Chapter 1

Introduction and Motivation

Contents

1.1 Digital communication systems and compression

Advances in the speed and degree of integration of digital circuits have led to a steady increase in the popularity of digital communications. Originally devoted to the transmission of data, which has relatively lax timing requirements, digital communication systems are increasingly used for transmission of real time signals as well, for which there typically exists some constraint on the delay between encoding and decoding. Digital technology has been widely used in telephone networks for years, (though normally within the network rather than in the subscriber loop) and the introduction

of the compact disc (CD) has shown that digital technologies can also spawn very successful consumer products. More recently, motivated by the advantages of digital technology for reliable transmission and efficient storage, and by decreasing costs of available Very Large Scale Integration (VLSI) technology, digital transmission of images and video has become increasingly popular.

Efficiency of transmission and storage make it very important to use compression of the digitized signals, that is, to use a procedure to reduce the amount of information needed to reproduce the signal at the decoder. While, arguably, transmission and storage capacities will be available more cheaply in the future (through improvements in fiber optic communications and high speed digital switching), it is also true that the increase in demand for digital communication will still make it necessary to compress the signals to be transmitted. Additionally, the continuing decline in the prices of memory and the circuitry required for compression will make it even cheaper to use increasingly complex algorithms for compression.

Thus, since both storage and transmission resources tend to be scarce, an efficient digital communications system must resort to some sort of compression scheme. In this thesis we will consider several issues concerning compression for images and video. We will concentrate on lossy compression, i.e. where the decoded signal is an approximation to the original signal fed into the encoder. While lossy compression would obviously be unacceptable for data communication, signal transmission is different in that judiciously administered "losses" to the input signal may be acceptable. In this context, acceptable losses are those that a user would not be able to detect (in the case of low compression systems) or those the user would not object to, even if they are perceptible (for lower quality, higher compression systems.)

The interest in compression for both images and video has been shown in international standardization efforts, such as JPEG [112] for image compression and H.261

[61] MPEG-1 [58] and MPEG-2 for video compression at different target rates. As the availability of standards has permitted affordable hardware compression it has also multiplied the number of applications for which digital image and video are proposed. Computers equipped with JPEG compression are becoming commonplace, digital transmission of video is already used in videophone and videconference applications [61], it will soon become reality in satellite broadcast and has also been selected as the delivery mechanism for future High Definition TV (HDTV) systems.

To study the performance of image and video compression techniques one has thus to consider two parameters: the achieved degree of compression and the degradation in the perceived quality due to the lossy scheme. Since image and video may be used in a real time communication environment, we also consider throughout this thesis the role of delay in judging the performance of such systems, taking into account the end-to-end transmission delays as well as the encoding delays.

1.2 Performance of image and video encoders

Consider a generic communications system (refer to Fig. 1-1) composed of five elements, a signal source, an encoder, a communication channel, a decoder and a display or output device. The signal source generates at fixed intervals one block or vector chosen among a (continuous or discrete) set $\mathcal{S}$ of possible blocks (these blocks are groups of samples of the analog signal that is being transmitted). Call $X_1, \cdots, X_i, \cdots, X_N$ the blocks in the sequence, where X_i will typically denote the current block. As blocks, depending on the application, we consider anything from single samples up to video frames. Also, in some cases, the numbering of the blocks will correspond to the temporal ordering (e.g. consecutive frames in a video sequence) while in other cases all the blocks may be available simultaneously and the numbering would correspond to some other ordering (e.g. spatial ordering of the blocks in

a block–based image coding scheme). The encoder maps each of the source blocks X_i into a reproduction block $\hat{X}_i$ chosen from a finite codebook $\mathcal{C} \subset \mathcal{S}$ and transmits the index corresponding to the chosen codeword to the receiver through the communication channel. The decoder maps the index to the reproduction codeword $\hat{X}_i$, which is then presented in the display or output device. Note that we use the terms codebook and index, commonly employed in the vector quantization (VQ) literature, as a conceptual tool, without implying that the encoding procedures or the codebook design are necessarily similar to those used in VQ.

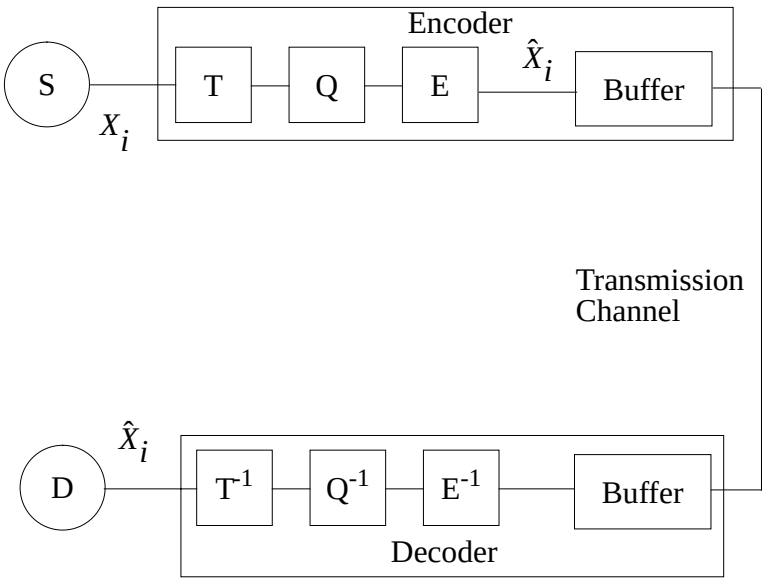

Figure 1-1: Block diagram of a generic communications system. The source outputs blocks that are encoded and transmitted. Note that the diagram depicts an encoder formed by a transform (T), quantizer (Q) and entropy coder (E), as well as an output buffer. However this is not the only configuration we will consider.

A good way of studying the performance of such a system is to resort to a rate-distortion (R-D) framework [99, 8, 24]. The encoder uses a certain rate, or number of bits, to transmit each of the blocks in $\mathcal{C}$ while the channel can transmit only a limited number of bits per unit time. For the given channel rate constraints, we can

determine the performance by measuring the distortion between the original block produced by the source and the corresponding output block at the receiver end. Such a distortion should be linked to how acceptable to the end user the decoded signal is. Then we can concentrate on the problem of minimizing the distortion achieved at a certain rate. We will look in more detail at the exact optimization framework and the cost functions in Section 1.4.

Let us now consider a generic encoder in more detail. The encoder is uniquely determined by two elements: the codebook (including both the available reproduction blocks in $\mathcal{C}$ and the corresponding codewords) and the encoding rule (which determines what codeword should be assigned to a given input block). Typically, the codebook is designed first, and optimized based upon either models, or, more likely, a training source, that are assumed to be representative of the expected sources. Although a fully adaptive scheme would potentially update the codebook as the source changes [39, 40, 17] most systems operate with a fixed pre-designed codebook and rely on the encoding rule to provide all the adaptivity.

1.2.1 Codebook design

We now explain some basic ideas regarding typical codebook design. Our aim will be to review some of the more basic principles to outline the assumptions and trade-offs that are required to implement practical codebooks. Since in the rest of this thesis (except Chapter 6) we will assume a generic codebook, this section does not intend to be exhaustive. Detailed treatments of codebook design techniques can be found in [39, 50].

1.2.1.1 Scalar quantization and entropy coding

The simplest case is that of scalar quantization, where each of the samples generated by the input source is quantized separately. Then the set of input blocks $\mathcal{S}$ will be some subset of $\mathcal{R}$, the set of real numbers. The codebook design problem consists in selecting a finite number, say L, of values $r_i \in \mathcal{R}$ to be used as reproduction levels, as well as the corresponding codewords to be transmitted, c_i. The encoding algorithm will be such that $\hat{X}_i = r_j$, where r_j is the reproduction level closest to X_i. Under the assumption of a stationary source with known statistics well known design techniques can be used to determine the r_i, c_i pairs. For instance, the Lloyd-Max quantizer design technique provides optimal, i.e. distortion minimizing, r_i's for fixed length c_i's (see Chapter 6 and [39, 50]). The model can be either explicit, through knowledge of the probability density function (pdf) of the source, or implicit, through the choice of a specific training sequence or set of blocks as being characteristic of the source.

If $\mathcal{S}$ is a finite set (as is the case for example with image pixels) then one can use *entropy coding* techniques to minimize the expected average number of bits needed to encode the source without distortion, i.e. where $\hat{X}_i = X_i$. This would be an example of *lossless* compression. Calling $s_1, \ldots, s_K$ the elements of $\mathcal{S}$, assume their probabilities $p(s_1), \cdots, p(s_K)$ are known. Then the first order entropy of the source in bits is:

$$H = -\sum_{k=1}^{K} p(s_k) \log_2(p(s_k)). \tag{1.1}$$

The entropy dictates a lower bound in the number of bits per input sample needed to transmit sequences generated by the source assuming that the source follows the model and that the blocks are to be coded individually. Well known techniques such as Huffman and arithmetic coding can provide performance close to the entropy. Note that given a continuous set $\mathcal{S}$ and a set of reproduction levels r_i one can measure the

expected probability of each of the r_i's and then use entropy coding to determine the c_i's. The resulting c_i's will have variable length.

The main point is to note that both scalar quantizer and entropy coder design rely on having available a model of the input source. In cases where the source is known to be non-stationary one can resort to techniques that adapt the codebook to the changing statistics. See Chapter 6 for a review of adaptive codebook techniques for both lossless and lossy compression.

1.2.1.2 Vector quantization

When we choose blocks X_i which contain n samples, i.e. for $\mathcal{S}$ a subset of $\mathcal{R}^n$, we are considering the more general Vector Quantization (VQ) codebook design problem. Similar principles, although without as simple an optimization procedure, can be defined for vector quantization schemes. See [39] for an excellent review of VQ techniques.

Note that, as the dimensions of the vector increase, it becomes increasingly unrealistic to obtain explicit models of the source, and therefore VQ techniques rely on training sequences to design the codebooks. Obviously the codebook obtained through training will do a good job of representing the blocks *within* the training set. However it is by no means clear what constitutes a "good" training set, as measured by how well the generated codebook performs for blocks *outside* the training set. Also, increasing the dimensions of the vector increases the complexity of both the codebook design and the encoding procedure.

For the above reasons VQ schemes of interest typically rely on relatively small values of n. Given the complexity of generating a codebook, typical schemes maintain a fixed one, though recent work has also explored different strategies for updating the codebook on the fly while encoding the source [40, 17, 14, 15, 23, 39].

1.2.1.3 Codebook design for image and video sources

For image and video applications, it would be possible, at least conceptually, to consider images or frames as being "blocks" in the sense introduced above, so that a codebook would be composed of full image size codewords, i.e. each of the codewords would have the size of an image (see [69] for an example where this model is used as an analysis tool). Even though all image coding schemes effectively do map each image into one among a discrete, albeit huge, set of possible codewords, it would be clearly impractical to use such an idea, due to the complexity of both to the codebook design and the encoding algorithm. Thus all practical image/video coding algorithms rely on the "divide and conquer" approach to make the codebook design and the encoding algorithm realizable. Discrete cosine transform (DCT) based coding [68] VQ [39] are examples of algorithms that decompose the input signal into sub-blocks, which are independently quantized (sub-block size codebooks can then be designed). Similarly, subband and wavelet coders produce several sets of samples (each of the bands) which can then be quantized independently.

We now briefly outline a typical codebook design technique for a linear transform DCT-based coder. An image is first decomposed into blocks on which the DCT is computed. Then the resulting set of coefficients (in the "transform domain") is quantized. While there is no theoretical justification to encoding independently the transform coefficients (or the subbands), in practice this approach is used under the generally correct assumption that after decorrelation, achieved by using the DCT, the gain to be expected by using vector quantization would be limited. Therefore one can treat the set of all coefficients corresponding to the same frequency as a single source, for which a quantizer has to be designed. The problem of determining how coarsely to quantize each of the coefficients is again one where models are typically used. For instance, in the case of DCT methods, one estimates the variance of each of the

coefficients based on an image or set of images, then bit allocation techniques are used to determine the coarseness of quantization to apply to each of the coefficients [50]. This results in a set of factors (usually denoted quantization matrix) which specify the relative coarseness of quantization for each of the coefficients. For simplicity of implementation uniform quantizers are normally used and then codewords for each of the possible quantized values are found in order to minimize the entropy, using variations of Huffman coding. Also, it should be pointed out that inter-coefficient correlation is exploited to some extent by using techniques such as zig-zag scanning and run-length coding [112]. Examples of this design technique can be found in [1, 112, 58].

Furthermore, to make the design of these systems more versatile an additional parameter, sometimes called quantization parameter or QP is also available. This parameter determines the coarseness of the quantization for a given block (as the MQUANT parameter in MPEG) or for all the blocks in an image (as in JPEG). While the quantization matrix determines the relative coarseness of quantization for each of the coefficients, the QP parameter scales the quantizer steps *equally* for all the coefficients in a block, while preserving the relative degrees of coarseness determined by the quantization matrix. Thus, if QP can be changed on block by block basis, it allows the encoder to assign different levels of quantization coarseness to each block. Typically, increasing QP results in higher compression at the cost of higher distortion. This parameter enables using the same coding scheme to achieve several operational R-D operating points for any given block and would be equivalent to having a set of M codebooks $\{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$ each corresponding to a given QP (the role of the QP is thus analogous to that of the gain in gain shape VQ schemes[39]). Therefore to transmit the codeword corresponding to a given input block it will be necessary to first communicate to the decoder which codebook was selected and then transmit the

index of the appropriate codeword. Thus the QP value can be seen as an overhead information.

For convenience we have been referring to codebooks for blocks even when considering transform coding schemes, thus ignoring the exact quantization process which entails computation of the transform and then quantization of the coefficients. Obviously both the codebook design and the mapping procedure are not arbitrary and the techniques used are specific to the transform coding environment.

Design for subband coders [114, 52] resorts to similar techniques where quantizers are designed a priori and trained on some representative set of images. Similar training approaches have also been used for wavelet based coding schemes [4]. Only more recent work [59, 100] has achieved a higher degree of adaptivity by exploiting dependencies between bands while not requiring training [100].

Finally we should note that choosing the codelengths for the different vectors in block-based motion compensated video coding also resorts to training. In this case, a first phase in the design process determines which vectors turn out to be more "popular" and the relative frequencies of occurrence are then used to compute an entropy code for the vectors.

1.2.2 Encoding algorithms and adaptivity

Note that a common thread through all of the abovementioned schemes is to operate on smaller coding units (blocks in DCT and VQ, bands in subband and wavelet coding, vectors in motion compensation) and define a codebook for such units. The main motivations for this approach are the reduced complexity of small-block approaches, as well as the difficulty of achieving good models for larger blocksizes. In most cases the codebook does not change after the design stage. Assume now that the codebook is fixed and our aim is to achieve some adaptivity.

We will consider two cases according to whether the encoder chooses the codeword corresponding to a block from a single codebook or from one among several possible codebooks. The latter is the most attractive alternative to implement adaptive encoding algorithms, but good results can be obtained using the former approach as well. As an example, so-called thresholding algorithms perform an adaptive encoding within a single codebook. For instance in [89, 88] thresholding is used to remove, in a R-D optimal way, coefficients after having quantized all the image using a single QP. Thus, for the codebook determined by QP, codewords are transmitted which are not necessarily the nearest, for our distortion measure, to the source block. A similar approach has been proposed to improve the performance of a wavelet–based encoder in [118].

Obviously, the simplest encoding algorithm is that which maps each of the input blocks into a codeword regardless of the context. In other words, each block symbol will be considered independently of the others and will be mapped to the nearest codeword (in terms of the distortion measure that is used). If several codebooks are used then the choices of codebook will also be made independently, so that for instance one could use the same $\mathcal{C}_i$ for all blocks in the source or choose a $\mathcal{C}_i$ for each block based on some other factor, e.g. the buffer fullness in a typical buffer control scheme. This approach has the advantage of not requiring encoding delay. In the example of DCT-based coding this solution would call for fixing one QP for all blocks in the image so that they can all be coded independently (this is the case in JPEG, for instance).

Roughly speaking, we can define adaptivity as the ability to change the choice of codeword for a given block *depending on the context.* More formally, if X_i is the current block and $X_i^* = f(X_i)$ is the codeword nearest to it within the codebook (in the sense of the distortion measure that has been selected), then an adaptive

encoding algorithm selects $\hat{X}_i$, which might be different from X_i^*. We will denote $R(\hat{X}_i)$ and $D(\hat{X}_i)$, the rate and distortion, respectively for the given codeword $\hat{X}_i$, where $R(\hat{X}_i)$ includes any possible overhead needed to specify the choice of codebook $\mathcal{C}_j$. Adaptivity can be summarized by writing the encoding algorithm for block i as

$$\hat{X}_i = f(\hat{X}_1, \ldots, \hat{X}_{i-1}, X_1, \ldots, X_{i-1}, X_i, \ldots, X_N). \tag{1.2}$$

If we had

$$\hat{X}_i = f(\hat{X}_1, \ldots, \hat{X}_{i-1}, X_i). \tag{1.3}$$

we would be considering *backward adaptation*. In some cases, if multiple codebooks are used, the information of which codebook is to be used would not necessarily have to be sent as overhead, since both encoder and decoder have access to the information needed to adapt the encoding rule (except for X_i). A predictive scheme would be an example of this type of adaptation, since past quantized blocks are used to predict the current one and the predictor defines the codebook to be used. In the context of rate control, memoryless schemes (where the buffer state is fedback to choose the next codebook) would not require overhead, while schemes where the incoming block X_i is analyzed before the decision would require overhead.

Conversely, if the encoding rule was such that

$$\hat{X}_i = f(X_i, \ldots, X_{i+D}). \tag{1.4}$$

we would have an example of *forward adaptation*. Again, if multiple codebooks are used, the encoder will have to rely on sending its choice of codebook to the decoder, since the decision will be based on information available only at the encoder. Note that in this example the *encoding delay* would be D blocks, since the encoder has to know the next D blocks in order to quantize block i.

Adaptivity has advantages, which will be described in what follows, but also increases the complexity of the encoders and/or decoders. Forward adaptation calls for an encoder more complex than the decoder, but requires that overhead information be sent. For example, the encoder in an JPEG scheme can specify different quantization tables (and thus different codebooks) for different images but would have to first optimize the table for a given image and then transmit it along with the image data to the decoder. Conversely, backward adaptation may not require overhead information to be sent but resorts to similar complexity in encoder and decoder, which may not be desirable in some applications. It is also less robust to channel errors, since future blocks will be coded based on past decoded ones and these may have been corrupted by transmission errors.

1.3 Adaptivity, bit allocation and delay constraints

We have thus far explained how adaptivity could be achieved via either update of the codebook or through the encoding algorithm. We now motivate why adaptivity is desirable in the first place. Parts of this thesis will be devoted to considering the case where a discrete set of codebooks is fixed (Chapters 2, 3, 4). Another chapter will deal with an adaptation scheme that acts upon a set of images (Chapter 5) and we will finally show a novel example of backward adaptive quantization (Chapter 6).

We now list some of the main motivations to set up adaptive quantization schemes. Our goal is to present situations where using the more complex encoding rules of (1.3) and 1.4) can be advantageous.

Rate constraints. A typical situation calls for compressing a set of blocks (e.g. an image) using a rate close to some target rate budget. In such cases, choosing $\hat{X}_i$ for each of the blocks of the source is a bit allocation problem where the resulting

decoded quality should be maximized for the given rate. Fixed quantization basically would assign the same relevance to all types of blocks. Because some blocks can withstand higher objective error without affecting the perceptual quality, it may be wise to reduce their rate allocation (i.e. operate with a poorer quality codebook) and use those bits for other blocks. Therefore, achieving a globally good performance under a rate constraint involves considering all blocks and thus being adaptive (see Chapter 2).

Buffer constraints. When transmitting the variable size codeword indices through a constant rate channel where the encoder and decoder are attached to synchronous source and display devices, the rate will have to be adjusted so that excessive end to end delay does not cause loss of information. Equivalently, maintaining a constant end to end delay can be seen as having finite buffers at the encoder and decoder as in Fig. 1-1. This is a special case of the previous global rate constraint where now local constraints on the rate have to be enforced as well. The problem of rate control is thus also one of adaptive encoding where, as will be seen, a short encoding delay may be sufficient to obtain good performance (see Chapter 2).

Memory in the encoder (dependent allocation). If we are generating predictors with some of the previously quantized blocks (e.g. DPCM, Motion-compensated DCT, etc) then achieving optimality would require adaptivity. Indeed, making a quantization choice for the current block as in (1.3) may turn out to be suboptimal in the long run since the current block is used as a predictor for future ones (see Chapter 4).

Inadequacy of the training set or the model with respect to the actual source. Obviously if there is a mismatch between the expected and actual sources

a scheme that is capable of adapting should be better (see Chapter 6 for an example of codebook adaptation). Note that the case of image and video is particular in that there are models available *only* for the coding units (e.g. DCT blocks, subbands) but not for the images as a whole, which would be too complex to model. Therefore the need for blockwise bit allocation can be seen as an adaptation to the specific input image.

Other constraints. Finally it may be possible that other constraints operating directly or indirectly on the rate may determine the use of an adaptive point. For example, some situations may require a rate control that meets certain requirements due to system performance issues, rather than purely source coding performance (see Chapter 3). Also, multiresolution (MR) schemes present a different problem in that they require that a codeword is chosen for each of the resolutions of the signal source. While choosing the codewords based on R-D criteria may be one of the objectives [85, 86], the relative rate of the different resolutions can also be chosen based on other constraints. For instance, in a multiresolution image database browsing scheme, our aim can be to operate with a minimum query delay. The problem is also one of adaptation since we would allocate the rates in a different manner depending on the characteristics of the image set to be queried (see Chapter 5).

1.4 Optimization framework

Traditional rate-distortion theory provides bounds on the performance for certain classes of sources and encoders. For a stationary source, characterized by its probability density function (pdf), the rate-distortion function will give a lower bound on the distortion that is achievable with a given rate. However, rate distortion theory does not specify how to construct encoders to achieve results close to those bounds.

Parts of this thesis (Chapters 2, 3, 4) will be concerned with studying and optimizing the performance of *actual encoders*, operating on sources for which we make no assumptions on stationarity. We therefore depart from the traditional R-D framework in two ways (refer to Fig. 1-2):

1. We use in our optimization actual values of rate and distortion as in the operational rate-distortion framework of [22], i.e. for every source block we measure the distortion and the length of the resulting codeword. Instead of looking for the expected values of rate and distortion, as in a model based approach relying on stationarity, we find time averages and our objective is to minimize these averages for the known source. Thus in Fig. 1-2 the available points marked with the crosses were directly measured on the source.

2. To achieve adaptivity we consider that the encoder can choose for each symbol a reproduction vector from one of the discrete set of M codebooks $(\mathcal{C}_1, \cdots, \mathcal{C}_M)$. Thus we assume that we are given a discrete set of admissible quantizers to operate with, as in [101], and our problem will be to choose the "best one" for every block in our source. Thus in Fig. 1-2 we observe a finite set of available operating points.

1.4.1 Defining a cost function

We have so far mentioned the existence of a distortion function without detailing the type of function that we expect to use. Due to its convenience and wide usage the mean square error (MSE) measure is very frequently chosen. If blocks of size n are considered and the $X_i(j)$ are the components of a block then $MSE = (1/n)\sum(\hat{X}_i(j) - X_i(j))^2$. Our results are valid for MSE but also for more general distortion measures, namely, those which can be computed *individually* on each block,

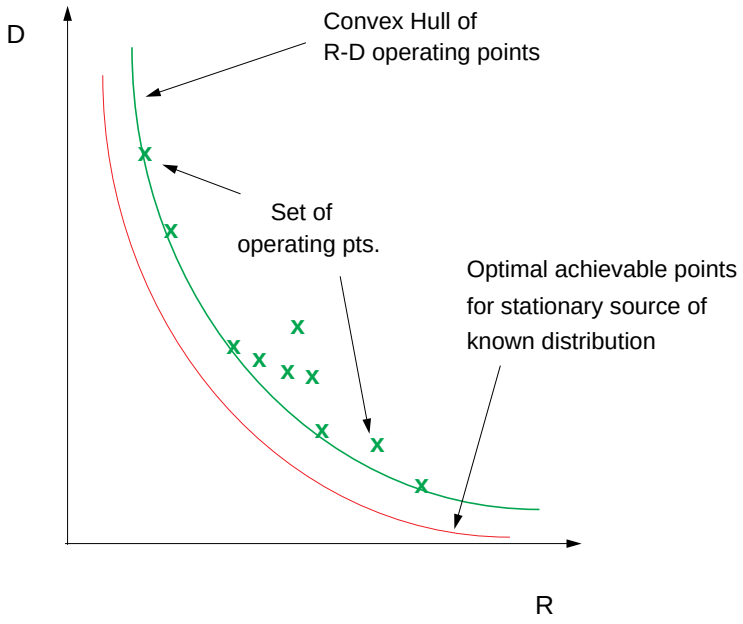

Figure 1-2: Operational vs. traditional rate distortion formulation. The traditional R-D formulation assumes a model for the source and is concerned with obtaining bounds on achievable performance, without providing constructive methods to approach those bounds. In an operational R-D framework the optimization has to find good actual operating points among the discrete set of those available (depicted as in individual points in the figure).

i.e. where $D = D(X_i, \hat{X}_i)$. Additionally, by aiming to minimize the average distortion we are implicitly assuming that the average distortion for a set of blocks is a good measure of the overall reconstruction quality of the set.

It is well known that MSE may not be correlated to the perceptual quality of the decoded images, so that images with similar MSE may have widely different perceptual quality. Also, it is questionable that averaging a distortion measure such as MSE is meaningful. For instance, can a very poor quality block within an image be "compensated" by the fact that adjacent blocks have good quality? To face the shortcomings of the MSE measure while still considering objective cost functions we

propose the following two modifications, which both produce distortion measures that *can be used within our framework.*

Weighted MSE. This is a well known technique to improve the correlation between the objective measure and the subjective quality. The idea is to weigh the MSE according to the type of block/image under consideration. For example, in a block-based DCT scheme one could weigh the MSE by the "activity" of the block as measured, for example, by the energy of the transform domain coefficients. By doing this, blocks with higher activity, i.e. more high frequency content, could be quantized more coarsely without affecting as much the subjective quality as in the case of low activity blocks. Note that in designing the quantization one is already implicitly using a frequency weighted MSE approach, since higher frequency coefficients tend to be more coarsely quantized. Furthermore, if relative weights derived from Human Visual System (HVS) factors can be attached to the different frequency coefficients or subbands, one can use a frequency weighted MSE in the bit allocation process. See [48] for a comprehensive review of perceptual factors in signal compression.

Thresholded MSE. Optimization of distortion across the blocks in the source may result in a misleading averaging of high quality and low quality blocks. To avoid this problem, we resort to setting thresholds on the MSE. For instance, assume that for a certain block i and codebook j we achieve a quality such that the next higher quality codebook $j-1$ provides a decoded block which is visually indistinguishable from that obtained with j. Then, we should set to zero the MSE corresponding to using either for that block. In this way, the optimization techniques will avoid increasing the quantization quality beyond the point where it no longer makes a difference. Similarly, if a certain quality level was deemed unacceptable for some block and a given application, we could set the cost of choosing the corresponding codebook,

and those of worse quality, to infinity. In this way we are in effect eliminating that codebook from the set of admissible quantizers for the block in question.

While it may not be straightforward to set up the abovementioned weights and the thresholds it is worth noting that they can help provide an objective measure closer to the perceptual quality criteria, while being readily usable within our optimization framework.

1.5 Problem formulation and complexity

1.5.1 Formulation

Assume we are given N blocks to be quantized, and that we have M possible operating points for each of the blocks. The rate and distortion for each of the blocks and each of the operating points are known also. Then our problem will be to find the encoding rule f to

$$\min \sum_{i=1}^{N} D(\hat{X}_i, X_i) \tag{1.5}$$

and such that for every i we have:

$$\hat{X}_i = f(\hat{X}_1, \ldots, \hat{X}_{i-1}, X_1, \ldots, X_{i-1}, X_i, \ldots, X_N).$$

while possibly meeting some set of additional constraints on the rates $R(\hat{X}_1, \ldots, \hat{X}_N)$.

Note that we have written f in the more general form as in (1.2) but we will restrict ourselves variously to forward adaptation as in (1.4) in Chapters 2, 3 and 4 or to backward adaptation as in (1.3) in Chapter 6. Also it should be emphasized that our objective is *the encoding rule*: our goal will be to obtain an algorithm f that will choose $\hat{X}_i$ based on the available information, for any source. We denote f as a function for convenience, although it should be clear that it represents an algorithm.

In the literature, two approaches have been used to deal with bit allocation

problems: an analytical model–based (continuous) framework and an operational rate–distortion based (discrete) framework. The model–based approach assumes various input distributions and quantizer characteristics from Gaussian inputs [47] to negative–exponential distributed quantizers [50, 97]. With this approach, one can sometimes get closed–form solutions based on the assumed models, using continuous optimization theory. Alternatively, bit allocation for completely arbitrary inputs and discrete quantizer sets has also been addressed in [101] in the operational rate–distortion framework we have chosen.

Note that while the problem can be conveniently expressed in terms of the rate and distortion there are other parameters of interest, namely the complexity of the encoding algorithm and the encoding delay.

1.5.2 Encoding delay

We define the encoding delay as the number of "future" blocks that have to be considered in order to encode a given block as in (1.4). For instance if in an image application we select the codebook and reproduction level independently for each block, then there is no encoding delay. If conversely we perform a global allocation of codebooks to blocks we are incurring a one-image encoding delay.

Note that some of the questions raised in Section 1.3 may be tackled without necessarily recurring to an increase in encoding delay. For example the buffer constraint may be dealt with, as has often been the case in the literature, by choosing the codebook based on the current block and the buffer occupancy. Also, assigning different codebooks to different blocks can be done by having some criterion based on the block itself. As an example one could measure the activity (for instance using the total power in the transformed block) in each block of DCT coefficients and assign the codebooks accordingly, so that higher activity blocks get coarser quantization.

In this thesis we will consider problems where we are seeking to optimize the encoding rule to meet some requirements while resorting to the encoding delay as one of the parameters that can be adjusted. Assuming relatively long encoding delay is legitimate for some applications for various reasons. (i) For non-interactive applications, e.g. image transmission, the encoding delay is not as critical as, say, in an interactive videoconferencing environment. (ii) The complexity required when operating with long encoding delays may be acceptable in asymmetric applications (e.g. broadcast) or "one-time coding many-time decoding" (as in coding for CD-ROM) where encoding delay results in only a "latency" at the start of the transmission and where complex encoders can be considered.

1.5.3 Data complexity

A further measure of the complexity of the optimization comes from the amount of data required to achieve optimality within our framework. To avoid making assumptions on the input source, we are required to know the rate and distortion data for each of the blocks. This assumption may be realistic in applications such as block-based image coding [101] where the R-D values can be computed independently and quite efficiently for each of the blocks. However, in environments such as video, where there exists a prediction, the number of R-D points to be computed grows exponentially (see Chapter 4). In order to make these algorithms practical it may be necessary to include some model-based assumptions on the R-D data. While this approach would not achieve optimality the trade-off may be worthwhile in that the amount of data to be computed could be substantially reduced.

1.5.4 Tools

Some of the problems that we will be tackling in this thesis involve searching among a finite, but possibly very large, set of operating points for one that minimizes a cost function while meeting a certain number of constraints. This type of problem is called Integer Programming [67, 66] and, due to the discreteness the set of possible solutions, tends to be particularly difficult since neither the objective, nor the cost functions have the properties (e.g. differentiability, convexity) that are used in problems where solutions have to be found within a continuous set. The main problem is that no specific properties are assumed for the set of points and we need to resort to techniques that will enable us to search through the data without resorting to an exhaustive search. For example, in Fig. 1-2 one would have to find "good" operating points among a discrete set for which no structure is assumed. We briefly mention the two well-known techniques that will be used the most, namely lagrangian optimization and dynamic programming. Here we just give the intuition of the techniques, which will be presented in more detail later on.

Lagrangian optimization. We will only introduce here the geometrical interpretation of this technique (see Fig. 1-3), while a more formal presentation is given in Chapter 2. We start by representing *all* possible solutions for our problem as points in an R-D plot. Note that we have a discrete number of possible operating points, denoted $x_1, x_2, \ldots$, where $x_k = (\hat{X}_1^k, \ldots, \hat{X}_N^k)$ is a possible solution and we are plotting $D(x_k) = \sum D(\hat{X}_i^k)$ versus $R(x_k) = \sum R(\hat{X}_i^k)$. Then, for a choice of a parameter λ, a real positive number, we have that

$$x^*(\lambda) = \arg\min_{x_k}(D(x_k) + \lambda R(x_k))$$

is the operating point that is first touched by a line of absolute slope λ. The main advantage of this technique is that it finds operating points that are good, in the sense of lying on the convex hull of the set of operating points. Further, by varying λ one can sweep all the points on the convex hull and, in particular, search for that point which meets some rate budget constraint. The main problem seems to be that we have to compute the combined R-D characteristic beforehand (i.e. find all the possible combinations of rate and distortion) which would be very complex. In fact, this is not necessary. In the case where all the possible combinations $D(x_k)$ and $R(x_k)$ are the sum of the rate and distortion at each of the blocks, the above minimization can be done *individually* for each block and the combined R-D characteristic of Fig. 1-3 need not be computed. This would be an "independent" coding environment, where each block's rate and distortion can be computed without knowing the other blocks. Details are given in Chapter 2. In some cases however, the "dependent" case, other quantized blocks need to be known in order to quantize a given block (typically because some prediction is used) and thus the optimization becomes more complex. This second case is studied in Chapter 4.

Dynamic programming. In this thesis we use dynamic programming to perform a sequential search through the set of possible solutions, while eliminating the ones that turn out to be suboptimal along the way. Dynamic programming [7, 9] can be used in finding the minimum cost path through a tree or trellis where each branch has a cost attached and the cost is additive over the path. The main result of Bellman's optimality principle (refer to Fig. 1-4) states that if the minimum cost path from A_1 to A_4 passes through A_3 then its subpath going from A_1 to A_3 is *also* the optimal path from A_1 to A_3. Thus to find the optimal path we can set out to find sequentially the optimal path from the initial stage to successive stages, while knowing that any paths that have been pruned along the way *would not* have been globally optimal. In

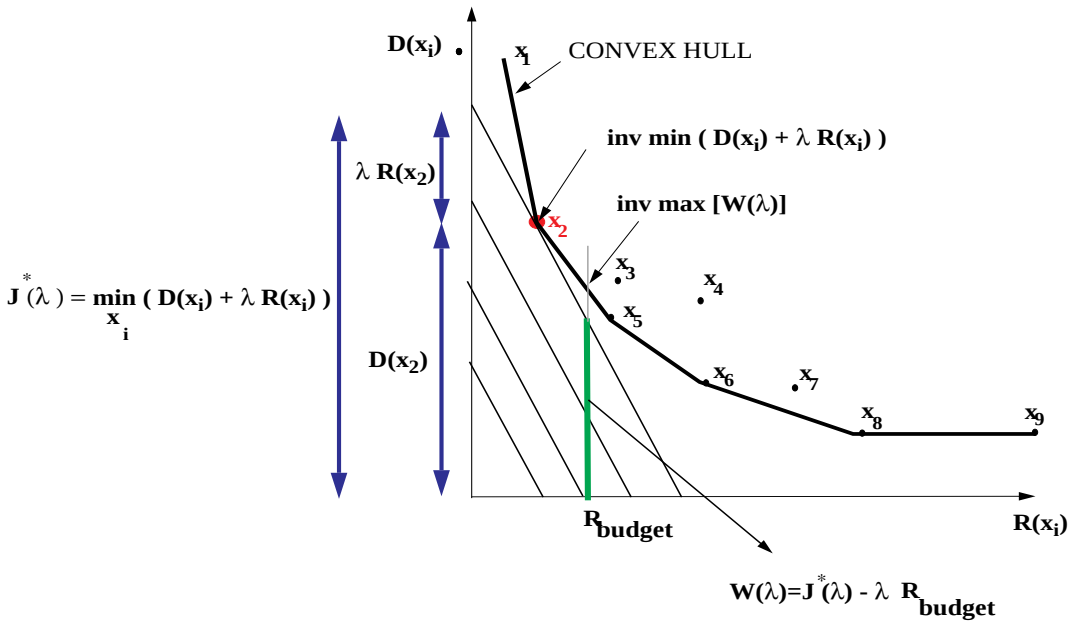

Figure 1-3: Lagrangian optimization. For a given real positive number λ the operating point x_i which minimizes $D(x_i) + \lambda R(x_i)$ is good in the sense of being on the convex hull of the R-D characteristic. For given λ, the point that minimizes the lagrangian cost is that point that is first "hit" by a line of slope λ and is thus guaranteed to lie on the convex hull. Using several values of λ enables to map the convex hull and search for points that meet a budget.

our context, we can use dynamic programming to sequentially eliminate suboptimal solutions. We can grow a tree where each stage represents one block and where the different states represent solutions which use the same number of bits up to that stage. Then we can rule out solutions for which there is an alternative providing less total distortion for the same rate. If two paths converge to the same node in the tree, i.e. the two solutions use the same number of bits for blocks considered, then we need only keep the minimum cost path. We will see how a modified version of this idea can be used to solve the problem of buffer constrained optimization in Chapter 2.

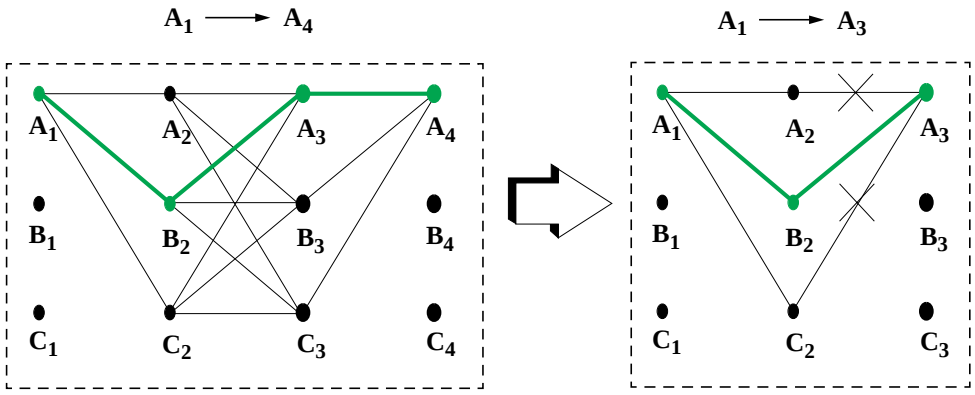

Figure 1-4: Dynamic programming. The problem is to find the minimum cost path in a trellis where each branch has a cost and the costs are additive. If the optimal path from A_1 to A_4 passes through A_3 then the subpath from of this solution that goes from A_1 to A_3 is also the optimal path from A_1 to A_3.

1.6 Overview and contribution of the thesis

Chapter 2 formulates the problem of buffer constrained allocation in a deterministic context. It describes an algorithm to find the optimal solution using dynamic programming and also provides several alternative approaches that achieve near-optimal performance with much reduced complexity. This chapter thus considers the problem of adaptivity through rate control, when a bounded encoding delay is allowed.

Chapter 3 considers the rate control problem in the case of transmission of video over packet networks. It argues that rate control algorithms different from those of Chapter 2 should be used in this environment to enable good overall performance, for *both* coders and network, and, in particular, to ensure existence of statistical multiplexing gain. Furthermore, simple policing strategies such as the leaky bucket may not be sufficient to guarantee the desired performance and we show how a double

leaky bucket approach may help solve this problem.

Chapter 4 is devoted to "dependent quantization", where the dependency comes from using a predictive encoding environment. There we extend the results of optimal independent allocation [101] and show how some additional assumptions are needed in order to avoid having to do an exhaustive search for the solution. Simple examples in the case of MPEG encoding are given and a solution is proposed for the buffer constrained case as well.

In Chapter 5 we consider the problem of allocating rate to the different layers of a multiresolution encoder under the constraint of minimizing the end-to-end delay in querying a multiresolution image retrieval system. In this case we use a simple model to motivate the fact that different allocations should be used depending on the set of images that is being queried.

Finally, Chapter 6 studies the problem of providing adaptivity without requiring that overhead be sent. Since many quantizer design algorithms start by assuming a model for the source, we propose that a way to achieve adaptivity is to have encoder and decoder *learn the model* from the quantized blocks and then use a quantizer design method for the approximated model. We show experimental results for scalar quantizers and study the convergence properties of our model learning technique.