①

A New Method for Image Compression and Progressive Transmission

（画像圧縮と順次再生のための新しい方法）

by

Martin J. Dürst

Dissertation Submitted to the
Graduate School of the
University of Tokyo
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Science
in Information Science

June 1990

A New Method for Image Compression and Progressive Transmission

(画像圧縮と順次再生のための新しい方法)

by

Martin J. Dürst

Dissertation Submitted to the
Graduate School of the
University of Tokyo
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Science
in Information Science

June 1990

A New Method for Image Compression and Progressive Transmission

by

Martin J. Dürst

Dissertation Submitted to the
Graduate School of the
University of Tokyo
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Science
in Information Science

June 1990

Abstract

Image compression has been a field of intensive research already for a long time. In recent years, progressive transmission of images has attracted considerable attention. Progressive transmission encodes and transmits an image in such a way that the receiver can get the best overall impression at an early time. This can lead to a net increase in bandwidth especially for slow transmission lines, but is usually obtained at the cost of a highly increased computational complexity or a much lower compression ratio than straightforward compression at a fixed rate.

In this thesis, progressive transmission is viewed in a more general way. Rate distortion theory applied to progressive transmission shows that in many cases, progressive transmission is theoretically possible without any overhead. As a new result, it is proved that in the high rate case, the cost of progressive transmission will in any case not be more than 1.8753 db (in terms of the mean square error). Also, the analysis of the requirements for an image compression algorithm shows that progressiveness of a compression method is a highly desirable property for most applications, especially in a heterogeneous environment.

Based on these requirements, a new method for image compression and progressive transmission is proposed. The basic principle of this method is the combination of hierarchical subdivision in the spatial domain (sampling) and in the gray scale domain (quantization), called hierarchical sampling restricted quantization (HSRQ).

The resulting image structure is captured in the bitwise condensed quadtree (BC quadtree) and the gray scale depth first expression (GDF). By dividing the image information into components based on the level in the spatial and in the gray scale hierarchy, it is possible to increase both spatial and gray scale resolution concurrently in a way well adapted to the human visual system.

The selection of the representatives for a given gray scale interval and the definition of the exact component sequence allow to optimize performance and to adapt transmission to the properties of the image, the needs of the user, and the possible bottlenecks in sender, transmission line, and receiver with a flexibility uncommon for other transmission methods. Additional performance gains are possible by optimizing the gray scale hierarchy or by using arithmetic coding.

A wide variety of experiments show that the new method in the present implementation performs comparable to methods like vector quantization and nonadaptive transform coding while being considerably less complex. Additional improvements are possible and are pointed out throughout the thesis. This new method of image compression and progressive transmission may therefore well become one of the methods of choice for a wide area of applications.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my thesis advisor Prof. Tosiyasu L. Kunii for the great confidence, encouragement, and support he always showed me, and for his valuable advice on countless occasions. I will always keep in mind his approach to science and research, not affraid to cross long established boundaries and discover new teritories, and always striving to be at the first front of new developments.

For all the advice during the examination of this thesis, I warmly thank the president and the members of the examination committee, Profs. Hiroshi Imai, Takashi Masuda, Akinori Yonezawa, Mamoru Maekawa, and Ken Sakamura.

My thanks also go to the past and present research associates of the Kunii Laboratory: Dr. Issei Fujishiro, now at the University of Tsukuba, Dr. Yoshiaki Takai, now at the University of Hokkaido, Yasuto Shirai, initially my tutor, and now at the University of Shizuoka, and Yoshihisa Shinagawa. They were always ready to help me with any kind of problem.

I am also very grateful to the many past and present members of the Kunii Laboratory. There friendship, help, and advice during all these years is unforgettable. These thanks are extended to the staff of the laboratory and the department.

For many interesting comments and suggestions in various stages of this work, I would like to thank Prof. Francis Chin of the University of Hong Kong, Prof. Hanan Samet of the University of Maryland, Prof. Gio Wiederhold, Dr. Eve Riskin, Dr. Will Equitz, and Dr. Ramin Samadani of Stanford University, Prof. Allen Klinger of UCLA, Prof. Satoru Kawai of the College of Arts and Sciences of the University of Tokyo, and Profs. Shun-ichi Amari and Suguru Arimoto of the Faculty of Engineering of the University of Tokyo.

Dr. Eve Riskin, together with Prof. Masao Sakauchi of the Institute of Industrial Science, University of Tokyo, again receives my thanks for providing the images used in the experiments. The help of the senior students of the Department of Information Science, University of Tokyo in the face recognition experiment of Section 6.5 was invaluable.

For the many equipment used, my thanks go to Kansei Iwata, president of Graphica Co., Ltd., and Norimasa Koyama, executive director of Cadtech Inc., for the high resolution drum scanner G-225C, Asahi Technocomputer, Ltd., for the AsahiStellar GS-1000 Graphics Super Workstation, and Dr. Hideko Kunii, director of the Software Research Center (SRC) of Ricoh Co., Ltd., for the Vax-750.

During the three and a half years of my stay in Tokyo, I was supported by a scholarship of the Japanese Ministry of Science, Education, and Culture

(Monbusho). The generosity of the Japanese Government made it possible for me to fully concentrate on my research.

For the application to the above scholarship, and during my whole stay in Japan, I have also received the support and encouragement of Profs. Peter Stucki, Kurt Bauknecht, and Edwin Rühli of the University of Zürich, and Prof. Jürg Nievergelt of the ETH Zürich, which is herewith warmestly acknowledged.

Last but not least, my parents have always supported me with their concern and encouragement. This thesis is dedicated to them.

Table of Contents

Abstract iii
Acknowledgments v
Table of Contents vii
List of Figures xi
List of Tables xiii

1 Introduction 1
1.1 Motivation 1
1.2 Historical Background and Related Work 3
1.2.1 Image Compression 3
1.2.2 Quadtrees 7
1.2.3 Progressive Transmission 12
1.3 Outline of the Thesis 18

2 Rate Distortion Theory for Progressive Transmission 19
2.1 Mathematical Foundations 19
2.1.1 Probability 19
2.1.2 Random Variables, Processes, and Fields 21
2.1.3 Information and Entropy 22
2.1.4 Mutual Information 22
2.1.5 Mutual Information for Three Variables 23
2.2 Rate Distortion Theory 24
2.2.1 The Model of Transmission 24
2.2.2 The Rate Distortion Function 25
2.2.3 The Rate Distortion Function for a Composite Source 27
2.3 Lower Bounds 28
2.3.1 Successive Refinement 28
2.3.2 Conditions for Successive Refinement 29
2.3.3 Discussion 32
2.4 An Upper Bound 33
2.4.1 High Rate Vector Quantization Advantages 33
2.4.2 The Space Filling Disadvantage For Progressive Transmission 34
2.4.3 Discussion 37

3 Requirements 39
3.1 Application Overview 39
3.1.1 Applications of Image Compression 39
3.1.2 Channel Structure 40
3.1.3 Applications of Progressive Transmission 42
3.2 Requirement Analysis 45
3.2.1 Efficiency 45
3.2.2 Soft Flexibility 47
3.2.3 Hardware Flexibility 50

4 The New Method 53
4.1 The Bitwise Condensed (BC) Quadtree 53
4.1.1 The Concept of the Bitwise Condensed Quadtree 53
4.1.2 Internal Storage 54
4.1.3 Construction of the BC Quadtree 56
4.2 The Gray Scale Depth First Expression (GDF) 57
4.2.1 DF and GDF 57
4.2.2 Binary Coding of GDF 58
4.3 Image Components 58
4.3.1 Partitioning of the Image into Components 59
4.3.2 Increasing Gray Scale Resolution 60
4.3.3 Increasing Spatial Resolution 61
4.3.4 Increasing Both Spatial and Gray Scale Resolution 62
4.3.5 Transmission Algorithms 63
4.4 Hierarchical Sampling Restricted Quantization (HSRQ) 65
4.4.1 Sampling and Quantization with PCM 65
4.4.2 Interval Coding 66
4.4.3 Hierarchical Intervals 67
4.4.4 Quantization Restricted by Hierarchical Sampling 68
4.5 Traces 70
4.5.1 Traces for Pixels 71
4.5.2 The Trace Tree 71
4.5.3 Subtraces 72
4.5.4 General Conditions for Decodability 73
4.6 Comparison with Other Methods 74
4.6.1 General Coding Principles 74
4.6.2 Image Complexity 76
4.6.3 Quadtree Oriented Methods 77

5 Optimization 81
5.1 Gray Level Reproduction 81
5.1.1 Black to White Reproduction 81
5.1.2 Random Background Reproduction 83
5.1.3 Center Value Reproduction 84
5.1.4 Average Reproduction 84
5.1.5 Global and Predictive Reproduction 91
5.2 Optimizing Component Sequence 92
5.2.1 The Optimal Component Sequence 92
5.2.2 Component Sequence Heuristics 93
5.2.3 Other Ways to Define Components 95
5.3 Optimizing the Gray Scale Hierarchy 97
5.3.1 Basics 97
5.3.2 Interval Statistics 98
5.3.3 Finding the Optimum 99
5.3.4 Considering Binary Coding 101
5.3.5 Display 102
5.3.6 Related Work 103
5.4 Arithmetic Coding 104

6 Experiments 107
6.1 Compression Methods Used 107
6.1.1 Previously Existing Methods 107
6.1.2 GDF Variants 109
6.2 Overall Compression 110
6.3 Visual Evaluation 112
6.4 Analytical Evaluation 115
6.5 Recognition of Faces 121
6.5.1 Experiment Setup 122
6.5.2 Results 123
6.5.3 Comparison with Transform Coding 125
6.5.4 Comparison with Perception Experiments 126

7 Extended Applications 129
7.1 Imprecise Data 129
7.2 Color Images 130
7.2.1 Basic Color Science 131
7.2.2 Cubic Color Spaces 133
7.2.3 Results 137
7.3 Image Sequences 139

8 Conclusions 141
8.1 Summary 141
8.2 Directions for Future Work 142

Appendices 143
Appendix A: Component Sequences 143
Appendix B: Results for Image "Lena" 144
Appendix C: Images Used In Face Recognition Experiments 146

Bibliography 149

Index 163

List of Figures

Figure 1.1. Example image 7
Figure 1.2. Quadtree for the image of Figure 1.1 8
Figure 1.3. G-quadtrees for the image of Figure 1.1 11
Figure 1.4. The natural gray scale hierarchy 11
Figure 1.5. The subsampling scheme of Endoh et al 14

Figure 2.1. Transmission system model 25
Figure 2.2. A typical rate distortion curve 26
Figure 2.3. Rate distortion curves for progressive transmission 28
Figure 2.4. Relative quantization error for different polytopes 37

Figure 3.1. Diagram to characterize a compression method 46

Figure 4.1. Quadtree for the image of Figure 1.1 54
Figure 4.2. BC quadtree for the image of Figure 1.1 54
Figure 4.3. The explicit quadtree for the BC quadtree of Figure 4.2 55
Figure 4.4. The BC quadtree with parentheses 57
Figure 4.5. The indexed BC quadtree 59
Figure 4.6. Progressive transmission increasing gray scale resolution 61
Figure 4.7. Progressive transmission increasing spatial resolution 62
Figure 4.8. Increasing both spatial and gray scale resolution 63
Figure 4.9. Sampling and quantizing a function 66
Figure 4.10. Approximating a function by intervals 66
Figure 4.11. Two ways of splitting and reducing intervals 68
Figure 4.12. Interval refinement, and corresponding symbols of GDF 68
Figure 4.13. Approximating a function by HSRQ 69
Figure 4.14. A set of refinements not leading to resolution problems 70
Figure 4.15. Traces for the example image of Figure 1.1 71
Figure 4.16. The trace tree for the example image 71
Figure 4.17 Comparison of transform coding and HSRQ 75

Figure 5.1. Black to white reproduction 82
Figure 5.2. Random background reproduction 84
Figure 5.3. Smallest spatial intervals quantized with one level 85
Figure 5.4. Smallest spatial intervals quantized with two levels 85
Figure 5.5. Reproduction using trace averages 90
Figure 5.6. Example image for optimization 98
Figure 5.7. Values of N*, T, D, and R in optimization algorithm 100

Figure 5.8. Optimal gray scale hierarchy for image of Figure 5.6 101
Figure 5.9. Progressive transmission with optimized gray scale hierarchy 102
Figure 5.10. Segmentation by optimizing the gray scale hierarchy 104

Figure 6.0. Component sequences o2, o3, and o4 109
Figure 6.1. Originals of the images from the Standard Image Data Base 111
Figure 6.2. Transmitting the image "Girl" with Dreizen's method 112
Figure 6.3. The image "Girl" using eight different methods 113
Figure 6.4. The image "X-ray" using eight different methods 113
Figure 6.5. The image "Fax" using eight different methods 114
Figure 6.6. PSNR of the image "Girl" for eight methods 116
Figure 6.7. PSNR of the image "Girl" for component sequence o5 117
Figure 6.8. (unused)
Figure 6.9. PSNR for the image "Lena" 119
Figure 6.10. Average SNR for five brain scans 120
Figure 6.11. Progressive transmission of an MR image 121
Figure 6.12. Average PSNR of the faces for each method used 124

Figure 7.1. Gray scale hierarchy allowing imprecision 130
Figure 7.2. The RGB color cube 131
Figure 7.3. Rotation of a square with the generalized ring method 134
Figure 7.4. The RGB color cube rotated by 45° about the G axis 135
Figure 7.5. The RGB color cube rotated by 45° about the B axis 135
Figure 7.6. The color cube of Figure 7.5 rotated by 45° about the R axis 135
Figure 7.7. The color cube of Figure 7.6 rotated by 45° about the G axis 136
Figure 7.8. The RGB color cube rotated by 30° about the B axis 137
Figure 7.9. The color cube of Figure 7.8 rotated by 30° about the R axis 137

Figure B.1. "Lena" at 1/64 and 1/32 bits per pixel 144
Figure B.2. "Lena" at 1/16 and 1/8 bits per pixel 145
Figure B.3. "Lena" at 1/4 and 1/2 bits per pixel 145
Figure B.4. "Lena" at 1 and 1.5 bits per pixel 145
Figure B.5. "Lena" at 2 and 8 bits per pixel 146

Figure C.1. Senior students of the Department of Information Science 146
Figure C.2. Senior students of the Department of Information Science 146
Figure C.3. Senior students of the Department of Information Science 147
Figure C.4. Famous personalities 147
Figure C.5. Famous personalities and the author 147

List of Tables

Table 4.1. Components of the example image 59
Table 4.2. Size (number of symbols) of components for image "Girl" 60

Table 5.1. Reproduction averages for image "Girl" 87

Table 6.1. Overall compression rate for various images and methods 111
Table 6.2. PSNR for various images and methods at 2 bits per pixel 118
Table 6.3. Average number of bits necessary to recognize a face 123
Table 6.4. Number of bits for person recognition using DCT 126

1

Introduction

1.1 Motivation

In image processing and computer vision, image compression forms one of the basic applications and one of the basic tools for other, higher level tasks. As a tool, compression helps to reduce the amount of data that has to be processed at higher levels like enhancement, matching, recognition, etc. As an independent application, compression is very useful in reducing bandwidth on communication channels and storage requirements for mass storage.

Image compression, and data compression in general, have a long history, and the amount of literature on the subject can not be overviewed easily any more. On the other side, the field has not at all reached its theoretical boundaries. Also, storage and computation capacity grow at unchanged speed, but are hardly able to keep up with the imagination of man devising new applications, and with the economic pressure to realize long foreseen ones.

Besides these outside motivations, there is, I believe, an inner motivation that attracts researchers to the field of data compression. Man has always been fascinated by the power of a poet to say much with few words. To separate the important from the meaningless is a truly human activity. To find ways to do something similar, even if at a much lower level, and to try to do it in a simple and straightforward manner, seems clearly worth the time and effort.

Owing to the large amount of research in the field of image and data compression, much work has concentrated on investigations for particular applications, with their particular image characteristics, quality requirements, and abilities or needs with regard to hardware and software.

However, recently, there is an increasing trend towards integration. Multimedia applications integrate data and operations of all kinds. Local, wide area,

and global networks connect computing facilities all over the world. This trend towards increasing integration is already considerable, but compared with the developments that can be expected in the future, it is only a first small step.

This has important consequences for the development of image coding methods. Instead of trying to develop algorithms suited for a very specialized application, general algorithms are necessary. An ideal coding algorithm should be usable both in hardware and in software, and on a wide variety of machine configurations. Also, to be usable on networks with different bandwidth, the coding method has to be progressive.

The present thesis proposes a new basic method of image coding. It is much more flexible in several aspects than previous methods, while being surprisingly simple and efficient. Although there exists nothing such as an ultimately optimal coding method, even for clearly defined requirements, I think that the proposal of this thesis is a definitive step towards better coding methods usable in a heterogeneous environment.

Nonetheless, some restrictions have to be made. First, the main subject of this work are gray scale still images; binary images, color images, and image sequences are discussed only marginally. Second, it is generally assumed that the channels and storage media work reliably and without error. Third, when not mentioned otherwise, it is assumed that the image is clear and free of noise, put to the extreme, that it cannot be enhanced anymore by any known technique. This is an assumption frequently made, but rarely mentioned.

During the development that led to the work presented in this thesis, it was many times an algorithm or a data structure in search of an application, rather than a method developed to answer clearly defined requirements. On a day to day basis, the progress of my research has been motivated, or directed and influenced, as follows:

The initial motivation has clearly been some inefficiency that intrigued me in the G-quadtree [Kni86], [Mao87] (Subsection 1.2.2.5). This lead to the development of GDF and the BC quadtree (Sections 4.1 and 4.2), first described in a report for the lecture of Prof. Kawai in Spring 1988, and later in [Dür88]. At that time, GDF was intended as a method for lossless compression.

Also, in winter 1987/88, the algorithm later published as [Dür89c] (Section 5.3) was devised and later implemented. The fact that overall compression ratios were not extremely high let to the consideration of GDF and the BC quadtree for progressive transmission (Section 4.3). Implementation in fall 1989 for a first publication [Dür89d] showed fairly good results.

The investigation of some discontinuities during the reproduction of the transmitted images in April 1990 revealed how additional improvements could be obtained (Section 5.1). This motivated me to study source coding theory and rate distortion theory and to try to apply it to progressive transmission in an effort towards unifying straightforward compression and progressive transmission (Chapter 2). It also led to the formulation of the HSRQ principle for combined hierarchical sampling and quantization (Section 4.4). Experiments showed that the gap between complex coding methods with fixed rate and simple methods for progressive transmission could be smaller than generally assumed (Chapter 6).

1.2 Historical Background and Related Work

The historically oriented overview in this section is divided into three subsections. The first subsection discusses image compression in general. The second subsection is devoted to the quadtree and related data structures. The third subsection treats progressive transmission. In the framework of this thesis, progressive transmission is treated as a natural consequence of a good compression scheme, but historically, progressive transmission was considered to be antagonistic to compression, and so it is discussed separately here.

1.2.1 Image Compression

The large amount of literature on image compression has already been mentioned. This subsection intends only to give a very short overview of the available methods and their performance; the interested reader is referred to the bibliography at the end of Chapter 5 in [Ros82], and some newer overviews like the review of Jain [Jai81] and a Special Issue of the Proceedings of the IEEE [IEE85].

1.2.1.1 Pulse Code Modulation

More than a form of data compression, pulse code modulation (PCM) is first and foremost the basic way in which images are digitized, stored in a computer, and displayed on a CRT display or printed on a printing device. The originally analog image is sampled with a square or rectangular grid. Then the samples at each grid location (picture elements, pixels, or pels) are quantized to a certain number of bits and represented in array form. This array is usually denoted as the canonical form of the image [Knt85].

Throughout this thesis, it will be assumed that this canonical form is the original. This provides a clear base of reference; in fact, with the current trend towards digital electronics, a persistent analog form of the image may not exist.

The main problem that has to be solved for PCM is the selection of the correct number of samples and gray levels that are necessary for a faithful and efficient representation of an image.

One way to determine the necessary spatial resolution (the number of samples in each direction) and gray scale resolution (the number of gray levels) is to set them somewhat higher than the maximal resolution capabilities of the human visual system. However, spatial and gray scale resolution are actually not independent. Spatial resolution is used to compensate for lacking gray scale resolution in applications like dithering and digital halftoning. On the other hand, gray scale fonts have recently become popular to balance the low spatial resolution of CRT screens.

The complexity of the relation between spatial and gray scale resolution has been pointed out by Huang [Hua65]. He showed that the relation between spatial and gray scale resolution considered optimal by a human viewer depends on the image contents, and is in no ways linear (different relations are preferred at different levels) or even convex (increase of resolution is not always perceived as an improvement in image quality). Huang attributes this mainly to the fact that a lower number of gray levels can increase contrast and thus enhance image quality.

Although a long time has passed since Huang's work, the relation between spatial and gray scale resolution is not really clear even now. Recently, some theoretical work has started in this area [Bru87], [Pel89], but it is difficult to find a way to a practical application of it.

Besides reducing the necessary memory by restricting sampling and quantization, there have been other attempts to compress an image based on PCM. The first mentioned here is block coding, devised by Kunt [Knt78]. It consists in assigning very short codes to very frequent block patterns, for example completely white or completely black blocks. It is therefore especially suited for binary images, and has been used for gray level images mainly by coding each bit plane separately.

Block truncation coding [Del79] also splits an image into blocks, and then uses two representative values in each block. These representatives are chosen so as to conserve mean and variance, or some other moments, inside the block. The representatives and the selection of the representative for each pixel are transmitted. Block truncation coding achieves good results at a rate of usually 2 bits/pixel with very low complexity.

The most general approach to PCM is vector coding or vector quantization. A statistic of all blocks of a training sequence of representative images is made.

Then a number of representative vectors, optimally distributed in the multidimensional sample space, is determined. The set of representative vectors is called the codebook. The number of the best-fitting vector for each block is transmitted.

Encoding with general codebooks is very time consuming, and so in many cases tree structured codebooks are used [Ris90]. Vector quantization can on the limit achieve the theoretical bounds for compression (see Chapter 2), but it is limited by the size of the codebook. Vector quantization in the context of image compression can also be used in many other situations than PCM [Nas88].

1.2.1.2 Differential Pulse Code Modulation

Differential pulse code modulation (DPCM) [Hun79] is based on the observation that neighboring pixels of an image frequently have similar values. A a concentration of the histogram can thus be achieved when not the pixels themselves, but their differences, are transmitted.

DPCM is the term used for such techniques in the field of signal processing. In image processing, the term predictive coding is more popular. It is due to the fact that in most variants, not the difference to a previous pixel, but the difference to the predicted value of the present pixel is used for coding.

Predictive coding has the advantage that it is usable in a range from satisfactory quality with 1 bit per pixel (so called delta modulation) to perfect (error-free) reproduction with usually about three to five bits/pixel. Also, it is easily implemented with a minimum of memory. On the other hand, in its basic form, it is the typical example of a nonprogressive transmission method. At the receiver, the image is displayed line by line and pixel by pixel.

1.2.1.3 Transform Coding

Transform coding is based on the theory of linear systems. The linear dependence in a block of pixel values is eliminated by an orthogonal transform (geometrically, a rotation of the sample space). The resulting values are requantized and transmitted. Compression is achieved because the variance of each coefficient after transformation, and its contribution to the quality of the image, varies greatly with its order. Thus a different number of bits can be used to quantize each component.

Several transforms have been proposed, but at present, the discrete cosine transform is the most used. It comes closest to the theoretically optimal Karhunen-Loève transform while being separable [Ahm74]. Transform coding achieves good results, but requires a considerable amount of calculation.

1.2.1.4 Hybrid and Adaptive Coding

To combine the advantages and reduce the disadvantages of the basic coding methods, a wide variety of hybrid coding methods has been proposed. Also, for most methods of image compression, improvement is possible by adapting some parameters of the method to the local statistical variations of the image.

1.2.1.5 Advanced Coding Methods

In recent years, a number of advanced methods for the coding of images has been developed. Their common feature is that they do not see the image as a simple heap of numbers that are statistically related in some way. Instead, they incorporate knowledge of the human visual system, the "final destination" of a coded image. A good overview of the human visual system and these methods can be found in the review by Kunt [Knt85].

The main property of the human visual system used by these methods is the the fact that a relevant part of the vision process seems to be based on the extraction of edges and directional information from the image. Thus good approximation and compression of an image can be obtained by detecting edges and regions, coding the edges as line or circle fragments, and approximating the gray levels in the regions with constants or two dimensional polynomial functions. Related methods use combinations of low pass filters and directional filters. For a more detailed explanation, please see [Knt85].

Recently, coding rates of 150:1 and more have been reported for these methods [NZZ90]. This does not mean that the same quality as with the traditional methods is achieved; the appearance of the reproductions is in many cases rather sketchlike. It however means that these methods may give a usable approximation of the image at such low rates, something which is hardly possible with traditional methods.

Besides the necessary high amount of computation needed, these methods may have other problems. They are not really useful for obtaining more accurate approximations of the image. It is also not clear whether it is advisable to have the computer simulate and maybe preempt some of the processing which the human visual system is best suited for.

The Gabor transform, proposed for image coding by Daugman [Dau88], may be seen as a combination of the visual system oriented techniques and transform coding.

1.2.1.6 Fractals

Image compression using deterministic fractals has recently attracted great attention [Bar88], [Zor88]. Compression factors of 10,000 to 1 and higher are

promised. However, for the time being, the amount of resources necessary to code and decode images, both in terms of manpower and computing time, is high, and the decoded images "resemble impressionistic renderings of their originals, rather than photographic copies" [Zor88]. It is impossible to foresee if, how, or when better results will be obtained. Fractals can therefore not at the moment be considered a serious alternative for image compression.

1.2.2 Quadtrees

An overview of quadtrees, with concentration on its application to gray scale images, is included here for two reasons: First, it should provide the necessary background for those readers not familiar with hierarchical data structures like the quadtree. Second, quadtrees were the starting point of the research that led to the results described in this thesis. In some aspects, it can actually be seen as part of a greater effort to identify principles and techniques for the design of efficient quadtree data structures and algorithms [Dür90b], [Dür90a].

The new method presented in this thesis is not in any way limited to the quadtree as a spatial hierarchy. However, the quadtree is the simplest and most popular hierarchy to which the new method can be applied. It is therefore used throughout most parts of this thesis.

1.2.2.1 What is a Quadtree?

The term quad tree was first used by Finkel and Bentley [Fin74] for what is now called the point quadtree: A recursive subdivision of space that uses the data points to determine the locations of the subdivisions. Nowadays, the term quadtree is mainly identified with the recursive, regular, and rectangular subdivision of a square area of interest called the universe. In this thesis, the word quadtree will always be used in this sense.

A very simple example of a gray scale image and its quadtree is shown in Figures 1.1 and 1.2. A broad overview over quadtrees and related data structures can be found in the reviews by Samet [Sam84,88a,b,90a,b].

000	001	011	011
001	001	011	011
010	011	100	100
010	011	101	110

Figure 1.1. Example image
(left: gray scale image; right: corresponding binary array)

The image of Figure 1.1 has a spatial resolution of r=2 and a gray scale resolution of b=3 bits per pixel, and thus a side length of 2^r=4 and a size of $2^r \cdot 2^r = 4 \cdot 4 = 16$ pixels and 8 gray levels. The quadtree of this image is obtained by recursively dividing the image into four squares called quadrants or subquadrants, down to the pixel level, and representing this subdivision hierarchy by a tree with outdegree four.

The root of the tree (level 0) corresponds to the whole image, and the leaves (level r) correspond to the individual pixels. Usually, the quadtree is condensed by merging the children of a node whenever they all contain the same value, and replacing them with a leaf node. The quadtree for the image of Figure 1.1 is shown in Figure 1.2. The children of each node are arranged top to bottom and left to right. This kind of quadtree is usually called region quadtree because it represents regions of equal value.

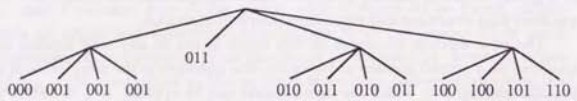

Figure 1.2. Quadtree for the image of Figure 1.1

1.2.2.2 Advantages and Disadvantages

Quadtrees, and their higher-dimensional analogs, octrees (three-dimensional) and 2^d-trees (d-dimensional), are not only used in image processing, but also in a wide variety of other fields such as computer graphics, solid modeling, and geographic and geologic information systems.

Compared to the many other hierarchical subdivision schemes and spatial indexing methods proposed in the literature, the quadtree has the following advantages: Calculations are simple because square side lengths are always a power of two. Also, the position of division lines or planes is not object dependent, so that corresponding parts of two quadtrees match easily. On the other side, quadtrees are still flexible enough to adapt to different resolution requirements in different areas of the universe, and thus to concentrate processing on the most interesting parts.

Also, algorithms working on quadtrees are neither trivial but time consuming, nor do they need complicated heuristics and are overly difficult to analyze. Basic techniques like neighbor finding [Sam84] and quadtree plane sweep [Sam85a], [Sam88c], [Dür90a] can be used for most problems. They lead to algorithms that use either average linear time (neighbor finding) or worst case

linear time (plane sweep). In both cases, the complexity of the problem is measured as the number of nodes of the tree[1]. The calculation of the Euler number and related functionals from a 2^d-tree of arbitrary dimension in linear time [Dür90a] is a particularly interesting example.

Quadtree data structures and algorithms can in most cases easily be extended to three and more dimensions. This is important as the use of higher dimensions can in many cases lead to a better understanding of a problem and to the discovery of new applications. In computer animation, for example, the use of the fourth dimension (time) let to the introduction of the operation of motion comparison [LeeM90].

The quadtree also has some disadvantages. Because of the fixed location and orientation of the subdivisions, translations, rotations, and scalings are not very efficient. This is important in applications such as solid modeling, where these geometric transformations are frequent, but not in image processing, where such transformations are rather rare and difficult even when carried out on the underlying grid of pixels.

Also, the definition of a certain quadtree variant in terms of the types of allowable leaf nodes is very important for the efficiency of a quadtree. If the leaf definition is too restricted, the number of leaves grows too fast, and efficiency is reduced. On the other side, if the leaf definition is too general, the number of leaves is reduced, but the algorithms working on each leaf become too complex.

A good example for this is the definition of the polytree, a combination of octree and boundary representation (B-rep) used in solid modeling. The initial, too restricted definition let to resolution problems, known as black holes, near some vertices and edges [Dür88a,b]. This can be solved by generalizing the leaf node definition, taking care that this does not affect the involved algorithms [Dür89a]. The careful examination of these algorithms lead to some new results in computational geometry [Dür89b] and to an additional generalization of the leaf definitions [Dür90b].

1.2.2.3 Quadtree Representations

The quadtree, as shown in Figure 1.2, is a conceptual construct. There exist many different ways to represent this structure in the main or secondary memory of a

---

[1] Because in the quadtree, all interior nodes have a fixed outdegree of four, the relation between the number of leaves n_l, the number of interior nodes n_i, and the total number of nodes n_t, is given by

$$n_i = (n_l-1)/3 \quad \text{and} \quad n_t = n_i+n_l = (4n_l-1)/3 \tag{1.1}$$

and so we have $O(n_l) = O(n_i) = O(n_t)$.

computer. Each representation has its advantages and disadvantages; for a more detailed discussion, see [Sam89] and [Dür90b (Part IV)].

The pointer quadtree allows fast access and changes through pointers. The linear quadtree [Gar82] stores the size, position, and contents of the leaf nodes in a linear sequence. The depth first expression (in this paper denoted by DF) [Kaw80] is a sequence of symbols resulting from a preorder depth first traversal of the tree. One symbol, usually "(", is used for interior nodes, and others for the different kinds of leaf nodes. The explicit quadtree [Woo82], [Burn83] stores a full quadtree with fixed structure. It is especially suited for cases where condensation is rare, like gray scale images.

The representation of the quadtree influences both memory requirements and accessibility. Memory requirements are considerably lower for DF than for the other representations (cf. [Tam84a]). Accessibility is best for the pointer quadtree and the explicit quadtree, and worst for DF.

A variant of the quadtree is the bintree [Kno80]. Binary subdivision in each of the coordinate directions are alternated, and represented by a binary tree.

1.2.2.4 Quadtrees for Gray Scale Images

After this general introduction to quadtrees, we concentrate on the uses of quadtrees for image processing. Some of the early works on quadtrees were mainly concerned with gray scale images (e.g. [Kli76]). However, for gray scale images, it is rare that even four neighboring pixels have exactly the same gray value. Condensation is much higher for binary quadtrees, and thus the research on quadtrees mainly concentrated on this field.

There were several attempts to solve this problem. Klinger et al. [Kli76] tried to allow condensation based on statistical attributes like the standard deviation. Oliver et al. [Oli83] placed average values in interior nodes. Kawaguchi et al. [Kaw83], after converting pixel values to a Gray code, coded each bit plane separately. Based on bintrees [Kno80], Tamminen [Tam84b] developed a coding suited for so called "maps", images that consist of rather large areas of unique color, but where there is no relation between the colors of adjacent areas. Woodwark [Woo84] proposed a similar scheme based on quadtrees. First proposed in [Kaw80], Oliver et al. [Oli83] and Kunii et al. [Kni86] presented slightly different variants of an extension of the (originally binary) DF to gray scale images.

1.2.2.5 The G-Quadtree

In [Kni86], a family of quadtrees called G-quadtrees was also introduced. Each G-quadtree uses a different number of most significant bits, changing the amount of condensation based on the needs of the application. This work was extended to

3D and refined in [Mao87]. Figure 1.3 shows the three G-quadtrees for our example image.

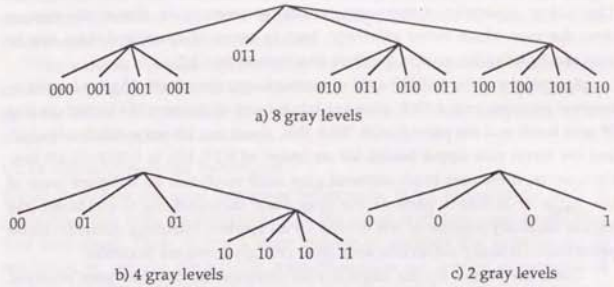

Figure 1.3. G-quadtrees for the image of Figure 1.1

The G-quadtree is the first combination of the spatial hierarchy of the quadtree with the natural gray scale hierarchy (shown for our example image in Figure 1.4). It was some of its efficiency problems that led to the research on quadtrees for gray scale images which resulted in the work presented in this thesis. It is therefore worth to analyze the strengths and problems of this representation.

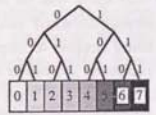

Figure 1.4. The natural gray scale hierarchy

The motivation for the development of the G-quadtree can be seen twofold. On one side, there were the needs of the application (the analysis of autoradiographs of rat brains) for fast automatic and human-guided image analysis with functions like thresholding, averaging, and so on. On the other side, there were the limitations of the hardware, like memory restrictions and the fact that the display was accessible only with a bandwidth of 9600 bits/second. Both sides suggested the use of a coarse representation, refinable by the user. As the original quadtree would have had too many nodes, the resolution was reduced by reducing the number of gray levels.

In general, the G-quadtree achieved its goals, but there are some efficiency problems that are overlooked easily. First, a dual representation is used to store the current quadtree (using pointers) and the remaining lower bits (as an array). This makes conversion of these representations necessary on almost any request from the user. Much better efficiency, both in terms of space and time, can be obtained by using the explicit quadtree (see Subsection 4.1.2).

Second, the variant of DF used to store a G-quadtree (with 2^k gray levels) on external memory, called SKF, uses k+1 bits for any of the only 2^k+1 symbols (the 2^k gray levels and the parenthesis). With this, about one bit per symbol is wasted, and the worst case upper bound for an image of $b \cdot 2^{2r}$ bits is $4/3 \cdot (b+1) \cdot 2^{2r}$ bits. Even worse, if the user sets a different gray scale resolution for different areas of the image or different parts of the gray scale (non-uniform G-quadtree), the highest necessary number of bits is used for all symbols [Mao90a]. Actually, better performance is easily achievable with small changes (compare [Kaw80]).

Third, the reason for the slight overall compression that has been achieved despite the problems mentioned above has been attributed to the G-quadtree; in reality it is mainly due to the fact that almost half of the area of the images used is uniform background. Much higher compression is possible using this fact efficiently. On the other hand, this shows that the G-quadtree is not able to achieve any compression on images without uniform background. A particularly striking example of this problem is the use of the triangular quadtree to construct a three dimensional translucent display of a rat brain [Mao90b]. A check by the author of this thesis revealed that there was no condensation at all in the object areas. This suggests that a straightforward calculation, using arrays, should perform much faster, besides being simpler to implement.

Whereas the above three problems are rather easily detected and solved, it is another point that is of most interest: The high condensation achieved on the more significant bits when using a low number of gray levels is completely lost when increasing the number of gray levels. Trying to find a solution for this problem let to the BC quadtree and GDF as described in Chapter 4.

1.2.3 Progressive Transmission

This subsection gives an overview of the different methods proposed to achieve progressive transmission. An overview of the various applications for progressive transmission is given is Subsection 3.1.3. These applications are basically independent of the method used. Some of the methods mentioned here are compared conceptually with the new method presented in this thesis in Section 4.6; some experimental results can be found in Chapter 6.

The basic principle used by all progressive transmission methods is to rearrange the image information so that the most important components of the image are transmitted first; this is usually combined with a transformation of the image data in one or another way, both to achieve compression and to make data items independent to allow their rearrangement.

The various methods proposed up to now for progressive transmission differ by what parts of the image information they consider most important, and by what kinds of transforms they use; this affects both the complexity of the method and the quality of the result.

1.2.3.1 Spatial Resolution Techniques

Trying to isolate the most important information in an image easily leads to identifying this with the components of the image with low spatial resolution (low frequency). There is thus a large number of methods that increase spatial resolution to achieve progressive transmission. In his review of progressive transmission, Tzou [Tzo87] called these methods pyramid-structured, as some of them use a pyramid data structure.

Progressive transmission was first proposed by Sloan and Tanimoto [Slo79], [Tan79]. Their basic proposal worked as follows: For an image of $2^r \cdot 2^r$ pixels, they recursively constructed images with reduced spatial resolution, having $2^{r-1} \cdot 2^{r-1}$ pixels, $2^{r-2} \cdot 2^{r-2}$ pixels, and so on, with the final image containing only one pixel. The smaller images were formed by taking the average of four pixels of the larger image. All these images can be arranged as a pyramid, with the single pixel image at the top and the full resolution image at the bottom.

Transmission of the image started at the top and proceeded down the pyramid. A rough, but in many cases useful, approximation of the image was obtained at an early stage. The overhead for sending the whole pyramid was 33%, but Sloan and Tanimoto argued that this could be well offset by the gains in efficiency achieved by the early approximations.

They also proposed ways of reducing the overhead. One possibility was to use the sum, and not the average, in the smaller images. Then only three out of four pixels of a lower level had to be transmitted. The fourth was regained by a subtraction operation. This left an overhead of 8.3% for an image with b=8 bits/pixel.

Knowlton [Kno80] eliminated the previously mentioned overhead by using a lookup table transforming two pixel values to their approximate average and difference with exactly the same number of bits/value. Hill et. al. [Hil83] later replaced the large table with a simple procedure. Knowlton [Kno80] also showed that it was possible to achieve overall compression because differences near to zero were much more frequent than large differences. Several prediction schemes

were investigated by Fanelli et al. [Fan84] for the progressive transmission of newspaper images.

The theoretically optimal construction of a low resolution image is not simple averaging of nonoverlapping areas, but the use of Gaussian filters. A pyramid based on this principle has been proposed by Burt and Adelson [Bur83]. They call it the Laplacian pyramid because the difference images between two levels of the pyramid, which are transmitted to increase resolution, are basically Laplacians. This form of pyramid formation has the advantage that the size of subsequent levels is not restricted to increase by a factor of two [Hof86]. Compression is achieved because the Laplacians have a very condensed histogram. A somewhat similar method has been proposed by Yasuda et al. [Yas79,80]. Other filters proposed for progressive transmission include the 2-D quadrature mirror filter (QMF), but its computation cost is extremely high (see [Tzo87] for additional references).

The other extreme, namely simple subsampling, has been proposed by Sloan et al. [Slo79], Dreizen [Dre87], and Endoh and Yamazaki [End87]. Sloan et al. and Dreizen use one of the pixels of a subblock, for example the top left, as a representative. Endoh et al. devised an interesting subsampling scheme that increases the number of pixels by two (instead of four) for each level of the pyramid (see Figure 1.5).

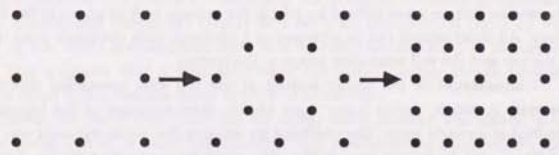

Figure 1.5. The subsampling scheme of Endoh et al.

Sloan et al. stress that in this way, any overheads are eliminated, and no calculations are necessary. Dreizen makes up for the inaccuracy of simple subsampling by first subdividing areas of the image where there are strong changes of gray values. This is nicely combined with a simple prediction scheme that achieves good overall compression.

On the other side, Endoh et al. use recursive linear interpolation and prediction, both based on the hierarchy of sampling points shown in Figure 1.5. This burdens the receiver, especially in the initial phases. Using cubic splines or cubic convolution for interpolation has been proposed by Sanz et al. [San84].

1.2.3.2 Gray Scale Resolution Techniques

Achieving progressive transmission by increasing the gray scale resolution, transmitting one bit plane after the other, is such a general idea that it is difficult to attribute it to any individual researcher. Tzou [Tzo87] calls this, and some related methods, spatial domain techniques, but this is somewhat misleading, as the methods based on the increase of spatial resolution also work in the spatial domain. Basically, this approach is simple, but not very efficient. At least one bit per pixel is necessary for the first approximation. This can however be improved dramatically by using compression methods for binary images (see also Subsection 1.2.3.5).

Several papers that use methods described in Subsection 1.2.3.1 to achieve progressive transmission mention on the fly that improvements are possible by initially restricting the number of bits per pixel [Loh82](for Knowlton's method), [Hil83], [San84]. Although this very easily limits the number of bits per pixel in an initial step, none of these papers propose a way to transmit the additional bits efficiently. The only method that can increase both spatial and gray scale resolution is that of [End87]; the differences between this method and that presented in this thesis will be discussed in Subsection 4.6.3.

1.2.3.3 Transform Coding Techniques

Sloan and Tanimoto [Slo79], [Tan79] mentioned the possibility of using transform coding for progressive transmission. However, they also pointed out the difficulties when using such methods, like the large amount of computation necessary and the need for additional storage.

There are several aspects that make transform coding suited for progressive transmission. First, it provides very good basic compression rates. Second, the transformed coefficients are independent of each other and can be transmitted in any desirable sequence. Third, the transformation usually results in band splitting. Starting transmission with the low frequency coefficients easily allows to increase spatial resolution progressively.

There are basically two ways to transmit the transformed coefficients. The first is to choose a fixed bit allocation scheme based on the desired final quality, and then to transmit one component after the other. Different transmission sequences have been investigated by Ngan [Nga84], and for NTSC coded images by Dubois and Moncet [Dub86].

When transmitting one component after the other, fast transformations like the FFT are not usable. However, Lohscheller [Loh82] showed how to reduce computation by taking advantage of the fact that in the case of the discrete cosine transform, many of the contributions of a coefficient to the retransformed pixels

are equal. Takikawa [Tak84] proposed another way to reduce computation, in his case for the discrete Fourier or the Hadamard transform, namely by recursively transmitting only the top left quarter of the coefficients.

Lohscheller [Loh82,83,84] also showed that additional performance could be gained by measuring the visibility threshold of each component, and eliminating coefficients that are below this threshold. He also introduced an adaptive scheme based on a class division. Performance is improved further, but this scheme can hardly be called progressive any more. The transmission of the classification information overly delays the early transmission of actual image data.

The second way to transmit the coefficients is to quantize all the coefficients incrementally finer and finer, i.e. to allocate more and more bits to each component. Optimal bit allocation schemes can be adapted from nonprogressive applications. With this, not only the spatial, but also the gray scale resolution is successively increased.

Tzou and Elnahas [Tzo86] use embedded coding and show that this approach leads to about the same reproduction quality with a rate half as high as when transmitting one component after the other. Instead of embedded coding, Wang and Goldberg [Wan88] proposed to requantize the quantization errors, and combine this with vector quantization. They showed that by repeatedly quantizing the residual errors of the coefficients, the coding error will approach zero. However, this is only true when numerical errors are ignored and the operations on real numbers are implemented exactly in the same way in the sender and the receiver. This is difficult to guarantee in a heterogeneous environment.

The results with this method are of high quality, but the computational requirements are extremely high. To be really able to use the good approximation quality at low bit rates, the retransformation for a whole image has to be carried out in times of one second or less. If at all, this is possible only with specialized hardware.

1.2.3.4 Miscellaneous Methods and Techniques

Vector quantization, if tree structured, can be used for progressive transmission [Tzo87], [Ris90]. Vector quantization can neither be grouped with the techniques increasing spatial resolution nor with those increasing gray scale resolution. Theoretically, it will improve resolution in the optimal way with every bit transmitted.

Vector quantization can achieve good results at limited complexity [Tzo87]. On the other hand, vector quantization also has its problems. Training of the quantizer for different types of images is necessary, and storing the codebook may take a considerable amount of memory. Also, the range of rates at which vector quantization can be used is limited by the size of the codebook. If a small block size is used, the initial rate is increased. For a large block size, the final rate is restricted.

Various authors, starting with Sloan and Tanimoto [Slo79] and Knowlton [Kno80], have proposed to allow feedback from the receiver. The viewer can then indicate the area of the image that is of most interest, and this can accelerate transmission of the necessary information. Algorithms to determine the area of interest at the sender in advance, based on statistical properties of image parts, have been proposed by Sanz et al. [San84].

A method of image coding and transmission somewhat akin to progressive transmission has been developed by Prusinkiewicz et al. [Pru85]. With this method, not only initial substrings, but in a hologram-like way any substring of the encoded image provides a description of the image at a lower resolution. the price to be paid is the high overhead.

1.2.3.5 Progressive Transmission for Binary Images

The main interest of this thesis is the coding and progressive transmission of gray scale images. Methods for binary (black and white) images can give an important additional insight into this problem.

One of the methods originally proposed by Sloan and Tanimoto [Slo79], [Tan79] is to form a pyramid by taking a popular gray value (instead of the average) as a representative. Pixels on any level not equal to this value are then transmitted by indicating their size, coordinates, and value. This method is suited almost only for binary images.

A related idea is the use of forests of quadtrees to approximate a binary image, as proposed by Samet [Sam85b]. This allows to slightly reduce the number of nodes when compared with a standard quadtree. However, the overhead due to the fact that locations and sizes have to be transmitted is large. Also, in the way the approximations are ordered, the first passes over the image contain most of the information and thus take a long time to transmit, whereas later passes only make some additional corrections. In addition, the approximation is very irregular. Some areas are approximated to the finest detail in the first pass, whereas other areas remain very coarse.

Besides the better known method for the progressive transmission of gray scale images, Knowlton proposed a method for binary images in the same paper [Kno80]. Knowlton uses what is now known as DF (see Subsection 1.2.2.3), reordered breadth first. Also, he uses gray to indicate undetermined areas of the image, and the information about neighboring areas to increase performance.

Growth-geometry coding has also been proposed for the progressive transmission of binary images [Fra80]. However, when compared with the necessary amount of computation, the achieved compression rate, especially for the first stage, is not very impressive, and the example used in [Fra80] is too small to judge the applicability of this method.

1.3 Outline of the Thesis

Chapter 2 and 3, together with the overview of Section 1.2, form the introductory part of this thesis. Chapter 2 concentrates on the theoretical aspects of the problem, giving theoretical arguments and bounds that allow to compare progressive transmission with transmission at a fixed rate. Chapter 3 then analyses the requirements of a wide variety of image transmission and storage applications and provides a framework for later reference.

The main part of this thesis is formed by Chapters 4 and 5. Chapter 4 introduces the basic principles of the new method of data compression presented in this thesis. Chapter 5 provides several ways to improve and optimize the performance of this method. Some of them are closely related to the basic method, whereas others may require a lot of additional work to be implemented successfully.

Chapters 6 and 7 then provide results of experiments (Chapter 6) and proposals for the extension of the method to other applications like color images (Chapter 7).

2

Rate Distortion Theory for Progressive Transmission

In this chapter, data compression and progressive transmission are discussed from a theoretical point of view. The main problem is: Does the fact that a transmission method is progressive mean that it will be less efficient overall than a nonprogressive transmission method, and if yes, by how much?

The first section of this chapter shortly introduces the basic concepts of probability and information theory that are necessary for the later development. The second section then is devoted to rate distortion theory in general. Section 2.3 shows how rate distortion theory can be applied to progressive transmission. Section 2.4 gives an upper bound for the distortion when using progressive transmission at high rates. This upper bound comes fairly near to the upper bounds available for the general (nonprogressive) case.

2.1 Mathematical Foundations

This section gives a short overview of the concepts of probability and information theory necessary to understand the later development. The main purpose of this section is to refresh the existing knowledge and to introduce the notation used. For a thorough introduction, the reader should consult a basic textbook on the subject, such as [Jons79] or [Ros82, chapter 2].

2.1.1 Probability

The *probability* of a certain *outcome* a_i of an *experiment* A, denoted $P(a_i)$, is usually defined as the expected relative occurrence of a_i if the experiment A is repeated a large number of times. The probability of any outcome is thus greater or equal to zero, and as different outcomes are mutually exclusive, and the expe-

riment always has an outcome, the probabilities of all outcomes of an experiment add up to one. An example of an experiment would be the selection of a pixel value in a certain location of an image, or of an image from an image database.

An *event* is defined by saying for each outcome whether or not the event occurs. The probability of an event is the sum of the probabilities of the outcomes for which the event occurs. New events can be constructed by combining existing ones; these are defined by set theoretic operations on the underlying outcomes. A set of mutually exclusive events whose probability adds up to one is called a *system* (of events). The same notation as for experiments and outcomes will be used for systems and events. Also, sometimes events will be treated as *letters* drawn from a certain *alphabet*.

The probabilities of several systems of events can be combined. The probability of the combined occurrence of event a_i in A and b_j in B is denoted by $P(a_i\, b_j)$. Given a complete table of the probabilities for all combinations from A and B, there are several ways of looking at these probabilities.

First, the probabilities $P(a_i\, b_j)$ themselves are called *joint probabilities*. They of course add up to one. Second, the probabilities $P(a_i)$ and $P(b_j)$ are called *marginal probabilities* or marginals. They can be obtained as

$$P(a_i) = \sum_j P(a_i\, b_j) \quad \text{and} \quad P(b_j) = \sum_i P(a_i\, b_j). \tag{2.1}$$

A third concept is *conditional probability*. The probability of a_i conditioned on b_j, denoted $P(a_i \mid b_j)$, is the probability of a_i given that b_j occurred, and can be calculated as

$$P(a_i \mid b_j) = \frac{P(a_i\, b_j)}{P(b_j)}. \tag{2.2}$$

If $P(b_j) = 0$, this means that $P(a_i b_j) = 0$, and in this case $P(a_i \mid b_j)$ is defined to be 0. A useful property when working with conditional probabilities is that general laws valid for unconditioned probability remain valid if all probabilities are conditioned by the same outcome or event. This becomes obvious when we remark that all probabilities are conditioned in some way, even if only by assuming that the experiment takes place in this universe.

If for all a_i and b_j, $P(a_i\, b_j) = P(a_i) \cdot P(b_j)$, then A and B are called *independent*, which means that the outcome of A does not depend on the outcome of B or vice versa. The concepts of joint, marginal, and conditional probabilities, as well as independence, can easily be extended to more than two variables (see also Subsection 2.1.5).

2.1.2 Random Variables, Processes, and Fields

A *random variable* is a variable that assumes a (real) value at random, according to some probability distribution. Random variables can be constructed by assigning a value to each outcome of an experiment, or by taking a function of another random variable. As with traditional variables, random variables can be grouped, leading to *random vectors* and random matrices. Random variables can also be classified by whether they can assume *continuous* values or are restricted to a finite set of *discrete* values.

The *expectation* $E(A)$ of a random variable A is defined (in the discrete case) as the average of its values weighted by the respective probabilities:

$$E(A) = \sum_i P(a_i) \cdot a_i. \tag{2.3}$$

Note that the same notation as for experiments is used. The difference is usually evident from the context. If A can assume more than a discrete set of values, the sum in (2.3) is replaced by an integral.

Taking a series of random variables indexed over time, for example a random variable describing the outcome of an experiment at regular time intervals, results in a *random process*, denoted A_t or $A(t)$. If time is discrete, this is a *discrete time random process* (also called *random sequence*); if time is continuous, this is a *continuous time random process*. As an example, the pixel values of an image when transmitted line by line can be seen as the realization of a discrete time discrete value random process.

Subsequent variables of a random process can be dependent (having memory) or independent (memoryless). Also, they can have identical probability distributions or different probability distributions. When working with independent and identically distributed (i.i.d.) random processes, the index t will usually be omitted.

A random process is called (strictly) *stationary* if the probability distributions of all the random variables that can be defined on any subset of the process are unaffected by shifting the time origin. Weaker versions of stationary processes can also be defined. Another related property of a random process is *ergodicity*. Roughly speaking, it refers to the fact that the process does not have infinite memory, or that dependence between two of its variables is decreasing with increasing time difference.

Replacing the time parameter of a random process by two or more parameters produces a *random field*. Images, before being digitized, can be seen as realizations of two-dimensional continuous parameter continuous valued

random fields. Sampling makes the parameter space discrete, and quantization has the same effect on the values.

2.1.3 Information and Entropy

Random variables (and processes) can be viewed as sources of information. The information obtained when the random variable A assumes value a_i, called *self-information* and denoted $I(a_i)$, depends on the probability of a_i as follows:

$$I(a_i) = -\log P(a_i). \tag{2.4}$$

It can be shown that this definition is the only one that exhibits all the properties conveniently assumed to hold for a measure of information. The base of the logarithm does not have to be specified supposed that it is always the same; it affects the result only by a constant factor. Taking the natural logarithm leads to information being expressed in units of *nats*; using base 2 logarithms gives *bits*.

The *entropy* $H(a_i)$ of a random variable A is its expected or average information:

$$H(A) = E(I(a_i)) = -\sum_i P(a_i) \cdot \log P(a_i). \tag{2.5}$$

Here and in the following, $0 \cdot \log x$ is defined to be zero irrespective of whether $\log x$ or x itself are defined. *Joint* and *conditional entropies* are defined accordingly:

$$H(A\,B) = -\sum_{i,k} P(a_i\,b_k) \cdot \log P(a_i\,b_k) \text{ and}$$
$$H(A \mid B) = -\sum_{i,k} P(a_i \mid b_j) \cdot \log P(a_i \mid b_j). \tag{2.6}$$

Using formula (2.2), it is easy to show that

$$H(A\,B) = H(A) + H(B \mid A) = H(B) + H(A \mid B), \tag{2.7}$$

i.e. the information (per symbol) conveyed by the combination of A and B is the information contained in A plus the information of B when A is already known (resp. vice versa).

2.1.4 Mutual Information

The most important measure of information in the context of coding and data transmission is mutual information. The *mutual information* $I(A;B)$ between two random variables A and B is the information contained in A about B or in B about A, and is calculated as

$$I(A;B) = \sum_{i,j} P(a_i\,b_j) \cdot \log \frac{P(a_i b_j)}{P(a_i) \cdot P(b_j)}$$
$$= H(A) - H(A \mid B) = H(B) - H(B \mid A). \tag{2.8}$$

An important property of mutual information is that it is never negative, as expressed in the following theorem (For a proof, see for example [Jons79, p.22]):

Theorem 2.1: *Mutual information is greater or equal to zero, with equality only in the case of independence.*

If the source random variable A is transmitted over a communication system or channel resulting in the random variable B at the output, then $I(A;B)$ is the information about A transmitted by the channel. Channel coding theory, which will not be discussed further here, says that $I(A;B)$ is also the capacity of the channel that is necessary to produce an output B statistically related to the source A with joint probabilities $P(a_i\,b_j)$.

On the one hand, if A and B are independent, and thus $I(A;B) = 0$, no transmission is necessary; B can just be produced by a random number generator. On the other hand, if we want B to reproduce A exactly, then the channel capacity necessary is $I(A;A)$, which can be shown to equal $H(A)$, as $P(a_i\,a_j) = P(a_i)$ if $i=j$, and $P(a_i\,a_j) = 0$ otherwise.

2.1.5 Mutual Information for Three Variables

Several kinds of mutual information can be defined for three (and more) variables. These kinds of information measures will play an important role in Section 2.3.

The mutual information between a random variable A and the combination of two random variables B and C, written $I(A;\,B\,C)$, is very naturally defined as

$$I(A;\,BC) = \sum_{i,j,k} P(a_i\,b_j\,c_k) \cdot \log \frac{P(a_i\,b_j\,c_k)}{P(a_i) \cdot P(b_j\,c_k)} \tag{2.9}$$

The mutual information between A and B when conditioned on C, denoted by $I(A;B \mid C)$, similarly is

$$I(A;B \mid C) = \sum_{i,j,k} P(a_i\,b_j\,c_k) \cdot \log \frac{P(a_i b_j \mid c_k)}{P(a_i \mid c_k) \cdot P(b_j \mid c_k)} \tag{2.10}$$

For conditional mutual information, a theorem similar to Theorem 2.1 holds:

Theorem 2.2: $I(A;B \mid C) \geq 0$, *with equality only if A and B are independent when conditioned on C.*

Proof: The conditional mutual information can be written as the weighted average of the mutual information between A and B for each c_k:

$$I(A;B \mid C) = \sum_k P(c_k) \cdot \sum_{i,j} P(a_i b_j \mid c_k) \cdot \log \frac{P(a_i b_j \mid c_k)}{P(a_i \mid c_k) \cdot P(b_j \mid c_k)}$$
$$= \sum_k P(c_k) \cdot I(A;B \mid c_l). \tag{2.11}$$

The conditional probabilities $P(a_i \mid c_k)$, $P(b_j \mid c_k)$, and $P(a_i b_j \mid c_k)$ together behave like ordinary probabilities, and so with Theorem 2.1, $I(A;B \mid c_k) \geq 0$. Also, $P(c_k) \geq 0$ for all k, from which it follows directly that $I(A;B \mid C) \geq 0$. Q.E.D.

Another theorem relates mutual information and conditional mutual information:

Theorem 2.3: $I(A; BC) = I(A; B) + I(A;C \mid B)$.

For a proof, see for example [Jons79, p. 28]. The theorem says that the information that both B and C together convey about A is equal to the information that B gives on A plus the information that C gives on A after B is known. The next theorem, which concludes this section, will be used in Section 2.3:

Theorem 2.4: $I(A; B) + I(A;C \mid B) \geq I(A; C)$, *with equality if and only if A and B are independent when conditioned on C.*

Proof: Using theorem 2.3, we have

$$I(A; BC) = I(A; B) + I(A;C \mid B) = I(A; C) + I(A;B \mid C), \tag{2.12}$$

and so, as from theorem 2.2, $I(A;B \mid C) \geq 0$,

$$I(A; B) + I(A;C \mid B) \geq I(A; C),$$

with equality if and only if $I(A;B \mid C) = 0$. Q.E.D.

2.2 Rate Distortion Theory

Consider the following problem: Given an image, how well (i.e. how near to the original image) can it be transmitted with a given number of bytes if the best possible coding is used? The investigation of this relation between transmission rate and reproduction quality (or distortion) is the subject of rate distortion theory. In this section, we give a short introduction to rate distortion theory as far as necessary for the development in the following sections. The reader interested in a more detailed treatment is referred to [Ber71] or [Gra90].

2.2.1 The Model of Transmission

The transmission system considered by rate distortion theory is shown in Figure 2.1. From a source random process X_t, the transmission system produces an output Y_t. In the following, we will assume that the source is discrete and i.i.d. Extensions to more complex cases are possible (see [Ber71, Chapter 7]). The transmission system is composed of a source coder, which for each x selects some value $\hat{x}$, a channel that takes $\hat{x}$ as input and outputs $\hat{y}$, and is able to transmit at rate R, and a source decoder which transforms $\hat{y}$ to y.

The channel may be composed of a channel coder, a noisy transmission line, and a channel decoder, but is assumed to work errorfree, so that we can assume $\hat{x} = \hat{y}$. Also, the decoder works deterministically, and so we can assume that it just copies its input to its output, so that $y = \hat{y}$. Then the responsibility of optimizing transmission is fully assumed by the source coder. This is why this field of coding theory is called source coding, in contrast with channel coding, which is mainly concerned with error detection and error correction codes.

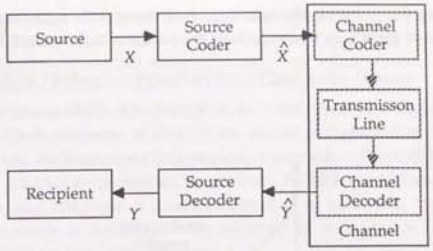

Figure 2.1. Transmission system model

2.2.2 The Rate Distortion Function

The rate distortion function $R(D)$ for a given distortion D is defined as the minimum rate necessary to transmit y so that the distortion is smaller or equal to D. Here, distortion is measured by assigning to each pair of x and y a value $d(x, y)$ that indicates the cost or distortion of the output y in view of the source value x. Greater values of $d(x, y)$ indicate less desired combinations of x and y. The distortion of the overall transmission is measured as the average distortion per letter, or

$$dist(Q) = \sum_{jk} Q_{jk}\, d(x_j, y_k) \qquad \text{where}$$

$$Q_{jk} = P(x_j\, y_k). \tag{2.13}$$

Q is called the transition probability matrix. The distortion measure of (2.13) is the so-called single letter fidelity criterion, and we will assume this fidelity criterion for the following discussion.

The rate distortion function $R(D)$ can now be formally defined as follows: If we define the set of acceptable transition probabilities as

$$Q_D = \{Q \mid dist\,(Q) \leq D\}, \qquad \text{then}$$

$$R(D) = \min I(X; Y), \tag{2.14}$$

where the minimum is over all $Q \in Q_D$. Many general properties of the rate-distortion function are known, such as convexity ($\cup$), continuity (except maybe at the point of maximal distortion D_{max}), and the fact that the slope of $R(D)$ tends to $-\infty$ as D approaches zero [Ber71, chapter 2].

Figure 2.2 shows a typical rate distortion curve. Unfortunately, in most cases it is impossible to find $R(D)$ analytically. However, an efficient numerical algorithm to compute points on $R(D)$ is available [Bla72]. It works parametrically,

approximating the point on the rate distortion curve with a given slope. This algorithm was of great help when checking the results of Subsection 2.3.

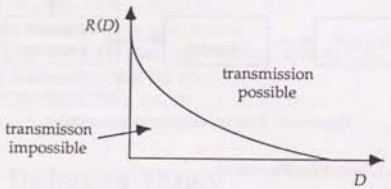

Figure 2.2. A typical rate distortion curve

The relation between transmission system input and output is given in terms of joint probabilities, but for the actual implementation, block coding is usually assumed. A *block code* of length n works by repeatedly taking n letters of the source, and assigning to this block a code word according to some rule. The number of code words is chosen so that it matches the capacity of the channel.

The most important property of $R(D)$ as defined by (2.14) is expressed in the source coding theorem and its converse, originally due to Shannon [Sha59]:

Theorem 2.5 (Source Coding Theorem): *Given a discrete memoryless source X and a single letter fidelity criterion $d(x, y)$, denote the corresponding rate distortion function with $R(D)$. Then, for any $\varepsilon>0$ and any $D\geq 0$, an integer n can be found so that there exists a code of block length n, distortion $\leq D+\varepsilon$ and rate $<R(D)+\varepsilon$.*

Theorem 2.6 (Converse Source Coding Theorem): *There are no source codes that achieve rate less than $R(D)$ with distortion less than D.*

For a proof of the above two theorems, we refer the reader to [Ber71] or [Gra90]. Theorem 2.5 is valid for block codes only, but similar coding theorems are possible for other kinds of codes.

The two theorems together exhibit the full meaning of the rate distortion curve: Performance above and arbitrarily near the rate distortion curve is possible, at the expense of maybe very long and complicated codes, but performance below the rate distortion curve is impossible. The rate distortion curve therefore can provide a good yardstick when designing codes. If we are fairly close to $R(D)$, it might not be worthwhile to search for additional

improvements. Also, we will never attempt to design a code that performs below $R(D)$.

2.2.3 The Rate Distortion Function for a Composite Source

A composite source [Ber71, Chapter 6.1] is the combination of a certain number of subsources. Each subsource is an ordinary source as discussed in the previous subsection, and the sources are independent of each other. For each time instant, one of the subsources is selected at random, based on a known probability distribution, and the letter it produces has then to be coded. Depending on whether the coder or the decoder are informed of the sequence of subsource selections, there are four different variants.

For the development in the next section, the only case of interest is that both the coder and the decoder know which subsource has been selected. In this case, the rate distortion function of the composite source is constructed by combining the rate distortion functions of the individual sources. For each rate distortion function, let the region of achievable rate distortion combinations, including the rate distortion curve itself, be called the *achievable region*. Then the following theorem holds:

Theorem 2.7: *For a composite source with both coder and decoder informed about the selection of the subsource, the achievable region of the composite source is the Minkowski sum of the achievable regions of the subsources weighted with the respective probabilities of the subsources.*

Proof (informal): Minkowski addition of two sets of points is defined as the set of points formed by the vector addition of any two points of the original sets [Min03], [Had57]. The definition of scalar multiplication and of the sum of more than two terms is obvious. A good introduction to the properties of Minkowski addition can be found in [Gia88].

If the subsources are so that they can be, individually, transmitted at rates $R_1,\ldots, R_k,\ldots, R_n$, with distortions $D_1,\ldots, D_k,\ldots, D_n$, and the probability of each subsource being used is denoted by $P(s_k)$, then for a single letter distortion measure, the average distortion is

$$D = \sum_k P(s_k)\cdot D_k, \tag{2.15}$$

and the average rate will be

$$R = \sum_k P(s_k)\cdot R_k. \tag{2.16}$$

This corresponds directly to the definition of the Minkowski sum. That the boundary of the resulting achievable region of points (D, R) is indeed the rate distortion function of the composite source, in the sense of the source coding theorems 2.5 and 2.6, is proved in Berger [Ber71, pp.183/4 and pp. 55-7]. The

specification of the rate distortion curve used there, namely to combine points of the original rate distortion functions with equal slope, exactly corresponds to the slope-matching rule for the outlines of Minkowski sums as explained in [Gui83]. The properties of the composite rate distortion function, as given in [Ber71, p. 57], are also easily recognizable as properties of Minkowski addition. Q.E.D.

2.3 Lower Bounds

2.3.1 Successive Refinement

Rate distortion theory has been developed for transmission at a fixed rate; the rate distortion curve shows the relation between rate and distortion when using a transmission scheme that does not necessarily allow to obtain intermediate results as transmission is proceeding. In other words, we are getting from the point of no transmission $(D_{max}, 0)$ *in one step* to a point $(D_1, R(D_1))$.

Rate distortion theory does not explicitly say whether we can, in a second step, achieve the point $(D_2, R(D_2))$, i.e. reduce the distortion to D_2 with the additional rate of not more than $R(D_2) - R(D_1)$ (Figure 2.3 (left)). Whether this is possible or not is of great importance for progressive transmission. If the point $(D_2, R(D_2))$ can be achieved from the point $(D_1, R(D_1))$, this shows that theoretically, progressive transmission is possible at the same rate and with the same distortion as transmission that does not care about intermediate results.

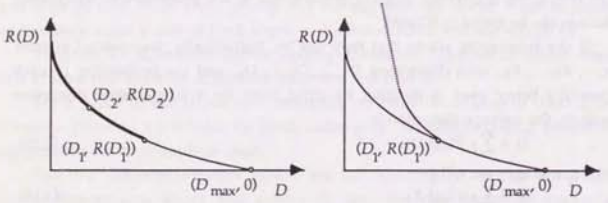

Figure 2.3. Rate distortion curves for progressive transmission

On the other hand, there are cases where the way the information was transmitted to reach the point $(D_1, R(D_1))$ precludes that further points on the original rate distortion curve can be reached (Figure 2.3 (right)). This means that

in this case we have to pay something if we want to make transmission progressive. Also, it makes theoretical analysis much more complicated, as the rate distortion combinations achievable in this case depend on the size of the steps.

The question of which of the two cases in Figure 2.3 applies has been investigated in a recent thesis by Equitz [Equ89], [Equ90]. He used the term "successive refinement" for the case where there is no loss with progressive transmission[1]. He found necessary and sufficient conditions for successive refinement to be possible, defined in terms of Markov conditions on the transition probabilities at the points $(D_1, R(D_1))$ and $(D_2, R(D_2))$.

Equitz's derivation of these conditions relies on previous results on the so called multiple description problem. The author of this thesis obtained equivalent conditions for discrete sources independently before getting notice of Equitz's work. The presentation of the conditions will therefore take a form somewhat different to that in [Equ89]. Although it is less formal, it is hoped that it will be somewhat easier to understand for readers not very familiar with rate distortion theory.

2.3.2 Conditions for Successive Refinement

Assume a source random variable A, with an alphabet of size m, and two decoder outputs B and C, both with the same alphabet of size n. The distortion matrix is given as $d(a_i, b_j) = d(a_i, c_j)$ $(0 \le i < m,\ 0 \le j < n)$. The transition probabilities $P(a_i|b_j)$ and $P(a_i|c_k)$ are chosen so as to yield points $(D_1, R(D_1))$ and $(D_2, R(D_2))$, respectively, on the rate distortion curve, with $R(D_2) > R(D_1)$. The rates and distortions will be indexed with the letters used for the decoder outputs (i.e. $D_1 = D_B$, $R(D_1) = R_B$, $D_2 = D_C$, and $R(D_2) = R_C$).

Based on the transmission of source A with output B, we will now try to construct a new output equal to C with rate $R_C - R_B$. Although the relation between A and B is defined in terms of conditional probabilities, the channel and the decoder work deterministically, so that both the coder and the decoder know B. Thus we can construct a composite source as discussed in Subsection 2.2.3 by using B to select the appropriate subsource. The probability of each subsource is $P(b_j)$, with source output probabilities $P(a_i|b_j)$. The distortion matrix remains unchanged. As the subsources are independent of each other, the resulting rate distortion function will be the actual rate distortion function when starting from point (D_B, R_B).

---

[1] It would probably be more appropriate to speak of "lossless successive refinement", but we will use "successive refinement" in the following.

We denote the overall output resulting from this composite source by C'. To decide whether successive refinement is possible, we have to compare C' with C, which can be done by setting the slope of the rate distortion function at $(D_{C'}, R_{C'})$ (and thus the slopes of all the subsource rate distortion functions) equal to the slope at (D_C, R_C).

If for this slope, there is only one point (D_C, R_C), with only one set of transition probabilities, then either

$$D_{C'}=D_C, \tag{2.17a}$$

$$R_{C'}=R_C, \quad \text{and} \tag{2.17b}$$

$$P(a_i c'_k)=P(a_i c_k) \quad \text{for all } i \text{ and } k. \tag{2.17c}$$

In this case successive refinement is possible. Otherwise $(D_{C'}, R_{C'})$ will not lie on the rate distortion curve, which means that successive refinement is impossible. If there is more than one set of transition probabilities, then if successive refinement is possible, one of these sets has to fulfill condition (2.17c). If the rate distortion curve contains a linear segment, then successive refinement can be tested by approaching both endpoints of this segment.

For the above three conditions, it can be shown that (2.17c) implies (2.17a), as

$$\begin{aligned} D_{C'} &= D_{C'|B} = \textstyle\sum_j P(b_j)\cdot D_{C'|b_j} \\ &= \textstyle\sum_j P(b_j)\cdot \sum_{ik} P(a_i c'_k \mid b_j)\cdot d(a_i, c'_k) \\ &= \textstyle\sum_{ik} P(a_i c'_k)\cdot d(a_i, c'_k) = \sum_{ik} P(a_i c_k)\cdot d(a_i, c_k) \\ &= D_C. \end{aligned}$$

On the other hand, using (2.17b) and (2.17a), we have

$$\begin{aligned} R_C &= I(A; C), && \text{and in addition} \\ R_{C'} &= R_B + R_{C'|B} \\ &= I(A; B) + \textstyle\sum_j P(b_j)\cdot I(A;C' \mid b_j) \\ &= I(A; B) + \textstyle\sum_j P(b_j)\cdot I(A;C \mid b_j) \\ &= I(A; B) + I(A;C \mid B), && \text{so that} \\ I(A; C) &= I(A; B) + I(A;C \mid B). && (2.18) \end{aligned}$$

Using theorem 2.4, the condition for successive refinement to be possible can now be deduced directly and formulated as follows:

Theorem 2.8: *Successive refinement of source A from decoder output B to decoder output C is possible iff A and B are independent when conditioned on C, or*

$$I(A;B \mid C) = 0. \tag{2.19}$$

This is a very natural result: When transmitting data progressively, any initial parts of the data are included in the data transmitted up to the current point.

Therefore they will not provide any additional information about the source if again sent separately.

Formula (2.19) can be expressed in terms of the individual probabilities using (2.10) as follows:

$$P(a_i b_j \mid c_k) = P(a_i \mid c_k)\cdot P(b_j \mid c_k), \quad \text{respectively} \tag{2.20}$$

$$P(a_i b_j c_k) = P(a_i c_k)\cdot P(b_j \mid c_k) \quad \text{or} \tag{2.21}$$

$$P(a_i b_j c_k)\cdot P(c_k) = P(a_i c_k)\cdot P(b_j c_k) \tag{2.22}$$

for all i, j, and k. Now given the points (D_B, R_B) and (D_C, R_C), the marginal probabilities $P(a_i b_j)$ and $P(a_i c_k)$ are known, but else, there are no conditions for the joint probabilities $P(a_i b_j c_k)$. Summing (2.21) over k leads to

$$P(a_i b_j) = \textstyle\sum_k P(a_i c_k)\cdot P(b_j \mid c_k). \tag{2.22}$$

This is a system of mn linear equations where the $P(a_i b_j)$ and $P(a_i c_k)$ are given, and the $P(b_j \mid c_k)$ are the unknowns. This system can be separated into n systems, one for each b_j, with m equations and n unknowns each. If the solutions to these systems are all nonnegative, i.e. if they indeed represent probabilities, then successive refinement is possible; else it is not. This condition also can be expressed geometrically: If for all b_j, the vector of probabilities $P(a\, b_j)$ lies inside the angle spanned up by the vectors of probabilities $P(a\, c_k)$, then and only then progressive transmission is possible.

Depending on m and n, the following cases can occur: If both the source alphabet and the reproduction alphabet have equal size ($m=n$), there is in general a unique solution for each system, which can be checked for negative values. If the source alphabet is smaller than the reproduction alphabet ($m<n$), in general the solution will be underdetermined; all possible solutions lie on a lineal manifold of dimension $n-m$. Whether a solution without negative values exists can be checked by using methods similar to linear programming. In the third case, when the source alphabet is larger than the reproduction alphabet ($m>n$), in general there is no solution. If $m>n$, it is therefore easy to find counterexamples to successive refinement.

If successive refinement is possible, formula (2.22) not only allows to check whether successive refinement is possible, but if so also allows to calculate the transition probabilities of all the subsources. As these subsource have source distributions different from the original source A, this may allow to calculate the rate distortion functions of large classes of source distributions comparatively easily by just solving a system of linear equations.

2.3.3 Discussion

The development in the previous subsection shows that there are indeed cases where theoretically, using progressive transmission is as efficient as using nonprogressive transmission. From the geometric explanations in the previous subsection, we can deduce the class where both source and reproduction alphabet contain two letters (m=n=0) as a simple example. On the other hand, (2.22) lets assume that in most cases where $m > n$, successive refinement is not possible.

Equitz [Equ89], [Equ90] investigated several other types of source distributions and distortion functions. He showed that for any discrete source with Hamming distortion measure (also called probability of error distortion measure), any Gaussian source with squared difference distortion measure, and any Laplacian source with absolute difference distortion measure, successive refinement is possible.

In addition, Equitz proved that successive refinement is always possible for discrete sources and small distortion. In general, successive refinement seems to be more frequent for small distortions, which can easily be explained by the geometric argument at the end of the previous subsection. For small distortions, the probability vectors come closer and closer to the coordinate axes and span up the whole first (hyper)quadrant. This is in accordance with the general tendency that rate distortion problems are easier to treat at small distortions.

Besides the practical reasons that advocate the use of progressive transmission (see Chapter 3), which will be discussed in Chapter 3, the above results represent an additional incentive that makes the study of progressive transmission worthwhile. However, in principle, these results are valid only under the assumptions of rate distortion theory.

The rate distortion function is the boundary of the region of possible rate distortion combinations, achievable only as the number of symbols transmitted grows towards infinity. In the case of successive refinement, the total block length necessary to reduce the distance to the rate distortion curve below a certain level will certainly grow with the number of refinements used. The results of the experiments of Equitz [Equ89, Chapter 7], for example, suggest that in the case of a Gaussian source, the necessary total block length will increase proportionally to the number of refinements.

In the case of finite length messages like single images, which are our main interest, the increase of total block length proportional with the number of refinements means that the number of refinements is limited by image size if we want to transmit with a certain efficiency. Cases where several images are transmitted together are very rare (see Subsection 3.1.3.1)

However, the reasoning developed in the previous subsection is strictly valid only in the case where we are actually on the rate distortion curve. The fact that we used a higher rate than actually necessary to achieve a given distortion does not mean that this was completely useless. An extreme example is receiving a cryptographically encoded message: Although the rate increases continuously, distortion does not decrease until the key is received.

The above reasoning made obvious that in addition to investigating lower bounds to the combinations of rate and distortion achievable with progressive transmission, as done in this section, realistic upper bounds could help greatly in assessing the possibilities of progressive transmission.

2.4 An Upper Bound

2.4.1 High Rate Vector Quantization Advantages

In this section, we will obtain an upper bound for the additional amount of distortion we have to "pay" for a progressive coding method. In order to obtain a numerical result, some simplifying assumptions are necessary. First, we choose the mean square error per symbol as the distortion measure. This is more or less the only mathematically tractable error criterion. Second, we use the so-called high rate assumption, i.e. we assume that the transmission rate is high enough that the probability over the region of the source represented by a given code vector is nearly constant, and that boundary effects can be neglected.

The high rate or high resolution assumption is used frequently when investigating the approximative behavior of coding algorithms. For this case, the individual components that contribute to the advantage of vector quantizers over scalar quantizers, including their magnitude, have been investigated recently by Lookabaugh et al. [Loo89]. These components are called space filling advantage, shape advantage, and memory advantage. The memory advantage results from the fact that subsequent signals are not independent. The shape advantage depends on the distribution of the individual signals. The space filling advantage is due to the fact that in higher dimensions, the Voronoi regions around each coding vector can be made to resemble the optimal sphere.

For the high rate case, Zador [Zad82] showed that the expected minimal distortion for a k-dimensional block or vector code with a code book of M vectors is given by

$$E(k, M, p) = G_k M^{-2/k} \left(\int_{R^k} p(x)^{k/(k+2)} dx \right)^{(k+2)/k} \qquad (2.23)$$

where $p(x)$ is the probability density function of the source vector x and G_k is the coefficient of quantization, which depends only on k. Zador also provided upper and lower bounds for G_k.

Gersho [Ger79] then conjectured that for an optimal quantizer, the Voronoi regions around each code book vector are congruent to the space-filling convex k-dimensional polytope P with minimal corresponding quantization coefficient $G(P)$. In this case, $G(P)$ can be evaluated as the dimensionless and normalized second moment of P:

$$G(P) = \frac{1}{k} \frac{\int_P \|x-\hat{x}\|^2 dx}{V(P)^{(k+2)/k}} \tag{2.24}$$

Here $V(P)$ is the volume of P, $\hat{x}$ is the centroid of P, which is used on the decoder side to represent source vectors that fall into P, $\|x\|$ denotes the Euclidian norm (the length) of x, and the numerator of (2.24) is the (unnormalized) second moment, also denoted by $U(P)$. Note that even if the conjecture of Gersho might not hold, (2.24) provides an upper bound for G.

The optimal polytope for k=1 is the interval with $G_1 = 1/12 = 0.08333\ldots$, for k=2 it is the hexagon with $G_2 = 0.08018\ldots$, for k=3 the truncated octahedron with $G_3 = 0.07854\ldots$, and so on. Further results are given in [Con82]. For $k\to\infty$, both the upper and the lower bounds of Zador converge to the case where P is a k-dimensional sphere and $G \to \frac{1}{2\pi e} = 0.05855$. These figures directly express the above mentioned space filling advantage for each dimension.

2.4.2 The Space Filling Disadvantage For Progressive Transmission

In the case of progressive transmission, the shape advantage and the memory advantage remain unaffected. On the other side, the space filling advantage cannot be used because the optimal polytopes (with the exception of k=1) are not recursively dividable. This problem could be avoided by again using the high-rate assumption to eliminate boundary effects. However, this would mean that each polytope had to be divided into far more than 2^k smaller polytopes, which would correspond to the transmission of more than 1 bit/symbol and make the progressive transmission steps much too coarse.

The only recursive subdivision that seems to be available for higher dimensions is the regular binary subdivision of the bintree or the 2^d-tree [Kno80], [Tam84a], [Jac83], [Dür90a]. For the 2^d-tree, the polytope is a hypercube, which corresponds to coding each dimension separately, and G is therefore equal to G_1=1/12. However, even in this case, the number of bins for one subdivision step is 2^k, with a corresponding incremental rate of 1 bit/symbol.

The boxes of the bintree at each level are not congruent if the bintree is used in its classical form with all edge lengths equal to powers of two (see below). However, according to the original intentions of Knowlton [Kno80], who used rectangles with aspect ratios of 2:3 and 4:3, the boxes can easily be made congruent. The side lengths for the box P, denoted by $a_0, a_1, \ldots, a_{k-1}$, are given as:

$$a_i = 2aq^i \qquad \text{where} \qquad q=2^{1/k}. \tag{2.25}$$

In two dimensions, the standard paper sizes like A3, A4, A5,... are constructed in this way. G can be calculated for this box as follows: We start by placing the center at the origin, so that $\hat{x}$ is eliminated and the numerator in (2.24) takes the form

$$U(P) = \int_P \|x\|^2 dx. \tag{2.26}$$

Then we express $\|x\|^2$ as a sum, getting

$$U(P) = \int_P \left[\sum_{i=0}^{k-1} x_i^2\right] dx. \tag{2.27}$$

The integral and the sum can be exchanged

$$U(P) = \sum_{i=0}^{k-1} \left[\int_P x_i^2 dx\right] \tag{2.28}$$

and the multiple integral replaced by a simple one and evaluated

$$U(P) = \sum_{i=0}^{k-1} \left[\frac{V(P)}{2a_i} \int_{-a_i}^{a_i} x_i^2 dx_i\right] = V(P) \sum_{i=0}^{k-1} \left[\frac{a_i^2}{3}\right]. \tag{2.29}$$

Using (2.25), we then get

$$U(P) = \frac{V(P)}{3} \sum_{i=0}^{k-1} a^2 q^{2i} = \frac{a^2\, V(P)}{3} \frac{q^{2k}-1}{q^2-1} = \frac{a^2 V(P)}{2^{2/k}-1}. \tag{2.30}$$

This leaves us with

$$G(P) = \frac{1}{k} \frac{V(P)}{V(P)^{(k+2)/k}} \frac{a^2}{2^{2/k}-1} = \frac{1}{k} \frac{1}{V(P)^{2/k}} \frac{a^2}{2^{2/k}-1}. \tag{2.31}$$

The volume $V(P)$ can be calculated as

$$\begin{aligned} V(P) &= \prod_{i=0}^{k-1} 2a_i = \prod_{i=0}^{k-1} 2aq^i \\ &= 2^k a^k q^{k(k+1)/2} = 2^k 2^{(k+1)/2} a^k. \end{aligned} \tag{2.32}$$

Combining this with (2.31), we obtain

$$G(P) = \frac{2^{1/k}}{8k\,(4^{1/k}-1)}. \tag{2.33}$$

That a is eliminated shows that indeed $G(P)$ is unaffected by scaling P. For k=1, (2.33) evaluates to 1/12 as expected. For $k\to\infty$, the numerator approaches 1. The

limit of the denominator can be found by setting $h = 1/k$ for convenience and using L'Hopital's rule:

$$\lim \frac{f(x)}{g(x)} = \lim \frac{f'(x)}{g'(x)} \tag{2.34}$$

so that

$$\lim_{(h\to 0)} \frac{h}{(4^h-1)} = \lim_{(h\to 0)} \frac{1}{\ln(4)\, e^h}. \tag{2.35}$$

This finally leads to the following theorem:

Theorem 2.9: *Under the high rate assumption, the space filling disadvantage of progressive transmission for $k\to\infty$ is*

$$G(P) = \frac{1}{8\ln(4)} = 0.090168.... \tag{2.36}$$

Therefore the space-filling disadvantage of progressive transmission compared with scalar quantization is $2/3 \cdot \ln(4) = 0.9242$ or -0.3424 db. Compared with the best known lattice quantizer, based on the lattice Λ_{16} [CheT90], it is 0.7574 or -1.2071 db, and compared with the optimal quantizer for $k\to\infty$, it is 0.6493 or -1.8753 db. As the gain per additional bit per symbol is a factor of 4.0 or 6.02 db [Jay84, p. 125], the progressive transmission disadvantage can also be expressed in terms of bits per symbol. This leads to an additional 0.057 bits, 0.200 bits, and 0.312 bits per symbol necessary to achieve the same distortion when compared with the above three cases.

It may be of interest to investigate the case where all edge lengths of the bintree are powers of two as mentioned above. In this case, if we denote by g the number of smaller edges, the side lengths of the box P are given as:

$$a_i = a \ (0\le i<g) \quad \text{and} \quad a_i = 2a \ (g\le i<k). \tag{2.37}$$

Evaluation of (2.24) in the same way as above leads to

$$G(P) = \frac{1}{12n} \cdot \frac{4n-3g}{4^{(n-g)/n}}. \tag{2.33}$$

For $g=0$ and $g=n$, this evaluates to $1/12$, as in these cases, all edges are of the same length. The maximum is attained for $g = n \cdot [4/3 - 1/\ln(4)] = 0.612n$ with $G(P) = 0.1053$.

This and the above results are summarized in Figure 2.4. Note that compared with the previous rate distortion diagrams, the axes have been exchanged, and the distortion logarithmized and turned upside down to be compatible with the graphs used in Chapter 6. The values on the coordinate axes are relative values, they should not be taken absolutely.

In Figure 2.4, only the lowest two curves are progressive; the three upper curves are just collections of points each reachable independently, but not successively during the same transmission. In fact, once a point on one of these curves has been reached, additional finer quantization (corresponding to

progressive transmission) leads to values much below the curve of optimal progressive quantization.

The curve for the case where all edge lengths of the bintree are powers of two (uniform progressive) is a good example of this. Once in a while, the scalar quantization curve is reached, but between these points and on average, performance is worse than for optimal progressive transmission.

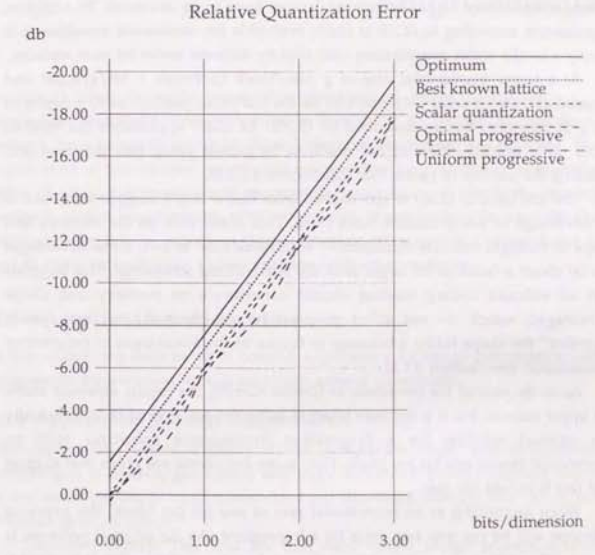

Figure 2.4. Relative quantization error for different polytopes.

2.4.3 Discussion

When compared to the memory and shape advantages that can be as high as 10 db [Loo89], the above space filling disadvantage for progressive transmission is clearly not very high. What is much more important is that such a constant bound exists at all. It suggests that something like a progressive rate distortion

function might indeed exist, and shows that in the high-rate region it is placed only slightly above and in constant distance from the "one shot" rate distortion function. This is of course under the condition that we reach the high-rate region without additional losses.

Also, it is important that the above result is obtained with an incremental transmission rate of one bit per vector (not per symbol), the smallest incremental rate practically realizable. This is in contrast to Section 2.3 and the work of Equitz, where considerably higher incremental rates have to be assumed. In addition, quantization according to (2.25) is easily realizable for continuous variables; it is simply a locally scalar quantization with slightly different scales for each variable.

At a larger incremental rate of g bits/block (between 1 bit/symbol and 1 bit/block), the optimal polytope will be the Cartesian product of k/g copies of the g-dimensional box as described by (2.25). As (2.33) approaches the limit of (2.36) quite quickly, there is not much to be gained using this polytope and reducing the number of passes (see also Section 3.2.2.3).

That the limit of (2.36) is approached from below might suggest that there is an advantage of using smaller block sizes. This is not true, as the memory and shape advantages increase considerably with block size. In fact, these advantages can be about a factor of 10 larger than the space filling advantage. This suggests that an efficient coding method should concentrate on memory and shape advantages, which do not affect progressive transmission, but may easily "sacrifice" the shape filling advantage in favour of the advantages of progressive transmission (see Section 3.1.3).

As in the case of the conjecture of Gersho [Ger79], the results obtained above are upper bounds, but it is not sure whether using the polytope of (2.25) is actually the optimal solution for a progressive transmission quantizer with an incremental rate of one bit per block. Here some arguments are given that suggest that this is indeed the case.

When quantizing at an incremental rate of one bit per block, the previous polytope will be cut into two parts by a hyperplane. As the original polytope is convex, the dihedral angle between the cutting hyperplane and the original polytope boundary will be smaller or equal to 90°. Angles greater than 90° will lead to high second moments and thus inefficient quantization. Therefore a subdivision scheme that keeps all angles at 90° can be conjectured optimal.

The results in this Section can also be applied to tree structured vector quantization [Ris90]. Tree structured vector quantization is used mainly as a tool to reduce the complexity of encoding, which is a limiting factor to the performance of vector quantization.

3

Requirements

In this chapter, the wide range of applications of image compression and progressive transmission is overviewed in Section 3.1. In Section 3.2, the requirements for a generally usable or "ideal" compression method are analyzed. The mains aims of this chapter are to give an overview over the many application fields of image compression and progressive transmission, to clarify the relation between image compression and progressive transmission, to provide the motivation for several aspects of the new method presented in Chapter 4 and 5, and to allow its evaluation and comparison with other methods.

3.1 Application Overview

In this section, the wide range of possible applications for image compression and progressive transmission is reviewed from several viewpoints.

3.1.1 Applications of Image Compression

The applications where digitally stored images are used, and thus image compression is desired, grow more and more numerous. In the following, some of the more important application areas are discussed shortly; this list is not intended to be complete:

- Satellite imagery: This was the first field where image processing and compression by computers were used on a large scale. Satellite images are used for astronomy, telemetry, weather forecast, intelligence, land use planning, agriculture, and so on. Special properties of these images include the use of three and more color or false color components and the highly varying image quality and content.
- Medicine: A greater and greater variety of new technologies that produce digital images, like computerized tomography, nuclear magnetic resonance, ultrasound, and digital radiography, combines with the trend to digitize

existing X-ray images for easier storage and accessibility. Properties of medical images include the very rare use of colors and the extremely high resolution requirements, for both space and gray scale, especially in the case of X-ray images. Radiologists have well trained eyes and are probably the best critics of image compression algorithms.

- Television and video: This includes areas like high definition television (HTV), video editing systems, teleconferencing and picture phones, digital cameras, and so on. In these applications, consideration is usually given to the complicated relationship between analogue and digital representations, although there is a strong trend to complete digitalization.
- Business applications: The use of images is increasing both for internal documentation and in the contact with the customer. At the moment, the use of graphs and diagrams in presentations and reports is state of the art, but it can be expected that in the near future, no decent business presentation or report will lack some impressive images. Also, digital images will be used more and more as parts of catalogues, estimates, contracts, and so on.
- Computer graphics: The number of computer generated images increases fast, and as the generation of such images may take much time, they should be stored carefully. For images generated by the computer, high compression rates are often possible. It may also be noted that in computer graphics, there is a technique somewhat similar to progressive transmission, namely the progressive refinement of an image from a scene description by increasing the amount of calculation [Berm86].
- Personal computing: The increased availability of graphics displays and scanners at low prices makes it possible to use images "just for fun" in the personal computing environment. Bulletin boards like CompuServe already provide a wide selection of images for download, and the exchange of images among users is increasing.

3.1.2 Channel Structure

In Figure 2.1, a simple model of the transmission system was presented. However, the channel connecting sender and receiver is rarely just one direct line, as in the case of a phone line. The structure of the channel, i.e. the way in which the sender(s) and the receiver(s) are connected, can vary greatly in several aspects.

3.1.2.1 Storage and Transmission

The aim of storing an image is to keep it over time, to be able to see the same image days, months, or years later. As time is fixed, compression tries to optimize the use of space. On the other hand, transmission makes the images overcome space. Here, compression is used to reduce time.

Although in some applications, the main aim may be transmission (e.g. remote surveillance) or storage (e.g. digital photography), in most applications, these two aspects are combined, as an image, to reach its destinations, has to cross both space and time. In an image database application, for example, the image is first stored to be available whenever needed, and then transmitted on the request of an user. In a heterogeneous environment, an image may even be stored on different media and transmitted over different channels many times. Although not absolutely necessary, it is clearly desirable for the same compression technique to be used during the whole "life" of an image.

The technical terms used for storage and for transmission unfortunately differ in many ways. In this thesis terms of both fields will be used, and will in most cases include both storage and transmission.

3.1.2.2 Number of Senders and Receivers

Depending on the application and the individual image, the number of senders and receivers and their relation may differ greatly. Some images may be stored with only a small chance of ever been looked at. Other images may be viewed by a large number of people. For some images, the destination is fixed from the beginning, but in other cases, an image may play very different and unforeseen roles.

For an individual image, there is always exactly one sender, but the number of receivers varies. For a collection of images, the case of one receiver and several senders is possible in applications like remote control and surveillance. However, in general there will be one central sender and several distributed receivers. The receivers can all receive the same image in the same form and at the same time in a broadcast application, or each can work independently as in an image database application.

3.1.2.3 Open and Closed Systems

Many applications of image transmission at the moment are closed. A fixed set of images is viewed by a fixed set of users using a fixed and uniform combination of hardware and software. The reasons for this are the optimization of a system towards a specific application, security considerations, and the fact that the number of systems and users is still small. Thus applications using images are

mostly islands in a sea of traditional applications like text processing and numerical calculation.

There is however an increasing tendency towards open and heterogeneous systems. To justify the relatively high costs of acquisition and storage of images, they should be accessible by as many users as possible, or, in the case of medical or similar applications, at least by all persons authorized to see the image. The lower hardware costs make this extension of the user base possible. A large group of users will usually be spread over various locations. This, together with the variety of the data, implies a heterogeneous environment. The users will access the data over various communication channels and use different types of hardware. Compared with fixed standalone applications, this makes it necessary to formulate completely different requirements for image compression methods.

3.1.3 Applications of Progressive Transmission

In Subsection 3.2.1, the various application fields for image compression have been listed up. Here the applications specific to progressive transmission and the relation between progressive transmission and image compression are discussed. An overview of the methods used for progressive transmission is given in Subsection 1.2.3.

3.1.3.1 Traditional Applications

The situation for which progressive transmission has traditionally been proposed is the transmission of images over a low capacity line to a user directly viewing the transmitted image. Bandwidths proposed include 1200 bps (bits per second) [Slo79], [Tan79], [Tzo87], 2400 bps [Yas80], 4800 bps [Loh82], [Kno80], and 9600 bps [End87]. With these bandwidths, the transmission of an image in canonical form may take from about one minute up to more than half an hour, depending on the size of the image. This can be somewhat reduced by compression, but the user still has to wait a long time.

In most cases, the user receiving the image is not primarily interested in its complete reception. Rather, he or she has to carry out some task or take some decision based on the image. Even if in some cases, the complete reception of the image may be necessary to complete the task at hand, in most cases this is not necessary.

In other cases, the complete reception of the image may be necessary to finish the task, but the task may be started with an image of lower resolution, and so be completed faster. This is due to the fact that many tasks consist of a sequence of decisions. The first decision will assess the general relevance of the image. This

includes the rejection of erroneously transmitted images and images of too low quality. Later, the user may decide which parts of the image are relevant, and then take some measurements or decisions directly related to the overall task.

Progressive transmission was therefore proposed to reduce the long time the user has to wait for the image to develop on the screen. The image is transmitted and displayed initially at a low resolution. This can be done very fast. Then the resolution is gradually or stepwise increased until it reaches a level where the task can be completed. At this point the user can interrupt transmission. This leads to a reduction of the effective bandwidth.

Typical applications where a low data rate is combined with decisions that can be taken based on a low resolution of the image include the following:

- Telebrowsing [Kno80] in remote image databases: The user may want to get a rough overview over the stored images. Also, he or she may formulate a query which is evaluated based on the textual descriptions stored. This is usually not enough to isolate the desired image(s), so that several images have to be rejected until the desired one is found.
- Teleconferencing [Kno80], teleconsulting [Fra80], and remote teaching [Dre87]: A speaker often speaks first about the outline of a diagram or the general structure of an image, and then discusses the details.
- Security applications: Person identification, security monitoring, remote surveillance.
- Electronic shopping (mail order houses, travel arrangements) [Loh84].
- Transmission from a sender threatened with destruction [Kno80], for example a satellite: This is one of the rare examples where it may make sense to transmit several images interleaved, and where therefore rate distortion theory can fully be applied (see Subsection 2.3.3)

3.1.3.2 Progressive Transmission with Higher Bandwidths

The advent of broadband wide area networks and high capacity storage may to some extent alleviate the problem of long transmission times. Thus one might conclude that on the long term, there is no need for progressive transmission. However, as technology advances, screen size and gray scale depth and with this image size are increasing.

Also, a method that can reduce the effective transmission time from 10 minutes to 1 minute for a slow network can as well be used to reduce the transmission time from several (tens of) seconds to a few seconds or less. It is well known that the efficiency of work carried out with the computer depends

critically on the response time of the system. Only average viewing times of less than a second make true browsing through image collections and image databases possible in the same way as browsing through a book. Progressive transmission is therefore useful over a wide range of bandwidths, provided that there are algorithms that are able to work at the necessary speed.

3.1.3.3 Progressive Transmission for Image Compression

Progressive transmission can be of great help to reduce the time a user has to wait to see and use an image. However, it was long believed that progressive transmission and efficient compression exclude each other or compete with each other. In Sections 2.3 and 2.4, theoretical arguments showed that this may not be the case. This means that progressive transmission may be made more efficient than it was until now.

The progressiveness of a compression method is however also useful when the only aim of compression is the reduction of storage requirements. The main reason for this is that if an image compression method is progressive, there is a clear and direct relation between rate and distortion, which may make it worth to include progressiveness in any compression algorithm possible.

It is well known that in general, variable rate coding methods perform better than fixed rate methods. This is due to the use of adaptivity and entropy coding. However, there are cases where it is important that an image be coded with a given number of bits. One case is television broadcast, or the assignment of different bandwidths on a common transmission line for different sources, which may change depending on the importance of each source. Another case is the need to fit one image or a certain number of images on a given storage unit like a floppy disk or a memory chip card.

Using a nonprogressive, but parametrizable variable rate coding method, the best image reproduction possible with the given rate can be found by trial and error, maybe guided by binary search. However, this will greatly increase the amount of computation necessary for coding. On the other hand, if the coding method is progressive, adaption to the memory available is possible just by interrupting coding at the given rate.

In the opposite case, namely when a given maximum level of distortion has to be maintained, progressiveness is also helpful. In many parametrizable coding methods, the parameter(s) are in some way related to the distortion, but this relation may not be clear. Thus again trial and error has to be used. This is also necessary if quality is selected by the user, because he or she will not in general be able to translate the quality requirements into the corresponding parameter values. Progressive methods avoid this problem. In addition, many progressive

methods update a given pixel only a few times, and thus a distortion measure like the mean square error can easily be calculated and checked on the fly.

The above two cases can be unified by assuming that a user or a system has a certain preference for quality and high compression rates. Microeconomic theory shows that the curves of equal preference are usually convex; if using the coordinate system of Figure 2.2, they will be monotonically decreasing and convex $\cap$. If the rate distortion curve of a progressive transmission method is indeed convex ($\cup$) similar to the theoretical rate distortion curve, then the optimal coding point is easily found. Note that using a single preference function is more general than the approach of Bruckstein [Bru87], who proposes independent additive cost functions for rate and distortion.

3.2 Requirement Analysis

In this section, the requirements for image compression and progressive transmission methods are analyzed. Rather than focussing on a particular application or hardware configuration, the requirements for an *ideal* compression algorithm are discussed.

First of all, an ideal algorithm should be efficient, delivering the best compression with least distortion and least computational complexity. This is discussed in more detail in Subsection 3.2.1. However, to be usable in a heterogeneous environment, an ideal compression method also needs a large degree of flexibility. This is treated in two parts, the flexibility regarding the user and the image (soft flexibility) in Subsection 3.2.2, and the hardware related flexibility (hardware flexibility) in Subsection 3.2.3.

3.2.1 Efficiency

3.2.1.1 Rate, Distortion, and Complexity

Traditionally, individual coding methods are evaluated by their location in the triangle shown in Figure 3.1. The three goals of low bit rate, low distortion, and low complexity obviously exclude each other to a certain degree. The relation between bit rate and distortion, without regard to complexity, is bounded by the rate distortion function as discussed in Sections 2.2 and 2.3. On the other hand, the relation between the distance to the rate distortion curve and the coding complexity is based on experience; it changes slightly with many smaller and sometimes bigger steps.

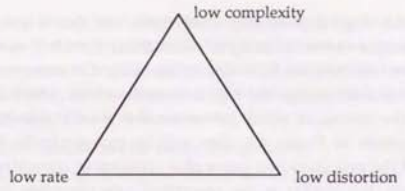

Figure 3.1. Diagram to characterize a compression method

The relation between image compression and progressive transmission has already been discussed in Subsection 3.1.3.3. In view of Figure 3.1, we can say that progressive transmission, instead of realizing only one point, allows to move from high distortion and low rate on the left to low distortion and high rate on the right. This is responsible for the advantage of using a progressive compression method.

Depending on the method used, this will be done at low complexity and far from the rate distortion curve (top) or at high complexity and near the rate distortion curve (bottom). However, with better compression methods, it may be possible to reduce complexity without compromising the result. The first requirement for an ideal compression method can therefore be formulated as follows:

Requirement 3.1: *An ideal compression method is progressive and follows the rate distortion curve as close as possible with a complexity as low as possible.*

Here it is interesting to observe another connection between progressive transmission and image compression: For progressive transmission frequent recalculations and updates may be necessary, and hence researchers concerned with progressive transmission usually prefer simple algorithms; this can lead to a reduced complexity of image compression in general. The new method presented in this thesis is a good example for this. On the other hand, researchers originally working only on image compression tend to propose methods with high complexity. This is especially the case for transform oriented methods.

The formulation of Requirement 3.1, taking the rate distortion curve as a reference, may be simple and theoretically correct. However, it turns out that human viewers usually tend to discretize the distortion dimension, even if they are aware of its continuity. Distortions frequently used as points of reference are the point where the image contents is barely recognizable (see Section 6.5), the "acceptable" distortion, the point where no distortion is recognizable, and the point where the image is completely (i.e. up to the last bit, lossless) reproduced.

3.2.1.2 Lossless or Lossy Compression

Data compression methods are sometimes divided into the two categories approximate (lossy) compression and error-free (lossless) compression. Approximate compression reduces the amount of data as much as possible with a tolerable quality reduction. Error-free compression encodes the canonical form of an image so that it can be reproduced exactly.

In some way, the requirement of losslessness for an ideal compression algorithm is already included in Requirement 3.1. However, there is some disagreement in the literature as to whether lossless reproduction is necessary. It is therefore worth to discuss this problem in somewhat more detail. Lohscheller [Loh82], considering television applications and using transform methods, denies it. Other authors [Kno80], [Dre87], [Sek89] argue that exact reproduction is strictly required for certain applications like medicine, law, satellite images, and research, and indispensable for basic services like image databases and image data exchange, which must not affect the data they manipulate.

The decision whether to use a lossless or a lossy compression method is of some importance because traditional compression methods for both cases differ greatly. However, if necessary, this gap can be filled. For most lossy methods, the coding error will tend to zero if the error images are repeatedly recoded. This has been proved by Wang et al. [Wan88] for the case of transform coding. For practical reasons, they also proposed to code the residual errors with a simple lossless coding technique after a certain number of steps.

It is therefore possible to transform a lossy method into a lossless method, but this requires an additional effort in software or hardware. Also, it may use a nonprogressive method and thus reduce progressiveness in the later part of transmission. On the other hand, some authors propose to transform a lossless method into a lossy one by using cutoff values to increase performance [Yas80], [Dre87]. For an optimal progressive method, this will obviously not be possible, as information is already transmitted exactly by decreasing relevance.

The above discussion leads to the following requirement:

Requirement 3.2: *The ideal transmission method should be lossless, and this should be achieved with a finite number of steps, all using the same algorithm, and without relying on the accuracy of the hardware.*

3.2.2 Soft Flexibility

3.2.2.1 Image Type

Many image compression algorithms are optimized to fit certain kinds of images, or can be trained for this, as in the case of vector quantization. An image

compression method for a heterogeneous environment, however, has to be able to deal with various kinds of images, and the cost of adaption should be as low as possible. Therefore,

Requirement 3.3: *An ideal image compression method has to be flexible to easily adapt to different kinds of images and different individual images.*

3.2.2.2 Spatial Resolution and Gray Scale Resolution

Progressive transmission is achieved by increasing the resolution of the image. There are basically two aspects of image resolution: Spatial resolution and gray scale resolution. In Subsection 1.2.3, we have seen that many methods either increase spatial resolution or gray scale resolution; combinations are rare and restricted in their flexibility.

That gray scale resolution can globally be traded for spatial resolution has been mentioned in Subsection 1.2.1.1. However, results on the properties of the human visual system [Sak77], [Knt85] show that in addition to this, there is an important local relation between spatial resolution and gray scale resolution. In areas with high spatial resolution (high frequency), only a few gray levels can be distinguished. On the other hand, in areas with slow gray level changes (low spatial resolution/low frequency) the visual system is very good at recognizing small gray level differences, and even differences in the first derivative (Mach banding effect). This stands in analogy to the uncertainty principle, where either velocity or position, but never both, can be measured with high accuracy.[1]

Although implicitly, this is already contained in the distortion mentioned in Requirement 3.1, it may be useful to formulate the relation between spatial and gray scale resolution explicitly:

Requirement 3.4: *The optimal progressive transmission method increases both the spatial and the gray scale resolution smoothly, continuously, and in a local and global optimal balance adapted to the properties of the human visual system and the needs of the user.*

3.2.2.3 Stepwise Improvement of the Image

The optimal way to achieve requirement 3.4 obviously is to display an image as if slowly adjusting focus and gradually increasing contrast on an analog device. This is however hardly possible, and so the improvement has to be stepwise. This means that the sizes of the steps and the change in the step size during the transmission has to be discussed.

[1] Wilson [Wil84] mentions the uncertainty principle in the context of data compression, but it is not clear whether this is meant in the same sense as above, or differently. Daugman [Dau88] also refers to the uncertainty principle.

There are different opinions on how large the steps should be. Hofmann et al. [Hof86] advocate the use of as few steps or passes over the image as possible, which should carefully be chosen to meet the requirements of the users. This is necessary because with their method, the overhead cannot be ignored; it lies between 2 to 4% per pass. However, it is rather doubtful whether the resolution necessary for each image and user can be predicted at all; Hofmann et al. do not give an example where this was actually possible. Also, if an image is seen by many different users, they may have different requirements. It would then be necessary to recode the image for each user.

In most other cases, the size of the passes is fixed by the method proposed. However, Hill et al. [Hil83] and Sanz et al. [San84] combine two passes of the bintree to one pass to avoid displaying rectangles (cf. Subsection 5.2.3.3). In tree structured vector quantization, additional steps can be introduced by using pruned tree structured vector quantization [Ris90]; this leads to faster improvement of the image.

A large number of passes is however preferable to come as close to the ideal of progressive transmission as possible. A large number of passes also distributes the increase in image quality better over the whole image. This is important because it affects the overall subjective image quality, which is not just the sum of the local image qualities. Section 2.4 also showed that at least in the high rate case, the steps can be as small as 1 bit per image without significant overhead.

Another problem is the relative size of the different passes (in bits per image), which is rarely discussed explicitly. Quadtree- or pyramid-based methods lead to exponential increases. On the other hand, transform based methods, especially if they transmit the information bitwise, use constant size passes to maintain the balance between transmission and calculation. The general form of the rate distortion curve suggests that an exponential or similarly nonlinear increase of the step size is preferable, as this leads to perceptually equal increases of image quality. This also allows to use the same pass sizes for a wide range of bandwidths.

This subsection can be summarized as follows:

Requirement 3.5: *The incremental steps of an ideal progressive transmission method should be small, particularly in the initial part of the transmission.*

3.2.2.4 Continuing Transmission or Transmission on Demand

Related to the problem of step size, some authors assume that each step is transmitted at the user's request, whereas others assume that transmission just continues until it is interrupted by the user. Basically, a continuing transmission is preferable, but there are two exceptions.

The first exception is the case where the transmission line is shared, and charges are calculated based on the amount of information transmitted and not on the time connected. In this case, the system should stop the transmission at least once, at the level of acceptable distortion or where no distortion is recognizable any more.

The second exception is the case where the decoding algorithm interferes with local calculations the user may initiate upon globally recognizing the image. In this case, the decoding algorithm, with the exception of the interface to the network, should run at a low priority.

Requirement 3.6: *In principle, transmission should continue until interrupted by the user.*

3.2.3 Hardware Flexibility

3.2.3.1 Bottlenecks

In most discussions about progressive transmission, it is assumed that the low bandwidth transmission line is the only bottleneck. There are however other bottlenecks, which become more important if progressive transmission is used for higher bandwidth channels as proposed in Subsection 3.1.3.2. Also, in a heterogeneous environment, the importance of these bottlenecks may change with every transmission.

Besides the bandwidth of the transmission line (or the storage capacity in the case of compression), the main bottlenecks are the amount of calculation necessary for coding and decoding, the amount of memory used by the algorithms, and the number of times the frame buffer is accessed or changed.

The amount of calculation can differ widely for different coding methods. Authors describing complex coding methods usually assume that specialized hardware will be used. However, although the design and production of special purpose hardware components is becoming cheaper and faster, in a heterogeneous environment it cannot easily be assumed that such hardware exists on all machines. Therefore, it is important that a method can both be implemented in hardware and in software. Actually, that a coding method works at tolerable speed in software is an important prerequisite for the wide distribution necessary to justify a hardware implementation.

Adaption to these bottlenecks can assume different forms. One possibility is that the sender or the receiver adapt to the abilities of the other side. With the exception of broadcast situations, usually the sender will adapt to the abilities of the receiver. For example, in a central database, images may be stored using a basic coding method and backend entropy coding. If the computation abilities of

the receiver are too small, the entropy coding may be decoded at the sender, possibly using specialized hardware. A good performance, relative to the means available, may still be possible.

The other possibility is to use the same format for all communications and to allow the sender or the receiver to produce or interpret the data depending on its abilities. At first sight, this may seem impossible or inefficient, but in Sections 5.1 and 5.2, examples of this kind of flexibility will be shown.

This subsection can now be summarized as follows:

Requirement 3.7: *An ideal coding method should be implementable both in software and in hardware and be adaptable to the different bottlenecks in the transmission line, the receiver, and the sender.*

3.2.3.2 Scalability

Different output devices at the receiver may have different resolutions, both in space and in gray scale. Endoh et al. [End87] therefore proposed to use progressive transmission to first transmit an image at a lower resolution for the display on a monitor, and then to increase resolution if a printout on a high resolution printer is desired. Also, they proposed to store the initial parts of all images as samples in a local database to allow the user to quickly select desired images, and then to obtain the remaining parts of the image description from a central database if needed.

A certain degree of scalability is inherent in every progressive transmission algorithm, but there are large differences regarding the degree to which the initial approximations are good reproductions of the final image at the appropriate size. Methods based on Gaussian filters will produce the best approximations for this purpose, whereas averaging and simple subsampling, in this sequence, lead to less pleasing results.

The change from an image for a monitor (low spatial, but high gray scale resolution) to an image for a printer (high spatial, but low gray scale resolution) is not possible simply by taking the appropriate subsamples or bit planes from an image at a very high resolution, as Endoh et al. [End87] seem to assume. Also, if the sizes of samples and original images in local and remote database differ by a factor of 2^4 or more in each direction (this may be necessary to use this idea efficiently), it might be better to use a high quality method independent of progressive transmission to produce the sample.

With these restrictions in mind, the last of the requirements for an ideal method for image compression and progressive transmission can be formulated:

Requirement 3.8: *An ideal progressive transmission method should include scalability.*

4

The New Method

In this chapter, the new method for image compression and progressive transmission and the concepts related to it will be developed. The data structure aspect of the new method is captured in the bitwise condensed quadtree (BC quadtree, Section 4.1). The gray scale depth first expression (GDF, Section 4.2) expresses the same structure as a sequence of symbols. Syntactical aspects of this symbol sequence are discussed in Section 4.5 using the concept of traces.

In Section 4.3, the image information condensed so far is divided into several components based on spatial and gray scale resolution; this allows progressive transmission adapted to the properties of the human visual system. A new approach to sampling and quantization suited to the method, the concept of hierarchical sampling restricted quantization (HSRQ, Section 4.4), is also presented. finally, the new concepts and the new method are compared with previous methods and concepts of data compression and image analysis.

In several sections, implementation details are interleaved with the conceptual discussion. This should provide additional insight into the concepts discussed. Also, showing that efficient implementation is possible is important to demonstrate the usability of the concepts and methods introduced. Nevertheless, readers interested mainly in the conceptual aspects may skip the corresponding subsections (Subsections 4.1.2, 4.1.3, 4.2.2, and 4.3.5).

4.1 The Bitwise Condensed (BC) Quadtree

4.1.1 The Concept of the Bitwise Condensed Quadtree

Our new method of image compression and progressive transmission is based on the concept of the *bitwise condensed quadtree* (BC quadtree), as first presented in [Dür88]. The BC quadtree combines gray scale and spatial hierarchies in a single

tree in a simple and efficient way. It can be best explained by recalling the example image of Figure 1.1 and its quadtree, which is again shown in Figure 4.1.

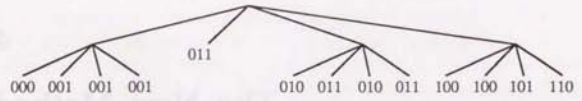

Figure 4.1. Quadtree for the image of Figure 1.1

For gray scale images, the condensation achievable with the (region) quadtree is not very high. It is rare that four neighboring pixels have the same gray values. However, neighboring pixels have mostly similar values. Thus in many cases their leading bits are the same.

Therefore to achieve a high degree of condensation without loosing any information, it is best to combine the spatial and the gray scale hierarchy and to condense the quadtree not based on the whole gray values, but bitwise. Starting from the most significant bit, whenever all the first bits of all the children of a given node are the same, these bits are removed and a corresponding bit is added to the entry of their parent node. For our example, the BC quadtree is depicted in Figure 4.2.

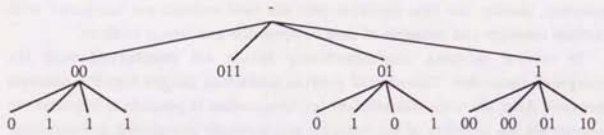

Figure 4.2. BC quadtree for the image of Figure 1.1

4.1.2 Internal Storage

The BC quadtree is a conceptual construct, and it is not very efficient to implement this tree as a pointer quadtree or as a linear quadtree. The data structure most suited for the internal representation of the BC quadtree is the explicit quadtree of [Woo82], [Burn83]. In the explicit quadtree, each node has a fixed location, and so references from a node to its parent or children are simple and fast. No overhead is needed to store any pointers or location codes.

Both Woodwark [Woo82] and Burton et al. basically store the same information in both leaf nodes and internal nodes. In the case of the BC quadtree, it is not advisable to store all the information shown for a node in Figure 4.2 in the same location. The number of bits to be stored may be very small, but the control information to indicate the number of bits and their position is difficult to organize.

The following approach, shown in Figure 4.3, is therefore used: In the leaf nodes, the full pixel values are stored, but no control information. This has the advantage that in most cases, the frame buffer can directly be used to hold these values, and no additional memory is necessary. The higher levels of the explicit quadtree do not store any of the common bit values. In the explicit quadtree, it is always easily possible to find any leaf below the current node, and any of these leaves will contain the necessary common bits.

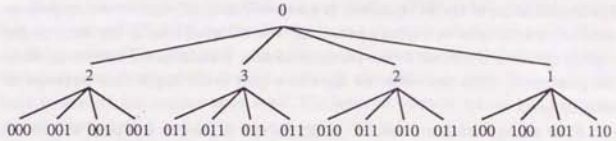

Figure 4.3. The explicit quadtree for the BC quadtree of Figure 4.2

In place of the common bits, the interior nodes in the explicit quadtree contain the bit position of the last common bit at that level, and 0 if there are no common bits. The bit position of the first common bit in a given node of the BC quadtree is then found by incrementing the entry in the parent node of the explicit quadtree by one. The root node does not have a parent node, but we can assume a parent node containing the value 0. For the leaf nodes, the last "common" bit is always bit b, and so there is no need to store it.

For an image with n pixels and b bits per pixel, this representation needs not more than

$$(n\text{-}1)/3 \cdot \lceil \log(b+1) \rceil \quad \text{bits} \tag{4.1}$$

in addition to the memory necessary for the original image. In many cases, b=8, and then it is simplest to arrange the explicit quadtree so that each node uses one byte. The total storage necessary then amounts to

$$4/3 \cdot (n\text{-}1) \quad \text{bytes.} \tag{4.2}$$

As for each interior node, only four of eight bits are used, it is possible to duplicate the entry in the parent node; this may accelerate some operations.

There are several ways to arrange the nodes of the explicit quadtree in memory. The variant of Burton et al. [Burn83] is a quaternary version of the heap

structure used in the heap sort algorithm. This structure was used in some early implementations of the new method. The variant of Woodwark [Woo82] is not very useful because neighboring nodes are scattered in memory.

Another variant, shortly mentioned in [Woo82], is to arrange the nodes on each level in scan line order. This makes the calculation of the location of related nodes somewhat more difficult, but has the advantage that it coincides with the order the pixels are arranged in most frame buffers. Also, when using the BC-quadtree for progressive transmission, it leads to a straightforward top to bottom update of the display for each pass, which may be visually more pleasing than the update according to the Morton sequence, the sequence of the subquadrants of a given size in a depth first traversal of the quadtree [Sam84].

4.1.3 Construction of the BC Quadtree

The construction of the BC quadtree is extremely simple; most of the operations used are simple bitwise logical operations. The BC quadtree, in the form of the explicit quadtree described in the previous section, is constructed bottom up from the pixel array. This can either be done in a post order depth first traversal or level by level.

If for a given interior node, let p_1, p_2, p_3, and p_4 denote any pixel in each of its four subquadrants, and the symbols "|", "&", and "^" bitwise logical OR, bitwise logical AND, and bitwise logical exclusive or (XOR). Then a value q can be calculated for any interior node on the first level above the pixel level as follows:

$$q = (p_1 | p_2 | p_3 | p_4) \wedge (p_1 \& p_2 \& p_3 \& p_4) \qquad (4.3)$$

This q will have all those bits set that are not common in p_1, p_2, p_3, and p_4. The entry for the explicit quadtree can now easily be found from q by table lookup. For all 2^b possible bit combinations of q, this table will contain the number of contiguous zeroes starting from the most significant bit. Another possibility is to store q, or an unified version of it, for example with all lower bits set, directly in the explicit quadtree. For interior nodes at higher levels, the intermediate results of their children have also to be considered. Denoting them by q_1, q_2, q_3, and q_4 leads to

$$q = [\,(p_1 | p_2 | p_3 | p_4) \wedge (p_1 \& p_2 \& p_3 \& p_4)\,] \;|\; (q_1 | q_2 | q_3 | q_4). \qquad (4.4)$$

There are several other possible forms for the formulas (4.3) and (4.4). It is easy to see that the construction of the BC quadtree can be done in time linear to the number of pixels in the image, with a very low constant due to the extremely simple operations used.

4.2 The Gray Scale Depth First Expression (GDF)

The BC quadtree described in the previous section allows the condensation of the information about the given image. To transmit or store this information efficiently, this section develops the gray scale depth first expression (GDF) and its binary coding.

4.2.1 DF and GDF

For binary quadtrees, the depth first expression (DF) was proposed by Kawaguchi et al. [Kaw80]. It is produced by traversing a quadtree in depth first preorder sequence, and outputting **(** for an interior node, **0** for a white leaf node, and **1** for a black leaf node[1].

A gray scale image can be coded in a way similar to DF, called *gray scale depth first expression* (GDF). GDF results directly from a preorder depth first traversal of the BC quadtree. The symbols **0** and **1** are used for individual bits instead of leaf nodes, whereas the symbol **(** is used to denote the end of an entry in an interior node instead of the interior node itself. The term *bit selection symbol* will be used later to denote both **0** and **1**. For our example, GDF is

(00(0 1 1 1 011 01(0 1 0 1 1(00 00 01 10.

Blanks have been added for legibility. The parentheses used in GDF can also be added conceptually to the BC quadtree, as shown in Figure 4.4.

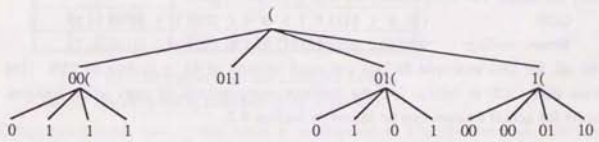

Figure 4.4. The BC quadtree with parentheses

Algorithms for the coding and decoding of GDF have been given in [Dür88c]. However, it is not necessary to construct GDF, and transmit the information in the BC quadtree, in depth first order. Actually, the fact that other transmission sequences are possible is crucial for the use of the BC quadtree for image compression and progressive transmission. Therefore, the term GDF will be used

[1] The bold letters **(**, **0**, and **1** will always be used for the symbols of DF or GDF, whereas the letters 0 and 1 will denote bit values.

later for any sequence of 0s, 1s, and (s derived from the BC quadtree, even if this is not in accordance with the original meaning of GDF.

It is of course necessary to show that GDF uniquely describes the image it has been derived from to show that Requirement 3.2 can be fulfilled. A proof that the original image can be reconstructed from its GDF was given in [Dür88c]. In this thesis, general conditions will be given in Section 4.5 for the orders of the symbols in GDF for which reconstruction of the image is possible. Also, it will be shows that the depth first form of GDF satisfies these conditions.

4.2.2 Binary Coding of GDF

Kawaguchi et al. [Kaw80] propose several ways to code the three symbols **0**, **1**, and **(** with the two digits of the binary system. On the lowest level of spatial subdivision, there are no (, and so coding is very simple. On the other levels, it is not clear which of the three symbols is the most frequent and therefore should be assigned a one-bit code. Contrary to Kawaguchi but in accordance with Knowlton [Kno80] and Tamminen [Tam84b], here **(** is coded with one bit. This preserves the symmetry between the bit selection symbols **0** and **1** and so simplifies program code and size calculations (see Section 5.3). Also, in this way, the maximal length of a "bad" image is shortest, namely (for an image of $2^{2r}{\cdot}b$ bits)

$$2^{2r}{\cdot}(b+1/3) \text{ bits.} \qquad (4.5)$$

Adapting "10" (on higher levels) or "0" (on the pixel level) for **0**, "11" or "1" for **1**, and "0" for (, and coding the GDF of our example image gives the following binary sequence. For comparison, we also give GDF again.

GDF: **(0 0 (0111 0 1 1 0 1 (0101 1 (00 00 01 10**

Binary coding: 0 10 10 0 0111 101111 10 110 0101 11 0 00 00 01 10

Over all, for this example 36 bits are used instead of 48, a saving of 25%. This shows that GDF is useful for the lossless compression of gray scale images. Results for actual images can be found in Section 6.2.

4.3 Image Components

Depth first transmission of the symbols in the BC quadtree is not suitable for progressive transmission. The symbols of GDF have to be reordered in some way so that more important symbols are transmitted first. A method to achieve this is to partition the symbols into several components which are transmitted one after the other. Inside each component, the original sequence is retained. In this section, a simple way to split the image information into components is presented. Other, more elaborate ways of forming components will be discussed in Subsection 5.2.3.

4.3.1 Partitioning of the Image into Components

Each symbol in the BC quadtree or in GDF can be assigned a spatial level s ($0 \le s \le r$) and a gray scale level c ($1 \le c \le b$). The spatial level is the level in the BC quadtree occupied by the symbol. The gray scale level is the level of the symbol in the gray scale hierarchy, equal to the bit position (1 for the most significant bit, b for the least significant bit). In Figure 4.5, the BC quadtree of our example is again depicted, indexing every symbol with its gray scale level. Contrary to [Dür88c], parentheses have been indexed with the gray level of the first noncommon bit.

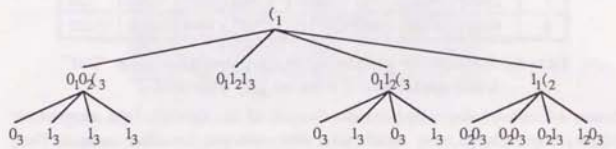

Figure 4.5. The indexed BC quadtree

The symbols of the BC quadtree can now be divided into $(r+1){\cdot}b$ components depending on their spatial level s and their gray scale level c. The components for our example image are shown in Table 4.1.

	1	2	3
0	(	-	-
1	0001	011(	(1(
2	-	0001	0111 0101 0010

Table 4.1. Components of the example image
(rows: spatial level s; columns: gray scale level c)

Components with low spatial level s correspond to big squares and thus low spatial resolution. Components with high s represent small squares and thus high spatial resolution. Similarly, components with low and high gray scale level c can be associated with low and high gray scale resolution.

As a more realistic example, Table 4.2 gives the size, in number of symbols, for each component of the image "Girl". This well known test image, as well as some others used in later chapters, is part of the Japanese standard image database (SIDBA) [Ono79]. It has a spatial resolution of r=8 (256·256 pixels) and a gray scale resolution of b=8 (256 gray levels). The original can be seen in Figure 4.6 or Figure 6.1.

	1	2	3	4	5	6	7	8
0	1	0	0	0	0	0	0	0
1	4	0	0	0	0	0	0	0
2	16	2	0	0	0	0	0	0
3	56	18	5	3	0	0	0	0
4	184	122	44	37	0	0	0	0
5	456	510	273	244	31	1	0	0
6	1160	1772	1616	1512	535	83	9	0
7	2560	4910	7150	8870	6274	2911	889	228
8	4424	10760	21432	39112	53748	61980	64624	65232

Table 4.2. Size (number of symbols) of components for image "Girl" (rows: spatial level *s*; columns: gray scale level *c*)

Tables 4.1 and 4.2 show that the distribution of the symbols into components follows a general pattern, which was also observed for other images. First, components at low spatial resolution contain only very few symbols. This is due to the fact that the description of an image at low resolution is much more concise than at higher levels, and to the fact that condensation, even for the most significant bit, seldom can be achieved for the whole image or very large parts of it.

Second, the size of components at intermediate spatial levels first grows with increasing bit position, but then shrinks again for bits with low significance. Obviously, more significant bits have already been condensed at higher levels of the spatial hierarchy, whereas less significant bits change too irregularly for a large amount of condensation to be achieved on the present level of the spatial hierarchy.

Third, most symbols are concentrated on the lowest level of the spatial hierarchy. Therefore, it might seem that the BC quadtree and GDF are not very efficient in condensing image information. It should however be noted that these symbols affect only a single pixel, whereas symbols on higher levels affect large squares. Also, all symbols on the lowest level of the spatial hierarchy can be coded with one bit, whereas symbols on higher levels may need more than one bit. The increasing size of the components also meets Requirement 3.5 (increasing size of passes).

4.3.2 Increasing Gray Scale Resolution

Progressive transmission can be realized increasing either gray scale resolution or spatial resolution. Using the BC quadtree and the component partitioning,

progressively increasing gray scale resolution is realized by first transmitting all the components with gray scale level 1, next those with gray scale level 2, and so on. The result is shown in Figure 4.6; the last image shows the original.

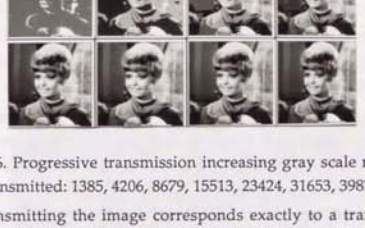

Figure 4.6. Progressive transmission increasing gray scale resolution (bytes transmitted: 1385, 4206, 8679, 15513, 23424, 31653, 39872, 48064)

This way of transmitting the image corresponds exactly to a transmission by bit plane as described in Subsection 1.2.3.2. Due to the properties of the quadtree, compression is achieved, especially in the more significant bit planes. Stored in canonical form, each bit plane occupies 8192 bytes, and the whole image together occupies 65536 bytes. Note also that the components in each bit plane are transmitted starting with the lowest spatial resolution. With this, an additional progressive transmission effect is obtained that is not visible in Figure 4.6.

Compression in all bit planes except the most significant could be increased by taking advantage of codes like the Gray code [Knt78], [Kaw83] or the unit-distance-equalized significance (UDES) code [Knt78]. These codes reduce the number of bits differing from one gray level to the other and thus lead to additional condensation if each bit plane is treated separately. However, in this case, the structure information, captured by the parentheses of GDF, cannot be shared by different bit planes. Experiments presented in Section 6.2 show that GDF is more efficient than the individual application of DF to each bit plane of a Gray coded image as proposed in [Kaw83].

4.3.3 Increasing Spatial Resolution

Using the partitioning of the image introduced in Subsection 4.3.1, progressive transmission by increasing spatial resolution can also be realized easily. The components are transmitted in order of increasing spatial resolution *s*, as shown in Figure 4.7.

Figure 4.7. Progressive transmission increasing spatial resolution
(bytes transmitted: 1, 3, 15, 80, 331, 1490, 7900, 48064)

This method of increasing spatial resolution is clearly different from the methods mentioned in Subsection 1.2.3.1. For each square, instead of an average value or the gray value of a specific pixel, only the value of the common most significant bits is displayed.

This has the consequence that the images in the upper row of Figure 4.7 contain much less visual information than images at the same resolution produced with other methods. On the other hand, the number of bytes transmitted is also much lower. In general, decreasing the spatial resolution by one level reduces the necessary number of bytes by about a factor of four; here, this factor is larger.

4.3.4 Increasing Both Spatial and Gray Scale Resolution

The most important consequence of partitioning GDF as in Subsection 4.3.1 is that this not only allows to progressive transmission increasing gray scale resolution or spatial resolution. It becomes possible to combine the increases in spatial and gray scale resolution in many different ways. This can be used to adapt the sequence of components to the properties of the human eye and the image according to Requirements 3.3 and 3.4.

There are some restrictions on the possible sequences of components, but these restrictions can be formulated in a very simple way. To be able to transmit a given component x, all the components that have smaller or equal resolution in both aspects (spatial and gray scale), i.e. all components for which both $s \leq s_x$ and $c \leq c_x$, have to have been transmitted already.

This says nothing more than that spatial resolution, for a given bit plane, and gray scale resolution, for a given subquadrant size, have to be increased step by step. Transmitting components with higher resolution before those with lower

resolution is any way inefficient to achieve progressive transmission, and so the above restrictions are not very stringent.

A first example of combining the increase of spatial and gray scale resolution is shown in Figure 4.8. More examples can be found in Chapter 5 and 6. The sequence of components used is included in Appendix A. It was determined by having a human observer compare several variants of component sequences using the image "Girl". This sequence was found to perform well for various other images of different classes, too. Ways to determine optimal component sequences and the problems that can be encountered when this is done are described in Section 5.2.

Figure 4.8. Increasing both spatial and gray scale resolution
(bytes transmitted: 300, 600, 1200, 1800, 2400, 3600, 4800, 9600)

4.3.5 Transmission Algorithms

In this subsection, outlines of the actual algorithms that implement progressive transmission based on the BC quadtree and GDF are presented. The data structure used by both the sender and the receiver is the explicit quadtree as explained in Subsection 4.1.2. The sender's algorithm can then be outlined as follows:

1. As a preliminary step, construct the explicit quadtree from the image data as described in Subsection 4.1.3.
2. Decide the order of the components and send this information.
3. Transmit each of the components. For the component with spatial level s and gray scale level c, traverse the nodes of level s and apply step 4 to each node.
4. If the gray scale level recorded in the current node of the explicit quadtree (the index of the last common bit) is one smaller than c, then output a parenthesis. Else, if c is greater or equal than the recorded gray scale level and

smaller than the gray scale level recorded in its parent, then output the corresponding bit in the frame buffer in the form of the corresponding bit selection symbol. If none of the above conditions apply, no output is necessary.

Step 2 can be omitted if the order of the components is fixed. Otherwise, a small improvement is possible if the identification for each component is sent just before the component itself.

The receiver does not need to keep the index of the last common bit of each internal node if it does not intend to use the BC quadtree structure at the receiver for purposes other than the progressively refined display of the transmitted image. A one-bit flag for each interior node, and a second one-bit flag for the nodes on the second-lowest level of the explicit quadtree, is all that is needed, reducing the storage besides the frame buffer to (cf. (4.1))

$$(n\text{-}1)/3 + n/4 \quad = (7n\text{-}4)/12 \quad \text{bits.} \tag{4.5}$$

The flag denotes whether a node is currently receiving symbols or not. The additional flags on the second-lowest level are used for the pixel level. On this level, it suffices to have one bit for every four pixels, as all four pixels in a two by two square will change their state at the same time. Then the outline of the algorithm on the receiver's side is as follows:

1. As a preliminary step, clear all flags except that of the root node, and initialize the frame buffer.
2. Receive the information concerning the order of the components.
3. Receive each of the components. For the component with spatial level s and gray scale level c, traverse the nodes of level s and carry out step 4 for those nodes whose flag is set.
4. Receive one symbol. If this is a parenthesis, then set the flag of this node to 0 and the flags of its children to 1. Else, if this is a bit selection symbol, then paint the square corresponding to the present node with the appropriate gray value.

Step 2 can be adapted according to step 2 of the sender's algorithm. The "appropriate gray level" mentioned in step 4 will be discussed in further detail in Section 5.1.

The extraordinary simplicity of both algorithms, especially the one on the receiver side, should be noted. No multiplications are necessary, and the only additions used occur for address calculations. In many graphics displays and workstations, a square can be painted with a single command. If this is not the case, then variants of the algorithm to reduce the number of times the frame buffer is accessed can be used (see Section 5.1).

The small amount of memory needed besides the frame buffer makes it easy to implement the algorithms in hardware. It may even be desirable to integrate the receiver's algorithm into the frame buffer. This is especially true if the grid of the quadtree is aligned with the addressing of the frame buffer. Much more complex additions to frame buffers are already in use. For a summary of the latest developments, see [Fuc89].

4.4 Hierarchical Sampling Restricted Quantization (HSRQ)

In the previous sections, the concept of bitwise condensation has been used to construct the BC quadtree and GDF, proceeding bottom up. In this section, the principle of Hierarchical Sampling Restricted Quantization (HSRQ) will be introduced as the top down counterpart of the BC quadtree and GDF. The examples in this section will be one-dimensional. This makes it possible to show both the spatial dimension and the gray scale dimension. This section also intends to show that the BC-quadtree, GDF, and HSRQ can be applied to data other than images, such as sound and three-dimensional density data.

4.4.1 Sampling and Quantization with PCM

Given a smooth function $g = f(t)$ $(t_0 \le t \le t_1)$, how can we represent it more or less accurately by a finite number of bits? This is usually done by *sampling* the function at regularly spaced sampling points and *quantizing* the sampled values to a fixed number of representative values. This way of coding a function is called PCM (see Subsection 1.2.1.1). This process is shown in Figure 4.9.

Coding the function in this way needs 64·4 = 256 bits. This number can be reduced by reducing the number of sampling points or quantization levels, but this will lead to problems. In areas where high frequency components dominate (the left part of Figure 4.9), the number of sampling points cannot be reduced too much, whereas in areas where low frequencies dominate, the number of quantization levels cannot be reduced too much without severely affecting reproduction quality.

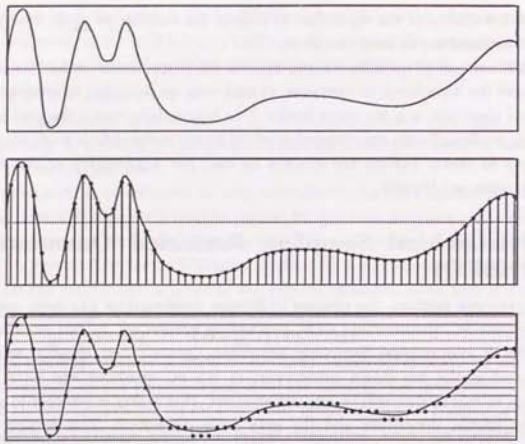

Figure 4.9. Sampling and quantizing a function
(top: original function $g = f(t)$; middle: function sampled at 64 regularly spaced sampling points; bottom: function quantized to one of 16 levels)

4.4.2 Interval Coding

Instead of performing sampling and quantization as two independent steps, and approximating the function by a number of points, it is also possible to approximate the function by intervals in both the sampling dimension (space or time) and the quantization dimension (gray value, voltage, etc.). In Figure 4.10, the same function as in Figure 4.9 has been roughly approximated by ten two-dimensional intervals.

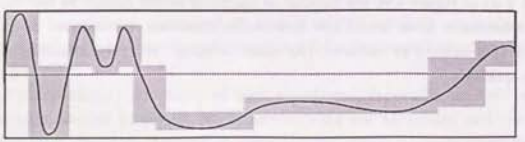

Figure 4.10. Approximating a function by intervals

The intervals in Figure 4.10 have been selected more or less arbitrarily, but so as to give a good idea of where the function passes. The facts that the intervals completely contain the function, and that they are tight, should allow a fairly good reproduction of the original function. If the interval boundaries are coded at the same resolution as the samples of Figure 4.9, then the interval representation needs $(4+6)\cdot 2\cdot 10 = 200$ bits.

It should be noted that the intervals have different aspect ratios in different areas. The intervals in the high frequency area are narrow and high, whereas the intervals in the low frequency area are wide but low. This shows that the interval representation is able to adapt to parts of the function with different characteristics.

4.4.3 Hierarchical Intervals

Arbitrary intervals as those in Figure 4.10 represent one possibility to approximate a function, but they do not easily allow to refine this approximation. To allow progressive transmission or compression at any rate, it is better to use a hierarchy of intervals. Starting with one interval that includes the whole domain and range of the function $f(t)$, this interval is successively replaced by more and smaller intervals.

Actually, in many cases the domain of the function may be very large, and so it may be advisable to start with a number of intervals of a certain size. This is especially important if the domain is not known in advance, as for example when coding sound. For image compression some small savings are also possible by starting with squares smaller than the whole image, because it is very rare that even the first bit can be condensed over the whole image. Squares of 64·64 or 32·32 pixels seem to perform best. This may also allow a certain parallelization if the method is implemented in hardware.

When going from large to small intervals, there are two steps: The *splitting* of a larger interval into two or more smaller intervals that together cover the same area, and the *reduction* of the size of an interval, eliminating parts that are not passed by the function. Basically, both of these steps can be carried out in both dimensions, as shown in Figure 4.11.

However, as general functions do not have an inverse, the natural way is to apply splitting to the sampling dimension and interval reduction to the quantization dimension. In this case, splitting of an interval in the sampling dimension can simply be called *sampling*, whereas reduction of an interval in the quantization dimension is called *quantization*. The subsequent samplings of the original interval define a sampling hierarchy.

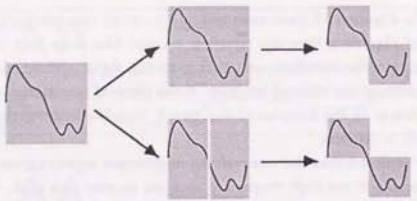

Figure 4.11. Two ways of splitting and reducing intervals

For the reduction of intervals in the quantization dimension (quantization), several not necessarily nonoverlapping subintervals may be provided. If this may lead to ambiguities, the priority of the subintervals has to be specified clearly. The simplest solution, which is adopted here in consistency with the bitwise condensation of the BC quadtree, is to use the two halves of an interval as subintervals. This leads to the three possibilities for the refinement of a given interval shown in Figure 4.12. These three possibilities correspond exactly to the three symbols of GDF.

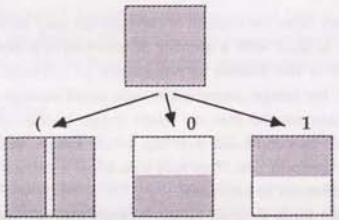

Figure 4.12. Interval refinement, and corresponding symbols of GDF

4.4.4 Quantization Restricted by Hierarchical Sampling

Obviously an interval can be sampled even when quantization is possible. However, this is inefficient, as the two sampled subintervals will have to be quantized in the same way. On the other hand, it is not possible to quantize an interval if sampling has not progressed enough and the function still passes both the upper and the lower half of the interval. Thus quantization is restricted by the sampling hierarchy. This is the principle of *hierarchical sampling restricted quantization*, or HSRQ.

HSRQ can be carried out in several steps, using several rules to decide on the next interval to be sampled or quantized, and eventually additional rules that decide what variants of sampling and quantization are allowed. These rules may, corresponding to Section 4.3, be given in the form of a component sequence, each component consisting of the intervals of a given length and height. Other variants are discussed in Section 5.2. Several steps of applying HSRQ to the function of Figure 4.9 are shown in Figure 4.13.

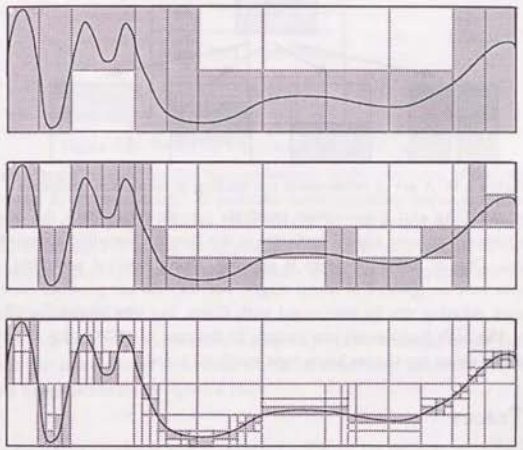

Figure 4.13. Approximating a function by HSRQ

Again, the aspect ratios of the intervals adapt to the local properties of the function. Ways to approximately reconstruct the function even for large intervals will be discussed in Section 5.1.

Please note that not all intervals have been completely quantized, especially in the area of high frequencies, and that the function many times passes a horizontal and a vertical grid line at the same time, which cannot be expected from arbitrary smooth functions. If the function is already represented in canonical form, and HSRQ is used only for resampling and requantization, then this poses no problems.

Otherwise, this leads to resolution problems that can be compared with the "black hole" phenomenon [Dür88a,b,89a], which can appear in the polytree, a variant of the octree for the representation of polyhedral objects. The solution in this case is to provide more possibilities for refinement. The smallest set of refinements that avoids resolution problems is shown in Figure 4.14.

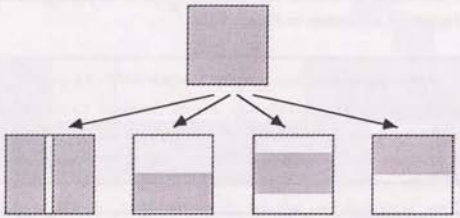

Figure 4.14. A set of refinements not leading to resolution problems

That the sampling and quantization intervals completely contain the original function means that some typical problems of the classical sampling approach are eliminated. One of these problems is aliasing. Using interval sampling, high frequencies may be ignored at initial stages, but they do not produce false low frequencies. Aliasing can be eliminated with filters, but this introduces another problem: The high frequencies just vanish. In the case of HSRQ, this is not the case; in some sense, we always know how much we know.

4.5 Traces

Kawaguchi et al. [Kaw80,83] have used a context free grammar to describe and analyze DF. Unfortunately, a grammar for GDF is not context free. It is possible to formalize GDF with an attributed grammar, using the positions in the spatial and gray scale hierarchy as attributes. However, this is overly tedious and not very rewarding. The concept of traces introduced in this section provides a lighter and more productive tool to treat some formal aspects of the BC quadtree and GDF. Traces will be used in Subsection 5.1.4 to describe methods of reproducing the image at intermediate stages, and in Subsection 5.2.3 to define other ways of splitting the image information into components.

4.5.1 Traces for Pixels

Traces can best be defined based on the BC quadtree. The *trace* of a pixel is the string of symbols encountered on the way from the root of the BC quadtree down to that pixel. The traces for our example image are shown in Figure 4.15.

(00(0	(00(1	(011	(011
(00(1	(00(1	(011	(011
(01(0	(01(1	(1(00	(1(00
(01(0	(01(1	(1(01	(1(10

Figure 4.15. Traces for the example image of Figure 1.1

The name "trace" is used for this concept because of the similarity to the processes traces used by Hoare [Hoa85] to describe communicating sequential processes (CSP). In the same way a process trace records all the events encountered by a process, and is used to reason about processes, a (pixel) trace records all the symbols affecting a pixel, and can be used to reason about the BC quadtree and GDF.

4.5.2 The Trace Tree

To get an overview over the possible traces, they can be arranged in the form of a *trace tree* (the more correct expression would be trace trie). The trace tree for our example image is shown in Figure 4.16.

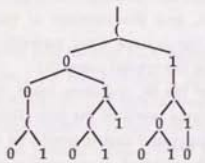

Figure 4.16. The trace tree for the example image

In the trace tree, all equal traces are merged. The maximum number of leaves in a trace tree can be calculated from the number of bits per pixel, b, and the resolution of the image, r, as follows: In a trace, there are exactly b bit selection symbols and at most r parentheses, in any order. Traces with less than r parentheses can be

uniquely extended by adding the missing parentheses at the end. This extension will be assumed frequently below, without always mentioning it. It simplifies argumentation about traces in many cases. The two bit selection symbols can appear in any order and frequency. Thus the number of possible traces is

$$\frac{(b+r)!}{b!\cdot r!}\cdot 2^b. \tag{4.6}$$

This evaluates to 80 for our small example and to 3,294,720 for images such as that used in Section 4.3. However, the number of traces of an actual image is much smaller in any case. First, it is limited by the number of pixels, 2^{2r}. Second, it is reduced by the coherence of the image. Third, all traces of an image will start with the same symbol, the first symbol of the GDF of the image.

The trace tree can be used to count the numbers of pixels in an image for each trace, and to calculate other properties as explained in Chapter 5. It can easily be constructed by a depth first traversal of the BC quadtree.

4.5.3 Subtraces

Traces cannot only be defined for pixels (leaf nodes in the BC quadtree), but also for larger squares (interior nodes in the BC quadtree). Obviously, they are the strings of symbols encountered from the root of the BC quadtree down to the corresponding nodes. Traces for general squares are one kind of subtraces, a general way of constructing new traces. If traces are interpreted as strings, the trace of a large square is by definition the initial substring (called *initial subtrace*) of the trace of any of the pixels, or subquadrants, in that square.

Turning this the other way round, any trace of a node in the BC quadtree contains as initial subtraces all the traces of its ancestors. As the entry for an internal node in the BC quadtree always ends with a parenthesis, these substrings also end with a parenthesis, and the number of parentheses they contain corresponds to the size of the square (less parentheses for larger squares and vice versa) and the level in the spatial hierarchy.

Contrary to the entries of the BC quadtree itself, the corresponding traces contain information about various levels of the tree. This is somewhat similar to a proposal by Klinger, who suggested that it would be of interest to investigate the expressive power of hierarchical structures like the quadtree whose condensation or splitting rules consider more than just one or two levels of the hierarchy [Kli89].

That traces contain information about ancestors means that they also contain some information about neighbors. This is important because it is essential to see an image not just as a set of pixels, but to consider the neighboring relations of these pixels. Traces provide an efficient way to do this.

There are other ways of forming subtraces. For example, it sometimes provides useful to extract the bit selection symbols from a trace α[1]. This is denoted by $bi(\alpha)$ and called the *bit selection subtrace* of α. It is not a substring in the original sense of the word because the bit selection symbols in α may be separated by parentheses.

Extracting the parentheses and so forming the parenthesis subtrace of a trace α is also sometimes useful. However, a simple string of parentheses does not contain very much information. The following convention is therefore adopted: To form the *parenthesis subtrace* $pa(\alpha)$, the bit selection symbols are not eliminated but replaced by the symbol $\Diamond$, called the *neutral bit selection symbol*. Still, when taking the length of $pa(\alpha)$ with $|pa(\alpha)|$, this will be defined as the number of parentheses only. Parenthesis subtraces do not say anything about the gray value of a pixel or quadrant. However, they provide valuable information about the degree of activity or flatness around that pixel or quadrant.

4.5.4 General Conditions for Decodability

After having introduced the trace concept, GDF can be seen as a sequence of the traces of the pixels in the image it describes. Various traces are interleaved, and if all the traces of the pixels in a given subquadrant have the same initial subtrace, then this subtrace is included only once and shared by all the pixels affected.

Therefore, for a string of symbols (GDF) to represent an image, may be at a lower resolution than the original, it has to fulfill the following conditions:

- The symbols contained in GDF have to appear in the original order.
- The trace(s) to which a symbol belongs have to be apparent to the receiver from the previously received symbols (or from side information sent beforehand).

It is easy to see that the depth first form of GDF fulfills these two conditions. The first condition is guaranteed by the top-down progression of a depth first traversal. The second condition is also guaranteed: Receiving a parenthesis indicates that a common initial subtrace ended and the following symbols belong only to a subquadrant of the present node. That a trace is completed, and we therefore have to pass from the present node to the next child of the first (bottom up) ancestor that still has some children, is detected by counting the number of bit selection symbols of the present trace.

---

[1] Greek letters are used to denote traces in accordance with the general notation for strings.

In the same way, it is possible to show that the conditions for the component sequences mentioned in Section 4.3 are necessary and sufficient for correct transmission. Necessity follows from the first of the above conditions. Sufficiency is fulfilled if we assume a fixed ordering of all the subquadrants at any given level of spatial resolution.

4.6 Comparison with Other Methods

This section gives a conceptual comparison of the BC quadtree, GDF, and HSRQ with some other (image) compression methods. For a comparison of compression rates, the reader is referred to Chapter 6. Besides providing deeper insight into the new method, the comparisons in this section may also lead to the combination of some aspects of the new method with other methods.

4.6.1 General Coding Principles

It turns out that HSRQ possesses some similarities with most of the important coding methods and principles. This may not be very surprising, as the principles that can be used for coding are limited, and any good coding method will take use of them in some way. More surprising is that with HSRQ, such a combination is possible at a very low complexity.

4.6.1.1 Predictive Coding

Predictive coding takes advantage of the fact that neighboring data values are highly correlated. This is also used by HSRQ, the difference being that HSRQ is limited by the arbitrary subdivisions of the spatial and the gray scale hierarchy. On the other hand, HSRQ can sometimes use coherence in the leading bits over a much larger part than predictive coding, which basically works from pixel to pixel and from line to line. How additional prediction can be introduced into the BC quadtree is shortly described in Section 5.1.5.

4.6.1.2 Transform Coding

The relation between transform coding and HSRQ can best be shown using Figure 4.17. Transform coding takes advantage of the correlation between neighboring data values by transforming a set of values so that correlation is eliminated. For two correlated values, the transformation usually results in the average and the difference of the values. As the average has more variance, it is usually transmitted with higher priority, and using more bits, than the difference.

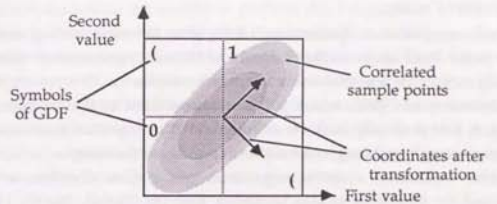

Figure 4.17 Comparison of transform coding and HSRQ

HSRQ also has a transforming effect. Value combinations whose average differs strongly from the general mean are quantized early using one of the bit selection symbols. Quantization of value combinations which differ greatly, but whose average therefore lies near the general mean, is delayed.

On a higher level, the splitting of the components according to spatial resolution in HSRQ can be compared with the frequency splitting effect of transform coding. However, contrary to traditional transforms, the frequencies in HSRQ are are distributed in octave steps; this seems to be more in accordance with the human visual system [Dau88].

4.6.1.3 Vector Quantization

HSRQ can be interpreted as a kind of tree-structured vector quantization with fixed tree structure, but variable vector size. A fixed tree structure, especially if it is as simple as in the case of HSRQ, greatly reduces the overhead for storing codebooks and for finding the nearest representative.

As vector quantization is optimized and trained to a certain class of images, it is difficult for HSRQ to achieve better results. However, the fixed vector size limits good performance of vector quantization to a certain range of rates. High rates, up to those necessary for lossless reproduction, are restricted by the codebook size, and rates lower than one bit per vector are not possible. Thus it is possible for HSRQ to outperform vector quantization; an example of this is discussed in Section 6.4.

The above discussion suggests that vector quantization may be improved by initially using large vectors, and then successively changing to smaller vectors. This has actually been proposed by Gersho and his group [Vai87], [Ho88]. The results obtained are good results, but the complexity of these methods is high.

4.6.1.3 Adaptive Coding

It has been mentioned in Subsection 1.2.1.4 that for most coding methods, adaption to the local image statistics allows additional improvements. However, this usually results in a considerably increased complexity. One reason for the good performance of HSRQ, which will be demonstrated by the experiments in Chapter 6, is that it already includes adaption, in the sense that it automatically uses different sampling strategies for different regions of the image.

Whereas traditional adaptive algorithms use a class division, adaptive coding based on quadtrees is also becoming popular [Far90], [Str90], [Vai87]. However, in these cases, the quadtree is just used to define a partition of the image into squares of different sizes. The quadtree and the actual coding method are not directly related as this is the case with the BC quadtree.

4.6.2 Image Complexity

One goal of the research in computer vision and image processing is to capture the structure of an image and the complexity of parts of an image, and to find ways to continuously reduce this complexity. In the framework of this thesis, only a very short comparison with such work can be given, but a more extensive investigation of this area may lead to interesting new discoveries.

4.6.2.1 Scale space

The theory of scale space (see [Lin90] for an introduction and additional references) defines image complexity by the number of local minima and maxima. It then tries to find ways to continuously reduce the (spatial) resolution without at any point increasing image complexity. In the spatially continuous case, this leads uniquely to Gaussian kernels of different sizes.

Applying this to the BC quadtree and HSRQ, it is of course not possible to increase resolution continuously. However, it may be possible to apply the theory of scale space to HSRQ as follows: As HSRQ only defines intervals of gray scale and space that are passed by the original function, the least complex function (in the sense of scale space) that passes these intervals can be selected. Scale space theorems can then be developed based on these functions. Note that in this case, the scale space parameter (resolution) cannot be specified by a single parameter.

4.6.2.2 Picture Information Measures

Chang [Cha89, Chapter 4] discusses information measures that indicate the complexity of an image or picture, and outlines the properties that such a measure should fulfill. Based on DF, Kawaguchi et al. define a measure for the complexity of binary images [Kaw80]. In the terminology of this thesis, it is defined as the number of bit selection symbols necessary to represent a given

subquadrant, divided by the number of pixels in this subquadrant. In [Kaw83], this measure was used to simplify noisy parts of lower bit planes of a gray scale image.

Similar information measures can be defined based on GDF. However, the advantage of the BC quadtree is that it uses a very simple rule, namely the equality of leading bits in all corresponding pixels, to very easily obtain a rough estimate of image complexity. This can then be used efficiently for progressive transmission, whereas other picture information measures are more complex and therefore less suited for progressive transmission.

4.6.3 Quadtree Oriented Methods

While some of the differences between the BC quadtree and other progressive transmission methods using quadtrees are obvious, it may be necessary to point out some less obvious differences.

4.6.3.1 The BC Quadtree: Quadtree or Pyramid

In the field of progressive transmission, the terms *quadtree* and *pyramid* are used interchangeably in most cases. However, it may be interesting to analyze whether the BC quadtree is indeed a quadtree, or rather a pyramid.

According to Samet [Sam90a, Preface], the quadtree is a *variable resolution* representation, whereas the pyramid is a *multiple resolution* representation. In this statement, Samet of course refers to spatial resolution. The BC quadtree obviously incorporates aspects of both variable spatial resolution and multiple spatial resolution. In its variable resolution aspect, it adapts the spatial resolution, for a given gray scale resolution, to the local properties of the image. On the other hand, multiple spatial resolution is present for different levels of gray scale resolution. Indeed, the BC quadtree goes beyond the simple multiple spatial resolution that allows to arrange pyramids (in the sense of image descriptions) as pyramids (in the geometric sense), as it incorporates a large number of different ways to increase resolution.

In analogy to Samet, introducing the concepts of variable gray scale resolution and multiple gray scale resolution is useful to show the differences between the BC quadtree and the G-quadtree [Kni86], [Mao87]. The G-quadtree is clearly a variable gray scale resolution representation. It provides different resolutions with different members of a G-quadtree family, or variable gray scale resolution in one and the same tree in the case of the nonuniform G-quadtree [Mao90a]. On the other hand, the BC quadtree incorporates all different gray scale resolutions in one and the same data structure. It is therefore a multiple gray scale resolution representation.

4.6.3.2 Knowlton's Progressive Transmission of Binary Images

It is interesting to compare the method proposed by Knowlton to transmit binary images progressively (introduced in Subsection 1.2.3.5) with the progressive transmission achieved when using the BC quadtree. It turns out that the reproduction achieved when applying the new method to an image with b=1 bits per pixel is the same as with Knowlton's method, with some small exceptions.

The differences are that Knowlton uses bintrees and starts with cells of 2·3 pixels at the second lowest level, mainly to equalize the aspect ratios at different levels of the bintree. He then adds the number of black (or white) pixels in such a cell as an intermediate level of refinement, using this to select between several gray levels. He also proposes an elaborate variant that increases compression by using prediction. It seems that the bintree is better suited for such a prediction scheme, as the number of cases that have to be treated is smaller.

The equality of the two methods in the binary case suggests that progressive transmission based on the BC quadtree is also highly efficient when transmitting images that contain mainly binary information, like text and line drawings. This is confirmed in Section 6.3.

4.6.3.3 Dreizen's Method

Dreizen's method [Dre87], described in Section 1.2.3.1, is similar in complexity to the present one, and incorporates some aspects of combined increase of spatial and gray scale resolution by first subdividing areas of the image where there are strong changes of gray values. Therefore, in the experiments of Chapter 6, the new method is mainly compared with Dreizen's method.

Dreizen uses the difference between the minimum and the maximum pixel value in a subquadrant to decide whether to split or not. This is obviously a more flexible criterion than bitwise condensation, as it is not influenced by the arbitrary splitting of the gray scale hierarchy. On the other hand, the cost of calculating the minima and maxima is slightly higher than the cost of the bitwise logical operations used when constructing the BC quadtree.

That Dreizen's method is less efficient than progressive transmission based on the BC quadtree (see Chapter 6) is due to the fact that he uses full quantization in all cases. This is partially compensated by using prediction, but prediction is not very efficient in the first stages where transmission is concentrated on the areas with large gray scale variances.

Although it is possible in the framework of his method, Dreizen never considers interleaving transmission of small squares with strongly differing gray values and larger squares with slightly differing gray values. This might somewhat increase the performance of this method.

4.6.3.4 The Method of Endoh et al.

The method of Endoh et al. [End87] is the only previously existing progressive transmission method that allows the independent increase of spatial and gray scale resolution. Similar to Section 4.3, the image information is split into components, each with different spatial and gray scale resolution. However, these components just consist of the corresponding bits at appropriate subsampling locations, and so basically always have the same size. This method does not capture the structure of the image in any way.

To achieve compression, the above method relies heavily on prediction and entropy coding using run length coding. This greatly increases complexity, but leads to somewhat better compression than the BC quadtree because prediction is fully utilized. A simpler version of their approach, not using prediction, will not produce acceptable results.

Also, the method uses interpolation to smooth the image. This increases the computation load at the receiver, especially in the initial stages of transmission. It is difficult to imagine the method without interpolation. The subsampling scheme used (see Figure 1.5) alternatingly produces squares and diamonds on the screen, which will greatly disturb the user. Also, whereas many displays can easily paint a square, painting other polygons is supported by a smaller number of display architectures, and even if supported, it usually takes longer.

However, the most important difference between the method of Endoh et al. and the work presented in this thesis are the component sequences considered. They propose component sequences corresponding to those in Subsections 4.3.2 (increasing gray scale resolution) and in Subsection 4.3.3 (increasing spatial resolution)[1]. For special applications like printout on a high-resolution printer, they also propose to additionally increase the spatial resolution after having increased the gray scale resolution for output to a display.

Their component sequences can therefore by characterized by a repeated selection of an increase in either gray scale or spatial resolution. If the gray scale resolution is increased, then this is done for all spatial levels already transmitted, and vice versa for spatial resolution. When the components are arranged as in Table 4.2, the area of the components transmitted therefore always forms a rectangle in the upper left corner.

---

[1] The result of increasing spatial resolution in the case of Endoh et al. is similar to that for Knowlton [Kno80] and cannot directly be compared with Figure 4.7.

On the other side, as explained in Subsection 4.3.4, the sequence of components when using the BC quadtree is less limited. Consequently, in the case of Endoh et al. gray scale resolution and spatial resolution can be increased *independently*, whereas in the case of the BC quadtree, gray scale resolution and spatial resolution are increased *concurrently*.

5

Optimization

The basic principles of the BC quadtree, GDF, and HSRQ have been introduced in the last chapter. In this chapter, various ways to optimize image compression and progressive transmission based on these principles are discussed. Section 5.1 deals with the selection of representative gray levels. This then allows to find rules to determine the best component sequence in Section 5.2. To achieve better overall compression, a deterministic algorithm is proposed in Section 5.3, and compression using arithmetic coding is treated in Section 5.4.

5.1 Gray Level Reproduction

With other image coding and progressive transmission methods, reproduction values are usually specified exactly for each stage and each pixel. Even when the gray scale resolution of an image is reduced, the remaining gray levels are usually reproduced at equal intervals. In the case of HSRQ, however, the gray levels are only restricted to intervals that are repeatedly subdivided. During the whole transmission, intervals of various sizes coexist.

In this section several rules and methods are proposed to specify the exact gray level with which each gray scale interval is represented (see also [Dür91]). The proposed methods differ in complexity and in the quality of their results, and their choice is basically independent of the sender; this offers flexibility for the receiver in accordance with Requirement 3.7.

5.1.1 Black to White Reproduction

Each bit selection symbol of GDF specifies an additional bit for a square of a certain size. It is therefore possible to reproduce the image by changing only one bit of each pixel of a square at a time. This is especially advantageous if the frame buffer is organized so that equivalent bits of neighboring pixels can be accessed

together. In this case, the access rate to the frame buffer can be reduced by a factor of b for an image with b bits per pixel.

As the lower bits of a pixel are not changed for a long time, they have to be initialized carefully. One way of doing this is to initialize the image to black (00000000). Now when receiving a **0**, no change to the frame buffer is necessary. When a **1** is received, the corresponding bits are set, and the corresponding square appears lighter. As a result of this, the access rate to the frame buffer is again reduced by a factor of 2, as we can assume that the number of **0**s and **1**s is more or less the same. This reduction is possible for all frame buffer organizations.

Figure 5.1 shows the results of this method. Compared with the methods discussed in later subsections, the information content of the developing image is rather low, because half of the bit selection symbols do not lead to a change of the display. To estimate this effect, this method was included in the experiments described in Chapter 6.

Figure 5.1. Black to white reproduction
(bytes transmitted: 300, 600, 1200, 1800, 2400, 3600, 4800, 9600)

On the other hand, starting with a black screen has the advantage that the image intensity is increasing monotonously. This eliminates a slightly disturbing effect that can appear, in one form or the other, with most other methods: A large area of the image quickly increases and decreases in intensity. This appears like blinking if fast enough. If slower, viewers may get the impression that some gray levels are unnecessarily changed back to their original value, as it is difficult for a human observer to remember absolute gray values. The monotonously changing intensity is also advantageous when displaying image sequences (see Section 7.3). It is also possible to display an image starting with a white background, but this is visually less pleasing (see, however, the comment to Figure 6.5).

Reducing the number of accesses to the frame buffer is of importance especially in the early stage of the transmission, when for every few bits received,

a large square has to be painted. If the access to the frame buffer is a bottleneck only at this stage, it is possible to use the black to white reproduction method for the initial stage and another reproduction method for later stages. The combination of other methods is possible, too.

It has however to be pointed out that it is not necessary that the rate at which the largest squares can be written into the frame buffer is exactly as high as the rate at which the information to paint these squares is received. Most of the squares painted will be rather small, and so it is possible to queue some of the commands to paint large squares. The queue will be reduced as soon as information for smaller squares arrives. The delay in painting large squares may have no effect on the transmission time at all, as a certain amount of detail is needed in any case for the user to get an initial idea about a transmitted image.

Another possibility to reduce the frame buffer access in the initial stages is to first display a smaller copy of the image, as this has been proposed in [Hof86]. Displaying the image with half the final side length is possible as long as no component of the lowest level of the spatial hierarchy (the pixel level) is transmitted. However, as the most significant bit component on the pixel level may not affect very many pixels, it may be necessary to start building up the full size image earlier. The same argument applies to earlier stages.

5.1.2 Random Background Reproduction

Instead of starting with a black background, it is possible to start with a random background. In this case, both **0**s and **1**s are written to the frame buffer and affect the displayed image. The distribution for the original background has to be uniform on the whole gray scale; the uniform distribution is the only one whose basic shape is not affected when derandomizing leading bits.

The result of this method is shown in Figure 5.2. The random noise has the effect of blurring the sharp boundaries of the subdivision and the flat areas of large squares. This leads to a subjective quality improvement in the early stages. This can be explained by the fact that the human eye is better in detecting shapes from a noisy pattern than from an arbitrarily smoothed one. The effect of the random pattern can be compared to dithering. An investigation of different kinds of pseudorandom patterns may lead to interesting results.

This method of gray level reproduction is very well suited for frame buffers where corresponding bits of neighboring pixels are stored together. On the other hand, if this is not the case, the access rate to the frame buffer is increased as the lower bits of each pixel have to be conserved. Each pixel value therefore has to be read one by one, the correct bit set or cleared, and the pixel value written back to the frame buffer. This problem can be circumvented by using a precomputed

pseudorandom pattern, which does not have to have the size of the whole image, but can be used repeatedly. This also eliminates the time necessary to calculate the initial background.

Figure 5.2. Random background reproduction
(bytes transmitted: 300, 600, 1200, 1800, 2400, 3600, 4800, 9600)

The random background method is also not suited for low quality displays such as TV monitors. The random pattern introduces high frequencies which lead to flickering that can badly disturb the viewer. This method was therefore not used in the experiments described in Chapter 6.

5.1.3 Center Value Reproduction

The traditional method used when transmitting an image bit plane by bit plane or with a reduced gray scale resolution is to display the gray value in the center of the corresponding gray scale interval. For example, if the first three bits are 011, then for an eight bit frame buffer the value chosen is 01101111 or 01110000. The image is thus initialized to 01111111 or 10000000. This is the method that has been used for the figures in Chapter 4 (see especially Figure 4.8).

Center value reproduction has the advantage that it is easy to understand and implement and leads to fairly good results. If the frame buffer is addressable by bit plane, only one or two out of the b bits per pixel have to be changed for each arriving bit selection symbol. For example, if the pixel value is 01101111 as in the last paragraph, and a **0** is received, the pixel value is changed to 01100111; if a **1** is received, the new pixel value is 01110111 (changed bits underlined).

5.1.4 Average Reproduction

5.1.4.1 The Problem

The methods for the reproduction of gray levels discussed in the previous subsections are simple and produce acceptable results. However, the fact that with

HSRQ, different areas of the developing image are quantized with different numbers of gray levels leads to some problems. A typical one-dimensional example is shown in Figures 5.3 and 5.4.

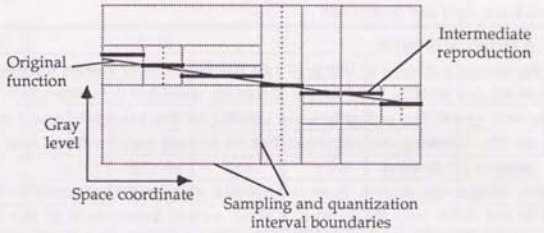

Figure 5.3. Smallest spatial intervals quantized with one level

In Figure 5.3, a slightly decreasing linear function is finely quantized in all spatial intervals but the smallest ones. On the other hand, only one quantization level is used for the smallest intervals. This produces a fairly nice approximation of the original function.

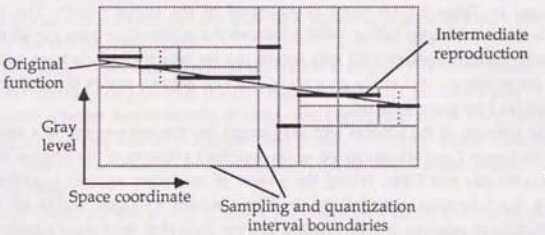

Figure 5.4. Smallest spatial intervals quantized with two levels

What happens if the quantization for the small intervals is increased to two levels is shown in Figure 5.4. This quantization introduces some false high frequency components into the image reproduction. In two dimensions the effect occurs mainly between light and dark areas and leads to false contours. On low quality displays, these high frequencies introduce flickering.

For Figures 5.3 and 5.4, the center of each gray scale interval has been chosen as the representative gray level according to Subsection 5.1.3. Similar effects appear with the reproduction methods of Subsection 5.1.1 and 5.1.2, although there is no fairly nice intermediate stage as in Figure 5.3. These effects are especially visible in Figure 5.1, top right and bottom left.

5.1.4.2 A Statistical Example

There are several solutions to this problem. The theoretically correct solution, if the mean square error is used as the distortion measure, is to reproduce the squares of a given size and gray scale interval by the average of the values quantized. The following analysis shows that the average actually differs from the center value of the interval.

Most images are smooth in most locations, and on average, neighboring pixels do not differ very much. So in a small square, for example of size 2·2, pixels will have gray values very near to the center of the whole gray scale if neither all of its pixels have gray levels in the upper nor in the lower half of the gray scale. Thus the average of the pixels with gray values in the upper half of the gray scale that lie in a two by two square that also contains pixels in the lower half of the gray scale will be very close to the center value of the whole gray scale, and not near the center value of the corresponding interval.

To explain this phenomenon using an actual example, these averages are calculated in Table 5.1 for some components of the image "Girl". The first column in the left (right) half of Table 5.1 shows the percentages (perc.) of all (not necessary aligned) squares with side length *size* for which all pixels have values in the lower (upper) half of the gray scale. The next column shows the gray level averages (av.) for these squares.

The averages of the squares with side length *size* that are not part of a square with side length 2·*size* are calculated as the weighted averages of the squares with side lengths *size* and 2·*size*, taking the weight of the larger squares negatively. For the exact formula, see (5.6) below. The resulting averages usable for the reproduction of squares of side length *size* when receiving their most significant bit (rep. av.), are shown in the last column of both parts of Table 5.1.

Table 5.1 clearly shows that the averages for squares of different sizes vary greatly, and that the smaller the squares are the closer these averages move to the center of the whole gray scale. On the other hand, for large squares the averages tend to be even farther away from the center of the whole gray scale than the centers of the corresponding subintervals. In fact, a large square that has an uniform most significant bit will not contain many pixels near the center of the gray scale, because such pixels have a high probability of being accompanied by a pixel with different most significant bit in their neighborhood.

A similar relation between the size of the leafs in a quadtree and their histogram was observed by Wu et al. [Wu82] and used for image segmentation.

size	≤127			≥128		
	perc.	av.	rep. av.	perc.	av.	rep. av.
1	88.347	62.5282	111.57	11.652	155.922	138.56
2	84.828	60.4939	93.18	8.370	162.730	150.41
4	79.091	58.1230	78.16	4.821	171.800	164.86
8	69.021	55.1996	64.62	1.708	184.453	182.40
16	52.388	52.2086	56.02	0.166	203.539	-
32	27.359	48.7213	-	-	-	-

Table 5.1. Reproduction averages for image "Girl"

5.1.4.3 Reproduction Averages for Traces

In Chapter 4, we introduced the concept of traces to analyze the BC quadtree and HSRQ. Traces are also a tool to describe and estimate reproduction averages, for two reasons: First, the average depends on the location in the spatial and in the gray scale hierarchy, information which is contained in a trace. Second, the receiver can easily reconstruct the trace of a square or a pixel and then use it to decide the reproduction value.

We will now present some rules that relate averages of different traces. With this, we intend to demonstrate the advantages of the trace formalism and to give the reader a better understanding of traces and averages. Some of the rules apply to single images as well as to random field models of images, whereas others are strictly true only for random fields that exhibit the necessary degree of stationarity and ergodicity.

We denote the average of all pixels with trace α by $av(\alpha)$, and the frequency of occurrence of pixels with trace α by $oc(\alpha)$. We will also use $av(A)$ and $oc(A)$ for the set of traces A with the obvious meaning. Laws formulated on sets of traces apply to single traces, too.

First, averages of mutually exclusive sets of traces can be added by weighting them with their occurrences:

$$\text{Iff } oc(A \cup B) = oc(A) + oc(B), \text{ then}$$
$$oc(A \cup B) \cdot av(A \cup B) = oc(A) \cdot av(A) + oc(B) \cdot av(B). \qquad (5.1)$$

Note that in the previous formula, it is not possible to use $A \cap B = \emptyset$ as a condition, as A could contain subtraces of traces in B or vice versa. A special case of (5.1) is the average of a common subtrace. As

$$oc(\alpha) = oc(\alpha 0) + oc(\alpha \diamond) = oc(\alpha 0) + oc(\alpha 0) + oc(\alpha 1), \tag{5.2}$$

it follows that

$$\begin{aligned} oc(\alpha) \cdot av(\alpha) &= oc(\alpha 0) \cdot av(\alpha 0) + oc(\alpha \diamond) \cdot av(\alpha \diamond) \\ &= oc(\alpha 0) \cdot av(\alpha 0) + oc(\alpha 0) \cdot av(\alpha 0) + oc(\alpha 1) \cdot av(\alpha 1). \end{aligned} \tag{5.3}$$

The law (5.1) was also applied in the previous subsection. As $size = 2^{(r-s)}$, the first column on the left side contains $oc([\alpha \mid \alpha = (^{*<s>}0)])$, where $(^{*<s>}$ denotes the string of up to s (including 0) copies of (. The second column contains the corresponding average $av([\alpha \mid \alpha = (^{*<s>}0)])$. The third column contains the average for a single trace, namely $av((^{<s>}0)$, where $(^{<s>}$ denotes exactly s repetitions of (. Obviously,

$$oc([\alpha \mid \alpha = (^{*<s>}0)]) = oc([\alpha \mid \alpha = (^{*<s-1>}0)]) + oc((^{<s>}0), \tag{5.4}$$

and so

$$\begin{aligned} &oc([\alpha \mid \alpha = (^{*<s>}0)]) \cdot av([\alpha \mid \alpha = (^{*<s>}0)]) \\ &\quad = oc([\alpha \mid \alpha = (^{*<s-1>}0)]) \cdot av([\alpha \mid \alpha = (^{*<s-1>}0)]) \\ &\qquad + oc((^{<s>}0) \cdot av((^{<s>}0), \end{aligned} \tag{5.5}$$

and $av((^{<s>}0)$ can thus be calculated as follows:

$$\begin{aligned} av((^{<s>}0) = [&oc([\alpha \mid \alpha = (^{*<s>}0)]) \cdot av([\alpha \mid \alpha = (^{*<s>}0)]) \\ &- oc([\alpha \mid \alpha = (^{*<s-1>}0)]) \cdot av([\alpha \mid \alpha = (^{*<s-1>}0)])] \\ &/ [oc([\alpha \mid \alpha = (^{*<s>}0)) - oc([\alpha \mid \alpha = (^{*<s-1>}0)])]. \end{aligned} \tag{5.6}$$

Exchanging **1** for **0** in (5.5) and (5.6) gives the corresponding formulas for the right part of Table 5.1.

For some cases, exact averages are known. Especially, if all bit selection symbols are known, the average, in binary notation, is the gray scale value of the corresponding bit selection subtrace:

$$\text{If } |bi(\alpha)| = b, \text{ then } av(\alpha) = bi(\alpha). \tag{5.7}$$

Here, $|\alpha|$ denotes the length of trace α. On the other side, we note that $av(\varepsilon)$ is the average of the whole image, where ε is the empty trace. Defining $\lceil \alpha \rceil$ and $\lfloor \alpha \rfloor$ as the highest and lowest full bit selection trace that contains the bit selection subtrace of α, i.e.

$$\lfloor \alpha \rfloor = bi(\alpha)0^{<b-|bi(\alpha)|>} \quad \text{and} \tag{5.8}$$

$$\lceil \alpha \rceil = bi(\alpha)1^{<b-|bi(\alpha)|>}, \tag{5.9}$$

the average of a given string can obviously be limited by

$$\lfloor \alpha \rfloor \le av(\alpha) \le \lceil \alpha \rceil. \tag{5.10}$$

As $\lceil \alpha 0 \rceil < \lfloor \alpha 1 \rfloor$, we can deduce

$$av(\alpha 0) < av(\alpha 1). \tag{5.11}$$

For well behaved images and image models, this can be extended to

$$av(\alpha 0) < av(\alpha \diamond) < av(\alpha 1), \tag{5.12}$$

and further, as we have already noted in the previous subsection, to

$$av(\alpha 0) < av(\alpha(0)) < av(\alpha \diamond) < av(\alpha(1)) < av(\alpha 1). \tag{5.13}$$

The most important aspect of the last two formulas is perhaps that they clearly show that transmitting a (is not a useless overhead of GDF, as it seems at first. Is fact, a (contains information that can be used to obtain a faster approximation to the original image.

5.1.4.4 Calculation and Transmission of Averages

The previous subsection discussed various relations between averages of different traces and sets of traces. In this subsection, we discuss ways to calculate and transmit the actual averages. The trace tree introduced in Subsection 4.5.2 allows to calculate averages for all traces and subtraces. However, the number of possible traces in an image is high, and therefore it may be advisable to find trace sets that have similar averages and similar structure. The following discussion applies both to individual traces and to sets of traces.

The first possibility to obtain averages for the reproduction of the image at the receiver is to assume fixed averages for all traces, based on the relations given in the previous subsection. This is especially useful for applications where all images are statistically similar or can be divided into a certain number of types. It is also possible that the receiver updates these averages after each image received; a receiver that disposes of the necessary computing power may thus obtain a very accurate statistical model of trace averages independent of the sender.

A second possibility to calculate trace averages is to calculate some characteristic values of the image, such as average, standard deviation, autocorrelations, etc., at the sender. These values are transmitted to the receiver, which uses them to calculate the average for each trace.

An example is to assume a Gauss Markov model for the image, as this is frequently done (The most famous example is perhaps [Ahm74]). As this model determines the probabilities of all pixel combinations, all the averages can theoretically be calculated. However, if this requires numerical integration or Monte Carlo simulation, this approach may not be very practical.

Also, the complexity of the model that allow a reasonable approximation of the actual averages has to be taken into account. If the number of characteristic values needed to describe the trace averages to a certain degree of accuracy is low, these values can just be sent to every receiver and be ignored if another reproduction scheme is used. Otherwise, the independence of the sender from the receiver (as postulated in Requirement 3.7) is lost.

A third method to obtain trace averages is to try to deduce them from the already received image components. For example, a main distinction can be made between images with flat areas and sharp edges, and images with a generally smooth change of intensity. The fine quantization of large and middle-sized areas will indicate to which category the transmitted image belongs, and this information can then be used to predict the reproduction values for the small squares. Also, as explained in Section 5.2, the optimal component sequence differs according to the type of the image, so that suitable trace averages may be deduced from this sequence.

A last possibility is to calculate the average for each trace or trace set at the sender and transmit it to the receiver. A variant of this method was used in the experiments in Chapter 6. This was done to estimate the effect of average value reproduction on the quality of the developing image while using a very simple model. It can be expected that with the other methods mentioned in this subsection, similar results can be obtained with less or no calculations at the sender.

The traces were grouped according to their number of parentheses $s = |pa(\alpha)|$ and their bit selection subtrace $bi(\alpha)$. In this way, for each component as defined in Section 4.3, at most $(b\text{-}c)\cdot 2^c$ bits, where $c=|bi(\alpha)|$, have to be transmitted. Here, the first term, $b\text{-}c$, indicates the number of remaining bits and the second term, 2^c, the number of different bit selection subtraces for this component. This added about 0.3% to the total transmission rate.

Figure 5.5. Reproduction using trace averages
(bytes transmitted: 300, 600, 1200, 1800, 2400, 3600, 4800, 9600)

The result of this scheme, for our example image, is shown in Figure 5.5. Figure 5.5 shows that reproduction by average values leads to a smoother and better looking image when compared to the previously presented methods, may be with exception of the initial stages of the random background method. Because

different gray values are used for different subquadrant sizes, the block boundaries are eliminated early.

The reproduction scheme used for Figure 5.5 has the advantage that it works with the same amount of memory for the BC-quadtree as the decoding algorithm of Section 4.3. However, it is far from being perfect. Transmitting averages for the least significant bit is really not necessary, as using the "wrong" reproduction value in this case has almost no effect on the visual appearance of the image.

On the other hand, significant deviations from the averages for the individual traces occur in the case of $s=r$ and $c=2$. Both traces of the form (((((((01 (we assume $r=8$) and traces like (((((((0(1, (((((((0((1, etc., are assigned the same average. Actually, the average for the former lies close to 01111111, whereas the averages for the later lie close to 01000000. In this case, a finer grouping of the traces seems necessary (see also the comment to Figure 6.6).

5.1.5 Global and Predictive Reproduction

The four reproduction methods discussed in the four previous subsections only change the pixel values in the square for which an additional symbol is received, based only on information about the current square. This leads to a simple and fast implementation and fairly good results, but such a limitation is not necessary if additional computing power is available.

Some improvement of reproduction quality can be expected if the information of neighboring pixels is considered. This can be done both globally and locally. In the first case, the image is reprocessed completely by the decoder at determined points of time, for example after the transmission of each component. An example of this is the use of cubic splines or cubic convolution for interpolation by Sanz et al. [San84]

In the second case, the pixels of the current square are changed taking into account the information available for neighboring squares and pixels, predicting the exact pixel values in the current square. The neighboring squares or pixels may then in turn be adjusted based on the information just received for the current square.

One goal of this method is to improve image quality by smoothing the image to eliminate block boundaries. On the other hand, if sophisticated enough, it can also detect and strengthen edges that are not parallel to the quadtree grid. The actual techniques used can range from simple decision rules to complex filtering and smoothing procedures. A simple way to ameliorate the problem described in Subsection 5.1.4.1, for example, is to restrict the reproduction values for the small spatial intervals by the gray values of the neighboring larger intervals.

On the other hand, it is not clear to what extent traditional filtering techniques can be applied to the nonuniform sampling and quantization in the case of HSRQ. In the case of regular subsampling, optimal reconstruction filters, and optimal sampling kernels for given reconstruction methods, are known [Chi89]. The problem of signal reconstruction from irregularly spaced samples has been studied [CheD87], but mainly to find the best of several sampling patterns.

In our case, sampling and quantization are not only nonuniform, but also specify intervals and not points. The reconstructed function will have to lie completely inside the intervals. This is a binary valued condition that is much more difficult to work with than for example the well used mean square error. Numerical methods that combine a variety of constraints, such as [Mal89], or morphological filters [Gia88], may be a solution to this problem.

A good reconstruction and interpolation is also desirable to achieve scalability as demanded by Requirement 3.8. The best reproduction of the original image at a smaller scale is obviously obtained easier from a good reproduction at the full scale than from a bad one.

5.2 Optimizing Component Sequence

Section 4.3 showed how changing the component sequence influences the appearance of the image reconstruction process and thus the efficiency of progressive transmission and the compression ratio for lossy compression. A more detailed discussion of the problem of determining a good (or even optimal) component sequence has been delayed up to this section because the optimal component sequence depends to some degree on the chosen reproduction method. In addition, the third subsection of this section discusses alternative ways to split GDF into components.

5.2.1 The Optimal Component Sequence

Assume that we are given a distortion function $d(\cdot,\cdot)$, a reproduction method, and the division of the image into components that will be transmitted one after the other. How can an optimal transmission sequence be defined, and does such a sequence indeed exist?

The optimal component sequence may be defined as follows: If a component sequence is optimal, then it achieves distortion equal or lower to that achieved by any other component sequence at any time during transmission. If such a component sequence does exist, it should clearly be used for transmission.

On the other hand, it is easily possible that there is no such component sequence. In this case, to be able to define the optimal sequence, the simple

distortion function that compares the original with the reproduced image is not sufficient to decide which transmission sequence to choose. The distortion function has to be integrated in a suitable way over time.

There are several reasons that such an integration might be necessary. First, there are restrictions on the possible component sequences, as given in Section 4.3. As these restrictions require nothing more than to transmit large squares before small squares, and more significant bits before less significant bits, it could be assumed that they do not restrict the optimal component sequence. However, this may not be true. The number of bits for the same number of squares is much lower on the pixel level, which makes these components more efficient. Also, there are components with many parentheses and others consisting almost only of bit selection symbols. Compared with bit selection symbols, parentheses do not affect the image directly, and so components with many parentheses perform somewhat worse.

Second, it is possible that some components worsen image quality rather than improve it. This depends heavily on the reproduction method chosen (see Figures 5.3 and 5.4). A good reproduction method is thus an important condition for an optimal component sequence to exist. On the other hand, the optimal component sequence depends on the reproduction method chosen.

5.2.2 Component Sequence Heuristics

Even if we are given a suitable distortion measure and a way to integrate it over time, it is not easy to find the optimal component sequence, as the number of possible sequences is extremely high. In the case of the image "Girl", for example, the number of allowed sequences is as high as $54 \cdot 10^{12}$.

A considerable simplification of the problem is obtained by assuming that the distortion measure is additive. This means that the change of distortion due to transmitting a given component is independent of exactly what other components have already been transmitted. This excludes almost all distortion measures which are not single letter distortion measures. On the other side, well used distortion measures like the mean square error are additive. Due to the restrictions on the component sequence, exchangeable components affect different pixels.

For each component, the quality improvement $\Delta d(\cdot,\cdot)$, the necessary number of bits $\Delta bits$, and thus the efficiency of each component, expressed by $\Delta d(\cdot,\cdot)/\Delta bits$, can be calculated. Then the components can be ordered by decreasing efficiency. If this order satisfies the restrictions on the component sequence given in Section 4.3, we have found the optimal component sequence.

This will not generally be the case, and so heuristics have to be used. One simple heuristic is to postpone "bad" components, i.e. components with negative or very low contribution, as long as possible. This leads to good initial results and is thus appropriate in applications like person recognition, where rarely much more than the initial stages of a developing image will be used. On the other hand, a sudden deterioration of quality in a late stage of image reproduction may destroy the confidence of the viewer into the transmission scheme used. This may do much more harm than can be measured numerically.

Other heuristics include the combination of a bad component with the component it delays, treating these two components as one, or transmitting bad components as soon as possible, favorizing a regularly improving reproduction over a good initial result. An important point is also that heuristics should concentrate on the larger components.

Such a heuristic, but still numerical approach, which tries to find the best sequence for a given image (or image model) by calculation of actual distortion improvements, can be complemented or replaced by a more rule-based approach. For fixed applications, it may be sufficient to use the same component sequence for all images. For all these images the reproduction quality for the chosen component sequence will differ only slightly from the optimal sequence. This has the additional advantage that the component sequence need not be transmitted. However, the number of bits needed to specify the component sequence is bounded by $(r+1)\cdot b\lceil \log b \rceil$ and thus practically negligible.

Large image databases will have stored some textual information for each image, and the component sequence may be selected using this information, as the optimal component sequence will largely depend on the type of the image.

The component sequence may also be adapted to the user's queries or the user itself. For form- or shape-oriented queries spatial resolution is more important than gray scale resolution, and so the component sequence may be adapted to such a query. On the other hand, if the user remembers an image mainly from the general distribution of light and shadow, gray scale resolution should be increased first.

Also, as an example, the same X-ray image may be viewed once by an orthopedist interested in the sharp outlines of the skeleton, and another time by an oncologist looking for the diffuse shades of a tumor. An intelligent transmission system can in such a case automatically adapt the component sequence to the user.

5.2.3 Other Ways to Define Components

Until now, division of the image information into components was assumed to be based on the level in the gray scale and in the spatial hierarchy, as defined in Section 4.3. However, there are several ways to increase the number of components and thus to achieve a smoother development of the image.

5.2.3.1 Components based on Traces

One way of increasing the number of components is to define components based on traces. This is directly based on the BC quadtree and HSRQ and thus not possible with other methods of progressive transmission. Starting with the empty trace, we transmit as a single component all those symbols that have the present trace as prefix. For example, when transmitting the component based on trace ((((0(1, all the symbols that expand this trace to either ((((0(10, ((((0(11, or ((((0(12 are transmitted.

This can lead to improvements in the following ways: First, the human visual system is more sensitive to contrast in the dark rather than in the light areas of an image [Mann74]. Thus, it is advantageous to transmit traces with low-valued bit selection subtraces somewhat earlier than those with high-valued bit selection substrings.

Second, as already explained in Chapter 3, the gray scale resolution necessary in areas with high frequency is not as high as in slowly changing areas. The basic way of defining components in Chapter 4 has been guided by this observation, but does not fully use it. The problem is that initially, areas of high activity are correctly sampled finer and quantized coarser than areas of low activity. As resolution is increased, additional quantization is necessary mainly in areas of low activity to eliminate Mach banding and similar effects, but actually, quantization is increased at the same pace in areas of high activity, where this is much less necessary.

To give a concrete example, a trace of (((((((011 suggests that we are in an area of high activity, so that additional quantization is not really necessary. On the other hand, a trace like ((((01(((10, although it has the same number of parentheses and the same bit selection symbols, belongs to an area of low activity and should therefore be quantized further with high priority.

It is not possible to deduce the amount of activity in a given area from a single trace with absolute certainty. To further improve the sequence of transmissions, neighboring traces can also be considered. This is similar to considering neighboring gray values when deciding on the reproduction value (see Subsection 5.1.5). However, this greatly increases the complexity of the algorithm.

The advantage of traces is that they provide a maximum of information about the current area in an efficient and seemingly local way.

5.2.3.2 Local Priorities

Letting the user choose parts of the image to be transmitted with higher priority (before the start of the transmission or based on the intermediate image) has been proposed first in [Kno80]. This can easily be implemented with most methods, the preferred region usually being a rectangle.

With the new method, smoother solutions are possible. First a priority function for each pixel or small square is defined. Then the priority for internal nodes of the BC quadtree are calculated as the maximum of its children. Components are then formed based on spatial resolution, gray scale resolution, and priority. Compared with defining priorities for large squares in a fixed grid [San84], this has the advantage that the reproduction can be adapted to the image and the human field of vision with a very small transmission overhead.

Note that in contrast to [San84], there is no need to let the sender calculate the importance of different areas of the image, because HSRQ automatically concentrates on those areas with high activity.

5.2.3.3 Bintrees

Another way to increase the number of components is to use a bintree instead of a quadtree as the spatial hierarchy. The bintree alternately leads to rectangles and squares, instead of displaying squares all the time, which is visually not so pleasing. Most authors therefore, even when actually using bintrees, avoided the display of the rectangular stages [Hil83], [San84], or treated the lowest level differently to obtain rectangles with more or less equal aspect ratios in all stages [Kno80].

In this respect, the unequal development of the image and the early elimination of visible block boundaries when using reproduction averages may hide the visual disadvantage of the bintree. On the other hand, with the simpler reproduction methods, neighboring squares are frequently quantized to exactly the same gray scale value, a fact which also hides rectangles, whereas this is not the case for methods based on averaging or subsampling. In general, it seems that the combination of the arbitrary subdivisions in the gray scale and the spatial hierarchy do not increase, but decrease blocking effects.

In addition, it may be possible to split some of the squares in horizontal direction and others in vertical direction. Whereas this distracts the viewer even more than the usual bintree if the image is developed regularly, it probably makes the use of the bintree completely invisible in the case of HSRQ.

5.2.3.4 Exact Transmission Sequence inside a Component

There are basically three ways of transmitting the symbols of each component. The first is to exactly follow the Morton sequence defined by the quadtree. The second is to scan the image at the present spatial resolution, line by line. The third is to use a pseudorandom sequence.

With regard to visual quality, the last solution is probably the best, as it leads to a smoother, less localized increase of resolution. The viewer is not seduced to concentrate on the location where the image is improving currently, and so will view the image as a whole, which leads to better recognition. The second solution comes next. At least this restricts the head movement of the user to up and down movements, and so is highly predictable. It was used in the experiments described in Chapter 6.

5.2.3.5 Algorithmic Aspects

The algorithms for progressive transmission in Chapter 4 basically scan the appropriate level of the explicit quadtree. If the number of components increases, scanning a whole level of the quadtree for each component becomes highly inefficient. In this case, the solution is to connect all the nodes in the same component into a linked list.

As the pixel level is treated separately in any case, an index size of two bytes is sufficient for images up to 512·512 pixels. Although this increases the necessary amount of memory compared to the algorithms of Subsection 4.3.5, the additional memory needed is still smaller than the original image. The trace tree can be used to store the root of each list. This method also reduces processing time. It is no more necessary to decide whether a symbol should be sent for the present node or not, it only necessary to decide which symbol to send.

5.3 Optimizing the Gray Scale Hierarchy

5.3.1 Basics

In most parts of this thesis, the quadtree is used as spatial hierarchy. However, HSRQ and GDF can easily be adapted to trees with outdegrees other than four. Of particular interest for image processing are bintrees [Kno80] and octrees [Sam88a,b]. In general, any tree structure is possible as long as the encoder and the decoder agree on it. The same applies to the gray scale hierarchy. In this section, we concentrate on increasing compression by optimizing the gray scale hierarchy.

We present a dynamic programming algorithm that, with respect to a given image and a given binary coding of the symbols of GDF, optimizes the gray scale

hierarchy. This algorithm has previously been published as [Dür89c]. It calculates, in time $O(2^{2r}+g^3)$, the gray scale hierarchy that leads to the highest overall compression. 2^{2r} is the number of pixels in the image, and $g=2^b$ is the number of gray levels. The example image we will use, and its unoptimized gray scale hierarchy, are shown in Figure 5.6.

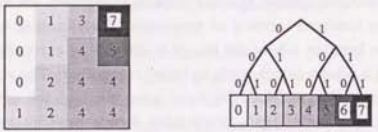

Figure 5.6. Example image for optimization
(left: image; right: gray scale hierarchy)

At first, it may seem difficult to find an efficient algorithm for the problem at hand because of the complicated relationships between the spatial and the gray scale hierarchy. The key idea of the algorithm is to consider all possible gray scale intervals. First, the quadtree is traversed. For each node, the gray scale interval it covers is calculated, and the number of nodes of each interval is counted. Next, nonexistent gray levels are eliminated and the number of independent trees for each interval is calculated. The actual optimization then decides the best division point for each interval working bottom-up.

In the next two subsections, the formal basis of the algorithm is presented, along with the actual values for the example image of Figure 5.6. The following three subsections then discuss binary coding, display, and related work. The reader should easily be able to understand the example and to implement the algorithm from the description given here.

5.3.2 Interval Statistics

To formalize the algorithm, we first note that the number of parentheses is independent of the gray scale coding and therefore parentheses can be ignored in the optimization algorithm. Then we denote by L_i the number of leaves of the (full) quadtree with gray level i, and by N_i^j $(i \le j)$ the number of nodes whose children's smallest gray level is i and whose largest gray level is j. N can be represented as a triangular matrix.

Inspecting L, gray levels that are not used at all can be eliminated in time $O(g)$. As the number of actually used gray levels is bounded by g, we will continue to use g for simplicity.

Now we can calculate the number T_i^j of independent trees in the quadtree that contain leaves in the interval form i to j. Each leaf in such a tree can be considered originally as an independent tree, and each internal node connects four subtrees to one tree and thus reduces the number of trees by three. Therefore,

$$T_i^j = \sum_{k=i}^{j} L_k - 3 \cdot \sum_{k=i}^{j} \sum_{l=k}^{j} N_k^l. \tag{5.14}$$

T_i^j is also the number of root nodes of independent trees with the corresponding interval and thus the number of nodes with that interval whose parent does not fit in that interval.

Actually, L and N can be integrated from the start into N^* so that

$$N^{*i}_i = L_i - 3 \cdot N_i^i \qquad \text{and} \tag{5.15}$$

$$N^{*j}_i = -3 \cdot N_i^j \qquad (i<j) \qquad \text{and so} \tag{5.16}$$

$$T_i^j = \sum_{k=i}^{j} \sum_{l=k}^{j} N^{*l}_k. \tag{5.17}$$

N^* needs $O(g^2)$ storage and can be obtained in a simple first-depth traversal of the quadtree using $O(2^{2r})$ time and $O(r)$ storage. For each quadtree node, the corresponding N^{*j}_i is incremented by 1 for leaf nodes and decremented by 3 for internal nodes. These values can be adjusted for trees or nodes with outdegrees other than four. T can be calculated from N^* in time $O(g^2)$. New space is not needed as the values of T can overwrite those of N^*. Figure 5.7 shows the values of N^* and T for the example image of Figure 5.6, together with some values introduced in the next subsection. There is no pixel with gray value 6, and so this gray value has been eliminated.

5.3.3 Finding the Optimum

The number of symbols (excluding parentheses) needed to code all trees in a given interval, denoted by S_i^j, can now be expressed as follows: For an interval with only one gray level, there is no information necessary to distinguish this gray level from others, and so

$$S_i^i = 0. \tag{5.18}$$

For larger intervals, S_i^j depends on the division point D_i^j ($i \le D_i^j < j$) of the interval, defined as the top gray value of the lower subinterval. Setting $k = D_i^j$, S_i^j can be calculated as

$$S_i^j = S_i^k + S_{k+1}^j + T_i^k + T_{k+1}^j \qquad (5.19)$$

where the first two terms represent the number of symbols to code all the trees of the two subintervals independently, and the later two terms correspond to the number of symbols, one for each independent tree of the subintervals, to code the subinterval the tree belongs to.

i \ j	0	1	2	3	4	5	7
0	3 3 - 3	-3 3 0 9	-3 2 1 13	0 3 2 17	0 5 2* 24	0 6 3 29	0 1 2 (25)
1		3 3 - 13	0 5 1 10	0 6 1 15	0 8 1* 24	0 9 1* 30	0 7 1 28
2			2 2 - 2	0 3 2 6	0 5 2* 13	0 6 3 18	0 4 2 18
3				1 1 - 1	0 3 3 6	0 4 3* 13	-3 2 4 12
4		N* T D R			2 2 - 2	0 3 4 6	0 4 4 10
5						1 1 - 1	0 2 5 4
7							1 1 - 1

Figure 5.7. Values of N^*, T, D, and R in optimization algorithm

Optimization is carried out by selecting D_i^j so that S_i^j is minimized. The optimum values of S_i^j are calculated for small intervals first, so that they are available when deciding the splitting values of larger intervals. The number of symbols needed for the whole tree is obviously S_0^{g-1}. The optimal gray scale hierarchy can be constructed from a subset of D, starting with D_0^{g-1}. The number of additions in (5.19) can be reduced by introducing

$$R_i^j = S_i^j + T_i^j \qquad \text{so that} \qquad (5.20)$$

$$S_i^j = R_i^k + R_{k+1}^j. \qquad (5.21)$$

Again, the R values can overwrite the T values, but separate space is needed for D. The time for this step is dominated by the number of the evaluations of (5.21), which is of the form $\sum_{k=1}^{g} k \cdot (g-k)$ and thus $O(g^3)$, so that the whole algorithm optimizing the gray scale hierarchy uses time $O(2^{2r}+g^3)$. D and R are shown in Figure 5.7. A star indicates that there are other division values that lead to the same optimal coding for the interval. The value in parentheses is S_0^7, the number of symbols (without parentheses) needed for the optimal coding of the example image, and not R_0^7, which has no meaning. The optimized gray scale hierarchy is shown in Figure 5.8.

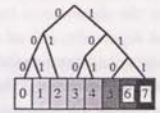

Figure 5.8. Optimal gray scale hierarchy for image of Figure 5.6

Below are the GDF based on conventional (upper row) and optimal (lower row) gray scale hierarchy. For this example, five symbols are saved. As the gray levels not used (g bits) and the structure of the optimized gray scale hierarchy ($<2g$ bits) have also to be transmitted, optimization does not pay off in this case. However, we will see in Section 6.2 that this is different for actual images.

GDF (original): (00(0101(011 111 100 101 0(00 10 01 10 100

GDF (optimized): (00(0101 1(00 11 01 10 0(00 1 011 101

5.3.4 Considering Binary Coding

Above, only the number of symbols was considered, regardless of the fact that a symbol, depending on its position, may be coded with one or two bits (see Subsection 4.2.2). The algorithm can be modified to take into account the number of bits per symbol as follows. First, imagine that separate statistics are collected for nodes at each level in the spatial hierarchy, and that (5.19) is extended, summing over the levels, each level weighted appropriately. The weighting factors can then be propagated to the first step similar to (5.15). The resulting increments for N^* are 1 for pixel nodes, -2 for the next higher level, and -6 for all other nodes. In this way, the later steps of the algorithm, and thus its complexity, remain unchanged.

The gray scale hierarchy is not limited to a binary tree. Of particular interest are ternary trees because they permit the four symbols **0**, **1**, **2**, and **(** to be coded with exactly two bits. Finding the optimal gray scale hierarchy in this case is still possible in time $O(g^3)$. First, D is calculated for the binary subdivision. When deciding on the division of an interval into three subintervals, extensive search is necessary only for one division point. The optimal second division point is identical to the binary division point for the remaining interval, as there are just two terms more in (5.19) or one more in (5.21).

Generalizing this, it is possible to find the optimal n-ary gray scale coding with $O(g^3 \cdot \log n)$ time and $O(g^2 \cdot \log n)$ space, and all gray scale codes up to n-ary with $O(g^3 \cdot n)$ time and $O(g^2 \cdot n)$ space. It is also possible to combine binary (for the lowest level of the spatial hierarchy) and ternary (for the upper levels) trees. The details of this are left to the reader.

A fractional number of bits for each symbol, as in the case of arithmetic coding, is also possible. However, the algorithm is limited in the sense that each binary coding has to be optimized separately, and all symbols except (have to be coded with the same number of bits. Another possibility is to introduce weights depending on other aspects of the image, like the difference to neighboring pixels, and so on. In this way, not the overall compression rate, but the increase of reproduction quality over the whole transmission may be optimized.

5.3.5 Display

When receiving image data compressed using the above optimization algorithm, basically all the reproduction methods described in Section 5.1 can be used. However, it is not anymore possible to just change one or two bits according to simple rules; table lookup has to be used to determine the appropriate reproduction values.

To be able to show the result of using the gray scale hierarchy optimization algorithm without having to build too complicated a table structure, the following approach was used: Each node in the gray level hierarchy is represented by exactly one gray level. For leaf nodes, the correct gray value has to be used in any case. The gray values for internal nodes are assigned so that no two internal nodes have the same gray value. A simple way to do this is to assign to each internal node the value of the highest leaf node in the lower subtree.

Figure 5.9. Progressive transmission with optimized gray scale hierarchy (bytes transmitted: 300, 600, 1200, 1800, 2400, 3600, 4800, 9600)

In this case only a single table is needed, with three entries for each gray value. The first two entries contain the gray value of the children if this gray value is used as an internal node. The last entry contains the level of the internal node in the gray scale hierarchy to avoid gray values used for leaf nodes being remapped to the children of the internal node with the same gray value. The result of this simplified approach, already very encouraging, is shown in Figure 5.9.

5.3.6 Related Work

Optimizing the gray level hierarchy can be compared to the normalized or optimal quadtree of [Li82] and [Gro83], an optimization of the spatial hierarchy. However, there only the *position* of the quadtree grid is optimized, which will not give substantial savings if there are no big rectangular blocks, whereas we are optimizing the *structure* of the gray scale hierarchy.

Trying to optimize the structure of the spatial hierarchy will lead to something like the k-d-tree [Ben75]. But for usual images, the resolution r and the number of bits per pixel b are of about the same order, and so the gray scale hierarchy ($g = 2^b$ leaves) is much smaller than the spatial hierarchy (2^{2r} leaves). Thus to improve compression, changes to the gray scale hierarchy are more promising than changes to the spatial hierarchy.

Some dynamic programming problems can be speeded up from $O(g^3)$ to $O(g^2)$ by utilizing the convexity inherent in the problem [Yao82]. Unfortunately, this is not possible in our case. The cost function for a given combination of subtrees in a given interval is not defined only by cost values associated to the trivial intervals, and the additional costs for true intervals can be both positive and negative.

The optimization algorithm presented above also can lead to a new approach to segmentation. The first bit of the optimized gray scale divides the image into two parts, a lighter and a darker one. This is done so that the dividing line between the two parts is short and the two parts themselves are as homogeneous as possible. This is exactly what a good segmentation algorithm is supposed to do.

Examples of this effect are shown in Figure 5.10. The originals in the top row have been processed with the algorithm developed in this section. The top eight bit planes of the resulting images are displayed in the bottom row using the standard gray scale. The rightmost image shows that the algorithm can have other effects than segmentation. Because in this image the white area dominates, the algorithm expands the gray scale for these gray levels to permit coding them with less bits.

Figure 5.10. Segmentation by optimizing the gray scale hierarchy (images: Girl, Couple, Moon, Fax data)

5.4 Arithmetic Coding

An alternative to the algorithm presented in the previous section to improve compression is arithmetic coding [Wit87]. Compared to the well known and used Huffman coding, arithmetic coding has attracted wide attention only recently, but is clearly superior to Huffman coding. The first reasons for its superiority is the use of "fractal bits" that allow coding nearer to the entropy limit for a given set of symbols and its probability distribution. The second reason is the easiness with which the probability distribution can be changed, allowing a clear separation of the source model and the coding procedure.

The algorithms for arithmetic coding have long been believed to be somewhat slow if the number of symbols is high. Although new algorithms alleviate this problem [Jonw88], [Mof90], the coding of binary symbols remains the best application for arithmetic coding. Compatible hardware and software implementations have been studied in great detail by Pennebaker, Mitchell, et al. [Pen88].

Arithmetic coding is therefore very well suited as an entropy coding method for GDF. To investigate the range of savings possible, we implemented arithmetic coding in the form proposed by Witten et al. [Wit87]. This implementation was also used instead of Huffman coding for some other compression algorithms in the experiments described in Chapter 6.

The model used was adapted to the trace grouping used to define trace averages as explained in the second half of Subsection 5.1.4.4. The first bit of the binary representation of each symbol, which distinguishes between parentheses and bit selection symbols, was modeled with decreasing cumulative frequencies. The total number of symbols in a given component is known from

the previously transmitted components. So if the number of parentheses is transmitted, the number of bit selection symbols can be calculated in the receiver. The frequency of each kind of symbol is then decreased by one for each such symbol received.

For the bits distinguishing between the two bit selection symbols, a cumulative frequency model was used, with independent frequencies for each internal node of the gray scale hierarchy. When reaching a total frequency of 4096 symbols, the individual frequencies were halved to allow adaption. Obviously, this value should be selected much lower.

With this model, some additional compression was achieved. Detailed results can be found in Section 6.2. The approach taken here was guided by simplicity considerations. An optimization of the frequency model, including short time adaption capabilities, can be expected to lead to much better results.

Still better compression can be achieved using multistate (predictive) models that are very efficient in coding binary images [Arp88]. The amount of compression possible with such models may be estimated from the results of [Kno80], [Wil84], and [End87], who combine hierarchical approaches with prediction. The frequency model should also be integrated with a reproduction model as discussed in Subsections 5.1.4 and 5.1.5.

Despite all the possibilities for optimization described in this and the previous sections, it should be kept in mind that the real advantage of the BC quadtree and HSRQ over other compression methods is that it provides relatively good performance, including progressiveness, at very low complexity. Before introducing complex optimization procedures, the advantages and costs have to be evaluated carefully.

6

Experiments

This chapter presents experiments that evaluate the new method of this thesis and its variants, and compare them with existing methods. In Section 6.1, each of the methods used in later sections is described shortly. Section 6.2 is devoted to lossless compression. Section 6.3 shows results of coding various frequently used images to allow visual comparison with previous methods. Section 6.4 measures the improvement of image quality over transmission time analytically. Section 6.5 presents results of a recognition experiment where the number of bits transmitted necessary to recognize a person's face was measured.

6.1 Compression Methods Used

In this section, the various compression methods and variants used in the different experiments, and the details of their implementation, are briefly explained. Please note that not all methods have been used in all experiments.

6.1.1 Previously Existing Methods

6.1.1.1 Predictive Compression

Predictive Huffman coding [Ros82,pp.181-188] is a simple and efficient lossless compression technique, a variant of DCPM (see Subsection 1.2.1.2), which however is not suited for progressive transmission. It has been implemented in its simplest form, comparing each pixel with its left neighbor [Ros82,p.182,(172)]. To entropy code the difference values, Huffman coding was replaced by arithmetic coding with a simple adaptive model, as used for character coding in [Wit87]. This has the advantage that only one pass over the data is necessary.

6.1.1.2 Lempel-Ziv Compression

Lempel-Ziv compression, in the variant known as LZW [Wel84], is a frequently used lossless compression method. It is well known for the fact that it compresses

most data better than or comparable to other methods. LZW was used as implemented in the UNIX™ command *compress*. Before using *compress*, the pixels were reordered to satisfy the requirements of [Lem86], although the Morton sequence [Sam84] was used, and not the Peano-Hilbert curve as proposed in [Lem86].

6.1.1.3 Bitwise DF with Gray Code

Kawaguchi et al. [Kaw83] transform pixels values to a Gray code and then use DF for each bit plane separately. Originally, they eliminated noisy regions in lower bit planes, which leads to approximate compression. However, this approximate compression is both difficult to implement and not applicable to progressive transmission. Therefore, here only the lossless version of this method is used.

6.1.1.4 Dreizen's Method

Of all methods described in the literature about progressive transmission, the method of Dreizen [Dre87] (see Subsection 1.2.3.1) is the most efficient method with low complexity and the one closest to the newly proposed method in several aspects. It directly includes lossless compression. Its complexity is very comparable to that of the new method (see Subsection 4.5.3.3). It concentrates the increase of spatial resolution to regions with high gray scale differences, which can be compared to the combined increase of spatial and gray scale resolution. A comparison of the two methods is therefore of great interest.

Dreizen's method was implemented exactly as described in [Dre87], with only one modification. The original fixed Huffman code was replaced by the more flexible and universal arithmetic coding with the cumulative frequency model described in [Wit87].

6.1.1.5 Dreizen's Method, Homogeneous Variant

The implementation of Dreizen's method as described above was also used in its homogeneous variant, i.e. with only one pass over the data, splitting the nodes in simple breadth first order. This can be seen as a good representative of the classical uniform spatial subdivision approach to progressive transmission. Averaging, as opposed to the simple subsampling used here, might provide slightly better results.

6.1.1.6 Pruned Tree Structured Vector Quantization (PTSVQ)

Tree structured vector quantization is a variant of vector quantization using a tree structured codebook that greatly simplifies encoding. Progressive transmission is achieved by transmitting one layer of the tree for each pass. Pruned Tree Structured Vector Quantization (PTSVQ) [Ris90] forms an embedded hierarchy of optimal subtrees of the original tree. Progressive transmission is

carried out by enlarging the tree by one node at a time and transmitting the information for the corresponding vectors.

6.1.1.7 Block Truncation Coding

Block truncation coding (see [Del79] and Section 1.2.1.1) is a simple image coding method that leads to relatively good coding results at a rate of 2 bits per pixel. Because it works very locally, it cannot be used for progressive transmission.

6.1.2 GDF Variants

The following aspects of the new method proposed in this thesis have been varied in the experiments: The way of compressing GDF, the method used to decide the reproduction values, and the sequence of the components. Additional compression of GDF was either not used (sometimes denoted by *plain*), achieved by optimizing the gray scale hierarchy as described in Section 5.3 (*optimized*), or by arithmetic coding as explained in Section 5.4 (*arithmetic*). Reproduction values were selected according to Subsection 5.1.1 (*black to white*), Subsection 5.1.3 (*center values*), or as explained in Subsection 5.1.4.4 (*averages, av.*).

The component sequences used can be described as follows: One component sequence, denoted *bi*, simply transmits the components bit plane by bit plane. Three component sequences, *o2*, *o3*, and *o4*, were constructed by arranging the components in a rectangle similar to Table 4.2, and then selecting the components starting in the upper left corner according to a rating function whose lines of equal value are shown in Figure 6.0. The full component sequences are included in Appendix A.

Figure 6.0. Component sequences *o2*, *o3*, and *o4* (left to right)

Component sequence *o3* was constructed on the assumption that in the case of the mean square error a difference of one level corresponds to a factor of four in both the spatial and the gray scale hierarchy. In the case of *o2*, the gray scale

resolution is increased faster, and in the case of *o4*, the spatial resolution is increased faster. These three sequences were constructed to test whether the balance assumed for sequence *o3* was more or less correct, and by how much a deviation from this balance would affect the reproduction results. The last row has been set off because the visual system is much less sensitive to high frequencies than to middle and lower frequencies [Mann74].

Only certain combinations of additional compression, reproduction value selection, and transmission sequence were used. The transmission sequence *bi* was combined with center value reproduction to approximate simple transmission bit plane by bit plane, with the reproduction value selection mostly used in this case. The black to white reproduction method was combined with the transmission sequence *o4*. A long delay of components of high spatial resolution would in this case have lead to ringing effects.

Sequences *o2*, *o3*, and *o4* were used with average value reproduction, and with or without arithmetic compression. With arithmetic compression, they represent the best variants of the newly developed method implemented so far; without additional compression, they represent the best variants when the receiver is not fast enough to decode arithmetic codes.

6.2 Overall Compression

One aspect of the performance of a compression scheme that can easily be measured is the overall compression rate. For many progressive transmission applications, this may not be of primary importance. However, for applications where no loss of information is tolerated and large amounts of image data are stored, but seldom retrieved, overall compression is very important.

Also, overall compression can give an indication of the performance at intermediate stages. Transmission methods that use similar ways to approximate an image can be expected to perform similarly if they have about the same overall compression rate.

The images used in the experiments in this section have been taken from version 1 of the Japanese Standard Image Data Base (SIDBA) [Ono79], and are shown in Figure 6.1. All images have 256·256 pixels with 8 bits per pixel. Table 6.1 shows the overall compression rates for those compression methods described in Section 6.1 that allow lossless compression. Results from [Dür88c] and [Dür89c] are combined with some newly calculated ones. Smaller values in Table 6.1 indicate better compression, and a value above 100% shows that in this case, no compression is possible.

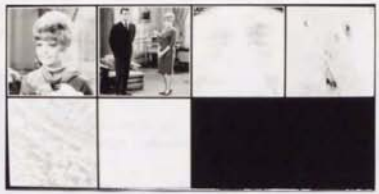

Figure 6.1. Originals of the images from the Standard Image Data Base

	Girl	Couple	X-ray	Moon	Aerial	Fax data
Sequential	64.43	61.28	65.97	70.13	77.98	50.48
DF (Gray code)	77.19	68.50	-	83.13	96.61	-
LZW	76.73	74.36	-	94.16	106.73	-
Dreizen	70.66	65.39	80.88	74.39	85.48	53.39
GDF (plain)	73.34	64.81	84.99	82.03	88.57	60.09
GDF (optimized)	59.52	57.47	57.38	71.76	82.33	47.09
GDF (arithmetic)	61.50	58.55	58.23	75.04	83.70	50.27

Table 6.1. Overall compression rate for various images and methods
(compression given as $\frac{\text{size of compressed image}}{\text{size of full image}}$ %)

Table 6.1 can be interpreted as follows: DF using a Gray code and LZW clearly perform worse than the other methods. For DF, this is due to the fact that parentheses are needed for every bit plane, even if the structure of adjacent bit planes is similar. In fact, Kawaguchi et al. [Kaw83] did not propose this method for lossless compression. LZW, on the other hand, theoretically can perform as well as any other method if the image conforms to some statistical assumptions [Lem86]. However, it obviously takes too long to adapt to the characteristics of an image. This is clearly due to the fact that gray levels are just interpreted as unrelated symbols[1].

[1] A recent paper [She90] gives successful experimental results for binary images constructed according to a Markov model. However, it is doubtful whether these results can be extended to natural gray scale images.

The other methods may approximately be ranked in the order Sequential, GDF (optimized), GDF (arithmetic), Dreizen, and GDF (plain). The performance of the sequential method can easily be improved by using a better predictor. However, this method is not predictive.

That GDF (optimized) and GDF (arithmetic) perform better than the sequential method in some cases is due to the fact that in these images, not all gray levels are used. This fact is directly used in these optimizing algorithms. However, it is effective only towards the end of the transmission, and up to a rate of two or three bits per pixel, the additional compression achievable with arithmetic coding is in no case larger than 10%.

GDF, without any entropy coding, performs almost as well as the method of Dreizen, which uses entropy coding. On the other hand, if even a simple entropy coder is added to GDF, or if the gray scale hierarchy is optimized, then GDF leads to better overall compression rates. When optimizing the arithmetic coding of GDF and introducing prediction (see Sections 5.1.5 and 5.4), compression rates will easily increase further.

6.3 Visual Evaluation

Sections 4.3, 5.1, and 5.3 already provided examples of results achievable with the new method of data compression and progressive transmission proposed in this thesis. This section provides additional results using some of the example images of Section 6.2.

Figure 6.2. Transmitting the image "Girl" with Dreizen's method
(bytes transmitted: 300, 600, 1200, 1800, 2400, 3600, 4800, 9600)

Figure 6.2 gives the result of applying Dreizen's method to the image "Girl". The reader is asked to compare the result with those shown in Section 4.3 (Figures 4.6, 4.7, 4.8), Section 5.1 (Figures 5.1, 5.2, 5.5), and Section 5.3 (Figure 5.9). Compared to

these figures, Figure 6.2 differs especially in the rather low amount of detail in initial stages and the still clearly visible large squares in areas of slow intensity change at later stages.

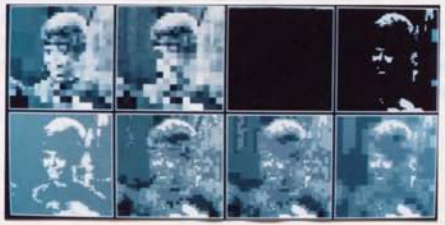

Figure 6.3. The image "Girl" using eight different methods
(bytes transmitted: 600)

In Section 6.5, an experiment comparing eight transmission methods is described. It is therefore of interest to present some results of applying these eight transmission methods to some of the example images used in the previous Section. These methods are Dreizen, Dreizen (homogeneous), sequential, *o4* (black to white), *bi* (center value), *o2* (av., arith.), *o3* (av., arith.), and *o4* (av., arith.), shown in Figure 6.3.

Figure 6.4. The image "X-ray" using eight different methods
(information transmitted: 8192 bytes = 1 bit per pixel)

Here the difference between all these methods is clearly visible, although a ranking cannot easily be established in all cases. Figure 6.4 shows the application of the eight methods to the image "X-ray". The images in the bottom row are difficult to distinguish, but all show much clearer lines than the method of

Dreizen (top left), a fact that is important in medical applications. The black to white reproduction shows more contrast than the original image, an effect that may be desirable in some cases.

Dreizen mentions the fact that his nonhomogeneous method adapts better to images that are close to binary images than methods which homogeneously increase spatial resolution. Figure 6.5 shows that in this case, the new method performs even better. However, in this case, using averaging to select reproduction values performs somewhat disappointingly.

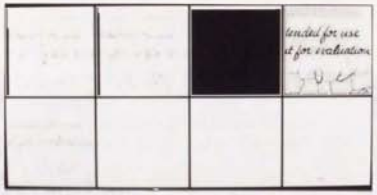

Figure 6.5. The image "Fax" using eight different methods (information transmitted: 1024 bytes = 0.125 bits per pixel)

The reason for this is that subquadrants that contain both background and foreground usually contain more background than foreground. Therefore the averages for the reproduction are close to the background. Due to this, the letters are shrunk and so hardly readable. On the other side, the black to white reproduction enlarges the letters. This, combined with the additional contrast, makes them easily readable. It also shows that the uppermost part of the image contains additional detail, an information that the user may use to concentrate transmission to that part of the image.

Note that if the text were white on a black background, the best reproduction method would be white to black, although this has been discouraged in Subsection 5.1.1. This example clearly shows that selecting the reproduction method based on the type of the image can easily improve the efficiency of transmission.

Another image used by many authors for image compression experiments is "Lena". It is the only 512·512 pixel image used for the experiments. Results at various rates are shown in Appendix B. Interested readers can compare these results with other results published, but should be aware of the following two points: First, the method presented in this thesis is a low complexity method, and comparisons with methods that have high computational requirements, such as transform coding methods, should be made with care. Second, no smoothing is used. Especially at initial stages, the images are much better visible when viewed from a greater distance.

6.4 Analytical Evaluation

To thoroughly evaluate the performance of the various methods and variants, a criterion that can readily be calculated and is related to image quality has to be used. The most popular such criterion is the average mean square error, mostly used in the form of the peak to peak signal to noise ratio (PSNR), measured in dB. The PSNR is defined as

$$\mathrm{PSNR} = 10 \log_{10} \frac{(2^b-1)^2}{E((x-\hat{x})^2)} \text{ db}, \tag{5.13}$$

where the numerator is the maximally possible energy per pixel for an image with a gray scale resolution of b bits per pixel. The numerator is used to scale the denominator, which denotes the average energy of the noise, the difference between corresponding pixels of the original image and the image reproduction.

The PSNR for the example image "Girl" and the eight methods used in Section 6.3 is shown in Figure 6.6. In the initial stage, the methods behave somewhat unpredictably; this part will be analyzed below. The steepness of the curves in the final part of the transmission is due to the fact that the quantization error reaches zero.

Between rates of about 1.5 bits per pixel and 4 bits per pixel, the behavior is very similar to that shown in Figure 2.4 for the case of a bintree with edges of sidelengths of powers of two. This is due to the fact that for all pixels, the sizes of the quantization intervals are powers of two. Because of the original quantization, it is difficult to change this directly. An improvement is however possible if the number of components is increased as proposed in Subsection 5.2.3.

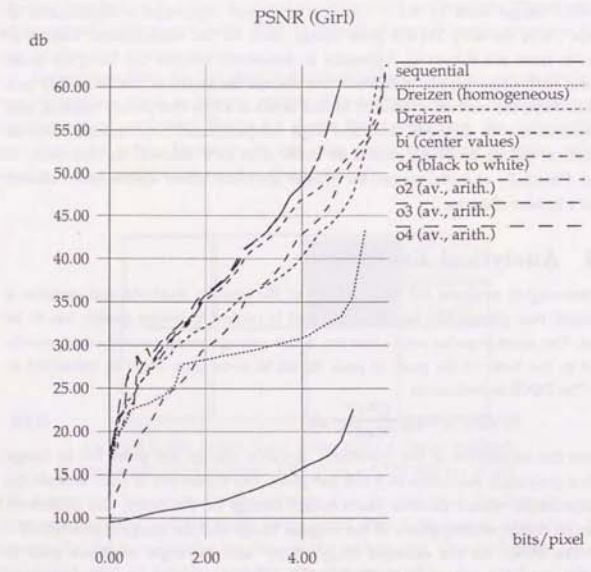

Figure 6.6. PSNR of the image "Girl" for eight methods

Some variants of the newly proposed method show a reduction of PSNR in some parts of the transmission. As already mentioned in Subsection 5.1.4.4, for the component with s=8 and c=2, the actual trace averages and the used reproduction values differ. This leads to a deterioration of the image. How transmission can be optimized by changing the component sequence is shown in Figure 6.7, using the manually optimized sequence *o5*. When constructing the sequence *o5*, the "bad" component with s=8 and c=2 was delayed as long as possible, and as a result, *o5* performs as good as any of the other sequences up to a rate of about 1 bit per pixel.

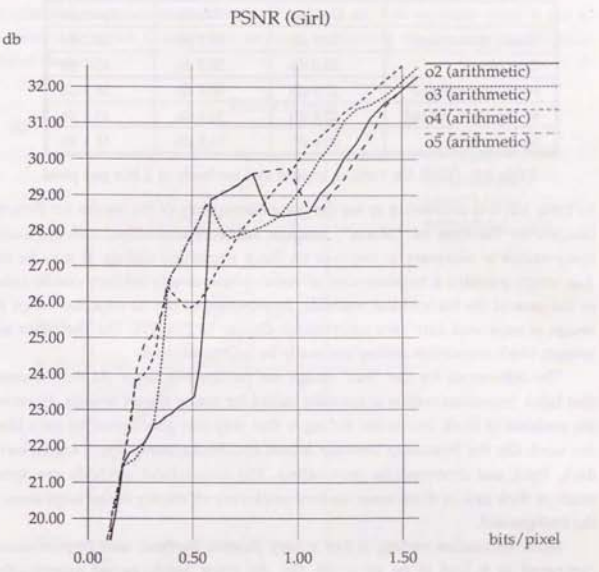

Figure 6.7. PSNR of the image "Girl" for component sequence *o5*

Besides the sequential method, used for reference, the methods used so far for comparison are basically designed for progressive transmission and not intended for pure data compression. It is therefore of interest to compare the newly developed method to some methods primarily used for compression. Table 6.2 compares several variants with block truncation coding, for which the results have been taken from [Gri87].

	Couple	Moon	Fax data
Block truncation	32.9 db	32.7 db	28.2 db
Dreizen	33.0 db	30.7 db	42.1 db
bi (center values)	37.9 db	30.7 db	38.4 db
o4 (black to white)	32.4 db	24.8 db	35.0 db
o4 (av., arith.)	38.6 db	34.8 db	46.4 db

Table 6.2. PSNR for various images and methods at 2 bits per pixel

In Table 6.2, it is interesting to see the great discrepancy of the results for different images. In the case of "Moon", average value reproduction and arithmetic compression is necessary to improve on block truncation coding. It may be that this image contains a high amount of noise, which overly inhibits condensation in the case of the hierarchical methods. Improvements can be expected when the image is separated into two components similar to [Yan77]. For the other two images, block truncation coding can easily be improved upon.

The differences for the "Fax" image are particularly large. At first, it seems that block truncation coding is specially suited for nearly binary images. However, the problem of block truncation coding is that only two gray levels for each block are used. On the boundary between letters and background, there are however dark, light, and intermediate gray values. The hierarchical methods can spend much of their rate in these areas, as they work very efficiently in the large areas of the background.

Block truncation coding is not a very flexible method, and improvements compared to it had to be expected. On the other hand, vector quantization, especially in the form of PTSVQ, is a flexible and efficient coding method. Due to the cooperation of Dr. Eve A. Riskin, it was possible to make a direct comparison of HSRQ and GDF with PTSVQ. Figure 6.9 shows the PSNR for the image "Lena" (for the original, please see Figure B.5 in Appendix B). The values for PTSVQ have been taken from [Ris90], Figures 4.8 and 4.9.

Here, the new method performs better at rates up to 0.15 bits per pixel, but is then overtaken by PTSVQ. PTSVQ reaches 32 db, but cannot continue compression because the codebook is exhausted. On the other side, the new method, after a step back similar to those visible in Figure 6.7, reaches 32 db around 1.5 bits per pixel and then steadily improves at about 6 db per bit.

In the case of PTSVQ, additional performance of about 2 db is possible by using prediction [Ris90]; this however eliminates the possibility of progressive transmission. On the other hand, using components based on traces as proposed in Subsection 5.3.2.1, true trace averages, and a better model for arithmetic

compression, improvements of the new method are also possible. Also, it has to be noted that vector quantization depends on training the quantizer to a certain kind of images.

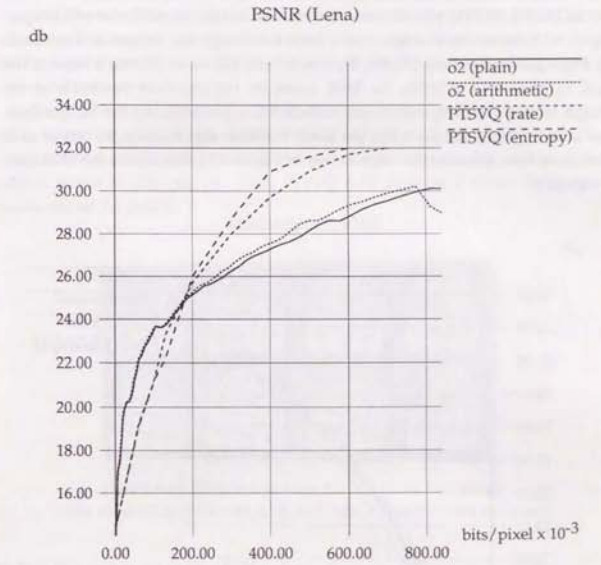

Figure 6.9. PSNR for the image "Lena"

HSVQ also compares well with some other compression methods. For "Lena", Westerink et al. [Wes88] give a value of about 29.2 db PSNR at a rate of 0.63 bits per pixel using subband coding combined with vector quantization; with the new method, 29.48 db are achieved at the same rate. Perkins [Per88] gives values of about 30 db, 34.5 db, and 39 db[1] at rates of 1, 2, and 3 bits per pixel for the discrete cosine transform with marginal bit allocation. As can be seen from Figure 6.9, the

[1] Perkins uses a different power to normalize the SNR; 5.687 db have to be added to his results to allow comparison.

new method reaches 30 db already at 0.8 bits per pixel; for 2 and 3 bits per pixel, the values are 35.5 db and 41.5 db, and all this without arithmetic coding. On the other hand, highly sophisticated adaptive coding methods can perform about 5 to 7 db better at a rate of about 0.7 bits per pixel [Pea90].

In [Ris90], PTSVQ was also used for medical images, in particular MR images. Figure 6.10 shows the average results for a five image test sequence. The values for PTSVQ are taken from [Ris90], Figures 4.2 and 4.6. As in [Ris90], instead of the peak to peak SNR (PSNR), the SNR based on the standard deviation of the images is used. The original images contain 9 bits per pixel. As the BC quadtree was only implemented for 8 bits per pixel, the most significant eight bits of each pixel have been used. At the rates shown in Figure 6.10, this affects the SNR only marginally.

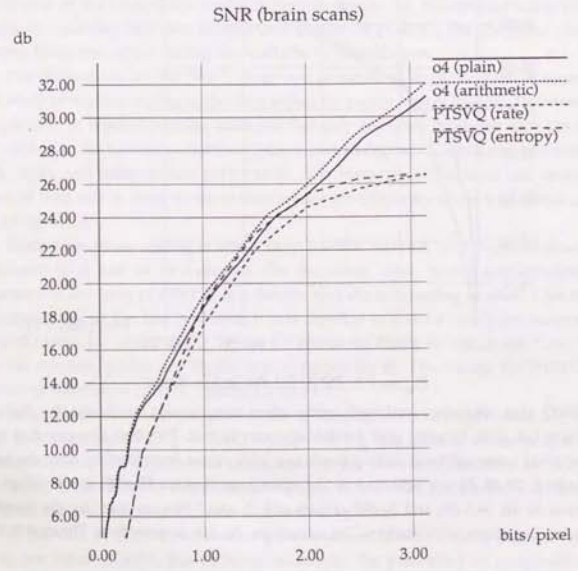

Figure 6.10. Average SNR for five brain scans

The results shown in Figure 6.10 are surprisingly favorable for the new method. As already explained in Subsection 4.6.1.3, vector quantization, although applicable over a wide range of rates, has to concentrate on a particular rate interval because of the relation between vector size, codebook size, and rate.

However, in Figure 6.10, the new method performs better than PTSVQ over all rates. For medical images like brain scans, where large areas of uniform background are combined with areas of high activity, a hierarchical approach is clearly advantageous.

The difference between PTSVQ and the new method is also shown in Figure 6.11, using the same image as in [Ris90], and partially the same rates, for progressive transmission. The difference in especially notable in the initial stages, where at 0.25 bits per pixel, PTSVQ only displays a binary image at a resolution of 2·2 pixels.

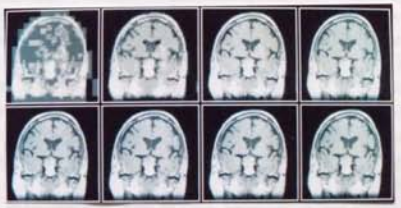

Figure 6.11. Progressive transmission of an MR image
(rates used: 0.125, 0.25, 0.46, 0.84, 1.17, 1.52, 2.0, and 8.0 bits per pixel)

6.5 Recognition of Faces

As explained in Chapter 3, there exist a wide range of applications for image coding and progressive transmission. It was impossible to evaluate the new method with respect to all of these applications, and so for more detailed evaluation, one such application, the recognition of faces, has been chosen. This has the following advantages: First, there are previous results in this area that allow a comparison with other methods (see Subsection 6.5.3). Second, the problem is well defined and easy to understand, and appears in many possible applications of progressive transmission. Third, the recognition of faces is an activity for which the human visual system is particularly suited and trained; the results in this section therefore complement the analytical criteria used in the

previous sections. Also, the rate necessary for face recognition is quite low; in this respect, too, there is a contrast to the analytical methods, which may not be very appropriate for low rates, and to the results of Chapter 2, which primarily apply to high rates.

6.5.1 Experiment Setup

The experiment was carried out with the senior (4th grade) students of the Department of Information Science, Faculty of Science, of the University of Tokyo. This is a group of altogether 31 students who know each other for about one and a half years. Two 35mm slide pictures were taken of 23 of the students, a teaching assistant, and the author of this thesis. The better of the two pictures was used in the experiment. In addition, slides of 15 well known personalities from politics, sports, etc., were taken from photographs in weekly or monthly magazines.

All the slides were scanned at a resolution of 25μ per pixel with a high resolution drum scanner. Then the images were scaled by a factor of three in each direction by a simple averaging algorithm to eliminate noise. A new pixel was calculated as the average of seven of the nine old pixels, after eliminating the pixels with the highest and lowest gray level values. The size of the final images was 256·256 pixels, with 8 bits per pixel. The averaging also allowed to stretch the histogram so that the whole range of 256 gray levels was used; many of the slides had been taken somewhat too dark. The images are shown in Appendix C.

In the recognition experiment, altogether 28 of the students and 2 assistants who knew the students well participated. Each of them was presented all of the 40 images, one after the other. They were instructed to press the return key as soon as the image developing on the screen was recognized. Then they had to enter a code number from a list that contained 62 names, those of all the students and of 30 famous personalities. Only correct results were considered for later processing. When the return key was pressed, transmission was stopped and the image stayed on the display while the number was entered, but it was impossible to resume transmission.

For each participant, each of eight methods was used for exactly five images. The sequence of the images and the methods used for display were randomized independently to eliminate the influence of the learning effect and of the relation between methods and images. To reduce the effect of human reaction time, all methods were tuned so that an answer was possible within 25 to 50 seconds from the start of the transmission, depending on individual ability and the image displayed.

6.5.2 Results

The number of bits needed on average to recognize a face are given for each method in Table 6.3. It clearly shows the improvements possible with the new method when compared to previously available methods of similar complexity. Compared with the method of Dreizen [Dre87], the best quadtree based progressive transmission method up to now, the new method can save about 40% of the bits to be transmitted, and thus the transmission time. This result is not very much affected even if arithmetic compression is not used for the last three variants. Compared to sequential transmission, an improvement by about a factor of 20 is possible. This clearly shows the advantages of progressive transmission in general.

Method	bits transmitted	pixels painted	squares painted
Sequential	177031	37981	37981
Dreizen (homogeneous)	16641	255114	1488
Dreizen	13107	224435	2227
o4 (black to white)	14982	72589	3660
bi (center value)	11561	90986	6015
o2 (av., arithmetic)	8917	147170	2945
o3 (av., arithmetic)	7914	128844	2713
o4 (av., arithmetic)	10158	109901	4522

Table 6.3. Average number of bits necessary to recognize a face

When comparing the different variants of the same method, it is interesting that the difference between homogeneous and nonhomogeneous subsampling is not very large. This also indicates that the improvement of the method of Dreizen over, for example, Knowlton, is not very large, at least for this application. The reasons for this are that with the prediction scheme used by Dreizen, the homogeneous regions of the image can be coded with fewer bits per square than the nonhomogeneous ones. Also, no additional bits are necessary for node selection.

Another point of interest is that the simple transmission bit plane by bit plane, if combined with increasing spatial resolution for each bit plane, is more efficient than increasing spatial resolution, even if done with Dreizen's method.

Comparing different variants of the new method shows that the selection of the component sequence and the method of reproduction affects the results in a way similar to what has been observed in the previous section. All component sequences used with average value reproduction perform better than previously

proposed methods, but the component sequence clearly affects the result. The average PSNR for the faces and methods used (with exception of the sequential method) are shown in Figure 6.12.

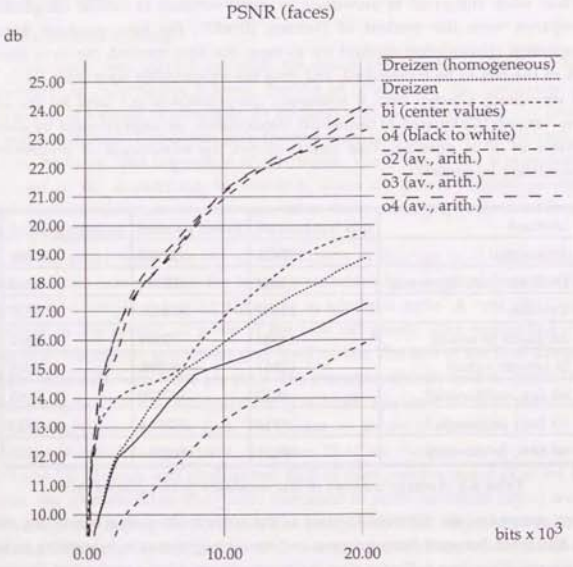

Figure 6.12. Average PSNR of the faces for each method used

Component sequence o3 shows the best PSNR in the region of interest, but the difference is not as large as the difference in Table 6.3. That the methods that do not use trace averages for reproduction achieve recognition with a much lower PSNR can be explained by the fact that using trace averages is specifically designed to reduce PSNR. Recognition is obviously also possible if the image is slightly shifted in space (Dreizen) or in gray scale (black to white and center value). However, there is nothing wrong with trying to optimize PSNR; the cost associated with transmitting correct averages is small, and the improvement, both in terms of recognition time and visual quality, is considerable.

Table 6.3 also gives the number of pixels and of squares (of any size, including pixels) painted on average by each method for the recognition of a face. This gives an indication of how the performance of each method is affected when the access to the frame buffer becomes a bottleneck.

If the frame buffer is accessed by bit plane and not by pixel, then the black-to-white and the center value variant of the new method will perform much better, as only one or two bits per pixel have to be updated in this case. For the other variants and methods, such savings are impossible or difficult.

In the last column, the method of Dreizen performs better than the others. It should, however, be kept in mind that if whole squares can be painted into the frame buffer, the speed of the frame buffer will be much less of a bottleneck. Also, the component sequences for the new method have been chosen so as to perform near the optimum for the bit average. Sequences that completely avoid the early transmission of the lowest level(s) of the spatial hierarchy will perform better if the frame buffer access, in numbers of squares, is a bottleneck.

The results in the last column of table 6.3 also can be used to project the number of pixels written to the frame buffer if smaller images, for example with 64·64 or 128·128 pixels, are transmitted, or if the presently used images are displayed at a smaller size initially as proposed by Hofmann and Troxel [Hof86].

6.5.3 Comparison with Transform Coding

As mentioned in the introduction, transform coding has also been proposed for the progressive transmission of images. Particularly Lohscheller [Loh82,83,84] investigated the use of the cosine transform for progressive transmission. In his dissertation [Loh82], he published some results of a face recognition experiment. Comparison with these results therefore provides an additional way to evaluate the new method.

The description of the experiments in [Loh82] is not very detailed, but it can be concluded that they were mainly carried out using the same principles as described above. The time it took a viewer to recognize an image is comparable. The number of experiments was considerably smaller, using seven portraits and between 10 and 15 viewers. The relation between the viewers and the displayed persons is not clear, but the smaller number suggests that it was closer than in our case.

A seemingly important difference between the two experiments is the size of the used images, 512·512 in the case of Lohscheller, and 256·256 here. However, the minimal amount of information necessary build up an image that allows to identify a person should be independent of the resolution of the original image, provided that the original resolution is not extremely low. The results of both

experiments can therefore well be compared in terms of the number of bits necessary for the recognition.

Block size	8·8	16·16	32·32
Bit average	33840	19827	10512

Table 6.4. Number of bits for person recognition using DCT (from [Loh82])

The results obtained by Lohscheller [Loh82, p.89] are given in Table 6.4 for the different transformation block sizes he used. For a block size of 32·32 pixels, these results come close, but clearly lie above to those of the new method; for the more practicable block sizes of 16·16 and 8·8, the results fall way behind those presented in this paper.

These results are not changed by the subjective adaption as introduced in [Loh84]. First, the complexity of this adaption makes it necessary to reduce the block size to 8*8. Second, the adaptive method starts slowly because the class numbers, transmitted first, do not contribute directly to the image buildup. The break-even point with the nonadaptive method lies at a rate of about 50,000 bits.

That the theoretically better founded transform coding approach does not perform better can mainly be explained as follows. The optimal bit allocation to the individual components usually quantizes the first (DC) component finer than the original pixels. As computational considerations make bitwise (as opposed to componentwise) transmission impracticable, image transmission is started with a low spatial and an overly high gray scale resolution.

6.5.4 Comparison with Perception Experiments

Harmon [Har73] investigated the amount of information absolutely necessary to identify a face. He gives a bound of 768 bits (a 16·16 grid with eight gray levels). This is about ten times less than the best result presented in Subsection 6.3. This difference seems very big, but there are several reasons that can explain it.

First, the pictures for Harmon's experiments were taken by a professional photographer in a photo studio. All pictures are simple frontal pictures and fill most of the available area. The background is completely uniform, and the illumination optimally shows the features of each face. Significant improvements of the results presented in this section can be expected in an application where the images have been taken with more care.

Second, the viewers did not have any choice between guessing based on the present image and waiting for a more detailed image. Recognition accuracy in the

case of Harmon was 48 percent, whereas here, it was higher than 90 percent, and the bit average of Table 6.3 includes only correct answers.

Harmon also reported that recognition efficiency was improved by optimally placing the image with respect to the grid. This can be expected for the methods discussed in this section, too. In particular, in the case of the variants based HSRQ, not only the placement with respect to the spatial grid, but also the placement with respect to the "grid" of the gray scale can be optimized.

Altogether, the results of this chapter clearly show that the new method presented in this thesis in an improvement over a variety of other methods that have been used for progressive transmission and image compression. Compared to other low complexity methods like that of Dreizen, higher compression is achieved. Compared to considerably more complex methods like transform coding, the complexity is reduced while obtaining similar compression rates. This is clearly due to the combination of the increase of spatial and gray scale resolution using HSRQ.

7

Extended Applications

This chapter discusses the application of the basic method to imprecise data, color images, and image sequences. In all three cases, the discussion centers about how particular features of the method lead to new approaches and solutions, and in some cases to new problems that still have to be solved.

7.1 Imprecise Data

The storage and processing of imprecise data by digital computers poses special problems that are not present if data and operations can be specified exactly. Areas where imprecise data is of particular importance are geology and geophysics. The very large amount of data to be stored in a geological information system also requires special consideration of storage efficiency without unduly decreasing accessibility. The large variety of the data to be stored makes it desirable to use an unified but flexible approach, where all types of data are stored with basically the same data structure, and processed with the same basic algorithms.

A special property of geological data is its relation to space. A geological data base will store data of a large region, but this data is obtained, and later most times used, on a smaller scale. Also, geological data has spatial continuity, which means that data for neighboring points and regions is in most cases similar. To design an efficient geological information system, using these properties is a simple necessity.

Some contributions towards an approach integrating various kinds of data while using basically the same data structures and algorithms have been given in [Dür90b]. Here those parts of this paper are included which discuss how to store imprecise density data using the BC quadtree.

Being able to store density data of different degrees of accuracy is important in geological applications. Using the BC quadtree, this can be done in several ways. First, the gray scale hierarchy can be changed. Second, several trees

describing different aspects of an uncertain density function can be stored and accessed in parallel.

The gray scale hierarchy can be changed by introducing an additional symbol "*", which indicates that for the present region of the image, there is no further precision available. The gray scale hierarchy for our example, without and with this extension, is shown in Figure 6.1. The use of altogether four symbols allows easy binary coding with 2 bits.

Figure 7.1. Gray scale hierarchy allowing imprecision
(left: original gray scale hierarchy; right: changed gray scale hierarchy)

There are other ways to change the gray scale hierarchy. If one part of the gray scale is of particular interest, a finer subdivision can be used there, increasing the number of significant bits in the range of interest. Also, the scale can be shifted while keeping the number of significant bits constant, for example by using a logarithmic scale instead of a linear one. If there is a particular density level that separates valuable ore from useless rock, the most significant bit can be made to distinguish these two density ranges, while less significant bits provide more detail. This is particularly interesting in connection with progressive transmission.

If high precision is needed in combination with uncertainty, two separate trees can be used. One of them can represent the minimum and the other the maximum of the expected density, or one of them can represent the mean and the other the standard deviation. The number of significant bits can be different for the two trees, particularly in the later case.

With this amount of possibilities, a careful choice becomes important. Criteria for the decision include the needs of the application, the probability of changes of critical density values, the ease of implementation in software or hardware, and time and space considerations.

7.2 Color Images

Progressive transmission can be extended to color as shown in [Hil83], where the basic method of Knowlton [Kno80] was applied to each of the color components of the image. However, more elaborate methods of progressive transmission,

when used in the same way, may not lead to as good results as might be expected from a simple extension of the gray scale case.

To understand the principles and problems inherent in the storage, compression, and progressive transmission of color images, the first subsection of this section gives a short introduction to color science and color spaces. In the second subsection, an new class of cubic color spaces is then proposed to allow the efficient and lossless progressive transmission of color images. The third subsection discusses the result obtained so far, the problems encountered, and possible improvements.

7.2.1 Basic Color Science

This subsection gives a short introduction to color science, as far as necessary for the later development. For more details, the interested reader is referred to the literature [Fol90], [Mey86].

In the human eye, *color* information is received by three kinds of cones, which are mainly located in the center of the retina. On the other hand, light *intensity* is sensed by the far more numerous, more widely distributed, and more sensitive rods. The number of kinds of cones suggests that to represent all color combinations visible to a human observer, three real quantities are sufficient. This has been confirmed by colorimetric experiments.

For digital representation, these three quantities are usually chosen to correspond to red, green and blue, in accordance with the phosphors of color monitors. The resulting color space is called the RGB color space. In this space, the colors representable on a monitor (a subset of the complete range of colors perceptible by the human viewer) form a cube. Cuts through this cube, orthogonal to the green axis, are shown in Figure 7.2. The individual cuts are taken in even intervals, and the first and last square represent the bottom and top face of the cube.

Figure 7.2. The RGB color cube

The RGB color space is well suited for storage, calculation, and display. However, it has several disadvantages that led to the development of a variety of other color spaces. First, it is difficult for human users to specify color in terms of RGB values. The HSV (hue, saturation, value) and HSL (hue, saturation, luminance) color spaces, which can take the form of (hex)cones, double (hex)cones, or cylinders, have been developed for this purpose.

Second, if the color coordinates are based on actual colors, it is impossible to represent all visible colors with three positive values. The cones of the retina work as inhibitors in certain areas of the spectrum. Thus for colorimetry and standardization, the CIEXYZ color coordinates are used (CIE: Commission Internationale d'Eclairage). With them, all visible colors can be expressed by three positive values.

In all the above color spaces, the distance between two colors does not correspond to the percepted difference between two colors. This led to the development of the so called uniform color spaces $L^*a^*b^*$ and $L^*u^*v^*$.

For bandwidth reduction of the television signal (NTSC), the Y_tIQ color space is used [Bad86]. Its development was directed by the observation that the resolution power of the human eye is greater for luminance and smaller for color changes, and by the necessity to make the composite signal compatible with the already existing black and white television signal and to fit the color signal into the available bandwidth.

In the NTSC signal, the luminance component is denoted by Y_t to distinguish it from the Y component of CIEXYZ. The I color component is oriented from reddish-orange to bluish-green. This is the direction of the color spectrum in which the human eye is most sensitive. The Q component is oriented orthogonally to the I component, from yellowish green to bluish magenta. The three components are transmitted using bandwidths of 4.2, 1.3, and 0.5 Mhz, which shows their relative importance to the human eye.

The transformation from the television monitor RGB primaries to Y_tIQ are given by

$$\begin{bmatrix} Y_t \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \qquad (7.1)$$

The first row of the transformation matrix indicates that the contribution to the luminance signal varies greatly for each of the RGB primaries. This fact will need our attention in the next subsection.

7.2.2 Cubic Color Spaces

The application of the requirements of Chapter 3 to color images seems straightforward. Each of the three components of the image is transmitted in parallel using the basic techniques outlined so far.

However, as pointed out in the last subsection, the resolution of the human eye is higher for luminance than for hue or saturation. It seems worth to try to apply these principles to progressive transmission, including all the requirements defined in Chapter 3. Basically, the image is transformed to the Y_tIQ color space at the sender, then the three components are transmitted independently, initially spending more of the transmission time on the Y_t component, and at the receiver, the original image is reconstructed. Transmitting the luminance component first has an additional advantage. Not only is the human eye more sensitive to luminance than to colors, but a black and white image also contains most of the information of the color image.

The transform of formula (7.1) leads to values for Y_t, I, and Q that have to be requantized. Thus a lossless reproduction of the original image becomes difficult. To overcome this problem, a new class of color spaces is proposed here. To avoid any loss of information, and any inefficiencies, the transformed color space is of the same form as the original RGB color space, namely a cube with $2^b*2^b*2^b$ locations (usually b=8). Also, the main axes of the new cube will correspond as much as possible to the Y_t, I, and Q directions of the NTSC color system.

In the Y_tIQ color space, the I and Q components are translated so that they have an average of 0. For the representation with b bits per pixels, this translation is obviously of no importance; what is important is the rotation included in (7.1). The three-dimensional rotation can be approximated by a combination of two-dimensional rotations.

To approximate the two-dimensional rotations, a newly developed generalization of a technique originally introduced by Knowlton [Kno80] is used. Knowlton approximated the linear transformation of two values to their average and difference, which can be understood as a rotation by 45°, by a permutation using a lookup table.

Hill et al. [Hil83] later discovered the ring structure of this approximation and implemented it with a simple algorithm. To use the same procedure for transformation and for retransformation, Hill et al. exchanged the two variables inside the procedure. However, the exchange can also be performed outside the procedure.

This ring method can be generalized by making the approximate angle selectable. An outline of the permutation algorithm, in unoptimized form, is

given below. Rotations by multiples of 90° are exact rotations; for other angles, this is of course not possible. Some rotations produced by the algorithm are shown in Figure 7.3.

1. Determine the ring i of the two input coordinates.
2. Determine the ring radius r of ring i, which is defined as the distance of the ring center line from the center of the square ($r = i - 0.5$).
3. Calculate the shift distance d for this ring from the rotation angle α and the ring radius r ($d = 8r\alpha/360°$).
4. Shift the input values by the number of steps d around the ring.
5. Output the coordinates of the new position.

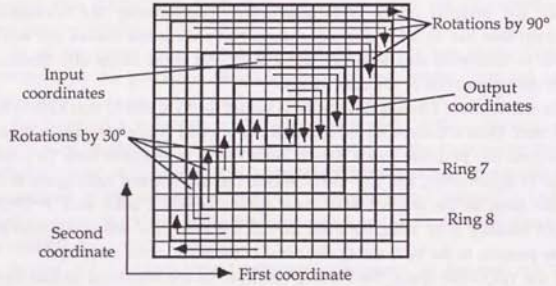

Figure 7.3. Rotation of a square with the generalized ring method

Steps (2) and (3) can be combined to a simple table lookup once the angle(s) used are known. The most complicated step of the algorithm is step (4); different cases have to be distinguished. It can be somewhat simplified by restricting the angles to the interval [0°, 45°]. This is easily possible, as other angles can be realized by a combination with rotations of multiples of 90°, which can be realized by variable exchange and subtraction.

Figure 7.4 shows the RGB color cube of Figure 7.2 rotated by 45° in the RB plane about the G axis. The diagonals appear lighter than their surroundings because they have been stretched. To actually approximate the transformation to the Y_1IQ color space, three two-dimensional rotations are combined. The first rotation keeps the B value constant, the second rotation maintains the original R value, and the third rotation the original G value. If for all three rotations, angles of 45° are used, the result is shown in Figures 7.5 through 7.7.

Figure 7.4. The RGB color cube rotated by 45° about the G axis

Figure 7.5. The RGB color cube rotated by 45° about the B axis

Figure 7.6. The color cube of Figure 7.5 rotated by 45° about the R axis

Figure 7.7. The color cube of Figure 7.6 rotated by 45° about the G axis

In Figure 7.7, the coordinate axes can definitely be identified with the Y_t, I, and Q components of the Y_tIQ color space. The horizontal axis corresponds to I, the vertical axis to Q, and different squares correspond to different values of Y_t. Using three angles of 45° has the advantage that the original black-white diagonal has I and Q values of about 2^{b-1}. This simplifies the reproduction of the color image in the initial stages where only the luminance component is transmitted.

On the other side, with this transform, even fairly light colors are mapped to the minimum (black) Y_t value, and fairly dark colors are mapped to the maximum (white) Y_t value. Partly, this is due to the fact that the cubic shape of the color space has to be maintained. Partly, however, this is due to the fact that with this transform, each of the RGB values affects the Y_t value by the same amount, where as in the original transform of (7.1), the influence of G on Y_t is strongest.

This problem can be alleviated by changing the rotation angles, as shown in Figures 7.8 and 7.9. In Figure 7.8, the rotation that led to Figure 7.5 has been performed with a rotation angle of only 30°. Figure 7.8 then has been rotated about the original red axis, but also with an angle of only 30°, to obtain Figure 7.9. Compared with the luminance distribution in Figures 7.6 and 7.7, the luminance distribution in the individual squares in Figure 7.9 is more uniform.

In this case, the third rotation was omitted. It is not of primary importance that the coordinate axes of the transformed color space correspond to the I and Q components of the NTSC color space. In this way, additional distortions that are introduced with each rotation approximation can be avoided.

Figure 7.8. The RGB color cube rotated by 30° about the B axis

Figure 7.9. The color cube of Figure 7.8 rotated by 30° about the R axis

7.2.3 Results

The progressive transmission of color images was simulated in the same way as for gray scale images in Chapter 6, using the color spaces developed in the previous subsection. At the moment, the results are not completely satisfying. However, it is expected that their description in this work can shed light on the problems encountered and lead to new and better solutions.

The main result can be summarized as follows: Using a cubic Y_tIQ color space, transmitted images can be recognized at an early stage solely from the Y_t component. This leads to an additional improvement, compared with gray scale images and relative to the canonical form of the color image, by a factor of about three. This is a considerable improvement, which however had to be expected.

On the other hand, an acceptable color image is obtained only at a rate that is considerably higher than three times the rate necessary for a gray scale image with comparable quality. This is not very satisfying; a rate lower than three times the

rate for a gray scale image should be possible according to the explanations in Subsection 7.3.1.

That the results with color images are not satisfactory has several reasons. The first is that with HSRQ, the quantization for the two color components progresses differently. This leads to unpleasant false colors, which greatly reduce the apparent quality of the image. Obviously, the human visual system is much better at deducing color for a gray scale image, or just viewing the gray scale image as it is, than correcting false colors. The second reason is that for the bandwidth reduction with the NTSC color space to work properly, the color components should contain low frequencies only.

The third reason is that banding effects in a gradually changing area are much better visible in color images. The changes of the different color components do not occur at the same locations, which leads to colored bands. For high quality computer graphic images, where smoothly changing areas are very frequent, such effects are observable even with 8 bits per pixel and color [Hal89]. In addition, the reproduction result is affected by the distortions introduced into the color space by the approximate rotations.

To improve the performance of progressive transmission for color images, the following solutions may be possible: First, it may be necessary to abandon the requirement of final lossless reproduction. This will allow the use of a wider range of color spaces. Also, the importance and semantics of color differ greatly for different applications, and it may not be possible to find one coding method that is suited for all applications.

Second, the gradual introduction of color into a gray scale image has to be studied. It seems desirable to introduce basic colors, like red, yellow, green, and blue, first. If the NTSC color system is used, the colors that appear on the screen first are orange, light blue, violet, and yellow-green, which leads to a very ugly image.

Third, it may be possible to merge the two color component hierarchies to a single hierarchy to synchronize the transmission of color information. Algorithms that construct a color hierarchy adapted to an image have been developed in connection with the quantization of color images to a restricted number of color values (usually 256) [Hec82]. This has the additional effect that dependence between the different color coordinates is used for compression.

Fourth, the color components have to be smoothed in the initial stages of the reproduction, as proposed for gray scale images in Section 5.1. Testing all these improvements and their combinations, it should be possible to achieve satisfactory color image quality at 2 bits per pixel (compare [Cam86]) and less.

7.3 Image Sequences

Using progressive transmission methods for video compression is basically a contradiction. Progressive transmission uses the time dimension to gradually improve an image, whereas video uses the time dimension to display an image sequence. However, as already pointed out in Subsection 3.1.3, the progressiveness of an image compression method allows fixed rate coding with a basically variable rate coding method. As variable rate coding generally achieves better compression, and a fixed rate per frame is necessary for smooth video decoding, progressive coding techniques are well suited for video coding.

In addition, the flexibility of progressive transmission techniques is very useful when transmitting sequences of images at a frame rate lower than the video frame rate. Possible applications include videophones, teleconferencing, and browsing through subsequent cross-sectional images in medicine, geology and other fields.

A sequence of similar images is usually coded as a sequence of differential images. In most cases, blocks of the image are separated into two kinds: Slightly changed blocks, which are coded by comparing them with the same block of the previous image (interframe coding), and greatly changed blocks, which are coded independently (intraframe coding). More elaborate algorithms estimate the motion of different parts of the image between two frames. Hereby, quadtrees are already used to concentrate coding on the areas of largest activity [Str90]. A combination of such an algorithm with the BC quadtree could lead to very good results.

The method of image compression and progressive transmission proposed in this paper is well suited for interframe coding using differential images. As differential images have a heavily biased histogram, the gray scale hierarchy is best changed in a way similar to that described in Section 5.3. Image changes are best reproduced in a way similar to the black-to-white reproduction method described in Subsection 5.1.1. This results in a monotone change from the old to the new image, which is important to eliminate flickering.

8

Conclusions

8.1 Summary

The main contribution of this thesis is a new method for image compression and progressive transmission. The results obtained show that the new method performs about as well as considerably more complex coding methods like vector coding or transform coding, and better than existing methods of comparable complexity.

The examination of the mathematical base and the requirements of present and future applications of image compression and progressive transmission led to the conclusion that the unification of progressive transmission and image compression is both highly desirable for practical reasons and theoretically possible in many cases. In particular, it was shown that in the high rate case, progressive transmission can perform at only 1.8753 db below any pointwise optimal transmission.

The requirements for an ideal compression method including progressive transmission have been analyzed, and the new method presented in this thesis in many aspects fulfills these requirements better than existing methods. It is fully progressive from the initial very low rates to the point of lossless compression while maintaining high compression rates and very low complexity. It is flexible and so can easily be adapted to different kinds of images, the needs of the user, and the possible bottlenecks in sender, transmission line, and receiver. The efficiency of the new method has been confirmed by a number of experiments of various types.

The new method has been developed and analyzed using a number of new principles, like bitwise condensation, the combined increase of spatial and gray scale resolution, sampling and quantization using intervals, and traces incorporating a considerable amount of neighborhood information in an efficient

way. These principles may prove to be of value by themselves in various areas related to data compression and image processing.

8.2 Directions for Future Work

The work in this thesis can be continued and extended in several ways. One way is to examine the various possibilities for additional performance improvements. They include the definition of components and averages fully based on traces, the development of models for trace averages and probabilities, the use of prediction to enhance these models, and the combination of the new method with other methods of data compression. Considerable improvements in performance can be expected, but care has to be taken that there is not too large an increase in complexity.

The new approach to sampling and quantization, represented by the HSRQ (hierarchical sampling restricted quantization) principle, can also lead to many interesting research problems. These include the theoretical analysis of the limits of performance of HSRQ for different source models, the investigation of various combinations of splitting and reduction, the reconstruction of smooth functions from the intervals, and the use of HSRQ for image processing applications other than pure compression. Some interesting aspects are the connection of HSRQ with morphological approaches and image information measures. Another aspect is the fact that HSRQ restricts the absolute deviation of any reconstruction, which means that the absolute error of further operations can be bounded exactly.

The new method leads to very good results for the compression of gray scale images, whereas for color images, the results are not yet satisfactory. The efficient use of progressive transmission for color images is an important, but difficult problem, as it is directly related to the problem of what information the human visual system obtains from color. Another such problem is how the dynamic improvement of the image during progressive transmission affects recognition and working efficiency. The application of the new method to image sequences and higher- and lower-dimensional data should also be studied.

A completely different, but not less interesting task is the implementation of the new method, in a number of suitable variants, for a single application or a heterogeneous environment, both in software and hardware.

Appendices

Appendix A: Component Sequences

The component sequences are listed separating different components by commas. For each individual component, the spatial hierarchy level s is followed by the gray scale hierarchy level c.

Component Sequence Used for Figure 4.8

This component sequence was optimized by hand for the center value reproduction.

0 1, 0 2, 0 3, 0 4, 0 5, 0 6, 0 7, 0 8, 1 1, 1 2, 1 3, 1 4, 1 5, 1 6, 1 7, 1 8, 2 1, 2 2, 2 3, 2 4, 2 5, 2 6, 2 7, 2 8, 3 1, 3 2, 3 3, 3 4, 3 5, 3 6, 3 7, 3 8, 4 1, 4 2, 4 3, 4 4, 4 5, 4 6, 4 7, 4 8, 5 1, 6 1, 5 2, 5 3, 5 4, 5 5, 5 6, 5 7, 5 8, 6 2, 7 1, 7 2, 8 1, 6 3, 6 4, 6 5, 6 6, 6 7, 6 8, 8 2, 7 3, 8 3, 7 4, 7 5, 7 6, 7 7, 7 8, 8 4, 8 5, 8 6, 8 7, 8 8

Component Sequence *o2*

See Figure 6.1 for an explanation of how this and the next two components have been constructed.

0 1, 0 2, 1 1, 0 3, 1 2, 2 1, 0 4, 1 3, 2 2, 0 5, 3 1, 1 4, 2 3, 0 6, 3 2, 1 5, 4 1, 2 4, 0 7, 3 3, 1 6, 4 2, 2 5, 0 8, 5 1, 3 4, 1 7, 4 3, 2 6, 5 2, 3 5, 1 8, 6 1, 4 4, 2 7, 5 3, 3 6, 6 2, 4 5, 2 8, 7 1, 5 4, 3 7, 6 3, 4 6, 7 2, 5 5, 3 8, 6 4, 4 7, 7 3, 5 6, 6 5, 4 8, 8 1, 7 4, 5 7, 6 6, 8 2, 7 5, 5 8, 6 7, 8 3, 7 6, 6 8, 8 4, 7 7, 8 5, 7 8, 8 6, 8 7, 8 8

Component Sequence *o3*

0 1, 1 1, 0 2, 2 1, 1 2, 0 3, 3 1, 2 2, 1 3, 0 4, 4 1, 3 2, 2 3, 1 4, 0 5, 5 1, 4 2, 3 3, 2 4, 1 5, 0 6, 6 1, 5 2, 4 3, 3 4, 2 5, 1 6, 0 7, 7 1, 6 2, 5 3, 4 4, 3 5, 2 6, 1 7, 0 8, 7 2, 6 3, 5 4, 4 5, 3 6, 2 7, 1 8, 8 1, 7 3, 6 4, 5 5, 4 6, 3 7, 2 8, 8 2, 7 4, 6 5, 5 6, 4 7, 3 8, 8 3, 7 5, 6 6, 5 7, 4 8, 8 4, 7 6, 6 7, 5 8, 8 5, 7 7, 6 8, 8 6, 7 8, 8 7, 8 8

Component Sequence *o4*

0 1, 1 1, 2 1, 0 2, 3 1, 1 2, 4 1, 2 2, 0 3, 5 1, 3 2, 1 3, 6 1, 4 2, 2 3, 0 4, 7 1, 5 2, 3 3, 1 4, 6 2, 4 3, 2 4, 0 5, 8 1, 7 2, 5 3, 3 4, 1 5, 6 3, 4 4, 2 5, 0 6, 8 2, 7 3, 5 4, 3 5, 1 6, 6 4, 4 5, 2 6, 0 7, 8 3, 7 4, 5 5, 3 6, 1 7, 6 5, 4 6, 2 7, 0 8, 8 4, 7 5, 5 6, 3 7, 1 8, 6 6, 4 7, 2 8, 8 5, 7 6, 5 7, 3 8, 6 7, 4 8, 8 6, 7 7, 5 8, 6 8, 8 7, 7 8, 8 8

Component Sequence *o5*

This component sequence was optimized, starting with sequence *o3*, to obtain the highest PSNR in the initial stages of transmission (differences to *o3* in italics).

0 1, 1 1, 0 2, 2 1, 1 2, 0 3, 3 1, 2 2, 1 3, 0 4, 4 1, 3 2, 2 3, 1 4, 0 5, 5 1, 4 2, 3 3, 2 4, 1 5, 0 6, 6 1, 5 2, 4 3, 3 4, 2 5, 1 6, 0 7, 7 1, *8 1*, 6 2, 5 3, 4 4, 3 5, 2 6, 1 7, 0 8, 6 3, 5 4, 4 5, 3 6, 2 7, 1 8, 7 2, 7 3, 6 4, 5 5, 4 6, 3 7, 2 8, 7 4, 6 5, 5 6, 4 7, 3 8, 7 5, 6 6, *7 6*, *8 2*, *8 3*, 5 7, 4 8, 8 4, 6 7, 5 8, 8 5, 7 7, 6 8, 8 6, 7 8, 8 7, 8 8

Appendix B: Results for Image "Lena"

For all the images shown below, the component sequence *o2*, average value reproduction, and arithmetic compression have been used. To compensate for the larger size of the image, the component sequence has been applied to each of the four quadrants of the images.

Figure B.1. "Lena" at 1/64 and 1/32 bits per pixel

Figure B.2. "Lena" at 1/16 and 1/8 bits per pixel

Figure B.3. "Lena" at 1/4 and 1/2 bits per pixel

Figure B.4. "Lena" at 1 and 1.5 bits per pixel

Figure B.5. "Lena" at 2 and 8 bits per pixel
(the image on the right is the original)

Appendix C: Images Used in Face Recognition Experiments

Figure C.1. Senior students of the Department of Information Science

Figure C.2. Senior students of the Department of Information Science

Figure C.3. Senior students of the Department of Information Science
and the teaching assistant

Figure C.4. Famous personalities

Figure C.5. Famous personalities and the author

Bibliography

[Ahm74] Ahmed, N., Natarajan, T., and Rao, K. R. Discrete cosine transform. *IEEE Trans. Comput.* **C-23**, 1 (Jan. 1974), 90-93.

[Arp88] Arps, R. B., Truong, T. K., Lu, D. J., Pasco, R. C., and Friedman, T. D. A multi-purpose VLSI chip for adaptive data compression of bilevel images. *IBM J. Res. Develop.* **32**, 6 (Nov. 1988), 717-726.

[Bad86] Badrkhan, Kamiran S. *Video Systems: Television Principles and Servicing*. John Wiley & Sons, New York, 1986.

[Bar88] Barnsley, Michael F., and Sloan, Alan D. A better way to compress images. *BYTE* **13**, 1 (Jan. 1988), 215-223.

[Ben75] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM* **18**, 9 (Sept. 1975), 509-517.

[Ber71] Berger, Toby. *Rate Distortion Theory*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971.

[Berm86] Bergman, Larry, Fuchs, Henry, Grant, Eric, and Spach, Susan. Image rendering by adaptive refinement. *Computer Graphics* (Proc. SIGGRAPH) **20**, 4 (August 1986), 29-37.

[Bla72] Blahut, Richard E. Computation of channel capacity and rate-distortion functions. *IEEE Trans. Inform. Theory* **IT-18**, 4 (July 1972), 460-473.

[Bru87] Bruckstein, Alfred M. On Optimal Image Digitization. *IEEE Trans. Acoust., Speech, Signal Processing* **ASSP-35**, 4 (April 1987), 553-555.

[Bur83] Burt, Peter J., and Adelson, Edward H. The Laplacian pyramid as a compact image code. *IEEE Trans. Commun.* **COM-31**, 4 (April 1983), 532-540.

[Burn83] Burton, Warren F., and Kollias, J. G. Comment on 'The explicit quad tree as a structure for computer graphics'. *The Computer Journal* **26**, 2 (March 1983), 118.

[Cam86] Campbell, Graham, DeFanti, Thomas A., Frederiksen, Jeff, Joyce, Stephen A., Leske, Lawrence A., Lindberg, John A., and Sandin, Daniel J. Two bit/pixel full color encoding. *Computer Graphics* (Proc. SIGGRAPH) **20**, 4 (Aug. 1986), 215-223.

[Cha89] Chang, Shi-Kuo. *Principles of Pictortial Information Systems Design.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[CheD87] Chen, David Shi, and Allenbach, Jan P. Analysis of error in reconstruction of two-dimensional signals from irregularly spaced samples. *IEEE Trans. Acoust., Speech, Signal Processing* **ASSP-35**, 2 (Feb. 1987), 173-180.

[CheT90] Chen, Ting-Chung. A lattice vector quantization using a geometric decomposition. *IEEE Trans. Commun.* **COM-38**, 5 (May 1990), 704-714.

[Chi89] Chin, Francis, Choi, Andrew, and Luo, Yuhua. An optimal generating kernel for image pyramids by linear fitting. *Proceedings of the 1989 International Symposium on Computer Architecture and Digital Signal Processing*, 11-14 Oct. 1989, Hong Kong, pp. 612-617.

[Con82] Conway, J. H., and Sloane, N. J. A. Voronoi regions of lattices, second moments of polytopes, and quantization. *IEEE Trans. Inform. Theory* **IT-28**, 2 (March 1982), 211-226.

[Dau88] Daugman, John G. Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression. *IEEE Trans. Acoust., Speech, Signal Processing* **ASSP-36**, 7 (July. 1988), 1169-1179.

[Del79] Delp, E. J., and Mitchell, O. R. Image compression using block truncation coding. *IEEE Trans. Commun.* **COM-27**, 9 (Sept. 1979), 1335-1342.

[Dre87] Dreizen, Howard M. Content-driven progressive transmission of gray-scale images. *IEEE Trans. Commun.* **COM-35**, 3 (March 1987), 289-296.

[Dub86] Dubois, E., and Moncet, J. L. Encoding and progressive transmission of still pictures in NTSC composite format using transform domain methods. *IEEE Trans. Commun.* **COM-34**, 3 (March 1986), 310-319.

[Dür88a] Dürst, Martin J., and Kunii, Tosiyasu L. Integrated polytrees: A generalized model for integrating spatial decomposition and boundary representation. Technical Report 88-002, Department of Information Science, Faculty of Science, University of Tokyo, Jan. 1988.

[Dür88b] Dürst, Martin J., and Kunii, Tosiyasu L. A proposal for polytree generalization. In *Proceedings of the 36th National Convention of the Information Processing Society of Japan* (Yokohama, Japan, March 16-18). Information Processing Society of Japan, Tokyo, Japan, 1988, pp. 2143-2144 (in Japanese).

[Dür88c] Dürst, Martin J., and Kunii, Tosiyasu L. Error-free image compression using gray scale quadtrees. Technical Report 88-024, Department of Information Science, Faculty of Science, University of Tokyo, Dec. 1988.

[Dür89a] Dürst, Martin J., and Kunii, Tosiyasu L. Integrated polytrees: A generalized model for the integration of spatial decomposition and boundary representation. In *Theory and Practice of Geometric Modeling*, W. Strasser and H.-P. Seidel, Eds. Springer-Verlag, Berlin, 1989, pp. 329-348. (Also available as Technical Report 89-016, Department of Information Science, Faculty of Science, University of Tokyo, May 1989.)

[Dür89b] Dürst, Martin J., and Kunii, Tosiyasu L. Vertex classification using the convex hull on a sphere. In *Proceedings of the International Workshop on Discrete Algorithms and Complexity* (Fukuoka, Japan, Nov. 20-22) (also *IPSJ Technical Report* 89-AL-12). Information Processing Society of Japan, Tokyo, Japan, 1989, pp. 25-32. (Also available as *IEICE Technical Report* **89**, 309 (paper Nr. COMP89-53). Institute of Electronics, Information and Communication Engineers, Tokyo, Japan, 1989, pp. 25-31.)

[Dür89c] Dürst, Martin J., and Kunii, Tosiyasu L. Error-free image compression with gray scale quadtrees and its optimization. In *Proceedings of the International Workshop on Discrete Algorithms and Complexity* (Fukuoka, Japan, Nov. 20-22) (also *IPSJ Technical Report* 89-AL-12). Information Processing Society of Japan, Tokyo, Japan,

1989, pp. 115-121. (Also available as *IEICE Technical Report* **89**, 310 (paper Nr. COMP89-66). Institute of Electronics, Information and Communication Engineers, Tokyo, Japan, 1989, pp. 25-31.)

[Dür89d] Dürst, Martin J., and Kunii, Tosiyasu L. Progressive transmission based on bitwise condensed quadtrees. In *Proceedings of the 3rd Sapporo International Computer Graphics Symposium* (Sapporo, Japan, Nov. 28-30), Yoshinao Aoki, Ed. Executive Committee, Sapporo International Computer Graphics '89, Sapporo, Japan, 1989, pp. 84-89 (in Japanese). (Slightly expanded version (in English) available as Technical Report 90-006, Department of Information Science, Faculty of Science, University of Tokyo, Feb. 1990.)

[Dür90a] Dürst, Martin J., Bieri, Hanspeter, and Kunii, Tosiyasu L. Two linear time algorithms for the Euler number of hierarchically represented digital pictures. Technical Report 90-007, Department of Information Science, Faculty of Science, University of Tokyo, Feb. 1990.

[Dür90b] Dürst, Martin J., and Kunii, Tosiyasu L. Methods for the efficient storage and manipulation of spatial geological data. To appear in *Three-Dimensional Modelling with Geoscientific Information Systems*, A. K. Turner, Ed. Kluver Academic Publishers, Dorndrecht, The Netherlands, 1990.

[Dür91] Dürst, Martin J., and Kunii, Tosiyasu L. Progressive transmission increasing both spatial and gray scale resolution. *Proceedings of the International Conference on Multimedia Information Systems* (Singapore, Jan. 16-18, 1991), to be published.

[End87] Endoh, Toshiaki, and Yamazaki, Yasuhiro. Progressive coding scheme for multi-level images. *Trans. Instit. of Electr., Inform. and Commun. Eng.* **J70-B**, 1 (Jan. 1987), 105-114 (in Japanese).

[Equ89] Equitz, William H. R. *Successive Refinement of Information*. PhD thesis, Dept. of Electrical Engineering, Stanford University, 1989.

[Equ90] Equitz, William H. R., and Cover, Thomas. Successive refinement of information. Submitted to *IEEE Trans. Inform. Theory*. (Also available as IBM Research Report RJ 7319, IBM Almaden Research Center, San Jose, 1990).

[Fan84] Fanelli, A. M., and Marangelli, B. Data compression for progressive transmission of screened pictures. *IEEE Trans. Syst., Man, Cybern.* **SMC-14**, 3 (May/June 1984), 519-524.

[Far90] Farrelle, Paul Michael. *Recursive Block Coding for Image Data Compression*. Springer-Verlag, New York, 1990.

[Fin74] Finkel, R. A., and Bentley, J. L. Quad trees - a data structure for retrieval on composite keys. *Acta Informatica* **4**, 1 (Nov. 1974), 1-9.

[Fol90] Foley, James D., van Dam, Andries, Feiner, Steven K., and Hughes, John F. *Computer Graphics: Principle and Practice* (Second Edition). Addison-Wesley, Reading, MA, 1990.

[Fra80] Frank, Amalie J., Daniels, J. D., and Unangst, Diane R. Progressive image transmission using a growth-geometry coding. *Proc. IEEE* **68**, 7 (July 1980), 897-909.

[Fuc89] Fuchs, Henry, Poulton, John, Eyles, John, Greer, Trey, Goldfeather, Jack, Ellsworth, David, Molnar, Steve, Turk, Greg, Tebbs, Brice, and Israel, Laura. Pixel-planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories. *Computer Graphics* (Proc. SIGGRAPH) **23**, 3 (July 1989), 79-88.

[Gar82] Gargantini, Irene. An effective way to represent quadtrees. *Commun. ACM* **25**, 12 (Dec. 1982), 905-910.

[Ger79] Gersho, Allen. Asymptotically optimal block quantization. *IEEE Trans. Inform. Theory* **IT-25**, 4 (July 1979), 373-380.

[Gia88] Giardina, Charles R., and Dougherty, Edward R. *Morphological Methods in Image and Signal Processing*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[Gra90] Gray, Robert M. *Source Coding Theory*. Kluver Academic Publishers, Boston, 1990.

[Gri87] Griswold, N. C., Halverson, D. R., and Wise, G. L. A note on adaptive block truncation coding for image processing. *IEEE Trans. Acoust., Speech, Signal Processing* **ASSP-35**, 8 (Aug. 1987), 1201-1203.

[Gro83] Grosky, William I., and Jain, Ramesh. Optimal quadtrees for image segments. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-5**, 1 (Jan. 1983), 77-83.

[Gui83] Guibas, Leo, Ramshaw, Lyle, and Stolfi, Jorge. A kinetic framework for computational geometry. In *Proceedings of the 24th Annual IEEE Symposium on the Foundations of Computer Science*. IEEE, 1983, pp. 100-111.

[Had57] Hadwiger, H. *Vorlesungen über Inhalt, Oberfläche und Isoperimetrie.* Springer-Verlag, Berlin, 1957.

[Hal89] Hall, Roy. *Illumination and Color in Computer Generated Imagery.* Springer-Verlag, New York, 1989.

[Har73] Harmon, Leon D. The recognition of faces. *Scientific American* **229**, 5 (Nov. 1973), 71-82.

[Hec82] Heckbert, Paul. Color image quantization for frame buffer display. *Computer Graphics* (Proc. SIGGRAPH) **16**, 3 (July 1982), 297-307.

[Hil83] Hill, F. S., Jr., Walker, Sheldon, Jr., and Gao, Fuwen. Interactive image query system using progressive transmission. *Computer Graphics* (Proc. SIGGRAPH) **17**, 3 (July 1983), 323-330.

[Hoa85] Hoare, C. A. R. *Communicating Sequential Processes.* Prentice Hall International, Englewood Cliffs, NJ, 1985.

[Hof86] Hofmann, William D., and Troxel, Donald E. Making progressive transmission adaptive. *IEEE Trans. Commun.* **COM-34**, 8 (Aug. 1986), 806-813.

[Ho88] Ho, Yo-Sung, and Gersho, Allen. Variable-rate multi-stage vector quantization for image coding. *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, ICASSP'88 (1988), 1156-1159.

[Hun79] Hung, Stephen H. Y. A generalization of DPCM for digital image compression. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-1**, 1 (Jan. 1979), 100-109.

[Hua65] Huang, Thomas S. PCM picture transmission. *IEEE Spectrum* **2**, 12 (Dec. 1965), 57-63.

[IEE85] *Proc. IEEE* (Special issue on Visual Communications Systems) **73**, 4 (April 1985).

[Jac83] Jackins, Chris L., and Tanimoto, Steven L. Quad-trees, oct-trees, and K-trees: A generalized approach to recursive decomposition of Euclidean space. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-5**, 5 (Sept. 1983), 533-539.

[Jai81] Jain, Anil K. Image data compression: A review. *Proc. IEEE* **69**, 3 (March 1981).

[Jay84] Jayant, N. S., and Peter Noll. *Digital Coding of Waveforms.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[Jons79] Jones, Douglas Samual. *Elementary Information Theory.* Oxford University Press, Oxford, 1979.

[Jonw88] Jones, Douglas W. Application of splay trees to data compression. *Commun. ACM* **31**, 8 (Aug. 1988), 996-1007.

[Kaw80] Kawaguchi, E., and Endo, T. On a method of binary-picture representation and its application to data compression. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-2**, 1 (Jan. 1980), 27-35.

[Kaw83] Kawaguchi, Eiji, Endo, Tsutomu, and Matsunaga, Jun-ichi. Depth-first picture expression viewed from digital picture processing. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-5**, 4 (July 1983), 373-384.

[Kli76] Klinger, Allen, and Dyer, Charles R. Experiments on picture representation using regular decomposition. *Comput. Graphics Image Process.* **5**, 1 (March 1976), 68-105.

[Kli89] Klinger, Allen. Oral communication with the author. Los Angeles, Dec. 7, 1989.

[Kni86] Kunii, Tosiyasu L., Fujishiro, Issei, and Mao, Xiaoyang. G-quadtree: A hierarchical representation of gray-scale digital images. *The Visual Computer* **2**, 4 (Aug. 1986), 219-226.

[Kno80] Knowlton, Ken. Progressive transmission of grey-scale and binary pictures by simple, efficient, and lossless encoding schemes. *Proc. IEEE* **68**, 7 (July 1980), 885-895.

[Knt78] Kunt, Murat. Source coding of X-ray pictures. *IEEE Trans. Biomed. Eng.* **BME-25**, 2 (March 1978), 121-138.

[Knt85] Kunt, Murat, Ikonomopoulos, Athanassios, and Kocher, Michael. Second-Generation Image-Coding Techniques. *Proc. IEEE* **73**, 4 (April 1985), 549-574.

[LeeM90] Lee, Meong Won, Kunii, Tosiyasu L., and Dürst, Martin J. Motion comparison in computer animation. In *Computer Animation '90* (Proceedings of Computer Animation '90, Geneva, Switzerland, April 25-27), Nadia Magnenat-Thalmann and Daniel Thalmann, Eds. Springer-Verlag, Tokyo, 1990, pp. 191-206. (Also available as Technical Report 90-001, Department of Information Science, Faculty of Science, University of Tokyo, Jan. 1990.)

[Lem86] Lempel, Abraham, and Ziv, Jacob. Compression of two-dimensional data. *IEEE Trans. Inform. Theory* **IT-32**, 1 (Jan. 1986), 2-8.

[Li82] Li, Ming, Grosky, William I., and Jain, Ramesh. Normalized quadtrees with respect to translations. *Comput. Graphics Image Process.* **20**, 1 (Sept. 1982), 72-81.

[Lin90] Lindeberg, Tony. Scale-space for discrete signals. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-12**, 3 (March 1990), 234-254.

[Loh82] Lohscheller, Herbert. *Einzelbildübertragung mit wachsender Auflösung*. Dissertation, Fakultät Elektrotech., RWTH Achen, Achen, West Germany, 1982.

[Loh83] Lohscheller, Herbert. Vision adapted progressive image transmission. In *Signal Processing II: Theories and Applications* (Proceedings of EUSIPCO-83, Second European Signal Processing Conference, Erlangen, West Germany, Sept. 12-16. 1983), Schüssler, H. W., Ed. Elsevier Science Publishers B. V., North-Holland, 1983, 191-194.

[Loh84] Lohscheller, Herbert. A subjectively adapted image communication system. *IEEE Trans. Commun.* **COM-32**, 12 (Dec. 1984), 1316-1322.

[Loo89] Lookabaugh, Tom D., and Gray, Robert M. High-resolution quantization theory and the vector quantizer advantage. *IEEE Trans. Inform. Theory* **IT-35**, 5 (Sept. 1989), 1020-1033.

[Mal89] Mallet, Jean-Laurent. Discrete Smooth Interpolation. *ACM Trans. Graph.* **8**, 2 (April 1989), 121-144.

[Mann74] Mannos, James L., and Sakrinson, David J. The effects of a visual fidelity criterion on the encoding of images. *IEEE Trans. Inform. Theory* **IT-20**, 4 (July 1974), 525-536.

[Mao87] Mao, Xiaoyang, Kunii, Tosiyasu L., Fujishiro, Issei, and Noma, Tsukasa. Hierarchical representations of 2D/3D gray-scale images and their 2D/3D two-way conversion. *IEEE Comp. Graph. and Appl.* **7**, 12 (Dec. 1987), 37-44.

[Mao90a] Mao, Xiaoyang. *A Hierarchical Data Representation-Oriented Image Processing System*. PhD thesis, Dept. of Information Science, Faculty of Science, The University of Tokyo, 1990.

[Mao90b] Mao, Xiaoyang, Fujishiro, Issei, and Kunii, Tosiyasu L. A translucent display algorithm for G-octree represented gray-scale images. *The Journal of Visualization and Computer Animation* **1**, 1 (August 1990), 22-25.

[Mey86] Meyer, Gary W. Tutorial on color science. *The Visual Computer* **2**, 5 (Sept. 1986), 278-290.

[Min03] Minkowski, H. Volumen und Oberfläche. *Math. Ann.* **57** (1903), 447-495.

[Mof90] Moffat, Alistair. Linear time adaptive arithmetic coding. *IEEE Trans. Inform. Theory* **IT-36**, 2 (March 1990), 401-406.

[Nas88] Nasrabadi, Nasser M., and King, Robert A. Image coding using vector quantization: A review. *IEEE Trans. Commun.* **COM-36**, 8 (Aug. 1988), 957-971.

[Nga84] Ngan, King N. Image display techniques using the cosine transform. *IEEE Trans. Acoust., Speech, Signal Processing* **ASSP-32**, 1 (Feb. 1984), 173-177.

[NZZ90] Extreme Kompression von Videosignalen. *Neue Zürcher Zeitung* (Swiss daily newspaper), Wednesday, April 18, 1990, p. 57.

[Oli83] Oliver, M. A., and Wiseman, N. E. Operations on quadtree encoded images. *The Computer Journal* **26**, 1 (Jan. 1983), 83-91.

[Ono79] Onoe, Morio, Sakauchi, Masao, and Inamoto, Yasushi. *SIDBA Standard Image Data Base*. MIPC Report 79-1, Multidimensional Image Processing Center, Institute of Industrial Science, University of Tokyo (March 1979).

[Pea90] Pearlman, William A. Adaptive cosine transform image coding with constant block distortion. **COM-38**, 5 (May 1990), 698-703.

[Pel89] Peleg, Shmuel, Werman, Michael, and Rom, Hillel. A unified approach to the change of resolution: Space and gray-level. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-11**, 7 (July 1989), 739-742.

[Pen88] Pennebaker, W. B., Mitchell, J. L., Langdon, G. G., Jr., and Arps, R. B. An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder. *IBM J. Res. Develop.* **32**, 6 (Nov. 1988), 717-726.

[Per88] Perkins, Michael G. A comparison of the Hartley, Cas-Cas, Fourier, and discrete cosine transforms for image coding. *IEEE Trans. Commun.* **COM-36**, 6 (June 1988), 758-761.

[Pru85] Prusinkiewicz, Przemyslaw, and Christopher, Mark. Hologram-like transmission of pictures. In *Computer-Generated Images: The State of the Art* (Proc. Graphics Interface '85), Magnenat-Thalmann, Nadia, and Thalmann, Daniel, Eds., Springer-Verlag, Tokyo, 1985, 125-134.

[Ris90] Riskin, Eve Ann. *Variable Rate Vector Quantization of Images.* Dissertation, PhD thesis, Dept. of Electrical Engineering, Stanford University, 1990.

[Ros82] Rosenfeld, Azriel, and Kak, Avinash C. *Digital Picture Processing (Second Edition), Volume 1.* Academic Press, New York, 1982.

[Sak77] Sakrinson, David J. On the role of the observer and a distortion measure in image transmission. *IEEE Trans. Commun.* **COM-25**, 11 (Nov. 1977), 1251-1267.

[Sam84] Samet, Hanan. The quadtree and related hierarchical data structures. *ACM Comput. Surv.* **16**, 2 (June 1984), 187-260.

[Sam85a] Samet, Hanan, and Tamminen, Markku. Computing geometric properties of images represented by linear quadtrees. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-7**, 2 (March 1985), 229-240.

[Sam85b] Samet, Hanan. Data structures for quadtree approximation and compression. *Commun. ACM* **28**, 9 (Sept. 1985), 973-993.

[Sam88a] Samet, Hanan and Webber, Robert E. Hierarchical data structures and algorithms for computer graphics, Part I: Fundamentals. *IEEE Comp. Graph. and Appl.* **8**, 3 (May 1988), 48-68.

[Sam88b] Samet, Hanan and Webber, Robert E. Hierarchical data structures and algorithms for computer graphics, Part II: Applications. *IEEE Comp. Graph. and Appl.* **8**, 4 (July 1988), 59-75.

[Sam88c] Samet, Hanan, and Tamminen, Markku. Efficient component labelling of images of arbitrary dimension represented by linear bintrees. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-10**, 4 (July 1988), 579-586.

[Sam89] Samet, Hanan, and Webber, Robert E. A comparison of the space requirements of multi-dimensional quadtree-based file structures. *The Visual Computer* **5**, 6 (Dec. 1989), 349-359.

[Sam90a] Samet, Hanan. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

[Sam90b] Samet, Hanan. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS.* Addison-Wesley, Reading, MA, 1990.

[San84] Sanz, Alberto, Muñoz, Carlos, and García, Narciso. Approximation quality improvement techniques in progressive image transmission. *IEEE J. Select. Areas Commun.* **SAC-2**, 2 (March 1984), 359-373.

[Sek89] Sekine, M. A Design of progressive transmission error-free image coding. Senior Thesis, Dept. of Information Science, Faculty of Science, University of Tokyo (in Japanese).

[Sha59] Shannon, C. E. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record, Part 4* (1959), 142-163.

[She90] Sheinwald, Dafna, Lempel, Abraham, and Ziv, Jacob. Two-dimensional encoding by finite-state encoders. *IEEE Trans. Commun.* **COM-38**, 3 (March 1990), 341-347.

[Slo79] Sloan, Kenneth R., Jr., and Tanimoto, Steven L. Progressive refinement of raster images. *IEEE Trans. Comput.* **C-28**, 11 (Nov. 1979), 871-874.

[Str90] Strobach, Peter. Tree-structured scene adaptive coder. *IEEE Trans. Commun.* **COM-38**, 4 (April 1990), 477-486.

[Tak84] Takigawa, Kei. Fast progressive reconstruction of a transformed image. *IEEE Trans. Inform. Theory* **IT-30**, 1 (Jan. 1984), 111-117.

[Tam84a] Tamminen, Markku. Comment on quad- and octtrees. *Commun. ACM* **27**, 3 (March 1984), 248-249.

[Tam84b] Tamminen, Markku. Encoding pixel trees. *Comput. Vision Graphics Image Process.* **28**, 1 (Oct. 1984), 44-57.

[Tan79] Tanimoto, Steven L. Image transmission with gross information first. *Comput. Graphics Image Process.* **9**, 1 (Jan. 1979), 72-76.

[Tzo86] Tzou, Kou-Hu, and Elnahas, Sharaf E. Bit-sliced progressive transmission and reconstruction of transformed images. *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, ICASSP'86 (1986), 533-536.

[Tzo87] Tzou, Kou-Hu. Progressive image transmission: a review and comparison of techniques. *Optical Engineering* **26**, 7 (July 1987), 581-589.

[Vai87] Vaisey, D. Jaques, and Gersho, Allen. Variable block-size image coding. *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, ICASSP'87 (1987), 1051-1054.

[Wan88] Wang, Limin, and Goldberg, Morris. Progressive Image Transmission by Transform Coefficient Residual Error Quantization. *IEEE Trans. Commun.* **COM-36**, 1 (Jan. 1988), 75-87.

[Wel84] Welch, Terry A. A technique for high-performance data compression. *IEEE Computer* **17**, 6 (June 1984), 8-19.

[Wes88] Westerink, Peter H., Boekee, Dick E., Biemond, Jan, and Woods, John W. Subband coding of images using vector quantization. *IEEE Trans. Commun.* **COM-36**, 6 (June 1988), 713-719.

[Wil84] Wilson, Roland. Quad-tree predictive coding: A new class of image data compression algorithms. *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, ICASSP'84 (1984), 29.3.1-4.

[Wit87] Witten, Ian H., Neal, Radford M., and Cleary, John G. Arithmetic coding for data compression. *Commun. ACM* **30**, 6 (June 1987), 520-540.

[Woo82] Woodwark, J. R. The explicit quad tree as a structure for computer graphics. *The Computer Journal* **25**, 2 (March 1982), 235-238.

[Woo84] Woodwark, J. R. Compressed quad trees. *The Computer Journal* **27**, 3 (May 1984), 225-229.

[Wu82] Wu, Angela Y., Hong, Tsai-Hong, and Rosenfeld, Azriel. Threshold selection using quadtrees. *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-4**, 1 (Jan. 1982), 90-94.

[Yan77] Yan, Johnson K., and Sakrinson, David J. Encoding of images based on a two-component source model. *IEEE Trans. Commun.* **COM-25**, 11 (Nov. 1977), 1315-1322.

[Yao82] Yao, F. Frances. Speed-up in dynamic programming. *SIAM J. Alg. Disc. Meth.* **3**, 4 (Dec. 1982), 532-540.

[Yas79] Yasuda, Yasuhiko, Takagi, Mikio, and Awano, Tomobumi. Hierarchical coding of still images. *Proc. Picture Coding Symp.* (Ipswich, England, July 79), Nr. 3.2.

[Yas80] Yasuda, Yasuhiko, Takagi, Mikio, Kato, Shigeo, and Awano, Tomobumi. Step by step image transmission and display from gross to fine information using hierarchical coding. *Trans. Instit. of Electr., Inform. and Commun. Eng.* **J63-B**, 4 (April 1980), 379-386 (in Japanese).

[Zad82] Zador, Paul L. Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE Trans. Inform. Theory* **IT-28**, 2 (March 1982), 139-149.

[Zor88] Zorpette, Glenn. Fractals: not just another pretty picture. *IEEE Spectrum* **25**, 10 (Oct. 1988), 29-31.

Index

2^d-tree 8, 34

A

Absolute error 142
Acceptable distortion 46
Adaptive coding 6, 16, 44, 76, 120
Additive distortion measure 93
Advanced coding methods 6
Aliasing 70
Alphabet 20
Analog image 3
Approximate compression 47
Arithmetic coding 102, 104-105, 108, 109
Aspect ratio 35, 67, 69, 78, 96
Attributed grammar 70
av( ) 87
Average 13, 21, 74, 96
  reproduction 84-91
Averaging 51, 62, 108

B

Bandwidth reduction 43
BC quadtree 2, 12, 53-56, 57, 58, 64, 68, 71, 77
  construction 56
BC quadtree (continued)
  example 54
  implementation 54
  indexed 59
  with parentheses 57
Berger 27
bi 109
bi() 73
Binary
  coding 101, 102, 104
    of GDF 58
  image 2, 4, 15, **17**, 78, 105, 121
  quadtree 10
  search 44
  valued condition 92
Bintree **10**, 34, 78, 96, 97
Bit 22
  allocation
    fixed 15
    incremental 16
  plane 4, 15, 61, 76, 83, 84
  selection
    subtrace 73, 88, 90, 95
    symbol **57**, 73, 76, 81, 84, 93, 95, 104
    trace 88
Bitwise
  condensation 68

Bitwise (continued)
condensed quadtree: See BC
DF 108
logical operations 56
Black hole 9, 70
Black to white reproduction 81, 109
Block
boundaries 91, 96
coding 4, 26
length 26
size 17, 32, 38
truncation coding 4, 109, 118
Blurring 83
Bottleneck 50, 83, 125
Boundary representation 9
Browsing 44
Bruckstein 45
Bulletin boards 40

C

Canonical form 3, 42, 47
Center value reproduction 84, 109
Chang 76
Channel
coder/decoder 24
coding theory 24
structure 24, 40
Chip card 44
CIEXYZ color coordinates 132
Closed application 41
Codebook: See Vector quantization
Coder 24
Coding
adaptive: See Adaptive
advanced methods 6
arithmetic: See Arithmetic coding
block coding 4, 26
block truncation 4
Coding (continued)
by bit plane 61
embedded 16
entropy 44, 104
Huffman 104
hybrid coding 6
interframe 139
intraframe 139
predictive 5, 74
run length 79
sound 67
subband coding 119
transform: See Transform
variable rate 44
Color
images 2, 130-138, 142
science 131
space 131
Common subtrace 88
Communicating sequential processes (CSP) 71
Component
efficiency 93
sequence 69, 74, 79, 90, 92
bi 109
heuristics 93-94
number of 93
o2, *o3*, *o4* 109, 143-144
o5 116, 144
restrictions on 62
size 60
Components
based on local priority 96
based on resolution 58
based on traces 95
Composite source 27
Computational geometry 9

Computer
animation 9
graphics 8, 40
vision 1
Concurrent
increase of both resolutions 80
Condensation 8, 60, 72
Cones (on retina) 131
Conditional
entropy 22
mutual information 23
probability 20
Conditions for decodability 73
Context free grammar 70
Continuous
time random process 21
valued random variable 21
Contrast sensitivity 95
Converse source coding theorem 26
Convexity 103
Cost function 45, 103
Cross-sectional images 139
Cubic
convolution 14, 91
splines 14,91
color spaces 133
Cumulative frequencies 104

D

Daugman 6, 75
Decoder 24
Decodability, conditions for 73
Decreasing cumulative frequencies 104
Delta modulation 5
Depth first expression: See DF
DF 10, 12, 57, 76
bitwise 108
for gray scale images 10
Difference 13, 74
Differential pulse code modulation: See DPCM
Digital halftoning 4
Dimensionless moment 34
Discrete
cosine transform 5, 15, 119
distortion measure 46
Fourier transform 16
source 32
time random process 21
valued random variable 21
Distortion 25
acceptable distortion 46
function 92
measure 25, 86
additive 93
discrete 46
Hamming 32
single letter 93
not recognizable 46
Dithering 4, 83
Division point 98, 100
DPCM 5, 107
Dreizen 14
Dreizen's method 78, 108, 112, 123, 127
homogeneous 108
Dual representation 12
Dynamic programming 97, 103

E

Edge
detection 6
enhancement 91
Electronic shopping 43

Embedded coding 16
Empty trace 88, 95
Endoh 14, 79
Entropy 22
  coding 44, 104
  conditional 22
  joint 22
Equitz 29, 32, 38
Ergodicity 21, 87
Erroneously transmitted image 43
Error requantization 16
Error-free compression: See Lossless compression
Euclidian norm 34
Euler number 9
Example image 7
Expectation 21
Explicit quadtree 10, 12, 54-56, 63, 97

F

Face recognition 94, 121-127, 146
False
  colors 138
  contours 85
  low frequencies 70
FFT 15
Fidelity criterion, single letter 25
Field of vision 96
Filter
  for reconstruction 92
  Gaussian 14, 51
  quadrature mirror 14
  to eliminate aliasing 70
  morphological 92
Filtering 91
Flickering 84, 85, 139
Forest of quadtrees 17
Form-oriented query 94
Fractals 6
Frame buffer 50, 55, 64, 65, 79, 81, 83, 102, 125

G

G-quadtree 2, 10-12, 77
  inefficiencies of 12
Gabor transform 6
Gauss Markov model 89
Gaussian
  filter 14, 51
  source 32
GDF 2, 12, 57-58, 68, 73, 104, 112
  binary coding 58
Generalized ring method 134
Geologic information systems 8, 129
Geometric transformations 9
Gersho 34
Global
  networks 2
  reproduction 91
Grammar 70
Gray code 10, 61, 108
Gray level reproduction 81-92, 102
Gray scale
  fonts 4
  depth first expression: See GDF
  hierarchy 11, 53
    n-ary 102
    optimization 97-104
    ternary 101
  images 2
    quadtrees for 10
  intervals 98
  level 59
  resolution 4, 8, 15, 59, 60, 77, 94, 95
Growth-geometry coding 18

H

Hadamard transform 16
Hamming distortion measure 32
Half size display 83
Hardware 16, 42, 50, 65, 67, 104, 142
Head movement 97
Heterogeneous environment 2, 16, 41, 42, 48, 50
Heuristics 8, 93
Hexagon 34
Hierarchical
  data structure 7
  intervals 67
  Sampling Restricted Quantization: See HSRQ
  subdivision 8
  vector quantization 75
Hierarchy
  gray scale 11
  spatial 7, 11
High
  rate assumption 33
  frequency 67, 69, 110
Higher dimensions 9
Histogram 5, 87, 98
Hologram 17
HSL color space 132
HSRQ 3, 65, 68-70, 74, 76, 85, 138, 142
HSV color space 132
Huang 4
Huffman coding 104, 108
Human visual system 4, 48, 95, 110, 121, 131, 138, 142
Hybrid coding 6
Hypercube 34
Hyperplane 38

I

Ideal compression algorithm 45
I.i.d. random process 21, 24
Image
  analog 3
  analysis 11
  array 12
  binary: See Binary
  canonical form 3
  characteristics 89
  color: See Color
  complexity 76
  components 58-65, 79, 90
  compression
    applications 39-40
    overview 3-7
    requirements for 45-51
  cross-sectional 139
  database 41, 43, 47, 94
  enhancement 1, 2, 4, 114
  erroneously transmitted 43
  example 7
  in business applications 40
  information measures 142
  gray scale 2
    quadtrees for 10
  "life" of 41
  medical 39
  model 87, 89
  of low quality 43
  original 3
  processing 1, 5, 8, 9, 10, 39
  relevant parts of 43
  satellite 39
  segmentation 87, 103
  sequence 2, 82, 139, 142
  stepwise improvement of 48

Image (continued)
storage 41
transmission 41
type 114, 89, 94
Implementation aspects 54, 56, 63, 67, 97, 104
Imprecise data 130
Increasing
both resolutions 62
gray scale resolution 15, 16, 48, 60
spatial resolution 13, 15, 16, 48, 61
Incremental transmission rate 38
Independence 20, 21
of sender and receiver 89
Independent
increase of both resolutions 80
Information
conditional mutual 23
measure 22
mutual 22
systems 8, 129
Initial subtrace 72, 73
Integration 1
Interframe coding 139
Interpolation 14, 79, 91
Interval 81, 92
coding 66
statistics 98
Intraframe coding 139
Irregularly spaced samples 92

J

Joint
entropy 22
probability 20

K

k-d-tree 103
Karhunen-Loève transform 5
Kawaguchi 10, 57, 58, 70, 76, 111
Klinger 10, 72
Knowlton 13, 15, 17, 35, 58, 77, 130
Kunt 4, 6

L

L'Hopital's rule 36
$L^*a^*b^*$ color space 132
$L^*u^*v^*$ color space 132
Laplacian
pyramid 14
source 32
Leaf node types 9
Learning effect 122
Lempel-Ziv compression 107
Length of parenthesis subtrace 73
Linear
dependence 5
equations 31
interpolation 14
quadtree 10
Linked list 97
Local
area networks 2
image quality 49
refinement 96
Lohscheller 16, 125
Lookabaugh 33
Lookup table: See Table lookup
Lossless
compression 2, 5, 16, 46, 58, 110
transform coding 47
Lossy compression 47

Low
frequencies 67
false 70
displays 84, 85
images 43
LZW 107, 111

M

Mach banding effect 95
Marginal
probability 20
Markov conditions 29
Mean square error 86, 93, 115
Medical images 39, 114, 120, 139
Memory
advantage 33, 38
for progressive transmission 34
chip card 44
requirements 10, 11, 12, 15, 16, 44, 50, 55, 58, 64, 72, 91, 97, 100, 102
Memoryless random process 21
Merging 8
Microeconomy 45
Minkowski (sum/addition) 27
Model of transmission system 24
Moment, second 34
Monotonously changing intensity 82
Monte Carlo simulation 89
Morphological
filters 92
methods 142
Morton sequence 56, 97, 108
Most significant bit 10, 54, 59, 60
Motion
comparison 9
estimation 139
Multimedia 1
Multiple resolution
gray scale 77
spatial 77
Mutual
information 22
information, conditional 23
Mutually exclusive
events 20
outcomes 19

N

n-ary gray scale hierarchy 102
Nat 22
Neighbor finding 8
Networks 2
Neutral bit selection symbol 73
Nonexistent gray levels 98
Nonuniform
G-quadtree 12, 77
quantization 92
sampling 92
Normalized
moment 34
quadtree 103
Not recognizable distortion 46
NTSC 15, 132
Number
of component sequences 93
of passes 38, 49
of Receivers 41
of Senders 41
Numerical errors 16

O

o2 109, 143
o3 109, 144
o4 109, 144

a5 116, 144
oc( ) 87
Octahedron,truncated 34
Octree 8, 9, 97
Oncologist 94
Open applications 42
Optimal
  component sequence 63, 90, 92-97, 116
  gray scale hierarchy 97-104, 109
  quadtree 103
  reconstruction filters 92
Orthogonal transform 5
Orthopedist 94
Overall image quality 49
Overhead of progressive transmission 13, 49

P

pa() 73
Parallel processing 67
Parenthesis subtrace 73
Pass size 17, 49
PCM 3-5, 65
Pel: See Pixel
Peak to peak signal to noise ratio 115
Peano-Hilbert curve 108
Person recognition: See Face recognition
Personal computing 40
Picture
  element: See Pixel
  information measure 76
Pixel 3
  trace 71
Point quadtree 7
Pointer quadtree 10, 12
Polytope 34
Polytree 9, 70
Prediction 14, 78, 79, 91, 105, 123, 142
Predictive
  coding 5, 74, 107
  reproduction 91
Preference function 45
Printing device 3, 51, 79
Priority 96
Probability 19
  conditional 20
  distribution 104
  joint 20, 26
  marginal 20
Progressive transmission
  algorithms 63-65
  applications of 42-45
  conceptual comparison with previous methods 77-80
  confidence into 94
  for binary images 17-18
  for higher bandwiths 43
  increasing both resolutions 62
  increasing gray scale resolution 15, 60
  increasing spatial resolution 13-14, 61
  memory advantage for 34
  methods 12-18
  of color images 131
  overhead of 13, 28, 49
  principle of 13
  requirements for 45-51
  shape advantage for 34
  space filling disadvantage 34
  traditional applications 42
  use for pure image compression 44
  using transform coding 15

Pruned tree structured vector quantization: See Vector quantization
Pseudorandom
  patterns 83
  sequence 97
PSNR 115, 124
Pulse code modulation: See PCM
Pyramid 13, 77

Q

Quadrants 8
Quadrature mirror filter 14
Quadtree 12, 77, 139
  advantages of 8
  BC: See BC quadtree
  binary quadtree 10
  disadvantages of 9
  explicit: See Explicit quadtree
  for gray scale images 10
  forest of 17
  linear quadtree 10
  node types 9
  normalized 103
  number of nodes in 9
  optimal 103
  plane sweep 8
  point quadtree 7
  pointer quadtree 10, 12
  region quadtree 8
  representations 9-10
  triangular quadtree 12
Quality improvement 93
Quantization 3, 65, 75, 78, 85, 142
  coefficient 34
  in HSRQ 67
  nonuniform 92
  scalar 38
Quantization (continued)
  vector: See Vector
Query 94

R

Radiologist 40
Random
  background reproduction 83
  field 21, 87
  matrix 21
  number generator 23
  process 21
  sequence 21
  variable 21
    continuous valued 21
    discrete valued 21
  vector 21
Rate distortion
  curve 25, 45
    computation of 26
    for progressive transmission 28
  function 25-28
    for composite source 27
    properties of 25
  theory 3
Receivers, number of 41
Recognition
  accuracy 126
  of faces 121-127
Reconstruction filters 92
Rectangle: See Aspect ratio
Recursive subdivision 7
Recursively dividable polytopes 34
Reducing spatial resolution 76
Reduction (HSRQ) 67, 142
Region
  detection 6
  quadtree 8

Regular subsampling 92
Reliability of channel 2
Remote
  control 41
  image database 43
  surveillance 43
  teaching 43
Requirement
  adaptability
    to different images 48
    to hardware bottlenecks 51
    to human visual system 48
    to needs of users 48
  continuous increase of resolution 48
  flexibility 48
  hardware independence 47
  losslessness 47
  low complexity 46
  number of passes 49
  pass size 49
  progressiveness 46
  scalability 51
  uninterrupted improvement of image 50
Reproduction
  average reproduction 84-91
  averages 86, 87, 96
  black to white 81
  center value 84
  global 91
  method 81-92, 93, 102, 109
  predictive 91
  random background 83
  white to black 82
Resolution
  gray scale: See Gray scale resolution
Resolution (continued)
  problems 9, 70
  requirements 8
  spatial: See Spatial resolution
RGB color space 131
Riskin 118
Rods (on retina) 131
Rotation 133
Rule-based approach 94
Run length coding 79

S

Samet 7, 17, 77
Sampling 3, 65, 142
  hierarchy 67
  in HSRQ 67
  kernels 92
  nonuniform 92
  strategies 76
Satellite images 39, 47
Scalability 51, 92
Scalar quantization 38
Scale space 76
Second moment 34
Segmentation 87, 103
Self-information 22
Sender
  threatened with destruction 43
  number of 41
Sequence of decisions 42
Sequential transmission 107, 123
Shape advantage 33, 38
  for progressive transmission 34
Shape-oriented query 94
Shared transmission line 50
SIDBA 59, 110

Signal
  processing 5
  to noise ratio 115
Single letter
  fidelity criterion 25
  distortion measure 93
Size
  of passes 49
  of trace tree 72
SKF 12
Slow transmission line 11, 42
Small size display 83
Smooth local refinement 96
Smoothing 83, 91, 142
SNR 120
Solid modeling 8, 9
Source
  composite 27
  coder/decoder 24
  coding theorem 26
  coding theory 3
  discrete 32
  Gaussian 32
  Laplacian 32
  model 104
  random process 24
  subsource 27
Space filling
  advantage 33
  disadvantage for progressive transmission 34
Spatial
  continuity 129
  hierarchy 7, 11, 53, 96
  indexing 8
  level 59, 90
  resolution 4, 8, 13, 59, 61, 76, 77, 94
Splitting (HSRQ) 67, 142
Standard image database: See SIDBA
Stationarity 21, 87
Step size 49
Subband coding 119
Subquadrants 8
Subsampling 14, 51, 62, 96, 108
  regular 92
Subsource 27
Subtrace 72
  bit selection 73, 88, 90
  common 88
  initial 72
  parenthesis 73
Successive refinement 29
  condition for 30
    expressed geometrically 31
  positive examples 32
System of events 20

T

Table lookup 13, 56, 102, 133
Tamminen 10, 58
Task 42
Telebrowsing 43
Teleconferencing 43
Teleconsulting 43
Television 40, 44
Ternary gray scale hierarchy 101
Time complexity 9, 56, 100, 102
Trace 70-74, 95, 104
  averages 87, 89, 104, 142
  bit selection trace 88
  empty trace 88
  for pixels 71
  occurrence 87
  set 87, 89
  subtrace: See Subtrace
  tree 71, 89, 97

Training sequence: See Vector
Transform
  coding 5, 46, 74, 119, 125, 127
    for progressive transmission 15
    lossless 47
  discrete cosine 5, 15
  discrete Fourier 16
  FFT 15
  Gabor 6
  Hadamard 16
  Karhunen-Loève 5
  orthogonal 5
Transmission
  line 24
  sequence 15, 57, 62
  system model 24
Tree structured vector quantization:
    See Vector quantization
Triangular
  quadtree 12
  matrix 98
Truncated octahedron 34

U

Uniform
  background 12, 126
  color spaces 132
Unit-distance-equalized significance
    code 61
Universe 7, 8

V

Variable
  rate coding 44
  resolution 77
Vector quantization 4, 16, 38, 49, 75,
    108, 118, 120
Vector quantization advantages 33
Video 40
Visibility threshold 16
Visual
  system: See Human visual system
  evaluation 112
Voronoi regions 33

W

Wide area networks 2
White to black reproduction 82, 114
Woodwark 10, 55

Y

Y_iIQ color space 132

Z

Zador 33