COMPLEXITY-DISTORTION TRADEOFFS IN IMAGE AND VIDEO COMPRESSION

by

Krisda Lengwehasatit

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
in Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

May 2000

Contents

List of Figures

List of Tables

Abstract

With the emergence of the Internet, a broader range of information transmission such as text, image, video, audio, etc. is now ubiquitous. However, the growth of data transferring is not always matched by the growth in available channel bandwidth. This has raised the importance of compression especially for images and video. As a consequence, compression standards for image and video have been developed since the early 90's and have become widely used. Those standards include JPEG [1] for still image coding, MPEG1-2 [2] for video coding and H.261/263 [3] for video conferencing.

We are motivated by observing that general purpose workstations and PCs have increased their speed to a level where performing compression/decompression in software of images and even video, can be done efficiently at, or near, real-time speed. Examples of this trend include software-only decoders for the H.263 video conferencing standard, as well as the wide use of software implementations of the JPEG standard to exchange images over the Internet. This trend is likely to continue as faster processors become available and innovative uses of software, for example usage of JAVA applets, become widespread. Optimizing the performance of the algorithms for the specific case of software operation is becoming more important.

In this thesis we investigate variable complexity algorithms. The complexities of these algorithms are input-dependent, i.e., the type of input determines the complexity required to complete the operation. The key idea is to enable the algorithm to classify the inputs so that unnecessary operations can be pruned. The goal of the design of the variable complexity algorithm is to minimize the average complexity over all possible input types, including the cost of classifying the inputs. We study two of the fundamental operations in standard image/video compression, namely, the discrete cosine transform (DCT) and motion estimation

(ME).

We first explore variable complexity in inverse DCT by testing for zero inputs. The test structure can also be optimized for minimal total complexity for a given inputs statistics. In this case, the larger the number of zero coefficients, i.e., the coarser the quantization stepsize, the greater the complexity reduction. As a consequence, tradeoffs between complexity and distortion can be achieved.

For direct DCT we propose a variable complexity fast approximation algorithm. The variable complexity part computes only DCT coefficients that will not be quantized to zeros according to the classification results (in addition the quantizer can benefit from this information by by-passing its operations for zero coefficients). The classification structure can also be optimized for a given input statistics. On the other hand, the fast approximation part approximates the DCT coefficients with much less complexity. The complexity can be scaled, i.e., it allows more complexity reduction at lower quality coding, and can be made quantization-dependent to keep the distortion degradation at a certain level.

In video coding, ME is the part of the encoder that requires the most complexity and therefore achieving significant complexity reduction in ME has always been a goal in video coding research. There have been several algorithms with variable complexity for ME. However, most of the research concentrate on reducing the number of tested vector points. We propose two fast algorithms based on fast distance metric computation or fast matching approaches. Both of our algorithms allow computational scalability in distance computation with graceful degradation in the overall image quality. The first algorithm exploits hypothesis testing in fast metric computation whereas the second algorithm uses thresholds obtained from partial distances in hierarchical candidate elimination.

Chapter 1

Introduction

1.1 Overview of compression

All current image and video compression standards are based on the same concept of transform based coding, illustrated by Figure 1.1. Basic building blocks of a transform based image/video encoder include *(i) blocking*, where data is partitioned into a smaller unit known as block (with size 8x8 pixels) or macroblock (16x16 pixels), *(ii) motion estimation (ME)* in predictive mode of video coding to exploit the temporal redundancy, *(iii) Discrete Cosine Transform (DCT)* to decompose the signal into its different frequency components, *(iv) quantization* to reduce the amount of information down to a level suitable for the channel (while introducing distortion) and *(v) entropy coding* to compress the quantized data losslessly. The decoding operation performs entropy decoding, inverse quantization, inverse DCT (IDCT) and motion compensation, sequentially to obtain the reconstructed sequence.

Typically, standards define only the syntax of the bit-stream and the decoding operation. They normally leave some room for performance improvement via better bit allocation at the encoders. Usually the more recent standards which provide better compression performance, have also more funtionalities and require more complex operation. With the existing standards, the coding gain comes at the price of significant complexity at major components of the basic structure such

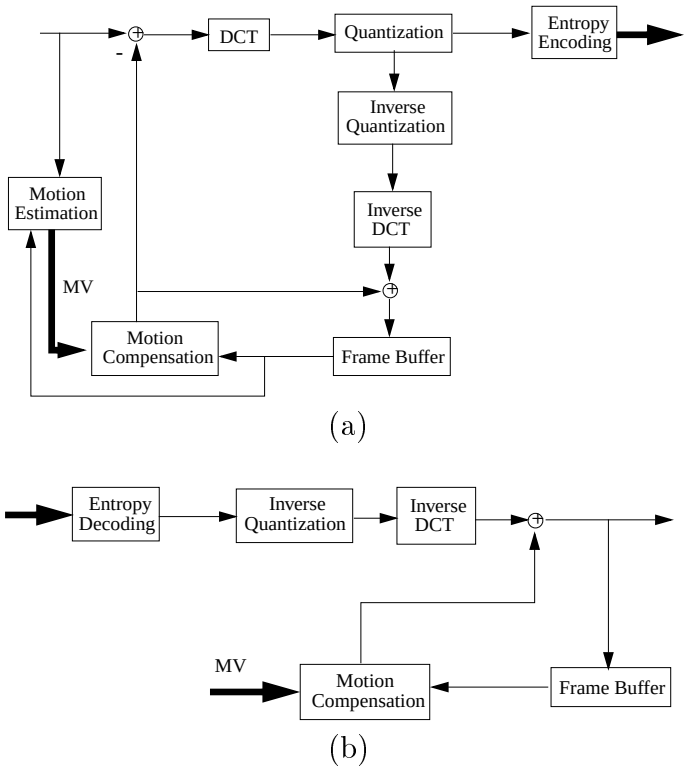


Figure 1.1: (a) Video encoding and (b) decoding system.

as DCT, inverse DCT and motion estimation. With the advent of real-time applications such as video conferencing, complexity issues of the codec have become more important. In many applications, including decoding on general purpose computers or on portable devices, significant complexity reduction is needed before video can be supported, especially if high resolution is required. Table 1.1 shows an example of a typical profile of computational complexity, listing the percentage of time spent in each of the major components in MPEG2 encoding. We can see that in a video encoding system, motion estimation is the most time consuming task. With fast motion search, one can achieve speedups of a factor of 4-5. However, the complexity still remains significant. Besides ME, video encoders have to perform DCT/IDCT, which is also a major complexity component. In Chapter 2-5, we will study the complexity-distortion tradeoff of these two components (DCT and ME) based on a variable complexity algorithm framework in which the complexity is input-dependent. We will propose algorithms such that

Table 1.1: Profile of component in MPEG2 encoder for the sequence "mobile& calendar". ES: exhaustive search, ST1: a spatio-temporal fast search [4], PDS: partial distance search.

component	ES	ES-PDS	ST1	ST1-PDS
Motion estimation	86.6%	69.3%	22.9%	20.2%
Quant. + Inv.Quant	3.7%	8.3%	20.0%	21.7%
DCT + IDCT	1.9%	5.9%	13.0%	12.6%
Others	7.8%	16.5%	44.1%	45.5%
Relative total time	1	0.44	0.2142	0.2106

their structure can be optimized for a given type of input sequences or images, so that the total complexity is minimal on the average sense.

1.2 Rate-distortion and complexity

In information theory, entropy is defined as a measure of the amount of information contained in random data. The amount of information can be expressed in terms of the number of bits needed to represent the data, so that more bits are needed to represent data containing more information. Entropy represents the amount of average information of a random source and also serves as a lower bound for the average code length needed to represent that source. The entropy of a random sequence $\bar{X}$ is defined as

$$H(\bar{X}) = -\sum_{x} p(\bar{x}) \log p(\bar{x}) \qquad \text{bits}$$

where $p(\bar{x})$ is probability mass function. An entropy encoder maps source symbol to codeword, $\mathcal{E} : \bar{x} \rightarrow c(\bar{x})$. From information theory one cannot compress data beyond the entropy (Shannon's source coding theorem). If we want to further compress the data, distortion must be introduced as a tradeoff and the goal is to find the minimal rate for a given distortion (Shannon's Rate-Distortion theorem [5]), i.e., one wants to find a coder with minimal rate for a given distortion constraint. Similar to the entropy, the rate-distortion function is defined as the

minimum rate such that the average distortion satisfies a distortion constraint.

Image and video compression algorithms aim at achieving the best possible Rate-Distortion (R-D) performance. Three main approaches are typically considered to improve R-D performance, namely, better transformation ([6, 7, 8]), quantization bit allocation ([9, 10, 11]), and efficient entropy coding ([12]). There are a number of algorithms that provide high efficiency in each of these areas. Examples include using the wavelet transform with progressive quantization and efficient entropy coding ([13, 14, 15, 16, 17, 18]), DCT with optimal thresholding and quantization [19], DCT with progressive coding [20], variable block size DCT [21], etc. Moreover, in all the above methods, complexity is not taken into account explicitly.

Shannon's Rate-Distortion theorem provides only an asymptotic result which is only valid for infinite memory coders. Typically, encoders with longer memory, more computational power and larger context statistics can perform better than encoders using less resources. In complexity-constrained environment, such as in software implementation of real-time video en/de-coding systems and in battery-limited pervasive devices, complexity becomes a major concern in addition to rate-distortion performance. Normally, one would prefer to have a system that encodes or decodes with higher frame rate with a small degradation in picture quality rather than to have a slightly better rate-distortion performance with much more complexity or delay. Thus, in order to achieve the best of rate-distortion-complexity performance, all three factors must be explicitly considered together.

With the fast increase in the clock speed and performance of general purpose processors, software-only solutions for image and video encoding/decoder are of great interest. Software solutions result in not only cheaper systems, since no specialized hardware needs to be bought, but also provide extra flexibility as compared to a customized hardware. In a software implementation, there are many factors that impact the computational complexity in addition to the number of arithmetic operations. For example, conditional logic, memory access or caching, all have an impact in overall speed. However, most of the work that has studied complexity issues has focused on arithmetic operation (additions and multiplications), and generally has not considered other factors. The work by Gormish [22]

and Goyal [23] are examples of research that addresses complexity-rate-distortion tradeoffs. In this research the transform block size is used to control the complexity, i.e., when the block size increases, the required complexity also increases while rate-distortion performance improves. Note, however, that complexity is determined only by the block size, and therefore, it is constant regardless of the input. Thus, as in most earlier work in this topic, complexity is analyzed only as a worst-case. Instead, in this thesis we will concentrate on algorithms designed to perform well *on average*.

1.3 Variable Complexity Algorithm (VCA)

In this work, we are interested in reducing the computational complexity of an algorithm, where complexity could be measured as actual elapsed time, CPU clock cycle or the number of operations consumed by the process. We develop algorithms with input-dependent complexity (see Fig. 1.2), such that for some types of input the complexity is significantly reduced, whereas for some other types the complexity may be larger than the original complexity (i.e., that achieved with an input-independent algorithm). For example, the IDCT does not have to be performed if the input block is all zero. The overhead cost of testing for the all-zero block is then added to the complexity of not-all-zero block, but, if the percentage of all-zero block is large enough, the complexity savings can outweigh the testing overhead. The same can be applied to more sophisticated VCAs in which the final goal is to achieve less complexity *on the average*. We will show that by using VCAs, it is possible to achieve a better complexity-distortion performance for a given rate than reported in [22].

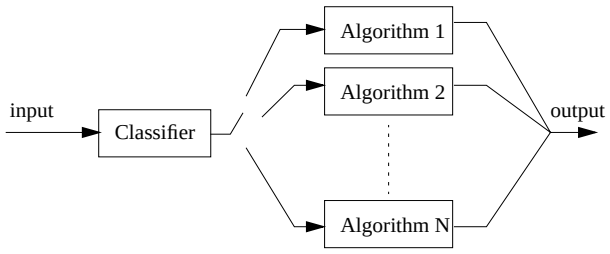


Figure 1.2: Variable complexity algorithm scheme.

In order to define an efficient VCA, as depicted in Fig. 1.2, it will be necessary to study 3 issues:

1. **Input Classification**

   In order for the complexity of an operation to be variable, the input must be classified into different classes where each class requires different amount of computation to complete the operation. Conceptually, we can say that a different algorithm (a "reduced algorithm") is associated with each class. The complexity of each reduced algorithm may not be the same and depends on the input class. The classification scheme thus plays an important role in VCA framework. This idea is similar to entropy coding where the code-length of the codeword corresponding to each input symbol can be different.

2. **Optimization**

   The goal of the input classification is to achieve average-case complexity that is below the complexity achieved when no classification is used and a single algorithm is used for all inputs. However, the classification process itself comes at the cost of additional complexity. Therefore, we use statistical information to design a classification algorithm such that the total complexity *including the classification cost* is minimum on the average. This is achieved by eliminating those classification tests that are not worthwhile, i.e., those that increase the overall complexity. This is analogous to the entropy code design problem where the codebook is designed in such a way that the average code-length is smaller than the fixed length code. Normally, symbols with higher probability are given shorter codewords and rare symbols are assigned longer codewords. However, in the complexity case, there are two factors in determining the total complexity, namely, the cost of classification and the cost of the reduced algorithms, which vary for different types of input. Unlike the entropy coding, here it is not possible to guarantee that the most likely inputs are assigned the least complex algorithms, since the complexity is a function of the input itself. Thus, in minimizing the complexity, we will consider both classification cost and reduced algorithm complexity along with the input statistics.

3. **Computational scalability**

   In order to gain additional complexity reduction, it may be necessary to sacrifice to some extent the quality of the algorithm. Normally, computational scalability can be introduced in an algorithm at the cost of increased distortion, or higher rate for the same distortion. The complexity-distortion tradeoffs are analogous to rate-distortion tradeoffs, i.e., for a given complexity constraint, we want to find an algorithm which yields the smallest distortion while satisfying the complexity budget. This complexity-distortion problem is more open than its rate-distortion counterpart because the complexity depends on the platform the system is running on. In this thesis, we present computationally scalable algorithms which perform reasonably well on Unix and PC platforms, but our methods can be easily extended to small embedded machine.

In this dissertation, we present variable complexity algorithms (VCA) for 3 main components of standard video coders, namely, IDCT, DCT and motion estimation. For each of these algorithms we studied the three main issues discussed above. In Sections 1.4 and 1.5, we give a comprehensive literature survey of DCT/IDCT and motion estimation, and provide summaries of our contributions.

1.4 Discrete Cosine Transform Algorithms

1.4.1 Definition of DCT

The Discrete Cosine Transform (DCT) [24], first applied to image compression in [25], is by far the most popular transform used for image compression applications. Reasons for its popularity include not only its good performance in terms of energy compaction for typical images but also the availability of several fast algorithms. Aside from the theoretical justifications of the DCT (as approximation to the Karhunen-Loeve Transform, KLT, for certain images [24]) our interest stems from the wide utilization in different kinds of image and video coding applications. The well-known JPEG and MPEG standards use DCT as their transformation to

decorrelate input signal (see [1]). Even with the emergence of wavelet transforms, DCT has still retained its position in image compression. While we concentrate on the DCT, most of our developments are directly applicable to other orthogonal transforms.

The N point DCT $\bar{\mathbf{X}}$ of vector input $\bar{\mathbf{x}} = [x(0), x(1), ..., x(N-1)]^T$ is defined as $\bar{\mathbf{X}} = \mathbf{D_N} \cdot \bar{\mathbf{x}}$ where $\mathbf{D_N}$ is the transformation matrix of size NxN with elements $\mathbf{D_N}(i,j)$

$$\begin{aligned} \mathbf{D_N}(i,j) &= \frac{c_i}{2} \cdot \cos \frac{(2j+1)i\pi}{2N} \qquad (1.1) \\ \text{where} \qquad c_i &= \begin{cases} \frac{1}{\sqrt{2}} & \text{for } i = 0 \\ 1 & \text{for } i > 0 \end{cases} \end{aligned}$$

Conversely, the inverse transform can be written as $\bar{\mathbf{x}} = \mathbf{D_N}^T \cdot \bar{\mathbf{X}}$ given the orthogonality property. The separable 2-D transforms are defined as

$$\begin{aligned} \mathbf{X} &= \mathbf{D_N} \cdot \mathbf{x} \cdot \mathbf{D_N}^T \\ \text{and} \qquad \mathbf{x} &= \mathbf{D_N}^T \cdot \mathbf{X} \cdot \mathbf{D_N}, \end{aligned}$$

respectively, where $\mathbf{X}$ and $\mathbf{x}$ are now 2-D matrices. This means that we can apply DCT or IDCT along rows first then across columns of the resulting set of coefficients, or vice versa, to obtain the 2-D transform. Each basis in the DCT domain represents an equivalent frequency component of the spatial domain real data sequence. After applying the DCT to a typical image, DCT coefficients in the low frequency region contain most of the energy. Therefore, DCT has a good energy compaction performance.

Computing the DCT/IDCT directly following the definition requires N^2 multiplications for a 1-D transform. There have been many fast algorithms proposed to reduce the number of operations. There are two ways to categorize these fast algorithms. First, they can be categorized as either exact or approximate algorithms depending on whether the transform result follows the DCT definition or (slightly) differs from the definition, respectively. Second, they can be categorized based on their complexities. If the complexities are fixed regardless of the input,

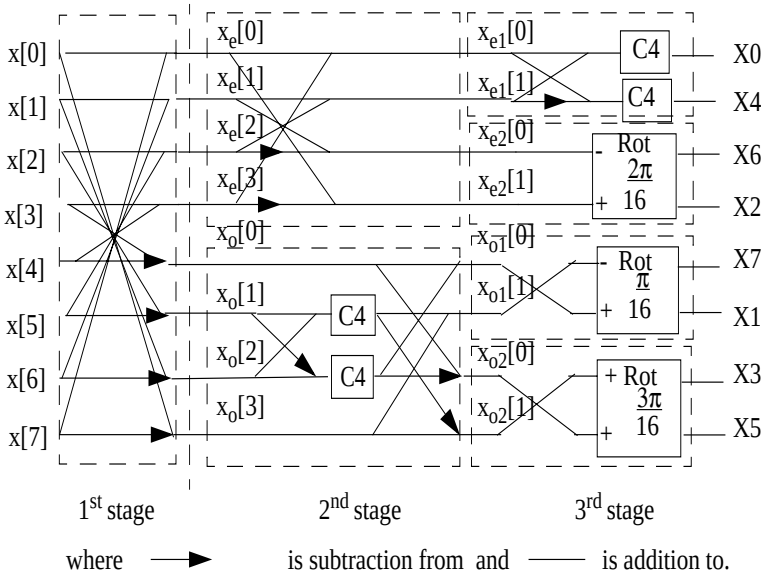
they are fixed complexity algorithm. On the other hand, if the complexity is input dependent, they are *variable complexity algorithms* (VCAs), i.e., there are different algorithms for different types of input (see Figure 1.2). Another related issue is computational scalability where the complexity can be adjusted with the penalty of distortion tradeoffs. We now review previously proposed algorithms for fast implementation of DCT/IDCT, classifying them according to the approaches that are used.

1.4.2 Exact vs. Approximate DCT

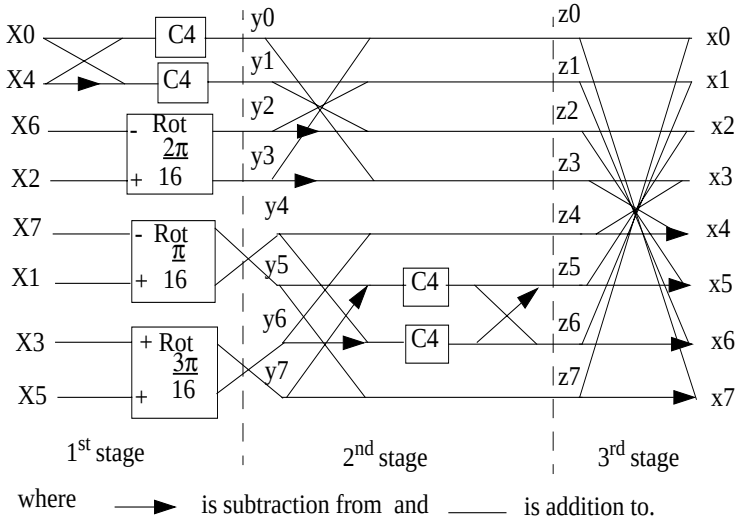
Exact Algorithms

Since elements in the DCT transformation matrix are based on sinusoidal functions, significant reductions in complexity can be achieved. For example, with the availability of the well known Fast Fourier Transform (FFT), one can obtain the DCT using the FFT and some pre-post processings as shown in [26] and [27]. A direct DCT computation fast algorithm was first proposed by Chen *et al.* in [28]. Since then, there are several other fast DCT algorithms proposed such as [29], [30] and [31], which aim at achieving the smallest number of multiplications and additions. The minimal number of multiplications required for a 1-D DCT transform was derived by Duhamel *et. al.* in [32]. Loeffler *et. al.* in [33] achieves this theoretical bound for size-8 DCT. An example of fast algorithm based on Vetterli-Ligtenberg's algorithm [29] for 1-D size-8 DCT and IDCT is depicted in Figures 1.3(a) and (b), respectively. This algorithm requires 13 multiplications and 29 additions and the structure is recursive, i.e., the top 4 branches starting from the 2nd stage correspond to a size-4 DCT. Table 1.2 shows the complexity of various other algorithms for DCT computation in terms of number of additions and multiplications.

It has been shown that a fast algorithm for 2-D requires less arithmetic operations than using 2 fast 1-D algorithms separately ([34], [35]). Similar to 1-D DCT, the theoretical bound for higher dimension was derived by Feig and Winograd in [36] and the bound can be achieved as pointed out by Wu and Man in [37] by incorporating 1-D algorithm of [33] to a 2-D algorithm by Cho and Lee in [38].



(a)



(b)

Figure 1.3: A fast (a) DCT (b) IDCT algorithm introduced by Vetterli-Ligtenberg where 'Rot' represents the rotation operation which takes inputs $[x, y]$ and produces outputs $[X, Y]$ such that $X = x\cos Q + y\sin Q$ and $Y = -x\sin Q + y\cos Q$, $C4 = 1/\sqrt{2}$.

Table 1.2: Number of operations required to compute a 1-D size N DCT using various fast algorithms (the number in the brackets is obtained when N=8).

Algorithm	**multiplications**	**additions**
matrix multiplication	N^2 [64]	$N(N-1)$ [56]
Chen *et.al*'77 [28]	$N\log_2 N - 3N/2 + 4$ [16]	$3N(\log_2 N - 1)/2 + 2$ [26]
Wang'84 [30]	$N(\frac{3}{4}\log_2 N - 1) + 3$ [13]	$N(\frac{7}{4}\log_2 N - 2) + 3$ [29]
Lee'84 [31]	$\frac{N}{2}\log_2 N$ [12]	$3\frac{N}{2}\log_2 N - N + 1$ [29]
Duhamel'87 [32] (Theoretical bound)	$N/2 - \log_2 N - 2$ [11]	n/a

Moreover, there are several other algorithms aiming for different criteria. For example, an algorithm for fused MULTIPLY/ADD architecture introduced in [39], a time-recursive algorithm [40] and a scaled DCT algorithm which computes a scaled version of the DCT, i.e., in order to get the exact DCT the scaling factor must be introduced in the quantization process. A scaled DCT algorithm proposed in [41] gives a significant reduction of the number of multiplications. Research is still ongoing on the topic of fast algorithms, but current goals in this research may involve criteria other than adds and multiplies, e.g., providing favorable recursive properties [42]. Some of them avoid multiplications by using a look-up table [43]. It is also worth mentioning the work by Merhav and Bhaskaran [44] in which image manipulations such as scaling and inverse motion compensation are done in the transform domain. This saves some computations as compared to doing the inverse transform, processing, and then the forward transform separately. The gain can be achieved given the sparseness of the combined scaling and transform matrix.

Approximate Algorithms

All of the above algorithms have aimed at computing the exact DCT with minimal number of operations. However, if the lower bound in the number of operations has been reached, obviously it is no longer possible to reduce the complexity while maintaining an exact algorithm. One way to further reduce the number of

operations is to allow the resulting computed coefficients to be different from their defined values, i.e., allow some errors (distortions) in the DCT computation. The following approaches are representative of those proposed in the literature.

One approach is to compute some of the DCT coefficients which represent low frequencies. Empirically, for typical images most of the energy is in the low frequencies as can be seen in Figs. 1.4 and 1.5. Thus, ignoring high frequency coefficients still results in acceptable reconstructed images in general. Earlier, a related work on computing the DFT with a subset of inputs/outputs is proposed by Burrus *et.al* [45]. This work analyses the number of operations required for the Discrete Fourier Transform (DFT) and provides an algorithm with minimal number of operations (so-called "pruned DFT") for the case where only a subset of input or output points are needed. This is achieved by pruning all the operations that become unnecessary when only a subset of input/output points is required. However, the work was restricted to the case where the required subset is known or fixed and where the required input/output points are always in order. For the DCT, one could compute only the first 4 coefficients of a size-8 DCT as in [46]. An alternative is to preprocess the data to get rid of empirically unnecessary information before performing a smaller size DCT. In [47] and [48], the simple Haar subband decomposition is used as the preprocessing. Then a size-4 DCT is performed on low band coefficients and the result is scaled and used as the first 4 coefficients of a size-8 DCT. For IDCT, a reduced size input in which only DC, 2x2 or 4x4 DCT coefficients are used to reconstruct the block has been implemented in [46]. Similarly, the reduced size output IDCT can also be used ([49, 50]) for applications such as reduced resolution decoder and thumbnail viewing. However, for video applications the motion drift problem can seriously deteriorate the video quality when an approximate IDCT is used instead of the exact one.

Another approach is to use the distributed arithmetic concept. DCT coefficients can be represented as a sum of the DCT of each input bit-plane, which can be easily computed by a look-up table [51]. Since the contribution of the least significant bit-plane of input is small, and most likely the LSB represents noise in image capturing process, the operation for those bit-planes can be removed. This idea originally came from DFT computation in [52] where it is called SNR

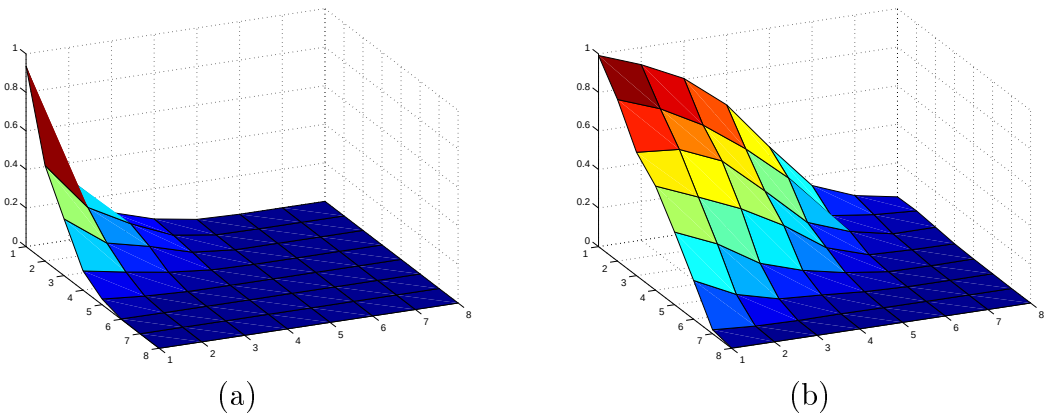
update. In [53], a class of approximate algorithms which is multiplication-free are proposed, and the error is analyzed. This method is similar to the distributed arithmetic approach in the sense that all DCT coefficients (not just a subset of coefficients) are computed, but with less accuracy.

1.4.3 VCA DCT

One common feature in all the fast algorithms discussed above (both exact DCT and approximate DCT) is that they aim at reducing the complexity of a generic direct or inverse DCT, *regardless of the input to be transformed*. This is obvious in the case of exact algorithms, since these have a fixed number of computations. Even for the approximate algorithms we just discussed, any input is computed in the same way, even though obviously some blocks suffer from more error than others. Therefore complexity is estimated by the number of operations which is the same for every input. In other words, all of the above algorithms lack adaptivity to the type of input. In this thesis we consider as possible operations not only typical additions/multiplications but also other types of computer instructions (for example `if, then, else`) such that additional reductions in complexity are possible, in an average sense, if the statistical characteristics of the inputs are considered.

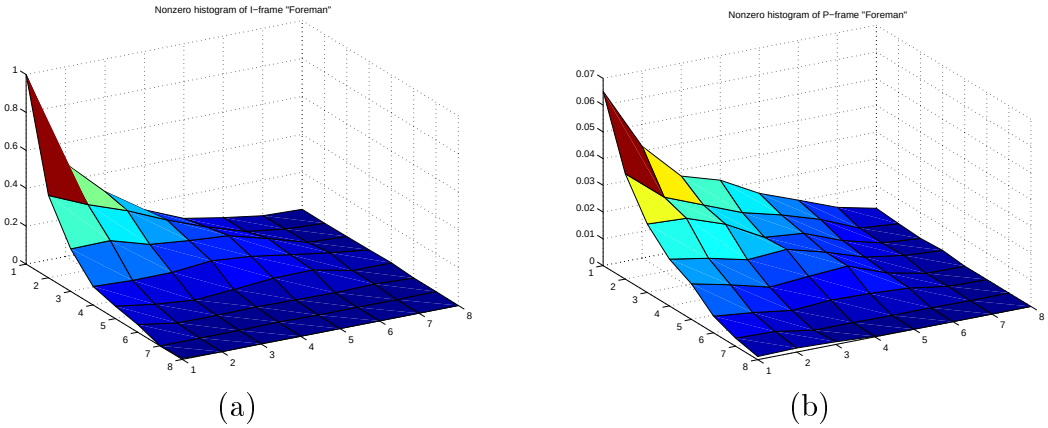
To explain the improved performance achieved with input dependent algorithms, consider the analogy with vector quantization (VQ). In VQ, the input can be a combination of many sources with different characteristics, e.g., background area, edge area and detail area for image. It was shown in [54] that by first classifying the source into different classes and then applying a VQ designed for each class, the rate-distortion performance is better than having a single VQ codebook. In the same context, [55] shows that by classifying a block of image into different classes with different VQ codebook reduces the overall computational complexity on the average.

In the case of DCT, we motivate the benefits of input-dependent operation by considering the sparseness of quantized coefficients in Figs. 1.4 and 1.5. Note that, as expected, the high frequency coefficients are very likely to be zero. This will also be the case for the difference images encountered in typical video coding



(a) (b)

Figure 1.4: Frequency of nonzero occurrence of 8x8 quantized DCT coefficients of "lenna" image at (a) 30.33 dB and (b) 36.43 dB.



(a) (b)

Figure 1.5: Frequency of nonzero occurrence of 8x8 quantized DCT coefficients of 10 frames for "Foreman" at QP=10 (a) I-frame (b) P-frame.

scenarios (e.g. P and B frames in MPEG), where the percentage of zero coefficients is likely to be even higher (see Fig. 1.5).

Therefore, for IDCT, it is straightforward to check the content of an input block which has already been transformed and quantized. Taking advantage of the sparseness of the quantized DCT coefficients can be easily done and is a widely used technique in numerous software implementations of IDCT available in public domain software such as the JPEG implementation by the Independent JPEG Group [46], the MPEG implementation by U.C.Berkeley [56], vic, the UCB/LBL

Internet video conferencing tool [57], or TMN's H.263 codec [58]. All these implementations take into account the sparseness to achieve image decoding speed-up by checking for all-zero rows and columns in the block to be decoded, since these sets do not require a transform to be performed. Also checking for DC-only rows and columns is useful since the transform result is simply the constant scaled DC vector.

In all these approaches there is a trade-off given that additional logic is required to detect the all-zero rows and columns and so the performance of the worst case decoding is worse than if tests were not performed. However the speed up for the numerous blocks having many zeros more than makes up for the difference and, on average, these simple schemes achieve faster decoding for "typical" images. This simple all-zero test method makes the IDCT algorithm become *a VCA since the complexity of IDCT operations for each input block depends on the class of that block, and the class is determined by the number of zeros in the block.*

Another example of VCAs is by Froitzheim and Wolf [59] (FW), which formally addresses the problem of minimizing the IDCT complexity in an input dependent manner, by deciding, for a given block, whether to do IDCT along the rows or the columns first. Different blocks will have different characteristics and in some one of the two approach, row first or column first, may be significantly faster.

A more sophisticated classification of inputs for 2-D IDCT is proposed in [60] for software MPEG2 decoders with multimedia instructions. The input blocks are classified into 4 classes which are (1) block with only DC coefficient, (2) block with only one nonzero AC coefficient, (3) block with only 4x4 low frequency components and (4) none of the above. The first 3 classes are associated with reduced algorithms that require less operations than the algorithm for class (4) (which uses the baseline DCT algorithm).

For the case of forward DCT, an example of block-wise classification can be found in [61], where each block is tested prior to the DCT transform to determine whether its DCT coefficients will be quantized to zero or not. Given the values of the input pixels $x(i, j)$ the proposed test determines whether the sum of absolute values exceeds a threshold which is dependent on the quantization and the confident region. However, this work has the limitation of assuming a

single quantizer is used for all the DCT coefficients (thus it is better suited for for interframe coding scenarios) and can classify only all-zero DCT block.

1.4.4 Computationally Scalable DCT

Algorithms in which the complexity can be scaled depending on the level of accuracy, or the distortion, can be called scalable complexity algorithm. Therefore, scalable complexity algorithms are also approximate algorithms. In the context of IDCT, the scalability can be achieved by introducing distortion at either the decoding or encoding side. The decoder can perform approximate IDCT operation and obtain a lower quality reconstruction, or the encode can assign coarser quantization parameters to produce more DCT coefficients quantized to zero and therefore enable a faster decoding. Thus, in general, low quality images lead to low complexity decoding. In the case of predictive video coding, the encoder based scalability is preferred in order to maintain the synchronization between the encoder and decoder. In [62], optimal quantization assignment at the encoder is studied to obtain minimal distortion for a given decoding time budget.

For DCT, the mechanism to control the complexity is by adjusting the accuracy of the transform which in turns reflects in the degradation in the coding performance. Examples of this approach are Girod's [63] in which a DCT block is approximated using only DC and first two AC component. The approximation error is then obtained from the sum of absolute difference between the original block and the block reconstructed using only 3 DCT coefficients. This error is compared with a threshold which is a function of the quantization parameter and a desired level of accuracy. Pao & Sun [64] proposed a statistical sum of absolute value testing (SSAVT) which classifies the DCT block into several reduced output classes with controllable degree of confidence.

1.4.5 Thesis Contributions

In chapter 2, we study the benefits of input-dependent algorithms for the IDCT where the average computation time is minimized by taking advantage of the

sparseness of the input data. We show how to construct several IDCT algorithms. We show how, for a given input and a correct model of the complexity of the various operations, we can achieve the fastest average performance. Since the decoding speed depends on the number of zeros in the input, we then present a formulation that enables the encoder to optimize its quantizer selection so as to meet a prescribed "decoding time budget". This leads to a complexity-distortion optimization technique which is analogous to well known techniques for rate-distortion optimization. In our experiments we demonstrate significant reductions in decoding time. As an extension of this work, we address the generalized quadtree optimization framework proposed in [21] by taking the complexity budget into account and using a VCA IDCT to assess the complexity cost. Therefore, we have a complete rate-complexity-distortion tradeoff in the sense that not only quantization parameter but also the block size are optimally selected for the best rate-complexity-distortion performance. The work in this chapter was published in part in [65] and [62].

In chapter 3, we propose two classes of algorithms to compute the forward DCT. The first one is a variable complexity algorithm in which the basic goal is to avoid computing those DCT coefficients that will be quantized to zero. The second one is an algorithm that approximates the DCT coefficients, without using floating point multiplications. The accuracy of the approximation depends on the quantization level. These algorithms exploit the fact that for compression applications (i) most of the energy is concentrated in a few DCT coefficients and (ii) as the quantization step-size increases an increased number of coefficients is set to zero, and therefore reduced precision computation of the DCT may be tolerable. We provide an error analysis for the approximate DCT compared to SSAVT DCT [64]. We also propose 3 hybrid algorithms where SSAVT, approximate DCT, and VCA approaches are combined. This work was published in part in [66].

1.5 Motion Estimation

Motion estimation (ME) is an essential part of well-known video compression standards, such as MPEG1-2 [2] and H.261/263 [3]. It is an efficient tool for video

compression that exploits the temporal correlation between adjacent frames in a video sequence. However, the coding gain comes at the price of increased encoder complexity for the motion vector (MV) search. Typically ME is performed on macroblocks (i.e., blocks of 16×16 pixels) and its goal is to find a vector pointing to a region in the previously reconstructed frame (reference frame) that best matches the current macroblock (refer to Fig. 1.6). The most frequently used criterion to determine the best match between blocks is the sum of absolute differences (SAD).

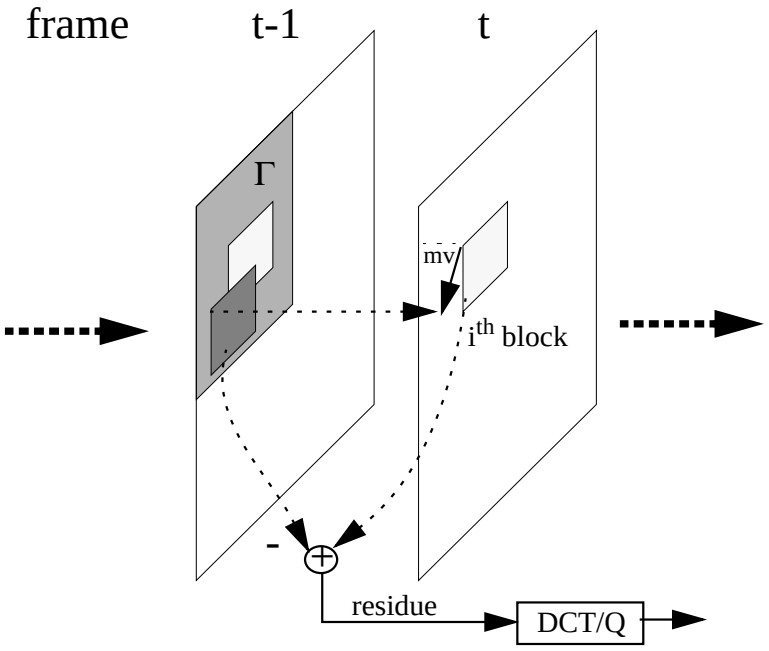


Figure 1.6: Motion estimation of i-th block of frame t predicted from the best block in the search region Γ in frame t-1.

1.5.1 Example: Conventional Exhaustive Search

We start by introducing the notation that will be used throughout the thesis (refer to Table 1.3). Let us consider the i-th macroblock in frame t. For a given macroblock and candidate motion vector $\vec{mv}$, let the sum of absolute difference

Table 1.3: Notation Table

$I_t(n_x, n_y)$	intensity level of (n_x, n_y) pixel relative to the upper-left-corner pixel of the macroblock.
B	the set of pixels constituting a macroblock
β	a subset of B
$\vec{mv} = (mv_x, mv_y)$	a candidate motion vector
$\Gamma = \{\vec{mv}\}$	the set of allowable $\vec{mv}$ in a search region e.g., $mv_x, mv_y \in \{-16, -15.5, ..., 15, 15.5\}\}$.
$\gamma \subset \Gamma$	a set of $\vec{mv}$ actually tested for a given search scheme

matching metric be denoted as $SAD(\vec{mv}, \beta)$, where[1]

$$SAD(\vec{mv}, \beta) = \sum_{(n_x, n_y) \in \beta} |I_t(n_x, n_y) - I_{t-1}(n_x + mv_x, n_y + mv_y)|, \tag{1.2}$$

and where β is a subset of the pixels in the macroblock. This notation will allow us to represent the standard SAD metric based on the set B of all pixels in a macroblock, as well as partial SAD metrics based on pixel subsets β. A ME algorithm will return as an output the best vector for the given search region and metric, $MV^*(\gamma, \beta)$, i.e. the vector out of those in the search region $\gamma \subseteq \Gamma$ that minimizes the SAD computed with $\beta \subseteq B$ pixels,

$$MV^*(\gamma, \beta) = \arg \min_{\vec{mv} \in \gamma} SAD(\vec{mv}, \beta).$$

In the literature, a search scheme is said to provide an *optimal* solution if it produces $MV^*(\Gamma, B)$, i.e., the result is the same as searching over all possible $\vec{mv}$ in the search region (Γ) and using a metric based on all pixels in the macroblock (B), $MV^*(\Gamma, B)$ can typically be found using an exhaustive full search (ES). In this thesis, we will term "exhaustive" any search such that $\gamma = \Gamma$ regardless of the particular β chosen.

In general, motion search is performed by computing the SAD of all the vectors

[1]Note that, since our approach will be the same for all macroblocks in all motion-compensated frames, we will not consider explicitly the macroblock and frame indices (i and t) unless necessary.

in the search region sequentially (following a certain order, such as a raster scan or an outward spiral), one vector at a time. For each vector, its SAD is compared with the SAD of the "best found-so-far" vector. Without loss of generality, let us assume that we are considering the i-th candidate vector in the sequence, $\vec{mv}_i$ for $i = 1, ..., |\Gamma|$, and we use B for the SAD computation. Thus we define the "best found-so-far" SAD as

$$SAD_{bsf}(\gamma_i, B) = \min_{\vec{mv} \in \gamma_i} SAD(\vec{mv}, B)$$

where $\gamma_i = \bigcup_{j=1}^{i} \{\vec{mv}_j\} \subseteq \Gamma$ is the set of vectors that have already been tested up to $\vec{mv}_i$ and the associated "best found-so-far" vector is denoted by $MV_{bsf}(\gamma_i, B)$. Note that when all vectors in the search region have been tested, $MV^*(\Gamma, B)$ is equal to $MV_{bsf}(\Gamma, B)$.

To complete the encoding process MV^* is transmitted. The residue block, which is the difference between the motion estimated block and the current block, is transformed, quantized, entropy coded and then sent to the decoder, where the process is reversed to obtain the reconstructed images.

1.5.2 Fast Search vs. Fast Matching

The computational complexity of motion search is a major concern for block-based video encoding systems with limited computation power resources. We now provide a quick overview of fast ME techniques. Our goal is to provide a rough classification of the various strategies that have been used to reduce complexity, and also to classify and clarify the novelty of the algorithms that will be introduced in Chapters 4 and 5.

The total complexity of the ME process depends on (i) the number of candidate vectors in the search region, Γ, and (ii) the cost of the metric computation to be performed for each of the candidates (e.g., computing a SAD based on the set B will be more costly than using a subset $\beta \in B$.) Thus, fast ME techniques are based on reducing the number of candidates to be searched (fast search) and/or the cost of the matching metric computation (fast matching).

Fast search (FS)

In order to improve the efficiency of the search, fast ME algorithms can restrict the search to a subset of vectors $\gamma \subset \Gamma$. This subset of vectors can be predetermined and fixed as in [67] or it can vary as dictated by the specific search strategy and the characteristics of the macroblock. Examples of the latter case are 2-D log search [68], conjugate directions and one-at-a-time search [69], new three step search [70], gradient descent search [71], and center-biased diamond search [72], which all exploit in various ways the assumption that the matching difference is monotonically increasing as a particular vector moves further away from the desired global minimum. For example, 2-D log search starts from a small set of vectors uniformly distributed across the search region and moves on to the next set more densely clustered around the best vector from the previous step (if there is a change in direction, otherwise, the next set would be the same farther apart). A good initial point can also be used to reduce the risk of being trapped in local minima. Approaches to find a good initial point include hierarchical and multiresolution techniques [73, 74, 75, 76, 77]. Another successful class of techniques seeks to exploit the correlations in the motion field, e.g., MVs of spatially and temporally neighboring blocks can be used to initialize the search as in [4] (referred to as the ST1 algorithm) and [78]. The ST1 algorithm employs the spatial and temporal correlation of motion vectors of adjacent macroblocks. It starts with the best candidate motion vectors from a set of neighboring macroblock both spatially and temporally, if available (Fig. 1.7 (a)). Then it performs local refinement on a small 3x3 window search until it reaches the minimum point (Fig. 1.7 (b)). In general, ST1 algorithm achieves a higher speed-up than 2-D log search, with also lower overall residue energy.

Fast matching (FM)

Another approach for fast ME, which can also be combined with a FS technique, consists of devising matching criteria that require less computation than the conventional sum of absolute difference (SAD) or mean square error (MSE). One example of this approach consists of computing a partial metric, e.g., the SAD based on $\beta \subset B$ [67]. Of particular relevance to our work are the partial distance

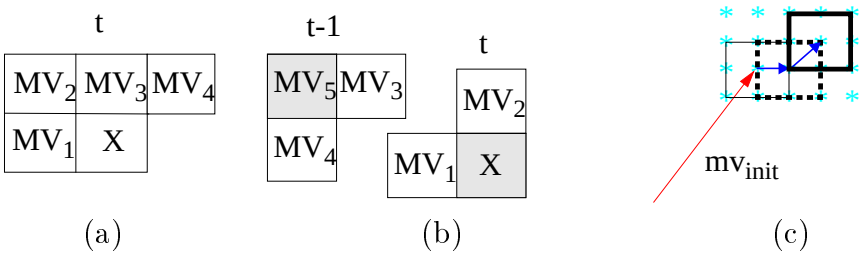


Figure 1.7: Summary of ST1 algorithm where (a) and (b) depict spatial and spatio-temporal candidate selection, respectively. Highlighted blocks correspond to the same spatial location in different frames, (c) illustrates the local refinement starting from the initial motion vector found to be the best among the candidates.

search techniques, which have also been proposed in the context of VQ [79, 54]. In a partial distance approach the matching metric is computed on successively larger subsets of B but the computation is stopped if the partial metric thus computed is found to be greater than the total metric of the "best found-so-far" vector. For example if $SAD(\vec{mv}_i, \beta \subset B) > SAD_{bsf}(\gamma_{i-1}, B)$ there is no need to complete the metric computation and calculate $SAD(\vec{mv}_i, B)$. Many implementations of FS algorithms include this partial distance technique to speed up their metric computation. Other early termination criteria have been proposed in [80]. Alternatively, matching metrics other than SAD or MSE can also be used. For example, in [81], adaptive pixel truncation is used to reduce the power consumed. In [82], the original (8-bit) pixels are bandpass filtered and edges are extracted, with the final result being a binary bit-map that is used for matching. Other approaches include hierarchical feature matching [83], normalized minimum correlation techniques [84], minimax matching criterion [85].

1.5.3 Fixed vs. Variable Complexity

We can also classify ME techniques into fixed complexity algorithm (FCA) and variable complexity algorithm (VCA). The complexity in FCA is input-independent and remains constant (e.g., a ME technique with fixed β and γ), while in this work we will consider VCA, where complexity is input dependent (e.g. β and γ are different for each macroblock and/or frame.) The goal when designing a VCA is then to achieve low complexity in the average case. Thus, we expect the "worst

case" complexity of the VCA to be higher than that of a comparable FCA, but hope that on the average, a VCA will have lower complexity. In practice, this is done by making reasonable, though typically qualitative, assumptions about the characteristics of typical sequences. For example, consider the algorithm of [4], which, as indicated earlier, exploits the correlations in the motion field. For this algorithm, performing ME in a scene with smooth motion (e.g. a scene with panning) tends to require less complexity (and to be closer to the optimal ES result) than finding the motion field for a scene with less correlated motion (e.g. a scene with several independent moving objects). Thus, such an algorithm provides a good average case performance under the assumption that typical video sequences have predominantly smooth motion. For similar reasons, algorithms in [68, 69, 70, 72] perform well for sequences with small motions.

A second example of a VCA algorithm can be found in the partial distance approach discussed earlier. The underlying assumption here is that the distribution of SADs for typical blocks has large variance, with few vectors having SAD close to the minimum (i.e. the SAD of the optimal vector). Thus, on average one can expect to eliminate many bad candidate vectors early (those having large metric) and thus to achieve a reduction in overall complexity. Once again this is making an implicit assumption about the statistical characteristics of these matching metrics for typical blocks. In this thesis we argue that substantial gains can be achieved by making these assumptions *explicit*, and therefore our probabilistic stopping criterion for the metric computation will be based on explicit statistical models of the distribution of SAD and partial SAD values (see Chapter 4).

1.5.4 Computationally Scalable Algorithms

Finally, we consider the computational scalability property, which is a desirable feature in many applications (e.g., to operate the same algorithm in different platforms, or to run at various speeds in the same platform). Computational scalability allows to trade-off speed with performance (e.g., the energy of the prediction residue in the case of ME). There has been some recent interest in computation scalability in the context of video coding in general and ME in particular. For example, [86] addresses computationally constrained motion estimation where the

number of vectors to be searched (the size of γ) is determined by a complexity constraint based on a Lagrangian approach. This technique adopts an idea similar to that in [87] but using complexity rather than rate as a constraint.

1.5.5 Thesis Contributions

In this thesis, we will focus on FM approaches based on the partial distance technique. In Chapter 4 we propose a novel fast matching algorithm to help speedup the computation of the *matching* metric, e.g., the sum of absolute difference (SAD), used in the search. Our algorithm is based on a partial distance technique in which the reduction in complexity is obtained by terminating the SAD calculation once it becomes clear that the SAD is likely to exceed that of the best candidate so far. This is achieved by using a hypothesis testing framework such that we can terminate the SAD calculation early at the risk of missing the best match vector. Furthermore, we discuss how the test structure can be optimized for a given set of statistics, so that unnecessary tests can be pruned. This work was first introduced in [88] and further refined in [89].

It should be emphasized that the FM techniques we propose can be applied along with any FS strategy and any other additive metrics such as MSE. We also note that, while our experimental results are provided for a software implementation, focusing on FM approaches may also be attractive in a hardware environment. For example, from a hardware architecture point of view, some FS designs have the drawback of possessing a non-regular data structure, given that the blocks that have to be searched in the previous frame depend on the selection of initial point. Thus the set of candidates considered varies from macroblock to macroblock. Conversely, ES algorithms have the advantage of operating based on a fixed search pattern (this could also facilitate parallelizing the algorithm). In general, FS algorithms such as that in [4] will have to be modified for hardware implementation, with one of the main goals being to minimize the overhead, even though it is relatively small, due to the non-regularity of the algorithm. As an alternative, if the goal is an efficient hardware design one may choose to design an efficient FM approach (e.g., [81, 90, 91, 92, 93]) and combine it with a simple search technique, such as ES.

In Chapter 5, we present a new class of fast motion estimation techniques that combine both fast search and fast matching based on partial distances. In these algorithms, the computation of the matching metric is done in parallel for all candidates. Unlike other fast search techniques that eliminate candidates based on the spatial location of the candidate, this technique uses only the partial distance to eliminate candidates, thus increasing the chance to discover an isolated global minimum. We also propose two multiresolution algorithms based on this technique to help speedup the computation and achieve performance comparable to other fast search techniques.

1.6 Laplacian Model

As mentioned in Section 1.3, the optimization of the VCAs has to take into account the statistics of the input. There are basically two approaches to acquire those statistics, namely, parametric and non-parametric. In the non-parametric case, the probability of an event of interest is obtained directly from the data. For parametric case, a model of the image is assumed and characterized by a set of parameters. The statistics thus can be obtained from the parameterized model. In Chapter 2 and 3, we will use both approaches to evaluate the performance of the proposed algorithms.

In this section, we will address the image modelling assumption used throughout this thesis. Similar to [22], we model a DCT coefficient in a 2-D block as an independent random variable of Laplacian source, i.e., the p.d.f. of $\mathbf{X}(u,v)$ can be written as

$$f_{\mathbf{X}(u,v)}(x) = \frac{\lambda_{(u,v)}}{2} e^{\lambda_{(u,v)}|x|} \tag{1.3}$$

where $\lambda_{(u,v)}$ is the Laplacian parameter of $\mathbf{X}(u,v)$, the DCT coefficient in position (u,v).

In VCA, the complexity savings mostly come from DCT coefficients quantized to zero. Therefore, given that uniform quantization, with stepsize $2QP$ (QP is a quantization parameter) and deadzone in the region $(-2QP, 2QP)$, is used for all DCT coefficients, the probability of $\mathbf{X}(u,v)$ being quantized to zero can be

written as

$$\begin{aligned} p_z(u,v) &= \Pr\{\lfloor |\mathbf{X}(u,v)|/2QP \rfloor = 0\} \\ &= \Pr\{|\mathbf{X}(u,v)| < 2QP\} \\ &= 2(1 - e^{\lambda_{(u,v)} 2QP}) \end{aligned} \tag{1.4}$$

Furthermore, in the case of residue frames, the model parameter $\lambda_{(u,v)}$ can be obtained directly in the spatial domain. From [64], it has been observed that the correlation between pixels in residue frames can be approximated to be separable horizontally and vertically. Thus the correlation can be expressed as $r(m,n) = \sigma^2 \rho^{|m|} \rho^{|n|}$ where m and n are horizontal and vertical displacement, ρ is the correlation coefficient, and σ^2 is the pixel variance. From our observation on five H.263 test sequences ("Miss America", "Suzie", "Mother&Daughter", "Foreman" and "Salesman"), the average correlation coefficient ranges from 0.92 to 0.97. Therefore, in our simulation we use the value 0.9. Let the correlation matrix be denoted by $\mathbf{R}$ and written as

$$\mathbf{R} = \begin{bmatrix} 1 & \rho & \rho^2 & & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & & \rho^{N-3} \\ & \vdots & & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & & \cdots & 1 \end{bmatrix}$$

Therefore, from [94], the variance of the DCT coefficients can be derived as

$$[\sigma^2_{\mathbf{X}(u,v)}] = \sigma^2 [\mathbf{D_N R D_N^t}]_{(u,u)} [\mathbf{D_N R D_N^t}]_{(v,v)} = \sigma^2 [\mathbf{\Gamma_N^2}(\mathbf{u},\mathbf{v})] \tag{1.5}$$

where $\mathbf{D_N}$ is again the DCT matrix of size N, and the scaling factor $\mathbf{\Gamma_N}^2(u,v)$ is a function of N and (u,v). For example, for DCT of size 8 and $\rho = 0.9$, $\mathbf{\Gamma_8}^2(u,v)$

is

$$[\mathbf{\Gamma_8}^2(u,v)] = \begin{bmatrix} 38.2606 & 6.2219 & 2.1408 & & 0.3383 \\ 6.2219 & 1.0118 & 0.3481 & \cdots & 0.0550 \\ 2.1408 & 0.3481 & 0.1198 & & 0.0189 \\ & \vdots & & \ddots & \vdots \\ 0.3383 & 0.0550 & 0.0189 & \cdots & 0.0030 \end{bmatrix} \tag{1.6}$$

where $[\mathbf{\Gamma_8}^2(u,v)]$ represents a square matrix with elements $\mathbf{\Gamma_8}^2$. Therefore, we can find the probability of a zero quantized DCT coefficient from the variance in pixel domain as

$$p_z(u,v) = 2(1 - e^{-\frac{\sqrt{2}QP}{\mathbf{\Gamma_N}(u,v)\sigma}}) \tag{1.7}$$

Rate and Distortion

For the sake of completeness, we finally show the rate and distortion of this Laplacian DCT source with uniform quantization presented earlier in this section. The probability of DCT coefficients in bin $(2QPi, 2QP(i+1))$ can be expressed as

$$p_i = \int_{2QPi}^{2QP(i+1)} f_{X(u,v)}(x)dx$$

The rate can be approximated by the entropy

$$\begin{aligned} R(u,v) &= -\sum_i p_i \log p_i \\ &= e^{-2\lambda QP}(1 + \frac{2\lambda QP\epsilon}{1 - e^{-2\lambda QP}}) - \log(1 - e^{-2\lambda QP}), \end{aligned} \tag{1.8}$$

and therefore, the total block rate is

$$R_{blk} = \sum_{(u,v)} R(u,v), \tag{1.9}$$

where $\epsilon = \log_2 e$ and $\lambda = \frac{\sqrt{(2)}}{\sigma\Gamma_N(u,v)}$.

For midpoint reconstruction in each quantization bin, the distortion can also

be derived as

$$
\begin{aligned}
D(u,v) &= -\sum_{i\neq 0}\int_{2QPi}^{2QP(i+1)}(x-QP(2i+1))^2 f_{X(u,v)}(x)dx + \int_{-2QP}^{2QP} x^2 f_{X(u,v)}(x)dx \\
&= \sigma^2\Gamma_N^2(u,v) - \frac{2QPe^{-2\lambda QP}(3-e^{-2\lambda QP})}{\lambda(1-e^{-2\lambda QP})} - 3e^{-2\lambda QP}QP^2 \qquad (1.10) \\
D_{blk} &= \sum_{(u,v)} D(u,v) \qquad (1.11)
\end{aligned}
$$

where D_{blk} is total block MSE.

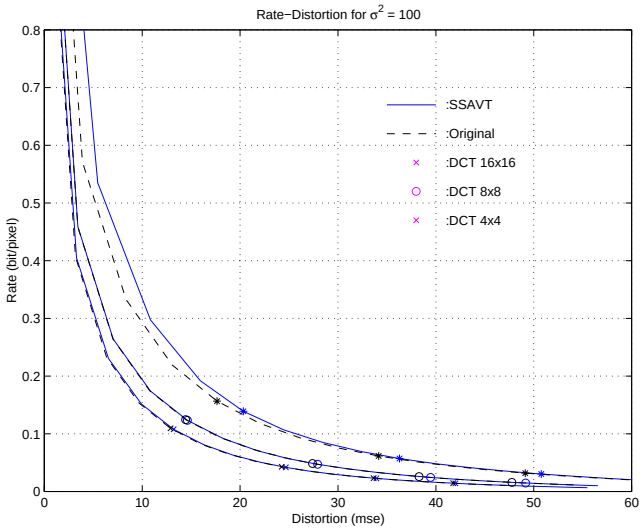


Figure 1.8: Rate-Distortion using standard-defined DCT ('dashed') and R-D for SSAVT ('solid') for $\sigma^2 = 100$ with varying QP at 3 block sizes, i.e., 4x4, 8x8 and 16x16. The rate and distortion is normalized to per pixel basis.

The R-D curves obtained from (1.9) and (1.11) are shown in Fig. 1.8 for different block sizes. It can be seen that larger block sizes give better coding efficiency.

Chapter 2

Inverse Discrete Cosine Transform

In this chapter, we propose fast VCA algorithms for IDCT. We treat IDCT before DCT because it is conceptually easier to explain the problem in that case and the VCA framework can be more easily introduced. For example, if the information about the position of the zero DCT coefficients is available, some operations involved with those zero coefficients can be pruned. The key idea of VCA IDCT is based on this concept such that knowledge of the position of the zero coefficients is used to prune the DCT algorithm. For typical images, this classification intuitively should give significant complexity reduction since high frequency DCT components tends to be quantized to zero (see Fig. 1.4 and 1.5). The speedup can be even more significant for lower quality images or in the P-frames of video sequences.

We propose 2 novel classification schemes, namely, sequential and tree-structured classification (TSC). We also discuss the design goal for optimal classification for both schemes. The best test structure is selected to achieve minimal average complexity for a given input statistics. For the above two classifications, we will consider only a 1-D IDCT for simplicity. However, the algorithms can be easily extended to the separable 2-D transform. Next we consider a heuristic 2-D dyadic classification similar to [60] as an alternative for faster but coarser classification. This method can be combined with other 1-D VCA IDCTs mentioned

above. Some experimental results on different image model parameters and on real image and video data are also shown.

The distortion/decoding time tradeoff problem is posed. The goal is to minimize the distortion with the constraint of a given decoding time budget, under the assumption that the IDCT at the decoder is VCA. The problem is solved using Lagrange multiplier techniques that provide the optimal quantization assignment for each block of image, i.e., the assignment that minimizes distortion under the given decoding time budget. In addition, a generalized rate-complexity-distortion (R-C-D) optimization framework using quadtree structure is proposed. Beside the quantization parameter, we allow the encoder to select the quadtree structure as an additional degree of freedom in the R-C-D tradeoffs.

2.1 Formalization of IDCT VCA

Our goal is to define a VCA version of the IDCT that can achieve minimal average complexity. We first select a baseline algorithm (which gives an exact IDCT for any input). We have chosen the algorithm used in [58] (similar to Fig. 1.3(b)) as our baseline algorithm because it requires the minimum number of operations. This algorithm uses fixed point arithmetic while still keeping enough accuracy to be compliant with the JPEG standard. The number of operations when applying this 1-D IDCT to an 8x8 block of a 2-D image along the row and column sequentially is as follows: 176 multiplications (11 multiplications per 1-D IDCT after factoring out division by $2\sqrt{2}$ to the last stage), 536 additions and 240 bit-wise shifting operations(which replace the divisions by 8 and are used to maintain accuracy).

This baseline algorithm is then pruned according to the classification information available about the inputs. We refer to pruned algorithms as "reduced IDCT" (RIDCT) algorithms. Each RIDCT is designed for each class such that it uses the minimal number of operations for the given class. The combined classification and RIDCTs is, thus, a VCA.

The accuracy of information obtained from classification is also a factor to determine the minimal operation numbers, e.g., a coarse classification will not

provide exact information about whether each coefficient is zero, and thus will limit the number of operations that can be pruned. In order to obtain precise information, finer classifications are generally required. Therefore, the goal of our formalization is to explore the tradeoffs between classification accuracy and classification cost overhead. This is achieved in the average sense by taking the cost of classification and the statistics of frequency of occurrence of each class into account for a given image or set of images.

2.1.1 Input Classification

We define a class of inputs to be a set of input vectors having zero coefficients at certain positions. Let X_j be the j-th DCT coefficient in the 1-D input DCT vector and $\bar{\mathbf{z}}_i$ be a vector representing the i-th class of input, elements in the $\bar{\mathbf{z}}_i$ vector represent zero positions in the input vector, i.e.,

$$\bar{\mathbf{z}}_i = Z[i_j]_{j=0}^{7}, \qquad \text{where} \quad i_j = \begin{cases} 0 & \text{if } X_j = 0 \\ 1 & \text{if } X_j \text{ is unknown} \end{cases} \tag{2.1}$$

It is clear that a larger number of zeros results in potentially faster IDCT implementations, since operations such as multiplication by zero and sum with a zero value need not be performed. For example, with a size-8 input vector, an all zero input (i.e., class $Z[00000000]$) would require no operations while a class $Z[11111111]$ input would require the maximum number of operations. However, for each input class one can find many RIDCT algorithms which give the exact IDCT output, e.g., one can apply RIDCT of $Z[11111111]$ to $Z[00000000]$ or any other classes and still get the correct IDCT results. Let $\mathcal{A}_i = \{A_j\}_{j=1}^{J(i)}$ be the set of RIDCTs that provide exact IDCT computation for class $\bar{z}_i$. There are $J(i)$ RIDCT algorithms for class $\bar{z}_i$. The one with minimal complexity is defined as

$$A_i^* = \arg \min_{A_j \in \mathcal{A}_i} c(A_j)$$

where $c(A_j)$ is complexity of A_j. Note here that the complexity is measured in a unit of interest, which is not restricted to be an arithmetic complexity measure. The most appropriate unit of complexity which can be used directly to judge

the performance of a system is the elapsed time or the CPU clock cycle spent in the process. For example, the classification, even though it may not involve any arithmetic operations, does require other types of instructions, and thus has a certain complexity cost. (In this work, we will use "cost" or "complexity" instead of "complexity cost".)

In order to further motivate the concept of classification, let us denote a primitive class in which all components are known as $\bar{\mathbf{s}}_i$, i.e.,

$$\bar{\mathbf{s}}_i = S[i_j]_{j=0}^{7}, \qquad \text{where} \quad i_j = \begin{cases} 0 & \text{if} \quad X_j = 0 \\ 1 & \text{if} \quad X_j \neq 0 \end{cases} \tag{2.2}$$

$\bar{\mathbf{s}}_i$ is the input bitmap of the i-th primitive class. Note that the main difference between classes $\bar{\mathbf{s}}_i$ and $\bar{\mathbf{z}}_i$ is that the former assumes that some coefficients are non-zero, while in the latter those coefficients were not known to be zero or non-zero. Note also that the class $\bar{\mathbf{z}}_i$ consists of primitive classes that share the same information about the zero of certain DCT coefficients (j, when $i_j = 0$). The cardinality of the set depends on the number of untested coefficients (j, when $i_j = 1$). Note here that beside zero-nonzero based classification, one can also classify the input differently, e.g., an input with coefficients value 1 or -1 which requires no operations for multiplication, etc. However, for simplicity we consider only zero-nonzero based classification.

The input bitmap can be obtained as a by-product of entropy decoding. Specifically, when zero-run length coding is employed (in JPEG and MPEG), this bitmap information is computationally cheap to obtain. However, the mapping between an input bitmap and a corresponding reduced algorithm (classification) is not so straightforward. For better visualization, we assume that the classification and the reduced algorithms are two separate operations. For example, for a DCT vector of size 8, there are 256 possible primitive classes. Brute force mapping of the bitmap to a corresponding reduced algorithm would require a large memory storage for 256 possible RIDCTs. In the case where we have a smaller number of RIDCTs due to memory space limitations, the problem of RIDCTs selection needs to be studied. In our implementation we get rid of the storage dependency by considering a single algorithm which consists of operations conditionally pruned

by testing their corresponding inputs. Therefore, the memory requirement is no longer an issue, but the designs of classification and RIDCTs are coupled.

At this point, we can generalize the definition of "class" in (2.1) to be any set of primitive classes. Therefore, starting with the space of all possible primitive classes, we can partition this space into groups of classes. Let us denote one such partition as $g \in \mathcal{G}$ where $\mathcal{G}$ is the set of all possible partitions of the set of primitive classes. Let N_g be the number of resulting classes in g, and $g(i)$, $i = 0, ..., N_g - 1$, represent the classes of input which are members of g. Each class $g(i)$ has an associated minimal complexity RIDCT A_i^*. In order to formulate the minimal complexity VCA problem, the classification cost has to be taken into account. Let S_g be a classification structure for g, $S_g \in \mathcal{S}_g$ which is the set of all possible testing structures resulting in partition g, and $S_g == \{S_g(0), ..., S_g(N_g-1)\}$ where $S_g(j)$ is the cost of classifying class $g(j)$. Therefore, our goal can be formalized as

Formulation 1 (Minimal Complexity VCA) *The goal is to find a partition g^* and a corresponding classification structure, S_g^* which gives the best tradeoff between complexity reduction by RIDCT and the complexity of the classification itself. In other words, we want to find g, S_g such that the average complexity, T, is minimized*

$$T^* = \min_{g \in \mathcal{G}} \min_{S_g \in \mathcal{S}_g} \sum_{i=0}^{N_g-1} (c(S_{g(i)}) + c(A^*_{g(i)})) \cdot P_{g(i)}, \tag{2.3}$$

where $P_{g(i)}$ be the probability of inputs in class $g(i)$ occuring[1]*, $c(S_{g(i)})$ be the cost of operations necessary to perform the test S_j.*

This involves two minimizations over (i) possible partitions, $\mathcal{G}$, and (ii) test structure, $\mathcal{S}_g$. It can also be viewed as a tradeoff between the fineness of classification and the classification cost. The more precise the class is (smaller group), the more compact the corresponding RIDCTs, i.e., $\sum_{j \in g(i)} P_j \cdot c(A_j^*) \leq P_{g(i)} \cdot c(A^*_{g(i)})$ for $P_{g(i)} = \sum_{j \in g(i)} P_j$ based on the assumption that $c(A_j^*) \leq c(A^*_{g(i)}), \forall j \in g(i)$. However the more classification cost, i.e., $\sum_{j \in g(i)} P_j \cdot c(S_j) \geq P_{g(i)} \cdot c(S_{g(i)})$. Without any further restrictions, the solution of this optimization problem is very

[1] This probability can be measured from typical training image data or can be obtained from a model of typical images.

difficult to find. In the next section, we impose constraints on both partition and the test structure such that the search space is limited to one of manageable size. We restrict ourselves to tests that seem intuitively reasonable given the baseline algorithm.

2.2 Proposed VCA for IDCT

In this section, we discuss three classification methods, namely (i) sequential, (ii) tree-structured and (iii) 2-D dyadic classification. For the first two methods we also address the optimization of the classification structure to obtain the minimal average complexity. For dyadic classification, we use a heuristic test structure based on experimental observations of typical image and video data. All of these classification techniques employ zero masks in testing for a certain class. In zero mask testing (ZMT), the bitmap of the input vector is tested via a bitwise "AND" operation with a mask, denoted by $\bar{\mathbf{m}}_i$ for the i-th mask,

$$\bar{\mathbf{m}}_i = M[i_j]_{j=0}^{7} \qquad \text{where} \qquad i_j \in \{0, 1\}$$

which represents the class being tested. Each bit of the mask represents a DCT coefficient of interest so that if the bit is set to one we test for a zero coefficient, but we do not care if the bit is set to zero. It can be seen that the mask can be related to the class $\bar{\mathbf{z}}_i$ for a certain i presented earlier: if the result of the bitwise 'AND' is zero, the input belongs to class $\bar{\mathbf{z}}_i$, otherwise, the input is not a member of class $\bar{\mathbf{z}}_i$.

2.2.1 Sequential Classification

Sequential classification is a scheme such that the input is zero-mask tested for one class at a time. There is no refinement for classes already tested. Therefore, the sequential classification can be expressed by a sequence of zero masks, which correspond to the order in which each class is tested. Let $M = \{\bar{\mathbf{m}}_0, ..., \bar{\mathbf{m}}_{N_M-1}\}$ be a sequence of zero masks representing a sequential classification, where N_M is the number of zero masks, and there are $N_M + 1$ resulting classes. From the

previous subsection, we know that $\bar{\mathbf{z}}_i$ is a class tested by $\bar{\mathbf{m}}_i$. However, it is possible that some primitive classes in $\bar{\mathbf{z}}_i$ are also members of $\bar{\mathbf{z}}_{i-1}$ which will be tested first. Therefore, the actual set of primitive classes tested at step i, denoted by $\bar{\mathbf{d}}_i$, is $\bar{\mathbf{z}}_i - \cup_{j=0}^{i-1}\bar{\mathbf{z}}_j$. The last class (default class) after the last test is $\bar{\mathbf{d}}_{N_M+1} = C_\forall - \cup_{j=0}^{N_M-1}\bar{\mathbf{z}}_j$, where $C_\forall = \cup_{j=0}^{255}\bar{\mathbf{s}}_j$ is the set of all primitive classes.

Without loss of generality, let us reuse the notation for the minimal complexity reduced algorithm, A_i^*, for class $\bar{\mathbf{d}}_i$. Therefore, we can write the expression for the average complexity of the sequential classification IDCT as

$$T_{seq} = \sum_{i=0}^{N_M-1} (c(ZMT)\cdot(i+1)+c(A_i^*))\cdot P_{d(i)} + (c(ZMT)\cdot N_M + c(A_{N_M+1}^*))\cdot P_{d(N_M+1)} \tag{2.4}$$

where $c(ZMT)$ is the cost of zero masking test and $P_{d(i)}$ is the probability of class $\bar{\mathbf{d}}_i$. Since the cost of ZMT is the same for every mask, the testing cost for the i-th mask is (i+1) times the ZMT cost since there must be i+1 ZMT prior to the i-th mask. The term outside the summation in (2.4) represents the cost for the default class. Therefore, we can formalize the complexity minimization *based on sequential test* as

Formulation 2 (Optimal Sequential Classification VCA) *The goal is to find the optimal zero mask sequence M^* such that the total average complexity is minimal. In other words, we want to minimize*

$$T_{seq}^* = \min_{M\in\mathcal{M}} T_{seq} \tag{2.5}$$

where $\mathcal{M}$ is the set of all possible zero mask sequences M.

2.2.2 Greedy Optimization

The problem formulation is fairly straightforward but in order to obtain the optimal solution, exhaustive search is unavoidable. One can view the exhaustive search as a tree search in which the tree grows to all possible directions. The time required for exhaustive search can be reduced by using the knowledge of the redundancy between zero masks, i.e., the fact that, as mentioned earlier, set $\bar{\mathbf{z}}_i$ and $\bar{\mathbf{z}}_j$ for given i and j may have a non-empty intersection. Thus, the path to a node

that corresponds to a subset of previous nodes can be omitted. Yet, it is still time consuming to find the optimal solution and it may not be practical to look for it when the size of the input vector is larger than 8. Finding suboptimal solutions is thus a more attractive alternative. Algorithms such as the M-algorithm [95], in which only M best paths are kept at each iteration, can be used. In this work, we use an even simpler and faster approach than the M-algorithm.

We use a greedy approach, in which the class that gives maximal complexity savings is tested first, and then the classes with less computational saving are tested later, until no further savings can be achieved from the classification, i.e., the savings from using a RIDCT is outweighed by the cost of testing. When that point is reached the default algorithm is applied to the remaining (untested) classes. Prior to performing a greedy optimization, the statistics of each class $\bar{\mathbf{s}}_i$, for all i are assumed to be known. Let $P_{s(i)}$ denote the probability of a primitive class $\bar{\mathbf{s}}_i$. Let us define the complexity savings of reduced algorithm A_i^* of class $\bar{\mathbf{z}}_i$ by $\tilde{c}_i = c(A_\forall) - c(A_i^*)$, where $A_\forall$ represents the baseline algorithm. The Greedy approach can be described as follows.

Algorithm 1 (Greedy Optimization)

Step 1: Assume known probabilities of all primitive classes, $P_{s(i)}$, $\forall\, i$. Initialize the list of sequential tests M as an empty set. Set $N_M = 0$.

Step 2: Find class i such that the incremental complexity savings is maximal, i.e., $i^* = \arg\max_{\bar{\mathbf{m}}_i \notin N_M} \Delta\tilde{c}_i$ where $\Delta\tilde{c}_i = \tilde{c}_i \cdot P_{d(i)} - c(ZMT) \cdot P_{untested}$ and $P_{untested} = \Pr\{C_\forall - \cup_{i=0}^{N_M-1} d_i\}$.

Step 3: If $\Delta\tilde{c}_{i^*} > 0$ put $\bar{\mathbf{m}}_i$ as the last element in M and increment N_M by 1. Otherwise, stop.

Step 4: If $N_M = 256$, stop. Otherwise go to Step 2.

The calculation of $P_{d(j)}$ in Step 4 has to take into account the redundancy of previous zero mask tests by excluding the probability of the primitive classes that have already been tested.

2.2.3 Tree-structured classification (TSC)

In this subsection, we propose an alternative to sequential classification. Unlike sequential classification where the tested classes are not refined or tested again, the tree-structured classification (TSC) allows hierarchical testing in which tested classes can be further classified for more accurate positions of zeros, which also implies better RIDCTs. The key idea of TSC is to exploit the structure of the baseline algorithm in classification. Consider the baseline full version of the IDCT algorithm we have chosen, which is shown in Fig. 1.3(b). In this IDCT algorithm there are three stages of operations, each of which involves 2, 4 and 8 input coefficients, respectively. This tree-like structure provides a natural way of hierarchically classifying the data, going from coarse to fine classification, i.e., ZMTs that test a larger group of coefficients and then smaller and smaller groups.

In Fig. 1.3(b), let us consider the issue of grouping first. Needless to say, when all 8 coefficients are zero, all branches up to stage 3 can be pruned. Given that 4 out of 8 coefficients are zero, the two scenarios that give the largest reductions in complexity will correspond to the cases when (X_7, X_1, X_3, X_5) or (X_0, X_4, X_6, X_2) are all zero, since branches of zero coefficients in stage 1 and stage 2 can be pruned. For the case when 2 coefficients are zero, (X_2, X_6), (X_1, X_7), (X_3, X_5) and (X_0, X_4) give the four minimal complexity RIDCTs since the corresponding operations of these 4 pairs in stage 1 can be pruned. When only one coefficient of those pairs is zero, stage 1 operation can be partially pruned. From these, we can find, for the case when any number of coefficients is zero, the grouping with maximal complexity reduction which is a combination of the above 4, 2 and 1 coefficients.

As far as the test structure is concerned, in order to achieve the above grouping with maximal efficiency regardless of the statistics of the input (assuming a uniform distribution of the input classes), groups with lower complexity RIDCT should be tested before groups with larger complexity RIDCT. In this case, the all-zero input class must be tested first. Then the two 4-zero classes, the four 2-zero classes and finally the eight 1-zero classes are tested in succession. This hierarchical testing results in at most 15 tests before a certain class can be identified. An example of pseudo code for testing 2 coefficients at the second stage of

the upper 4 branches of Figure 1.3(b) is shown in Figure 2.1.

```
    if(X2==0 and X6==0)
       z0 = z3 = y0
       z1 = z2 = y1
/********* save computation for y2 and y3 **********/
   else if(X0==0 and X4==0)
    z0 = y3 , z1 = y2
    z2 = −y2 , z3 = −y3
/********* save computation for y0 and y1 **********/
   else        z0 = y0 + y3
       z1 = y1 + y2
       z2 = y1 − y2
       z3 = y0 − y3
/********* performing full version for this part **********/
```

Figure 2.1: TSC pseudo code for testing upper 4 branches of Fig. 1.3 (b) in stage 2.

This TSC is illustrated in Figs. 2.2 and 2.3. For simplicity in explanation, we denote $C_{i_0 i_1 i_2 i_3 i_4 i_5 i_6 i_7}$ as a class of input obtained from TSC. The classification information i_j indicates the following.

$$i_j = \begin{cases} E_k & k \in \{1, 2, ...\} \text{ means that at least one of coefficients indexed with a given } E_k \text{ is nonzero} \\ 0, 1 & \text{means that } x_j \text{ is "known" to be zero or nonzero} \\ U & \text{means } x_j \text{ can be either zero or nonzero (unknown)} \end{cases}$$

As an example, $C_{E_1 U E_2 0 E_1 0 E_2 U}$ means that at least one of $\{x_0, x_4\}$ and one of $\{x_2, x_6\}$ must be nonzero, x_1 and x_7 are unknown, and x_3 and x_5 are *both* known to be zeros. Thus, in order to perform IDCT correctly an RIDCT must assume that all coefficients except x_3 and x_5 are zero. Note that the class in the example above can be represented as $Z^c[01110111] \cap Z^c[11011101] \cap Z[11101011]$, where $Z^c[.]$ is a complement of set $Z[.]$. The newly introduced notation is obviously more concise.

Fig. 2.2 illustrates the tree-structured classification of size 8 DCT coefficients. Each node represents a class with some knowledge about the positions of zero coefficients. Descendents of each node are the results of finer classifications, i.e., they contain more information about zero coefficients than their parent nodes. For

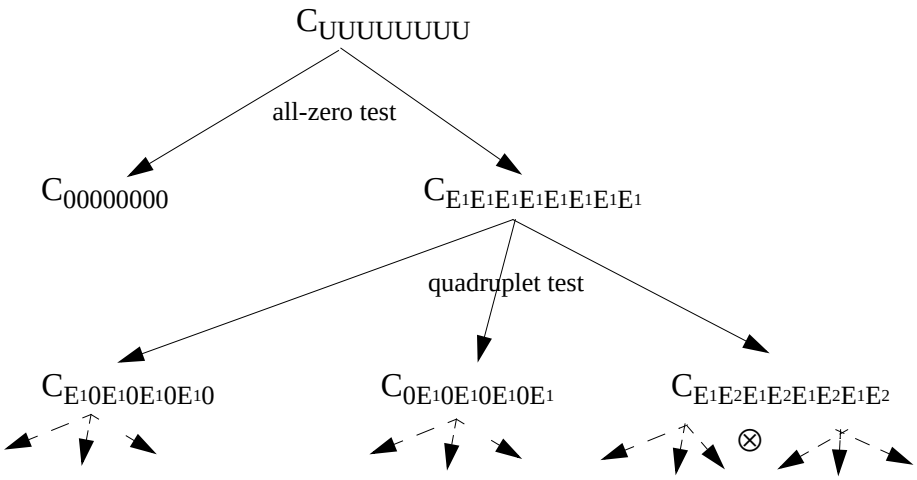


Figure 2.2: TSC diagram showing zero information of classes after all-zero test and 4-zero tests.

simplicity, in the figure the tree is grown to the level of 4-zero classes only. Finer classifications are shown in Fig. 2.3 where 2-zero and 1-zero tests are performed. It can be verified that a total of 256 primitive classes can be obtained with this TSC if the tree is constructed from all-zero level down to 1-zero test level, i.e., combining Fig. 2.2 and 2.3.

Every nodes of the tree in Fig. 2.2 and 2.3 has an associated RIDCTs. The RIDCTs for those classes are based on pruning only operations that involve coefficients that are known to be zero. Thus, we cannot prune operations involving coefficient labeled with U or E_k unless we are willing to have a non-exact IDCT.

2.2.4 Optimal Tree-structured Classification (OTSC)

Note that in Section 2.2.3, we outlined the classes that are induced by testing in a hierarchical manner, starting with tests on 4 coefficients and continuing with two and then one coefficient. At each node, in order to classify to 3 classes, two logical operations must be performed. For example, class $C_{E_1E_1E_1E_1}$ can be classified to $C_{0E_10E_1}$ and $C_{E_1UE_1U}$ using AND logic between the input bitmap and the mask $M[1010]$: if the result is zero, the input belongs to C_{0U0U}, otherwise it belongs to $C_{E_1UE_1U}$. Then $C_{E_1UE_1U}$ is further classified to $C_{E_10E_10}$ and $C_{E_1E_2E_1E_2}$ by performing an AND of the input bitmap with $M[0101]$: if the result is zero, the

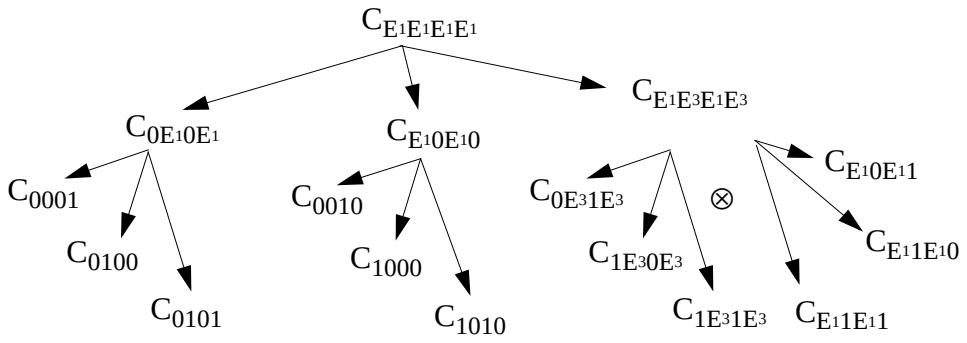


Figure 2.3: TSC diagram showing zero information of classes after 2-zero test and 1-zero tests. The most right-handed class after 2-zero test is further classified into 9 classes which are tensor products of descendent classes shown, i.e., $A \otimes B = \{(A_i \cap B_j)\}_{\forall\, i,j}$ where $A = \{A_i\}, B = \{B_i\}$. Therefore, the number of leaves is 3+3+(3x3) = 15.

input belongs to $C_{E_10E_10}$, otherwise it belongs to $C_{E_1E_2E_1E_2}$. However, the order in which $M[0101]$ and $M[1010]$ are applied can be switched, and the resulting 3 classes can still be obtained. However, the complexities of both orderings are different when the frequency of occurence of the resulting classes is taken into account. For example, in the above example, if class $C_{E_10E_10}$ occurs more often than $C_{0E_10E_1}$, then regardless of RIDCTs complexity[2] it is better to apply the mask $M[0101]$ before $M[1010]$.

Therefore, for optimal TSC tests, ordering has to be carefully chosen. Furthermore, we allow more freedom to perform only one logic operation at a node to classify only 2 classes, e.g., in the above example, we can choose to have only the second test performed on $C_{E_1E_1E_1E_1}$ to get $C_{E_10E_10}$ and $C_{UE_1UE_1}$. Thus, the goal of our optimization procedure is to find the best classification, that is, to determine (i) the best order in which to apply the tests and (ii) which tests it is worth performing (in the sense of reducing the average complexity).

In Figs. 2.4, we show the possible choices for testing 4 coefficients. We denote the two logic comparisons by A and B. In the figure there are five possible choices which are A-B, B-A, A only, B only and no-test. Each choice results in different classifications except A-B and B-A which have the same final result. Given the

[2]In our work, the RIDCTs complexity has to be taken into account in order to obtain minimal total complexity.

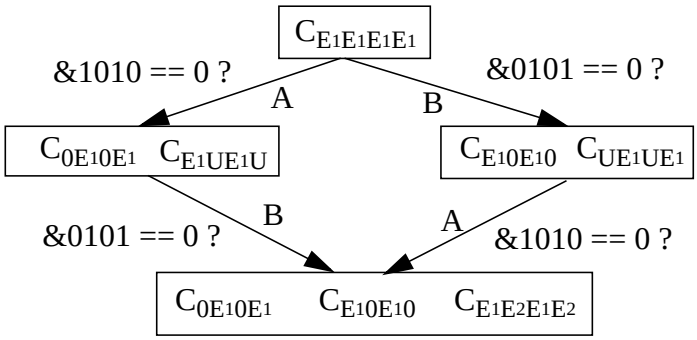


Figure 2.4: Diagram showing possible choices for classification at level 2 where A, B represent testing with $M[1010]$ and $M[0101]$, respectively.

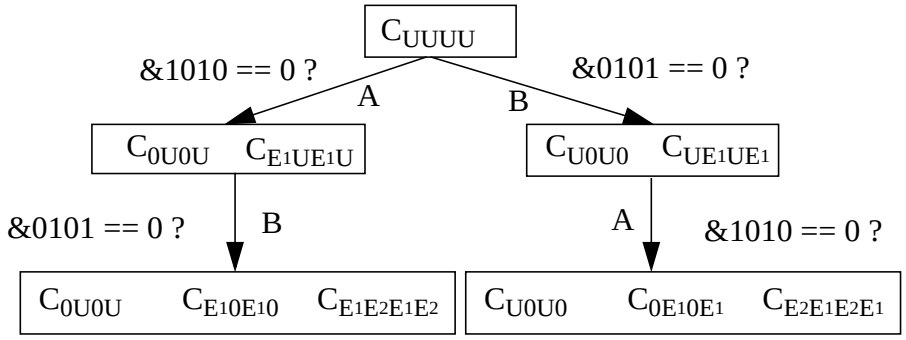


Figure 2.5: Another diagram when the result from level 1 is different.

statistics of the input, we can then compare the complexity including the test to find the best choice on the average sense. The complexities of the five choices as shown in Fig. 2.4 are

$$
\begin{aligned}
T_{\texttt{A-B}} &= c(S_A)P_{E_1E_1E_1E_1} + c(S_B)P_{E_1UE_1U} + \\
&\quad c(A^*_{0101})P_{0E_10E_1} + c(A^*_{1010})P_{E_10E_10} + c(A^*_{1111})P_{E_1E_2E_1E_2} \\
T_{\texttt{B-A}} &= c(S_B)P_{E_1E_1E_1E_1} + c(S_A)P_{UE_1UE_1} + \\
&\quad c(A^*_{0101})P_{0E_10E_1} + c(A^*_{1010})P_{E_10E_10} + c(A^*_{1111})P_{E_1E_2E_1E_2} \\
T_{\texttt{A-only}} &= c(S_A)P_{E_1E_1E_1E_1} + \\
&\quad c(A^*_{0101})P_{0E_10E_1} + c(A^*_{1111})P_{E_1UE_1U} \\
T_{\texttt{B-only}} &= c(S_B)P_{E_1E_1E_1E_1} + \\
&\quad c(A^*_{1010})P_{E_10E_10} + c(A^*_{1111})P_{UE_1UE_1}
\end{aligned}
$$

$$T_{\texttt{no-test}} \quad = \quad c(A^*_{1111})P_{E_1E_1E_1E_1}$$

We can perform the comparison of these 5 choices at all level of testing i.e., all-zero, 4-zero, 2-zero and 1-zero test. In the case, where A-only, B-only or no-test are chosen in the previous level of zero testing, the current level will start with a class which no information has been obtained, see Fig.2.5. From the TSC tree in Fig. 2.2 and 2.3 we compare all possible combinations of 5 choices in every node to find the test structure with minimal complexity.

For consistency with previous formalization, let H be redefined as a set of choices at each node in Fig. 2.2 and 2.3.

$$H = \{h^0, h^1, \{h^2_i, h^3_{4i}, h^3_{4i+1}, h^3_{4i+2}, h^3_{4i+3}\}^3_{i=0}\}$$

where $h^l_i \in \{\texttt{A}-\texttt{B}, \texttt{B}-\texttt{A}, \texttt{A}-\texttt{only}, \texttt{B}-\texttt{only}, \texttt{no}-\texttt{test}\}$ is the choice selected at level l and node i. There are 4 levels corresponding to all-zero, 4-zero, 2-zero and 1-zero test. If the previous level performs classification with less than 3 classes, not all of next level choices are available. For example, if $h^1 = \texttt{A}-\texttt{only}$, the resulting classification is $\{C_{E_10E_10E_10E_10}, C_{UE_2UE_2UE_2UE_2}\}$ then h^2_1 has no value because the class $C_{0E_10E_10E_10E_1}$ is not classified but included in $C_{UE_2UE_2UE_2UE_2}$. Also $h^3_4, h^3_5, h^3_6, h^3_7$ have no values either. Thus our optimization goal can be formalized as

Formulation 3 (Optimal Tree-Structured Classification VCA) *The goal is to find H^* from all possible combinations $\mathcal{H}$ such that the overall complexity is minimal i.e., we minimize the average complexity*

$$T^*_{\texttt{TSC(binary)}}(P) = \min_{H\in\mathcal{H}} T_{\texttt{TSC(binary)}}(H, P) \tag{2.6}$$

for a given statistics of input classes.

Let H^* represent an *optimized TSC* (OTSC) algorithm which can be found by searching overall possible test structure space, $\mathcal{H}$, satisfying the above binary TSC framework.

2.2.5 Computation of 2-D IDCT

As mentioned earlier, there are several approaches to classify a 2-D DCT input block using the previously proposed algorithms. For example, we can directly test the coefficients in a 2-D block as in [60]. We can also apply OTSC to each row of input in the row-wise 1-D IDCT and then apply the OTSC for each column of the row-wise IDCT output in the column-wise 1-D IDCT. For simplicity, we choose to perform the latter (separable 2-D IDCT). Note that, we can have different OTSCs for different row and column positions but the overhead of caching different functions in the software implementation outweighs the complexity reduction of having more OTSCs for different row/column statistics. Thus we use the same algorithm for all rows/columns.

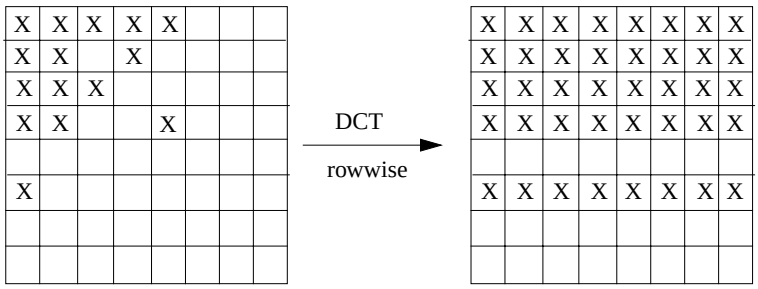


Figure 2.6: Content of bitmap after first (row-wise) 1-D IDCT where 'x' represents nonzero coefficient.

Except for the 2-D dyadic classification in Section 2.2.6, the TSC and sequential classification schemes require re-classification for the intermediate result between the 1-D separable IDCTs. After the rowwise 1-D IDCT, the input to the columnwise 1-D IDCT has to be classified again. The bitmap for the second IDCT can be approximated very accurately from the first bitmap as shown in Figure 2.6. The approximation relies on the fact that if at least one frequency component is nonzero it is more likely that all the outputs from the IDCT (i.e. the result of applying the IDCT to the row or column) will be all different from zero. There are very unlikely cases where magnitude cancellation occurs at some output points and the output is zero even though there were non-zero inputs but we ignore these cases. The complexity savings from this bitmap implementation is about 5% over direct coefficient testing.

2.2.6 2-D Dyadic Classification

In the above proposed classification schemes, i.e., Sequential Classification and TSC, 1-D IDCT is the target the classifications are applied on. They both can be extended to 2-D classification but the complexity is also increased and the optimization process may become impractical. Therefore, we propose an alternative classification in which a block of 2-D DCT coefficients is classified instead of classifying each row/column separately. We propose 2-D dyadic classification in which an NxN DCT block is classified into all-zero, DC-only, low-2x2, low-4x4, low-8x8, ... , $\frac{N}{2}$x$\frac{N}{2}$, and full-NxN classes. For each of these classes, a corresponding 1-D RIDCT is applied along the nonzero rows and then all columns. The testing order follows the above list. The reason behind this classification is from the fact that in typical image high frequency components are more likely to be quantized to zero. Figure 2.7 shows how the classification works.

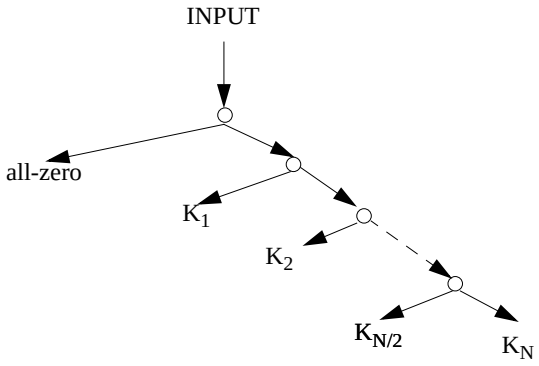


Figure 2.7: Dyadic classification scheme. A DCT input of size NxN is classified into all-zero, DC-only (K1), low-2x2 (K2),..., low-$\frac{N}{2}x\frac{N}{2}$ ($K_{N/2}$), and full-NxN (K_N) classes.

When the baseline algorithm has a recursive structure in the sense that larger size IDCT computation consists of smaller sizes IDCT computations, we can express the complexity of the RIDCTs, accordingly. For example, the Chen,Smith &Fralick [28] or the Vetterli-Ligtenberg [29] (Fig. 1.3) algorithms can be decomposed to a normal half-size IDCT (type II) and a half-size IDCT of type IV [24] and the normal half-size IDCT can be further decomposed and so on. Therefore, it is equivalent to having smaller size IDCTs as the RIDCTs. The complexity

for each class tested by dyadic classification on an NxN DCT block can then be written as

$$c(A_i^*) = (N + \frac{N}{2^{\log_2 N - i + 1}}) K_{N/2^{\log_2 N - i + 1}} \tag{2.7}$$

for class $i = 1, 2, ..., \log_2 N + 1$ where K_N is the complexity of DCT/IDCT of size N and when $i = 0$, $c(A_0^*) = 0$ for all-zero block. The term that multiplies K represents the number of rows and columns on which 1-D reduced IDCT is performed. Here, we use a separable reduced size 1-D IDCT to perform 2-D IDCT. The reduced rowwise transform is performed only for rows with nonzero coefficients. Then the reduced columnwise transform is applied to all columns. In terms of classification structure, it can be seen that, unlike the greedy approach, the order of classification (or threshold testing) is fixed and the closed form complexity expressions of the reduced algorithms are known. Let B_i denote class i input. Thus, the complexity of B_i can be written as

$$T_{dyadic}(B_i) = c(A_i^*) + i \cdot T_{dtest} \tag{2.8}$$

where T_{dtest} represents the testing cost (2-D zero mask or series of 1-D zero mask tests). For simplicity, we assume that T_{dtest} is constant for all classes. From (2.8), the average complexity can be expressed as

$$\bar{\mathbf{T}}_{dyadic} = \sum_{i=1}^{\log_2 N + 1} (T_{dyadic}(B_i) \cdot \Pr(B_i) \tag{2.9}$$

where $\Pr(B_i)$ is the probability of class B_i which is a function of QP, σ^2 and N.

2.3 Results

In our experiments we implement a real software image decoder using the OTSC. For sequential testing, we only implement a greedy optimization based on an image model.

The main problem now is obtaining accurate models for the costs of the operations used in ZMT, and RIDCT, such as addition, multiplication, shift, OR, NOT,

Table 2.1: Weight for different logical operations.

Operation	Weight
Addition	1
Binary Shift	1
Binary OR	1
Multiplicatio	3
Conditional Branching	5

IF, etc. While this may be relatively straightforward in hardware implementations or for assembly language level programming, it is more complicated if we use a high level programming language, since the effect of memory accesses, size of the code and so forth are more difficult to characterize. This problem is very difficult to solve since it highly depends on compiler and machine architecture. To obtain more accurate estimates one could resort to an assembly language implementation or to measuring the average time complexity for each of the possible classification trees directly. This latter approach would clearly be too complex and still would not guarantee an exact cost measure, since many different factors affect the runtime performance.

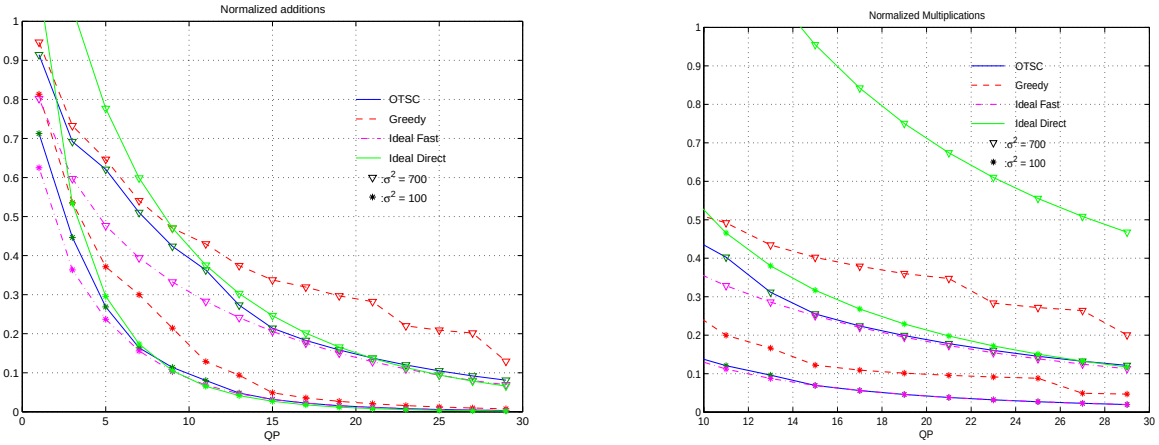
For the sake of practicality, we just use a set of approximated weights for the operations involved. By feeding these approximated parameters into our search algorithms we can get a fairly accurate approximation to the optimal solution. We have used a Sun Sparcstation 5 running Solaris 2.5 with the gcc compiler, and Pentium 450 MHz running WindowNT 4.0 with Visual C++ compiler. We use the assessment in Table 2.1 for each operation [96].

Note that one IF statement consists of comparing to zero and a jump operation. One more thing to be considered here is memory access. Whenever one variable is involved in an operation, there is either a read-in or write-out time. However, since this memory access can be highly optimized using an optimization option by the compiler, we do not take that into account. We emphasize this issue because our costs are only approximated and very machine dependent. Hence, we do not guarantee that the result of the OTSC will be the same for all environments.

2.3.1 Results Based on Image Model

In this section we assume that the image characteristic to be tested follows the model presented in Section 1.6 (assuming INTER-frame coding). Therefore, given that the model parameter is known, the probability of X(u,v) being zero ($p_z(u,v)$) can also be computed. Thus, we can obtain the average complexity of any algorithms. We perform our experiments on different values of the model parameter and the quantization parameter.

First we will show the results of both the greedy and OTSC compared to that of an ideal algorithm in which the classification comes for free. That is, we assume we know the exact position of the zeros in the input so that all corresponding operations can be pruned without any testing overhead. Fig. 2.8 shows the number of additions and multiplications as functions of the quantization parameter (QP) of the Greedy and OTSC compared to the ideal case. In addition to using the Vetterli-Ligtenberg algorithm (Fig. 1.3 (b)) as the baseline algorithm, we show the ideal case when direct matrix multiplication method is used instead. It can be seen that in all cases the ideal direct matrix multiplication is inferior to the ideal fast search based algorithm.



(a) (b)

Figure 2.8: Number of additions (a) and the number of multiplications (b) needed for $\sigma^2 = 100$ and 700, using OTSC ('solid'), Greedy ('dashed'), Ideal fast IDCT ('dashed-dotted'), and Ideal matrix multiplication ('light-solid'). The OTSC and Greedy are optimized for each QP.

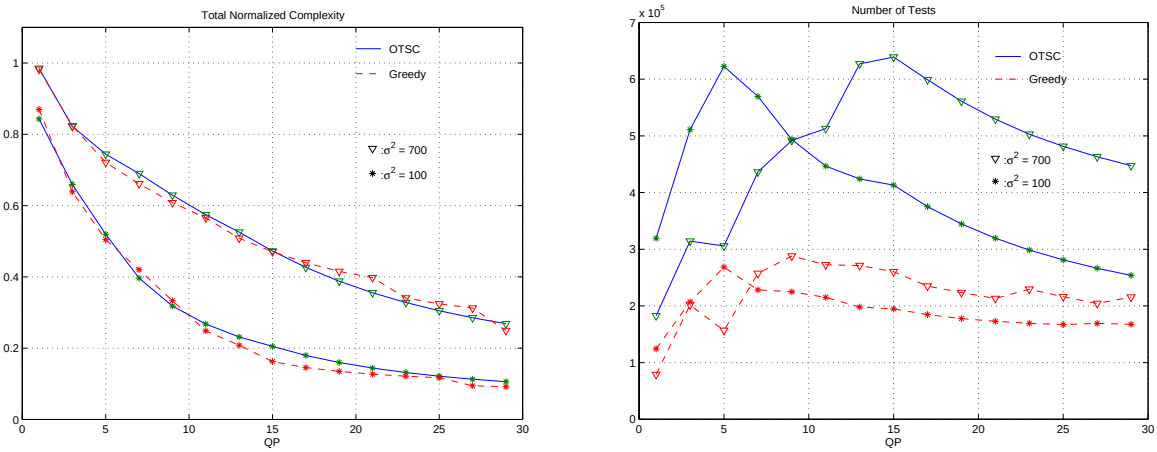
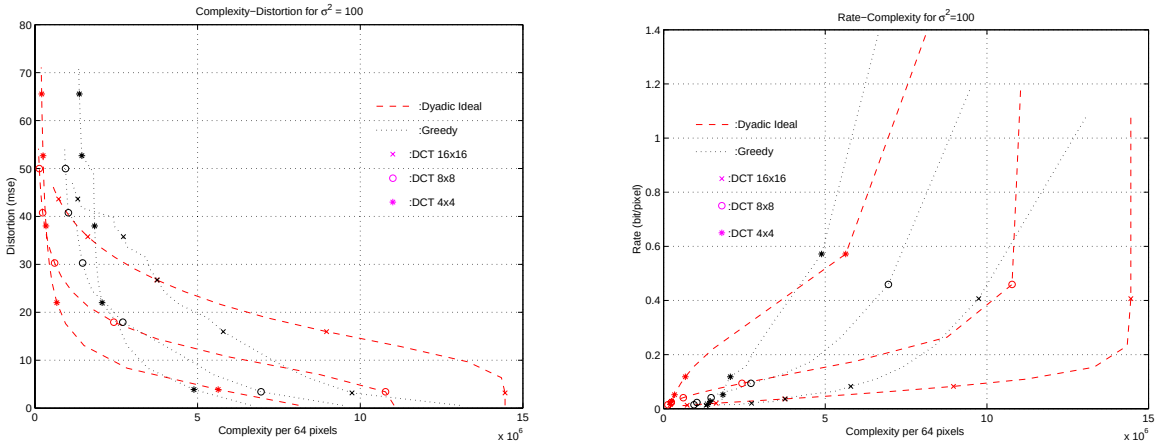


Figure 2.9: Total complexity (a) and the number of tests (b) needed for $\sigma^2 = 100$ and 700 using OTSC ('solid') and greedy ('dashed') algorithms.

One can also observe that in terms of number of additions and multiplications, the OTSC requires less operations than the Greedy approach. This is because the ability to refine the tests within OTSC allows better classification than the greedy approach. The result of the OTSC in terms of number of additions and multiplications is very close to the ideal case at high quantization and moves away from the ideal as finer quantization is used, see Fig. 2.8. This can be easily explained since in the low rate region most of the quantized DCTs are zero and therefore the zero tests become very efficient. On the other hand, at high rate there are less zero DCTs, thus resulting in a waste of zero mask test computations. Also in Fig. 2.9, the complexity reduction of the OTSC comes at the price of using numerous tests (as can be seen in Fig. 2.9 (b)), which essentially makes the algorithm very machine-dependent, i.e., the total cost could vary a lot from machine to machine. On the other hand, the number of tests (bit map testings) required by the greedy approach is much less than OTSC, thus compensating its larger number of multiplications and additions and resulting in similar overall complexity.

We emphasize that the greedy approach performs very close to the OTSC in terms of total complexity while the number of tests is much less. Therefore, it can be accurately used as an indicator of the complexity of the OTSC at various

block sizes with much less optimization effort. We show the distortion-complexity and rate-complexity (R-C) performance of different IDCT algorithms at different QP and block sizes in Figure 2.10 (a) and (b), respectively.



(a) (b)

Figure 2.10: (a) Complexity-distortion and (b) Rate-complexity curves for different algorithms, i.e., greedy ('dotted') and 2-D dyadic ('dashed'), for DCT size 16x16 ('x'), 8x8 ('o') and 4x4 ('*') at $\sigma^2 = 100$. The complexity unit is weighted operations per 64 pixels.

From Figure 2.10 (a), we can see that, for a given distortion, which block size gives the least complexity depends on σ^2 and QP. It can be seen that at low distortion, smaller block size is cheaper than large block size, whereas the reverse is true at high distortion. From Figure 2.10 (b), it can be seen that as rate becomes small, the complexity difference between each block size becomes smaller as well. Therefore, the conclusion is that using large block sizes at low rates results in comparable complexity and distortion performances to using smaller block size.

The C-D and R-C results of 2-D dyadic are also shown in Fig. 2.10 (a) and (b), respectively. At this point, we can also compare the 2-D dyadic testing to the greedy approach. We can see that at high rate the dyadic test requires higher complexity than the greedy algorithm because the classification is coarser for the higher frequency components. However, at lower rate the complexity of the dyadic test is lower since there are fewer tests in the dyadic test and the test is performed on a whole 2-D block, instead of each row or column, thus resulting in more efficient classification cost. In fact, the 2-D test can be performed efficiently

based on 1-D bitmaps. For example, to test for low-4x4 class of an 8x8 block, the lower 4 bits of column bitmap and lower 4 bits of first 4 row bitmaps are bitwise or and compared with zero. An alternative is to check the EOB symbol which represents the last non-zero coefficient in the zigzag scan order. From the EOB, one can determine the all-zero DCT area and apply a corresponding a reduced IDCT algorithm.

Therefore, we can combine the OTSC or greedy approach with the 2-D dyadic test, i.e., the classification will follow the dyadic testing first, if the resulting class is a full-NxN block, then the greedy or OTSC approach is applied for each row/column. At low rates, this method has the advantage of low testing cost from dyadic test while being able to refine the test at high rate using the greedy approach.

2.3.2 Real Image Data Results

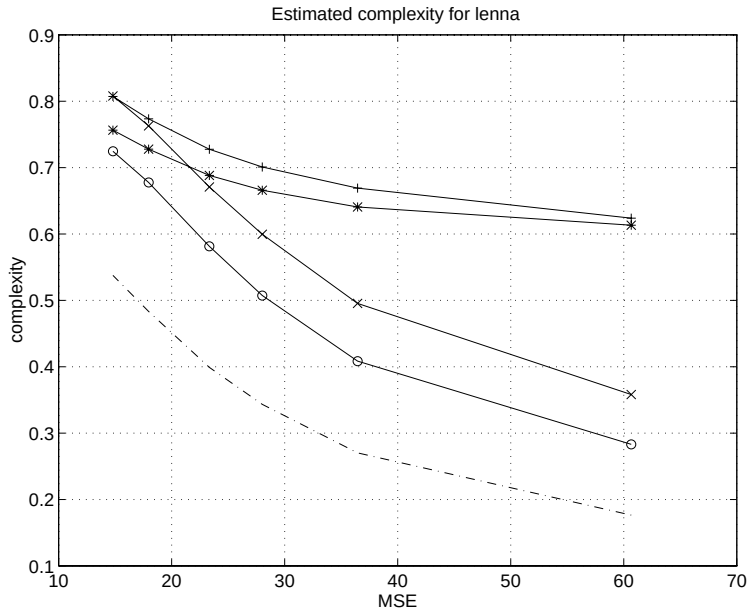


Figure 2.11: Normalized estimated complexity for "lenna" using CW with all-zero test algorithm ('+'), CW with all-zero test for the first 1-D IDCT and ac-zero test for the second 1-D IDCT ('x'), FW algorithm ('*'), and OTSC algorithm ('o').

We now show the results obtained with real image and video data using a real

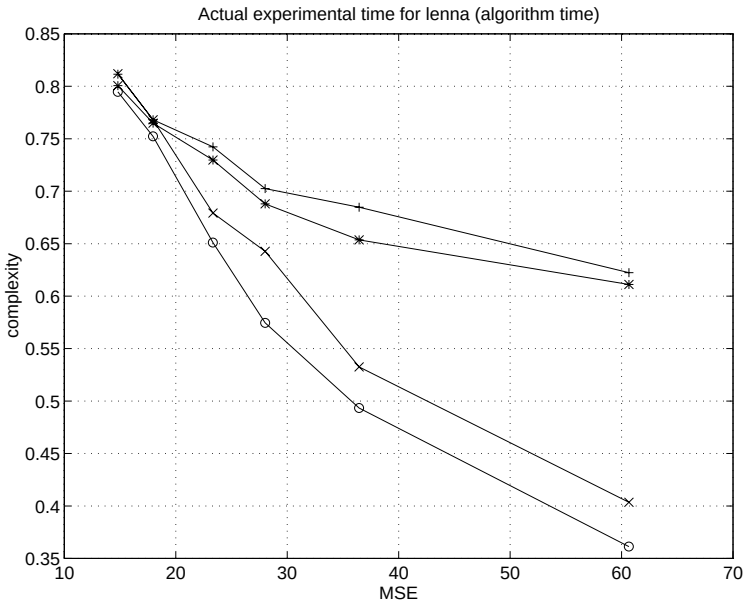


Figure 2.12: Normalized actual time complexity (only IDCT algorithm part) for "lenna" using CW with all-zero test algorithm ('+'), CW with all-zero test for the first 1-D IDCT and ac-zero test for the second 1-D IDCT ('x'), FW algorithm ('*'), and OTSC algorithm ('o').

image and video decoder. First we show the result of OTSC on image data (see Fig. 2.11). The results show the estimated complexity comparison between (i) IDCT based on our proposed method (OTSC), (ii) the baseline algorithm (CW) with all-zero test, (iii) the baseline (CW) algorithm with all-zero test for the first 1-D IDCT and AC-zero test for the second 1-D IDCT (since after the first 1-D IDCT, it is more likely for typical images that only the DC coefficient in the 1D vector is non-zero) and (iv) the FW method, for various mean squared error values obtained by changing the quantization parameter. We use the example quantization table from the JPEG standard [1]. All the values are normalized by the complexity of the baseline algorithm without all-zero test. Next, the actual implementation times are shown for IDCT algorithm part (Fig. 2.12) and total decoding time (Fig. 2.13), which includes the time for read-in and write-out of the input data. Also shown in Fig. 2.14 is the mismatch case, i.e., we use the IDCT algorithm optimized for a specific image at a certain quality factor for a different image and/or quality factor.

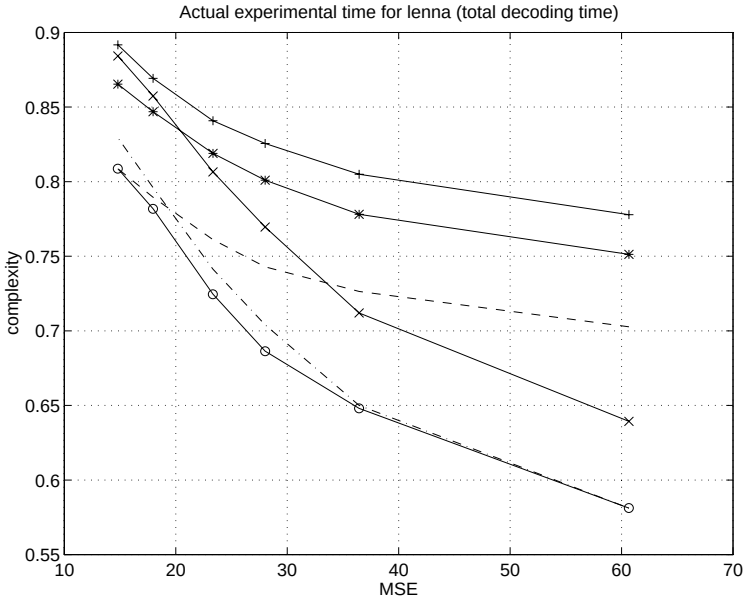


Figure 2.13: Normalized actual time complexity (total decoding time) for "lenna" using CW with all-zero test algorithm ('+'), CW with all-zero test for the first 1-D IDCT and ac-zero test for the second 1-D IDCT ('x'), FW algorithm ('*'), OTSC algorithm ('o'). OTSC for lenna at MSE 14.79 ('–'), and OTSC for lenna at MSE 60.21 ('-.').

Our proposed method (OTSC algorithm) achieves up to 50% complexity reduction for the IDCT part. When there is a mismatch between the statistics of the training data and those of the actual data being processed, the performance can be degraded. Another source of mismatch comes from the assessment for the cost of each operation, which must be precise enough to maintain the consistency between the estimated complexity and the actual complexity. As seen for both experimental results, even with mismatches, the optimized algorithm is a little bit inferior because of the lack of an exact model for the complexity. However, if our estimation for cost of operations is precise enough, a conclusion drawn from the result when applying optimized IDCT algorithm for "lenna" with MSE 60.21 could be that the degradation for using an algorithm optimized for a higher MSE (which overestimates the number of zeros) is less than the degradation for an algorithm optimized for a lower MSE (which underestimates the number of zeros). This may not be true for other types of images. Also it has to be pointed out that

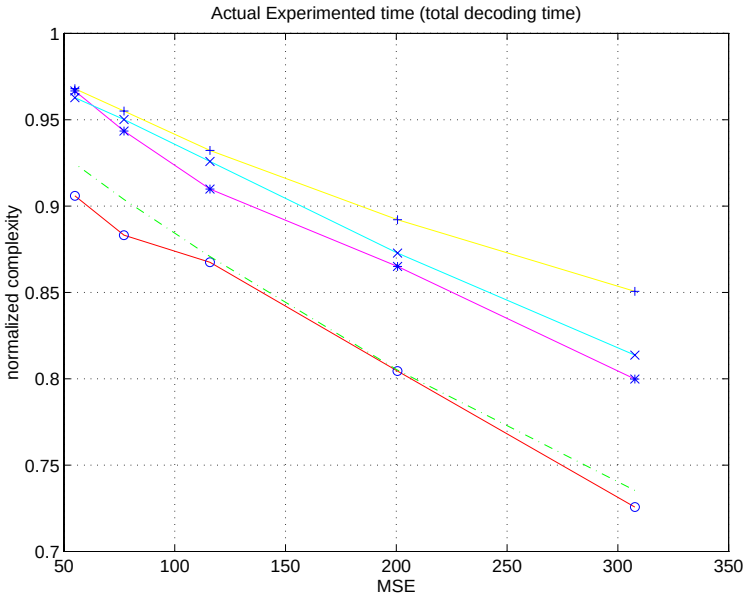


Figure 2.14: Normalized actual time complexity (total decoding time) for baboon image using CW with all-zero test algorithm ('+'), CW with all-zero test for the first 1-D IDCT and ac-zero test for the second 1-D IDCT ('x') , FW algorithm ('*'), OTSC algorithm ('o'), and OTSC for lenna with MSE 14.79 ('–')

our current implementation has concentrated mostly on the issue of structuring the tests, and thus uses a very simple programming style. Our goal at this stage was to implement all the algorithms with a similar style so that a fair comparison is possible.

The results of the combined 2-D dyadic and OTSC IDCT are shown in Fig. 2.15 using TMN's codec [58] and coding 249 frames of the "Foreman" sequence at different target bit rates. The inverse quantization complexity is also considered since there is an overhead cost of bitmap generation for both OTSC and dyadic. It can be seen from Fig. 2.15 that the combined dyadic-OTSC gets an extra 6-9% complexity reduction from the OTSC. An explanation is that the 2-D classification is cheaper than separate 1-D classification, if it matches the statistics of the data well. In typical image and video data, nonzero coefficients tend to cluster around DC component. That is why we observe extra speedup using dyadic classification.

We now discuss the issue of statistical estimation. One scenario to obtain the statistics for the proposed VCA IDCTs in a practical encoding system would be to

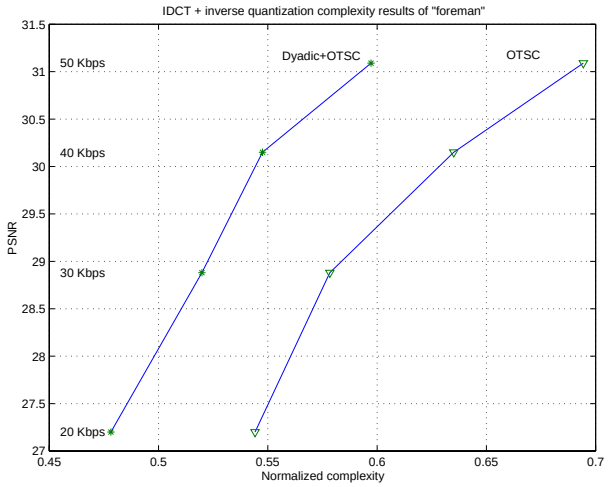


Figure 2.15: The complexity (CPU clock cycle) of the IDCT + Inv. Quant. normalized by the original algorithm in TMN at various PSNRs (different target bit rates) using OTSC ('△') and combined dyadic-OTSC ('*').

have the encoder send side information about the optimized tree-structure zero-test IDCT algorithm to the decoder along with the data, such that the decoder can do the IDCT operation with the minimum complexity. However, this comes with the price of bit rate overhead. As an alternative, a tree-structured or greedy optimization can be performed at the decoding end using the past statistics of input data. This seems to be more compatible with the existing image and video standards. However, there is an issue of run-time algorithm change. It is impossible to compile the optimal IDCT at run-time. Therefore, instead of optimizing the VCA on the fly, a more practical way is to have a set of pre-optimized VCAs that covers a wide range of image statistics at run-time and choose the one with smallest complexity for the image being operated.

2.4 Distortion/decoding time tradeoffs

As we mentioned earlier for the scalability issue, one way to achieve IDCT computational scalability is by changing QP at the encoder side. This can be applied to the case where a limited complexity decoder is the major concern. The C-D tradeoff will be addressed, i.e., within a given computational complexity at the

decoder, the encoder adjusts the quantization parameter such that the distortion is minimized while satisfying the complexity constraint. Note that this requires the encoder to have enough information about the operation at the decoder end in order to change the QP accordingly.

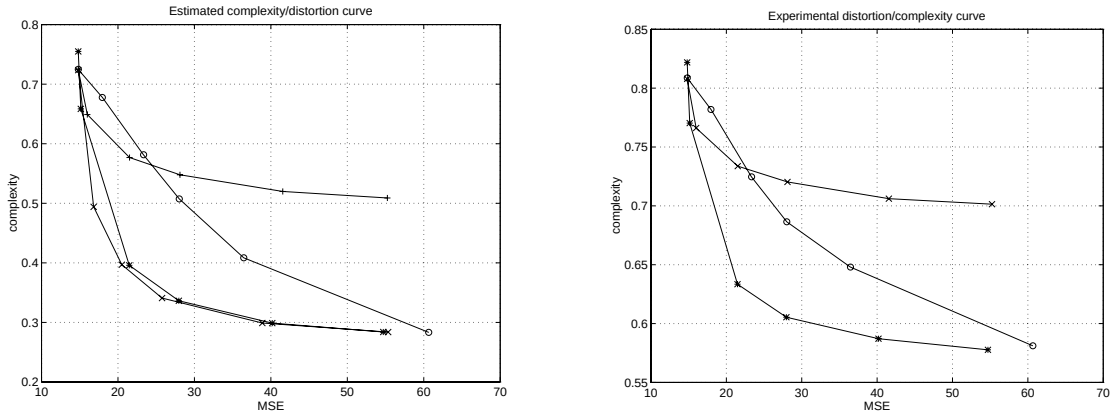
As described in the previous section a VCA implementation of the IDCT can be obtained given a "typical" set of input images. As is clear from the results, the coarser the quantization the faster the decoder can run. Thus, a decoder based on a VCA IDCT is inherently *computationally scalable*. Unlike the JPEG case in which the quantization parameter is fixed for all blocks in a coded image, for video encoding standards such as MPEG1-2 or H.261-3 the encoder can adjust its quantization parameter on every macroblock in order to try to find the best picture quality for a given bit budget. The same idea can be applied to a complexity-distortion (C-D) framework where the encoder can control the decoding speed by assigning to each block one out of several available quantizers (coarser quantizers result in faster operation). The question then arises on how to optimally select those quantization steps *for a given decoding time budget*. This leads to a formulation where the encoder operates based on a complexity-distortion (C-D) tradeoff, rather than on the traditional rate-distortion (R-D) tradeoff. As an application, one could for instance store images to be downloaded by different types of hosts so that the slower hosts can access lower quality images, which can be decoded more quickly. Similarly, the quality of the decoded images can be selected based on the load of the shared host. The problem can be formalized as finding the quantizer assignment $\{j,i\}_{opt}$ such that

$$\sum_j D_j(i) \;\text{ is minimized while }\; \sum_j T_j(i)) < T_{budget},$$

where T_{budget} is the total time (or complexity) budget, $D_j(i)$ and $T_j(i)$ are, respectively, the distortion and decoding time when quantizer i is used for block j. The problem can be solved using the well-known Lagrangian multiplier method [9]

$$\{j,i\}_{opt} = arg\min_{\{j,i\}}(\sum_j D_j(i) + \lambda \cdot \sum_j T_j(i)) \tag{2.10}$$

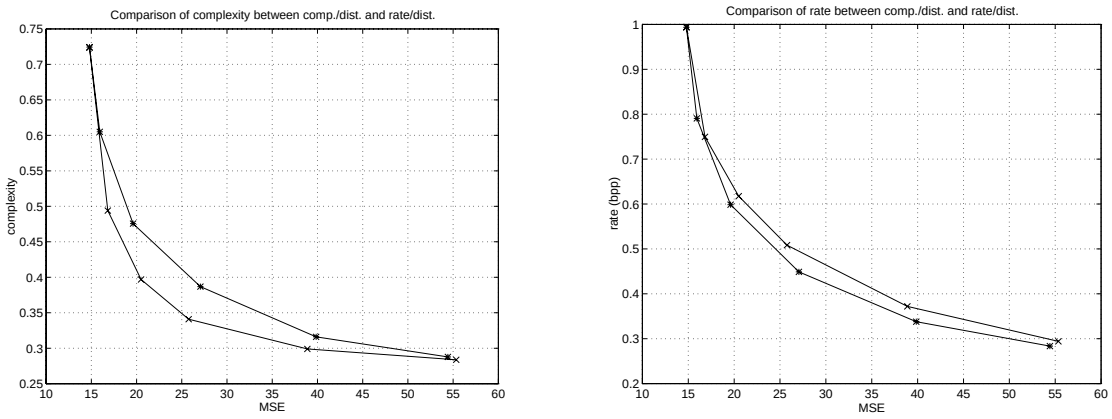
where $\lambda \geq 0$ is the Lagrange multiplier, which will have to be adjusted so that the budget is met.



(a) (b)

Figure 2.16: Distortion versus (a) estimated IDCT complexity (b) experimental decoding time of "lenna" using fixed quantizer encoder and OTSC decoder ('o'), Lagrange multiplier results ('x'), encoder follows Lagrange multiplier results but decoder uses a single OTSC for MSE=60.66 ('*') and MSE=14.80 ('+'), respectively.

Figs. 2.16 and 2.17 summarize our results. In this experiment, we design six OTSC algorithms for six different quantization parameters based on the statistics of the test image. The OTSC design is based on the approach discussed in the previous section. We then have the complexity and distortion for each block and each quantizer, as required by the Lagrange optimization of (2.10). For a given λ, we find the quantizer that minimizes (2.10) for each block. Finally, we repeat the minimization at different values of λ until the complexity constraint is met. Therefore, the encoder applies different quantizers for different blocks according to the optimization result. The quantizer information must also be sent to the decoder for proper inverse quantization. At the decoder side, the OTSC algorithm that corresponds to the quantizer selected for the block is used. However, in most of the cases, having many OTSCs at the decoder results in excessive overhead. A more practical approach is to have a single OTSC optimized for a single quantizer and to use this OTSC for all blocks.



(a) (b)

Figure 2.17: (a) Complexity-Distortion curve obtained from C-D ('x') and R-D ('*') based optimization. (b) Rate-Distortion curve achieved when C-D ('x') and R-D ('*') based optimization. The complexity is normalized by the complexity of the baseline Vetterli-Ligtenberg algorithm.

Fig. 2.16 indicates that quantizer allocation for each block optimized for complexity results in very significant reductions in complexity. As expected, the C-D allocation outperforms the other methods. Note that in Fig. 2.16 (a) we compare the C-D allocation when multiple quantization-dependent algorithms are used and when a single algorithm is used (OTSC optimized for fine quantizer or coarse quantizer). When a coarse quantizer is used the performance is very close to that of the multiple algorithm approach. This result is consistent with the previous result in Fig. 2.13 where we can use an OTSC optimized for a coarse quantizer for blocks coded with finer quantization with only small degradation in C-D performance. For the actual decoding tests we thus use a single algorithm. Each point in the C-D optimized curves is obtained with a given parameter λ.

One may argue that the complexity can be controlled via rate control since in general, the quantizer is determined by bit budget. If the bit budget is small, a coarse quantizer will be used, which in turn results in more DCT coefficients being set to zero, so that decoder with VCA IDCT will be faster. However, Figs. 2.17 (a) and (b) demonstrate how a complexity driven quantizer allocation results in better C-D performance than its rate driven counterpart. Even though C-D and R-D curves from both C-D based and R-D based optimization are close together, as

expected better performance can be observed when the correct budget constraint parameter is used. However, we may also use a rate budget constraint instead of a complexity constraint to obtain sub-optimal C-D performance.

2.5 Rate-Complexity-Distortion Quadtree Optimization

There are only a few works addressing the rate-complexity-distortion tradeoffs. Among those are [22] and [23] in which the complexity is considered as a factor to determine the R-D performance, and can be varied by changing the size of the block transform. In [23], the convexity of the R-D-C curve of Gauss-Markov sources is proved. Furthermore, in [23] the C-D comparison for a given rate between KLT and DCT is shown. An interesting conclusion can be drawn that with this particular source the C-D curve of the DCT is better. However, in these works the complexity is not input-dependent, i.e., it considers the worst-case scenario which implies the complexity is a function of the block size. If we use the variable complexity approaches proposed in the previous sections, it can be seen that the C-D relation also follows a similar tradeoff as R-D when QP changes (see Figure 2.10(a)) whereas (see Figure 2.10(b)) the R-C have the tradeoff when the block size, not QP, varies. It is true because when the block size is larger, the complexity required for block transform is also larger while the rate is smaller as a result of the transform coding gain, and vice versa. However, since the number of available block sizes is limited, in order to obtain a convex hull of the R-C curve, the Lagrange multiplier method can be used as it is in R-D applications [9]. In [97] and [21], quadtree-based coding is used for better R-D performance by using a different block size for different regions and types of inputs. In [21], the R-D optimization criterion is used with the optimal tree pruning algorithm [98]. The optimization already incorporates the fact that the encoder has to send the block size information to the decoder.

In this section, we present a more general R-C-D tradeoff based on the Lagrange multiplier technique, which incorporates the selection of QP and block size. We will first formalize a problem similar to the optimal quadtree based coding of

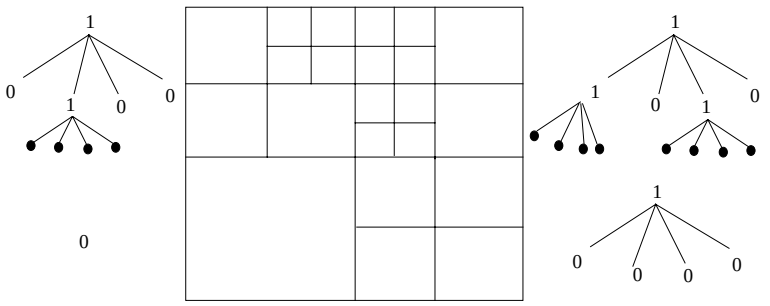


Figure 2.18: Quadtree structures of four 16x16 regions and the corresponding representative bits.

[21], with an additional constraint on the complexity. Then the QP as well as the block size are optimally selected for each block of the image or the video frame. The complexity constraint is the target total decoding time at the decoder.

We use the same notations as [21]. Based on Figure 2.18, let $X_{n,i}$ denote the i-th DCT block at level n. As in the figure, the leaves (smallest blocks) correspond to a block size of 4x4, the middle nodes represent 8x8 blocks and the root of each tree corresponds to block size of 16x16. Let $n = 0, 1, 2$ represent blocks of size 4x4, 8x8 and 16x16, respectively. The scheme can be generalized to smaller block size (i.e., 1x1 block) or larger block size such that level n corresponds to block size 2^nx2^n. However, given the limitation on computation, we use only 3 block sizes. The side information needed to be sent to the decoder about the structure of the quadtree is '0' if the block is not split and '1' otherwise. If the parent node at level $n+1$ is split, then there are 4 children blocks, i.e., $X_{n,i}$ for $i = 0, 1, 2, 3$. Let $r_k(X_{n,i}), d_k(X_{n,i})$ and $t_k(X_{n,i})$ denote the bits, distortion and complexity of $X_{n,i}$ in block k. The quadtree optimization is based on the bottom up approach in [98], in which the decision is made on whether smaller blocks with suboptimal R-C-D operating points are better off merged for better R-C-D performance.

The goal of the optimization is to *minimize the total distortion* $\bar{\mathbf{D}}$ *such that the total rate* $\bar{\mathbf{R}}$ *and complexity* $\bar{\mathbf{T}}$ *are under their respective budget constraints* R_b *and* T_b. Using the Lagrange multiplier method, we can convert this constrained problem into an unconstrained problem by minimizing the following objective function

$$\min \sum_k d_k + \lambda_r \sum_k r_k + \lambda_t \sum_k t_k \tag{2.11}$$

where λ_r and λ_t are Lagrange multipliers controlling the tradeoffs between R-D and C-D, respectively. These parameters need to be searched such that the constraints are met.

The optimization of the quadtree can be described as follows. Assume that the optimal rate, distortion and complexity of block k at level n are known to be $r_k^*(X_{n,i}), d_k^*(X_{n,i})$ and $t_k^*(X_{n,i})$ for $i = 0, 1, 2, 3$. These four subtrees will be merged if

$$d_k(X_{n+1}) + \lambda_r r_k(X_{n+1}) + \lambda_t t_k(X_{n+1}) \leq \sum_{i=0}^{3} [d_k^*(X_{n,i}) + \lambda_r r_k^*(X_{n,i}) + \lambda_t t_k^*(X_{n,i})] \tag{2.12}$$

where the left side is the minimal cost function (over all possible quantizer choices) for the parent node. Then the optimal rate, distortion and complexity of level $n+1$ are updated and the optimization proceeds to the next level. When the merge decision is made, a one-bit side information has to be sent to the decoder as well. Therefore, the new suboptimal solution at level $n+1$ becomes

$$r_k^*(X_{n+1}) = \begin{cases} 1 + r_k(X_{n+1}) & \text{if merge} \\ 1 + \sum_{k=0}^{3} r_k^*(X_{n,i}) & \text{if split} \end{cases} \tag{2.13}$$

The resulting distortion and complexity are

$$d_k^*(X_{n+1}) = \begin{cases} d_k(X_{n+1}) & \text{if merge} \\ \sum_{k=0}^{3} d_k^*(X_{n,i}) & \text{if split} \end{cases} \tag{2.14}$$

$$t_k^*(X_{n+1}) = \begin{cases} t_k(X_{n+1}) & \text{if merge} \\ \sum_{k=0}^{3} t_k^*(X_{n,i}) & \text{if split} \end{cases} \tag{2.15}$$

The process continues until the root node is reached and the optimal quadtree of every region in an image is computed. Then the Lagrange parameters λ_r and λ_t are adjusted, and the whole process repeats until the bit budget and the complexity budget constraint are met. There are several methods for adjusting the Lagrange multipliers. In this work, we use the linear approximation algorithm proposed in [99] in which the Lagrange multipliers are shrunk (by a factor $\gamma < 1$) and expanded (by a factor $1/\gamma$) when the total constrained quantity is below or

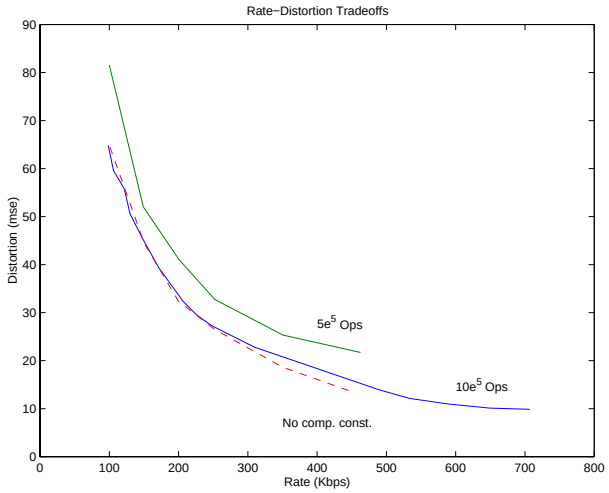


Figure 2.19: Constant-complexity rate-distortion curves. When complexity constraint is loosen, the rate-distortion performance can be better. 'dashed' curves show unconstrained complexity result.

above the budget, respectively. Using this algorithm, we can generate constant-complexity R-D curves and constant-rate C-D curves. In order to find constant distortion R-C curves, we rewrite (2.11) as

$$\min \frac{1}{\lambda_r} \sum_k d_k + \sum_k r_k + \frac{\lambda_t}{\lambda_r} \sum_k t_k \tag{2.16}$$

which can be viewed as a rate minimization problem with distortion and complexity constraint. Then, the linear approximation can be applied to find the new Lagrange multipliers, i.e., $\frac{1}{\lambda_r}$ and $\frac{\lambda_t}{\lambda_r}$.

Figures 2.19-2.21 show experimental results of the quadtree optimization for the first 30 INTER frames of the "Miss America" sequence. We can select the QP out of 5 possible choices, namely, 4, 10, 16, 22 and 28, and 3 block sizes are allowable for the DCT, i.e., 4x4, 8x8 and 16x16. As in [21], we do not take the bits necessary for motion vector coding into consideration since it can be considered as a fixed overhead expense and depends on the motion estimation. The distortion is in the unit of MSE and the complexity is the estimated number of operations[3].

[3]Since the complexity depends on machine and load condition, we only use the estimate number of operation as it provides framework for future development.

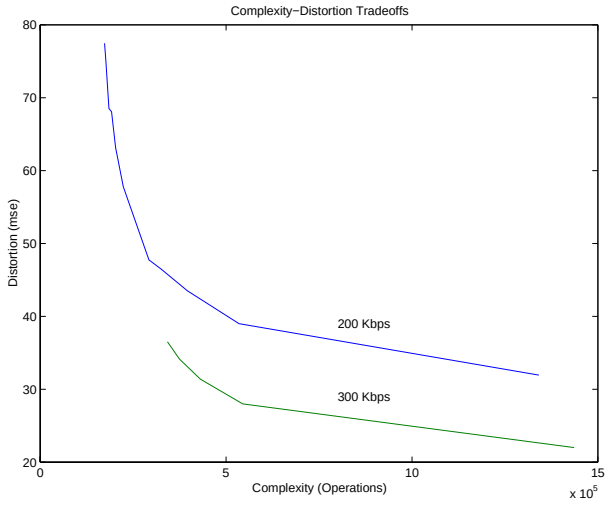


Figure 2.20: Constant-rate complexity-distortion curves at 200 Kbps and 300 Kbps. As rate is more constrained, C-D performance gets worse.

In Figure 2.19, we show R-D curves for constant complexities. It can be seen that when the complexity constraint is more limited, the R-D performance is degraded. As the complexity budget becomes tighter, the coder is forced to use smaller block size or coarser quantization. We also show the result of the optimal R-D quadtree optimization without complexity constraint, which gives the best R-D result.

In Figures 2.20 and 2.21, constant-rate C-D curves and constant-distortion R-C curves are shown. Again, it can be seen that when one parameter of the R-C-D triplet is more strictly limited, the tradeoff functions of other 2 parameters get worse. For example, as the rate budget becomes tighter, a larger block size and a larger QP tend to be chosen. As a result, complexity increases whereas the larger QP means more distortion. On the other hand, as distortion requirement is more demanding, smaller QP and larger block size are likely to be selected. Consequently, higher rate and higher complexity are unavoidable. In addition, note that all the results follow the convexity theorem in [23].

In the case of real-time encoding, in order to avoid the DCT computational load for all 3 block sizes that are necessary for computing the rate and distortion information, a fast approximate DCT (as will be presented in chapter 3) can be employed. In addition, the encoder may assume that the decoder uses only the dyadic testing for IDCT, instead of combined dyadic and OTSC. Therefore,

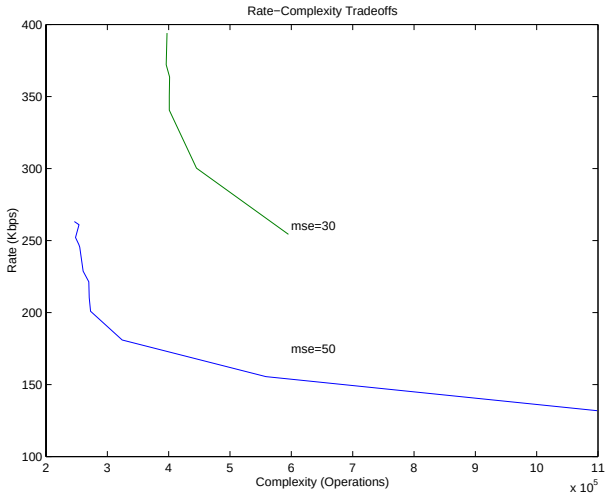


Figure 2.21: Constant-distortion rate-complexity curves at MSE = 30 and 50. As distortion requirement is more rigid, the R-C performance becomes worse.

the complexity evaluation can be done faster without actually testing the DCT coefficients.

2.6 Summary and Conclusions

We formalized a variable complexity algorithm for IDCT by discussing the classification problem and the reduced complexity algorithms that can exploit the knowledge obtained from the classification. Then, we presented a sequential classification technique using zero mask test as a tool for testing. We proposed a greedy optimization to select the order in which the classes should be tested by sequential classification that achieve an acceptable suboptimal solution with reduced complexity. We provided R-C-D results for a given input model and at different block sizes. Next, we proposed an alternative tree-structured classification that exploits the structure of the baseline algorithm in the classification process. Given the statistics of the input, we showed how to construct an optimal classifier based on the tree-structured concept. The experimental results show significant complexity savings for typical images at low rates. We also proposed a combined algorithm in which the heuristic 2-D dyadic classification is performed first based on observation of empirical data, and the VCA IDCT can be applied

later for finer classification. The result gives extra 5% complexity reduction compared to the OTSC. This implies that since the image and video content is in general similar, one can apply a heuristic VCA IDCT without much degradation compared to the optimal solution. Also, the 2-D classification gives better classification gain at low rate.

Finally, we addressed the problem of distortion/decoding time tradeoff using OTSC. The problem is solved using a Lagrange multiplier technique for optimal quantization selection at the encoder side. We extended this work to quadtree optimization for rate-complexity-distortion tradeoff. We showed some experimental results on a video sequence where the goal is to minimize the distortion in a rate and complexity constrained environment. Using VCA approaches, it may be possible to use larger block sizes efficiently, thus improving the coding gain. The applicability of these techniques is not limited to DCT. There are also some works trying to achieve fast inverse wavelet transform, e.g., work by Fernandez and Ortega in [100], where a similar tree-structure classification for a VCA implementation of inverse wavelet transform is proposed.

Chapter 3

Forward Discrete Cosine Transform

In this chapter, we propose 2 classes of fast DCT algorithms, namely a VCA approach and an approximate approach. In the VCA approach, we try to determine, from the set of inputs or intermediate outputs of the operations, which set of outputs will be quantized to zero. The classification is performed in a hierarchical manner. We also show how to optimize the test structure for a given training data, with techniques similar to those presented for IDCT. However, the complexity saving of this approach at high bit rate coding (for both MPEG2 and JPEG) are very marginal. Therefore, we resort to lossy approximate DCT algorithms in which, instead of selectively computing a subset of the DCT, all DCT coefficients are approximated. These algorithms are multiplication-free, thus yielding great reduction in complexity, and the degree of approximation can be selected based on the quantization level, so that the complexity is quantization dependent. The error analysis is also presented. Unlike VCA, at high bit rate, the approximate DCT still manifests significant gain (e.g., around 30% reduction in computation time) with a slight degradation in R-D performance. As a benchmark for the performance, Pao and Sun's statistical sum of absolute value test (SSAVT) is reviewed. The error analysis of SSAVT is then derived to explain the good performance of SSAVT. Finally, we introduce several hybrid algorithms, e.g., combined approximate and VCA, combined SSAVT and approximate DCT,

and combined SSAVT, approximate and VCA DCT. The experimental results for low-bit rate coding (H.263) show significant gain of more than 50% complexity saving using the combined algorithm, as compared with a standard fast DCT algorithm.

3.1 Exact VCA DCT

Our goal is to avoid the computation of those coefficients that will be later quantized to zero. The main difficulty is that DCT coefficients which are quantized to zero are not known unless they have already been computed. If the zero (to be quantized to zero) coefficients are known, only a subset of output needs to be computed and only necessary operations are performed. We now explore an algorithm which can deal with nonuniform quantization and allows a finer classification of the inputs.

3.1.1 Input Classification: Pre-transform Deadzone Test

The problem of determining whether the output DCTs will be quantized to zero can be viewed geometrically as that of determining whether the input vector is in the dead-zone region of the corresponding quantizers or not (see Figure 3.1). Since the DCT is an orthonormal transform, if the input is in the dead-zone so is the output. In Figure 3.1, the dead-zone is the solid rectangle in the (y_1, y_2) output coordinate and corresponds to the region where the DCT coefficients are quantized to zero. The input coordinate is (x_1, x_2). The test region equivalent to [61] (see Chapter 1 is shown as a dashed square which is equivalent to thresholding the absolute sum of the input.

Let $\mathbf{x}(i)$ be an input element from the spatial domain input vector $\bar{\mathbf{x}}$, and let $\bar{\mathbf{X}}$ be the corresponding DCT output vector. Let $\bar{\mathbf{Q}}$ and $\bar{\mathbf{q}}$ be a vector of the quantization and a vector of the thresholds for the coefficients in the input. $\bar{\mathbf{Q}}$ can also be viewed as the vector of deadzone boundary. We present a test region based on the following test,

Formulation 4 (Pre-transform Deadzone Test) *We want to find* $\bar{\mathbf{q}}$ *with maximal hypercube volume,* $\prod_i q(i)$*, satisfying the triangle inequality* $|\mathbf{D_N}| \cdot \bar{\mathbf{q}} < |\bar{\mathbf{Q}}/2|$ *such that if input* $|\mathbf{x}(i)| < \mathbf{q}(i)/2$ *,* $\forall$ *i, then* $\mathbf{X}(i) \leq \mathbf{Q}(i)$*,* $\forall$ *i, i.e.,* $\bar{\mathbf{X}}$ *will be quantized to zero and, thus, can be ignored.*

$|\mathbf{D_N}|$ is the elementwise absolute value of the DCT matrix of size NxN. The solution to the above can be obtained numerically by searching over all values of $\bar{\mathbf{q}}$. Note that the ideal test would be one such that the dead-zone corresponding to the test fit as close as possible within the dead-zone corresponding to the actual quantization. This test region is equivalent to a tilted solid square in the dead-zone in Figure 3.1. For simplicity, we use a square dead-zone i.e. $\mathbf{q}(i) = q$ for all i, since its resulting volume is almost as large as the maximal non-square dead-zone. In order to perform this test we will need to compute at most N absolute values, N comparisons and $N-1$ logical operations.

This test classifies the input into 2 classes to which we assign either full operation DCT or no operation DCT. Consider now the baseline fast algorithm in Figure 1.3. It can be seen that the computation is divided into three stages. From Figure 1.3, let

$$\mathbf{D_8} = \begin{bmatrix} \mathbf{H}_1^3 & 0 & 0 & 0 \\ 0 & \mathbf{H}_2^3 & 0 & 0 \\ 0 & 0 & \mathbf{H}_3^3 & 0 \\ 0 & 0 & 0 & \mathbf{H}_4^3 \end{bmatrix} \begin{bmatrix} \mathbf{H}_1^2 & 0 \\ 0 & \mathbf{H}_2^2 \end{bmatrix} \mathbf{H}_1^1$$

where $\mathbf{H}_1^1$ has dimension 8×8, $\mathbf{H}_1^2$ and $\mathbf{H}_2^2$ have dimension 4×4 and $\mathbf{H}_1^3, \mathbf{H}_2^3, \mathbf{H}_3^3$ and $\mathbf{H}_4^3$ are 2×2 matrices. Therefore, we can write the output of each stage as follows,

$$\mathbf{H}_1^1 \bar{\mathbf{x}} = [\bar{\mathbf{x}_1^2}\ \bar{\mathbf{x}_2^2}]^t, \tag{3.1}$$

$$\begin{bmatrix} \mathbf{H}_1^2 & 0 \\ 0 & \mathbf{H}_2^2 \end{bmatrix} \mathbf{H}_1^1 \bar{\mathbf{x}} = [\bar{\mathbf{x}}_1^3\ \bar{\mathbf{x}}_2^3\ \bar{\mathbf{x}}_3^3\ \bar{\mathbf{x}}_4^3]^t. \tag{3.2}$$

Let us denote $\bar{\mathbf{x}}_1^1 = \bar{\mathbf{x}}$. Therefore, $\bar{\mathbf{x}}_j^i$ represents the j-th output vector from stage i. From the above, we can apply the proposed test method at the beginning

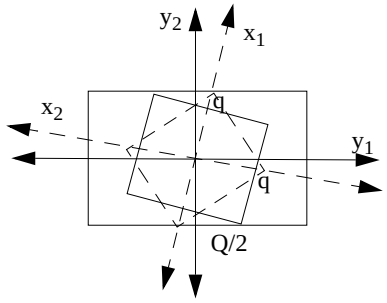


Figure 3.1: Geometric representation of dead-zone after rotation.

of each stage, i.e., testing $\bar{\mathbf{x}}_1^1$ before $\mathbf{H}_1^1$ operation, $\bar{\mathbf{x}}_1^2$ and $\bar{\mathbf{x}}_2^2$ before $\mathbf{H}_1^2$ and $\mathbf{H}_2^2$, and $(\bar{\mathbf{x}}_1^3, \bar{\mathbf{x}}_2^3, \bar{\mathbf{x}}_3^3, \bar{\mathbf{x}}_4^3)$ before $(\mathbf{H}_1^3, \mathbf{H}_2^3, \mathbf{H}_3^3, \mathbf{H}_4^3)$. Let the associated threshold vectors be denoted by q_1^1, (q_1^2, q_2^2), and $(q_1^3, q_2^3, q_3^3, q_4^3)$, respectively. The test structure is shown in Fig. 3.2. Starting from the root of the diagram, at each test, if the condition is satisfied no further operations are performed and coefficients at the end of paths starting from the test are set to zero. Otherwise, we continue toward the leaves.

In computing the thresholds of each test, we must incorporate the remaining operations need to be done after each stage, e.g., q_1^1 is computed from the DCT matrix, $\mathbf{D_8}$, while q_1^2 is computed based on the remaining transformation of $\begin{bmatrix} \mathbf{H}_1^3 & 0 \\ 0 & \mathbf{H}_2^3 \end{bmatrix} \mathbf{H}_1^2$ and q_2^2 is computed based on $\begin{bmatrix} \mathbf{H}_3^3 & 0 \\ 0 & \mathbf{H}_4^3 \end{bmatrix} \mathbf{H}_2^2$.

Note that this classification is not restricted to detecting all-zero blocks as in [61] and can thus be used to determine whether subsets of the output coefficients will be zero. This method can also be extended to a separable 2-D DCT (row-column 1-D DCT) with the use of Kronecker (or tensor) product, i.e., $\mathbf{D_N} \cdot \mathbf{x} \cdot \mathbf{D_N^t} = (\mathbf{D_N} \otimes \mathbf{D_N})\bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is a row-column scanned 1-D vector from the matrix $\mathbf{x}$, and the thresholds are obtained in the same manner. Furthermore, hierarchical classification for a non-separable 2-D DCT is possible by simply post-multiplying (3.1) and (3.2) with $\mathbf{H}_1^{1t}$ and $\mathbf{H}_1^{1t} \begin{bmatrix} \mathbf{H}_1^2 & 0 \\ 0 & \mathbf{H}_2^2 \end{bmatrix}^t$, respectively.

However, when the proposed VCA, which is based on the triangular inequality, is used for larger vector size the efficiency of detecting input in the dead-zone greatly decreases. We will show how the vector dimension affect the efficiency of

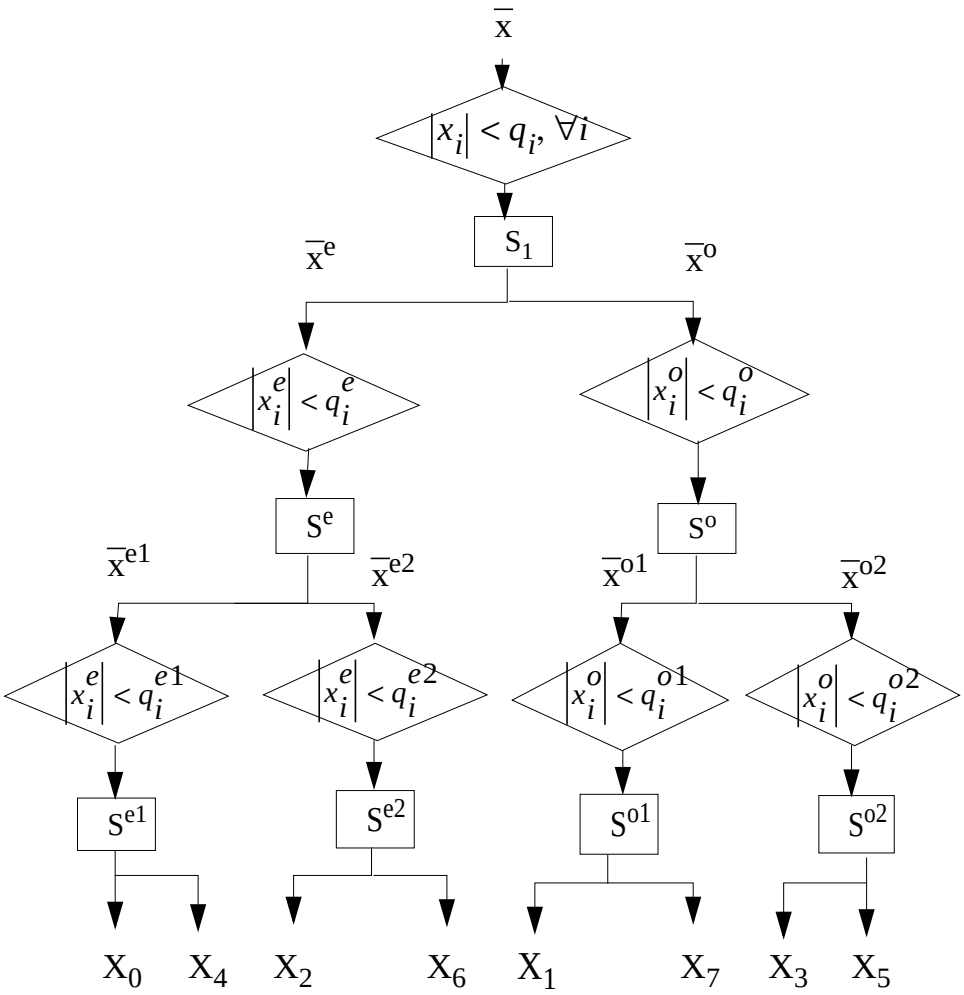
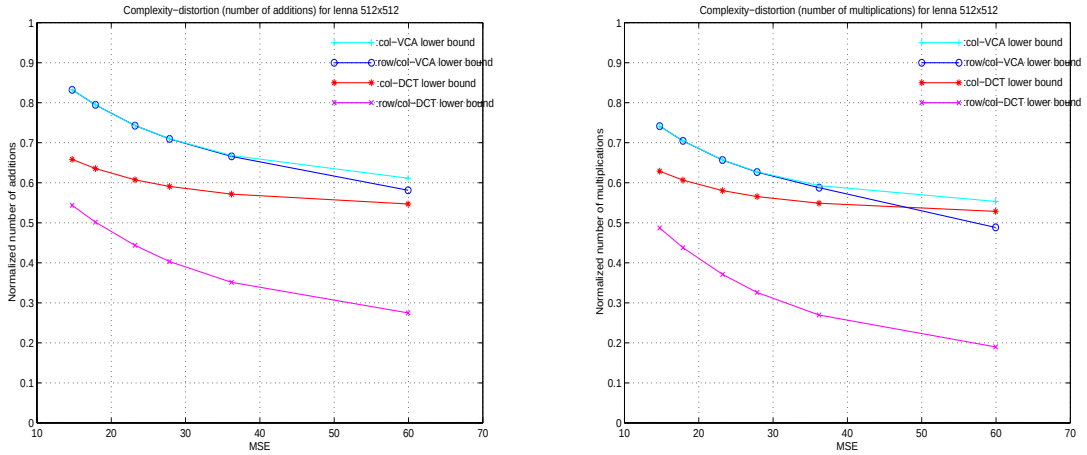


Figure 3.2: Proposed VCA algorithm.

the test. Our threshold designing problem is equivalent to the problem of fitting the biggest circle into the output dead-zone in Fig. 3.1. Assume the dead-zone is shrunk to a square with the width equal to the narrowest side of the dead-zone, then the ratio of the biggest sphere area and dead-zone area (2-D) is $\pi/4$. For a size n input vector, let V_n° be the volume of the sphere, and let $V_n^\square$ be the volume of the pre-transform dead-zone. It can be derived that

$$\frac{V_n^\circ}{V_n^\square} = \frac{\prod_{i=2}^{n} \int_{-\pi/2}^{\pi/2} \cos^n(\theta)\, d\theta}{2^n}, \tag{3.3}$$



(a) (b)

Figure 3.3: Comparisons of original and pruned algorithms for different distortions (a) number of additions, (b) number of multiplications. The DCT lower bound corresponds to computing only the subset of coefficients that will be non-zero after quantization. The VCA lower bound corresponds to pruning subject to the classification mechanisms of Section 3.1.2, i.e., we can only prune subsets of coefficients which are tested jointly.

$$\text{with} \quad \int_{-\pi/2}^{\pi/2} \cos^n(\theta)\, d\theta = \begin{cases} 4\sum_{q=0}^{\lfloor \frac{n}{2} \rfloor - 1} \binom{n}{q} \frac{\sin((n-2q)\frac{\pi}{2})}{(n-2q)} & \text{for n odd} \\ \binom{n}{n/2} \frac{\pi}{2^n} & \text{for n even} \end{cases}$$

It can be evaluated that as the dimension grows larger, this ratio becomes smaller, which in turn means that the ratio of our threshold hypercube to the actual dead-zone volume is even smaller. This means that the area that is not covered by the threshold hypercube but is part of dead-zone gets larger and thus left unexploited. As mentioned earlier that 2-D DCT is equivalent to a 1-D transform using a Kronecker multiplication and the resulting dimension of the transform is the square of the original dimension. Therefore, as can be seen in Fig. 3.3, the efficiency of the classification becomes less significant when we try to classify input before the first direction (rowwise) 1-D DCT, since the equivalent size of input vector becomes 64 instead of 8. On the other hand, if we apply the proposed VCA to the second direction (columnwise) 1-D DCT only, the size of each input vector is 8 because the DCT operation of each column can be done independently, thus giving more classification gain.

The result of the classification is shown in terms of number of operations as normalized by the fast algorithm in Figure 1.3, which is used for separable row-column 1-D DCT. We encode the "lenna" image of size 512x512 using different quantization parameters to obtain the complexity at different qualities for the decoded images. The DCT computation is exact and the only distortion is that introduced by quantization. The results of pruning for each quantization level are shown in Figure 3.3 (a), (b). In these figures, the classification costs are not included in order to see the efficiency of the pre-transform deadzone test compared to the ideal case where all input points inside the deadzone are detected. Thus the results indicate the minimal achieveable complexities or the lower bounds. In Figs. 3.3, we see that the lower bound on complexity reduction using our proposed VCA is close to the ideal case when applied to the column 1-D DCT only.

3.1.2 Optimal Classification

When the complexity of the tests is taken into account the total complexity can be higher than that of the original fixed complexity algorithm, as seen in Figure 3.4. As in Chapter 2, we optimize the classification such that only tests that provide reductions in average complexity are kept (i.e., the savings achieved when operations are skipped outweigh the overhead of testing for all inputs). This optimization is based on training on a set of images and is performed through exhaustive search (since the number of tests involved is small.)

We use "baboon", "boat", "creek" and "lake" as training data to design the VCA at each quantization parameter for "lenna" image. The result of both estimated complexity and CPU clock[1] savings are shown in Figure 3.4. We use the same set of weights for operations as in Table 2.1. It can be seen that when the quantization parameter is small, i.e., in the low MSE region, the complexity is the same as that of the original fixed complexity algorithm. This means that there is no test in the algorithm because in the optimization process it is determined that the tests would not result in any savings. On the other hand, in the high MSE region, given that there will be more zero outputs, there is something to be

[1]The implementation is run on a Sun Ultra-1 running Solaris 2.5.

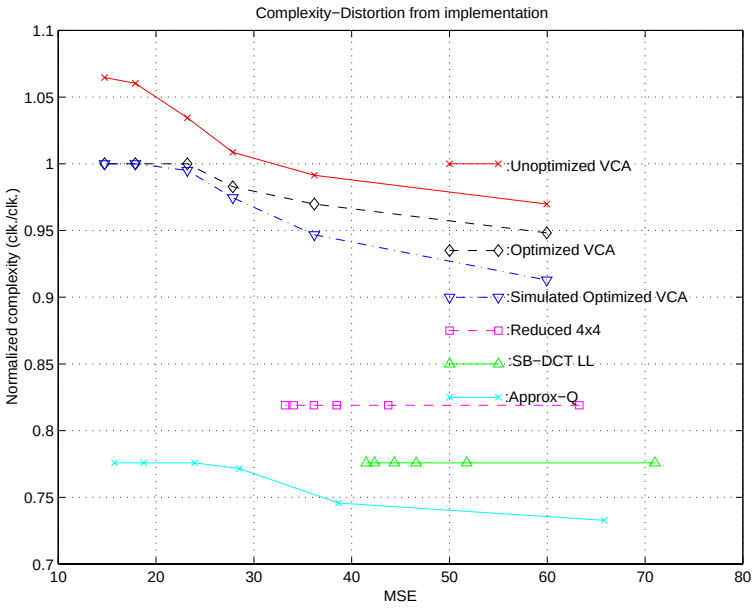


Figure 3.4: Complexity(clock cycle)-distortion comparison with "lenna" JPEG encoding.

gained from the VCA approach as will be seen later in this chapter.

From our results we can see that the gains are modest due in part to the overhead involved in testing. However a major reason for the lack of more significant gains is the fact that we are still computing an exact DCT, when in fact at high MSEs an approximate DCT would be acceptable given that data is going to be coarsely quantized.

3.2 Approximate DCT

The DCT can be approximated using a subband decomposition as in [48] and [47]. This approach exploits the fact that, with appropriate post-processing, the DCT coefficients can be obtained after the subband decomposition, and in typical natural images one can disregard high frequency subband components without greatly affecting the accuracy of the calculated DCT. Therefore, the DCT coefficients can be approximated from post-processing only low frequency subband coefficients. Because a simple subband decomposition can be used (Haar filters for example) the overhead for pre-processing is small. A subband decomposition

hence can be viewed as a pre-processing that reduces the interdependencies among the inputs and gives some clues about what information can be disregarded.

Instead of approximating the whole DCT block with a subset of DCT coefficients, another approach is to estimate all the coefficients but with less accuracy. In this section, we first give a review of the statistical sum of absolute value testing (SSAVT) [64], and analyze the distortion contributed by the approximation. Also, our proposed approximate algorithm is described, and its error analysis is discussed.

3.2.1 SSAVT Review

We first review the SSAVT [64] algorithm. From the Laplacian model for residue pixels presented in Section 1.6, we have that the variance of each pixel is (see (1.5))

$$\sigma^2_{X(u,v)} = \sigma^2 \Gamma^2_N(u,v)$$

where $\Gamma^2_N(u,v)$ is a function of the correlation coefficient of spatial domain pixels. We can find the variance of each DCT coefficient from the scaled version of the spatial-domain variance. In [64], the residue pixels are modeled in spatial domain to be Laplacian distributed with zero mean as in (1.3). This assumption is also applicable to some extent to pixels in INTRA coded frames. Thus, in this work we apply the Laplacian model to INTRA coding. Since the fundamental idea of the SSAVT is to allow looser thresholding by introducing a probabilistic criterion (as opposed to the deterministic criterion in [61]), a model mismatch for I-frames would only result in a small deviation in the probabilistic criterion from the actual one. From this Laplacian assumption, we can estimate σ^2 from the Sum of Absolute Value (SAV) as

$$\sigma \approx \sqrt{2} \cdot SAV/N^2 \tag{3.4}$$

where N^2 is the number of pixels in an NxN block. In the case of a P-frame, the SAV can be obtained as a by-product of the motion estimation, since the Sum of Absolute Difference (SAD) is computed and compared in order to find the best motion vector. However, it is worth mentioning that the SAD corresponding to a macroblock can only be used to obtain a value of σ that will be the same for all

four 8x8 blocks. In an advanced prediction mode (OBMC), the SAD of each block is computed and therefore we can have different σ for each block. Alternatively, it would be possible to explicitly compute the SAD for each of the subblocks so as to have a different σ for each.

The key to reducing the complexity comes from the fact that given the model we can always find the range of values that the DCT coefficient will take with some probability. For example, the probability that a value will fall within $(-3\sigma, 3\sigma)$ is about 99%, and as a result, we can skip the computation of a certain coefficient if the value is 99% sure to be quantized to zero. The testing would then consist of checking if

$$3\sigma_{X(u,v)} < 2QP + DZ, \tag{3.5}$$

where QP and DZ are the quantization parameter and the additional deadzone factor, respectively. That is, the 99% probability interval is completely embedded in the deadzone.

$$\begin{aligned} 3\Gamma_N(u,v)\sigma &< (2QP + DZ) \\ SAV &< (2QP + DZ) \cdot N^2/(3\sqrt{2}\Gamma_N(u,v)) \end{aligned} \tag{3.6}$$

Equation (3.6) is from (3.4). Therefore, the test is equivalent to a threshold testing of the SAV. The confidence level can also be changed depending on the complexity reduction versus accuracy tradeoff. Furthermore, from (1.6) it can be seen that the variances are decreasing from the DC to higher frequency AC coefficients. This implies that if a lower frequency DCT is determined to be zero, so are all higher frequency DCTs. As a result, the classification can be performed by testing the SAV with a set of thresholds which corresponds to classifying the output 8x8 DCT block to *i) all-zero, ii) DC-only, iii) low-4*`x`*4, and iv) all-64*. For each of the test, $\Gamma_8(0,0), \Gamma_8(1,0)$ and $\Gamma_8(4,0)$ are used in (3.6), respectively.

The result of this classification is very promising. Only a slightly degradation in PSNR is observed because the distortion caused by not computing high frequency components is compensated by the corresponding bit savings. We now provide an analysis of the distortion introduced by the SSAVT that was not provided by the original authors [64] and which will help us evaluate the degradation

introduced by the SSAVT and compare it with that of our proposed approximate algorithms.

3.2.2 Error Analysis of SSAVT

For each outcome of the SAV test, a corresponding reduced output DCT is applied. We can then consider the reduced output DCT as an approximation of the exact DCT. For example, the equivalent transform matrix of the low-4x4 DCT is $\begin{bmatrix} \mathbf{I}_4 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{D}_8$ where $\mathbf{I}_4$ is the 4x4 identity matrix. In words, it is $\mathbf{D}_8$ with all the lower 4 rows, and rightmost 4 columns, set to zero. Therefore, different approximate DCTs are used for different values of σ^2. Let $\{Th_0, Th_1, ..., Th_G\}$ be a set of thresholds defined as functions of QP for classifying an input into G classes (e.g., for the SSAVT, G is 4) where Th_0 is 0 and $Th_G = \infty$. *The i-th reduced output DCT is applied if* $Th_{i-1} \leq \hat{\sigma} < Th_i$. where $\hat{\sigma}$ is the approximation of σ computed from the SAV and

$$Th_i = \frac{2QP + DZ}{3 \max\limits_{(u,v)\in B_i} \Gamma_N(u,v)} \tag{3.7}$$

where B_i is a set of output coefficients computed. Therefore, the block distortion of class i input can be expressed as

$$D_{ssavt}(B_i) = \sum_{(u,v)\in B_i} D(u,v) + \sum_{(u,v)\notin B_i} \sigma^2 \Gamma_N^2(u,v) \tag{3.8}$$

where $D(u,v)$ is obtained from (1.10).

The first term is the sum of the distortion of coefficients that are computed, and the second term corresponds to the coefficients that are computed and thus are not coded. Figure 3.8 shows the normalized increase in distortion using the SSAVT at various σ^2 and QP (ratio between the second term and the first term of (3.8)). This result is based on the assumption that the variance of the signal can be perfectly estimated from the SAV.

It can be seen that the increases in distortion have a seesaw shape as a function of the pixel variance. This can be explained as follows. For a given QP, there is a set of thresholds that specifies which reduced DCT will be used within a certain

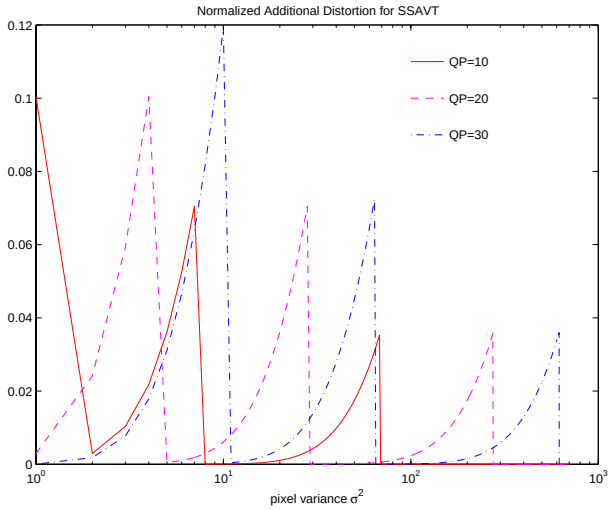


Figure 3.5: Additional distortion (normalized by the original distortion) when using SSAVT at various levels of the pixel variance σ^2.

interval. At the beginning of each new interval, an algorithm which computes more coefficients is selected thus pushing the distortion down, and as the σ^2 grows, the extra distortion starts to become larger again. For the SSAVT algorithm, there are 3 thresholds. After the third threshold has been exceeded (at the right end of each curves in Fig. 3.8), there is no additional distortion since all 64 DCTs are always computed. As an extension to [64], we perform SSAVT for all possible reduced output DCTs, i.e., we also consider *low-2*$\mathtt{x}$*2* output class as well. In other words, $c(A_i^*)$ will have the form shown in (2.7).

The reduction in computation and the increase in distortion from computing only a subset of DCT coefficients has a benefit in that the overall rate required to code the DCT coefficients is reduced. Therefore, the rate and complexity of a class g input can be written as

$$R_{ssavt}(B_i) = \sum_{(u,v)\in B_i} R(u,v) \tag{3.9}$$

$$T_{ssavt}(B_i) = c(A_i^*) + i \cdot c(Av), \tag{3.10}$$

when $Th_i \leq \hat{\sigma} < Th_i + 1$.a $c(Av)$ is the cost for SAV threshold testing.

In order to find $\bar{D}_{ssavt}$, $\bar{R}_{ssavt}$, and $\bar{T}_{ssavt}$, the average values of distortion, rate and complexity, the probability of the outcomes of the SAV test has to be taken

into account. For a given model of the source, the variance σ^2 cannot be perfectly estimated by the SAV. The mean absolute sum $m = SAV/N^2$ can be viewed as a random variable representing the time average of the source. It can be derived from the model of the source (first order Markov source) that the mean of m is

$$E\{m\} = \sigma/\sqrt{2}$$

and the variance is

$$\sigma_m^2 = E\{(m - E\{m\})^2\} = \sigma^2(\frac{1}{N^2} + \frac{2((N^2-1)\rho - N^2\rho^2 - \rho^{N^2+1})}{N^4(1-\rho)^2}),$$

where, again, ρ is the correlation coefficient of the source.

Let us denote the approximation of σ^2 based on m and $\hat{\sigma}^2 = 2m^2$. This $\hat{\sigma}^2$ is tested with a set of thresholds $\{Th_0, Th_1, ..., Th_G\}$ where $G = \log_2 N + 1$ for all possible reduced output DCT. The probability of each outcome of the threshold test can be written as

$$\begin{aligned} P_i &= \int_{Th_i}^{Th_{i+1}} f_{2m^2}(x)dx \\ &= \int_{\sqrt{Th_i/2}}^{\sqrt{Th_{i+1}/2}} f_m(x)dx \end{aligned} \tag{3.11}$$

where $f_m(x)$ is the p.d.f. of m. From [88] (see also Chapter 2), the difference between the partial mean of absolute pixel difference (PMAD) and the actual mean of absolute pixel difference (MAD) is modeled as a Laplacian distribution. Therefore, in this thesis, we assume that m has a truncated Laplacian distribution with mean and variance as above, i.e.,

$$f_m(x) = \begin{cases} \frac{\lambda_s}{2L} e^{-\lambda|x - E\{m\}|} & \text{for} \quad x > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $L = \int_0^\infty \lambda_m e^{-\lambda|x-E\{m\}|}$ is a constantnormalization factor that ensures that a valid p.d.f. is obtained and $\lambda_m = \sigma_m/\sqrt{2}$. Thus, the complexity, rate and

distortion can be expressed in a similar fashion to (2.9) as

$$\bar{T}_{SSAVT} = \sum_{i=1}^{\log_2 N+1} P_i \cdot T_{dyadic}(B_i) \tag{3.12}$$

$$\bar{R}_{SSAVT} = \sum_{i=1}^{\log_2 N+1} P_i \cdot R_{dyadic}(B_i) \tag{3.13}$$

$$\bar{D}_{SSAVT} = \sum_{i=1}^{\log_2 N+1} P_i \cdot D_{dyadic}(B_i) \tag{3.14}$$

Note that all the above quantities are functions of QP, σ^2 and N. The R-C-D characteristic of the SSAVT is shown in Figure 3.6.

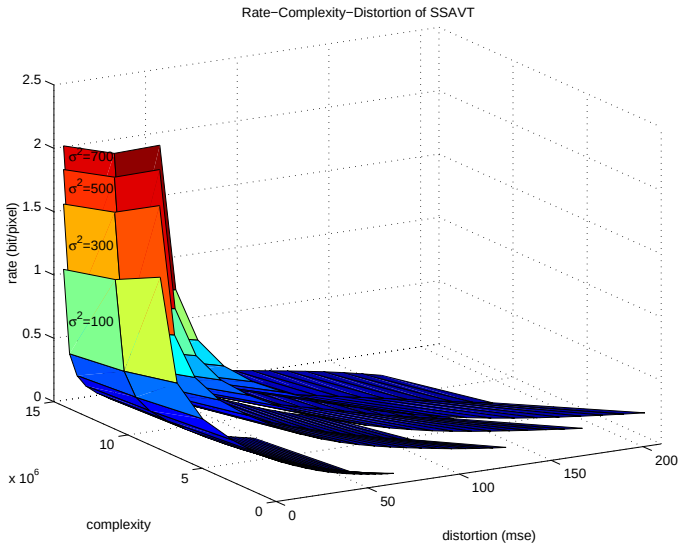


Figure 3.6: Rate-complexity-distortion functions of SSAVT. The R-C-D results of SSAVT and ideal dyadic test are very close and hence cannot be visually distinguished in the figure.

Figure 3.7 shows the complexity comparison between the SSAVT and the 2-D dyadic classification. Since the DCT and IDCT operations are transpositions of each other, the 2-D dyadic classification for IDCT in Chapter 2 can be considered as the ideal case for SSAVT where all instances of reduced output DCT are 100% correctly detected. The addition and multiplication results follow a similar trend as the total complexity curve, and are thus omitted here. The R-D function of the SSAVT is already shown in Figure 1.8. It can be seen that the R-D performance

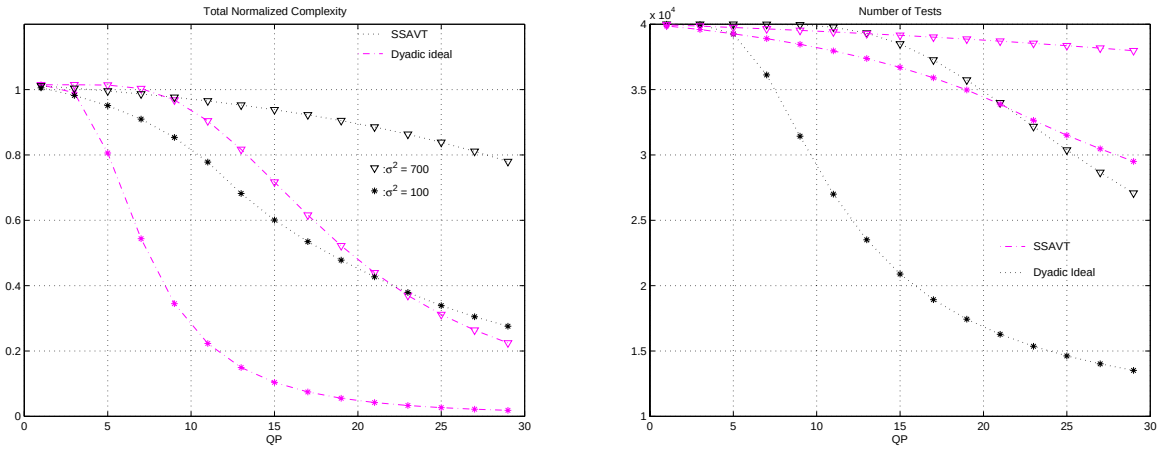
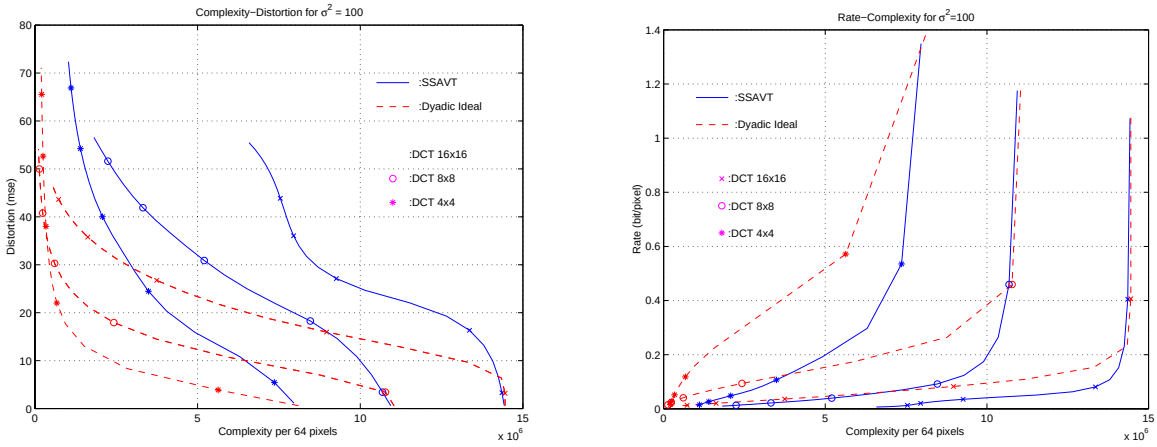


Figure 3.7: Total complexity and the number of tests needed for SSAVT algorithms and 2-D dyadic at $\sigma^2 = 100$ and 700. 2-D dyadic is considered as an ideal case for the SSAVT where reduced classes are 100% detected.

does not degrade much when complexity is reduced. While distortion increases, fewer bits are also needed for the reduced output DCT. The C-D and R-D functions of the SSAVT and the ideal dyadic test are shown in Figures 3.8 (a) and (b), respectively.

In Figure 3.7, the complexity performance as a function of QP of the SSAVT is significantly inferior to that of the ideal dyadic test. It implies that the ability to capture input classes with zero outputs is not very good. However, when considering the resulting rate and distortion of the SSAVT, the overall R-C-D (Fig. 3.6) is only slightly degraded from the ideal dyadic test. The reason to support this claim can be found by considering Fig. 3.8. One can see that in Fig. 3.8(a) the C-D performance of the SSAVT is worse than the ideal at the same complexity, the SSAVT results in higher distortion. This is because it incorrectly classifies nonzero coefficients to be zero coefficients. On the other hand, when it classifies zero coefficients to be nonzero, unnecessary computations are spent on those zero coefficients. However, in Fig. 3.8(b) the SSAVT R-C performance is better than the ideal dyadic test at low rate, because the bits saved from incorrectly classifying nonzero coefficients to zero. As a result, the overall R-C-D performance degradation from the ideal case is mitigated by this R-D tradeoff.



(a) (b)

Figure 3.8: (a) Complexity-distortion and (b) Rate-complexity curves for different algorithms, i.e., SSAVT ('solid') and 2-D dyadic ('dashed'), for DCT size 16x16 ('x'), 8x8 ('o') and 4x4 ('*') at $\sigma^2 = 100$. 2-D dyadic is considered as an ideal case for the SSAVT where reduced classes are 100% detected.
b

SSAVT is an example of the superior performance of approximation approaches as compared to exact VCA approaches. However, in the next section, as an alternative to the SSAVT in which only a subset of the outputs is computed, we propose another class of approximate algorithms in which all the outputs are computed at an accuracy lower than the minimum requirement of the standard DCT.

3.2.3 APPROX-Q DCT

Here we introduce an approximate algorithm (see also [66]) where in the DCT coefficient computation we replace the multiplications with additions and binary shifts in a manner that is quantization dependent. For large QP, a coarse approximate DCT is used because even if the error in coefficient computation is large, the large quantization step means that their quantized representation will introduce even more error. Similarly, finer approximate DCT is used when QP is small. Our proposed approximate DCT is shown in figure 3.9. This figure is derived from the fast DCT in 1.3 by simplifying the operations. The matrix $\mathbf{p_o}$ ranges from the

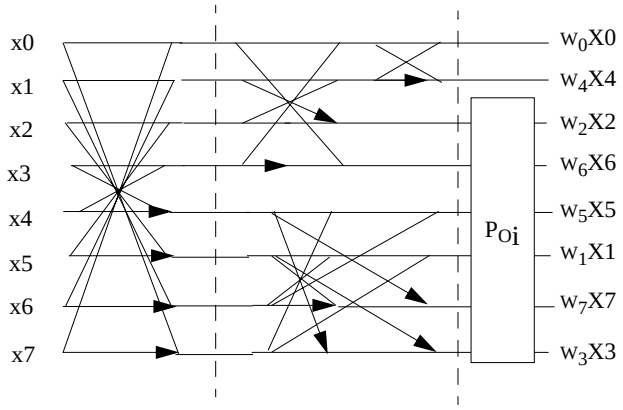


Figure 3.9: The approximate DCT algorithm.

identity matrix, $\mathbf{p}_{o1} = \begin{bmatrix} \mathbf{q}_1 & 0 \\ 0 & I \end{bmatrix}$, to more complex matrix $\mathbf{p}_{o2}$,$\mathbf{p}_{o3}$,$\mathbf{p}_{o4}$,and $\mathbf{p}_{o5}$ shown below.

$$\mathbf{p}_{o2} = \begin{bmatrix} \mathbf{q}_1 & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t \\ \bar{\mathbf{0}} & 1 & 0 & 1/8 & -1/8 \\ \bar{\mathbf{0}} & -1/8 & 1 & 1/8 & 0 \\ \bar{\mathbf{0}} & 0 & -1/8 & 1 & 1/8 \\ \bar{\mathbf{0}} & 1/8 & 1/8 & 0 & 1 \end{bmatrix}$$

$$\mathbf{p}_{o3} = \begin{bmatrix} \mathbf{q}_2 & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t \\ \bar{\mathbf{0}} & 1 & 1/8 & 1/8 & -1/8 \\ \bar{\mathbf{0}} & -1/8 & 1 & 1/8 & -1/8 \\ \bar{\mathbf{0}} & -1/8 & -1/8 & 1 & 1/8 \\ \bar{\mathbf{0}} & 1/8 & 1/8 & -1/8 & 1 \end{bmatrix}$$

$$\mathbf{p}_{o4} = \begin{bmatrix} \mathbf{q}_3 & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t \\ \bar{\mathbf{0}} & 1 & 1/8 & 1/8 & -1/4 \\ \bar{\mathbf{0}} & -1/8 & 1 & 1/4 & -1/8 \\ \bar{\mathbf{0}} & -1/8 & -1/4 & 1 & 1/8 \\ \bar{\mathbf{0}} & 1/4 & 1/8 & -1/8 & 1 \end{bmatrix}$$

Table 3.1: Number of operations required for proposed approximate algorithms where Alg. No. i corresponds to using the transform matrix P_{oi}.

Alg. No.	Additions	Multiplication	Binary Shifts
1	24	0	2
2	33	0	7
3	38	0	8
4	42	0	12
Vetterli's [29]	29	13	0

$$\mathbf{p}_{o5} = \begin{bmatrix} \mathbf{q}_3 & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t & \bar{\mathbf{0}}^t \\ \bar{\mathbf{0}} & 1 & 1/8 & 1/8 & -3/16 \\ \bar{\mathbf{0}} & -1/8 & 1 & 3/16 & -1/8 \\ \bar{\mathbf{0}} & -1/8 & -3/16 & 1 & 1/8 \\ \bar{\mathbf{0}} & 3/16 & 1/8 & -1/8 & 1 \end{bmatrix}$$

where

$$\mathbf{q}_1 = \begin{bmatrix} 1 & -1/2 \\ 1/2 & 1 \end{bmatrix}$$

$$\mathbf{q}_2 = \begin{bmatrix} 1 & -3/8 \\ 1/2 & 1 \end{bmatrix}$$

$$\mathbf{q}_3 = \begin{bmatrix} 1 & -3/8 \\ 3/8 & 1 \end{bmatrix}$$

$$\bar{\mathbf{0}} = [\ 0 \quad 0\ ]$$

The number of operations required for these approximate DCTs are shown in Table 3.1.

Let an approximate DCT matrix using one of the above approximations be denoted by ($\mathbf{D_{approx}}$). After these approximations, if one wants to obtain the exact DCT coefficient values, the post-transform matrix would be $\mathbf{D_8} \cdot \mathbf{D_{approx}}^{-1}$. This post-transform matrix gets closer to the identity matrix (up to scaling factors) as the approximation gets more accurate. The scaling factors needed at the output can be found to be $\mathtt{diag}(\mathbf{D_8} \cdot \mathbf{D_{approx}}^{-1})$, and can be absorbed in quantization,

i.e., in our approximate DCT, the post-transform is approximated by setting off-diagonal elements to zero leaving only the diagonal elements as scaling factors. Since this algorithm is lossy, the approximation introduces more distortion to the reconstructed images. The rate-distortion curves of these approximate DCTs using JPEG coding are shown in Figure 3.10.

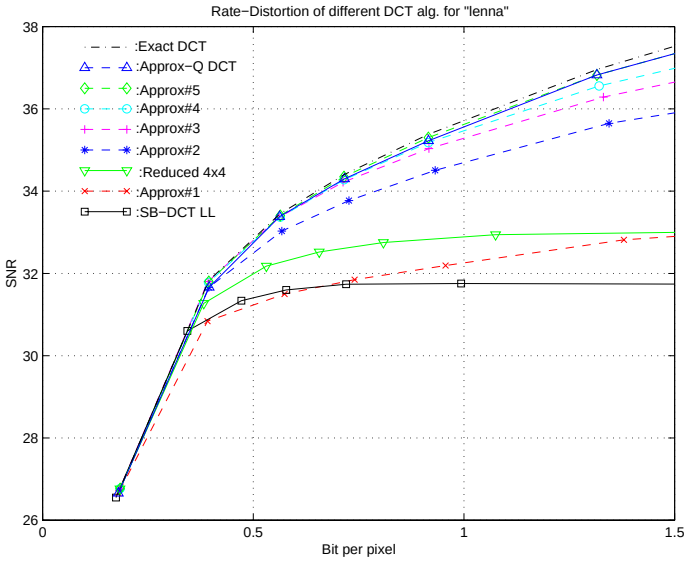


Figure 3.10: Rate-Distortion curve of 512x512 lenna image JPEG coding using various DCT algorithms. Note that at high bit rate coarser approximate algorithm performances deviate from the exact DCT performance dramatically. The quantization dependent approximation can maintain the degradation level over wider range of bit rate.

In this experiment, we encode the "lenna" image with JPEG compression using the example quantization matrix [1]. It can be seen that, as expected, in the high distortion region the increase in distortion introduced by the approximate DCTs does not affect performance because it is less than the distortion due to quantization. Therefore, we develop an algorithm which is quantization parameter dependent (shown as a solid line, Approx-Q, in Figure 3.10) in which the selection of the approximate algorithm is made at the beginning of the encoding process depending on the quantization parameter. It uses a coarser DCT approximation algorithm for low quality coding and finer DCT approximation for high quality coding for small degradation. The degradation of the decoded image introduced by the approximate DCT is 0.12 dB at 0.18 bpp, 0.15 dB at 0.91 bpp and 0.64 dB

at 3.17 bpp. From Figure 3.10, it is obvious that even with the Approx#2, the R-D curve is better than other reduced complexity approximate algorithms such as pruned DCT (computing only low frequency 4x4 DCT) and subband DCT (using low-low subband [48]). This is because we do not lose high frequency information which is present at high rates.

3.2.4 APPROX-Q Error Analysis

We now analyze the additional distortion introduced by the approximation of the DCT and, based on that, we show how to automatically select which (coarser or finer) approximate DCT should be used for a given QP and σ^2 . Our goal is selecting the algorithm to ensure that the resulting overall distortion does not exceed a certain level.

Let us denote the transform matrix of the i-th approximate DCT by $\hat{\mathbf{D}}_i$, the input spatial domain block $\mathbf{x}$, and the DCT computed by this reduced matrix by $\hat{\mathbf{X}}_i$. Therefore, the difference between the exact and approximate computation can be written as

$$\begin{aligned} \mathbf{e} &= \mathbf{X} - \hat{\mathbf{X}}_\mathbf{i} \\ &= \mathbf{D}\mathbf{x}\mathbf{D}^\mathbf{t} - \hat{\mathbf{D}}_\mathbf{i}\mathbf{x}\hat{\mathbf{D}}_\mathbf{i}^\mathbf{t} \\ &= (\mathbf{D} - \hat{\mathbf{D}}_\mathbf{i})\mathbf{x}(\mathbf{D} - \hat{\mathbf{D}}_\mathbf{i})^\mathbf{t} \\ &= \hat{\mathbf{E}}_\mathbf{i}\mathbf{x}\hat{\mathbf{E}}_\mathbf{i}^\mathbf{t} \end{aligned} \tag{3.15}$$

where $\hat{\mathbf{E}}_\mathbf{i} = \mathbf{D} - \hat{\mathbf{D}}_\mathbf{i}$ is the approximation error matrix. With a similar analysis to that in Section 1.6, the variance of the error in DCT domain can be derived as

$$\sigma_e^2(u,v) = \sigma^2[\hat{\mathbf{E}}_\mathbf{i}\mathbf{R}\hat{\mathbf{E}}_\mathbf{i}^\mathbf{t}]_{(u,u)}[\hat{\mathbf{E}}_\mathbf{i}\mathbf{R}\hat{\mathbf{E}}_\mathbf{i}^\mathbf{t}]_{(v,v)} = \sigma^2\phi_i^2(u,v), \tag{3.16}$$

where $\mathbf{R}$ is the correlation matrix of the input vector and $\phi_i^2(u,v)$ is a scaling constant that is a function of the approximation matrix. In general $\phi_i^2(u,v)$ should be much smaller than $\Gamma_N^2(u,v)$, since their ratio is the relative additional distortion introduced by the approximation. The total block approximate error can be

written as $D_e(q)$,

$$D_e(q) = \sigma^2 \sum_{(u,v)} \phi_q^2(u,v) \tag{3.17}$$

We can model this approximation error as an additive white Gaussian noise (AWGN) that is to be added to another AWGN error due to quantization, i.e., from the approximate quantized DCT, $\tilde{X}$ can be written as

$$\tilde{\mathbf{X}} = \mathbf{X} + \mathbf{n_q} + \mathbf{n_a}$$

where $\mathbf{n_q}$ is a Gaussian r.v. modeling the error due to quantization and $\mathbf{n_a}$ represents another Gaussian r.v. representing the approximation error. It is reasonable to assume that $\mathbf{n_q}$ and $\mathbf{n_a}$ are independent. Therefore, the distortion can be expressed in terms of the summation of the original distortion and the distortion from approximation.

$$D_{APPROX}(QP, \sigma^2, N) = \sum_{(u,v)} D(u,v) + D_e(q) \tag{3.18}$$

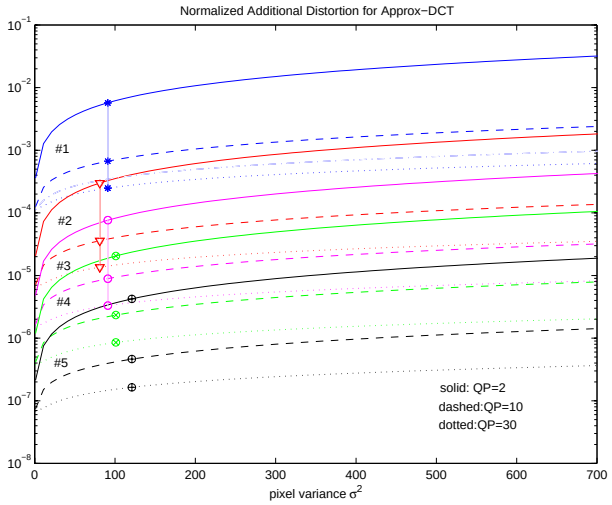


Figure 3.11: Additional distortion (normalized by original distortion) using approximate DCT algorithms #1 ('$*$'), #2 ('$\bigtriangledown$'), #3 ('o'), #4 ('$\otimes$'), and #5 ('$\oplus$') at various pixel variance σ^2.

Figure 3.11 shows the approximation error results of the 5 approximate DCT algorithms normalized by the distortion due to quantization. It can be seen that

not only the approximation error depends on the QP, it also increases as the pixel variance σ^2 grows, i.e., for a fixed pixel variance the additional error ratio increases as quantization stepsize decreases. Furthermore, for a fixed QP, the additional error ratio increases with pixel variance. However, for a given approximate algorithm, QP still plays a bigger role in the resulting error. Therefore, we can systematically select which approximate algorithm to be used for a given QP and σ^2, so as to ensure that the approximation error is below a certain limit, i.e.,

$$\text{the } q\text{-th algorithm is selected if } \quad D_e(q) < \eta \sum_{(u,v)} D(QP, \sigma^2, N)$$

where η is the level of desired additional error . For example, when coding a frame with fixed QP for all blocks, low variance blocks (associated with low activity) require less accurate DCT approximation whereas high variance blocks must use finer approximation in order to maintain the same level of additional error throughout the entire frame.

3.3 Results and Hybrid Algorithms

In this section we discuss a series of hybrid algorithms which combine the strengths of each the previously proposed algorithms. Experimental results are shown based on the baseline H.263 TMN's codec [58] used to encode 249 frames of the "Foreman" sequence at different target bit rates. The Visual C++ compiler with optimization option is used. The CPU clock cycles spent in both the DCT *and* quantization are measured because for the VCA exact DCT, not only the DCT can be prunned but also the quantization for those zero coefficients can be omitted. For Approx-Q, since the approximated coefficients have to be scaled, the extra scaling operation has to be done in the quantization process thus affecting the overall complexity. However, it will be seen later that the speedup from the DCT part is much more significant than the slowdown in quantization for the Approx-Q case. As can be seen in Fig. 3.12, the speedup of the Approx-Q is much higher than the VCA exact algorithm with only 0.01-0.06 dB degradation.

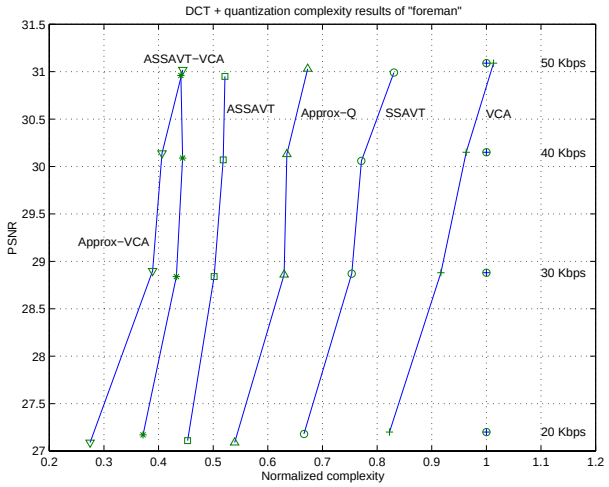


Figure 3.12: The complexity (CPU clock cycle) of the DCT + Quant. normalized by the original algorithm in TMN at various PSNRs (different target bit rates) of original DCT ('+'), SSAVT ('o'), Approx-Q ('△'), ASSAVT ('□'), Approx-VCA ('▽'), and ASSAVT-VCA ('*').

3.3.1 Approx-SSAVT (ASSAVT)

We first combine the SSAVT and the Approx-Q DCT in such a way that the SAV is first tested and the input is thus classified in the same manner as the SSAVT. Then the reduced DCT is approximated using the approximate algorithm. That is, we use approximate algorithms also for the reduced dimension DCTs used in SSAVT. The total block distortion for this combined algorithm, which we call ASSAVT, can be expressed as

$$D_{ASSAVT}(QP, \sigma^2, N) = \sum_{(u,v)\in B_i} [D(u,v) + \sigma^2 \phi_i^2(u,v)] + \sum_{(u,v)\notin B_i} \sigma^2 \Gamma_N^2(u,v) \quad (3.19)$$

The first summation on the right side is the distortion plus approximation error of the computed coefficients. The second term is for uncomputed coefficients. Figure 3.13 shows the result of ASSAVT with the target approximation error at $2\text{x}10^{-4}$ of the original distortion. For simplicity, this experiment assumes that the variance of the input is perfectly estimated by the SAV. Therefore, the class that the input belongs to is deterministic. Therefore, the estimated SSAVT distortion

is known with certainty. In practice, the estimation of the pixel variance is random, thus resulting in a probabilistic estimation of the SSAVT distortion. The actual deviation from the SSAVT distortion must be averaged over the support of the estimated pixel variance. In Figure 3.13, it can be seen that the additional distortion follows the SSAVT seesaw-shaped results except that it can now be kept under the specified error bound. Thus, when the error from SSAVT becomes too small, i.e., sufficient coefficients are computed for the given σ, then we can observe the error due to the approximation. Except in those cases, the additional error due to approximation is negligible. Moreover, these situations occur when error is small, anyway, and therefore adding the approximate algorithm to SSAVT is always beneficial.

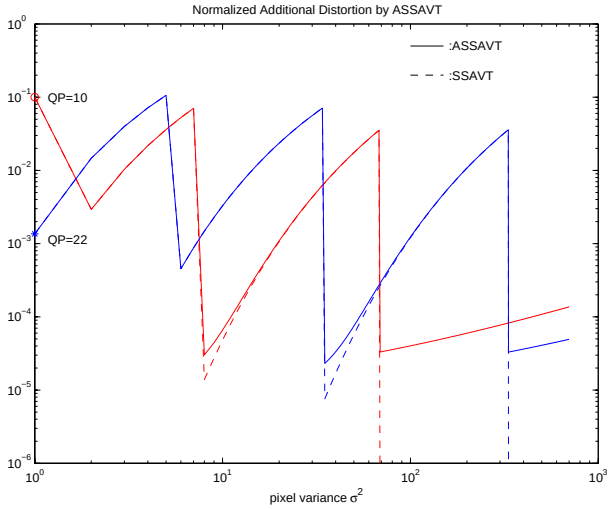


Figure 3.13: Additional distortion (normalized by original distortion) using the ASSAVT with 2x10^{-}4 target deviation from the original distortion ('- -'), SSAVT ('–'), at QP = 10 ('o') and 22 ('*'), respectively.

We apply the ASSAVT to the DCT subroutine of the TMN [58] H.263 codec with some modifications. First, the reduced DCT, i.e., 2x2-DCT and 4x4-DCT, use Approx#4 and Approx#5, respectively. This follows the result in Fig 3.13 that as the variance σ^2 gets larger, the error becomes larger too. This also results in larger size DCT after SSAVT classification. Therefore, in order to maintain a small error, finer approximation algorithm has to be used. For DC-only class, the DC coefficient can be computed precisely using only additions and a binary shift. For

the full DCT class, we use the APPROX-Q algorithm presented earlier in which the selection of the approximate algorithm is quantization parameter dependent. The quantization can also be done faster using the result of SSAVT classification, i.e., the high frequency DCT components do not need to be quantized if they are not computed. Note that the approximate algorithm assignment can be changed according to the performance degradation requirement. In our experiments, we simply select a scheme that gives reasonable complexity reduction and degradation tradeoffs.

The results of the proposed ASSAVT as compared to SSAVT are shown in Figure 3.12. The CPU clock cycle of both DCT and quantization are measured as mentioned above. It can be seen that the ASSAVT can obtain 47-55 % complexity reduction, as compared to 17-33% reduction by the SSAVT, while the coding performance is degraded only by 0.04-0.14 dB as compared to 0.02-0.08 dB by SSAVT with respect to the exact algorithm.

3.3.2 Approx-VCA DCT

It is worthwhile pointing out here that since the structure of the approximate DCT algorithm is similar to the fast algorithm in Figure 1.3 we can apply the classification structure and the optimization technique to obtain the optimal VCA as in Section 3.1 for the Approx-Q DCT algorithm presented in Section 3.2. The algorithm is now data-dependent and will be called Approx-VCA DCT. Experimental results are also shown in Fig. 3.12. In this experiment, we also take advantage of the zero information from the VCA in the quantization process as mentioned earlier, i.e., if the resulting coefficients from VCA are zero, no quantization operation is needed. In practice, one could combine VCA DCT and quantization into one subroutine, but for simplicity we still separate them and simply introduce a zero test before quantizing each coefficient.

3.3.3 ASSAVT-VCA DCT

Finally, we can combine SSAVT, Approx-Q and VCA exact DCT into one algorithm. Based on the ASSAVT in Section 3.3.1, we modify the approximate

algorithm in the case of all-8x8 class to have pre-transform deadzone test capability as in Approx-VCA. The result is also shown in Fig. 3.12. It can be seen that the C-D result is improved from the ASSAVT by about 8-10%. However, when compared to the Approx-VCA algorithm, the ASSAVT-VCA improves just a very small fraction at low rates and shows almost no improvement at high rate. This is because the SSAVT classification efficiency drops at high rates.

3.4 Summary and Conclusions

We have proposed two techniques for fast VCA DCT computation. The first one computes exact DCT coefficients in a VCA manner by checking on the pre-transform deadzone region. This algorithm is hierarchical in the sense that the classification can be broken down to smaller group of coefficients. However, the drawback of this algorithm is the efficiency of the pre-transform deadzone drops rapidly as the dimension of the transform grow. Thus we apply it only to the second 1-D DCT. At high rate, this algorithm shows almost no gain. The second technique was then considered. Unlike the first approach, the resulting DCT output is not exact. We gave a review of the reduced DCT using SSAVT classification and analyzed the performance. Then we proposed an approximate algorithm that computes all the DCT coefficients in a block with less accuracy than in the exact case. The complexity reduction comes from using cheaper arithmetic operation (no multiplications) with the sacrifice in slightly poorer coding performance. Finally, we showed some experimental results and suggested several hybrid algorithms that combine all the above mentioned algorithms.

It can be seen that the result of approximate techniques are much more promising than the exact algorithms in terms of complexity at the cost of only a slight degradation in picture quality. Since the DCT cannot be computed exactly using any fixed precision computer, the error is already introduced. Also, the quantization effect masks out the computational error in the case of low rate. This results in a slight decrease in the overall performance. Another conclusion is that using the statistical model, such as SSAVT, allows more classification gain with a small coding performance degradation.

Chapter 4

Motion Estimation: Probabilistic Fast Matching

In this chapter, we propose a matching algorithm for motion estimation that uses probabilistic stopping criterion based on the partial distance search [79]. It can be categorized as a computationally scalable Fast Matching (FM) algorithm. The basic idea is to stop the computation of the matching metric when the partial metric indicates that the total metric is *likely* to exceed that of the best candidate so far. This allows us to save in computation when "bad" candidate vectors are being tested, since these can be discarded before all pixels in the macroblock have been used to compute the metric. To achieve this goal we formalize a hypothesis testing framework where the decision to stop the metric computation is done based on probabilistic models of the distribution of the actual metric based on the calculated partial metric. We select a given probability of error (i.e., missing a "good" vector) based on these models and by varying this probability we can achieve a computationally scalable calculation. From the nature of the test, we will refer to the original partial distance search as determinictic testing fast matching (DTFM) as opposed to the hypothesis testing fast matching (HTFM) ([88],[101]) for the proposed algorithm.

This chapter is organized as follows. In Section 4.1, the original partial distance algorithm is reformalized and a new macroblock partitioning is introduced. In Section 4.2, we provide a formal description of the hypothesis testing framework

that forms the basis of our algorithm. We model our variables of interest and assume that their second order statistics are known. With a given error probability or probability of false alarm, we design the hypothesis testing such that the error probability meets the requirements. We also show simulation results on some test sequences where we assume all necessary statistics of a particular sequence are known to the encoder. In Section 4.3, since each sequence has different characteristics which are unknown to the encoder before the encoding process takes place, we discuss how to obtain the statistics adaptively during the encoding. As an extension to [86], we propose an alternative computationally scalable Fast Search (FS) in Section 4.4.1 which can be combined with our HTFM. Finally, in Section 4.4, we show the result of our proposed HTFM with adaptive parameter estimation as compared to DTFM using ES, 2-D Log search [68] and ST1 search [4]. Finally, concluding remarks are given in Section 4.6.

4.1 Partial Distance Fast Matching

In this work, we use SAD as the matching criterion[1] (refer to Table 1.3 for necessary notations). In all our experiments SAD calculations are based on at most 128 pixels out of the 256 pixels of a macroblock. As in [67], this subset $\beta_q \subset B$ is obtained by subsampling using a quincunx grid (see Fig. 4.1.) As shown by Fig. 4.2, this particular subsampling tends to provide sufficient accuracy for the SAD calculation (see [67, 4]), i.e., the overall MSE does not increase significantly if we use β_q instead of B.

Our work is based on splitting each set β_q into b subsets of pixels, y_i, for $i \in \{1, 2, ..., b\}$, such that $\bigcup_{i=1}^{b} y_i = \beta_q$ and $y_i \cap y_j = \phi$ for $i \neq j$ (see for example Fig. 4.1). Let us define $x_i = \bigcup_{j=1}^{i} y_j$. During the SAD calculation of $\vec{mv}_j$, we compare the partial calculation of the SAD, $SAD(\vec{mv}_j, x_i)$ with the best SAD obtained out of all previously tested candidates, $SAD_{bsf}(\gamma_{j-1}, \beta_q)$. If the partial SAD is greater than SAD_{bsf}, we can terminate the computation and continue to

[1]Note, however, that our approach could be easy generalized to other additive distance metrics such as MSE (see for example our work in applying this approach to searching of a Vector Quantizer codebook in [101]).

the next vector. Otherwise, we compute $SAD(\vec{mv}_j, x_{i+1})$ for the next stage and perform the test again. If no early termination occurs, the process repeats until the final stage, b, is reached.

There are many possible ways to partition β_q into subsets y_i. In this work, we propose two methods, which both have $|y_i| = 8$, $\forall i$, and thus result in 16 stages of testing: these are the uniform partition (UNI)[2] and the row-by-row partition (ROW) shown in Fig.4.1(a) and 4.1(b), respectively. In Fig.4.1(a), the partition is designed such that the correlation between $SAD(\vec{mv}, \beta_q)$ and $SAD(\vec{mv}, x_i)$ is maximum, given that the pixels are uniformly spaced. This results in fewer pixel comparisons on the average, since early termination is more likely. However, for a hardware implementation, UNI may not be desirable as it results in more irregular memory access patterns. Conversely, ROW (see Fig.4.1 (b)) provides a more regular memory access that could simplify the implementation, although we can expect ROW to be worse than UNI in terms of producing a reliable estimate of the total SAD.

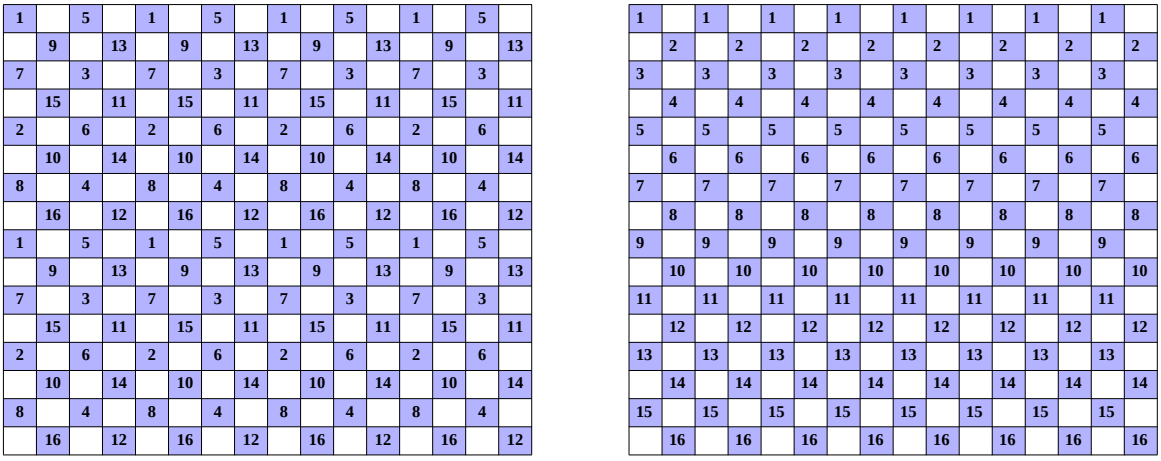
To simplify the notation, when there is no ambiguity we omit to write the terms $\vec{mv}_j$, γ_{j-1} and B. Also we use $PSAD_i$ to represent the partial SAD at stage i, i.e., $SAD(\vec{mv}, x_i)$ for $i = 0, ..., b-1$. Note that the partial SAD can be computed recursively as

$$PSAD_{i+1} = PSAD_i + SAD(\vec{mv}, y_i) \tag{4.1}$$

where $PSAD_0 = 0$ and $PSAD_b = SAD$.

It is clear from (4.1) that $PSAD_i \leq SAD$, for $\forall i$. Therefore, if $PSAD_i$ is greater than the SAD_{bsf}, there is no need to complete the SAD computation and we can move on to evaluate the next vector. Otherwise, we compute $PSAD_{i+1}$ and perform the test again. As a result the MV^* obtained by the partial distance method is obviously the same as that obtained by computing directly the full metric. Thus we call this technique a *deterministic testing fast matching (DTFM)*,

[2]During the reviewing process of our paper [89], we became aware of the work by Cheng and Sun [102], which independently proposed using a uniform partition (dithering pattern pixel decimation). It is important to note, however, that neither this or other FM works use a variable number of pixels for matching.



(a) (b)

Figure 4.1: Subset partitionings for 128 pixels subsampled using a quincunx grid into 16 subsets for partial SAD computation. Only highlighted pixels are used to compute SAD. Two types of subsets are used (a) uniform subsampling (UNI) and (b) row-by-row subsampling (ROW). Partial distance tests at the i-th stage are performed after the metric has been computed on the pixels labeled with i (corresponding to y_i).

as it deterministically provides the optimal solution. Note that in this work we call "optimal" the motion vector that is obtained with SAD computed from β_q (i.e., 128 pixels). Therefore a solution based on $x_i \subset \beta_q$ will tend to be "sub-optimal" since we cannot guarantee that it will produce the same motion vector selection obtained using β_q. The DTFM approach can be summarized as follows

Algorithm 2 (DTFM)

Step 1: *At the beginning of motion search for a particular block, compute the SAD of the first candidate MV, assign it to* SAD_{bsf} *and set* MV_{bsf} *accordingly.*

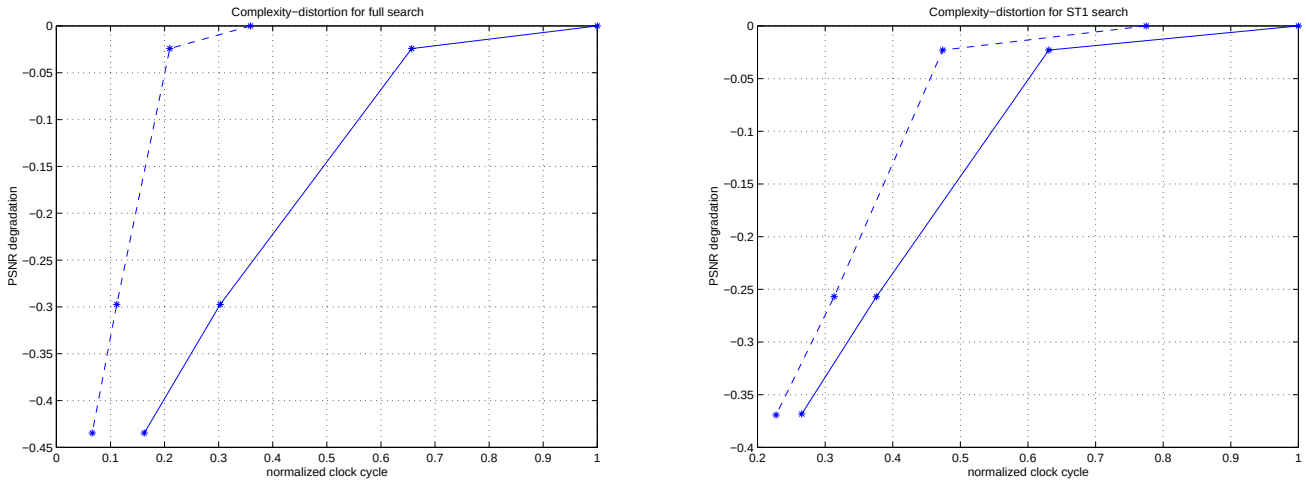
Step 2: *Every time a new* $\vec{mv}$ *is considered, as dictated by the FS strategy, set SAD to zero. Set* $i = 0$. *If there is no next unevaluated vector,* $MV^* = MV_{bsf}$.

Step 3: *Compute* $PSAD_i$

Step 4: *If* $i < b$, *go to step 5, else let* $SAD = PSAD_b$ *and go to step 6.*

Step 5: *If* $PSAD_i \geq SAD_{bsf}$, *we eliminate the current candidate and go to step 2. Otherwise, let* $i = i + 1$ *and repeat step 3.*

Step 6: *If* $SAD < SAD_{bsf}$, $SAD_{bsf} = SAD$ *and* $MV_{bsf} = \vec{mv}$. *Go to step 2.*



(a) (b)

Figure 4.2: Complexity-Distortion of reduced set SAD computation with ROW DTFM ('dotted') and without DTFM ('solid') using (a) ES and (b) ST1 search, averaged over 5 test sequences. Points in each curve from right to left correspond to $|\beta| = 256$, 128, 64 and 32, respectively. Note that there is a minimal difference between computing the SAD based on 256 and 128 pixels. For this reason in all the remaining experiments in this work we use at most 128 pixels for the SAD computation.

The partial distance technique we have just described is well-known and is implemented in many actual software implementations, where ROW subsampling is typically used (e.g. [58],[103]). The complexity savings of this technique come from the possibility of early termination in Step 5. The amount of complexity reduction varies depending on the nature of the sequence. Also, since we can use DTFM with any FS algorithm, the efficiency of the FS algorithm will affect the savings stemming from DTFM. For example, for efficient FS algorithms the tested MVs are likely to have small SAD and their SAD values will tend to be fairly similar. Therefore there is less chance to terminate the matching computation at an early stage, and the benefits of DTFM will be reduced. In general, the complexity reduction contributed by DTFM can be significant, e.g., about 3-5 times speedup in the ES case. In Fig.4.2, complexity-distortion (C-D) results with and without DTFM are compared. The C-D curves are obtained by changing the set β. One can observe that the relative gain in using DTFM is lower when a fast search algorithm is used (see also Table 1.1).

4.2 Hypothesis Testing Framework

In this section, we propose an algorithm, *hypothesis testing fast matching* (HTFM), that enables additional complexity savings as compared to DTFM by allowing an early termination of the SAD calculation based on the *likelihood* that the SAD will be greater than SAD_{bsf}, given the current $PSAD_i$. This complexity reduction over DTFM comes with the cost of potentially not finding the best motion vector for some blocks, which leads to an increase in the energy of the motion compensated predicted frame.

In our formulation, we will use the mean absolute difference (MAD) defined as $MAD = SAD/|B|$, where $|B|$ is the number of pixels in set B. Similarly, we write the "best-found-so-far" MAD as $MAD_{bsf} = SAD_{bsf}/|B|$ and the partial MAD as $PMAD_i = PSAD_i/|x_i|$. It is worth noting that *SAD is always greater than or equal to $PSAD_i$ but MAD can be either greater or smaller than $PMAD_i$.*

Our proposed approach comes from the observation that the PMAD becomes increasingly correlated with the MAD as the partial metric computation proceeds to more stages, i.e., more and more pixels are used. For example, consider Fig.4.3(a) where scatter plots of MAD vs. $PMAD_i$ are shown. It can be seen that there is a high correlation and $PMAD_i$ is an increasingly good estimate of MAD as i grows. The histograms of the difference between MAD and the PMADs are also shown in figure 4.3(b). From these figures we can conclude that the following are good approximations: (i) the partial MAD is a good estimate of the final MAD, i.e., $E\{MAD|PMAD_i\} \approx PMAD_i$, and (ii) there exists a reliable model for the error, and this model is about the same for any values of PMAD, i.e., $p_{MAD|PMAD_i}(x) \approx p_{MAD-PMAD_i}(x - PMAD_i)$.

In DTFM, we stopped the metric computation as soon as $PSAD_i$ was greater than SAD_{bsf}. In HTFM, given the $PMAD_i$ at the i-th stage we want to decide whether the MAD is *likely* to be larger than MAD_{bsf}. If that is the case, we can terminate the matching distance computation early, with the risk that the actual MAD may turn out to be smaller than MAD_{bsf}. We refer to this risk as the probability of false alarm, P_F. More formally, our goal is

Formulation 5 (HTFM) *Given $PMAD_i$ at the i-th stage ($i = \{1, 2, ..., b\}$) and*

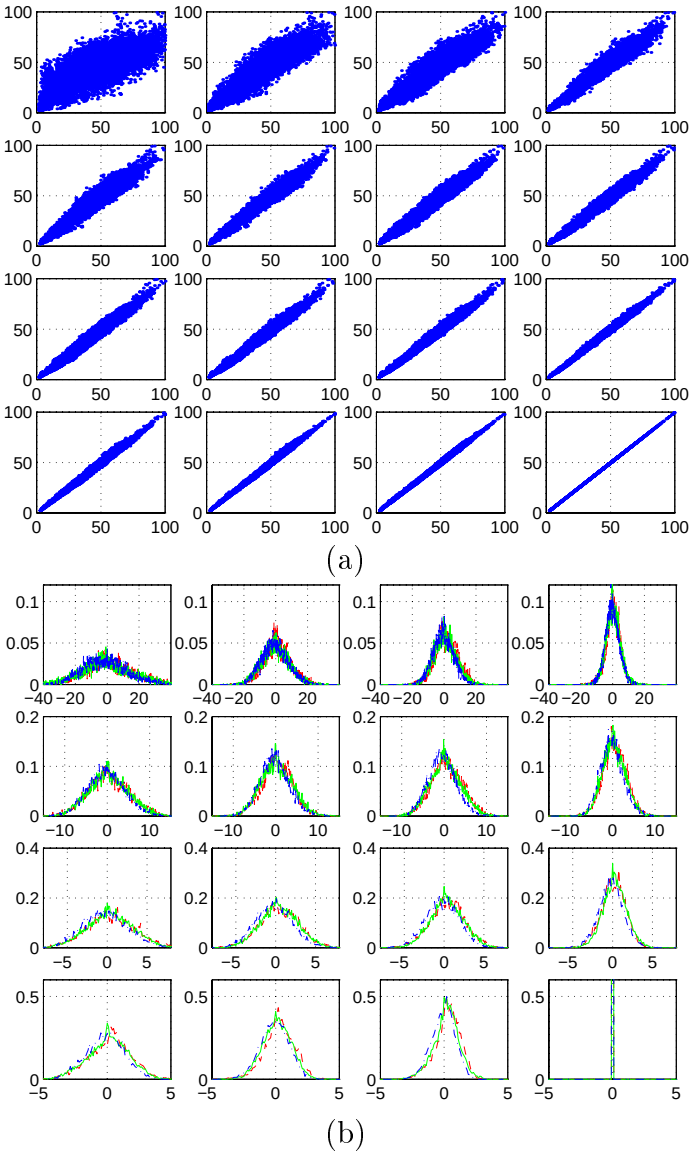


Figure 4.3: (a) Scatter plot between MAD (y-axis) and $PMAD_i$ (x-axis) and (b) corresponding histograms of $MAD - PMAD_i$. These are plotted for 16 stages of UNI subsampling, with number of pixels ranging from 8 (top left) to 128 (bottom right). We use UNI subsampling and ES on the "mobile &calendar" sequence. Similar results can be obtained with other sequences, search methods and subsampling grids.

MAD_{bsf}, we want to decide between the following two hypotheses:

$$\begin{aligned} H_0 &: MAD < MAD_{bsf} \\ H_1 &: MAD \geq MAD_{bsf} \end{aligned}$$

such that the constraint

$$P_F \equiv Pr(H_0|decide\ \ H_1) < P_f \ \textit{is satisfied.}$$

where P_f is the targeted P_F.

If H_1 is chosen, we stop the SAD computation. Otherwise, we compute $PMAD_{i+1}$ and perform another hypothesis testing at the $i+1$-th stage. Note that we could formulate the problem using the Neyman-Pearson criterion in which $Pr(\text{decide}\ \ H_1|H_0)Pr(H_0)$ (probability of false alarm) is constrained and $Pr(\text{decide}\ \ H_0|H_1)Pr(H_1)$ (probability of missed detection) is minimized. However, the resulting decision rule turns out to be the same. Also, we treat MAD_{bsf} and $PMAD_i$ as constants rather than assuming some prior distribution for them. Thus, we only need to consider and model the conditional probability of MAD given $PMAD_i$. P_F can then be rewritten as (see also Fig. 4.4),

$$P_F = \int_{-\infty}^{MAD_{bsf}} p_{MAD|PMAD_i}(x)dx = \int_{-\infty}^{MAD_{bsf}-PMAD_i} p_{MAD-PMAD_i}(x)dx$$

Given this probability, we can find a threshold parameter, Th_i, such that P_F is less than some threshold P_f,

$$\int_{-\infty}^{-Th_i} p_{MAD-PMAD_i}(x)dx = P_f \tag{4.2}$$

so that the decision rule becomes (see Fig.4.4)

$$PMAD_i - MAD_{bsf} \underset{H_0}{\overset{H_1}{\gtrless}} Th_i \tag{4.3}$$

Now we can replace the PSAD testing at step 5 of Algorithm 2 (DTFM) by (4.3). As illustrated in figure 4.4, $Pr(H_0|\text{decide}\ \ H_1)$ is the area under the $p_{(MAD|PMAD_i)}(x)$ function to the left of MAD_{bsf}. In general, we could select

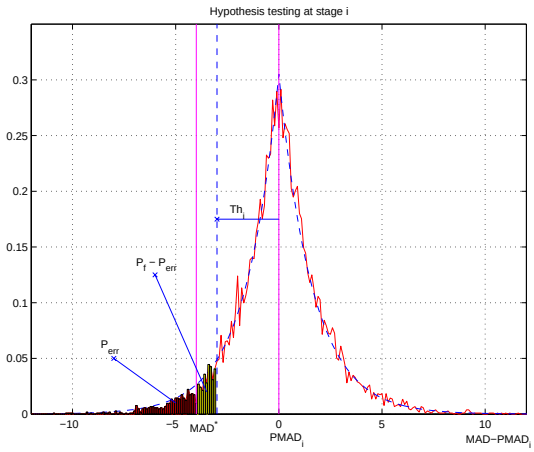


Figure 4.4: Empirical pdf of $MAD - PMAD_i$ (estimation error) obtained from histogram of training data (solid line) and the corresponding parametric model (dashed line). HTFM terminates the matching metric computation at stage i if $PMAD_i - MAD_{bsf} > Th_i$.

different P_f thresholds for each stage in the hypothesis testing. However, for simplicity, we fix P_f to a constant at all stages in our experiments.

From the histogram in Fig.4.3(b) we can model the difference between MAD and $PMAD_i$ as a random variable with Laplacian distribution, i.e., $p_{(MAD-PMAD_i)}(x) = \frac{\lambda_i}{2} e^{-\lambda_i |x|}$ where λ_i is the Laplacian parameter for stage i. We found that this model is accurate for many test sequences and FS methods. With a Laplacian model, the threshold (4.2) can be written as a function of λ_i and P_f as follows

$$Th_i = \begin{cases} -\frac{1}{\lambda_i} ln(2P_f) & P_f \leq 0.5 \\ -\frac{1}{\lambda_i} ln 2(1 - P_f) & P_f > 0.5 \end{cases} \tag{4.4}$$

Note that the Th_i of each stage is different because the model (and therefore λ_i) is different even if the same P_f is used for each stage.

So far, we have formalized a hypothesis testing framework based on likelihood testing of PMAD. However, there is a situation where it is useful to combine HTFM and DTFM. In some cases $PSAD_i$ is already larger than SAD_{bsf} but our probabilistic estimate indicates that $PMAD_i$ is not large enough to be eliminated (for example if P_f is very small), i.e., $PSAD_i|X|/|x_i| - SAD_{bsf} < Th_i|X|$ but $PSAD_i > SAD_{bsf}$. This would result in computing successive refinements of

PMAD. This situation happens more in the last few stages and when SAD_{bsf} has a small value, i.e., when we are close to finding the best point in the search region . Therefore, in order to take full advantage of all information available at a certain stage, our HTFM also incorporates the partial SAD test in conjunction with the likelihood test at each stage. The HTFM, thus, can be summarized as

Algorithm 3 (HTFM)

Same as Algorithm 2 except that **Step 5** *is replaced by:*

Step 5 *If* $PSAD_i \geq SAD_{bsf}$ *or* $PMAD_i > MAD_{bsf} + Th_i$*, we eliminate the current candidate and go to step 2. Otherwise, let* $i = i + 1$ *and repeat step 3.*

The proposed HTFM technique reduces matching metric computation cost, but introduces an overhead due to the hypothesis test at each stage (one more comparison and two additions). While this additional complexity is outweighed in general by the gain from early termination, it is also possible to optimize the testing. Thus some tests could be pruned with the goal of minimal overall complexity for a specific data with known statistics, i.e., the probability mass distribution of being terminated at certain stage as in [88]. However, we have found that the optimization does not give significant speedup, therefore, we do not discuss nor elaborate this issue in this thesis.

4.3 Parameter Estimation

In Section 4.2, we assumed that the conditional p.d.f. of MAD given $PMAD_i$ at stage i is known, so that the appropriate thresholds can be derived from this distribution. In practice, however, these statistics are not known a priori for a particular sequence. A possible solution would be to select in advance these probability distributions, through training over a set of video sequences. However, we have observed that the statistics of different sequences can differ significantly, depending on the frequency content of each frame, motion of objects in a frame and the FS techniques used. For example, a frame consisting of only low spatial frequency components tends to have less MAD estimation error than a frame with high frequency components. A frame with many moving objects causing

uncorrelated motion vector also gives higher MAD estimation error. Moreover, with initialization-refinement approaches to FS (e.g. [4]), the MAD estimation error is smaller than for ES because the statistics based on a set of candidate vectors that are already expected to be good (i.e., their SAD will be close to the optimal one). For these reasons, we focus on approaches that estimate the probability models on line, with updates taking place every few frames in a video sequence, under the assumption that the statistics do not change rapidly over a small group of frames.

We model the conditional density to be a Laplacian distribution with parameter λ_i. Thus we will only need to estimate the λ_i's in order to update the HTFM thresholds. Furthermore, λ_i is related to the variances, $\sigma_i^2 = E\{(MAD - PMAD_i)^2\}$, and the first order absolute moments, $\mu_i = E\{|MAD - PMAD_i|\}$ by

$$\lambda_i = \sqrt{2/\sigma_i^2} = 1/\mu_i \tag{4.5}$$

Therefore, obtaining any one of these three parameters is equivalent. Obviously, obtaining exact statistics would require that we compute the full MAD for each block, so that no fast matching speed up would be possible for the training frame. We now propose two training techniques for fast approximation of the threshold parameters for both ROW and UNI subsampling. In both techniques, our goal is to maintain the speed up due to DTFM while estimating the statistics reliably.

4.3.1 Model Estimation for ROW

Assume that when using DTFM for one block, the SAD calculation is terminated at stage t. In this case we have no information about $PMAD_i$ for $i > t$, but, given that we terminated early, the corresponding $PMAD_t$ can be thought to be a good approximation of the overall true MAD. Thus, our estimate $\tilde{\sigma}_i^2$ for σ_i^2 can be computed by assuming that for each block $MAD \simeq PMAD_t$, so that our estimated error variance is

$$\begin{aligned} \tilde{\sigma}_i^2 &= E_{t|i}\{(PMAD_t - PMAD_i)^2\} \quad \text{for } t > i \\ &\approx 2E_{t|i}\{|PMAD_t - PMAD_i|\}^2 \quad \text{with the Laplacian assumption} \end{aligned} \tag{4.6}$$

The update algorithm then becomes

Algorithm 4 (Variance estimation for ROW)

Step 1: *For a selected training frame, for every tested MV, perform DTFM in SAD calculation but also save* $|PMAD_t - PMAD_i|$ *for* $i < t$ *where* t *is the stage of DTFM termination.*

Step 2: *Compute* $\tilde{\sigma}_i^2$ *from* $2E_{t|i}\{|PMAD_t - PMAD_i|\}^2$.

Step 3: *Compute* $\lambda_i = \sqrt{2/\tilde{\sigma}_i^2}$ *and update thresholds for HTFM using this new estimate set of variances and (4.4).*

Thus for the training frames, we collect PMAD data only before the DTFM termination. The result of variances obtained this way is shown in Fig.4.5 averaged over 150 frames for each of the test sequences (ES is used in the experiment).

Two main observations can be made from Fig.4.5. First, it can be seen that $\tilde{\sigma}_i^2$ is obviously lower than the σ_i^2, thus resulting in smaller value of Th_i for a given targeted P_f which means that $Pr(SAD < SAD_{bsf}|\text{decide } H_1)$ is larger. Therefore, in order to obtain the same probability of error, P_f must be set to a smaller value. Second, we can further reduce the complexity of the data collection/manipulation and using linear interpolation to find $\hat{\sigma_i^2}$ for $i \neq \{1, 2, 4, 8\}$ by only computing $\sigma_1^2, \sigma_2^2, \sigma_4^2$, and σ_8^2.

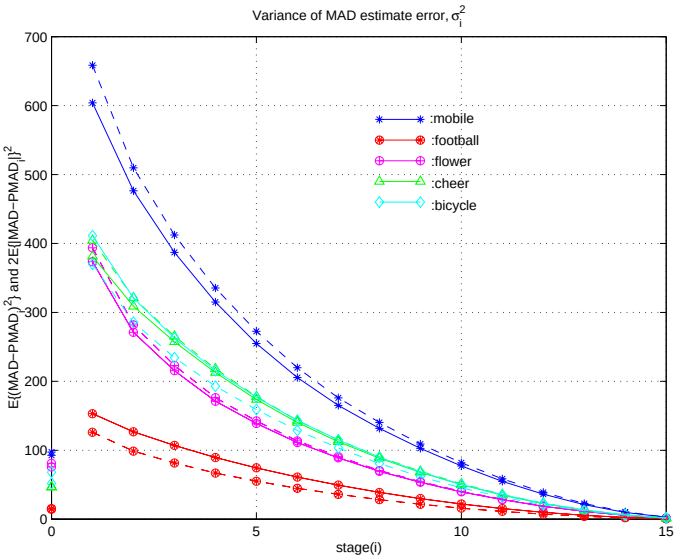
4.3.2 Model Estimation for UNI

In general, we can still use Algorithm 4 to approximate the model parameters for UNI subsampling. However, a better estimate $\hat{\sigma_i^2}$ of the true variance σ_i^2 can be obtained in this case. Let us consider the pixel at position $\vec{k} = (k_x, k_y)$ and denote $d(\vec{k})$ the pixel difference at that position,

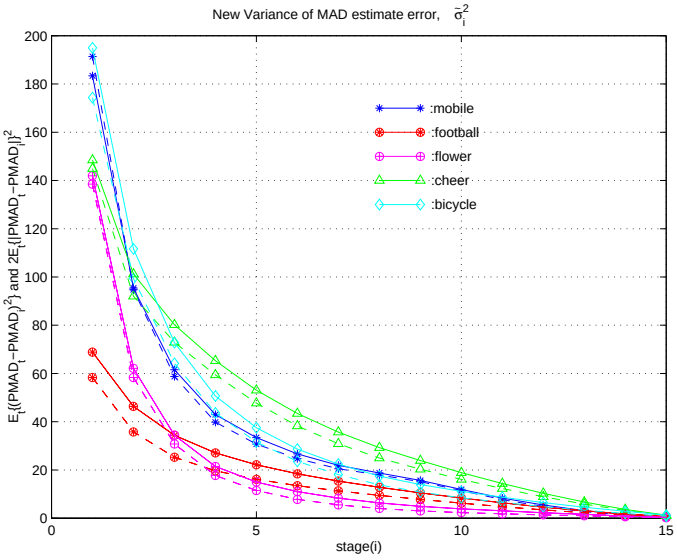
$$d(\vec{k}) = |I_t(\vec{k}) - I_{t-1}(\vec{k} + \vec{mv})|$$

We can write $PMAD_i$ as

$$PMAD_i = \sum_{\vec{k} \in x_i} d(\vec{k})/|x_i| \tag{4.7}$$



(a)



(b)

Figure 4.5: (a)σ_i^2 and (b) $\tilde{\sigma}_i^2$ of ROW computed from the definition (mean square) ('solid') and computed from the first order moment ('dashed'). The left-most points in (a) are $E\{(PMAD_1 - PMAD_2)^2\}$ and $2E\{|PMAD_1 - PMAD_2|\}^2$.

Consider the first two stages, $PMAD_1$ and $PMAD_2$. Because the pixels are uniformly spaced we can assume that the pixel difference, $d(\vec{k})$, is an i.i.d. random sequence with average MAD and variance σ_d^2. Hence, $PMAD_1$ and $PMAD_2$ can be viewed as time averages. Therefore, we have

$$\begin{aligned} E\{(PMAD_1 - PMAD_2)^2\} &= \sigma_d^2(|x_2| - |x_1|)/|x_2||x_1| \\ &= \sigma_1^2(|x_2| - |x_1|)/|x_2| \\ &= \sigma_2^2(|x_2| - |x_1|)/|x_1| \end{aligned} \tag{4.8}$$

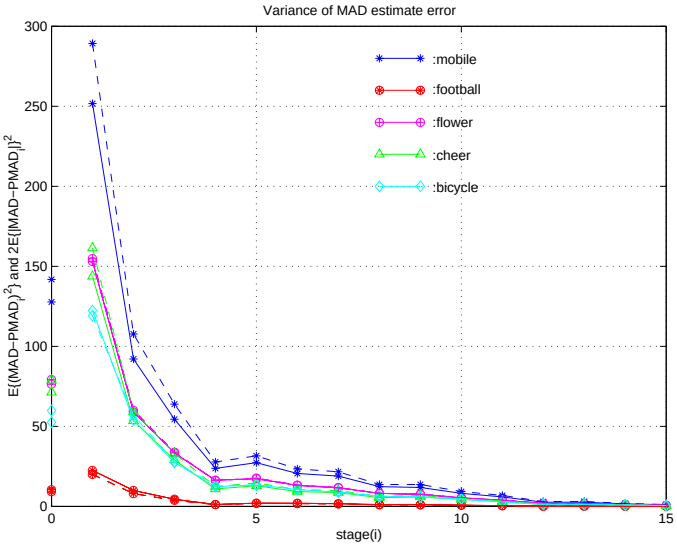
where σ_i^2 for $i = 1, 2$ are as defined previously. Therefore, using (4.8) for the first two test stages ($|x_2| = 16$ and $|x_1| = 8$), we can approximate σ_1^2 as $2E\{(PMAD_1 - PMAD_2)^2\}$. Besides, $PMAD_1 - PMAD_2$ can also be modeled to have Laplacian distribution. Hence its variance can be obtained without a square operation from the first order moment (expected absolute value), i.e., we can approximate σ_1^2 by

$$\sigma_1^2 \simeq 4E\{|PMAD_1 - PMAD_2|\}^2 = \hat{\sigma_1^2}.$$

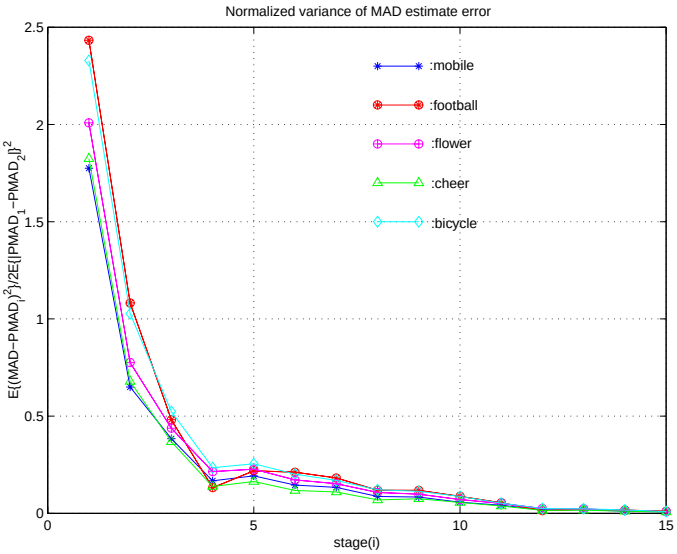
Our experiments (see Fig. 4.6(a)) show that this is a fairly accurate approximation. To approximate the variances at other stages, $\hat{\sigma_i^2}$, we observe that (see Fig.4.6(b)) the ratios between the first stage variance σ_1^2 and other σ_i^2 are almost constant regardless of the sequence and FS scheme used (see Fig. 4.6(b)). A possible explanation is based on the i.i.d. assumption of the pixel residue. From (4.7), it can be derived that the ratio of variances of $PMAD_i$ and $PMAD_j$ does not depend on σ_d^2 but a function of only i and j. Since we can approximate pixels under UNI as i.i.d., it makes sense to see consistent ratios among all test sequences. As a result, in our approach we only estimate $\hat{\sigma_1^2}$ and obtain the remaining $\hat{\sigma_i^2}$ by applying the scaling factors shown in Fig. 4.6(b). We also note (see Fig. 4.6(a)) that the variances can be estimated fairly accurately from the first order moment, μ_i^2.

Therefore, the algorithm to estimate model parameter for UNI can be summarized as

Algorithm 5 (Variance estimation for UNI)



(a)



(b)

Figure 4.6: (a) σ_i^2 ('solid') and $2\mu_i^2$ ('dashed') of MAD estimate error at 15 stages using ES and UNI, respectively. The left-most points shows $E\{(PMAD_1 - PMAD_2)^2\}$ and $2E\{|PMAD_1 - PMAD_2|\}^2$ for each sequence. (b) Ratio of $\sigma_i^2/\hat{\sigma}_1^2$ for each sequence. Note that this ratio is nearly the same for all sequences considered.

Step 1: *For a selected training frame, for every tested MV, always compute* $PMAD_1$ *and* $PMAD_2$ *and save* $|PMAD_1 - PMAD_2|$. *(DTFM or HTFM tests can be used for the following stages.)*

Step 2: *Compute* $\hat{\sigma}_1^2 = 4E\{|PMAD_1 - PMAD_2|\}^2$. *Compute other* $\hat{\sigma}_i^2$ *by dividing* $2\hat{\sigma}_1^2$ *with ratios in Fig. 4.6(b).*

Step 3: *Compute* $\lambda_i = \sqrt{2/\hat{\sigma}_i^2}$ *and update thresholds for HTFM using this new estimate set of variances and (4.4).*

This variance approximation technique is fast because the only data we have to collect is $PMAD_1 - PMAD_2$ and we can apply DTFM test (when no previous statistics are available) or HTFM test (using previous HTFM test parameters) at stage 2, 3 and so on, i.e., we still gain some computation saving while performing the training. Once again, the limitation of Algorithm 5 is that the i.i.d. assumption is only reasonable when using UNI. For ROW, it is obvious that we cannot apply the UNI parameter estimation technique.

Finally, to demonstrate the effectiveness of our online training scheme we show in Fig. 4.7 a comparison between the σ obtained by online training and the actual error variances for the corresponding frames. Our results show that these methods provide a good approximation without a significant impact in the complexity of the training frames. In our experimental results these training techniques will be used as appropriate and the corresponding training overhead will be included in the measured computation cost.

4.4 Experimental Results

We discuss the conditions of our experiments first. We use 5 test sequences, namely, "Mobile&Calendar", "Football", "Flower", "Cheer" and "Bicycle". All of them are 150 frames of size 360 by 240. We encode them with an MPEG2 coder based on the "mpeg2encode" source code of [103]. We use frame prediction mode only (in MPEG2 there are other modes of motion compensation such as field and dual-prime prediction). The range of motion search is -15 to 15. We set the target rate at 5 Mbps. All the results are generated on a PentiumII 300 MHz processor.

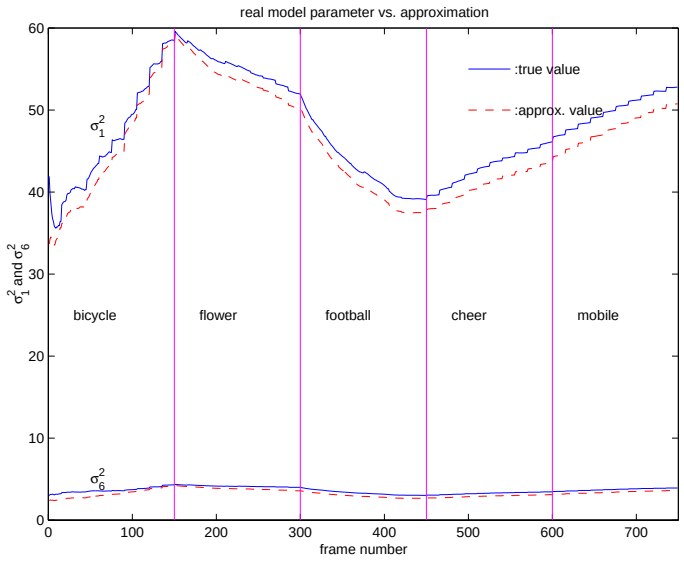


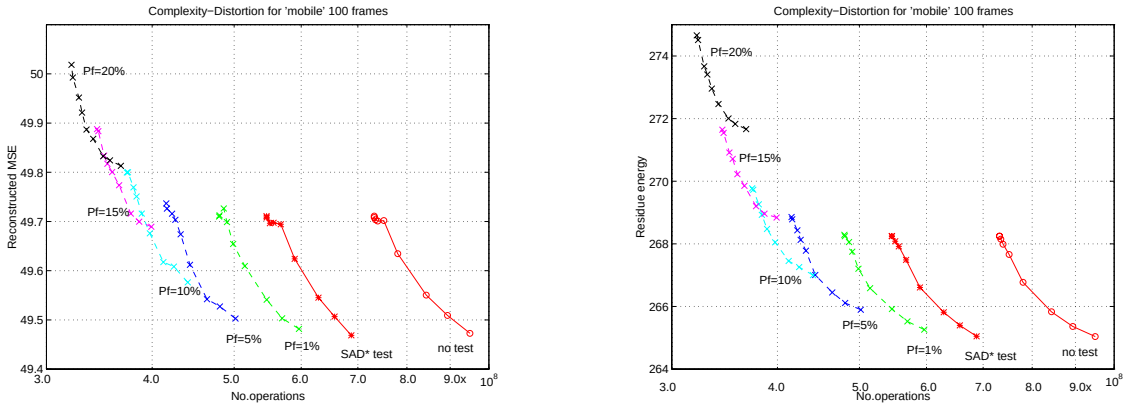
Figure 4.7: Example of tracking of statistics σ_i under UNI subsampling. Note that the approximated values track well the actual ones, even though the parameters do change over time. We use several different sequences to provide the comparison. This serves as motivation for using online training, rather than relying on precomputed statistics.

In addition to ES for the best motion vector, we also use 2-D log search [68] and ST1 algorithm [4] as fast search techniques. We use each one of these three FS algorithms at integer pel motion accuracy. In order to obtain half-pel motion accuracy, we perform a one step search over a small 3x3 window (on half-pel grid) around the best integer motion vector from the FS algorithms. We compare results between these three techniques with and without our HTFM using either UNI or ROW. This allows us to assess how our FM algorithm performs when a more efficient FS is also used.

4.4.1 VCA-FM versus VCA-FS

We compare the proposed VCA under FM approaches with that of the FS counterpart based on the idea in [86]. Unlike [86], we use ST1 algorithm [4] which also belongs to initialize-refine approach as the baseline algorithm. Similar to [86], we trade off computational complexity and distortion during the refinement

steps. The goal is to minimize the distortion under a constraint on the computational complexity. In the refinement process, the complexity constraint limits how far the local search goes. The Lagrange multiplier approach is used to get an unconstrained minimization for both parts.



(a) (b)

Figure 4.8: Complexity-Distortion curve for first 100 frames of "mobile" sequence (a) with MSE of the reconstructed sequence and (b) with residue energy as distortion measures and search without test ('o'), partial distance (SAD^* test) ('*') and combined hypothesis-SAD^* test ('x') each curve at fixed P_f labeled on each curve and varying λ from 0.01-4.

At a particular refinement iteration i, before starting the search for the minimal SAD within a 3x3 window centered around the current best result, we test if the Lagrangian cost condition

$$J_{i-1} = d_{i-1} + \lambda t_{i-1} > J_i = d_i + \lambda t_i \tag{4.9}$$

is met, where λ is the Lagrange multiplier, and d_i and t_i are the minimal distortion (SAD) and total complexity at the i-th iteration, respectively. If this condition is satisfied, the search continues, otherwise we terminate the search and return the vector with the minimal SAD so far. The variable complexity comes from early terminations of the refinement iteration before the original stop condition is met, i.e., before having a minimal SAD point in the center of a 3x3 window. λ is the factor that indicates how early the termination occurs and can be adjusted to

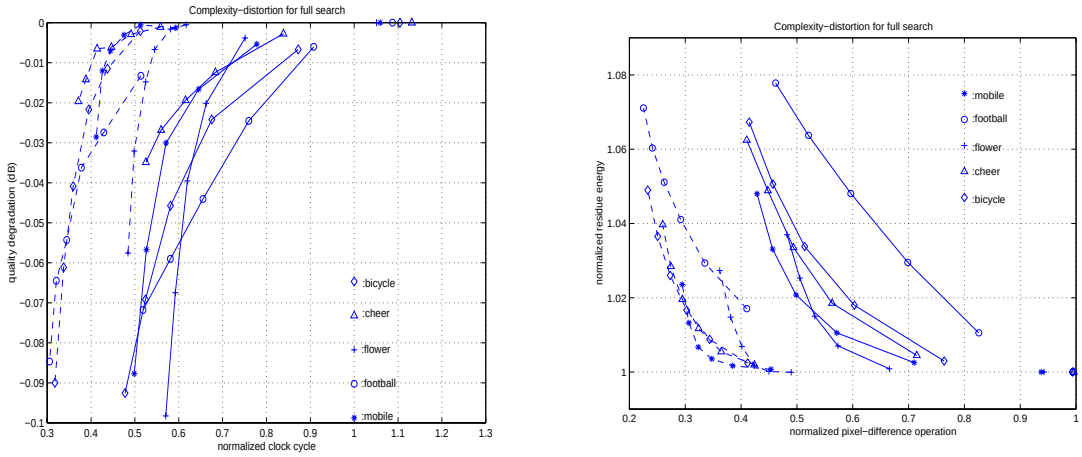
make the total complexity meet a certain budget. Intuitively, the iteration terminates if the reduction in distortion is outweighed by the monotonically increasing complexity, this will prevent further refining of a motion vector if the current residue energy is already small relative to the additional complexity required in the search.

Experimental results in terms of number of operations for different pairs of (P_f, λ) are shown in figure 4.8. We obtain the model parameter for HTFM "off-line", i.e., without using Algorithm 5 or 4. For each curve, the P_f is fixed while the value of λ changes to obtain the C-D tradeoffs. It can be seen that using λ allow scalability to ST1 algorithm with DTFM. However, by considering two parameters λ and P_f together, the best performance in terms of C-D tradeoff is to fix λ to zero and changing only P_f. This implies that the VCA-FS yields worse complexity-distortion tradeoff than the VCA-FM. Therefore, in the next section, we show only the result of our VCA-FM HTFM.

4.4.2 UNI versus ROW

In Figure 4.9 (a), we show the complexity-distortion of 5 test sequences using ES motion estimation with the HTFM. The units of complexity is the actual CPU clock cycles spent in motion estimation normalized by the same quantiy using ROW DTFM. The distortion is in terms of PSNR degradation from the DTFM (both ROW and UNI result in the same quality). If UNI is applied to DTFM, the resulting complexity is poorer, as can be seen by isolated points on 0 dB line. However, with HTFM, the UNI complexity is less than ROW by about 15% as the pixel difference saving overcomes the data access complexity and the likelihood of making wrong decision is smaller due to the reduced estimation error variance.

Also shown in Figure 4.9 (b) the same result as (a) but the number of pixel-difference operations is shown instead of the CPU clock cycle, and the average energy of residue pixel instead of the reconstructed PSNR degradation. It can be seen that the savings measured in terms of number of pixel-difference are always greater by about 5% because this measure does not take the complexity of search strategy, the test (DTFM or HTFM test), and parameter estimation into account.



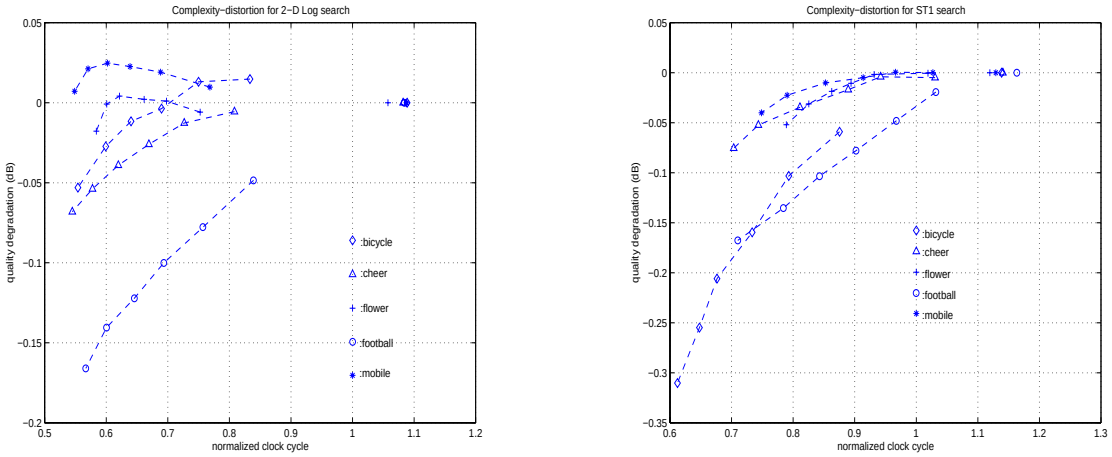
(a) (b)

Figure 4.9: Complexity-distortion of HTFM with ES and variance estimation on-the-fly, ROW ('solid') and UNI ('dashed'), (a) PSNR degradation vs. clock cycle and (b) residue energy per pixel vs. number of pixel-difference operations. Both clock cycle and number of pixel-difference operations are normalized by the result of ES with ROW DTFM. It can be seen that UNI HTFM performs better than ROW HTFM. The transform coding mitigates the effect of the increase of residue energy in the reconstructed frames. The testing overhead reduces the complexity reduction by about 5%. The complexity reduction is upto 65% at 0.05 dB degradation.

Therefore, it can be seen that the DTFM with UNI yields lower number of pixel-difference operations than ROW, but the actual CPU clock cycle is higher. It can also be seen that the decrease in reconsructed frame quality is less than the increase in residue energy as a result of the effect of bit allocation of remaining bits from motion vector coding to transform coding.

4.4.3 Scalability

In Fig.4.9 and 4.10, we show the computational scalability of the HTFM. In order to obtain a complexity-distortion curve we plot the complexity and distortion pair at different P_f values (ranging from 0.05 to 0.30 for UNI, and 0.01 to 0.20 for ROW.) For ROW we have to select lower P_f due to the underestimation of σ_i^2 by $\tilde{\sigma}_i^2$. Therefore, the real probability of error is larger than the targeted P_f. The statistics are updated using the proposed methods every GOP of size 15. Both



(a) (b)

Figure 4.10: Complexity-distortion of UNI HTFM with variance estimation on-the-fly of (a) 2-D Log search and (b) ST1 search. The axes are clock cycle and PSNR degradation normalized/compared to the 2-D Log search (a) or ST1 search (b) with ROW DTFM. The complexity reduction is upto 45% and 25% at 0.05 dB degradation for 2-D Log and ST1 search, respectively.

complexity and distortion are normalized by the complexity and distortion values of ROW DTFM.

Given that, as mentioned earlier, the UNI performs better than ROW, in Fig 4.10(a) and 4.10(b), we only show complexity-distortion using UNI HTFM for 2-D Log search and ST1 search, respectively. We can see that even though the results are not as good as for the ES case, we still gain complexity reduction with these FS algorithms. For example, we achieve a 45% complexity reduction with around 0.05 dB loss for 2-D Log search and a 25% complexity reduction for ST1 search. In terms of subjective quality, we observe no perceptual difference between DTFM and the HTFM at this level of degradation. In all experiments, one can see that the complexity-distortion performance of HTFM on the "Football" sequence is the worst because the high motion content results in high MAD estimation error variance. Therefore, ideally the HTFM works the best for sequence with low MAD estimation error variance. In other words, the residue has to be relatively smooth, which implies smooth moving objects with smooth background content.

4.4.4 Temporal Variation

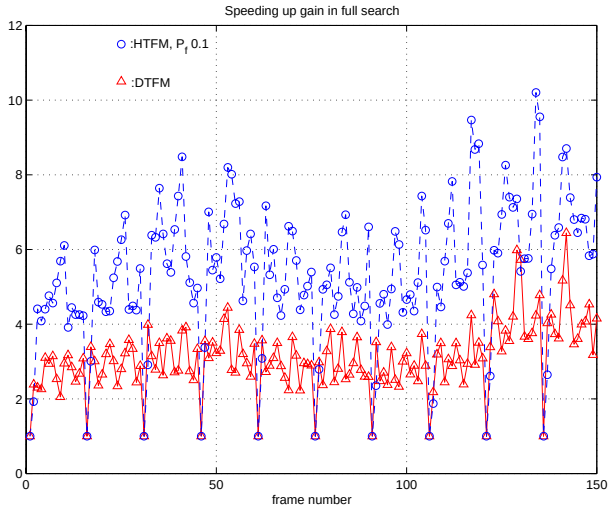


Figure 4.11: Frame-by-frame speedup factor for ES using ROW and DTFM ('Δ'), and ROW HTFM ('o') with $P_f = 0.1$ and 0.01 dB degradation.

Figs. 4.11, 4.12 and 4.13 show frame by frame speedup in clock cycle average for "Mobile" and "Football" using ES, 2-D Log search and ST1 search, respectively. Speedup is computed by comparing with the original FS algorithm (2-D Log or ST1 search) *without FM*. The P_f for HTFM is set to 0.1 or 10% for ES, 20% and 30% for 2-D Log and ST1 search, respectively. One can see that with fast model parameter estimation which takes place in the first P-frame of a GOP (size 15), we still perform almost as well as DTFM. By comparing Figs. 4.11 and 4.13, we can see that with an efficient FS algorithm, the extra speedup from DTFM is smaller, and thus speedup from HTFM algorithm is more difficult to get. Otherwise, the P_f can be set to larger value for greater speedup but that comes with the price of relatively larger distortion increase. It can also be seen from Fig. 4.13 that in some frames, the DTFM results in slower motion estimation (speedup less than 1). This is because the candidates being evaluated by ST1 are all almost equally good, thus resulting in almost no early terminations. This case is not observed by HTFM because of the probabilistic testing. In such case, the P_f can be set to as high as 40-50% without much PSNR degradation since any of the candidates evaluated by the FS are equally good.

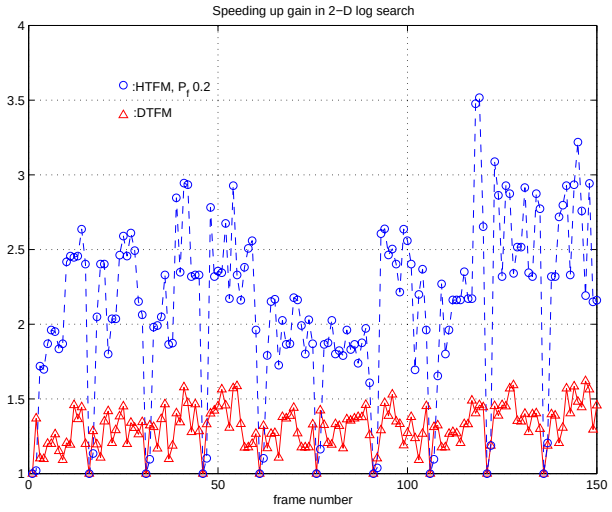


Figure 4.12: Frame-by-frame speedup factor for 2-D Log search using ROW and no FM ('*'), DTFM ('Δ'), and ROW HTFM ('o') with $P_f = 0.2$ and 0.04 dB degradation.

4.4.5 Overall Performance

Table 4.1 and 4.2 show the results in terms of *total encoding time* (seconds) needed to encode 150 frames of 5 test sequences using i) $|B| = 128$ pixels without FM, ii) $|B| = 128$ pixels with ROW DTFM iii) $|B| = 64$ pixels with ROW DTFM, iv) $|B| = 32$ pixels with ROW DTFM, v) UNI HTFM $P_f = 20\%$, vi) UNI HTFM $P_f = 30\%$, and vii) UNI HTFM $P_f = 40\%$ (all HTFM results are based on 128 pixels). Once again, we can see that the relative speedup due to FM is much more significant when using ES rather than a FS. Furthermore, we can see that in general our HTFM with $P_f = 30\%$ provides speedup as good as or better than using 64 pixels with ROW DTFM but with less distortion.

4.5 HTFM for VQ

Hypothesis testing fast matching can also be applied to VQ nearest neighborhood search. In VQ, the encoder must search for a codeword or code vector with distance nearest to an input vector. The matching metric normally used is the *Euclidean distance* between the input and the codeword, defined as $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 = \sum_{m=1}^{k}(x_m - y_m)^2$, where $\mathbf{x}$ is the input vector of dimension k and

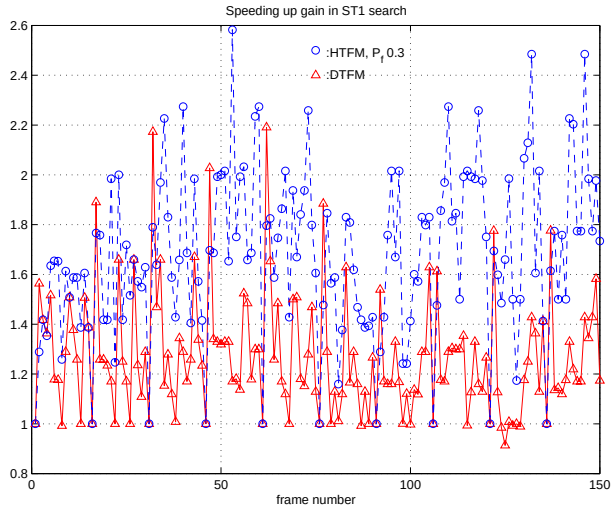


Figure 4.13: Frame-by-frame speedup factor for ST1 algorithm using ROW and no FM ('*'), DTFM ('Δ'), and ROW HTFM ('o') with $P_f = 0.3$ and 0.12 dB degradation.

$\mathbf{y}$ is a codeword in the codebook $C = \{\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_n}\}$ of size n. Therefore, the *quantization rule* is $Q(\mathbf{x}) = \mathbf{y_i}$ if $\|\mathbf{x} - \mathbf{y}_i\|^2 < \|\mathbf{x} - \mathbf{y}_j\|^2$, $\forall j \neq i$. The search time is linearly increasing with the dimension of the code vector and the size of the codebook.

As in the ME case, we first change the unit of the distance to be a per-dimension distance. In this case we define $\tilde{d}(\mathbf{x}, \mathbf{y}) = d(\mathbf{x}, \mathbf{y})/k$ and $\tilde{d}_{k'}(\mathbf{x}, \mathbf{y}) = d_{k'}(\mathbf{x}, \mathbf{y})/k'$. Our first goal is to estimate $\tilde{d}(\mathbf{x}, \mathbf{y})$ from the $\tilde{d}_{k'}(\mathbf{x}, \mathbf{y})$. Then we find the estimation error pdf and model it such that we can design a decision rule based on hypothesis testing to meet the targeted probability of false alarm. As in the ME case we found that $E\{\tilde{d}|\tilde{d}_{k'}\}$ can be well approximated by $\tilde{d}_{k'}$ For simplicity, we can also approximate the estimated error pdf using a Laplacian distribution, as in the ME case, and design the decision rule based on this assumption. The Laplacian parameter in this VQ case is obtained from the training vectors. The complexity-distortion result using HTFM is shown in Figure 4.14, which shows the result of DTFM at different code sizes, as well as HTFM curves corresponding to one codebook size with P_f ranging from 0.05 to 0.55. Figure 4.14(a) is for an i.i.d. Gaussian source with unit variance and Figure 4.14(b) is the high-high band from subband decomposition of "lenna" image. In both cases, the codebook

Table 4.1: Total time for encoding 150 frames and PSNR.

sequence	FM	ST1		Log		ES	
		sec.	PSNR	sec.	PSNR	sec.	PSNR
mobile	128 pels	37.78	32.4619	43.79	32.1536	472.75	32.3670
	DTFM (128 pels)	35.42	32.4619	39.33	32.1536	152.37	32.3670
	DTFM (64 pels)	33.63	32.1194	35.54	31.3711	98.25	31.9531
	DTFM (32 pels)	32.58	31.9322	33.59	30.9696	71.05	31.6990
	HTFM 20%	34.44	32.4518	35.14	32.1612	84.59	32.36
	HTFM 30%	33.78	32.4218	34.50	32.1436	80.39	32.3385
	HTFM 40%	33.52	32.3548	34.11	32.0778	77.88	32.2721
football	128 pels	39.05	40.4285	44.72	40.0528	449.64	40.5188
	DTFM (128 pels)	37.75	40.4285	42.19	40.0528	231.80	40.5188
	DTFM (64 pels)	34.40	40.2436	36.34	39.7743	129.63	40.3235
	DTFM (32 pels)	32.69	40.1621	33.62	39.6797	88.31	40.2269
	HTFM 20%	36.39	40.3264	36.70	39.8961	102.43	40.4645
	HTFM 30%	35.16	40.2576	35.64	39.8524	93.38	40.4341
	HTFM 40%	34.25	40.1938	34.92	39.7897	87.94	40.3937
flower	128 pels	37.75	35.3020	44.67	34.6225	466.02	35.3124
	DTFM (128 pels)	35.75	35.3020	40.69	34.6225	148.64	35.3124
	DTFM (64 pels)	33.19	35.0397	35.71	33.7696	94.88	35.0197
	DTFM (32 pels)	32.37	34.9337	33.41	33.2857	68.80	34.9056
	HTFM 20%	34.67	35.2834	35.81	34.6291	92.80	35.2976
	HTFM 30%	34.00	35.2499	35.24	34.6072	87.78	35.2548
	HTFM 40%	33.50	35.1884	34.66	34.5820	82.23	35.1661

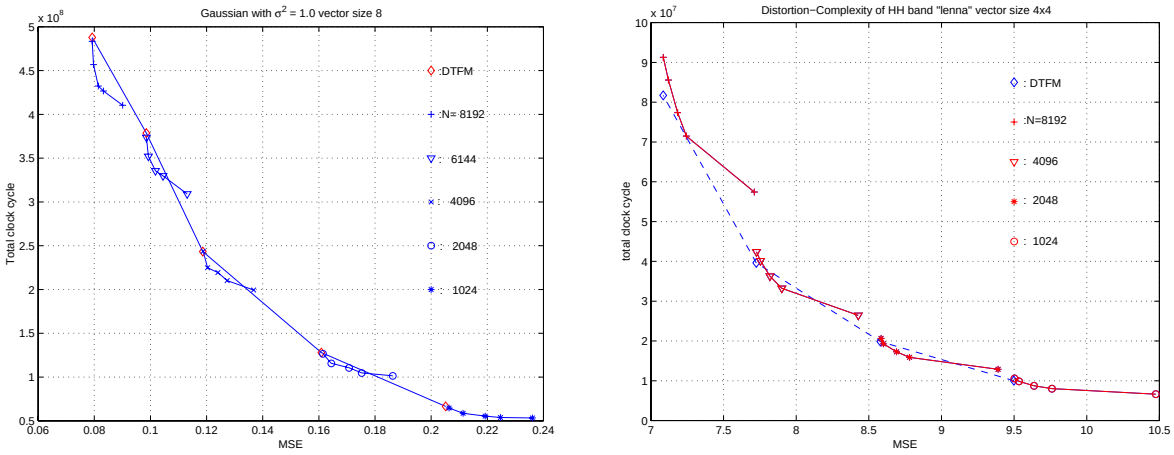
is designed using the LBG [104] algorithm from training vectors which are i.i.d. Gaussian and typical images, respectively.

We can see that unlike the ME case, in which the equivalent codebook size is fixed by the search region, in this VQ case the size of the codebook can be chosen to meet a complexity-distortion requirement. Note, however, that in order to operate in a computation scalable mode, in the DTFM case the codebook size has to be modified, while scalability is achieved with a constant codebook size for the HTFM case. In Figure 4.14, complexity-distortion performance achieved by HTFM is approximately the same as that achieved with DTFM using different codebook size within a certain range. This is due to several factors. First, the

Table 4.2: *cont.* Total time for encoding 150 frames and PSNR.

sequence	FM	ST1		Log		ES	
		sec.	PSNR	sec.	PSNR	sec.	PSNR
cheer	128 pels	42.61	35.0416	47.30	35.27	464.34	35.01
	DTFM (128 pels)	40.67	35.0416	43.34	35.00	175.66	35.01
	DTFM (64 pels)	37.96	34.9403	39.26	34.8811	107.44	34.9068
	DTFM (32 pels)	36.77	34.9051	37.37	34.8461	77.58	34.8624
	HTFM 20%	39.21	35.01	39.30	34.9610	92.25	35.00
	HTFM 30%	38.31	34.9662	38.60	34.9319	86.29	34.9879
	HTFM 40%	37.66	34.9219	37.85	34.9011	83.32	34.9712
bicycle	128 pels	42.28	35.1687	47.58	34.4392	461.28	35.1983
	DTFM (128 pels)	40.47	35.1687	45.04	34.4392	224.13	35.1983
	DTFM (64 pels)	36.78	34.8938	38.99	34.0651	128.43	34.8431
	DTFM (32 pels)	35.06	34.7540	36.16	33.9563	87.87	34.6779
	HTFM 20%	37.66	34.9628	39.56	34.3932	102.76	35.1574
	HTFM 30%	36.89	34.8585	38.42	34.3518	93.73	35.1083
	HTFM 40%	36.18	34.7495	37.53	34.2934	87.95	35.0401

speedup from using DTFM alone is already large, i.e., about 3 to 4 times faster than the original MSE computation. More than 90% of the distance computations are terminated early by DTFM, and most of the terminations occur at early stages. Second, the HTFM introduces more overhead cost for testing while providing more early termination at first few stages. However, the number of additional early termination is relatively small compared to the overall gain achieved by DTFM. Finally, the vector dimension in VQ case is still far less than in the ME case (16x16 macroblock). Thus, a few extra early terminations at an earlier stage are outweighed by the overhead cost for extra testing. Therefore, in order to get the maximum speedup performance, in our experiment for subband data VQ, Figure 4.14(b), we apply the HTFM test to the first one quarter of dimensions and simply use the DTFM test for the rest. As a conclusion, the HTFM for VQ, even though it does not provide a significant speedup over DTFM, provides computational scalability for a given fixed codebook, while scalability can only be achieved with different codebook sizes with DTFM.



(a) (b)

Figure 4.14: Complexity-distortion of HTFM VQ with vector size (a) 8 for i.i.d. source and (b) 16 (4x4) for high-high band of "lenna" image.

4.6 Summary and Conclusions

To summarize the HTFM, we propose a fast motion estimation by using fast matching in variable complexity framework where the number of pixels used in calculation varies depending on the likelihood that the MAD which is estimated by partial MAD will be larger than the "best found-so-far" SAD. The complexity is input-dependent i.e. it varies depending on the nature of sequences, and is adjustable by the degree of probability to make wrong decision. We formalize the problem with the assumption of the knowledge of the distribution of the MAD estimation error. Then the hypothesis testing is applied to minimize the probability of error. We call this novel algorithm HTFM.

We can apply HTFM to the matching criterion computation of any FS algorithm and also the nearest neighbor search for VQ encoding. However, in the case of ME the complexity reductions as observed from our experiment are less when more efficient FS is used because the set of considered MVs becomes so highly competitive that it is more difficult to predict which vector would win. Nevertheless, with a fast search algorithm we still achieve significant complexity reduction e.g., 45% with 0.1 dB PSNR degradation of the reconstructed sequences for 2-D Log search whereas we get 65% complexity saving for ES case. Therefore, our

algorithm allows a complexity-distortion tradeoff in fast matching that was not achieved by previous work. For example, in [67], attempts to reduce the complexity are done with reduced but fixed complexity algorithm, i.e. the complexity is not input-dependent. We also show the result of using reduced subset fast matching with DTFM. However, our HTFM still performs better. While other authors [86] have studied variable complexity approaches, these were targetting the C-D trade-off in FS, rather than FM as we propose.

Chapter 5

Motion Estimation: PDS-based Candidate Elimination

In this chapter we propose algorithms that perform both FM and FS approaches for motion estimation simultaneously using the technique of PDS-based candidate elimination. Similar to the HTFM in Chapter 4, this approach is based on the partial distance search (PDS) [79] in which the SAD calculation is broken into several steps, and at each step the partial SAD is updated and the calculation is allowed to stop once it is clear that the SAD will be greater than the best-so-far SAD. However, in the proposed approach, instead of using the PDS algorithm to provide early termination of SAD computation of candidates evaluated sequentially, in this work, SAD of several candidates are computed simultaneouly step-by-step and at each step some of them are eliminated from the pool of candidates based on their potential to become the best candidate. The proposed algorithm can be viewed as a path search algorithm where the goal is to find a path with minimal SAD which represents the best candidate. Unlike the HTFM in Chapter 4, we do not have a model of the relationship between the partial and the full SAD. We use a simpler threshold for the likelihood testing. In addition, we also propose a multiresolution approach in which the spatial dependency is used to help reduce the number of candidates considered in early stages. Only few candidates considered as representatives of their neighbors are evaluated. If these representatives show the tendency to be good motion vectors, their neighbors will be evaluated

next. We propose two variants of the multiresolution approach, i.e., breadth-first and depth-first algorithms, respectively.

This chapter is organized as follows. In Section 5.1, we propose the ideal dynamic programming which achieves minimal complexity in terms of arithmetic operations. In Section 5.2, faster, albeit, suboptimal, variations of the multiple step PDS approach are proposed. In Section 5.3, the multiresolution algorithms are presented. The complexity-distortion results of all the proposed algorithms are also shown in this section. The conclusions are drawn in Section 5.4.

5.1 PDS-based Candidate Elimination

The complexity reduction from the PDS when applying to various search strategies can vary widely (e.g. 10-80%) depending on two major factors, namely, (i) how the macroblock B is partitioned, and (ii) what fast search (FS) algorithm is used. In the HTFM approach of Chapter 4, we propose a uniform partition (UNI) as a better method for macroblock partitioning as compared to row-by-row partition (ROW) in that it requires fewer pixel comparisons on the average. Therefore, in this work, we will also use the UNI scheme. In this chapter we also provide some experimental results on H.263 sequences in which we calculate the SAD using up to 256 pixel instead of 128 pixel as in the MPEG2 case in Chapter 4 (Fig. 4.1). The UNI partitioning for H.263 sequences is thus shown in Fig. 4.1.

In order to better visualize the effect of macroblock partitioning and associated FS algorithm, the cumulative probability of early termination in a 16-stage PDS is shown in Figure 5.2. The cumulative probability is defined as

$$F_t(i) = Pr\{\mathbf{t} \leq i\}$$

where t is the stage in which the termination occurs, and $t = 1, 2, ..., 16$. From Figure 5.2, the average number of stages required before the PDS termination can be easily seen from the area above each curve, $E\{\mathbf{t}\} = \int(\mathbf{1} - \mathbf{F_t(i)})\mathbf{di}$. This average termination stage varies from sequence to sequence. Furthermore, it can be seen from the figure that the average termination stage using UNI is less than

1	**13**	**3**	**16**	**1**	**13**	**3**	**16**
9	**5**	**12**	**7**	**9**	**5**	**12**	**7**
4	**15**	**2**	**14**	**4**	**15**	**2**	**14**
11	**8**	**10**	**6**	**11**	**8**	**10**	**6**
1	**13**	**3**	**16**	**1**	**13**	**3**	**16**
9	**5**	**12**	**7**	**9**	**5**	**12**	**7**
4	**15**	**2**	**14**	**4**	**15**	**2**	**14**
11	**8**	**10**	**6**	**11**	**8**	**10**	**6**

Figure 5.1: Uniform macroblock partition into 16 subsets, showing only upper-left 8x8 region. Partial distance tests at the i-th stage are performed after the metric has been computed on the pixels labeled with i.

that of ROW for exhaustive search, thus resulting in less complexity. In Chapter 4, it was shown that the relative speedup from PDS when applied to a FS is less than when applied to exhaustive search. This is because the subset of candidate vectors considered in a FS contains only good candidates, i.e., their SADs are already small. Therefore, the SAD termination tends to occur during the last few stages. This fact is well illustrated in Figure 5.2 when a FS is used. Therefore, for a certain FS strategy, the smaller the average of termination stage, the more efficient the PDS. This raises the issue of whether the order in which the SAD of each candidate is calculated can be optimized.

It is obvious that if a candidate with smaller SAD is computed first, it is certain that the computation of worse candidates (with larger SAD) will be terminated at earlier stage. In fact, if the metric of best candidate is computed first, the order of the remaining candidates to be tested can be disregarded because the order does not affect the stage at which the termination takes place for each of the remaining candidates. However, in practice, the best candidate is obviously not known beforehand. Only a good initial candidate can be guessed. In UBC's software implementation of H.263+ encoder [58], the full search is performed by taking this ordering issue into consideration. Starting from a good initial candidate (following the motion vector predictive coding), all the candidates surrounding the initial points are searched in a spirally outward direction.

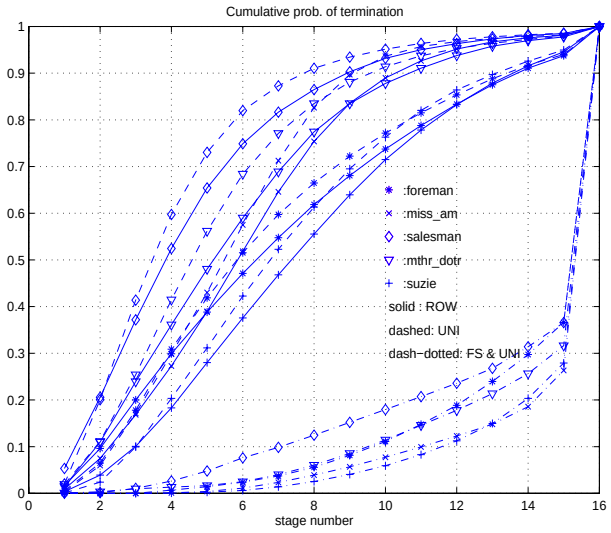


Figure 5.2: Cumulative probability of termination using 16 stage PDS and exhaustive search with ROW ('solid') and UNI ('dashed'), and using TMN's fast motion search with UNI ('dash-dotted') of 150 frames of five H.263 sequences coded at 56 Kbps. The efficiency of the PDS relatively drops as a FS is used.

5.1.1 Ideal Candidate Elimination

The dynamic programming is proposed as one way to achieve the minimal number of pixel comparison operations, which would correspond to the case when as if the best candidate is evaluated first. The basic idea of the ideal elimination is to update only a candidate with the smallest value of the partial SAD (PSAD) at a time. Using the same notation as in Chapter 4, the ideal candidate elimination algorithm is summarized as follows.

Algorithm 6 (Ideal Candidate Elimination (ICE-PDS))

Step 1:*Set SAD^* to infinity. Compute the $PSAD_1$ of every candidate in the search region. Set the stage index, $n(i)$, of every candidate to one, i.e., $n(i) = 1$ for all i candidates.*

Step 2:*Select candidate, i^*, with smallest value of PSAD satisfying i) $PSAD < SAD^*$ and ii) $n(i^*) < 16$. If no such candidate exists, return with the SAD^* result.*

Step 3:*Compute $PSAD_{n(i^*)+1}$ of this candidate. Set $n(i^*) = n(i^*) + 1$.*

Step 4:*If $n(i^*) = 16$, compare the PSAD with SAD^* and select the minimal to be the next SAD^* and the corresponding best-so-far MV, $\vec{\mathbf{MV}}^*$.*

Step 4:*Repeat Step 2.*

It can be verified that with this method, every candidate is terminated as earliest as possible since the PSAD will remain not updated as long as it is not the minimal PSAD at any given time. However, this algorithm requires multiple access to each of the candidates as well as determining the one with minimal PSAD at each step. These two overhead components may result in higher complexity than for the sequential candidate evaluation approach.

5.1.2 Reduced Steps Candidate Elimination

One way to reduce the amount of overhead mentioned above is to limit the number of times each candidate is considered. This is done by reducing the number of stages at which the PSAD calculation can be stopped. In this chapter, we propose a two-step CE-PDS. In the first step we compute the partial SAD of all candidates up to stage m. Then the candidate with minimal $PSAD_m$ is predicted to be the best vector and therefore its SAD is computed first by continuing SAD calculation from stage m. Then other candidates' SADs are computed using the PDS. Note that we can also apply the PDS technique in the first step. The stage number where termination occurs in the first step (Step 1 of Algorithm 7), $n(j)$, must be kept as the starting point from which the second step SAD calculation takes off. The algorithm is summarized as follows.

Algorithm 7 (Two-Step Candidate Elimination (2-CE-PDS))

Step 1: *With a preselected m value, find $MV^*(\Gamma, x_m)$ using PDS and keep the partial SAD, $SAD(\vec{mv}_j, x_{n(j)})$ where $n(j) \leq m$ is defined above for the j-th candidate.*

Step 2: *Update $SAD(MV^*(\Gamma, x_m), B)$ first, set it to SAD_{bsf} and use PDS to compute $SAD(\vec{mv}_j, B)$ $\forall j, \vec{mv}_j \neq MV^*(\Gamma, x_m)$, starting from $PSAD_{n(j)}$.*

This method does not employ an initialization technique like other FS algorithms. It is based solely on the matching metric. With this algorithm, the memory corresponding to the search area has to be accessed only twice. In Table 5.1, the complexity in terms of the number of stages in the SAD calculation (or

equivalently, number of pixel comparison operation) of the 2-CE-PDS algorithm is shown and compared with the original PDS and the ideal PDS in which the best candidate is found first. Note that all algorithms result in the same rate-distortion performance. The experiment is based on UBC's TMN version 3.2 [58]. The experimental environment is as follows, 56 Kbps target bit rate, code first 150 frames of each sequence, search distance 5x5 around the initial guess (offset) motion vector, code I and P-frame only, and use full search[1]. There are 16 stage in computing SAD as in Figure 5.1. The complexity in the unit described above reduces from the original PDS with UNI by the amount of 0.5-5% when $m = 1$ as compared to the ideal PDS complexity reduction which is about 1-7.5%.

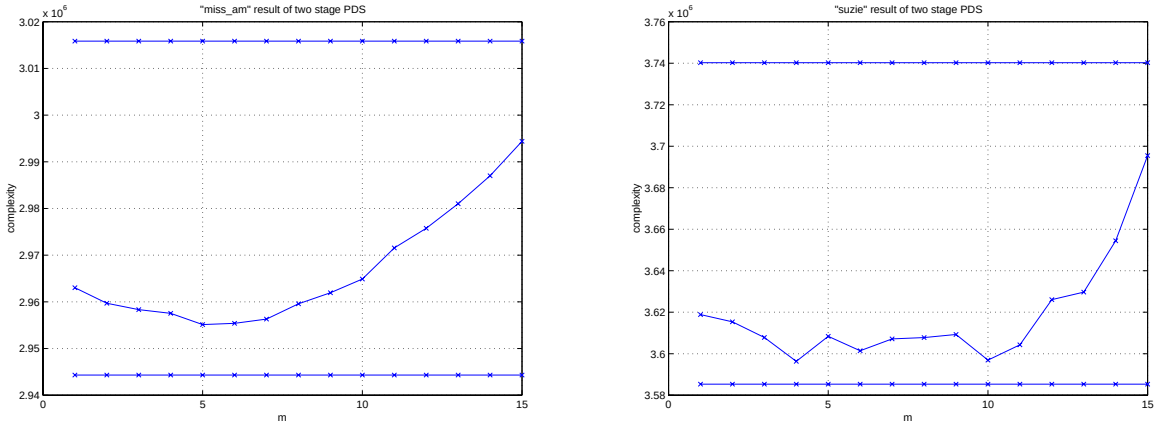
Table 5.1: Number of PSAD metric computation stages for different PDS variation for 150 frames of H.263 sequences coded at 56 Kbps.

H.263 sequences	**Orig. PDS (ROW)**	**Orig. PDS (UNI)**	**Ideal PDS (UNI)**	**2-CE-PDS**	
				$m = 1$	m_{min}
miss_am	3214159	3015848	2944313	2962825	2955141
foreman	3528755	3407654	3151037	3222447	3181377
salesman	2355461	2166800	2144171	2154860	2150489
mthr_dotr	2975387	2725827	2680578	2700361	2690790
suzie	3930649	3740283	3585322	3618637	3596301

Figure 5.3 shows the complexity of the 2-CE-PDS algorithm with different values of m. The value of m controls the tradeoff between the complexities of the computations before and after the m-th stage. Clearly, if m is small, we have a less accurate guess of the best candidate, whereas if m is large, we get a good guess but spend more on computation in the first step. However, with the optimal value of m, m_{min}, the minimal complexity is just less than 1% over the ideal PDS result for all sequences. The value of m_{min} for "Miss America", "Foreman", "Salesman", "Mother&Daughter", "Suzie" are 5, 5, 8, 8 and 4, respectively. The value of m_{min} can be related to the cumulative probability of termination in Fig. 5.2. In sequences where the initial candidate is not good enough, the termination can occur at any stage (see "Suzie") whereas with good initial candidate the termination tends to be seen earlier (see "Salesman"). With the former type

[1]The processor type is Pentium II 450 MHz with Windows NT 4.0.

of sequences, the first step PDS significantly helps reducing the complexity by finding a good guess of initial candidate. For the latter type of sequences where the initial "guess" is good enough, in order to get even better initial candidate, the number of stages m needs to be larger. However, the difference between the original PDS (UNI) and the ideal PDS is already small (from 1-10%) because of the good initial candidate provided by the original algorithm, thus using any small value of m yields similar results. Therefore, for simplicity, $m = 1$ would be a good choice.



(a) (b)

Figure 5.3: Complexity (number of stages) versus m for (a) "Miss America" and (b) "Suzie". The top and bottom lines in each figure are the original PDS with UNI and the ideal PDS, respectively.

One can further generalize the two-step algorithm to an algorithm having an arbitrary number of step (P-CE-PDS), i.e., we divide the calculation into p steps, for each step we update the best partial SAD first. Let $m(i)$, $i = 1, ..., p$, denote the SAD calculation stage at which step i stops, and $m(1) < m(2) < ... < m(p) = b$, i.e., $x_{m(p)} = B$. The p-step PDS can be written as

Algorithm 8 (P-Step Candidate Elimination (P-CE-PDS))

Step 1: *Assume there are p steps to compute SAD. The i-th step stops at stage $m(i)$, $i = 1, .., p$. Set $i = 1$*

Step 2: *Find $MV^*(\Gamma, x_{m(i)})$ using PDS and keep the partial SAD, $SAD(\vec{m}v_j, x_n(j))$ where $n(j) \leq m(i)$ for the j-th candidate.*

Step 3: *Compute* $SAD(MV^*(\Gamma, x_{m(i)}), x_{m(i+1)})$ *first, set it to* SAD_{bsf} *and find* $MV^*(\Gamma, x_{m(i+1)})$ *using PDS by computing* $SAD(\vec{mv}_j, x_{m(i+1)})$ *for other* j*-th candidate starting from* $PSAD_{n(j)}$.

Step 4: $i = i + 1$. *If* $i < p$ *repeat Step 3. Otherwise, return* $MV^*(\Gamma, B)$.

When the number of the steps of the P-PDS is equal to number of steps of the PDS, this algorithm is equivalent to the ICE-PDS in Algorithm 6. However, from our preliminary result the additional speedup using P-CE-PDS does not significantly improve 2-CE-PDS results. Also, it can be seen that even with the ideal PDS, the additional speedup is small when the original algorithm already starts with a good initial candidate, as in our experiment. In order to achieve further complexity reduction, we have to resort to a fast search which may not reach a global minimum vector.

5.2 Suboptimal CE-PDS

In this section, we further motivate a suboptimal solution using the result of the CE-PDS algorithms. Unlike other FS algorithms, which have an assumption of monotonically increasing error surface as the distance between a candidate and the local minimum grows, we reduce the number of the candidates tested using their own partial metric information. The key idea would be analogous to a tree growing where only a few branches are kept at a given stage. We propose 2 techniques to limit the number of branches grown, one is computationally nonscalable and the other is computationally scalable.

5.2.1 Computationally Nonscalable Fast Search

We start with an example on 2-CE-PDS. In the Step 2 of Algorithm 7 we update the SAD of candidates whose $PSAD_m$ has been computed, i.e., for those candidates terminated before stage m, the SAD computation is never completed. This makes sense because it is likely that candidates surviving the first step PDS will be good candidates. We can then generalize this idea to p-step which can be summarized as follows using the same notation as previously.

Algorithm 9 (P-Step Fast CE-PDS (P-FCE))

Step 1: *Assume there are p steps to compute SAD. The i-th step stops at stage* $m(i)$, $i = 1, .., p$, *and* $x_{m(p)} = B$. *Set* $i = 1$, *and* $\gamma_1 = \Gamma$.

Step 2: *Find* $MV^*(\gamma_1, x_{m(i)})$ *using PDS and keep the partial SAD of the* j*-th candidate,* $SAD(\vec{mv}_j, x_n(j))$, **only for those whose** $n(j) = m(i)$.

Step 3: *Update* $SAD(MV^*(\gamma_i, x_{m(i)}), x_{m(i+1)})$ *first, set it to* SAD_{bsf}.

Step 4: *Let* $\gamma_{i+1} = \{\vec{mv}_j | n(j) = m(i)\}$. *Find* $MV^*(\gamma_{i+1}, x_{m(i+1)})$ *using PDS by updating* $SAD(\vec{mv}_j, x_{m(i+1)})$ $\forall j$. *Update* $n(j)$.

Step 5: $i = i + 1$. *If* $i < p$ *repeat Step 3. Otherwise, return* $MV^*(\gamma_p, x_{m(p)})$.

Table 5.2: Results of 2-FCE complexity reduction with respect to the original PDS.

Sequences	SNR difference	m_{min}	Complexity (# stages)	% comp. reduction	CPU clk. cycle ratio	% clk. reduction
miss_am	0 dB	3	1030878	67.9%	1853/3277	43.5%
foreman	-0.01 dB	4	1566098	55.6%	2157/3719	42.0%
salesman	0 dB	3	808212	65.7%	1545/2687	42.5%
mthr_dotr	+0.01dB	3	1049889	64.7%	1807/3108	41.9%
suzie	+0.02dB	4	1695524	56.9%	2362/3607	34.5%

The intrinsic complexity of this algorithm stems from having to test whether a candidate is qualified for the next step PSAD computation or not. As there are more steps, this cost becomes larger and may outweigh the reduction of pixel comparisons. Therefore, similar to the discussion in the previous section, only 2-step FCE is considered. In Table 5.2, the result of 2-FCE with optimal value of m is shown. It can be seen that the complexity reduction, in terms of number of stages and the actual execution time, is obviously promising, with near optimal motion search performance.

Table 5.3 shows the result of 8-Step FCE with 2 stage interval between steps. We can see that the number of stage operation is further reduced but the total CPU clock cycle increases by 1-5% because of more testing operations overhead for valid candidates in each step. Therefore, our conclusion is that using 2 step suffices for significant speedup gain in terms of the execution time.

Table 5.4 shows the result of the fast search implemented in TMN H.263 codec [58]. It can be seen that even though our algorithm shows near optimal

Table 5.3: Results of 8-FCE (2 stage PSAD per 1 step testing).

Sequences	SNR difference	Complexity (no. of stages)	% comp. reduction	CPU clk. cycle ratio	% clk. reduction
miss_am	+0.01 dB	811838	74.7%	1983/3277	39.5%
foreman	-0.04 dB	1055540	70.1%	2145/3719	57.7%
salesman	0 dB	663683	71.8%	1740/2687	35.2%
mthr_dotr	-0.01dB	796422	73.2%	1889/3108	39.2%
suzie	-0.01dB	1142971	70.9%	2187/3607	39.4%

Table 5.4: Result of using UBC's Fastsearch option.

Sequences	SNR difference	Complexity (no. of stages)	%comp. reduction	CPU clk. cycle ratio	% clk. reduction
miss_am	-0.02	1139316	64.5%	1689/3277	48.5%
foreman	-0.11	1225935	65.3%	1702/3719	54.2%
salesman	0	816724	65.3%	1399/2687	47.9%
mthr_dotr	-0.05	1004437	66.2%	1543/3108	50.4%
suzie	-0.04	1267551	67.8%	1749/3607	51.5%

performance and less number of stage computation, in terms of execution time our algorithm is a bit inferior. This is due to extra clock cycles spent on candidates testings before computing the next step PSAD and multiple memory access for the search area. Another disadvantage of this algorithm is that the complexity is not scalable, i.e., the complexity-distortion operating point is fixed. Therefore, in order to be able to compare the performance with UBC's fast search at the same execution time or same distortion, we resort to a computationally scalable solution which allows significant speeding up gain with the sacrifice in some quality degradation.

5.2.2 Computationally Scalable Fast Search

To further reduce the complexity, we can limit the number of candidates passing to the next stage either by comparing the PSAD with a threshold or selecting only a limited number of candidates with the least PSAD. In this work, we choose to

follow the first approach because of simplicity in implementation. If the second approach is to be used, an extra complexity for sorting must be taken into account which will lessen the overall speedup. Consider the 2-step algorithm, after the first step we have an information of the best partial metric. Also, after updating the best partial SAD to obtain SAD_{bsf}, we can surmise that the final SAD would have the value around (strictly equal or less) the SAD_{bsf}. We can then set up a threshold with the value proportional to the SAD_{bsf} and the number of stage left in the 2nd step. Specifically, the threshold, T, is defined as

$$T = t \cdot SAD(MV^*(\Gamma, x_m), B) \cdot m/16$$

where t is a control parameter to adjust the tradeoff between the complexity and quality. The threshold is proportional to the linearly scaled version of the best-so-far SAD. Unlike Algorithm 7, in this scheme we only test for the PSAD thresholding. We do not test the value of $n(j)$, in order to maintain a reasonable complexity. In fact, for t less than or equal to 1, all the candidates that do not survive the first step PDS will also fail the threshold testing. For t greater than 1, the scheme allow more candidates to be considered even if they fail the first stage PSAD.

The computationally scalable algorithm can be summarized as follows.

Algorithm 10 (2-Step with Threshold Fast CE-PDS (2T-FCE))

Step 1: *Find* $MV^*(\Gamma, x_m)$ *using PDS and keep the partial SAD,* $SAD(\vec{mv}_j, x_n(j))$ $\forall \vec{mv}_j \in \Gamma$ *where* $n(j) \leq m$ *for the* j*-th candidate.*

Step 2: *Update* $SAD(MV^*(\Gamma, x_m), B)$ *first, set it to* SAD_{bsf} *and find* $MV^*(\gamma, B)$ *using the PDS where* $\gamma = \{\vec{mv}_j | SAD(\vec{mv}_j, x_n(j)) < T\}$ *where* $n(j)$ *is the starting stage for candidate* j*.*

Note that this 2-step algorithm can be easily extended to a p-step version. Table 5.5 shows the result of applying Algorithm 10 with $m = 1$ and $t = 1$. One can see that in most cases, we perform better than the UBC's fast search algorithm in both SNR and complexity (both execution time and number of operations). Besides, our algorithm is computationally scalable, i.e., the complexity can be made smaller by adjusting t to be smaller. As a result of smaller t, the quality

Table 5.5: Result of 2-Step with Threshold when $m = 1$ and $t = 1$.

Sequences	SNR difference	Complexity (no. of stages)	%comp. reduction	CPU clk. cycle ratio	% clk. reduction
miss_am	-0.02	647465	79.9%	1505/3277	54.1%
foreman	-0.08	839797	76.2%	1655/3719	55.5%
salesman	-0.03	581914	75.3%	1488/2687	44.6%
mthr_dotr	-0.02	657439	77.9%	1524/3108	50.9%
suzie	+0.01	831575	78.8%	1621/3607	55.1%

will be slightly degraded. Examples of the complexity-distortion behavior of the algorithm will be provided in the next section.

5.3 Multiresolution Algorithm

The proposed CE-PDS algorithm can be viewed as a variance of multiresolution algorithm (see [75], [77], etc.) where the pyramid decomposition and subsampling without anti-aliasing filter are used[2]. In general, the variable block size multiresolution algorithm can be implemented using the PDS technique. In the simplest scheme where no filtering operation is performed prior to subsampling, the coarsest resolution motion search is equivalent to searching over a subsampled grid of pixel using $PSAD_n$ such that x_n correspond to the subsampled pixels in the lowest resolution. In a finer resolution, the center of the search is obtained from the result of the coarser resolution search and the effective search area in the finer resolution can be reduced. The PSAD of candidates on the grid of finer resolution are then evaluated using a corresponding subset of pixel. The coarsest resolution grids come from the set of pixels in the first set of the UNI partitioning. Each set of pixels can be considered as one polyphase of the data. A finer resolution consists of more polyphases than a coarser resolution.

However, one advantage of the original multiresolution approach is the use of correlation between matching metrics of adjacent motion vectors to speedup the

[2]As discussed in [77] lack of an anti-aliasing filter is not crucial in pyramid coding.

search. In our algorithms in previous sections, this fact is not taken into consideration, thus resulting in redundant computations for candidates in the same neighborhood with similar matching metric values. Therefore, in this section, we propose algorithms that in addition to allowing scalable complexity based on our previous method, also take advantage of the candidate vector correlation as in the conventional multiresolution approach. There are several ways to obtain computational scalability. For example, the constraint of finding the best candidate in a particular resolution can be loosened in such a way that a "good" but not optimal candidate can be chosen if it can be found much faster than the optimal one. An approach is to allow more than one candidate to be the center of the search in the finer resolution using the partial metric criterion. This can be viewed as a soft-decision on what the initial candidate should be in the next resolution. We use a threshold approach similar to Algorithm 10 where the threshold is computed first from the initial motion vector at the center of the search region in the finest resolution. At the coarsest resolution, we compute the first stage PSAD on the coarsest grid. Candidates with PSAD less than the first threshold are then considered for the next step/resolution PSAD, as well as their neighbors in the finer resolution.

The order of the search in finer resolutions can be performed by a breadth-first or a depth-first approach. In breadth-first, the update of the partial metrics is performed for all candidates in the next level first before continuing to the following level. In depth-first, there are only two steps in partial metric computation, i.e., the update of partial metric is from the current resolution to the finest resolution. We propose two algorithms in Algorithm 11 and 12 corresponding to the breadth-first and depth-first approaches, respectively. Prior to describing the algorithm, let us introduce some new notations. Let us assume that there are R levels of resolution. Each resolution is obtained from subsampling the finer resolution by a factor of four. The macroblock is thus partitioned such that x_1 consists of only pixels in the coarsest resolution and there exist $x_{n(r)}$ for resolution r such that pixels in $x_{n(r)}$ correspond to pixels in resolution r ($n(1) = 1$). In our experiment for H.263 sequences, there are 3 resolutions and $n(1) = 1, n(2) = 4$,and $n(3) = 16$ (refer to Figure 5.1). Let γ_r be the set of candidates on the subsampled grid at

the r-th resolution. Let $\phi_r(\vec{mv})$ be the set of motion vectors in the neighborhood of $\vec{mv}$ in the r-th resolution, i.e., they are $\vec{mv} + [\pm 2^{R-r}, \pm 2^{R-r}]$ where R is the level number of the finest resolution. From these notations, the proposed multiresolution (MR) algorithms can be summarized as follows.

Algorithm 11 (Breadth-First Multiresolution (MR1-FCE))

Step 1: *Find the SAD of the initial motion vector, set it to* SAD_{bsf}. *Compute* $T(r)$ *for* $r = 1, ..., R$ *from* $T(r) = SAD_{bsf} \cdot 1/4^{(R-r)} \cdot t$. *Set* $r = 1$.

Step 2: *Compute* $SAD(\vec{mv}, x_1)$ *using PDS,* $\forall \ \vec{mv} \in \gamma_1$. $r \leftarrow r + 1$.

Step 3: *If* $SAD(\vec{mv}, x_{n(r-1)}) < T(r-1)$, *i) use PDS to update to* $SAD(\vec{mv}, x_{n(r)})$ *and ii) compute the neighboring* $SAD(\vec{v}, x_{n(r)})$ *using PDS* $\forall \vec{v} \in \phi_r(\vec{mv})$.

Step 4: $r \leftarrow r + 1$. *If* $r < R$, *repeat step 3. Otherwise, go to step 5.*

Step 5: *If* $SAD(\vec{mv}, x_{n(R-1)}) < T(R-1)$, *i) use PDS to update to* $SAD(\vec{mv}, x_{n(R)})$ *and ii) compute* $SAD(\vec{v}, x_{n(R)})$ *using PDS* $\forall \ \vec{v} \in \phi_R(\vec{mv})$.

Step 6: *Return* MV_{bsf} *as the best motion search result.*

Algorithm 12 (Depth-First Multiresolution (MR2-FCE))

Step 1: *The same as Step 1 in MR1-FCE.*

Step 2: *The same as Step 2 in MR1-FCE.*

Step 3: *If* $SAD(\vec{mv}, x_{n(r-1)}) < T(r-1)$, *i) use PDS to update to* $SAD(\vec{mv}, x_{n(R)})$ *and remove* $\vec{mv}$ *from the list of the candidate to be processed, and ii) compute the neighboring* $SAD(\vec{v}, x_{n(r)})$ *using PDS* $\forall \ \vec{v} \in \phi_r(\vec{mv})$.

Step 4: $r \leftarrow r + 1$. *If* $r < R$, *repeat step 3. Otherwise, go to step 5.*

Step 5: *The same as Step 5 in MR1-FCE.*

In both algorithms, the SAD_{bsf} and MV_{bsf} are updated with the PDS at the finest resolution. In MR2-FCE, the number of steps for SAD calculation for each candidate is at most two since the update is always to the finest resolution. The advantage of this approach is less number of data access (only twice) but the computation could be higher for some candidates which have small PSAD at early stages and large PSAD at later stages.

In Table 5.6, the result of the proposed multiresolution algorithms are shown for $t = 1.0$ and compared with the conventional multiresolution algorithm that use only one candidate as initial point in each resolution. It can be seen that

Table 5.6: Results of MR1-FCE and MR2-FCE complexity reduction at $t = 0.8$ with respect to the original multiresolution algorithm.

Sequences	MR1-FCE		MR2-FCE	
	SNR difference	% clk. increase	SNR difference	% clk. increase
miss_am	+0.31 dB	29.1%	+0.32 dB	23.7%
foreman	+0.73 dB	44.1%	+0.82 dB	51.8%
salesman	+0.04 dB	37.9%	+0.04 dB	27.6%
mthr_dotr	+0.29 dB	25.7%	+0.28dB	19.6%
suzie	+0.56 dB	22.2%	+0.56 dB	18.9%

the SNR is improved with the cost of extra computation. However, our proposed algorithm is also computationally scalable.

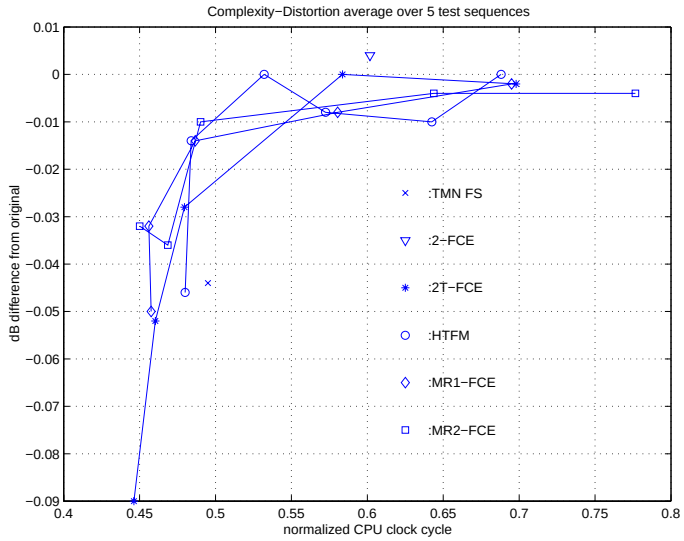


Figure 5.4: Complexity-Distortion using various algorithms average over 5 test sequences. The complexity unit is the clock cycles normalized by the original ROW PDS.

As shown in Fig. 5.4 and 5.5, the complexity-distortion curves of these two MR algorithms with varying t from 0.8 to 2 is compared with the scalable 2T-FCE and the nonscalable method 2-FCE from Table 5.2. The UBC's fast search [58], and the complexity scalable hypothesis testing fast matching (HTFM) are

also shown in the figures. Note that the results of the conventional multiresolution algorithm do not fit in the range displayed in Figure 5.4. The experimental environment is the same as before. The result is averaged over 5 test sequences which are "Miss America", "foreman", "salesman", "mother & daughter", and "suzie". One can see that 3 of our proposed algorithms (2T-FCE, MR1-FCE and MR2-FCE) outperform the fast algorithm in TMN3.0 and the HTFM at low quality-low complexity region. Furthermore, MR1-FCE and MR2-FCE perform almost equally well. In particular, MR1-FCE performs better at low quality (lower thresholds) while MR2-FCE performs better at high quality area (higher thresholds). This is reasonable because at high thresholds, there are more candidates passing to the finer resolutions, thus resulting in more overhead if there are many steps in PSAD calculation as in MR1-FCE. Compared to 2T-FCE, the proposed MR algorithms do not significantly achieve further speedup. This is due to the small search distance limit which has been set to ± 5 from the initial candidate at the finest resolution, i.e., only ± 1 at the coarsest resolution. If the search area is larger, the gain from MR approach will be clearer. The result in terms of number of pixel comparisons is shown in Figure 5.5. It can be seen that since this measure does not take the other complexity such as data access, PSAD testing, candidate selection, boundary checking, etc. into account, which contribute about 20% margin in the clock cycle measure.

5.4 Summary and Conclusions

We have presented a novel algorithm (PT-FCE) that takes advantage of the partial metric information to determine which candidates are worth to be fully evaluated. This algorithm can be viewed as a generalized version of multiresolution approach in the sense that the partial metric is obtained from a subset of pixel in a macroblock that corresponds to a certain resolution, and at each resolution the candidates to be considered are not necessary spatially related to the best candidate in the coarser resolution. Furthermore, the algorithm is computationally scalable, i.e., the complexity can be adjusted to meet the computational constraint with some sacrifice on poorer motion estimation performance. The result shows

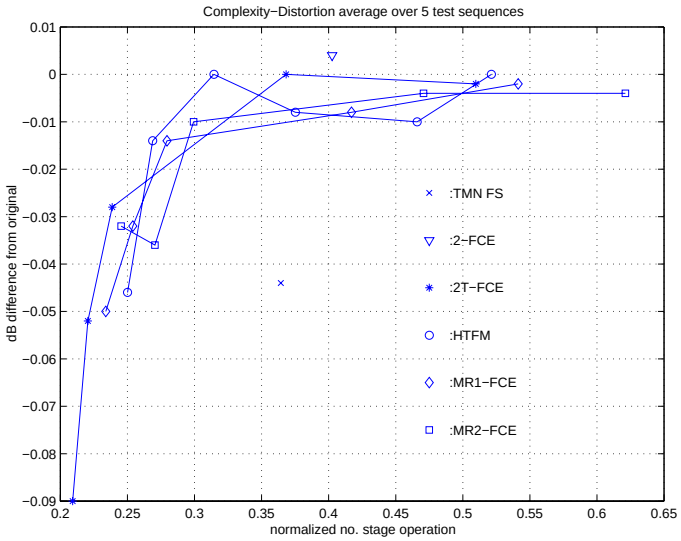


Figure 5.5: Complexity-Distortion using various algorithms average over 5 test sequences. The complexity unit is the the number of pixel comparisons normalized by the original ROW PDS.

that in terms of complexity-distortion tradeoff this novel algorithm (2T-FCE) outperforms the hypothesis testing algorithm (HTFM) with also less sophisticated algorithm.

Then, we propose two algorithms (MR1-FCE and MR2-FCE) that also use the spatial information in deciding the candidates to be evaluated. These algorithms possess a similar parent-child relationship as in the pyramid-based multiresolution approach without anti-aliasing filtering. They thus take advantage of spatial correlation of motion vectors to speedup the search. It also performs better than the original multiresolution because it allows soft-decision to be made based on partial metric. Thus, it is computationally scalable depending on the threshold factor that control the complexity-distortion tradeoff. The result shows that this approach performs little bit better than 2T-FCE. However, when the search area becomes larger, the difference in performance will be more visible.

References

[1] W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1994.

[2] J. Mitchell, W. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*. New York: Chapman and Hall, 1997.

[3] S. G. 16, "H.263 video coding for low bit rate communication," tech. rep., ITU-T, 1997.

[4] J. Chalidabhongse and C.-C. Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans.Cir.Sys. for Video Tech.*, April 1997.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 1991.

[6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. on Image Proc.*, 1992.

[7] *Wavelet and Subband Coding*. Prentice Hall P T R, 1995.

[8] K. Ramchandran, Z. Xiong, K. Asai, and M. Vetterli, "Adaptive transforms for image coding using spatially varying wavelet packets," *IEEE Trans. on Image Proc.*, 1996.

[9] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. on Signal Proc.*, vol. 36, pp. 1445–1453, Sept. 1988.

[10] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders," *IEEE Trans. on Image Proc.*, 1994.

[11] L.-J. Lin and A. Ortega, "Bit-rate control using piecewise approximated rate-distortion characteristics," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1998.

[12] C. Chrysafis and A. Ortega, "Efficient context-based entropy coding for lossy wavelet image compression," in *In Proc. of DCC'97*.

[13] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE. Trans. on Signal Processing*, 1993.

[14] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Trans. on Image Proc.*, 1994.

[15] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1996.

[16] H.-J. Wang and C.-C. Kuo, "A multi-Threshold wavelet coder (MTWC) for high fidelity image compression," 1997.

[17] J. Li and S. Lei, "An embedded still image coder with rate-distortion optimization," *IEEE Trans. on Image Proc.*, 1999.

[18] B.-B. Chai, J. Vass, and X. Zhuang, "Significant-linked connected component analysis for wavelet image coding," *IEEE Trans. on Image Proc.*, 1999.

[19] M. Crouse and K. Ramchandran, "Joint thresholding and quantizer selection for transform image coding: Entropy-constrained analysis and applications to baseline JPEG," *IEEE Trans. on Image Proc.*, 1997.

[20] J. Li, J. Li, and C.-C. Kuo, "Layered DCT still image compression," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1997.

[21] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," in *Proc. of ICASSP'91*, 1991.

[22] M. J. Gormish, *Source Coding with Channel, Distortion and Complexity Constraints*. PhD thesis, Stanford University, Mar. 1994.

[23] V. Goyal and M. Vetterli, "Computation distortion characteristics of block transform coding," in *Proc. of ICASSP'97*, (Munich, Germany), Apr. 1997.

[24] K. Rao and P. Yip, *Discrete Cosine Transform, Algorithms, Advantages, Applications*. Academic Press, 1990.

[25] T. N. N. Ahmed and K. R. Rao, "Discrete cosine transform," *IEEE Trans. on Computer*, 1974.

[26] M. Narasimha and A. Peterson, "On the computation of the discrete cosine transform," *IEEE Trans. on Comm.*, 1978.

[27] J. J.-I. Hong, *Discrete Fourier, Hartley and Cosine Transforms in Signal Processing*. PhD thesis, Columbia University, 1993.

[28] C. S. Wen-Hsiung Chen and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. on Comm.*, 1977.

[29] A. Ligtenberg and M. Vetterli, "A discrete fourier/cosine transform chip," *IEEE J. on Selected Areas in Comm.*, vol. SAC-4, pp. 49–61, Jan 1986.

[30] Z. Wang, "Fast algorithms for the discrete w transform and for the discrete fourier transform," *IEEE Trans. on Signal Proc.*, vol. ASSP-32, pp. 803–816, Aug. 1984.

[31] B. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. on Signal Proc.*, vol. ASSP-32, pp. 1243–1245, December 1984.

[32] P. Duhamel and H. H'Mida, "New 2n DCT algorithms suitable for VLSI implementation," in *Proc. of ICASSP'87*, (Dallas), p. 1805, Apr 1987.

[33] C. Loeffler, A. Ligtenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *In Proc. of ICASSP'89*.

[34] E. Feig and E. Linzer, "Discrete cosine transform algorithms for image data compression," *in Proc. of Electronic Imaging'90*, p. 84, 1990.

[35] M. Vetterli, "Tradeoffs in the computation of mono and multi-dimensional DCTs," tech. rep., Ctr. fo Telecommunications Research, Columbia University, June 1988.

[36] E. Feig and S. Winograd, "On the multiplicative complexity of discrete cosine transform," *IEEE Trans. on Information Theory*, 192.

[37] H. R. Wu and Z. Man, "Comments on "fast algorithms and implementation of 2-D discrete cosine transform," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1998.

[38] N. I. Cho and S. U. Lee, "A fast 4x4 DCT algorithm for the recursive 2-D DCT,"

[39] E. Linzer and E. Feig, "New scaled DCT algorithms for fused MULTIPLY/ADD architectures," in *Proc. of ICASSP'91*, vol. 4, pp. 2201–2204, 1991.

[40] V. Srinivasan and K. J. R. Liu, "Vlsi design of high-speed time-recursive 2-D DCT/IDCT processor for video applications," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1996.

[41] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. of IEICE*, vol. 71, p. 1095, Nov 1988.

[42] C. W. Kok, "Fast algorithm for computing discrete cosine transform," *IEEE Trans. on Signal Proc.*, 1997.

[43] V. S. Dimitrov, G. A. Jullien, and W. C. Miller, "A new DCT algorithm based on encoding algebraic integers," in *Proc. of ICASSP'98*.

[44] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image down-sampling and for inverse motion compensation," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 7, no. 3, pp. 458–475, June 1997.

[45] H. V. Sorensen and C. S. Burrus, "Efficient computation of the dft with only a subset of input or output points," *IEEE Trans. on Signal Proc.*, 1993.

[46] "The independent JPEG's group software JPEG, version 6." `ftp://ftp.uu.net`.

[47] A. Hossen and U. Heute, "Fast approximation DCT: Basic-idea, error analysis, application," in *Proc. of ICASSP'97*.

[48] S. Jung, S. Mitra, and D. Mukherjee, "Subband DCT: Definition, analysis, and applications," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1996.

[49] J. Bao, H. Sun, and T. Poon, "Hdtv down-conversion decoder," *IEEE Trans. on Consumer Electronics*, 1996.

[50] J. Song and B.-L. Yeo, "Fast extraction of spatially reduced image sequences from MPEG-2 compressed video," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1999.

[51] J. B. Lipsher, "Asynchronous DCT processor core," Master's thesis, Univ. of Cal. Davis.

[52] J. Winograd and S. Nawab, "Incremental refinement of DFT and STFT approximations," *IEEE Signal Proc. Letters*, 1995.

[53] W. K. Cham and F. S. Wu, "Compatibility of order-8 integer cosine transforms and the discrete cosine transform," in *Proc. of IEEE Region 10 Conf. on Computer and Communication Systems*, 1990.

[54] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Acad. Pub., 1992.

[55] K. S. Wang, J. O. Normile, H. Wu, and A. A. Rodriguez, "Vector-quantization-based video codec for software-only playback on personal computers," *Multimedia Systems 2(5)*, pp. 191–203, 1994.

[56] "MPEG video software decoder, v2.2." `http://bmrc.berkeley.edu/projects/mpeg/mpeg_play.html`.

[57] "vic:UCB/LBL video conferencing tool." `ftp://ftp.ee.lbl.gov/conferencing/vic/`.

[58] C. University of British Columbia, "Tmn h.263+ encoder version 3.0," *http://www.ee.ubc.ca /image/h263plus/*.

[59] K. Froitzheim and H. Wolf, "A knowledge-based approach to JPEG acceleration," in *Proc. of IS&T/SPIE Symp. on Electr. Imaging Science and technology*, (San Jose), Feb. 1995.

[60] E. Murata, M. Ikekawa, and I. Kuroda, "Fast 2D IDCT implementation with multimedia instructions for a software MPEG2 decoder," in *Proc. of ICASSP'98*.

[61] X. Bo and Z. Xulong, "A new algorithm for the implementation of h.263 video coding," *IEEE Trans. on Circ. and Sys. for Video Tech.*, To appear.

[62] K.Lengwehasatit and A.Ortega, "Distortion/decoding time tradeoffs in software DCT-based image coding," in *Proc. of ICASSP '97*, 1997.

[63] B. Girod and K. W. Stuhlmüller, "A content-dependent fast DCT for low bit-rate video coding," in *Proc. of ICIP'98*.

[64] I.-M. Pao and M.-T. Sun, "Modeling DCT coefficients for fast video encoding," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1999.

[65] K. Lengwehasatit and A. Ortega, "DCT computation with minimal average number of operations," in *Proc. of VCIP'97*, (San Jose, CA), Feb. 1997.

[66] K.Lengwehasatit and A.Ortega, "DCT computation based on variable complexity fast approximations," in *Proc. of ICIP'98*.

[67] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans.Cir.Sys. for Video Tech.*, vol. 3, pp. 148–157, April 1993.

[68] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. on Comm.*, 1981.

[69] R. Srinivasan and K. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans.Comm.*, vol. COMM-33, pp. 888–895, August 1985.

[70] R. Li, B. Zeng, and M. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 4, pp. 438–442, August 1994.

[71] L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion esitmation in video coding," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 6, pp. 419–422, August 1996.

[72] J. Y. Tham, S. Ranganath, and M. Ranganath, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 8, pp. 369–377, August 1998.

[73] M. Bierling, "Displacement estimation by hierarchical block matching," in *Proc. of VCIP'88*.

[74] F. Dufaux and M. Kunt, "Multigrid block matching motion estimation with an adaptive local mesh refinement," in *Proc. of VCIP'92*.

[75] S. Zafar, Y. Q. Zhang, and B. Jabbari, "Multiscale video representation using multiresolution motion compensation and wavelet decomposition," *IEEE J. on Sel. Areas in Comm.*, 1993.

[76] Y. Q. Zhang and S. Zafar, "Motion-compensated wavelet transform coding for color video compression," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1992.

[77] K. M. Uz, M. Vetterli, and D. J. LeGall, "Interpolative multiresolution coding of advanced television with compatible subchannels," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 1, pp. 86–99, March 1991.

[78] J.-B. Xu, L.-M. Po, and C.-K. Cheung, "Adaptive motion tracking block matching algorithm for video coding," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 9, pp. 1025–1029, October 1999.

[79] C.-D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Trans. on Comm.*, 1985.

[80] Y.-C. Lin and S.-C. Tai, "Fast full-search block-matching algorithm or motion-compensated video compression," *IEEE Trans. on Comm.*, 1997.

[81] Z.-L. He, K.-K. Chan, C.-Y. Tsui, and M. L. Liou, "Low power motion estimation design using adaptive pixel truncation," in *In Proc. of Int'l Symp. on Low Power Electronics and Design 1997*, pp. 167–172.

[82] B. Natarajan, V. Bhaskaran, and K. Konstantinides, "Low-complexity block-based motion estimation via one-bit transforms," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 7, pp. 702–706, August 1997.

[83] X. Lee and Y.-Q. Zhang, "A fast hierarchical motion-compensation scheme for video coding using block feature matching," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 6, pp. 627–634, December 1996.

[84] M. Khosravi and R. Schafer, "Low complexity matching criteria for image/video applications," *ICIP'94*, pp. 776–780, November 1994.

[85] M.-J. Chen, L.-G. Chen, T.-D. Chiueh, and Y.-P. Lee, "A new block-matching criterion for moion estimation and its implementation," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 5, pp. 231–236, June 1995.

[86] F. Kossentini and Y.-W. Lee, "Computation-constrained fast MPEG-2 encoding," *IEEE Signal Proc. Letters*, vol. 4, pp. 224–226, August 1997.

[87] F. Chen, J. Villasenor, and D. Park, "A low-complexity rate-distortion model for motion estimation in h.263," *Proc.ICIP'96*, vol. 2, pp. 517–520.

[88] K. Lengwehasatit, A. Ortega, A. Basso, and A. Reibman, "A novel computationally scalable algorithm for motion estimation," in *Proc. of VCIP'98.*

[89] K. Lengwehasatit and A. Ortega, "Probabilistic partial distance fast matching algorithms for motion estimation," *Accepted for publication in IEEE Trans on Circ. and Sys. for Video Tech.*, 1999.

[90] V. L. Do and K. Y. Yun, "A low-power vlsi architecture for full-search block-matching motion estimation," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1998.

[91] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures 2nd Edition.* Boston: Kluwer Academic Publishers, 1997.

[92] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architecture for video compression - A survey," *Proceedings of IEEE*, vol. 83, pp. 220–246, February 1995.

[93] H. Yeo and Y. H. Hu, "A novel modular systolic array architecture for full-search block matching motion estimation," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 5, pp. 407–416, October 1995.

[94] A. K. Jain, *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[95] F. Jelinek and J. B. Anderson, "Instrumentable tree encoding of information sources," *IEEE Trans. on Info. Th.*, 1971.

[96] D.A.Patterson and J.L.Hennessy, *Computer Architecture : a Quantitative Approach 2nd Ed.* Morgan Kaufmann Publishers, 1996.

[97] C. T. Chen, "Adaptive transform coding via quadtree-based variable block-size DCT," in *Proc. of ICASSP'89*, 1989.

[98] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. on Info. Th.*, 1989.

[99] G. Cheung, "Optimal bit allocation strategy for joint source/Channel coding of scalable video," Master's thesis, Electrical Engineering and Computer Sciences, U.C. Berkeley, 1998.

[100] P. Fernandez and A. Ortega, "An input dependent algorithm for the inverse discrete wavelet transform," in *In Proc. of 32nd Asilomar Conf.*

[101] K. Lengwehasatit and A. Ortega, "Complexity-distortion tradeoffs in vector matching based on probabilistic partial distance techniques," in *Proc. of Data Compression Conference, DCC'99*, (Snowbird, UT), Mar. 1999.

[102] F.-H. Cheng and S.-N. Sun, "New fast and efficient two-step search algorithm for block motion estimation," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 9, pp. 977–983, October 1999.

[103] M. S. S. Group, "Mpeg2 video codec version 1.2," *http://www.creative.net /~tristan/MPEG/mssg/mpeg2vidcodec_v12.tar.gz.*

[104] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Comm.*, 1980.