Dynamic Rate Shaping of Compressed Digital Video

Alexandros Eleftheriadis

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Graduate School of Arts and Sciences

Columbia University

1995

ABSTRACT

Dynamic Rate Shaping of Compressed Digital Video

Alexandros Eleftheriadis

We introduce the concept of Dynamic Rate Shaping (DRS), a technique to adapt the rate of compressed video bitstreams (e.g., MPEG-1, MPEG-2, H.261, JPEG) to dynamically varying rate constraints by operating directly in the compressed signal domain. Such a scheme is shown to be critical for multimedia communication systems, since it can guarantee universal interoperability between encoders/decoders and networks with widely different (and even time-varying) quality of service guarantees. The concept is shown to evolve naturally by considering first the operation of an actual multimedia communication system, in the form of the "Xphone" testbed that we have developed. It is shown that a key technique for enabling video communication using a system based on a best-effort operating system and network is, among others, adaptive rate control of the bit rate of video (JPEG in this case).

A natural extension of the adaptive rate control approach in Xphone is Data Partitioning. This scheme splits a compressed bitstream into two parts, and achieves robustness by transmitting them over channels with different quality of service guarantees. This provides the first critical step in detaching the manipulation of the rate from the encoder. An analysis of optimal data partitioning is provided using an operational rate-distortion context, and several algorithms are proposed. The DRS concept then arises by eliminating the second bitstream and allowing the rate constraints to vary over time.

DRS provides an interface between the encoder and the network, with which

the encoder's output can be perfectly matched to the network's quality of service characteristics. In essence, DRS bridges the gap between constant and variable bit rate video, providing a continuum of possibilities between the two. The problem of optimal DRS is analyzed, and a family of optimal and fast algorithms is described. We also show that Data Partitioning is a special case of "clustered constrained" DRS. Some of the fast DRS algorithms perform extremely close to optimal; their low complexity allows even purely software-based real-time implementation, thus making them attractive candidates for incorporation in actual multimedia communication systems.

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Architectures for Multimedia Communications 4

1.2.1 The Computer 5

1.2.2 The Network 12

1.2.3 The Video Codec 17

1.3 Overview and Contributions of the Thesis 19

1.3.1 The Xphone System 19

1.3.2 Data Partitioning 22

1.3.3 Dynamic Rate Shaping 23

2 The Xphone System **27**

2.1 Introduction 27

2.2 The Xphone System Architecture 32

2.2.1 Call Management 33

2.2.2 Scheduling 34

2.2.3 Network Transport 36

2.2.4 Media-Specific Support 37

2.3 Source Bit Rate Control 38

2.3.1 The JPEG Compression Algorithm 39

2.3.2 A Quantizer–Rate Model 40
2.3.3 Adaptive Rate Control Algorithm 42
2.4 End-To-End Delay 45
2.4.1 Sources of End-to-End Delay 46
2.4.2 End-to-End Delay Bounding 48
2.4.3 Average End-to-End Delay Minimization 49
2.5 Audio/Video Synchronization 52
2.5.1 The Synchronization Problem 53
2.5.2 Audio/Video Synchronization Algorithm 54
2.6 Concluding Remarks 58

3 Data Partitioning **61**

3.1 Introduction 61
3.2 Motion-Compensated Transform Coding 65
3.2.1 The Optimal Transform—Karhunen-Loeve 66
3.2.2 The Discrete Cosine Transform 67
3.2.3 Video Compression Using Transform Coding 69
3.2.4 The MPEG Picture Structure 72
3.2.5 Motion Compensation 75
3.3 The Data Partitioning Problem 80
3.3.1 Data Partitioning 80
3.3.2 General Problem Formulation 83
3.4 Data Partitioning in Non-Predictive Coding 84
3.4.1 Problem Formulation 84
3.4.2 The Optimal Algorithm 86
3.4.3 Performance Evaluation 94
3.5 Data Partitioning in Predictive Coding 97

3.5.1 Problem Formulation 97

3.5.2 The Causally Optimal Problem 99

3.5.3 The Causally Optimal Algorithm 101

3.5.4 Performance Evaluation 102

3.5.5 The Memoryless and Rate-Based Algorithms 104

3.6 Concluding Remarks 106

4 Dynamic Rate Shaping **108**

4.1 Introduction 108

4.2 The Rate Properties of Compressed Video 112

4.2.1 Variable Bit Rate Video 112

4.2.2 Constant Bit Rate Video and Rate Control 114

4.2.3 Video Rate and Communications Systems 117

4.3 Dynamic Rate Shaping 121

4.3.1 The Formulation of the Dynamic Rate Shaping Problem . . . 122

4.3.2 The Family of Dynamic Rate Shaping Algorithms 125

4.4 Constrained Dynamic Rate Shaping 126

4.4.1 Problem Formulation 128

4.4.2 Causally Optimal, Memoryless, and Rate-Based Algorithms . 132

4.4.3 Performance Evaluation 135

4.4.4 Clustering 140

4.5 Unconstrained Dynamic Rate Shaping 143

4.5.1 Problem Formulation 145

4.5.2 Optimal Breakpoint Vector Selection 147

4.5.3 Performance Evaluation 152

4.6 Applications 154

4.6.1 Transcoding and Codec Interoperability 154

4.6.2 Trick Modes 155
4.6.3 Communication in Heterogeneous Networks 156
4.7 Concluding Remarks 157

5 Conclusions **160**

References **164**

List of Figures

1-1 A projection of the multimedia systems space. 5

2-1 Xphone in a projection of the multimedia systems space. 31

2-2 Xphone application layout. 33

2-3 Xphone call management state model. 34

2-4 Bandwidth vs. quantizer step size ($B(Q)$) for various image sizes. . . 41

2-5 Example scenario of video bit rate control algorithm. 43

2-6 Quantizer (Q) hysteresis-based adaptation mapping ($\Delta Q = Q(t) - \hat{Q}(t)$, $\Delta Q' = Q(t + \delta t) - \hat{Q}(t)$). 44

2-7 Adaptation of Q to network load in an actual videoconference (3 min duration). 45

2-8 End-to-end delay increments and audio buffer occupancy in an actual videoconference. 47

2-9 Restart protocol for end-to-end delay bounding. 48

2-10 Estimated end-to-end delay with and without silence detection in actual videoconferences (2 min duration). 51

2-11 Estimated transmission delay with and without silence detection, for the sessions of Figure 2-10. 52

2-12 Audio output buffer occupancy. 55

3-1 Segmentation of images or video frames into blocks. 70

3-2 Zig-zag scanning pattern of transform coefficients. 72
3-3 The MPEG Picture Structure. 74
3-4 The MPEG sequence structure (display and coding orderings). 78
3-5 Block diagram of a motion-compensated transform coder. 79
3-6 Block diagram of a Data Partitioning system. 81
3-7 Breakpoint position in the zig-zag pattern of DCT coefficients. 82
3-8 Overview of the bisection algorithm. 89
3-9 Slice 20 (full-width, frame 0) from "Flower Garden", coded at 24 Mbps (x) and 12 Mbps (o). 93
3-10 Data partitioning of frame-based, intra coded "Flower Garden", from 24 Mbps to 12 Mbps, using optimal and rate-based algorithms. . . . 96
3-11 Slice 20 (full-width, frame 3, P-picture) from "Mobile" coded at 4 Mbps and partitioned at 3.2 Mbps: (x) $\hat{D}(B_i)$, (o) $D(B_i)$. 101
3-12 Bit distribution for the "Mobile" sequence coded at 4 Mbps, with I period 12, and B period 3 (the overhead bits include all non-DCT bitstream components). 103
3-13 PSNR (Y only) for "Mobile" sequence, frame-based coded at 4 Mbps and partitioned at 3.2 Mbps using the causally optimal, memoryless, and rate-based algorithms. 104
3-14 Distribution of accumulated error, causally optimal, and memoryless distortions across all slices of a picture ("Mobile", coded at 4 Mbps and partitioned at 3.2 Mbps, frame 3, P picture). 105

4-1 Variable bit rate motion-compensated transform encoder. 113
4-2 Constant bit rate motion-compensated transform coder with rate control. 115
4-3 Definition of Dynamic Rate Shaping. 123

4-4 The family of Dynamic Rate Shaping algorithms. 126
4-5 Breakpoint position in the zig-zag pattern of DCT coefficients. 127
4-6 $R(D)$ curves for an intra macroblock from "Flower Garden", coded at 24 Mbps (x) and 12 Mbps (o). 136
4-7 $R(D)$ curves for the total (A and E) and current picture only (E) shaping distortion for a macroblock from a P picture of "Flower Garden", coded at 4 Mbps and rate shaped at 3.2 Mbps. 136
4-8 Rate shaping of "Flower Garden" using the optimal, rate-based, and recoding approaches. The source is coded at 24 Mbps and rate shaped at 12 Mbps. 137
4-9 PSNR performance for rate shaping of "Mobile" using the (causally) optimal, memoryless, rate-based, and recoding approaches. The source is originally coded at 4 Mbps and rate shaped at 3.2 Mbps (I period 12, P period 3). 138
4-10 Average PSNR performance for rate shaping of "Mobile" using the (causally) optimal, memoryless, rate-based, and recoding approaches, for various target rates. The source is originally coded at 4 Mbps (I period 12, P period 3). 139
4-11 Clustering approach for the reduction of algorithmic complexity. . . . 141
4-12 PSNR performance for clustered ($C(44)$) rate shaping of "Mobile" using the (causally) optimal, memoryless, and rate-based approaches. The $C(1)$ optimal algorithm is also shown. The source is originally coded at 4 Mbps, and rate shaped at 3.2 Mbps (I period 12, P period 3). 142
4-13 The definition of breakpoint in the Constrained and Unconstrained DRS algorithms. 144

4-14 $R(D)$ "clouds" from a macroblock of "Mobile", coded at 4 Mbps. The 12 DCT coefficients generate 4,096 different breakpoint vector possibilities. 148
4-15 Recursive nature of run-length code generation. 149
4-16 PSNR performance for rate shaping of "Mobile" using the unconstrained optimal and memoryless algorithms, as well as the constrained optimal algorithm. The source is originally coded at 4 Mbps and rate shaped at 3.2 Mbps (I period 12, P period 3). 153

List of Tables

2.1 Coefficients of polynomials $p_1(x)$ and $p_2(x)$ in eq. (2.1) ($B(Q)$ model). 42

3.1 Typical quantizer weights for frequency-weighted quantization (larger weights correspond to coarser quantization). 71

Acknowledgements

I would like to express my sincere gratitude to Prof. Dimitris Anastassiou who supervised the research work contained in this thesis. Perhaps more important than the tangible results of a doctoral dissertation is the process of intellectual growth that produces them. I am particularly indebted to his guidance, his acute understanding of the importance of system-level issues, and his encouragement to pursue my own interdisciplinary interests.

During the development of this thesis I also had the benefit of interacting with a number of colleagues and friends. I thank in particular Professors Shih-Fu Chang and Martin Vetterli, and my ex-colleagues Professors Antonio Ortega and Kannan Ramchandran. Special thanks go to my colleague Sassan Pejhan with whom we closely collaborated on a number of projects, and also Ilana Pelzig-Cellum and Moleka Ahmed who assisted me in the development of the Xphone system. I would also like to thank Drs. Arun Netravali and Joel Zdepski for taking time off their busy schedules to discuss with me several issues of the problems addressed in this thesis. Finally, I would like to thank Drs. Arnaud Jacquin and Nikil Jayant for providing me with the opportunity of spending two very stimulating summers at AT&T Bell Laboratories.

The quality of the thesis has been significantly improved by the diligent efforts of the members of my defense committee: Profs. H. Meadows, S.-F. Chang, S. Feiner, and D. Duchamp, whom I all thank.

Finally, I would like to express my gratitude to my wife, Sonia, for her support throughout the lengthy years during which this thesis was prepared.

Chapter 1

Introduction

1.1 Introduction

The concept of a multimedia computer system has emerged due, in large part, to significant advances in computing and networking technologies that allow the implementation of sophisticated systems. The sheer computation speed and memory sizes of computers have grown exponentially during the past decade, and it seems that this trend will continue well into the next decade; network bandwidth evolution actually leapfrogged computers with the introduction of fiber-based transmission technology. As the computer continues to penetrate into everyday life, users and developers strive to find new ways of information representation and exchange [72, 10]. The multimedia concept promises to fulfill the need for expanded flexibility and more direct interaction with computers; in the same way that windows were the dominant "presentation technology" of the past decade, multimedia will play a similar role for future applications. The former uses high-quality, computer-generated graphics, while the latter will be able to use natural video and audio as well. The concepts of flexible image/video also found their way into consumer electronics products (e.g., Compact Disk Interactive—CD-I—and Photo-CD).

The enabling technology for multimedia systems is video compression algorithms.

The raw bit rate of PCM-coded video (24 bits per pixel, 640×480 pixel frames, 30 frames per second) is approximately 220 Mbps, far exceeding today's general purpose computer capabilities. Recent advances in compression technology for images and video have resulted in reductions of two orders of magnitude, which are well within today's limits. In addition, the work of international standardization organizations and the increased interest in video applications for computers and consumer electronics products have resulted in VLSI implementations of these algorithms which can be used for the development of real systems [11, 121, 5, 102, 81, 83, 4, 6, 91, 3].

Video coding, however, is just one of the components of a multimedia system. The support of continuous, high-volume and real-time data (like video or audio) in both computers and networks represents a tremendous shift in design methodology, resulting in a re-evaluation of basic principles. Time dependency of information as a concept only existed in dedicated systems (e.g., the telephone network, or embedded systems); with multimedia, it becomes an issue for practically any application. The focal point of multimedia research is to provide bit-pipe characteristics (guaranteed bandwidth, low and constant delay, accurate synchronization) to packet-based systems, using algorithms and architectures that can be widely deployed.

In this thesis we provide solutions to a number of open problems in multimedia communications, with particular emphasis on the treatment of compressed video signals. Central to the whole thesis are the notions of "flexibility" and "interoperability," i.e., algorithms that allow different components of a multimedia system to work together, even if their design assumptions are radically different. Our attempt has been to utilize signal processing principles, and apply them in a computer networking context. This implies that a detailed understanding of issues in a number of traditionally separate fields was required, ranging from source coding theory to advanced operating systems. We have found this atypical interaction extremely interesting and fruitful. In fact, due to the inherently interdisciplinary nature of multimedia, such an approach is often the only one that can provide meaningful and useful results.

In this introductory chapter we first provide a brief overview of recent research in the area of multimedia systems and multimedia communications (Section 1.2). This will help us construct a reference framework for later analyses, and introduces several interesting problems that can be directly or indirectly solved using the results presented in this thesis. We then present a brief outline of the key problems addressed in this thesis, as well as a summary of its major original contributions (Section 1.3).

In this overview we establish that the basic conclusions drawn concerning multimedia communications are:

- there is a high degree of uncertainty in terms of guaranteeing time constraints in a general purpose computer, even with real-time scheduling support,
- there is a high degree of uncertainty in terms of guaranteed quality of service in the network, due to the tradeoff between latency and reliable transport and also due to the bottlenecks of reliable multicast transport protocols,
- the structure of standards-based video codecs (e.g., JPEG, MPEG-1, MPEG-2) which will be widely used in computers is not robust enough to efficiently accommodate such an environment.

These fundamental principles are the driving forces behind much of the developments in this thesis, and will be key themes in the chapters to follow.

1.2 Architectures for Multimedia Communications

The broad range of applicability of the term "multimedia system" necessitates the establishment of some basic assumptions about the underlying components. We identify three basic parameters that can serve as a rough basis for classification of such systems, all related to support of time-dependent operations. The first parameter is the video codec (a key component of any truly multimedia system). As will become evident in the sequel, the structure of the codec's algorithm and the way the codec is interfaced to the system plays a significant role in determining the system's overall design. The second parameter is the equipment hosting the codec; it can range from a general purpose computer—with or without real-time support—to a specially designed dedicated device (e.g., a terminal with only display and user input capabilities). Finally, the third parameter relates to the quality of service (QoS) that the underlying network provides. The range here spans from unreliable, multi-access networks, to ones with deterministic guarantees in terms of both bandwidth and delay.

The 3-D parameter space is depicted in Figure 1-1. This parameterization is "video-centric" and ignores several other potential media of a multimedia system, such as multichannel sound or graphics. The assumption is that their overall contribution to the bandwidth requirements of the system is small compared with that of video. When this is not the case (e.g., in high-resolution 3-D animated graphics), in several instances the same concepts applied to video can potentially be used for these types of media.

We are mostly interested in applications that utilize general purpose computing equipment; the relevant design area in Figure 1-1 is shown in light gray. This represents the most interesting, and most complex, case; simpler cases, however, like dedicated set-top units are also of importance. The area of high-end systems

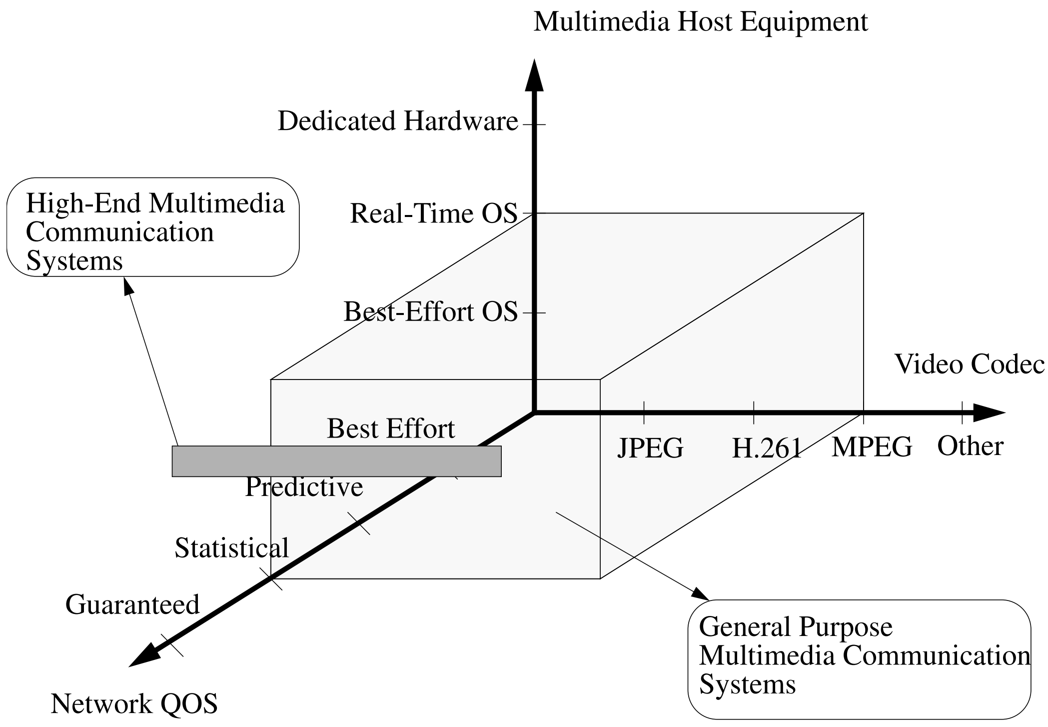

Figure 1-1: A projection of the multimedia systems space.

is depicted in dark gray, and it includes real-time operating systems and network transmission with guaranteed quality of service. In the following, we provide an overview of the architectural considerations—and supporting algorithms—affecting the computer, the network, as well as the codec itself. A broader overview, and from different perspectives, can be found in [62].

1.2.1 The Computer

Traditionally, general purpose computers are designed around a shared bus, with the operating system controlling the access of resources by various application programs. In simple systems, where only a single application is allowed to run, the design is straightforward and "flat." In more complex multitasking or multiprocessor systems the operating system's layering is thicker, and its responsibilities are more involved. The basic guidelines in both cases are fair resource sharing, and reliable processing or transfer of data between the various components of the system (mass storage, memory etc.). During the years, a number of techniques have been employed to maintain a geometric growth in terms of speed and memory sizes (bus width increase, DMA, caches, pipelining, parallelism etc.). It should be noted that the simultaneously increasing complexity of software has always been able to tax the available hardware resources.

In supporting multimedia applications involving video and audio, a computer must be able to handle continuous and real-time data flow. In addition, it must provide time-dependent services for the timely transfer of information between its various components. The difficulty in supporting these requirements is their time and bandwidth scale; video for example may require the transfer of many kilobytes per frame (i.e., every 33 msec, for a 30 frame per second video signal). Furthermore, video and audio should be made available to applications with the same ease as text and graphics. Although there are many systems which use dedicated connections for video and audio (by bypassing the bus and/or the CPU, e.g., [9, 95]), their lack of media integration precludes their classification as multimedia systems.

1.2.1.1 The Operating System

The operating system has a central responsibility for supporting multimedia, as the primary resource coordinating entity of the system. Current operating systems utilize static priority scheduling for processes, using locks and blocking where appropriate to facilitate resource sharing. As explained in [90] locking does not seem to pose a problem for multimedia applications, due to their limited expected use of shared data structures. Consequently, the primary objective of a multimedia operating system is supporting time constraints specified by application programs.

In order to meet such constraints it is essential to make the kernel fully preemptible (by introducing extensive locking) [55, 80]. In addition, time constraints have to be communicated to the system by applications. The form of these constraints can vary; due to the fact that some requests may be infeasible, time limits are usually combined with priorities to facilitate the selection of requests to be deferred.

It is important to note that software alone can not provide sufficient time-related guarantees. Delays caused by interrupt and/or DMA processing in the lower half of the kernel are substantial, but their adequate bounding requires hardware support [55]. Although bus scheduling and data path optimization techniques have been proposed [120, 37], it seems unlikely that hardware support in a large scale is possible. It is worth noting, however, that commercial operating systems have started to incorporate timing information in their process scheduling algorithms (primarily those derived from AT&T's Unix System V Release 4, e.g., SunOS 5.x, IRIX 5.x), presumably to aid multimedia applications development.

An area which has received a lot of attention is disk scheduling for multimedia data. In order to support video on-demand and general multimedia database services, special purpose disk scheduling and admission control algorithms have to be employed. Techniques proposed constrain either the order of requests or the physical block placement on the disk, in order to meet continuity requirements, minimize buffering and maximize the number of users that can be concurrently supported [107, 129, 118, 127].

1.2.1.2 Multimedia Services for Applications

Application-level multimedia services constitute a layer between the operating system and the actual application. This layer has the responsibility of interfacing the application with the actual media hardware (via the operating system), while providing features like media synchronization and coordination. A similar structure, for example, is employed in the X Window system which provides distributed text, raster graphics and coordination services. The literature on the subject is rather extensive (with even commercial implementations available, e.g., Apple's QuickTime), and a detailed description of the various approaches is outside the scope of this section. It will be useful, however, to provide a summarized view of the approaches taken.

The primary service to application programs is synchronization; it is typically classified as intra-media or inter-media. In the former case, also referred to as rate synchronization, it guarantees the possible rate constraint imposed by the media at hand (e.g., 64 Kbit/sec for audio). Inter-media synchronization guarantees that actions pertaining to different media but related either implicitly (video and audio) or explicitly (by the user, as in an image and associated text), occur simultaneously. The so-called "lip-sync," which refers to the synchronization of lip movement (video) and the associated speech signal, is an example of inter-media synchronization.

Since an extremely attractive application of multimedia communication is multipoint conferences, the multimedia service should also provide distributed access and synchronization services (e.g., floor control). These usually take the form of serialization in computer-supported collaborative work environments, as well as synchronization in the communications subsystem. In the following we discuss synchronization at all levels of a multimedia system.

Clearly, in describing a multimedia document one has to specify the timing interrelationships between the various component objects. Techniques that have been proposed include timelines (as in the MAEstro system [36] and QuickTime), temporal scripting languages, Timed Petri Nets [84], and formal description techniques (as in protocol validation) [30]; in addition, international standardization activities are underway to construct an object-oriented representation of multimedia and hypermedia objects (ISO's Multimedia and Hypermedia Experts Group—MHEG) [77]. In cases where the temporal characterization is relative (i.e., not based on a globally known clock), linear programming techniques can be used during run-time to compute and execute a schedule [16]. Note that such an approach greatly simplifies editing, but on the other hand imposes a considerable burden on the run-time environment.

At a lower level, specific time-dependent interrelationships have to be maintained. Although the subject has received a lot of attention for the case of distributed systems (discussed below), little work has been reported for local synchronization: it is generally assumed that adequate support is provided locally at each host. One plausible proposed approach is restricted blocking [115]; as an example, if a video stream has to be blocked at a synchronization point, instead of staying idle (blocked) it displays the previous frame. This alone, however, does not provide enough operational versatility (consider for example the case of a network connection: restricted blocking will essentially increase the end-to-end delay by an amount equal to its duration).

A more elaborate approach is based on physical and logical synchronization frames [94]. The former are handled internally by the system, while the application applies synchronization control at the logical frame level (or in exception handling) via "upcalls." The device specifics are abstracted using a device specification language. Two problems with this approach are: 1) the communication overhead between the system and the application may be substantial if logical frames are close in size to physical frames, and 2) accurate corrective operation by the application may require knowledge of the underlying device state, which is hidden by the layering. An essential issue here—considering that decisions have to be reached and enforced in few tenths of milliseconds for video and audio—is the observability of the system; a more detailed discussion of this concept is presented in Section 2.5

In distributed systems, where the network intervenes between the source and the destination, similar synchronization problems appear. In multipoint environments an additional problem concerns sequencing of operations. Fundamentally, synchronization problems over networks are caused by delay variance (jitter) and clock drift. The latter can be eliminated if the network itself provides an accurate clock, or bounded (to a few milliseconds per day) using special host-to-host protocols [86, 87]. Depending on the underlying networking technology, the jitter scale can vary very much; contrast for example a multi-access environment and a constant bit-rate link[1].

In order to eliminate jitter, a well-known technique is buffering at the receiver. For example, in [31] extensive buffering (up to 1 sec) of multiple user reaction scenarios is proposed in order to accommodate jitter and minimize the response time experienced by the user. In other words, the system "anticipates" the user's reactions and prefetches the necessary material, essentially providing a caching mechanism. This, however, is unacceptable for person-to-person communication, where the end-to-end delay has to be less than 250 msec[2]. Another approach, taken in [69], consists of selective video frame dropping at the transmitter, together with audio buffering at the receiver. Frames are dropped when the asynchrony between video and audio exceeds a prespecified limit. In order to facilitate accuracy, the system actually decides on a possible drop only after it has acquired access to the—token ring—network. Note that this requires an extremely tight coupling between the

[1] A discussion of the network's implications in multimedia systems is provided in Section 1.2.2.

[2] This number is drawn from traditional long-distance telephony requirements. For other collaborative tasks this number is significantly reduced

application and the network controller, which may not always be possible (in [69] the authors used a real-time operating system of their own design). A similar approach is taken in [105], where again the sender controls synchronization based on periodic feedback from the receiver. The feedback substitutes the knowledge of the precise network access time and, coupled with suitable delay bounds (maximum and minimum), is used to monitor the playback process at the receiver. We should note that a significant drawback in placing the control in the sender is that it complicates one-to-many connections.

A well known problem in multipoint applications is sequencing. This is a form of synchronization, but pertains only to adequate ordering of requests or messages [124]. A simple example can be seen in a distributed editing environment; clearly, user actions must be performed in the same order for all participants of the session. The simplest approach in this case is to allow access to the "conference floor" to a single user at a time [8], with an obvious reduction in interactivity and flexibility. Another approach is to have centralized control; all messages are sent to a central coordinating entity (a bridge) which orders them and further distributes them to all recipients. Such a scheme, with multi-level bridging, is proposed in [119]. The drawback in this case is that the bridge must be able to handle substantially higher load than each individual participant. Since this bridging is application-specific, it cannot be handled by networking hardware. Of course, not all data need to be subject to serialization (in [119] sequencing is a per-packet option). For example, if video and audio are used only for communication support purposes, they may be sent directly and not through the bridge. The possibly different transmission delays, however, may be noticeable to the conference participants. Another framework for multi-point synchronization has been proposed in [111], and consists of maintaining logical clocks bound to logical input or output devices. Clocks may be active if incremented in real-time, or passive if incremented by the devices themselves. By binding different devices to the same logical clock, synchronization can be achieved. A possible drawback in this case is that, in order to avoid blocking, continuous communication must be maintained between all participants to ensure proper management of the clocks. Even worse, the proposed system breaks down in—very common—symmetric configurations.

1.2.2 The Network

Digital video coding alone would not be sufficient to allow widespread use of multimedia communications; although the achieved bandwidth reduction is significant (up to two orders of magnitude), it still required a similar increase in available network bandwidth. The introduction of fiber-based transmission technology provided the potential for multi-gigabit transmission [7]. Although this potential has not been fully tapped yet—being limited by the speed of electronics—many commercial computer networks today operate at rates of 100-150 Mbps (even with twisted-pair or coaxial cabling), or even 622 Mbps and beyond.

The integration provided by Asynchronous Transfer Mode (ATM) broadband networks [2, 56] will allow a single network base to provide a multiplicity of communication services [24]. As the ratio of communication over computing resources increased, and quality of service considerations were introduced, it became apparent that a reevaluation of networking concepts was necessary [76]. We should also not exclude from consideration existing technologies, like the current Internet Protocol (IP version 4) [70] or its forthcoming replacement, IP version 6 [33].

Seeing the network from a user's perspective, we are primarily interested in the transport and network layers and the various services they will provide in future communication networks. Clearly, these services will be critical in shaping the structure of multimedia applications, and vice versa. In the following we give an overview of the current status of research in high-speed integrated network services. Our intention is to identify characteristics from which an operating environment for future multimedia applications can be extrapolated.

At a high-level, it is realized that significant streamlining is necessary in protocol stack design, in order to eliminate replication of functionality; although layering is conceptually attractive, it should not hinder efficient implementation [28]. One of the most direct approaches that can be taken in high-speed protocol processing is parallelism [65, 103, 133]. As network bandwidth leapfrogged CPU speeds, the number of available CPU cycles for processing a single packet was significantly reduced; parallelism provides a natural solution, applicable to modern protocol designs as well [92]. Noting that processing power is a more scarce resource than bandwidth, additional enhancements can be achieved by reduced protocol processing requirements [52].

1.2.2.1 Transport-Layer Protocols

Current transport protocol designs (e.g., TCP [71]) are based on window flow control and error control; the window essentially specifies the number of packets that can be in the network, unacknowledged by the receiver. A large window size improves throughput, but at the same time increases the necessary buffering at the transmitting site. In Gbps transmission rates, the size of the data that can be in-transit in the network until an acknowledgment is received by the sender can be quite large (across the US this time is approximately 30 msec, translating to many MBytes of data that may have to be retransmitted) [103]. In addition, fiber-based channels have a very low bit-error rate; consequently, reliability will be mostly affected by packet losses due to congestion rather than actual bit errors. Current error protection schemes are targeted towards the latter; also, in the case of windowing, they are strongly coupled with flow control.

In order to eliminate the drawbacks of windows, rate-based flow control is proposed as an alternative for high-speed communication. In order to handle packet losses, selective retransmissions of groups of packets may be used [25, 26, 92]; such a scheme can be augmented by packet-based forward error correction in order to reduce the number of retransmissions [53]. In addition, periodic state exchange can be employed to eliminate almost all error recovery procedures and reduce the error recovery timers that have to be maintained [92, 103]. An important consideration for transaction-oriented applications (e.g, Remote Procedure Calls—RPC) is the call setup time. In addition to the well-known handshake techniques, timer based methods can be employed in which a connection is implicitly setup by the reception of the first data packet; connection establishment can be acknowledged with the data packet sent back, while connection termination can be effected via timers [103, 53].

An important issue related to the quality of service provided by the network layer is logical multiplexing; if multiple streams are multiplexed at the transport layer then the network layer will have to apply the most demanding requirements to all of them, and hence significantly reduce efficiency [28, 51]. Consequently, logical multiplexing above layer 3 (network) should be avoided.

1.2.2.2 Network-Layer Protocols

Although not directly interfaced to the users, the network layer plays an important role, since it is there where quality of service is enforced. The advent of integrated, multimedia services introduces stringent constraints on bandwidth and delay requirements. For example, person-to-person communication may require constant availability of bandwidth (ranging from a few Kbps to many Mbps), and a delay acceptable for human communication (less than 250 msec). An issue further complicating the problem is variable bit rate sources (e.g., coded video), whose analytical modeling is difficult.

In order to accommodate the requirements of individual users, the network layer must provide admission control (to control admission of new users) and policing functions (to enforce that prenegotiated traffic agreements are not violated). These functions are closely related to the service discipline employed at the output queues of the network switches. A comparison of such rate-based service disciplines is provided in [131]. Apart from deterministically guaranteed services, statistical guarantees may be useful as well [54]. In a similar vein, observing that network users may be able to adapt to variable delay, a predictive service type was introduced in [27]. In this case the bound can not be statistically characterized; the system merely asserts that some very high fraction of the packets obey the bound. An admission control scheme for predictive service is described in [66].

A facility that is crucial for multipoint multimedia communication is the capability of the network to support multicasting. Clearly, the overhead involved in emulating one-to-many communication via replicated packet transmissions can very quickly deplete network resources. The traditional IP was extended to support multicasting, as described in [34, 32]. Efficient and scalable multicasting in ATM-based networks is still a research topic (see for example [125] and references therein). The lack of source addressing in the connection-oriented ATM scheme does not allow direct cell demultiplexing by mere examination of the cell header (VPI/VCI).

The capability of the network—and lower—layers to support multicasting must somehow be incorporated in the transport layer (both in terms of data transport and in terms of session establishment). In [99], the multicast channel is approached in a fashion similar to tune-in TV broadcast channels. The authors recognize that receiver processing may be required to effect error concealment; hierarchical coding is suggested for this purpose. This approach is of course only applicable to video and audio, with no sequencing requirements. A reliable multicast protocol based on negative acknowledgments is described in [49], but without regard for flow control or sequencing. In the current version of IP (version 4), multicasting is accessed by applications through the UDP protocol, which is connectionless and unreliable. The same is expected to hold for the next generation of IP (IPng), i.e., version 6.

Regarding session establishment, a number of paradigms can be adopted. The most difficult case is the support of dynamically set up conferences. Due to the fact that multicast addresses are essentially a network resource, some form of management must be enforced. Clearly, it would be unacceptable for two independent sessions to be accidentally merged together by using the same addressing parameters. Although this "crosstalk" can be eliminated by appropriate filtering, since this has to be done above the network layer it unnecessarily taxes computing resources. Issues related to session control are addressed in [112].

We should note that a number of protocols incorporating parts of the above concepts have been actually implemented. Designs specifically addressing multimedia communication issues include the Tenet protocol suite developed at U.C. Berkeley based on the RTIP protocol (Real-Time Internet Protocol) [54, 130], and the ISI/BBN suite based on the ST-II network protocol coupled with the NVP and PVP transport protocols for audio and video respectively [18]. Detailed feature and performance comparisons between various transport protocols can be found in [35, 65, 88, 60]. Most recently, efforts within the Internet Engineering Task Force (IETF) have resulted in a new light-weight transport protocol for real-time communications over IP, called RTP [113]. RTP, however, is essentially a packet header specification rather than an actual protocol; it is up to the applications to implement end-to-end algorithms for their particular requirements.

1.2.3 The Video Codec

Although the primary function of the video codec is to reduce the video source bit rate, there are a number of parameters that can significant affect a system's architecture. A major consideration is the characterization of the codec's output stream: it can be constant or variable rate. The "natural" rate of most codecs is variable, which results in constant image quality. In recognition of the need of constant bit rate video (e.g., for CD-ROM applications), buffer control techniques are widely used to constrain the output rate to a constant value. Buffer control is a feedback technique, in which the encoder monitors the contents of the output buffer which is emptied at a constant rate (the channel rate); the encoder adapts the coding process in order to ensure that the buffer never underflows or overflows [6, 3, 96].

In applications such as residential video distribution, video-on-demand, or consumer electronics products, the above scheme—coupled with appropriate audio coding—is sufficient. In a B-ISDN environment, the equipment can be attached to ATM Adaptation Layers (AAL) 1 (constant bit rate—CBR) or 2 (variable bit rate—VBR) [2, 24, 117, 114], or even AAL 5 (intended primarily for data transmission). The inherent ease with which dedicated hardware can be accurately clocked does not create any synchronization or related problems in the equipment hosting the codec. Of course, we do not claim that efficient VBR traffic support in a network is easy; if however it is supported, the end-user equipment can easily utilize it.

As was discussed in Section 1.2.1, when the codec is hosted in a general purpose computer, the latter's architecture greatly affects the system's performance. Furthermore, the extent to which computers will be supporting AAL's 1 and 2 is not yet clear. Considerations related to interface hardware cost, actual connection costs (billing) and the bursty nature of network traffic generated by computers in non-multimedia applications may affect the extent of this support. In order to reduce the analytical source rate modeling difficulties posed by VBR video, special purpose codecs can be devised whose equivalent buffer occupancy feedback information is provided by the network policing function. Although such an approach is indeed attractive, we should note that general purpose computers will most likely be equipped with general purpose codecs, that is, very few assumptions will be made on the underlying channel (be it the computer's bus and/or a possible network connection).

From the above discussion we see that it is quite likely that the codec will have to be interfaced to an environment where uncertainty may prevail. On the computer architecture side (Section 1.2.1.1) we saw that real-time support may be available, but will not completely guarantee "hard" real-time constraints; on the network side (Section 1.2.2), unreliability may be present in the form of packet loss due to congestion. In addition, inherent inefficiencies of reliable multicast transport protocols may enforce unreliability in multipoint multimedia applications. We should also not exclude current communication and computing facilities as a possible environment. Although quality of service cannot be guaranteed, multimedia communication may still be possible albeit with considerable quality loss. Currently proposed standard codecs—JPEG, MPEG-1, MPEG-2, H.261—however, were not designed for unreliable channels. Although a hierarchical design may have been used (as for example in MPEG), the degree with which it can be put in use to accommodate unreliability in a computer/network environment as previously described is rather limited.

1.3 Overview and Contributions of the Thesis

As mentioned throughout this system-level overview, a basic feature dominating general-purpose, computer-based multimedia communications is that of uncertainty: time-based constraints can not be fully guaranteed by the host computer, the network connection may not be fully reliable, and the network delay and jitter may not be fully guaranteed. On the other hand, economies of scale, flexibility and compatibility considerations will result in the deployment of standards-based image and audio-video codecs (JPEG, MPEG, or H.261). The structure, however, of these codecs does not fully match the abovementioned constraints. The mismatch may be even more pronounced in multipoint environments over heterogeneous networks.

These considerations have motivated us to explore techniques that can provide high-quality communication even under these adverse conditions. Our approach begins with the examination of algorithms for a general-purpose environment, consisting of standard workstations and networking equipment. This environment exhibits essentially all of the difficulties identified in Section 1.2.

1.3.1 The Xphone System

A complete testbed (Xphone) has been implemented, and has been used to test and evaluate several different original algorithms. The testbed is based on standard workstations running a multiuser, multitasking operating system (SunOS 4.1), equipped with cameras and JPEG video compression hardware boards. The workstations are connected with each other using a 10 Mbps Ethernet network, shared among approximately 25 workstations. This setup represents a typical configuration for computer-supported collaborative work environments, as well as for simple desktop videoconferencing.

Xphone has been designed as a distributed multimedia communication application development system; in that sense, it is not a stand-alone application but it provides core services which facilitate the development of applications that require real-time audio-visual communication support. This broader approach introduces several interesting problems which would not be present under a more narrow perspective. Its architecture is novel and, in its latest version, is comprised of several different modules. The core of the system is the Xphone library which is linked to the application process, and serves as the principal coordinating entity between the application and the communication facilities that Xphone provides. Xphone introduces the concept of intra-application scheduling, which effectively implements a micro-scheduler within an application and totally transparent to it. We show that such scheduling is critical for supporting continuous data flow for video and audio within the event-driven architecture of windowing systems (e.g., the X Window System). It is also essential for implementing effective synchronization algorithms. In addition to scheduling, Xphone also introduced full session control services for point-to-point communication. A complete protocol was designed for that purpose, operating between applications and special Xphone servers residing in each machine. The semantics of the session layer are identical to that of a telephone call, emulating traditional videoconferencing systems.

Within the Xphone testbed we have introduced and evaluated several different algorithms. A key open problem is that of audio-video synchronization. Since audio and video signals follow different paths within the host computer and may also be subject to different delays by the network, achieving accurate synchronization is not trivial. This is particularly so because of the sensitivity of the human perception to even very small deviations. We introduce a technique which bounds the temporal misalignment by the video frame rate; this guarantees that, as frame rates increase with faster computers, the accuracy will also increase. Note that the frame period is a very natural “unit of measurement” for synchronization purposes, as can be seen by considering that for very high frame periods synchronization (at least when speech signals are concerned) is rendered meaningless.

Another open problem which has been addressed in Xphone is the rate control of the video signal. Due to the high variability of the best-effort, multi-access network used, any a priori assumptions regarding the available network bandwidth are inappropriate. We introduce a novel technique in which the instantaneously available bandwidth is monitored, and a non-linearly filtered version of it is used to control the compression ratio used. Since the exact compression ratio cannot be controlled (since it depends also on the source material), a statistical relationship is derived between it and the encoder’s quantizer step size for typical head-and-shoulders images.

Finally, a critical parameter for person-to-person communication is the end-to-end delay. Very long delays can significantly inhibit interaction, as they introduce unnatural “pauses.” Long distance telephony requirements mandate that delay is kept below 250 msec. Due to the nature of Ethernet, end-to-end delays for audio-visual signals can reach 1 sec in less than a minute. To mitigate this problem, we introduced the concept of “on-demand” compression and utilized silence detection. The former compresses video frames and submits them for transmission only when sufficient network resources are available. The latter, in addition to reducing the overall bandwidth requirements, introduces “relief intervals” during which the audio output buffers at the receiver are allowed to drain (and hence the current end-to-end delay drops to zero).

The Xphone system’s architecture and algorithms are discussed in detail in Chapter 2.

1.3.2 Data Partitioning

In Xphone, the algorithm utilized for video compression (JPEG) is relatively simple. Temporal independence of individual frames is very helpful in terms of adapting the continuous video source to a bursty one (albeit at a lower frame rate), using on-demand compression and transmission. Hence we proceed to examine more closely techniques with which more complex codecs can be interfaced to the network. We focus on motion-compensated, block-based transform coders which represent the state-of-the-art in video compression. The ISO/IEC MPEG-2 standard is used as an example, since it is the most complex in this family of algorithms, and is also expected to be the dominant technique used for several years to come.

Our approach explores the concept of hierarchical transmission: the original information is segmented into layers of high priority (a coarse-level representation) and low priority (high-detail refinement). The two layers are transmitted either with different priorities or, if the underlying network does not support prioritized transmission, with different levels of forward error correction. In cases of network congestion or other error-inducing situations, the low priority layer is discarded first. When a single-layer approach is used, any losses can be quite detrimental to the video quality at the receiver. Since any redundancy has, to a great extent, been removed in the coded representation of the signal during the source coding process, recovery from losses using interpolation or extrapolation can be quite difficult and oftentimes ineffective. The dual-layer approach achieves a much better visual quality at the receiver, since a usable signal can always be decoded. We solve the open problem of optimally partitioning a bitstream into two such layers, and in particular within the context of the so-called "Data Partitioning" scalability mode of MPEG-2.

In data partitioning, transform coefficients of each block of the compressed video signal are segmented into two parts, hence forming the two different layers. Low-frequency coefficients are included in the high-priority layer (for guaranteed transmission) while the high-frequency ones are included in the low-priority layer (and are potentially subject to losses). The technique is similar to the well-known techniques of hierarchical or pyramidal coding, but a key difference here is that Data Partitioning can be applied even after encoding has taken place. This makes it applicable not only to live video applications (e.g., news broadcasts, videoconferencing), but also to stored video applications such as video-on-demand. We analyze the problem of data partitioning in an operational rate-distortion context, and derive optimal algorithms as well as fast approximations. The mathematical approach (based on constrained minimization using Lagrange multipliers) is very similar to the one employed in the optimal JPEG thresholding work in [73, 74]; in this case, however, we also take into account the temporal recursiveness inherent in the MPEG coding scheme. We should note that although the technique is discussed within the context of MPEG-2, it is applicable to essentially any block-based transform coding scheme, including MPEG-1 and H.261.

1.3.3 Dynamic Rate Shaping

Finally, motivated by the work done in Xphone and the Data Partitioning problems, and as a direct extension of it, we introduce the novel concept of "Dynamic Rate Shaping." The point of view taken is that in order to optimize the information transport mechanism, perfect knowledge of the transported data can only have a positive effect. Consider for example the case where only constant bit rate audio is being transferred over an unreliable network. In order to filter out network and computer-induced jitter, extensive buffering is required at the receiver. This, however, is contrary to low end-to-end delay requirements of personal communication applications. A technique which can be used (and has also been implemented in the

Xphone system) in order to reduce both network load and end-to-end delay is silence detection. Typically, the speech signal has a 50% (or less) activity; by detecting and filtering out silence, we reduce the bandwidth required, but we also provide "relief" periods where jitter can be absorbed.

The above technique is nothing more than utilization of knowledge of the structure of the transported data to maximize the actual information throughput. In order to properly detect silence, the system has to perfectly know how to operate under the underlying audio coding scheme. A similar technique can be effectively used for video. Such an approach also permits the use of priorities, if provided by the underlying network mechanisms or via different levels of forward error correction.

For the case of compressed video signals, of primary interest is the bandwidth required for their transmission. A process with which the rate of a compressed video bitstream can be modified to comply with some given constraint will be called Rate Shaping. If the constraint is allow to vary over time, then it will be called Dynamic Rate Shaping.

One of the major concerns here is that the processing required for proper interfacing of a coded video bitstream to the network has to be within acceptable bounds; fast algorithms have to be used, preferably implementable in software. This requirement precludes the obvious, but extremely expensive, approach of decoding the signal and recoding it at the new target rate. As will be shown in Chapter 4, recoding can also yield inferior quality, which—at first glance—is a somewhat surprising result. In manipulating the coded bitstream, a dynamic rate shaping algorithm must have knowledge of the bit-stream syntax and heavily capitalize on codec properties (which themselves are linked to video signal properties). Conceptually, the codec transforms the signal from its natural domain to one where redundancy is drastically reduced; the codec-network interface, or dynamic rate shaper, has to operate on the mapped domain of the codec's output, to achieve improved quality as perceived by the end-user.

We analyze the problem of Dynamic Rate Shaping for motion-compensated, block-based transform coders, with particular emphasis on the MPEG-2 scheme. Our approach follows the treatment of Data Partitioning, that is, it uses an operational rate-distortion framework. We identify two major categories of algorithms for DRS, namely requantization and selective transmission. Our focus is the latter category, but comparisons are made with requantization using a recoding approach.

A family of algorithms is proposed, providing optimal solutions under different assumptions, as well as fast approximations. We segment selective transmission algorithms into constrained and unconstrained. An optimal constrained algorithm is obtained using Lagrangian optimization, and shown to be extremely complex in terms of computational resources, but primarily unacceptable due to the excessive delay it introduces.

A "causally optimal" algorithm is then proposed, which provides an optimal solution when the added constraint of causality is introduced. By sidestepping the recursive component of the coding process, a "memoryless" algorithm is shown to perform almost identically to the causally optimal one (within 0.5 dB), hence providing an extremely good tradeoff between quality and implementation complexity. This is a key result since it allows extremely simple and practical implementations. Another key property is that, under a very large range of target bit rates, the causally optimal and memoryless algorithms perform better than recoding.

Further enhancements are introduced in the form of "clustered" algorithms, which significantly decrease the degrees of freedom of the optimization problem to reduce computational complexity. Two particular algorithms, $C(4)$ and $C(8)$, are identified as representing suitable compromises between complexity and performance for real-time software-based implementation.

Finally, the unconstrained optimal DRS problem is attacked, and a fast algorithm is derived using a combination of Lagrange multipliers and dynamic programming. The algorithm is shown to be extremely complex, but does not provide substantial improvement over the constrained case (about 1 dB). This result augments the already good properties of the memoryless constrained scheme.

The results presented in this thesis have also been published in [45, 47, 38, 39, 40, 21, 12]. For the benefit of uniformity and cohesiveness, work which was not directly related to the main theme of Dynamic Rate Shaping has not been included, but can be found in [122, 46, 20, 19, 100, 41, 42, 43, 48, 44, 64, 101, 63].

Chapter 2

The Xphone System

2.1 Introduction

One of the enabling technologies for multimedia systems is video compression algorithms. Recent advances in compression technology for images and video (JPEG, MPEG-1, MPEG-2) have resulted in bandwidth reductions of two orders of magnitude, down to 1–2 Mbit/sec. In addition, the work of international standardization organizations and the increased interest in video applications for computers and consumer electronics products have resulted in VLSI implementations of these algorithms which can be used for the development of real systems [4, 6, 5, 102, 91, 3, 121, 81, 83, 8].

Video coding, however, is just one of the components of a multimedia system. The support of continuous, high-volume and real-time data (like video or audio) in both computers and networks represents a tremendous shift in design methodology, resulting in a re-evaluation of basic principles. Time dependency of information as a concept existed only in dedicated systems (e.g., the telephone network, or embedded systems) and Computer-Supported Collaborative Work (CSCW) environments; with multimedia, it becomes an issue for practically any application. The focal point of multimedia research is to provide bit-pipe characteristics (guaranteed bandwidth, low and constant delay, accurate synchronization) to packet-based systems, using algorithms and architectures that can be widely deployed.

The availability of some kind of real-time support from the underlying operating system and network is important for high-quality, wide-area multimedia communications, and is currently a very active area of research (see e.g., [27, 59, 66, 130] and references therein). It is nevertheless possible to provide multimedia communication even in environments where delay uncertainty prevails (best-effort systems), albeit with some quality degradation. In addition, algorithms employed in non-real-time and real-time systems can be the same; although the latter will definitely perform better, the techniques used to achieve this performance can be similar (especially if the real-time support is not "hard"). Throughout this chapter we assume the use of a best-effort operating system and network; in other words, no time-related guarantees are provided.

A number of systems and techniques have appeared in the literature, addressing various aspects of multimedia systems. Early efforts provided audio communication only [8]. Some systems use analog video and audio communication [9], with the corresponding self-evident limitations in terms of media integration in user applications. A significant volume of work has been reported at the system architecture level [31, 94, 119], describing the interface between applications and multimedia services and the latter's structure.

In the area of media synchronization, a number of techniques have been proposed. These include incorporation of time constraints and scheduling of multimedia documents [16, 77, 84], media synchronization for database access applications (where a high end-to-end delay is acceptable) [31, 105], and synchronization for interactive multimedia communications [69, 68]. In the first and second areas, the proposed techniques are basically used to derive time-stamps (or their equivalent) with no further analysis of how these time-stamps will be enforced; in addition, strong assumptions are usually made in terms of the performance of the underlying network and host equipment [105]. In the third area, which is more directly related to our work, the techniques described in [69, 68] require very tight coupling of software/hardware layers (the authors use their own operating system).

In this chapter we describe the architecture and associated algorithms of the Xphone multimedia communications system, which has been developed to support the use of multimedia information both locally and across networks by end-user applications. Our primary focus—and the major contribution of this chapter—is on the techniques used for source bit rate control, audio/video synchronization and end-to-end delay control, as well as the mechanisms with which these techniques can be integrated into a coherent and usable service for application developers. Source bit rate control uses features provided by the video encoding hardware to accommodate network load variations and window size changes performed by users. Audio/video synchronization and low end-to-end delay are essential for acceptable human communication. Synchronization is achieved via an algorithm based on time-stamps, while end-to-end delay minimization is achieved via silence detection and a restart protocol.

The performance of these algorithms is demonstrated by a number of graphs depicting system parameters in actual conferencing sessions. The current implementation supports point-to-point connections over TCP/IP; the above techniques however are potentially applicable to other transport protocols, and can also be used in multi-point connections. The selection of TCP was primarily dictated by two factors. First, the effectiveness of the algorithms could be demonstrated with minimal protocol-related complications, since all of them operate at a higher layer. Second, use of the video source bit rate control algorithm and protection of the LAN from congestion required the use of a flow-controlled transport protocol.

We should note that, since Xphone's original design (1991), several other desktop videoconferencing systems have been designed and implemented. The most popular ones are those used in the so-called "MBONE," i.e., the multicast-enabled part of the Internet, and include `nv`, `vat`, `ivs`, etc. All these applications are dissemination-oriented: they multicast audio and video material over the network, and users "tune-in" into the appropriate multicast addresses in order to receive it. The lack of any transmitter-receiver interaction precludes any form of quality control, and the received video and audio quality is typically very poor. Furthermore, audio and video are transmitted by different programs, and no mechanisms are provided to effect synchronization. These characteristics make them suitable for very large conferences, but not for small group, highly interactive collaboration. A small number of commercial products have now also been introduced, utilizing algorithms extremely similar to the ones used in Xphone.

The specific environment in which the system has been implemented and evaluated is composed of Sun SPARCstation 2 workstations connected via an Ethernet LAN used by 25 hosts. Video compression (JPEG) is provided by an XVideo board from Parallax Graphics, while audio acquisition/playback is performed through the workstation's audio hardware. Basic communication parameters of the system are (averages):

- 8 fps for 320×240 24-bit video,
- 250 msec end-to-end delay, and
- 1 Mbps bandwidth (full-duplex, including 64 Kbps audio).

The positioning of this system in the space of multimedia communication systems is shown in Figure 2-1. The axes denote increasing complexity and/or support features for each of the three major system components, i.e., the network, the operating system and the video codec.

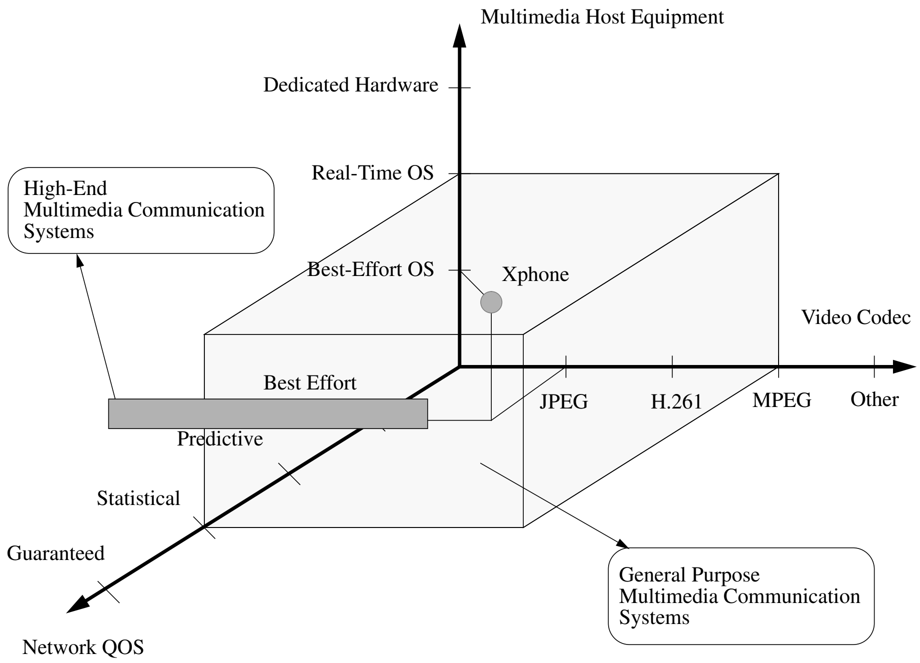

Figure 2-1: Xphone in a projection of the multimedia systems space.

The structure of the chapter is as follows. In Section 2.2 we briefly describe the architecture of the system, and show how the individual algorithms described later on are integrated. In Section 2.3 we describe our source bit rate control algorithm. Section 2.4 describes the end-to-end delay properties of the system, and shows how silence detection has been employed for its reduction by a factor of 50%. In Section 2.5 we describe the audio/video synchronization algorithm used. We conclude the chapter with some remarks and a summary of the major contributions in Section 2.6

2.2 The Xphone System Architecture

The objective of the system is to provide distributed multimedia services to application programmers. In other words, Xphone is not an application per se, but rather a facility that multimedia system developers can employ for their specific needs. Basic features that had to be provided are:

1. support for continuous data streams, such as video and audio, and intra-application scheduling,
2. synchronization facilities, especially for video and audio, both locally and across a network connection,
3. an easy to use and robust session management facility, and
4. compatibility with existing interactive application environments and development practices (e.g., the X Window System).

It is not our intention to provide a specific multimedia object structuring like the one found in multimedia documents; such constructions are located hierarchically higher than Xphone, and can be easily accommodated by it.

The system comprises four main subsystems: call management, scheduling, network transport and media-specific support. The latter includes support for I/O operations for various media types and also the appropriate synchronization mechanisms (which for fine-grain synchronization are dependent on the media device specifics). In the following we briefly describe each component. The structure of a typical Xphone application is shown in Figure 2-2.

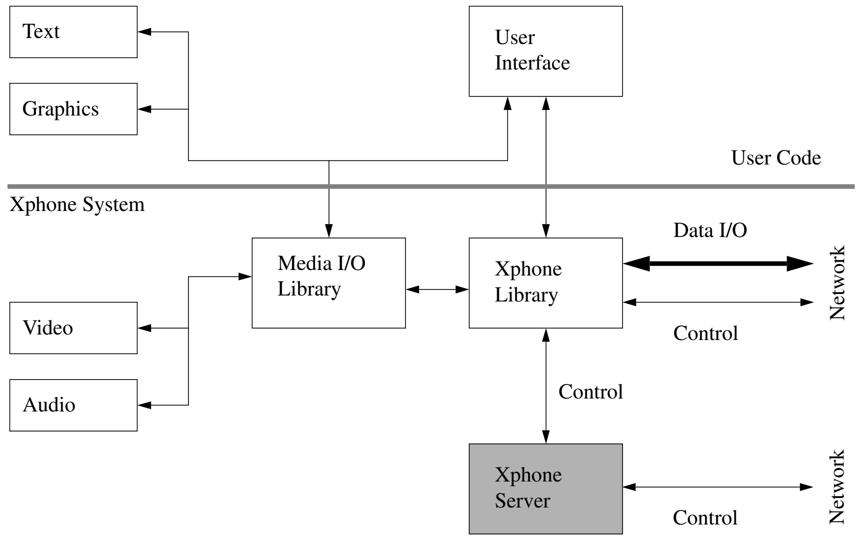

Figure 2-2: Xphone application layout.

2.2.1 Call Management

Call management in the Xphone system is a fully symmetric operation. It is performed by a server process—which must be available in each workstation—that receives and dispatches call control information to and from application programs. When a server receives a connection request it notifies the relevant user either by periodically printing a message on the screen, or via the peer application if it is currently running. A connection request can be accepted or rejected by the end-user, aborted by the caller or it can fail if an error occurs.

After successfully establishing a connection, the application processes exchange data directly. Connections are terminated either by the the applications (hang-up), or by system errors (connection failures). The call state model presented to applications is shown in Figure 2-3. Note that all error-related transitions have

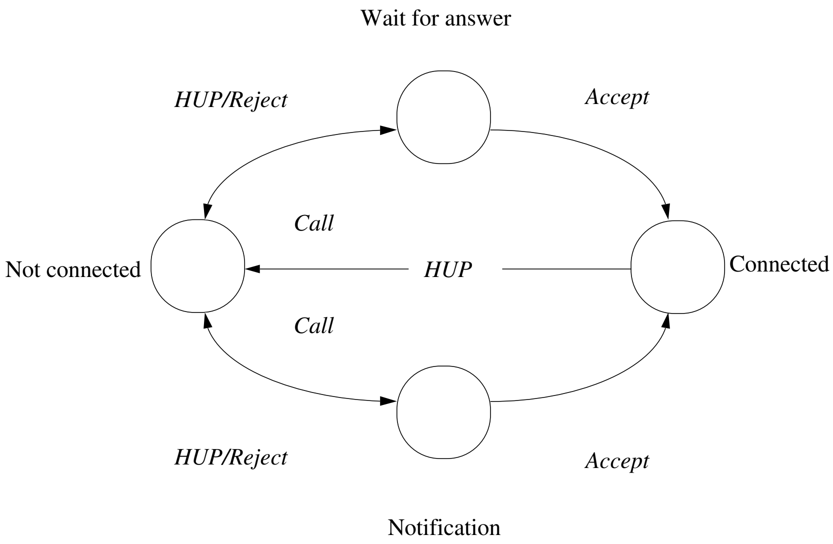

Figure 2-3: Xphone call management state model.

been removed for clarity. This model is similar to the one presented by a classical telephone service, with the added benefit of caller identification.

During the connection establishment, the peer applications exchange the port numbers [116] that they have already bound for actual data transmission. The server has been implemented using the Remote Procedure Call (RPC) package, and is registered to `inetd` for automatic invocation [116].

2.2.2 Scheduling

Scheduling at the application level is essential for providing continuity of data flow. This is amplified by the event-driven architecture of interactive, window-based graphical user interface environments (like those based on the X Window System). In these environments, the application is designed to react to prescribed events generated by the user or the system (e.g., when a button is pressed); the main program control is handled internally by the supporting windowing software. Consequently, a scheduling facility is provided so that: 1) software development can still be based on the established call-back architecture, and 2) continuity of data flow is guaranteed. In doing so, the facility has to be seamlessly integrated to the windowing environment; this has the additional benefit that existing applications will be able to use multimedia services with no modifications of their already developed code. In our software we provide support for the XView and X Toolkit Intrinsics packages (other toolkits can of course be easily added). The support consists of equivalent substitutes to main-loop control functions of these packages, which use the Xphone scheduler for window system event processing. Xphone event processing (e.g., a call request) is performed synchronously with the scheduler; in other words events are only dispatched between scheduler tasks in order to guarantee state consistency.

The scheduler can be seen as a static priority one, with the difference that tasks are usually not removed from the scheduling queue. The scheduler processes tasks in a round robin fashion, starting from the ones with highest priority. The application program has the option of restarting a round, hence skipping low priority tasks. Certain tasks—like windowing system event processing—are always given the highest priority, as they can adversely affect the interactive response time of the application.

Data I/O is performed via the scheduler as follows. Each medium (video, audio etc.) is assigned a unique identifier by the application. For each such medium read and write functions have to be provided. The former reads data from the medium device (or a storage device) and submits it to Xphone for network transmission. The latter receives data from Xphone, originating from the network, and plays it back on the medium device (or perhaps stores it in a file). These two functions are registered to the scheduler under the medium identifier with a specified priority. If the priority is non-zero, the scheduler will automatically invoke the read function when appropriate. The write function is invoked by the scheduler whenever a packet with data of this specific type is retrieved from the network. The system attempts to read data from the network between tasks and, if successful, it immediately dispatches them.

With this scheme, the application can guarantee continuous data flow with a single call to the scheduler that registers the appropriate I/O operations. The fact that event processing is synchronous greatly simplifies the application's code. Moreover, the overhead of these operations is very small, and when appropriately optimized allows the application to operate with very high performance.

2.2.3 Network Transport

The system currently uses the TCP/IP protocol stack, and hence the processing here is minimal. The system structures the transmitted data with a header which includes the medium identifier, packet length and time-stamp information. When a packet is received by Xphone, the header is transformed to a larger one which includes an entry for a "reception" time-stamp. This can be used later on for time keeping purposes (e.g., to monitor the end-to-end delay).

It is possible in a network connection to not be able to completely read or write a medium packet from or to the network. While network read operations may be incomplete (the system will complete the operation at a later time), write operations must be completed when ordered. Although an output queue could be used, it would increase the end-to-end delay considerably. To avoid this problem and also to help increase throughput, after incomplete write attempts a read operation is performed which, if successful, will dispatch a packet to the appropriate medium write function. The incomplete write attempt is then resumed. This functionality essentially makes flow control decisions visible to the application layer: incomplete writes are triggered by filled buffers at the transport protocol layer, which signifies a congestion situation. This property is critical in ensuring a small end-to-end delay, since it prevents the creation of excessive queueing backlogs caused by congestion. Of course, the tradeoff here is a temporarily lower frame rate for video, but this is deemed much less severe than the effects of large end-to-end delays.

2.2.4 Media-Specific Support

This component is responsible for handling I/O and control operations for the various media types. These operations depend heavily on the specific hardware platform selected, and its accompanying software interfaces. In our environment the audio hardware is treated at the application level as a regular Unix device, while the XVideo board is operated through X Window System based operations. Although a generic device interface would help application developers (and it has frequently been proposed in the literature), it is extremely difficult to capture the richness of the various interfaces under a single entity. In addition, layering such a generic interface on top of differing native interfaces may degrade performance. Our approach consists of providing support for I/O operations that conform to the Xphone scheduler interface, but allowing the application the option to fully control other operations (e.g., the video window size or its placement in a user interface).

Media synchronization is performed in this component, as it heavily depends on the specifics of the implementation. Synchronization in our system is based on timestamps, which are placed by the acquisition routines (the media read functions) in the medium data header. Fine-grain inter-media synchronization (basically between video and audio) is performed by "supervised output:" the media write routines of the media types to be synchronized are encapsulated under a single write operation which performs the necessary decisions and invokes the appropriate media write operations when necessary. Coarse-grain synchronization can be effected by the time-stamp time line.

In the current implementation, the system uses the Sun audio device which provides 8-bit μ-law companded audio at an 8 KHz sampling rate, and the XVideo board from Parallax Graphics which provides "on-demand" JPEG coded video frames of sizes up to NTSC resolution (640×480). On-demand implies that frame acquisition/playback and compression/decompression are under complete program control; in other words, there is no buffering of the video source at the device driver level (as opposed to audio which is continuously sampled).

2.3 Source Bit Rate Control

An important parameter of a multimedia communication system is the target bit rate of video. Factors affecting its selection include the available network resources, the capabilities of the computer hosting the video codec, as well as the structure of the codec itself. For example, the bandwidth provided by today's local and wide area networks spans more than two orders of magnitude, from 10 Mbps Ethernet to 100 Mbps using FDDI or the newest variants of Ethernet, and even more with ATM. Most importantly, in environments where the network does not provide guaranteed bandwidth or delay, the available bandwidth, as seen by the application, is often highly variable. Use of a constant target bit rate in this case may adversely affect both the end-to-end delay of the system and the video frame rate (the latter will be affected when on-demand video coding or frame skipping is used).

The capabilities of the host computer also place limitations on the system performance, as there are specific limits of data throughput sustainable by the various components (bus, CPU etc.). Finally, the actual codec used has a dominant effect on the range of achievable bit rates, as it directly controls both the compression ratio and the maximum attainable frame rate. In cases where the compression parameters are fixed, the only possible way to control the source rate is by modifying the frame rate. Most codecs (including JPEG), however, have the capability to trade-off image quality and bit rate. By exploiting this capability, one can adapt to large variations of the network load. The algorithm described here assumes the use of the JPEG compression algorithm, and it also provides for adaptation of the source rate to video display window size (possibly performed by the user).

2.3.1 The JPEG Compression Algorithm

The JPEG algorithm for still image compression can be briefly described as follows [5, 121, 102]. Each color component of the original image is divided into non-overlapping blocks of 8×8 pixels. Each block is first offset by -2^{P-1}, where P is the number of bits per color component (8 for true color). It is then transformed by a forward Discrete Cosine Transform (DCT) [108] to yield 64 frequency coefficients.

The DCT coefficients are then quantized according to an implementation-dependent quantization table, which is under user control. High frequencies, to which the human eye is less sensitive, are quantized using a coarse step size, while low frequency components are subject to a much finer quantization. This quantization step is the principal source of lossiness in the JPEG algorithm. Next, the quantized coefficients are rearranged in ascending order of spatial frequency by starting with the DC (top-left) coefficient and proceeding in a zig-zag manner. The DC coefficients are then differentially encoded. The other 63 coefficients are run-length encoded [91, 29] to produce a string of zero AC coefficients followed by a non-zero AC coefficient. The run-lengths are then entropy coded (Huffman or arithmetic coding) to achieve compression. Huffman tables are also customizable. At the decoder the reverse procedure takes place.

By varying the quantization tables, applications can achieve a trade-off between compression ratio and output image quality: the coarser the quantization, the higher the compression ratio since the quantized coefficients will be smaller and the strings of zeros preceding a non-zero coefficient longer. The quality of the output image, however, will become poorer. In the specific video coding equipment that we used, the quantization process is controlled by a single parameter Q; the higher the value of Q, the coarser the quantization(i.e., Q is proportional to the quantizer's step size). In addition, the achievable frame rate is an increasing function on Q (a higher Q yields a higher frame rate). This is due to various system-level (bus, device driver etc.) bandwidth bottlenecks.

2.3.2 A Quantizer–Rate Model

In order to adapt to network load and image size variations, it is necessary to find an explicit relationship between Q, the source bit rate and the image size. An analytical derivation of such a formula is not possible, as the resultant bit rate is dependent on the source material. We have derived such a relationship by fitting a non-linear model to experimentally obtained data. We have used 18 different image sizes ranging from 96×72 to 640×480 (aspect ratio 4/3). For each image size, several minutes of video data (head and shoulders) were recorded and played back, for values of Q ranging from 25 up to 600 (in steps of 25). For each such combination, an average source bit rate was estimated (using the instantaneous values of frame size over inter-frame time). These values where fitted using minimum squared error techniques to the following non-linear model:

$$B = p_1(W) + p_2(W)\log(Q) \tag{2.1}$$

where B is the source bit rate in Mbps (here 1 Mbit $= 1024 \times 1024$ bits), W is the image area (measured in pixels), and $p_1(\cdot)$ and $p_2(\cdot)$ are 5-th order polynomials.

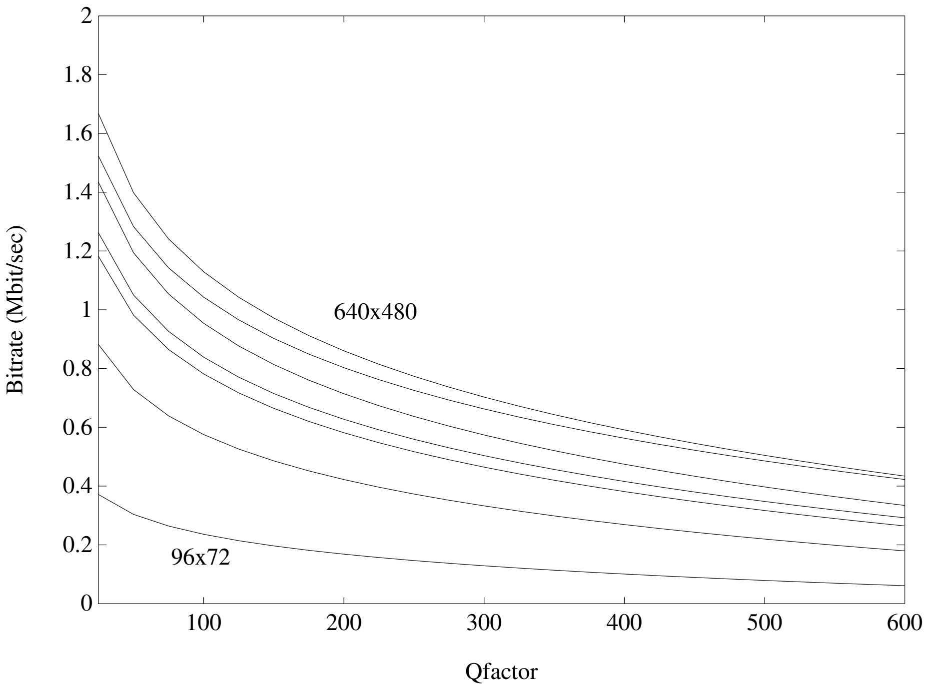

Figure 2-4: Bandwidth vs. quantizer step size ($B(Q)$) for various image sizes.

The selection of this specific model was based on its total squared error performance. The coefficients of p_1 and p_2 are given in Table 2.1. Figure 2-4 depicts the relationship between B and Q for various values of W. We should note that a tradeoff exists between the extent of the applicability of the model (in terms of the values for p_1 and p_2) for various video material, and the performance that it allows to be achieved. Furthermore, it should be emphasized that the above model encompasses the whole video subsystem (i.e., acquisition, encoding, transfer to main memory via the system's bus), and not just the encoder.

	p_1	p_2
x^0	0.1290	-0.0463
x^1	5.3007e-05	-1.6840e-05
x^2	-7.7014e-10	2.5060e-10
x^3	5.3620e-15	-1.7803e-15
x^4	-1.7126e-20	5.7934e-21
x^5	2.0236e-26	-6.9550e-27

Table 2.1: Coefficients of polynomials $p_1(x)$ and $p_2(x)$ in eq. (2.1) ($B(Q)$ model).

2.3.3 Adaptive Rate Control Algorithm

Our algorithm employs eq. (2.1) to adapt to network load variations as follows. The system (video input function) maintains an estimate of the available network bandwidth, based on measurements of actual throughput of the video stream only. This is given by the average frame length times the average frame rate, over a 10-frame window. Every 10 frames, this estimate is consulted and a possibly new value of Q is selected. Assume that $B(t)$ is the current bandwidth estimate, $W(t)$ the current image area, and $Q(t)$ is the current value of Q, as shown in Figure 2-5. Then under no network load, the output bit rate would be given by: $B(t) = p_1(W(t)) + p_2(W(t))\log(Q(t))$. If the network is loaded, then the actual bit rate observed will be lower, say $\hat{B}(t)$, corresponding to a Q value given by:

$$\hat{Q}(t) = \exp_{10}\left\{\frac{\hat{B}(t) - p_1(W(t))}{p_2(W(t))}\right\} \tag{2.2}$$

By selecting this new value, and hence moving from the operating point A_1 to A_3 in Figure 2-5, the system can lower its bandwidth requirements, and yet maintain a sufficiently high frame rate. This, of course, has the effect of degrading spatial image quality; the objective here is to sustain a frame rate which may already be marginally acceptable, by tolerating a small degradation in spatial quality.

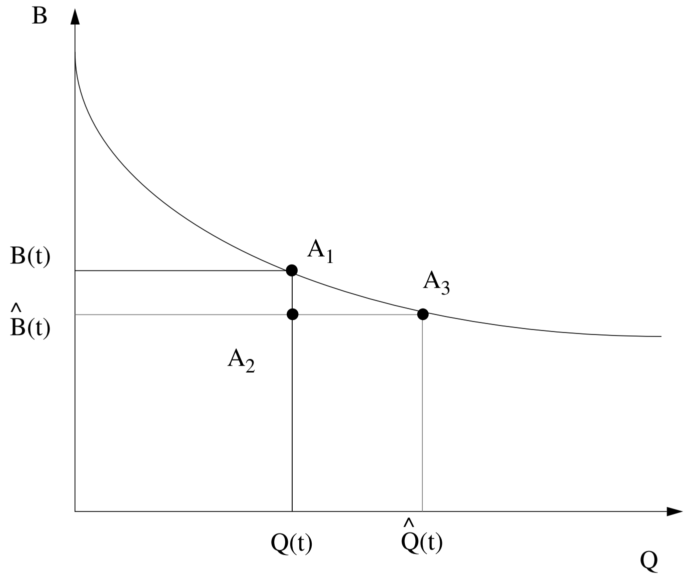

Figure 2-5: Example scenario of video bit rate control algorithm.

It should be noted that the measured bandwidth cannot exceed the one specified by eq. (2.1) (although this may some times happen since this is only an experimental, statistical estimate). If the actual available bandwidth was known, then eq. (2.2) could be applied directly to derive the new optimum value of Q. Since in our environment this information is not available, a procedure must be provided which will enable the increase of the source bit rate when the network load allows it.

The decision process used for the adaptation of Q is governed by the following three mutually exclusive cases:

1. if $\hat{Q}(t) > Q(t) + 50$, then $Q(t + \delta t) = Q(t) + 25$,

2. if $Q(t) + 50 > \hat{Q}(t) \geq Q(t) - 25$, then $Q(t + \delta t) = Q(t) - 25$, and

3. if $\hat{Q}(t) \leq Q(t) - 25$, then $Q(t + \delta t) = \hat{Q}(t)$.

This "hysteresis" mapping is depicted in Figure 2-6. Clearly, this process tries to favor the reduction of Q as it is the only means to increase quality. In order to reduce the sensitivity of the algorithm to small or short-time variations in bandwidth, large thresholds are used to trigger modifications of Q. The values of $Q(t)$ are always kept within the range between 25 and 350, which correspond to acceptable levels of quality for casual communication purposes.

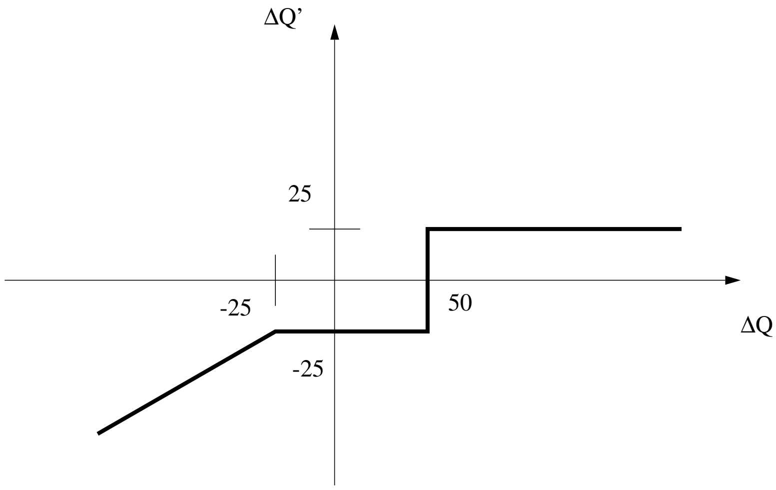

Figure 2-6: Quantizer (Q) hysteresis-based adaptation mapping ($\Delta Q = Q(t) - \hat{Q}(t)$, $\Delta Q' = Q(t + \delta t) - \hat{Q}(t)$).

In Figure 2-7 we show the variations of $Q(t)$, $B(t)$ and the frame rate $F(t)$ over an actual 3-minute session ($W = 320 \times 240$). The values have been scaled as shown, to facilitate comparative examination of the plots. Network load was introduced by TCP/IP traffic between two other hosts. As can be observed from the plots, $Q(t)$ increases whenever the available bandwidth as given by $B(t)$ decreases. On the other hand, if the network load permits it, reductions of $Q(t)$ result in larger

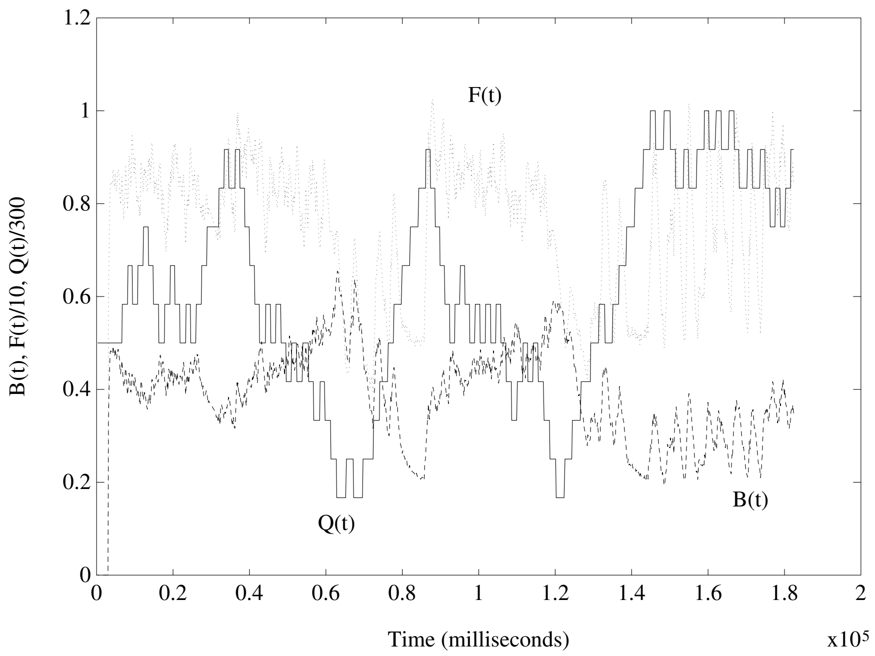

Figure 2-7: Adaptation of Q to network load in an actual videoconference (3 min duration).

output bit rate (and further attempts to reduce $Q(t)$). More significantly, however, we note that although a 50% reduction occurs in the output bit rate during the last minute, the effect on the frame rate is much smaller (a 10% reduction). This is accomplished by a reduction in quality, as shown by the high Q values.

2.4 End-To-End Delay

The end-to-end delay in audio communication systems is a very important factor, and is limited by the requirements imposed for human interaction. Acceptable end-to-end delay values prescribed for long-distance telephony are in the range of a few hundred milliseconds. Consequently, and since video in the Xphone system is on-demand coded, the end-to-end delay requirements are dictated by that of the audio signal, which is subject to a constant output processing rate. We should note that this is not just an artifact of the particular audio representation used: audio must always be sampled at a much higher temporal frequency compared to video in order to at least guarantee intelligible speech quality.

2.4.1 Sources of End-to-End Delay

We define the end-to-end delay as the time between acquisition and playback of an audio sample. This delay consists of several components. There is the acquisition time of the samples of an audio frame (a complete frame—which can have a variety of sizes—must be acquired before the operating system dispatches it to the application). Additional delay is introduced by network transmission, which includes transport and lower layer protocol processing and physical transmission of the data over the link(s). Finally, queuing delay is introduced at the audio output buffer as audio frames arrive in a bursty fashion. Other components such as buffer copying are ignored, as their effect is at a much smaller scale.

When a session is setup, the initial end-to-end delay consists simply of the acquisition delay of the first audio frame[1], plus the transmission delay associated with it. From that point on, this delay stays constant as long as the audio output buffer at the receiver is never emptied. If the buffer is emptied for a period of time, then the overall end-to-end delay of the session is increased by exactly that time; since the audio data can not be processed faster than their natural sampling rate, they accumulate at the receiver's buffer. This effect is demonstrated in Figure 2-8, where we show the increments in the end-to-end delay after the reception of the first audio packet and the corresponding audio output buffer occupancy.

[1]Note that the audio acquisition buffer size in our system is set by the operating system at 1024 bytes, which placed a lower limit on the acquisition delay of the first packet at 128 msec.

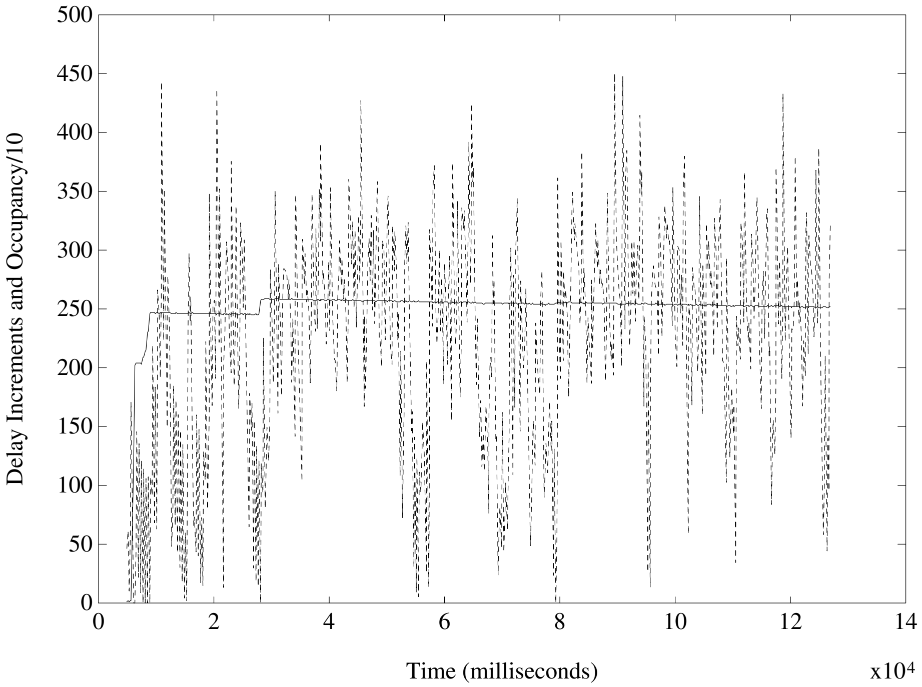

Figure 2-8: End-to-end delay increments and audio buffer occupancy in an actual videoconference.

2.4.2 End-to-End Delay Bounding

In order to keep the end-to-end delay bounded, a restart mechanism has been used to temporarily stop the acquisition and transmission of audio and video. Normal operation is resumed only after the receiver's audio output buffer is completely emptied, in which case the delay's state is identical to the one during the session's startup. The mechanism follows a simple handshake rule, as shown in Figure 2-9.

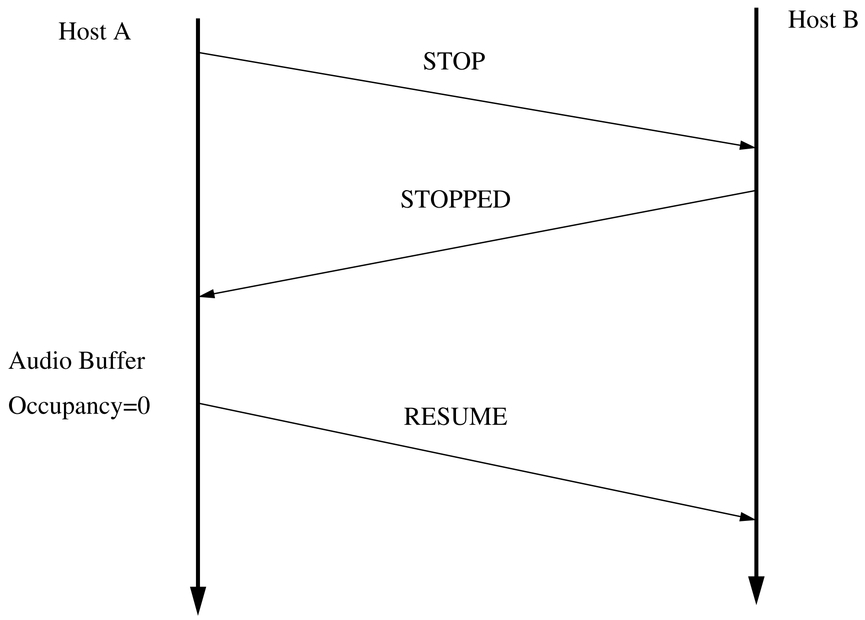

Figure 2-9: Restart protocol for end-to-end delay bounding.

When the receiver senses the average delay to be larger than the prespecified threshold, it sends a STOP message to the transmitter. Upon its reception, the transmitter stops acquiring and sending audio and video frames and sends a STOPPED message to the receiver. Meanwhile, the latter continues playing the frames that it receives or are already in its audio buffer. This is done in order to avoid dropping audio packets that were sent prior to the sender being notified of the temporary interruption of communication.

Once the receiving host receives the STOPPED message, it knows that no more audio packets are on the way. It then starts to monitor its audio buffer, and once it is empty it sends a RESUME message to the transmitter. When the transmitter receives this RESUME message, it resumes normal operation. Note that the restart procedure should only be used infrequently, as it interrupts the communication process. The duration of a restart procedure—and hence of communication disruption—follows closely the current end-to-end delay. The actual estimation of the end-to-end delay is described below.

2.4.3 Average End-to-End Delay Minimization

In order to mitigate the adverse effects of jitter, we employed silence detection in the audio signal. Silence detection is widely used for bandwidth reduction purposes in voice communication; here, however, we also use the silence parts of the speech signal to reduce the end-to-end delay. Essentially, silent parts of audio provide "relief" periods in which the output buffer is allowed to drain. The waiting time at the output buffer is then considerably reduced.

The effectiveness of this technique is directly related to the speech activity factor, which for telephone conversations is approximately 50%. In our system we have found that the activity factor is actually lower (around 40%) due to the effect of the higher end-to-end delay (similar to a long distance connection). Clearly, in the case of an audio stream with no silence such an algorithm will have no effect. We should note that an alternative approach in which the output buffer occupancy is reduced by selectively discarding very small audio segments (receiver drops) suffers from very rapid deterioration of speech quality due to temporal non-linearities.

The silence detector that we have employed is triggered by the difference between successive samples of audio. We opted here for simplicity and minimal processing overhead. Silence detection is always performed on a frame-by-frame basis, and is applied from the beginning of the frame until a non-silent part is reached. To avoid erroneous decisions, an initial segment of a frame is classified as silence only if it is at least one third of the frame's total length. For the same purpose, the first silent part detected after a non-silent one is never classified as silence. Although more sophisticated designs could have been used, this suffices to illustrate the effectiveness of the approach. Note that frame headers are always transmitted, even if the entire frame was classified as silence; also, the size of the initial segment of the frame that was classified as silence is transmitted in the frame's header.

In order to demonstrate the effectiveness of the technique, the end-to-end delay was estimated with and without the use of silence detection. The estimates (which are also used to trigger the restart protocol) are based on per-frame measurements of the abovementioned three principal components of the end-to-end delay (i.e., acquisition time, transmission delay, and output queueing), averaged over a window of size 10. The acquisition time is simply given by the ratio of the length of the frame (including silence, if any) to the audio sampling rate. The output queueing time can similarly be computed by the ratio of the current output buffer occupancy to the audio sampling rate.

The estimation of the transmission delay is more involved, as timing information from a single host has to be used in order to avoid clock synchronization problems. For this purpose, transmission delay is estimated as half the round-trip delay. The latter is obtained by sending a special packet with no data, that is immediately transmitted back to the sender. The round-trip delay is then the difference between the time this packet was sent, and the time it was received.

A new estimate is obtained between successive audio frame acquisitions. Due to the very small frame header size the added overhead is quite small. Moreover, the whole process is completely transparent to the application as it only involves the registration of the appropriate modules to the Xphone scheduler during initialization. The accuracy of this transmission delay estimate is restricted by a number of factors; in all cases where it is used (performance evaluation and restart triggering), however, an error of few tens of milliseconds is not significant.

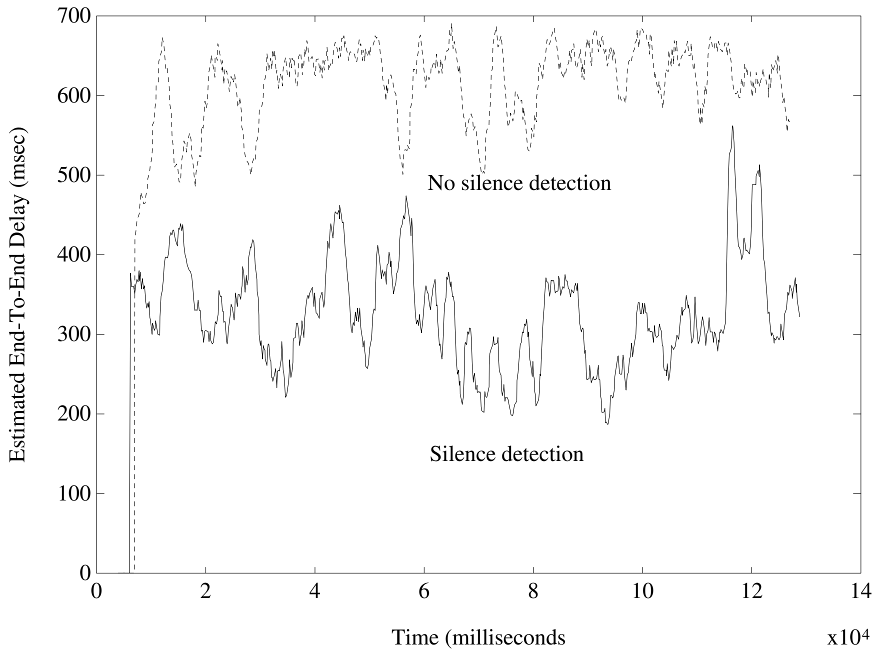

Figure 2-10: Estimated end-to-end delay with and without silence detection in actual videoconferences (2 min duration).

Figure 2-10 compares the estimated end-to-end delay with and without silence detection, over two 2-minute sessions. A speech activity factor of 50 % was maintained. To demonstrate that the two experiments were carried out under similar conditions (i.e., network load), the estimated transmission time for both cases is shown in Figure 2-11. As can be seen, the end-to-end delay with the use of silence detection has effectively been kept around 300 milliseconds, whereas with no silence

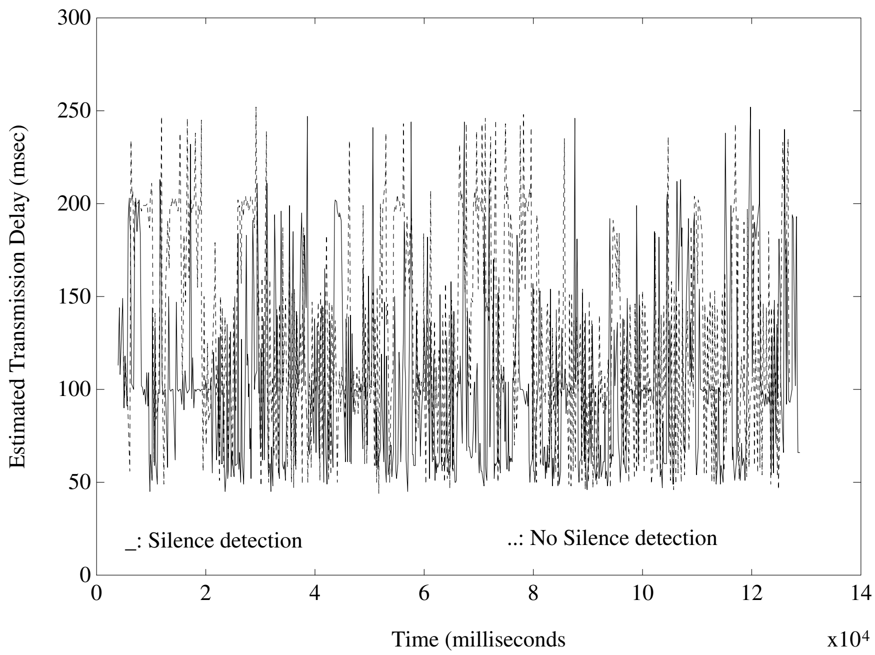

Figure 2-11: Estimated transmission delay with and without silence detection, for the sessions of Figure 2-10.

detection it reached 600 milliseconds.

A more sophisticated non-linear algorithm has also been developed (based on the so-called "Average Magnitude Factor"), and is described in detail in [63]. Experimental results with this more accurate silence detection technique are essentially identical with the ones presented here. The primary difference is that the perceptual speech quality is improved, due to the elimination of false alarms (speech segments classified as silence). The average end-to-end delay is also reduced due to the reduction of false positives (silent audio segments classified as speech).

2.5 Audio/Video Synchronization

Synchronization is an essential part of any multimedia system, regardless of local or distributed (across a network) operation. Synchronization can be intra-medium (or rate synchronization) where it pertains to maintaining the natural rate of the source (e.g., 64 Kbit/sec audio), or inter-media where it guarantees that the explicit (user specified) or implicit (as in audio and video) time relationships between different media types are enforced.

On the basis of different time scales between the synchronization requirements of different media types, one can also distinguish between fine-grain and coarse-grain synchronization. The latter refers to cases where the misadjustment tolerances are larger than those posed by video and audio (which are in the order of tens of milliseconds). An example of this case is the display of a still image and its associated text.

The most difficult task is arguably the fine-grain, inter-media synchronization between video and audio, as the tolerances prescribed by human perception criteria are very tight. Here we describe the algorithms that we have developed to attack this problem. The results we obtained were very good, as judged by subjective evaluation. As mentioned in Section 2.4, silence detection may be employed to help maintain a low average end-to-end delay between restart operations. The use or not of silence detection changes the algorithm slightly; both cases are analyzed below.

2.5.1 The Synchronization Problem

The objective of a synchronization algorithm can be stated as follows. Let $t^a_{o_i}$ and $t^p_{o_i}$ be the time of acquisition and playback of the i-th object of type o. Then, for accurate synchronization the following conditions must hold:

1. $t^p_{o_i} - t^p_{o_{i-1}} = t^a_{o_i} - t^a_{o_{i-1}}$ for all i and o (intra-medium synchronization), and

2. $t^p_{o_i} - t^p_{k_i} = t^a_{o_i} - t^a_{k_i}$ for all i, o and k (inter-media synchronization).

Since obtaining the time from a computer—especially a multitasking one—does not guarantee accuracy, both acquisition and playback times can only be approximated.

The above conditions can then only be approximately satisfied.

In view of time-stamp uncertainty, the task of our synchronization algorithm is to ensure that the following conditions hold:

1. $t^p_{o_{i-1}} \leq t^p_{k_j} < t^p_{o_i}$ if $t^a_{o_{i-1}} \leq t^a_{k_j} < t^a_{o_i}$ for all o, k, i and j, and

2. $t^p_{o_i} < t^p_{o_j}$ if $t^a_{o_i} < t^a_{o_j}$ for all i and j.

Of course the latter is simply an ordering condition. We note that the above conditions bound the synchronization misalignment by the time interval required for two successive media acquisitions. As the performance of the system increases (e.g., a higher video frame rate can be supported), the synchronization accuracy increases. Such coupling of accuracy to CPU performance is both intuitive and desirable, given the rapid pace with which CPU instruction speeds increase. We should also note that the video frame period is a very natural "unit of measurement" of temporal misalignment. For example, for very high frame periods (e.g., in the order of a second), it does not make sense to consider synchronization at a fine scale, since it becomes perceptually irrelevant.

2.5.2 Audio/Video Synchronization Algorithm

At the acquisition phase, both audio and video frames are time-stamped with millisecond resolution before they are delivered to the network. Time-stamping occurs immediately after acquisition; this implies that the time-stamp for audio marks the end of the audio frame rather than its beginning. Note that it is essential that time references are always based on the same clock, to avoid clock synchronization requirements. For the audio signal which is subject to continuous sampling, intra-medium synchronization has to be used to ensure that the full 64 Kbps rate is serviced. This is done at the acquisition point, by simply always reading the full contents of the audio input buffer.

As data transmission takes place, multiplexed video and audio frames are received by the Xphone system and are delivered to the appropriate media playback routines. The audio frames are always immediately submitted to the audio output buffer. Due to possible non-zero buffer occupancy, the actual playback time may vary. The synchronization decision is performed for video frames only, and is based on both time-stamps and audio buffer occupancy.

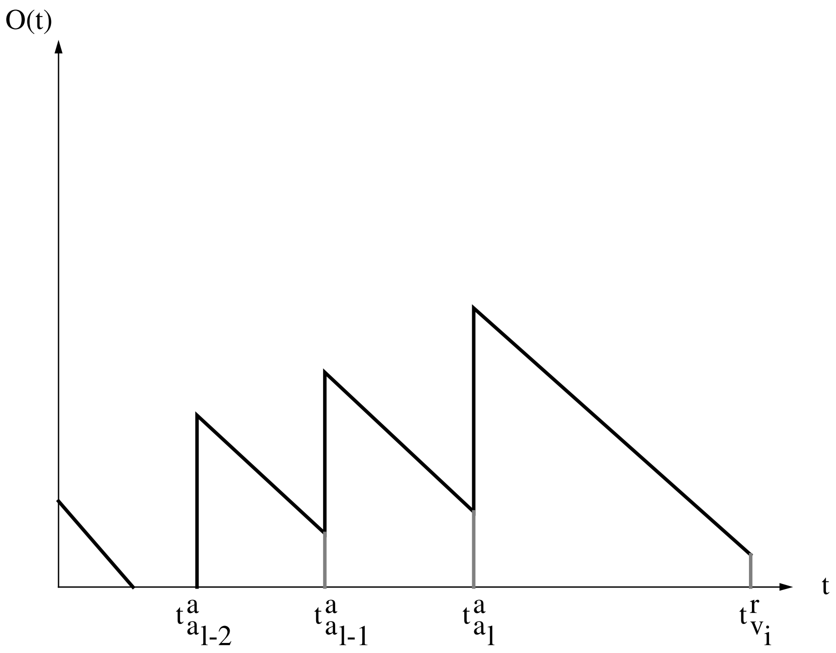

Figure 2-12: Audio output buffer occupancy.

Let $t^r_{v_j}$ and $t^a_{v_j}$ denote the reception and acquisition time-stamps of the j-th video frame respectively, with similar notation for the audio frames ($t^r_{a_i}$ and $t^a_{a_i}$). Let also $O(t)$ denote the output buffer occupancy at time t in audio samples, and r_a the playback rate (here 8,000 samples/sec). In Figure 2-12 we depict the audio buffer occupancy evolution until the time of the j-th video frame's arrival $t^r_{v_j}$.

The first task of the algorithm is to position itself in the playback time-line. To that end, it must find the acquisition time-stamp of the currently played (or last played, if the output buffer is empty) audio frame, which may not be the most recently received. For this purpose, a finite history of received audio frames is kept, and the audio output buffer occupancy is queried ($O(t^r_{v_j})$). This audio frame history is scanned until an audio frame k is found which satisfies:

$$\sum_{i=k}^{l} L(a_i) \geq O(t^r_{v_j}) > \sum_{i=k+1}^{l} L(a_i) \tag{2.3}$$

where $L(a_i)$ denotes the length of the i-th audio frame in samples, and l is the most recent audio frame received.

2.5.2.1 Synchronization with no Silence Detection

We assume now that silence detection is not used, and distinguish between two different cases: 1) $O(t^r_{v_j}) = 0$, and 2) $O(t^r_{v_j}) \neq 0$. The first case implies that the audio output buffer is in "starved" state (with the corresponding consequences in the end-to-end delay), and that the last audio frame has already been played out. The synchronization algorithm then decides to drop or queue the video frame, depending on if $t^a_{a_k}$ is greater or less than $t^a_{v_j}$ respectively (note that the audio time-stamp refers to the end of the audio frame).

In the second case ($O(t^r_{v_j}) \neq 0$), the decision has three branches, i.e., to drop, playback or queue. The criteria are:

1. if $t^a_{v_j} < t^a_{a_{k-1}}$ then drop,
2. if $t^a_{a_{k-1}} \leq t^a_{v_j} < t^a_{a_k}$ then play back, and
3. if $t^a_{a_k} \leq t^a_{v_j}$ then queue.

When no information is available for the $(k-1)$-th audio frame, then the estimate $t^a_{a_k} - L(a_k)/r_a$ is used instead of $t^a_{a_{k-1}}$. Due to time-stamp inaccuracy, incorrect decisions may be made if this estimate were used all the time. Clearly, as the end-to-end delay increases, both the video and audio output queues will increase in occupancy. Whenever the synchronization decision is initiated, it processes all video frames currently resident in the video output queue until it is either empty, or a frame that has to be queued up (wait) is found.

2.5.2.2 Synchronization with Silence Detection

When silence detection is used, the audio frame headers of completely silent frames are still transmitted to the receiver; the length of frames that partially consist of silence is conveyed in the frame header. In this case, an audio buffer occupancy of zero does not always designate starvation, since it may correspond to a silent part. Moreover, this silent part may belong to an audio frame that has not yet arrived, and hence it is not possible to accurately decide if the situation is normal or abnormal.

For this reason, the three-part decision described above is employed in all cases, except when the audio buffer occupancy is zero and the last audio frame received was not entirely silence (note that when the occupancy is zero, the last audio frame received is always the reference one). When this happens and the video acquisition time-stamp satisfies:

$$t^a_{a_{k-1}} \leq t^a_{v_j} < t^a_{a_k} \tag{2.4}$$

(criterion 2), then the video frame is dropped. In the case where a video frame is queued up but the following audio frame is silence, then this algorithm will cause a slight delay in the video frame's playback; since this however corresponds to silence, there is no synchronization problem.

We should note that with the above bounded synchronization scheme, long-term intra-medium synchronization for video is maintained, although its short-term, local accuracy is traded-off with audio/video inter-media synchronization. This tradeoff is more heavily pronounced with a high video/audio interleave factor, which in turn depends on the video acquisition hardware and the audio input buffering. In our system this factor is typically 1:1 or 2:1. For much higher values—or if the video encoding/decoding processes are highly asymmetric in terms of delay—an extra control step would be required for intra-medium synchronization of video packets during the time their associated audio packet was played back. For normal video frame rates (up to 30 fps) and a configurable audio input buffer it is always possible to enforce a low interleave factor, and hence avoid any such complication.

2.6 Concluding Remarks

We have presented the architecture and the algorithms used in the Xphone multimedia communication system. This system assumes the use of a best-effort operating system and network, and provides for synchronized video/audio playback with bounded and minimized end-to-end delay, as well as source bit rate adaptation to the network load. The major contributions of this chapter can be summarized as follows:

1. The novel concept of intra-application scheduling for multimedia communication systems was introduced, and was shown to be essential for the transparent support of continuous, multimedia data flow, as well as for achieving seamless integration with interactive windowing environments.

2. A non-linear quantizer-rate model was proposed to describe the relationship of quantizer step size and compressed bit rate for typical head-and-shoulders video sequences.

3. An adaptive rate control mechanism was introduced in order to maximize the perceptual video quality according to the available network bandwidth, and also to adapt to changing video display window sizes. The algorithm trades-off spatial quality in order to achieve a sufficiently high frame rate.

4. We have shown that silence detection is a very efficient mechanism for reducing the average end-to-end delay. Further bounding of the maximum acceptable end-to-end delay can be achieved using an appropriate restart protocol, albeit with temporary disruption of communication.

5. We have introduced a novel synchronization algorithm for computer-based environments which, based on time-stamps, bounds the temporal misalignment by the video frame period. The algorithm guarantees that as CPU speeds scale up, so will synchronization accuracy. The approach also shows that very good video/audio synchronization is possible, even with no real-time support.

Using the above techniques, the current system's implementation was shown to achieve a frame rate of 8 frames/sec for a frame size of 320×240, an average bit rate of 1 Mbit/sec (full-duplex), and an average end-to-end delay of 250-300 msec.

Further improvements can be achieved by a number of ways. For example, the end-to-end delay can be further minimized by reducing the audio acquisition buffer size, by applying a more sophisticated silence detection algorithm, and by coding the companded audio signal. Quality can also be improved by applying echo cancellation techniques; the current levels of end-to-end delay can generate undesirable echo phenomena. Experimentation has shown that their severity depends heavily on the quality of the audio equipment used, and it can even be completely eliminated with the use of a high-quality lavalier microphone. The video source rate bit rate control algorithm can also be improved by providing a mechanism that will enable adaptive calibration of the bit rate model been used.

Finally, a considerable improvement in overall performance can be attained by substituting the TCP layer with an unreliable one that would, however, be able to recover in cases of errors with no retransmissions. Although full reliability is desirable for audio, packet losses can be tolerated for video. As the three primary components of multimedia communication systems (codec, computer and network) have competing requirements, techniques such as those described in this chapter will become essential for their smooth cooperation.

Chapter 3

Data Partitioning

3.1 Introduction

The traditional problem in video coding for the past several decades has been that of compression: describe the signal with as few bits as possible. The signal is treated as a single waveform, while compression may be subject to transform and/or prediction techniques [67, 57]. In several cases, however, it is beneficial to segment the original signal into multiple parts, and handle each one independently. Such an approach was originally applied to speech using the so-called sub-band coding approach [67], which partitioned the signal into multiple frequency bands. The primary motivation is that, since the human aural system perceives the various frequency bands in different ways, one could apply different compression techniques to each of the sub-bands.

A more general application of this principle is the so-called pyramidal, or hierarchical approach [91]. Here the signal is again decomposed into a number of different layers, but now each layer represents a successive refinement of the previous one. During compression, each layer is formed by compressing the difference between the original signal and its reconstructed version up to the particular layer. Consequently, an equivalent point of view is that each layer represents the compression error of its immediately lower layers. This representation is again typically (but not necessarily) lossy, hence leaving a residual compression error. In sub-band coding approaches, the individual layers are orthogonal to each other (or approximately so, depending on the filter bank used). For this reason, compression of the different layers can occur in parallel.

The benefits and applications of both approaches are numerous. A key feature is that they can facilitate interoperability. A classical example from the analog world is that of monochrome and color television receivers: the base layer here is the luminance signal (the monochrome component), while the added layers are the chrominance. The two layers are transmitted separately (modulated at slightly different frequencies), and hence allow monochrome receivers to process color signals by "decoding" only the luminance part.

A more modern example from the digital domain is compatible High Definition Television coding: a layered approach allows the base component to be compatible with standard resolution television receivers, and hence a single signal can be used to service both systems.

Another important feature of layered compression is that it can be made robust to channel errors. In particular, one can associate a better transmission environment for the base layer, and less protected ones for the higher layers. In some transmission environments this association is directly supported. In packet-based networks for example, it is possible to mark higher layer packets so that, when congestion is created in intermediate routers or switches, they are dropped first. In systems that are not capable of prioritized transmission, one can essentially emulate the same effect by using different levels of forward error correction. By utilizing more efficient error-correcting codes for the lower layers, one can ensure—given some assumptions about the channel "noise"—that the base layer will arrive intact at the receiver.

Examples of such channels are over-the-air broadcast, as well as individual virtual circuits within packet-based networks.

A significant drawback of layered approaches is that they are, in general, less efficient than their single-layer counterparts. In other words, for a particular signal to noise ratio (SNR), it is better to compress a signal using a single layer than multiple ones. “Better” here implies that fewer bits are required to represent the signal. In several applications, however, this is acceptable due to the added flexibility. We should also note that layered approaches also tend to be much more expensive to implement that single-layer ones, due to the added encoding and decoding complexity.

From a multimedia communications perspective, the most important drawback of layering is that it is embedded in the encoding method. In other words, the layers can only be constructed during the encoding process; doing so at a later stage would require significant computational resources, in essence consisting of full decoding and recoding. There are several reasons, however, to desire a layered structure even without support from the encoder. First, due to the cost of hierarchical (or scalable) encoding, it is likely that single-layer, general-purpose encoders will dominate in actual systems. Second, the exact partitioning point in terms of bit rates is not obvious, due to the potentially large number of channel types over which the signal may be transported. Finally, due to the loss in compression efficiency it is likely that, in applications such as video-on-demand, only a single-layer high-quality version of the signal will be utilized (stored for retrieval by the users). It is then important to examine approaches in which layering is provided after encoding has already taken place.

In this chapter we examine one such approach, called “data partitioning”, which is applicable to any block-based, transform coding scheme. Our primary focus will be the MPEG-2 coding scheme [6], which provides direct support in its bitstream syntax to effect data partitioning, and in which partitioning was first introduced.

We analyze the problem of optimal data partitioning using an operational rate-distortion approach. The optimal algorithm is characterized, and is shown to have significantly high complexity and delay, as a result of the temporal structure of predictive compression. A "causally optimal" algorithm based on Lagrangian multipliers is described; it optimally solves the problem when the additional constraints of causal operation and/or low-delay are imposed. A memoryless version of the algorithm, theoretically optimal for non-predictive compression only, is shown to perform almost identically but with significantly lower computational complexity. Finally, a fast, suboptimal algorithm using "rate-based" optimization is also proposed, and is shown to perform quite close (within 1 dB) to the causally optimal one.

The structure of this chapter is as follows. In Section 3.2 we present an overview of motion-compensated transform coding, with particular emphasis on the MPEG scheme. Although the description is somewhat extensive, this will help us delineate the data partitioning problem, and define it in mathematical terms. It will also be useful for the development of Chapter 4, where the properties of the coding algorithm are again at the core of the analysis. In Section 3.3 we introduce the problem of data partitioning, and formulate it in an operational rate-distortion context. In Section 3.4 we present the optimal solution for non-predictive coding, whereas in Section 3.5 we analyze the more general predictive coding case. The chapter concludes with Section 3.6, where a summary of the major contributions is presented.

3.2 Motion-Compensated Transform Coding

The algorithmic foundation of practically all state-of-the-art, general-purpose video coding schemes today is transform coding [91, 67]. The effectiveness of the technique is evidenced by its incorporation into all modern image and video compression standards, namely MPEG-1 [4], MPEG-2 [6], H.261 [3], and JPEG [5]. The key idea in transform coding is to map the original signal into a domain where compression becomes easier. Since the objective of compression is to eliminate redundancy, the optimal transformation is one that completely decorrelates the source signal.

Assume that we have N samples of a zero-mean signal:

$$\mathbf{x} = [x(0), x(1), \ldots, x(N-1)]^T \tag{3.1}$$

If $\{\mathbf{\Phi_i}\}$ is a set of linearly independent vectors spanning the N-dimensional vector space, then x can be expanded in terms of $\mathbf{\Phi_i}$'s as:

$$\mathbf{x} = \sum_{i=0}^{N-1} X(i)\mathbf{\Phi_i} \tag{3.2}$$

where $X(i)$ is the coefficient of expansion given by

$$X(i) = \frac{\langle \mathbf{x}, \mathbf{\Phi}_i \rangle}{\langle \mathbf{\Phi}_i, \mathbf{\Phi}_i \rangle} \tag{3.3}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. Clearly, the vector $\mathbf{x}$ can be equally well represented by either the N samples x_i in the original domain, or the N numbers $X(i)$ (or coefficients) in the space spanned by the basis functions $\mathbf{\Phi}_i$. If we assume that only the first M ($M < N$) coefficients in (3.3) are non-zero, then we will be able to represent our signal in the $\{\mathbf{\Phi}_i\}$ space with just M coefficients, and hence achieve compression.

3.2.1 The Optimal Transform—Karhunen-Loeve

In practice, the calculation of optimal basis vectors $\mathbf{\Phi}_i$ is quite involved. Let $\hat{\mathbf{x}}$ be the coded representation of $\mathbf{x}$, i.e.,

$$\hat{x} = \sum_{i=0}^{M-1} X(i)\mathbf{\Phi}_i \tag{3.4}$$

The Mean Squared Error (MSE) of the coded representation can then be described by

$$\epsilon = E\left[(x - \hat{x})^2\right] \tag{3.5}$$

Assuming that $\mathbf{x}$ is real and that the $\mathbf{\Phi}_i$ are orthonormal, it can be shown [108] that minimization of the MSE leads to the following eigenvalue problem:

$$(C_{\mathbf{xx}} - \mu_i I)\mathbf{\Phi}_i = 0, \qquad i = 0, 1, \ldots, N-1 \tag{3.6}$$

where $C_{\mathbf{xx}}$ is the autocovariance matrix of the random vector $\mathbf{x}$:

$$C_{\mathbf{xx}} = E[\mathbf{xx}^T] \tag{3.7}$$

The set of basis vectors $\mathbf{\Phi}_i$ that are solutions of (3.6) form the so-called Karhunen-Loeve Transform (KLT) expansion, and the matrix

$$\mathbf{\Phi} = [\mathbf{\Phi}_0, \mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_{N-1}] \tag{3.8}$$

is the KLT matrix of $\mathbf{x}$. Due to (3.6), $\mathbf{\Phi}$ diagonalizes $C_{\mathbf{xx}}$, i.e.,

$$\mathbf{\Phi}^{-1} C_{\mathbf{xx}} \mathbf{\Phi} = \mathrm{diag}\,[\mu_0, \mu_1, \ldots, \mu_{N-1}] \tag{3.9}$$

The MSE due to the truncation in (3.5) can be written as

$$\epsilon = \sum_{i=M}^{N-1} \mu_i \tag{3.10}$$

and is minimized by ranking the eigenvalues μ_i in a descending order.

Note that the vector $\mathbf{X}$ of coefficients $X(i)$ can be directly expressed as

$$\mathbf{X} = \mathbf{\Phi}^{-1}\mathbf{x} \tag{3.11}$$

hence the term transform. The KLT is optimal, in the sense that [108]: 1) it completely decorrelates the signal in the transform domain, 2) it minimizes the MSE in bandwidth reduction or data compression, 3) it contains the most variance (energy) in the fewest number of transform coefficients, and 4) it minimized the total representation entropy of the sequence.

As we can see from eq. (3.6), the appropriate set of basis functions $\{\mathbf{\Phi}_i\}$ depends directly on the signal's autocovariance matrix, hence the signal's statistics. The problem can be solved analytically only in the simplest of cases, when particular models are assumed for $\mathbf{x}$ [108]. A well-known example is Markov-1 signals, whose autocovariance matrix has the form:

$$[C_{\mathbf{xx}}]_{ik} = \rho^{|i-k|} \qquad i, k = 0, 1, \ldots, N-1 \tag{3.12}$$

for $0 < \rho < 1$, where ρ is the adjacent sample correlation coefficient.

3.2.2 The Discrete Cosine Transform

The dependence of the optimal transform on the source signal makes KLT an impractical approach for real applications. Of interest, however, are constant transform matrices who are good approximations to the KLT in particular cases. One such approach, which has also proven to have perceptually good performance in image and video compression, is the Discrete Cosine Transform (DCT). The N point forward DCT transform $\mathbf{X}$ of a vector $\mathbf{x}$ is defined by[1]:

$$X(m) = \sqrt{\frac{2}{N}} k_m \sum_{n=0}^{N-1} x(n) \cos\left[\frac{(2n+1)m\pi}{2N}\right], \qquad m = 0, 1, \ldots, N-1 \tag{3.13}$$

where

$$k_m = \begin{cases} 1/\sqrt{2} & \text{if } m = 0 \text{ or } N \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

It can be shown that the DCT is a unitary transform, and hence its inverse is its transpose:

$$x(n) = \sqrt{\frac{2}{N}} k_n \sum_{m=0}^{N-1} X(m) \cos\left[\frac{(2n+1)m\pi}{2N}\right], \qquad n = 0, 1, \ldots, N-1 \tag{3.15}$$

Defining the DCT matrix as

$$[C]_{mn} = \sqrt{\frac{2}{N}} k_n \sum_{m=0}^{N-1} X(m) \cos\left[\frac{(2n+1)m\pi}{2N}\right], \qquad m, n = 0, 1, \ldots, N-1 \tag{3.16}$$

we have

$$\mathbf{X} = C\mathbf{x} \quad \text{and} \quad \mathbf{x} = C^{-1}\mathbf{X} = C^T\mathbf{X} \tag{3.17}$$

A key property of the DCT is that it is asymptotically equivalent[2] to the KLT for Markov-1 signals, when the correlation coefficient ρ tends to 1 [108]. Note that this property holds independently of the value of N. As a result, the higher

[1] This is the DCT Type II; other versions are also possible (see [108]) but have not found significant use in image and video compression.

[2] Asymptotic equivalence here is formally defined under the weak Hilbert-Schmidt norm of a matrix, which is proportional to the sum of the squares of its eigenvalues.

the correlation of the source, the better the performance DCT will provide for compression purposes.

3.2.3 Video Compression Using Transform Coding

For the purposes of image and video compression, a two-dimensional transformation must be applied. This is done in a separable fashion. Let

$$[\mathbf{x}]_{mn} = x(m,n), \qquad m,n = 0,1,\ldots,N-1 \tag{3.18}$$

be a two dimensional $N \times N$ signal. Its 2-D DCT is defined by the $N \times N$ matrix:

$$\mathbf{X} = C\mathbf{x}C^T \tag{3.19}$$

The DCT is typically applied in non-overlapping blocks of an image or video frame (or picture), as shown in Figure 3-1. Extensive experimentation has shown that a good tradeoff in terms of complexity and performance is obtained with block sizes of 8×8 pixels (for picture sizes of CCIR 601 [1] resolution, i.e., 704×480 pixels for the US television system) [91].

The DCT-transformed blocks of the picture are then subject to quantization. Due to the particular sensitivities of the human visual system, quantization is perceptually, or frequency weighted. In other words, the quantization step size is not uniform across the different transform "frequencies". Typically, the DC coefficient (the signal's mean) is subject to very fine quantization, whereas AC coefficients are quantized with progressively coarser step sizes. A typical quantizer weight matrix is shown in the following table 3.1 [6]. The actual quantization step size can be controlled, in order to achieve a particular target bit budget or rate (for the case of video). Denoting by $X(i,j)$ the DCT coefficient, $\tilde{X}_{ij}$ its quantized version, W_{ij} the

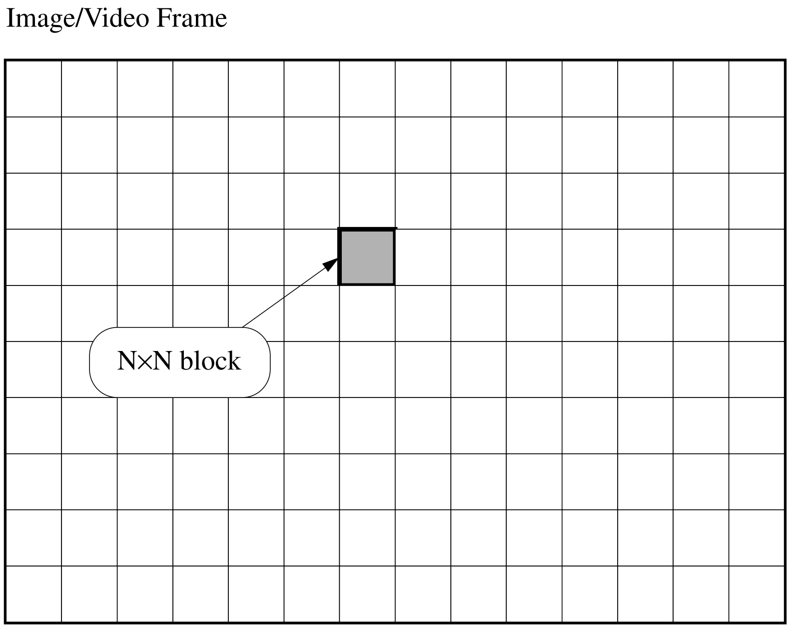

Figure 3-1: Segmentation of images or video frames into blocks.

quantizer weight, and by Q the so-called quantizer scale, we have:

$$\tilde{X}(ij) = \left\lfloor \frac{\alpha X(i,j)}{W_{ij}Q} + \frac{1}{2} \right\rfloor \tag{3.20}$$

where α is a parameter (for the weight values of Table 3.1 we would have $\alpha = 16$).

Quantization is the single source of quality loss in transform coding (hence making it a "lossy" compression technique). In general, the frequency distribution in the DCT domain tends to be concentrated around the DC component, with higher-order AC coefficients having values very close to zero. By using a large step size for these coefficients, their quantized representation becomes zero, making compression much easier. Again, due to the lower sensitivity of the human visual system at these frequencies, the resultant distortion is typically non-perceptible (for small to medium compression ratios).

	0	1	2	3	4	5	6	7
0	8	16	19	22	26	27	29	34
1	16	16	22	24	27	29	34	37
2	19	22	26	27	29	34	34	38
3	22	22	26	27	29	34	37	40
4	22	26	27	29	32	35	40	48
5	26	27	29	32	35	40	48	58
6	26	27	29	34	38	46	56	69
7	27	29	35	38	46	56	69	83

Table 3.1: Typical quantizer weights for frequency-weighted quantization (larger weights correspond to coarser quantization).

After quantization, the DCT coefficients of each block are scanned in a zig-zag pattern. A typical configuration is shown in Figure 3-2; small variations of this pattern are also used. The zig-zag scan is used to create an ordered, one-dimensional array of DCT values. The ordering priority is essentially the perceptual significance of each DCT coefficient.

After the zig-zag scan phase, the resultant one-dimensional array is run-length encoded [91, 29]. In other words, we convert the array to a representation in which we jointly describe the number of consecutive zero-valued DCT coefficients and the value of the next non-zero coefficient. Such a run-length is depicted in Figure 3-2. This representation tends to be quite compact, since many coefficients (particularly higher-order AC ones) have been zeroed out due to the quantization process.

The encoder then utilizes Huffman coding [29, 91] to efficiently code the run-lengths. Huffman codes employ variable length codewords (VLCs): the most likely run-lengths are assigned shorter codewords, while less likely ones are assigned longer ones. This results, on the average, to fewer bits per symbol compared to a fixed-length representation. The resultant VLCs are directly inserted in the bitstream. We should note that for the particular case of the DC component, differential PCM

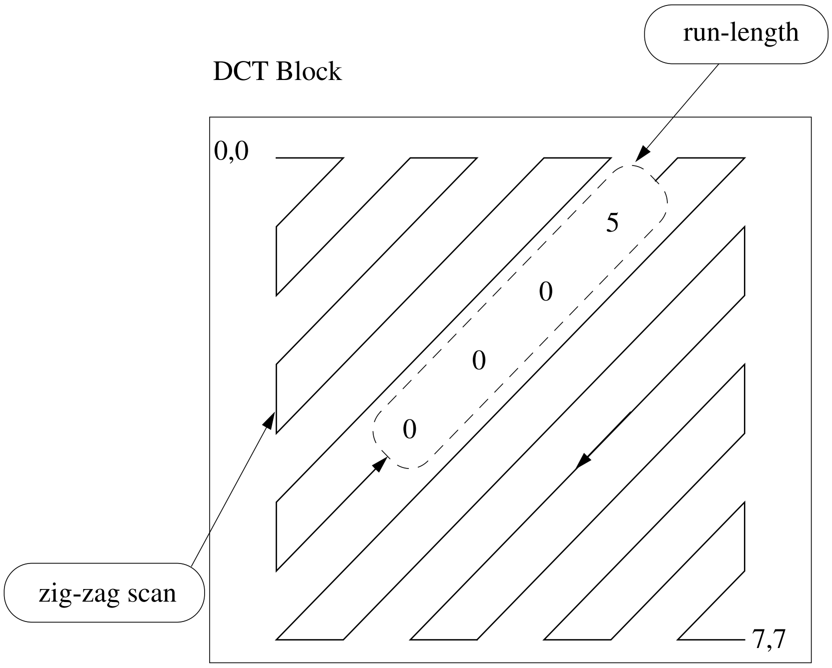

Figure 3-2: Zig-zag scanning pattern of transform coefficients.

encoding is used; i.e., the encoder uses the value of the previous block as a prediction, and only encodes the resultant difference.

3.2.4 The MPEG Picture Structure

The way compressed data is structured in the particular case of MPEG is of importance for the operation of the algorithms presented later on. We should note that this structure is not used exclusively in MPEG, but in other schemes as well (e.g., in H.261 [3], which has minor modifications).

Up to now we have considered the video pictures without regard to the color component. In reality, each individual picture is in fact composed of three elementary ones, one per color component. In video compression the YUV format [91] is exclusively used, and is a linear transformation of the more familiar RGB color space. This is not just the result of an analog tradition: because the Y (or luminance) component contains the fundamental picture structure, it is extremely useful for signal processing purposes (e.g., motion estimation and compensation as we discuss later on). In addition, because the human visual system is less sensitive to the quality of the chrominance components U and V, they can be represented with fewer samples than Y. Typically only one sample of each of U and V is used for every square block of 4 pixels of the luminance component, resulting in the so-called 4:2:0 chroma format. Essentially, the U and V components are subsampled by 2 in each direction. Other formats include 4:2:2, in which the U and V components are horizontally subsampled by a factor of 2, and 4:4:4 in which all components have exactly the same number of samples in both directions.

The MPEG picture structure follows a hierarchical approach. Each layer of the hierarchy utilizes a header, followed by the data of lower layers. An exception is the lowest layer (the block), which does not employ any header information for efficiency purposes. Individual blocks are delineated using variable length end-of-block markers after all DCT run-length codes have been inserted in the bitstream. As shown in Figure 3-3, at the lowest level we have the 8×8 block, which is the unit of transform coding. A 2×2 square of luminance blocks—together with their associated 2, 4, or 8 chrominance blocks, depending on the chroma format—forms a macroblock, which is the unit for quantizer selection and motion compensation (see Section 3.2.5). A horizontal strip of a contiguous series of macroblocks forms a slice, which is the unit for bitstream synchronization.

All recursively computed quantities (except in the temporal dimension), for example DC predictors, are reset at the start of each slice, thus allowing the decoder to quickly recover in cases of channel errors. Although a slice is shown in Figure 3-3 to span the whole width of the picture, this does not have to be the case. A slice cannot, however, span multiple lines of macroblocks. Finally, the set of coded slices forms the picture.

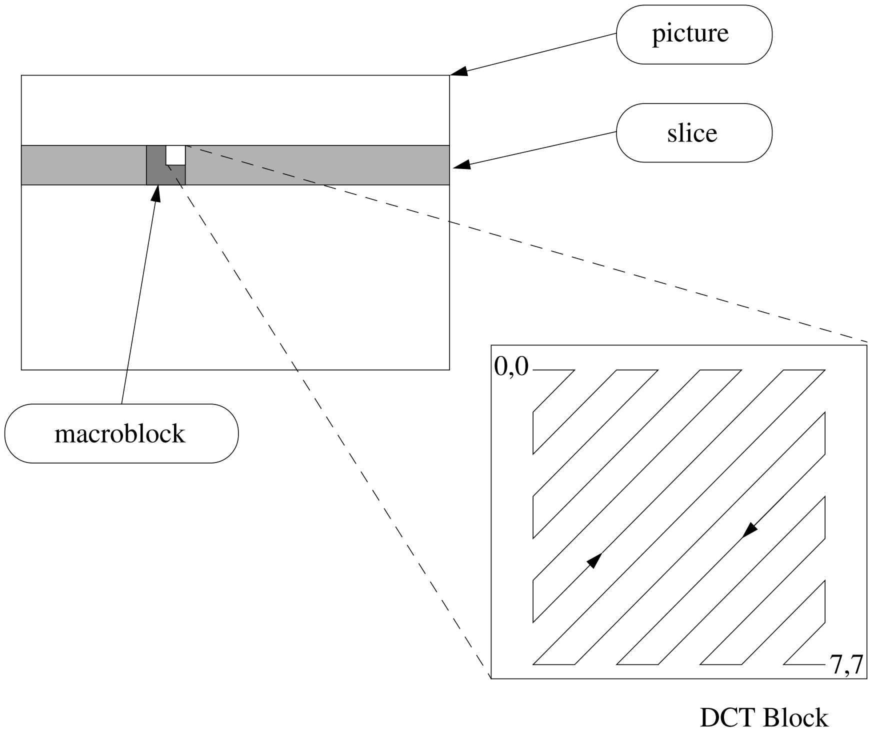

Figure 3-3: The MPEG Picture Structure.

We should also note that individual video frames can be either progressive or interlaced. In the latter case, the frame is composed of two temporally displaced fields: the odd field consists of the even-numbered lines (0, 2, and so on), while the even field consists of the odd-numbered ones. It is important to observe that the two fields represent sampling of the scene at two different time intervals. MPEG allows for the fields to be coded either together as a single frame, or individually if it is preferential to do so. In our discussions both in this chapter and also in Chapter 4, interlacing will not play a significant role.

3.2.5 Motion Compensation

The compression algorithm, as described above, does not take into account any temporal redundancy that may exist between successive pictures. It is evident, however, that such redundancy exists and that it may be extremely helpful in achieving very high compression ratios. Traditional techniques to remove redundancy in the temporal dimension use a recursive approach, in the form of prediction.

Based on the past history of the (one or two-dimensional) signal, an estimate of its future value is formed. The encoder then computes the difference between the actual and predicted values, and transmits it after appropriate encoding. If the prediction is successful, the energy of the prediction signal will be very small, hence reducing the required transmission bit rate and increasing the quality. A significant amount of work has been done for the design of predictors for various types of stochastic signal models [97, 98], with particular emphasis on speech applications.

Very simple zero-order prediction is typically used for video signals, due to their particular structure. In order to improve performance, however, motion is taken into account in the prediction process, thus giving rise to motion-compensated predictors. Here the prediction for any given pixel is not formed just by the value of the collocated pixel in the previous frame (as in a simple linear predictor), but by a spatially displaced one. The displacement from the collocated pixel is a two dimensional quantity, and is appropriately termed *motion vector*. These motion vectors are calculated at the encoder, and are embedded in the compressed bitstream for use by the decoder. At the decoder side, the motion vectors are used to form the predicted pixel values, to which the decoded prediction error will be added to reconstruct the final picture. It has been shown [22, 23] that the same transform coding technique used for the original images is near optimal for the case of prediction error (difference) images as well (the KLTs are identical assuming a first-order Markov model for the source).

In MPEG, motion estimation (or rather compensation, since the standard prescribes only the decoding algorithm) is performed on a macroblock basis. In other words, all the pixels of a macroblock share the same motion vector. This represents a tradeoff between motion estimation accuracy and the overhead bits required to transmit the motion vector values to the receiver. Several different techniques can be used to compute the motion vector values; the most popular and effective one is full-search block matching [91]. For each macroblock of a picture, we try to find a similar one in a small neighborhood of the previous picture. The search range is typically ± 16 to ± 32 pixels in each direction. Similarity is defined by the sum of the absolute values of the difference of the pixels (L_1 norm). Letting $x_t(m,n)$ denote the pixel value of picture t, the prediction error is defined by:

$$E(\Delta x, \Delta y) = \left\{ \sum_{i=0}^{15} \sum_{j=0}^{15} |x_t(m+i, n+j) - x_{t-1}(m + \Delta x + i, n + \Delta y + j)| \right\} \quad (3.21)$$

and the optimum motion vector is given by:

$$(\Delta x^*, \Delta y^*) = \arg \min_{\Delta x, \Delta y} E(\Delta x, \Delta y) \qquad |\Delta x|, |\Delta y| \leq S \quad (3.22)$$

where S defines the (typically symmetric) search range. Note that motion estimation is exclusively performed on the luminance component; motion compensation, however, is applied to all three color components.

In order to improve motion-compensated prediction, non-integer motion vector values can also be used. In this case pixels that do not fall exactly on the sampling lattice of the signal are interpolated from neighboring ones, typically using bilinear interpolation [91].

The above two approaches, i.e., transform coding and motion-compensated transform coding, provide very good compression efficiency. It is possible, however, to improve even further by using a slightly more complex prediction approach, allowing bidirectional motion estimation. Here the encoded picture is not predicted only from a past picture, but from a future one as well. This obviously requires that: 1) the appropriate coding delay is sustained, and 2) a frame reordering is utilized during transmission, so that the future picture is encoded prior to the current one.

In bidirectional prediction we can use either of three different modes: prediction from the past picture, prediction from the future picture, or prediction (interpolation) from both. In the first two cases, only one motion vector needs to be sent to the decoder. In the third, two such motion vectors will have to be sent. In all three cases, appropriate flags must be available in the bitstream so that the appropriate prediction mode can be signalled. When the interpolative prediction mode is used, the prediction block is formed as the average of the past and future picture blocks. It has been found that the interpolative prediction mode is the most effective in achieving high compression ratios in MPEG-1 and MPEG-2.

Motion vectors are encoded in a differential fashion, similar to DC coefficients of individual blocks. In other words, their difference from the previously computed motion vector is encoded, using Huffman coding. These motion vector predictors are reset at the start of each slice, similarly to all other recursively computed quantities.

In the MPEG compression scheme, the above three prediction options give rise to three different picture categories. Pictures that are encoded with no prediction are called intra, or I pictures. Those that are predicted from past pictures are called predicted or P pictures. Finally, those that are bidirectionally interpolated are called B pictures. The I picture is typically employed in a periodic fashion by encoders, so that any errors introduced by transmission and propagated due to the recursive nature of the decoding process are eliminated. The I picture period is typically 12 or 15 frames, while the P-picture period is typically 2 to 3 frames.

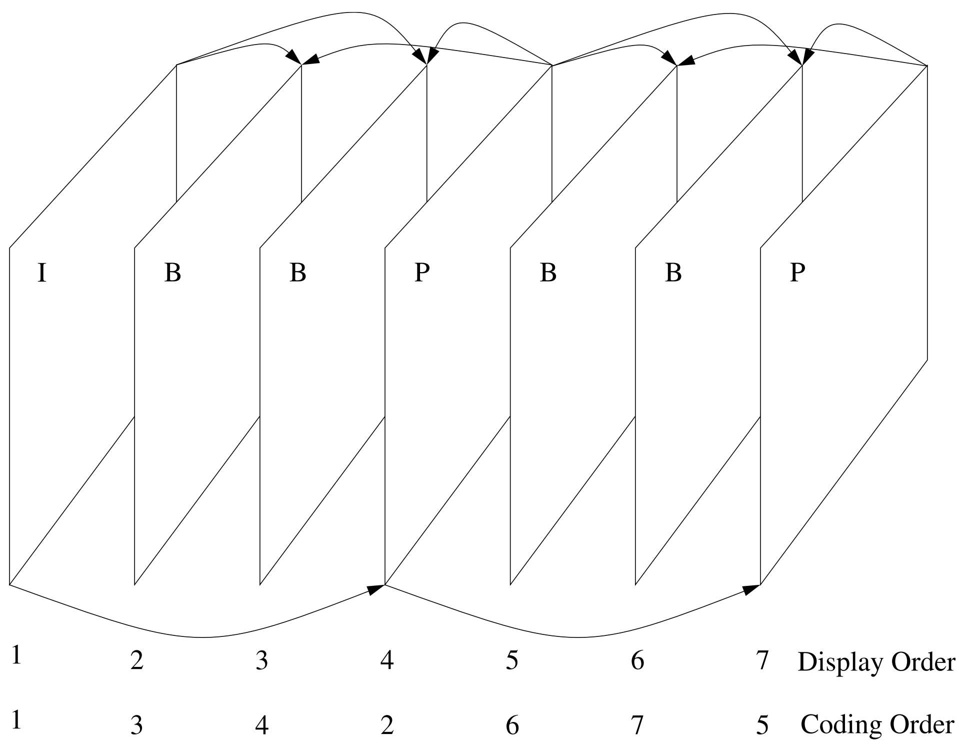

Figure 3-4: The MPEG sequence structure (display and coding orderings).

A typical sequence structure for MPEG is shown in Figure 3-4. The first frame is coded independently (I or intra), the fourth frame is predicted from the first (P), while the intervening second and third frames are interpolated from the first and fourth frames, and so on. Observe that the order in which the frames (or, more accurately, their prediction errors) are transmitted (coding order) is not the same as the one in which they are displayed (display order). Since the fourth frame (P) is used as a reference for the second and third (B), it has to be coded (and transmitted) first. This entails a coding delay equal to three frames in this particular case, so that the encoder has access to the P frame following the B ones.

The sequence structure is strictly hierarchical; in other words, I-pictures are independently coded, P-pictures are predicted from the closest past I or P picture,

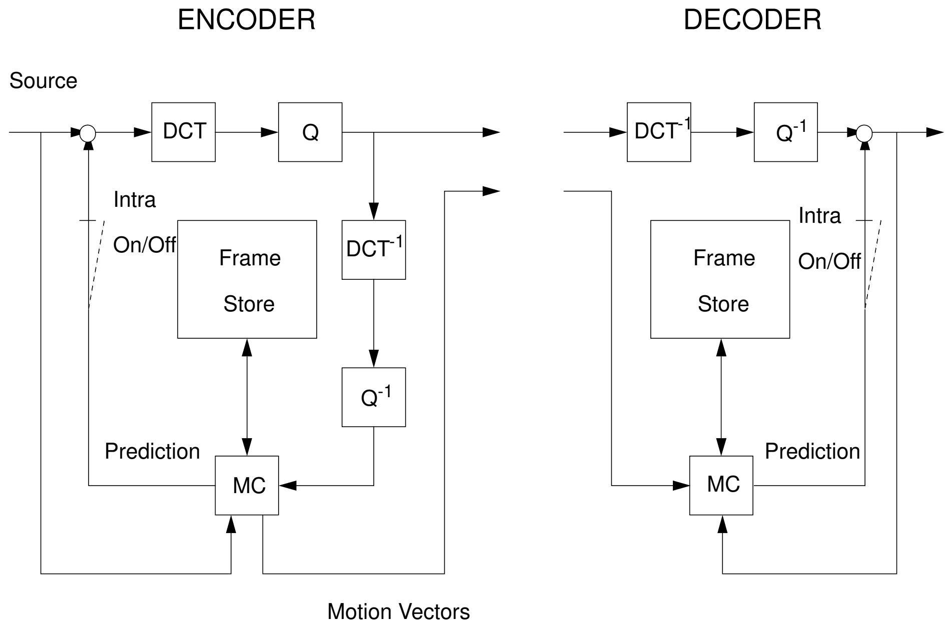

Figure 3-5: Block diagram of a motion-compensated transform coder.

while B picture are interpolated from the closest past and future I or P pictures. We should note that, within a particular picture type, not all of the macroblocks have to be encoded with the same prediction type. For example, it is possible to have intra-coded macroblocks (i.e., not predicted by any mode) in both P and B pictures.

In closing, we should mention that the above overview, although extensive, is by no means exhaustive. In particular, many details of the MPEG compression scheme were omitted, particularly with respect to features targeting interlaced video compression. As we mentioned earlier, however, interlacing does not have any effect on the results presented in this thesis, and hence these issues will be ignored. A block diagram of a motion-compensated transform encoder-decoder pair is shown in Figure 3-5.

3.3 The Data Partitioning Problem

3.3.1 Data Partitioning

Data partitioning is a feature of the MPEG-2 draft standard that provides for the segmentation of a coded signal bitstream into two components or partitions [6]. It can be a very effective tool for the transmission of video over channels that allow selective protection of each of the partitions. Channels of this type can be implemented, for example, using increased forward error correction, or employing high priority transmission in an ATM-based networking environment. By transmitting the most critical information with high reliability, i.e., over the highest quality channel, the average quality of the signal reconstructed at the receiver can be significantly increased for the same level of channel distortion. This feature is one of the major benefits of pyramidal or—more generally—hierarchical, multi-layer coding schemes.

An important characteristic of data partitioning is that it can be employed even after encoding has taken place, in contrast with other hierarchical approaches, such as the SNR, spatial, or temporal scalability modes of MPEG-2 [6], or the embedded DCT coding approach proposed in [109]. This is because the encoder does not need to maintain a prediction loop per each signal layer, a necessary requirement for a pyramidal scheme in which each coding layer is an enhancement of its previous one. As a direct consequence, it is also less robust in the sense that neither partition is self-contained; loss of information in either one will cause error propagation and accumulation during the decoding process if temporal predictive/interpolative modes are used. As we will see later on, error accumulation can in fact be kept under control. Although data partitioning is currently supported only in MPEG-2, it is trivial to incorporate into other coding schemes.

The system diagram of the data partitioning scheme is shown in Figure 3-6. In between an MPEG-2 encoder/decoder pair, the bitstream (assumed here to be coded

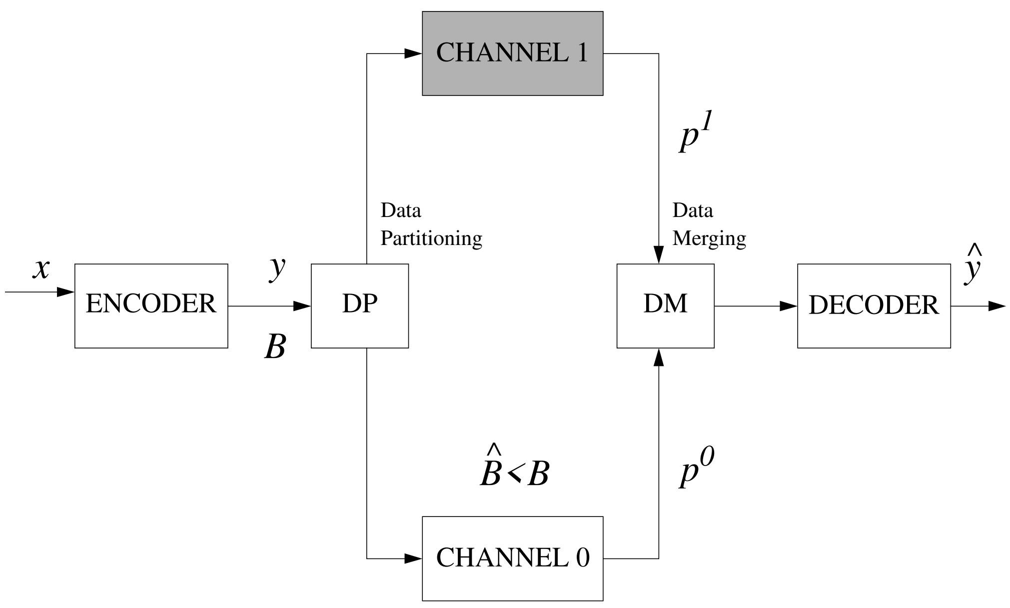

Figure 3-6: Block diagram of a Data Partitioning system.

at the constant rate of B Mbps) is split into two parts, each being transmitted on a different "channel". Throughout this chapter we will assume that channel 0 is a perfect one, i.e., it exhibits no losses, errors, or insertions. We will also assume that it has a given fixed available bandwidth $\hat{B} < B$. Channel 1 is assumed to exhibit arbitrary stochastic behavior (a minimax problem formulation will make the details of this behavior irrelevant).

In other words, we are given a bitstream of bit rate B, but our communication resources only allow us to reliably transmit at a bit rate of $\hat{B}$. The problem is then how to optimally split the bitstream into two parts, the base one complying with the rate constraint $\hat{B}$, so that the quality of the decoded signal at the receiver is maximized.

Partitioning is performed at well-defined points in the bitstream syntax [6], called *breakpoints*. These can occur at various levels of the bitstream hierarchy. For our purposes, and to ensure that partition 0 is independently decodable, we will constrain the allowable breakpoint positions so that critical quantities such as macroblock address increment (indicating the relative position of a macroblock with respect to the previously coded one) and DCT DC differential values (for intracoded macroblocks) are included in partition 0. As a result, partitioning will only affect the number of coefficient run-length codes that will be carried in partition 0, while the rest will be assigned to partition 1. This is depicted in Figure 3-7.

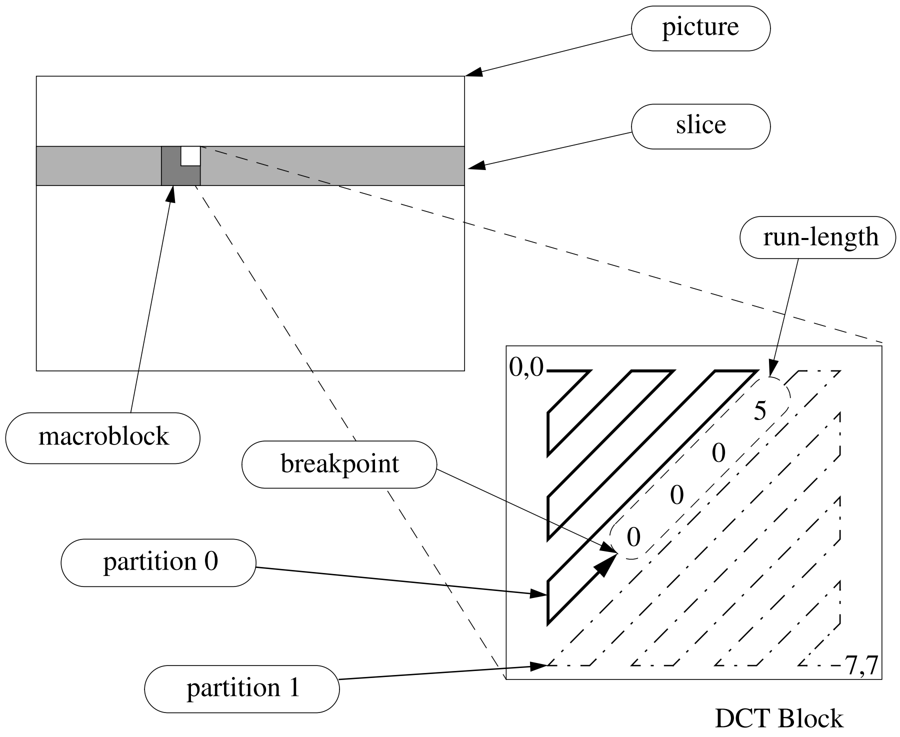

Figure 3-7: Breakpoint position in the zig-zag pattern of DCT coefficients.

Note that, in MPEG-2, the breakpoint value is the same for all blocks of a given slice. The breakpoint value, i.e., a fixed-length code indicating the number of run-length codes that are included in partition 0, is included in the slice header. Sequence headers are replicated in partition 1 to increase robustness, and hence the total rate for the transmission of the signal is slightly increased. In partition 1, the breakpoint value is set to 0 since it is not needed. We now proceed to a mathematical formulation of the problem.

3.3.2 General Problem Formulation

Denoting by y the coded video signal, by $\hat{y}$ the output of the decoder, by p^i the signal of the i-th partition, and by $R(\cdot)$ the bit rate, the problem of optimal data partitioning can be expressed as follows:

$$\min_{R(p^0) \leq \hat{B}} \{\|y - \hat{y}\|\} \tag{3.23}$$

The metric $\|\cdot\|$ above denotes the squared error criterion:

$$\|\mathbf{x}\| \equiv \mathbf{x}^T\mathbf{x} = \sum_{i=0}^{N-1} x(i)^2 \tag{3.24}$$

and is applied only in the luminance component. The rate constraint, however, refers to all three color components. Since channel 1 is assumed to exhibit stochastic behavior, we consider the deterministic problem of minimizing the *maximum average distortion* D, i.e.,

$$\min_{R(p^0) \leq \hat{B}} \left\{ D \stackrel{\text{def}}{=} \max\{\|y - \hat{y}\|\} \right\} \tag{3.25}$$

This corresponds to the case where the entire partition 1 is lost. D will be referred to in the sequel as the "partitioning distortion".

The optimization window in (3.25) is not specified, and it can span from just a part of a picture, up to any number of pictures. In general, and taking into account that data partitioning as described here is performed after encoding has taken place, it is desirable to keep the end-to-end delay low. Computational complexity considerations impose additional constrains on the window length, as will be made evident later on. Consequently, we will typically be interested in solutions of (3.25) that consider up to a single complete picture.

An important aspect of the problem not readily evident in (3.25) is its recursive nature, caused by the corresponding recursive process with which y and $\hat{y}$ are generated (decoded) when P and B pictures are involved. In the following we separately consider the two cases: optimal data partitioning in non-predictive coding (I pictures only), and optimal data partitioning in predictive coding (I, P, and B pictures).

3.4 Data Partitioning in Non-Predictive Coding

3.4.1 Problem Formulation

In intra-picture only partitioning, there is no temporal dependence between pictures. Consequently, the partitioning distortion will simply consist of the DCT coefficients that were assigned to partition 1. As we noted in Section 3.2.2, the DCT matrix is unitary, i.e.,

$$C^T C = I \tag{3.26}$$

As a result, the energy of the signal in the spatial domain is equal to its energy in the transform domain. In other words, and considering the one-dimensional case for simplicity, if:

$$\mathbf{X} = C\mathbf{x} \tag{3.27}$$

then

$$\mathbf{x}^T\mathbf{x} \equiv \sum_{i=0}^{N-1} x(i)^2 = \sum_{i=0}^{N-1} X(i)^2 \equiv \mathbf{X}^T\mathbf{X} \tag{3.28}$$

Now let b denote the truncation point, i.e., all DCT coefficients from b up to $N - 1$ are moved to partition 1. Considering the truncated—in the DCT domain—

representation $\tilde{\mathbf{x}}$ of $\mathbf{x}$ and using (3.28) we have:

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| = \|\mathbf{X} - \tilde{\mathbf{X}}\| = \sum_{i=b}^{N-1} X(i)^2 \tag{3.29}$$

eq. (3.29) provides an expression for the truncation, or partitioning distortion directly in the DCT domain. Generalization to two dimensions is straightforward.

Let us now consider the partitioning distortion in two dimensions for a group of blocks. We recall that the breakpoint values are identical for all blocks of a given slice i (Section 3.3.1). This value will be denoted by b_i, and indicates that the b_i-th and higher-order DCT run-length codes of all blocks of this slice will be removed and placed in partition 1. The domain of b_i is the set of values $\{0, 1, \ldots, 64\}$. Since blocks are transformed independently, the partitioning distortion for a set of blocks will simply be the sum of the partitioning distortions of the individual blocks.

Denoting the DCT coefficient of the k-th run-length of block j of slice i by $X_j^i(k)$, and the set of blocks that belong to slice i by $\mathbf{S}_i$, we can express the partitioning distortion for a particular slice as:

$$D_i(b_i) = \sum_{j \in \mathbf{S}_i} \sum_{k \geq b_i} X_j^i(k)^2 \tag{3.30}$$

Note that this is a function of the breakpoint value b_i. Since error calculations are only done on the luminance component, we will assume that, for chrominance blocks,

$$D_i(b_i) = 0 \tag{3.31}$$

Returning to eq. (3.25), we can now explicitly express the problem of minimizing the maximum partitioning distortion D as:

$$\min_{R(p^0)\leq\hat{B}} \{\max\{\|y-\hat{y}\|\}\} \Longleftrightarrow \min_{\sum_{i=1}^{S} R_i(b_i)\leq\hat{B}} \left\{\sum_{i=1}^{S} D_i(b_i)\right\} \tag{3.32}$$

where $R_i(b_i)$ denotes the rate required to encode slice i when the breakpoint value b_i is used, and S is the total number of slices considered (may span several pictures).

Our objective here is to find those values b_i^*, $i = 1,\ldots,S$ that minimize the maximum distortion as given in (3.32). An exhaustive search would be clearly impossible, as the number of possible combinations that would have to be examined can be huge (65^S). We recall that data partitioning is typically applied immediately prior to transmission (when the value of $\hat{B}$ becomes known), and hence complexity and delay considerations are very important.

3.4.2 The Optimal Algorithm

This constrained minimization problem can be solved using the approach of Lagrange multipliers [50]. A similar algorithmic approach but in a different context has been used in [73, 74, 126]. The Lagrange multipliers approach converts the constrained optimization problem to an unconstrained one, by adding more dimensions to the parameter space. Consider the following problem. Given a constaint B, find

$$\min_{b\in A} D(b) \tag{3.33}$$

subject to

$$R(b) \leq B \tag{3.34}$$

Then the following theorem holds [50].

Theorem 1 *For any $\lambda \geq 0$, the solution $b^*(\lambda)$ to the unconstrained problem*

$$\min_{b \in A} \{D(b) + \lambda R(b)\} \tag{3.35}$$

is also the solution to the constrained problem (3.33)–(3.34) with the constraint $B = R(b^*(\lambda))$, that is, with $R(b) \leq R(b^*(\lambda))$.

The proof is quite simple and can be found in [50]. It is important to properly appreciate the consequences of what Theorem 1 provides. In particular, it does not guarantee any solution to the constrained problem (3.33)–(3.34) (in other words, the two problems are not equivalent). It only indicates that for every nonnegative λ, there is a corresponding constrained problem whose solution is identical to that of the unconstrained one. If, however, $R(b^*(\lambda))$ happens to be equal to B, then $b^*(\lambda)$ is the desired solution for the constrained problem.

Since the constraint B in our problem is given (the reliable channel bandwidth $\hat{B}$), our algorithm will have to find an appropriate value for λ so that $R(b^*(\lambda) = B$. Since the domain of b in our case is discrete, such an exact solution may not be attainable. We will consequently be satisfied with a solution for which $R(b^*(\lambda))$ is as close as possible to B.

Returning to our original problem, we can rewrite (3.32) as an unconstrained problem as follows

$$\min \left\{ \sum_{i=1}^{S} D_i(b_i) + \lambda \sum_{i=1}^{S} R_i(b_i) \right\} \tag{3.36}$$

By defining the per-slice quantity

$$L_i(\lambda, b_i) \stackrel{\text{def}}{=} D_i(b_i) + \lambda R_i(b_i) \tag{3.37}$$

the above can be rewritten as

$$\min\left\{\sum_{i=1}^{S} L_i(\lambda, b_i)\right\} \tag{3.38}$$

We observe that, given a particular λ, the minimization problem above can be solved independently for each slice, since $L_i(\lambda, b_i) \geq 0$. In other words, each L_i can be minimized independently of the others. This structure helps to significantly reduce the complexity of the problem. Within each particular slice, one can even use exhaustive search to obtain the optimum breakpoint value b_i^*, since the possibilities are limited (65 in the worst case). Hence the complexity of the problem becomes proportional to $65S$, where S is the number of slices (recall that for a full-search it is 65^S).

In order to find the optimal solution b_i^*, however, we must also find the appropriate value λ^* for λ. This can be accomplished using an iterative bisection algorithm [73, 74]. The algorithm starts with two initial estimates for λ (typically its extreme values 0 and ∞), and continuously refines its estimate until convergence is achieved.

Figure 3-8 illustrates the procedure for the simple case of a single slice. The graph shows the various rate-distortion points (marked with "x") when different breakpoint values are selected. For example, point A (first from the left) corresponds to the breakpoint value $b = 0$, i.e., to the case where all DCT coefficients are carried in partition 1. This gives rise to a particular partitioning distortion D and rate R that is required to represent the slice, as depicted in the figure. The rate is non-zero, since there are also overhead bit for headers etc. Similarly, point C corresponds to the breakpoint value $b = 2$. Since more DCT coefficients are included in partition 0,

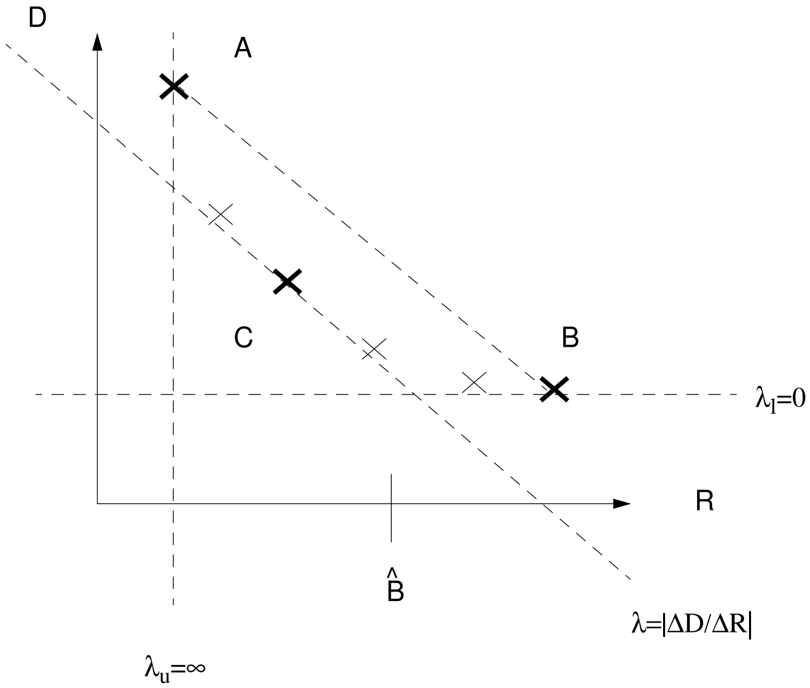

Figure 3-8: Overview of the bisection algorithm.

the rate is slightly increased but the distortion is reduced. Hence the $R(D)$ curve[3] is monotonically non-increasing. The curve does not necessarily have to have 64 points, since typically only a small number of run-lengths are needed to encode each block.

We should note that these $R(D)$ curves are not results of a stochastic minimization problem as in rate-distortion theory [29, 14], but discrete point curves that result from actual compression and differ from slice to slice. This is the reason why the term "operational" rate distortion minimization is used to differentiate it from the stochastic case. An $R(D)$ curve obtained from actual data is shown in Figure 3-9 at page 93.

[3] Although the term "$R(D)$ curve" will be used throughout this thesis, we should note that, strictly speaking, it is imprecise as it implies continuity.

As initial values for λ we select the two extreme cases $\lambda_l = 0$ and $\lambda_u = \infty$ (the subscripts are for "lower" and "upper' respectively). In the former case the minimum is achieved by independently minimizing the distortion, and hence the optimal breakpoint for this value of λ (denoted by $b^*(\lambda)$) is obtained by using the maximum possible value of b: $b^*(0) = b_{\max} \leq 64$. This solution is indicated at point B in Figure 3-8. In the case $\lambda_u = \infty$, the minimum is achieved by independently minimizing the rate, corresponding to $b^*(\infty) = 0$ or point A of Figure 3-8[4]. At points A and B we also show the lines that pass from these points and have as a slope the negative value of their corresponding λ. Observe that these points minimize the expression: $D(b) + \lambda R(b) + c$, and hence for some value of the constant c (for the particular λ) the optimum solution is *on* the line, while all other points are above it.

We observe that our initial estimates $R(\lambda_l)$ and $R(\lambda_u)$ contain the desired target rate $\hat{B}$, which ensures that the problem is feasible. The next step is to select a new value for λ, which can be done in any number of different ways. Lacking any a priori information on the $R(D)$ curve, and given its high variability from slice to slice, a plausible selection is the slope of the line segment interconnecting our original points A and B. We thus have:

$$\lambda_{\text{next}} = \frac{D(\lambda_u) - D(\lambda_l)}{R(\lambda_l) - R(\lambda_u)} \tag{3.39}$$

Next, we minimize $D(b) + \lambda R(b)$ for this particular λ, and obtain as a solution, say, point C. Note that the new optimal breakpoint value will be between those of points A and B.

[4] Using this formulation, and for purposes of precision, the rate should be exactly 0; one can always, however, subtract the overhead bit rate from the rate constraint $\hat{B}$, ensuring this way that $R(0) = 0$.

We then examine which of the intervals

$$[R(b^*(\lambda_u)), R(b^*(\lambda_{\text{next}}))) \quad \text{and} \quad (R(b^*(\lambda_{\text{next}})), R(b^*(\lambda_l))] \tag{3.40}$$

contains our target bit rate $\hat{B}$, and repeat the procedure from the start using these new values of λ as starting points. If it turns out that $R(b^*(\lambda_{\text{next}})) = \hat{B}$ then we have found an exact solution. In practice, convergence will occur when the new $R(D)$ point coincides with one of the initial two points.

We now present the detailed description of the algorithm, applicable to any number of slices. Since in an actual implementation it is more convenient to deal directly with the number of bits instead of the rate, in the following we can consider that rate-related quantities refer to just quantities of bits. The two are proportional to each other, related by a normalization constant, and hence the two interpretations are equivalent.

We denote by $b_i^*(\lambda)$ the optimal breakpoint for slice i for the particular value of λ, and $R_i^*(\lambda)$ and $D_i^*(\lambda)$ the optimal rate and distortion respectively of slice i for the given value of λ, i.e., we have:

$$R_i^*(\lambda) \stackrel{\text{def}}{=} R_i(b_i^*(\lambda)) \quad \text{and} \quad D_i^*(\lambda) \stackrel{\text{def}}{=} D_i(b_i^*(\lambda)) \tag{3.41}$$

We also denote by R_{budget} the target bit budget for the particular set of slices $\{\mathbf{S}_i\}$. We should note that although the average rate of partition 0 has to be less than or equal to $\hat{B}$, there is flexibility on how the target bit budget for any given set of slices is allocated.

Lagrangian Minimization Algorithm

Step 1: Initialization

Set $\lambda_l = 0$ and $\lambda_u = \infty$. If the inequality:

$$\sum_{i=1}^{N} R_i^*(\lambda_u) \leq R_{\text{budget}} \leq \sum_{i=1}^{N} R_i^*(\lambda_l) \tag{3.42}$$

holds as an equality for either side, an exact solution has been found. If the above does not hold at all, then the problem is infeasible (this can happen if the target rate $\hat{B}$ is too small). Otherwise go to Step 2.

Step 2: Bisection and Pruning

Compute:

$$\lambda_{\text{next}} := \left| \frac{\sum_{i=1}^{N} [D_i^*(\lambda_u) - D_i^*(\lambda_l)]}{\sum_{i=1}^{N} [R_i^*(\lambda_u) - R_i^*(\lambda_l)]} \right| \tag{3.43}$$

and find $R_i^*(\lambda_{\text{next}})$ and $D_i^*(\lambda_{\text{next}})$ such that $B_i^*(\lambda_u) \leq B_i^*(\lambda_{\text{next}}) \leq B_i^*(\lambda_l)$.

Step 3: Convergence Test

If

$$\sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) = \sum_{i=1}^{N} R_i^*(\lambda_u) \quad \text{or} \quad \sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) = \sum_{i=1}^{N} R_i^*(\lambda_l) \tag{3.44}$$

then stop; the solution is $B_i^*(\lambda_u)$, $i = 1, \ldots, N$. If

$$\sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) > R_{\text{budget}} \tag{3.45}$$

then $\lambda_l := \lambda_{\text{next}}$, else $\lambda_u := \lambda_{\text{next}}$.

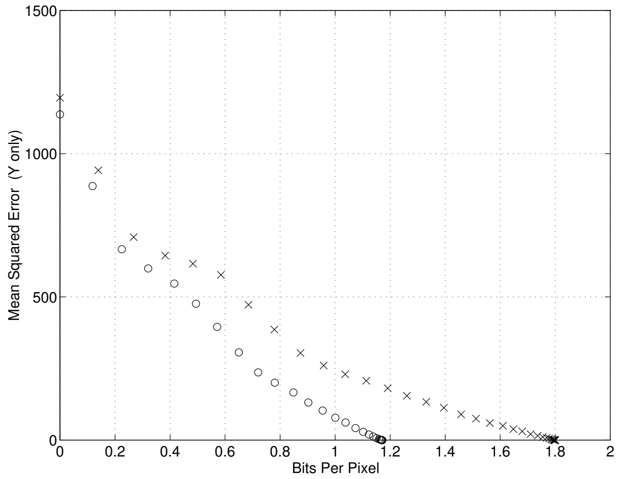

Figure 3-9: Slice 20 (full-width, frame 0) from "Flower Garden", coded at 24 Mbps (x) and 12 Mbps (o).

The bisection algorithm operates on the convex hull of the $R(D)$ curve of each slice. Consequently, points which lie above that, and hence are not $R(D)$ optimal, are not considered by the algorithm. Figure 3-9 shows the $R(D)$ plots for an actual slice (frame-based, intra coding of "Flower Garden" at 24 and 12 Mbps, slice 20—full-width—of frame 0). Worth noting is the locally non-convex behavior in both cases. This property can be traced back to the structure of the MPEG-2 run-length encoding tables [6], where specific examples of non-convexity can be easily found. In most cases (and particularly for P and B pictures as we will see later on), the number of $R(D)$ points which lie above the convex hull is small, and hence in practice they do not represent a significant problem.

In some cases, if the $R(D)$ curve of a slice is sufficiently misbehaved, the bisection algorithm can be set off track, with a resulting underutilization of the target bit budget. In order to mitigate this effect, and also to speed up operation, each iteration considers a continuously shrinking interval of possible breakpoint values ("pruning"). This will result in convergence of the algorithm to a much smaller set of non-convex points, and is a byproduct of convexity.

3.4.3 Performance Evaluation

The collection of necessary data in eq. (3.36) that is needed to run the algorithm, requires only parsing of the input bitstream up to inverse quantization of the DCT coefficients. In other words, all operations can be performed directly in the compressed domain. Note that distortion data are computed from the luminance component only, whereas rate data are computed from all three components. The parsing process represents a very small fraction of the complete decoding process; the dominant processing step in decoding is in fact the inverse DCT.

The window S (number of consecutive slices) in which the algorithm operates has been considered up to now a design parameter. Since data partitioning is performed after encoding, it is desirable to minimize the additional delay introduced by the extra processing step. Even in cases where partitioning is applied on stored material prior to transmission, delay is a very important parameter for interactive applications. A plausible selection is then a complete single picture (frame or field).

As we mentioned in the previous section, the target bit budget R_{budget} can be set quite arbitrarily, given however that the average rate does not exceed $\hat{B}$. This represents another degree of freedom which is not (and cannot be) optimized by the above algorithm. A convenient selection is obtained by choosing for each picture the value

$$R_{\mathrm{budget}} = (\hat{B}/B)R - R_o \tag{3.46}$$

where R is the size (in bits) of the currently processed frame (or,more generally, set of slices), and R_o is the number of bits spent for coding components of the bitstream that are not subject to data partitioning (overhead bits for headers, motion vectors, etc.). Allocated bits that are left over from one picture are carried over to the subsequent picture. Note also that R is immediately available after the complete picture has been parsed.

It is very easy to show that the budget selection in (3.46) guarantees that the target bit rate is not exceeded. We have:

$$R_{\text{budget}} + R_o = \frac{\hat{B}R}{B} \tag{3.47}$$

and the average value $\bar{R}$ of R over time is $\bar{R} = B$. Hence the average rate for the partitioned picture will be:

$$\overline{R_{\text{budget}} + R_o} = \frac{\hat{B}\bar{R}}{B} = \hat{B} \tag{3.48}$$

In addition, we can also show that this allocation can satisfy any scaled buffering constraints that may be imposed. This will be further discussed in Chapter 4.

The value given by eq. (3.46) carries over to the partitioning algorithm several properties of the encoder. In particular, we observe that bit allocation is performed in a manner proportional to the one decided by the encoder. Assuming that an "intelligent" encoder has been used, the original bit allocation may have been meticulously optimized; utilizing a proportional allocation in the partitioning process can help to improve the overall video quality. In the case where buffering constraints are imposed to partition 0 for placement prior to transmission, one can convert the problem to a buffer-constrained optimization problem. The approach would be similar to [96], although the problem there was focused on quantizer selection. It is doubtful, however, that the significant extra complexity of the problem can in fact achieve improved results (an optimal fast algorithm for this problem is not known).

The computational overhead of the iterative algorithm is small, with convergence achieved typically within 8–10 iterations. Figure 3-10 shows the results of applying the algorithm to 20 frames of "Flower Garden", using frame-based intra coding at 24 Mbps, and with a target rate of 12 Mbps for partition 0. The quality metric used is

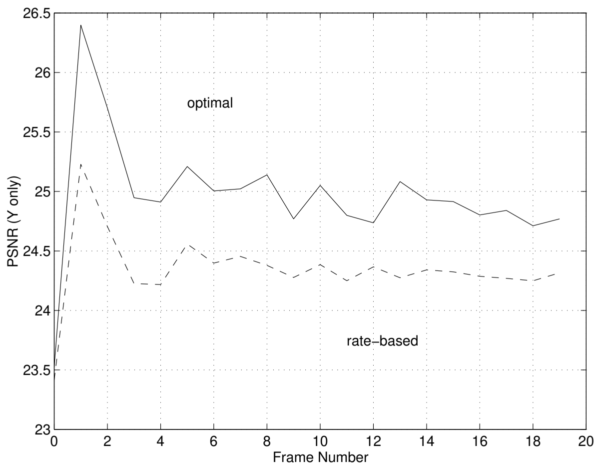

Figure 3-10: Data partitioning of frame-based, intra coded "Flower Garden", from 24 Mbps to 12 Mbps, using optimal and rate-based algorithms.

"Y-PSNR", i.e., the Peak SNR of the luminance component only. PSNR is derived from the squared error $e = \|y - p^0\|$ using:

$$\text{PSNR} = -10\log_{10}\left(\frac{255^2}{e}\right) \quad \text{in dB} \tag{3.49}$$

where 255 is the peak value for the luminance signal (using an 8-bit representation).

Also shown in Figure 3-10 are the results of a simpler algorithm that uses *rate-based* optimization. In this latter case each slice is independently assigned a target bit budget proportional to its original size R_i, and a breakpoint is selected so that this budget is not exceeded (leftover bits are carried over to the next slice). In other words, we select the breakpoint of each slice as:

$$b_i^* = \max\left\{b_i : R_i(b_i) \leq R_{\text{budget}_i}\right\} \tag{3.50}$$

The bit budget for each slice is set according to:

$$R_{\text{budget}_i} = \frac{R_i}{R_{\text{budget}}/S} \tag{3.51}$$

where S is the number of slices. In order to compute R_{budget_i}, a complete picture is read; this makes the algorithms comparable in terms of the optimization window used. Note that this algorithm is purely rate-based, i.e., the distortion is completely ignored. Lagrangian optimization outperforms in this case the rate-based algorithm by 0.6 dB on the average.

3.5 Data Partitioning in Predictive Coding

3.5.1 Problem Formulation

When all variants of picture coding types are used (I, P, and B), the problem of data partitioning becomes significantly more complex. The decoding process (see Figure 3-5 in page 79) can be described by:

$$y_i = \mathcal{M}_i(y_{i-1}) + e_i \tag{3.52}$$

where P_i denotes the i-th decoded picture (in coding order), $\mathcal{M}_i(\cdot)$ denotes the motion compensation operator for picture i, and e_i denotes the coded prediction error. The first picture is assumed to be intra-coded, and hence

$$e_0 = y_0 \tag{3.53}$$

Although, for simplicity, a single reference picture is shown above for motion compensation, the expression can be trivially extended to cover the general case (which includes B pictures).

By applying data partitioning and decoding partition 0, eq. (3.52) becomes:

$$\hat{y}_i = \mathcal{M}_i(\hat{y}_{i-1}) + \hat{e}_i \tag{3.54}$$

where $\hat{e}_i$ denotes the partitioned prediction error. Recall that the original partitioning problem was set in its minimax form in eq. (3.25) of Section 3.3.2 (page 83) as follows:

$$\min_{R(p^0)\leq \hat{B}} \left\{ D \stackrel{\text{def}}{=} \max \left\{ \|y - \hat{y}\| \right\} \right\} \tag{3.55}$$

Using (3.52) and (3.54), and observing that the maximum average distortion is maximized when $\hat{y} = p^0$ (i.e., partition 1 is lost), eq. (3.55) becomes:

$$\min_{\sum_{i=1}^{N} R_i(b_i)\leq \hat{B}} \left\| \sum_{p=1}^{M} \mathcal{M}_i(y_{i-1}) - \mathcal{M}_i(\hat{y}_{i-1}) + e_i - \hat{e}_i \right\| \tag{3.56}$$

where M is the number of pictures over which optimization takes place. Note that:

$$\mathcal{M}_i(y_{i-1}) - \mathcal{M}_i(\hat{y}_{i-1}) \neq \mathcal{M}_i(y_{i-1} - \hat{y}_{i-1}) \tag{3.57}$$

i.e., motion compensation is a non-linear operation, because it involves integer arithmetic with truncation away from zero [6].

From eq. (3.56) we observe that, in contrast with the intra-only case, optimization involves the accumulated error:

$$a_i \stackrel{\text{def}}{=} \mathcal{M}_i(y_{i-1}) - \mathcal{M}_i(\hat{y}_{i-1}) \tag{3.58}$$

Furthermore, due to the error accumulation process, partitioning decisions made for a given picture will have an effect in the quality and partitioning decisions of subsequent pictures. As a result, an optimal algorithm for (3.56) would have to examine a complete group of pictures (I-to-I), since breakpoint decisions at the initial I-picture may affect even the last B or P picture. Not only the computational overhead would be extremely high, but the delay would be unacceptable as well. It is desirable then to seek fast solutions with small delay, that are able to control error propagation in a well-defined fashion.

An attractive alternative algorithm is one that solves eq. (3.56) on a picture basis, and where only the error accumulated from past pictures is taken into account. This algorithm will be referred to as *causally optimal*. In addition, in order to avoid similar complications that arise when the optimization window spans more than one picture, we will restrict our discussion for the case where the windows is up to a complete single picture. This property is also an indirect consequence of the causality argument.

Note that in order to accurately compute a_i, two prediction loops have to be maintained: one for a decoder that receives the complete signal, and one for a decoder that receives only partition 0. This is because of the nonlinearity of the integer arithmetic of motion compensation expressed by eq. (3.57). With the penalty of some lack in arithmetic accuracy, these two loops can be collapsed together. In the following we will assume that the (optimal) dual-loop operation is always used.

3.5.2 The Causally Optimal Problem

The causally optimal problem can now be formulated as follows. Substituting eq. (3.58) in (3.56) we have

$$\min_{R(p^0)\leq \hat{B}} \{\|a_i + e_i - \hat{e}_i\|\} \tag{3.59}$$

We must now obtain an expression for the total partitioning distortion $a_i + e_i - \hat{e}_i$.

As in the non-predictive case of Section 3.4.1, we first consider a single block. Let $A(k)$ denote the k-th DCT coefficient of the accumulated error a (in zig-zag scanning order), $E(k)$ the corresponding coefficient of the decoded picture e, $\hat{E}(k)$

the one of the partitioned picture, and b the breakpoint value. We then have

$$\begin{aligned}
\|a_i + e_i - \hat{e}_i\| &= \sum_{i=0}^{N-1} \left(A(k) + E(k) - \hat{E}(k)\right)^2 \\
&= \sum_{i=0}^{N-1} A(k)^2 + 2\sum_{i=0}^{N-1} A(k)\left(E(k) - \hat{E}(k)\right) + \sum_{i=0}^{N-1} \left(E(k) - \hat{E}(k)\right)^2 \\
&= \sum_{i=0}^{N-1} A(k)^2 + 2\sum_{i \geq b} A(k)E(k) + \sum_{i \geq b} E(k)^2 \qquad (3.60)
\end{aligned}$$

since

$$\hat{E}(k) = \begin{cases} E(k) & \text{if } k < b \\ 0 & \text{otherwise} \end{cases} \qquad (3.61)$$

Observe that the total distortion involves not only the accumulated error and the current picture's partitioning error (identical to the non-predictive case), but crossterms as well.

Due to the independence of individual blocks, we can extend (3.60) for a complete slice. We should note, however, that while the prediction error DCT coefficients are represented by their run-lengths, and the truncation point is also defined by the number of run-length to be included in partition 0, the accumulated error has no such representation. Consequently, a mapping function $\mathcal{I}(k)$ is necessary that maps the k-th run-length of a block into the appropriate position in the zig-zag scanning pattern.

Denoting by $\hat{D}_i(b_i)$ the total partitioning distortion of slice i for the breakpoint value b_i, we have

$$\hat{D}_i(b_i) = \sum_{j \in S_i} \left\{ \sum_{k=0}^{N-1} A_j^i(k)^2 + \sum_{k \geq b_i} 2A_j^i(\mathcal{I}_j^i(k))E_j^i(k) + \sum_{k \geq b_i} E_j^i(k)^2 \right\} \qquad (3.62)$$

where S_i denotes the blocks of slice S_i, $A_j^i(k)$ is the k-th DCT coefficient (in zig-zag scanning order) of the j-th block of the i-th slice of the accumulated error a_i, and

$E_j^i(k)$ is the DCT coefficient of the k-th *run-length* of the j-th block of the i-th slice of the coded prediction error. Using (3.62), the data partitioning problem (3.59) for the predictive case can be formulated as:

$$\min_{R(p^0)\leq\hat{B}} \{\|a_i + e_i - \hat{e}_i\|\} \Longleftrightarrow \min_{\sum_{i=1}^{N} R_i(b_i)\leq\hat{B}} \left\{\sum_{i=1}^{S} \hat{D}_i(b_i)\right\} \tag{3.63}$$

where S is the total number of slices in the optimization window.

3.5.3 The Causally Optimal Algorithm

The minimization problem in (3.63) can be solved using the Lagrangian optimization approach of the non-predictive case in Section 3.4.2, using the more general definition of the distortion $\hat{D}$ given by eq. (3.62).

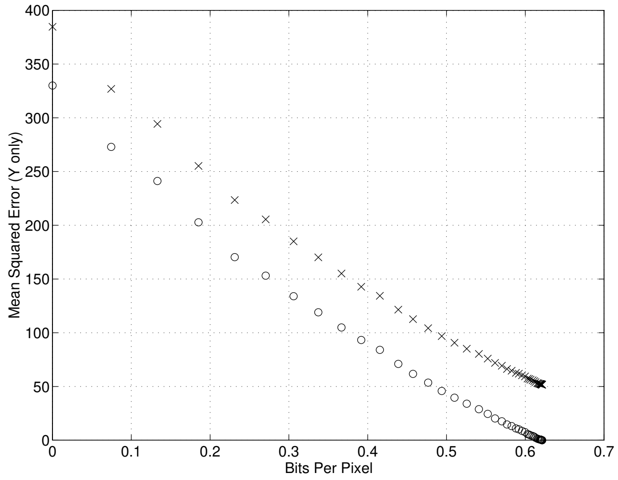

Figure 3-11: Slice 20 (full-width, frame 3, P-picture) from "Mobile" coded at 4 Mbps and partitioned at 3.2 Mbps: (x) $\hat{D}(B_i)$, (o) $D(B_i)$.

Of particular concern is the convexity of the $R(D)$ curves when the total distortion (including the accumulated error) is taken into account. Figure 3-11 shows the R(D) curve for slice 20 of frame 3 (P-picture) from the sequence "Mobile" coded at 4 Mbps (frame-based coding) and partitioned at 3.2 Mbps using the causally optimal algorithm. The upper curve takes into account the accumulated error a_i, whereas the bottom one involves only the prediction error partitioning distortion $e_i - \hat{e}_i$.

We observe that convexity is clearly present. In fact, for predicted pictures, $R(D)$ curves tend to be uniformly convex, compared with intra pictures which tend to have a concave middle segment. We have experimentally verified that this property holds even for small slice sizes (e.g., 11 or 4 macroblocks per slice, instead of the regular 44 which amounts to the whole picture width), although the curves become progressively flatter.

3.5.4 Performance Evaluation

An important issue in mixed-mode coding, as in non-predictive coding, is the target bit budget that will be set for each picture (or more generally, set of slices). The matter is more complicated than in the intra-only case, due to the irregular bit distribution among pictures of different types. In a typical situation, I and P picture DCT coding requires a significant number of bits, while B picture sizes are dominated by header and motion vector coding bits. Figure 3-12 depicts the number of total vs. overhead bits for the "Mobile" sequence coded at 4 Mbps. "Overhead" here includes everything except the DCT coefficients which are subject to partitioning. Observe the evident periodic pattern between I pictures, and the irregularity of the bit distribution between I, P, and B pictures.

As a result of the bit distribution irregularity, B pictures provide much less flexibility for data partitioning. In order to accommodate this behavior, I and P pictures are assigned proportional bit budgets as in Section 3.4.3. For B pictures the same is done, except when the resulting bit budget is negative, in which case it is set to 0. The negative budget, however, is accounted for, so that the bits spent for the B picture are subtracted from the budget of the immediately following picture.

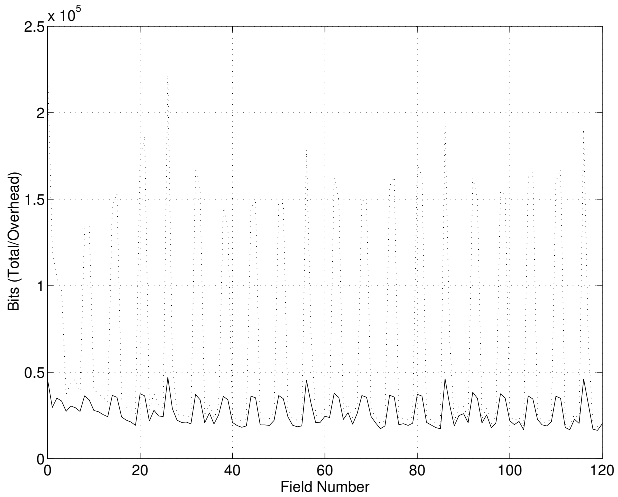

Figure 3-12: Bit distribution for the "Mobile" sequence coded at 4 Mbps, with I period 12, and B period 3 (the overhead bits include all non-DCT bitstream components).

Note that an optimal bit allocation for each picture would be a direct by-product of the optimal non-causal algorithm.

Figure 3-13 shows the Y-PSNR resulting from the causally optimal algorithm on 15 frames of the "Mobile" sequence (I distance N=15, I/P frame distance M=3), frame-based coded at 4 Mbps and partitioned at 3.2 Mbps (80% of the rate goes to partition 0). This is the signal quality that would be observed by a decoder that receives only partition 0, compared with one that receives both partitions. We see that I and P frames suffer the most, while B frames are in general up to 1 dB better.

The complexity of solving eq. (3.63) is significant, as it requires at least one complete decoding loop for the luminance signal. If the non-linearity of the motion compensation is taken into account, then two such loops are required. In addition, since motion compensation is performed in the spatial domain while partitioning is performed in the DCT domain, a forward DCT computation module is required as well in order to compute $A^i_j(\cdot)$. As a result, the implementation complexity is between that of a decoder and an encoder.

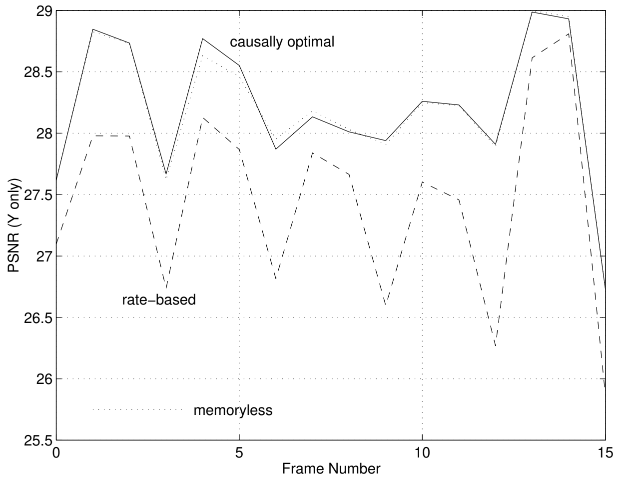

Figure 3-13: PSNR (Y only) for "Mobile" sequence, frame-based coded at 4 Mbps and partitioned at 3.2 Mbps using the causally optimal, memoryless, and rate-based algorithms.

3.5.5 The Memoryless and Rate-Based Algorithms

Given the complexity of the causally optimal algorithm, it is interesting to examine the benefit of error accumulation tracking. This can be evaluated by applying the algorithm of Section 3.4.2 (intra-only case) to the mixed-mode case, since the only difference is the accumulated error term a_i. Bit budget allocation, however, is performed as discussed in Section 3.5.4 (mixed-mode case).

Surprisingly, the results of this *memoryless* mixed-mode partitioning algorithm are almost identical to the causally optimal one. Figure 3-13 shows the relevant PSNR values for the "Mobile" sequence. The difference is in general less than 0.1 dB and the curves can hardly be distinguished. We have experimentally verified that this holds for a very wide range of bit rates (i.e., down to 50% reduction, or more depending on the original rate and picture resolution) and slice sizes. The difference, however, increases slightly to 0.2–0.3 dB. We should note that these difference values are perceptually insignificant.

This is a very important result, as it implies that we can dispense completely with the error accumulation calculation and its associated computational complexity, for a minimal cost in performance. The quality degradation between the causally optimal and memoryless algorithms will be perceptually insignificant, across the spectrum of slice sizes and partition rates.

This property is hinted at by Figure 3-11 (page 101) upon closer examination. The upper and lower curves are almost identical, except for a vertical shift. Figure 3-14 depicts the two types of distortions (from the causally optimal and memoryless algorithms) as well as the accumulated error across all slices of a picture. We observe that the two distortions behave in very similar ways as we move along the picture. In order for the accumulated error to affect the partitioning decisions, either the slope of the $R(D)$ curves or the overall accumulated error distribution across a picture would have to be significantly affected. This, however, is not the case, because at each picture the partitioning decisions are optimally made.

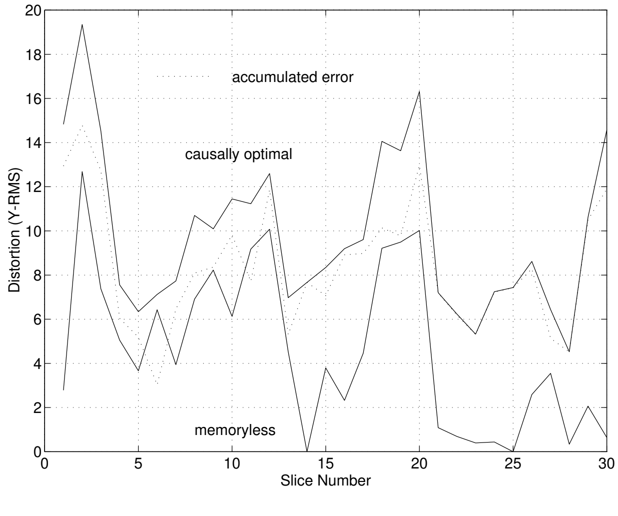

Figure 3-14: Distribution of accumulated error, causally optimal, and memoryless distortions across all slices of a picture ("Mobile", coded at 4 Mbps and partitioned at 3.2 Mbps, frame 3, P picture).

Finally, we examine the performance of the rate-based optimization algorithm introduced in Section 3.4.3 (eq. (3.50)), in a mixed-mode coding environment. Since, as was previously pointed out, rate-based optimization does not take into account the distortion, there is no difference whether or not the accumulated error is tracked. Consequently, the only difference lies in the bit budget allocation. Figure 3-13 depicts the results obtained on the "Mobile" sequence, with the same coding and partitioning parameters as before. We see that the rate-based algorithm is inferior by about 1 dB. The complexity, however, is significantly reduced as well, as the Lagrangian optimization iteration is avoided. This makes the rate-based algorithm attractive, when complexity and/or implementation costs are of importance.

3.6 Concluding Remarks

The problem of optimal data partitioning in motion-compensated transform coding was analyzed, with particular emphasis in its use by the MPEG-2 video coding standard. Data partitioning can be a very effective tool for transmission of single-layer video bitstreams over unreliable channels, including channels that provide prioritized transmission (i.e., virtual channels in packet-based networks). A key property of the approach is that it can be applied even after encoding has already taken place, and thus is applicable not only for live transmission systems, but also for stored video applications such as video-on-demand. A potential drawback of the approach is that, in contrast with other scalable coding approaches, neither of the two partitions in which the bitstream is split is self-contained. Consequently, due to the recursive nature of motion-compensated compression, if part of the bitstream is lost (namely, from partition 1), error accumulation will occur.

The major contributions of this chapter can be summarized as follows:

1. We provided an analysis of data partitioning in an operational rate-distortion context. An optimal algorithm based on Lagrange multipliers was derived for non-predictive (intra-only) coding, and shown to be less complex than a full

decoder.

2. For the predictive coding, or mixed-mode case (I, P, and B pictures) the optimal algorithm was characterized and shown to possess significantly high complexity and delay, as a complete group of pictures was required to be processed at a time. As an alternative, a "causal" minimization formulation was proposed, in which only the accumulated error from past pictures is taken into account (while the one propagated to future pictures is ignored).

3. An optimal algorithm for the causal problem was developed as a generalization of the non-predictive case. Experimental results have shown that the algorithm performs quite well, with P and B pictures having about 1 dB higher quality than I ones.

4. It was then shown that tracking the error accumulation from one frame to the next does not actually benefit the partitioning process in any significant way, and hence that a memoryless algorithm employing Lagrangian optimization is sufficient.

5. Finally, a faster but suboptimal algorithm that uses rate-based optimization was also proposed for comparison purposes. It was shown to perform quite close (within 0.6 dB on the average) to the optimal one for the intra-only case, but proved to be inferior by 1 dB on the average for the mixed-mode case.

The discovery of the extremely good performance of the—relatively simple—memoryless algorithm is an important result, as it drastically reduces the complexity of the algorithm. The significance of this property is strengthened by observing that, as has been experimentally verified, it holds across the whole spectrum of partitioning rate ratios and slice sizes. The results of this chapter will be generalized in the following one, where we discuss the Dynamic Rate Shaping concept.

Chapter 4

Dynamic Rate Shaping

4.1 Introduction

In applications of digital video communications there are many cases where control of the bit rate of video is needed, even after encoding has already taken place. One example is video-on-demand services, in which transmission of precoded material may occur over a wide variety of channels. Users may access the video server using dial-up lines (with a capacity of few tens of Kbps), wireless channels (capacity of few Mbps), traditional local area networks (in the order of 10 Mbps), high-speed LANs (100 Mbps or more). The material that has been encoded and placed in the server may potentially have to be transmitted over all these different types of channels. Using a single-layer coding approach, a decision has to be made for which channel type to optimize the representation. Choosing a low-rate channel will compromise quality for high rate users; choosing a high rate will make the service unusable for low rate users. The alternative of storing multiple versions of the same material at different rates is unattractive, since it multiplies storage requirements. Furthermore, multiresolution coding with too many layers would be undesirable, due to the loss in coding efficiency.

Another example is transmission of real-time or precoded video material over channels with limited or no quality of service guarantees. This includes CSMA/CD LANs (e.g., Ethernet), as well as virtual circuits in ATM networks. Although techniques have been developed to employ rate control for live sources based on network feedback [47, 75] (one such technique is discussed in Section 2.3), no solution is currently available for prerecorded material. Similarly, consider a variable bit rate video source that is fed to an ATM virtual circuit: due to the difficulties in modeling such traffic, the traffic characterization used for admission control and policing will not necessarily match that of the source. Instead of dropping vital information at the source or in internal network nodes (when congestion occurs at intermediate switches), an operation that would manipulate the bitstream so that it complies with what the network can deliver would be an extremely useful proactive measure against resource exhaustion.

Another environment that could potentially benefit from such "post-encoding rate control" operation would be multipoint communication with mobile hosts. Since the mobile link is typically of much lower bandwidth than wired ones, by reducing the video rate at the base-to-mobile link, wired participants would still be able to utilize the full bandwidth available to them without being compromised by the presence of wired ones. The same argument holds for heterogeneous (at least in terms of bandwidth) internetworks.

Finally, an environment that continuously grows in importance is that of general purpose computers. Due to the variety of network transport mechanisms that can be employed and the potential use of video for applications not involving network transmission, it is most likely that general-purpose (transport-independent) video codecs will be used. This includes MPEG-1 and single-layer (non-scalable) MPEG-2. It is desirable, then, to provide a mechanism that can gracefully interface the codec with the particular transport facilities used, if any. Other environments where such techniques would be extremely useful include "trick-mode" operations (fast forward, rewind, etc.), decoder interoperability, transcoding, etc. These will be discussed in more detail in Section 4.6.

In all the above cases, the common theme is the need to manipulate the coded bitstream so that it complies with the bandwidth availability of the underlying communication resources, or the processing capability of the receiver. In general, this manipulation is performed at the transmitting host, just above the transport layer, and interfaces the coded video bitstream with the transport service. It is also possible to perform it in intermediate nodes within the network, which would be specially configured to provide such "value-added" services, similar to today's videoconferencing bridges.

We refer to this rate manipulation operation as *Dynamic Rate Shaping* (DRS). The term dynamic refers to the possibility that rate constraints are time-varying, while shaping is used instead of rate control to: 1) differentiate our approach from that of classical encoder rate control in which the variable rate of an entropy-coded bitstream is matched to a fixed channel rate, and 2) to more accurately capture the posterior (with respect to coding) nature of the operation. Note that DRS is quite different from traffic shaping (e.g., in DRS the traffic's average rate can change). Also, DRS can be used in new types of hybrid guaranteed/best-effort services, such as the ones described in [17].

In order for rate shaping to be viable it has to be implementable with reasonable complexity and yield acceptable visual quality. With respect to complexity, the straightforward approach of decoding the video bitstream and recoding it at the target rate would be obviously unacceptable; the delay incurred would also be an important deterrent. Hence algorithms of complexity less than that of a cascaded decoder and encoder will be sought. These algorithms will ideally operate directly in the compressed domain of the signal. In terms of quality, it should be noted that recoding does not necessarily yield optimal conversion; in fact, since an optimal encoder (in an operational rate-distortion sense) is impractical due to its complexity, recoding can only serve as an indicator of an acceptable quality range. As will be shown, regular recoding can be quite lacking in terms of quality, with DRS providing significantly superior results.

We present a family of algorithms that solve the problem of dynamic rate shaping for motion-compensated transform coders, including MPEG-1, MPEG-2, H.261, and JPEG. We identify two major categories of algorithms, quantizer-based and selective-transmission-based. Our focus is on the latter category, for which we derive several different algorithms, optimal under different circumstances. The operational rate-distortion approach utilized in Chapter 3 is used here as well, and it is shown that data partitioning is a special case of dynamic rate shaping. We also discuss several fast approximations and analyze their performance. It is shown that some of those perform almost identically to their corresponding optimal ones, thus making them excellent candidates for (even software-based) implementation. While the approach is applicable to any motion-compensated block-based transform codec, the MPEG-2 [6] standard is used for all simulation results presented.

The structure of this chapter is as follows. In Section 4.2 we discuss some key properties of the rate behavior of video signals, their impact in communications applications, as well the mechanism of rate control. This section complements Section 3.2, which provided an overview of the motion-compensated transform coding process, and motivates the DRS concept. In Section 4.3, we present a mathematical formulation of the problem of dynamic rate shaping, and identify its several variants. Section 4.4 discusses the Constrained DRS algorithm (optimal when certain structural constraints are imposed) and its variants, and shows that it is a generalized version of the data partitioning problem. Section 4.5 presents the Unconstrained DRS algorithm, which is optimal for the selective-transmission based category. Section 4.6 discusses several possible applications of DRS, and how the results of this chapter would impact them. Finally, we conclude in Section 4.7 with a summary of the major contributions of this chapter.

4.2 The Rate Properties of Compressed Video

The representation of uncompressed digital video requires a constant rate, as a result of the regular three-dimensional sampling lattice and the use of a constant number of bits per pixel and per color component. This rate is typically extremely high, in the order of 200 Mbps depending on the picture size and the chroma format used. By employing compression as with the motion-compensated transform coding techniques discussed in Section 3.2 (or other), one can reduce this rate by several orders of magnitude, down to a few Mbps, with small—if any—perceptually noticeable loss in quality. This "mapping", however, of the signal from its original spatio-temporal domain to the compressed one, significantly distorts its rate behavior. Since the compressed signal will be subsequently transmitted across a network, or handled internally by a local system, it is very important to identify the impact of this rate behavior on overall system design.

4.2.1 Variable Bit Rate Video

As we have seen from the encoding algorithm description in Section 3.2, the compressed video bitstream is composed of three primary components: header information, motion vectors, and DCT coefficients. Header information requires, in general, a roughly constant number of bits across different pictures. Slight variations are possible, since header information also includes prediction modes, macroblock types, etc., which may dynamically vary between pictures or even across the same picture. Motion vectors are encoded differentially, using Huffman—or variable length—codes (VLC). Similarly, DCT coefficients, which are encoded in the form of run-lengths, are also represented via variable length codewords. Figure 4-1 depicts the block diagram of a motion-compensated transform encoder, where the Huffman (VLC) coding stage is shown.

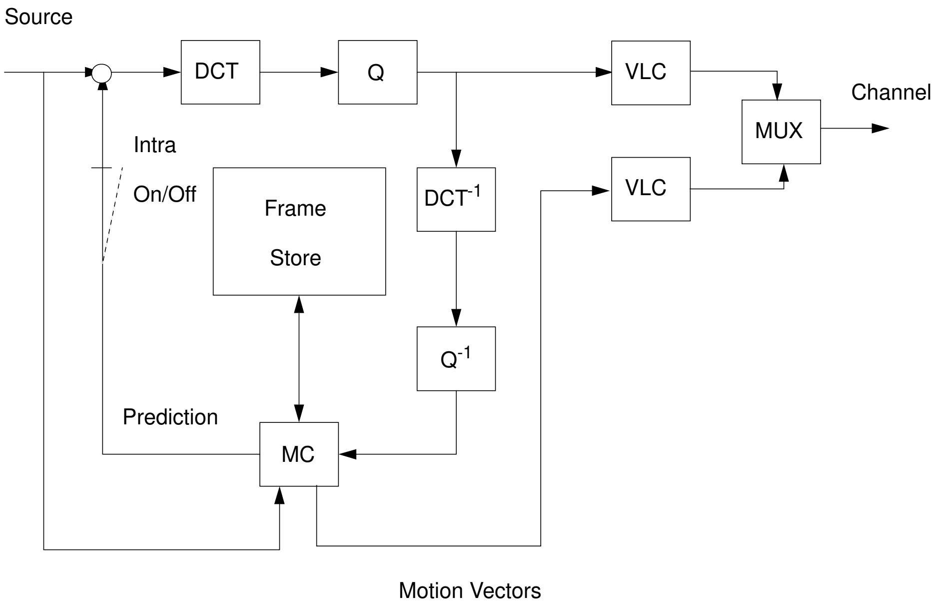

Figure 4-1: Variable bit rate motion-compensated transform encoder.

As a result, the rate of a compressed video bitstream is inherently variable. When the same quantizer parameter (i.e., quantization step size) is used for all macroblocks and pictures, the resultant bitstream is called a Variable Bit Rate (VBR) bitstream. This is, of course, not the only way to produce a variable rate bitstream, but the term VBR has been used almost exclusively to denote this coding approach. Use of the same quantizer across the whole video sequence results in identical quality (at least in terms of theoretical SNR) for all pictures, and hence VBR encoding is also known as constant quality or constant Q encoding (where Q refers to the quantizer step size, which essentially defines quality).

4.2.2 Constant Bit Rate Video and Rate Control

In traditional communication applications (i.e., those based on circuit-switching), the notion of a channel involves a constant bandwidth (capacity) and delay, together with some statistical characterization of the unavoidable noise. For this reason, significant efforts have been expended over the years to convert the variable length bitstream produced by transform coders into a constant bit rate one.

The classical model of constant bit rate conversion consists of attaching a buffer of size B at the output of the encoder. The buffer is continuously emptied at the constant channel rate. The buffer's occupancy is monitored, and the encoder takes appropriate actions to avoid overflow or underflow. Overflow would obviously lead to loss of information, while underflow is undesirable when the coded bitstream is used to generate transmission timing information at the receiver. Let $B(t)$ denote the buffer's occupancy just prior to coding picture unit t (a unit may be a block, macroblock, slice, or even a whole picture), $r(t)$ the number of bits required to encode picture unit t, and r_C the channel rate expressed in terms of the average bits per picture (i.e., r_C is the channel rate multiplied by the picture period). The buffer's occupancy evolution is then governed by the recursive equation

$$B(t+1) = B(t) + r(t) - r_C \qquad B(0) = 0 \tag{4.1}$$

whereas the encoder must ensure that, for all t,

$$0 \leq B(t) \leq B \tag{4.2}$$

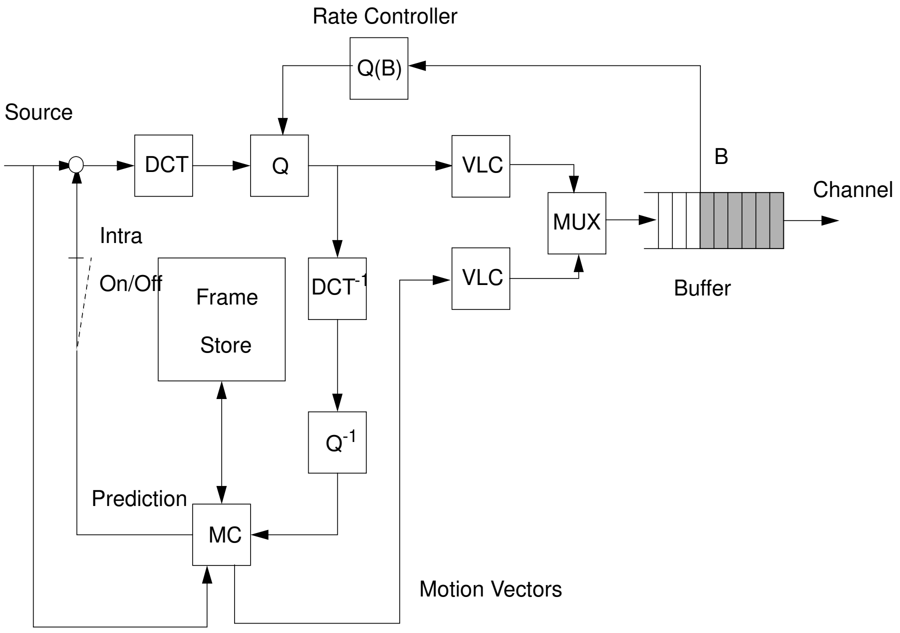

Figure 4-2: Constant bit rate motion-compensated transform coder with rate control.

The process with which the buffer's occupancy is controlled is called rate control. The rate controller modifies the encoder bitstream rate by adapting the quantization process based on the buffer occupancy, as shown in Figure 4-2. In its simplest form, the rate controller is simply a linear relationship between the quantizer step size and the buffer occupancy. Assuming that the quantizer scale Q (which is proportional to the step size) can take values in the set $\{1, 2, \ldots, Q_{\max}\}$ (typically $Q_{\max} = 31$), then the rate controller uses the equation:

$$Q(t) = \left\lfloor \frac{B(t)}{B} Q_{\max} \right\rfloor + 1 \tag{4.3}$$

to set the value of Q for unit t. When $B(t) = B$, Q(t) is simply set to $Q_{\max}$. Eq. (4.3) establishes a closed loop between the buffer and the encoder, and by virtue of (4.1) ensures that the average number of bits used per unit is r_C. When the buffer occupancy becomes high, coarser quantization is utilized so that fewer bits are required to code each unit, and vice versa. Note that the above scheme does not guarantee that overflow or underflow will never occur. If the statistics of the source, however, do not vary very quickly with time, the probability of that happening becomes very low.

Much more complicated rate approaches are also possible, that take also into account the source signal [104, 58]. There are also rate control schemes that take into account semantic information (e.g., the location of faces or facial features), like the model-assisted rate control algorithms we have proposed in [47, 42, 43, 44, 64]. These techniques achieve a better distribution of the bit budget to critical segments of a video sequence, but all tend to be heuristic. More rigorous analyses of the problem have also been developed (see, for example, [96, 82, 126] and references therein).

Note that modern systems deviate significantly from the above model. While it is still applicable to purely hardware-based solutions, such as an encoder that is directly interfaced to a channel, computer-based encoders and networks have totally different paradigms of communication. For the case of the host, the fundamental unit of transmission is no longer the bit but is rather a memory buffer (e.g., a page, for systems using virtual memory). Furthermore, memory space in general purpose computers is less likely to be tightly bounded. Similarly, computer networks utilize packet-based transmission, and their bandwidth and delay behavior is much more complicated than what the above model suggests.

4.2.3 Video Rate and Communications Systems

As we have discussed in Chapters 1 and 2, the quality of service (QoS) is one of the most important parameters of a communication system. Traditional telecommunication facilities based on circuit-switching technology provide constant bandwidth bit pipes with fixed delay and well-defined error behavior. Data communication facilities, on the other hand, typically provide no QoS guarantees, and only operate on a best-effort basis. They include individual LANs as well as the Internet as a whole. Modern network architectures, and more specifically ATM-based networks, attempt to bridge the gap between the two by combining features from both: connection-oriented operation, packet-based transmission, and small (fixed) packet length for low delay in real-time communication applications [56].

4.2.3.1 Networks with Quality of Service Guarantees

A key feature of the integrated networking approach in ATM (inherited from data networking) is the concept of statistical multiplexing. The key principle in statistical multiplexing is that, if we multiplex a large number of variable rate streams, the bandwidth required for their transmission will be less than the sum of the bandwidths required for each one individually. This is because rate peaks in some stream will most likely coincide with valleys in others, and hence the variation will tend to cancel out. In fact, as one would expect, the aggregate bandwidth required will simply be the sum of the average rates of the individual streams.

When a user requests a connection from an ATM network, he or she establishes an agreement, or *traffic contract*, with the network with respect to the traffic that will be generated and transported. The traffic contract is based on a predefined traffic characterization vector, which may include parameters such as peak and average cell (packet) rates, maximum burst size, cell delay variation, etc. [56]. The network management layer will decide to accept the connection if the projected traffic load can be accommodated by the network without compromising the traffic contracts of other users; otherwise, the request will be declined. This process is called *admission control*.

In order to protect itself from abuse, the network monitors the user's traffic at the point it enters the network, and ensures that the agreed-upon traffic bounds are not exceeded. If a user exceeds the parameters of the traffic contract, then the system will discard the cells corresponding to the excess traffic. Alternatively, the network may simply mark them as excessive and if, during the course of the transmission, a switch becomes congested, it will discard them at that time. This process is called *policing*.

The above model assumes that the user's traffic can be adequately modeled by a standardized, universal traffic characterization vector. The importance of the traffic's behavior in designing admission control algorithms has prompted a significant amount of research in the area of video traffic modeling and multiplexing. Very sophisticated models have been developed, including Markov-Modulated Poisson Processes (MMPP), Transform Expand Sample (TES), etc. (see for example [78, 61, 79, 110, 85] and references therein). Despite their complexity, these models have not been proven to be universally applicable.

The difficulty of the modeling problem can be traced back to the structure of the encoder. The rate variability is due to the use of Huffman encoding, which employs variable length codes. These are designed so that highly likely events are encoded with shorter codewords, while less likely ones with longer codewords. Depending on the source material, the number of bits needed to encode a picture may exhibit large variations: pictures with uniform appearance and few details will in general require fewer bits; complex sequences with very high-speed motion, on the other hand, will require much more. In essence, the richness of visual content itself prohibits the development of a simple, yet accurate, traffic model.

The consequences of inadequate traffic models can take two different forms. For optimistic models (i.e., those that underestimate the source traffic), we will have information loss since excess traffic will be filtered out by the policing function. For pessimistic models (i.e., those that overestimate the source traffic), we are wasting network resources. The example of peak rate allocation (i.e., allocation of a channel based on the maximum rate of a variable rate video bitstream) belongs to the latter category. Obviously, such an approach defeats the whole purpose of using an integrated network architecture, and falls back to the traditional circuit-switching system model.

4.2.3.2 Networks without Quality of Service Guarantees

The category of networks without QoS guarantees includes traditional data networks (e.g., Ethernet LANs), as well as virtual circuits in which no service guarantees have been established [93]. Due to the higher cost of establishing high-quality connections, and also because this type of virtual connection will be popularized by distributed computing applications (e.g., Internet connections), it is expected that they will represent a significant proportion of the total number of connections.

In this case the traffic that can be handled by the network can be arbitrary, and can also vary significantly over time. Clearly, any a priori assumption that can be made by an encoder's rate controller are totally useless. Unless the network happens to be in a state in which the offered traffic can be accommodated, high-quality communication is totally impossible. Potential adverse phenomena include severe end-to-end delays, very high jitter (delay variation), and very high loss rates.

In order to accommodate audio-visual applications in such an environment, applications must exhibit "elasticity." In other words, they must be able to adapt to the changing network conditions in order to keep the quality of service as high as possible. We have used such an approach in Chapter 2, in the context of the Xphone system's JPEG coding adaptive rate control algorithm. A similar approach has been followed in [75] in the context of MPEG. Both ideas utilize network feedback in order to control the encoding process[1]. Observe the similarity with traditional rate control, in which the feedback was produced by a channel buffer.

Although these techniques have merit, they have the significant problem of not being able to handle precompressed material. Since they require direct communication of the network state to the encoder, they are only applicable to live video applications. Another drawback is that they require specialized encoders (i.e., systems that are specifically designed to accept and utilize network feedback), and hence are more expensive.

Current applications (e.g., `nv`, `vat`, `vic`, etc.) that are routinely used in the MBONE (the multicast-enabled part of the Internet) follow a very simple approach

[1] For the JPEG case, and due to the very challenging environment used in Xphone, we have also seen that in many cases it is necessary to drop frames altogether in order to maintain an adequate communication environment.

towards quality of service. Since they are designed for dissemination-oriented communication (e.g., broadcast of seminars) and not highly interactive sessions, they employ a best-effort approach. A target bit rate is fixed (in the order of 128 Kbps) depending on the "scope" of the session (local, regional, national, or international), and transmission is done with no regard to the reception quality. As one might expect, the quality is quite poor. Although this open-loop and best-effort approach is reasonable when the number of receivers may potentially be in the thousands (and given the current Internet's capabilities), person-to-person or small group communication has much stricter performance requirements. Efforts are underway within the Internet Engineering Task Force (IETF) to provide improved quality of service in IP (and IPng) using reservations (RSVP) [132]; this will at least provide good delay bounds for real-time traffic.

4.3 Dynamic Rate Shaping

From the preceding discussions, we observe that we essentially have two different viewpoints in terms of the video rate produced or transported. On the one hand we have video encoders that live in the dual worlds of traditional CBR and highly unpredictable VBR compressed video. On the other hand we have networks that can carry either CBR traffic, or statistically characterized VBR traffic, or only provide best-effort capabilities. The problem of carrying CBR video is well understood. As we pointed out earlier, though, there are cases when the video and channel rates can be incompatible. In the most interesting case of VBR video, the principles assumed by these two viewpoints on how the traffic behaves are very different, and will almost always result in very poor performance.

The objective of dynamic rate shaping is then to bridge the wide gap between CBR and VBR video with a continuum of alternatives and hence, using hopefully simple and efficient algorithms, to allow complete interoperability of the video signal with any kind of transport mechanism. By separating the dependency of the encoder and the transport mechanism we allow their designers to independently use the best possible designs. For example, the network designer will select a traffic characterization vector for which admission control becomes tractable and efficient to implement throughout a network. Similarly, the encoder designer will use the rate behavior that best represents the source material. If necessary, a dynamic rate shaping processor can then ensure interoperation between the two, if necessary. Note that by optimizing the two components independently it is almost certain that we will not achieve a global optimum. The search for such an optimum, however, is—in all likelihood—a futile exercise due to the large number of parameters involved. At present no general "optima" are known, even for the individual problems.

We thus define rate shaping as an operation which, given an input video bitstream and a set of rate constraints, produces a video bitstream that complies with these constraints. For our purposes, both bitstreams are assumed to meet the same syntax specification, and we also assume that a motion-compensated transform coding scheme is used. This includes both MPEG-1 and MPEG-2, as well as H.261 and JPEG. If the rate constraints are allowed to vary with time, the operation will be called *dynamic rate shaping*. Throughout this discussion we assume that MPEG-2 is used as the video coding syntax. An overview of the general transform coding approach, including MPEG-specific features, was provided in Section 3.2.

4.3.1 The Formulation of the Dynamic Rate Shaping Problem

The rate shaping operation is depicted in Figure 4-3. We are given an input bitstream y with some rate characteristic $B(t)$ and a rate constraint $B_T(t)$, and we are asked to produce a bitstream $\hat{y}$ whose rate $\hat{B}(t)$ complies with this constraint.

Note that no communication path exists between the rate shaper and the source of the input bitstream, which ensures that, by design, no access to the encoder is necessary.

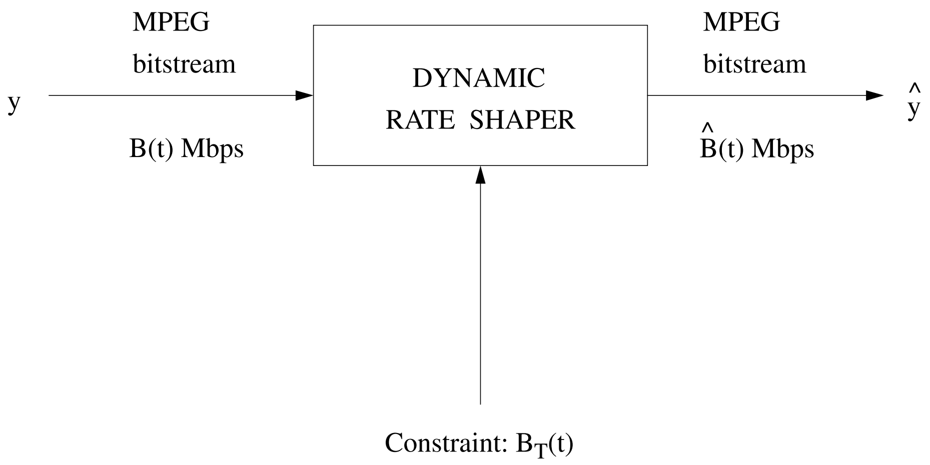

Figure 4-3: Definition of Dynamic Rate Shaping.

Of particular interest is the source of the rate constraints $B_T(t)$. In the simplest of cases, $B_T(t)$ may be just a constant and known a priori, e.g., the bandwidth of a circuit-switched connection. This is the case when we are converting a high-rate bitstream for transmission over a low-capacity channel. It is also possible that $B_T(t)$ has an a priori well-known statistical characterization. This would be the case, for example, in ATM networks, where the statistical characterization is governed by the traffic contract and the policing function.

Finally, another alternative is that $B_T(t)$ is generated by the network over which the output bitstream is transmitted. This information could be potentially provided by the network management layer, or may be the result of end-to-end bandwidth availability estimates. This would be used, for example, in best-effort networks. Such an approach has been followed in the Xphone adaptive rate control scheme (Section 2.3), and also in [75].

The objective of an optimal rate shaping algorithm is to minimize the conversion distortion, i.e.,

$$\min_{\hat{B}(t) \leq B_T(t)} \{\|y - \hat{y}\|\} \tag{4.4}$$

The metric $\|\cdot\|$ denotes the squared error criterion. Note that no assumption is made about the rate properties of the input bitstream, which can indeed by arbitrary. As we will see, the attainable rate variation $(\hat{B}/B)$ is in practice limited, and depends primarily on the number of B pictures of the bitstream and the original rate $B(t)$.

In addition, there is no indication in eq. (4.4) of either the optimization window, or the optimization parameters (i.e., those for which we are called to find the optimal values). The nature of the optimization parameters gives rise to structurally different problems, which lead to slightly different types of algorithms. The various alternatives are discussed in the next section.

The optimization window, as in the case of data partitioning in Chapter 3, is considered a design parameter. There are several issues affecting its choice. A very important one is the computational complexity that we are prepared to accept; longer windows will require significantly more processing time and buffering space than shorter ones. Another key factor is the dynamics of the transport mechanism used, if any. If the constraints provided to the rate shaper change very quickly compared to the picture period (33 or 16.5 msec), a short optimization window should be used in order to capture these variations. For more slowly varying networks, the performance benefits of larger windows can be exploited. In the simulation results presented in this chapter, all optimization windows are set to cover a complete picture.

Finally, we should point out that, as in the data partitioning case, y and $\hat{y}$ may be generated by a recursive process. Rate shaping will introduce error accumulation phenomena, which will have to be taken into account.

4.3.2 The Family of Dynamic Rate Shaping Algorithms

Assuming that a motion-compensated transform coding technique is used to generate the input bitstream and decode the output one, there are two fundamental ways to reduce the rate:

1. modifying the quantized transform coefficients by employing coarser quantization, and

2. eliminating transform coefficients.

In general, both schemes can be used to perform rate shaping. Requantization leads to recoding-type algorithms, for which it has been shown (at least for the buffered-constraint case) that optimal solutions exist, but are very complex. In addition, as we will see later on, they do not necessarily perform as well. Our focus here is in "selective transmission" algorithms (i.e., those that only select transform coefficients for elimination from the bitstream). This concept is directly linked, as we will see, with the data partitioning problem of Chapter 3, traditional zonal sampling [67], as well as the optimal thresholding of JPEG-compressed images analyzed in [73, 74], and the mathematical treatment is very similar.

For purposes of comparison between the two approaches, we will use recoding as a representative algorithm of requantization. This serves a dual purpose, since recoding is also the brute-force approach in performing rate shaping (although too complex and expensive to be of practical value). We should note that a requantization approach has recently been proposed (independently of our work) for rate conversion of H.261 video in [89, 128].

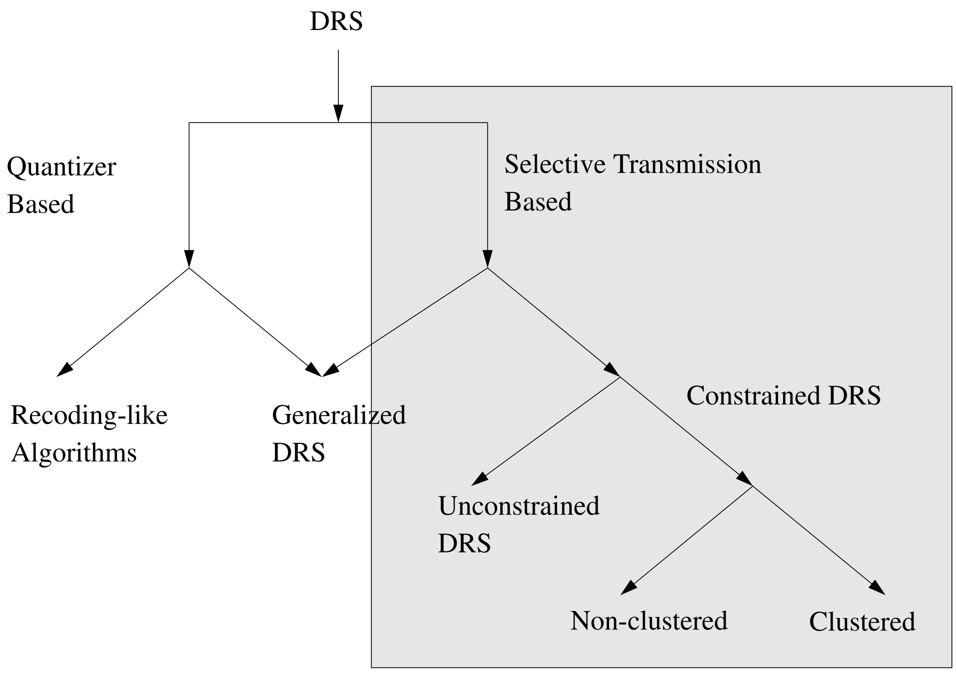

Figure 4-4: The family of Dynamic Rate Shaping algorithms.

These two broad categories of algorithms give rise to a number of special cases. The complete family is shown in Figure 4-4. Algorithms that combine requantization and selective transmission will be referred to as *generalized* rate shaping algorithms, and will not be discussed here. The selective transmission category (indicated by the gray area) is further segmented into constrained and unconstrained algorithms. The former is subdivided into clustered and non-clustered. We first discuss the constrained DRS algorithm first, as it is the simplest.

4.4 Constrained Dynamic Rate Shaping

In the Constrained Dynamic Rate Shaping (CDRS) algorithm, we add a structural constraint to the optimization problem of eq. (4.4): rate reduction will occur by eliminating a contiguous string of DCT coefficients at the end of each block. In that sense, CDRS performs a truncation in the transform domain. The number of DCT run-length codes to be kept in each block will be called the *breakpoint*, as shown in Figure 4-5. The process is very similar to data partitioning, with the difference that now the breakpoint value is different for each block. Assuming that the MPEG scheme is used, we require that at least one DCT coefficient will remain in each block. This is necessary in order to avoid certain syntax complications, which include recoding the coded block patterns (which indicate which blocks in each macroblock are included in the bitstream), and reexecuting the DC prediction loops. As a result, breakpoint values will range from 1 to 64.

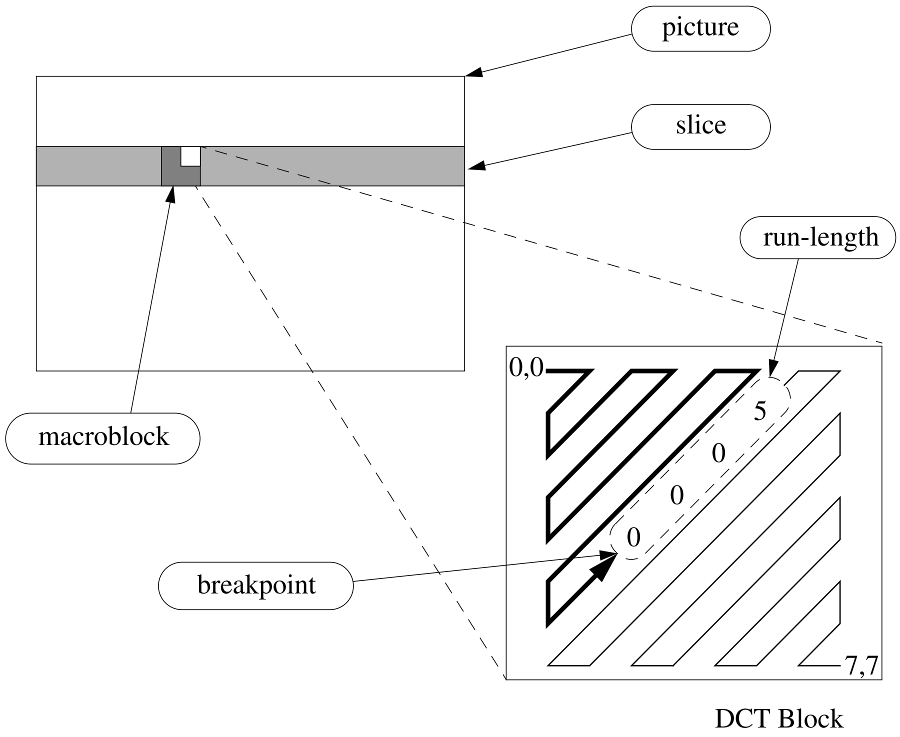

Figure 4-5: Breakpoint position in the zig-zag pattern of DCT coefficients.

4.4.1 Problem Formulation

4.4.1.1 The Intra-Only Case

We can formulate the problem of Constrained DRS by borrowing the line of analysis used in Sections 3.4.1 and 3.5.1. In particular, we first express the distortion for a single block, and then proceed to consider the total rate shaping distortion. We consider one-dimensional quantities for simplicity. Let

$$\mathbf{x} = [x(0), x(1), \ldots, x(N)]^T \tag{4.5}$$

denote the pixel values of a (one-dimensional) block, and

$$\mathbf{X} = [X(0), X(1), \ldots, X(N)]^T \tag{4.6}$$

its DCT coefficients. Let also $\hat{\mathbf{x}}$ and $\hat{\mathbf{X}}$ denote the corresponding quantities of the truncated representation; i.e., for a breakpoint value of $b > 0$ we have:

$$\hat{X}(k) = \begin{cases} X(k) & \text{if } 0 \leq k < b \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

and

$$\mathbf{x} = C^T \mathbf{X} \tag{4.8}$$

where C is the N-point DCT matrix. Using the unitarity of the DCT transform matrix, we have shown in (3.29) that the truncation error can be expressed as:

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| = \sum_{i=b}^{N-1} X(i)^2 \tag{4.9}$$

In other words, the distortion is simply the energy of the coefficients that are removed.

Due to the independence of the blocks of a picture (and considering just the intra case at this time), the total shaping distortion will simply be the sum of the individual distortions for each block. A matter of importance is that, in contrast with data partitioning, we are now free to select a different breakpoint value per block. This includes not only luminance blocks, but chrominance ones as well. Consequently, the total distortion will have to have contributions from all three color components. Such a combined metric is very difficult to derive in a meaningful sense, since it depends heavily on the way color is perceived by the human visual system, as well as the particular color space used [91].

If we consider for example a difference Δ in the luminance component only, and the same difference in a chrominance component, a metric which would treat all three components identically would give the same result. The perceptual effect, however, of these two distortions would be radically different, particularly for large Δ's. One could possibly use a weighted average of the distortions in the three components, but again the selection of appropriate weights would be arbitrary.

To avoid this complication, we only define distortion for the luminance component. In order to be able to accommodate all different chroma formats, the selection of a breakpoint will be performed at the macroblock level. This represents the "least common denominator" in terms of matching the spatial area of luminance and chrominance blocks.

We can now express the total distortion, again for the intra-only case, as follows. Let $D_i(b_i)$ denote the distortion for macroblock i when the breakpoint value b_i is used, and let $X_j^i(k)$, $0 \leq k \leq 63$ denote the value of the DCT coefficient of the k-th run-length of the j-th block of the i-th macroblock. We then have:

$$D_i(b_i) = \sum_{j \in \mathcal{Y}} \sum_{k \geq b_i} X_j^i(k)^2 \tag{4.10}$$

where $\mathcal{Y}$ denotes the (four) luminance blocks of a macroblock. Consequently, the minimization problem of eq. (4.4) can be formulated as follows

$$\min_{\hat{B}(t) \leq B_T(t)} \{\|y - \hat{y}\|\} \Longleftrightarrow \min_{\sum_{i=1}^{M} R_i(b_i) \leq B_T(t)} \left\{ \sum_{i=1}^{M} D_i(b_i) \right\} \tag{4.11}$$

where M denotes the number of macroblocks in the optimization window, and $R_i(b_i)$ denotes the rate required to encode *all three* color components of macroblock i using the breakpoint value b_i, i.e.,

$$R_i(b_i) = \sum_{m \in \mathcal{Y} \cup \mathcal{U} \cup \mathcal{V}} R_i^m(b_i) \tag{4.12}$$

where $R_i^m(b_i)$ denotes the rate for block m, and $\mathcal{U}$ and $\mathcal{V}$ are the sets of block numbers of the chrominance components U and V of a macroblock.

4.4.1.2 The Mixed Mode (I, P, B) Case

In order to derive a similar expression to (4.10) for the mixed-mode coding case, we need to take into account the recursive nature of the decoding operation. As we noted in Section 3.5.1, when we make rate shaping decisions for a given picture we not only have to take into account the error accumulated from past pictures, but also the error that will be propagated to future pictures. This requires that a complete group of pictures, from I to I, is considered jointly in the optimization process. Complexity and delay considerations of such an attempt totally rule out this possibility.

As in data partitioning, a causality argument will be invoked in order to constrain the optimization window and avoid a "look-ahead" optimization algorithm. Using this argument, when rate shaping decisions are taken for a given picture we only consider the error accumulated from shaping decisions from *past* pictures; the error that will be propagated to future pictures is ignored. We should note that this error component will be revisited when we will process a following picture, since it will appear in the accumulated error from past pictures. Also observe that the causally optimal problem is unconditionally optimal for the intra-only case.

We start from the expressions for the decoding process with, and without rate shaping, i.e.,

$$y_i = \mathcal{M}_i(y_{i-1}) + e_i, \qquad e_0 = y_0 \tag{4.13}$$

and

$$\hat{y}_i = \mathcal{M}_i(\hat{y}_{i-1}) + \hat{e}_i, \qquad \hat{e}_0 = \hat{y}_0 \tag{4.14}$$

where $\mathcal{M}(\cdot)$ is the motion compensation operator, and e_i is the coded prediction error. By defining the accumulated error a_i as

$$a_i \stackrel{\text{def}}{=} \mathcal{M}_i(y_{i-1}) - \mathcal{M}_i(\hat{y}_{i-1}) \tag{4.15}$$

the minimization problem of eq. (4.4) can be written as:

$$\min_{\hat{B}(t) \leq B_T(t)} \{\|a_i + e_i - \hat{e}_i\|\} \tag{4.16}$$

Finally, using a proof similar to the one of Section 3.5.2, we can express the shaping distortion $D_i(b_i)$ for macroblock i when breakpoint value b_i is used as:

$$\hat{D}_i(b_i) = \sum_{j \in \mathcal{Y}} \left\{ \sum_{k=0}^{N-1} A_j^i(k)^2 + \sum_{k \geq b_i} 2A_j^i(\mathcal{I}_j^i(k))E_j^i(k) + \sum_{k \geq b_i} E_j^i(k)^2 \right\} \tag{4.17}$$

where N is the block size, $\mathcal{Y}$ denotes the luminance blocks of a macroblock, $A^i_j(k)$ is the k-th DCT coefficient (in zig-zag scanning order) of the j-th block of the i-th macroblock of the accumulated error, and $E^i_j(k)$ is the DCT coefficient of the k-th *run-length* of the j-th block of the i-th macroblock. Note that a mapping function $\mathcal{I}^i_j(k)$ is also used to map run-length positions in the prediction error to zig-zag scan positions in the accumulated error.

The causally optimal problem can now be expressed from (4.16) as:

$$\min_{\sum_{i=1}^{M} R_i(b_i) \leq B_T(t)} \left\{ \sum_{i=1}^{M} \hat{D}_i(b_i) \right\} \tag{4.18}$$

Eq. (4.18) generalizes (4.11) by considering the more general definition of distortion $\hat{D}_i(b_i)$ given by (4.17). This is because, for the intra-only coding case, we have $D_i(b_i) = \hat{D}_i(b_i)$. As a result, the same algorithm that solves (4.18) also solves (4.11). Such algorithms are presented below.

4.4.2 Causally Optimal, Memoryless, and Rate-Based Algorithms

For the solution of the constrained minimization problem expressed by eq. (4.18) we use the Lagrange multiplier approach employed for the data partitioning problem in Section 3.4.2. In particular, by introducing the Lagrange multiplier λ, we consider the unconstrained minimization problem

$$\min \left\{ \sum_{i=1}^{M} \hat{D}_i(b_i) + \lambda \sum_{i=1}^{M} R_i(b_i) \right\} \tag{4.19}$$

As we have discussed in Section 3.4.2 (Theorem 1), for some particular value of λ which our algorithm will have to find (if it exists), the solutions to (4.18) and (4.19) become identical. By defining

$$L_i(\lambda, b_i) \stackrel{\text{def}}{=} \hat{D}_i(b_i) + \lambda R_i(b_i) \tag{4.20}$$

Eq. (4.19) can be written as

$$\min \left\{ \sum_{i=1}^{M} L_i(\lambda, b_i) \right\} \tag{4.21}$$

We note that

$$L_i(\lambda, b_i) \geq 0 \tag{4.22}$$

since all quantities involved are non-negative.

From the representation in (4.21) we observe that, for a given λ, minimization with respect to the breakpoint value b_i can be performed independently for each macroblock. This reduces the complexity of the problem to the order of $64M$, compared with a brute-force full-search method that would require examination of 64^M possibilities.

In order to find the optimum value λ^* for λ, and hence the optimal breakpoints b_i^*, the same iterative bisection algorithm used in the data partitioning problem can be employed. The detailed description can be found in Section 3.4.2 (page 92); we should note that $\hat{D}$ should be substituted in place of D.

Again, an issue is the target bit budget R_{budget} that is to be set for each optimization window. In our simulations we used a constant target rate, and hence the proportional bit allocation used in (3.46) can still be used:

$$R_{\text{budget}} = (\hat{B}/B)R - R_o \tag{4.23}$$

where R_o denotes overhead bits (for headers, motion vectors, etc.). In case the target bit budget is not complete utilized (which happens almost always), leftover bits from one iteration of the algorithm are carried over to the next. The problem of selecting an appropriate bit budget in the general case of dynamically varying constraints is a quite complex one, and also depends on the specifics of the implementation. In this chapter we do not address this issue, focusing rather on the signal processing problem of optimally performing the required rate conversion. This would be an important issue for any implementation, however, and warrants further detailed study.

In order to simplify the algorithm, we can also utilize a memoryless algorithm. Here we ignore the accumulated error, and treat each picture as an intra one. In other words, the definition of distortion is the one used for D instead of $\hat{D}$. The benefits of such an approach are discussed in the next section.

A very simple alternative approach that can serve as a lower bound in terms of quality is a purely rate-based approach. Here we do not take into account the distortion at all (either current or accumulated), and focus only on meeting the prescribed rate constraints. The selection of the optimal breakpoint b_i^* for each macroblock is selected depending on:

$$b_i^* = \max\left\{b_i : R_i(b_i) \leq R_{\text{budget}_i}\right\} \tag{4.24}$$

where the bit budget R_{budget_i} is set proportional to the original size R_i of the macroblock:

$$R_{\text{budget}_i} = \frac{R_i}{R_{\text{budget}}/S} \tag{4.25}$$

Note that the same optimization window as in the optimal and memoryless algorithms has to be used, in order to compute R_{budget_i}, hence ensuring fairness in the comparison of the various approaches. As before, if the target bit budget is not met exactly for a particular macroblock, leftover bits from one macroblock are carried over to the next.

4.4.3 Performance Evaluation

As in the case of data partitioning, the only data needed to execute the optimal algorithm for the intra-case are the values of the DCT coefficients. Hence an implementation needs only to be able to parse the input bitstream up to the point of inverse quantization of DCT coefficients. This represents a small percentage of the complexity of a complete decoder, since the motion compensation and inverse DCT transformation modules are not needed. Furthermore, the output bit generation module is very simple, since the only modification in the bitstream is the removal of bits corresponding to run-lengths that are eliminated.

In the mixed-mode case, the algorithm becomes significantly more complicated due to the need to track the accumulated error a_i. This involves a dual motion compensation loop (as was shown in Section 3.5.1), as well as inverse and forward DCT transform modules. The latter is necessary since motion compensation is performed in the spatial domain. As a result, the complexity of the causally optimal algorithm is between that of an encoder and a decoder.

Of concern in terms of the performance of the algorithm is the fact that the iterative bisection operates in the convex hull of the $R(D)$ curves. As we saw in the case of data partitioning, points which lie above are not $R(D)$ optimal, and hence are not considered by the algorithm.

Figure 4-6 depicts the $R(D)$ curves for an intra macroblock from "Flower Garden", coded at 24 and 12 Mbps. This macroblock was purposely selected to show that a quite erratic behavior can sometimes be present. For the 24 Mbps case, for example, the convex hull is defined by just 6 points; all the rest are ignored. Comparing with Figure 3-9 (page 93), where the slice-based $R(D)$ curve for the data partitioning problem is depicted, we see that the latter is much more well-behaved due to the averaging across the full slice. When we consider the mixed-mode coding case, the $R(D)$ curves are much more well-behaved. An example is shown in Figure 4-7.

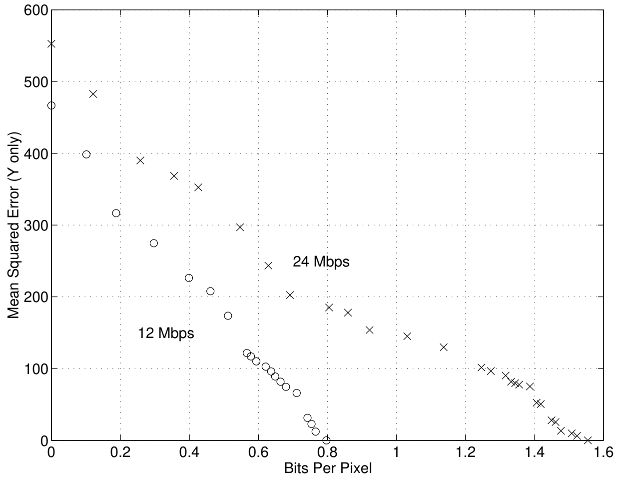

Figure 4-6: $R(D)$ curves for an intra macroblock from "Flower Garden", coded at 24 Mbps (x) and 12 Mbps (o).

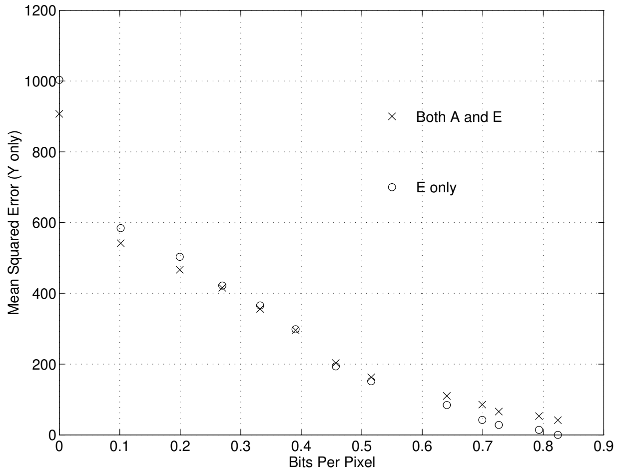

Figure 4-7: $R(D)$ curves for the total (A and E) and current picture only (E) shaping distortion for a macroblock from a P picture of "Flower Garden", coded at 4 Mbps and rate shaped at 3.2 Mbps.

Figure 4-8 shows the results of various rate shaping approaches on the "Flower Garden" sequence, intra coded at 24 Mbps and rate shaped at 12 Mbps. The optimization window is a full picture, and typically requires 10–12 iterations of the bisection algorithm to obtain the optimal breakpoint configuration. We observe that the recoding approach outperforms the optimal rate shaping algorithm by about 4 dB on the average, which in turn outperforms the rate-based approach by about 2 dB. We should note that this behavior is not typical, and depends on the selection of the original compression rate, as well as the rate shaping ratio (here 50%).

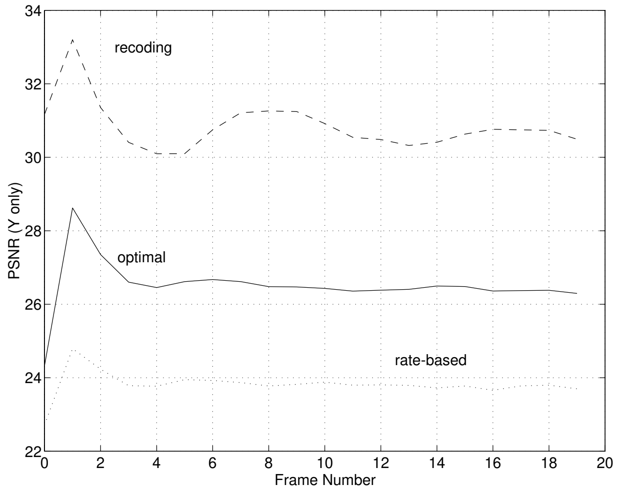

Figure 4-8: Rate shaping of "Flower Garden" using the optimal, rate-based, and recoding approaches. The source is coded at 24 Mbps and rate shaped at 12 Mbps.

Figure 4-9 shows the results of the various mixed-mode rate shaping algorithms on the "Mobile" sequence, coded at 4 Mbps and rate shaped at 3.2 Mbps. The optimization window is again a complete picture. Our first observation is that the causally optimal algorithm outperforms recoding. In fact, as we can see in the figure, the recoding algorithm has a rapidly declining performance. This is because of the initial state assumed by the encoder, which is the same one used for the original compression at 4 Mbps. We purposely selected this situation to depict the effects of using a general-purpose encoder that cannot be easily adapted to changing conditions. After this initial quality drop, the encoder will eventually converge to a quality around 29–30 dB. Hence the optimal algorithm turns out to be about 1 dB better than recoding. Depending on the "intelligence" of the encoder, this difference can be reduced even further.

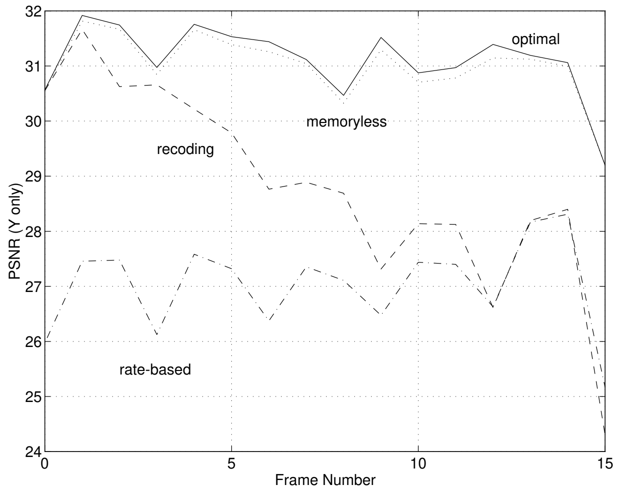

Figure 4-9: PSNR performance for rate shaping of "Mobile" using the (causally) optimal, memoryless, rate-based, and recoding approaches. The source is originally coded at 4 Mbps and rate shaped at 3.2 Mbps (I period 12, P period 3).

We also observe that the rate-based approach performs very poorly, about 5 dB worse than the causally optimal one. Most important, however, is the behavior of the memoryless algorithm. As we can see from the figure, its performance is almost indistinguishable from the optimal algorithm, with a difference of about 0.2 dB. This level of differentiation is extremely hard, if not impossible, for a human observer to detect. This reinforces the results of Chapter 3, where a similar behavior was identified, and can be explained using identical reasoning.

Figure 4-10 shows rate shaping results (average PSNR) for "Mobile" across a spectrum of different rates. We see that the causally optimal algorithm outperforms recoding up to about 2.8 Mbps, after which the curves cross each other. This result indicates that, if the rate reduction is not severe, selective transmission performs better than requantization approaches. In other words, the penalty of dropping high-frequency DCT coefficients is not severe. When the reduction becomes too high, then coefficients very close to the DC have to be eliminated, hence rapidly deteriorating quality. In this case it is better to requantize, so that a different bit distribution is created among the various DCT coefficients. This result corroborates the similar behavior that has been observed in optimal thresholding of JPEG images [73, 74].

We should note that the exact position of the crossover point depends on the initial coding rate, as well as the sophistication of the encoder (or requantizer). Better designs will push the intersection further to the right, thus narrowing the window in which selective transmission performs better.

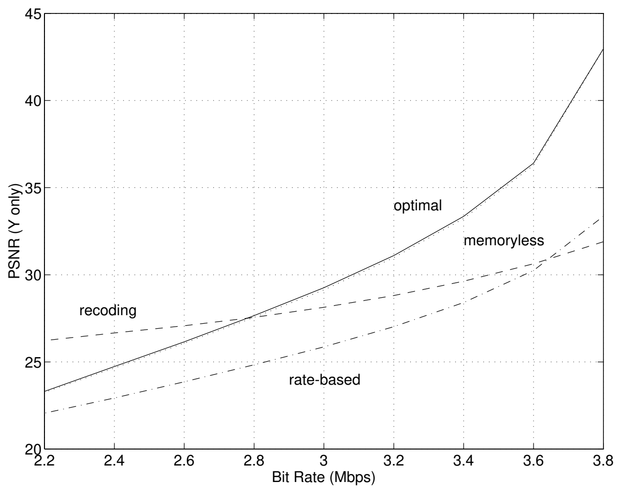

Figure 4-10: Average PSNR performance for rate shaping of "Mobile" using the (causally) optimal, memoryless, rate-based, and recoding approaches, for various target rates. The source is originally coded at 4 Mbps (I period 12, P period 3).

We also see from the figure that the rate-based approach performs consistently poor. As in the isolated example of Figure 4-8, however, we see that the memoryless algorithm performs almost identically to the causally optimal one across the whole range of the various target rates (the two curves can hardly be distinguished).

This confirms that the near-optimality of the memoryless algorithm is a universal property.

The importance of this result is significant. As we stated in the introduction to this chapter, the viability of the concept of dynamic rate shaping relies heavily on the capability of implementing it with simple algorithms. This result indicates that this is indeed possible. Given that the complexity of the memoryless algorithm is less than that of an encoder, and that real-time MPEG decoding is today possible for low-resolution video, even purely software-based implementations of DRS should soon be possible on general purpose microprocessors.

4.4.4 Clustering

Since a full resolution standard TV picture (704×480) may contain up to 3,960 macroblocks (for a 4:4:4 chroma format), the processing required within each iteration in order to find the breakpoint value that minimizes $D_i(b_i) + \lambda R_i(b_i)$ can be significant. Consequently, even for the memoryless algorithm, it is worth examining approaches with which this complexity can be further reduced.

An attractive possibility is to use a *clustering* approach, in which a common breakpoint value is selected for a set of macroblocks. The concept is depicted in Figure 4-11. We refer to such algorithms as $C(n)$, where n is the number of sequential macroblocks contained in each cluster. The causally optimal, memoryless, and rate-based algorithms discussed in the previous section all fall under the $C(1)$ category, i.e., they obtain one breakpoint value per macroblock. We implicitly used a clustering argument in Section 4.4.1.1, when we defined a common breakpoint value for all blocks of a macroblock in order to avoid an ill-defined distortion metric.

Several different algorithms can be obtained for the clustering approach by appropriate modification of the variables used in the causally optimal, memoryless,

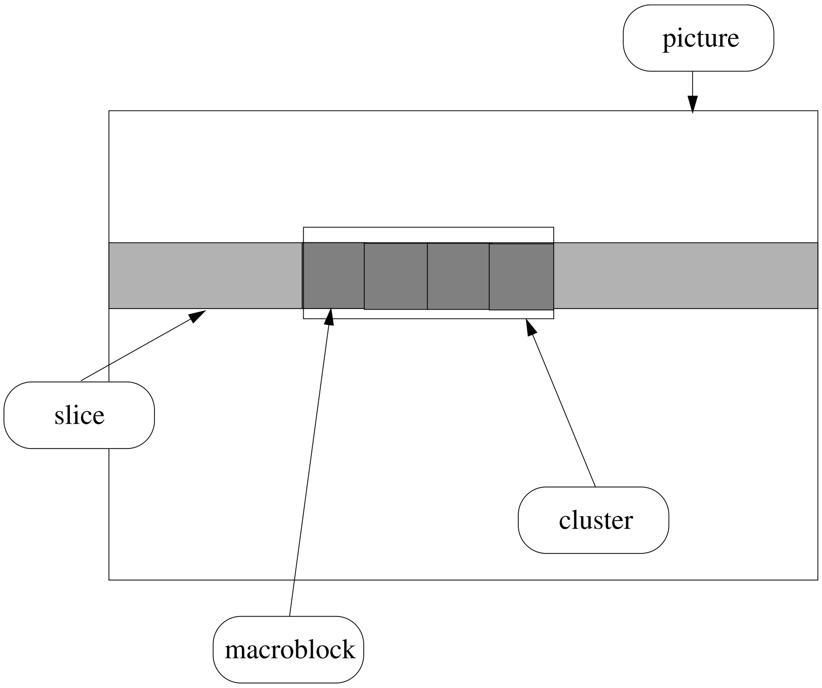

Figure 4-11: Clustering approach for the reduction of algorithmic complexity.

and rate-based algorithms. In particular, define by $\mathcal{C}_k$ the set of macroblock belonging to the k-th cluster. We can define the cluster distortion $D_i^{\mathcal{C}}(b_i)$ (either with, or without the accumulated error), as:

$$D_i^{\mathcal{C}}(b_i) = \sum_{k \in C_i} D_k(b_i) \tag{4.26}$$

where $D_k(b_i)$ denotes the distortion for an individual macroblock. Similarly, we can define the cluster rate as:

$$R_i^{\mathcal{C}}(b_i) = \sum_{k \in C_i} R_k(b_i) \tag{4.27}$$

where $R_k(b_i)$ denotes the rate of an individual macroblock. Using these definitions for the rate and distortion, all previous algorithm can be applied directly.

Clearly, when $\mathcal{C}_k$ spans a complete slice, the problem becomes mathematically equivalent with that of data partitioning. In the latter case, instead of eliminating the truncated DCT coefficients, we create a secondary bitstream (partition 1) in which they are transmitted to the receiver. The mechanisms, however, with which the optimal breakpoints are obtained in the two cases are identical. Thus data partitioning can be considered a special case of dynamic rate shaping, where the target bit rate is constant and the cluster is a complete slice.

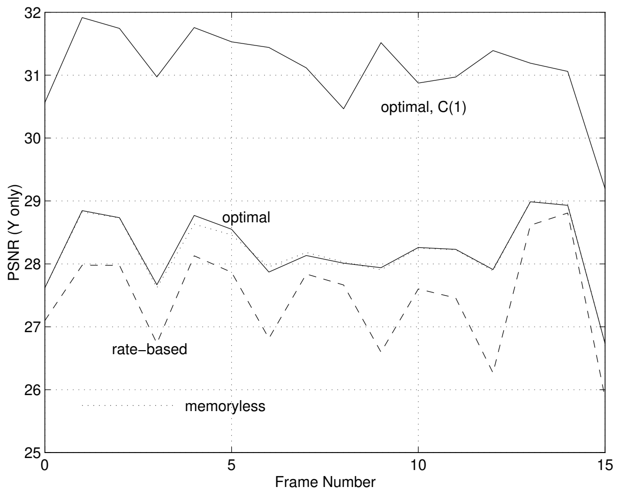

Figure 4-12: PSNR performance for clustered ($C(44)$) rate shaping of "Mobile" using the (causally) optimal, memoryless, and rate-based approaches. The $C(1)$ optimal algorithm is also shown. The source is originally coded at 4 Mbps, and rate shaped at 3.2 Mbps (I period 12, P period 3).

As one might expect, the reduction of degrees of freedom in the clustering approach can only degrade performance. Figure 4-12 depicts the results of the various clustered rate shaping algorithms for the case where the cluster spans a complete slice (C(44), for the particular resolution used, and given that the slices in this case have the width of the picture). Also shown is the $C(1)$ causally optimal algorithm. We can see that all the curves have essentially shifted down by about 2.5–3 dB, without modification of their relative positioning. Note that this represents the worse quality drop possible, since the largest reasonable (but not largest possible) cluster size was used. Of course, the complexity was also significantly reduced, since at each iteration of the bisection algorithm we only have to compute optimal breakpoints for just 30 clusters.

Intermediate cluster sizes can provide a better tradeoff between complexity and performance. In particular, we expect that the $C(4)$ and $C(8)$ memoryless algorithms are very good candidates for software-based implementation in today's general purpose microprocessors.

4.5 Unconstrained Dynamic Rate Shaping

In the constrained dynamic rate shaping problem discussed in the previous section, we considered the DRS problem with the added structural constraint that rate reduction would only occur by eliminating contiguous strings of DCT coefficients at the end of each of the blocks. In this section we lift this constraint, and allow our algorithm to arbitrarily select DCT coefficients for elimination from the bitstream. This represents the most general case of the category of the selective transmission DRS algorithms (see Figure 4-4, page 126), and will be called Unconstrained Dynamic Rate Shaping.

The difference between the two approaches is shown in Figure 4-13. In the constrained case, the breakpoint b is defined as an integer indicating the run-length position at which we start eliminating run-length codes. The range for breakpoint values in this case is $1, \ldots, 64$, where we assume that at least one DCT run-length code will remain in each block (to avoid the complications of recoding the coded block pattern and reexecuting the DC prediction loops when MPEG-2 is used). In the unconstrained case the breakpoint becomes a binary vector $\mathbf{b}$, indicating which DCT coefficients are kept and which are eliminated. In other words we have:

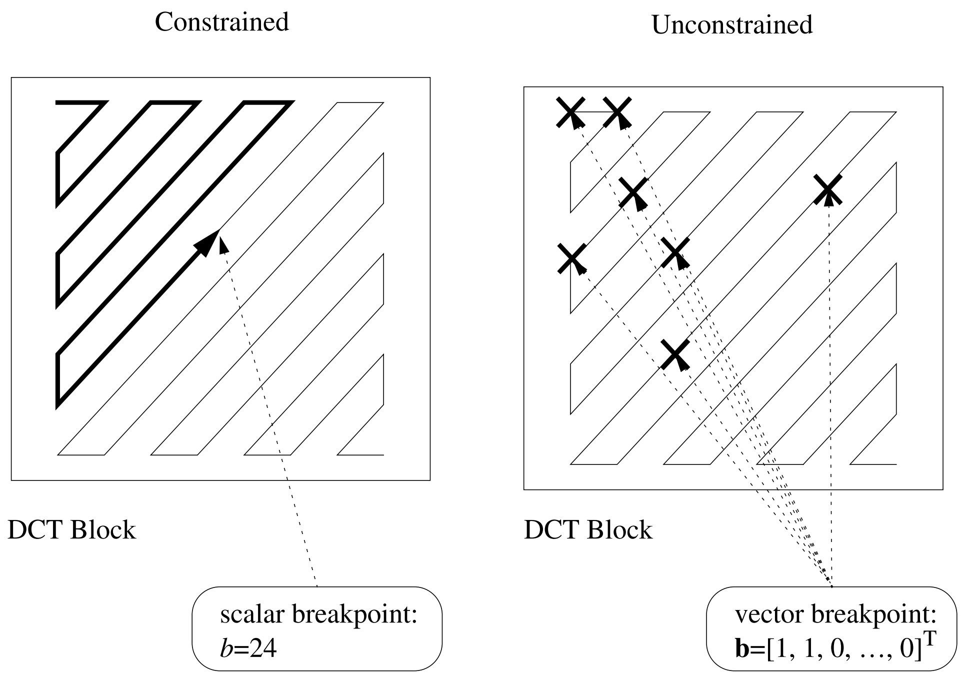

Figure 4-13: The definition of breakpoint in the Constrained and Unconstrained DRS algorithms.

$$\mathbf{b} = \left[b^0, b^1, \ldots, b^{N-1}\right] \tag{4.28}$$

where N is the block size, and

$$b^j = \begin{cases} 0 & \text{if coefficient run-length } j \text{ is eliminated} \\ 1 & \text{otherwise} \end{cases} \tag{4.29}$$

Note that typically the number of run-lengths is much less than N, even in intra blocks.

4.5.1 Problem Formulation

In order to properly formulate the problem we need to explicitly express the rate and distortion in eq. (4.4), which we repeat here for convenience:

$$\min_{\hat{B}(t) \leq B_T(t)} \{\|y - \hat{y}\|\} \tag{4.30}$$

As in the case of the constrained problem, we proceed with an analysis of the distortion starting from a single block. Using similar arguments to the constrained case (the orthonormality of the DCT), we can easily show that if the block $\mathbf{x}$ is rate shaped with the breakpoint vector $\mathbf{b}$, then the distortion between the original block and the modified version $\hat{\mathbf{x}}$ can be expressed as:

$$\|\mathbf{x} - \hat{\mathbf{x}}\| = \sum_{i=0}^{N-1} \overline{b^i} X(i)^2 \tag{4.31}$$

where $\overline{b^i}$ denotes the logical complement of b_i (the value 1 becomes 0, and vice versa), and N is the length of the block.

Due to the independence of the individual blocks of a picture, the total shaping distortion in (4.30) for the intra-only case can be expressed as the sum of the distortions of the individual blocks. To avoid a perceptually meaningless distortion measure, we will restrict the measurement of the distortion to luminance blocks only. Consequently, in order to be able to meaningfully select breakpoint vectors for chrominance blocks, we use the same breakpoint vector for all blocks of a macroblock. With these considerations, we can define the distortion $D_i(\mathbf{b}_i)$ of macroblock i when the breakpoint vector $\mathbf{b}_i$ is used as:

$$D_i(\mathbf{b}_i) = \sum_{j \in \mathcal{Y}} \sum_{k=0}^{N-1} \overline{b_i^k} X_j^i(k)^2 \tag{4.32}$$

where $\mathcal{Y}$ denotes the set of block numbers of the luminance (Y) component of a macroblock.

As in the constrained case, the total rate can be expressed as the sum of the individual rates $R_i(\mathbf{b}_i)$ used to encode each macroblock using breakpoint vector $\mathbf{b}_i$, computed over all blocks of a macroblock:

$$R_i(\mathbf{b}_i) = \sum_{m \in \mathcal{Y} \cup \mathcal{U} \cup \mathcal{V}} R_i^m(b_i) \tag{4.33}$$

where $R_i^m(\mathbf{b})_i)$ denotes the rate for block m. Using (4.32) and (4.33) we can express the minimization problem of (4.30) as follows:

$$\min_{\sum_{i=1}^{M} R_i(\mathbf{b}_i) \le B_T(t)} \left\{ \sum_{i=1}^{M} D_i(\mathbf{b}_i) \right\} \tag{4.34}$$

The mixed-mode case analysis involves the consideration of error propagation due to the recursive nature of the encoding process. As in the constrained DRS case (Section 4.4.1.2), we only consider algorithms that take into account the error accumulated up to the current picture, and not the one that will be propagated to future pictures (causality). We can then show that the per-macroblock distortion $\hat{D}_i(\mathbf{b}_i)$ is given by:

$$\hat{D}_i(\mathbf{b}_i) = \sum_{j \in \mathcal{Y}} \sum_{k=0}^{N-1} \left\{ A_j^i(k)^2 + 2\overline{b_i^k} A_j^i(\mathcal{I}_j^i(k)) E_j^i(k) + E_j^i(k)^2 \right\} \tag{4.35}$$

where $A_j^i(k)$ is the k-th DCT coefficient (in zig-zag scanning order) of the j-th block of the i-th macroblock, $E_j^i(k)$ is the DCT coefficient of the k-th *run-length* of the same block, and $\mathcal{I}_j^i(k)$ is the mapping function between run-length and zig-zag scan positions of the block. Eq. (4.34) with this more general definition of distortion expresses the unconstrained DRS problem in the compressed domain.

4.5.2 Optimal Breakpoint Vector Selection

The Lagrange multiplier approach (Section 4.4.2) can be used to transform (4.34) into an unconstrained minimization problem:

$$\min \left\{ \sum_{i=1}^{M} \hat{D}_i(\mathbf{b}_i) + \lambda \sum_{i=1}^{M} R_i(\mathbf{b}_i) \right\} \tag{4.36}$$

The bisection algorithm can then be used as an iterative algorithm that computes the optimal value λ^* for λ. Within each iteration, however, obtaining the optimum breakpoint vector is no longer trivial. Due to the independent coding of individual blocks, the problem can still be partitioned.

In particular, if we define:

$$L_i(\lambda, \mathbf{b}_i) \stackrel{\text{def}}{=} D_i(\mathbf{b}_i) + \lambda R_i(\mathbf{b}_i) \tag{4.37}$$

we can write (4.36) as:

$$\min \left\{ \sum_{i=1}^{M} L_i(\lambda, \mathbf{b}_i) \right\} \tag{4.38}$$

Hence within each iteration of the bisection, we can separately minimize $L_i(\lambda, \mathbf{b}_i)$ for each macroblock. In contrast with the data partitioning and constrained dynamic rate shaping cases, though, the breakpoint is no longer a scalar, but an N-dimensional binary vector. Consequently, the worst-case number of possible solutions grows from 64^M to 2^{64M}. Considering that M may possibly be in the thousands (e.g., when the optimization window is a complete picture), a full-search approach is impossible.

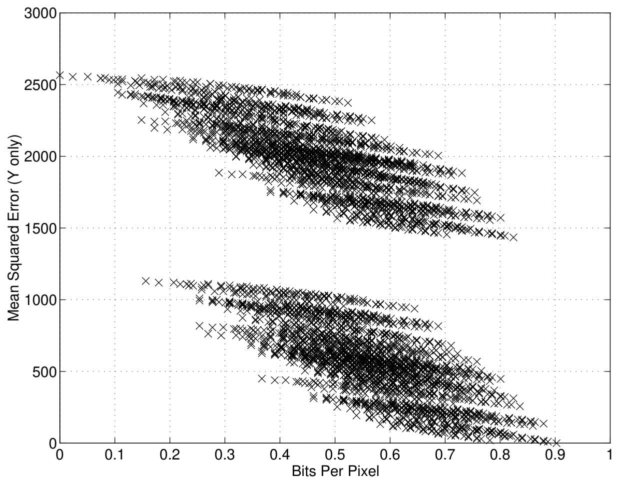

Figure 4-14: $R(D)$ "clouds" from a macroblock of "Mobile", coded at 4 Mbps. The 12 DCT coefficients generate 4,096 different breakpoint vector possibilities.

Figure 4-14 shows the $R(D)$ plot for a macroblock from the "Mobile" sequence, coded at 4 Mbps. We observe that the graph is no longer a curve, and it becomes a "cloud". This particular macroblock had only 12 DCT run-length codes, and hence the number of possible combinations (vector values) is "only" 4,096. Of particular interest is that we, in fact, have two identical clouds, which are just spatially shifted versions of each other. The reason is that there is a dominant DCT coefficient (DC), which in the left-most cloud is excluded, while in the right-most it is included. Particularly important is the fact that the number of points who are $R(D)$-optimal, i.e., define the convex hull, is actually very small. In fact, from the left-most cloud, just one point is included. The consequence of this behavior is that the convergence speed of the bisection iteration for λ is not affected by the dimensionality of the breakpoint space. In our experiments, convergence was achieved within 10–12 iterations, just as in the constrained DRS case.

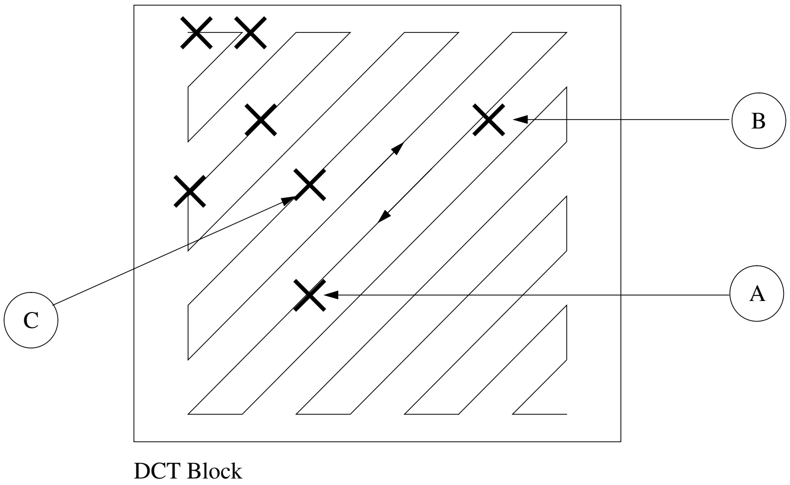

Figure 4-15: Recursive nature of run-length code generation.

We now return to the problem of finding the optimal breakpoint vector within each iteration of the bisection algorithm. A significant complication towards obtaining a solution is the recursive nature of the run-length code generation process. Figure 4-15 illustrates the situation. Recall that a run-length code jointly indicates the number of contiguous zero-valued DCT coefficients and the value of the next non-zero coefficient.

Let us assume that we are processing a block, and are considering whether or not to include the DCT coefficient at point A by examining its contribution to the macroblock rate and distortion. Since the run-length code that will represent A is defined based on the previous non-zero DCT coefficient, the precise rate to be used for A depends on if the coefficient at point B is included or not. In the former case the run-length code is computed starting from point B, while in the latter it is computed starting at point C. These two rates will be different, hence giving two different contributions for $L_i(\lambda, \mathbf{b}_i)$ of this macroblock. Note that the distortions in both cases, however, are identical.

This recursive dependence of the DCT coefficient run-length codes suggests that a fast algorithm may be possible using dynamic programming [13, 15]. In the dynamic programming approach (pioneered by Bellman [13]), a complex multidimensional decision process is broken into a series of successive steps; Bellman's principle of optimality suggests that if the total decision is optimal, then each of the individual ones is also optimal. This allows us to map the complex problem into a much simpler sequential decision process.

Such a dynamic programming algorithm has been used in [106, 73, 74] in the context of optimal quantizer thresholding of JPEG images. The major differentiating factor here is the distortion measure, as in our case it involves the accumulated distortion in the temporal dimension. An additional difference is that here we are dealing with run-length codes, rather than DCT coefficients; in that sense, the process can be considered as iterative, temporally recursive thresholding. In the following we briefly describe the application of the algorithm in the context of DRS.

Let us consider a single macroblock and a given λ. For notation simplicity, we will drop all macroblock indices in the sequel. The algorithm starts from the DCT coefficient of the first run-length and, moving towards the end, examines the benefit of including each run-length. At initialization (or step 0), we then have an all-zero breakpoint configuration and an optimal (at step 0) Lagrangian distortion L_0^* which equals the maximum:

$$L_0^* = \sum_{j \in \mathcal{Y}} \sum_{k=1}^{N-1} \{A_j(k) + E_j(k)\}^2 \tag{4.39}$$

Note that the first run-length code is always included, as was explained in the beginning of this section.

In succeeding steps, we consider the incremental cost reduction ΔL_{ij} of going from run-length n directly to m, skipping those inbetween:

$$\Delta L_{nm} = \sum_{j \in \mathcal{Y}} \left\{ -A_j(\mathcal{I}_j(m) E_j(m) + E_j(m)^2 + \lambda R_j(n,m) \right\} \tag{4.40}$$

where $R_j(n,m)$ is the number of bits needed to encode the run-length code of the DCT coefficient of the m-th run-length code when the run begins at the position of run-length code n. Note that these values can be precomputed at the beginning of the bisection algorithm, and thus be reused for all different values of λ.

The algorithm then proceeds to examine the following coefficients, maintaining the following internal variables: L_k^* indicates the minimum Lagrangian cost associated with having k as the last run-length, S_k is the set of all candidate optimal predecessor run-lengths for k, and k^* denotes the optimal last run-length.

Optimal Breakpoint Selection Algorithm

Step 1: Find current optimal cost and predecessor

Set:

$$J_k^* := \min_{i \in S_{k-1}} \{L_i^* + \Delta L_{ik}\} \tag{4.41}$$

If $L_k^* \leq L_{k^*}^*$ then $k^* = k$. Also set:

$$\text{pred}(k) = \arg \min_{i \in S_{k-1}} \{L_i^* + \Delta L_{ik}\} \tag{4.42}$$

This step finds the optimal Lagrangian cost for this iteration, and marks the run-length as the optimal last if the total cost is less than or equal to the currently optimal one. It also stores the optimal predecessor run-length to k.

Step 2: Prune predecessor set

Set:

$$S_k = \{k\} \cup \{i | i \in S_{k-1} \text{ and } L_i^* < L_k^*\} \tag{4.43}$$

This step only keeps as candidate optimal predecessors those run-lengths which have *less* optimal Lagrangian cost that the current one. This step is responsible for the fast operation of the algorithm [106], and is possible because of the monotonicity property of the Huffman run-length code tables used in both MPEG and JPEG: for any given DCT coefficient level, longer zero run-lengths correspond to codes that have non-decreasing lengths. Thus if we consider a predecessor l to k with equal or higher optimal Lagrangian cost from k, the cost of going to a future run-length from either l or k will always be less than or equal if we do so from k (since the run-length encoding for the longer path will take at least as many bits).

Step 3: Termination test

If all run-length codes have been processed, then stop. Otherwise set $k := k + 1$ and go to Step 1.

At the end of the algorithm, we have the optimal last run-length k^*, and we also have the linked list pred($\cdot$) of optimal predecessors. Starting from k^* and traversing the list towards the first DCT run-length code, we obtain the optimal list of all run-length codes whose DCT coefficients are to be included in the rate shaped bitstream.

4.5.3 Performance Evaluation

The complexity of the unconstrained DRS approach is dominated by the dynamic programming algorithm, which is arguably quite complex to implement. The amount of computation required depends heavily on the number of run-length codes, and whether or not the second step of the dynamic programming algorithm can quickly prune a large number of possible predecessors. Even though in practice this is typically the case, the fact that the algorithm is executed for every macroblock and for every step of the bisection iteration represents a significant concern.

The situation is accentuated in mixed-mode coding, where the additional overheads of dual loop motion compensation, and both forward and inverse DCT computations are incurred. Although a memoryless approach eliminates these latter tasks, it has only a small effect on the complexity of the dynamic programming step. Finally, the unconstrained approach has the additional processing overhead of having to generate new Huffman run-length codes for the resulting breakpoint configurations of each block, whereas the constrained algorithm merely performs truncation.

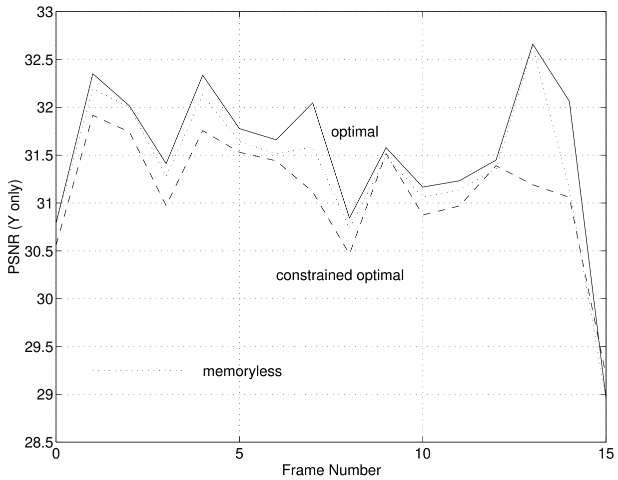

Figure 4-16: PSNR performance for rate shaping of “Mobile” using the unconstrained optimal and memoryless algorithms, as well as the constrained optimal algorithm. The source is originally coded at 4 Mbps and rate shaped at 3.2 Mbps (I period 12, P period 3).

As a result, the unconstrained DRS scheme is mostly useful as a benchmark of optimal behavior for faster algorithms (recall that the unconstrained algorithm is optimal among all those of the selective transmission category). Figure 4-16 depicts the results of applying the unconstrained optimal algorithm on the “Mobile” sequence, coded at 4 Mbps and rate shaped at 3.2 Mbps. We also show the results of the memoryless version of the algorithm, as well as those of the *constrained* optimal algorithm. Note how close the memoryless algorithm tracks the performance of the optimal one, as in the constrained rate shaping case.

We observe that, as expected, the unconstrained algorithm outperforms the constrained one. The difference though is very small, in this case just about 0.5 dB on the average. This is a quite important and perhaps surprising result, since it indicates that the constrained approach produces results which are extremely close to optimal. This includes the memoryless unconstrained algorithm, which as we have seen is very simple to implement.

The reason for this behavior is the structure of the DCT coefficient zig-zag scanning pattern: it provides a quite successful ordering of the DCT coefficients according to their importance. In other words, most of the time the unconstrained algorithm removes DCT run-length codes at the end of the blocks and following a zig-zag scanning order, with very few discontinuities. The decisions taken, then, by the constrained and unconstrained schemes vary very slightly, leading to a correspondingly small difference in terms of decoded video quality.

4.6 Applications

Although the concept of dynamic rate shaping was presented in Section 4.1 primarily from the point of view of an interface between a compressed video source and the network transport, the idea of modifying the rate of compressed video can have several other applications. Some examples are briefly described below.

4.6.1 Transcoding and Codec Interoperability

The existence of multiple different standards for video compression naturally raises the issue of their interoperability. As use of digital video content finds more widespread use, it is expected that in several cases users will need to be able to convert material from one format to another. An additional reason for transcoding is that the various standards were designed for different applications. For example, MPEG-1 [4] addresses digital video for storage media (e.g., CD-ROM), MPEG-1 [6] addresses primarily broadcast TV applications, whereas H.261 [3] is targeting videoconferencing applications over low bit rate lines.

Most of the recent standards share a common algorithmic foundation in their design: use of motion compensation, and use of block-based transform coding. This makes it possible to design relatively simple transcoding algorithms, since the task involves primarily generation of new headers etc. As the new bitstream is constructed according to the target standard, one cannot guarantee (or even predict) its exact rate behavior. When rate constraints are imposed by the application in which the resultant bitstream will be utilized, rate shaping can be used as a filtering stage to provide compliance. Although in some cases the rate mismatch may be small, it still can be sufficient to generate disturbing errors at a decoder (e.g., due to buffer overflows).

A similar example is interoperability between codecs of the same standard. Implementation considerations in actual systems result in decoders with various different performance capabilities. Typically, systems hosted on general purpose computers are limited by the sustainable speed of the shared data bus. Dedicated systems, on the other hand, can achieve significantly higher rates. Hence rate shaping can also be used in this case to reduce the rate to appropriate levels.

4.6.2 Trick Modes

The term "trick modes" refers the capability of a video playback system to perform fast forward and reverse operations. Although in the analog domain the operation is relatively straightforward, the digital domain represents several challenges [123].

Let us consider the case of an MPEG-2 bitstream. In order to perform fast forward or fast reverse, one will rely on either the I pictures, or a combination of I and P pictures, depending on the desired playback speed and the sequence structure employed in the bitstream (note that typically this structure is not known in advance). During the trick mode operation, the playback system will simply skip over any pictures that do not require decoding, and submit to the decoder only those that construct the fast forward/reverse sequence. This operation can be performed quite easily, since pictures can be very easily located from their associated fixed-length start codes.

When the individual pictures are merged to form the new trick mode bitstream, any rate constraints that the original bitstream may had satisfied will no longer be met. Since the decoder will always have a rate constraint, this can cause data loss across the communication path. Even with extensive buffering, disregarding the bitstream's rate will result in a variable playback rate which can be very annoying to viewers. In other words, frame rate continuity is preferable to spatial quality continuity across frames. Rate shaping can be used in this case, to properly modify the trick mode bitstream.

4.6.3 Communication in Heterogeneous Networks

Finally, another case where the DRS concept can be used is communication in heterogeneous networks. An interesting example is multipoint communication with both wired and wireless (e.g., mobile) hosts. The bandwidth of wireless channels is typically an order of magnitude less than that of wired ones, in the order of few Mbps. It is also time varying, due to multipath distortion phenomena.

Let us consider the case of a multi-point videoconference with three participants, one of which is using a wireless channels. Due to the capacity mismatch between the different channels, the brute-force approach in providing viable communication would be to limit the transmission rate to that of the least capable participant. This will severely degrade the quality for the wired participants, in order to accommodate the wireless one.

A better alternative would be to use a rate shaping processor at the base station of the wireless user, in order to filter the video traffic originating from the wired participants. This way, the latter would be able to fully utilize the bandwidth available to them, and still not exclude the wireless participant from the session. If the wireless channel capacity is too low to be handled by rate shaping, one can combine scalable coding techniques and DRS to achieve much larger rate reductions. A more detailed description of how DRS can be used in a networking architecture providing scalable flows can be found in [12].

4.7 Concluding Remarks

The novel concept of Dynamic Rate Shaping was introduced as a general mechanism for modifying the bit rate of motion-compensated, transform coded digital video (including MPEG-1, MPEG-2, H.261, and JPEG). One of its primary applications is as an adaptation mechanism between compressed video rate characteristics and network transport service capabilities, where it can provide universal interoperability between VBR and CBR codecs and both guaranteed-service and best-effort networks. Other applications include transcoding, trick mode operation, as well as communication in heterogeneous network environments (including mobile systems). DRS attempts to eliminate the system-level barriers posed by the current dominance of CBR and VBR compressed video, by providing a continuum of alternatives between the two.

We have provided an analysis of DRS in an operational rate-distortion context. We have derived a family of different algorithms, classified in two broad categories: requantization-based, and selective transmission-based. Although the focus was on the latter category, comparative examination of the two was provided using recoding. The major contributions of this chapter can be summarized as follows:

1. We have derived an optimal algorithm for the problem of constrained rate shaping using Lagrange multipliers, and have shown that it outperforms recoding across a wide range of target rates. The constraint mandates that only contiguous strings of DCT coefficients at the end of each block can be eliminated. The algorithm was shown to have a complexity between an encoder and a decoder.

2. We have shown that a memoryless constrained algorithm (optimal for the intra-only case) performs almost identically to the optimal one, and hence is an excellent candidate for actual implementation since its complexity is less than a decoder's.

3. In order to reduce the computational complexity, the clustering approach was introduced for both the optimal and the memoryless constrained algorithms. Due to the tradeoff between complexity and quality, two levels of clustering (4 and 8) were identified as possible candidates for software-based implementation.

4. We have shown that clustered rate shaping is essentially identical to data partitioning when the cluster spans a whole picture slice. Although in data partitioning two output bitstreams are generated, the mathematical problems are identical. Hence data partitioning can be considered as a special case of rate shaping.

5. We have derived an optimal algorithm for the problem of unconstrained rate shaping, where selection of DCT coefficients for elimination can be arbitrary. The algorithm was based again on the Lagrange multipliers approach and the bisection algorithm. A fast algorithm based on dynamic programming had to be used, however, in order to obtain optimal configurations for breakpoint vectors. We showed that the quality improvement of the unconstrained approach is marginally better than that of the constrained one, even though the complexity of the former scheme is significantly higher. This favors the constrained algorithm, and particularly its memoryless version, for actual implementations.

A most interesting result, from a practical point of view, is the performance of the constrained memoryless algorithm vs. the unconstrained optimal one, especially in view of their difference in complexity. As we have mentioned in the introduction of this chapter, the implementation complexity of DRS is a key component of its viability in real multimedia systems. In order to be of most use, it must ideally be implementable using a purely software-based approach. In Section 4.4.4 we saw that, by using the clustering approach, sufficient reduction in complexity can be achieved for the memoryless algorithm in order to make this possible. The penalty in performance will be small (yet perceptible), in the order of 1 dB compared to the optimal result.

Chapter 5

Conclusions

We have introduced the concept of Dynamic Rate Shaping (DRS), a technique to adapt the rate of compressed video bitstreams (e.g., MPEG-1, MPEG-2, H.261, JPEG) to dynamically varying rate constraints by operating directly in the compressed signal domain. This capability of post-encoding rate control has been shown to be critical for multimedia communication systems, due to the unavoidable incompatibilities between the design of general-purpose codecs and networks. DRS can guarantee universal interoperability between the two, even when the quality of service guarantees provided by the network are time-varying. There are other domains in which DRS can be utilized, including transcoding and codec interoperability, trick modes, as well as communication in heterogeneous networks.

We have shown that the concept evolves quite naturally by considering first the operation of an actual multimedia communications testbed—Xphone—that we have developed. Due to the very challenging communication environment in which Xphone is based upon (best-effort operating system and network), maintaining acceptable quality requires that the system is capable of quickly adapting to changing network conditions.

For that purpose, an adaptive rate control scheme was developed, which ensures sufficiently high frame rates by trading off spatial quality. To mitigate the adverse effects of jitter to the end-to-end delay of videoconferencing sessions, silence detection was used to transform the audio signal to a "bursty" one that can more easily be transported by CSMA/CD (Ethernet) internetworks. Finally, in order to provide accurate audio-video synchronization, a scalable synchronization algorithm was developed that bounds the synchronization misadjustment by the frame period. All these are examples of algorithms that adapt different parts of a multimedia system to the behavior accepted by others.

A natural extension of this approach was followed for the video signal, in the form of data partitioning. This process, first introduced in the MPEG-2 standard, segments a compressed bitstream into two layers or partitions. The one containing the most important information is transported with reliability, while the other one uses a less robust "channel". This can achieve improved average quality at the receiver, since critical information is guaranteed to be received. Channel reliability, when not directly supported by the network, can be emulated by increased forward error correction. We have posed the problem of optimal data partitioning in an operational rate-distortion context, and discussed several optimal and fast algorithms.

By eliminating the second partition, and allowing the bit rate to be variable, we naturally arrive at the idea of dynamic rate shaping (DRS). DRS provides an interface between the encoder and the network, with which the encoder's output can be perfectly matched to the network's quality of service characteristics. This helps to avoid wasting network resources (when quality of service guarantees can be provided), or avoids information loss that can lead to severe quality impairments.

DRS bridges the gap between constant and variable bit rate video, providing a continuum of possibilities between the two. It frees the network designer from the need to derive accurate and universally applicable models for video traffic and, more importantly, it opens up the design options for efficient and scalable admission control algorithms. Following the analysis of data partitioning, the optimal DRS problem was posed in an operational rate-distortion context, and a family of algorithms was identified. Not surprisingly, we also showed that data partitioning is a special case of DRS, and more specifically "clustered constrained" DRS.

A key result was the discovery of very simple DRS algorithms that perform very closely to the optimal ones. The consequences of this simplicity is that even software-based real-time implementations are possible in modern general purpose microprocessors. This can greatly facilitate the incorporation of DRS into actual multimedia communication systems.

A number of issues can be raised from the results of this thesis, warranting further investigation. An interesting question is the examination of the dynamics of rate shaping in real communication environments. The constraints presented to the DRS processor essentially form a closed-loop feedback control system, whose behavior should be carefully characterized. The fact that DRS can be applied to an arbitrary time scale (i.e., with an arbitrary optimization window), guarantees that, whatever the dynamics of the network, DRS can be applied in a stable manner. Closely related to that is the transformation of rate constraints to optimization window bit budgets, since it depends on the specifics of the particular communication architecture. Optimal architectures for DRS processors are also of interest, particularly using software-based implementations. Finally, joint requantization and selective transmission approaches ("Generalized" DRS) are worth exploring, particularly for very large rate conversion ratios.

A more deeply rooted question is whether or not one can construct video coding algorithms that are inherently amenable to rate shaping. Clearly all coding schemes can be subjected to DRS, as long as we can associate a distortion and rate with each of the bitstream components. Whether or not DRS will result in acceptable quality, however, depends on the structure of the encoding scheme. A good example of a representation in which rate shaping is extremely simple, is the decimal numbering system: assuming our source is a real number, its coded representation could be one that utilizes only the first decimal digits. If rate shaping is required, then we can simply truncate the representation to fewer digits with no processing whatsoever; the new representation is guaranteed to be optimal under a squared error criterion. We should note, however, that supporting such functionality can potentially degrade the coding performance of the algorithm, since the design space becomes constrained.

References

[1] Encoding Parameters of Digital Television for Studios. ITU-R (formerly CCIR) Recommendation 601. 1982.

[2] B-ISDN Recommendations, CCITT COM XVIII-R 23-E. February 1990.

[3] Video Codec for Audio Visual Services at $p \times 64$ kbits/s. ITU-T (formerly CCITT) Recommendation H.261, CDM XV-R 37-E. August 1990.

[4] Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1,5 Mbit/s, ISO/IEC 11172-2 International Standard (MPEG-1). 1993.

[5] Information Technology – Digital Compression and Coding of Continuous-Tone Still Image's, ISO/IEC 10918 International Standard (JPEG). 1994.

[6] Information Technology – Generic Coding of Moving Pictures and Associated Audio, ITU-T Draft Recommendation H.262, ISO/IEC 13818 Draft International Standard (MPEG-2). 1994.

[7] A. S. Acampora and M. J. Karol. An Overview of Lightwave Packet Networks. *IEEE Network Magazine*, 3(1):29–41, January 1989.

[8] L. Aguilar, J. J. Garcia-Luna-Aceves, D. Moran, E. J. Craighill, and R. Brungardt. Architecture for a Multimedia Teleconferencing System. In *Proceedings, ACM SIGCOMM '86 Symposium*, pages 126–136, August 1986.

[9] Sudhir R. Ahuja and J. Robert Ensor. Coordination and Control of Multimedia Conferencing. *IEEE Communications Magazine*, 30(5):38–43, May 1992.

[10] Scientific American. Special Issue: Communications, Computers and Networks. September 1991.

[11] P. H. Ang, P. A. Ruetz, and D. Auld. Video Compression Makes Big Gains. *IEEE Spectrum*, pages 16–19, October 1991.

[12] C. Aurrecoechea, A. Campbell, A. Eleftheriadis, and H. Hadama. Meeting End-to-End QoS Challenges for Adaptive Digital Video Flows. In *Proceedings, 6th IFIP International Conference on High Performance Networking (HPN-95)*, Palma de Mallorca, Spain, September 1995 (to appear).

[13] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[14] T. Berger. *Rate Distortion Theory*. Prentice Hall, Englewood Cliffs, New Jersey, 1971.

[15] D. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[16] M. C. Buchanan and P. T. Zellweger. Scheduling Multimedia Documents Using Temporal Constraints. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 223–235, November 1992.

[17] A. Campbell, D. Hutchinson, and C. Aurrecoechea. A Dynamic QoS Management Scheme for Adaptive Hierarchically Coded Flows. In *Proceedings, 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 107–118, April 1995.

[18] S. Casner, C. Lynn Jr., P. Park, K. Schroder, and C. Topolcic. Experimental Internet Stream Protocol, Version 2. RFC 1190, USC/Information Sciences Institute, 1989.

[19] S.-F. Chang, D. Anastassiou, A. Eleftheriadis, J. Meng, S. Paek, S. Pejhan, and J. R. Smith. Development of Advanced Image/Video Servers in a Video on Deman Testbed. In *Proceedings, IEEE Visual Signal Processing and Communications Workshop*, New Brunswick, New Jersey, September 1994.

[20] S.-F. Chang and A. Eleftheriadis. Error Accumulation of Repetitive Image Coding. In *Proceedings, IEEE International Symposium on Circuits and Systems*, pages 3.201–3.204, London, England, May–June 1994.

[21] S.-F. Chang, A. Eleftheriadis, and D. Anastassiou. Development of Columbia's Video on Demand Testbed. *Image Communication Journal, Special Issue on Video on Demand and Interactive Television*, 1995 (to appear).

[22] C.-F. Chen and K. K. Pang. Hybrid Coders with Motion Compensation. *Multidimensional Systems and Signal Processing*, 3:241–266, 1992.

[23] C.-F. Chen and K. K. Pang. The Optimal Transform of Motion-Compensated Frame Difference Images in a Hybrid Coder. *IEEE Trans. on Circuits and Systems–II: Analog and Digital Signal Processing*, 40(6):393–397, June 1993.

[24] N. K. Cheung. The Infrastructure for Gigabit Networks. *IEEE Communications Magazine*, pages 60–68, April 1992.

[25] D. D. Clark, M. L. Lambert, and L. Zhang. NETBLT: A Bulk Data Transfer Protocol. RFC 998, 1987.

[26] D. D. Clark, M. L. Lambert, and L. Zhang. NETBLT: A High Throughput Transport Protocol. In *Proceedings, ACM SIGCOMM '87 Workshop*, pages 353–359, August 1987.

[27] D. D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network. In *Proceedings, ACM SIGCOMM '92*, pages 14–26, August 1992.

[28] D. D. Clark and D. L. Tennenhouse. Architectural Considerations for a New Generation of Protocols. In *Proceedings, ACM SIGCOMM '90 Symposium*, pages 200–208, September 1990.

[29] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley & Sons, New York, New York, 1991.

[30] L. F. R. da Costa Carmo, P de Saqui-Sannes, and J.-P. Courtiat. Basic Synchronization Concepts in Multimedia Systems. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 85–96, November 1992.

[31] R. B. Dannenberg, T. Neuendorffer, J. M. Newcomer, and D. Rubine. Tactus: Toolkit-Level Support for Synchronized Interactive Multimedia. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 264–275, November 1992.

[32] S. Deering. Host Extensions for IP Multicasting. RFC 1112, Standford University, 1989.

[33] S. Deering and B. Hinden. Internet Protocol, Version 6 (IPv6) Specification. IETF IPng Working Group Draft, March 1995.

[34] S. E. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proceedings, ACM SIGCOMM '88 Symposium*, pages 55–64, August 1988.

[35] W. A. Doeringer, D. Dykeman, M. Kaiserswerth, B. W. Meister, H. Rudin, and R. Williamson. A Survey of Light-Weight Transport Protocols for High-Speed Networks. *IEEE Transactions on Communications*, 38(11):2025–2039, November 1990.

[36] G. Drapeau and H. Greenfield. MAEstro - A Distributed Multimedia Authoring Environment. In *Proceedings, Summer 1991 USENIX Conference*, pages 315–328, 1991.

[37] P. Druschel, M. B. Abbott, M. Pagels, and L. Peterson. Analysis of I/O Subsystem Design for Multimedia Workstations. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 251–263, November 1992.

[38] A. Eleftheriadis and D. Anastassiou. Optimal Data Partitioning of MPEG-2 Coded Video. In *Proceedings, 1st IEEE International Conference on Image Processing*, pages I.273–I.277, Austin, Texas, November 1994.

[39] A. Eleftheriadis and D. Anastassiou. Meeting Arbitrary QoS Constraints Using Dynamic Rate Shaping of Coded Digital Video. In *Proceedings, 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 95–106, Durham, New Hampshire, April 1995.

[40] A. Eleftheriadis and D. Anastassiou. Constrained and General Dynamic Rate Shaping of Compressed Digital Video. In *Proceedings, 2nd IEEE International Conference on Image Processing*, Washington, DC, October 1995 (to appear).

[41] A. Eleftheriadis and A. Jacquin. Model-Assisted Coding of Videoteleconferencing Sequences at Low Bit Rates. In *Proceedings, IEEE International Symposium on Circuits and Systems*, pages 3.177–3.180, London, England, May–June 1994.

[42] A. Eleftheriadis and A. Jacquin. Automatic Face Location Detection and Tracking for Model-Assisted Coding of Video Teleconferencing Sequences at Low Bit Rates. *Image Communication Journal*, 1995 (to appear).

[43] A. Eleftheriadis and A. Jacquin. Automatic Face Location Detection for Model-Assisted Rate Control in H.261-Compatible Coding of Video. *Image Communication Journal, Special Issue on Very Low Bit Rate Video Coding*, 1995 (to appear).

[44] A. Eleftheriadis and A. Jacquin. Low Bit Rate Model-Assisted H.261-Compatible Coding of Video. In *Proceedings, 2nd IEEE International Conference on Image Processing*, Washington, DC, October 1995 (to appear).

[45] A. Eleftheriadis, S. Pejhan, and D. Anastassiou. Algorithms and Performance Evaluation of the Xphone Multimedia Communication System. In *Proceedings, ACM Multimedia 93 Conference*, pages 311–320, August 1993.

[46] A. Eleftheriadis, S. Pejhan, and D. Anastassiou. Multicast Group Address Management and Connection Control for Multi-Party Applications. Technical Report CU/CTR/TR 351–93–31, Center for Telecommunications Research, Columbia University, November 1993.

[47] A. Eleftheriadis, S. Pejhan, and D. Anastassiou. Architecture and Algorithms of the Xphone Multimedia Communication System. *ACM/Springer Verlag Multimedia Systems Journal*, 2(2):89–100, August 1994.

[48] A. Eleftheriadis, S. Pejhan, and D. Anastassiou. Multicast Address Management and Connection Control for Multi-Party Applications. In *Proceedings, IEEE Infocom '95 Conference*, pages 386–393, Boston, Massachusetts, April 1995.

[49] A. Erramilli and R. P. Singh. A Reliable and Efficient Multicast Protocol for Broadband Broadcast Networks. In *Proceedings, ACM SIGCOMM '87 Workshop*, pages 343–352, August 1987.

[50] H. Everett. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*, 11:399–417, 1963.

[51] D. C. Feldmeier. Multiplexing Issues in Communication System Design. In *Proceedings, ACM SIGCOMM '90 Symposium*, pages 209–219, September 1990.

[52] D. C. Feldmeier. A Framework of Architectural Concepts for High Speed Communications Systems. *IEEE Journal on Selected Areas in Communications*, 11(3), March 1993.

[53] D. C. Feldmeier. An Overview of the TP++ Transport Protocol Project. In Ahmed Tantawy, editor, *High Performance Communication*. Kluwer Academic Publishers, 1993.

[54] D. Ferrari and D. C. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, April 1990.

[55] T. Fisher. Real-Time Scheduling Support in Ultrix 4.2 for Multimedia Communication. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 282–288, November 1992.

[56] ATM Forum. *ATM User-Network Interface Specification, Version 3.0.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[57] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Boston, Massachusetts, 1992.

[58] C. A. Gonzales and E. Viscito. Motion Video Adaptive Quantization in the Transform Domain. *IEEE Transactions on Circuits and Systems for Video Technology*, 1(4):374–378, December 1991.

[59] I. W. Habib and T. N. Saadawi. Multimedia Traffic Characteristics in Broadband Networks. *IEEE Communications Magazine*, 30(7):48–54, July 1992.

[60] R. G. Herrtwich and L. Delgrossi. Beyond ST-II: Fulfilling the Requirements of Multimedia Communication. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 23–29, November 1992.

[61] D. Heyman, A. Tabatabai, and T. V. Lakshman. Statistical Analysis and Simulation Study of Video Teleconference Traffic in ATM Networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2(1):49–59, March 1992.

[62] Editor J. F. Koegel Buford. *Multimedia Systems.* ACM Press SIGGRAPH Series. Addison-Wesley, New York, New York, 1994.

[63] S. Jacobs, A. Eleftheriadis, and D. Anastassiou. Silence Detection in Multimedia Networks. Technical Report CU/CTR/TR 407–95–13, Center for Telecommunications Research, Columbia University, May 1995.

[64] A. Jacquin and A. Eleftheriadis. Automatic Face Location Detection for Model-Assisted Rate Control in H.261 Compatible Coding of Video. In *Proceedings, International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, June 1995 (to appear).

[65] N. Jain, M. Schwartz, and T. R. Bashkow. Transport Protocol Processing at GBPS Rates. In *Proceedings, ACM SIGCOMM '90 Symposium*, pages 188–199, September 1990.

[66] S. Jamin, S. Shenker, L. Zhang, and D. D. Clark. An Admission Control Algorithm for Predictive Real-Time Service. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 308–315, November 1992.

[67] N. S. Jayant and P. Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice Hall, Englewood Cliffs, New Jersey, 1984.

[68] K. Jeffay, D. L. Stone, and F. D. Smith. Transport and Display Mechanisms for Multimedia Conferencing Across Packet-Switched Networks. *Computer Networks and ISDN Systems*, 26(10):1281–1304, July 1994.

[69] K. Jeffay, D. L. Stone, T. Talley, and F. D. Smith. Adaptive, Best-Effort Delivery of Digital Audio and Video Across Packet-Switched Networks. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 1–12, November 1992.

[70] J.Postel. Internet Protocol. RFC 791, USC/Information Sciences Institute, 1981.

[71] J.Postel. Transmission Control Protocol. RFC 793, USC/Information Sciences Institute, 1981.

[72] Ronald K. Jurgen. Digital Video. *IEEE Spectrum*, pages 24–30, March 1992.

[73] K. Ramchandran and M. Vetterli. Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility. In *Proceedings, Picture Coding Symposium '93*, March 1993.

[74] K. Ramchandran and M. Vetterli. Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility. *IEEE Transactions on Image Processing, Special Issue on Video Sequence Compression*, 3(5):700–704, September 1994.

[75] H. Kanakia, P. P. Mishra, and A. Reibman. An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport. In *Proceedings, ACM SIGCOMM '94 Conference*, pages 20–31, September 1993.

[76] L. Kleinrock. The Latency/Bandwidth Tradeoff in Gigabit Networks. *IEEE Communications Magazine*, 30(4):36–40, April 1992.

[77] F. Kretz and F. Colaitis. Standardizing Hypermedia Information Objects. *IEEE Communications Magazine*, 30(5):60–70, May 1992.

[78] A. A. Lazar, G. Pacifici, and D. E. Pendarakis. Modeling Video Sources for Real Time Scheduling. *ACM/Springer Verlag Multimedia Systems Journal*, 1(6):253–266, April 1994.

[79] D. Lee, B. Melamed, A. R. Reibman, and B. Sengupta. TES Modeling for Analysis of a Video Multiplexer. *Performance Evaluation*, pages 21–34, November 1992.

[80] J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman. *The Design and Implementation of the 4.3 BSD UNIX Operating System*. Addison-Wesley, 1989.

[81] Didier LeGall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[82] D. W. Lin, M.-H. Wang, and J.-J. Chen. Optimal Delayed Coding of Video Sequences Subject to a Buffer Size Constraint. In *Proceedings, SPIE Visual Communications and Image Processing Conference*, Cambridge, Massachusetts, November 1993.

[83] Ming Liou. Overview of the $p \times 64$ kbit/s Video Coding Standard. *Communications of the ACM*, 34(4):59–63, April 1991.

[84] T. D. C. Little and A. Ghafoor. Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communications*, 8(3):413–427, April 1990.

[85] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J. Robbins. Performance Models of Statistical Multiplexing in Packet Video Communications. *IEEE Transactions on Communications*, pages 834–844, July 1988.

[86] D. L. Mills. On the Accuracy and Stability of Clocks Synchronized by the Network Time Protocol in the Internet System. *Computer Communications Review*, 20(1):65–75, January 1990.

[87] D. L. Mills. Internet Time Synchronization: The Network Protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, October 1991.

[88] M. Moran and R. Gusella. System Support for Efficient Dynamically Configurable Multi-Party Interactive Multimedia Applications. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 133–146, November 1992.

[89] D. G. Morrison, M. E. Nilsson, and M. Ghanbari. Reduction of the Bit-Rate of Compressed Video While in its Coded Form. In *Proceedings, Packet Video Workshop '94*, pages D17.1–D17.4, 1994.

[90] R. Needham and A. Nakamura. An Approach to Real-Time Scheduling but is it Really a Problem for Multimedia ? In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 30–37, November 1992.

[91] A. N. Netravali and B. G. Haskell. *Digital Pictures: Representation, Compression, and Standards (2nd ed.).* Plenum Press, New York, New York, 1995.

[92] A. N. Netravali, W. D. Roome, and K. Sabnani. Design and Implementation of a High-Speed Transport Protocol. *IEEE Transactions on Communications*, 38(11):2010–2023, November 1990.

[93] P. Newman. Traffic Management for ATM Local Area Networks. *IEEE Communications Magazine*, 32(8):44–50, August 1994.

[94] C. Nikolaou. An Architecture for Real-Time Multimedia Communication Systems. *IEEE Journal on Selected Areas in Communications*, 8(3):391–400, April 1990.

[95] J. D. Northcutt, G. A. Wall, J. G. Hanko, and E. M. Kuerner. A High Resolution Video Workstation. *Image Communication*, 4(4-5), August 1992.

[96] A. Ortega, K. Ramchandran, and M. Vetterli. Optimal Buffer-Contrained Source Quantization and Fast Approximations. In *Proceedings, IEEE Intl. Symposium on Circuits and Systems, ISCAS '92*, San Diego, May 1992.

[97] A. Papoulis. *Signal Analysis.* McGraw-Hill, New York, New York, 1977.

[98] A. Papoulis. *Probability, Random Variables, and Stochastic Processes, (3rd ed.).* McGraw-Hill, New York, New York, 1991.

[99] J. C. Pasquale, G. C. Polyzos, E. W. Anderson, and V. P. Kompella. The Multimedia Multicast Channel. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 185–196, November 1992.

[100] S. Pejhan, A. Eleftheriadis, and D. Anastassiou. Multiparty multimedia communication systems. In *Proceedings, Iranian Conference on Electrical Engineering*, Tehran, Iran, May 1994.

[101] S. Pejhan, A. Eleftheriadis, and D. Anastassiou. Distributed Multicast Address Management in the Global Internet. *IEEE Journal on Selected Areas in Communications, Special Issue on Internetworking*, 1995 (to appear).

[102] W. Pennebaker and J. Mitchell. *The JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, New York, 1993.

[103] T. F. La Porta and M. Schwartz. Architectures, Features, and Implementation of High-Speed Transport Protocols. *IEEE Network Magazine*, 5(5):14–22, May 1991.

[104] A. Puri and R. Aravind. Motion-Compensated Video Coding with Adaptive Perceptual Quantization. *IEEE Transactions on Circuits and Systems for Video Technology*, 1(2):210–221, June 1991.

[105] S. Ramanathan and P. V. Rangan. Continuous Media Synchronization in Distributed Multimedia Systems. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 289–296, November 1992.

[106] K. Ramchandran. *Joint Optimization Techniques in Image and Video Coding with Applications to Multiresolution Digital Broadcase*. PhD thesis, Columbia University, New York, New York, 1993.

[107] P. V. Rangan, H. M. Vin, and S. Ramanathan. Designing an On-Demand Multimedia Service. *IEEE Communications Magazine*, 30(7):56–64, July 1992.

[108] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, San Diego, California, 1990.

[109] A. Reibman. DCT-Based Embedded Coding of Packet Video. *Image Communication*, 3:231–237, 1991.

[110] A. R. Reibman and A. Berger. Traffic Descriptors for VBR Video Teleconferencing over ATM Networks. In *Proceedings, Globecom '92 Conference*, pages 1135–1139, December 1992.

[111] K. Rothermel and G. Dermler. Synchronization in Joint-Viewing Environments. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 97–109, November 1992.

[112] E. M. Schooler. The Impact of Scaling on a Multimedia Connection Architecture. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 302–307, November 1992.

[113] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF Audio-Video Transport Working Group Draft, March 1995.

[114] W. Stallings. *ISDN and Broadband ISDN.* Macmillan Publishing Company, New York, 1992.

[115] R. Steinmetz. Synchronization Properties in Multimedia Systems. *IEEE Journal on Selected Areas in Communications*, 8(3):401–412, April 1990.

[116] W. R. Stevens. *UNIX Network Programming.* Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[117] J. Sutherland and L. Litteral. Residential Video Services. *IEEE Communications Magazine*, 30(7):36–41, July 1992.

[118] H. Vin and P. V. Rangan. Admission Control Algorithms for Multimedia On-Demand Servers. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 50–62, November 1992.

[119] G. Vonderweidt, J. Robinson, C. Toulson, J. Mastronardi, E. Rubinov, and B. Prasada. A Multipoint Communication Service for Interactive Applications. *IEEE Transactions on Communications*, 39(12):1875–1885, December 1991.

[120] G. A. Wall, J. G. Hanko, and J. D. Northcutt. Bus Bandwidth Management in a High Resolution Video Workstation. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 236–248, November 1992.

[121] Gregory K. Wallace. The JPEG Still Picture Compression Standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[122] F. M. Wang, R. Mokry, T. H. Chiang, A. Eleftheriadis, and D. Anastassiou. Compatible Coding of Digital Interlaced HDTV Using Prediction of the Even Fields from the Odd Fields. In *Proceedings, Fourth International Workshop on HDTV and Beyond*, Torino, Italy, September 1991. This paper is reprinted in the book "Signal Processing of HDTV", III, H. Yasuda and L. Chiariglione, Eds., Elsevier, 1992.

[123] J. Watkinson. *The Art of Digital Video.* Focal Press, Oxford, England, 1990.

[124] R. W. Watson. *IPC-Interface and End-to-End Protocols*, volume 107 of *Lecture Notes in Computer Science*, chapter 7, pages 140–174. Springer Verlag, 1981.

[125] L. Wei, F.-C. Liaw, D. Estrin, A. Romanow, and T. Lyon. Analysis of a Resequencer Model for Multicast over ATM Networks. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 197–208, November 1992.

[126] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(9):1445–1453, 1988.

[127] J. Yee and P. Varaiya. An Analytical Model for Real-Time Multimedia Disk Scheduling. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 276–281, November 1992.

[128] M. Yong, Q.-F. Zhu, and V. Eyuboglu. VBR Transport of CBR-Encoded Video over ATM Networks. In *Proceedings, Packet Video Workshop '94*, pages D18.1–D18.4, 1994.

[129] P. S. Yu, M.-S. Chen, and D. D. Kandlur. Design and Analysis of a Grouped Sweeping Scheme for Multimedia Storage Management. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 38–49, November 1992.

[130] H. Zhang and T. Fisher. Preliminary Measurement of the RMTP/RTIP. In *Proceedings, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 173–184, November 1992.

[131] H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. In *Proceedings, ACM SIGCOMM '91 Conference*, pages 113–121, 1991.

[132] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network Magazine*, pages 8–18, September 1993.

[133] M. Zitterbart. High-Speed Transport Components. *IEEE Network Magazine*, 5(1):54–63, January 1991.