Algorithms for Transform Selection in Multiple-Transform Video Compression

by

Xun Cai

S.B. EE, University of Science and Technology of China, 2010

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
May 16, 2012

Certified by [illegible]
Professor Jae S. Lim
Thesis Supervisor

Accepted by .... [illegible] ....
Professor Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

Algorithms for Transform Selection in Multiple-Transform Video Compression

by

Xun Cai

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 2012, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Selecting proper transforms for video compression has been based on the rate-distortion criterion. Transforms that appear reasonable are incorporated into a video coding system and their performance is evaluated. This approach is tedious when a large number of transforms are used. A quick approach to evaluate these transforms is based on the energy compaction property. With a proper transform, an image or motion-compensated residual can be represented quite accurately with a small fraction of the transform coefficients. This is referred to as the energy compaction property. However, when multiple transforms are used, selecting the best transform for each block that leads to the best energy compaction is difficult.

In this thesis, we develop two algorithms to solve this problem. The first algorithm, which is computationally simple, leads to a locally optimal solution. The second algorithm, which is more intensive computationally, gives a globally optimal solution. We provide a detailed discussion on the ideas and steps of the algorithms, followed by the theoretical analysis of the performance. We verify that these algorithms are useful in a practical setting, by comparing and showing the consistency with rate-distortion results from previous research.

We apply the algorithms when a large number of transforms are used. These transforms are equal-length 1D-DCTs in 4x4 blocks, which try to characterize as many 1D structures as possible in motion-compensation residuals. By evaluating the energy compaction property of up to 245 transforms, we quickly determine whether these transforms will bring potential performance increase in a video coding system.

Thesis Supervisor: Professor Jae S. Lim
Title: Professor of Electrical Engineering

Acknowledgments

First of all, I would like to express my sincere gratitude to my research advisor, Professor Jae Lim, for his support and guidance during my M.S. research. His group provides me a chance to explore the research topic that interests me. His immense knowledge and vision helps me better understand the signal processing world. His constructive feedbacks, encouragement and patience have a remarkable influence on the completion of this thesis.

I would like to thank all the members that I have worked with in ATSP. Harley Zhang has provided a lot of technical help when I joined the group. Gun Bang has discussed both the research and general video processing technologies with me. I feel grateful for Cindy LeBlanc, who provides all kinds of support during my stay in this group.

Finally, I would like to thank my family members in China. My parents financially supported me during my first semester at MIT, which allows me to settle down in my dream school. I am grateful for my fiancee and soulmate, Mengyu Zhang, who supports my career and respects my choice of life style.

Contents

List of Figures

List of Tables

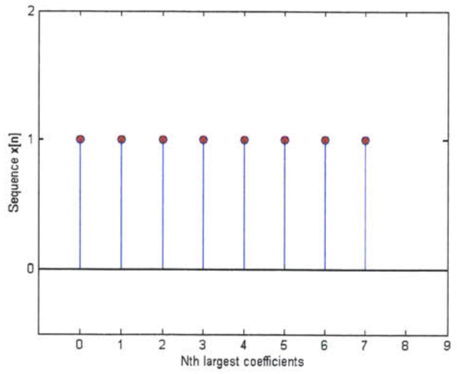
Chapter 1

Introduction

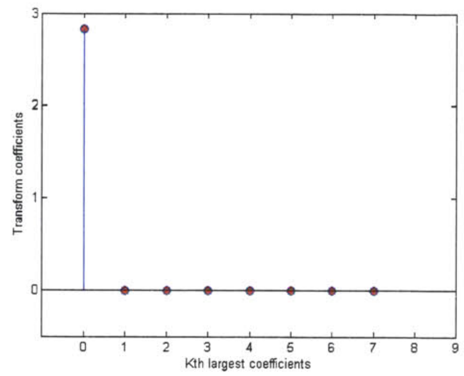
1.1 Energy Compaction

Transforms are used in many transform-based image and video compression systems. It is well known that these signals can be represented quite accurately with only a small fraction of the transform coefficients using proper transforms. This phenomenon is referred to as the energy compaction property [1]. Energy compaction can be measured as a function of preserved energy with respect to the number of largest-magnitude transform coefficients. A transform that preserves more energy when a fixed number of coefficients are used is considered desirable, since the loss of energy is related to the distortion present in image or video signals which the human visual system perceives.

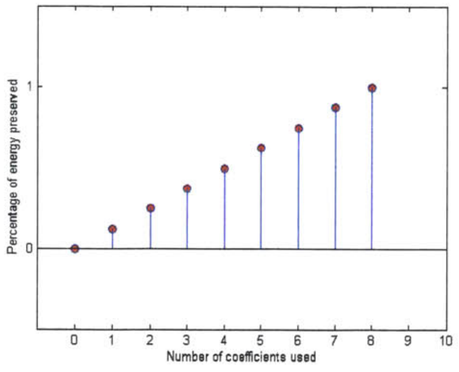
A transform with good energy compaction property is an indication of an effective signal representation. Consider the following example shown in Figure 1-1a, where a discrete sequence x[n]= u[n]-u[n-8] is approximated with different numbers of largest coefficients in both the time domain and the DCT domain. If this process is done in the time domain, the energy preserved is linear to the number of coefficients we use, which is shown in Figure 1-1c. This means using one coefficient only preserves 12.5% of its total energy. Intuitively, this representation is inefficient since each coefficient is equally important and it is not reasonable to discard any one of them over any other. However, if the signal is transformed with the DCT of length eight, the resulting transform coefficients only have a nonzero DC value, as shown in Figure 1-1b. This implies that all the energy can be preserved with one coefficient. The corresponding preserved energy function is shown in Figure 1-1d. In order to preserve the same amount of the energy, the time domain representation typically uses coefficients than the DCT representation. In other words, when the same amount of coefficients are used, the DCT representation preserves more energy than the time domain representation. Therefore, in this case, using DCT is a more efficient representation of the signal.



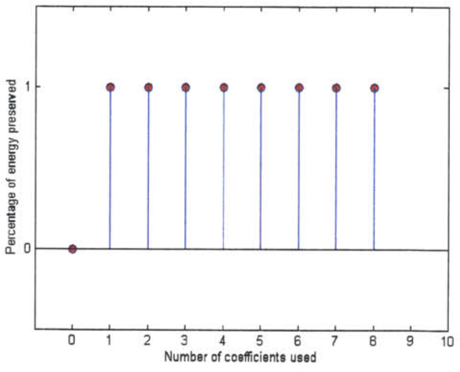
(a) Sequence x[n]



(b) DCT coefficients X[k]



(c) Energy preservation in the time domain

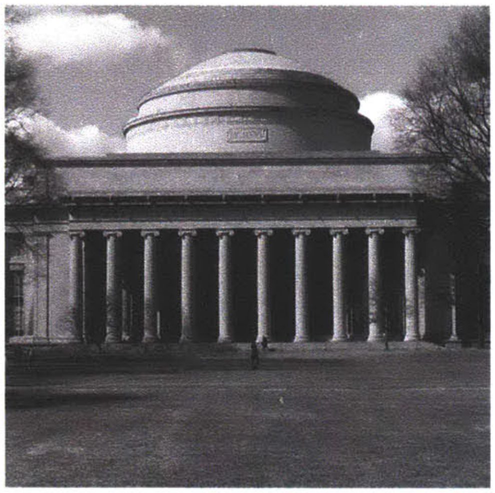


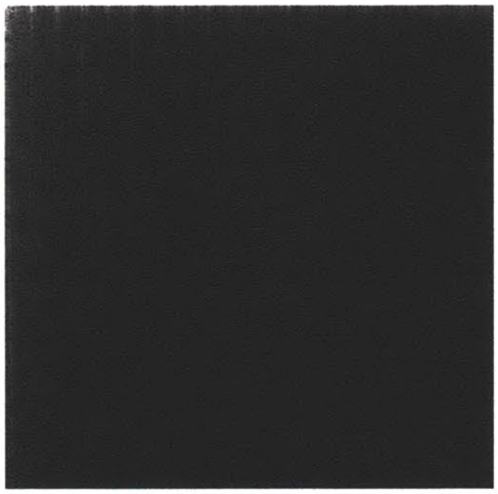
(d) Energy preservation in the transform domain

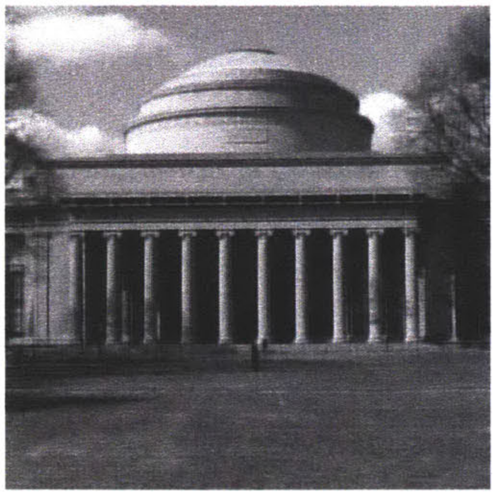
Figure 1-1: Example of the energy compaction property of DCT. The sequence in the time domain has equal-valued coefficeints and the energy is not compacted, while in the DCT domain, the energy is compacted into only one transform coefficient, which leads to a more efficient representation of the signal.

For most typical image and video frame signals, the 2D-DCT behaves in a similar manner, which can preserve much of their energy with a small number of coefficients. Figure 1-2 shows a typical image, its 2D-DCT coefficient magnitudes and the reconstructions from 5% and 15% of the largest transform coefficients. Figure 1-2b shows the DCT coefficient magnitudes of the original image. Most of the transform coefficients have small amplitudes (shown as darker), except those low frequency coefficients located at the upper left corner. This indicates a concentration of image energy by using a few transform coefficients that have large magnitudes. By only keeping these large-magnitude coefficients, most of the information in the image signal can be preserved and the image can be reconstructed with a fairly acceptable visual quality. Typically, fifteen percent of coefficients lead to a reconstruction that is not distinguishable from the original version by human eyes, as shown in Figure 1-2d.

Such an effective representation of a signal is important for a coding system, where these signals need to be compressed and transmitted without too much distortion. At some reasonable distortion level, a coding system attempts to minimize the number of bits used to represent the signal. For transforms with good energy compaction property, such as the 2D-DCT, most of the transform coefficients are close to zero. The length of the bit sequence produced by a 2D-DCT based compression system can be significantly reduced by discarding these coefficients with small magnitudes. As a result, the 2D-DCT has been extensively used in many image and video compression systems.


(a) Original image


(b) Magnitude of 2D-DCT of original image


(c) Reconstruction from 5% of coefficients

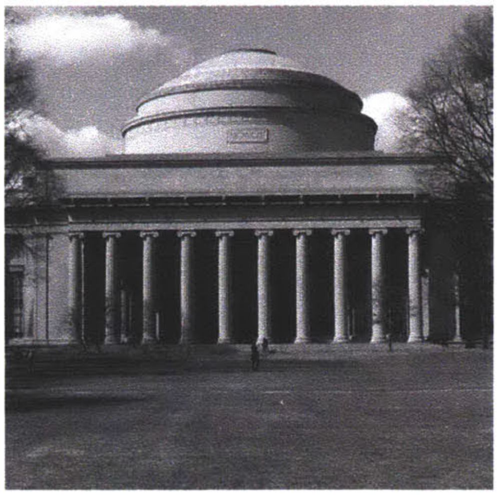

(d) Reconstruction from 15% of coefficients

Figure 1-2: An example of 2D-DCT and energy compaction property. The original image in (a) is transformed with 2D-DCT and the magnitudes of the transform coefficients are shown in (b). We reconstruct the image by keeping only the largest 5% and 15% transform coefficients and setting others to zero, and then applying the inverse 2D-DCT.

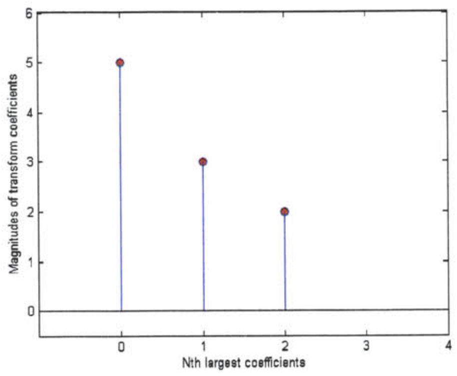
1.2 Evaluating the Energy Compaction in Multiple-Transform Environment

Many practical image and video coding systems are block-based. For example, in JPEG image coding system, images are segmented into 8x8 blocks, and 8x8 2D-DCT is computed for each block. The resulting block transform coefficients are further processed to bit streams. The block-based transform design is reasonable due to its effective implementation and decorrelation. First, compared to the local block-based transforms, global transforms are typically computationally more intensive. Second, a block transform decorrelates the signals well enough if the block size is chosen properly according to the signal contents. Most practical pictures have local features that can be better characterized by a block transform than a global transform. This issue will be further discussed in the next chapter.

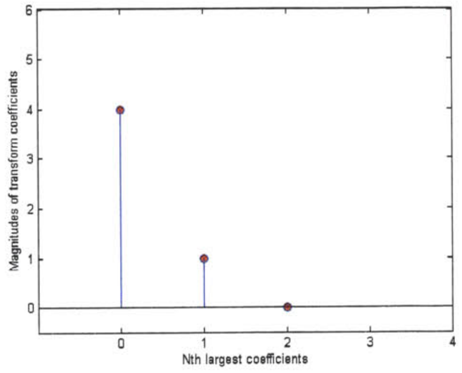
In a coding system that utilizes only one transform, preserving the maximum energy with a given number of transform coefficients can be accomplished simply in the following way. We can compute the transform coefficients for all the blocks of interest and select the transform coefficients whose magnitudes are above a certain threshold. By varying the threshold we can select the given number of coefficients that preserve the most energy.

This process can be illustrated with the following example shown in Figure 1-3. Suppose we have two blocks whose transforms coefficients have magnitudes 5,3,2 and 4,1,0, as shown in Figures 1-3a and 1-3b. If we choose all the coefficients whose magnitudes are no smaller than five, only one coefficient is chosen. That results an energy preservation of 25 with one coefficient. If we change the threshold to four, we choose two coefficients, one from each block. The preserved energy is 41 with two largest coefficients. By repeatedly changing the threshold and taking more coefficients into energy count, we can evaluate the energy preservation as a function of all possible numbers of coefficients used, as shown in Figure 1-3c.

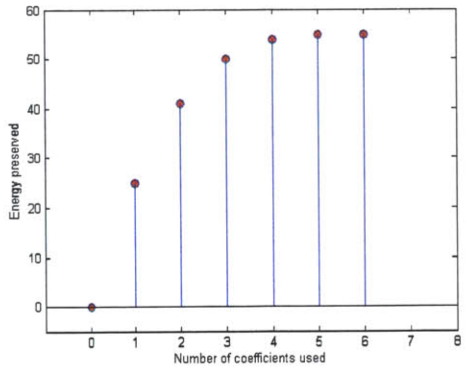
It is well known that the characteristics of image intensities vary significantly from one region to another region within the same video frame. Though motion-



(a) Transform coefficients in Block 1



(b) Transform coefficients in Block 2



(c) Energy preserved with different numbers of coefficients used

Figure 1-3: An example of energy preservation computation with single block transform.

compensated residuals share similar characteristics as typical images, recent studies have found that the characteristics of the image and the motion-compensated residual are often quite different [2]. Figure 1-4 shows an example of an image and a motion-compensated residual. These phenomena are observed in [3] that Motion compensated-residuals have many local one-dimensional structures. This result leads to the possibility that using more than one transform to allow a different transform for each block may significantly improve the performance of a video compression system.

In this multiple transform environment, choosing the best transform for each block and selecting the transform coefficients that preserve the most energy for a given number of transform coefficients is a very difficult task. Consider a single block. The best

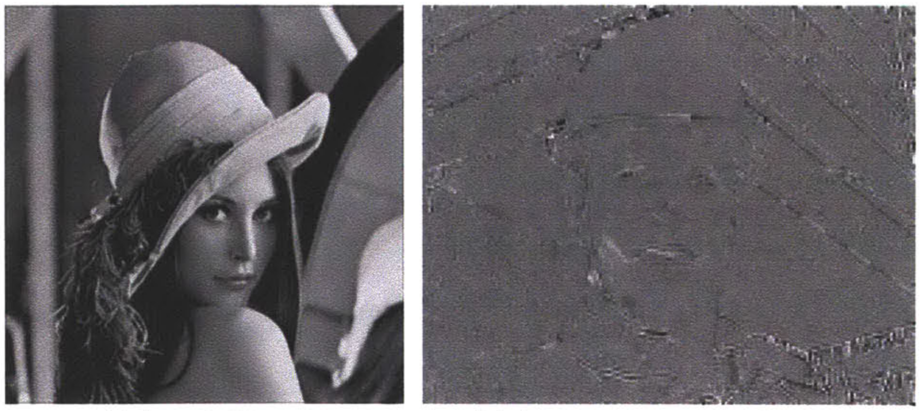

(a) A typical image (b) A motion-compensated residual

Figure 1-4: An example of a typical image and motion compensated residual. The typical image shown here has many visible two-dimensional structures. While it also has one-dimensional structures, these are different from the one-dimensional structures present in motion-compensated residuals. This is mainly because most of the image intensities in the motion-compensated residual are zero-valued (shown in grey color) and the one-dimensional structures display non-zero intensities. In typical images, one dimensional structures are generally separating two areas with different intensities.

transform for this block depends on the number of transform coefficients selected. However, the coefficients selected for a specific block depend on the coefficients selected in other blocks due to the total number of coefficients constraint. This implies the best transform for this block depends on the transforms used in other blocks. As a result, the optimal transform for each block cannot be determined independent of other blocks.

A brute force method that considers all possible transform combinations for all the blocks may conceptually solve this problem. For example, if we have two transforms A and B for each block and we have two blocks, we may consider using transforms A for both blocks, denoted as combination AA. The computation of preserved energy with given number of coefficients follows and this is compared with the cases using the other three transform combinations AB, BA and BB. However, we note that the number of possible combinations is the number of transforms to the power of the number of blocks, which is very large even for the simplest case of two transforms. Some recent studies [2] have considered a set of as many as 17 different transforms that can be chosen for each block. This implies that brute force method is not practical. As a result, effective methods to evaluate the energy compaction with multiple transforms need to be developed.

1.3 Overview of Thesis

In Chapter 2, we review some recent research on the development of multiple transforms in video compression applications. Specifically, we observe that these transforms are selected and tested by incorporating them in video compression systems. We point out that this methodology is tedious and may not be efficient when a very large number of transforms are used. We then propose an approach to select transforms based on the energy compaction property.

Evaluating the energy compaction in multiple-transform environment is a difficult task. In Chapter 3, we develop two algorithms to solve this problem. The first algorithm is computationally and conceptually simple, but leads to a locally optimal solution. The second algorithm is more computationally intensive but leads to a globally optimal solution. We describe the algorithms and analyze their performance in theory.

In Chapter 4, the performance of the two algorithms is tested with practical settings. The energy compaction results are compared with the rate-distortion performance under similar settings.

In Chapter 5, we demonstrate the power of the algorithms when a very large number of transforms need to be tested and analyzed. We evaluate these transforms by investigating the energy compaction of some transform sets.

In Chapter 6, we make conclusions and discuss the future research.

Chapter 2

Previous Research

2.1 Direction-Adaptive 1D DCTs

Energy compaction occurs due to the decorrelation of signals with proper transforms. For a specific signal, some information of the current signal value may be obtained observing previous or future values of that signal. This is referred to as the correlation within a signal. In the example shown in Section 1.1, the time domain representation of a signal fails to consider the correlation between different coefficients. In contrast, the DCT has the ability to explore the global pattern in the signal and effectively utilizes the information provided by the correlation within the signal. This functionality leads to a very effective representation of the signal in the DCT domain.

Similar to DCT used in the above example, many transforms have the ability to explore the correlation within a signal. Some transforms are designed in such a way that they minimize the correlation. A statistical model that characterizes the correlation within a category of signals is first developed. Based on this model, the transform that minimizes the correlation in theory is developed. For example, Markov-1 process is used extensively to model images. This model leads to an auto-covariance function of images signals, defined as:

$$C(I, J) = \rho_1^{|I|}\rho_2^{|J|}$$

where I and J represents the horizontal and vertical distance between image pixels, and ρ_1 , ρ_2 represent the correlation parameters along horizontal and vertical direction, respectively. The 2D-DCT is then developed by minimizing the correlation of images given this auto-covariance function, with an additional assumption that the correlation parameters along both directions are close to one. The Markov-1 model that characterizes the statistics of images approaches typical images well enough in practice. As a result, the 2D-DCT demonstrates fairly good performance when decorrelating the image signals, and thus is useful in image compression. This is consistent with the observation that the most energy of an image signal can be preserved with a few DCT coefficients.

It was generally believed that the motion-compensated residuals share the same statistics with typical images. Therefore, the 2D-DCT has been extensively used as a residual transform in video coding system such as H.264 [4]. Recently, it has been observed that the statistics of motion-compensated residuals is different from that of typical images. Specifically, the work in [2] reports that the correlation of motion-compensated residuals can be better characterized as:

$$C(I, J, \theta) = \rho_1^{|I\cos\theta + J\sin\theta|}\rho_2^{|-I\sin\theta + J\cos\theta|}$$

where the additional parameter θ represents the directionality of the correlation. This model can be interpreted as the Markov-1 model rotated by a degree θ . Based on this new model, the correlation parameters ρ_1, ρ_2 have been estimated. The result implies one correlation parameter approaches zero while another approaches one for typical residual signals. In other words, for a residual signal, the correlation may be strong only along a certain direction, meaning these one-dimensional structures may appear more frequently in residual signals.

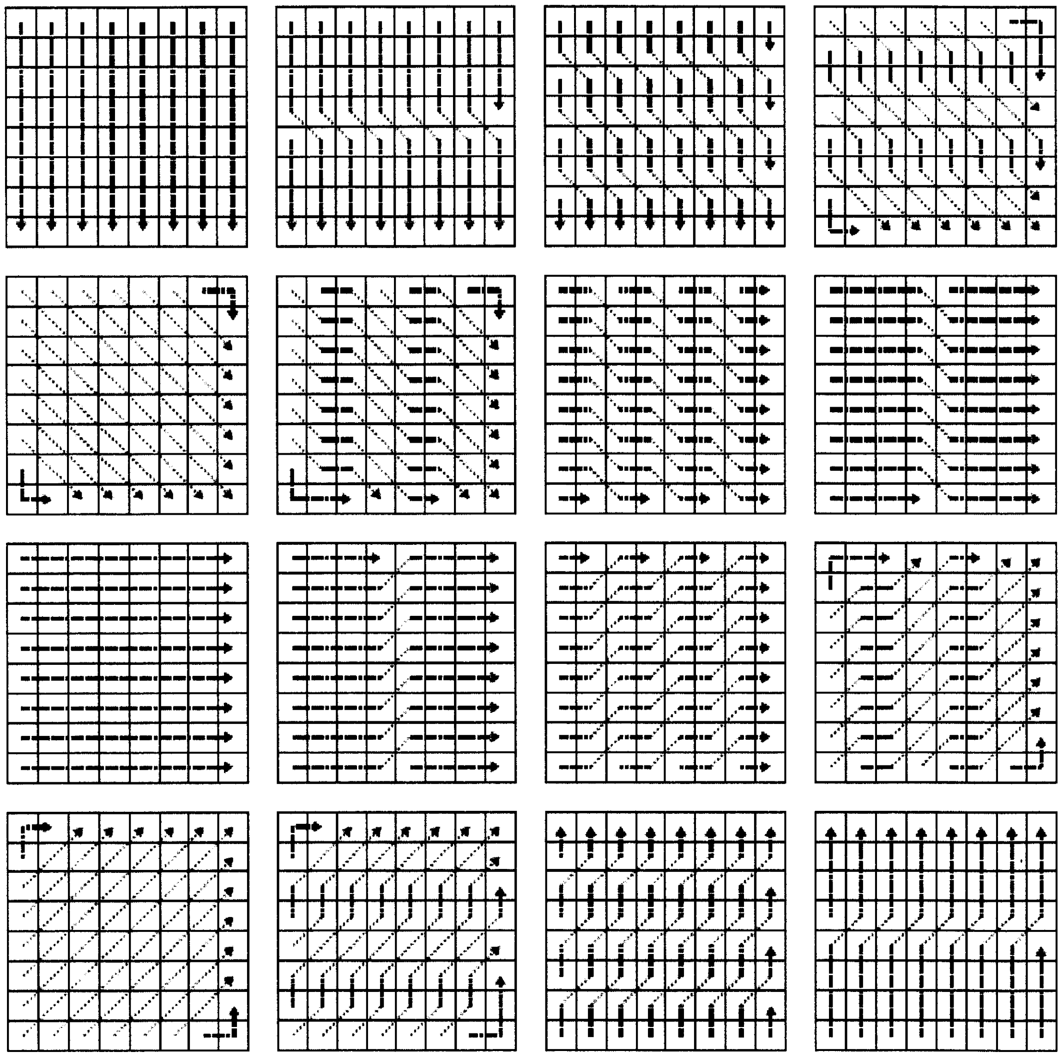


Figure 2-1: Sixteen 1D-DCTs of block size 8x8.

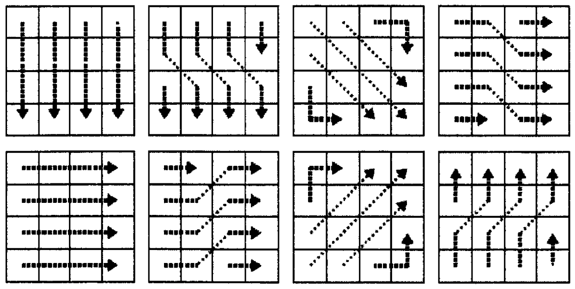


Figure 2-2: Eight 1D-DCTs of block size 4x4.

Based on this observation, a set of 1D-DCTs have been developed to exploit the 1-D structures present in the residual signals. Figure 2-1 shows sixteen block 1D-DCTs of size 8x8. Figure 2-2 shows eight block 1D-DCTs of size 4x4. The DCT is only applied in one direction along the lines shown in the figures. These transforms cover many possible 1D directions within a block.

These transforms have been incorporated into the H.264 system and their performance is verified with rate-distortion metric. The simulation results have shown that the rate-distortion performance increases significantly with these additional 1D-DCTs [2]. Further analysis of the 1D-DCTs is investigated in [5], where only vertical and horizontal 1D-DCTs are used instead of 1D-DCTs of many different directions. Simulations are performed with three different transform settings: 2D-DCT only, 2D-DCT with all 1D-DCTs, and 2D-DCT with only vertical and horizontal 1D-DCTs. By comparing the rate-distortion performance of three different settings, it is reported that for typical video sequences, much bit saving is due to the use of only vertical and horizontal 1D-DCTs.

2.2 Rate-Distortion Performance of 1D-DCTs

In video coding systems, the transform performance is evaluated using the rate-distortion metric. It measures the distortion of the reconstruction of a video sequence from a coded bit stream as a function of the bitrate. When multiple transforms are incorporated into a video coding system, it is necessary to choose the correct transforms that optimize the coding performance. In other words, when we allow each coding block to use a different transform, each different choice of transform will produce a different bit stream. Among these bit streams, there is a specific one that introduces least distortion given a specific bit rate, which is considered optimal.

In practice, there are many algorithms that can help to choose the optimal transforms. In [2], the Lagrangian multiplier method is used. In this method, the rate profile associated with each transform is designed according to some proper choice of quantization and entropy coding. Then for each block, a cost function $f(i) = R_i + \lambda D_i$

for all possible i is computed, where i represents the index of a specific transform, and R_i, D_i represent the rate and distortion, respectively, using transform i . The parameter λ is referred to as the multiplier, which is a fixed positive parameter that controls the video quality. Intuitively, a small λ will almost ignore the distortion. A larger λ instead, tends to minimize the distortion regardless of the bitrate. By choosing a proper λ , one may balance the tradeoff between minimizing the rate and distortion. Finally, for all these transforms, the one that leads to the minimum cost function $f(i)$ is chosen as the optimal transform for the block. For more details, readers are referred to [6].

The Lagrangian multiplier method is proved to be an optimal solution in the sense that it can choose the optimal transforms that lead to the optimal rate-distortion performance, given transforms and corresponding codewords. Still, well-designed codewords for transforms are necessary in order to achieve desirable performance. Poorly designed codewords will severely degrade the performance, even if the transforms are correctly chosen. Meanwhile, designing these codewords in terms of quantization and entropy coding is quite involved. In [2], the codewords are designed simply by modifying the existing quantization and entropy coding within H.264 system. Still, the rate-distortion performance increases significantly, indicating that these transforms work well on the residual images. The potentials of these transforms can be better explored if the codewords are more carefully designed.

2.3 Motivation for Thesis

As discussed in the previous section, in the work of [2], a set of transforms that appear reasonable were first determined. For this given set of transforms, their rate profiles which depend on the quantization and entropy coding methods were obtained and incorporated into a practical H.264 system, which is a tedious process. This type of process is, of course, necessary when we have determined a specific set of transforms and wish to incorporate them into a specific video compression system.

Suppose we have a large number of possible sets of transforms. Evaluating the performance of each possible set in a video compression system can involve a great deal of efforts. In the end, we may find that only a small number of the many transforms are useful and are incorporated into the video coding system. Evaluating the transform performance based on the rate-distortion performance relies on well-designed codewords, so that the potentials of these transforms will be fully explored. The work involved in designing the codewords for the many transforms that are not used is quite wasteful.

In this thesis, we evaluate the performance of a set of transforms based on the energy compaction capability. The energy compaction indicates the performance of the transforms independent of how the codewords are designed. It is true that eventually the performance of these transforms needs to be tested in a practical coding system, we can still use this method to screen out some transforms that may not have any potential to increase the performance. By doing so, the codeword designing workload will be significantly reduced. This approach is more effective especially when a large number of transforms are involved.

Chapter 3

Algorithms

Suppose we have a given set of transforms. In this chapter, we develop two algorithms that choose for each block the best transform within the set and the best transform coefficients to preserve the largest amount of energy for a given total number of transform coefficients to be selected.

This chapter is organized as follows. We first formulate the energy compaction evaluation as a maximization problem. Then we develop two algorithms to solve this problem. The first algorithm is an iterative procedure that consists of two steps in each iteration. The first step chooses the best transform for each block given the number of coefficients used in that block. The second step chooses the best coefficients in each block that lead to the optimal energy preservation, given the chosen transforms. The second algorithm computes the block optimal energy function, which contains all the information necessary to obtain the optimal solution. The best coefficients and transforms are determined by minimizing some cost function associated with the block optimal energy function. For each of the two algorithms, we analyze their performance in theory. We summarize this chapter by comparing their performance.

3.1 Problem Formulation

Suppose a signal is segmented into N blocks, indexed by $n = 1, 2...N$. Without loss of generality, we assume that each block has M data point. For each block, we can choose from K candidate transforms.

The preserved energy is a function of two factors. The first factor is the transform used for each block, which we denoted as $\mathbf{T}$. Once the transforms are fixed, we can choose some transform coefficients to preserve the energy, which leads to the second factor. The second factor is the coefficients selected in each block, which we denote as $\mathbf{C}$. Note that in order to preserve the maximal amount of energy, we should only choose largest coefficients in a block. As a result, these coefficients can be indicated by the number of coefficients used in each block. Note that both $\mathbf{T}$ and $\mathbf{C}$ consider the transforms and numbers of coefficients chosen in N blocks. As a result, they are vectors of length N. The energy preserved is denoted as $E(\mathbf{T}, \mathbf{C})$

Our notation can be illustrated with an example. Suppose $N = 2$, $M = 3$ and $K = 4$,which corresponds to two blocks with size three, and we can choose from four candidate transforms. $\mathbf{T} = (2, 4)$ and $\mathbf{C} = (1, 3)$ means that for these two blocks, we use transform 2 for the first block and choose one largest coefficient, and we use transform 4 for the second block and choose three largest coefficients. In this example, we use a total of four coefficients to preserve the energy.

With this notation, our problem can be formulated as follows:

$$\begin{array}{ll} \textit{maximize} & E(\mathbf{T}, \mathbf{C}) \\ \textit{subject to} & |\mathbf{C}|_1 = C_0 \end{array}$$

where $|\mathbf{C}|_1$ is the 1-norm of $\mathbf{C}$, representing the total number of coefficients used, which equals the given total number of coefficients constraint C_0.

The notation used in this section will be used throughout the rest of this thesis.

3.2 Algorithm A

3.2.1 Algorithm Description

The first algorithm, denoted as Algorithm A, is based on two observations. In one observation, when the best transform for each block is given, the transform coefficients can be optimally chosen by selecting in the order of the largest magnitude transform coefficients until the given total number of coefficients is selected. In the other observation, if we know the number of transform coefficients to be selected for each block, choosing the optimal transform for each block is straightforward. For each block, we can compute the transform coefficients for each possible transform, choose the known number of coefficients starting from the largest magnitude coefficient, and choose the transform that preserves the largest energy for that block.

These two observations suggest an iterative procedure. In each iteration, we choose the best transform for each block from the most recent number of coefficients selected for the block, and then choose the best set of transform coefficients from the most recent transforms chosen for all the blocks.

If we denote the transforms of the *ith* iteration as $\mathbf{T}_i$, and the numbers of coefficients used as $\mathbf{C}_i$, each iteration consists of two steps described below:

Step A1: From Step A2 of the previous iteration $i-1$, we are given the number of coefficients $\mathbf{C}_{i-1}$ used in each block. Based on this, we update the transforms in current iteration $\mathbf{T}_i$ as:

$$\mathbf{T}_i = \underset{\mathbf{T}}{\operatorname{argmax}}\, E(\mathbf{T}, \mathbf{C}_{i-1})$$

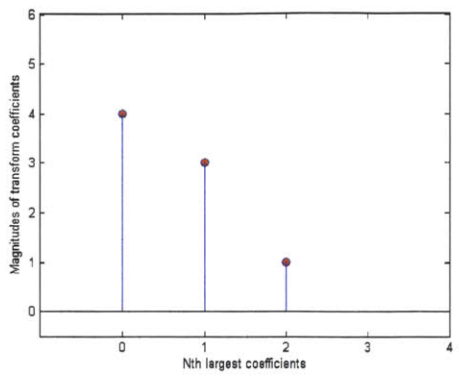
Since the number of coefficients used in each block is given, this optimal transform selection can be carried out in each block independently by comparing the energy preserved with this specific number of coefficients over all possible transforms.

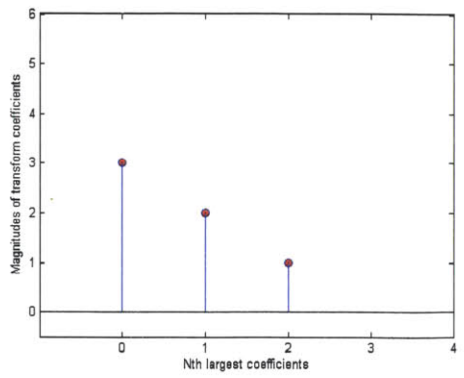
Step A2: From Step A1, we are given the transforms selected in the current iteration, denoted as $\mathbf{T}_i$. We compute the transform coefficients $\mathbf{C}_i$ for all the blocks based on the chosen transforms as:

$$\mathbf{C}_i = \underset{\mathbf{C}}{\operatorname{argmax}}\, E(\mathbf{T}_i, \mathbf{C})$$
$$\textit{subject to} \qquad |\mathbf{C}|_1 = C_0$$

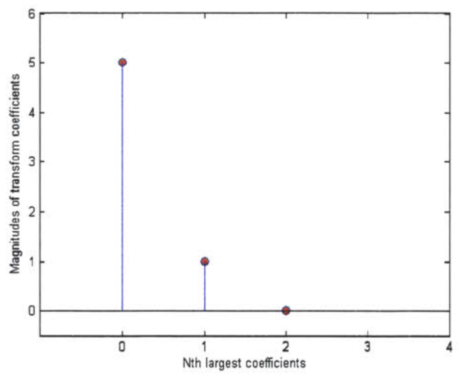
This maximization can be performed by selecting the largest C_0 coefficients in all blocks. Then we trace these selected coefficients back to the blocks where they belong, and these numbers of coefficients selected are used as the $\mathbf{C}_i$ in the next iteration.

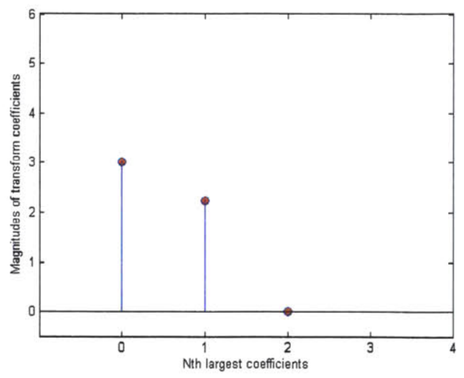
To illustrate the algorithm, we show the process of one iteration with an example. Suppose we have a signal that has two blocks of length three, which are transformed by two candidate transforms. We consider preserving the energy with two coefficients. In other words, $N = 2$, $M = 3$, $K = 2$ and $C_0 = 2$. Figure 3-1 shows the magnitudes of coefficients when two different transforms are applied on two blocks. Suppose we initiate the iteration by choosing two coefficients from the first block and nothing from the second block, which means $\mathbf{C}_{i-1} = (2, 0)$. In Step A1, by comparing Figures 3-1a and 3-1c, we find that for the first block, the first transform preserves energy 25 while the second transform preserves energy 26 with two coefficients. Therefore, we decide to choose the second transform for the first block. For the second block, we can choose either transform since in both Figures 3-1b and 3-1d, zero energy is preserved. Then, we decide to choose the first transform. In this case, we get $\mathbf{T}_i = (2, 1)$ after Step A1 is finished. In Step A2, with the selected transforms, we choose two largest coefficients from 5, 3, 2, 1 and 1. This is achieved by choosing one coefficient 5 from the first block and one coefficient 3 from the second block, resulting in $\mathbf{C}_i = (1, 1)$. The preserved energy is now 34, which is larger than 26. We can verify that the algorithm converges at this point, by repeating the above iteration process. In this example, we can verify that this result is optimal by comparing the results with enumerating all transform combinations. For the performance of the algorithm A in general cases, further analysis is carried out in the following sections.





(a) Block 1 with Transform 1: Coefficients: 4,3,1 (b) Block 2 with Transform 1: Coefficients: 3,2,1





(c) Block 1 with Transform 2: Coefficients: 5,1,0 (d) Block 2 with Transform 2: Coefficients: $3,\sqrt{5},0$

Figure 3-1: An example for iteration process of Algorithm A.

3.2.2 Convergence

The Algorithm A discussed in Section 3.2.1 can be shown to converge, under any initial conditions. From the computation in Step A1, $E(\mathbf{T}_i, \mathbf{C}_{i-1}) \geq E(\mathbf{T}_{i-1}, \mathbf{C}_{i-1})$. This is because the transforms are selected such that the resulting energy is maximized when $\mathbf{C}_{i-1}$ is given, thus larger than the energy when any other transform combination is used. Similarly, from Step A2, $E(\mathbf{T}_i, \mathbf{C}_i) \geq E(\mathbf{T}_i, \mathbf{C}_{i-1})$ since the chosen transform coefficients are the largest ones that lead to the maximal amount of preserved energy. From both inequalities, we obtain $E(\mathbf{T}_i, \mathbf{C}_i) \geq E(\mathbf{T}_{i-1}, \mathbf{C}_{i-1})$, which implies a non-decreasing sequence $E(\mathbf{T}_i, \mathbf{C}_i)$ with respect to i. In addition, this sequence is upper bounded by the total energy of the signal. Therefore, a monotonically non-decreasing and bounded sequence must be convergent. This result indicates that Algorithm A is guaranteed to converge in terms of the preserved energy. Notice that this result is consistent with the example in Figure 3-1, while the preserved energy is actually increasing as the iteration proceeds.

3.2.3 Initial Conditions and Local Convergence

While this algorithm is guaranteed to converge, it may be sensitive to initial conditions. In other words, this algorithm may be trapped at different local maxima when different initial conditions are used. To show this point, we consider the energy map with respect to both $\mathbf{C}$ and $\mathbf{T}$.

For a specific transform combination $\mathbf{T}^*$, we first ignore the constraint $|\mathbf{C}|_1 = C_0$. The unconstrained energy $E(\mathbf{T}^*, \mathbf{C})$ can be expressed as the sum of the energy in each block, denoted as $E_i(\mathbf{T}^*, \mathbf{C}), i = 1, 2...N$. It can be seen that $E_i(\mathbf{T}^*, \mathbf{C})$ is a concave function with respect to the number of coefficients used in that block. This is because the incremental energy when one more coefficient is used is non-increasing. Therefore, $E(\mathbf{T}^*, \mathbf{C}) = \sum_{i=1}^{N} E_i(\mathbf{T}^*, \mathbf{C})$ is also concave since it is the sum of several concave functions. Now if we consider the constraint $|\mathbf{C}|_1 = C_0$, the result is the intersection of the unconstrained concave energy function with a affine linear subspace described as $|\mathbf{C}|_1 = C_0$, which is still a concave function. Based on this argument, we conclude that Step A2 finds the optimal solution on a concave surface. For a specific coefficient distribution $\mathbf{C}^*$, Step A1 searches the highest energy for all different possible transform combinations $\mathbf{T}$.

Graphically, the process of Algorithm A is illustrated in Figure 3-2. In this figure, we plot the energy $E(\mathbf{T}^*, \mathbf{C})$ with respect to the coefficient distribution $\mathbf{C}$. Each curve represents the energy function of a specific transform combination $\mathbf{T}^*$. Note that in practice, there are a very large number of curves and $\mathbf{C}$ is multi-dimensional. We simplify the figure just to show the idea.

Suppose we start from the state A and perform the iteration Step A2. Step A2 searches on the concave surface until it reaches the maximal point B. Then Step A1 checks all possible transform combinations and reaches the maximal point C where

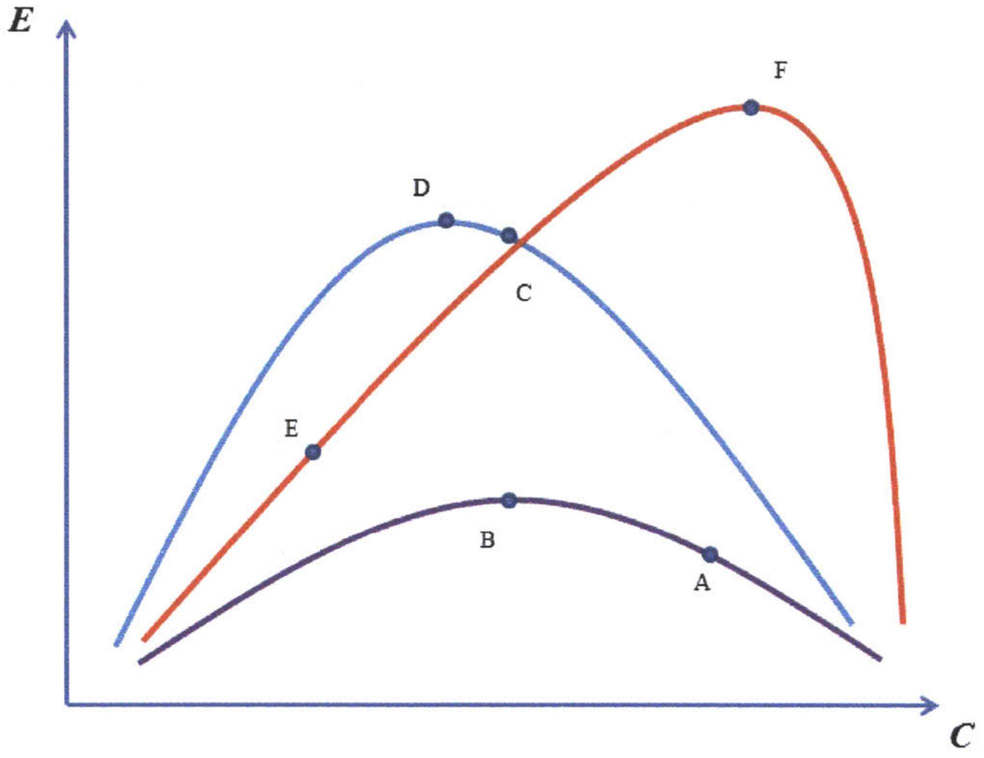


Figure 3-2: Graphical illustration of the iteration process of Algorithm A.

the number of coefficients used is fixed. The iteration stops when it reaches D, which is a locally maximal solution. However, as we can see, D is different from the globally optimal solution F, which implies that the algorithm is trapped at a local maximum. If the algorithm starts from E and Step A2, with one step we reach the global optimum F. We can see from this argument that the convergence may be sensitive to initial conditions.

The best case for this algorithm is that all the locally optimal solutions are the same. In this case, the algorithm converges to the global optimum under any initial conditions. Even if they do not, the locally optimal solution may be very close to the globally optimal solution. Since the structure of the energy function is dependent on the specific signal to be evaluated, analyzing the structure in theory is difficult. Instead, we test this property in the next chapter, by comparing the locally optimal solution with the globally optimal solution obtained from the other proposed algorithm.

In practice, there are many possible choices of initial conditions. For example, one may specify certain transforms, such as 2D-DCT for each block, and start from Step A2. One may also use an equal number of coefficients in each block and start from Step A1. The results from our simulations in the next chapter indicate that the preserved energy after convergence is not sensitive to reasonable choices of initial conditions.

3.2.4 Computational Complexity

The overall computational complexity depends on the number of iterations. In practice, for typical image and video signals, the convergence occurs within several iterations, as shown in the next chapter. The computational complexity of one iteration can be roughly estimated as follows. We make the assumption that each transform is comparable in complexity. With the Big O notation which evaluates the asymptotic complexity, the computational complexity in Step A1 is $O(KN)$, when all transforms are performed and compared once in each block. For Step A2, we need to find the largest C_0 coefficients. This can be accomplished by sorting the coefficients with a complexity of $O(NM \log NM)$, where NM equals the number of pixels. This complexity, however, can be improved using the median-of-medians algorithm [7], where the complexity of the worst case is $O(NM)$ that is linear to the number of pixels. Furthermore, we note that for those blocks for which the number of coefficients chosen or the transform chosen does not change from a prior iteration, the computation can be reduced by using the results from the prior iteration. Since the majority of blocks remain to use the same transform, the intensive computations needed in computing the transforms can be drastically reduced. In conclusion, with these optimizations on the implementation of Algorithm A, it becomes a very efficient algorithm computationally.

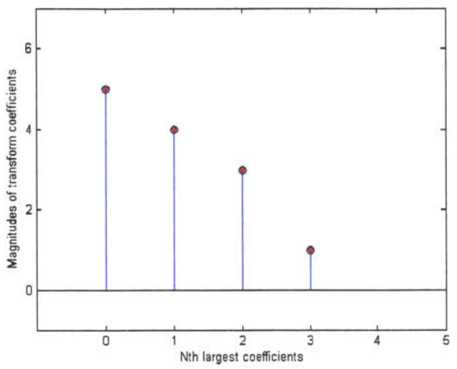
3.3 Algorithm B

3.3.1 Algorithm Description

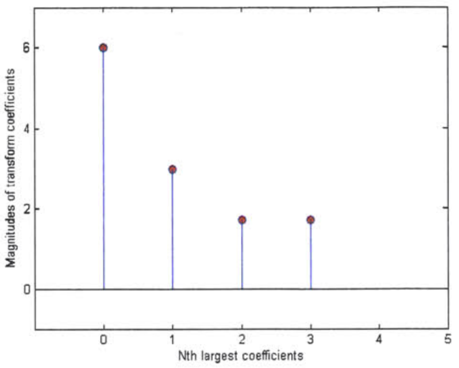
The second algorithm, denoted as Algorithm B, finds the globally optimal solution in two steps. The first step finds the optimal energy function for each block. The second step searches for the optimal solution.

In the first step, to compute the block optimal energy function, we consider a single block. For this block, when the number of coefficients used in this block is fixed, choosing the optimal transform that gives the highest energy is straightforward using the same procedure in Step A1 of Algorithm A. By varying the number of coefficients used in this block, we can determine the optimal energy preserved as a function of the number of coefficients. We will refer to this function as the block optimal energy function and denote it as $E(c)$, where c represents the number of coefficients. Note that the chosen transforms in this block optimal energy function is the only possible ones that will be selected in the optimal solution. This is because once the optimal number of coefficients used in this block is obtained in some way, the corresponding chosen transform gives the highest possible energy. As a result, it is clear that this block optimal energy function carries all the information that will be used to obtain an optimal solution.

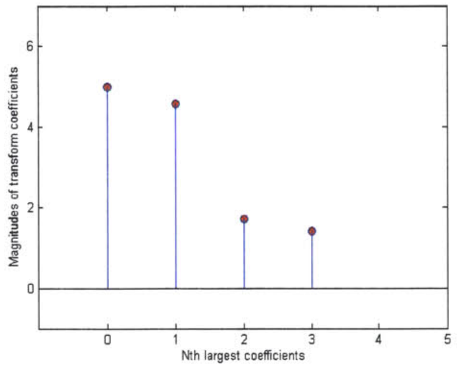
Figure 3-3 shows an example of computing the block optimal energy function. A block is transformed by three transforms T_1, T_2 and T_3 with length four, which result in the transform coefficients (5,4,3,1), $(6,3,\sqrt{3}\ ,\sqrt{3}\ )$ and $(5,\sqrt{21}\ ,\sqrt{3}\ ,\sqrt{2}\ )$, shown in Figures 3-3a, 3-3b and 3-3c respectively. To compute the block optimal energy function, we start from $c = 0$. For this case, we decide not using anything to preserve the energy, and the energy preservation is 0. Next, we can preserve the energy, with one coefficient, of 25,36,25 for these three transforms. Obviously, we should choose T_2 to obtain the maximal amount of energy 36. If we preserve the energy with two coefficients, we sum up the square of the magnitudes of two largest coefficients for three transforms, which result in the energy preservation of 41,45 and 46. As a result, the optimal energy is 46 and the best transform T_3 is selected. In the same manner, when three and four coefficients are used, the optimal energy is 50 and 51, and the best transform is T_1 and T_1 (or T_2 , T_3), respectively. The block optimal energy function for this specific block is given in Figure 3-3d.



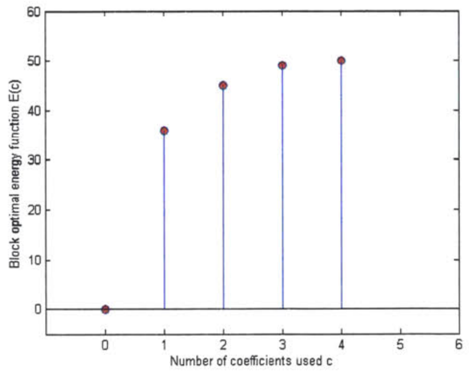
(a) Transform coefficients of T1



(b) Transform coefficients of T2



(c) Transform coefficients of T3



(d) Block optimal energy function

Figure 3-3: An example of computing the block optimal energy function.

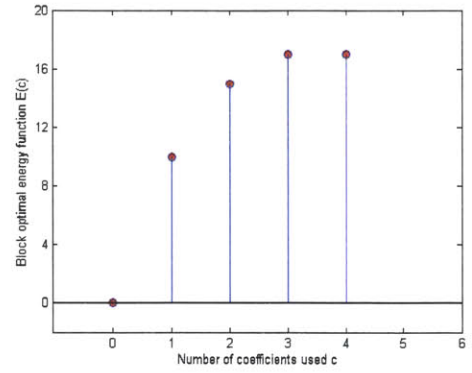
In the second step, we use a method to determine the optimal number of coefficients used in each block. In this method, we first fix a positive parameter λ and then minimize the cost function $f(c) = c - \lambda^{-1}E(c)$ for each block. The parameter c is the number of coefficients (from zero to the size of the block) and $E(c)$ is the block optimal energy function. The number of coefficients that leads to the minimum $f(c)$ is chosen as the optimal number of coefficients used in the block.

To verify that we can obtain the optimal number of coefficients used in each block with this method, we first show that this is true for a special case when all block optimal energy functions are concave, where λ has an intuitive explanation related to the incremental energy. We then extend to the case where all block optimal energy functions are not concave.

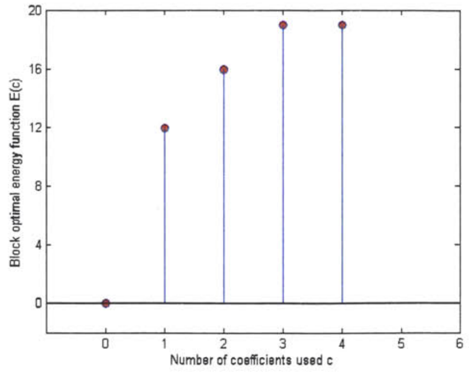
The case when all the block optimal energy functions are concave

When all the block optimal energy functions are concave, we consider the incremental energy when one more coefficient is added. If the incremental energy is maximized for each new coefficient added among all the remaining coefficients in all the blocks, the cumulative energy is also maximized. This can be accomplished by computing the incremental energy from each block optimal energy function and selecting all the coefficients in all the blocks that contribute the incremental energy above a certain threshold λ. All the selected coefficients can be traced back to the blocks where they come from and the optimal number of coefficients used in each block can be obtained. It will be shown that the parameter λ used in the cost function $f(c)$ represents the incremental energy threshold.

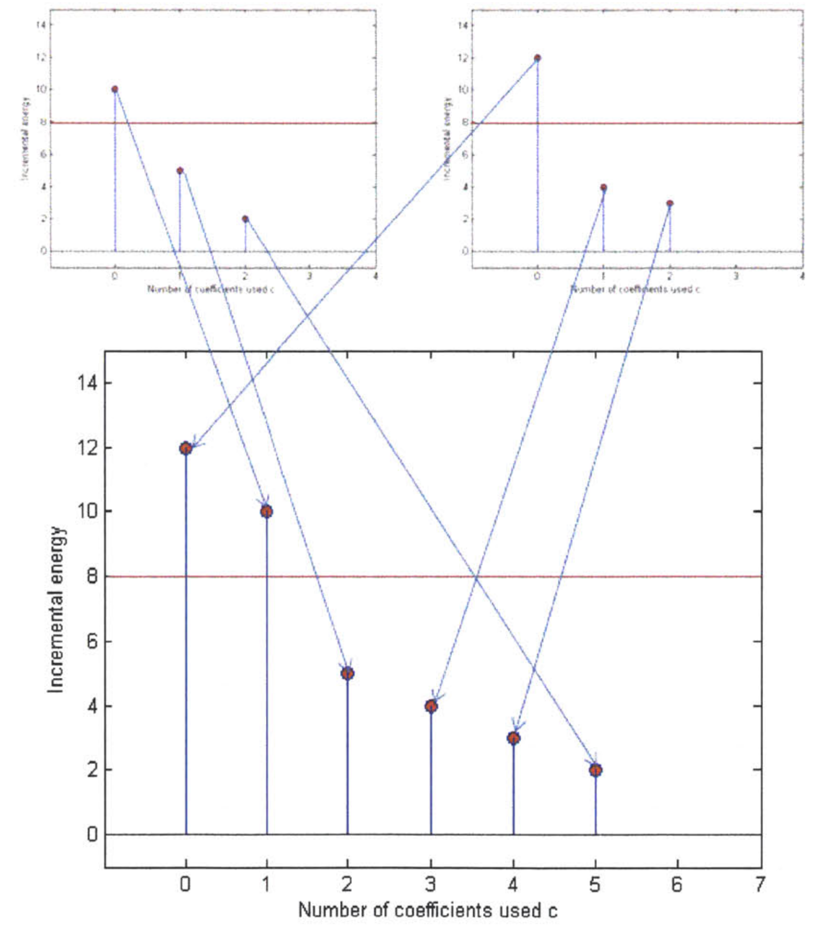
This maximization process can be illustrated with Figure 3-4. Figures 3-4a and 3-4b show optimal energy functions of two blocks. The incremental energy where one more coefficient is added in each block is shown in the left upper and right upper of Figure 3-4c. The incremental energy of all blocks can be aligned together, as is shown in the lower part of figure (c). The mapping of the incremental energy in each block to the overall incremental energy is illustrated with arrows. By decreasing the energy threshold λ from 12 to 0, we can select the coefficient one by one that has the largest incremental energy in the rest of unselected coefficients. Since when all the block optimal energy functions are concave, the incremental energy is decreasing with respect to the number of coefficients used. Therefore, no smaller incremental energy will be chosen ahead of a larger one.



(a) Block 1 Optimal Energy Function



(b) Block 2 Optimal Energy Function



(c) Incremental energy in each block and its mapping to the overall incremental energy

Figure 3-4: Illustration of thresholding the incremental energy.

When a λ is fixed, we can choose some optimal energy and the corresponding optimal numbers of coefficients used in the blocks. For example, the horizontal lines in Figure 3-4c correspond to the energy threshold λ of 8. It can be seen that these two selected coefficients are distributed to both blocks using the same threshold. In this case, the optimal energy is 22 and each block will use one coefficient. This is indeed the highest amount of energy that we can preserve with two coefficients.

We now relate the process discussed above with minimizing the cost function $f(c)$. Specifically, we discuss how the incremental energy thresholding can be related with minimizing the cost function. We consider selecting all the incremental energy within a specific block that are larger than λ. This can be achieved by computing the incremental energy from the block optimal energy function as $E(c+1) - E(c)$. Another approach is to directly consider $E(c)$. Figure 3-5 shows a concave block optimal energy function, which is linearly interpolated. It is clear that the incremental energy when a certain number of coefficients c is used is the slope of segment right to this point. In order to choose the coefficients that have incremental energy larger than λ, we can instead determine the point where the slope of left segment is larger than λ and that of right segment is smaller than λ. This point can be determined by pushing a line with slope λ until it is tangent to the interpolated function. Equivalently, pushing the line corresponds to make the intersection of the y-axis and this line as large as possible when this line is intersecting with the function. From a simple computation, we can see the intersection of this line with y axis is $E(c) - \lambda c$. Maximizing $E(c) - \lambda c$ is the same as minimizing $c - \lambda^{-1}E(c)$ with respect to all c. This minimizes the cost function $f(c) = c - \lambda^{-1}E(c)$ discussed above.

The case when all the block optimal energy functions are not concave

When all the block optimal energy functions are not concave, the above discussion based on the concavity of the functions case does not apply. However, every non-concave block optimal energy function can be modified to a concave function in the following way. We consider the convex hull of the block optimal energy function. If some values do not lie on the convex hull, we modify the values of those points to the

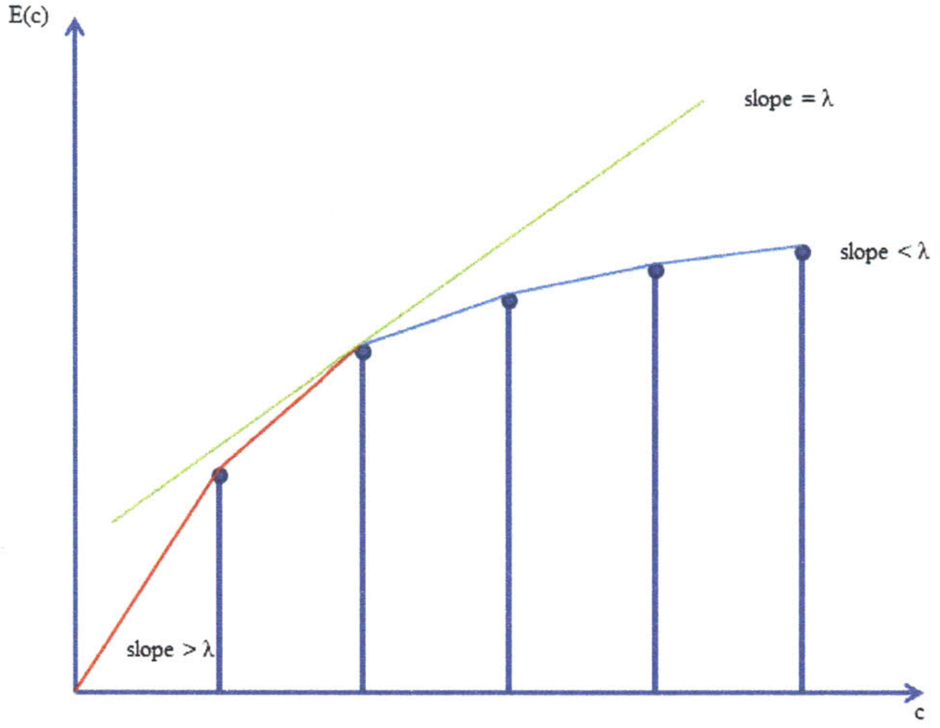


Figure 3-5: Illustration of minimizing the cost function.

convex hull. This is illustrated in Figure 3-6. Note that this function is not concave due to the point lying under the convex hull, which is outlined in this figure. We modify this point straight up to the convex hull, as marked by the arrow. When all these non-concave block optimal energy functions are modified to the concave ones, we can apply the method discussed above to the modified problem where all the block optimal energy functions are replaced by the concave ones.

We now show that these modifications of block optimal energy functions do not affect the optimal solution obtained by using the minimization method discussed above. We first observe that the values that have been modified increase due to the property of convex hull. Intuitively, a sequence point either lies on the convex hull or lies under the convex hull. If it lies above the convex hull, it means that the convex hull we found is not correct. Instead, it should be modified to include the one that lies above. This implies that the optimal energy E^* we obtain in the modified version is not smaller than that of the original E. In other words, E^* is an upper

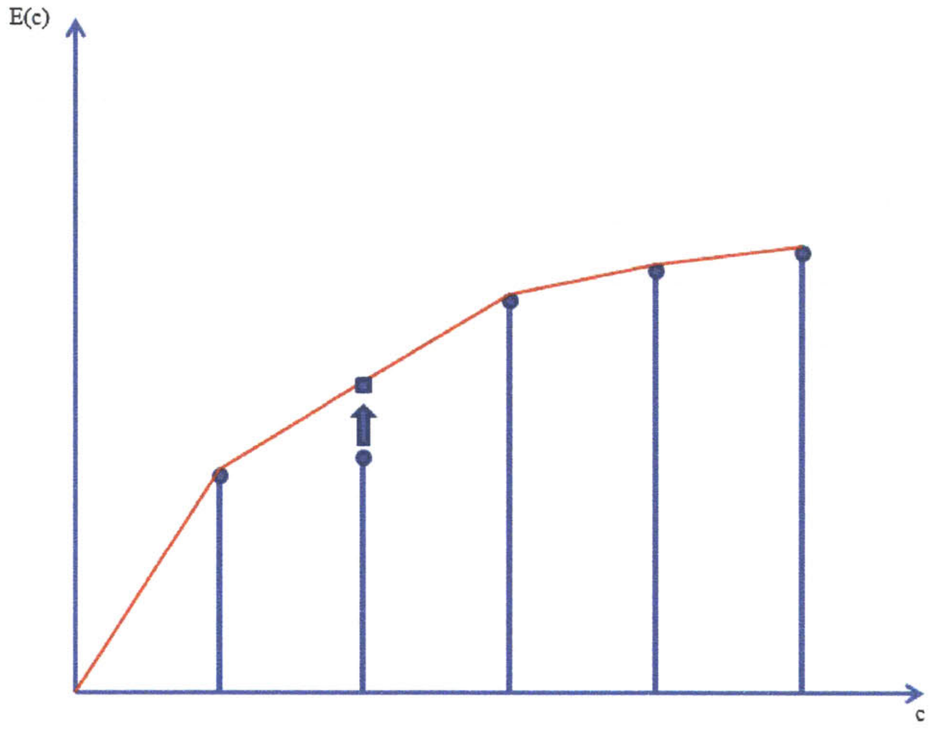


Figure 3-6: Modification of the non-concave block optimal energy function.

bound of the optimal solution for the original problem. Another observation is that the optimal solution obtained in the modified problem is a feasible solution in the original problem. This can be seen from the following argument. By pushing a line with a certain slope λ, only the points that lie on the convex hull can be reached. Therefore, those modified points will never be included into the optimal solution since they can never be reached by the proposed method. This means the optimal numbers of coefficients obtained will not be affected by modifying those points that lie under the convex hull. As a result, the upper bound E^* is actually feasible in the original problem. We can conclude that the minimization process discussed accomplishes the optimal number of coefficients used in each block.

We also refer to [6] for complement, since the rate-distortion optimization discussed in [6] has a similar mathematical structure with our method. In addition, it provides an alternative explanation based on the Lagrangian multiplier method. The relationship between these two explanations is quite involved. An intuitive view is that Lagrangian multiplier method is actually derived from the duality theory, where the dual variables consider whether the optimal solution is sensitive to the infinitesimal change of some prime variables. In our explanation, we show that the optimal solution is not sensitive to the sequence points that lie under the convex hull but is sensitive to the sequence points that lie on the convex hull.

To conclude, we outline the algorithm steps of Algorithm B:

Step B1: Compute the block optimal energy function $E(c)$ for each block. For each block, we compute the optimal energy among the transforms within the transform set, as a function of the number of coefficients (from 0 to the size of the block M).

Step B2: Choose a fixed positive parameter λ. For each block, compute $f(c) = c - \lambda^{-1}E(c)$ where c is the number of coefficients and $E(c)$ is the block optimal energy function. Choose the number of coefficients that minimizes $f(c)$ as the chosen number of coefficients used in this block. Then the optimal transform for that block is the one that leads to the highest energy when the number of coefficients is given.

3.3.2 Global Optimality

As shown in the previous section, Algorithm B finds the globally optimal solution. This means for the number of coefficients used in Algorithm B, the resulting energy is the highest among all possible transform combinations and used coefficients. This global optimality is not guaranteed in Algorithm A, when the algorithm may fall into the local maximal trap. The global optimality of Algorithm B can be also be observed from the following chapter, where Algorithm B preserves higher energy relative to Algorithm A in the simulations.

3.3.3 The difficulty of specifying the total number of coefficients

The total number of coefficients used is explicitly specified in Algorithm A, while it is controlled by λ in Algorithm B. The relationship between the number of coefficients used and λ is complicated and cannot be determined explicitly. This brings another issue about how to choose the correct λ, which is out of the scope of this thesis. One possibility is that in order to use some specified number of coefficients, one may iterate Step B2 back and forth among different λ values to match the desired number of coefficients. This method is used in this thesis. Furthermore, the possible range of λ does not cover all possible total numbers of coefficients. This is because some total numbers of coefficients may be related to those points that lie under the convex hull, which can not be reached by this method. In other words, the optimal solution cannot be obtained for a certain set of total numbers of coefficients for any choice of λ.

Despite this limitation, Algorithm B is useful in verifying the convergence quality of Algorithm A. Specifically, we can specify the total number of coefficients in Algorithm A to exactly match that generated by a particular λ in Algorithm B, and see how close their solutions are. This method is used in the next chapter to evaluate the performance of Algorithm A.

3.3.4 Computational Complexity

For each block, in order to determine the block optimal energy function in Step B1, each transform needs to be performed once and compared when each number of coefficients is used. This results in $O(KM)$ tests for one block. For all the N blocks, Step B1 has a complexity of $O(NMK)$. Step B2 has a complexity of $O(NM)$ since the cost function needs to be computed M times in order to find the minimum in each block. In addition, finding the value of λ that leads at least approximately to the total given number of coefficients involves performing Step B2 for multiple values of λ . This fact further increases the complexity of Algorithm B.

3.4 Comparison between Two Algorithms

Table 3.1 summarizes the major differences between the two proposed algorithms.

Algorithm A is more practical than Algorithm B mainly because of the computational complexity. As analyzed in previous sections, Step A1 has a complexity of

Table 3.1: Major differences between Algorithm A and Algorithm B.

Algorithm	Algorithm A	Algorithm B
Quality of solutions	Locally optimal solutions, close to globally optimal solutions in practice (shown in Chapter 4)	Globally optimal solutions
Complexity	Low	Higher than Algorithm A
Convergence	Preserved energy converges within several iterations	No convergence issue regarding the preserved energy
Parameter specified	The number of coefficients that are used to preserve the energy	The parameter λ, which is related to but not an explicit function of the number of coefficients
Useful Context	Compare the preserved energy of different transform sets, given the same number of coefficients used	Determine the upper bound of the preserved energy, verifying the convergence of Algorithm A

$O(KN)$ and Step A2 has a complexity of $O(NM)$, while Step B1 has a complexity of $O(MNK)$ and Step B2 has a complexity of $O(NM)$. Due to the extensive computation of transforms in Algorithm B relative to Algorithm A, we conclude that Algorithm B has a higher computational complexity than Algorithm A, with a factor of M, the number of data points within a block. To illustrate the idea, we consider a practical example where we use 8x8 block transforms. In this example, M is equal to 64. By using Algorithm A, the computational complexity is significantly reduced.

Chapter 4

Simulations

In Chapter 3, we developed two algorithms for the energy evaluation problem in multiple-transform environment. We discussed the theoretical performance of both algorithms. In this chapter, we test these algorithms in simulations. Section 4.1 discusses the implementation of the algorithms and the experimental setup used to obtain the simulation results. Section 4.2 tests the performance of Algorithm A. We show that Algorithm A performs well enough to be practical regarding the convergence speed, the insensitivity to initial conditions, and convergence quality. Section 4.3 compares the energy compaction with the rate-distortion performance. We present that our energy compaction results are consistent with the rate-distortion performance in previous research.

4.1 Implementation and Experimental Setup

In this chapter, we test the performance of different transform sets based on their energy compaction capabilities. The transforms include 2D-DCT and 1D-DCTs, as described in Chapter 2. The details of the implementation for these transforms along with the codeword design can be found in [2]. In our algorithm implementation, we use the transform matrices from the source code developed in [2] to perform the transforms. Since we only evaluate the energy compaction of these transforms, the quantization and entropy coding parts are not included. This is the major difference between our experimental setup and a practical video coding system.

We now evaluate the energy compaction of multiple transforms for the motion-compensation residuals. Specifically, the motion-compensation residual of the QCIF sequence (resolution 177 x 144) "foreman", generated from H.264 codecs, is used as the source. We use this specific residual image as a representative since it is observed that most results obtained from other residual images share the same characteristics. The source image is segmented into 4x4 or 8x8 blocks, depending on different transforms used. This residual is shown below in Figure 4-1:

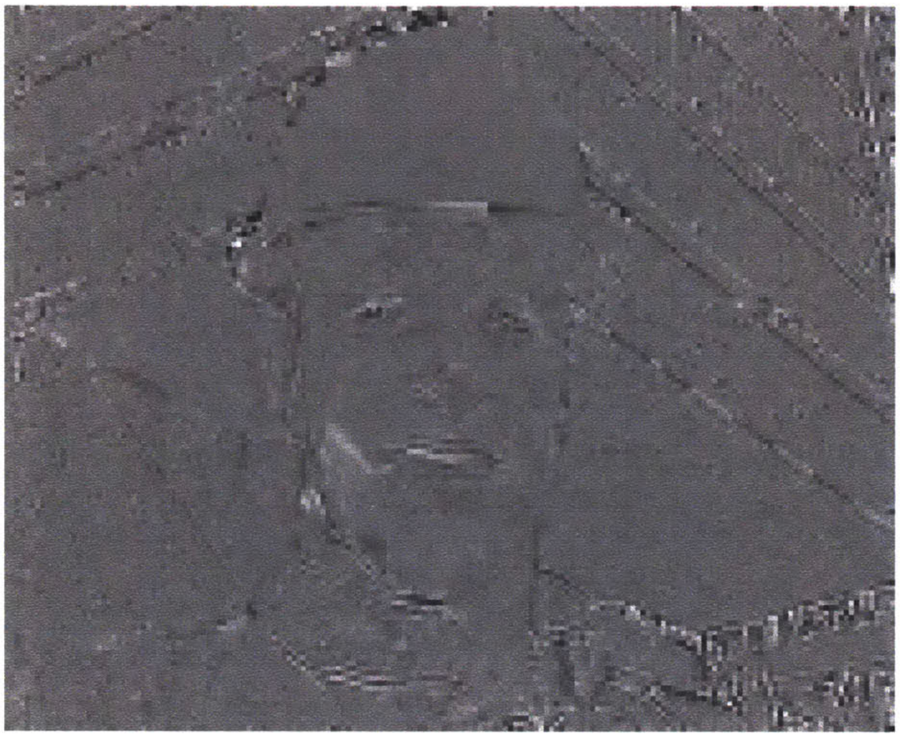


Figure 4-1: Source image used in simulations: QCIF sequence (resolution 177 x 144) MC residual "foreman".

In order to quantify the energy compaction, we plot the preserved energy as a function of the used number of coefficients. This number of coefficients used is specified in Algorithm A and generated according to a specific choice of λ in Algorithm B.

The preserved energy is in terms of the percentage relative to the total energy. The number of coefficients used is represented in terms of the percentage relative to the total number of coefficients in the entire image. When comparing the energy preservation of different transforms sets, we plot the preserved energy function for each of the transform set in one figure. At the same used number of coefficients, higher energy preservation indicates the better performance of the corresponding transform set.

We initialize Algorithm A by using the 2D-DCT for all the blocks. This initial condition is conceptually reasonable since other transforms are used as complements of the 2D-DCT, when the 2D-DCT does not characterize certain areas well enough. It should be noted that the results of Algorithm A are not quite sensitive to the initial conditions, as is shown in the next section.

4.2 Algorithm Performance

In Chapter 3, we analyzed the theoretical performance of the algorithms. We have shown that Algorithm A is guaranteed to converge to locally optimal solutions under any initial condition, and Algorithm B gives globally optimal solutions. In practice, Algorithm A is more desirable since it is less computationally intensive and the number of coefficients used can be easily specified. However, several issues regarding the performance of Algorithm A remain to be addressed. We need to find out the number of iterations required before convergence, the choice of initial conditions, and the quality of locally optimal solutions. In this section, we show that Algorithm A converges fast, is not very sensitive to initial conditions, and converges to locally optimal solutions that are close enough to globally optimal solutions. This implies that Algorithm A performs well enough to be a practical method.

We use 8x8 2D-DCT along with sixteen 8x8 1D-DCTs in this section as the transform set.

4.2.1 Number of iterations required in Algorithm A

Figure 4-2 shows, with Algorithm A, the preserved energy in the first two iterations and after convergence, when different numbers of coefficients are used. From this figure, we can see that the preserved energy increases with more iterations, which is consistent with the theoretical analysis. Another observation is that the preserved energy after convergence is very close to that of the second iteration. This implies that Algorithm A converges very fast. Numerically, when three percent of the coefficients are used, the preserved energy for the first four iterations is: 58.6%, 60.5%, 60.6% and 60.6% respectively. This result suggests that the energy converges after the third iteration. In all the simulations we have performed, the convergence occurs within at most four iterations even if we use around 250 transforms in later sections.

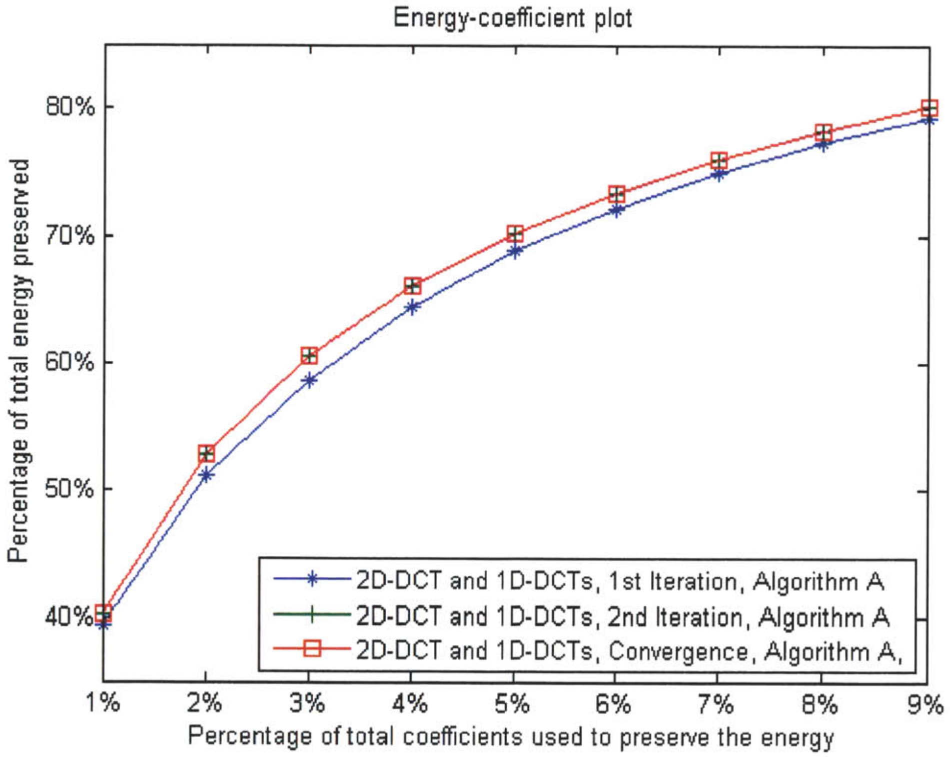


Figure 4-2: Preserved energy in the first, second iterations and after convergence.

4.2.2 Initial conditions of Algorithm A

In practice, there are a very large number of possible initial conditions that we can start from. For example, we can use a specific same transform for all the blocks and start from Step A2. We can also use the same number of coefficients in each block and start from Step A1. With different initial conditions, Algorithm A may give different locally optimal solutions.

Figure 4-3 shows the preserved energy after convergence with the above two different initial conditions. The preserved energy differs slightly between the two cases. The difference is around 0.1%, which implies that Algorithm A is not sensitive to initial conditions. Similar results are observed for other transform sets and video sequences.

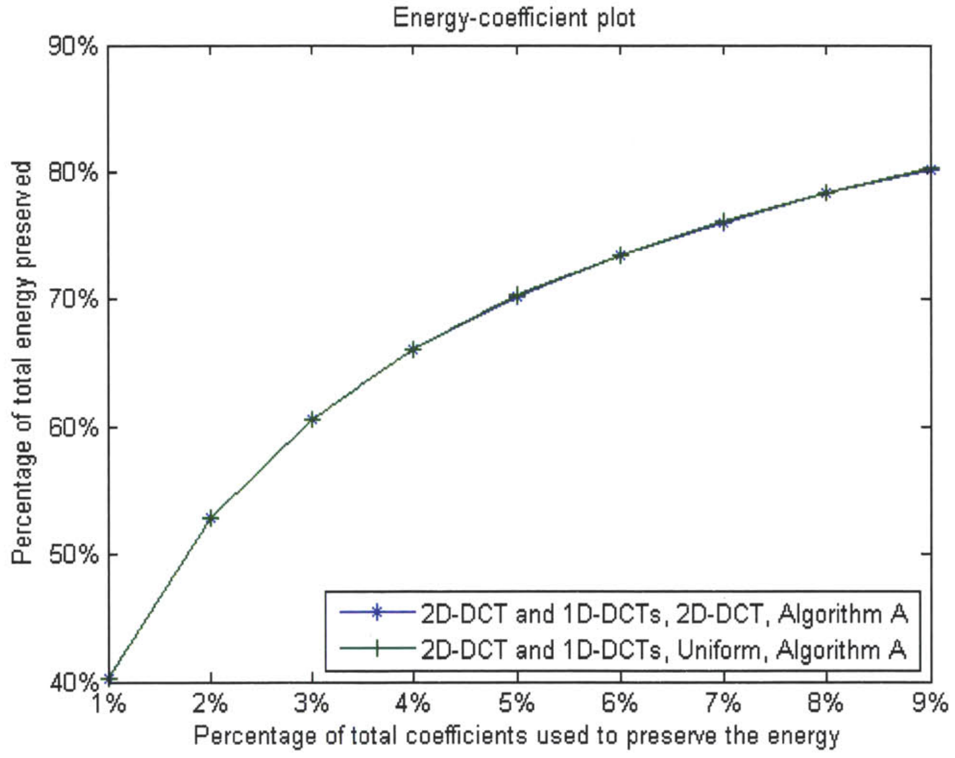


Figure 4-3: Preserved energy with different initial conditions.

4.2.3 Quality of the locally optimal solutions obtained by Algorithm A

Figure 4-4 shows the preserved energy obtained after convergence by Algorithm A and Algorithm B. It is difficult to specify the number of coefficients used in Algorithm B. Therefore, we first generate the preserved energy profile by different λ in Algorithm

B. Then we specify the same number of coefficients used in Algorithm A as those generated in Algorithm B. From this figure, we can see that Algorithm A converges to locally optimal solutions that are slightly worse than the globally optimal solutions in Algorithm B. The difference between them is no more than one percent and is hardly visible in this figure. This result indicates the results obtained using Algorithm A are quite close to the globally optimal solution in this example. We have observed similar results for many other video sequences.

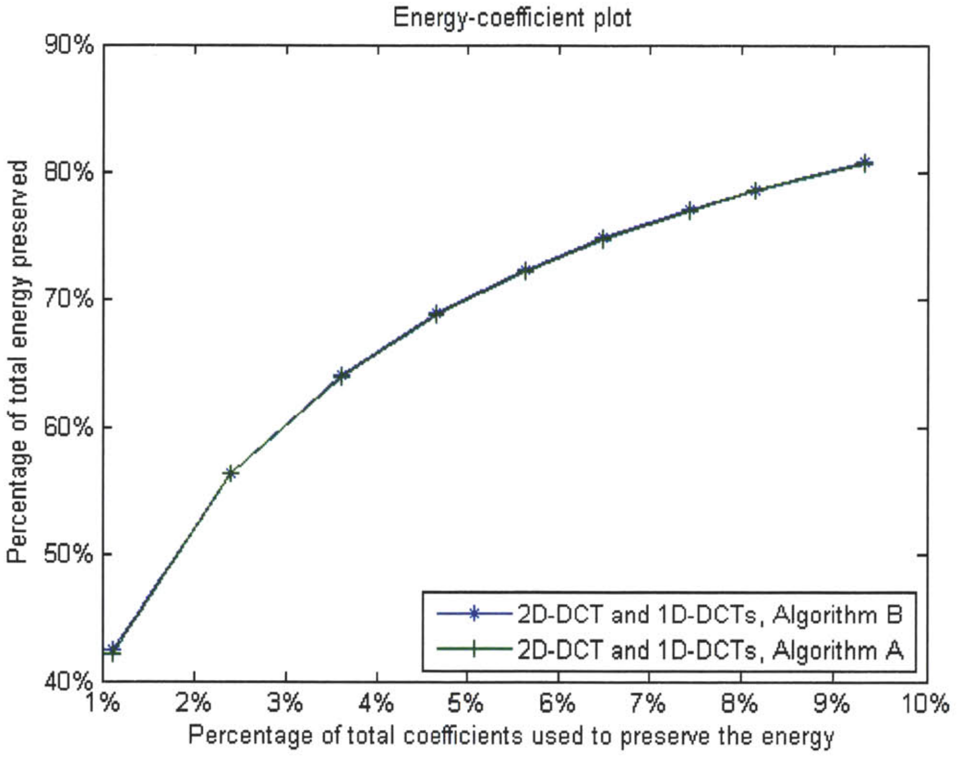


Figure 4-4: Preserved energy of Algorithm A and Algorithm B.

4.3 Comparison with R-D Performance

The algorithms we propose can be used to obtain the optimal energy profiles in multiple transform environments. These energy profiles can be used to determine whether a specific transform set will possibly be useful in a real video coding system.

The work in [2] reports that with additional 1D-DCTs of 16 different directions, the rate-distortion performance of a video coding system significantly increases. The work in [5] analyzes the performance of these 1D-DCTs, and reports that much of the performance gain by using the 1D-DCTs is due to the use of only horizontal and vertical 1D-DCTs. In this section, we evaluate the energy compaction of three transforms sets. 1), Using 8x8 2D-DCT only. 2), Using 8x8 2D-DCT along with all sixteen 8x8 1D-DCTs. 3), Using 8x8 2D-DCT and only horizontal/vertical 8x8 1D-DCTs. We compare the energy compaction capabilities with the rate-distortion of the same transform sets.

Figure 4-5 shows the preserved energy when these three transform sets are used. For these three settings, we notice that when more transforms are used, the preserved energy increases. When only 2D-DCT is used, the energy compaction is the worst. When horizontal/vertical 1D-DCTs are incorporated, the energy compaction performance significantly increases. The preserved energy continues to increase when another fourteen 1D-DCTs are incorporated, but not as much in proportion to the number of 1D-DCTs used.

Qualitatively, this result is consistent with the rate-distortion performance reported in [5], where these transforms are incorporated into a H.264 video coding system. The preserved energy is related with the distortion present in a compressed video sequence, and the number of coefficients used roughly indicates the number of bits in coding a residual image. When more coefficients are used, the preserved energy increases and the distortion decreases. This corresponds to a higher bit rate coding. Besides, compared to using only 2D-DCT, the performance increases, in terms of preserved energy, when only horizontal and vertical 1D-DCTs are incorporated. When additional fourteen 1D-DCTs are used, the performance increases further, but still, not as much when performance gain per transform is considered. This performance gain is even smaller when we take the side information into account, since it requires more bits to signal a specific transform from seventeen transforms rather than only from three.

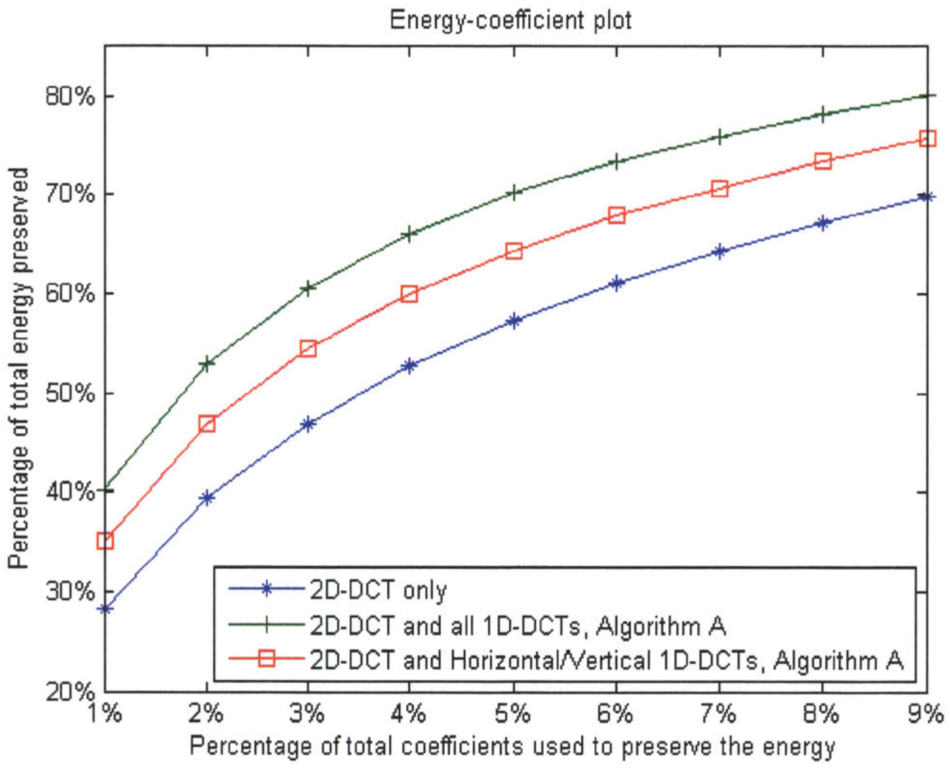


Figure 4-5: Preserved energy when 1), 2D-DCT only. 2), 2D-DCT with all 8x8 1D-DCTs. 3), 2D-DCT with horizontal/vertical 1D-DCTs are used.

Table 4.1: Percentage of coefficients used at different preserved energy levels.

Energy Level	2D-DCT only	2D-DCT with vertical horizontal 1D-DCTs	2D-DCT with all 1D-DCTs
40%	2.1%	1.4%	1.0%
50%	3.5%	2.3%	1.8%
60%	5.6%	4.0%	2.9%
70%	9.0%	6.7%	4.9%

Table 4.2: Coefficients saving at different preserved energy level.

Energy Level	2D-DCT only	2D-DCT with vertical horizontal 1D-DCTs	2D-DCT with all 1D-DCTs
40%	-	33.3%	52.4%
50%	-	37.1%	48.6%
60%	-	28.6%	48.2%
70%	-	25.5%	45.6%

Table 4.1 shows the percentage of coefficients used at different preserved energy level. Table 4.2 shows the relative coefficients saving with respect to using only 2D-DCT. These results are consistent with the Bjontegaard-Delta bitrate results reported in [5]. The coefficients saving, as well as the bitrate saving, increases when more transforms are used. When we first incorporate only horizontal and vertical transforms in addition to 2D-DCT, the coefficient saving is significant. When we further increase the number of transforms to seventeen, the additional coefficient saving is still significant. The coefficients saving introduced by additional horizontal/vertical 1D-DCTs is slightly more than that introduced by other additional fourteen 1D-DCTs. However, using only vertical and horizontal transforms may be more efficient than using the additional fourteen transforms, if we consider the average coefficient saving introduced by each transform.

There are two major differences between energy-compaction results and rate-distortion results. First, in addition to the number of coefficients, the magnitudes of the coefficients and the locations of the coefficients also affect the performance of the transforms in real video coding systems. In general, larger coefficients require more bits to code and smaller ones require less. Even if we can throw away some small coefficients in energy compaction evaluation, these small coefficients may not be quantized to zero and still need several bits, especially in high bit rate situations. Second, the side information in the bit stream may be significant. Specifically, we need to transmit the information of which transform is used for a block. These issues are related to the transforms, quantization, and entropy coding that we do not consider in this thesis.

Evaluating the energy compaction performance is probably effective in obtaining some qualitative results. For example, if a certain transform set significantly increases the energy compaction performance, we may conclude that incorporating the transform set in a video compression system to test in a practical setting may be worth the effort. In general, it is hard to get quantitative results regarding the transform performance in a real coding system by evaluating only the energy compaction performance. However, we show in the next chapter that some approximated approach can be used to accomplish such a task.

Chapter 5

Applications: Evaluating the Energy Compaction for Equal-Length 1D-DCTs

In this chapter, we demonstrate the power of the proposed algorithm when the performance of a very large number of transforms needs to be tested. In this situation, designing the codeword for every single transform and then incorporating these transforms into a video coding system to test their performance is not practical. Instead, we use our algorithms to obtain some preliminary results on energy compaction, so that we can decide whether these transforms can possibly be useful in a real video coding system.

We use 4x4 2D-DCT, 4x4 directional 1D-DCTs and 4x4 equal-length 1D-DCTs. Section 5.1 discusses the designs of the equal-length 1D-DCTs. In the remainder of this chapter, we present and discuss the energy compaction results obtained by using different sets of transforms.

5.1 Designs of Equal-Length 1D-DCTs

In [2], sixteen 8x8 directional 1D-DCTs and eight 4x4 directional 1D-DCTs are designed based on the one-dimensional structures within residual images. These transforms are effective in representing these one-dimensional structures of certain directions. We illustrated the transforms in Figure 2-1 and Figure 2-2 for 8x8 and 4x4 cases, respectively.

We observe that these transforms are not of the equal length. In other words, the number of data points in a single directional 1D-DCT is not the same. For some transforms such as the vertical and horizontal transforms, every single directional 1D-DCT uses the same number of data points as other transforms in one block. Every transform spans four or eight pixels for 4x4 and 8x8 cases, respectively. For the rest of these transforms, each transform within a block spans a different number of data points. For example, the 8x8 diagonal transforms have lengths varying from three to eight. This varying transform length also brings the varying number of transforms within a block. The vertical and horizontal 8x8 directional 1D-DCTs have eight transforms in a block while the diagonal one has thirteen.

Many modern video codecs are accelerated by using parallel computing and hardware implementation of frequently used processes. For example, butterfly structure is proposed as a hardware implementation of transforms in recent video codec standard HEVC [8]. In addition, different transforms in different blocks can be performed in a parallel manner. In these cases, transforms with equal lengths are desirable since they simplify the hardware structures. Every hardware transform processor has the same number of inputs and every block uses the same number of processors.

The 1D-DCTs proposed in [2] only considers the one-dimensional structure along certain straight directions. However, some one-dimensional structures may not be exactly straight. If these curved one-dimensional structures appear, neither 2D-DCT nor straight directional 1D-DCTs are effective in representing them.

Based on these observations, we design a set of equal-length 1D-DCTs. We find all possible one-dimensional structures of the same length. Then we segment one block to these one-dimensional structures. Each one-dimensional structure is transformed with a length four 1D-DCT. Every different partition of a block corresponds to an equal-length 1D-DCT. We only design the 4x4 transforms, since the number of all possible partitions of an 8x8 block is estimated to be much more than one million, which is too large for exhaustive search of possible partitions.

Figure 5-1 shows 12 possible one-dimensional structures used in 4x4 equal-length transforms. These one-dimensional structures are rotated for 90, 180 and 270 degree. They are also flipped in the horizontal, vertical and diagonal direction. The 4x4 blocks are tiled using different combinations of these length four one-dimensional structures. Figure 5-2 shows an example of an equal-length 1D-DCT. In this example, the block is tiled using the two rotated or flipped versions of the fifth structure, one fourth structure and one tenth structure. We find all possible partitions which segment a 4x4 block into four one-dimensional structures. This process results in 237 equal-length 4x4 1D-DCTs.

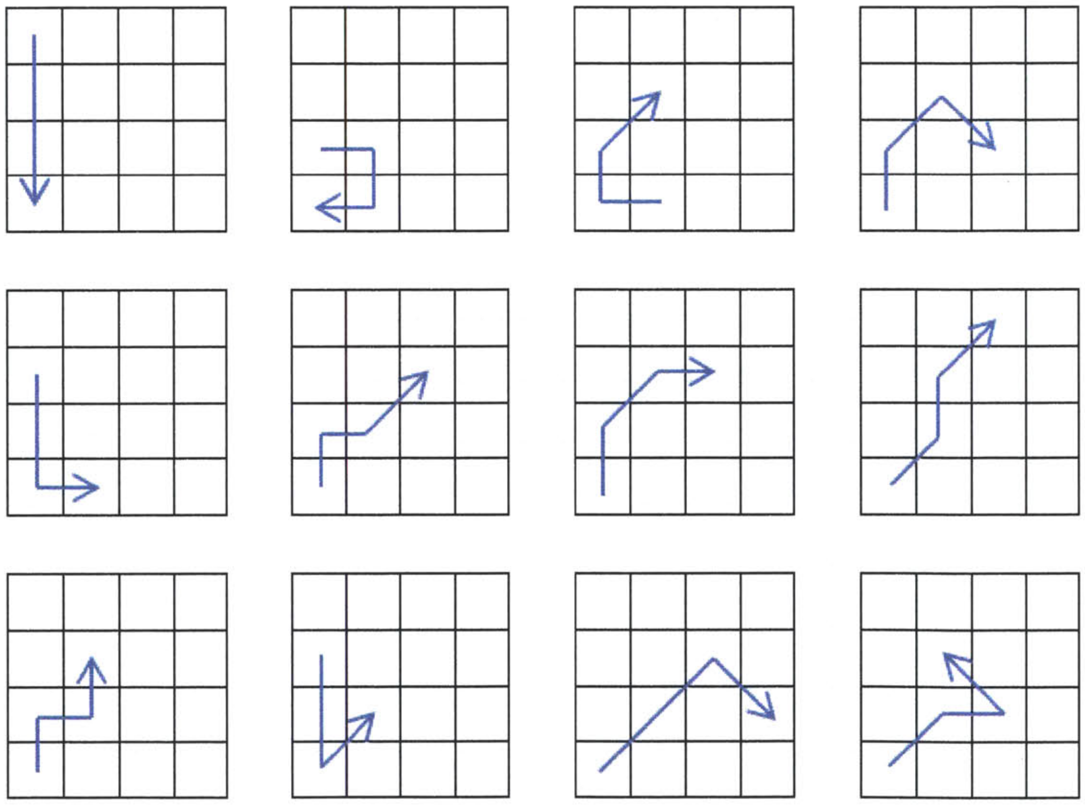


Figure 5-1: All possible structures used in 4x4 equal-length transforms.

With these additional equal-length 1D-DCTs, the one-dimensional structures in a residual image can be better characterized. The energy must compact into fewer

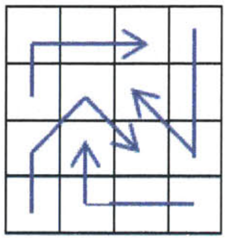


Figure 5-2: An example of 4x4 equal-length 1D-DCT.

coefficients since we are allowed more options of transforms. However, this performance gain in energy compaction does not come without costs. Since we are using more transforms, it implies that in a real coding system, the bits used to transmit side information increase as well. There is a tradeoff between using more transforms to achieve better transform performance and using fewer bits for overhead. One may incorporate all the transforms, or only incorporate a subset of them. The problem of choosing the optimal subset of transforms from a huge set of transforms arises.

The best way to solve this transform selection problem is by incorporating these transforms into a video coding system and test the rate-distortion performance of certain reasonable subsets of transforms. However, designing codewords for hundreds of transforms, which is necessary for obtaining rate-profile, is incredibly tedious. The rest of this chapter tests and analyzes the performance of these equal-length 4x4 1D-DCTs, along with the 2D-DCT and 4x4 directional 1D-DCTs, based on their energy compaction property. From this process, one can roughly determine the possibly reasonable subset for which further work on designing codewords may be worthwhile.

5.2 Overall Performance

We evaluate the energy compaction of all possible transforms before testing on any specific transform set. These transforms include: 4x4 2D-DCT, 4x4 directional 1D-DCTs and 4x4 equal-length 1D-DCTs. The source image we use is the motion-compensation residual of the QCIF sequence (resolution 177 x 144) "foreman", generated from H.264 codecs, as shown in Figure 4-1. We first verify that Algorithm A converges well enough where so many transforms are used. Then based on the results of Algorithm A, we inspect the frequencies for selection of transforms. This can help us roughly estimate which transforms are used frequently. By utilizing this kind of information, we can eliminate the transforms that are seldom selected and select sets of transforms that look reasonable, which will be further tested in the following sections.

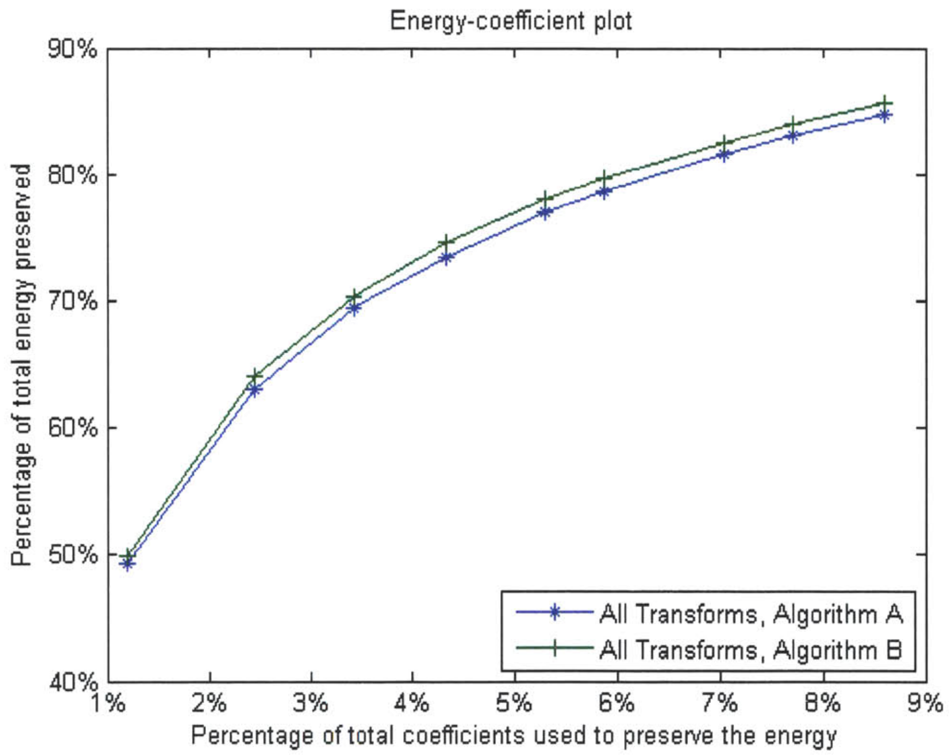


Figure 5-3: Preserved energy of Algorithm A and Algorithm B.

Figure 5-3 shows the preserved energy using both algorithms, when all the transforms are used. In this figure, we notice that Algorithm A converges to locally optimal solutions worse than the globally optimal solutions obtained by Algorithm B. The energy difference between these two curves is larger than that in previous chapter. This is due to the number of transforms we use increases from 17 to around 250. Still, this difference is much smaller compared to the energy difference between different transform sets, as we will see in the next sections. Therefore, Algorithm A is still considered accurate enough even when so many transforms are used.

Figure 5-4 shows the frequencies of selection of transforms. We use three percent of coefficients in Algorithm A. When Algorithm A converges to the locally optimal solution, every block selects a specific transform. For every transform, we count the number of blocks that choose this transform. The transforms are ordered in the following way: 2D-DCT is numbered #0, and directional 1D-DCTs are numbered from #1 to #8, among which the #1 is the vertical one and #5 is the horizontal one. The rest are equal-length 1D-DCTs, numbered from #9 to #245.

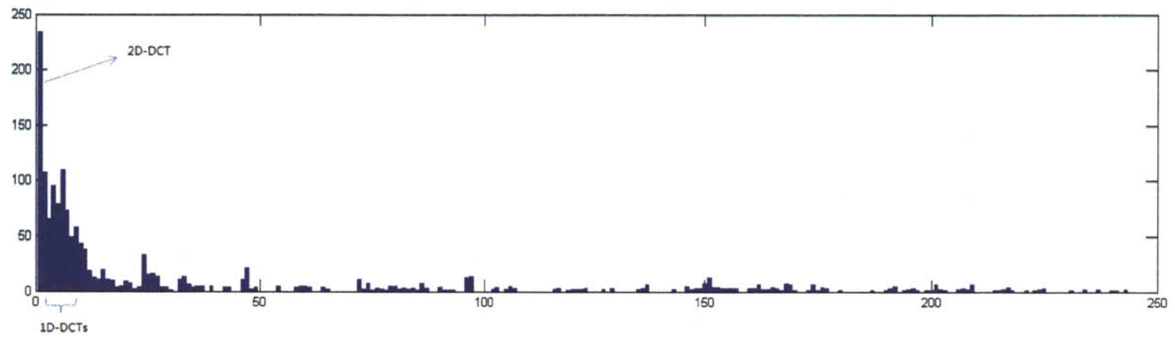


Figure 5-4: Frequencies for selection of transforms.

From this figure, we can see that 2D-DCT is chosen most frequently. This is reasonable since motion prediction performs well in most smooth areas. In these areas, the residual signal has close to zero pixel values and is basically two-dimensional, which can be effectively characterized by 2D-DCT. Next frequently chosen transforms are #1 and #5, which correspond to vertical and horizontal 1D-DCTs. This is because in motion-compensation residuals, many one-dimensional structures are aligned vertically and horizontally, as analyzed in [5]. Besides 2D-DCT, horizontal and vertical 1D-DCTs, other directional 1D-DCTs are chosen more frequently than the equal-length transforms, which implies directional 1D-DCTs are strong enough to characterize most one-dimensional structures. This result suggests that equal-length 1D-DCTs may only be potentially useful as complements, not replacements of these directional 1D-DCTs.

Among all the equal-length transforms, the frequencies for choice of transforms

Table 5.1: Frequencies for selection of different subsets of transforms.

Transforms	2D-DCT only	H/V 1D-DCTs	Other 1D-DCTs	Equal-Length
#Transforms	1	2	6	237
Frequencies	234	217	419	714
Percentage	14.8%	13.7%	26.5%	45.1%

significantly differ. Only a few equal-length 1D-DCTs are used frequently, and the frequencies of other transforms are close to zero. This indicates even if we are using equal-length 1D-DCTs, we may not need all of them. Only a small subset of these equal-length 1D-DCTs are sufficient in characterizing those curved one-dimensional structures. This is intuitively reasonable, due to the special structures of these equal-length 1D-DCTs. These equal-length 1D-DCTs are different combinations of a same set of tiling components, shown in Figure 5-1. That means there are a number of transforms sharing a specific one-dimensional structure at the same location within the 4x4 block. If such one-dimensional structure happens to exist in the residual image, using one representative from these transforms to characterize this structure is sufficient. Therefore, the frequencies cluster to a single transform when this specific one-dimensional structure exists in the residual image. The frequencies for choice of these transforms are summarized numerically in Table 5.1.

Based on these observations, we divide all the transforms into four subsets. 2D-DCT, horizontal and vertical 1D-DCTs, other directional 1D-DCTs and equal-length 1D-DCTs. The rest of this chapter deals with evaluating and analyzing the energy compaction of different combinations of these transform sets. By doing so, we attempt to see if these equal-length 1D-DCTs can potentially help to improve the performance when incorporated into a video coding system.

5.3 Using all equal-length 1D-DCTs

In this section, we evaluate the energy compaction of five transform sets of size 4x4, listed below:

Transform Set 1: 2D-DCT only;

Transform Set 2: 2D-DCT and horizontal and vertical 1D-DCTs;

Transform Set 3: 2D-DCT and all eight directional 1D-DCTs;

Transform Set 4: 2D-DCT, horizontal and vertical 1D-DCTs, and equal-length 1D-DCTs;

Transform Set 5: 2D-DCT, all eight directional 1D-DCTs and equal-length 1D-DCTs.

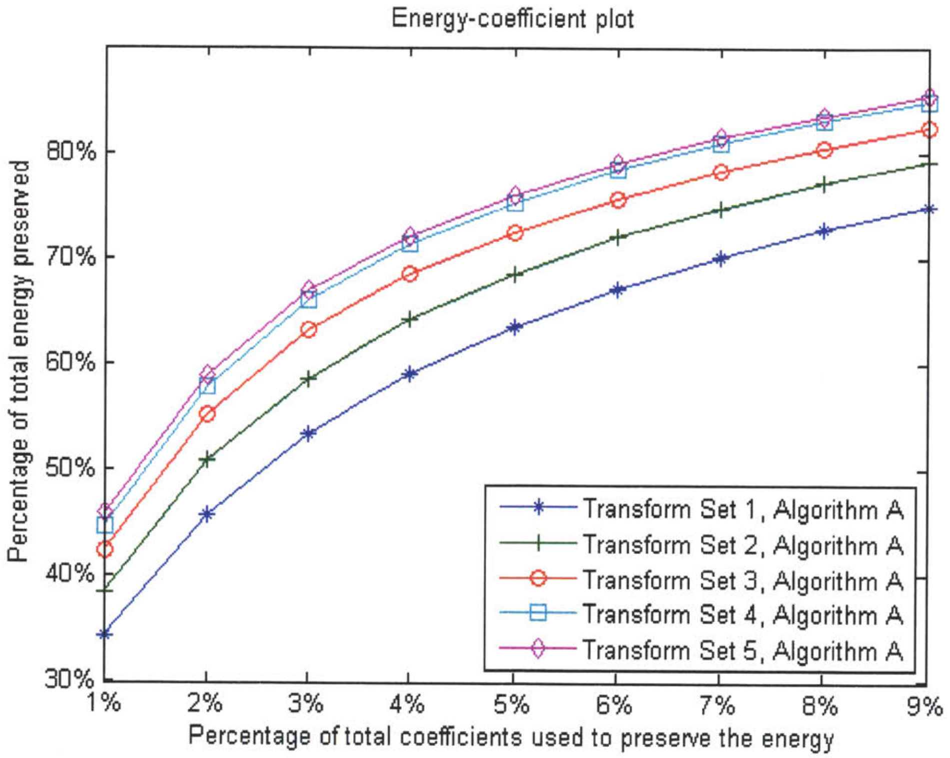


Figure 5-5: Preserved energy of five 4x4 transform sets with all equal-length 1D-DCTs.

Figure 5-5 shows the preserved energy when using these five transform sets. We design these transforms sets as combinations of the subsets listed in Table 5.1. We note that in this section, we only consider the 237 equal-length transforms as a whole. Improvement by using part of these transforms is included in the next section.

Transform Sets 1-3 only consider using subsets of directional 1D-DCTs. These three curves display similar patterns as in the 8x8 case discussed in Chapter 4. By incorporating more transforms, the energy compaction increases. The increase introduced by only using horizontal and vertical transforms is larger than that of using the rest six directional 1D-DCTs.

Transform Set 4 replaces six tilt directional 1D-DCTs with 237 equal-length 1D-DCTs. We include the horizontal and vertical transforms in this set due to two reasons. First, horizontal and vertical transforms can characterize horizontal and vertical one-dimensional structures that appear so frequently in the residual images. Second, horizontal and vertical transforms are equal-length transforms as well. In other words, we include 2D-DCT and all equal-length transforms in this set. The result shows that the preserved energy using this transform set is higher than that of using Transform Sets 1-3.

Compared to the number of transforms we use, the increase in energy compaction is not that significant. The preserved energy increase is almost the same when we progressively include three transform sets: horizontal and vertical 1D-DCTs, six directional 1D-DCTs and 237 equal-lengths 1D-DCTs. However, the number of transforms used increases from one, three, nine to over two hundred. Intuitively, this selection of transforms that uses 237 equal-length transforms may not be desirable. For the rest of this section, we provide an argument to illustrate this point.

Table 5.2 shows the percentage of coefficients used at different preserved energy level. We use this information to estimate number of bits used in a coded video sequence. The same energy level indicates the same distortion level in the coded sequence. The number of coefficients is roughly proportional to the number of bits used to represent these coefficients with a scaling factor. To transmit a specific block, we need to transmit the additional side information regarding the type of transform we use. If we use one bit to signal using 2D-DCT, and using code words with equal lengths for other transforms, this scheme results in the overhead load of 1 bit for 2D-DCT and $1 + log(N - 1)$ bits for other transforms. For Transform Set 2, we need 1 extra bit is we use transforms other than 2D-DCT. For Transform Set 3 and Transform Set 4, we need three and eight more bits, respectively.

Having a heavy overhead does not necessarily mean the transform set is useless. It is useless only when the side effect of heavy overhead cannot be compensated by

Table 5.2: Number of coefficients used at different preserved energy levels.

Energy Level	Transform Set 1	Transform Set 2	Transform Set 3	Transform Set 4
50%	2.6%	1.9%	1.6%	1.4%
60%	4.3%	3.2%	2.6%	2.2%
70%	7.0%	5.5%	4.3%	3.8%

the coefficient saving from a better energy compaction. In order to compare the effect of overhead and coefficient saving, the relationship between them has to be established. That is, we need to convert the coefficient saving to bit saving. The number of coefficients is not linear to the number of bits. However, linearity is a fair approximation when we consider the performance difference within a small range. Therefore, we only need to consider the scaling factor between number of bits and percentage of coefficients. If the scaling factor is known, we can convert the coefficient saving into bit saving and see if that is sufficient to overcome the side effect of heavy overhead.

It is shown in [5] that using Transform Set 3 slightly increases the rate-distortion performance than using Transform Set 2. Roughly speaking, we consider their rate-distortion performance to be the same. That means at the same energy level, the difference between the percentage of coefficients in Transform 2 and Transform 3 can be compensated by the 2 extra overhead bits. Using this information, we can estimate the scaling factor between percentage of coefficients and number of bits. For example, the percentages of coefficients when we preserve 50% using Transform Set 2 and Transform Set 3 differ by 0.3%, which means we can scale 0.3% coefficients to two bits. Therefore, using eight bits to transmit the overhead is beneficial only when the percentage of coefficients is decreased by at least 0.3x7/2=1.05% from Transform Set 2. However, this decrease of percentage of coefficients from Transform Set 2 to 4 is only 0.5%, which is much smaller than 1.05%. This implies using Transform Set 4 will introduce too large overheads. The results are similar when we preserve 60% and 70% energy.

We can obtain similar results for Transform Set 5 using the same type of analysis.

In conclusion, using all the equal-length 1D-DCTs specifically used in this experiment introduces too large overheads. Therefore, this scheme may not be practical in a real video coding system.

5.4 Using part of equal-length 1D-DCTs

From the previous section, we can see that using all the equal-length transforms costs too much in transmitting the side information. One way to reduce the size of the overhead is to reduce the number of transforms used, while maintaining a fair amount of energy preservation. This is discussed in this section.

When only part of the equal-length 1D-DCTs are used, the specific subset of transforms to be included should be chosen carefully. We first determine the number of transforms that are used. This can be accomplished by utilizing the information from Table 5.2. Similar to the argument in the last section, we can see that Transform Set 2 needs one extra bit to signal the transform we use while Transform Set need three bits. This indicates that the coefficient percentage difference between Transform Set 2 and 3 at the same energy level roughly corresponds to two bits in a video coding system. We further observe that this difference between Transform Set 3 and Transform Set 4 is around half of that between Transform Set 2 and 3. This indicates that the number of bits in the side information of any transform set probably should not exceed that of Transform Set 3 by more than one. In other words, besides 2D-DCT, we should use no more than 16 transforms, whose side information can be coded with no more than four bits. Suggested by this observation, we modify the transform sets we use as the following:

Transform Set 1: 2D-DCT only;

Transform Set 2: 2D-DCT and horizontal and vertical 1D-DCTs;

Transform Set 3: 2D-DCT and all eight directional 1D-DCTs;

Transform Set 4: 2D-DCT, all eight directional 1D-DCTs, and eight equal-length 1D-DCTs with the highest frequencies shown in Figure 5-4.

Figure 5-6 shows the energy profile with these four transform sets. By incorporating only eight more equal-length 1D-DCTs in Transform Set 4, the preserved energy slightly increases compared to using Transform Set 3. It is graphically clear that the extra 1 bit side information introduced in Transform Set 4 can hardly be compensated by the small amount of coefficient saving.

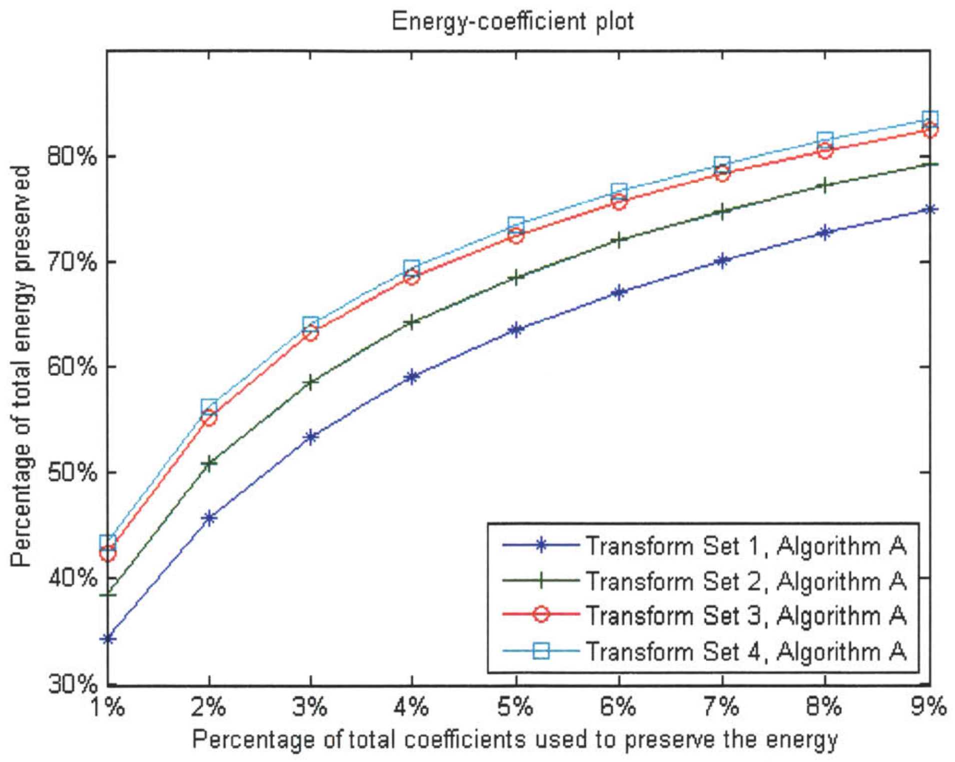


Figure 5-6: Preserved energy of four 4x4 transform sets with part of equal-length 1D-DCTs.

5.5 Conclusions

In this chapter, we design a set of equal-length transforms to illustrate the power of an algorithm developed in Chapter 3. The equal-length transforms were considered based on two considerations. One is simplification and acceleration of both software and hardware implementation of transforms. The second is that these equal-length transforms can be used to characterize those curved one-dimensional structures which are not considered in directional 1D-DCTs.

We design a total number of 237 equal-length 1D-DCTs. It is not practical to test the performance of these transforms based on the rate-distortion performance after incorporating them into a real coding system. Instead, we test them based on their energy compaction performance using the proposed algorithms.

Different transform sets that seem reasonable are proposed to be tested. We first use all equal-length 1D-DCTs as a whole and estimate the preserved energy in the best case. With some reasonable approximation, we show that it is inefficient to use all these transforms. Then we get an upper bound of the number of transforms that can be used, and redesign these transforms that can give best possible results, based on this upper bound. We evaluate the transform performance again by doing simulations on the new transform sets. The results show that the newly incorporated eight transforms do not bring in much preserved energy increase.

This result implies that the specific set of the proposed equal-length transforms may not be effective if they are incorporated into a video coding system. It is suggested that further work on designing the codewords for these transforms is not necessary. We also conclude that using more complex transforms may not bring additional performance gain to a coding system. The directional 1D-DCTs may exhaust the performance of exploiting the one-dimensional structures.

Chapter 6

Conclusions

6.1 Summary

Transforms are used in many image and video coding system. For example, 2D-DCT has been used extensively in image coding systems such as JPEG [1], and in video coding systems such as H.264 [4]. It is generally believed that the performance of these coding systems can be improved by using more than one transform. Recent studies consider using up to seventeen one-dimensional transforms in a video coding system, and it is reported the performance significantly increases with multiple transforms [2].

The performance of a coding system is generally measured by its rate-distortion profile. Transforms are incorporated into a coding system and the rate-distortion performance is tested. To obtain the rate-distortion performance of a system, one needs the rate profiles of all used transforms. These rate profiles are not dependent exclusively on the transforms themselves. The corresponding quantization and entropy coding parts have to be carefully designed as well, in order to achieve the best performance of the transforms. Since the designs of these parts are typically tedious, evaluating the transforms based on rate-distortion performance is not effective, especially when we have a large number of transforms.

Instead of evaluating the rate-distortion performance, we evaluate the energy compaction capability of a specific set of transforms. We find the largest possible energy that we can preserve using some given number of coefficients. Since the energy is related to the distortion and the number of coefficients is related to rate, the energy compaction property roughly indicates the actual performance of the transforms in a practical coding setting.

When multiple transforms are used, obtaining the largest amount of energy is a difficult task. One may use the brute force method by enumerating all transform combinations used in all blocks. However, this type of approach is not practical in terms of computation due to the exponentially increasing complexity. We propose two effective algorithms to solve this problem and make energy compaction evaluation possible in a multiple transform environment.

The first algorithm is based on the two observations. First, finding the best transforms given the number of coefficients used in each block can be easily accomplished. Second, finding the optimal number of coefficients used in each block given selected transforms can be easily accomplished as well. We develop this algorithm by iteratively applying these two steps. We show that this algorithm is guaranteed to converge at least to locally optimal solutions. The complexity of this algorithm is fairly low and can be used in practice.

The second algorithm searches the optimal solutions in the block optimal energy functions. We show that this algorithm always finds the globally optimal solutions. However, this algorithm is more computationally intensive than the first one. Specifying the number of coefficients used in this algorithm is not as easy as the first algorithm. As a result, we use this algorithm as the benchmark of the first algorithm.

We test the performance of these two algorithms in simulations by applying 2D-DCT and directional 1D-DCTs on a motion-compensation residual image. We demonstrate that the first algorithm converges to locally optimal solutions that are slightly worse than the globally optimal solutions. Other issues regarding this algorithm are discussed to verify that it can be used in practice. The energy compaction property obtained with our algorithms is shown to be consistent with the rate-distortion results in a similar setting.

We propose another set of more than two hundred transforms in complements to the transforms we already have. It is not practical to implement these transforms into a coding system. Therefore, we test the energy compaction performance of all these transforms. We analyze the relation between the energy compaction and the rate-distortion performance after trying different sets of transforms that seem reasonable. The results show that the side information needed by potentially incorporating these particular sets of transforms is too large. We conclude that these transforms may not be useful in a practical video coding system. By doing this, we verify that our proposed methods are useful in quickly determining whether a large set of transforms may be useful.

6.2 Future Research

In this thesis, we demonstrated the power of the proposed methods with a limited number of examples. They can be used to determine if other transforms sets can be useful for a particular type of images. For example, we may apply our algorithms to 3D video coding applications. We can determine if these 1D-DCTs can be useful for binocular prediction residuals and depth map prediction residuals.

The interpretation of the energy compaction results can be improved by taking consideration of other issues besides the number of coefficients used. We may also include the statistics of the magnitudes of the coefficients. Another improvement is to explore a more accurate relationship between the numbers of coefficients and the rate profiles, which can better characterize the transform performance in a video coding system. This can be done, for example, by assuming varying-length code words for different transforms based on their choice of frequencies. In many video and image coding systems, the transform coefficients are weighted in a frequency-dependent manner due to different sensitiveness to different spatial frequencies of human visual system. In this case, our algorithms can also be applied to the weighted transform coefficients, which better reflect the actual coding process in a real coding system.

Our algorithms are designed for determining which sets of transforms can be useful in video compression. In addition to solving this specific problem that the algorithms were designed for, they can also be useful in other contexts. The basic approach of the algorithm can be useful in a more general optimization problem, where it is desirable to preserve the energy of a signal with a small number of parameters. In other words, while Algorithm B can be extended to a general but slow approach based on duality, Algorithm A suggests an effective and fairly accurate approach to solve this type of optimization problems.

As another example of possible application of our algorihtms, when the constraint is bit rate instead of number of coefficients, our algorithms can be modified to work in a real coding system. We note that while Algorithm B is degenerated into the Lagrangian multiplier method in this case, Algorithm A is particularly useful when we want to have a strong control over the bitrate. This is because when using Lagrangian multiplier method, the resulting bitrate is controlled by a specific multiplier parameter and is strongly dependent on the video contents. If we use an idea similar to Algorithm A to control the bitrate, the bit rate can be at least roughly specified. This is an area of our current research.

Bibliography

[1] J. S. Lim. *Two-Dimensional Signal and Image Processing.* Prentice Hall, 1990.

[2] F. Kamisli. "Transforms for Prediction Residuals in Video Coding" Ph.D. thesis, Dept. Elect. Eng. and Comput. Sci., MIT, Cambridge, MA, 2010.

[3] F. Kamisli and J. S. Lim. "Video compression with 1-D directional transforms in H.264/AVC" IEEE ICASSP 2010, pp. 738-741, March 2010.

[4] Iain E. G. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia.* Wiley, 2003.

[5] H. Zhang, "Analysis of one-dimensional transforms in coding motion compensation residuals for video applications" M.Eng thesis, Dept. Elect. Eng. and Comp. Sci., MIT, Cambridge, MA, 2011.

[6] A. Ortega and K. Ramchandran. "Rate-distortion methods for image and video compression" Signal Processing Magazine, IEEE, 15(6):23-50, Nov 1998

[7] M. Blum, R.W. Floyd, V. Pratt, R. Rivest and R. Tarjan, "Time bounds for selection," J. Comput. System Sci. 7 (1973) 448-461.

[8] Han, Woo-Jin "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools" IEEE Transactions on Circuits and Systems for Video Technology 2010 Vol. 20 Iss. 12 p. 1709 - 1720