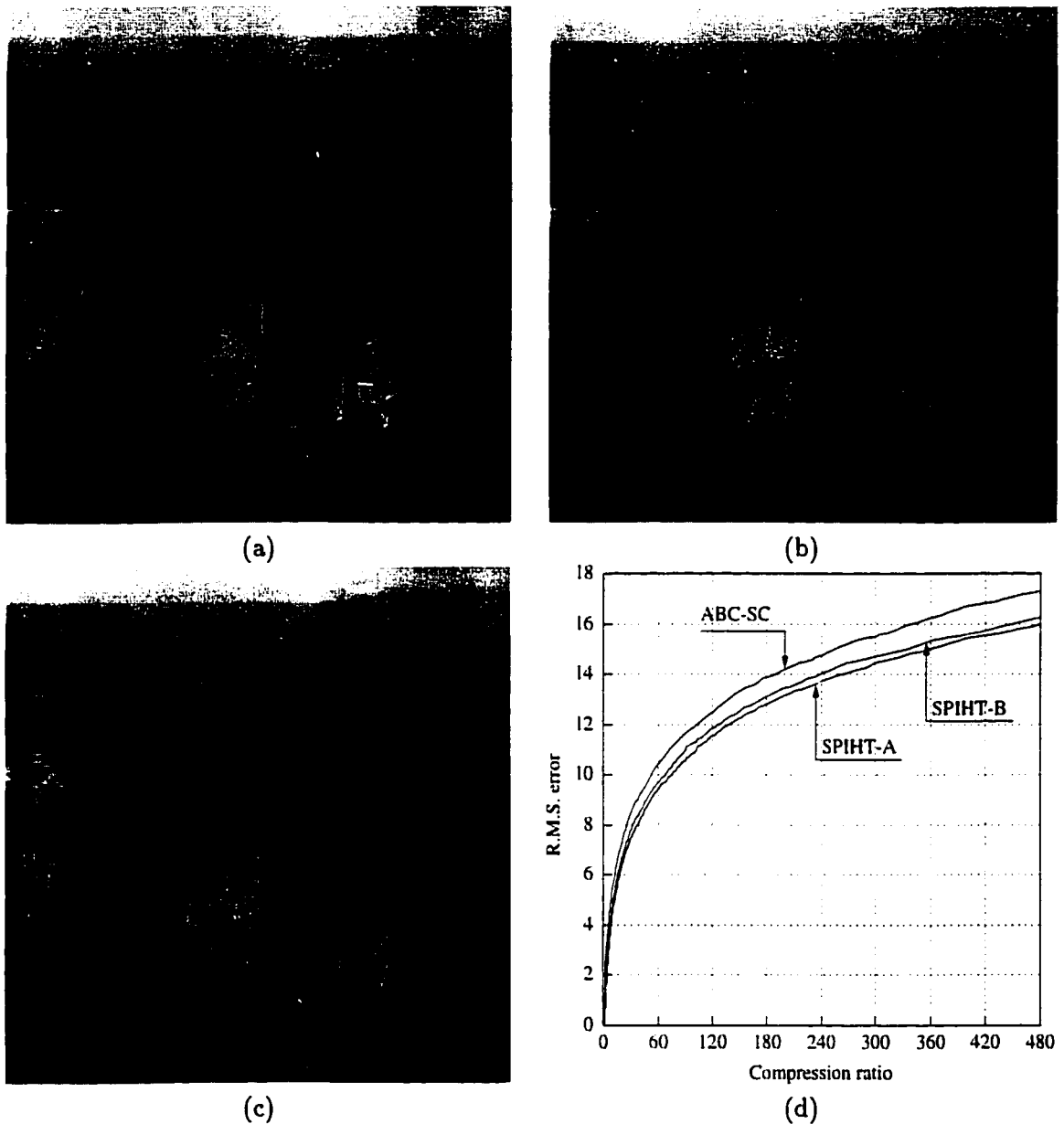


Figure B.7: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Goldhill image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 310.23$, and $RMSE = 15.63$); (b) SPIHT-A ($CR = 310.23$, and $RMSE = 14.54$); (c) SPIHT-B ($CR = 310.23$, and $RMSE = 14.79$); and (d) rate-distortion curves.

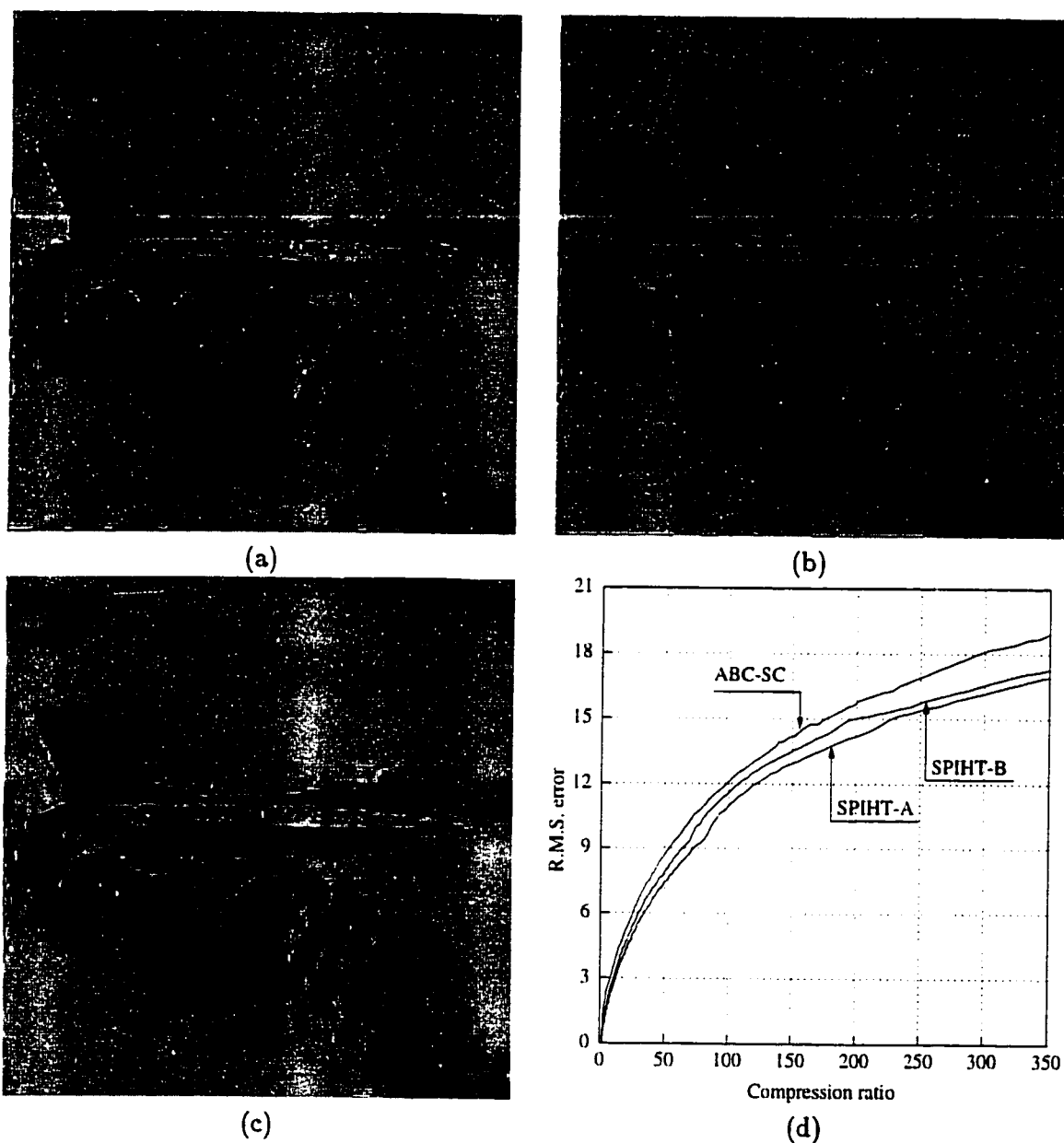


Figure B.8: ABC-SC, SPIHT-A, and SPIHT-B compression results for the F16 image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 203.53$, and $RMSE = 15.84$); (b) SPIHT-A ($CR = 203.53$, and $RMSE = 14.30$); (c) SPIHT-B ($CR = 203.53$, and $RMSE = 15.06$); and (d) rate-distortion curves.

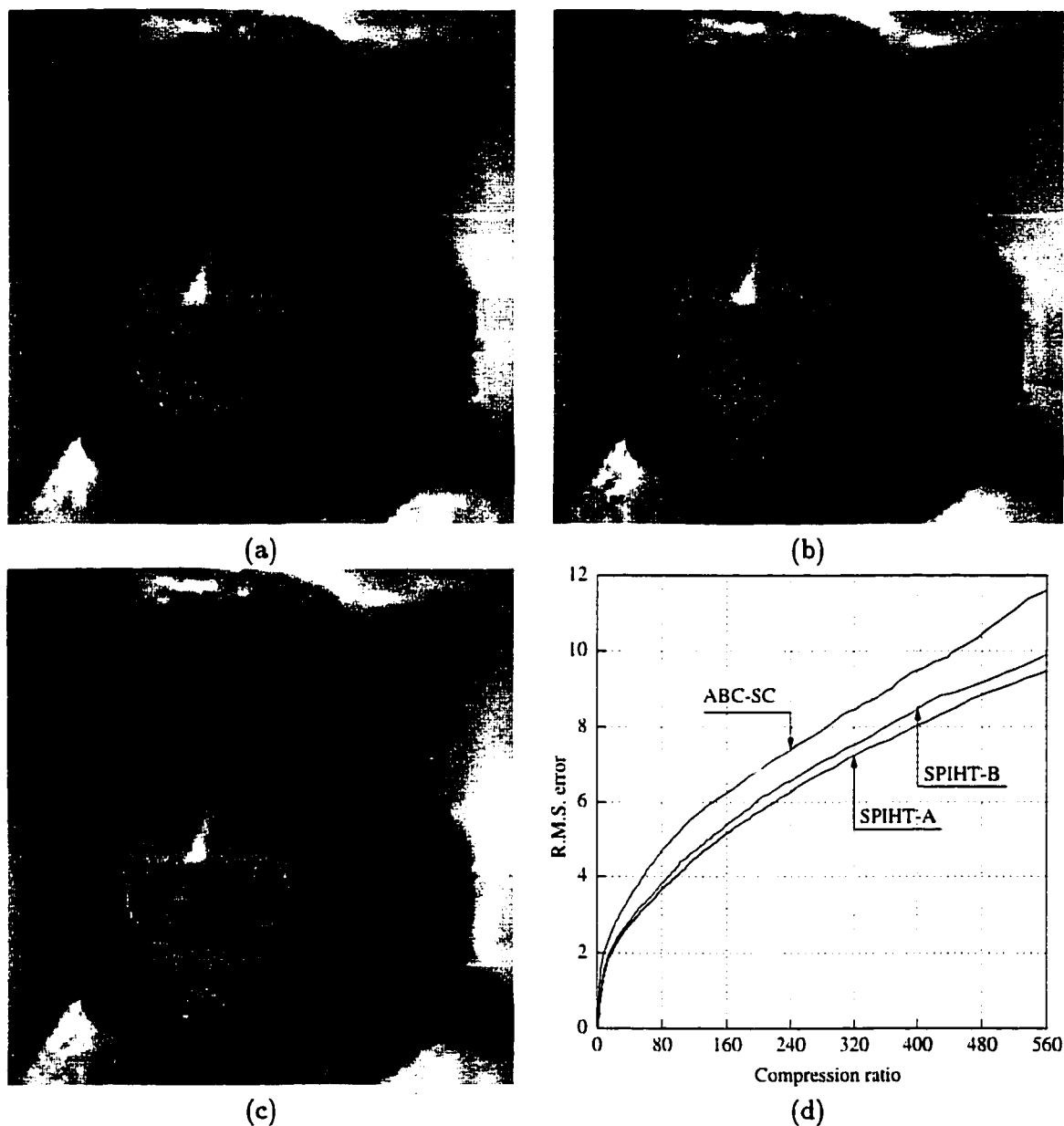


Figure B.9: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Woman image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 422.13$, and $RMSE = 9.74$); (b) SPIHT-A ($CR = 421.45$, and $RMSE = 8.25$); (c) SPIHT-B ($CR = 422.13$, and $RMSE = 8.82$); and (d) rate-distortion curves.

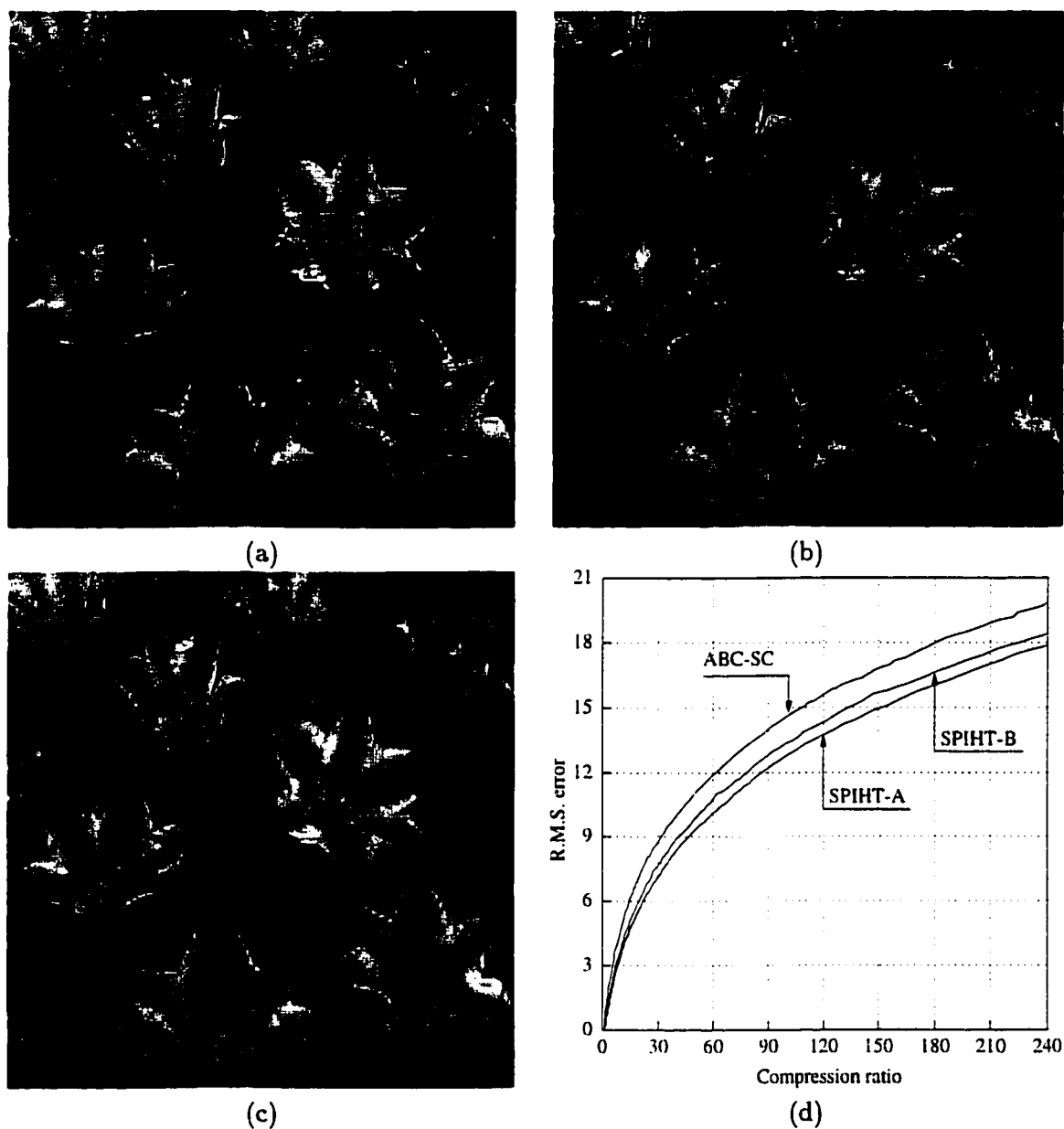


Figure B.10: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Tulips image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 148.78$, and $RMSE = 16.77$); (b) SPIHT-A ($CR = 148.86$, and $RMSE = 14.90$); (c) SPIHT-B ($CR = 148.78$, and $RMSE = 15.68$); and (d) rate-distortion curves.

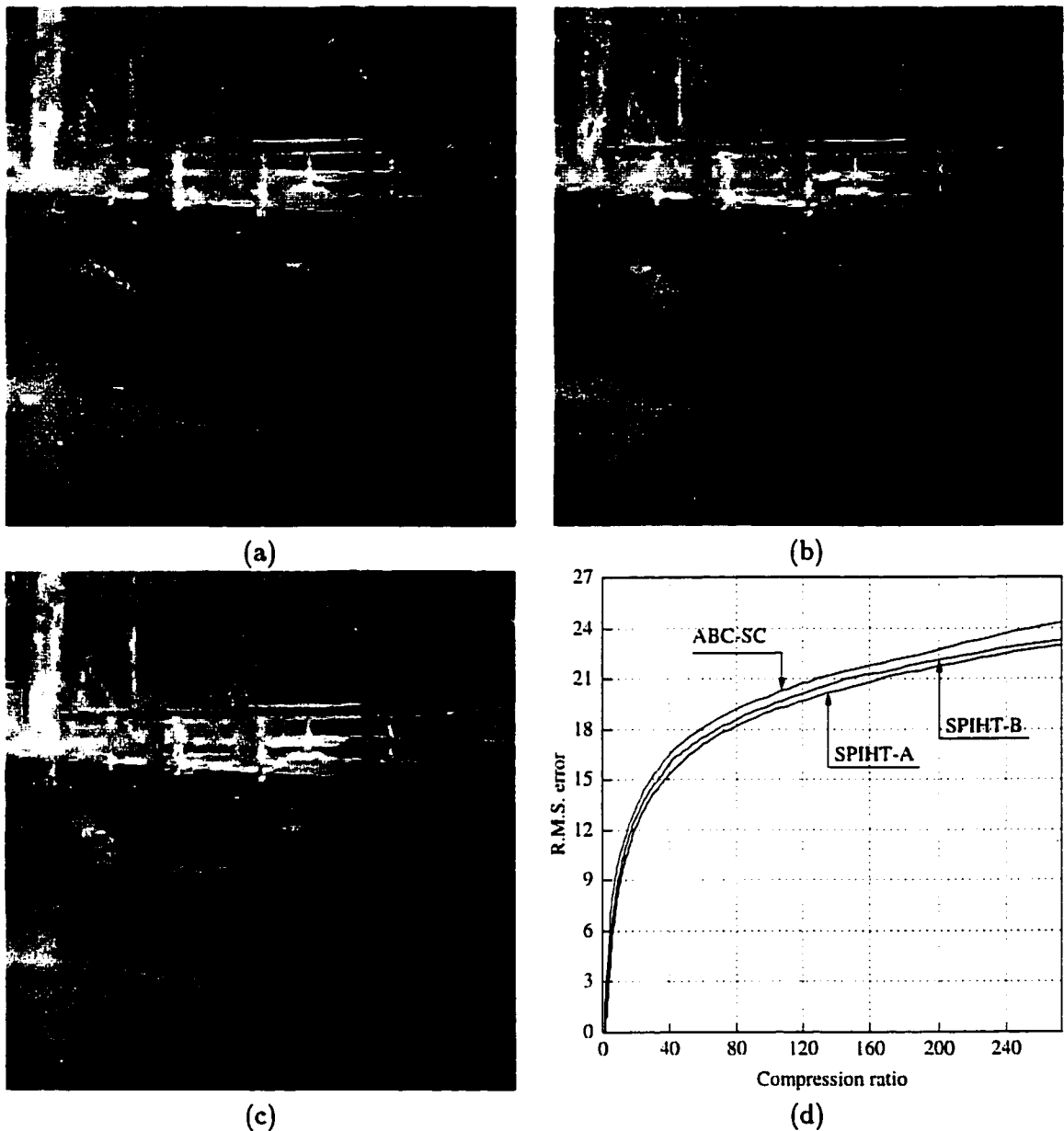


Figure B.11: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Bridge image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 156.04$, and $RMSE = 21.73$); (b) SPIHT-A ($CR = 156.13$, and $RMSE = 20.71$); (c) SPIHT-B ($CR = 156.13$, and $RMSE = 21.22$); and (d) rate-distortion curves.

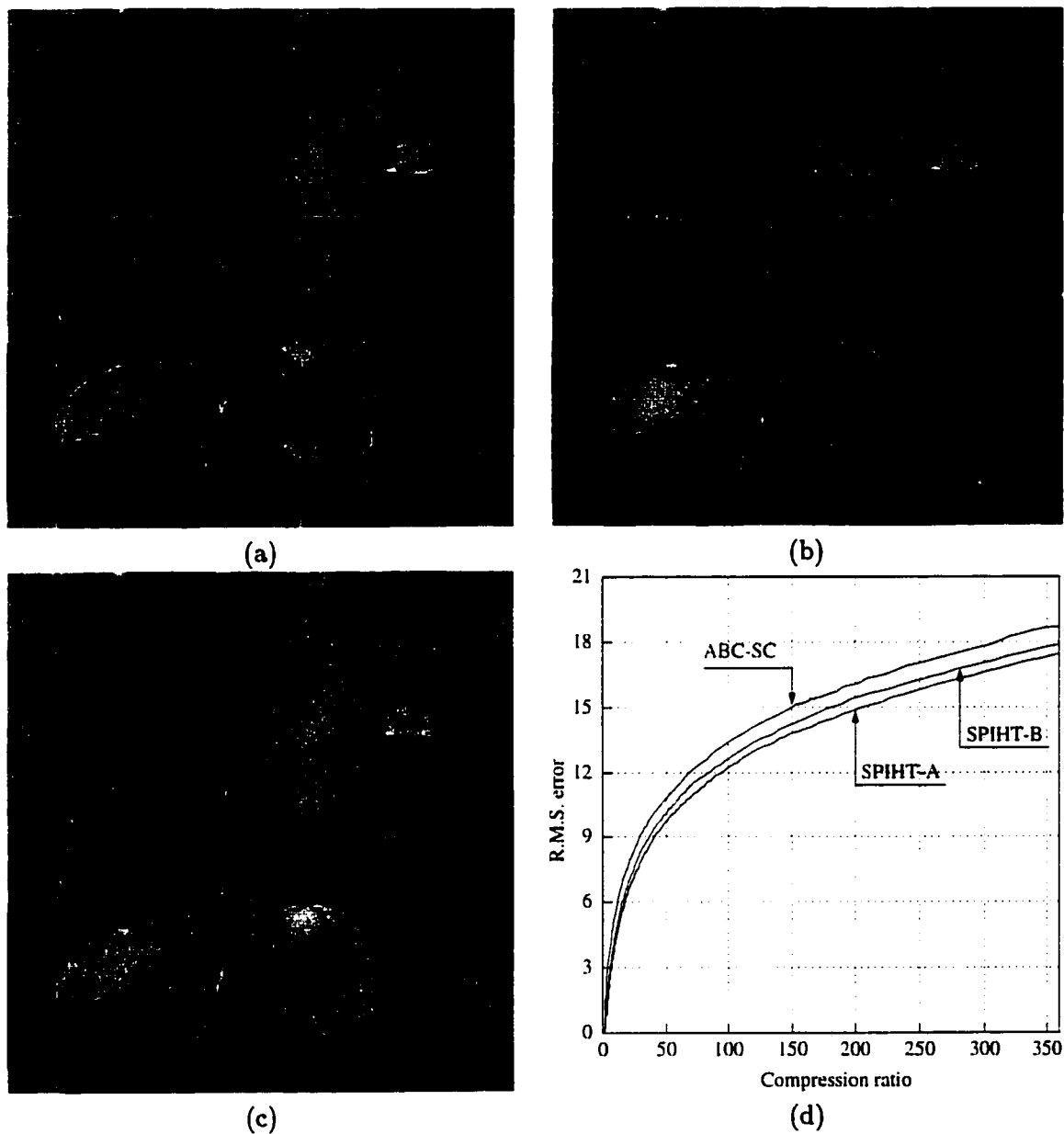


Figure B.12: ABC-SC, SPIHT-A, and SPIHT-B compression results for the Man image. (a) ABC-SC ($QF = 32$, $TQR = 1.00$, $CR = 222.53$, and $RMSE = 16.54$); (b) SPIHT-A ($CR = 222.53$, and $RMSE = 15.31$); (c) SPIHT-B ($CR = 222.34$, and $RMSE = 15.80$); and (d) rate-distortion curves.

Table B.1: A summary of rate-distortion performance for the decompressed images shown in Figures B.1–B.12.

Compression technique	Monarch		Boats		Rocks	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	104.03:1	17.61	212.09:1	16.38	156.22:1	17.93
SPIHT-A	104.03:1	14.89	211.92:1	15.11	156.22:1	16.21
SPIHT-B	104.03:1	15.77	212.09:1	15.60	156.22:1	16.87

Compression technique	Barbara		Zelda		Peppers	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	209.38:1	19.28	391.85:1	11.75	188.32:1	13.94
SPIHT-A	209.38:1	18.04	391.85:1	10.00	188.32:1	12.50
SPIHT-B	209.38:1	18.44	391.85:1	10.34	188.32:1	13.20

Compression technique	Goldhill		F16		Woman	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	310.23:1	15.63	203.53:1	15.84	422.13:1	9.74
SPIHT-A	310.23:1	14.54	203.53:1	14.30	421.45:1	8.25
SPIHT-B	310.23:1	14.79	203.53:1	15.06	422.13:1	8.82

Compression technique	Tulips		Bridge		Man	
	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>	<i>CR</i>	<i>RMSE</i>
ABC-SC	148.78:1	16.77	156.04:1	21.73	222.53:1	16.54
SPIHT-A	148.86:1	14.90	156.13:1	20.71	222.53:1	15.31
SPIHT-B	148.78:1	15.68	156.13:1	21.22	222.34:1	15.80

Bibliography

- [1] T. Cornsweet, *Visual Perception*, Academic Press, 1971.
- [2] R. Gonzalez and R. Woods, *Digital Image Processing*, Addison-Wesley, 1992.
- [3] X. Ran and N. Farvardin, "A Perceptually Motivated Three-Component Image Model—Part I: Description of the Model", *IEEE Transactions on Image Processing*, Vol. 4, No. 4, pp. 401–415, April 1995.
- [4] E. Delp and R. Mitchell, "Image Compression Using Block Truncation Coding", *IEEE Transactions on Communication*, Vol. COM-27, No. 9, pp. 1335–1342, September 1979.
- [5] P. Franti, O. Nevalainen, and T. Kaukoranta, "Compression of Digital Images by Block Truncation Coding: A Survey", *Computer Journal*, Vol. 37, No. 4, pp. 308–332, 1994.
- [6] H. Musmann, "Predictive Image Coding", in *Image Transmission Techniques* by W. Pratt (Ed.), Academic Press, pp. 73–112, 1979.
- [7] P. Wintz, "Transform Picture Coding", *Proceedings of the IEEE*, Vol. 60, No. 7, pp. 809–820, July 1972.
- [8] R. Clarke, *Transform Coding of Images*, Academic Press, 1985.

- [9] Y. Chien and K. Fu, "On the Generalised Karhunen-Loève Expansion", *IEEE Transactions on Information Theory*, Vol. IT-13, No. 3, pp. 518–520, July 1967.
- [10] N. Ahmed, T. Natarajan, and K. Rao, "Discrete Cosine Transform", *IEEE Transactions on Computers*, Vol. C-23, No. 1, pp. 90–93, January 1974.
- [11] K. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, and Applications*, Academic Press, 1990.
- [12] V. Algazi and D. Sakrison, "On the Optimality of the Karhunen-Loève Expansion", *IEEE Transactions on Information Theory*, Vol. IT-15, No. 2, pp. 319–321, March 1969.
- [13] W. Chen, C. Smith, and S. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform", *IEEE Transactions on Communications*, Vol. COM-25, No. 9, pp. 1004–1009, September 1977.
- [14] F. Kamangar and K. Rao, "Fast Algorithms for the 2-D Discrete Cosine Transform", *IEEE Transactions on Computers*, Vol. C-31, No. 9, pp. 899–906, September 1982.
- [15] B. Lee, "A New Algorithm to Compute the Discrete Cosine Transform", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, No. 6, pp. 1243–1245, December 1984.
- [16] N. Cho and S. Lee, "Fast Algorithm and Implementation of 2-D DCT", *IEEE Transactions on Circuits and Systems*, Vol. CAS-38, No. 3, pp. 297–305, March 1991.
- [17] E. Feig and S. Winograd, "Fast Algorithms for the Discrete Cosine Transform", *IEEE Transactions on Signal Processing*, Vol. 40, No. 9, pp. 2174–2193, September 1992.

- [18] A. Hung and T. Meng, "A Comparison of Fast Inverse Discrete Cosine Transform Algorithms", *Multimedia Systems*, Vol. 2, No. 5, December 1994.
- [19] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code", *IEEE Transactions on Communications*, Vol. COM-31, No. 4, pp. 532–540, April 1983.
- [20] M. Unser, "Improved Least Squares Laplacian Pyramid for Image Compression", *Signal Processing* Vol. 27, No. 2, pp. 187–203, May 1992.
- [21] B. Escalante-Ramirez, S. Venegas-Martinez, and F. Garcia-Ugalde, "Pyramidal Predictive Image Coding with Polynomial Transforms", *Still-Image Compression*, Vol. SPIE-2418, pp. 107–117, February 1995.
- [22] T. Yu, "Novel Contrast Pyramid Coding of Images", *IEEE International Conference on Image Processing (ICIP'95)*, Vol. 3, pp. 592–595, October 1995.
- [23] C. Chui, *An Introduction to Wavelets*, Academic Press, 1992.
- [24] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-11, No. 7, pp. 674–693, July 1989.
- [25] S. Mallat, "Multifrequency Channel Decompositions of Image and Wavelet Models", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-37, No. 12, pp. 2091–2110, December 1989.
- [26] J. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients", *IEEE Transactions on Signal Processing*, Vol. 41, No. 12, pp. 3445–3462, December 1993.
- [27] A. Said and W. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Tree", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 3, pp. 243–250, June 1996.

- [28] D. Pearson and J. Robinson, "Visual Communication at Very Low Bit Rates", *Proceedings of the IEEE*, Vol. 73, No. 4, pp. 795–812, April 1985.
- [29] S. Carlsson, "Sketch Based Coding of Grey Level Images", *Signal Processing* Vol. 15, No. 1, pp. 57–83, July 1988.
- [30] A. Pentland and B. Horowitz, "A Practical Approach to Fractal Compression", *Visual Communications and Image Processing '90*, Vol. SPIE-1605, pp. 467–474, November 1991.
- [31] A. Jacquin, "Fractal Image Coding: A Review", *Proceedings of the IEEE*, Vol. 81, No. 10, pp. 1451–1465, October 1993.
- [32] A. Habibi, "Hybrid Coding of Pictorial Data", *IEEE Transactions on Communications*, Vol. COM-22, No. 5, pp. 614–624, May 1974.
- [33] Y. Zhao and B. Yuan, "Hybrid Image Compression Scheme Combining Block-based Fractal Coding and DCT", *Signal Processing: Image Communication*, Vol. 8, No. 2, pp. 73–78, March 1996.
- [34] X. Ran and N. Farvardin, "A Perceptually Motivated Three-Component Image Model—Part II: Applications to Image Compression", *IEEE Transactions on Image Processing*, Vol. 4, No. 4, pp. 430–447, April 1995.
- [35] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, pp. 591–598, 1990.
- [36] J. Max, "Quantizing For Minimum Distortion", *IRE Transactions on Information Theory*, Vol. IT-6, No. 1, pp. 7–12, March 1960.
- [37] S. Lloyd, "Least Squares Quantization in PCM", *IEEE Transactions on Information Theory*, Vol. IT-28, No. 2, pp. 129–137, March 1982.

- [38] C. Shannon, "Coding Theorems For Discrete Source with a Fidelity Criterion", in *IRE National Convention Record*, Part 4, pp. 142–163, 1959.
- [39] Y. Linde, A. Buzo, and R. Gray, "An Algorithm For Vector Quantizer Design", *IEEE Transactions on Communications*, Vol. COM-28, No. 1, pp. 84–95, January 1980.
- [40] N. Nasrabadi and R. King, "Image Coding Using Vector Quantization: A Review", *IEEE Transactions on Communications*, Vol. COM-36, No. 8, pp. 957–971, August 1988.
- [41] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.
- [42] W. Equitz, "Fast Algorithms for Vector Quantization Picture Coding", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'87)*, Vol. 2, pp. 725–728, April 1987.
- [43] D. Cheng and A. Gersho, "A Fast Codebook Search Algorithm For Nearest-Neighbor Pattern Matching", *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'86)*, Vol. 1, pp. 265–268, April 1986.
- [44] L. Guan, "Fast Algorithms for Vector Quantization of Image Data", *Ph.D. Dissertation*, Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, 1991.
- [45] M. Fu and J. Mui, "A Survey on Image Segmentation", *Pattern Recognition*, Vol. 13, No. 1, pp. 3–16, 1981.
- [46] R. Haralick and L. Shapiro, "Survey, Image Segmentation Techniques", *Computer Vision, Graphics, and Image Processing*, Vol. 29, No. 1, pp. 100–132, January 1985.

- [47] P. Sahoo, S. Soltani, A. K. C. Wong, and Y. Chen, "A Survey of Thresholding Techniques", *Computer Vision, Graphics, and Image Processing*, Vol. 41, No. 2, pp. 233–360, February 1988.
- [48] M. Pal and S. Pal, "A Survey on Image Segmentation Techniques", *Pattern Recognition*, Vol. 26, No. 9, pp. 1277–1294, September 1993.
- [49] H. Samet, "The Quad-Tree and Related Data Structures", *ACM Computing Surveys*, Vol. 16, No. 2, pp. 188–260, June 1984.
- [50] C. Chen, "Adaptive Transform Coding Via Quadtree-Based Variable Blocksize DCT", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'89)*, Vol. 3, pp. 1854–1857, May 1989.
- [51] P. Nasiopoulos, R. Ward, and D. Morse, "Adaptive Compression Coding", *IEEE Transactions on Communications*, Vol. 39, No. 8, pp. 1245–1254, August, 1991.
- [52] M. Lee and G. Crebbin, "Classified Vector Quantisation with Variable Block-Size DCT models", *IEE proceedings: Vision, Image and Signal Processing*, Vol. 141, No. 1, pp. 39–48, February 1994.
- [53] J. Vaisey and A. Gersho, "Image Compression with Variable Block Size Segmentation", *IEEE Transactions on Signal Processing*, Vol. 40, No. 8, pp. 2040–2060, August 1992.
- [54] X. Wu, "Image Coding by Adaptive Tree-Structured Segmentation", *IEEE Transactions on Information Theory*, Vol. IT-38, No. 6, pp. 1755–1767, November 1992.
- [55] H. Radha, M. Vetterli, and R. Leonardi, "Image Compression Using Binary Space Partitioning Trees", *IEEE Transactions on Image Processing*, Vol. 5, No. 12, pp. 1610–1624, December 1996.

- [56] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-Generation Image-Coding", *Proceedings of the IEEE*, Vol. 73, No. 4, pp. 549–574, April 1985.
- [57] M. Kunt, M. Benard, and R. Leonardi, "Recent Results in High-Compression Image Coding", *IEEE Transactions on Circuits and Systems*, Vol. CAS-34, No. 11, pp. 1306–1336, November 1987.
- [58] H. Freeman, "On the Encoding of Arbitrary Geometric Configuration", *IRE Transactions on Electronic Computers*, Vol. EC-10, pp. 260–268, June 1961.
- [59] D. Neuhoff and K. Castor, "A Rate and Distortion Analysis of Chain Codes for Line Drawings", *IEEE Transactions on Information Theory*, Vol. IT-31, No. 1, pp. 53–67, January 1985.
- [60] G. K. Wallace, "The JPEG Still Picture Compression Standard", *IEEE Transactions on Consumer Electronics*, Vol.38, No. 1, pp.xviii–xxxiv, February 1992.
- [61] M. El-Sakka and M. Kamel, "Adaptive Image Compression Based on Segmentation and Block Classification", submitted to *IEEE International Conference on Image Processing (ICIP'97)*.
- [62] M. El-Sakka and M. Kamel, "Adaptive Image Compression Based on Segmentation and Block Classification", submitted to *IEEE Transactions on Image Processing*.
- [63] S. Golomb, "Run-Length Encodings", *IEEE Transactions on Information Theory*, Vol. IT-12, No. 3, pp. 399–401, July 1966.
- [64] J. Gimlett, "Use of 'Activity Classes' in Adaptive Transform Image Coding", *IEEE Transactions on Communications*, Vol. COM-23, No. 7, pp. 785–786, July 1975.
- [65] W. Chen and W. Pratt, "Scene Adaptive Coder", *IEEE Transactions on Communications*, Vol. COM-32, No. 3, pp. 225–232, March 1984.

- [66] J. Woods and S. O'Neil, "Subband Coding of Images", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-34, No. 5, pp. 1278–1288, October 1986.
- [67] M. El-Sakka and M. Kamel, "A Segmentation Criterion for Digital Image Compression", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'95)*, Vol. 4, pp. 2551–2554, May, 1995.
- [68] I. Witten, R. Neal, and J. Cleary, "Arithmetic Coding for Data Compression", *Communications of ACM*, Vol. 30, No. 6, pp. 520–540, June 1987.
- [69] T. Bell, J. Cleary, and I. Witten, *Text Compression*, Prentice Hall, 1990.
- [70] D. Huffman, "A Method For the Construction of Minimum Redundancy Codes", *Proceedings of the IRE*, Vol. 40, pp. 1098–1101, September 1952.
- [71] H. Reeve and J. Lim, "Reduction of Blocking Effects in Image Coding", *Optical Engineering*, Vol. 23, No. 1, pp. 34–37, January 1984.
- [72] K. Tzou, "Post-Filtering of Transform-Coded Images", *Applications of Digital Image Processing XI* Vol. SPIE-974, pp. 121–126, August 1988.
- [73] B. Ramamurthi and A. Gersho, "Nonlinear Space-Variant Post-Processing of Block Coded Images", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-34, No. 5, pp. 1258–1268, October 1986.
- [74] K. Sauer, "Enhancement of Low Bit-Rate Coded Images Using Edge Detection and Estimation", *Computer Vision Graphics and Image Processing: Graphical Models and Image Processing*, Vol. 53, No. 1, pp. 52–62, January 1991.
- [75] D. Slawewski and W. Li, "DCT/IDCT Processor Design for High Data Rate Image Coding", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 2, No. 2, pp. 135–146, June 1992.

- [76] C. Wang and C. Chen, "High-Throughput VLSI Architecture for the 1-D and 2-D Discrete Cosine Transforms", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, No. 1, pp. 31–40, February 1995.
- [77] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI Architectures for Video Compression—A Survey", *Proceedings of the IEEE*, Vol. 83, No. 2, pp. 220–246, February 1995.
- [78] Y. Chang and C. Wang, "New Systolic Array Implementation of the 2-D Discrete Cosine Transform and its Inverse", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, No. 2, pp. 150–157, April 1995.