Chapter 5

Models for Rate Distortion Prediction

5.1 Introduction

In this chapter, we introduce models for predicting the rate R and distortion D of frames coded using matching pursuit. We introduce two basic model classes, which will be called Full Pre-Code and Reduced Complexity Pre-Code. Each presents a different tradeoff between model complexity and accuracy, and useful applications for each model class will be discussed. This work is the first in which matching-pursuit specific models have been developed, however several models for DCT-based systems do exist (e.g. [16, 27, 39]). These will be reviewed in Section 5.2.2.

Rate-distortion models are useful in two respects. First, model development often gives useful insights into the process being modeled [27]. This is particularly true of the Full

Pre-Code model which will be shown in Section 5.3. The development of this model will give us a better understanding of the matching pursuit decomposition and illustrate how in-loop quantization affects the process. Second, the resulting models are useful for rate control, particularly for a well-known class of applications which attempt to find the optimal bit allocation across frames in a sequence [29, 8, 54]. The bit allocation is usually chosen to minimize some overall distortion measure subject to a given rate constraint. There are many variations on this theme, but typical algorithms base their optimization decisions on many calculated rate-distortion (R, D) points for each candidate frame to be coded. By designing an accurate model to predict these points instead of explicitly coding them, the complexity of the optimization algorithm can be greatly reduced.

The chapter is arranged as follows. Section 5.2 reviews the relevant literature and formulates the modeling problem. We begin in Section 5.2.1 with a review of (R, D) optimization methods. We identify four candidate methods from the literature to which our (R, D) models might be applied. Three of these methods are shown to fit within a common *dependency tree* framework which will prove useful in Section 5.5.1 for complexity analysis. We then discuss how modeling could be used in such (R, D) based optimization algorithms, and we review existing (R, D) models for the DCT case in Section 5.2.2. Section 5.2.3 then discusses how (R, D) optimization applies to the matching pursuit case, and relevant differences from the well-researched DCT case are identified.

In Section 5.3, we introduce our first (R, D) model for the matching pursuit process. This model relies on a single matching pursuit decomposition using the full dictionary, and is thus called the Full Pre-Code Model. Given such an analysis pass using arbitrary quantizer $Q(\cdot)$ and stopping point N, the model predicts the (R, D) performance of the decomposition with a second arbitrary operating point $\{Q'(\cdot) \neq Q(\cdot), N' \leq N\}$. In this way many such (R, D) points may be generated economically from a single analysis pass. The model works by predicting the interaction between the matching pursuit transform and the in-loop quantizer. During the process, it also generates accurate estimates of other properties of the decomposition such as the absolute modulus value $|p_i|$ as a function of atom number i, and the quantizer rate-distortion (r, d). This model will be shown to be mainly useful in combination with the 2-Pass quantizer design algorithm developed earlier in Chapter 4.

A second class of models is considered in Section 5.4. These models reduce the complexity of the analysis pass by applying a subset of the full dictionary, and are thus called Reduced Complexity Pre-Code Models. The pre-coding results are then used to predict the (R, D) behavior of the full dictionary decomposition using the equivalent mapping from a previously encoded frame. The resulting models are less accurate than the Full Pre-Code Model, but they allow a tradeoff between complexity and (R, D) prediction accuracy for cases in which computational complexity is the constraining factor. A Hybrid Model Extension which combines the Full and Reduced Pre-Code model classes is also developed. This extends the tradeoff flexibility by allowing a continuum of complexity vs. accuracy tradeoff points to be conveniently achieved. This class of models will be shown to be mainly useful in combination with the 1-Pass and Fixed quantizer designs considered in Chapter 4.

In Section 5.5, we evaluate the two matching pursuit model classes in the context of (R, D) based optimization algorithms. Both complexity and accuracy of the models are considered. In Section 5.5.1, the four specific (R, D) optimization methods reviewed earlier are analyzed and the complexity reductions due to the use of our models are computed and evaluated. Significant complexity reductions are shown to be possible. In Section 5.5.2, the accuracy of the matching pursuit models is compared to that of the cubic spline model, which has been successfully applied to a (R, D) optimization problem [39]. Experimental results show that the Full Pre-Code model achieves about the same accuracy as the spline, and the Reduced Complexity models are shown to be slightly less accurate.

Based on the evaluations of complexity and accuracy, we conclude that the models are suitable for use in (R, D) based optimization algorithms. These and additional conclusions are summarized in Section 5.6.

5.2 Models and their application

To understand why rate-distortion models are useful, it is first necessary to explore the optimization algorithms in which they may be usefully applied. We begin with a review of such algorithms in the next subsection. We then describe the models themselves and review existing models for DCT-based systems in Section 5.2.2. Optimization and modeling for the matching pursuit case are then considered in Section 5.2.3.

5.2.1 (R, D) Based Optimization Methods

Problem Formulation

Optimal bit allocation for video compression is a topic that has lately received much attention [48]. The basic problem involves distributing a limited number of bits among multiple coding units such that some overall distortion metric is minimized [54]. A variety of different coding units have been considered; in the DCT case, these may be individual macroblocks, groups of blocks (slices), or entire video frames. Each coding unit has a control parameter which determines the local tradeoff between rate and distortion. The quantization step size is typically the control parameter in the DCT case, chosen from a finite set of possibilities (e.g. $q \in 1, 2, 3...31$). The distortion metric is usually the average mean square error (MSE) taken across all coding units, but other metrics such as perceptually weighted MSE and maximum MSE have also been considered [49, 61].

The problem is then to choose the control parameters for each coding unit such that the global distortion by the chosen metric is minimized, subject to some constraint on the rate. The rate constraint may be a simple budget constraint, e.g $R \leq R_{budget}$. It may also be more complex. For example, in variable bit rate (VBR) transmission for ATM networks, the constraint is derived from the need to meet end-to-end delay requirements while avoiding buffer overflow and simultaneously satisfying network policing constraints [29].

For complexity-limited applications, care must be taken when defining the scope of the optimization. One could in theory optimize the macroblock-level quantizer step sizes over an entire video sequence using the general framework introduced in [54]. However, the resulting optimization would be far too complex to implement. Instead, practical schemes tend to separate the optimization into more tractable subproblems. A typical strategy is to perform the frame-level bit allocation first, and then separately distribute bits to the various coding units (e.g. macroblocks) within each frame [57, 16]. This approach is used in the test models for standard codecs as well; see for example TMN-10 for H.263+ [30] and

Verification Model 11.0 for MPEG-4 [35]. Even with such separation, true optimization of each subproblem is still quite complex [64]. For this reason, the standard test models and many published works are willing to sacrifice optimality in order to lower the complexity or reduce the coding delay [70, 57, 69].

For the remainder of this chapter, we limit our attention to the frame level bit allocation problem. Doing so allows us to leverage existing DCT-based solutions in order to evaluate our matching pursuit (R, D) models. This is possible because many such methods are not DCT-specific, and so are easliy applied to the matching pursuit case. For bit allocation within a frame, DCT-based techniques are not so easily mapped to the matching pursuit case. This is because matching pursuit decomposition is a frame-wise procedure which does not rely on macroblock-style coding units. We thus consider the case where a frame-level optimization method is used to choose the target rate for matching pursuit, and the greedy bit allocation native to the matching pursuit algorithm is used to distribute bits within the frame.

We begin with a sample formulation for the case of budget-constrained bit allocation across frames in a DCT-based MPEG coder. Define (R_k, D_k) to be the rate distortion pair associated with a particular frame k. The problem is then formulated as:

Formulation 5.1 *For frames $k = 1...M$, find the corresponding quantizer step sizes $\{q_k, k = 1...M\}$ with $q_k \in \{1, 2, 3, ...31\}$ such that distortion $D = \frac{1}{M}\sum_{k=1}^{M} D_k$ is minimized subject to rate constraint $R = \left(\frac{1}{M}\sum_{k=1}^{M} R_k\right) \leq R_{budget}$.*

Obtaining a global optimization over multiple frames may be very complex, especially as M increases. The solution will depend on the local tradeoffs (R_k, D_k) which can be achieved in encoding each frame k with all available quantizer steps. An additional complication arises because of inter-frame dependency. Because most frames of motion compensated video are coded using prediction from other encoded frames, the tradeoffs available for frame k will depend on the quantizer steps used to encode all frames on which frame k depends. For example, suppose M frames are to be coded, each using one of C possible quantizers. If the frames are independently coded (i.e. without motion compensation), then the rate distortion tradeoffs (R_k, D_k) for all M frames may be investigated by coding each frame using each quantizer. This requires a total of CM frame encodings. However, if each frame is coded using prediction from the previous frame, then the residual image for frame k becomes a function of the quantizers used for the $k-1$ previous frames. In this case, computing all possible (R_k, D_k) requires

$$C + C^2 + ... + C^M = \sum_{k=1}^{M} C^k \tag{5.1}$$

frame encodings. This quickly becomes too expensive for a brute force search as M increases.

Optimization and Models

Several solutions to the optimal bit allocation problem have appeared in the literature. Before reviewing these, we note that such solutions typically base their optimization decisions on sampled (R, D) points. This concept is illustrated in Figure 5.1. The large box in the figure represents an optimization method which solves a problem of the type stated in Formulation 5.1. The problem is generalized slightly in that generic control parameters σ_k are used to control the rate-distortion tradeoff for each frame k. These replace quantizer steps q_k, which are specific to the MPEG case. The optimization algorithm thus takes the original source frames as input, along with additional information to specify the rate constraint. It then solves the bit allocation problem by choosing control parameters $\{\sigma_k, k = 1...M\}$ which minimize global distortion subject to the given rate constraint. Call this optimal control parameter set $\{\sigma_{k,opt}\}$. This set forms the output of the optimization algorithm. Other useful information, such as the optimally coded frames, may also be returned.

As will be discussed shortly, the contents of the "Optimization Control" box in Figure 5.1 differ greatly among the various optimization methods. However, all of the methods reviewed in the next section base their optimization on obtaining sampled (R, D) points. Consider the rate-distortion generation unit, labeled "Exact (R_k, D_k)" in the figure. Given a specific source image $\vec{f_k}$, this unit computes the exact rate-distortion tradeoff point which would be obtained by encoding $\vec{f_k}$ using a candidate control parameter σ. Note that $\vec{f_k}$ may be either an original image or a prediction residual image, depending on the frame type to be coded. Prediction dependency is thus handled within the optimization algorithm itself.

The algorithm thus probes the local (R, D) tradeoff possibilities by feeding the (R, D) generation unit with candidate images and control parameters, and collecting the resulting computed rate-distortion points. We call this process "rate distortion farming." These points are then used to find $\{\sigma_{k,opt}\}$. Typically many candidate values of σ must be tried with each image $\vec{f_k}$ during the course of the optimization.

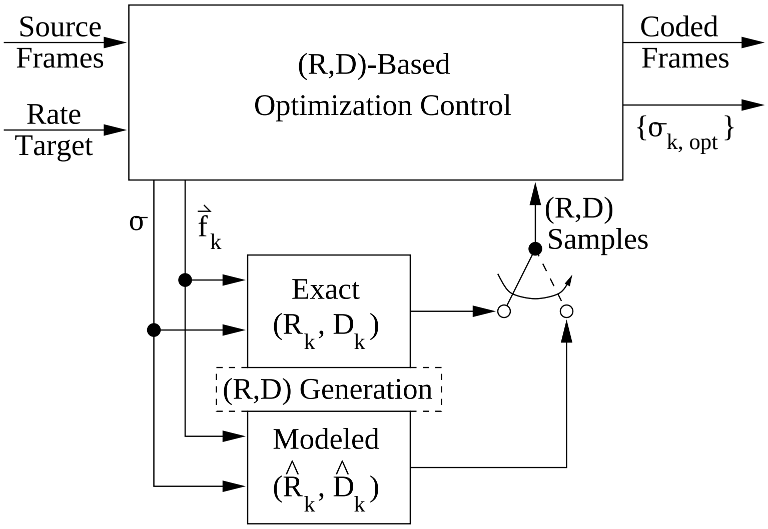
As noted in [54], the optimization complexity is dominated by the rate distortion farming process. We further note that obtaining the *exact* (R_k, D_k) points generally requires the (R, D) generation unit to actually encode each candidate $\vec{f}_k$ using the candidate control parameter. This complexity can be reduced significantly through the use of models. In Figure 5.1, this is shown as a second switchable rate-distortion generation unit labeled "Modeled $(\hat{R}_k, \hat{D}_k)$". When this unit is activated, $\vec{f}_k$ is not explicitly encoded with control parameter σ. Instead, a reduced complexity model is used to estimate the rate distortion tradeoff.

This is the proposed application of the matching pursuit (R, D) models which will be developed in Sections 5.3 and 5.4. In order to be useful in the context of Figure 5.1, such models must meet two requirements. First, they must reduce the complexity of the optimization process compared to the case where exact (R, D) computation is used. Second, the models must be sufficiently accurate so that the obtained parameter set $\{\sigma_{k,opt}\}$ performs nearly as well as the set obtained through the farming of exactly computed (R, D) points. Both complexity reduction and accuracy of the matching pursuit models will be explored in Section 5.5.

Existing Optimization Methods

We now review solutions to the optimal bit allocation problem which have appeared in the literature. Each method is a specific realization of the "Optimization Control" box in Figure 5.1. To maintain a video coding focus, we restrict our attention to solutions which take inter-frame dependency into account. Four such solutions will be identified and described in detail. These will form the test cases for the model-based complexity reduction analysis to be presented in Section 5.5.1. Of these solutions, the first three may be described within a common *dependency tree* framework, as will be shown in the next subsection. The

Figure 5.1: An overview of (R, D) based optimization. The large box represents the optimization method, which relies on sampled (R_k, D_k) points in order to solve for the optimal control parameters $\{\sigma_{k,opt}\}$. The samples may be exactly computed or may be generated by a reduced complexity (R, D) model.



fourth solution handles inter-frame dependency in a unique way and thus will be treated as a special case. In the following descriptions, we necessarily stress the details which are relevant to model-based complexity reduction. The interested reader is referred to the original papers [54, 8, 39, 29] for a more complete description.

1. ***Ramchandran, et.al. [54]:***

   This method is the first known work to handle the dependent bit allocation case. A powerful optimization framework is introduced which is capable of finding the true optimal control parameter set $\{\sigma_{k,opt}\}$ for dependent coding with a simple budget constraint (more complex rate constraints are not considered). The framework is also flexible, and extensions to subband and pyramid coding are shown. We restrict our focus to their solution for video coding with inter-frame dependency.

   The proposed solution framework is based on a combination of the popular Lagrangian and trellis-based methods. A substantial review of these methods is given in [48]. Our summary will necessarily be brief.

   Lagrangian optimization transforms a constrained optimization problem into a series of unconstrained problems through the use of a Lagrange Multiplier (λ). Consider the constrained problem we presented earlier in Formlulation 5.1. The Lagrangian method solves this problem by combining rate and distortion into a unified Lagrangian cost function:

$$J_k(\lambda, \sigma) = D_k(\sigma) + \lambda R_k(\sigma)$$

Consider λ to be fixed. Each choice of control parameter σ will then result in some value of $J_k(\lambda, \sigma)$. The method then seeks to minimize the Lagrangian cost over the frame range of interest, that is:

$$\{\sigma_{k,opt}\} = \min_{\{\sigma_k\}} \sum_{k=1}^{M} J_k(\lambda, \sigma_k) \tag{5.2}$$

This optimization is not subject to an additional constraint, and so is easier to solve than the constrained problem of Formlulation 5.1. However, the lack of a bit rate constraint means that the total rate produced by the optimal control parameter set will actually depend on λ. That is,

$$R = R(\lambda) = \sum_{k=1}^{M} R_k(\sigma_{k,opt})$$

A useful interpretation is that the choice of λ controls the relative importance between rate and distortion in the metric J_k to be minimized.

A key result states that the solution to Equation (5.2) actually solves the constrained problem of Formulation 5.1 for the special case of $R_{budget} = R(\lambda)$. This was originally shown for the case of independent coding units [62, 49] and was later extended to the dependent case [54]. The original constrained problem may thus be solved by using Equation (5.2) with various λ until the total rate $R(\lambda)$ matches the desired R_{budget}.

If this method is applied to independent coding units (e.g. I-frames), then the minimization of Equation (5.2) may be done independently for each frame. That is, $\sigma_{k,opt}$ for frame k is the value of σ which minimizes $J_k(\lambda, \sigma)$, and thus may be found by encoding frame k with all possible σ and computing the resulting Lagrangian costs [49]. The solution is not so simple when the frames are interdependent. In this case, the Lagrangian cost becomes

$$J_k(\lambda, \sigma_k, \{\sigma_{k'}, k' \in \Phi_k\})$$

where Φ_k is the set of all frames on which frame k depends. For this case, the authors divide the optimization frame range into independent groups of frames. Each such frame grouping begins with an I-frame and so does not depend on any previously encoded frames. Computation of the Lagrangian costs within the group are essentially computed by a full expansion of all possible coding combinations. The optimization across many such independent frame groupings is then performed using a trellis search. We omit the details, but the concept is similar to the trellis optimization which will be considered below for the method of Hsu, et.al. [29].

Because full coding expansion within a group of frames may be very complex,[1] the authors introduce a pruning method based on the Monotonicity property [54]. This allows them to eliminate some coding paths from the dependent expansion. It follows from the idea that frames which depend on a particular frame k will benefit in the form of a lower Lagrangian cost when k is coded at a higher quality. The authors claim this property generally holds for DCT-based coding systems, although others have presented cases in which it is violated [8]. Monotonicity-based pruning will not be considered in our present work because it has not yet been demonstrated for the matching pursuit case. It also would result in a data-dependent complexity analysis, which we consider to be beyond our current scope.

We thus consider the rate-distortion farming process of Ramchandran, et.al. [54] to consist of the full dependent expansion of each independent frame grouping. We use the particular grouping on which their main results were based:

$$\{I_1/B_1/B_2/P_1/B_3/B_4/P_2/B_5/B_6\}$$

Our frame grouping notation follows the usual convention of using symbols I, P, and B to represent Intra, Predicted, and Bi-directionally predicted frames, respectively. Each I-frame is coded without prediction and so does not depend on other coded frames. Each P-frame depends on the most recent previously coded I or P-frame. Each B-frame depends on the closest preceding I or P-frame, and also the closest following I or P-frame. B-frames must thus be encoded non-causally.

The (R, D) farming complexity for the above frame grouping will be computed for the exact (R, D) generation case in Section 5.2.3, and the complexity reduction due to our matching pursuit models will be explored in Section 5.5.1.

2. Chen and Lin [8]:

This work solves the dependent bit allocation problem, but the simple budget constraint is replaced by a more complex rate constraint derived for the case of video transmission over an ATM network. The constraint is too complicated to derive here, but it follows from three main factors:

1. The encoder and decoder have physical buffers of a limited size, which must not overflow. This imposes a buffer-state dependent upper limit on the instantaneous bit rate.
2. There is an end-to-end delay requirement. Each encoder bit contributes to a particular reconstructed frame and thus must arrive at the decoder *before* that frame is to be decoded and displayed. This imposes a buffer-state dependent lower limit on the instantaneous bit rate.

[1] A simple example of such dependent expansion was presented earlier, along with the resulting complexity. See Equation (5.1). Complexity for the specific frame grouping presented in [54] will be explored in Sections 5.2.3 and 5.5.1.

3. The encoder must abide by any agreements negotiated with the network, and must not violate any network policing functions which are in place to enforce such agreements.

The authors demonstrate that the bit allocation problem under this more complex rate constraint may be solved for the independent case with Lagrangian optimization. For the dependent case, however, a Lagrange-based solution becomes too cumbersome. Instead, the authors resort to a full tree search. That is, all possible dependent combinations of $\{\sigma_k\}$ are encoded, and $\{\sigma_{k,opt}\}$ is the minimum distortion solution which does not violate the rate constraint. Unfortunately, an exponential explosion of states with tree depth limits the solution window (M in Formulation 5.1) to a small number of frames. For this reason, they use a sliding window approach. Specifically the tree search optimization uses frames k, $k+1$ and $k+2$ to obtain the optimal set $\{\sigma_k, \sigma_{k+1}, \sigma_{k+2}\}$. The first control parameter σ_k of the optimal set is retained and used to code the current frame, but the remaining values are discarded. The optimization window is then shifted by one frame, that is k is incremented, and the process is repeated.

For their experiments, the authors consider the following frame grouping of arbitrary length:

$$\{I/P/P/P/P/P/P...\}$$

The same structure will be used for our complexity analysis in Section 5.5.1. Specifically, we will compute the complexity per location of the length 3 sliding window with and without (R, D) modeling.

3. Lin and Ortega [39]:

This optimization method is completely different from the previous two, and yet we will show that it fits into both the framework of Figure 5.1 and the dependency tree framework which will be developed in the next subsection. The method is also of interest because it already relies on (R, D) models, specifically to account for the dependency. However, it still uses rate-distortion farming to generate these models, and so our own models will be shown to provide further complexity reductions for the matching pursuit case.

The authors specifically consider the following frame grouping:

$$\{I_1/B_1/B_2/P_1/B_3/B_4\}$$

This choice is carefully made in order to limit inter-frame dependency to at most three levels. Also, a constant rate per frame-grouping is assumed. This allows simpler searching and indexing of the resulting video, and has the convenient property of limiting the optimization to a single frame grouping. The goal is thus to distribute a fixed number of bits optimally among the six frames in the grouping so as to minimize the average distortion.

The optimization is done using a gradient-descent method. This allows a fast, directed search through the space of control parameter possibilities. Consider a "distortion surface" Υ which depends on the control parameters used to encode the six frames in the grouping:

$$\Upsilon(\{\sigma_k, k = 1...6\})$$

The surface is composed of contributions from both distortion and from the rate and buffer constraints. It is carefully constructed so that the control parameter set which minimizes Υ will approximately minimize distortion while meeting the rate constraint [40].

The method of Lin and Ortega seeks a local minimum in Υ by selecting an initial point on the surface, and then iterating the following procedure:

1. Compute the gradient at the current location. This indicates the direction in control-parameter space where Υ descends most steeply.
2. Perform a line-search for a local minimum. That is, move along surface Υ in the gradient direction until a local minimum is found.
3. Update the current location to that of the newly found local minimum. If the distance moved is above some threshhold, repeat steps 1-3.

Note that the first two steps require the generation of rate-distortion points for arbitrary control parameter combinations. Dependency would normally require the entire frame grouping to be recoded for each point probed on surface Υ. To avoid this complexity, the authors develop a dependency model based on cubic spline interpolation. Rate distortion farming is first performed using a greatly subsampled set of the dependent control parameter possibilities. The interpolation model is then based on this sparse set of exact (R, D) points. Once the model is defined, an approximation of the rate and distortion may be interpolated for any arbitrary set of control parameters $\{\sigma_k\}$. This in turn allows arbitrary points on surface Υ to be approximately computed. The spline models are much less complex than exact (R, D) compuation, and so the overall optimization complexity is significantly reduced.

Note that this optimization method still falls within the framework shown in Figure 5.1 because of the required initial (R, D) farming step. The "Optimization Control" box thus calls an (R, D) generation unit to obtain enough sampled (R, D) points to create the interpolation models. These models then exist within the control unit, and are used to generate the optimal control parameter set. Further complexity reductions are thus possible using modeled (R, D) generation, and such reductions will be explored in Section 5.5.1.

We refer the reader to the original paper for a description of the spline-based dependency models [39]. In our current framework, we consider rate distortion farming complexity only. For this reason, our complexity reduction analysis will not depend on the specifics of the spline-based models, but it will depend on the (R, D) farming process used to define the models. For this reason, we briefly summarize the process here.[2] As is usual for the DCT case, Lin and Ortega use quantizer stepsize q as the basic control parameter. Consider $\mathcal{S} = \{3, 5, 8, 13, 21, 31\}$ to be the subsampled quantizer set from which each q_k must be chosen. First, the I-frame is encoded using all six possibilities $q_I \in \mathcal{S}$. For two of these I-frame encodings, the dependency path is extended to the P-frame. Specifically, the extended paths are $\{q_I, q_P\}$ with $q_I \in \{5, 13\}$ and $q_P \in \mathcal{S}$. Finally, three of these partial paths are extended to the B-frames. The partial paths are specified by:

$$\{q_I = 5, q_P = 13\}, \{q_I = 13, q_P = 5\}, \{q_I = 13, q_P = 13\}$$

and each is extended to the B-frames with all possible $q_B \in S$.

4. Hsu, et.al. [29]:

Like the work of Chen and Lin [8], this optimization method addresses optimal bit allocation among frames under the complex rate constraints found in ATM network environments. The optimization method is unique among the four in that it does not require tree expansions to account for the dependency. Instead, the authors handle dependency in an iterative fashion. Specifically they use the following algorithm:

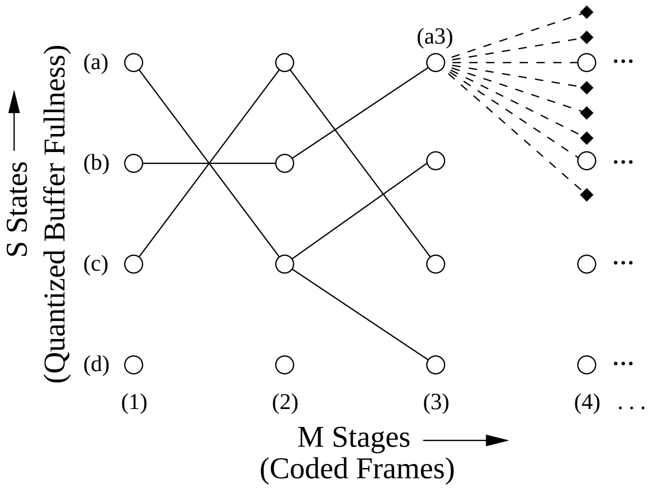
1. Encode the M frames in some suboptimal way to "fix" the predictors (e.g. the motion residual images for each frame k).
2. Ignoring interframe dependency, solve for $\{\sigma'_{k,opt}\}$, the optimal control parameter set given the fixed predictors.
3. Recode the M frames using the parameter set found in Step 2, and use the results to update the fixed predictor set.
4. Repeat Steps 2 and 3 until convergence.

Because inter-frame dependency can be ignored, the complexity of the optimization in Step 2 is greatly reduced. However, there is some question of whether the solution converges to the optimal *dependent* solution, i.e. whether $\{\sigma'_{k,opt}\}$ approaches $\{\sigma_{k,opt}\}$ as the number of iterations increases. The authors do not prove this, but encouraging experimental results are shown.

The solution in Step 2 of the above algorithm relies on a trellis based optimization. This is illustrated in Figure 5.2. The trellis has M stages, one for each frame to be coded. Each stage has S states corresponding to quantized levels of encoder buffer fullness. A branch connecting two nodes at stages $k-1$ and k represents the encoding of frame k with some choice of control parameter σ_k. The nodes themselves indicate the buffer fullness before and after encoding the frame, and a complete path through the tree shows the result of coding all M frames using some combination of control parameters $\{\sigma_k\}$.

[2] Those who find this dependency expansion description to be unwieldy should note that a compact graphical notation for such expansions will be developed in the next subsection.

Figure 5.2: Trellis optimization used in the method of Hsu, et.al. [29]. The extension of node $(a3)$ to the next stage is highlighted. Filled diamonds show resulting buffer fullness levels before state quantization, and each results from a different choice of control parameter σ_4.



The optimization is done progressively in order of stages. For each stage $k-1$, all nodes are extended to stage k by encoding frame k with all possible control parameters. Two pruning methods are used to limit the expansion of paths on the trellis. First, any path extension which would violate the rate constraint is discarded. The focus is on the ATM-network based rate constraint, but *any* arbitrary rate constraint could be handled in this way. Second, all paths leading to each node at stage k are pruned down to a single survivor via a minimum distortion criterion. The interpretation is that all coding paths which arrive at stage k with a particular buffer state are viewed as identical with respect to rate considerations, so only the minimum distortion path need be considered. Distortion here refers to the average MSE across the first k frames.

As an illustration, consider node $(a3)$ in the trellis of Figure 5.2. The path leading to the node represents the encoding of the first three frames using some control parameter set $\{\sigma_1, \sigma_2, \sigma_3\}$. To extend this path to the fourth stage, frame 4 is encoded using all possible choices for σ_4. The resulting buffer fullness levels are shown as filled diamonds on the figure. Any such path extensions which violate the rate constraint are discarded,[3] and the remaining path extensions are quantized and associated with specific states at stage 4. This procedure is repeated for all stage 4 nodes, and then all paths leading to each stage 4 node are pruned to a single minimum distortion survivor path. When stage M is reached, the solution $\{\sigma'_{k,opt}\}$ corresponds to the path which reaches stage M with the minimum distortion. This procedure corresponds to Step 2 of a single iteration of the overall algorithm as described earlier.

[3]For example, if state (a) is the quantized version of a "full" buffer, then it is possible that the top two path extensions of $(a3)$ would result in buffer overflow. In this case, the extended paths would be discarded.

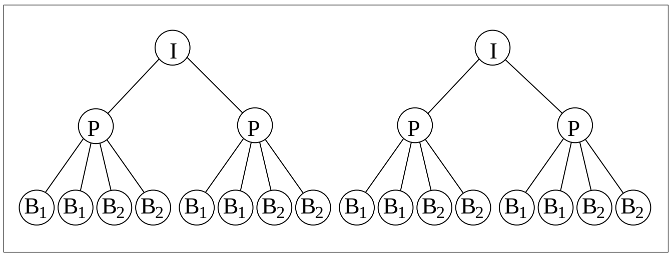
As before, we consider the rate-distortion farming process of the method, since this will be useful for our later analysis of model-based complexity reduction. First note that the method fits nicely into the framework of Figure 5.1, since the path extension performed at each stage k requires (R, D) generation for frame k using all possible σ_k. It is important to note that only a single prediction version of frame k exists at each stage, due to the "fixed" predictors. (R, D) farming for a single trellis iteration thus requires CM frame encodings, where C is the number of control parameter choices. This differs greatly from the other three optimization methods, each of which considers parallel versions of each frame k with varying prediction "parents." For this reason, the method of Hsu, et.al. [29] will not fit within the common framework instroduced in the next subsection. Model-based complexity reduction for this method is thus considered a special case, and will be handled separately in Section 5.5.1.

Common Dependency Tree Framework

The four optimization methods will be used in the complexity reduction analysis of Section 5.5.1. Specifically, we will show how matching pursuit models can be used to reduce the rate-distortion farming complexity of each of the four methods. To simplify the upcoming analysis, we note that the (R, D) farming process for three of the four optimization methods may be described as the expansion of a *dependency tree.* In this section, we develop a compact notation for describing such a tree, and a tree description is provided for each of the three optimization methods which fit into this framework.

To illustrate the concept of a dependency tree, we consider a simple $\{I/B_1/B_2/P\}$ frame grouping. Prediction for the P-frame depends on the I-frame, and each B-frame depends on both the I- and P-frames. Suppose each frame is to be coded with a specific control parameter σ of which there are $C = 2$ different choices. (We choose a small C to keep the example simple). The rate distortion farming process which investigates all possible encodings of these four frames can be shown in a traditional tree form, as shown in Figure 5.3. Each node of the tree corresponds to a *frame instance*, that is a particular

Figure 5.3: Simple example of a dependency tree in traditional form. Full expansion of the $\{I/B_1/B_2/P\}$ frame grouping is shown with $C = 2$ control parameter choices per frame.



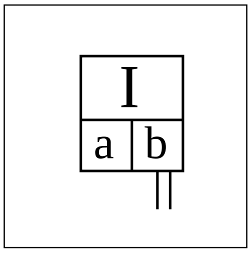
source image encoded using a specific choice of σ. All instances of frame k make up a single tree level, and we place all B-frames with the same level of dependency (e.g. B_1, B_2) on the same level of the tree. The tree is constructed so that the frame instance represented by a particular node depends on its ancestors at higher levels of the tree.

If frame is a P or B-frame, then the source residual image to be encoded will depend on how its parents were encoded. For this reason, parallel instances of each frame with different prediction parents may exist at each tree level. We use the term *cousins* to specify nodes which are related in this way. It is also possible for multiple instances of a single frame to exist under the same set of prediction parents. This occurs when a single unique cousin frame is encoded multiple times using various choices of σ. Each node of this traditional tree representation thus corresponds to a single (R, D) point that must be generated during the (R, D) farming process which the tree represents.

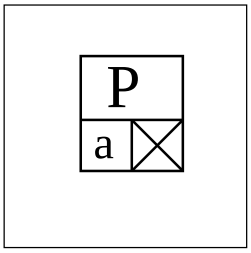
The tree representation is valuable for complexity analysis because it shows explicitly the number of frame instances that must be encoded to complete the rate-distortion farming process. Inter-frame dependence can also be derived from the tree by tracing any path from a root node (top) to a leaf (bottom). Each node visited along this path will generally depend on all previously visited nodes.[4]

[4]We will see an exception to this rule for any frame grouping in which B-frames are followed by a trailing

Figure 5.4: Compact notation for multiple frame intstances on one level of a dependency tree. (a) An I-frame node with a frame instances, each coded using a different control parameter σ. Of these, b are non-terminal and connect to the next level of the tree. (b) A P-frame node with a frame instances, all of which are terminal.



(a)

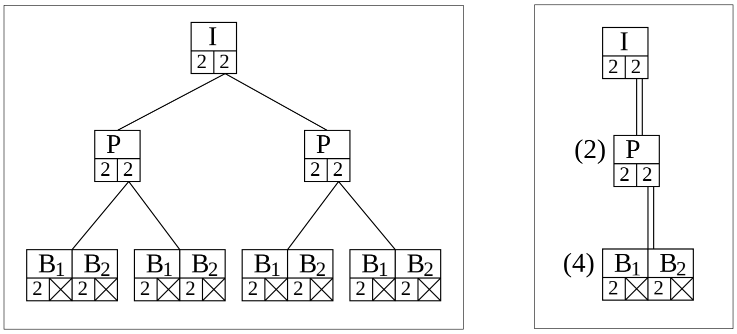


(b)

The traditional tree notation shown in Figure 5.3 has a few major drawbacks. First, because the number of nodes of a full tree is exponential in tree depth, a notation which explicitly draws each node is only suitable for small frame groupings. Second, because each dependency branch is shown explicitly, there is no way to represent a variable number (e.g. C) of branches. To remedy these problems, we introduce a novel compact notation for representing the dependency tree. Each node in the new notation consists of a label and two parameters, as shown in Figure 5.4. The label will typically be the frame type (I, P or B) with an optional identifying subscript. The first parameter a indicates the number of frame instances, that is the number of different control parameters with which the frame is to be encoded. The second parameter $b < a$ shows the number of instances which are non-terminal, that is the instances on which later frames may depend. If b is nonzero, as shown in Figure 5.4a, connecting branches will lead to these later dependent frames. The $b = 0$ case means that all frame instances are terminal. No branches are shown in this case, and parameter b is replaced with the "×" symbol as shown in Figure 5.4b.

I-frame. In this case, the B-frames will depend on the I-frame, which is then placed in the tree to show the dependency. However, the I-frame is of course coded independently of its ancestors in the tree. Such cases will be treated with care in the upcoming analysis.

Figure 5.5: Compact graphical representations for the simple dependency tree shown in Figure 5.3. (a) New representation with fanned branches. (b) Folded-branch representation of the same dependency tree.



(a) (b)

The new representation of our simple $\{I/B_1/B_2/P\}$ frame grouping is shown in Figure 5.5a. To conserve space, multiple B-frames with common parents are grouped together into compound nodes. Comparing this new representation to the traditional tree of Figure 5.3, we see that the new form is more compact and that the total number of nodes has been reduced by a factor of $C = 2$. The number of frame instances which must be encoded during the rate-distortion farming process may still be computed simply by adding the frame instance parameters (a in Figure 5.4) across all nodes. The new representation is thus both compact and useful.

Symmetric dependency trees may be represented even more compactly by folding together identical cousins at each level of the tree. A set of cousin nodes Z may be folded together only if they are *identical* with respect to the following two criteria:

1. All nodes $z \in Z$ must descend from a common parent.

2. All nodes $z \in Z$ must have identical child trees. Here the *child tree* of a node z is defined as the subtree consisting of z and all descending branches and nodes (i.e.

the subtree with z as the root node). Two such child trees are identical only if all branches, nodes, and node parameters (a, b) match.

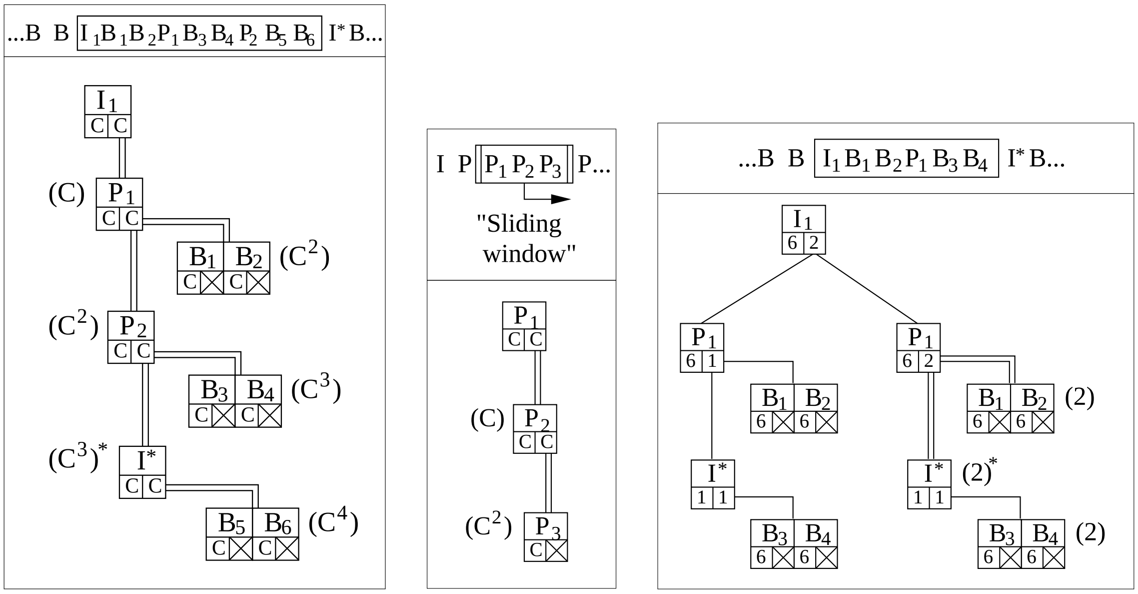
We show the folded representation of our simple example in Figure 5.5b. A double-line is used to represent multiple branches which have been folded together. The number of combined branches is equal to the b parameter of the parent node, which is always found at the top of the folded branch. Note that no information is lost in the folding process; the fanned-branch representation as in Figure 5.5a may be recovered from the more compact folded-branch form by traversing the tree from the bottom up and fanning out each folded branch. However, complexity analysis information is somewhat obscured by the folding process, since the total number of frame instances on a tree level can no longer be simply counted by adding up the a parameters. For more convenient access to this information, we annotate each folded node with a multiplicity factor, A. This is the number in parentheses which accompanies each folded node in Figure 5.5b. The multiplicity factor specifies the total number of identical nodes that are hidden within the folded node. Compound nodes consisting of multiple B-frames are folded and annotated in the same way.

Figure 5.6 shows compact dependency tree representations of the first three previously reviewed optimization methods. Trees for Ramchandran, et.al. [54] and Chen and Lin [8] are shown in Figures 5.6a and 5.6b, respectively. Each of these cases performs a full tree expansion with C control parameter choices per frame. The tree thus follows directly from the frame grouping pattern, which is displayed above the tree for each method. Note that for the method of Ramchandran, et.al., instances of frame I^* appear at the sixth level of the tree. Frame I^* refers to a trailing I-frame which is outside the current frame grouping. However, it must be considered because of B-frame dependency. This case must be treated with care when counting coded frame instances for complexity analysis. The coded instances of frame I^* occur along C^4 different dependency paths, since there are C^4 different ways to encode the pair (P_2, I^*) on which frames (B_5, B_6) depend. However, because I^* is an independent frame it does not really depend on its parents in the tree structure. In fact, the C^4 coding paths at this tree level only require I^* to be coded in C different ways. As a reminder of this case, we will flag the frame instance counter with an asterisk and take special care when counting the complexity.

The tree representation for Lin and Ortega [39] is shown in Figure 5.6c. This case warrants special attention because it expands a subset of the full dependency tree. This essentially means that although a instances of a frame may be encoded at a certain tree level, only a subset consisting of $b < a$ of these continue as dependency branches to the next level. This of course explains why the compact tree representation requires a second node parameter b.

Our earlier review of Lin and Ortega's method specified which control parameter (i.e. DCT quantizer step size) combinations make up the subsampled dependency tree. This level of detail is not required for complexity analysis, and so is not shown on the tree itself. The tree merely indicates how many frame instances must be encoded at each level, and shows the dependency paths which link those instances. The fourth level of the tree contains a trailing I-frame I^*, similar to that already discussed for the case of Ramchandran, et.al. [54]. We note that while frame I^* connects a total of three dependency paths on which frames (B_3, B_4) depend, only two unique encodings of I^* are actually performed.

Figure 5.6: Dependency tree representations for three (R, D) optimization methods. (a) Ramchandran, et.al [54]. (b) Chen and Lin [8]. (c) Lin and Ortega [39].



(a) (b) (c)

Complexity analysis using the tree representations of Figure 5.6 will be performed in Section 5.5.1, and the results will be used to evaluate the complexity reduction due to matching pursuit (R, D) models used within each optimization method. The fourth optimization method due to Hsu, et.al. [29] does not fit into the dependency tree framework and will thus be analyzed as a special case. We now move on to a discussion of (R, D) models and review existing models for the DCT case.

5.2.2 Models for (R, D)

Consider the second (R, D) generation unit labeled "Modeled $(\hat{R}_k, \hat{D}_k)$" in Figure 5.1. This unit produces reduced complexity estimates of the rate-distortion tradeoff that would result from encoding source image $\vec{f}_k$ with some control parameter σ. We now consider possible designs for such models, and we review existing models from literature.

Source models come in two basic types. Analytical models are based on an analysis of the internal workings of the system, and this internal behavior is then modeled. Empirical models instead attempt to model the input/output relationship of the system by looking at experimental output data, regardless of what is really going on inside [27]. As one might expect, empirical models tend to be less complex than their analytical counterparts. However, analytical models may capture the behavior of the system more completely and perform more robustly; empirical models may perform poorly when the input data differs greatly from the data used to develop the model.

In this work, we develop the first known rate distortion models for the matching pursuit algorithm. Models of both types will be considered; the Full Pre-Code model of Section 5.3 is an analytical model, and the Reduced Complexity Pre-Code models of Section 5.4 are empirical. A hybrid extension of these models presented in Section 5.4.1 is essentially a cross between the two types. Although no other matching pursuit models are available for comparison, various models for DCT based coders do exist. In reviewing these, we will comment on whether extensions to matching pursuit (R, D) are possible for each case. We will also show that a particular model due to Lin and Ortega [39] may be used to benchmark the accuracy of our matching pursuit models.

Analytical Models

An interesting analytical model was developed by Hang and Chen [27]. They combine existing theoretical models for the components of a DCT encoder, and they adjust the resulting model to account for real effects not covered by the theory (e.g. the difference between VLC coding and entropy). Given the variances of the DCT coeficients, their model predicts the relationship between rate R, distortion D, and the quantizer step size. They do not apply the model to the optimal bit allocation problem, but they do show an application to adaptive frame dropping [28].

A second analytical model was introduced by Uz, Shapiro and Czigler [66]. This is a theoretical model of motion dependency for DCT based video, and closed form solutions for optimal bit allocation are derived from the model. The accuracy of the model is in doubt, however, since no experimental results were presented, and so the model prediction was never shown to match a real DCT encoder.

One problem with the above analytical models is that each is based in some way on the internal workings of a DCT-based system. While perfectly valid models result from this process, they are not useful for our purposes because they are too DCT-specific. Any comparison to the matching pursuit models which will be developed in Sections 5.3 and 5.4 requires a model which is easily adapted to matching pursuit. In search of such a case, we turn our attention to existing empirical models.

Empirical Models

Simple empirical models for motion dependency and DCT-based residual coding are developed by Katto and Ohta [36]. Two distinct models are created to solve two different problems. The first model is used to solve for the optimal arrangement of I, P, and B frames within an MPEG group of pictures (GOP). A second model is then used to solve for the optimal average bit allocation among the frame types (I,P,B). Note that the bit allocation problem solved by Katto and Ohta is "easier" than that of Formulation 5.1, because they optimize over only a single GOP, and also because they assume all frames of the same prediction type get the same target bit allocation. Their simple closed form solution cannot be expected to perform as well as the more expensive methods we reviewed in Section 5.2.1. Still, the authors show experimental gains of 0.4 dB over standard DCT-based rate control methods. Nearly identical gains are shown in a later work by Mizuno, et.al. [42], in which an identical problem formulation is solved using a very different empirical model. However, the models presented in both of these works rely on the assumption that quantization error variance in the transform (DCT) coeficient domain is equal to that in the reconstructed image domain. While this assumption generally holds for orthogonal basis sets, it does not hold in the case of matching pursuit with an overcomplete basis.

Another empirical model was developed by Ding and Liu [16]. The model is based on fitting approximations of the form

$$R(\sigma) \approx \hat{R}(\sigma) = \alpha + \frac{\beta}{\sigma^{\gamma}}$$

where $R(\sigma)$ is the rate required to encode the current frame using control parameter σ, which in the DCT case refers to the quantization step size. Parameters $\{\alpha, \beta, \gamma\}$ are then used to fit the approximation to the true function $R(\sigma)$ using a few exactly computed points. Unfortunately, distortion models $\hat{D}(\sigma)$ were not considered in this work. A rate model of the above form could be considered for matching pursuit, however a different model proposed by Lin and Ortega [39] has been shown to be more accurate, and covers distortion modeling as well. This model will now be considered in detail.

Cubic Spline Interpolation

Recall our earlier review of the optimization method of Lin and Ortega [39], which used a model based on cubic spline interpolation to account for inter-frame dependency. Before extending the cubic spline model to the multi-frame dependency case, the authors considered a single frame (R, D) model based on cubic spline, and they showed experimental results to compare the accuracy of the model to others presented in literature. The spline model was shown to be the clear winner in terms of accuracy. Also, the generic nature of the model allows for easy extension to the matching pursuit case (or virtually any coding method). For these reasons, we consider the cubic spline model in detail as a candidate for benchmarking the accuracy of our matching pursuit (R, D) models.

The spline model is empirical and conceptually very simple. Rate and distortion are separately modeled as functions of the control parameter σ. Consider the distortion model, $\hat{D}(\sigma)$. This model is based on N_c exactly computed distortion points, e.g. $\{D(\sigma_j),$ $j = 1...N_c\}$. These serve as "control points" for the model, which then uses cubic spline interpolation [52] to estimate the points which fall between the known control points. The result is a reduction in (R, D) data generation complexity, so long as the number of (R, D) points interpolated by the model is greater than the initial investment N_c. At least four points are required for the interpolation [52], though the prediction accuracy may be scaled at the cost of complexity by computing more control points. To achieve the accuracy needed for the bit allocation optimization, Lin and Ortega typically use five or six control points per frame [39].

In justifying the spline model, the authors presented accuracy comparisons to three other DCT-based models. The first two were existing models due to Hang and Chen [27] and due to Ding and Liu [16] which were mentioned in our earlier review. The third was a piecewise linear interpolation model proposed by the authors. In an experimental comparison of accuracy, the cubic spline model was the clear winner, achieving average prediction errors of 0.99 and 0.51 percent for rate and distortion prediction, respectively.

We hypothesize that such accuracy levels are sufficient to enable (R, D) models to be used within an optimal bit allocation algorithm. The precedent for this claim is also in the Lin and Ortega paper, since the authors did incorporate the spline interpolation model into their gradient-based optimization algorithm, and the resulting system showed significant improvement over existing rate allocation methods [39]. We will thus use cubic spline interpolation as a benchmark for judging the accuracy of our matching pursuit (R, D) models. An experimental comparison will be presented in Section 5.5.2. In order to present a controlled accuracy comparison for the matching pursuit case, it is necessary to extend the cubic spline interpolation method to model $R(\sigma)$ and $D(\sigma)$ for the matching pursuit encoder. Such an extension is straightforward. Because the spline model simply interpolates points along functions $R(\sigma)$ and $D(\sigma)$, it may be easily applied to any coding method.

We have thus far reviewed the literature for both (R, D) based optimization methods and (R, D) models, and we have laid the groundwork for evaluating our models in terms of complexity and accuracy. We now consider the optimization problem specifically for the matching pursuit case, and we identify relevant differences from the DCT case on which previous literature has focused.

5.2.3 The Matching Pursuit Case

We now turn to the matching pursuit case. First we note that the optimization framework described in Figure 5.1 is not specific to DCT-based systems, and readily applies to matching pursuit. In this case, the "Exact" (R, D) generation is an exact copy of the matching pursuit residual encoder, and the "Modeled" unit is some model which predicts matching pursuit rate and distortion. The definition of control parameter σ has not yet been defined for the matching pursuit case, but will be considered shortly. The choice of σ for DCT-based systems was simple because the (R, D) relation for such systems is solely driven by the quantizer step size q. Matching pursuit (R, D) may rely on both the number of atoms N and also the design of the in-loop quantizer $Q(\cdot)$. To address this issue, we begin with a review of the matching pursuit quantizer-design methods which were developed in the previous chapter.

Matching Pursuit and Quantizer Design

Recall that matching pursuit encoding relies on an in-loop quantizer $Q(\cdot)$ to quantize the modulus p_i for each atom stage i. Here "in-loop" means that the quantized atom is subtracted from the remaining signal energy at each stage, and so the quantization error from one stage may be corrected by atoms at future stages. Early encoder designs used a fixed modulus quantizer, specifically a uniform quantizer with a fixed stepsize [45]. Adaptive designs were considered in Chapter 4. These allow the quantizer design to adapt to the characteristics of each frame, and were shown to improve the coding efficiency of the matching pursuit encoder, particularly at high bit rates. Two different adaptive schemes were considered, a near-optimal 2-Pass design, and a reduced complexity 1-Pass design with slightly reduced performance.

Consider the optimization framework of Figure 5.1 where the system to be optimized is a matching pursuit encoder, and the optimization method is fixed. We will show that the rate-distortion farming complexity needed for the optimization will depend on the quantizer design choice used for matching pursuit. Knowing this, it is not surprising that the complexity reductions due to matching pursuit (R, D) modeling will also depend on the quantizer design. Before discussing these effects on complexity, we first review the three matching pursuit quantizer design methods.

1. ***Fixed Quantizer*:**
   A fixed quantizer design was used in our early matching pursuit work [46, 44, 45] and has been used as the default in extensions by other authors [15, 55]. This is a simple mid-tread uniform quantizer with midpoint reconstruction, which was shown earlier as Figure 4.1. The stepsize is fixed at $QP = 30$, which was experimentally determined to work well across the MPEG-4 test sequence set. The idea of a fixed stepsize is foreign to DCT-based systems, since such systems must vary the quantizer to achieve rate control. The (R, D) tradeoff for matching pursuit is actually more sensitive to the number of atoms N, so rate-control can still be achieved by fixing the quantizer and varying N. By using the average rate per atom from the previous frame as an estimate for that of the current frame, N may be chosen to accurately target a particular bit rate, as was demonstrated in Chapter 2.

   Obviously for a fixed quantizer design, no initial analysis of the current frame is required in order to specify the quantizer. Also, because of the progressive nature of the coding method, a single matching pursuit pass to N_{max} atoms traces out all possible rate distortion tradeoffs due to encoding N atoms where $N < N_{max}$.[5] Thus if a particular optimization method requires many rate-distortion points for some frame k, these may be efficiently computed using a single encoding pass with N_{max} atoms, where N_{max} is the maximum N that would be required across all desired encodings of the frame.

[5]The encoder explicitly tracks the remaining signal energy $\vec{f_i}$ as atoms are subtracted. Distortion can be computed progressively using $\left\|\vec{f_i}\right\|^2 = \left\|\vec{f}_{i-1}\right\|^2 - p_i^2 + d_i^2$, where p_i and d_i are the current stage modulus and quantzation error, respectively. This is simple combination of Equations (4.3) and (4.2). Computing the progressive rate for stage i requires a few VLC table lookups and a local position order re-sorting, which are of negligible complexity compared to the atom search procedure.

2. Adaptive 2-Pass Design:

Introduced in the previous chapter, this design exploits the fact that matching pursuit (R, D) is sensitive to both the number of atoms N and the design of quantizer $Q(\cdot)$. The quantizer is effectively uniform with an expanded dead zone, where both dead zone size DZ and quantizer step size QP are adapted to the current frame. An initial analysis decomposition is first performed, consisting of full matching pursuit encoding without quantization. This pass determines the number of atoms needed to hit the target bit rate, and also the resulting quantizer dead zone size. The step size is then chosen to maintain an optimal ratio QP/DZ (typically 0.6), and the frame is then encoded to N atoms using the quantizer defined by parameters (DZ, QP).

For basic matching pursuit encoding, the complexity of the 2-Pass design is twice that of the fixed design, since each coded frame requires an extra initial pass to define the quantizer. For rate-distortion farming, each frame is typically encoded using many different control parameters σ, and each such encoding will require a different quantizer design. Because the initial analysis pass is always done without quantization, it may be shared among all choices of σ. It is thus performed once to some number of atoms N_{max}. This defines the optimal parameters $\{N(\sigma), DZ(\sigma), QP(\sigma)\}$ for all desired encodings which satisfy $N(\sigma) < N_{max}$. However, each choice of σ then requires a separate quantized pass using its unique quantizer to complete the encoding. Obtaining exact (R, D) points for C control parameter choices thus requires a total of $1 + C$ matching pursuit passes. [6]

3. Adaptive 1-Pass Design:

This is an approximation to the 2-Pass design method, but the initial analysis pass is eliminated. Instead, the quantizer parameters (DZ, QP) are estimated from the previous frame coding results and the starting energy of the current frame. Note that these quantizer design parameters are independent of the choice of σ. In other words, a single predicted quantizer design is used for all σ of interest. Given this design, rate control must be based only on the number of encoded atoms. Specifically, the current frame is progressively encoded until some desired target rate is met. The point at which this occurs effectively defines the number of coded atoms, $N(\sigma)$.

Because the quantizer is independent of the control parameter, all choices of σ will encode the frame using the same quantizer but with different values of $N(\sigma)$. This is identical to the fixed quantizer case with respect to complexity. Thus

[6] We adopt the total number of matching pursuit passes as our measure of data farming complexity. This simplifies the analysis and avoids dependence on the source sequences and control parameter choices. However, it ignores the fact that each pass varies in complexity with associated number of atoms $N(\sigma)$. We justify this omission by noting that C passes with different control parameters are typically considered, and the resulting values of $N(\sigma)$ will typically form a cluster about some N_{avg}. For complexity analysis, such structures are closely approximated as C passes of N_{avg} atoms each. Total complexity is thus nearly proportional to the total number of passes, which is our adpoted measure.

data farming for a frame with many control parameter choices can be efficiently performed using a single matching pursuit pass to some suitable number of atoms N_{max}.

Control Parameter Definition

The local (R, D) tradeoff for matching pursuit encoding is sensitive to both the number of atoms N and the design of quantizer $Q(\cdot)$, which for our purposes may be specified by parameters DZ and QP. For the case of Fixed quantizer design, only N is used for rate control, and thus N could be used as the control parameter. However, this is a poor choice for the adaptive designs, in which the quantizer parameters are also allowed to vary.

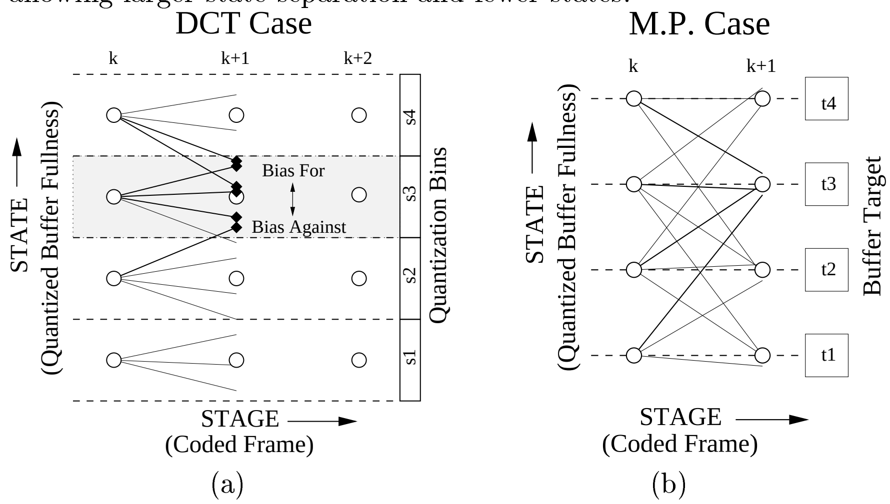
Instead, we choose the *target encoding rate* for the current frame to be the control parameter for matching pursuit based (R, D) optimization. In doing so, we unify all three quantizer design methods under a common control paradigm, since each method has an associated way of choosing parameters $\{N, DZ, QP\}$ to accurately hit a given target rate specified by σ. This simplifies the design of the "Optimization Control" layer in Figure 5.1, since the details of designing $Q(\cdot)$ and choosing N may be abstracted to the (R, D) generation layer. The optimization method itself need only be concerned with target rates, which are much more relevant to the problem it seeks to solve. We illustrate this advantage through the use of an example.

***Example: Rate targeting for trellis search*:**

Consider the trellis search used in the optimization method of Hsu, et.al. [29]. This was reviewed earlier in Section 5.2.1, and illustrated in Figure 5.2. We begin with the DCT case, in which the control parameter is specifically the quantizer step size q. Path extension at each stage k is thus performed by encoding frame k using all possible step sizes from some candidate set $\mathcal{S}$.

Recall that buffer fullness (in bits) is used as the state of the system, and quantization of buffer fullness is used to reduce the number of states and thus the complexity of the optimization. An earlier work by Ortega, et.al. [49] explored

Figure 5.7: Transition between quantized states for the dynamic programming solution to optimal bit-allocation. (a) The DCT case. Paths arrive randomly in bin $s3$ and are merged by the minimum distortion criterion. Bin size is limited by rate disparity across the bin. (b) The matching pursuit case. Candidate rates chosen to target each state. Rate disparity is limited, allowing larger state separation and fewer states.



the loss of optimality which results from state quantization. There it was shown that quantization bins as large as 50 bits result in negligible losses, but optimization performance degrades for larger bin sizes.

State quantization losses may be interpreted in the following manner. Consider the set of all candidate coding paths which arrive at stage k with buffer fullness $B(k)$ within a particular quantization bin. All such paths are treated identically with respect to rate, and are in fact *merged* into a single surviving path based solely on minimum distortion. This is clearly suboptimal, since paths with $B(k)$ near the top of the bin have the advantage of a higher bit rate than those where $B(k)$ is near the bottom of the bin. Paths of the former category thus have a lower expected distortion than those of the latter, and so a bias for survival exists for paths which happen to arrive near the top of the quantization bin of any state. If a particular path survives because of this bias, then alternate paths with lower rate and higher distortion could have been discarded, even though they may have been superior to the surviving minimum distortion path in an overall (R, D) sense. This effect certainly limits the useful bin size for state quantization, and contributes to the losses demonstrated in [49] at higher bin sizes.

An illustration of this problem is shown in Figure 5.7a. Note that the problem is caused by the rate disparity of the various path extensions, each due to coding some frame with some quantization step size. A random distribution of $B(k)$ levels results within each state quantization bin.

This problem may be avoided for the matching pursuit case with an appropriate design for the control parameter set. Recall that control parameters σ for this case correspond to target rates, and that accurate rate targeting is intrinsic to matching pursuit under each of the three quantizer design methods. To take advantage of this ability, we consider the states to be target buffer fullness values rather than quantization bins. Consider $\mathcal{S}$ to be the set of all target rate points which are used to extend each previous stage survivor path. Set $\mathcal{S}$ would then be designed to target each possible buffer fullness state at the current stage.

An illustration of the effect of such rate targeting is shown in Figure 5.7b. This could be interpreted as an extension of state quantization but with the bin center *intentionally targeted.* State progression in this manner would be more orderly than it was for the DCT case, and accurate targeting of the state values would allow competing paths to be merged on an equal bit rate footing. Specifically, all paths arriving at a given state would be within one or two atoms (essentially 40 bits) of the target budget, even if the inter-state spacing were increased to a much higher value. This in theory allows the matching pursuit system to retain fewer states and thus reduce the complexity for the same PSNR loss due to state quantization. However, the idea could also be implemented for a DCT-based system with the addition of a sophisticated rate targeting model (e.g. [16]).

We conclude that target rate points have an advantage over other available control parameter options, and are more relevant to the design of the optimization system. We are now ready to formulate the optimal bit allocation for the matching pursuit case.

Problem Formulation

We now formulate a sample matching pursuit bit allocation problem. The equivalent DCT-based problem was given as Formulation 5.1. For that formulation, quantization step size was used for the control parameter. In our new formulation, σ_k corresponds to the target rate chosen for the encoding of a particular frame k. It is chosen from a pool of candidate rates $\{\bar{R}_c, c = 1...C\}$. The resulting optimal bit-allocation problem is thus stated as follows.

Formulation 5.2 *For frames $k = 1...M$, find the corresponding frame level target rates $\{\sigma_k, k = 1...M\}$ with $\sigma_k \in \{\bar{R}_c, c = 1...C\}$ such that distortion $D = \frac{1}{M}\sum_{k=1}^{M} D_k$ is mini-*

mized subject to rate constraint $R = \left(\frac{1}{M}\sum_{k=1}^{M} R_k\right) \leq R_{budget}$.

Recall that more general versions of this bit allocation problem have been solved for the DCT case. For example, generic coding units such as macroblocks have been used. For matching pursuit, all existing rate control has been done at the frame level, and so we restrict our attention to bit allocation across frames in a sequence. This does not rule out further optimization within frames. In fact, an improvement to the in-frame spatial bit allocation was the effective result of the heuristic energy weighting scheme presented in [3], and further improvements might be shown using these weights as the control parameters for (R, D) optimization. Such improvements will not be considered here.

As in the DCT case, the matching pursuit bit allocation problem may be easily extended to other distortion metrics and more complex rate constraints. This is not a problem because all of the optimization methods reviewed in Section 5.2.1 were formulated in terms of generic distortion measures, and three of the four [8, 39, 29] could be used with generic rate constraints. Most solutions for inter-frame dependency [54, 29, 8] are also not DCT-specific, and thus could be extended to the matching pursuit case.

Now that the matching pursuit version of the (R, D) optimization problem has been suitably defined, we move on to consider the complexity of the resulting (R, D) farming process.

Exact (R, D) Generation Complexity

We now consider the complexity of the exact (R, D) farming process for matching pursuit with each combination of optimization method and quantizer design type. This is the farming task performed by the exact (R, D) generation unit pictured in Figure 5.1. Using an example, we show how such complexities may be derived from the compact dependency tree representations developed in Section 5.2.1. We then generalize the process to allow simple complexity analysis of any arbitrary dependency tree. The resulting analysis is used to compute the (R, D) generation complexity for each of the three methods which were earlier shown to fit within the dependency tree framework. Analysis of the fourth optimization method due to Hsu, et.al. [29] is presented as a separate case.

1. ***Complexity of dependency tree expansion:***

   Consider the full expansion of the $\{I/B_1/B_2/P\}$ frame grouping. This simple example was presented earlier in Section 5.2.1 and illustrated in compact form as Figure 5.5b. We now show separate complexity analysis of the corresponding (R, D) farming task for each matching pursuit quantizer design.

   For fixed quantization, the quantizer is known in advance and so a single source image may be coded to many choices of σ using a single shared encoding pass. When multiple cousins are hidden within a single folded node, each corresponds to a unique source image and so each hidden cousin requires a separate encoding pass. For our simple example with $C = 2$, the I-frame layer requires 1 encoding pass to compute both control parameter instances.[7] The P-frame layer consists of 2 unique cousin nodes, each of which requires a single encoding pass. Finally, the B-frame layer consists of 2 sets of 4 unique cousin nodes, one corresponding to each of the two B-frames. Each of these eight frame instances requires an additional pass. The total number of required passes is thus:

   $$1 + 2 + 2(4) = 11$$

   For the 1-Pass adaptive quantizer, a predictive design is used, and so again the quantizer is known at the start of coding. Complexity is thus equivalent to the Fixed design case, as noted in our earlier review. Thus, matching pursuit with a 1-Pass quantizer also requires a total of 11 encoding passes.

   Before proceeding to the 2-Pass design case, we consider generalizing the above procedure to apply to arbitrary dependency trees. We first note that the com-

[7] To maintain a single encoding environment for complexity analysis, we assume that matching pursuit is used to encode each frame type, including the I-frames. This effectively upper bounds the I-frame complexity, since less complex methods such as DCT or wavelet are used in practice. We could alternately ignore I-frame complexity, since even our upper bound is insignificant in most real cases, due to the exponential expansion of P and B-frame instances with dependency tree depth.

plexity analysis used above can be summarized compactly as:

$$\sum_{\forall z \in \mathcal{Z}} A_z$$

where z is a node, $\mathcal{Z}$ is the collection of nodes which forms the complete tree, and A_z is the multiplicity factor associated with node z. By convention, $A_z = 1$ for single (non-folded) nodes.

For the general case, the above summation procedure must be modified to account for the special case of trailing I-frames. These were introduced in Section 5.2.1 and were labeled as I^* on Figures 5.6a and 5.6c. Such a frame lies immediately beyond the end of the current frame grouping, but must be expanded to account for B-frame dependency. The frame must thus be included in all dependency paths leading to the B-frames. However, because I^* is independently coded, the various cousin representations of I^* in the tree actually represent encodings of the same intra source image. The multiplicity factor annotation for I^* is thus typically greater than the number of I-frame encodings that must be actually performed. In fact, for Fixed and 1-Pass quantizer designs, a single shared encoding pass may be used to realize all required encodings of frame I^*. A generalized rule for computing dependency tree expansion complexity κ_1 for Fixed and 1-Pass quantizer designs may thus be stated as:

$$\kappa_1 = \mathcal{I}_{(I^*)} + \sum_{\forall z \in \mathcal{Z}'} A_z \tag{5.3}$$

where $\mathcal{Z}'$ is the set of all nodes z in tree $\mathcal{Z}$ which do *not* correspond to a trailing I-frame, and $\mathcal{I}_{(I^*)}$ is an indicator function which is equal to 1 if the expansion of a trailing I-frame must be considered, and 0 otherwise. Note that for typical cases, the $\mathcal{I}_{(I^*)}$ term is insignificant and may be ignored.

We now move on to the case of 2-pass adaptive quantizer design. For this case, coding a single source image to C different control parameter choices requires $(1+C)$ encoding passes. The first shared pass is performed without quantization and is used to define the parameters $\{N(\sigma), DZ(\sigma), QP(\sigma)\}$ to be used with each control parameter σ. Coding is then completed for each choice of σ, which accounts for the remaining C passes.

For our simple example, $C = 2$, and so the I-frame layer requires 1 analysis pass and 2 completion passes. The P-frame layer has 2 unique cousins, each of which requires 1 analysis pass and 2 completion passes. Finally, the B-frame layer consists of 2 sets of 4 unique cousins. Each of these eight frame instances requires 1 analysis pass and 2 completion passes. The total number of required passes is thus:

$$(1+2) + 2(1+2) + 2(4)(1+2) = 33$$

Using the previously introduced notation, we summarize this complexity analysis procedure as:

$$\sum_{\forall z \in \mathcal{Z}} A_z \cdot (1+C)$$

We now extend this rule to cover the expansion of arbitrary dependency trees. Since each node may be encoded to a different number of control parameters, we define C_z as the number of σ choices encoded for node z. This is node parameter a in the notation of Figure 5.4. To handle trailing I-frames, we define $C_{(I^*)}$ as the total number of encodings of I^* which are needed for B-frame dependency. We set $C_{(I^*)} = 0$ for any case in which a trailing I-frame need not be considered. Exact (R, D) complexity κ_2 for matching pursuit with a 2-Pass adaptive quantizer is thus computed as:

$$\kappa_2 = \mathcal{I}_{(I^*)}(1 + C_{(I^*)}) + \sum_{\forall z \in \mathcal{Z}'} A_z \cdot (1 + C_z) \tag{5.4}$$

As before, the term corresponding to frame I^* is often small enough that it may be ignored.

The complexity computation rules defined by Equations (5.3) and (5.4) may be applied to arbitrary dependency tree representations. We thus apply them to the three dependency trees shown in Figure 5.6. These correspond to the optimization methods which were earlier found to be within the dependency tree framework. A summary of the resulting complexities is given in Table 5.1.

2. Complexity analysis for Hsu, et.al.:

We now consider the fourth optimization method due to Hsu, et.al. [29]. Recall that this method handles dependency in an iterative manner by alternately "fixing" the predictors and using trellis optimization to find $\{\sigma'_{k,opt}\}$, the optimal control parameter set given the fixed predictors. Because dependency is essentially ignored during each optimization iteration, the dependency tree framework does not apply. We thus treat this as a special case, and proceed to analyze the complexity for each matching pursuit quantizer design.

We begin by noting that each dependency iteration consists of two steps. The first step is the trellis optimization, given the fixed predictors defined in the previous iteration. Recall that an illustration of this was shown in Figure 5.2. The second step encodes the entire sequence using $\{\sigma'_{k,opt}\}$, the optimal control parameter set found using the fixed predictors. We count the number of encoding passes used in these two steps as a measure of per-iteration complexity. We assume generic values for the number of frames M, the number of control parameter choices C, and the number of quantized buffer states S.

Consider first a Fixed quantizer design, and assume the "fixed" predictors were generated during the previous iteration. In order to compute these fixed predictors, each frame has already been encoded along the single dependency path defined by the previous $\{\sigma'_{k,opt}\}$. Since the trellis optimization step is fixed to this same prediction path, the (R, D) tradeoffs for encoding each frame with the fixed quantizer are already known,[8] and trellis optimization requires no additional passes. Once the new set $\{\sigma'_{k,opt}\}$ is found, re-encoding all frames to

[8] As before, we require each frame to be coded to some number of atoms N_{max}, where $N(\sigma) < N_{max}$ for all σ of interest, and so exact (R, D) computation may be efficiently shared across all choices of σ.

Table 5.1: Exact (R, D) generation complexities for the three optimization methods covered by the dependency tree framework. Each entry is the total number of encoding passes per frame grouping required for the (R, D) farming task. Where applicable, C is the number of control parameters considered per frame.

Optimization Method	Matching Pursuit Quantizer Design	
	Fixed or 1-Pass Adaptive	2-Pass Adaptive
Ramchandran, et.al. [54]	$2 + C + 3C^2 + 2C^3 + 2C^4$	$(1 + C)(2 + C + 3C^2 + 2C^3 + 2C^4)$
Chen and Lin [8]	$1 + C + C^2$	$(1 + C)(1 + C + C^2)$
Lin and Ortega [39]	16	108

update the fixed predictors to a new dependency path requires M passes. Thus the total complexity per iteration is M encoding passes.

We now consider the 1-Pass adaptive case. Recall that the quantizer design is predicted using information from the previous encoded frame. Matching pursuit with the resulting quantizer is then used within the trellis optimization to extend paths from each node of the current stage into the next stage. Consider the specific node $(a3)$ for which path extension was illustrated in Figure 5.2. Because the quantizer is predicted, its design will depend on the path leading to node $(a3)$. Consequently, the path extension for each node of the trellis requires a single matching pursuit pass with the appropriate predicted quantizer design. This one pass covers all σ choices used for path extension of a single node. To extend all nodes in this manner requires a total of SM encoding passes and completes the trellis optimization. The sequence is then re-encoded using the newly found $\{\sigma'_{k,opt}\}$. The fixed predictors are thus updated at a cost of M additional encoding passes. The total cost per iteration is thus $(S + 1)M$ encoding passes.

For the 2-Pass adaptive case, quantizer design requires a single unquantized analysis pass, and each σ encoding requires a quantized completion pass using the new quantizer. During the previous iteration, all M frames were encoded using the 2-Pass design in order to update the fixed predictors. Unquantized analysis passes for all frames are thus available and need not be recomputed. Trellis optimization does require each frame to be encoded using C different quantizers, one for each choice of σ. A total of CM completion passes are thus required. This allows $\{\sigma'_{k,opt}\}$ for the current iteration to be found. The fixed predictors are then updated using matching pursuit with the 2-Pass quantizer design at a cost of $2M$ passes. The total complexity per iteration is thus $(C + 2)M$ encoding passes.

A summary of the resulting complexities is given in Table 5.2.

The above complexity analyses were mainly developed to allow us to compute complexity reduction factors due to the use of matching pursuit models instead of exact

Table 5.2: Exact (R, D) generation complexity for the optimization method of Hsu, et.al. [29]. The optimization is done over M frames using S quantized buffer fullness states. Each frame is coded with C choices of control parameter σ. Each complexity entry is the total number of encoding passes per dependency iteration required for the (R, D) farming task.

M.P. Quantizer Design	Complexity
Fixed	M
1-Pass Adaptive	$(S+1)M$
2-Pass Adaptive	$(C+2)M$

(R, D) generation. This analysis will be presented in Section 5.5.1. However, the compact summary given in Tables 5.1 and 5.2 also allows some comparison to be made between the various optimization methods and matching pursuit quantizer design types. Some comments should be made in this regard.

First, any comparison across optimization methods should be done with caution, since each method optimizes over a different frame grouping, and each uses a different level of approximation. For example, one might note from Table 5.1 that the method of Ramchandran, et.al. [54] is greater in complexity than that of Chen and Lin [8] by a factor of about $2C^2$. However, it is important to consider that the former method performs its optimization over a nine-frame grouping with five levels of dependency, and finds the true optimal solution over that window. The latter method considers a much smaller three-frame window, and derives from it the control parameter for only a single frame. Although less expensive, this sliding window approach does not find the globally optimal bit allocation, where the method of Ramchandran, et.al. actually does. Either method may be useful, depending on a user's complexity constraints and coding efficiency requirements.

We thus conclude that the results in Tables 5.1 and 5.2 cannot provide a "fair" exact complexity comparison across optimization methods. However, we can use the table results to roughly classify the optimization methods into two categories. The methods of Ramchandran, et.al. [54] and Chen and Lin [8] use full dependency tree expansions. This is an expensive way to handle inter-frame dependency, and produces the polynomial terms in C which are seen in the first two rows of Table 5.1. Such terms are absent for the other two methods, which avoid direct expansion of the dependent frames and are thus substantially less complex. Lin and Ortega [39] accomplish this by using a spline model to interpolate the effects of dependency, and Hsu, et.al. [29] handle dependency in an iterative manner. Neither method is guaranteed to find the optimal dependent solution, but the complexity of each is greatly reduced compared to full dependency tree expansion.

We may also use the results of Tables 5.1 and 5.2 to compare complexity across the different matching pursuit quantizer design methods. Such a comparison is fair in the sense that the design method is the only variable, that is the frame grouping and optimization method are held constant in the comparison. However, keep in mind that the adaptive designs have a significant coding efficiency advantage over a Fixed quantizer, as demonstrated in the previous chapter. Among the adaptive designs, the 2-Pass quantizer has a small advantage over the 1-Pass. Note also that our current comments apply only to the exact (R, D) generation case. The complexity situation will naturally change when models are applied.

For the dependency tree framework cases shown in Table 5.1, the Fixed and 1-Pass methods have identical complexities, and so there seems to be no advantage to selecting a Fixed quantizer over 1-Pass for the encoder to be optimized. Complexity for the 2-Pass case is substantially higher than that of the 1-Pass case, increasing by a factor of $(1 + C)$

for the first two methods and by a factor of 6.75 for the third method. Thus the 1-Pass case would be used where computational complexity is the main concern, and the 2-Pass case would be used where near-optimal coding efficiency is required.

The situation is a bit different for the method of Hsu, et.al. [29] as shown in Table 5.2. Here the cost per iteration with Fixed quantizer is M encoder passes, which translates to a single encoder pass per frame. This low cost results because the passes used by one iteration to fix the predictors are recycled and used to compute the (R, D) tradeoffs for the trellis optimization of the next iteration. The cost per iteration for the 1-Pass and 2-Pass adaptive designs are $(S + 1)$ and $(C + 2)$ encoder passes per frame. These costs will be comparable for typical choices of S and C, so there is little if any complexity advantage to choosing the 1-Pass design method over the 2-Pass.

We close this section by considering how the exact (R, D) generation process for matching pursuit differs from that of the DCT case. Rather than consider all four optimization methods for DCT, we keep the discussion simple by considering the case of a single source image which is to be encoded to C different choices of σ. Because the DCT is an orthogonal basis, quantization need not be done in-loop. Instead, the DCT transform may be computed once on the current frame, and each choice of σ (quantization step for the DCT case) then requires the coeficients to be requantized and VLC encoded. Transform computation of all σ choices for a single source image is thus shared, which is similar but not identical to the case of matching pursuit with a Fixed quantizer. The important differences are:

1. A matching pursuit encoder pass is typically more expensive than a DCT transform

of the same image [45]. This is expected, since matching pursuit typically uses a large redundant dictionary and thus must compute many more coeficients than DCT. However, fast matching pursuit methods have been introduced to narrow this gap (e.g. [15][55][12][18]). The topic of fast implementations will be treated in more detail in Chapter 6.

2. As mentioned earlier, matching pursuit rate and distortion may be progressively updated during the encoding pass. This costs only a few arithmetic operations and table lookups each time an atom is found. Many (R, D) points may thus be computed from a single encoding pass at very little additional cost. The requantize/recode operation used to generate each desired (R, D) point for the DCT case is much more expensive, since it cannot be computed progressively. The reason is that global requantization essentially changes all quantized coeficients, which then must be VLC encoded from scratch.

The first of these two items shows a specific complexity advantage for DCT, while the second item shows an advantage for matching pursuit. Our experience suggests that the effects of the first item will outweigh those of the second, although the exact complexities will be implementation dependent. We thus expect optimization for a Fixed quantizer matching pursuit system to have a greater complexity cost than that of a comparable DCT system. Recall that in Tables 5.1 and 5.2, the 2-Pass matching pursuit quantizer design was shown to be several times more expensive than a Fixed quantizer. This suggests that for matching pursuit with 2-Pass quantizer design, the optimization complexity will be many times greater than that of DCT. Rate-distortion modeling and the resulting complexity reductions are thus much more important for the matching pursuit case than for DCT. Given this motivation, we move on to the development of our first matching pursuit (R, D) model.

5.3 The Full Pre-Code Model

In this section, we develop our first model for matching pursuit rate distortion prediction. This is basically an analytical model for the effects of the in-loop quantizer. Because the model is based on an initial matching pursuit pass, it will be called the Full Pre-Code model. We now move on to a more specific description of the problem addressed by the model.

Assume that the current frame has been encoded once with some initial quantizer $Q(\cdot)$ and number of atoms N. The problem is to predict the rate distortion performance for encoding the same frame with a different quantizer $Q'(\cdot)$ and number of atoms $N' < N$. By solving this problem, we enable any number of rate-distortion points to be quickly estimated from a single analysis encoding pass. This is particularly useful in combination with the 2-Pass matching pursuit quantizer design, since for this case each choice of control parameter σ results in a unique quantizer.

Given the initial pass with quantizer $Q(\cdot)$, our primary goal will be to predict the overall rate distortion behavior (R, D) for an arbitrary second quantizer $Q'(\cdot)$. We first concentrate on modeling overall distortion D as a function of $Q'(\cdot)$ and N'. This distortion model is developed in Section 5.3.1. A model for the overall rate R is then shown in Section 5.3.2. An evaluation of the combined rate-distortion prediction model using sample

Figure 5.8: Data given by initial decomposition of current frame $\vec{f}$ using operating point $\{Q(\cdot), N\}$.

Pass 1 Moduli:	$\{p_i,$	$i = 1...N\}$
Pass 1 Quant Error:	$\{d_i = p_i - Q(p_i),$	$i = 1...N\}$
Overall Distortion:	$\{D_i = \|\vec{f_i}\|^2,$	$i = 0...N\}$

frames from across the MPEG-4 test set is presented in Section 5.3.3.

5.3.1 Distortion Model

We begin by formally stating the problem and introducing some notation. Suppose motion compensation has been done on the current frame to produce motion residual $\vec{f} = \vec{f_0}$. This residual image is to be encoded with matching pursuit in the usual way. Suppose an initial N atom encoding pass is performed using quantizer $Q(\cdot)$. At each stage i, an atom is encoded with basis function $\vec{b}_{\gamma_i}$ and modulus p_i. The modulus is quantized with distortion $d_i = p_i - Q(p_i)$, and the resulting atom is then subtracted from the remaining signal energy as:

$$\vec{f_i} = \vec{f}_{i-1} - Q(p_i)\vec{b}_{\gamma_i}$$

The first encoding pass thus provides us with ordered sets of modulus, quantization error and overall distortion. This given data is summarized as Figure 5.8.

Based on this set of Pass 1 knowledge, we seek to predict the overall distortion $\{D'_i, i = 0...N\}$ for a second encoding of $\vec{f}$ using a different quantizer $Q'(\cdot) \neq Q(\cdot)$. Additional Pass 2 parameters such as $\{p'_i, d'_i, i = 1...N\}$ will be predicted on the way to our main goal. To distinguish predicted Pass 2 values from their true counterparts, we adopt the hat notation (e.g. $\hat{p}'_i, \hat{d}'_i$, etc.).

Contributions to Distortion

Recall that in the previous chapter, matching pursuit was evaluated in terms of energy reduction, that is the amount of energy removed by the encoding of each atom. From earlier Equations (4.2) and (4.3), the single-stage energy reduction is:

$$\begin{aligned}\overline{ER}_i &\triangleq \|\vec{f}_{i-1}\|^2 - \|\vec{f}_i\|^2 \\ &= p_i^2 - d_i^2 \end{aligned} \tag{5.5}$$

and the cumulative energy reduction due to the first i atoms may thus be written as:

$$\begin{aligned}\overline{CR}_i &\triangleq \|\vec{f}_0\|^2 - \|\vec{f}_i\|^2 \\ &= \sum_{j=1}^{i} p_j^2 - \sum_{j=1}^{i} d_j^2 \end{aligned} \tag{5.6}$$

Knowing the energy reduction together with the initial energy of the frame is equivalent to knowing the overall distortion, since

$$D_i = D_0 - \overline{CR}_i \tag{5.7}$$

From Equation (5.6), the energy reduction may be naturally split into separate components from the modulus and quantization error terms. Define the p^2 and d^2 contributions to $\overline{CR}_i$ respectively as:

$$\rho_i \triangleq \sum_{j=1}^{i} p_j^2 \tag{5.8}$$

$$\delta_i \triangleq \sum_{j=1}^{i} d_j^2 \tag{5.9}$$

In order to predict Pass 2 distortion D', we will first model the changes in the p^2 and d^2 contributions which arise due to the change in quantizers from $Q(\cdot)$ to $Q'(\cdot)$. Define these contribution changes as:

$$\Delta\rho_i \stackrel{\triangle}{=} \rho_i' - \rho_i \tag{5.10}$$

$$\Delta\delta_i \stackrel{\triangle}{=} \delta_i' - \delta_i \tag{5.11}$$

where $\{\rho_i', \delta_i'\}$ are the contributions to $\overline{CR}_i'$, the Pass 2 energy reduction term. The distortion for Pass 2 may then be computed using

$$D_i' = D_i - \Delta\rho_i + \Delta\delta_i \tag{5.12}$$

Thus an accurate prediction of $\Delta\rho_i$ and $\Delta\delta_i$ will allow us to compute predicted Pass 2 distortion $\hat{D}_i'$ from known Pass 1 distortion D_i. Before describing the prediction methods, we illustrate the above definitions. Some sample waveforms are shown for a single frame of the Mobile sequence encoded at a constant 1 Mbit/s rate. The frame was encoded using two different quantizers. The first quantizer, $Q(\cdot)$, was defined using the two-pass quantizer design algorithm of the previous chapter using a pre-set ratio of $QP/DZ = 0.6$. The second quantizer, $Q'(\cdot)$, is the two-pass result with a ratio of 1.0. The second quantizer is thus more coarse, introducing more quantization error than the first. The energy reduction contributions ρ_i and δ_i for each of the two quantizers are plotted against atom number i in Figures 5.9a and 5.9b. Note that at full scale it is difficult to see a difference between the p^2 contributions ρ_i and ρ_i'. Changing the quantizer ratio does introduce a noticeable change in the d^2 contribution, however. The amount of these changes is shown in Figures 5.9c and 5.9d. From the vertical scales of these two plots, the d^2 contribution difference is seen to be about five times larger than the p^2 difference. This makes an accurate prediction of $\Delta\delta_i$ the most important task, with accuracy in $\Delta\rho_i$ desirable but of secondary importance. Figure 5.9e shows the overall distortion vs. atom number i for the two different quantizers.

For this plot, we zoom in on the final 600 atoms. This highlights an area of interest, since the Mobile sequence is not typically coded at lower bit rates. It also shows the maximum difference between the two distortion curves, since the curves tend to diverge as i increases.

Predicting d^2 Contribution

We begin by considering the quantization error contribution δ_i. Since the Pass 1 moduli are known, it is possible to approximate the Pass 2 quantization error by requantizing the p_i using the Pass 2 quantizer $Q'(\cdot)$. In doing so, we make the implicit assumption that the moduli are minimally affected by the change in quantizer, i.e.

$$p_i' \approx p_i \tag{5.13}$$

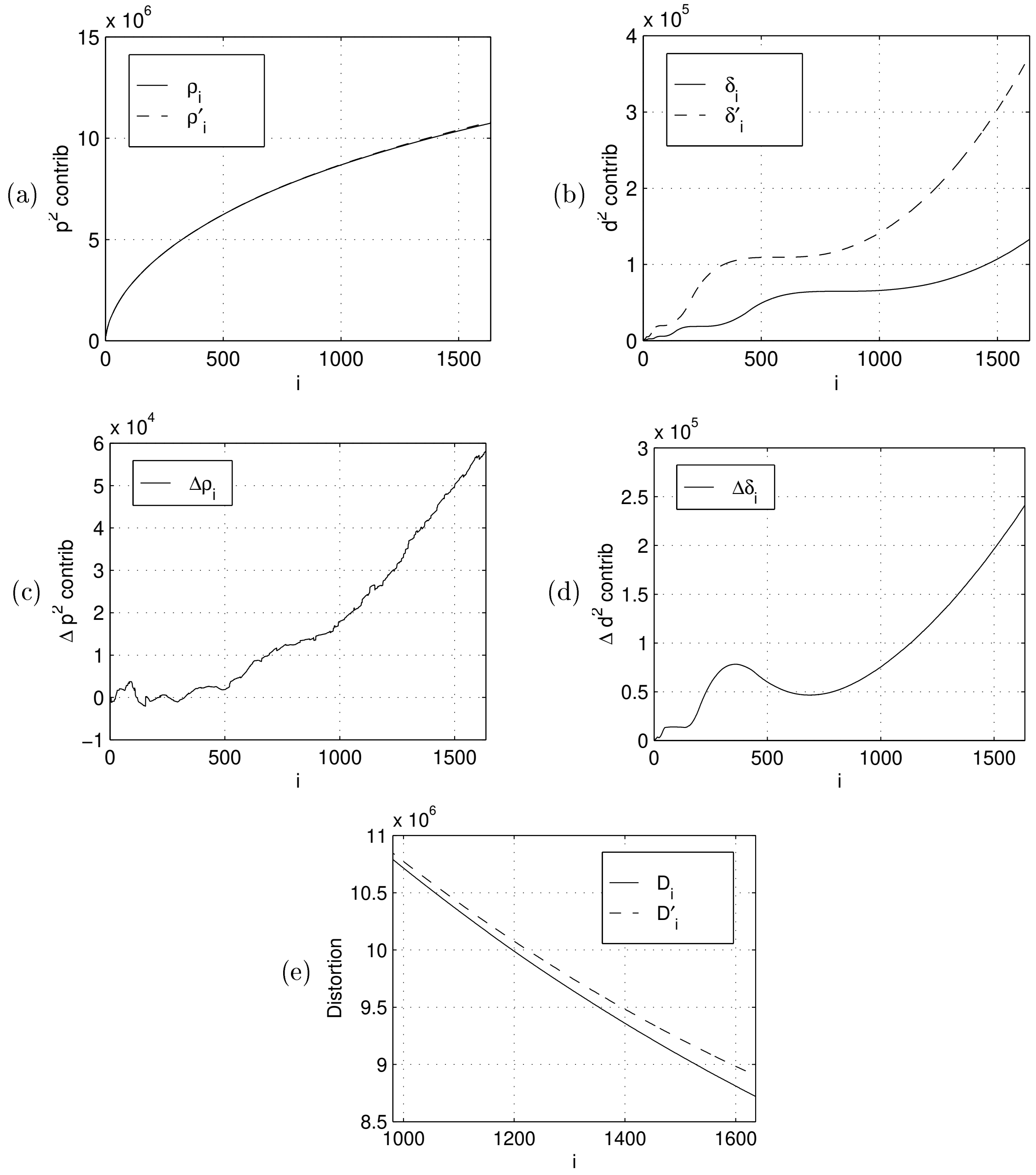
This leads to the following approximation of the quantization error:

$$\begin{aligned} d_i' &= p_i' - Q'(p_i') \\ &\approx p_i - Q'(p_i) \end{aligned} \tag{5.14}$$

Approximations (5.13) and (5.14) allow us to predict $\{p_i', d_i', i = 1...N\}$. We can then use the Pass 2 equivalents of Equations (5.8) and (5.9) to compute predicted values for the energy reduction contributions $\{\rho_i', \delta_i', i = 1...N\}$. Since the final goal is to predict the overall Pass 2 distortion D_i' using Equation (5.12), it is most useful to evaluate the prediction results by comparing predicted and real values of the contribution changes $\{\Delta\rho_i, \Delta\delta_i\}$, and ultimately comparing the predicted overall distortion to the real value D_i'.

In Figure 5.10, the prediction performance is illustrated for the same sample frame of Mobile as used in the previous illustration. Figure 5.10a shows the predicted and real

Figure 5.9: Illustration of distortion contributions for encoding a single frame of Mobile at approximately 1 Mbit/s. Quantizers $Q(\cdot)$ and $Q'(\cdot)$ have QP to DZ ratio values of 0.6 and 1.0 respectively. (a) p^2 distortion contributions ρ_i and ρ'_i. (b) d^2 distortion contributions δ_i and δ'_i. (c) $\Delta\rho_i = \rho'_i - \rho_i$. (d) $\Delta\delta_i = \delta'_i - \delta_i$. (e) Distortion D_i and D'_i resulting from quantizers $Q(\cdot)$ and $Q'(\cdot)$ respectively.



p^2 contribution differences. Our predicted p^2 contribution difference is forced to be zero, according to Approximation (5.13). The d^2 contribution difference is predicted fairly well, as shown in Figure 5.10b.[9] Using the predicted values of $\Delta\rho_i$ and $\Delta\delta_i$, the predicted value of distortion D'_i is computed and compared to the real Pass 2 and Pass 1 distortions in Figure 5.10c. It is seen that the distortion prediction is reasonable, but a noticeable difference between real and predicted distortion can be seen. We show this absolute prediction error in Figure 5.10d. The error is seen to be generally increasing with atom number i.

Figures 5.10e and 5.10f evaluate the relative error. Two types of relative error are considered. The first shows the prediction error relative to the gap between D' and D and is defined as:

$$e_{|D'-D|} = \frac{|D' - \hat{D}'|}{|D' - D|} \tag{5.15}$$

The second shows the prediction error relative to Pass 2 distortion D', which is the end quantity being predicted. This error measure is defined as:

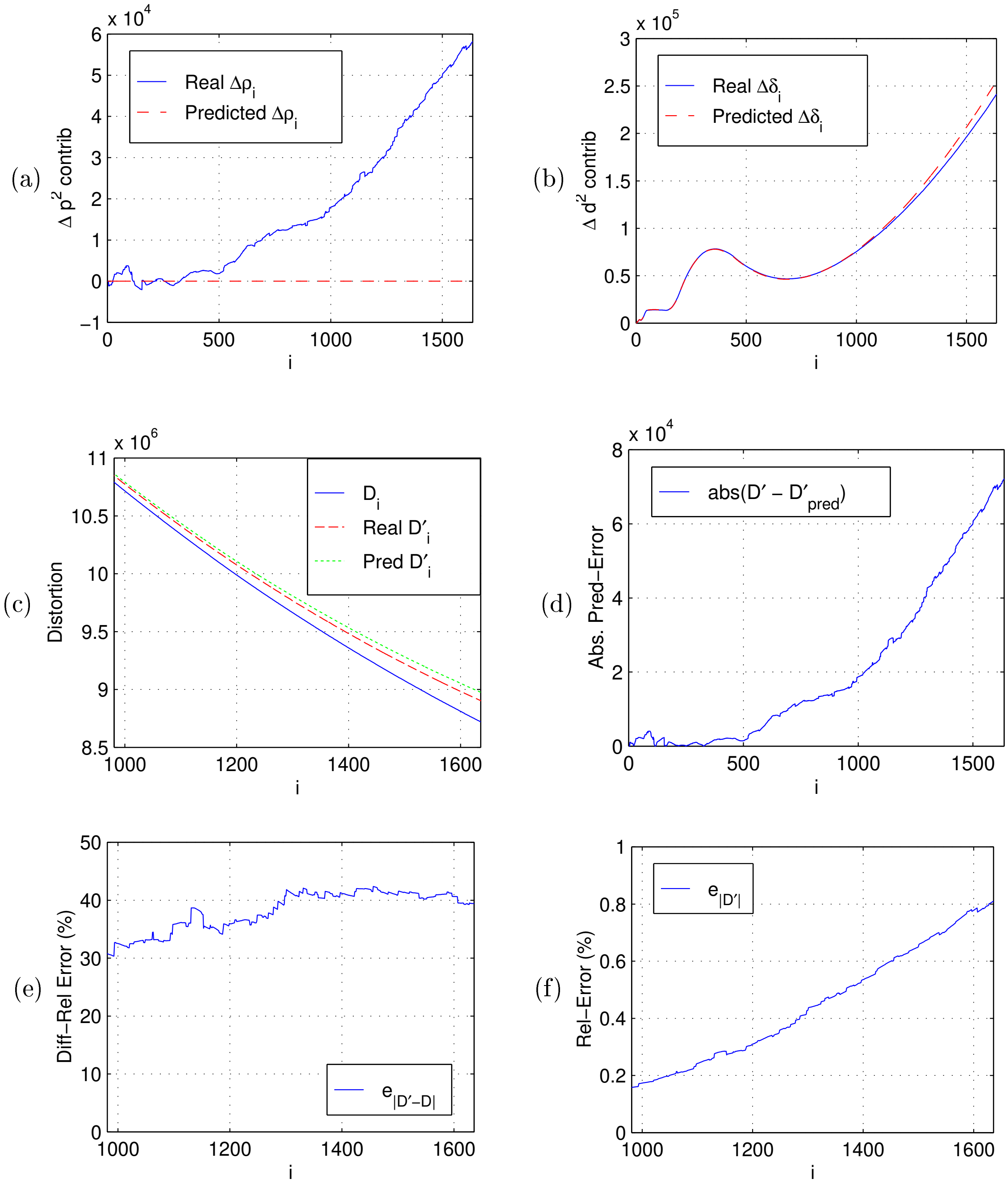
$$e_{|D'|} = \frac{|D' - \hat{D}'|}{|D'|} \tag{5.16}$$

In Figure 5.10e, we see that the simple distortion model does not provide a good prediction of the change in distortion, since $e_{|D'-D|}$ shows a relative error between 30 and 40 percent. The end prediction of D' has been improved, with values of $e_{|D'|}$ ranging from 0.2 to 0.8 percent. However, a steady increase in relative error with atom number can be seen on the plot.

The approximation in Equation (5.14) assumes that the p^2 contribution term ρ_i

[9] Figure 5.10b reveals some prediction error in $\Delta\delta_i$ toward the end of the encoding process, but this error will nearly vanish when the improved $\Delta\rho_i$ prediction is added in the next subsection. Because the predicted d'_i values come from quantization of the predicted p'_i values as in Equation (5.14), a more accurate estimate of p'_i in turn increases the accuracy of the predicted d'_i. See Figure 5.12b for a comparision.

Figure 5.10: Illustration of a simple distortion model with p^2 distortion contribution ignored. (a) Real and predicted change in p^2 contribution $\Delta\rho_i$. (b) Real and predicted change in d^2 contribution $\Delta\delta_i$. (c) Pass 1 distortion D_i compared to real and predicted values of Pass 2 distortion D_i'. (d) Absolute prediction error. (e) Percent differential relative prediction error, $e_{|D'-D|}$. (e) Percent Relative prediction error, $e_{|D'|}$.



does not change with the quantizer. This is clearly a major source of the prediction error seen in Figure 5.10. In the next section, this approximation will be replaced with a refined prediction for the Pass 2 modulus values. By more accurately predicting p'_i, we will show that our approximations of both contributions ρ'_i and δ'_i will be improved resulting in an improved prediction of overall distortion.

Predicting p^2 Contribution

The example of Figure 5.9c shows that the p^2 contribution to $\overline{CR}_i$ does change with the quantizer. This is because quantization is done in-loop, and so changing the quantizer affects the atoms chosen during the matching pursuit process. This change manifests itself as nonzero values of $\Delta\rho_i$, which can be seen in the figure. The change in p^2 contribution may be attributed to two sources. First, a change in the quantizer affects the accuracy of the coded moduli and thus changes the effectiveness of the matching pursuit process. For example, if $Q'(\cdot)$ introduces more quantization error than $Q(\cdot)$, then each Pass 2 atom is less effective at removing energy then the corresponding Pass 1 atom. The energy reduction process thus becomes slower. This results in an overall increase in the modulus values, since at a given stage i there will typically be more energy remaining in the progressive residual image $\vec{f}_i$, and so the computed inner product for that stage will typically be larger. Another way to state this is that if $D'_i > D_i$ on the average, then $|p'_i| > |p_i|$ will also be true on average. Larger $|p'_i|$ values will in turn result in an increase in the p^2 contribution to energy reduction, as suggested by Definition 5.8. This subsection attempts to model this change and thus improve our prediction of overall distortion D'_i.

A second and less predictable effect is also working to change the moduli for the second pass. Recall the definition of the *atom path* as the ordered list of coded atoms. Since the selection of atom i depends on all preceeding atoms, any change in the in-loop quantizer $Q(\cdot)$ used in one stage may change the atom selection result at any future stage. In effect, a change in the quantizer at stage i may cause the atom path to diverge beyond stage i. This affects all atom parameters, and in particular causes small random changes in the absolute modulus values.

Fortunately, the atom path effects on $\{p_i'^2, i = 1..N\}$ are typically much smaller than those due to more predictable quantization effects. Since the noise-like effects of atom path changes are complex functions of the signal $\vec{f}$ and the dictionary functions $\vec{b}_\gamma \in \mathcal{G}$, it is difficult to predict them without repeating the entire matching pursuit process using the new quantizer. This of course would be a trivial solution, but would be computationally very expensive.

For our system, we ignore the more subtle atom path changes and instead concentrate on modeling the change in $\{p_i', i = 1...N\}$ due to the expected change in the remaining energy for each stage. This energy change is related to all the previously found modulus values and their resulting quantization errors. Suppose that the energy remaining after a particular stage of Pass 1 is D_i, and we use the change in quantizer to predict the distortion at the corresponding atom stage in Pass 2. Call this predicted distortion $\hat{D}_i'$. Given the known Pass 1 modulus p_i, we propose to predict the Pass 2 modulus based on the ratio between the sum-squared energies from the two passes:

$$\hat{p}_i' = p_i \sqrt{\frac{\hat{D}_{i-1}'}{D_{i-1}}} \tag{5.17}$$

This heuristic energy correction is based on the empirical observation that the energy removed by a coded atom (p_i^2 if quantization is neglected) is roughly proportional to the remaining signal energy $D_{i-1} = ||\vec{f}_{i-1}||^2$. Although the relationship is not exact, we will show that adoption of this correction will improve our overall distortion model.

The effect of the change in energy on modulus and vice versa are cumulative. In other words, a change in the remaining energy at one stage will affect the modulus value of the next stage. This in turn affects both the quantization error and the expected energy of the next stage. For this reason, we propose to compute these effects on a per atom basis. For each stage, we predict the current modulus $\hat{p}_i'$ as a rescaled version of p_i according to Equation (5.17). We requantize this predicted modulus to approximate the quantization error as $\hat{d}_i'$. By substituting predicted values $\{\hat{p}_i', \hat{d}_i'\}$ into the single-stage energy reduction relation (5.5), we estimate the overall distortion for the next stage. This predicted value $\hat{D}_i'$ is in turn used to rescale the predicted modulus at the next stage. In this way we are able to predict both absolute moduli and overall distortion without the cost of performing a matching pursuit decomposition using the new quantizer. The resulting distortion model algorithm is detailed in Figure 5.11.

We apply this algorithm to predicting the p^2 and d^2 distortion contributions for our sample frame of Mobile. The result is shown in Figure 5.12. Both contributions are now modeled, as opposed to our earlier result in Figure 5.10 in which only changes in the d^2 contribution were modeled. The improvement in p^2 contribution prediction is obvious in comparing Figure 5.12(a) to Figure 5.10(a). The newly refined model captures the main trend of the p^2 contribution change, however it fails to track the more random noisy component due to atom path change. One interesting side effect of a better p^2 contribution

Figure 5.11: Summary of the Full Pre-Code distortion model.

Given: p_i, d_i, D_i from Pass 1 with $Q(\cdot)$
Predict: p'_i, d'_i, D'_i for Pass 2 with $Q'(\cdot)$

for $i = 1{:}N$ /* For each atom stage */
    if $(i == 1)$ /* Predict modulus */
        $\hat{p}'_i = p_i$
    else
        $\hat{p}'_i = p_i \sqrt{\frac{\hat{D}'_{i-1}}{D_{i-1}}}$
    endif

    $\hat{d}'_i = \hat{p}'_i - Q'(\hat{p}'_i)$ /* Predict Q-error */

    $\hat{D}'_i = \hat{D}'_{i-1} - (\hat{p}'_i)^2 + (\hat{d}'_i)^2$ /* and Distortion */
end

model is that a more accurate prediction of the p'_i values also improves the prediction of d'_i. This happens because the d'_i are computed through requantization of the predicted p'_i values.

The plot for overall distortion prediction is shown in Figure 5.12c. This shows that the distortion prediction has improved over the simpler model which was shown in Figure 5.10c. To highlight the improvement, compare the absolute and relative error plots in Figures 5.12d-f to their counterparts in Figure 5.10. Such a comparison shows that the absolute error of Figure 5.10d does not generally increase with atom number, as was the case for the simpler model. Overall, the full distortion model reduces this error by about a factor of four. Relative error measures $e_{|D'-D|}$ and $e_{|D'|}$ have also improved by about a factor of four. We conclude that the prediction of the change in distortion $(D' - D)$ and also the prediction of the final distortion D' are greatly improved with the addition of a p^2

contribution model.

This illustration using a frame of the Mobile sequence has served to illustrate the Full Pre-Code distortion model and show the type of improvement that can be expected. Additional plots and numeric data will be presented in Section 5.3.3 for a more thorough evaluation of prediction performance.

5.3.2 Rate Model

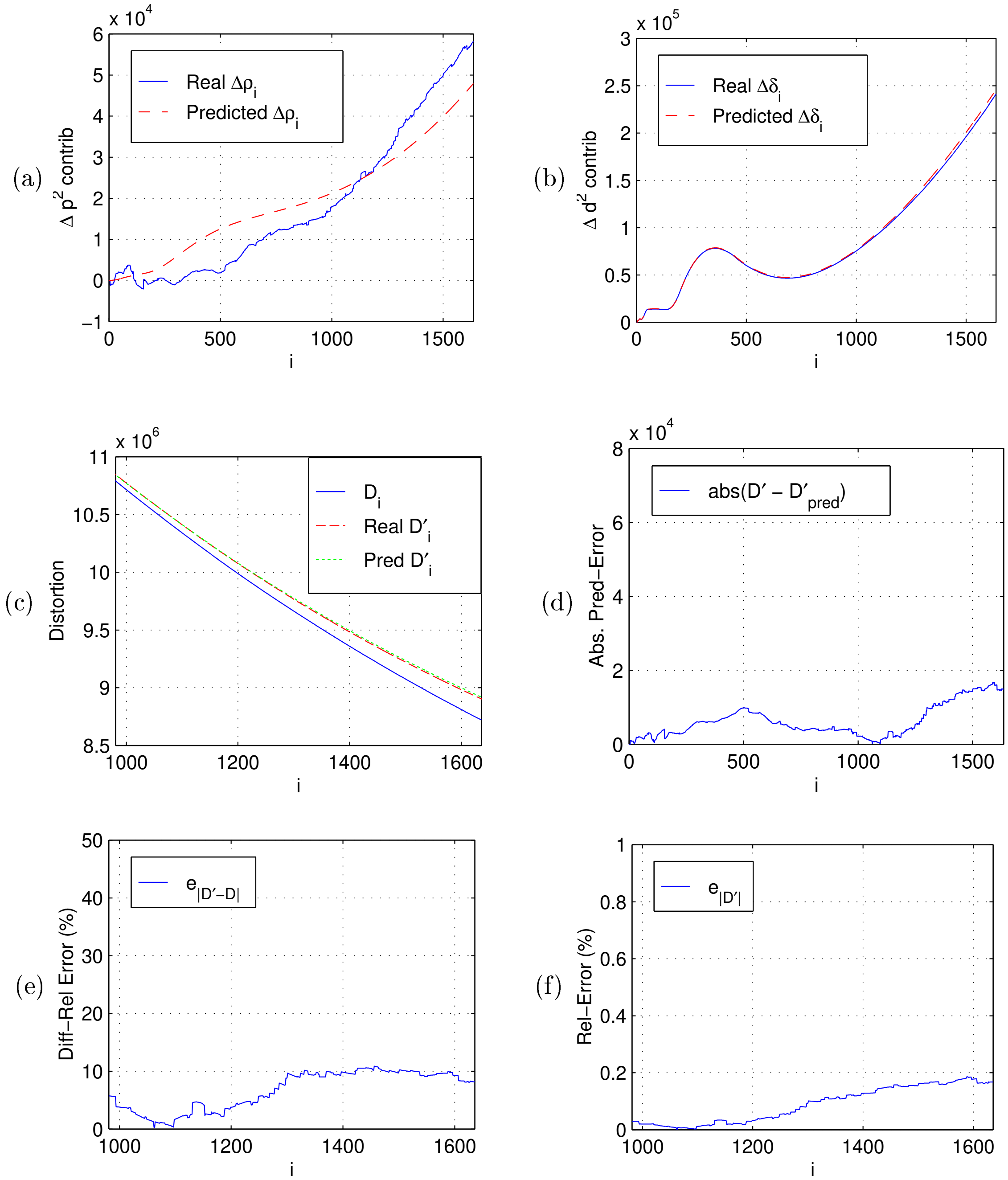
We now present a rate model to complement the Full Pre-Code distortion model of the previous subsection. Before proceeding, we note that the accurate rate-targeting methods intrinsic to matching pursuit with each quantizer design may obviate the need for such a model when solving the optimal bit allocation problem of Formulation 5.2. The reason is that target rate points are used for the control parameters. Within the optimization context, it is thus logical to model the rate as

$$R_k \approx \hat{R}_k = \sigma_k$$

where σ_k is the target rate to which frame k is encoded. Prediction accuracy for this case is thus equivalent to the intrinsic targeting accuracy for matching pursuit rate control, and high rate-targeting accuracy has been demonstrated. For example, bit rate targeting errors of about 1 percent were shown for the simple rate control mechanism introduced in Chapter 2.

Although the optimization problem of Formulation 5.2 can potentially be solved using the rate targeting approximation shown above, the quality of the solution will necessarily depend on model accuracy. Thus a rate model which predicts R_k with less than

Figure 5.12: Illustration of the full distortion model with both p^2 and d^2 distortion contributions. (a) Real and predicted change in p^2 contribution $\Delta\rho_i$. (b) Real and predicted change in d^2 contribution $\Delta\delta_i$. (c) Pass 1 distortion D_i compared to real and predicted values of Pass 2 distortion D'_i. (d) Absolute prediction error. (e) Percent differential relative prediction error, $e_{|D'-D|}$. (e) Percent relative prediction error, $e_{|D'|}$.



a 1 percent average error would be an improvement over the simple rate-targeting model. For this reason and for completeness, we now develop a Full Pre-Code rate model. The model is conceptually much simpler than the corresponding distortion model, and typical rate prediction errors of less than 1 percent will be shown.

Using the notation of the previous chapter, R' is the total bit rate used to encode the frame, and is the sum of bits to header, motion, and atoms:

$$R' = R'_{hdr} + R'_{mot} + R'_{atm} \tag{5.18}$$

The prediction task falls entirely to predicting R'_{atm}, since the bit budgets to header and motion vectors do not depend on the quantizer and are known at the start of atom coding.

The atom budget for the current frame is further divided into bits to modulus quantization, atom position, and basis indices as:

$$R'_{atm} = R'_Q + R'_P + R'_I \tag{5.19}$$

The rate due to quantization may be predicted in a straightforward manner. Recall that within the full distortion model of Figure 5.11, the Pass 2 modulus values are predicted and requantized using $Q'(\cdot)$. By simply VLC coding the predicted quantizer indices, we can predict the Pass 2 quantizer rate R'_Q. Call this predicted rate $\hat{R}'_Q$.

To approximate the overall rate R' from the quantizer rate, we note that the effect on overall bit rate due to a change in the quantizer is seen almost entirely in the rate to quantization. This was shown in Figure 4.12 of the previous chapter. Recall that for a constant number of atoms N, $(R_{atm} - R_Q)$ was shown to remain approximately constant as quantizer $Q(\cdot)$ varies. Since the rate variation is almost entirely expressed in R_Q, we

Figure 5.13: Summary of the Full Pre-Code rate model.

`Given:`	p_i, d_i, D_i `from Pass 1 with` $Q(\cdot)$ `and all Pass 1 rate information.`
`Predict:`	R'_i `for Pass 2 with` $Q'(\cdot)$

`For any` $N' <= N$

1. `Run full distortion model to predict quantized Pass 2 moduli.`
2. `Encode these moduli to predict quantizer rate as` $\hat{R}'_Q$.
3. `Predict atom rate using` $\hat{R}'_{atm} = R_{atm} - R_Q + \hat{R}'_Q$.
4. `Predict overall rate from` $\hat{R}' = R_{hdr} + R_{mot} + \hat{R}'_{atm}$.

approximate the Pass 2 bit rate to atoms as:

$$\hat{R}'_{atm} = R_{atm} - R_Q + \hat{R}'_Q \tag{5.20}$$

The predicted overall bit rate $\hat{R}'$ for the frame is then easily computed from the predicted atom budget and the known budgets to header and motion vectors, as in Equation (5.18). A summary of this rate prediction algorithm is shown in Figure 5.13. Experimental results for this model will be shown together with results from the earlier distortion model in the next section.

5.3.3 Prediction Results

In this section, we present additional results for the distortion and rate prediction models described in Figures 5.11 and 5.13. Sample frames from the MPEG-4 test set are encoded using an initial quantizer, and then distortion and rate for additional quantizers are predicted using the models. Plots and tables are presented to compare the predicted values to the true result and thus allow evaluation of the prediction accuracy. Each sample quantizer will be defined by the stepsize to dead-zone ratio as used in the previous chapter.

For example, a quantizer with $RATIO = 0.6$ is defined for the current frame by running the 2-pass quantization algorithm of Figure 4.14 on that frame to obtain the QP and DZ values with the desired ratio. By using a constant ratio to define the quantizer, we put the prediction methods into the framework of the previous chapter. However, it is important to notice that the prediction methods do not rely on this method of quantizer design, and could also be used with arbitrary quantizers $Q(\cdot)$ and $Q'(\cdot)$.

Three subsections are presented. The first shows results for the distortion prediction model, the second shows rate prediction results, and the third evaluates joint prediction of rate and distortion.

Distortion Prediction Results

Figure 5.14 shows distortion prediction results for sample frames from three of the standard MPEG-4 sequences. The three rows of plots correspond respectively to Mother QCIF at 24 kbit/s, Coast CIF at 112 kbit/s, and Stefan SIF at 1 Mbit/s. Each frame is coded with a reference quantizer $Q(\cdot)$ with ratio 0.6, and prediction is done for a second quantizer $Q'(\cdot)$ with ratio 1.0. For each of these sequences, three plots are shown. The first two plots for each sequence show a comparison of the contribution changes $\Delta\rho_i$ and $\Delta\delta_i$. In each plot, the real contribution change is compared to that predicted by the full distortion model. In each case, the change in the d^2 contribution is predicted very well, however the more erratic behavior of the p^2 prediction change is less successfully predicted. This is particularly true for the sample Stefan frame. In the lower left plot of Figure 5.14, the predicted p^2 contribution change is positive. This is what we would expect when moving from a finer to a coarser quantizer; since distortion should increase for the same number of atoms i, we expect that on average $|p_i'| > |p_i|$. However, in the real encoding with coarser quantizer $Q'(\cdot)$, the p^2 contribution change begins and remains negative for the first thousand atoms. This indicates that for this region $|p_i'| < |p_i|$ is true on the average. This is likely an example of the atom path effects described earlier. Toward the end of encoding, the real p^2 contribution change does turn positive again, and nearly catches up to the predicted values.

The rightmost plots in Figure 5.14 show a comparison of the reference pass distortion D_i and the real and predicted distortions D_i' for the second quantizer $Q'(\cdot)$. Note that in each of the three cases, the predicted distortion matches the true distortion extremely well. A closer look at this error is provided in Figure 5.15, which plots relative prediction error in distortion for the same three sequences. Figure 5.15a shows the prediction error relative to the change in distortion, $|D' - D|$. This error is seen to be 5 percent or less for the first two sequences. Larger relative errors are seen early in the encoding of Stefan, but the prediction is seen to improve as additional atoms are coded. The error relative to absolute distortion $|D'|$ is shown in Figure 5.15b. These plots show that the overall prediction error is extremely low, ranging from 0.0 to 0.3 percent of the actual distortion value.

For a more quantitative analysis, Table 5.3 shows distortion prediction error for a wider range of sequences and sample quantizers. Consider the first three prediction error columns of the table. These were produced by encoding a sample frame from each of the standard MPEG-4 sequences is using three different pairs of quantizers $\{Q(\cdot), Q'(\cdot)\}$, each pair showing a progressively larger change in stepsize to deadzone ratio. The distortion for the second quantizer is then predicted from the encoding pass made with the first quantizer.

For this table, the prediction error is measured in terms of PSNR. Each value in the table represents the difference between the predicted and true values of the PSNR of the encoded frame, with PSNR computed from mean square distortion D as:

$$PSNR = 10 \log_{10}(255^2/D)$$

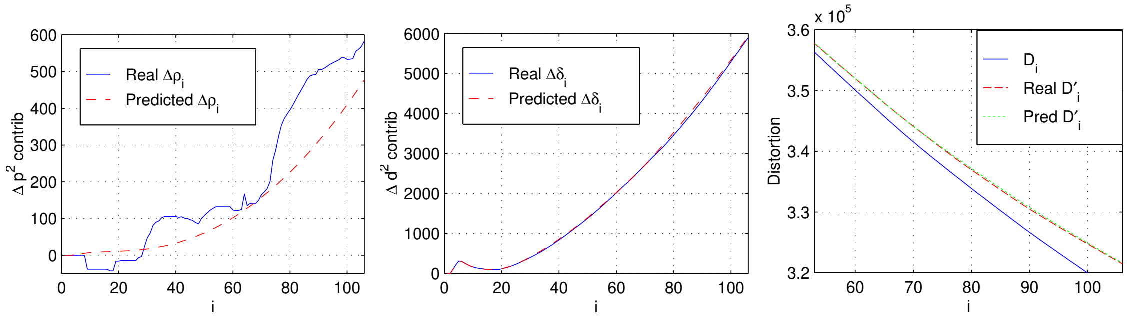
The smallest quantizer ratio change is $0.8 \rightarrow 1.2$ as shown by the first of the PSNR Error columns. For this quantizer change, the worst case error is .015 dB for Container at 24 kbit/s, and all other sequences have prediction errors of less than .01 dB. For the larger quantizer change of $0.4 \rightarrow 1.6$, the maximum prediction error is 0.0304 dB, and eight of the fourteen sequences have prediction errors less than .01 dB. Average absolute errors taken across all sequences are shown at the bottom of the table. These range from .005 to .01 dB, depending on the size of the quantizer change.

The final column of Table 5.3 demonstrates Full Pre-Code prediction using infinitely fine quantization for the first pass. This is equivalent to choosing initial quantizer $Q(\cdot)$ to have $RATIO = 0.0$, as annotated at the top of the column. Note that this is a perfectly valid case for the Full Pre-Code distortion model, since no restrictions on $Q(\cdot)$ were assumed in the development of the model. The table data shows that accurate prediction of the distortion for the $RATIO = 1.0$ case may be made from an unquantized pre-coding pass. This special case will be shown to be useful for some of the model applications which will be discussed in Section 5.5.1.

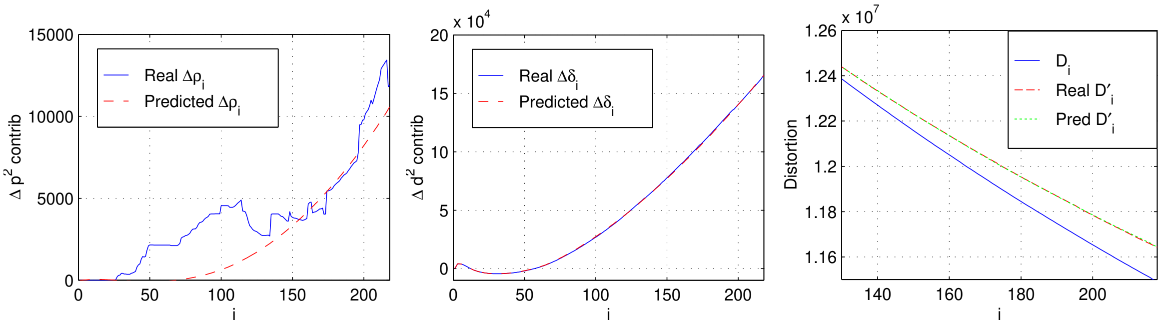
We thus conclude that our overall distortion model gives an extremely accurate prediction for D_i', even when the change in quantizer ratio is large. We now proceed to evaluate prediction results for the Full Pre-Code rate model.

Figure 5.14: Distortion prediction results for sample frames from three sequences with quantizer ratio change 0.6 → 1.0.

Mother at 24 kbit/s:



Coast Guard at 112 kbit/s:



Stefan at 1 Mbit/s:

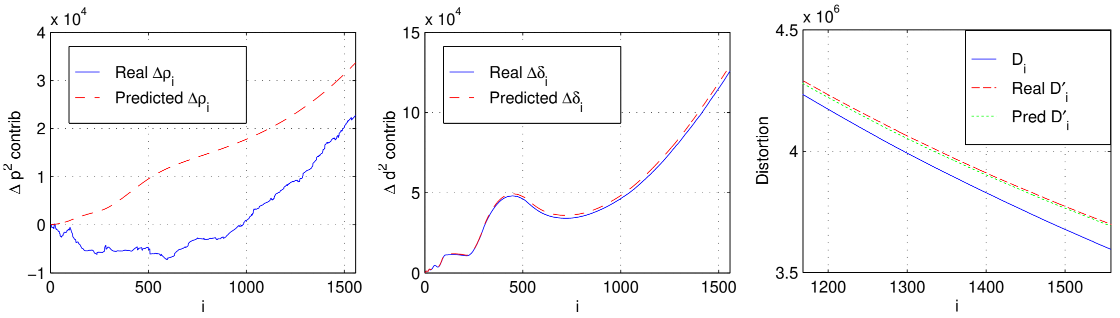


Figure 5.15: Percentage relative error for the prediction experiments of the previous figure. (a) Error relative to the distortion difference $|D' - D|$. (b) Error relative to absolute distortion $|D'|$.

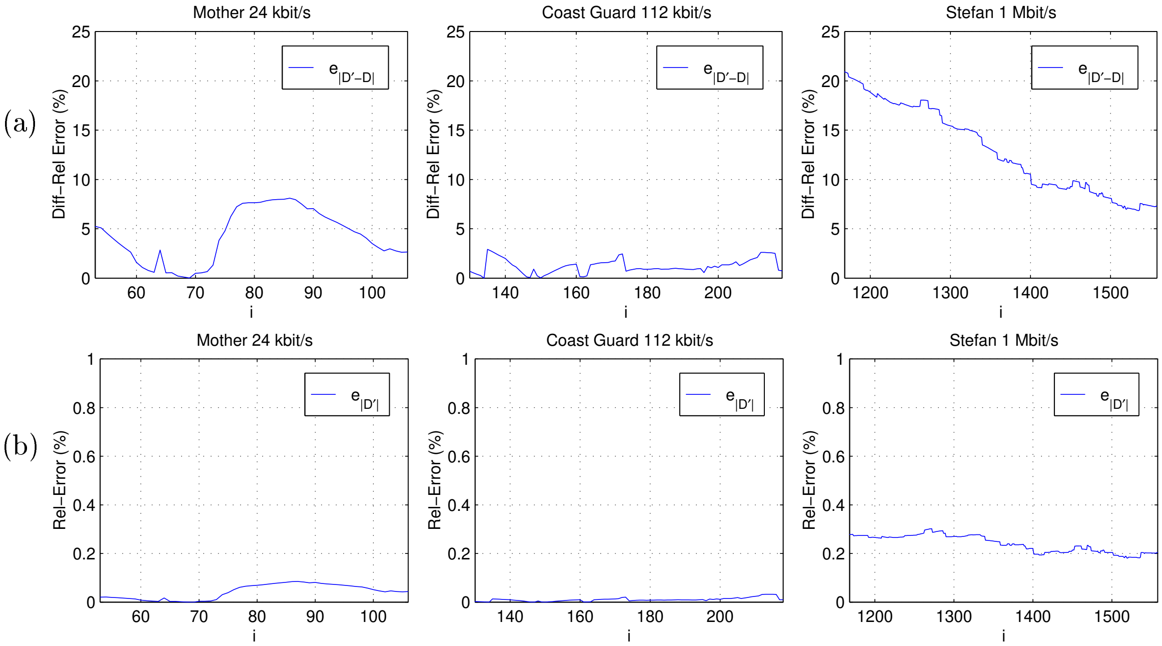


Table 5.3: Table of distortion prediction results for sample frames from each of the standard MPEG-4 test sequences.

Seq	Label	Atoms	Prediction error in PSNR (dB) for these RATIO changes:			
			$0.8 \rightarrow 1.2$	$0.6 \rightarrow 1.4$	$0.4 \rightarrow 1.6$	$0.0 \rightarrow 1.0$
1	Cont-10	54	0.0032	0.0043	0.0052	0.0050
2	Hall-10	54	-0.0009	-0.0092	-0.0116	0.0011
3	Mom-10	51	-0.0040	-0.0063	-0.0103	0.0004
4	Cont-24	114	-0.0150	0.0123	0.0068	0.0076
5	Silent-24	106	0.0010	-0.0058	-0.0016	0.0010
6	Mom-24	105	-0.0014	-0.0010	0.0091	0.0074
7	Coast-48	206	0.0086	0.0032	-0.0026	0.0071
8	Fore-48	138	0.0079	-0.0063	0.0228	0.0254
9	News-48	180	0.0065	-0.0136	-0.0062	0.0018
10	Coast-112	217	-0.0013	0.0003	0.0000	-0.0002
11	Fore-112	64	0.0033	0.0010	0.0020	-0.0058
12	News-112	238	0.0089	0.0189	0.0195	0.0147
13	Mob-1024	1606	0.0080	0.0087	0.0170	0.0052
14	Stef-1024	1520	0.0054	0.0225	0.0304	0.0045
Average absolute error (dB):			0.0054	0.0081	0.0104	0.0062

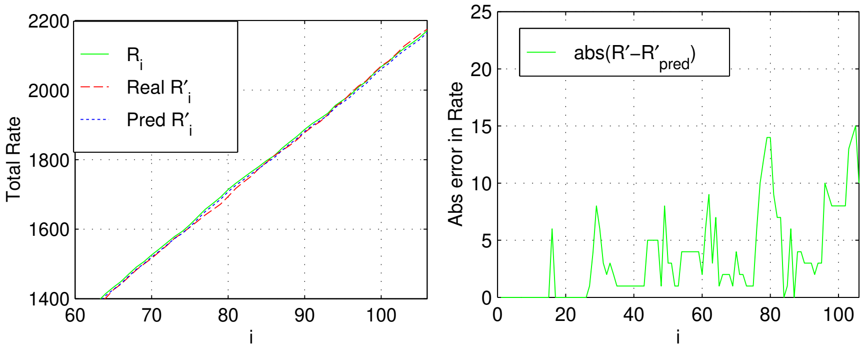
Rate Prediction Results

The plots of Figure 5.16 evaluate the accuracy of the rate prediction model for three sample frames from the MPEG-4 set. The sequences are the same three as were shown in Figure 5.14, and the same quantizer change of $0.6 \rightarrow 1.0$ is used. The left column of plots shows the overall rate used to encode the first i atoms. Note that only bits to atoms are shown in the plots; motion and header budgets, which are not affected by the quantizer change, are ignored. In each plot, the rate R_i used for the reference quantizer $Q(\cdot)$ is compared to the real and predicted versions of R_i', the rate obtained with the second quantizer $Q'(\cdot)$. For the two lower bit rate sequences, the rate change due to the quantizer is very small, and so all three lines coincide. The rate change for the Stefan 1Mbit/s frame is much larger for the same quantizer change, and here we see that the prediction closely matches the true value of R_i'. For each of the three cases, the absolute prediction error in rate is shown in the right column of plots. For the lower two bit rates, the error is very small, remaining below 15 bits throughout the entire encoding process. This shows an extremely accurate prediction, since the cost of a single atom is generally in the 20 bit range. For the higher rate Stefan run with its greater observed bit rate change, the prediction error is less than 20 bits for the first thousand coded atoms, and has increased to around 60 bits by the end of the approximately 1500 coded atoms. This is again a very accurate rate prediction, falling always within three atoms worth of bit budget. Expressed as a percentage of total bit rate, the prediction error is about 0.21 percent.

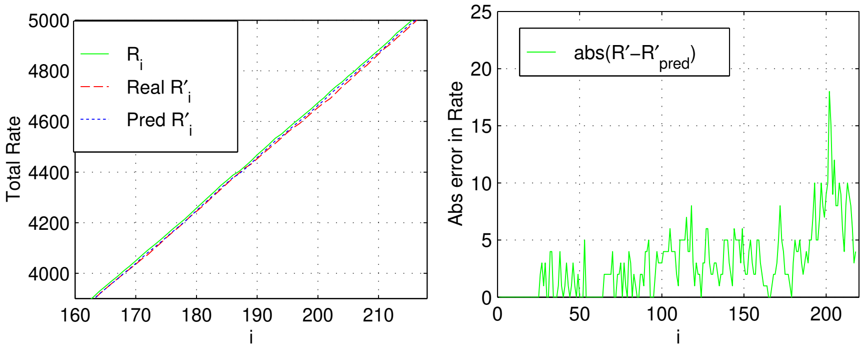
Numerical data for a wider range of sequences and quantizer pairs is presented in Table 5.4. As in the distortion results table, a sample frame from each of the fourteen

Figure 5.16: Rate prediction results for sample frames from three sequences with quantizer ratio change 0.6 → 1.0. Rates shown are for encoding the first i atoms, with motion and header bit rates ignored.

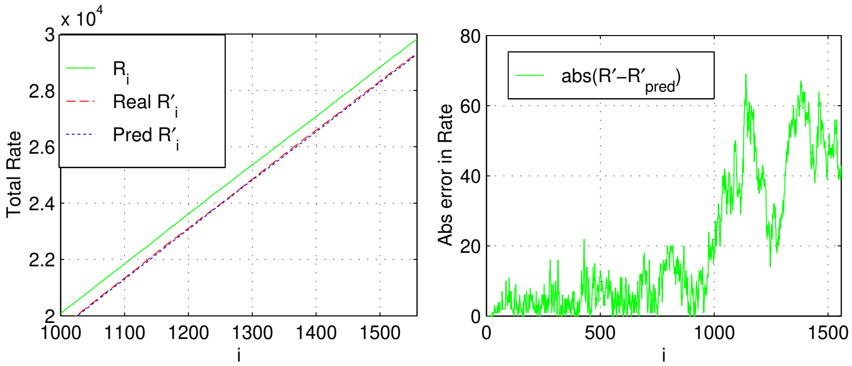
Mother at 24 kbit/s:



Coast Guard at 112 kbit/s:



Stefan at 1 Mbit/s:



MPEG-4 sequences is encoded using three quantizer pairs, each with a progressively larger change in stepsize to dead-zone ratio. For each sequence and quantizer pair, the percentage error in predicted rate is shown. Looking at the results, we see that the errors are extremely small. Of the 42 reported trials, only three cases show a prediction error above 1%. The largest such error is 1.62% for Container at 10 kbit/s with the largest quantizer ratio change, $0.4 \to 1.6$. Even for this quantizer pair, only two of 14 sequences show greater than a 1% error in predicted rate. The average error across all sequences ranges from .28 to .70 percent, depending on the extent of the quantizer change. Thus the prediction error due to the Full Pre-Code rate model is typically below 1%, even for large changes in quantizer.

Note that the numbers are computed for rate to atoms only, ignoring the motion and header bits. Since these additional bits do not change with the quantizer, including them in the rate prediction results would slightly reduce the reported error percentages.

Combined Rate Distortion Results

By combining the distortion and rate models, we show some combined (R, D) prediction results for four sample frames using a much wider range of quantizer changes. We also show prediction results for quantizer rate and distortion (r, d). Recall from the previous chapter that the quantizer rate r is the average number of bits spent to encode each modulus value, and quantizer distortion is:

$$d = mean((p_i - Q(p_i))^2) = mean(d_i^2)$$

The quantizer values (r, d) are normally hidden from the end user, and so the overall bit rate R and sum-squared distortion D of the coded frame are typically of greater interest.

Table 5.4: Table of rate prediction results for sample frames from each of the standard MPEG-4 test sequences. Percent error compares predicted rate $\hat{R}'_{atm}$ to actual rate R'_{atm}.

			Percent error in rate to atoms for these RATIO changes:		
Seq	Label	Atoms	$0.8 \rightarrow 1.2$	$0.6 \rightarrow 1.4$	$0.4 \rightarrow 1.6$
1	Cont-10	54	0.8921	0.7162	1.6245
2	Hall-10	54	0.0855	0.1708	0.8547
3	Mom-10	51	0.0850	0.0853	0.9442
4	Cont-24	114	0.2779	1.1210	0.7907
5	Silent-24	106	0.4470	0.4466	0.9870
6	Mom-24	105	0.6961	0.8784	1.1537
7	Coast-48	206	0.6286	0.0724	0.7029
8	Fore-48	138	0.0357	0.0718	0.4697
9	News-48	180	0.1015	0.1524	0.3313
10	Coast-112	217	0.5606	0.6222	0.5017
11	Fore-112	64	0.0570	0.3432	0.5166
12	News-112	238	0.0000	0.1604	0.7025
13	Mob-1024	1520	0.0070	0.1022	0.2188
14	Stef-1024	1606	0.0652	0.1545	0.0654
Average percent error:			0.2814	0.3641	0.7046

However, internal models of r and d are produced within the Full Pre-Code model, and it is illustrative to view the accuracy of these representations. We will show that the internal (r, d) models are extremely accurate, and thus Full Pre-Code is based on a good analytical model for the quantization effects.

The four frames examined here are taken from Silent QCIF at 24 kbit/s, News CIF at 112 kbit/s, Mobile SIF at 1 Mbit/s, and Stefan SIF at 3 Mbit/s. In order to effectively show a wide range of predicted quantizers, the following technique is used. The residual frame in question is first coded using reference quantizer $Q(\cdot)$, defined using the 2-Pass quantizer design algorithm of the previous chapter with stepsize to dead-zone ratio of 1.0. The full prediction model is then run using fourteen additional quantizer ratios from the set:

$$RATIO \in \{.4, .5, .6, .7, .8, .9, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8\}$$

The actual encoding runs using these $RATIO$ values are also performed for comparison to the prediction results. All runs are coded to a common target bit rate, however the prediction comparison is carried out using a constant number of atoms, N_{min}, taken as the minimum number of atoms produced in any of the above runs. All runs are truncated to N_{min} atoms, and the Full Pre-Code model is used to predict the rate and distortion at this common stopping point. The predicted results for changing from the reference quantizer to each of the additional quantizers defined by the above set is then compactly displayed on a single plot of rate or distortion vs. $RATIO$. Plots of rate vs. distortion are also shown.

We start with quantizer rate and distortion results (r, d), as shown in Figure 5.17. Each row of plots represents one of the four sample sequences. For each sequence, plots are shown for rate r vs. quantizer ratio, distortion d vs. ratio, and finaly r vs. d. Performance using the reference quantizer is shown as (*) in the plot. Real and predicted rate and distortion points are shown as (o) and (+) respectively. It is important to note that the performance points for each sequence are shown for a constant number of atoms, regardless of the chosen quantizer. The target bit rate is actually matched for the lowest tested ratio value 0.4, with the resulting minimum number of atoms used as the stopping point for encoding the remaining prediction quantizers.

Across the four tested sequences, the predicted and real points match very well, suggesting almost perfect prediction of quantizer (r, d). Slight prediction errors are visible in two cases, prediction of rate for News at 112 kbit/s, and prediction of distortion for

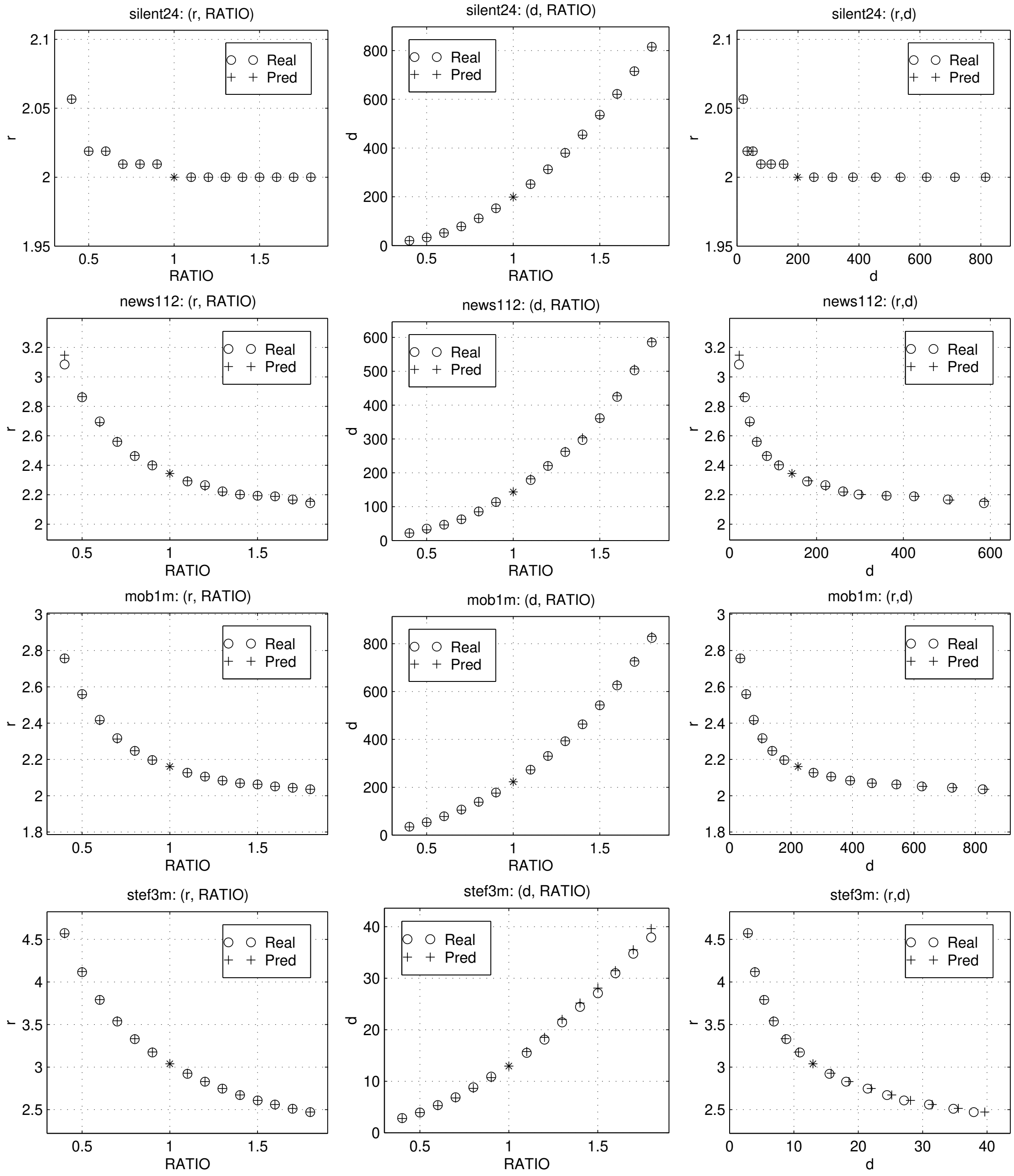
Stefan at 3 Mbit/s. Plots of r vs. d, shown in the last column of the figure, show very accurate prediction of most (r, d) points.

Overall rate R and distortion D are examined in a similar fashion with results shown in Figure 5.18. The same four sample sequences are used with the same reference and prediction quantizers under the same conditions. The first column of plots shows overall rate R plotted against prediction quantizer ratio. By scanning down the column, prediction of overall rate appears to be poor at low rates, but steadily improves as the rate increases. However, closer inspection reveals that the seemingly poor low bit rate performance is really an artifact of scale. At low bit rates, the overall rate R is relatively insensitive to the quantizer change, which is reflected by the nearly flat rate vs. ratio curve predicted for the Silent 24 kbit/s case. Atom path effects produce small deviations from the predicted rate on the order of ten to twenty bits, or about the cost of a single atom. The absolute error is thus small, but appears significant due to the small vertical scale of the graph. The corresponding error as a percentage of total bit rate is at most 1%.

The second column of plots in the Figure 5.18 shows overall distortion D plotted against prediction quantizer ratio. The model appears to track the true distortion very well, with some prediction error seen at the low end of News at 112 kbit/s. A good match is seen on nearly every point of the remaining three sample sequences.

Finally, overall rate R is plotted against distortion D in the rightmost column of plots in Figure 5.18. The rate prediction effects noted earlier can be seen to cause some mismatch in the two lower rate sequences, especially for Silent at 24 kbit/s. At higher bit rates, however, the combined prediction of R, D points is shown to work extremely well.

Figure 5.17: Quantizer rate and distortion (r, d) performance for four sample frames over a wide range of prediction quantizers $Q'(\cdot)$. In each case, the reference run uses quantizer $Q(\cdot)$ defined using $RATIO = 1.0$. The (r, d) point for the reference run is shown as (*) in each plot.



This is illustrated by the plots shown for Mobile at 1 Mbit/s and Stefan at 3 Mbit/s.

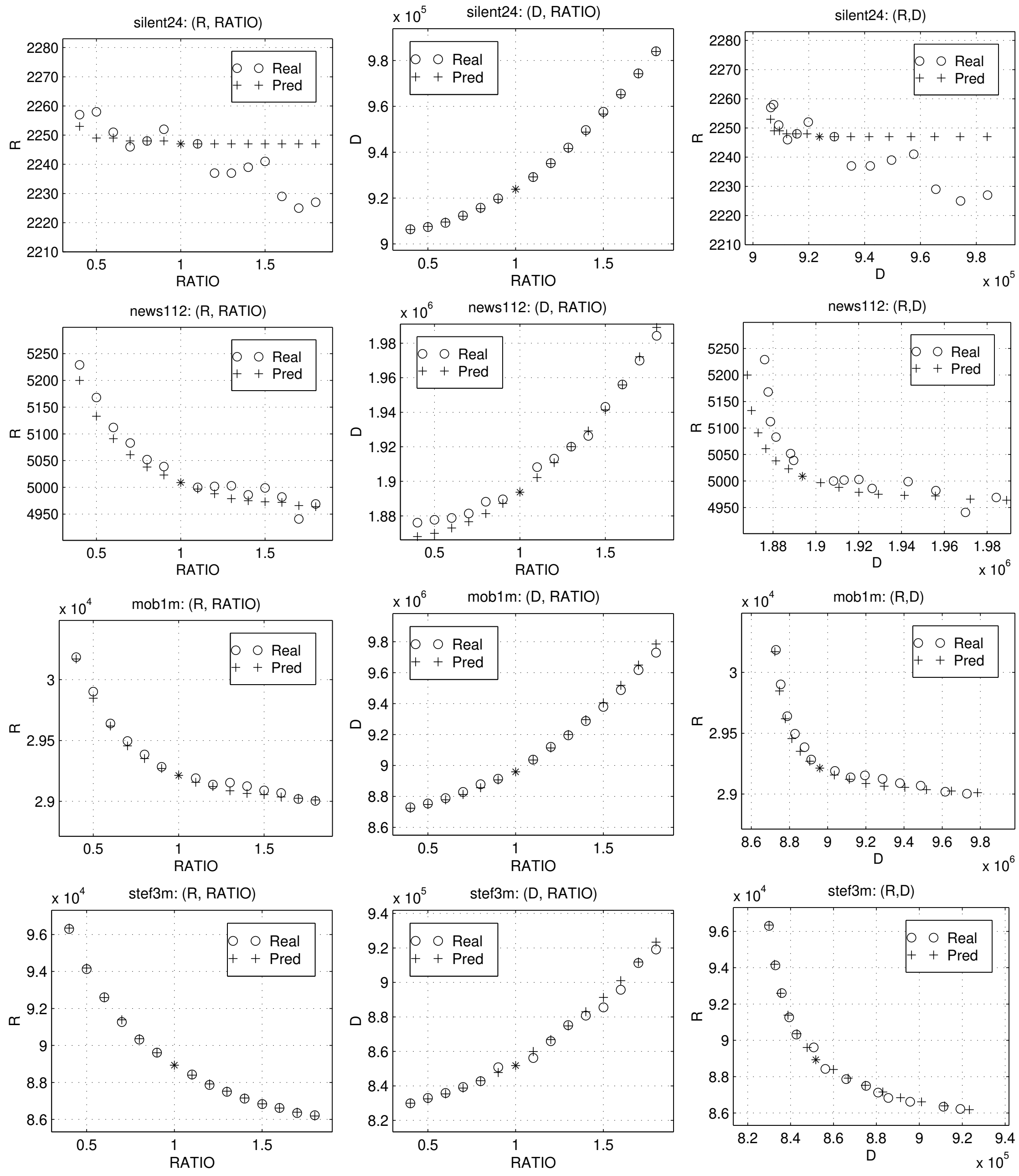
The above results show that the rate distortion performance of the matching pursuit encoder can be approximated for any operating point $\{Q'(\cdot), N'\}$ given an initial matching pursuit pass with a different quantizer and stopping point. This is a useful model for (R, D) optimization algorithms, particularly in combination with adaptive matching pursuit quantizer designs. The application of this model within specific (R, D) optimization algorithms will be investigated in Section 5.5. But first we show that useful rate distortion prediction may be done with less complexity than a full matching pursuit encoding pass. Such models will be introduced in the next section. While these are not as accurate as the Full Pre-code Model, they allow useful tradeoffs between the analysis complexity and the prediction accuracy. They are also suitable for situations where Full Pre-Code cannot be applied, as will be shown in Section 5.5.1.

5.4 Reduced Complexity Pre-Code Models

The previous section modeled the internal workings of the matching pursuit process in order to predict changes in (R, D) performance resulting from a change in the operating point $\{Q(\cdot), N\}$. This results in an accurate model, but it requires an initial analysis using a full matching pursuit decomposition. It also internally generates predictions of numerous parameter sets such as $\{\hat{p}'_i, \hat{d}'_i, i = 1...N'\}$. While these may be of interest to some users, the optimization applications reviewed in Section 5.2.1 require only the (R, D) information, and so modeling these additional parameters may be considered a waste of resources.

For situations where (R, D) is the only desired result and computational complex-

Figure 5.18: Overall rate and distortion (R, D) performance for the same four sample frames using the same prediction quantizers as in the previous figure. In each case, the reference quantizer $Q(\cdot)$ defined using $RATIO = 1.0$. The (R, D) point for the reference run is shown as (*) in each plot.



ity is a constraint, we present in this section a class of prediction algorithms which do not require a full matching pursuit pre-coding pass. The prediction is instead based on a reduced dictionary pre-coding pass, that is a matching pursuit decomposition using a subset of the full dictionary. The choice of dictionary subset will be shown to effectively set the tradeoff between the complexity of the pre-coding pass and the accuracy of the resulting prediction. We also present an extension of the model which combines a single reduced dictionary analysis pass with additional full-dictionary atoms to provide a continuum of complexity vs. accuracy tradeoff points.

The section is arranged as follows. The distortion model based on reduced complexity pre-coding is introduced in Section 5.4.1. Four dictionary subsets are considered, and tradeoffs in complexity and accuracy are explored and compared to the Full Pre-Code Model of the previous section. A compatible rate prediction model is then presented in Section 5.4.2.

5.4.1 Reduced Pre-Code Distortion Model

Motivation

We begin by motivating our current model and relating it to the Full Pre-Code model of the previous section. Consider model-based (R, D) generation as shown in the framework of Figure 5.1. For such applications, each source image $\vec{f}$ must typically be encoded with many choices of control parameter σ in the course of the optimization. It is thus efficient to implement the model in the following manner:

1. Pre-analyze the frame to be encoded, preferably using as little computation as possi-

ble.

2. Generate a model from the pre-analysis data of the current frame. Known data from previous frames may also be used.

3. Predict the desired (R, D) for each σ of interest from the resulting model.

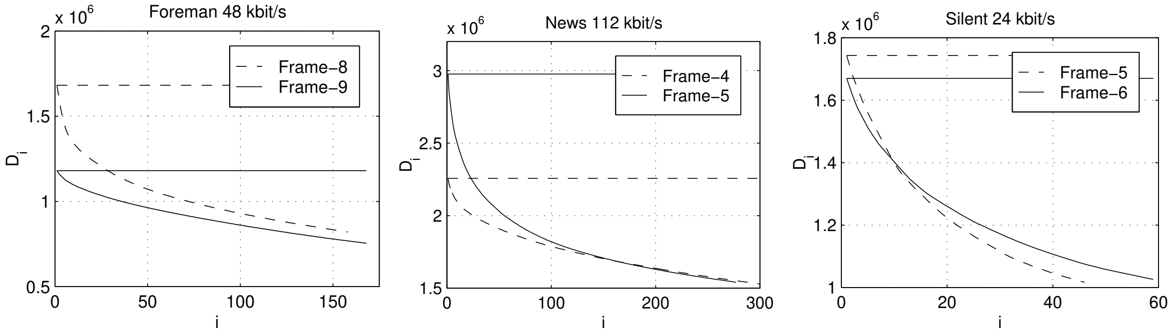
The above framework applies to the Full Pre-Code model of the previous section. For that method, the pre-analysis step consists of performing a full matching pursuit encoding pass with some initial quantizer, and the resulting distortion model is summarized in Figure 5.11. It is interesting to note that this three-step framework covers some DCT-based (R, D) models as well (e.g. [27, 39, 16]).

The complexity of using a model is typically concentrated within the pre-analysis step. This is certainly true for the Full Pre-Code model, for which the complexity of the initial encoding pass is much greater than the few arithmetic operations per atom needed to generate each $(\hat{R}, \hat{D})$ point from the resulting model. A significant reduction in model complexity thus requires a reduction in the complexity of the pre-analysis step. We thus hypothesize that useful prediction may be done without a full encoding pass used in the Full Pre-Code model, and we proceed to investigate the possibilities.

The first question to ask is whether a nontrivial pre-analysis of the current frame is even necessary. Suppose instead that we could predict the current frame distortion curve D_i as a possibly rescaled version of the previous frame curve. This approach would eliminate the expensive matching pursuit pass on the current frame, and require only minimal current frame computation (e.g. compute starting energy D_0 to determine the scaling factor). However, such a simple scheme can only work if the distortion behavior is very consistent from frame to frame. This is not generally the case, as shown by the plots in Figure 5.19. Each plot shows the distortion curves D_i for two successive frames of the same sequence encoded using the 2-pass adaptive quantizer algorithm of Chapter 4. Previous and current frame D_i are shown respectively by dotted and solid curves, and matching horizontal lines are added to highlight the starting energy of each frame. In viewing the curves, keep in mind that the task is to predict the solid curve given only the dotted curve and the solid horizontal line.

In Figure 5.19a, the two D_i curves are shown to have a very different curvature. The "Frame-8" curve begins with a sharp descent, while the "Frame-9" curve becomes almost instantly linear. A simple rescaling of the dotted curve would be unlikely to predict this change in curvature. In Figure 5.19b, "Frame-4" has a much lower starting energy than "Frame-5", but the two distortion curves seem to converge after about 100 atoms. Figure 5.19c shows an even more extreme case. In this plot we see that although "Frame-5" starts with higher energy then "Frame-6", after about 15 coded atoms the energy remaining in "Frame-5" is actually lower than that of "Frame-6". Unlike the previous two examples, the distortion curves in this plot diverge as the number of atoms increases. These examples show that the behaviour of D_i varies greatly from frame to frame, depending on how well the available basis functions match the remaining signal structures at each stage of the decomposition. Thus a simple model based only on D_i from the previous frame and starting energy of the current frame is unlikely to perform well. Accurate prediction of current frame D_i will require some non-trivial analysis of the current frame.

Figure 5.19: Distortion D_i vs. atom number i for two successive frames of the same sequence. Encoding based on the 2-pass quantization method of the previous chapter. (a) Foreman 48, frames 8-9. (b) News 112, frames 4-5. (c) Silent 24, frames 5-6.



(a) (b) (c)

Reduced Dictionary Analysis

To provide information useful to the prediction of current frame distortion, we propose an initial matching pursuit analysis pass using a reduced set of dictionary elements. There are two criteria for the design of such a reduced set. First, the basis functions included in the reduced set should be representative of those in the full dictionary set. This increases the likelihood that the two dictionary sets will have similar distortion performance, allowing the full dictionary distortion to be predicted from that of the reduced dictionary. A second design criterion relates to the complexity of the reduced dictionary analysis pass. Complexity is determined by the number of basis functions in the reduced set and the spatial extent of those functions. These factors must be considered in order to meet any constraints on analysis complexity.

For our experiments, we consider four reduced complexity dictionary sets, in order of decreasing complexity:

$$\{Red0, Red1, Red2, Red3\}$$

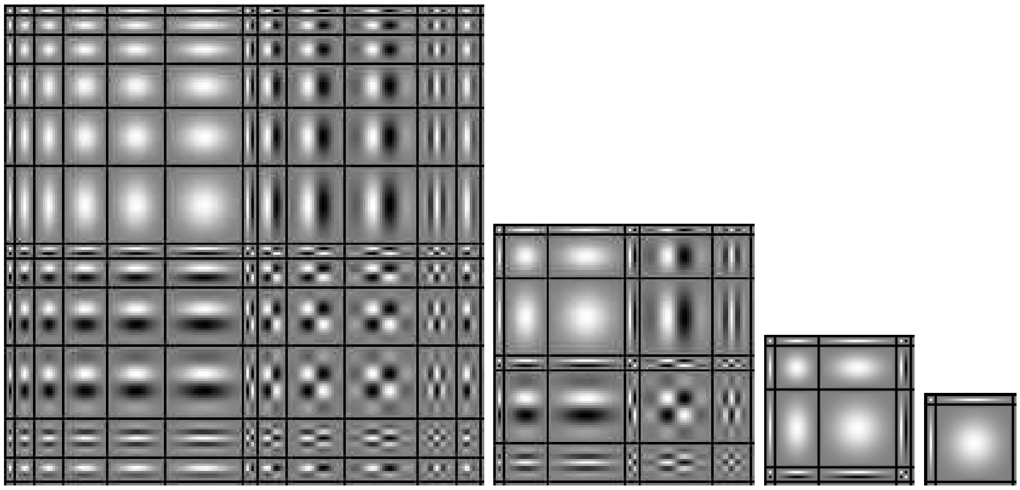
Table 5.5: Definition of reduced complexity dictionary sets as subsets of the full Gabor dictionary. An 'X' indicates that the 1-D basis is retained in a particular reduced set.

n	s_n	ξ_n	ϕ_n	Size	$Red0$	$Red1$	$Red2$	$Red3$
0	1.0	0.0	0	1	X	X	X	X
1	3.0	0.0	0	5	X			
2	5.0	0.0	0	9	X			
3	7.0	0.0	0	11				
4	9.0	0.0	0	15	X	X	X	
5	12.0	0.0	0	21	X			
6	14.0	0.0	0	23				
7	17.0	0.0	0	29	X	X	X	X
8	20.0	0.0	0	35				
9	1.4	1.0	$\pi/2$	3	X	X	X	
10	5.0	1.0	$\pi/2$	9	X			
11	12.0	1.0	$\pi/2$	21	X			
12	16.0	1.0	$\pi/2$	27	X	X		
13	20.0	1.0	$\pi/2$	35				
14	4.0	2.0	0	7				
15	4.0	3.0	0	7				
16	8.0	3.0	0	13	X	X		
17	4.0	4.0	0	7				
18	4.0	2.0	$\pi/4$	7	X			
19	4.0	4.0	$\pi/4$	7				

These are defined in Table 5.5 as subsets of the full Gabor dictionary. Each row of the table represents a 1-D function from the full dictionary, and an 'X' indicates that the function is retained in the corresponding reduced dictionary set. The reduced sets thus retain the separable structure of the full dictionary, and so reduced complexity separable computation may be used [45]. Note that these reduced dictionary sets were selected in an ad-hoc manner, and no automatic selection or optimization methods were used. However, the two design criteria mentioned above were considered during the manual selection process.

A visualization of the resulting 2-D basis functions is shown in Figure 5.20. It is interesting to compare these graphical basis images to that of the full dictionary, as shown in Figure 2.2. Note that the larger sets $Red0$ and $Red1$ seem to retain the overall four-quadrant

Figure 5.20: Reduced complexity dictionary sets. These are subsets of the full dictionary, which is shown in Figure 2.2.



$Red0$ $Red1$ $Red2$ $Red3$

structure of the original dictionary. These quadrants might be labeled clockwise from upper left as smooth Gaussian, vertical features, high frequency features, and horizontal features. However, the two smaller reduced sets, $Red2$ and $Red3$, consist mainly of the smooth Gaussian functions. The lack of variety is necessary here in order to achieve a reduced computational complexity. Given the 1-D functions which compose a reduced dictionary set, the complexity per atom for a reduced-dictionary encoding pass may be simply computed using the separable inner product computation described in [45]. The resulting complexity numbers are presented in Table 5.6. To summarize the table, the four reduced dictionary sets respectively have per-atom complexities of 35.7, 11.6, 4.9 and 2.2 percent of the full dictionary shown in Figure 2.2.

Once the reduced analysis pass for the current frame has been completed, it is used to predict the full dictionary performance. Define D_i^F and D_i^R as the distortion for the current frame coded to i atoms using the full and reduced dictionaries, respectively. The

Table 5.6: Relative complexity of the full and reduced Gabor dictionaries.

Dictionary	Number of 2-D Bases	Per-Atom Complexity (*)	Relative Complexity, β
Full Gabor	400	1.733	1.0
*Red*0	144	0.607	0.357
*Red*1	36	0.199	0.116
*Red*2	16	0.084	0.049
*Red*3	4	0.037	0.022

(*) Measured in millions of multiply-accumulate operations.

equivalent curves from the previous frame are $\dot{D}_i^F$ and $\dot{D}_i^R$. The problem is thus to predict D_i^F from known previous frame curves $\dot{D}_i^F$ and $\dot{D}_i^R$, and current frame pre-analysis curve D_i^R. Define a simple ratio mapping function as:

$$\dot{M}(i) \triangleq \frac{\dot{D}_i^F}{\dot{D}_i^R} \tag{5.21}$$

This is termed a mapping function because it can be used to map the reduced analysis distortion curve onto prediction space for the full dictionary curve. For the previous frame, we have:

$$\dot{D}_i^F = \dot{M}(i)\dot{D}_i^R \tag{5.22}$$

By using the known previous frame mapping function, we propose to predict the current frame distortion result as:

$$D_i^F \approx \dot{M}(i)D_i^R \tag{5.23}$$

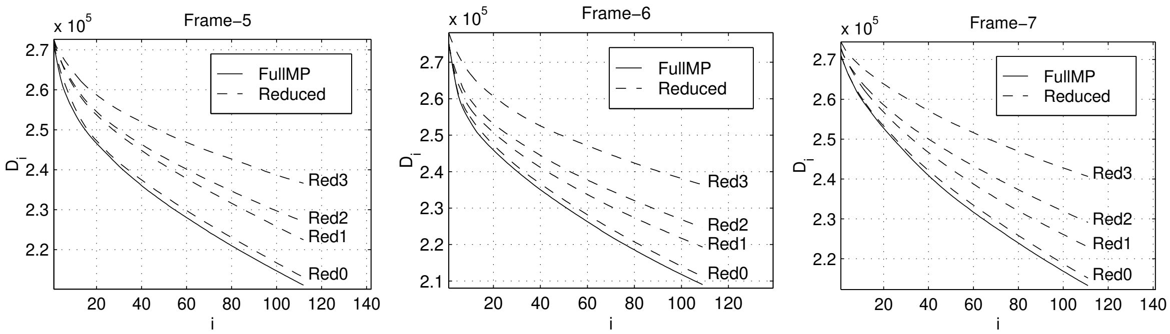
A problem arises if the stopping point for the current frame is larger than the stopping point used for the previous frame, since in this case the mapping $\dot{M}(i)$ does not extend far enough to allow prediction for the desired number of atoms. When this happens, we extend $\dot{M}(i)$ to the desired length using linear extrapolation.

The accuracy of the above approximation depends both on how "different" the full and reduced dictionaries are, and on how much the mapping function changes from frame to frame. If the reduced dictionary is very close in performance to the full dictionary, then Equation (5.21) will set $\dot{M}(i) \approx 1$, and the prediction according to Equation (5.23) effectively reverts to $D_i^F \approx D_i^R$. We expect prediction to work very well in this case, and this will be verified shortly for the *Red*0 case. Suppose instead that the reduced dictionary performance is significantly worse than that of the full dictionary. This will be true for the smaller reduced dictionary sets such as *Red*2 and *Red*3. Consistency of the mapping function from frame to frame becomes more critical in this case. If the mapping function varies slowly from frame to frame (that is, if $M(i) \approx \dot{M}(i)$ is true for most frames) then prediction will still work very well. However, the model breaks down whenever $M(i)$ and $\dot{M}(i)$ are mismatched.

We now illustrate the prediction procedure using sample distortion curves from a few frames of the Mother 24 kbit/s sequence. Figure 5.21 shows the basic D_i curves for the dictionaries under consideration. In each plot, the solid line shows the distortion curve for the full dictionary, and the dotted lines show the performance of the various reduced dictionaries. Note that we compute the reduced dictionary curves without quantization, and thus $M(i)$ relates the quantized full dictionary decomposition to the unquantized reduced complexity pass. In-loop quantization could be used, but then the choice of quantizer for the reduced pass becomes an issue. Such a design is not considered here.

The prediction method of Equation (5.23) requires that the relationship between reduced and full dictionary curves be consistent from frame to frame. Figure 5.21 suggests

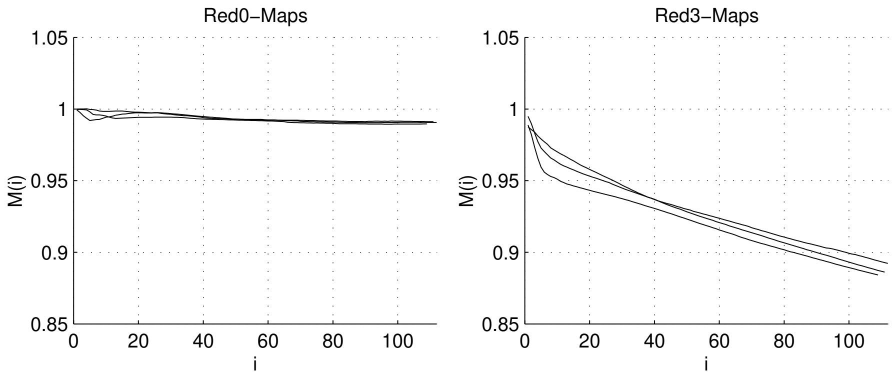
Figure 5.21: Distortion curves D_i produced by both the full Gabor dictionary (FullMP) and the various reduced dictionaries. Three successive frames of Mother at 24 kbit/s are shown.



that this is indeed the case, since the three subplots are nearly identical. The relationship may be examined further by directly plotting the mapping functions as defined in Equation (5.21). Figure 5.22a overlays the mapping functions which relate $Red0$ to the full dictionary for the three frames shown in the previous figure. Note that the mapping function is very consistent from frame to frame, and that $M(i) \approx 1$. This shows that the $Red0$ dictionary comes close in performance to the full dictionary, even though it has only about a third of the complexity. Figure 5.22b shows the mapping functions which relate $Red3$ to the full dictionary for the same three frames. The mappings are no longer close to 1, indicating that the less expensive $Red3$ dictionary lags significantly behind the full dictionary in performance. However the mapping functions are fairly consistent from frame to frame. This suggests that pre-analysis using the $Red3$ dictionary is still useful for predicting full dictionary performance.

A sample prediction result is shown in Figure 5.23. Here the D_i performance of the full dictionary for Frame-7 is predicted from reduced complexity pre-analysis using the mapping function from Frame-6. Figure 5.23a uses the $Red0$ dictionary for the pre-analysis, while Figure 5.23b uses the substantially less complex $Red3$ dictionary. In these two plots,

Figure 5.22: Simple ratio mappings overlaid for successive frames of Mother coded at 24 kbit/s. (a) Mapping Red0 $\rightarrow$ FullMP for frames $\{5, 6, 7\}$. (b) Mapping Red3 $\rightarrow$ FullMP for same three frames.



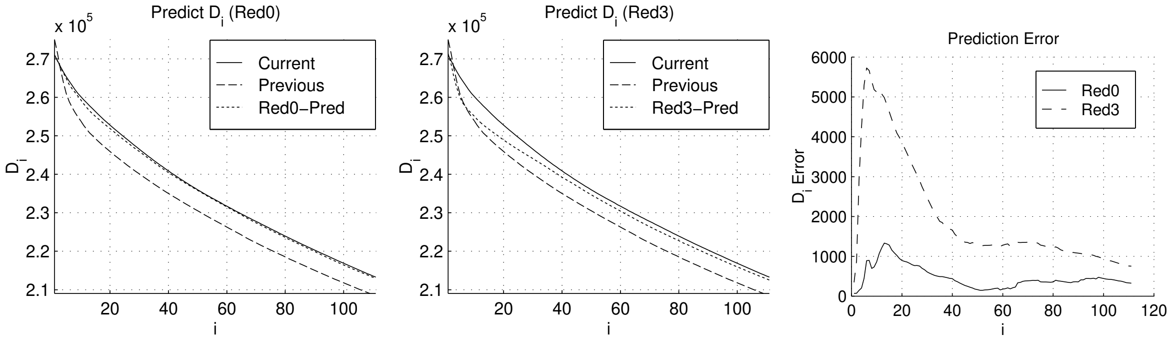
(a) (b)

the solid line represents the true D_i for the current frame, while the fine dotted line shows the predicted curve. Comparing the two plots, it is apparent that prediction using $Red0$ is more accurate than that of $Red3$. This is confirmed in Figure 5.23c, which shows the absolute prediction error for both cases. It illustrates an inherent tradeoff between analysis complexity and prediction accuracy. This tradeoff will be explored further in the next two subsections.

Hybrid Model Extension

In the reduced complexity analysis model of the previous section, the tradeoff between complexity and accuracy is effectively set by the design of the reduced analysis dictionary. The $Red0$ dictionary, for example, uses about one third of the full dictionary complexity to achieve a particular level of prediction accuracy. For some applications, it may be desirable to have a continuum of complexity vs. accuracy tradeoff options available. Suppose for example that the available computational resources are time varying. It is thus

Figure 5.23: Illustration of D_i prediction with reduced complexity pre-coding methods on Mother at 24 kbit/s. D_i of Frame-7 is predicted from reduced dictionary pass via Frame-6 mapping. (a) Prediction using Red0 dictionary. (b) Prediction using smaller Red3 dictionary. (c) Comparison of error in predicted D_i.



(a) (b) (c)

desirable to increase the accuracy of the prediction when more processor cycles are available to do so, and reduce the accuracy when cycles are scarce. In this case the model of the previous section becomes inconvenient, since each possible tradeoff between complexity and accuracy requires a redesigned reduced-complexity dictionary, and design methods for such dictionaries are still ad-hoc.

We now introduce a hybrid combination between the Full Pre-Code and Reduced Dictionary Pre-Code methods which addresses this problem and produces a continuous set of tradeoffs between complexity and prediction accuracy. An analysis pass using a reduced dictionary set is first performed, which allows the previous frame mapping technique of Equation (5.23) to be used. One or more full dictionary atoms are then computed and used to improve the prediction by effectively correcting the mapping. Each additional full-dictionary atom increases the analysis complexity and improves the expected prediction accuracy, allowing a continuous set of tradeoffs to be achieved. Since these tradeoffs are visited progressively, it is possible to stop at any time, thus the desired complexity point need not be known in advance.

We now define the specifics of the hybrid model. First define the anchor point a as the number of full dictionary atoms which have been computed for the current frame and are available for the prediction. Since the end model may be used to predict many (R, D) points with varying quantizers, we do not commit to a particular quantizer in computing these anchor atoms. Instead they are computed without in-loop quantization,[10] and the Full Pre-Code model of the previous section is applied to estimate quantization effects. This is economical, adding only a few arithmetic operations per full dictionary atom. We can thus assume that a limited number of Full Pre-Code distortion points are available. Denote these as $\{D_i^{PC}, i = 1...a\}$. Next define a mapping correction factor μ_a as:

$$\mu_a \triangleq \left. \frac{D_i^{PC}}{D_i^R \dot{M}(i)} \right|_{i=a}$$

This factor is defined in such a way that it corrects the previous frame mapping at the anchor point. Consider modifying the prediction of Equation (5.23) by multiplying the right side by the correction factor. The corrected prediction is:

$$D_i^F \approx \mu_a \dot{M}(i) D_i^R$$

Evaluating the resulting approximation at the anchor point yields:

$$D_a^F \approx \mu_a \left(\dot{M}(i) D_i^R \right) \Big|_{i=a} = D_a^{PC}$$

The Full Pre-Code result is thus recovered. We further claim that the corrected prediction for the remaining points $\{D_i^F, a < i \leq N\}$ is expected to improve because frame-to-frame

[10] A second option is to code the anchor atoms using an "average" quantizer taken from the set of quantizers $\{Q_c(\cdot), c = 1...C\}$ to be used for the C candidate rate-distortion points. Figure 5.24 will show that there is little to be gained in doing this, since the current method (without quantization) already produces an accurate mapping correction at the anchor point.

mapping inconsistencies due to earlier coded atoms have been corrected for. In particular, the first few atoms are typically of high energy and the effect on distortion due to coding these atoms is difficult to predict. The correction factor μ is used to adjust the previous frame mapping in a way that overcomes these early inconsistencies. Effectively, the correction factor allows us to "attach" the previous frame mapping to the current frame mapping at the anchor point.

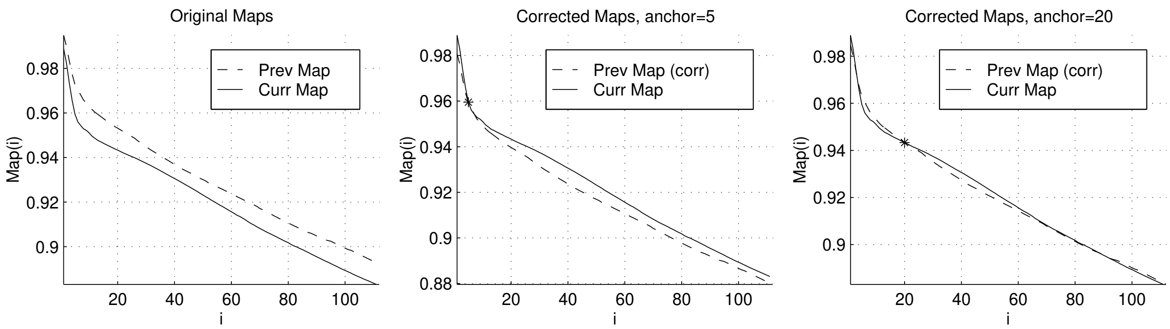
An illustration of this is shown in Figure 5.24. All three plots use our earlier Mother 24 kbit/s example, and the "current" frame refers to Frame-7. Figure 5.24a shows the original uncorrected mappings. Figure 5.24b shows the effect of correcting the previous frame mapping using an anchor point of 5. Note that the corrected map seems to match the current frame map perfectly at the anchor point. Moreover the maps are much closer together for the points beyond the anchor point $(i > 5)$. The consistency improvement is even greater for an anchor point of 20, as shown in Figure 5.24c. Since prediction accuracy relies on frame-to-frame mapping consistency, this implies that prediction of the distortion points for $(i > a)$ are improved.

We complete the hybrid model by noting that because the Full Pre-Code model is generally more accurate than even a corrected Reduced model, it is logical to use D_i^{PC} as the prediction in the $(i < a)$ region. The resulting hybrid model extension for any anchor point $a < N$ may thus be summarized as:

$$D_i^F \approx \begin{cases} D_i^{PC} & i < a \\ \mu_a \dot{M}(i) D_i^R & i \geq a \end{cases} \tag{5.24}$$

Two important points should be noted. First, even with the correction factor applied, the previous frame mapping will not exactly match the current frame mapping

Figure 5.24: An illustration of the hybrid model extension. Frame 7 of Mother at 24 kbit/s is predicted using the *Red*3 dictionary set. (a) Original mapping functions. (b) Previous mapping corrected with $a = 5$ full dictionary atoms. (c) Previous mapping corrected with $a = 20$. The (*) symbol denotes the anchor point in each case.



(a) (b) (c)

at the anchor point, since Full Pre-Code model distortion D_a^{PC} was used in place of true distortion D_a^F to compute the correction factor μ_a. Because the Full Pre-Code model is typically very accurate, this source of error is in practice limited to a small nonzero value. For example, the error is too small to be visible in the illustration of Figure 5.24. Second, although we expect each additional full dictionary atom to improve the prediction accuracy (of say a particular value $\{D_i, a < i < N\}$), such an improvement cannot be guaranteed. The tradeoffs that can be achieved will be explored in the next subsection.

Complexity vs. Accuracy

We now explore experimentally the complexity vs. accuracy tradeoffs which can be achieved using the reduced complexity distortion models. To this end, we code and compile prediction results using ten matching-pursuit coded frames from each of the fourteen standard MPEG-4 sequences. The frames are coded using the 2-pass quantizer design of the previous chapter. For each coded frame, the distortion achieved at the actual stopping point N is predicted using both the Full Pre-Code model and the Reduced Complexity Pre-Code models with all four reduced dictionaries. Prediction accuracy is then evaluated in terms of relative error in predicted distortion:

$$e_{|D_N|} = \frac{|\hat{D}_N - D_N|}{|D_N|}$$

The achieved relative prediction errors are compiled in Table 5.7. The column labeled "FullMP" refers to the Full Pre-Code model, and the remaining columns refer to Reduced Pre-Code using the specified dictionary. As expected, Full Pre-Code is seen to achieve the smallest error, with an average of 0.14 percent. The $Red0$ model is not far behind, with an 0.47 percent average error. The remaining models show progressively poorer prediction with average relative errors of 1.4, 2.0 and 3.0 percent. The maximum relative error across all tested frames is shown in Table 5.8. Again we see that FullMP and $Red0$ do a good job at containing the maximum errors, and that the remaining reduced complexity dictionaries do progressively worse. For a few sequences such as Hall-10 and Silent-24, the maximum error for the $Red3$ dictionary is extremely high. This result reflects cases where the mapping function is very different from frame to frame. The $Red3$ case, with only 2.2 percent of the full matching pursuit analysis complexity, is particularly sensitive to this problem. Because so little current frame analysis is being done, prediction relies much more heavily on consistency of the mapping function, and large errors occur when this consistency is violated.

Although the prediction accuracy of the $Red3$ model seems to be low, it may be improved by adding full dictionary atoms according the the Hybrid Model plan. The low analysis complexity of the $Red3$ dictionary makes it a prime candidate for such an

Table 5.7: Average % relative prediction error.

Seq	Label	FullMP	Red0	Red1	Red2	Red3
1	Cont-10	0.0403	0.4859	0.7092	0.9184	1.0441
2	Hall-10	0.2218	1.1986	2.7779	3.7255	6.2328
3	Mom-10	0.0477	0.3492	1.3693	1.8743	2.4651
4	Cont-24	0.1513	0.2655	0.4345	0.5472	0.6689
5	Silent-24	0.1246	0.6333	2.0069	4.2683	7.3063
6	Mom-24	0.1390	0.3516	1.4572	1.8786	2.5570
7	Coast-48	0.2344	0.2237	0.9817	1.4397	1.3739
8	Fore-48	0.3651	0.5281	2.5813	4.2459	7.3988
9	News-48	0.0963	0.5363	1.7749	2.1981	2.3014
10	Coast-112	0.0941	0.3430	0.3107	0.4686	0.3719
11	Fore-112	0.0324	0.2855	0.4847	1.0391	2.0126
12	News-112	0.1016	0.4862	2.8977	3.7956	5.9767
13	Mob-1024	0.1052	0.3889	0.5989	0.6145	0.9835
14	Stef-1024	0.1913	0.5615	0.6435	1.0561	1.3849
	Average	0.1389	0.4741	1.3592	2.0050	3.0056

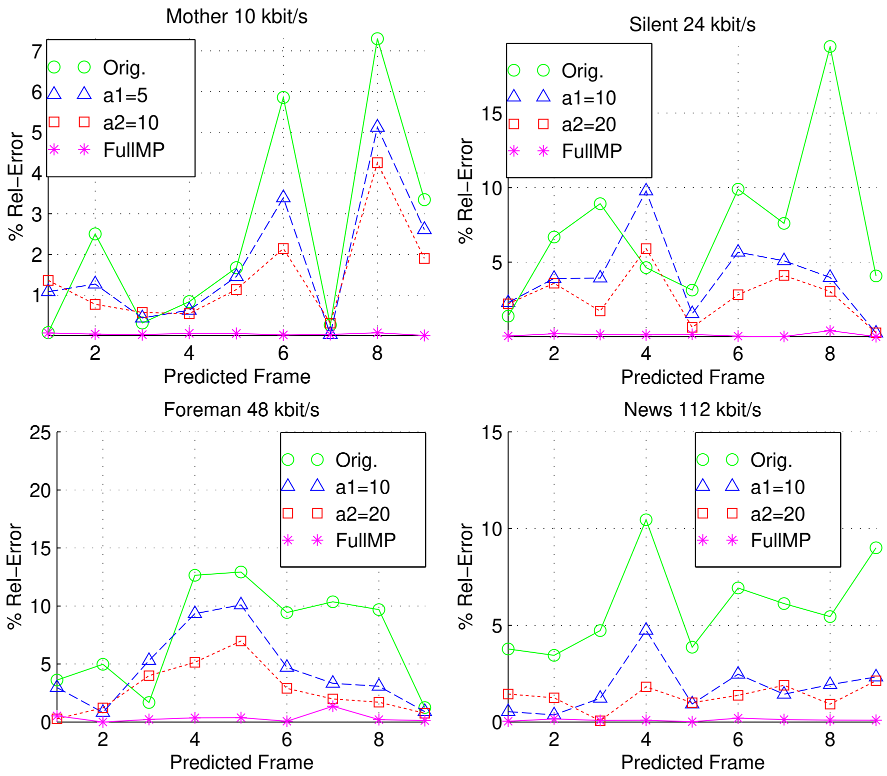
Table 5.8: Maximum % relative prediction error.

Seq	Label	FullMP	Red0	Red1	Red2	Red3
1	Cont-10	0.1458	1.2198	1.3567	1.3474	1.5414
2	Hall-10	0.5378	2.2464	12.1500	16.7474	35.8755
3	Mom-10	0.0789	0.6092	4.3268	5.9131	7.2962
4	Cont-24	0.3473	0.8749	1.1965	1.2750	1.3773
5	Silent-24	0.4092	1.5553	4.8747	11.5143	19.4636
6	Mom-24	0.3100	0.9305	5.6412	8.7735	11.4906
7	Coast-48	0.6363	0.6181	2.4576	4.3410	3.9637
8	Fore-48	1.3532	1.2731	4.6747	7.9676	12.9279
9	News-48	0.2148	1.0720	2.8339	3.8945	6.7626
10	Coast-112	0.1361	0.6102	0.5938	1.0031	1.6869
11	Fore-112	0.0616	0.6329	1.2362	3.2190	7.8295
12	News-112	0.1998	1.2638	5.6020	6.6106	10.4537
13	Mob-1024	0.1895	0.6242	1.4770	1.3331	1.8219
14	Stefan-1024	0.5796	1.2923	1.4995	2.6312	2.4994
	Average	0.3714	1.0588	3.5658	5.4693	8.9279

extension. We evaluate the improvement due to this hybrid model extension in Figure 5.25. The relative error vs. frame is plotted for each of four sequences. Separate error curves are shown for the original $Red3$ prediction model, and the extended model with anchor points of $a1$ and $a2$ atoms. Anchor points $a1$ and $a2$ are chosen in each case to keep the overall prediction complexity low. Relative error is shown experimentally to decrease as the anchor point increases. This is expected, and in fact Equation (5.24) shows that the predicted distortion curve approaches the Full Pre-Code prediction D_i^{PC} as $a \to N$. When $a = N$, the accuracy of the Full Pre-Code model will be achieved. This is not surprising, since by this time all N full-dictionary atoms have been found, and the complexity of Full Pre-Code has in fact been exceeded by the amount of the reduced dictionary computation.

While the models may be evaluated on prediction accuracy alone, it is more useful to look at the complexity vs. accuracy tradeoffs. These are compared for all three model classes in Figure 5.26. In each plot, relative prediction error is shown on the horizontal axis and relative complexity is shown on the vertical axis. Complexity is measured relative to the Full Pre-Code method. Each plot shows results averaged across all coded frames in a bit rate category. The tradeoff provided by the Full Pre-Code model is represented by an asterisk. This point is seen to achieve high accuracy at a relative complexity cost of 1. The basic Reduced Complexity Pre-Code models are represented by four symbols spread along the bottom axis of each plot. In general these are both less complex and less accurate than the Full Pre-Code model. However, it is interesting to note that the $Red0$ dictionary typically has much less complexity than Full Pre-Code without sacrificing much accuracy. This is most descriptive of the 48 kbit/s plot, but is seen at the other bit rates as well.

Figure 5.25: Improvement in accuracy due to the hybrid prediction model. Original *Red*3 prediction is compared to the hybrid model with anchor point set at $a1$ and $a2$. Full Pre-Code prediction is also shown. Results are shown for 9 frames encoded using the 2-pass quantizer encoder.



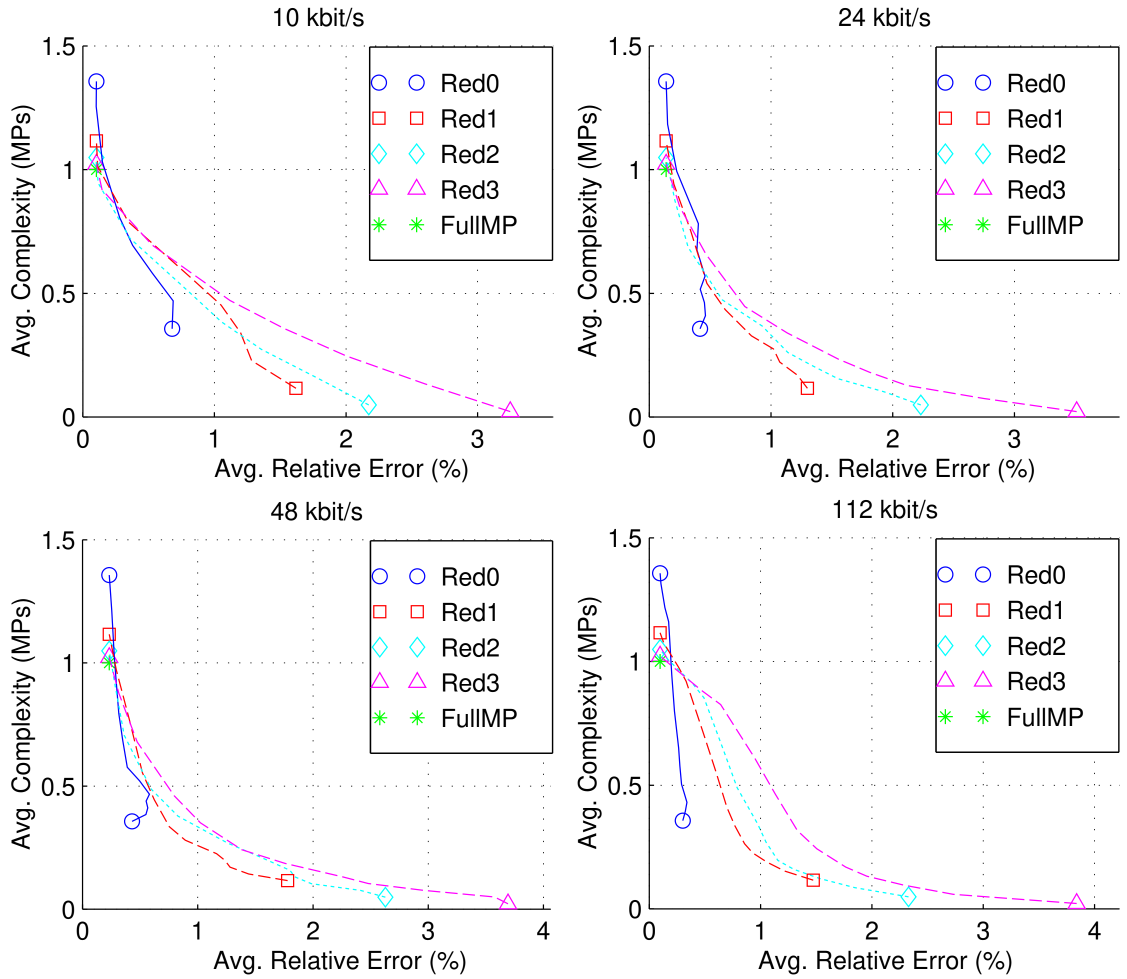
The connecting curves shown in each plot represent the tradeoffs achieved by the Hybrid Model extension as the anchor point a varies from 0 to N. With $a = 0$, no full dictionary atoms have been added, and the Reduced Complexity models result. With $a = N$, the accuracy of the Full Pre-Code model is achieved, but with the added cost of the Reduced Complexity Pre-coding step. For intermediate anchor points, we see a smooth curve that traces out a continuum of complexity vs. accuracy points. Although the Hybrid Model simplifies the generation of additional tradeoff points, the resulting performance is not as good as would be achieved by using the basic model with a reduced dictionary designed to hit the tradeoff. This can be seen on the plots. For example, the intrinsic $Red0$ tradeoff point, that is the $Red0$ point with $a = 0$, is shown in each plot to have smaller relative error than the Hybrid Model curves for $Red1$, $Red2$, and $Red3$ at the same average complexity level.

The results thus far show how the various models perform in predicting a single coding path, that of the 2-pass quantizer algorithm with an approximately constant bit rate. For (R, D) optimization algorithms, the task is different in that multiple rate-distortion points must be predicted for each candidate frame. A comparison more tailored to such algorithms will be shown in Section 5.5. But first we look briefly at a rate prediction algorithm to accompany the reduced complexity distortion models.

5.4.2 Reduced Complexity Rate Model

We now introduce a rate model compatible with the Reduced Pre-Code distortion model. Recall from the comments in Section 5.3.2 that the accurate rate targeting intrinsic to matching pursuit may remove the need for additional rate models for the (R, D) farming

Figure 5.26: Complexity vs. Accuracy tradeoffs, averaged across all sequences in a bit rate category. Full Pre-Code method is (*). Other symbols show Reduced Pre-Code models. Connecting lines show the Hybrid Model tradeoffs achieved as additional full dictionary atoms are added.



applications considered so far. We develop this model mainly for completeness, and also to show that a conceptually simple model yields reasonable prediction results.

Recall that the Full Pre-Code models are based on predicting the effects of a changing quantizer. It is thus natural that the associated rate model works by predicting the change in bits to quantization, with other bit rate contributions held constant, as described by Equation (5.20). For the Reduced Pre-Code case, rate information is generally available for the previous encoded frame, however the unquantized reduced dictionary pass yields no rate information for the current frame. We must thus resort to a prediction model based entirely on previous frame data. We do this by simply assuming that the cumulative rate function for atom coding is approximately constant from frame to frame. Here the cumulative rate function $\bar{R}_{atm}(k,i)$ is defined as the number of bits used to code the first i atoms of frame k. Since motion estimation for the current frame is typically performed prior to distortion prediction, we may assume that bits to motion and header are known exactly. The total rate to atoms for any desired stopping point N may be then predicted as:

$$\bar{R}_{atm}(k,N) \approx \bar{R}_{atm}(k-1,N)$$

If the number of previous frame atoms is insufficient, the cumulative rate curve may be extended using linear extrapolation.

Using this model, we predict the rate for the same ten matching pursuit encoded frames per sequence as were used in the results of the previous subsection. The results are summarized in Table 5.9. With the exception of Stefan, absolute errors are typically less than 2 or 3 atoms of bit budget. This can be seen by comparing the "Avg. Error"

Table 5.9: Results from the simple rate prediction model.

Seq	Label	Atoms/ Frame	Bits/ Atom	Avg. Error (bits)	Avg. % Rel-Error
1	Cont-10	53.89	20.63	10.23	0.92
2	Hall-10	40.78	22.64	16.56	1.67
3	Mom-10	41.22	22.96	15.09	1.60
4	Cont-24	117.89	18.63	29.58	1.35
5	Silent-24	75.00	21.80	31.28	2.05
6	Mom-24	99.89	20.38	28.64	1.32
7	Coast-48	201.33	19.94	39.42	0.98
8	Fore-48	150.67	20.35	58.82	1.98
9	News-48	207.11	21.64	39.17	0.89
10	Coast-112	158.44	23.74	26.82	0.71
11	Fore-112	83.33	25.56	30.74	1.69
12	News-112	293.44	20.68	58.76	0.96
13	Mob-1024	1612.78	18.36	65.08	0.22
14	Stefan-1024	1554.78	19.18	133.91	0.45
		Average:	21.17	41.72	1.19

column to the "Bits/Atom" column. The corresponding relative error percentages are at most 2 percent, and are below 1 percent for half the tested sequences. The average rate prediction error across all sequences is 1.19 percent, or about 2 atoms of bit budget. Note that because the previous frame cumulative rate curves are generally known, rate prediction for the current frame by this method requires no additional computation.

5.5 Evaluation of models

In the previous two sections, we introduced two classes of (R, D) models for matching pursuit. We now turn to the task of evaluating these models in the optimization framework originally posed in Figure 5.1. Each of the two model classes may be used to define the "Modeled $(\hat{R}_k, \hat{D}_k)$" generation box in the figure. Using the groundwork laid in Section 5.2, we now proceed to evaluate the performance of these models in such an optimization context.

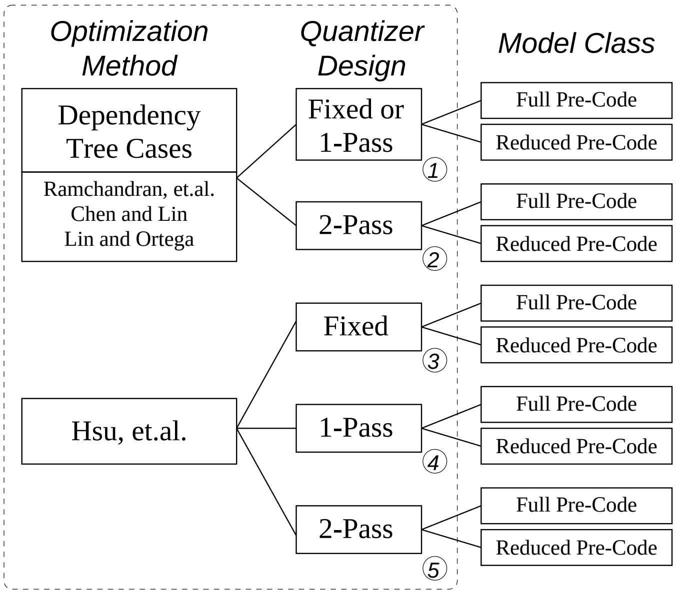
We begin in Section 5.5.1 by evaluating the complexity reductions which can be achieved using the models. The accuracy of the models will then be assessed and compared to that of the cubic spline interpolation model in Section 5.5.2.

5.5.1 Complexity

In this subsection, we evaluate the reduction in (R, D) data farming complexity obtained through the use of the two matching pursuit model classes introduced in Sections 5.3 and 5.4. Recall that the exact (R, D) generation complexity was analyzed earlier in Section 5.2.3, and the results were summarized in Tables 5.1 and 5.2. That analysis revealed that the exact (R, D) farming complexity depends both on the optimization method and the matching pursuit quantizer design type. The same dependencies apply when models are used, and so we are obligated to consider all combinations of four optimization methods, three quantizer design types, and two model classes.[11] Fortunately, the common dependency tree framework introduced in Section 5.2.1 will greatly simplify the analysis of the three optimization methods to which it applies. Complexity considerations will also eliminate some of the combinations, and our analysis will ultimately show that at most one of the two model classes may be usefully applied for each combination of optimization method and quantizer design type.

[11] Although the Hybrid Model Extension developed in Section 5.4.1 could be counted as a third model class, we omit the analysis of this case. We instead note that the Hybrid model may be substituted for the Reduced Pre-Code model in any application. This will result in additional model complexity vs. accuracy tradeoffs, essentially interpolated between those of the Reduced Pre-Code models with various reduced dictionaries. The intuition for this is illustrated in Figure 5.26.

Figure 5.27: A map of cases to be considered for modeled (R, D) complexity analysis. Twelve possible combinations of optimization method and quantizer design type are organized into five enumerated cases which are each considered separately. Both model classes must be considered for each case.



We now show complexity analyses for the various combinations of bit allocation optimization method, quantizer design type, and matching pursuit model class. A "map" of the various cases to be considered is shown in Figure 5.27. The figure describes graphically how some combinations of optimization method and quantizer design type may be clustered together for our complexity analysis. Specifically, the dependency tree framework will be used to combine the complexity analysis for three of the four optimization methods. For this framework, we earlier noted that the complexity for Fixed and 1-Pass quantizer designs is identical, and so these are also merged. The result is a reduced set of five enumerated cases, and each must be considered in combination with each of the two model classes. We now proceed to analyze each case in detail. Analysis results will then be compactly summarized in Tables 5.10- 5.12.

Case 1: Dependency Tree with Fixed or 1-Pass Quantizer

We now consider the complexity of model-based (R, D) generation for the dependency tree optimization methods in combination with either Fixed or 1-Pass adaptive quantizer design. Before delving into the analysis, we briefly review the process for exact (R, D) generation, since this will help us determine whether a model gives a useful reduction in complexity over the exact case. In this spirit, consider the exact (R, D) expansion of an arbitrary cousin node of the dependency tree. For Fixed or 1-Pass quantizer design, all σ choices for the corresponding frame are computed using a single shared encoding pass. This pass generates the exact (R, D) points for each σ, and it also provides the prediction images needed for (R, D) computation of any dependency children of the node.

Consider application of the Full Pre-Code model. This model requires a full initial encoding pass before any points $(\hat{R}, \hat{D})$ may be predicted. The cost of using the model is thus the same as the exact (R, D) cost, and so in this case Full Pre-Code yields no complexity advantage.

For the Reduced Pre-Code models, we note a distinction between terminal and non-terminal nodes of the dependency tree. For non-terminal nodes, a full encoding of each frame instance must be performed in order to generate prediction images for the children of the node. Because this full encoding pass exactly generates the desired (R, D) points, there is no advantage in modeling them. Non-terminal nodes of a dependency tree should thus be computed without the use of models.

Terminal nodes have no children, and so do not require a full encoding pass. For these nodes, we may apply the Reduced Pre-Code models. Note that the resulting complexity must be less than one full encoding pass per unique cousin node in order to realize any gain over exact (R, D) computation. Such gains can be realized, as we now illustrate with our usual example.

Example - Expansion of $\{I/B_1/B_2/P\}$:

Consider the full dependency tree expansion of the $\{I/B_1/B_2/P\}$ frame grouping, illustrated in compact form in Figure 5.5. The first two levels of the tree consist of non-terminal nodes, and thus should be exactly computed in order to generate the prediction children. The complexity of such exact computation may be obtained by applying Equation (5.3) to the first two levels of the tree, which for this example results in a complexity of $1 + 2 = 3$ encoding passes.

The third level of the tree consists of terminal nodes which may be divided into two sets of four cousins. The (R, D) points for each of these eight frame instances will be predicted using a Reduced Pre-Code model. Let β be the relative complexity of the reduced dictionary set on which the model is based. This represents the per-atom complexity for the reduced dictionary as a fraction of the full dictionary complexity. Refer to Table 5.6, which shows values of β for the various reduced dictionary sets developed earlier.

The Reduced Pre-Code model requires a reduced dictionary encoding pass for each of the eight frame instances in the terminal layer. It also requires passes with both the full and reduced dictionaries to be performed for the "previous" frame along each dependency path, which for our example corresponds to all nodes in the second layer of the tree. Because full dictionary passes were already performed in the exact encoding of this layer, we need only count the complexity of the reduced passes. We thus add 2 reduced passes for the second tree layer and 8 for the third (terminal) layer. Expressed in terms of full encoding passes, the added complexity for the reduced dictionary analysis is $(2 + 8)\beta$ encoding passes.

The total (R, D) generation complexity using the Reduced Pre-Code models is thus

$$3 + 10\beta$$

full encoding passes. Recall that exact (R, D) generation complexity for this example was earlier computed to be 11 encoding passes. Thus a model-based complexity reduction will be seen provided $\beta < 0.8$, which is true for all reduced complexity dictionary sets listed in Table 5.6.

It is useful to extend the complexity computation method used above to apply to an arbitrary dependency tree $\mathcal{Z}$. We start with the exact (R, D) generation complexity given by Equation (5.3), along with the associated notation. Consider $\mathcal{Z}'$, the collection of nodes which do *not* correspond to trailing I-frames. We now define two subsets of $\mathcal{Z}'$ which will be useful to the complexity calculation. Define set $\mathcal{Z}_1$ to consist of all nodes $z \in \mathcal{Z}'$ for which full encoding passes are required. $\mathcal{Z}_1$ thus consists of all non-terminal nodes in $\mathcal{Z}'$. Define a second set $\mathcal{Z}_2$ to consist of all nodes in $\mathcal{Z}'$ for which reduced complexity passes are required. This will contain all terminal nodes in $\mathcal{Z}'$, as well as those nodes from which previous frame mappings $\dot{M}(i)$ must be generated, as defined in Equation (5.21). The total complexity using a Reduced Pre-Code model with reduction factor β is thus:

$$\hat{\kappa}_1 = \mathcal{I}_{(I^*)} + \left(\sum_{\forall z \in \mathcal{Z}_1} A_z \right) + \beta \left(\sum_{\forall z \in \mathcal{Z}_2} A_z \right) \tag{5.25}$$

We now summarize our analysis for dependency tree cases with Fixed or 1-Pass quantizer design. We showed that application of the Full Pre-Code model gives no reduction in complexity, but that gains can be realized by applying the Reduced Pre-Code models to the terminal nodes of the tree. Exact (R, D) generation is used for the remaining (non-terminal) nodes. Note that although modeling gains are realized only for terminal nodes of the tree, these are typically the most populous due to the exponential expansion of nodes with tree depth.

The resulting complexity is computed using Equation (5.25). This method will be used shortly to evaluate the model-based complexity reductions for our three specific dependency tree cases. But first we consider the remaining four cases from Figure 5.27.

Case 2: Dependency Tree with 2-Pass Quantizer

Next consider model-based (R, D) generation for dependency tree expansion with 2-Pass quantizer design. Recall that the exact (R, D) generation process requires encoding each unique frame instance with a total of $(1+C)$ encoder passes. The first pass is unquantized and is used to design quantizer and stopping point parameters $\{N(\sigma), DZ(\sigma), QP(\sigma)\}$ for C different choices of control parameter σ. The remaining C passes are quantized completion passes, one for each choice of σ with its associated quantizer and stopping point. As we found in the Fixed and 1-Pass cases, exact generation must be used for all non-terminal nodes of the dependency tree, since for these nodes prediction children must be computed for use in the next tree level. Models may thus be applied only to terminal nodes of the tree.

Even with modeling, the first unquantized pass is required in order to obtain quantizer and stopping point parameters for the various σ choices. This initial pass is also sufficient to define the Full Pre-Code model for the given frame, which may then be used for (R, D) prediction without additional encoding steps. To see this, consider the summary of the Full Pre-Code distortion model given in Figure 5.11. In that figure, take initial quantizer $Q(\cdot)$ to be infinitely fine (i.e. no quantization) and then apply the prediction algorithm to predict the distortion for the combination of N' and $Q'(\cdot)$ corresponding to each σ of interest. In this way, one initial pass is used both to define the quantizers for each σ and to predict the resulting (R, D) tradeoffs via the Full Pre-Code model.

Before analyzing the resulting complexity for the full dependency tree, we note that there is no advantage to using the Reduced Pre-Code models in this case. Because the unquantized initial pass is required for quantizer design, it cannot be eliminated through the models. Thus any application of Reduced Pre-Code would only add computation (in the form of additional reduced dictionary passes) and reduce the modeling accuracy (since

Reduced Pre-Code is generally less accurate than Full Pre-Code). The Full Pre-Code model is clearly a better choice.

We now revisit our dependency tree example in order to illustrate the complexity reduction due to Full Pre-Code (R, D) modeling.

Example - Expansion of $\{I/B_1/B_2/P\}$**:**
Again we consider expansion of the $\{I/B_1/B_2/P\}$ frame grouping with $C = 2$, as illustrated in Figure 5.5. This time we assume a 2-Pass quantizer design, and apply the Full Pre-Code model to estimate (R, D) for the terminal nodes. Exact computation should be used for the first two levels of the tree which consist of non-terminal nodes. The resulting complexity may be found by applying Equation (5.4) to the first two levels of the tree. Doing this reveals a cost of $1(1+2) + 2(1+2) = 9$ encoding passes.

The third level of the tree consists of two sets of four cousins, all of which are terminal nodes. The initial unquantized pass is applied to each of these 8 frame instances in order to design the quantizers for the various σ choices. We then base Full Pre-Code prediction on this initial pass, and no additional encoding passes are necessary.

The total (R, D) farming cost when Full Pre-Code is used is thus $9 + 8 = 17$ encoding passes. Recall that our earlier analysis of the exact (R, D) generation case revealed a total of 33 passes. Use of the Full Pre-Code model thus reduces the complexity of our example by about a factor of 2.

We now extend the above complexity analysis method to cover arbitrary dependency trees. Begin with Equation (5.4) along with the associated notation. Recall that by our earlier definitions, $\mathcal{Z}'$ is the set of all nodes in tree $\mathcal{Z}$ which are not trailing I-frames, and $\mathcal{Z}_1$ consists of all non-terminal nodes in $\mathcal{Z}'$. Define $\mathcal{Z}_3$ as the set of all terminal nodes in $\mathcal{Z}'$. Thus $\mathcal{Z}_1$ and $\mathcal{Z}_3$ form a partition of $\mathcal{Z}'$. The total complexity for this case using the Full Pre-Code model may be expressed as:

$$\hat{\kappa}_2 = \mathcal{I}_{(I^*)}(1 + C_{(I^*)}) + \left(\sum_{\forall z \in \mathcal{Z}_1} A_z \cdot (1 + C_z) \right) + \left(\sum_{\forall z \in \mathcal{Z}_3} A_z \right) \tag{5.26}$$

We may now summarize the analysis of dependency tree cases with 2-Pass quantizer design. We found that although the Reduced Pre-Code models cannot be used, complexity reductions can be shown by applying the Full Pre-Code model to the terminal nodes of the tree. Exact (R, D) generation must be used for the remaining (non-terminal) nodes. The resulting complexity may be computed using Equation (5.26).

Case 3: Hsu, et.al. with Fixed Quantizer

Consider the iterative trellis-based optimization of Hsu, et.al. [29] paired with a Fixed quantizer design. Recall that exact (R, D) generation requires M encoding passes per iteration, one for each frame under consideration. These passes are required in order to fix the motion prediction frames for the next iteration of the optimization. They are also sufficient to enable exact (R, D) computation, as noted in our earlier analysis of the exact case. Thus no complexity advantage can be gained from modeling for this combination of optimization method and quantizer design.

Case 4: Hsu, et.al. with 1-Pass Quantizer

The same optimization method is now considered with a 1-Pass adaptive quantizer. Recall that the trellis optimization was illustrated in Figure 5.2. Our earlier analysis of the exact (R, D) case revealed that the 1-Pass quantizer design used to extend each node into the next stage will depend on the path leading into the node from previous stages. Because of pruning, only one such incoming path may exist for each node. Path extension for the exact (R, D) case thus requires a single encoder pass for each trellis node with its associated quantizer design.

Although the extension of each node requires a new predictive quantizer design, each node does not correspond to a different source image. In fact, the use of fixed motion prediction frames means that all nodes at a given stage k share an identical source image. It is thus possible to perform a single encoding pass on that source image, and then apply the Full Pre-Code model to predict the effects of the unique quantizer design associated with each node at stage k. All path extensions for all nodes at stage k may thus be predicted from a single encoding pass, and one such pass is required for each of the M frames under consideration. Better still, an appropriate encoding pass is already available, since it was performed for each source image during the previous iteration in order to update the fixed motion prediction frames for the current iteration. Thus the total (R, D) generation complexity for this case may be reduced to M encoding passes per iteration. These passes are used both to update the fixed motion predictors and to define the Full Pre-Code model used for modeling the path extension for each node.

Note that M is the minimum number of encoding passes required to fix the motion predictors for the next trellis optimization iteration. No further complexity reduction is possible, and so there can be no advantage to using the Reduced Pre-Code models.

Case 5: Hsu, et.al. with 2-Pass Quantizer

Finally we consider the method of Hsu, et.al. [29] with the 2-Pass quantizer design. As before, we note that the fixed motion predictors used in the trellis optimization result in a single unique source image per stage. Exact (R, D) computation for each stage thus requires a quantizer definition pass and C completion passes. However, the definition pass was already computed during the previous iteration in order to fix the current motion predictors. Total complexity for exact (R, D) computation thus consists of CM completion passes used in the current trellis optimization, plus $2M$ additional passes to fix motion predictors for the next stage.

Suppose Full Pre-Code is used for predicting the path extensions for the current trellis optimization. For each stage, we base a Full Pre-Code model on the quantizer definition pass which has been already computed in order to fix the current motion predictors. These models may be used to complete the trellis optimization with no additional encoding steps. Thus the total (R, D) generation complexity per iteration is $2M$ encoding passes when using the Full Pre-Code model.

Because $2M$ is the minimum number of passes per iteration required to fix the motion predictors, no additional complexity gain can be realized by applying the Reduced Pre-Code models.

Summary of complexity results

We now summarize the complexity analysis of the preceding five cases in compact table form. The results will then be used to compute model-based complexity reduction factors for each combination of optimization method and quantizer design. Before proceeding, we note that each such combination was shown to be compatible with at most one of the two model classes. Table 5.10 provides a summary of which models provide complexity reductions for each of these combinations. Note that the dependency tree entry covers all three methods for which representations were shown in Figure 5.6. There are thus 12 combinations of optimization method and quantizer design, and models were shown to provide useful complexity reduction in all but one of these cases. The exception is the method of Hsu, et.al. [29] combined with a Fixed quantizer design.

We now summarize in table form the model-based complexity for each combination

Table 5.10: A summary of the "useful" combinations of optimization method, quantizer design type, and (R, D) model class as revealed by complexity analysis of the various cases.

Optimization	Quantizer Design		
	Fixed	1-Pass	2-Pass
Dependency Tree	Reduced-PC	Reduced-PC	Full-PC
Hsu, et.al.	–	Full-PC	Full-PC

of method and quantizer design. Table 5.11 shows this for the three optimization methods covered by the complexity tree framework. For easy comparison, the exact (R, D) generation complexity from Table 5.1 has been reproduced as Table 5.11a. The modeled (R, D) complexity is then given in Table 5.11b.

The first of the two complexity columns corresponds to the Fixed or 1-Pass quantizer design types. For these two designs, our analysis showed that the Reduced Pre-Code models could be used to reduce the complexity. The complexity expressions shown in the table were thus produced by applying Equation (5.25) to the dependency tree representations for each optimization method. These were presented in Figure 5.6. Similarly, the second complexity column of the table corresponds to the 2-Pass adaptive quantizer design, for which our analysis identified complexity reductions due to application of the Full Pre-Code model. The tabulated complexities were thus computed by applying Equation (5.26) to the dependency tree representations in Figure 5.6.

By comparign Tables 5.11a and 5.11b for the Fixed and 1-Pass quantizer design, we see that the highest order polynomial term for the exact (R, D) case is multiplied by a factor of β in the modeled case. As an example, the highest order term for Chen and Lin [8] is C^2 for the exact case, but is reduced to βC^2 for the modeled case. Turning to the 2-Pass adaptive quantizer, we see that modeling typically reduces the polynomial order by

Table 5.11: Comparison of exact and modeled (R, D) generation complexity for the three optimization methods covered by the dependency tree framework. Each entry is the number of encoding passes per frame grouping required for (R, D) farming. Where applicable, C is the number of control parameters considered per frame, and β is the complexity reduction factor for the reduced dictionary set.

(a) Exact (R, D) generation complexity, copied from Table 5.1

Optimization Method	Matching Pursuit Quantizer Design	
	Fixed or 1-Pass Adaptive	2-Pass Adaptive
Ramchandran, et.al. [54]	$2 + C + 3C^2 + 2C^3 + 2C^4$	$(1 + C)(2 + C + 3C^2 + 2C^3 + 2C^4)$
Chen and Lin [8]	$1 + C + C^2$	$(1 + C)(1 + C + C^2)$
Lin and Ortega [39]	16	108

(b) Modeled (R, D) generation complexity

Optimization Method	Matching Pursuit Quantizer Design	
	Fixed or 1-Pass Adaptive	2-Pass Adaptive
Ramchandran, et.al. [54]	$2 + C + C^2 +$ $\beta(C + 3C^2 + 2C^3 + 2C^4)$	$2 + 3C +$ $4C^2 + 3C^3 + 2C^4$
Chen and Lin [8]	$1 + C + \beta(C + C^2)$	$1 + 2C + 2C^2$
Lin and Ortega [39]	$4 + 14\beta$	36

one. For Chen and Lin, exact (R, D) generation complexity yields a fifth degree polynomial in C, but only a fourth degree polynomial when modeling is used.

Table 5.12 shows the results of our model-based complexity analysis for the method of Hsu, et.al. [29] with each quantizer design. Again, the exact (R, D) generation results are reproduced for comparison as Table 5.12a, and the modeled (R, D) generation results are summarized in Table 5.12b. Recall that the models did not show a complexity reduction for the Fixed quantizer case, but reductions were shown for both adaptive cases. For 1-Pass quantization, Full Pre-Code was shown to reduce the complexity by a factor of $(S + 1)$, where S is the number of buffer fullness states used for trellis optimization. For 2-Pass quantization, Full Pre-Code reduces the complexity by a factor of $(C + 2)/2$, where C is the number of σ choices considered for each frame.

Table 5.12: Comparison of exact and modeled (R, D) generation complexity for the optimization method of Hsu, et.al. [29]. The optimization is done over M frames using S quantized buffer fullness states. Each frame is coded with C choices of control parameter σ. Each complexity entry is the total number of encoding passes per dependency iteration required for the (R, D) farming task.

(a) Exact (R, D) generation complexity, copied from Table 5.2

M.P. Quantizer Design	Complexity
Fixed	M
1-Pass Adaptive	$(S+1)M$
2-Pass Adaptive	$(C+2)M$

(b) Modeled (R, D) generation complexity

M.P. Quantizer Design	Complexity
Fixed	–
1-Pass Adaptive	M
2-Pass Adaptive	$2M$

To show a more tangible demonstration, we assume reasonable values for the various parameters and tabulate the model-based complexity reduction factors for each choice of optimization method and quantizer design. For all four optimization methods, we assume $C = 12$ different control parameter choices. For the case of Hsu, et.al. [29], we further assume $S = 20$ quantized buffer states. The resulting complexity reduction factors are shown in Table 5.13. Essentially the table shows that model based complexity reductions factors between 2 and 20 are possible for most of the analyzed cases.

5.5.2 Accuracy

The previous subsection showed that the two model classes developed in Sections 5.3 and 5.4 produce significant complexity reductions when used to model the (R, D) farming process for various optimization methods from literature. However, model-based (R, D) generation introduces some error into the optimization process. To minimize this

Table 5.13: Model-based complexity reduction factors for the various combinations of optimization method and quantizer design. Reasonable values of $C = 12$ and $S = 20$ are assumed.

Dependency Tree	Fixed or 1-Pass Adaptive (Red0)	(Red1)	(Red2)	(Red3)	2-Pass Adaptive (Full-PC)
Ramchandran, et.al. [54]	2.77	8.37	19.05	39.24	12.48
Chen and Lin [8]	2.29	5.05	7.61	9.55	6.52
Lin and Ortega [39]	1.78	2.84	3.41	3.71	3.00

Non-Dependency Tree	Fixed	1-Pass Adaptive (Full-PC)	2-Pass (Full-PC)
Hsu, et.al. [29]	–	21.00	7.00

source of error, model prediction error must be kept low. In this section, we evaluate the accuracy of our models. We focus on distortion model accuracy, since the use of target rates for our control parameters σ removes the need for explicit rate modeling. To benchmark the distortion accuracy, we use the cubic spline model proposed by Lin and Ortega [39]. Before proceeding, we briefly review the cubic spline model, and we restate why the model is suitable for our comparison.

Using Spline to Benchmark Accuracy

At the end of Section 5.2.2, we reviewed the cubic spline interpolation model proposed for (R, D) modeling by Lin and Ortega [39]. The model is based on several exactly computed control points, and interpolation is used to predict the points that fall between these control points. For example, the distortion model $\hat{D}(\sigma)$ is based on a set of exactly computed distortion points $\{D(\sigma_j), j = 1...N_c\}$. Cubic spline interpolation is then used to predict the distortion that would result for any choice of σ which falls between two of the known control points. The interpolation requires a minimum of four control points, but five or six are typically used [39].

In our earlier review, we concluded that the spline model is a suitable accuracy benchmark for our matching pursuit models. The reasons for this are briefly restated here.

1. The spline model is the clear winner when compared against the prediction accuracy of three other models, as shown in a comparison by Lin and Ortega [39].

2. The spline model is sufficiently accurate to be used for model-based (R, D) optimization. This was demonstrated by by Lin and Ortega [39, 40], who successfully incorporated it into their gradient-based optimization method.

3. Spline interpolation is generic, and so is easily extended to the matching pursuit case, or for that matter any encoder.

Accuracy Comparison

We now construct an experiment to compare the accuracy of our matching pursuit models to that of the cubic spline model. Our experiment is patterned after the model verification results presented in Lin and Ortega [39]. As usual for the matching pursuit case, target rate is used for the control parameter, and so we investigate only the distortion model, $\hat{D}(\sigma)$.

The experiment is set up in the following way. Suppose motion compensation has been performed on some frame k, and the source residual image is available. C choices of σ are to be predicted using the models. We choose candidate rate points uniformly distributed between $\frac{1}{2}R_{tgt}$ and $3R_{tgt}$ for some target number of bits R_{tgt}. The combinations $\{N(\sigma), DZ(\sigma), QP(\sigma)\}$ needed to hit each target rate σ are derived using the 2-Pass quantizer design and are considered to be known. Each of the three model types is then used to predict the distortion which would result from encoding the motion residual for frame k using the quantizer and stopping point defined for each rate point σ.

We now briefly review the three model types which are compared in the experiment, and we specify the input data on which each model is based. For the Full Pre-Code model, an initial unquantized encoding pass to an appropriate number of atoms N_{max} is performed. The model is then used to predict the distortion for the quantizer and stopping point corresponding to each of the C candidate rate points. For the Reduced Pre-Code model, a reduced dictionary pass to some N_{max} atoms is performed on the current frame residual. Both full and reduced dictionary passes are performed on the previous frame residual, since these are needed to generate the previous frame mapping function as defined in Equation (5.21). The model is then used to predict the distortion at stopping point $N(\sigma)$ defined for each rate point σ. The specific quantizer design for rate point σ is not explicitly used by the model. Reduced Pre-Code prediction results are separately computed for each of the four reduced dictionary sets listed in Table 5.6.

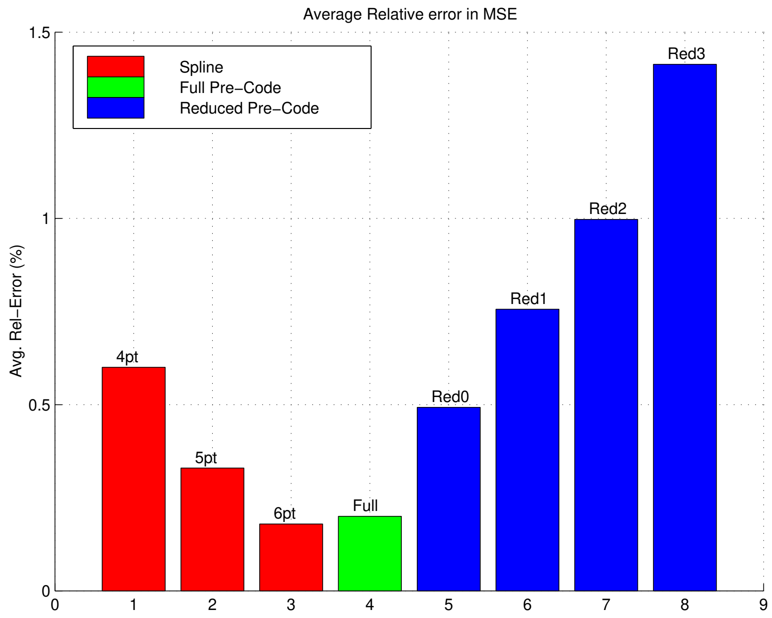
The cubic spline model is based on exactly computed distortion at N_c control points, i.e. $\{D(\sigma_j), j = 1...N_c\}$. Separate prediction results are generated for three different values of $N_c \in \{4, 5, 6\}$. Recall that additional control points improve the interpolation accuracy at the cost of additional model complexity. Control points σ_j are selected as a subset of the C candidate rate points on which the models are to be tested. This gives the spline model a slightly unfair advantage, since N_c of the C test points are known *exactly* to the model and thus are predicted without error. This is reasonable for a real application in which candidate rate points σ are specified in advance. For such a case, a subset of these points would naturally be chosen for the spline control points.

Experimental results are generated using a single frame from each of the 14 MPEG-4 test sequences. Each frame is then coded to $C = 16$ different candidate rate points distributed as described above. For each sequence, R_{tgt} is set to be the average bit rate per frame according to standard MPEG-4 test conditions. These conditions are shown in Table 1.2.

The resulting accuracy for the various models is shown in Figure 5.28. This figure shows that the Full Pre-Code model achieves about the same relative error as the 6-point cubic spline. The most accurate of the reduced complexity models, $Red0$, shows an accuracy somewhere between the 4-point and 5-point spline models. The remaining reduced complexity models are progressively less accurate. However, even the worst of the reduced complexity models, $Red3$, has an average relative error below 1.5 percent.

Using the cubic spline models as a benchmark, our experiments suggest that the Full Pre-Code models and the $Red0$ and $Red1$ reduced complexity models are accurate enough to be considered for use in (R, D) based optimization methods. Results are not conclusive for the less expensive $Red2$ and $Red3$ models, which were shown to be slightly less accurate. We further note that our models are not limited to the discrete levels of accuracy shown in Figure 5.28. Additional levels of accuracy can be effectively interpolated between these levels by either adding redesigned reduced-complexity dictionaries, or by applying the Hybrid Model extension as demonstrated in Figure 5.26.

Figure 5.28: Average Relative-Error Comparison.



Complexity of Spline Model

At this point, it is logical to ask the following question: "If the accuracy of the cubic spline is equal to or greater than that of the matching pursuit models developed earlier, then why not replace these models with the spline?" The answer stems from complexity considerations, which must be considered separately for each matching pursuit quantizer design. To simplify the discussion, we consider predicted (R, D) for a single frame over some known set of σ choices.

Suppose either the Fixed or 1-Pass quantizer designs are to be used. For these cases, the quantizer is known at the start of coding, and so a single encoding pass to N_{max} atoms is required in order to define the exact (R, D) control points to be used with the spline model. However, this same single encoding pass will trace out all the exact (R, D)

tradeoffs for encoding the given frame, thus removing the need for a model. For this reason, the spline model should not be used for Fixed or 1-Pass quantizer designs.

For 2-Pass quantizer design, a single unquantized matching pursuit pass to N_{max} atoms is needed to define the quantizers for each rate to be considered. Once the quantizers are defined, each control point used to define the spline requires a matching pursuit completion pass with a rate-specific quantizer. An additional N_c such passes must thus be performed. The resulting model does give a complexity reduction over exact (R, D) computation so long as the number of (R, D) points interpolated from the model is greater than the initial investment N_c. However, the Full Pre-Code model obtains similar accuracy and is completely specified by the single analysis pass which was used to define the quantizers. Full Pre-Code is thus about $(1 + N_c)$ times less complex than the spline, and so is a better modeling choice.

5.6 Conclusion

In this chapter, we developed the first known rate-distortion models for the matching pursuit algorithm. Two basic model classes were shown. The first model class, Full Pre-Code, is an analytical model based on a full matching pursuit encoding pass. It works by modeling the effects of a changing quantizer. It is thus especially useful for modeling systems with 2-Pass adaptive quantizer design, since in this case the quantizer varies with the target rate. A second model class, Reduced Complexity Pre-Code, was also developed. This empirical model is built on a matching pursuit pass with a reduced complexity dictionary, and so reduces the analysis complexity below that of one full dictionary pass per frame. This model class is most useful for situations where the quantizer does not change with the rate, such as Fixed or 1-Pass adaptive quantizer design. A hybrid cross between the two model classes was presented in order to allow a continuous set of complexity vs. accuracy tradeoffs.

We also investigated the suitability of these models for model-based (R, D) generation within optimal bit allocation algorithms. In doing so, we first united the various optimization algorithms and (R, D) models under a common "rate-distortion farming" framework. We then reviewed four recent optimization methods from literature and developed a "dependency tree" framework under which three of the methods could be described. A compact and useful dependency tree notation was developed in order to facilitate complexity analysis of the various methods. Such complexity analysis was then performed for all four optimization methods, first for exact (R, D) generation, and then for the reduced complexity case using models.

Our main conclusion is that both model classes may be usefully applied to reduce the complexity of (R, D) based optimization algorithms. We base this conclusion on two results presented in the final section. First, significant model-based complexity reductions were demonstrated for all four optimization algorithms in combination with nearly all matching pursuit quantizer design methods. Such complexity analyses also showed the complementary nature of the two model classes, since each model is appropriate for different combinations of optimization method and quantizer design. Second, the accuracy of our models was shown experimentally to be comparable to the cubic spline model of Lin and Ortega [39], which has been successfully used in a model-based optimization algorithm.

This suggests that our models are sufficiently accurate for use within similar optimization methods. Additional work is needed to adapt one or more of the optimization techniques to matching pursuit before the models may be usefully applied.