Fast and Efficient Algorithms for Video Compression and Rate Control

Dzung Tien Hoang and Jeffrey Scott Vitter

Draft, June 20, 1998

Vita

Dzung Tien Hoang was born on April 20, 1968 in Nha Trang, Vietnam. He immigrated to the United States of America in 1975 with his parents, Dzuyet D. Hoang and Tien T. Tran, and two sisters. He now has three sisters and one brother. They have been living in Harvey, Louisiana.

After graduating in 1986 from the Louisiana School for Math, Science and the Arts, a public residential high school in Natchitoches, Louisiana, he attended Tulane University in New Orleans with a full-tuition Dean's Honor Scholarship and graduated in 1990 with Bachelor of Science degrees in Electrical Engineering and Computer Science, both with Summa Cum Laude honors.

He joined the Department of Computer Science at Brown University in Providence, Rhode Island, in 1990 under a University Fellowship and later under a National Science Foundation Graduate Fellowship. He received a Master of Science in Computer Science from Brown in 1992 and a Doctor of Philosophy in Computer Science from Brown in 1997. From 1993 to 1996, he was a visiting scholar and a research assistant at Duke University in Durham, North Carolina. From 1991 to 1995, he spent summers working at the Frederick National Cancer Research Facility, the Supercomputing Research Center, and the IBM T. J. Watson Research Center.

In August 1996, he joined Digital Video Systems, in Santa Clara, California, as a Senior Software Engineer. He is currently a Senior Software Systems Engineer at Sony Semiconductor Company of America.

Jeffrey Scott Vitter was born on November 13, 1955 in New Orleans, LA. He received a Bachelor of Science with Highest Honors in Mathematics from the University of Notre Dame in 1977, and a Doctor of Philosophy in Computer Science from Stanford University in 1980. He was on the faculty at Brown University from 1980 until 1993. He is currently the Gilbert, Louis, and Edward Lehrman Professor and Chair of the Department of Computer Science at Duke University, where he joined the faculty in January 1993. He is also Co-Director and a Founding Member of the Center for Geometric Computing at Duke.

Prof. Vitter is a Guggenheim Fellow, an ACM Fellow, an IEEE Fellow, an NSF Presidential Young Investigator, a Fulbright Scholar, and an IBM Faculty Development Awardee. He is coauthor of the book *Design and Analysis of Coalesced Hashing* and is coholder of patents in the areas of external sorting, prediction, and approxi-

mate data structures. He has written numerous articles and has consulted frequently. He serves or has served on the editorial boards of *Algorithmica*, *Communications of the ACM*, *IEEE Transactions on Computers*, *Theory of Computing Systems* (formerly *Mathematical Systems Theory: An International Journal on Mathematical Computing Theory*), and *SIAM Journal on Computing*, and has been a frequent editor of special issues. He serves as Chair of ACM SIGACT and was previously Member-at-Large from 1987–1991 and Vice Chair from 1991–1997. He was on sabbatical in 1986 at the Mathematical Sciences Research Institute in Berkeley, and in 1986–1987 at INRIA in Rocquencourt, France and at Ecole Normale Supérieure in Paris. He is currently an associate member of the Center of Excellence in Space Data and Information Sciences.

His main research interests include the design and mathematical analysis of algorithms and data structures, I/O efficiency and external memory algorithms, data compression, parallel computation, incremental and online algorithms, computational geometry, data mining, machine learning, and order statistics. His work in analysis of algorithms deals with the precise study of the average-case performance of algorithms and data structures under various models of input. Areas of application include sorting, information storage and retrieval, geographic information systems and spatial databases, and random sampling and random variate generation. Prof. Vitter's work on I/O-efficient methods for solving problems involving massive data sets has helped shape the subfield of external memory algorithms, in which disk I/O can be a bottleneck. He is investigating complexity measures and tradeoffs involving the number of parallel disk accesses (I/Os) needed to solve a problem and the amount of time needed to update a solution when the input is changed dynamically. He is actively involved in developing efficient techniques for text, image, and video compression, with applications to GIS, efficient prediction for data mining, and database and systems optimization. Other work deals with machine learning, memory-based learning, and robotics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this book, we investigate the compression of *digital data* that consist of a sequence of symbols chosen from a *finite* alphabet. In order for data compression to be meaningful, we assume that there is a standard representation for the uncompressed data that codes each symbol using the same number of bits. For example, digital video can be represented by a sequence of frames, and each frame is an image composed of pixels, which are typically represented using a binary code of a fixed length. Compression is achieved when the data can be represented with an average length per symbol that is less than that of the standard representation.

Not all forms of information are digital in nature. For example, audio, image, and video exist at some point as waveforms that are continuous both in amplitude and in time. Information of this kind is referred to as *analog* signals. In order to be representable in the digital domain, analog signals must be discretized in both time and amplitude. This process is referred to as *digital sampling*. Digitally sampled data is therefore only an approximation of the original analog signal.

Data compression methods can be classified into two broad categories: *lossless* and *lossy*. As its name suggests, in lossless coding, information is preserved by the compression and subsequent decompression operations. The types of data that are typically compressed losslessly include natural language texts, database files, sensitive medical images, scientific data, and binary executables. Of course, lossless compression techniques can be applied to any type of digital data; however, there is no guarantee that compression will actually be achieved for all cases. Although digitally sampled analog data is inherently lossy, no additional loss is incurred when lossless compression is applied.

On the other hand, lossy coding does not preserve information. In lossy coding, the amount of compression is typically variable and is dependent on the amount of loss that can be tolerated. Lossy coding is typically applied to digitally sampled data or other types of data where some amount of loss can be tolerated. The amount of loss that can be tolerated is dependent to the type of data being compressed, and quantifying tolerable loss is an important research area in itself.

By accepting a modest amount of loss, a much higher level of compression can be achieved with lossy methods than lossless ones. For example, a digital color image can typically be compressed losslessly by a factor of roughly two to four. Lossy techniques can compress the same image by a factor of 20 to 40, with little or no noticeable distortion. For less critical applications, the amount of compression can be increased even further by accepting a higher level of distortion.

To stress the importance of data compression, it should be noted that some applications would not be realizable without data compression. For example, a two-hour movie would require about 149 gigabytes to be stored digitally without compression. The proposed Digital Video Disk (DVD) technology would store the same movie in compressed form using only 4.7 gigabytes on a single-sided optical disk. The efficacy of DVD, therefore, relies on the technology to compress digital video and associated audio with a compression ratio of about 32:1, while still delivering satisfactory fidelity.

A basic idea in data compression is that most information sources of practical interest are not random, but possess some structure. Recognizing and exploiting this structure is a major theme in data compression. The amount of compression that is achievable depends on the amount of redundancy or structure present in the data that can be recognized and exploited. For example, by noting that certain letters or words in English texts appear more frequently than others, we can represent them using fewer bits than the less frequently occurring letters or words. This is exactly the idea behind Morse Code, which represents letters using a varying number of dots and dashes. The recognition and exploitation of statistical properties of a data source are ideas that form the basis for much of lossless data compression.

In lossy coding, there is a direct relationship between the length of an encoding and the amount of loss, or distortion, that is incurred. Redundancy exists when an information source exhibits properties that allow it to be coded with fewer bits with little or no perceived distortion. For example, in coding speech, distortion in high frequency bands is not as perceptible as that in lower frequency bands. As a result, the high frequency bands can be coded with less precision using fewer bits. The nature of redundancy for lossy coding, especially as it relates to video coding, is explored in Chapter 2.

In data compression, there is a natural tradeoff between the speed of a compressor and the level of compression that it can achieve. In order to achieve greater compression, we generally require more complex and time-consuming algorithms. In this manuscript, we examine a range of operational points within the tradeoff possibilities for the application of video compression.

Motion Estimation at Low Bit Rates

In Chapter 3, we explore the speed-compression tradeoffs possible with a range of motion estimation techniques operating within a low-bit-rate video coder that adheres to the H.261 international standard for video coding. At very low rates, hybrid video

coders that employ motion compensation in conjunction with transform coding of the residual typically spends a significant portion of the bandwidth to code the motion information. We focus on motion estimation with hopes of improving the compression performance.

Initially, we construct motion estimation algorithms that explicitly minimize bit rate and a combination of rate and distortion. In coding experiments, these computationally intensive algorithms produce better compression (with comparable quality) compared to the standard motion estimation algorithm, which does not require as much computation. Based on insights gained from the explicit minimization algorithms, we propose a new technique for motion estimation that minimizes a quickly computed heuristic function of rate and distortion. The new technique gives compression efficiency comparable to the computationally intensive explicit-minimization algorithms while running almost as fast as the standard algorithm.

In Chapter 4, the bit-minimization philosophy is further applied to a non-standard quadtree-based video coder that codes motion information hierarchically using variable-sized blocks. The motivation is to explore the limits of bit-minimization when applied to an efficient scheme for coding motion vectors. By designing the quadtree encoding so that a subtree is coded independently of other disjoint subtrees, we are able to compute motion vectors that globally minimize the total bit rate using a dynamic programming algorithm. Experimental results confirm that the quadtree-based coder gives additional gains over the H.261-based coders.

Optimal Rate Control

In Chapters 5 through 11, we focus our attention on optimal rate control algorithms for video coders. Existing optimal rate control techniques typically regulate the coding rate to minimize a sum-distortion measure. While these techniques can leverage the wealth of tools from least-mean-square optimization theory, they do not guarantee constant-quality video, an objective often mentioned in the literature. We propose a framework that casts rate control as a resource allocation problem with continuous variables, non-linear constraints, and a novel lexicographic optimality criterion that is motivated for uniform video quality. With this framework, we redefine the concept of coding efficiency to better reflect the constancy in quality that is generally desired from a video coder.

Rigorous analysis within this framework reveals a set of necessary and sufficient conditions for optimality for coding at both constant and variable bit rates. With these conditions, we are able to construct polynomial-time algorithms for optimal rate control. Experimental implementations of these algorithms confirm the theoretical analysis and produce encodings that are more uniform in quality than that achieved with existing rate control methods. As evidence of the generality and flexibility of the framework, we show how to extend the framework to allocate bits among multiple variable-bit-rate bitstreams that are to be transmitted over a common constant-bit-

rate channel and to encompass the case of discrete variables.

Chapter 2

Introduction to Video Compression

In this chapter, we present an introduction to aspects of video compression that will be useful for understanding the later chapters. We begin by describing the generation and representation of digital video. With standard representations defined, we then motivate video compression with several illustrative examples that underscore the need for lossy compression. To better understand the tradeoffs inherent in lossy coding systems, an introduction to rate-distortion theory and practice is presented. Next, we describe existing international standards for video coding and present an overview of the fundamentals of these standards. This chapter is by no means intended to be comprehensive; for an in-depth introduction to video coding fundamentals, the reader is referred to [2, 26, 57, 59].

2.1 Digital Video Representation

Video belongs to a class of information called *continuous media*. Continuous media is characterized by the essentially continuous manner in which the information is presented.[1] This is in contrast to *discrete media*, in which there is no essential continuous temporal component. Text, images, and graphics are examples of discrete media, while movies, sound, and computer animation are examples of continuous media. Even though a slide show is a time-based presentation of images, it is not a continuous medium since each image is viewed as an individual item instead of a part of a bigger entity. On the other hand, a video clip, while also consisting of a sequence of images, is a continuous medium since each image is perceived in the context of past and future images.

For compression to be meaningful, a standard representation should be defined for the data to be compressed. In this section, we give an overview of some of the more popular standard representations for digital video that are in use today.

[1] The information may be discrete in representation, but it should be presented to give an *illusion* of continuity.

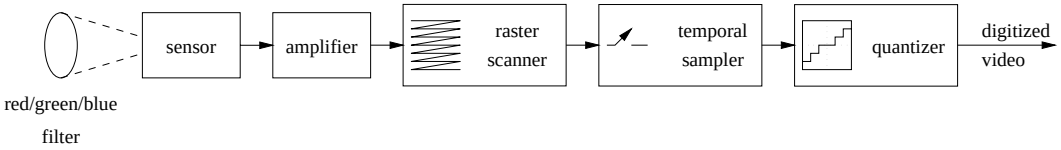


Figure 2.1: Block diagram of a video digitizer.

2.1.1 Color Representation

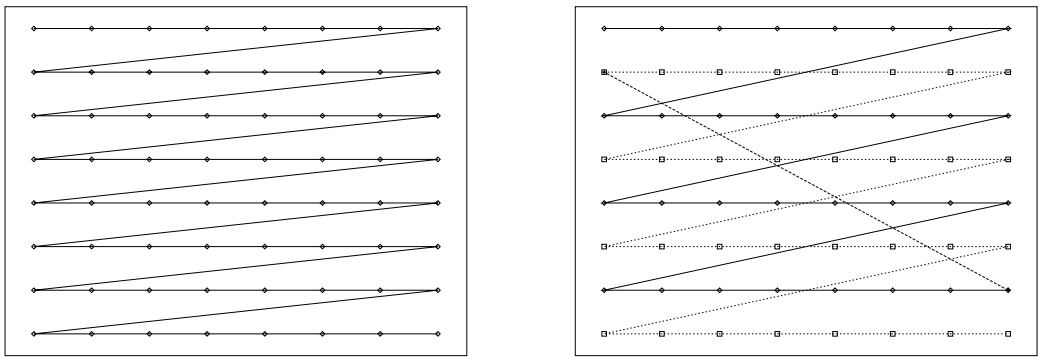
Excluding synthetic (computer-generated) sources, video originates in the physical world. In a general sense, video can be characterized as a time-varying, two-dimensional mix of electromagnetic signals. Being too general, this characterization is not practical for representing visual information relevant to human observers. Although visible light consists of a continuum of wavelengths, it has been known for several centuries that a small set of primary colors, mixed in the right proportions, can simulate any perceived color. In painting, for example, one system of primary colors is cyan, magenta, and yellow; this is a *subtractive* system since the absence of all primary colors yields the color white. Red, green, and blue light sources form another set of primary colors; this is an *additive* system since the presence of all the primary colors at their maximum intensities results in the perception of the color white. This phenomenon of color perception is caused by the way that the human eye detects and processes light, which makes it possible to represent a visual image as a set of three intensity signals in two spatial dimensions.

2.1.2 Digitization

In order to be processed by computers, analog video that is captured by a light sensor must first be digitized. Digitization of video consists of three steps: 1) spatial sampling, 2) temporal sampling, and 3) quantization. A block diagram of the digitization process is depicted in Figure 2.1 for one color component. The steps need not be performed in the order indicated and some steps can even be combined into one operation.

2.1.2a Spatial Sampling

Spatial sampling consists of taking measurements of the underlying analog signal at a finite set of sampling points in a finite viewing area (or *frame*). To simply the process, the sampling points are restricted to lie on a lattice, usually a rectangular grid, say of size $N \times M$. The two dimensional set of sampling points are transformed into a one-dimensional set through a process called *raster scanning*. The two main ways to perform raster scanning are shown in Figure 2.2: *progressive* and *interlaced*. In a progressive (or non-interlaced) scan, the sampling points are scanned from left to right and top to bottom. In an interlaced scan, the points are divided into odd and



(a) Progressive Scan

(b) Interlaced Scan

Figure 2.2: Scanning techniques for spatial sampling of a video image.

even scan lines. The odd lines are scanned first from left to right and top to bottom. Then the even lines are scanned. The odd (respectively, even) scan lines make up a field. In an interlaced scan, two fields make up a frame. It is important to note that the odd and even fields are sampled and displayed at different time instances. Therefore the time interval between fields in an interlaced scan is half of that between frames. Interlaced scanning is commonly used for television signals and progressive scanning is typically used for film and computer displays.

2.1.2b Temporal Sampling

The human visual system is relatively slow in responding to temporal changes. By taking at least 16 samples per second at each grid point, an illusion of motion is maintained. This observation is the basis for motion picture technology, which typically performs temporal sampling at a rate of 24 frames/sec. For television, sampling rates of 25 and 30 frames/sec are commonly used. (With interlaced scanning the number of fields per second is twice the number of frames per second.)

2.1.2c Quantization

After spatial and temporal sampling, the video signal consists of a sequence of continuous intensity values. The continuous intensity values are incompatible with digital processing, and one more step is needed before this information can be processed by a digital computer. The continuous intensity values are converted to a discrete set of values in a process called *quantization* (or *discretization*.

Quantization can be viewed as a mapping from a continuous domain to a discrete

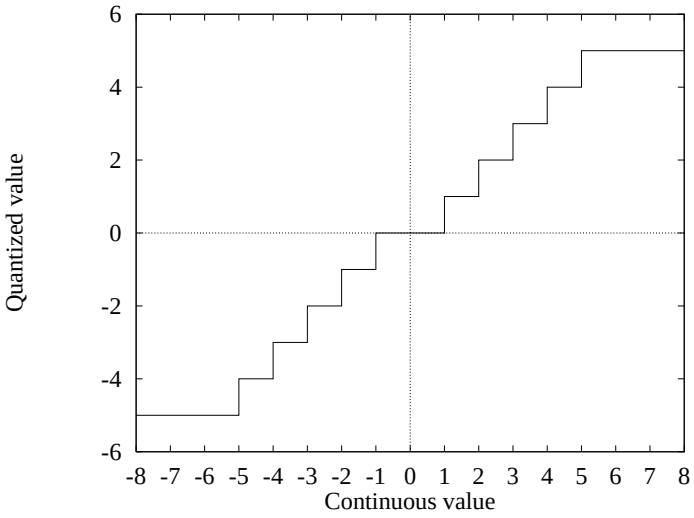


Figure 2.3: Example of uniform quantization.

range.[2] A particular quantization mapping is called a *quantizer*. An example is shown in Figure 2.3. In the figure, there are eleven discrete quantization levels, also called *bins*. Each bin has an associated size, which is the extent of the continuous values that map to that bin. In the example, each bin, except for the bins for -5, 0, and 5, has the same size, which is sometimes referred to as the *quantizer step size*. This type of quantizer is called a *uniform* quantizer.

A binary encoding can be assigned to each of the bins. Typically the initial quantization of a continuous source is done using a number of quantization levels that is a power of 2, so that a fixed number of bits can be used to represent the quantized value.[3] This process of representing a continuous value by a finite number of levels using a binary code is often referred to as *pulse code modulation* (PCM). Thus, after spatial sampling, temporal sampling, and quantization, we have $N \times M$ data points, commonly called *pixels* or *pels*, represented using a fixed number of bits.

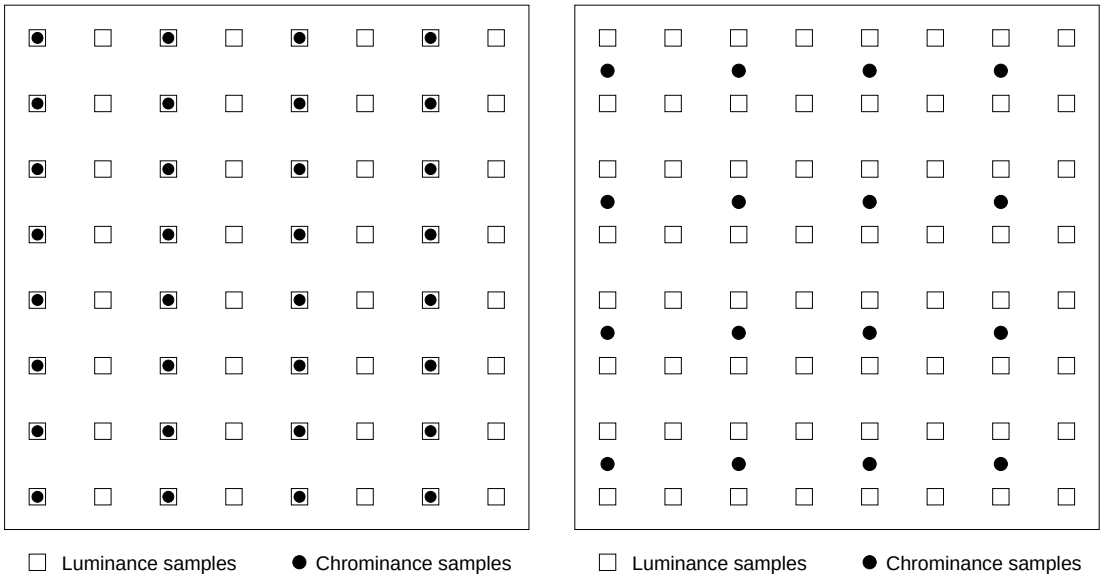
2.1.3 Standard Video Data Formats

To promote the interchange of digital video data, several formats for representing video data have been standardized. We now review some of the more popular standard representations.

The CCIR-601 [4] format for video frames specifies spatial sampling of 720×480 and temporal sampling at 30 frames/sec for NTSC (U.S. and Japan) television systems and 720×576 at 25 frames/sec for PAL (Europe) television systems. Color is

[2]This definition is intended also to encompass mappings from a discrete domain to a discrete range.

[3]Further quantization of digitized data may use a number of quantization levels that is not a power of 2 and employ variable-length entropy coding.



(a) 4:2:2 subsampling (b) 4:2:0 subsampling

Figure 2.4: Color subsampling formats, as specified in the MPEG-2 standard.

represented using three components: a luminance (Y) component and two chrominance components (Cb and Cr). The luminance component encodes the brightness or intensity of each pixel and the chrominance components encode the color values.[4] Each component is quantized linearly using eight bits. For NTSC (respectively, PAL), there are 720×480 (720×576) luminance values, one for each pixel, and 360×480 (360×576) values for each chrominance component. The chrominance components are subsampled horizontally with respect to the luminance component to take advantage of reduced human sensitivity to color. This subsampling process is referred to as the 4:2:2 format and is depicted in Figure 2.4(a).

The Source Input Format (SIF) specifies spatial sampling of 360×240 (respectively, 360×288) and temporal sampling at 30 (25) frames/sec for NTSC (PAL) television systems. As with CCIR-601, color is represented using three components: Y, Cb, and Cr. Each component is quantized linearly using eight bits. For NTSC (respectively, PAL), there are 360×240 (360×288) luminance values, one for each pixel, and 180×120 (180×144) values for each chrominance component. This subsampling format is referred to as the 4:2:0 format[5] and is depicted in Figure 2.4(b).

One drawback with the CCIR-601 and SIF formats is that they specify different

[4]The Y-Cb-Cr color space is related to the red-green-blue (RGB) color space by a matrix multiplication.

[5]This should not be confused with the older 4:1:1 format in which the chrominance components are subsampled by a factor of 4 only in the horizontal direction.